Ujian Akhir Semester
Manajemen Proyek Perangkat Lunak

Kualitas Perangkat Lunak

Oleh:
Nia Saurina - 5106.201.811

Dosen:
Fajar Baskoro, S. Kom., MT

PROGRAM PASCASARJANA
TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI SEPULUH NOPEMBER SURABAYA
2007

KUALITAS PERANGKAT LUNAK

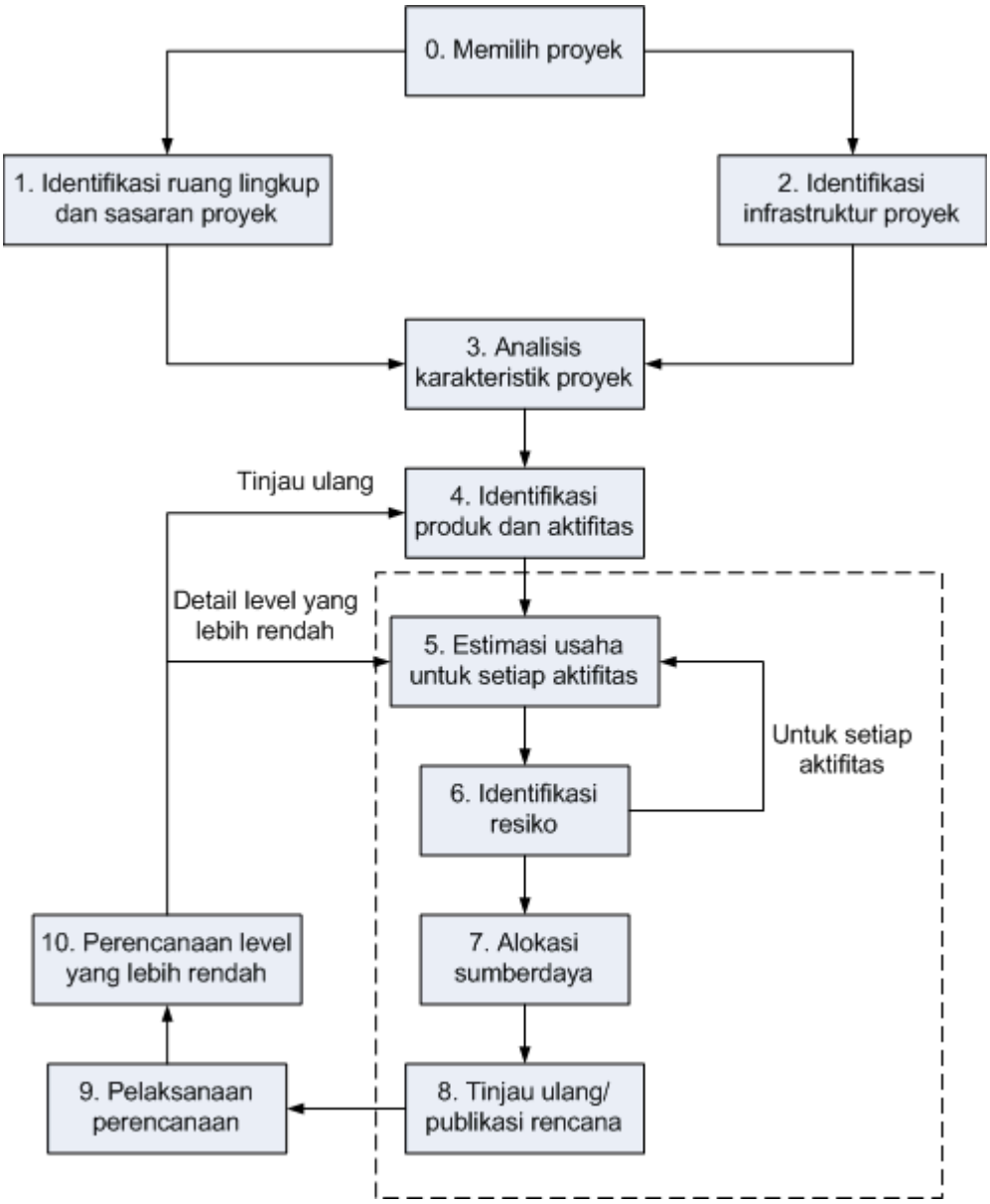
1. Pendahuluan

Ketika kualitas pada umumnya disetujui menjadi ‘barang bagus’, apa yang dimaksud orang pada kata ‘kualitas’ pada sebuah sistem dapat menjadi hal yang meragukan. Oleh karena itu dibutuhkan definisi yang lebih tepat untuk kualitas pada sistem. Bagaimanapun juga, hal ini tidak cukup- kita membutuhkan keputusan secara obyektif dimana sistem menemui kebutuhan kualitas dan pengukuran. Hal ini akan menjadi fokus seseorang seperti Brigette pada Britghtmouth College pada proses pemilihan paket.

Untuk seseorang-seperti Amanda pada IOE-pengembang software, pengukuran kualitas tidak hanya berfokus pada hasil akhir, tetapi juga pada proses. Amanda mungkin ingin menilai kualitas sistem akhir selama dalam proses pengembangan, dan juga meyakinkan pengembangan metode yang digunakan untuk menghasilkan kualitas. Hal ini akan mengarah ke penekanan yang berbeda-daripada berfokus pada kualitas sistem akhir, *customer* yang berpotensi (*customer* yang memiliki keinginan untuk membeli software) mungkin akan mencoba untuk melakukan cek ke *supplier* dalam penggunaan metode terbaik.

2. Menempatkan Kualitas Perangkat Lunak pada Perencanaan Proyek

Kualitas akan berfokus pada semua langkah perencanaan dan pelaksanaan proyek, tetapi akan merujuk ke kerangka Step Wise (gambar 1)



Gambar 1. Penempatan Kualitas Perangkat Lunak pada Step Wise

- Langkah 1: identifikasi ruang lingkup dan sasaran proyek. Beberapa sasaran berhubungan dengan kualitas aplikasi yang akan dikerjakan.
- Langkah 2: identifikasi infrastruktur proyek. Dengan langkah ini maka memerlukan identifikasi standard instalasi dan prosedur. Beberapa diantaranya akan berbicara tentang kualitas
- Langkah 3: analisis karakteristik proyek. Pada analisis karakteristik proyek berdasarkan kualitas, aplikasi akan diimplementasikan untuk melihat kebutuhan khusus kualitas. Misalnya pada keamanan kritis

(keamanan data) yang ekstrim dari keseluruhan tambahan batas aktifitas yang bisa digunakan sebagai pengembangan versi ke n dimana tim pengembang pada software versi sebelumnya akan sama berjalan secara paralel dengan output yang akan di cross cek untuk bahan perdebatan.

- Langkah 4: identifikasi produk dan aktifitas proyek. Poin ini sangat penting dimana masukan, keluaran dan kebutuhan proses di identifikasi untuk setiap proyek. Kebutuhan alami digambarkan lebih jelas pada bab ini.
- Langkah 8: tinjau ulang dan publikasi perencanaan. Pada langkah ini keseluruhan aspek kualitas perencanaan proyek akan di tinjau ulang.

3. Keutamaan Kualitas perangkat lunak

Kita berharap kualitas menjadi fokus pada semua prosedur barang dan jasa. Bagaimanapun juga, karakteristik khusus perangkat lunak menjadi hal yang tak mudah dimengerti (intangibility) dan kompleksitas, membuat permintaan khusus

- Meningkatkan perangkat lunak secara kritis. *End user* umumnya tertarik tentang kualitas umum perangkat lunak, terutama yang dapat diandalkan. Hal ini akan menyebabkan peningkatkan ketergantungan suatu organisasi pada sistem komputer dan perangkat lunak sehingga lebih berguna untuk area keamanan kritis (pengamanan data), misalnya pengontrolan pesawat terbang (aircraft)
- Hal yang tidak dimengerti (intangibility) oleh perangkat lunak membuat kesulitan untuk mengetahui pekerjaan tertentu dalam memuaskan *customer* di proyek. Hasil pekerjaan ini menjadi sesuatu yang dapat dimengerti (tangible) dengan menuntut developer menghasilkan ‘deliverables’ yang dapat diuji untuk kualitas
- Penjumlahan kesalahan selama pengembangan perangkat lunak. Sebagai pengembangan sistem komputer , jumlah langkah output dibuat dari satu langkah input ke langkah selanjutnya, kesalahan pada deliverables awal akan

ditambahkan pada langkah berikutnya, yang mengarahkan ke penjumlahan efek merugikan. Umumnya pada proyek berikutnya dimana kesalahan ditemukan menjadi lebih mahal untuk diperbaiki. Sebagai tambahan, karena jumlah kesalahan sistem tidak diketahui, fase debugging proyek menjadi sulit sekali untuk dikontrol.

Untuk alasan inilah manajemen kualitas merupakan bagian utama keseluruhan manajemen proyek.

4. Mendefinisikan Kualitas Perangkat Lunak

Kualitas merupakan kata yang meragukan dan dibutuhkan untuk definisi yang secara hati-hati untuk dimaknai. Untuk sistem perangkat lunak apapun, seharusnya ada tiga spesifikasi yaitu ;

- Spesifikasi fungsional

  Spesifikasi ini menggambarkan apa yang dikerjakan sistem – hal ini merupakan fokus utama metodologi seperti Unified Software Development Process
- Spesifikasi kualitas (atau atribut)

  Spesifikasi ini memfokuskan pada , bagaimana fungsi beroperasi.
- Spesifikasi sumber daya

  Spesifikasi ini berfokus pada seberapa banyak yang harus dihabiskan oleh sistem

Latihan 1 di Brightmouth College, Brigette telah memilih paket best of the shelf payroll untuk universitas. Bagaimana metode yang seharusnya dia lakukan untuk membuat paket itu?

Satu elemen pendekatan dapat menjadi identifikasi kriteria yang bertentangan dengan paket payroll yang akan diputuskan. Kriteria apa yang seharusnya diikutkan? Bagaimana kita bisa menggabungkan kriteria paket tersebut dibuat?

a. Menghasilkan pengamatan untuk mengetahui kebutuhan user. Pengamatan ini untuk membedakan kebutuhan kelompok user yang berbeda.

b. Mengatur kebutuhan kelompok yang berhubungan dengan kualitas individu dan atribut. Pengaturan ini dapat dilakukan, misalnya *functionality* (batas fitur yang dimiliki software), harga, usability, kapasitas, efisiensi, *flexibility* dan *reliability*
c. Terdapat beberapa kebutuhan yang akan menjadi kebutuhan mendasar. Misalnya sebuah aplikasi memerlukan penyimpanan untuk sejumlah karyawan. Jika penyimpanan tersebut tidak dapat dilakukan, maka kebutuhan tersebut akan segera dihilangkan.
d. Dalam kasus sebuah kebutuhan dapat bersifat relative (tidak tetap). Beberapa kebutuhan relative tersebut lebih penting dari kebutuhan yang lain. Harga software yang murah dapat dipertimbangkan tetapi software yang lebih mahal tidak dapat dihiraukan. Hal ini nampak pada pemberian rating untuk setiap kebutuhan, katakan nilai yang diberikan adalah 10 adalah hal yang penting.
e. Batas paket kandidat kebutuhan yang memungkinkan perlu dikenali. Jika pada satu paket terdapat beberapa proses seleksi awal, misalnya pengenalan harga dapat digunakan untuk mengurangi pesaing sehingga dapat mengatur daftar harga (shortlist).
f. Cara sederhana untuk mengukur kualitas software yang diinginkan perlu dipertimbangkan. Pada beberapa kasus, misalnya mengenai harga dan kapasitas, literatur / daftar penjualan atau spesifikasi tehnis dapat dikomunikasikan. Pada contoh kasus lain mengenai efisiensi, pra-pelatihan dapat dipengaruhi selama survey yang dilakukan user dapat memberikan informasi yang dibutuhkan.
g. Pada kegiatan pelatihan beberapa software memiliki kemungkinan kekurangan informasi, tetapi tidak akan merugikan kerugian kualitas lain. Cara sederhana mengkombinasi perbedaan kualitas adalah dengan memberikan nilai 10 pada ada atau tidaknya kualitas. Setiap nilai akan dikalikan dengan nilai 10 untuk kepentingan kualitas (lihat d) dan hasil dari semua perkalian dapat dihitung untuk memberikan nilai keseluruhan pada software.

Beberapa kualitas dapat diidentifikasi untuk menghasilkan perangkat lunak yang mencerminkan pandangan eksternal perangkat lunak yang akan diliki user, sebagaimana pada kasus usability. Kualitas eksternal akan dipetakan ke faktor internal dimana developer akan bersikap waspada. Hal ini bisa diperdebatkan, misalnya kode yang tersusun dengan baik (well-constructed code) umumnya memiliki kesalahan lebih sedikit dan peningkatan kemampuan untuk digunakan.

Mendefinisikan kualitas tidak cukup hanya dengan jika kita melakukan keputusan yang sesuai dengan kebutuhan sistem sehingga kita perlu melakukan pengukuran kualitas. Untuk setiap karakteristik kualitas, satu atau lebih pengukuran yang ditemukan akan menghasilkan nilai derajat kualitas.

Pengukuran yang baik harus dapat menghubungkan jumlah unit sampai keadaan yang maksimal. Jumlah maksimum kesalahan pada program, misalnya kesalahan yang berhubungan dengan ukuran program sehingga pengukuran kesalahan per seribu baris kode (fault per thousand line of code) akan lebih membantu daripada jumlah kesalahan (total fault) pada program sebagai jaminan kualitas program.

Mencoba untuk menemukan pengukuran untuk kualitas tertentu membantu mengklarifikasi apakah kualitas itu. Yang dipertanyakan adalah, dampak terhadap ‘Bagaimana kita mengetahui kapan hal ini akan berhasil?’ menjawab tentang sasaran kualitas akan dijelaskan lebih lanjut.

Kemungkinan pengukuran kualitas dapat dilakukan secara langsung atau indirect dimana sesuatu yang diukur bukan kualitasnya sendiri tetapi indikator kualitas yang dihadirkan. Misalnya jumlah permintaan oleh user diterima dengan bantuan pengoperasian aplikasi perangkat lunak tertentu dapat menjadi pengukuran usability secara indirect. Dengan identifikasi manajemen pengukuran pengaturan program bagi tim anggota proyek sangat penting untuk meningkatkan kualitas pengukuran itu sendiri. Misalnya jumlah kesalahan yang ditemukan di baris program tidak bisa dihitung, pada dasarnya hal itu memerlukan proses pemeriksaan yang lebih teliti, sehingga kesalahan lebih mudah ditemukan. Meningkatkan pengukuran ini bisa saja dilakukan, tentu saja, dengan mengetahui kesalahan yang ada melalui langkah pemeriksaan daripada menghilangkan kesalahan dari awal – dimana hal ini bukan faktor utama.

Kebutuhan utama untuk spesifikasi karakteristik kualitas adalah

- Definisi / deskripsi: definisi karakteristik Kualitas.
- Skala: unit pengukuran.
- Uji: uji praktis yang dilakukan dimana dihasilkan atribut Kualitas
- Penerimaan secara sederhana : nilai terburuk yang mungkin saja diterima jika karakteristik lain dibandingkan dengan kualitas, dan saat produk belum diterima.
- Batas target: batas nilai dimana perencanaan pengukuran nilai Kualitas seharusnya diperbaiki.
- Sekarang: hasil yang diharapkan sekarang

Oleh karena itu ada pengukuran yang dapat diaplikasikan lebih dari satu ke karakteristik kualitas. Saat melakukan daftar (drafting) spesifikasi kualitas karakteristik kualitas mungkin saja akan terbagi dalam beberapa sub-karakteristik. Misalnya sub-karakteristik dari 'usability' mungkin saja dapat dijelaskan (*communicativeness*). Satu aspek yang mungkin saja menjadi pemahaman struktur menu, adalah begitu mudahnya menemukan perintah untuk menyelesaikan beberapa fungsi. Aspek lain dari penjelasan tersebut adalah bagaimana mengatakan keberadaan pesan kesalahan yang muncul, melalui halaman "help"

Latihan 2 : Jelaskan spesifikasi Kualitas untuk paket pengolah kata . yang berfokus pada cara pengujian praktis atribut ini !

Ada banyak istilah yang dapat dijelaskan dan hanya dua contoh yang akan diberikan. Satu poin yang dapat muncul ketika software akan memiliki nilai yang paling rendah pada sejumlah perbedaan area fungsi, setiap elemen dapat dievaluasi secara terpisah, misalnya persiapan dokumen, penyajian, *mail merging*, misalnya:

- Quality: kemudahan pembelajaran
- Definition: waktu yang digunakan oleh *novice* (user baru) untuk mempelejari bagaimana menjalankan paket dalam menghasilkan dokumen yang standard
- Test: interview *novice* untuk memastikan pengalaman sebelumnya melalui penggunaan *word processing*. Didukung dengan sistem, software, pelatihan

manual dan dokumen standard yang telah dibuat. Waktu dibutuhkan untuk mempelajari bagaimana dokumen dapat dibuat

- Minimally acceptable: > 2.5 sampai 4 jam
- Target range: 1 sampai 2.5 jam
- Now: 3 jam

Atau

- Quality: kemudahan penggunaan
- Definition: waktu yang digunakan untuk *experienced user* (user yang berpengalaman) untuk menghasilkan dokumen yang standard
- Scale: menit
- Test: waktu yang dibutuhkan *experienced of packages* untuk menghasilkan dokumen yang standard
- Minimally acceptable: 40 sampai 45 menit
- Target range: 30 sampai 40 menit
- Current: 45 menit

Pembahasan mengenai penilaian adalah perluasan dan petunjuk dari semua pertanyaan yang belum terjawab. *Reader* dapat mengembangkan topik ini dengan membaca buku referensi lain.

5. ISO 9126

Setelah beberapa tahun, beberapa daftar karakteristik Kualitas perangkat lunak ditampilkan, seperti James McCall dan Barry Boehm. Mengetahui kesulitan pada definisi kualitas perangkat lunak yang baik dengan cara, misalnya menjadikan kesenangan kepada kesalahan perangkat lunak yang dapat ditolerir dan diperbaiki. Untuk beberapa ‘ketahanan’(robustness) yang berarti toleransi kesalahan input pada perangkat lunak, dengan kemampuan untuk merubah kode program tanpa menampilkan kesalahan. Standard ISO 9126 pertama kali diperkenalkan pada tahun 1991 melalui pertanyaan tentang definisi Kualitas perangkat lunak. Dokumen halaman-13 yang asli didesain sebagai fondasi lebih jauh, lebih detail, dan memiliki standard yang dapat diolah. Dokumen standard ISO 9126 sangat

panjang. Hal ini dikarenakan orang memiliki motivasi berbeda yang memungkinkan untuk tertarik pada kualitas perangkat lunak :

- Acquirer adalah orang yang memperoleh perangkat lunak dari supplier eksternal.
- Developer adalah orang yang membangun produk perangkat lunak.
- Evaluator independent adalah orang yang menetapkan kualitas produk perangkat lunak – tidak untuk dirinya sendiri tetapi untuk komunitas user – misalnya melalui jenis tool tertentu dari sebuah perangkat lunak sebagai bagian dari aktifitas profesional.

ISO 9126 telah membagi dokumen menjadi tiga bagian kebutuhan. Disamping ukuran bagian dokumentasi, ISO 9126 tidak hanya mendefinisikan atribut kualitas perangkat lunak. Standard ISO 14598 memisahkan prosedur yang seharusnya dibawa saat menaksir derajat produk perangkat lunak untuk menyesuaikan diri pada karakteristik kualitas ISO 9126 yang dipilih. Hal ini mungkin saja tidak diperlukan, tetapi disetujuinya ISO 14598 dapat digunakan untuk menyelesaikan penilaian dalam membedakan bagian karakteristik kualitas pada ISO 9126 yang dibutuhkan.

Perbedaan antara atribut kualitas internal dan eksternal telah dicatat, ISO 9126 juga memperkenalkan tipe kualitas – quality in use – dimana mengikuti elemen yang telah diketahui :

- Effectiveness merupakan kemampuan untuk mencapai tujuan user melalui akurasi dan kelengkapan.
- Productivity merupakan upaya menghindari kelebihan penggunaan sumber daya, seperti biaya staff dalam mencapai tujuan user.
- Safety merupakan upaya menghindari kejahatan level resiko untuk orang dan entitas lain seperti business, perangkat lunak, property dan lingkungan
- Satisfaction merupakan kepuasan user dalam menggunakan perangkat lunak.

User pada konteks ini adalah orang yang tidak hanya bekerja secara nyata pada sistem perangkat lunak yang akan dibuat, tetapi juga orang yang akan merawat dan meningkatkan perangkat lunak. Ide kualitas dalam penggunaan underlines adalah Bagaimana mempersiapkan kualitas perangkat lunak sebagai atribut yang

tidak hanya berlaku pada perangkat lunak tetapi juga pada konteks penggunaan. Mengambil skenario IOE sebagai contoh, misalnya variasi prosedur invoicing yang akan dipertimbangkan, tergantung pada tipe produk yang akan disajikan. Hal ini mungkin saja terdapat perbedaan input yang dibutuhkan pada situasi yang berbeda untuk perhitungan jumlah klien. Katakan invoices 95% yang digunakan dimiliki tipe produk A dan sisanya 5% ke produk B. Jika perangkat lunak ditulis secara khusus untuk aplikasi ini, maka di samping pengujian yang baik, beberapa kesalahan yang mungkin akan ditemukan, terdapat pada cara sistem operasional. Selagi dilaporkan dan diperbaiki, perangkat lunak mungkin saja dapat menjadi lebih 'dewasa' sehingga kesalahan perangkat lunak menjadi jarang. Hal ini terjadi jika ada kecepatan menukar antara produk B lebih mudah mengeluarkan faktur daripada peningkatan jumlah transaksi produk B. Oleh karena itu, perubahan penggunaan perangkat lunak harus melibatkan perubahan kebutuhan perangkat lunak, apa yang dapat diterima ke satu user mungkin tidak diterima oleh user lain.

ISO 9126 mengidentifikasi enam karakteristik kualitas perangkat lunak utama yaitu:

- Functionality: kemampuan menutupi fungsi produk perangkat lunak yang menyediakan kepuasan kebutuhan user.
- Reliability: kemampuan perangkat lunak untuk perawatan dengan level performansi.
- Usability: kemampuan yang berhubungan dengan penggunaan perangkat lunak.
- Efficiency: kemampuan yang berhubungan dengan sumber daya fisik yang digunakan ketika perangkat lunak dijalankan.
- Maintainanility: kemampuan yang dibutuhkan untuk membuat perubahan perangkat lunak
- Portability: kemampuan yang berhubungan dengan kemampuan perangkat lunak yang dikirim ke lingkungan berbeda.

ISO 9126 menyarankan sub-karakteristik untuk setiap karakteristik utama

Karakteristik	Sub-karakteristik
Functionality	Suitability Accuracy Interoperability Functionality compliance Security

‘Functionality compliance’ mengacu pada bagian perangkat lunak untuk mengaplikasikan standard atau kebutuhan legal. Umumnya hal ini digunakan untuk kebutuhan auditing. Sejak daftar asli 1999, sub-karakteristik yang disebut dengan ‘compliance’ telah ditambahkan pada ke-enam karakteristik ISO eksternal. Pada setiap kasus, hal ini mengacu pada standard spesifik manapun yang dapat menerapkan atribut kualitas tertentu.

‘Interoperability’ merupakan gambaran yang bagus pada usaha ISO 9126 untuk mengklarifikasi terminologi ‘interopability’ yang mengacu pada kemampuan perangkat lunak untuk berinteraksi dengan sistem lain. Kerangka ISO 9126 telah dipilih pada kata ini daripada ‘comparability’ karena nantinya akan mengakibatkan kebingungan dengan karakteristik yang dituju oleh ISO 9126 sebagai ‘replacability’

Karakteristik	Sub-karakteristik
Reliability	Maturity Fault tolerance Recoverability Reliability compliance

‘Maturity’ mengacu pada frekuensi kesalahan produk perangkat lunak yang memberikan dampak pada perangkat lunak yang digunakan sehingga kesalahan menjadi tidak nampak dan mudah dihilangkan. Hal ini menarik untuk ketahui bahwa ‘recoverability’ telah menjadi hal yang berbeda dari ‘security’ yang menggambarkan kontrol akses pada sistem.

Karakteristik	Sub-karakteristik
Usability	Understandability Learnability Operability Attractiveness Usability compliance

Catatan bagaimana ‘learnability’ dibedakan dari ‘operability’. Tool perangkat lunak dapat dengan mudah dipelajari tetapi menghabiskan waktu untuk menggunakannya dikarenakan oleh cara penggunaannya membutuhkan jumlah menu besar. Hal ini dapat diaplikasikan untuk paket yang singkat, tetapi tidak ada sistem yang menggunakannya untuk sepanjang waktu setiap hari. Pada kasus ini ‘learnability’ telah disatukan pada biaya ‘operability’.

‘Attractiveness’ adalah tambahan terbaru pada sub-karakteristik usability dan sangat penting dimana user tidak dipaksa untuk menggunakan produk perangkat lunak tertentu, Misalnya, kasus game dan produk entertainment lain.

Karakteristik	Sub-karakteristik
Efficiency	Time behaviour Resource utilization Efficiency compliance
Maintainability	Analysability Changeability Stability Testability Maintainability compliance

‘Analysability’ merupakan kemudahan untuk menentukan penyebab kesalahan. ‘Changebility’ merupakan kualitas lain dari ‘flexibility’ yang kemungkinan nantinya disebut sebagai ‘changebility’ yang memiliki arti tambahan yang sederhana dalam perbendaharan kata bahasa English – hal ini mungkin saja menandakan pemasok perangkat lunak yang selalu berubah.

Di sisi lain pengertian ‘Stability’, adalah tidak berarti perangkat lunak itu tidak pernah berubah. Hal ini berarti juga terdapat resiko yang kecil pada modifikasi perangkat lunak yang memiliki dampak tidak diduga.

Karakteristik	Sub-karakteristik
Portability	Adaptability Installibility Co-existence Replaceability Portability compliance

‘Portability compliance’ berhubungan erat dengan standard kemampuan yang digunakan pada platform manapun (*portability*). Standard bahasa pemrograman umum pada kasus ini digunakan pada lingkungan hardware/software. ‘Replacebility’ mengarah ke faktor yang memberikan ‘upward compatibility’

antara komponen software lama dan yang baru. Sedangkan ‘downwards compatibility' tidak dijelaskan pada pengertian definisi.

‘Co-existence’ mengarah pada kemampuan software untuk berbagi sumber daya dengan komponen software lain, tetapi tidak seperti ‘interopability’, dimana tidak ada data yang digunakan.

ISO 9126 menyediakan petunjuk untuk menggunakan kualitas karakteristik. Variasi dalam membedakan karakteristik kualitas tergantung pada tipe produk yang ditekankan.

Terdapat 5 langkah yang dikembangkan untuk kebutuhan produk software yaitu :

1. Menilai pentingnya masing-masing karakteristik kualitas aplikasi. Reliability akan berfokus pada sistem kerentanan keamanan (*safety-critical)* selama efisiensi itu merupakan keutamaan untuk sistem real time.
2. Memilih pengukuran kualitas eksternal menggunakan kerangka ISO 9126 yang berhubungan dengan prioritas utama kualitas. Realiability berarti memiliki waktu antara kesalahan yang akan menjadi keutamaan pengukuran, Dimana efisiensi dan lebih utamanya pada ‘*time behaviour’*, -respon waktu yang digunakan untuk pengukuran.
3. Pemetaan pengukuran pada tingkat kepuasan user. Untuk waktu respon, misalnya pemetaannya dapat dilihat pada Tabel 1.

Respon waktu (detik)	Rating
<2	Sangat puas
2-5	Puas
6-10	Cukup puas
>10	Tidak puas

Tabel 1. Pemetaan Pengukuran untuk Kepuasan User

4. Identifikasi yang berhubungan dengan pengukuran internal dan produk yang dihasilkan. Hal ini akan menjadi sesuatu yang penting saat software dikembangkan daripada mengevaluasi software yang dihasilkan. Untuk software baru, umumnya kualitas akhir produk akan membutuhkan penilaian selama pengembangan. Misalnya, pada kualitas eksternal dalam pertanyaan *time behaviour*, pada langkah desain software , perkiraan waktu eksekusi untuk sebuah proses dapat dihasilkan melalui pengujian kode software dan menghitung waktu untuk setiap perintah pada eksekusi proses secara umum.

Menurut buku ini pemetaan antara karakteristik kualitas internal dan eksternal serta pengukuran disarankan sesuai standard ISO 9126 yang setidaknya meyakinkan elemen sebuah pendekatan.. Technical report merupakan bagian dari keseluruhan standart. Standart ISO 9126, digunakan untuk pemetaan pengukuran internal dan eksternal serta validasi pengukuran yang memiliki hubungan erat diantara dua kondisi yang semestinya dilakukan. Hal ini mencerminkan masalah sebenarnya pada pengembangan software terutama pengujian struktur kode dan prediksi kualitas eksternal secara akurat seperti reliability.

Menurut ISO 9126, pengukuran dapat bertindak sebagai indikator akhir kualitas software yang dilakukan pada pengembangan life cycle yang berbeda. Untuk langkah awal dalam penentuan kualitas produk dapat bersifat kualitatif. Penentuan tersebut berdasarkan ceklist , dimana pemenuhan kriteria definisi awal dibantu oleh penilaian ekspert (*expert judgement*). Sebagai produk yang berhubungan dengan penyelesaian, sasaran, kuantitatif, maka peningkatan pengukuran akan dilakukan kemudian.

5. Penilaian keseluruhan kualitas produk. Apakah penilaian tersebut memungkinkan untuk memperoleh nilai yang menghasilkan keputusan kualitas produk software? Hal ini merupakan perdebatan utama. Dapat dilihat bahwa kualitas yang didiskusikan dapat berbeda antara satu dengan yang lainnya. Pada beberapa kasus penilaian, dapat dihadirkan hasil kualitas produk. Misalnya karakteristik efisiensi *time behaviour* dan pemanfaatan sumber daya dapat ditingkatkan dengan cara memaksimalkan karakteristik sistem operasi dan hardware untuk menghasilkan software. Bagaimanapun juga penilaian dapat mempengaruhi biaya *portability*. Faktor ketakutan lain adalah menggabungkan penilaian karakteristik jaminan yang berbeda , pada prakteknya, bisa sangat berbeda dan dapat diukur dengan cara yang berbeda, dimana membuat perbandingan dan penggabungan yang bisa sangat bermasalah.

Pengukuran telah dicatat pada penilaian kualitas yang dapat dilaksanakan untuk beberapa alasan yang berbeda : untuk menilai pengembangan software, *acquisition* atau penilaian independent.

Selama pengembangan produk software, penilaian diarahkan kepada kebutuhan yang utama dari keinginan developer terhadap kualitas. Tujuannya digunakan untuk identifikasi kemungkinan kelemahan sejak awal dan kebutuhan pada kasus yang menghasilkan pembobotan kualitas secara keseluruhan

Pada situasi ini dimana user yang berpotensi adalah menilai sejumlah perbedaan produk software untuk memilih yang terbaik sesuai keinginan mereka, hasilnya akan mengarah ke produk A yang lebih memuaskan daripada produk B atau C. Di sini terdapat beberapa ide mengenai hasil kepuasan relatif dan pertimbangan dalam berusaha untuk membuat suatu model kepuasan user. Satu pendekatan mengenali beberapa level kualitas menjadi sebuah kewajiban. Jika sebuah produk gagal meraih kewajiban level rating maka produk tersebut harus ditolak, tanpa melihat seberapa bagus kemungkinan di masa yang akan datang. Karakteristik kepuasan user mungkin saja diinginkan tetapi tidak dibutuhkan. Untuk rating kepuasan user dapat dibagi menjadi beberapa range, misalnya 0-5. Rating kepuasan user berdasarkan pada pengukuran yang obyektif untuk beberapa fungsi dan berhubungan dengan perbedaan nilai pengukuran ke level kepuasan user yang berbeda– lihat tabel 2.

Respon waktu (detik)	Skor kualitas
<2	5
2-3	4
4-5	3
6-7	2
8-9	1
>9	0

Tabel 2. Pemetaan Respon Waktu terhadap Kepuasan User

Sebagai tambahan pada rating kepuasan, rating pada range 1-5 dapat digunakan sebagai patokan untuk kepentingan setiap karakteristik kualitas, dll. Skor yang menggambarkan tentang kurang lebihnya kualitas dapat dilakukan dengan cara mengkalikan setiap rating dengan pembobotan yang diperoleh

melalui perkalian antara (*importance rating)* keutamaan rating dan nilai kualitas (*quality score*). Skor pembobotan ini dapat dijumlahkan untuk mendapatkan skor keseluruhan produk. Skor untuk produk yang berbeda dapat dipesan untuk mendapatkan pilihan skala awal. Misalnya kualitas dua produk kemungkinan dibandingkan dalam hal usability, efisiensi dan maintainaibility. Keutamaan setiap kualitas kemungkinan masing-masing bernilai 3, 4 dan 2, dengan nilai maksimum 5. Pengujian kualitas dapat menghasilkan situasi seperti pada tabel 3.

Situasi akhir , dimana penilaian kualitas dapat bertindak sebagai tim penilai yang bekerja atas nama komunitas user secara keseluruhan. Misalnya praktisi mungkin dapat menilai tool software yang mensupport kinerja anggotanya. Tidak seperti pemilihan oleh individual user / purchaser, percobaan dibuat untuk menghasilkan penilaian software yang obyektif, pada lingkungan user secara independent. Hal ini jelas bahwa hasil pengukuran akan memiliki banyak pertimbangan tergantung pada pembobotan yang diberikan pada setiap karakteristik software, dan perbedaan user akan memiliki kebutuhan nilai yang berbeda.

Produk A				Produk B	
Kualitas produk	Keutamaan rating (a)	Skor kualitas (b)	Skor pembobotan (a x b)	Skor kualitas (c)	Skor pembobotan (a x c)
Usability	3	1	3	3	9
Efficiency	4	2	8	2	8
Maintainability	2	3	6	1	2
TOTAL			**17**		**19**

Tabel 3. Pembobotan Skor Kualitas

6. Pengukuran Kualitas Perangkat Lunak secara Praktek

Dibawah ini terdapat beberapa metode yang mungkin digunakan pada kualitas tertentu. Metode ini menekankan bahwa mengukur hanyalah sebuah gambaran dan sudah pasti tidak didefinisikan. Setiap proyek membutuhkan pemikiran untuk

pengukuran ketika menemukan kebutuhan khusus. Pengukuran menggambarkan hubungan dengan produk akhir software dari sebuah proyek.

Reliability

Reliability dapat diukur melalui:

- Availability: prosentase jarak waktu tertentu yang digunakan sistem.
- Mean time between failure: jumlah total layanan waktu dibagi oleh total jumlah kesalahan.
- Failure on demand: kemungkinan bahwa sistem tidak tidak dapat dijalankan yang memungkinkan terjadi kegagalan.
- Support activity: jumlah laporan kesalahan yang telah diproses dan dihasilkan.

Latihan 3: Sistem account telah di-instal oleh kelompok IOE, dan umumnya digunakan oleh user dari jam 08.00 sampai 16.00, dari hari Senin sampai dengan hari Jumat. Setelah empat minggu sistem tidak bisa digunakan satu hari penuh karena terdapat beberapa masalah dengan disk drive dan tidak bisa dijalankan selama dua hari sampai jam 10.00 dikarenakan masalah proses batch. Apa yang dimaksud dengan *Mean time between failure* dan kebutuhan yang harus disediakan pada sebuah layanan?

Setiap hari, sistem bekerja dimulai jam 08.00 sampai 16.00, ± 10 jam

Jumlah jam selama sistem dijalankan dalam empat minggu: 10 jam x 5 hari (senin-jumat) x 4 minggu = 200 jam

Sistem tidak dapat dijalankan selama satu hari, yaitu 10 jam

Sistem tidak dapat dijalankan sampai jam 10.00 selama dua hari = 4 jam

Sehingga didapatkan: 200 – 10 – 4 = 186 jam

Availability = 186 / 200 x 100 = 93%

Jumlah tiga kesalahan yang dihitung, maka *mean time between failure* : 186 / 3 = 62 jam

Maintainability

Komponen utama dari *changebility*, yang berhubungan dengan kemudahan dalam memodifikasi software. Bagaimanapun juga, sebelum perkembangan dibuat, kesalahan seharusnya telah diketahui terlebih dahulu. Oleh karena itu *maintainability* dapat dilihat sebagai *changebility* yang ditambah kualitas baru yaitu *analysability*, untuk memudahkan identifikasi kesalahan.

Extendibility

Saat mengukur *changebility* kita dapat mengidentifikasi dua aspek: yang pertama adalah menghasilkan kode yang dapat dirubah dengan kode lainnya dan selain itu merupakan kemudahan penambahan *functionality* baru. Kita dapat menggunakan unsur sub-karakteristik *'extendibility'* dan mendefinisikan unsur tersebut sebagai produktifitas yang dibutuhkan untuk menyertakan fitur baru ke dalam sistem yang dihasilkan sebagai prosentase produktifitas normal saat pengembangan software.

Sistem billing IOE memiliki 5000 Source Line of Code dan membutuhkan 400 hari kerja untuk digunakan. Sebuah proses pengembangan sistem utama memiliki 100 SLOC dengan membutuhkan 20 hari kerja untuk implementasi, maka prosentase extendiblity adalah :

- Produktifitas untuk sistem asli = 5000/400 = 12.5 SLOC/staff-day
- Produktifitas untuk pengembangan = 100/20 = 5 SLOC/staff-day

  Extendiblity = 5/12.5 x 100 = 40%

7. Perbedaan Produk dengan Manajemen Kualitas Proses

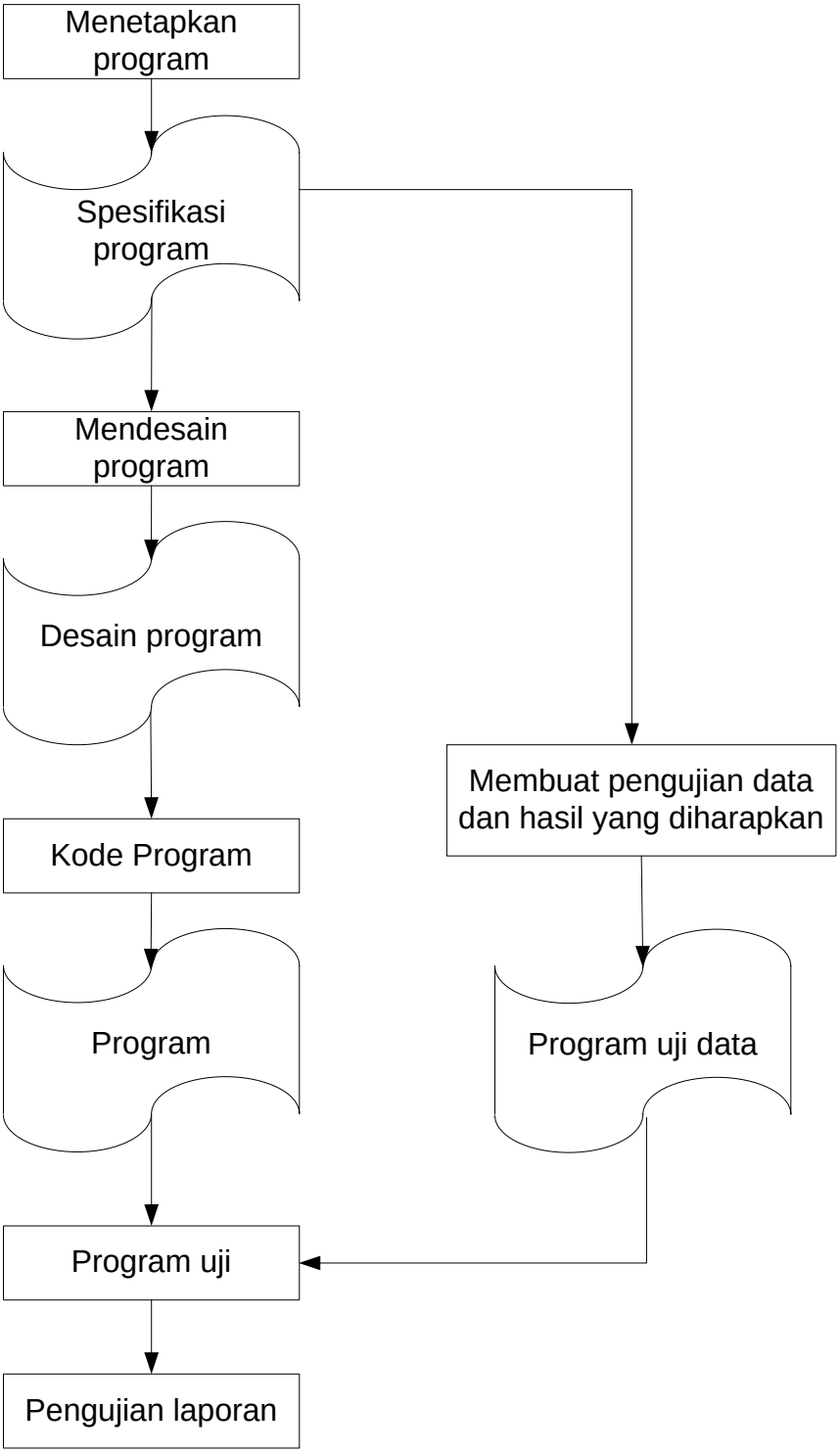
Pengukuran yang dijelaskan pada bab sebelumnya berhubungan dengan produk. Dimana berdasarkan produk sebuah pendekatan mengenai perencanaan dan pengontrolan proyek digunakan melalui metode manajemen proyek PRINCE2 – lalu berfokus ke produk yang dapat memuaskan *customer*. Metode ini lebih mudah untuk mengukur kualitas produk selama membuat aplikasi komputer dibandingkan selama proses pengembangan. Selama kita dapat mengukur atribut produk yang dibuat pada langkah awal pengembangan proyek dan mencoba

menggunakannya untuk memprediksi kualitas aplikasi akhir, tahap ini merupakan implementasi yang cukup sulit. Sebuah pendekatan alternatif akan dikembangkan pada bab ini untuk pengamatan kualitas proses selama pengembangan produk software.

Pengembangan proses sistem dibuat dari banyaknya aktifitas yang dihubungkan sehingga output dari satu aktifitas adalah input untuk proses yang berikutnya (gambar 2). Pengujian program ini tergantung pada pengujian program yang akan dilanjutkan ke langkah pengkodean program. Kesalahan dapat terjadi pada langkah proses manapun. Kesalahan tersebut dapat terjadi karena ketidaksempurnaan dari proses yang dihasilkan , atau karena informasi tidak tersampaikan dengan jelas selama langkah pengembangan.

Kesalahan yang terjadi pada langkah awal akan lebih sulit untuk diperbaiki pada langkah berikutnya, karena beberapa alasan:

- Apabila kesalahan ditemukan pada langkah akhir, maka akan membutuhkan pengulangan pekerjaan yang dimulai dari awal. Sebuah penemuan kesalahan spesifikasi pada pengujian , berarti pengulangan pekerjaan pada semua langkah antara langkah spesifikasi sampai langkah pengujian
- Kecenderungan umum untuk setiap langkah pengembangan adalah penjelasan yang lebih detail dan memiliki perubahan yang sedikit.



Gambar 2. Sebuah Contoh Urutan Proses dan Deliverables

Oleh karena itu kesalahan seharusnya dapat dikurangi oleh pengujian yang teliti pada setiap langkah sebelum melalui langkah selanjutnya. Untuk melakukan hal ini, kebutuhan proses seharusnya dijelaskan pada setiap kegiatan :

- *Entry requirement*: Penempatan sebelum aktifitas dimulai. Misalnya; Pada pengujian data dan hasil yang diharapkan memerlukan persetujuan sebelum pengujian program dimulai.
- *Implementation requirement*: adalah definisi bagaimana proses akan dipengaruhi. Misalnya Pada fase pengujian, sebuah kesalahan dapat diletakkan dimanapun sehingga mudah ditemukan dan diperbaiki, semua pengujian seharusnya diulangi, meskipun sebelumnya telah menjalani proses dengan benar.
- *Exit requirement*: Adalah aktifitas yang harus dipenuhi sebelum proses dinyatakan berakhir. Misalnya fase pengujian telah selesai, dimana semua pengujian akan dijalankan tanpa kesalahan.

Latihan 4. Dalam kondisi bagaimana sebuah aktifitas dikatakan berbeda dari hasil sebelumnya yang telah ada?

Kondisi yang memungkinkan melakukan satu aktifitas sebelum melaksanakan kegiatan yang telah selesai dilakukan. Pada kasus ini, *entry requirement* (kebutuhan input) dijalankan untuk pemenuhan kebutuhan, meskipun *exit requirement* belum dijalankan. Misalnya modul software dapat digunakan untuk pengujian performansi platform hardware meskipun berdampak pada pengurangan *screen layout*

Pelaksanaan *exit requirement* menghasilkan *entry requirement* yang membutuhan sumberdaya tertentu untuk melakukan aktifitas baru

Latihan 5. Amanda pada IOE telah memiliki kualitas manual yang dapat didiskusikan. Brigette pada Universitas Brightmouth telah melakukan spesifikasi kebutuhan masukan dan keluaran. Apa yang akan spesifikasikan oleh Amanda sebagai kebutuhan masukan dan keluaran pada proses pengkodean program seperti yang ditunjukkan pada gambar 2 di Universitas Brightmouth?

- *Entry requirement*. sebuah desain program yang harus dihasilkan dan ditinjau ulang oleh pengkaji yang nantinya dihasilkan dan diperiksa (inspeksi) oleh pimpinan
- *Exit requirement*: sebuah program harus dihasilkan dan digabungkan dan menghasilkan penggabungan yang bebas dari kesalahan pemrogaman. Kode program seharusnya dikaji ulang oleh sebuah tim yang hasilnya di inspeksi oleh pimpinan

Perlu dicatat untuk tim pengkaji ulang sebaiknya menggunakan checklist pada setiap tipe produk yang dikaji ulang dan seharusnya langkah ini ditandai sebagai pengembangan dari *entry/exit requirement*

8. Standard External

BS EN ISO 9001-2000

Pada IOE, sebuah keputusan mungkin dibuat dengan menggunakan contractor luar untuk menghasilkan tim account maintenance subsistem daripada mengembangkan software inhouse. Sebagai klien yang menggunakan layanan dari contractor luar , mereka akan mengikuti keinginan contractor dan kualitas *best practice* yang terbaik. Kondisi ini merupakan hal yang umum untuk melibatkan istilah yang digunakan oleh kontraktor. Standard variasi nasional dan internasional, termasuk British Standard Institution (BSI) pada United Kingdom, telah menjadi standard untuk sistem manajemen kualitas. British Standard sekarang disebut BS EN ISO 9001:2000 yang sama dengan standard ISO 9001:2000. Standard seperti ISO 9000 mencoba untuk memastikan melakukan pengawasan dan pengontrolan sistem untuk men-cek kualitas. ISO 9000 berfokus pada sertifikasi proses pengembangan, tidak berfokus pada produk akhir. Standard ISO 9000 menangani sistem kualitas pada istilah umum dan tidak hanya pada lingkungan pengembangan software.

ISO 9000 menggambarkan fitur dasar pada Quality Management System (QMS) dan mendefinisikan terminologi yang digunakan. ISO 9001 menggambarkan bagaimana QMS dapat diaplikasikan untuk pembuatan produk dan ketetapan layanan. ISO 9004 digunakan untuk peningkatan proses.

Ada beberapa kontroversi tentang nilai standard. Stephen Halliday, menulis di The Observer, memiliki perasaan khawatir pada standard ini dengan mengambil pendapat customer yang menyiratkan sertifikasi standard produk akhir meskipun seperti yang dikatakan Halliday "tidak ada sesuatu yang dilakukan dengan kualitas produk diluar batas". Customer menuliskan spesifikasinya sendiri dan melakukan perawatan meskipun memiliki nilai yang rendah". Hal ini juga disarankan dengan mendapatkan sertifikasi yang lebih mahal dan menghemat waktu proses, tetapi masih berjalan dengan baik. Akhirnya ada beberapa keuntungan melalui sertifikasi yang mungkin mengacaukan perhatian dari masalah sebenarnya saat menghasilkan kualitas produk.

Sebuah tugas utama untuk identifikasi kebutuhan kualitas. Pendefinisian kebutuhan sistem harus ditempatkan pada cek kebutuhan yang akan diisi dalam langkah pembetulan yang akan dilakukan jika dibutuhkan.

Panduan tentang Kebutuhan QMS BS EN ISO 9001:2000

Standard dibangun sesuai dengan :

- Pemahaman organisasi tentang kebutuhan customer sehingga dapat memenuhi kebutuhan customer.
- Kepempimpinan untuk menyediakan tujuan dan arah yang dibutuhkan untuk meningkatkan kualitas secara obyektif.
- Keterlibatan seluruh staff pada semua level.
- Berfokus pada proses pembuatan atau penyerahan produk dan layanan
- Berfokus proses yang berhubungan dengan penyerahan produk dan layanan.
- Proses peningkatan berkelanjutan.
- Pembuatan keputusan berdasarkan bukti nyata.
- Membangun keuntungan diantara supplier.

Penerapan prinsip utama melalui aktifitas berikut ini :

1. Menentukan kebutuhan dan keinginan customer.
2. Menetapkan kebijakan kualitas, yang merupakan kerangka untuk melibatkan sasaram organisasi yang berhubungan dengan kualitas yang telah didefinisikan.

3. Mendesain proses yang akan menghasilkan produk dan layanan, yang menampakkan kualitas sasaran organisasi.
4. Mengalokasikan pertanggung jawaban untuk memenuhi kebutuhan untuk setiap tahapan proses.
5. Memastikan sumberdaya yang cukup untuk menjalankan proses dengan baik.
6. Mendesain metode untuk mengukur efektifitas dan efisiensi setiap proses yang akan berkontribusi ke kualitas sasaran organisasi.
7. Mengumpulkan pengukuran.
8. Mengidentifikasi tentang segala perbedaan antar pengukuran nyata dengan nilai target.
9. Menganalisis penyebab perbedaan dan menguranginya dengan langkah nyata.

Prosedur di atas seharusnya didesain dan dijalankan pada peningkatan yang berkelanjutan. Jika berjalan dengan baik, maka akan menghasilkan QMS yang efektif. Kebutuhan ISO 9001 lebih detail meliputi:

- *Documentation* of objective, prosedur (pada form quality manual), perencanaan dan dokumentasi berhubungan dengan proses pengoperasian secara nyata. Dokumentasi sebagai bahan utama untuk pengontrolan sistem utama yang dengan pasti dapat berjalan. Secara sederhana dapat dikatakan bahwa kebutuhan bisa ditampilkan ke pihak lain yang menghasilkan QMS.
- *Management responbility* – kebutuhan organisasi untuk menampilkan QMS dan proses yang menghasilkan barang dan layanan membutuhkan kesesuaian pengaturan sasaran kualitas secara aktif dan baik.
- *Resource* – sebuah organisasi harus memastikan memiliki sumber daya yang cukup, termasuk pelatihan staff dan infrastruktur , yang akan digunakan pada langkah proses .
- *Production* seharusnya memiliki karakteristik:
  - Perencanaan.
  - Penentuan dan tinjau ulang kebutuhan customer.
  - Komunikasi efektif antara customer dan supplier.
  - Desain dan pengembangan yang menjadi prinsip perencanaan, kontrol dan review.

- Kebutuhan dan informasi tambahan pada langkah desain yang disimpan dengan baik dan jelas.
- Hasil desain yang telah di verifikasi, divalidasi dan di dokumentasikan dengan menyediakan informasi cukup untuk semua orang menggunakan desain tersebut; perubahan desain seharusnya dikontrol dengan baik.
- Komponen telah diperoleh , seharusnya ada pengukuran cukup untuk membuat spesifikasi dan evaluasi dari kualitas yang dimiliki.
- Produksi barang dan ketetapan layanan seharusnya berada di bawah kondisi pengaturan – kondisi ini meliputi ketetapan informasi, instruksi kerja, peralatan, pengukuran peralatan dan aktifitas *post-delivery*.
- Pengukuran – untuk menampilkan standarisasi produk dan QMS yang efektif serta peningkatan efektifitas proses pembuatan produk dan layanan

Latihan 6. Salah satu proses yang dipilih dalam pengembangan pengujian dan modifikasi software berikut adalah untuk menemukan kesalahan. Bagaimana seharusnya dampak pengujian sistem dalam menemukan kesalahan yang sesuai standard BS EN ISO 9001:2000?

Ada beberapa prosedur dokumentasi yang mengatur pengujian sistem. Tujuan kualitas untuk pengujian sistem dapat dipastikan sebagai sarana penyesuaian software kepada kebutuhan *user*.

Beberapa proses dilakukan untuk memenuhi kebutuhan dokumentasi *cross-reference* (apakah telah sesuai dengan referensi) dari bagian kasus pengujian spesifikasi

Hasil pengujian membutuhkan pendokumentasian kemudian dilakukan perbaikan yang nantinya juga akan didokumentasikan

Latihan 7. Menghadapi pemikiran kritis BS EN ISO 9001 yang telah dijelaskan, Jelaskan langkah pencegahan yang dilakukan oleh manajer proyek untuk kepentingan hasil yang berkualitas ?

Proyek manajer dapat melakukan pemeriksaan yang menghasilkan standarisasi. Mereka juga menemukan area kerja yang perlu distandarisasi oleh BS EN ISO 9001. Misalnya standarisasi digunakan pada proses pembuatan produk dan bukan lainnya. Poin utamanya adalah proyek manajer akan membutuhkan ketetapan untuk disampaikan ke kontraktor sebagai pedoman bagi kebutuhan organisasi klien.

Pemodelan Proses Kemampuan

Dibandingkan hanya memeriksa sistem dalam menemukan kesalahan, customer mungkin saja berharap untuk dapat melihat metode dan tool yang digunakan supplier dalam menghasilkan produk software yang berkualitas. Customer mungkin saja lebih yakin, jika ia mengetahui struktur metode software yang digunakan oleh supplier. Di United State, sejumlah pengembangan Capability Maturity Model (CMM) telah diterapkan pada Software Engineering Institute (SEI), bagian dari Carniege-Mellon University, yaitu SW-CMM yang berhubungan dengan pengembangan software. Pengembangan terakhir dalam pembuatan versi baru CMM, adalah CMM Integration atau CMMI, yang akan menyertakan perbedaan model yang diaplikasikan pada lingkungan yang berbeda ke dalam satu kesatuan sistem. Model ini menempatkan organisasi pada lima level proses pendewasaan yang memiliki indikasi kenyamanan dan kualitas produk. Lima level ini didefinisikan sbb:

- Level 1: initial. Prosedur memiliki aturan yang belum jelas. Beberapa proyek mungkin saja berhasil, tetapi karena kemampuan individual tertentu, termasuk proyek manajer. Tidak ada level 0 sehingga semua organisasi akan berada di level ini sebagai proses awal.
- Level 2: managed. Organisasi pada level ini memiliki prosedur dasar manajemen produk. Bagaimanapun juga, cara kerja individual yang dihasilkan tergantung pada apa yang dikerjakan orang.
- Level 3: defined. Organisasi memiliki cara untuk setiap pada pengembangan life cycle software yang perlu dilakukan.

- Level 4: quantitatively managed. Produk dan proses yang diikutkan pada pengembangan software sebagai hal yang utama untuk pengukuran dan pengontrolan.
- Level 5: optimizing. Prosedur peningkatan yang dapat didesain dan diimplementasikan dengan menggumpulkan data dan proses pengukuran.

Untuk setiap level, bagian dasar dari level 1, Key Process Area (KPA) telah diidentifikasi sebagai level sekarang dengan level sebelumnya. Hal ini terdapat pada tabel 4.

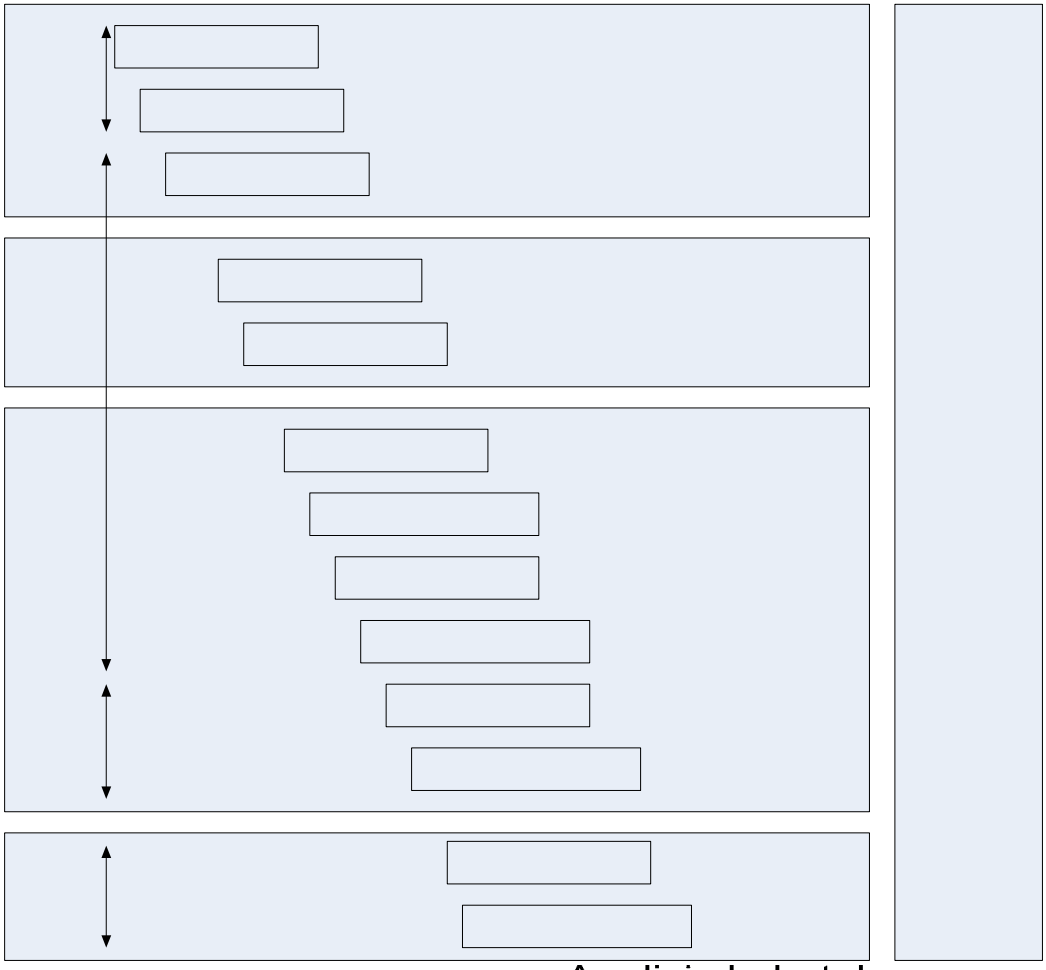
Level	Proses Area Utama
1. Initial	▪ Tidak bisa diterapkan.
2. Managed	▪ Kebutuhan manajemen, perencanaan proyek dan pengawasan dan pengontrolan, manajemen perjanjian supplier, pengukuran dan analisis, proses dan jaminan kualitas produk, majamen konfigurasi
3. Defined	▪ Kebutuhan pengembangan, solusi teknis, integrasi produk, verifikasi, validasi, pendefinisian fokus proses, pelatihan, manajemen integrasi proyek, manajemen resiko, pengintegrasian tim, manajemen integrasi supplier, analisis keputusan dan resolusi, integrasi lingkungan organisasi
4. Quantitatively Managed	▪ Performansi proses organisasi, manajemen proyek secara kuantitatif.
5. Optimizing	▪ Inovasi dan pengembangan organisasi, analisis dan resolusi sebab-akibat.

Tabel 4. Proses Area Utama CMMI

Penilaian dilakukan oleh tim penilai dari organisasi dan me-wawancarai staff mengenai kinerja mereka, menggunakan quesioner standard untuk mendapatkan informasi. Sasaran utama tidak hanya untuk menilai, tetapi juga merekomendasikan langkah khusus untuk membawa organisasi ke level yang lebih tinggi.

Proses Penilaian ISO 15504

ISO/IEC 15504 merupakan standard untuk proses penilaian yang memiliki kesamaan (share) dengan konsep CMMI. Ada dua standard yang seharusnya disesuaikan. Seperti standard CMMI yang didesain untuk menyediakan petunjuk penilaian proses pengembangan software. Untuk melakukan hal ini maka diperlukan beberapa *benchmark* atau yang berbeda dengan proses yang dapat dibandingkan. *Process reference model* menunjukkan pengembangan life cyle ideal yang bertentangan dengan proses nyata yang dibandingkan. Variasi *process reference model* dapat digunakan tetapi model dasar pada ISO 12207 yang telah dijelaskan pada bab 1 dan 10 yang menggambarkan proses utama – seperti analisis kebutuhan dan desain arsitektur – pada pengembangan software life cycle klasik.



Gambar 3. Pengembangan Life Cyle Software ISO 12207

Analisis kebutuhan

Sistem

Desain arsitektur

Analisis kebutuhan

- Analisis kebutuhan – dimulai dengan pengumpulan kebutuhan yang menyelidiki sejauh mana potensi user, manajer dan karyawan diperlukan sebagai fitur dan kualitas sistem yang baru. Analisa kebutuhan berhubungan dengan sistem secara keseluruhan. Apa yang menjadi kebutuhan kualitas yang digunakan user sebagai tambahan seharusnya dapat diselesaikan pada waktu tertentu. Pada kasus ini waktu pemrosesan akan dipengaruhi oleh ketrampilan manusia, seperti halnya pada kemampuan hardware dan software. Kebutuhan menghadapi *customer (customer facing)* yang tidak diterjemahkan ke dalam kebutuhan teknis dari pengembang sistem yang baru bekerja.
- Desain arsitektur – pemetaan kebutuhan ke dalam komponen sistem yang dibangun. Pada level ini, keputusan dibuat berdasarkan proses sistem baru yang akan dihasilkan dan diperhitungkan oleh user. Desain sistem arsitektur sebagai masukan dalam pengembangan *software requirement*. Kedua proses arsitektur desain diambil dari pemetaan *software requirement* ke *software component*
- Penjelasan desain – setiap komponen software terbuat dari beberapa sub-komponen software yang dari dikodekan dan diuji secara terpisah. Penjelasan desain sub-komponen secara terpisah telah dihasilkan.
- Kode dan pengujian – hal ini didasarkan pada penulisan unit kode software pada bahasa pemrograman seperti C# atau Java, atau juga mengarah ke penggunaan *application-builder* seperti Microsoft Access. Pengenalan pengujian untuk kesalahan software dihasilkan pada langkah ini.
- Integrasi – komponen individual yang dikumpulkan bersama dan diuji untuk melihat apakah pemenuhan kebutuhan telah tercapai. Integrasi dapat dicapai pada level software dimana perbedaan komponen software itu dikombinasikan, atau pada level sistem sebagai keseluruhan software dan komponen lain dari sistem seperti platform hardware dan network serta prosedur user yang dilakukan bersama.
- Kualifikasi pengujian – sistem, termasuk komponen software, harus diuji secara hati-hati untuk memastikan semua kebutuhan telah terpenuhi.

- Instalasi – Langkah ini merupakan pembuatan sistem operasional baru. Kegiatan instalasi hampir sama dengan melakukan pengaturan data (seperti sistem payroll). Kegiatan instalasi termasuk juga dalam parameter pengaturan sistem, instalasi software ke dalam platform hardware dan pelatihan user.
- Acceptance support (ketersediaan sistem pendukung) – langkah ini merupakan langkah penyelesaian dengan menggunakan instalasi sistem terbaru, termasuk pembetulan kesalahan yang mengganggu sistem, dan perluasan serta peningkatan yang dibutuhkan pada langkah ini. Hal ini memungkinkan untuk melihat perawatan software sebagai bagian kecil software. Pada lingkungan dimanapun, kebanyakan pengembangan software hanya merupakan kegiatan perawatan.

Setiap proses yang dinilai akan diputuskan sesuai dengan sembilan atribut proses dasar, sesuai pada tabel 5

Level	Atribut	Komentar
0. Incomplete		▪ Proses tidak diimplementasikan atau tidak berhasil
1. Performed Process	1.1 Process Performance	▪ Prosedur yang mendefinisikan hasil
2. Managed Process	2.1 Performance Management	▪ Proses yang telah direncanakan dan diawasi
	2.2 Work Product Management	▪ Hasil kerja yang telah didefinisikan dan dikaji ulang untuk memastikan kebutuhan telah terpenuhi
3. Established Process	3.1 Process Definition	▪ Proses yang menghasilkan penjelasan secara teliti
	3.2 Process Deployment	▪ Proses yang telah dijelaskan pada langkah sebelumnya
4. Predictable Process	4.1 Process Measurement	▪ Target pengukuran secara kuantitatif yang menjadi bagian setiap sub-proses dan pengumpulan data pada pengawasan performansi
	4.2 Process Control	▪ Pada dasar pengumpulan data di langkah 4.1, tindakan pembetulan telah diambil jika terdapat ketidaksesuaian variasi dengan target
5. Optimizing	5.1 Process Innovation	▪ Sebagai hasil pengumpulan di langkah 4.1, peluang untuk mengetahui perkembangan proses
	5.2 Process Optimization	▪ Peluang untuk peningkatan proses yang dievaluasi dengan baik dan diimplementasikan secara efektif

Tabel 5. Kemungkinan Atribut Proses Indikator untuk ‘Analisis Kebutuhan’

Saat tim penilai menyetujui peningkatan level atribut proses yang memiliki arti sesuai dengan nilai di bawah ini:

Level	Arti
N – Not achieved	Peningkatan 0-15 %
P – Partially achieved	Peningkatan > 15%
L – Largely achieved	Peningkatan 50-85 %
F – Fully achieved	Peningkatan > 85%

Untuk penilaian proses atribut sebagai kepastian peningkatan level, memerlukan penyediaan indikator sebagai bukti penilaian. Untuk melihat kemungkinan kerja dicontohkan pada proses *requirement analysis*. Pada tabel 6, contoh penggunaan indikator ditunjukkan pada setiap proses atribut yang berhubungan dengan *requirement analysis*.

Atribut proses	Contoh indikator
1.1 Process Performance 2.1 Performance Management 2.2 Work Product Management 3.1 Process Definition 3.2 Process Deployment 4.1 Process Measurement 4.2 Process Control 5.1 Process Innovation 5.2 Process Optimization	▪ Menghasilkan dokumen analisis kebutuhan ▪ Menghasilkan perencanaan analisis kebutuhan yang selalu *up to date* sebagai penyelesaian atau perubahan perencanaan yang telah dibuat ▪ Menghasilkan standard template dan pertemuan untuk mengkaji ulang dokumen analisis ▪ Menghasilkan prosedur manual yang mempengaruhi bagian analisis kebutuhan ▪ Menghasilkan dokumen pengontrolan yang telah disetujui sebagai langkah proses analisis kebutuhan, termasuk pemeriksaan kualitas ▪ Menghasilkan data tentang performansi analisis kebutuhan dan dapat diujikan ▪ Menghasilkan sasaran performansi kuantitatif dari analisis kebutuhan yang telah diidentifikasi dan didokumentasikan, misalnya adanya beberapa kesalahan dalam pembuatan software yang disebabkan oleh kesalahan analisis ▪ Menghasilkan laporan‘*lesson learnt*’ pada akhir proyek yang menjelaskan proses pengembangan ▪ Menghasilkan dokumen kelayakan tentang pengajuan peningkatan proses dan menunjukkan rekomendasi dari *stakeholder* tentang implementasinya

Tabel 6. Kemungkinan Atribut Proses Indikator untuk ‘Analisis Kebutuhan’

9. Tehnik Meningkatkan Kualitas Perangkat Lunak

Sampai dengan bab ini telah membahas kemungkinan customer dalam memastikan kualitas perangkat lunak yang diperkenalkan oleh supplier. Saat ini kita memerlukan tehnik bagi tim proyek dalam membantu meningkatkan proses pengembangan software. Terdapat tiga tema yang dapat dikemukakan yaitu:

- *Increasing visibility* – yang paling utama pada pengembangan proses pembuatan software yang telah diperkenalkan oleh American Software guru, Gerald Weinberg, 'Vegoless programming'. Weinberg yang melalui pelatihan pemrograman sederhana dalam mencari setiap kode
- *Practical structure* – pada awalnya programer sedikit banyak menulis program dengan menggunakan petunjuk umum. Setelah beberapa tahun kemudian telah berkembang metodologi pada setiap proses tahapan pengembangan software
- *Checking intermediate stages* – melihat keterkaitan dengan kehidupan manusia untuk mendukung percepatan pengembangan unit sasaran dalam kerangka model 'kerja' yang belum sempurna, masih terdapat 'kesalahan'. Salah satu cara untuk mengadakan pelatihan kualitas adalah melakukan pemeriksaan ketepatan langkah awal pekerjaan

Pada beberapa tahun terakhir, telah dilakukan pemeriksaan produk pada pertengahan proses dan mengarah ke pembuatan aplikasi sejauh mungkin pada komponen yang relatif bisa dikembangkan secara cepat dan diuji lebih awal. Sebagai catatan, kondisi ini dapat menghindari beberapa masalah dalam memprediksi kualitas software eksternal dari desain dokumen awal.

Saat ini peningkatan kualitas dapat dilakukan melalui dua hal yaitu inspeksi dan review yang pergerakannya lebih mengarah ke struktur prosedur untuk membahas struktur tehnik pemrograman yang nantinya menghasikan ide pengembangan software clean-room

Peningkatan kualitas produk oleh Negara Jepang telah menjadi bahasan dalam penggunaan tehnik kualitas produk, seperti variasi tema inspeksi dan pengembangan clean-room, tetapi dilihat dari sudut yang berbeda.

Inspeksi

Prinsip inspeksi dapat diperluas pada dokumen dan langkah manapun dalam proses pengembangan. Sebagai contoh, kasus pengujian membutuhkan review – kegiatan produksi yang kurang memiliki standard kualitas kesalahan dapat diabaikan dalam pengerjaan operasional disebabkan oleh kualitas yang rendah.

Inspeksi dilakukan saat proses pekerjaan itu berlangsung, kemudian dikerjakan oleh tim pelaksana dokumentasi. Pertemuan dilakukan untuk membahas pekerjaan dan membuat daftar kebutuhan pekerjaan perbaikan. Suatu pekerjaan dapat dilanjutkan apabila program tidak memiliki daftar kesalahan.

Beberapa aktifitas yang menggunakan tehnik ini adalah:

- Mencari solusi dalam mengurangi kesalahan
- Memotivasi developer untuk menghasilkan struktur yang lebih baik dan software yang mudah dijelaskan (*self-explanatory*) sehingga orang lain dapat memberikan kritik
- Mengadakan pelatihan *good-programming* sebagai tempat untuk membahas kualitas produk
- Meningkatkan semangat tim

Aktifitas di atas akan selalu dikaji ulang oleh rekan kerja yang melibatkan beberapa departemen, termasuk departemen pemrograman misalnya melakukan review terhadap produk yang melibatkan programmer. Untuk mengurangi masalah komunikasi dapat dilakukan melalui pembahasan perbedaan langkah awal penyajian dengan hasil pekerjaan yang telah dikaji ulang

IBM menggunakan proses review pada struktur dan bentuk dasar serta menghasilkan data statistik yang menunjukkan efektifitas proses. Inspeksi Fagan (pertama kali digunakan di IBM) merupakan awal mula prosedur baku, tanpa jaminan kerja tetapi melalui pelatihan khusus ‘moderator’

Prinsip Umum selain Metode Fagan

- Inspeksi dilakukan pada setiap tindakan pengiriman
- Semua tipe dicatat – baik kesalahan logis maupun fungsi
- Inspeksi dapat dilakukan oleh semua level pekerja kecuali pada level paling atas
- Inspeksi menggunakan pendefinisian langkah awal (*predefined*)
- Inspeksi membutuhkan waktu tidak lebih dari dua jam
- Inspeksi dipimpin oleh *moderator*, yang telah mengikuti pelatihan tehnik khusus
- Peserta lain memiliki peran, misalnya satu orang akan bertindak sebagai pencatat (*recorder*), yang lain mencari kesalahan, dan yang lainnya akan bertindak sebagai pembaca (*reader*) dan peserta lain melakukan inspeksi melalui dokumen
- Daftar ceklist digunakan untuk proses penilaian penemuan kesalahan
- Materi yang di diperiksa dengan kecepatan 100 baris per jam
- Data statistik perlu dilakukan pengawasan sehingga proses lebih efektif

Latihan 8. Bandingkan proses review dengan pair programming yang didukung oleh Xtreme Programming

Pair programming	Peer Review
▪ Akan lebih baik bila suatu pemrograman dikerjakan oleh sebuah tim ▪ *Driver* dan *navigator* bertanggung jawab bersama untuk menghasilkan produk software ▪ Menghasilkan desain program ▪ Interaksi *real time* antar peserta ▪ Usaha maksimal pada pengembangan produk software	▪ Kelompok *peer review* dapat beranggotakan beberapa orang ▪ *Developer* bertanggung jawab untuk pembuatan produk awal yang nantinya akan dikaji ulang ▪ *Reviewer* hanya melihat produk akhir ▪ Orientasi batch akan berfokus pada dokumentasi ▪ Waktu yang dibutuhkan untuk beberapa staff dalam mempelajari dokumentasi dan mengadakan seminar (*review meeting*)

Latihan 9. Kerjakan pada kelompok. Pilih salah satu rekan kerjamu sebagai penulis. Pilih moderator, *reader* dan *recorder*. Gunakan waktu 20 menit untuk membuat daftar pengujian program secara individual dan lakukan review bersama-sama dari kode yang telah digabungkan

Struktur Pemrograman dan Pengembangan Software Clean-room

Salah satu orang yang berhubungan dekat dengan struktur pemrograman adalah Djikstra. Pada akhir 1960, perkembangan software lebih kompleks dalam penggunaannya. Hal ini mungkin disadari bahwa pengujian software dilakukan secara menyeluruh – ada banyak kemungkinan kombinasi input. Pengujian sebaiknya dapat membuktikan sebuah kesalahan. Saran ini diberikan oleh Djikstra karena dapat menyelesaikan masalah mengenai ketepatan software dalam mencari kode secara nyata.

Memperluas pemikiran manusia merupakan tantangan dalam berhubungan dengan sistem komplek. Untuk sistem besar yang memiliki komponen hierarki dan sub-komponen, diperlukan dekomposisi agar bisa bekerja dengan baik, dimana setiap komponen akan berisi self-contained dengan satu input dan satu output.

Ide struktur pemrograman akan dibahas lebih jauh dalam pengembangan software clean-room oleh Harlan Mills dari IBM. Dengan tipe pengembangan software clean-room maka terdapat tiga bagian tim yaitu:

- *a specification team* adalah tim yang mendapatkan kebutuhan user dan dapat memperkirakan jumlah penggunaan setiap fitur dalam sistem
- *a development team* adalah tim yang mengembangkan kode tetapi tidak menghasilkan mesin penguji kode program
- *a certification team* adalah tim yang menyelesaikan pengujian

Apapun yang dihasilkan dalam peningkatan sistem, setiap sistem seharusnya dapat dijalankan oleh *end user*. Tim pengembang tidak hanya menguji kesalahan, semua software akan diverifikasi dengan tehnik matematis. Software dibuat melalui kode program, lalu diuji datanya melalui sistem dan membuat tahapan pengembangan program.

Sertifikasi tim menghasilkan pengujian yang berkelanjutan sampai model statistik yang akan menampilkan banyaknya kesalahan sesuai dengan level yang dapat diterima.

Metode Formal

Pada pengembangan clean-room menggunakan tehnik verifikasi matematis. Tehnik yang digunakan sangat jelas sesuai dengan dasar matematis, sebagai contoh yaitu spesifikasi bahasa Z dan VDM. Mereka menggunakan definisi *pre-defined* dan *post-condition* untuk setiap prosedur. *Pre-condition* mendefinisikan status yang diijinkan (*allowable states*), sebelum proses berlangsung pada variasi item data dalam tahapan pekerjaan. *Post-condition* mendefinisikan kondisi item data setelah prosedur dijalankan. Karena notasi matematis yang tepat, spesifikasi ditampilkan dengan jelas. Notasi matematika seharusnya menampilkan bukti matematis (dengan cara yang sama pada teorema 'Pythagoras') sesuai algoritma tertentu yang akan bekerja melalui data yang terdefinisi oleh *pre-condition.* Dengan langkah yang sama seperti pada prosedur *post-condition*. Akan lebih jelas jika dilakukan pada kasus yang lebih luas. Pada struktur pemrograman, yang berorientasi pada pengembangan obyek, dapat menampilkan analisa struktur permrograman yang lebih rinci, *self-contained* adalah prosedur yang memberikan saran pada pendekatan formal.

Perputaran Kualitas Perangkat Lunak

Telah menunjukkan sesuatu hal yang menarik dalam pelatihan kualitas perangkat lunak di Negara Jepang. Tujuan pendekatan ala Jepang digunakan untuk menjalankan dan membuat perubahan dalam pengembangan proses untuk mengurangi jumlah kesalahan pada produk akhir. Pengujian dan Inspeksi Fagan dapat membantu mengurangi kesalahan – tetapi kesalahan yang sama dapat terjadi secara berulang pada proses pembuatan produk secara berurutan. Dengan mengabaikan kesalahan, pengulangan langkah pengujian dan Inspeksi Fagan dapat dikurangi.

Keterlibatan staff pada identifikasi kesalahan diletakkan pada kerangka *quality circle*. Hal ini digunakan pada semua departemen organisasi dalam menghasilkan software yang dikenal dengan Software Quality Circle (SWQC)

Perputaran kualitas dijalankan oleh empat sampai sepuluh pekerja sukarelawan dalam area yang sama untuk proses identifikasi, analisis dan pemecahan masalah. Pada satu kelompok terdapat satu *facilitator* adalah orang yang memberikan saran pada permasalahan tentang prosedur. Dalam pembuatan perputaran kualitas membutuhkan pelatihan yang efektif.

Secara bersama-sama mengelompokkan kualitas untuk menyelesaikan masalah yang memberikan dampak pada kinerja mereka. Kelompok tersebut mengidentifikasi penyebab masalah dan memutuskan langkah nyata untuk menyelesaikan masalah tersebut. Seringkali dikarenakan memperoleh hambatan organisasi untuk menghadirkan ide dalam melakukan pengaturan, mereka memerlukan persetujuan sebelum melakukan pelaksanaan proses pengembangan

Latihan 10. Apa perbedaan utama antara perputaran kualitas dan tim review?

Proses secara umum dalam perputaran kualitas pada tim review akan menemukan produk tertentu. Kegunaan tim review sendiri dapat menjadi tidak efisien karena tim tersebut dapat mengulangi tipe kesalahan yang sama daripada hanya mengelompokkannya, dan perputaran kualitas berguna untuk menemukan sumber kesalahan

Brainstrorming merupakan tehnik tertentu yang berhubungan dengan perputaran kualitas. Pokok masalahnya adalah banyak membutuhkan ide dan suatu kelompok menyarankan beberapa ide yang memungkinkan untuk dilaksanakan. Ide yang disarankan akan ditulis pada flip-chart. Angggota lain pada kelompok tersebut, tidak boleh berkomentar. Pada akhir sesi mendapatkan daftar ide dan memadukan ide yang hampir sama (overlapping). Umumnya tehnik ini digunakan untuk memperoleh daftar masalah atau daftar kemungkinan pemecahan masalah.

Begitu pula dengan perputaran kualitas yang merupakan daftar kumpulan (daftar kesalahan sangat mungkin terjadi) *most probable error list*. Misalnya IOE,

Amanda menemukan penundaan perawatan karena kesalahan pada penentuan kebutuhan. Tim proyek seharunya mendampingi dan menghabiskan banyak waktu untuk menghasilkan daftar kesalahan yang terjadi pada penentuan kebutuhan. Kemudian digunakan untuk mengidentifikasi pengukuran yang dapat mengurangi kesalahan pada waktu yang bersamaan sebagai penentuan kebutuhan dan pengujian yang seharusnya dilakukan pada saat inspeksi. Cara lain dapat menggunakan ceklist untuk menentukan kebutuhan.

Latihan 11. Kerjakan dalam satu kelompok. Pilih area latihan tertentu dimana permasalahan telah muncul terlebih dahulu. Misalnya jika kamu berada di satu kelompok murid, kamu dapat menggunakan pelatihan atau modul yang kamu sampaikan, atau tugas terakhir yang baru saja kamu selesaikan. Dengan Brainstorming, identifikasikan semua permasalahan dari semua anggota kelompok yang hadir. Pada sesi akhir Brainstorming kelompokkan masalah yang sama dan kombinasikan apabila terjadi overlapping. Aktifitas lebih jauh dapat dilakukan dengan memilih masalah dan ide brainstrom digunakan sebagai solusi.

Efektifitas Perputaran Kualitas

Dalam perputaran kualitas harus didukung penuh oleh seluruh level manajemen. Manajemen *First-line* dapat membuat sebuah kebijakan baru sesuai dengan kapasitas mereka. Lagipula, pemecahan masalah tetap menjadi tugas utama. Konsultan dari perputaran kualitas melihat karyawan sebagai cara memberikan perubahan pada manajemen terlebih lagi pada saat perencanaan dan pengembangan. Beberapa manajer akan menyediakan waktu '*fire-fighting*', berhubungan dengan munculnya masalah (ad hoc) sehingga dapat mengurangi aktifitas *long-term* yang dapat meningkatkan efektifitas organisasi.

10. Perencanaan Kualitas

Beberapa organisasi menghasilkan *quality plan* untuk setiap proyek. Intinya menunjukkan bagaimana standard prosedur kualitas dan standard tersebut seharusnya ditulis pada (manual mutu organisasi: panduan organisasi dalam mencapai mutu yang diinginkan) *quality manual organization* yang akan diaplikasikan ke proyek tertentu. Jika pendekatan untuk perencanaan seperti Step Wise, aktifitas yang berhubungan dengan kualitas dan kebutuhan seharusnya di identifikasi melalui proses perencanaan pokok dan perencanaan kualitas yang secara terpisah tidak dibutuhkan. Saat software dihasilkan untuk customer eksternal, jaminan kualitas staff membutuhkan perencanaan kualitas yang sama untuk memastikan pengiriman produk memiliki kualitas yang unggul. Perencanaan kualitas dibutuhkan sebagai semacam daftar/ceklist dari kepentingan kualitas yang berhubungan dengan proses perencanaan. Pada kasus ini kebanyakan akan mengacu pada dokumen lain.

Perencanaan kualitas terdiri dari:

- Tujuan – ruang lingkup perencanaan
- Daftar referensi ke dokumen lain
- Pengaturan manajemen, termasuk organisasi, tugas dan pertanggung jawaban
- Dokumentasi yang dihasilkan
- Standard, latihan dan persetujuan
- Review dan audit
- Pengujian
- Masalah laporan dan tindakan koreksi
- Tool, tehnik dan metodologi
- Kode, media dan kontrol supplier
- Kumpulan dokumentasi, perawatan dan hambatan
- Pelatihan
- Manajemen resiko – metode manajemen resiko yang telah digunakan

11. Kesimpulan

Poin utama untuk mengingat semua mengenai kualitas perangkat lunak yaitu:

- Kualitas itu sendiri adalah konsep yang meragukan dan kebutuhan kualitas secara praktis perlu didefinisi secara hati-hati
- Terdapat cara yang praktis dalam pengujian yang berhubungan dengan ada atau tidaknya kualitas
- Sebagian besar kualitas software yang muncul ke user hanya dapat diuji saat sistem telah selesai dikerjakan
- Dibutuhkan pemeriksaan selama pengembangan sistem mengenai bagaimana kualitas yang seharusnya dihasilkan
- Beberapa kualitas - meningkatkan tehnik yang berfokus pada proses pengujian produk dalam proses pengembangan untuk evaluasi kualitas pengembangan proses yang digunakan.

12. Latihan

1. Sebuah organisasi membicarakan pembelian tool proyek perencanaan software seperti MS Project, dan telah diputuskan untuk menspesifikasikan kualitas paket tersebut

   Fitur ini akan berfokus pada:

   - Pengaturan proyek baru secara detail
   - Alokasi sumberdaya yang dibutuhkan untuk menyelesaikan proyek
   - Melakukan perbaruan pada proyek secara detail dengan informasi yang komplit dan aktual
   - Penyajian perencanaan secara efektif

   Gambarkan spesifikasi kualitas yang berkaitan dengan

   - Usability
   - Reliability
   - Recoverability

2. Dengan mengikuti kutipan dari laporan yang dihasilkan dari sistem

Modul	Tanggal Laporan Kesalahan	Pembetulan Kesalahan	Usaha (jam)
AA247	1.4.2004	2.4.2004	5
AA247	10.4.2004	5.5.2004	4
AA247	12.4.2004	5.5.2004	3
AA247	6.5.2004	7.5.2004	2

   Penilaian maintainability modul AA247 dari pandangan

   - Manajemen user
   - Manajemen developer

3. Diskusikan seberapa pentingnya pengukuran

   a. Jumlah kesalahan yang dihasilkan pada perpaduan awal dari program
   b. Usaha untuk implementasi perubahan yang diminta oleh user pada sistem
   c. Prosentase baris program yang dikomentari
   d. Jumlah halaman pada dokumen kebutuhan

4. Bagaimana kita dapat mengukur efektifitas dari petunjuk penggunaan paket software? Pertimbangkan kedua pengukuran yang dapat diaplikasikan dan prosedur pengukuran yang mungkin dapat dilakukan

5. Kemungkinan input apa yang digunakan, implementasi dan kebutuhan yang dihasilkan untuk proses *design program structure*?
6. Pada tabel 6, indikator yang memungkinkan memberikan saran pada proses analisis kebutuhan melalui kepuasan organisasi pada variasi proses level atribut. Tunjukkan kesamaan indikator yang dapat diidentifikasi untuk proses pengujian
7. Identifikasi tugas yang dilakukan sebagai bagian dari pekerjaanmu sehari-hari. Dimulai dari identifikasi input, proses dan hasil
8. Apakah kebutuhan BS EN ISO 9001 telah memenuhi kebutuhan untuk konfigurasi manajemen sistem secara efektif?