Jurnal Sistem Informasi (JSI), VOL. 1, NO. 3, Desember 2009, *Halaman 100-112*
ISSN Print : 2085-1588
ISSN Online : 2355-4614
http://ejournal.unsri.ac.id/index.php/jsi/index

MEMASYARAKATKAN DAN MEMPEROLEH CONCERN AKAN *SOFTWARE QUALITY*, Sebagai FAKTOR PENDORONG PENERAPAN CMMI atau CMM-SW

Ari Wedhasmara
Jurusan Sistem Informasi
Fakultas Ilmu Komputer Universitas Sriwijaya
E-mail: a_wedhasmara@unsri.ac.id, a_wedhasmara@yahoo.com

Abstrak

Perangkat lunak kini sudah menjadi kekuatan yang menentukan. Perangkat lunak menjadi mesin yang mengendalikan pengambilan keputusan di dalam dunia bisnis; berfungsi sebagai dasar dari semua bentuk pelayanan serta penelitian keilmuan modern. Hal inilah yang mengubah pandangan masyarakat tentang perangkat lunak. Program-program perangkat lunak sudah tersebar luas, dan masyarakat memandangnya sebagai suatu kenyataan teknologi dalam kehidupan. Akan tetapi banyak individu dan perusahaan masih mengembangkan perangkat lunak secara sembarangan sehingga kualitas perangkat lunak yang dihasilkan menjadi kurang bagus, sehingga diperlukan metode-metode modern dari pendekatan rekayasa perangkat lunak untuk mencapai kematangan secara utuh.

1. PENDAHULUAN

Pendekatan rekayasa perangkat lunak dalam makalah ini bekerja kearah tujuan tunggal, yaitu menghasilkan perangkat lunak berkualitas tinggi. Namun sebelumnya harus dipertanyakan, apakah sebenarnya kualitas perangkat lunak itu?

Philip Cosby dalam bukunya menjelaskan bahwa kualitas sangat mirip dengan seks. Setiap orang memerlukannya (pada saat tertentu tentunya). Setiap orang merasa memahaminya (meskipun mereka tidak ingin menjelaskannya). Setiap orang berpikir bahwa eksekusi hanya mengikuti inklinasi alami. Dan tentu saja, sebagian besar orang merasa bahwa masalah-masalah dalam area ini disebabkan oleh orang lain.

Banyak pengembang perangkat lunak percaya bahwa kualitas perangkat lunak merupakan sesuatu yang mulai Anda khawatirkan setelah kode-kode dilahirkan. Jaminan kualitas merupakan aktivitas mendasar bagi banyak bisnis yang menghasilkan produk yang akan digunakan oleh orang lain, oleh karena itu dibutuhkan pola tindakan yang sistematik dan terencana yang dibutuhkan untuk menjamin kualitas perangkat lunak. Sekarang, implikasi itu adalah banyak pilihan yang berbeda dalam suatu organisasi yang memiliki jaminan kualitas perangkat dan teknisi perangkat lunak, manajer proyek, pelanggan, penjual, dan individu yang bekerja dalam sebuah kelompok *SQA* (*Software Quality Assurance*).

Adapun kegiatan penjaminan kualitas yang akan dijelaskan pada makalah ini adalah Konsep Kualitas dan Pergerakan Kualitas.

ISSN Print : 2085-1588
ISSN Online : 2355-4614
http://ejournal.unsri.ac.id/index.php/jsi/index

2. KONSEP KUALITAS

Bagaimana organisasi pengembang perangkat lunak melakukan kontrol variasi dari satu proyek ke proyek lain, kita perlu meminimalkan perbedaan diantara sumber-sumber daya, sehingga diharapkan mampu menyelesaikan sebuah proyek dan sumber daya aktual yang digunakan, termasuk penataan staf, perlengkapan, dan waktu penanggalan. Secara umum kita perlu memastikan bahwa program pengujian kita sudah mencakup persentase perangkat lunak yang diketahui dari satu peluncuran ke peluncuran yang lain.

Bila kita amati sebuah item dengan didasarkan pada sifat pengukurannya, ada dua jenis kualitas yang ada, yaitu kualitas desain dan kualitas konformansi.

Kualitas desain mengacu pada karakteristik yang ditentukan oleh desainer terhadap suatu item tertentu. nilai material, toleransi, dan spesifikasi kinerja, semuanya memberikan kontribusi terhadap kualitas desain.

Kualitas konformansi adalah tingkat dimana spesifikasi desain terus diikuti selama pembuatan.

Dalam pengembangan perangkat lunak, kualitas desain mencakup syarat, spesifikasi, dan desain sistem. Kualitas konformansi adalah suatu masalah yang difokuskan pada implementasi. Bila implementasi mengikuti desain dan sistem yang dihasilkan memenuhi syarat serta tujuan kinerja, maka kualitas konformansi menjadi tinggi.

a. Kontrol Kualitas

Kontrol kualitas merupakan serangkaian pemeriksaan, kajian, dan pengujian yang digunakan pada keseluruhan siklus pengembangan untuk memastikan bahwa setiap produk memenuhi persyaratan yang ditetapkan. Kegiatan kontrol kualitas meliputi umpan balik pada proses yang menciptakan produk kerja. Kombinasi pengukuran dan umpan balik memungkinkan kita memperbaiki proses bila produk kerja yang diciptakan gagal memenuhi spesifikasi mereka.

1. Jaminan Kualitas
   Jaminan kualitas terdiri atas fungsi *auditing* dan pelaporan manajemen. Tujuan jaminan kualitas adalah memberikan data yang diperlukan oleh manajemen untuk menginformasikan masalah kualitas produk, sehingga dapat memberikan kepastian dan konfidensi bahwa kualitas produk dapat memenuhi sasaran.
2. Biaya Kualitas
   Biaya kualitas mencakup semua biaya yang diadakan untuk mengejar kualitas atau untuk menampilkan kualitas yang berhubungan dengan aktivitas. Biaya kualitas dapat dibagi kedalam biaya-biaya yang dihubungkan dengan pencegahan, penilaian, dan kegagalan. Pada biaya pencegahan meliputi: perencanaan kualitas, kajian teknis formal, perlengkapan pengujian, dan pelatihan. Biaya penilaian adalah inspeksi didalam proses, pemeliharaan dan kalibrasi peralatan, serta pengujian. Biaya kegagalan adalah biaya yang akan hilang bila tidak ada cacat yang muncul sebelum produk disampaikan kepada pelanggan. Biaya kegagalan dapat dibagi lagi kedalam biaya kegagalan internal dan eksternal. Biaya kegagalan internal adalah biaya yang diadakan bila kita mendeteksi suatu kesalahan dalam produk sebelum produk dipasarkan. Biaya

ISSN Print : 2085-1588
ISSN Online : 2355-4614
http://ejournal.unsri.ac.id/index.php/jsi/index

kegagalan internal meliputi: pengerjaan kembali, perbaikan, dan analisis mode kegagalan. Biaya kegagalan eksternal adalah biaya yang berhubungan dengan cacat yang ditemukan setelah produk disampaikan kepada pelanggan. Contoh-contoh biaya kegagalan eksternal meliputi: resolusi keluhan, penggantian dan pengembalian produk, dukungan *help line* dan kerja jaminan.

b. Pergerakan Kualitas

Pada masa sekarang ini, para manajer senior diseluruh dunia industri mengetahui bahwa kualitas produk yang tinggi diterjemahkan kedalam penghematan biaya dan peningkatan garis dasar, biasanya hal ini disebut dengan istilah *Total Quality Management* (*TQM*). Ada empat langkah kemajuan dasar yang biasanya ditemui dan menjadi pondasi dari banyak program *TQM* yang baik.

Langkah pertama disebut *kaizen* dan mengacu kepada sistem peningkatan proses yang kontinyu. Tujuan *kaizen* adalah mengembangkan sebuah proses (dalam hal ini, proses perangkat lunak) yang terlihat, dapat diulang, dan dapat diukur.

Langkah kedua yang dilakukan hanya setelah *kaizen* dicapai, disebut *atarimae hinshitsu*. Langkah ini mengamati hal yang tidak terlihat dan mempengaruhi proses, serta bekerja untuk mengoptimasi pengaruhnya terhadap proses tersebut.

Sementara dua langkah pertama berfokus pada proses, langkah selanjutnya yang disebut *kansei* (diterjemahkan sebagai lima indera), berkonsentrasi pada pemakai produk dalam hal ini perangkat lunak. Pada dasarnya, dengan mengamati cara pemakai mengaplikasikan produk, *kansei* membawa kepada peningkatan dalam produk itu sendiri, dan secara potensial kepada proses yang diciptakannya.

Akhirnya langkah yang disebut *miyokuteki hinshitsu*, meluaskan perhatian manajemen pada produk langsung. Langkah ini merupakan langkah yang berorientasi pada bisnis yang mencari peluang dalam area yang berkaitan, yang dapat diidentifikasi dengan mengamati penggunaan produk di pasar. Dalam dunia perangkat lunak, *miyokuteki hinshitsu* dapat dilihat sebagai godaan untuk menemukan produk baru dan menguntungkan, atau aplikasi yang merupakan perkembangan dari sebuah sistem berbasis komputer yang sudah ada.

Bagi sebagian besar perusahaan, *kaizen* menjadi perhatian utama, sampai suatu proses perangkat lunak yang matang dicapai.

3. PERKEMBANGAN PERANGKAT LUNAK DI INDONESIA

Perkembangan perangkat lunak di Indonesia masih sangat tertinggal dengan Negara-negara berkembang lainnya. Hal ini terbukti dengan baru disyahkannya Undang-undang Hak Kekayaan Intelektual tahun 2002, namun yang lebih mencengangkan lagi adalah berdasarkan data survei yang dipaparkan pada sosialisasi pemanfaatan *software* legal dilingkungan departemen pertanian pada bulan mei kemarin menyebutkan, bahwa Indonesia pada tahun 2003 setahun setelah disyahkannya Undang-undang Hak Cipta, ternyata 88% *software* yang beredar adalah bajakan. Hal tersebut berarti bahwa Indonesia merupakan negara pengguna *software* ilegal tertinggi ke-3 di Asia, setelah Vietnam dan

ISSN Print : 2085-1588
ISSN Online : 2355-4614
http://ejournal.unsri.ac.id/index.php/jsi/index

China, ditingkat dunia Indonesia termasuk negara terbesar pengguna *software* ilegal ke-4 setelah China, Vietnam, dan Ukraina.

Tahun berikutnya, yaitu pada tahun 2004, walaupun telah dilakukan sosialisasi Undang-undang No. 19 Tahun 2002 tersebut, ternyata jumlah *software* bajakan di Indonesia hanya turun 1% saja, menjadi 87% dan masih tetap diposisi ke-3 di Asia dan posisi ke-4 di dunia. Karena itulah pemeriksaan terhadap perangkat lunak bajakan marak dilakukan akhir-akhir ini. Pembersihan *software* bajakan yang biasanya hanya dilakukan oleh polisi di pusat penjualan materi bajakan tersebut (tingkat retail), bahkan sekarang sudah masuk ke perusahaan-perusahaan yang menggunakan *software* bajakan untuk keperluan komersil, termasuk warnet-warnet yang sebagian konsumennya adalah pelajar dan mahasiswa.

Bagi perusahaan yang terkena razia dan terbukti menggunakan perangkat lunak bajakan dapat dikenakan pasal 27 ayat 33 UU No. 19 Tahun 2002 tentang hak cipta, dengan ancaman pidana penjara paling lama 5 tahun atau denda 500 juta. Gencarnya razia menunjukkan keseriusan pemerintah. dalam melindungi hak kekayaan intelektual *software*, disamping upaya menciptakan lingkungan bisnis yang lebih baik di Indonesia.

a. Mengapa Terjadi Pembajakan ?

Pembajakan perangkat lunak terjadi karena masyarakat tidak menganggap perangkat lunak sebagai suatu kekayaan, padahal tanpa adanya perangkat lunak tidak ada yang dapat dikerjakan dengan komputer, karena itu diperlukan sosialisasi untuk mengubah *mind-set*, sehingga *software* dipandang sebagai kekayaan yang berharga. Oleh karenanya diharapkan pemerintah menjadi pelopor dalam proses ini. Tujuan pengakuan tentang Hak Kekayaan Intelektual ini adalah perangkat lunak yang merupakan karya cipta tersebut dihargai sebagai kekayaan milik penciptanya, dan karena itu siapapun tidak boleh menggunakannya tanpa izin atau membajak.

Tindakan yang dapat dikategorikan sebagai pembajakan adalah :

a. Memperbanyak dengan mengkopi :
   - Instalasi perangkat lunak yang dilakukan perusahaan melebihi jumlah lisensi yang dimiliki.
   - Meminjamkan perangkat lunak untuk di*install* di komputer lain diantaranya teman atau keluarga.

b. *Counterfeiting* atau Pemalsuan
   Menduplikasi dan mendistribusikan perangkat lunak imitasi dalam jumlah tertentu.

c. Harddisk Loading
   Biasanya dilakukan oleh perakit komputer yang melakukan instalasi menggunakan perangkat lunak bajakan.

d. Mischanelling
   Perangkat lunak dijual atau didistribusikan ke tipe *customer* yang salah, misalnya *charity* atau *academic price* (yang memang dapat dibeli dengan harga yang lebih murah) tetapi kemudian dijual ke customer biasa atau perusahaan komersial.

e. *Fake Licensing* atau Lisensi Palsu
   Fake Licensing atau Lisensi Palsu Hal ini terjadi jika *End User License Agreements* (*EULAs)* dijual secara terpisah sebagai lisensi.

ISSN Print : 2085-1588
ISSN Online : 2355-4614
http://ejournal.unsri.ac.id/index.php/jsi/index

f. *Internet Piracy* atau Penjualan Barang Palsu lewat Internet
   Penggunaan internet untuk pendistribusian perangkat lunak ilegal, dengan harga lebih rendah dari harga sesungguhnya.

b. Kualitas Perangkat Lunak

Secara prinsip sebuah *software* dikatakan baik apabila dapat secara utuh dan "sempurna" memenuhi kriteria spesifik dari organisasi atau perusahaan yang membutuhkannya. Hal ini sering diistilahkan sebagai pemenuhan terhadap "*user requirements*" (kebutuhan pengguna *software* yang telah terlebih dahulu didefinisikan secara jelas dan detail).

Disamping itu, terlepas dari apakah *software* tersebut dibeli jadi (*off-the-shelf shelf software)* atau dikembangkan secara khusus (*tailor-made software*), sebuah *software* yang baik haruslah pula berkualitas. Dalam salah satu referensi disebutkan bahwa sebuah *software* dikatakan berkualitas apabila memenuhi tiga ketentuan pokok:

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

McCall dan kawan-kawan pada tahun 1977 telah mengusulkan suatu penggolongan faktor-faktor atau kriteria yang mempengaruhi kualitas *software*. Pada dasarnya, McCall menitikberatkan faktor-faktor tersebut menjadi tiga aspek penting, yaitu yang berhubungan dengan:

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

c. Product Operations

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

a. Correctness
   Sejauh mana suatu *software* memenuhi spesifikasi dan *mission objective* dari *users*.
b. Reliability
   Fungsinya dengan ketelitian yang diperlukan.
c. Efficiency

ISSN Print : 2085-1588
ISSN Online : 2355-4614
http://ejournal.unsri.ac.id/index.php/jsi/index

Banyaknya sumber daya komputasi dan kode program yang dibutuhkan suatu *software* untuk melakukan fungsinya.

d. *Integrity*
   Sejauh mana akses ke *software* dan data oleh pihak yang tidak berhak dapat dikendalikan.
e. *Usability*
   Usaha yang diperlukan untuk mempelajari, mengoperasikan, menyiapkan *input*, dan mengartikan *output* dari *software*.

d. Product Revision

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

Faktor-faktor McCall yang berkaitan dengan kemampuan software untuk menjalani perubahan adalah:

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

e. Product Transition

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

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

a. *Portability*
   Usaha yang diperlukan untuk mentransfer *software* dari suatu *hardware* dan atau sistem *software* tertentu agar dapat berfungsi pada *hardware* dan atau sistem *software* lainnya.
b. *Reusability*
   Sejauh mana suatu *software* (atau bagian *software*) dapat digunakan ulang pada aplikasi lainnya.
c. *Interoperability*
   Usaha yang diperlukan untuk menghubungkan satu software dengan lainnya.

ISSN Print : 2085-1588
ISSN Online : 2355-4614
http://ejournal.unsri.ac.id/index.php/jsi/index

Usaha yang diperlukan untuk menghubungkan satu *software* dengan lainnya. Dalam pengembangannya lebih lanjut, ketiga aspek tersebut kerap dihubungkan dengan sejumlah *metric* yang sering digunakan sebagai alat ukur dalam membandingkan kualitas *software* satu dengan lainnya. Adapun *metric* yang dimaksud dalam skema pengukuran di atas adalah sebagai berikut:

a. *Auditability*
   Kemudahan untuk memeriksa apakah *software* memenuhi standar atau tidak.
b. *Accuracy*
   Ketelitian dari komputasi dan kontrol.
c. *Communication Commonality*
   Sejauh mana *interface*, protokol, dan *bandwidth* digunakan.
d. *Completeness*
   Sejauh mana implementasi penuh dari fungsi-fungsi yang diperlukan telah tercapai.
e. *Conciseness*
   Keringkasan *program* dalam ukuran *LOC* (*line of commands*).
f. *Consistency*
   Derajat penggunaan teknik-teknik desain dan dokumentasi yang seragam pada seluruh proyek pengembangan *software*.
g. *Data Commonality*
   Derajat penggunaan tipe dan struktur data baku pada seluruh program.
h. *Error Tolerance*
   Kerusakan yang terjadi apabila program mengalami *error*.
i. *Execution Efficiency*
   Kinerja *run-time* dari program.
j. *Expandability*
   Sejauh mana desain prosedur, *data*, atau arsitektur dapat diperluas.
k. *Generality*
   Luasnya kemungkinan aplikasi dari komponen-komponen program.
l. *Hardware Independence*
   Sejauh mana *software* tidak bergantung pada kekhususan dari *hardware* tempat *software* itu beroperasi.
m. *Instrumentation*
   Sejauh mana program memonitor operasi dirinya sendiri dan mengidentifikasi *error* yang terjadi.
n. *Modularity*
   Functional independence dari komponen-komponen program.
o. *Operability*
   Kemudahan mengoperasikan program.
p. *Security*
   Ketersediaan mekanisme untuk mengontrol dan melindungi program dan *data* terhadap akses dari pihak yang tidak berhak.
q. *Self-Dokumentation*
   Sejauh mana *source-code* memberikan dokumentasi yang berarti.
r. *Simplicity*

ISSN Print : 2085-1588
ISSN Online : 2355-4614
http://ejournal.unsri.ac.id/index.php/jsi/index

Kemudahan suatu program untuk dimengerti.

s. *Traceability*
Kemudahan merujuk balik implementasi atau komponen program ke kebutuhan pengguna *software*.

t. *Training*
Sejauh mana *software* membantu pemakaian baru untuk menggunakan sistem.

Melihat penjelasan tersebut, maka terlihat bahwa faktor harga dan tipe *vendor* tidak secara langsung berpengaruh terhadap baik atau berkualitas tidaknya sebuah produk *software*. Namun, adalah merupakan suatu kenyataan bahwa kebanyakan vendor ternama telah memiliki pengalaman selama berpuluh-puluh tahun dalam menerapkan metodologi pengembangan *software* yang berpegang teguh pada pencapaian aspek-aspek kualitas standar yang ada.

Jika metodologi yang digunakan perlu menerapkan langkah-langkah yang menyerap cukup banyak sumber daya perusahaan, maka dengan sendirinya *software* yang dijual atau dikembangkan pun pada akhirnya menjadi mahal.

Khusus untuk perusahaan yang ingin bekerjasama dengan pihak ketiga atau *vendor* dalam mengembangkan perangkat lunak yang spesifik, maka ada baiknya dicek apakah *vendor* yang bersangkutan telah memiliki sertifikat *CMM* (*Capability Maturity Model*) dan berada pada level berapa perusahaan tersebut. Paling tidak, yang bersangkutan harus memiliki sertifikat dengan minimum level 3 (dari skala 5) sebagai jaminan bahwa *software* yang dihasilkan benar-benar berkualitas.

CMM adalah sebuah standar pengembangan software berkualitas yang diperkenalkan oleh *Software Engineering Institute* (*SEI*) dan diakui kehandalannya di seluruh dunia.

4. CAPABILITY MATURITY MODEL

Kepercayaan pada penggunaan teknik rekayasa perangkat lunak yang baru tidak akan langsung menghasilkan kenaikan profit dan produktivitas, hal ini dikarenakan penyebab dari seluruh permasalahan dalam rekayasa perangkat lunak adalah bagaimana mengatur pada saat proses perangkat pembuatan perangkat lunak tersebut berlangsung.

CMM membantu organisasi dalam penyediaan infrastruktur kedisiplinan dan kematangan dari proses perangkat lunak tersebut. Untuk setiap tingkat *maturity* (kematangan) organisasi, *SEI* telah menyiapkan sekumpulan *Key Process Area* (*KPA*) sebagai acuan untuk naik ketingkat berikutnya. *SEI* (*Software Engineering Institute*) juga telah membuat kuesioner-kuesioner untuk melakukan *assessment* pada tingkat mana suatu organisasi berada.

Pada tingkat *level* pertama *Key Process Area* (*KPA*) kita tidak menemukan hal atau persyaratan spesifik yang direkomendasikan oleh *SEI*, hal tersebut adalah wajar karena keseluruhan organisasi pengembangan perangkat lunak masuk dalam kategori *level* pertama pada *CMM level*. Pada tingkat *level* yang kedua diperlukan berbagai syarat yang harus dipenuhi bagi seluruh pengembang rekayasa perangkat lunak untuk mencapai *level kedua*, adapun *KPA* yang dipersyaratkan adalah :

- *Requirement Management*

ISSN Print : 2085-1588
ISSN Online : 2355-4614
http://ejournal.unsri.ac.id/index.php/jsi/index

- *Software Project Planning*
- *Software Project Tracking and Oversight*
- *Software Subcontract Management*
- *Software Quality Assurance*
- *Software Configuration Management*

Pada tingkat level yang ketiga diperlukan juga syaratsyarat juga untuk dapat sampai pada level ketiga dalam *CMM*, adapun syarat-syarat tersebut adalah :

- *Organization Process Definition*
- *Training Program*
- *Integrated Software Management*
- *Intergroup Coordination*
- *Peer Reviews*

Pada tingkat level yang keempat juga harus dipenuhi berbagai persyaratan, yaitu :

- *Process Measurement and Analysis*
- *Quality Management*
- *Defect Prevention*

Pada tingkat akhir, yaitu pada level kelima *CMM*, suatu organisasi juga diwajibkan memenuhi persyaratan yang telah distandarkan oleh *SEI* pada *CMM*, adapun persyaratan tersebut adalah :

- *Technology Innovation*
- *Process Change Management*

Jika lihat pada *level-level* pada pemodelan *CMM* dapat kita ambil kesimpulan bahwasanya kualitas suatu perangkat lunak adalah suatu keharusan yang harus dipenuhi oleh setiap pengembang perangkat lunak dalam menjalankan setiap proyek yang mereka miliki.

5. PENGEMBANGAN PERAN PERANGKAT LUNAK DIDALAM MASYARAKAT

Saat ini, perangkat lunak memiliki dua peran. Di satu sisi berfungsi sebagai sebuah produk, dan disisi lain sebagai kendaraan yang mengantarkan sebuah produk. Sebagai produk, perangkat lunak mengantarkan potensi penghitungan yang dibangun oleh perangkat lunak komputer. Tidak peduli apakah perangkat lunak ada di dalam sebuah telepon seluler atau beroperasi disebuah mainframe komputer, perangkat lunak merupakan transformer informasi yang memproduksi, mengatur, memperoleh memodifikasi, menampilkan, atau memancarkan informasi, dimana pekerjaan ini dapat menjadi sesederhana suatu bit tunggal atau sekompleks sebuah simulasi multimedia. Sebagai kendaraan yang dipakai untuk mengantarkan produk perangkat lunak berlaku sebagai dasar untuk kontrol komputer (sistem operasi), komunikasi informasi (jaringan), dan penciptaan serta kontrol dari program-program lain (piranti dan lingkungan perangkat lunak).

Perangkat lunak mengantarkan informasi yang dipercaya sebagai produk yang paling penting diabad 21 ini. Perangkat lunak mengubah data personal (contohnya transaksi keuangan individual sehingga *data* dapat menjadi lebih berguna dalam konteks lokal; perangkat lunak mengelola informasi bisnis untuk mempertinggi tingkat kompetensi;

perangkat lunak menjadi pintu gerbang menuju jaringan informasi ke seluruh dunia (misal:*internet*); dan menyediakan berbagai alat untuk memperoleh informasi dalam segala bentuk.

Peran perangkat lunak komputer mengalami perubahan penting selama paruh kedua abad ke-20. Perkembangan dramatis pada unjuk kerja perangkat keras, perubahan-perubahan besar dalam arsitektur komputer, pertambahan yang pesat pada memori dan kapasitas penyimpanan serta variasi pilihan input dan output yang luas, semuanya mempercepat sistem berbasis komputer yang sudah kompleks dan canggih. Kecanggihan dan kompleksitas tersebut dapat memberi hasil yang mengagumkan jika sistem berjalan, tetapi juga dapat menjadi masalah besar bagi mereka yang harus membangun sistem yang kompleks.

Selama masa awal era komputer, perangkat lunak dilihat hanya sebagai suatu perenungan. Pemrograman komputer menjadi sebuah seni "*seat of pants*" dimana terdapat beberapa metode yang sistematis. Perkembangan perangkat lunak sebenarnya tidak bisa diatur, sampai terjadi jadwal yang bergeser atau biaya yang mulai melonjak. Para pemrogram kemudian mulai berusaha untuk membuat semuanya benar kembali, dengan cara yang heroik akhirnya mereka berhasil.

Pada masa awal perangkat keras dengan tujuan umum menjadi hal yang biasa, sementara perangkat lunak dirancang secara khusus untuk aplikasi tertentu saja dan hanya memilik areal industri yang terbatas. Produk perangkat lunak (misal program yang dikembangkan untuk dijual kepada satu pelanggan atau lebih), masih langka. Kebanyakan perangkat lunak dikembangkan dan digunakan oleh orang atau organisasi yang sama. Anda menulisnya kemudian memakainya, dan bila terjadi kegagalan. Anda sendiri yang memperbaikinya. Karena mobilitas kerja yang masih rendah, para manajer yakin bahwa Anda akan berada ditempat pada saat terjadi *bug*. Oleh karena itu lingkungan perangkat lunak yang personal seperti itu, rancangan menjadi sebuah proses implisit yang dilakukan oleh satu kepala, bahkan kadangkadang tidak ada prose pendokumentasian.

Era kedua juga ditandai dengan penggunaan produk perangkat lunak serta kehadiran *software houses*. Perangkat lunak dikembangkan untuk distribusi yang luas pada pasar yang multidisiplin. Selama jumlah sistem berbasis komputer mulai tumbuh, perpustakaan tentang perangkat lunakpun mulai berkembang. Proyek ini dikembangkan dirumah produksi dan menghasilkan puluhan ribu pernyataan sumber program. Tetapi produk perangkat lunak yang dikemas dari luar ditambah lagi dengan ratusan bahkan ribuan pernyataan baru. Semua program tersebut, semua pernyataan sumber harus dikoreksi ketika mengalami kesalahan. Dimodifikasi sesuai dengan keperluan pemakai, atau disebut juga *software maintenance*. Usaha yang dilakukan untuk pemeliharaan perangkat lunak mulai menyerap sumber daya dalam jumlah yang sangat tinggi.

Era ketiga evolusi komputer dimulai pada pertengahan tahun 1970 an dan berlangsung lebih dari satu dekade penuh. Sistem terdistribusi multikomputer, yang masing-masing melakukan fungsi secara konkuren dan berkomunikasi satu dengan yang lain, sangat menambah kompleksitas sistem berbasis komputer. Jaringan area global dan lokal, jaringan komunikasi digital dengan *bandwidth* yang tinggi serta pertambahan permintaan untuk akses data sesaat, sangat mendongkrak permintaan akan pengembang

ISSN Print : 2085-1588
ISSN Online : 2355-4614
http://ejournal.unsri.ac.id/index.php/jsi/index

perangkat lunak. Tetapi dalam sistem dan perangkat lunak yang diperoleh tetap saja berada dalam industri dan akademi. Pemakaian secara personal masih jarang.

Evolusi sistem komputer era keempat menjauhkan kita dari komputer individual dan program komputer dan menuju pengaruh kolektif dari komputer dan perangkat lunak. Mesin desktop yang kuat, yang dikontrol oleh sistem operasi yang canggih, jaringan lokal, dan jaringan global, serta digabung dengan aplikasi perangkat lunak yang maju, menjadi sebuah aturan.

Dalam dunia ekonomi, industri perangkat lunak tidak lagi menjadi buruk. Keputusan-keputusan yang dibuat oleh pengembang-pengembang perangkat lunak memiliki resiko besar. Tetapi selama era evolusi sistem berbasis komputer, serangkaian masalah yang berhubungan dengan perangkat lunak masih muncul, dan intensitas masalah-masalah tersebut terus bertambah.

1. Kemajuan perangkat keras terus berlanjut, melampaui kemampuan kita untuk membangun perangkat lunak yang sesuai dengan perangkat keras yang ada.
2. Kemampuan kita untuk membangun program baru tidak dapat memenuhi kebutuhan akan program baru, juga tidak dapat membangun program yang cukup cepat untuk memenuhi kebutuhan bisnis dan pasar.
3. Pemakaian komputer yang tersebar luas membuat masyarakat semakin tergantung pada operasi perangkat lunak yang reliabel.
4. Membangun perangkat lunak komputer dengan fleksibilitas dan kualitas yang sangat tinggi.
5. Kemampuan kita untuk mendukung program yang ada terhambat oleh buruknya desain serta sumber daya yang tidak memadai.

5.1 Perspektif Industri

Pada masa-masa awal sistem berbasis komputer dikembangkan dengan menggunakan manajemen yang berorientasi pada perangkat keras. Pemrograman masih dilihat sebagai “bentuk kesenian”. Hanya sedikit saja metode yang ada dan lebih sedikit lagi orang yang memahaminya. Para pemrogram bahkan kadang-kadang harus mempelajarinya dengan cara coba-coba. Jargon dan tantangan untuk membangun perangkat lunak komputer menciptakan sebuah mistik yang membuat para manajer menjadi sangat berhati-hati.

Sekarang distribusi biaya untuk pengembangan sistem yang berbasis komputer sudah berubah secara dramatis. Dibanding perangkat keras, perangkat lunak merupakan salah satu bentuk tunggal yang berbiaya paling tinggi. Selama hampir dua dekade ini, para manajer dan para praktisi melontarkan pertanyaan-pertanyaan sebagai berikut :

- Mengapa membutuhkan waktu yang sangat lama untuk menyelesaikan suatu program?
- Mengapa biayanya sangat tinggi ?
- Dapatkah kita menemukan kesalahan yang ada pada program sebelum mendistribusikan kepada pelanggan ?
- Mengapa sangat sulit bagi kita untuk mengukur kemajuan pada saat perangkat lunak dikembangkan?

ISSN Print : 2085-1588
ISSN Online : 2355-4614
http://ejournal.unsri.ac.id/index.php/jsi/index

5.2 Daya Saing Perangkat Lunak

Selama beberapa tahun pengembang perangkat lunak diperusahaan-perusahaan besar dan kecil hanya menonton saja di kota besar. Dan memang kadang-kadang mereka bersikap seperti itu. Ini karena komputer yang ada sudah *custom-build*, meskipun sekarang semua itu sudah berubah.

Sekarang ini perangkat lunak sudah menjadi lahan yang sangat kompetitif. Perangkat lunak yang dulu dibangun secara internal didalam komputer sekarang sudah diproduksi secara terpisah. Perusahaan yang pada awalnya harus membayar sepasukan pemrogram untuk menghasilkan aplikasi tertentu, sekarang dapat mengambil tenaga dari luar dan dilakukan pada *third party*.

Biaya, jangka waktu yang tak terbatas, dan kualitas merupakan pengendali utama yang membuat persaingan usaha perangkat lunak tidak pernah berhenti selama beberapa dekade terakhir.

6. KESIMPULAN

Jaminan kualitas perangkat lunak merupakan "aktivitas pelindung" yang diaplikasikan pada setiap langkah dalam proses perangkat lunak. *Software Quality Assurance* mencakup berbagi prosedur untuk aplikasi yang efektif dari metode dan piranti, kajian teknik formal, strategi dan teknik pengujian, prosedur untuk mengontrol perubahan, prosedur untuk menjamin kesesuaian dengan standar yang ada, serta pengukuran dan mekanisme pelaporan.

SQA kompleks karena kualitas perangkat lunak pun bersifat kompleks, atribut program komputer yang didefinisikan sebagai "penyesuaian dengan syarat yang ditetapkan secara eksplisit dan implisit". Tetapi bila diperhatikan secara lebih umum, kualitas perangkat lunak mencakup banyak faktor produk dan proses yang berbeda serta metrik yang berhubungan.

Kajian perangkat lunak merupakan salah satu aktivitas *SQA* yang terpenting. Kajian berfungsi sebagai filter bagi proses perangkat lunak, penghapus kesalahan bila biaya yang diperlukan untuk menemukan dan mengkoreksi kesalahan relatif tidak mahal. Kajian teknik formal atau *walkthrough* adalah pertemuan kajian yang disesuaikan dengan kebutuhan yang terbukti sangat efektif untuk menemukan keasalahan.

Perangkat lunak dirancang dari program-program, data, dan dokumen. Masing-masing dari item tersebut berdiri terdiri dari sebuah konfigurasi yang diciptakan sebagai bagian dari proses pengembangan perangkat lunak. Tujuan rekayasa perangkat lunak adalah menyediakan sebuah kerangka kerja guna membangun perangkat lunak dengan kualitas yang lebih tinggi.

7. DAFTAR PUSTAKA

Adler, Paul S. *Practice and Process: The Socialization Software Development*. 2006

Indrajit, Richardus Eko. *Kiat Memilih Software*. EbizzAsia, 2004

Pertanian, Departemen. "Sosialisasi Pemanfaatan Software Ilegal", www.deptan.go.id, 2005

ISSN Print : 2085-1588
ISSN Online : 2355-4614
http://ejournal.unsri.ac.id/index.php/jsi/index

Pressman, Roger S. *Rekayasa Perangkat Lunak (Pendekatan Praktisi)*. Yogyakarta: Penerbit Andi, 2002