

**TUGAS I
MANAGEMENT PROYEK
SOFTWARE ENGINEERING**



Disusun Oleh :

KELOMPOK 5

Angga Cahya S.N	I1A004015
Bayu Jati Kusuma	I1A004063
Elektrika Dwi W	I1A005037
Dian Eko Prabowo	I1A005057
Riawan	I1A005063
Oko Setia P	I1A006006
Yunan Ahmad B	I1A006011
Firman Hadi S	I1A006052

**DEPARTEMEN PENDIDIKAN NASIONAL
UNIVERSITAS JENDERAL SOEDIRMAN
FAKULTAS SAINS DAN TEKNIK
JURUSAN TEKNIK ELEKTRO
PURWOKERTO
2009**

PENGERTIAN SOFTWARE ENGINEERING

Mengutip dari Ian Sommerville dari bukunya Software Engineering, di sebutkan Software Engineering adalah disiplin ilmu yang membahas semua aspek produksi perangkat lunak, mulai dari tahap awal spesifikasi sistem sampai pemeliharaan sistem setelah digunakan, dimana ada dua kata kunci yaitu:

1. 'Disiplin Rekayasa' dimana perekayasa membuat suatu alat bekerja dengan menerapkan teori, metode dan alat bantu yang sesuai serta penggunaan secara selektif dan selalu mencoba mencari solusi terhadap permasalahan walaupun tidak ada teori atau metode yang mendukung.
2. 'Semua aspek produksi perangkat lunak', rekayasa perangkat lunak tidak hanya berhubungan dengan proses teknis dari pengembangan perangkat lunak tetapi juga dengan kegiatan seperti manajemen proyek perangkat lunak dan pengembangan alat bantu, metode, dan teori untuk mendukung produksi perangkat lunak.

Menurut Fritz Badar, software engineering adalah disiplin ilmu yang menerapkan prinsip-prinsip engineering agar mendapatkan software yang ekonomis yang dapat dipercaya dan bekerja lebih efisien pada mesin yang sebenarnya.

Apakah itu software (perangkat lunak)? menurut IEEE definisi software merupakan program komputer, prosedur, data dan semua dokumentasi yang berhubungan operasi pada sistem komputer. jadi bisa disimpulkan bahwa software merupakan kumpulan dari object membentuk konfigurasi yang didalamnya termasuk program, dokumen, dan data.

Tantangan dalam pembuatan software antara lain adalah menciptakan software yang memperbolehkan berbagai macam mesin untuk saling berkomunikasi melewati jaringan intranet dan internet, menciptakan arsitektur aplikasi yang sederhana namun canggih sesuai dengan kebutuhan pasar di era globalisasi, mendistribusikan source code sehingga pelanggan bisa membuat modifikasi lokal sesuai dengan kebutuhannya, serta menciptakan aplikasi yang memfasilitasi komunikasi dan distribusi produk secara massal menggunakan konsep yang berkembang.

Sering kali terdapat program yang telah berumur tua namun masih bernilai dalam bisnis, program ini harus tetap dijaga namun sering kali menimbulkan masalah karena program tersebut tidak didesain untuk berubah. padahal sebuah program harus bisa beradaptasi dengan lingkungan dan teknologi baru, selalu bisa mengimplementasikan kebutuhan bisnis yang baru, bisa berkomunikasi dengan sistem yang modern, dan lain-lain. untuk itu perlu software evolution

yaitu suatu proses dimana program melakukan perubahan bentuk, beradaptasi dengan pasar, dan menurunkan karakteristik yang ada pada program pendahulunya. Namun perubahan sendiri juga mempunyai biaya yang mahal, biaya perubahan pada tahap development membutuhkan biaya 6 kali lebih mahal dibandingkan jika perubahan dilakukan pada tahap definisi, dan melipat 60 sampai 100 kali lipat jika perubahan dilakukan setelah software dirilis.

Ada beberapa salah persepsi tentang software, dari sisi pelanggan berpandangan bahwa pernyataan umum tentang tujuan sudahlah cukup, dan melengkapi detilnya kemudian, dalam kenyataan pendefinisian kebutuhan yang salah dan kurang pada tahap awal pembuatan software merupakan sebab utama kualitas yang buruk dan penyelesaian software menjadi terlambat. pandangan lainnya yang salah adalah pelanggan mengangap bahwa perubahan bisa diakomodasi dengan mudah karena software itu fleksible, padahal biaya yang dibutuhkan untuk perubahan semakin besar jika mendekati tahap akhir sebuah software.

Dari sisi pengembang ada beberapa myth yang perlu diluruskan, diantaranya adalah anggapan bahwa pekerjaan developer telah selesai setelah program ditulis dan berjalan, padahal 50 sampai 70 % dari kerja developer terjadi ketika program di deliver ke pelanggan. selain itu pandangan tentang satu-satunya deliver adalah program perlu diluruskan, bahwa software didalamnya termasuk dokumentasi, data dan program.

Pada sisi manajemen berpendapat bahwa menambah programmer ketika deadline sudah didepan mata bisa membantu mempercepat waktu pengerjaan, namun pendapat itu tidak sepenuhnya benar karena orang baru membutuhkan masa transisi yang bisa jadi merusak ritme kerja tim.

Pendekatan dalam pengembangan, pengoperasian, dan pengelolaan perangkat lunak disebut dengan software engineering (rekayasa perangkat lunak). rekayasa perangkat lunak dimaksudkan untuk membuat perangkat lunak yang tepat dengan metode yang tepat, karena ketepatan perangkat lunak bisa di kelola. ada 2 hal yang perlu di pertimbangkan dalam rekayasa perangkat lunak, produknya yaitu perangkat lunak serta prosesnya. produk terdiri dari program, dokumen, dan data, proses terdiri dari proses manajemen dan proses teknikal.

Produk dari perangkat lunak dipantau melewati beberapa tahap pengembangan (software development life cycle / SDLC). contoh dari SDLC antara lain model waterfall, model V, model spiral, prototyping dan lain-lain. Sedangkan proses manajemen dalam rekayasa perangkat lunak terdiri atas manajemen proyek, Configuration management, Quality Assurance management.

proses teknikal merupakan metode yang diaplikasikan pada tahap tertentu dalam pengembangan software, yang didalamnya termasuk metode analisis, metode desain, metode pemrograman, dan metode testing.

Perbedaan antara rekayasa perangkat lunak dan ilmu komputer adalah jika ilmu komputer berkonsentrasi pada teori dan dasar-dasar komputer, rekayasa perangkat lunak berkonsentrasi pada praktik dalam pengembangan perangkat lunak yang tepat.

DELAPAN PRINSIP PROFESIONAL SOFTWARE ENGINEERING

Dari ACM/IEEE sendiri ternyata juga mengeluarkan suatu kode etik sebagai profesional Software Engineering, antara lain harus mengikuti Delapan Prinsip berikut :

1. **MASYARAKAT** — Perekayasa perangkat lunak akan bertindak secara konsisten sesuai dengan kepentingan masyarakat.
2. **KLIEN DAN ATASAN** — Perekayasa perangkat lunak akan melakukan yang terbaik bagi klien dan atasan mereka, konsisten dengan kepentingan masyarakat.
3. **PRODUK** — Perekayasa perangkat lunak akan mejamin bahwa produk mereka dan modifikasi yang mereka lakukan terhadapnya memenuhi standar profesional yang setinggi-tingginya.
4. **PENILAIAN** — Perekayasa perangkat lunak akan mempertahankan integritas dan independensi penilaian profesional mereka.
5. **MANAJEMEN** — Manajer dan pemimpin rekayasa perangkat lunak akan mengikuti dan mempromosikan pendekatan etis terhadap manajemen pengembangan dan pemeliharaan perangkat lunak.
6. **PROFESI** — Perekayasa perangkat lunak akan mempertinggi integritas dan reputasi profesinya konsisten dengan kepentingan masyarakat.
7. **KOLEGA** — Perekayasa perangkat lunak akan bersifat adil dan mendukung terhadap koleganya.
8. **DIRI SENDIRI** — Perekayasa perangkat lunak akan berpartisipasi dalam pembelajaran seumur hidup mengenai praktek profesi mereka dan akan mempromosikan pendekatan etis terhadap praktek profesi tersebut.

Kode Etik ini dibuat terkait dengan perilaku dan keputusan yang dibuat oleh para Software Engineering Profesional yang mencakup profesi praktisi, pendidik, manajer,

supervisor, pembuat kebijakan dan termasuk trainee dan mahasiswa profesi Software Engineering.

KOMPONEN SOFTWARE ENGINEERING

Komponen adalah pokok bahasan yang sangat menarik jika kita membicarakan masalah reuse. Membangun *software* yang bersifat *reusable* akan sangat sulit dikarenakan desain komponen yang tidak baik. Hal ini dapat mengakibatkan komponen yang satu dengan yang lain sulit untuk dipisahkan. Di dalam membangun dan memelihara komponen dibutuhkan kejelasan pengertian dari dan antar komponen. Suatu komponen dapat saja memiliki hubungan *external* atau ketergantungan antar komponen (komponen *dependence*). Untuk itu diperlukan sebuah *tools* yang dapat melihat komponen *dependence*.

AKTIVITAS DALAM MENGEMBANGKAN SOFTWARE

Krisis software tidak dapat hilang dalam satu malam, di mana tidak ada suatu pendekatan yang baik dalam mengatasi krisis software, namun gabungan dari metode untuk semua fase dalam pengembangan software seperti peralatan yang lebih baik untuk mengotomatisasi metode-metode ini, teknik yang lebih baik untuk mengontrol kualitas, dan filosofi untuk koordinasi kontrol, serta manajemen dipelajari dalam suatu disiplin ilmu yang kita sebut *software engineering*.

TIGA ELEMEN SOFTWARE ENGINEERING

Software engineering terdiri dari 3 elemen kunci, yaitu :

1. Metode

Metode software engineering memberikan teknik-teknik bagaimana membentuk software. Metode ini terdiri dari serangkaian tugas:

- Perencanaan & estimasi proyek
- Analisis kebutuhan sistem dan software
- Desain struktur data
- Arsitektur program dan prosedur algoritma
- Coding

- Testing dan pemeliharaan

2. Peralatan (tools)

Peralatan software engineering memberikan dukungan atau semiautomasi untuk metode.

Contohnya :

- CASE (Case Aided Software Engineering), yaitu suatu software yang menggabungkan software, hardware, dan database software engineering untuk menghasilkan suatu lingkungan software engineering.
- Database Software Engineering, adalah sebuah struktur data yang berisi informasi penting tentang analisis, desain, kode dan testing.
- Analogi dengan CASE pada hardware adalah : CAD, CAM, CAE.

3. Prosedur

Terdiri dari :

- urutan-urutan di mana metode tersebut diterapkan
- dokumen
- laporan-laporan
- formulir-formulir yang diperlukan
- mengontrol kualitas software
- mengkoordinasi perubahan yang terjadi pada software

Ketiga elemen tersebut memungkinkan manajer mengontrol proses pengembangan software dan memberikan praktisi dasar yang baik untuk pembentukan software berkualitas tinggi.

EMPAT METODE YANG DIGUNAKAN DALAM PENGUASAAN ATAS MODEL SOFTWARE ENGINEERING ATAU SOFTWARE ENGINEERING PARADIGM

Dalam penguasaan atas model software engineering atau software engineering paradigm, dikenal ada 4 metode yang luas dipergunakan, yaitu :

1. Classic Life Cycle Pradigm – Model Water Fall – Model Siklus Hidup Klasik

Keterangan :

1. System Engineering and Analysis

Karena software merupakan bagian terbesar dari sistem, maka pekerjaan dimulai dengan cara menerapkan kebutuhan semua elemen sistem dan mengalokasikan sebagian kebutuhan tersebut ke software. Pandangan terhadap sistem adalah penting, terutama pada saat software harus berhubungan dengan elemen lain, seperti :

- Hardware
- Software
- Database

2. Analisis kebutuhan software

Suatu proses pengumpulan kebutuhan software untuk mengerti sifat-sifat program yang dibentuk software engineering, atau analis harus mengerti fungsi software yang diinginkan, performance dan interface terhadap elemen lainnya. Hasil dari analisis ini didokumentasikan dan direview / dibahas / ditinjau bersama-sama customer.

3. Design

Desain software sesungguhnya adalah *proses multi step* (proses yang terdiri dari banyak langkah) yang memfokuskan pada 3 atribut program yang berbeda, yaitu :

- Struktur data
- Arsitektur software
- Rincian prosedur

Proses desain menterjemahkan kebutuhan ke dalam representasi software yang dapat diukur kualitasnya sebelum mulai coding. Hasil dari desain ini didokumentasikan dan menjadi bagian dari konfigurasi software.

4. Coding

Desain harus diterjemahkan ke dalam bentuk yang dapat dibaca oleh mesin

5. Testing

Segera sesudah objek program dihasilkan, pengetesan program dimulai. Proses testing difokuskan pada logika internal software. Jaminan bahwa semua pernyataan

atau statements sudah dites dan lingkungan external menjamin bahwa definisi input akan menghasilkan output yang diinginkan.

6. Maintenance

Software yang sudah dikirim ke customer data berubah karena

- Software mengalami error
- Software harus diadaptasi untuk menyesuaikan dengan lingkungan external, misalnya adanya sistem operasi baru atau peripheral baru.
- Software yang lebih disempurnakan karena adanya permintaan dari customer.
- Masalah yang dihadapi dari model siklus hidup klasik adalah :
- Proyek yang sebenarnya jarang mengikuti aliran sequential yang ditawarkan model ini. Iterasi (Pengulangan) selalu terjadi dan menimbulkan masalah pada aplikasi yang dibentuk oleh model ini.
- Seringkali pada awalnya customer sulit menentukan semua kebutuhan secara eksplisit (jelas).
- Customer harus sabar karena versi program yang jalan tidak akan tersedia sampai proyek software selesai dalam waktu yang lama.

2. Working Prototype

Adalah prototype yang mengimplementasikan beberapa bagian dari fungsi software yang diinginkan seperti pada pendekatan pengembangan software. Model ini dimulai dengan :

- Pengumpulan kebutuhan developer dan customer
- Menentukan semua tujuan software
- Mengidentifikasi kebutuhan-kebutuhan yang diketahui

Hasil dari pengumpulan kebutuhan diteruskan pada Quick Design. Quick Design ini memfokuskan pada representasi aspek-aspek software yang dapat dilihat oleh user, misalnya format input dan output, selanjutnya dari desain cepat diteruskan pada pembentukan prototype (langkah ke 3). Prototype ini dievaluasi oleh customer / user dan digunakan untuk memperbaiki kebutuhan-kebutuhan software. Proses iterasi terjadi agar

prototype yang dihasilkan memenuhi kebutuhan customer, juga pada saat yang sama developer mengerti lebih baik tentang apa yang harus dikerjakan.

Masalah yang dihadapi oleh prototyping paradigm ini adalah :

- Customer hanya melihat pada apa yang dihasilkan oleh software, tidak peduli pada hal-hal yang berhubungan dengan kualitas software dan pemeliharaan jangka panjang.
- Developer seringkali menyetujui apa yang diterangkan oleh customer agar prototype dapat dihasilkan dengan cepat. Akibatnya timbul pemilihan sistem operasi / bahasa pemrograman yang tidak tepat.

3. Fourth Generation Tehnique Paradigm – Model tehnik generasi ke 4 / 4GT

Istilah Fourth Generation Technique (4GT) meliputi seperangkat peralatan software yang memungkinkan seorang developer software menerapkan beberapa karakteristik software pada tingkat yang tinggi, yang kemudian menghasilkan *source code* dan *object code* secara otomatis sesuai dengan spesifikasi yang ditentukan developer. Saat ini peralatan / tools 4GT adalah bahasa non prosedur untuk :

- DataBase Query
- Pembentukan laporan (Report Generation)
- Manipulasi data
- Definisi dan interaksi layar (screen)
- Pembentukan object dan source (Object and source generation)
- Kemampuan grafik yang tinggi, dan
- Kemampuan spreadsheet

Masalah yang dihadapi dalam model 4GT adalah adanya sebagian orang yang beranggapan bahwa :

- peralatan 4GT tidak semudah penggunaan bahasa pemrograman,
- source code yang dihasilkan oleh peralatan ini tidak efisien,
- pemeliharaan sistem software besar yang dikembangkan dengan 4GT masih merupakan tanda tanya.

4. Model Kombinasi – Combining Paradigm

Keterangan :

Model ini menggabungkan keuntungan-keuntungan dari beberapa model sebelumnya. Seperti pada model sebelumnya, model kombinasi ini dimulai dengan langkah pengumpulan kebutuhan.

Pendekatan yang dapat diambil adalah pendekatan siklus hidup klasik (Analisis sistem dan analisis kebutuhan software) atau dapat juga menggunakan pendekatan seperti prototyping jika definisi masalahnya tidak terlalu formal.

Jika kebutuhan untuk fungsi dan performance software diketahui dan dimengerti, pendekatan yang dianjurkan adalah model siklus hidup klasik. Sebaliknya, jika aplikasi software menuntut interaksi yang sering antara manusia dan mesin, membutuhkan algoritma yang tidak dapat dibuktikan, atau membutuhkan tehnik output / kontrol, maka pendekatan yang dianjurkan adalah model prototyping.

Pada kasus seperti ini, 4GL dapat digunakan untuk mendapat prototype dengan cepat. Segera sesudah prototype dievaluasi dan disempurnakan, langkah desain dan implementasi dalam siklus hidup klasik diterapkan.

Dari model yang disebut di atas dapat diambil suatu kesimpulan, bahwa proses pengembangan software terdiri dari 3 fase, yaitu :

1. Fase Definisi

Fase definisi memfokuskan pada “*What*”. Selama definisi ini, developer software berusaha untuk :

- Mengidentifikasi informasi apa yang dikerjakan proses
- Fungsi dan performance apa yang diinginkan
- Interface apa yang dibutuhkan
- Hambatan desain apa yang ada, dan
- Kriteria validasi apa yang dibutuhkan untuk menetapkan keberhasilan sistem.

1. Sistem Analis

Sistem analis menetapkan peranan dari setiap elemen dalam sistem berbasis komputer, terutama mengalokasikan peranan software.

2. Sistem Software Planning

Dalam sistem ini, setelah lingkungan software dialokasikan, maka langkah dari sistem software planning ini adalah :

- Pengalokasian sumber / resource
- Estimasi biaya
- Penetapan tugas pekerjaan dan jadwal.

3. Requirement Analysis

Penetapan lingkup untuk software memberikan petunjuk / arah. Namun definisi yang lebih rinci dari informasi dan fungsi software diperlukan sebelum pekerjaan dimulai.

2. Fase Pengembangan (Development)

Fase pengembangan berfokus pada “*How*”. Selama pengembangan, developer software berusaha menjelaskan :

- Bagaimana struktur data dan arsitektur software yang didesain
- Bagaimana rincian prosedur diimplementasikan (diterapkan)
- Bagaimana desain diterjemahkan ke dalam bahasa pemrograman atau bahasa non prosedur, dan
- Bagaimana pengetesan akan dilaksanakan.

1. Desain software (Software Design)

Desain menterjemahkan kebutuhan-kebutuhan software ke dalam sekumpulan representasi (grafik, tabel, diagram, atau bahasa yang menjelaskan struktur data, arsitektur software dan prosedur algoritma).

2. Coding

Representasi desain harus diterjemahkan ke dalam bahasa tiruan / artificial language yang menghasilkan perintah-perintah yang dapat dieksekusi oleh komputer.

3. Software Testing

Segera sesudah software diimplementasikan dalam bentuk yang dapat dieksekusi oleh mesin, software perlu ditest untuk menemukan kesalahan (merupakan fungsi logika dan implementasi).

3. Fase Pemeliharaan (Maintenance)

Fase pemeliharaan berfokus pada “*Change*” atau perubahan. Ini dapat disebabkan :

1. Perubahan karena software error (Corective Maintenance)
2. Perubahan karena software disesuaikan / diadaptasi dengan lingkungan external, misalnya munculnya CPU baru, sistem operasi baru (Adaptive Maintenance)
3. Perubahan software yang disebabkan customer / user meminta fungsi tambahan, misalnya fungsi grafik, fungsi matematik, dll (Perfective Maintenance)