BAB I
SOFTWARE ENGINEERING

Arti Software Engineering :

Ilmu yang mempelajari tehnik pembuatan software yang baik dengan pendekatan tehnik (*Engineering approach)*

Dalam membuat softrare yang baik, ada beberapa cara :

1. Fase Perencanaan (Planning) :
   a) Rencana software
   b) Analisa kebutuhan software
   c) Analisa *cost banefit* (Salah satu bagian dari studi kelayakan)
2. Fase Pengembangan (Development) :
   a) Coding
   b) Testing

   Macam-macam test program :
      i) Unit test (Test per modul)
      ii) Integreated test (Test penggabungan dari modul-modul yang telah diuji)
      iii) Validated test (Diuji dengan data sebenarnya)
      iv) System test (Test dilakukan dengan lingkungan sebenarnya)
      v) Topdown test (Test gabungan dari atas ke bawah)
      vi) Bottom up test (Test gabungan dari bawah ke atas)
3. Fase Pemeliharaan (Maintenance) :

Jenis-jenis maintenance

   a) Koreksi (Corection)
   b) Adaptasi (Adaptive)

   Software dikembangkan sesuai dengan tuntutan perkembangan jaman

   c) Adaptasi yang berkembang pada dewasa ini terbagi atas :
      i) Sistem Operasi
         ◊ Pengarahan sistem operasi yang bersifat multi user. Contoh : UNIX
         ◊ Sistem operasi yang bersifat jaringan. Contoh : NOVELL
      ii) RDBMS - Relational DataBase Management System
         ◊ Berkembang dalam bentuk bahasa SQL (Structure Query Language).
      iii) Bahasa

      Mengarah pada perkembangan bahasa generasi ke empat (4GL - Fourth Generation Language)

      Bahasa 4GL adalah suatu bahasa yang dibuat untuk meningkatkan produktifitas programmer dan end user. Contoh :
         a) INFORMIX - Dapat dijalankan pada PC dengan minimum RAM 4MB + 640KB dan disk storage > 40MB
         b) ORACLE
         c) INGRES
         d) AS / SET - Digunakan pada IBM AS 400

e) POWER HOUSE - digunakan pada HR 3000

iv) Perfective

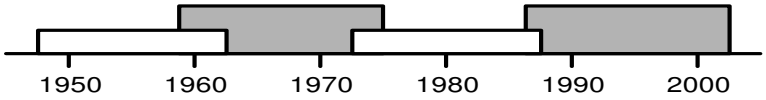
Menyempurnakan software yang ada biasanya dilakukan karena permintaan / saran / kritik user.

SOFTWARE AND SOFTWARE ENGINEERING

Selama tiga dekade pertama dari era komputerisasi, tantangan utama adalah mengembangkan hardware komputer yang dapat mengurangi biaya pengolahan dan penyimpanan data. Selama dekade tahun 1980 an, kemajuan yang pesat dari mikro elektronik menghasilkan kemampuan komputer yang lebih baik pada tingkat biaya yang lebih rendah. Namun masalah sekarang berbeda. Tantangan utama adalah mengurangi biaya dan memperbaiki kualitas solusi berbasis komputer (Solusi yang diimplementasikan dengan mempergunakan software). Software merupakan faktor kunci dalam keberhasilan suatu usaha, software dapat membedakan satu perusahaan dari perusahan saingannya.

EVOLUSI PERKEMBANGAN SOFTWARE

Evolusi software



Tahun-tahun awal :

- ◊ Batch orientation
- ◊ Limmited distribution
- ◊ Custummer software

Era kedua :

- ◊ Multi user
- ◊ Real time
- ◊ Database

Era ketiga

- ◊ Distibuted system
- ◊ Embedded intellegence
- ◊ Low cost hardware
- ◊ Consumer infact

Era keempat :

- ◊ Expert system
- ◊ A I Machine
- ◊ Parallel architecture

TAHUN-TAHUN PERTAMA :

- ◊ Batch Orientation

  Suatu orientasi di mana proses dilakukan setelah data dikumpulkan dalam satuan waktu tertentu, atau proses dilakukan setelah data terkumpul, lawan dari batch adalah ONLINE atau Interactive Process.

  Keuntungan dari Interactive adalah mendapatkan data yang selalu up to date.
- ◊ Limmited distribution

  Suatu penyebaran software yang terbatas pada perusahaan-perusahaan tertentu.
- ◊ Custom software

  Software yang dikembangkan berdaasarkan perusahaan-perusahaan tertentu.

ERA KEDUA :

- ◊ Multi user

  Suatu sistem di mana satu komputer digunakan oleh beberapa user pada saat yang sama.
- ◊ Real Time

  Suatu sistem yang dapat mengumpulkan, menganalisa dan mentransformasikan data dari berbagai sumber, mengontrol proses dan menghasilkan output dalam mili second.

◊ Database

Perkembangan yang pesat dari alat penyimpan data yang OnLine menyebabkan muncul generasi pertama DBMS (DataBase Management System).

◊ Product Software

Adalah software yang dikembangkan untuk dijual kepada masyarakat luas.

ERA KETIGA :

◊ Distributed system

Suatu sistem yang tidak hanya dipusatkan pada komputer induk (Host computer), daerah atau bidang lainnya yang juga memiliki komputer yang ukurannya lebih kecil dari komputer induk. Lawan dari distributed system adalah Centralized System.

◊ Embedded Intelegence

Suatu product yang diberi tambahan "Intellegence" dan biasanya ditambahkan mikroprocessor yang mutakhir. Contohnya adalah automobil, robot, peralatan diagnostic serum darah.

◊ Low Cost Hardware

harga hardware yang semakin rendah, ini dimungkinkan karena munculnya Personal Computer.

◊ Consummer Inpact

Adanya perkembangan komputer yang murah menyebabkan banyaknya software yang dikembangkan, software ini memberi dampak yang besar terhadap masyarakat.

ERA KEEMPAT :

◊ Expert system

Suatu penerapan A.I. (Artificial Intellegence) pada bidang-bidang tertentu, misalnya bidang kedokteran, komunikasi, dll.

◊ AI Machine

Suatu mesin yang dapat meniru kerja dari sebagian otak manusia. Misalnya mesin robot, komputer catur.

◊ Parallel Architecture

Arsitektur komputer yang memungkinkan proses kerja LAN paralel, yang dimungkinkan adanya prosesor berbeda dalam satu komputer

ARTI SOFTWARE

1. Instruksi

Atau program komputer yang ketika dieksekusi akan memberi fungsi dan hasil yang diinginkan.

2. Struktur data

Yang memungkinkan program memanipulasi informasi

3. Dokumen

Yang menggambarkan operasi dan penggunaan program.

SIFAT DAN KARAKTERISTIK SOFTWARE

1. Software merupakan elemen sistem logik dan bukan elemen sistem fisik seperti hardware
2. Elemen itu tidak aus, tetapi bisa rusak.
3. Elemen software itu direkayasa atau dikembangkan dan bukan dibuat di pabrik seperti hardware
4. Software itu tidak bisa dirakit.

KOMPONEN SOFTWARE

1. Bentuk bahasa

Terbagi 2, yaitu

A. High Level, contoh PASCAL, COBOL, FORTRAN.

B. Middle Level, contoh C

2. Bentuk translator

Terbagi 3 , yaitu :

A. Interpreter

Menerjemahkan dari bahasa tingkat tinggi ke bahasa tingkat rendah secara satu persatu (statemen demi statemen)

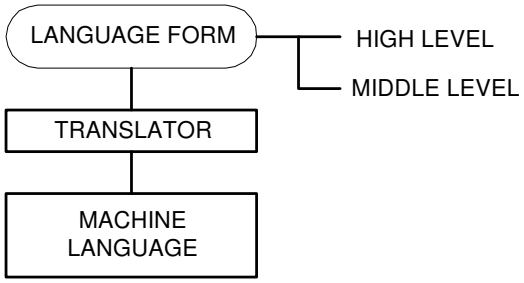
B. Compiler

Menerjemahkan secara keseluruhan, proses lebih cepat dari interpreter

C. Assembler

Menerjemahkan dari bahasa rakitan ke bahasa mesin

3. Bentuk mesin :



APLIKASI SOFTWARE

1. Sistem Software

Adalah sekumpulan program yang ditulis untuk melayani atau menunjang program lainnya. Beberapa sistem software seperti compiler, editor, komponen-komponen sistem operasi, driver dan prosesor telekomunikasi.

2. Real Time software

Software yang mengukur, menganalisis dan mengontrol kejadian yang sesungguhnya terjadi di dunia. Elemen-elemen real time software terdiri dari :

A. Komponen pengumpul data

Yang mengumpulkan dan menyusun informasi dari lingkungan external.

B. Komponen analisis

Yang mentransformasikan informasi yang diperlukan oleh aplikasi

C. Komponen kontrol

Yang memberikan respon kepada lingkungan external

D. Komponen monitor

Yang mengkoordinasi semua komponen-komponen lainnya, sehingga respons real time yang berkisar 1 milisecond sampai 1 menit dapat dipertahankan.

Perlu dicatat bahwa istilah real time berbeda dari istilah interactive atau time sharing. Sistem real time harus memberikan respons pada waktu yang ditentukan, sedangkan pada sistem interactive atau time sharing respons time biasanya melebihi batas waktu yang ditentukan tanpa merusak hasil.

3. Business software

Software yang palinmg banyak digunakan dalam bidang aplikasi software. Software ini digunakan oleh manajemen untuk mengambil kepitusan (Decision Making) dalam bidang bisnis. Contoh :

- ◊ DAC EASY ACCOUNTING
- ◊ FINANCE MANAJER

4. Engineering and sciencetific software

Software yang dicirikan dengan algoritma numerik, aplikasinya berkisar dari astronomi sampai vulkanologi, dari analis ketegangan otomotif sampai dinamika orbit ruang angkasa. Software ini banyak digunakan dalam bidang engineering dan science. Contoh

- ◊ CAD / CAM (Computer Aided Design / Computer Aided Manufacture - Ssimulasi sistem)

5. Emdebed software

Suatu software disimpan dalam memori tetap - ROM - Read Only Memory, dan digunakan untuk mengontrol product dan sistem software ini dijalankan dengan berbagai fungsi terbatas.

6. PC software (Personal Computer)

Software yang banyak digunakan di komputer pribadi (PC). Contoh :

- ◊ Word Processing : WS, WP
- ◊ Spreadsheet : Lotus, Supercalc
- ◊ Computer Graphics : Printshop, Print Magic
- ◊ Games : Paoman, Load Runner
- ◊ DBMS : Dbase III+, Foxbase, Clipper
- ◊ Network : LAN, Novell

7. Artificial Intelegence software

Software yang banyak menggunakan algoritma non numerik dalam memecahkan masalah kompleks yang tidak dapat dianalisis dengan analisis komputasi biasa. Saat ini bidang AI yang paling aktif adalah expert system atau knowledge base system. Bidang aplikasi lain dari software AI adalah pengenalan citra dan suara (image and voice pattern recognition), teorema pembuktian dan permainan / games.

KRISIS SOFTWARE

Adalah sekumpulan masalah yang ditemukan dalam pengembangan software computer. Masalahnya tidak hanya terbatas pada software yang tidak berfungsi sebagaimana mestinya, tetapi krisis software ini terdiri dari masalah yang berhubungan dengan :

1. Bagaimana mengembangkan software
2. Bagaimana memelihara software ynag ada, yang berkembang dalam jumlah besar
3. Bagaimana mengimbangi permintaan software yang makin besar.

MASALAH

Krisis software oleh beberapa masalah :

1. Estimasi jadual dan biaya yang seringkali tidak tepat
2. Produktivitas orang-orang software yang tidak dapat mengimbangi permintaan software
3. Kualitas software yang kurang baik.

Penyebab :

Masalah yang berhubungan dengan krisis software disebabkan oleh :

1. Karakteristik software itu sendiri

Karakteristik software adalah software yang bersifat logika dibandingkan fisik, oleh karena itu mengukur software harus merupakan suatu kesatuan, tidak seperti hardware. Software yang bersifat tidak aus ini menyebabkan kesalahan yang terjadi pada software. Umumnya terjadi pada tahap pengembangan. Manajer tingkat menengah dan tingkat atas yang tidak mempunyai latar belakang software, seringkali diberi tanggung jawab untuk mengembangkan software. Padahal tidak semua manajer itu dapat me-manage semua proyek.

Praktisnya : software programmer atau software engineering mendapatkan latihan formal yang sedikit dalam hal tehnik baru pengembangan software.

2. Kegagalan mereka yang bertanggung jawab dalam pengembangan software.

MITOS SOFTWARE

1. Mitos managements
   A. Kita tidak perlu mengubah pendekatan terhadap pengembangan software, karena jenis pemrograman yang kita lakukan sekarang ini sudah kita lakukan 10 tahun yang lalu.

   Realitasnya : Walau hasil program sama, produktivitas dan kualitas software harus ditingkatkan dengan menggunakan pendekatan software developments

   B. Kita sudah mempunyai buku yang berisi standarisasi dan prosedur untuk pembentukan software.

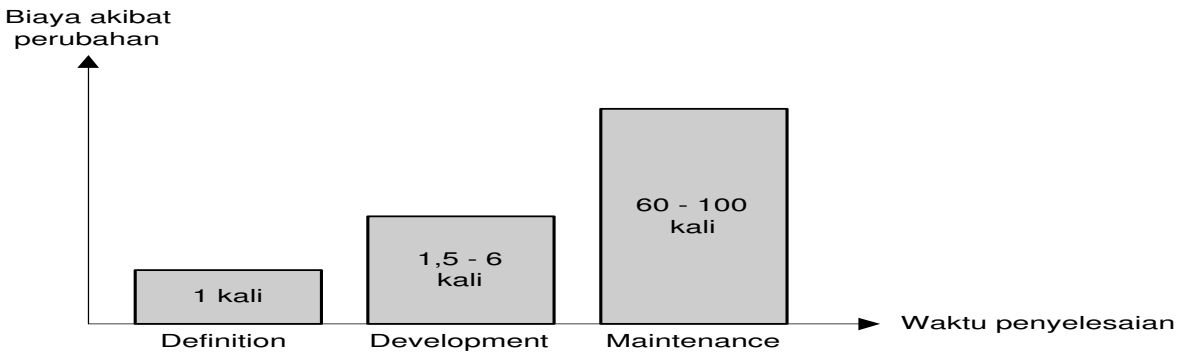
   Realitasnya : Memang buku tersebut ada, tetapi apakah buku tersebut sudah dibaca atau buku tersebut sudah ketinggalan jaman (out of date).

   C. Jika kita tertinggal dari jadwal yang ditetapkan, kita menambah beberapa programmer saja. Konsep ini sering disebut **Mongolian harde concept**.
2. Mitos Langganan / Customer

A. Pernyataan tujuan umum sudah cukup untuk memulai penulisan program. Penjelasan yang lebih rinci akan menyusul kemudian.

Realitasnya : Definisi awal yang buruk adalah penyebab utama kegagalan terhadap usaha-usaha pembentukkan software. Penjelasan yang formal dan terinci tentang informasi fungsi performance interface, hambatan desain dan kriteria validasi adalah penting. Karakteristik di atas dapat ditentukan hanya setelah adanya komunikasi antara customer dan developer.

B. Kebutuhan proyek yang terus menerus berubah dapat dengan mudah diatasi karena software itu bersifat fleksibel. Kenyataannya memang benar bahwa kebutuhan software berubah, tetapi dampak dari perubahan berbeda dari waktu ke waktu.



Kesimpulan : Jika perubahan mendekati akhir penyelesaian, maka biaya akan lebih besar.

3. Mitos Praktisi

A. Tidak ada metode untuk analisis disain dan testing terhadap suatu pekerjaan, cukup menuju ke depan terminal dan mulai coding.

Realitasnya : Metode untuk analisis desain dan testing diperlukan dalam pengembangan software.

B. Segera setelah software digunakan, pemeliharaan dapat diminimalisasikan dan diatasi dengan cara "CATCH AS CATCH CAM".

Realitasnya : Diperlukan budget yang besar dalam maintenance software. Pemeliharaan software harus diorganisir, direncanakan dan dikontrol seolah-olah sebagai suatu proyek besar dalam sebuah organisasi.

MODEL SOFTWARE ENGINEERING

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

Definisi :

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.

Software engineering terdiri dari 3 elemen kunci, yaitu :

1. Metode,
2. Peralatan (tools),
3. Prosedur,

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

1. Metode Software Enginnering

Metode software engineering memberikan tehnik-tehnik 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 Software Engineering

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

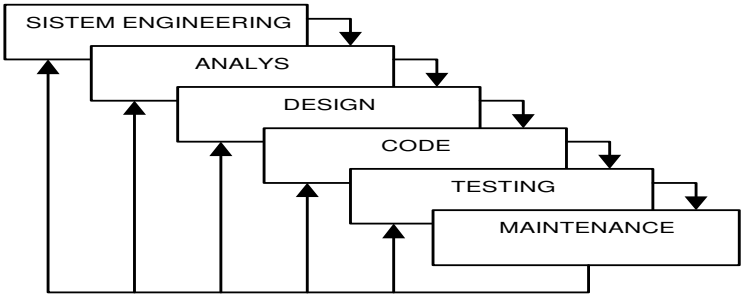
3. Prosedur Software Engineering

Terdiri dari :

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

Dalam penguasaan atas model software engineering atau software engineering paradigm, dikenal ada 3 metode yang luas dipergunakan, yaitu :

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



Keterangan :

A. 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

B. 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.

C. 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.

D. Coding

Desain harus diterjemahkan ke dalam bentuk yang dapat dibaca oleh mesin

E. 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.

F. Maintenance

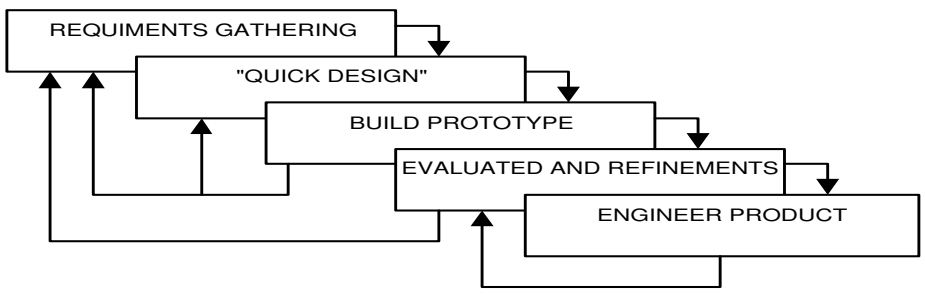
Software yang sudah dikirim ke customer data berubah karena

- ◊ 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 pda aplikasi yang dibentuk oleh model ini.
- ◊ Seringkali pada awalnya customer sulit menentukan semua kebutuhan secara explisit (jelas).
- ◊ Customer harus sabar karena versi program yang jalan tidak akan tersedia sampai proyek software selesai dalam waktu yang lama.

2. **Prototype Paradigm**



Keterangan :

Seringkali seorang customer sulit menentukan input yang lebih terinci, proses yang diinginkan dan output yang diharapkan. Tentu saja ini menyebabkan developer tidak yakin dengan efisiensi alogoritma yang dibuatnya, sulit menyesuaikan sistem operasi, serta interaksi manusia dan mesin yang harus diambil. Dalam hal seperti ini, pendekatan prototype untuk software engineering merupakan langkah yang terbaik. *Prototype sebenarnya adalah suatu proses yang memungkinkan developer membuat sebuah model software*.

Ada 2 bentuk dari model ini, yaitu :

A. Paper Prototype

Menggambarkan interaksi manusia dan mesin dalam sebuah bentuk yang memungkinkan user mengerti bagaimana interaksi itu terjadi.

B. Working Prototype

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

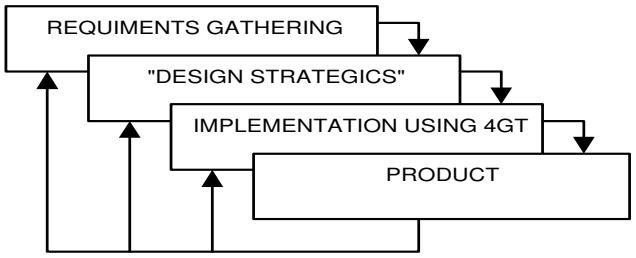
- ◊ Pengumpulan kebutuhan developer dan customer
- ◊ Menentukan semua tujuan software
- ◊ Mengidentifikasi kebutuhan-kebutuhan yang diketahui

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, selanjutanya 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

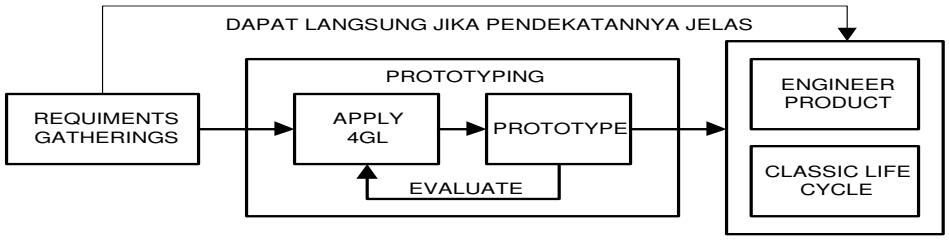
Keterangan gambar :

- ◊ Model 4GT untuk software engineering dimulai dengan rangkaian pengumpulan kebutuhan. Idealnya, seorang customer menjelaskan kebutuhan-kebutuhan yang selanjutnay diterjemahkan ke dalam prototype. Tetapi ini tidak dapat dilakukan karena customer tidak yakin dengan apa yang diperlukan, tidak jelas dalam menetapkan fakta-fakta yang diketahui dan tidak dapat menentukan informasi yang diinginkan oleh peralatan 4GT.
- ◊ Untuk aplikasi kecil adalah mungkin bergerak langsung dari langkah pengumpulan kebutuhan ke implementasi yang menggunakan bahasa non prosedur fourth generation (generasi ke 4). Tetapi untuk proyek besar, pengembangan strategi desain sistem tetap diperlukan, sekalipun kita menggunakan 4GL. Penggunaan 4GT tanpa desain untuk proyek besar akan menyebabkan masalah yang sama yang ditemui dalam pengembangan software yang menggunakan pendekatan konvensional.
- ◊ Implementasi yang menggunakan 4GL memungkinkan developer software menjelaskan hasil yang diinginkan yang kemudian diterjemahkan ke dalam bentuk source code dan object code secara otomatis.
- ◊ Langkah yang terakhir adalah mengubah implementasi 4GT ke dalam sebuah product. Selanjutnya developer harus melakukan pengetesan, pengembangan dokumentasi dan pelaksanaan semua aktifitas lainnya yang diwujudkan dalam model software engineering.

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

A. peralatan 4GT tidak semudah penggunaan bahasa pemrograman,
B. source code yang dihasilkan oleh peralatan ini tidak efisien,
C. 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
2. Fase Pengembangan (Development)
3. Fase Pemeliharaan (Maintenance)

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.

A. Sistem Analis

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

B. 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 jadual.

C. 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

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.

A. 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).

B. Coding

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

C. 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

Fase pemelihaaan berfokus pada "*Change*" atau perubahan. Ini dapat disebabkan :

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