EBOOK PEMODELAN PERANGKAT LUNAK

NAMA : M. IRFAN NAWAWI
KELAS: XI RPL 3

KATA PENGANTAR

Puji syukur bagi Allah SWT yang telah memberikan nikmat serta hidayah-Nya sehingga penyusun dapat menyelesaikan paper yang berjudul “Pemodelan Perangkat Lunak”. Paper ini disususn bertujuan untuk memenuhi salah tugas mata kuliah Pengantar Basis Data.

Penyusun mengucapkan terima kasih yang sebesar-besarnya kepada Bapak Zul Hilmi S.T. selaku pengajar mata pelajaran PPL dan kepada segenap pihak yang telah membantu menyelesaikan penulisan paper ini.

Penyusun menyadari bahwa paper ini jauh dari kesempurnaan, masih banyak terdapat kekurangan-kekurangan dalam penulisan paper ini, maka dari itu penyusun mengharapkan kritik dan saran yang konstruktif dari para pembaca demi kesempurnaan paper ini.

Terima kasih, dan semoga paper ini bisa memberikan sumbangsih positif bagi kita semua

Tasikmalaya, November 2017

M Irfan Nawawi

PENGERTIAN

Pemodelan dalam suatu rekayasa perangkat lunak merupakan suatu hal yang dilakukan di tahapan awal. Di dalam suatu rekayasa dalam perangkat lunak sebenarnya masih memungkinkan tanpa melakukan suatu pemodelan. Namun hal itu tidak dapat lagi dilakukan dalam suatu industri perangkat lunak. Pemodelan delam perangkat lunak merupakan suatu yang harus dikerjakan di bagian awal dari rekayasa, dan pemodelan ini akan mempengaruhi perkerjaan-pekerjaan dalam rekayasa perangkat lunak tersebut.

Di dalam suatu industri dikenal berbagai macam proses, demikian juga halnya dengan industri perangkat lunak. Perbedaan proses yang digunakan akan menguraikan aktivitas-aktivitas proses dalam cara-cara yang berlainan. Perusahaan yang berbeda menggunakan proses yang berbeda untuk menghasilkan produk yang sama. Tipe produk yang berbeda mungkin dihasilkan oleh sebuah perusahaan dengan menggunakan proses yang berbeda. Namun beberapa proses lebih cocok dari lainnya untuk beberapa tipe aplikasi. Jika proses yang salah digunakan

akan mengurangi kualitas kegunaan produk yang dikembangkan.

Pada rekayasa perangkat lunak, banyak model yang telah dikembangkan untuk membantu proses pengembangan perangkat lunak. Model-model ini pada umumnya mengacu pada model proses pengembangan sistem yang disebut *System Development Life Cycle (SDLC)* seperti terlihat pada gambar dibawah ini,

Setiap model yang dikembangkan mempunyai karakteristik sendiri. Namun secara umum ada persamaannya, yaitu :

1. Kebutuhan terhadap definisi masalah yang jelas.
   Input utama dari setiap model pengembangan perangkat lunak adalah pendefinisian masalah yang jelas. Semakin jelas akan semakin baik karena akan memudahkan dalam penyelesaian masalah. Oleh

karena itu pemahaman masalah merupakan bagian penting dari model pengembangan perangkat lunak.

2. Tahapan-tahapan pengembangan yang teratur. Meskipun model-model pengembangan perangkat lunak memiliki pola yang berbeda-beda, biasanya model-model tersebut mengikuti pola umum analysis – design – coding – testing – maintenance.

3. Stakeholder berperan sangat penting dalam keseluruhan tahapan pengembangan. Stakeholder dalam rekayasa perangkat lunak dapat berupa pengguna, pemilik, pengembang, pemrogram dan orang-orang yang terlibat dalam rekayasa perangkat lunak tersebut.

4. Dokumentasi merupakan bagian penting dari pengembangan perangkat lunak. Masing-masing tahapan dalam model biasanya menghasilkan sejumlah tulisan, diagram, gambar atau bentuk-bentuk lain yang harus didokumentasi dan merupakan bagian tak terpisahkan dari perangkat lunak yang dihasilkan.

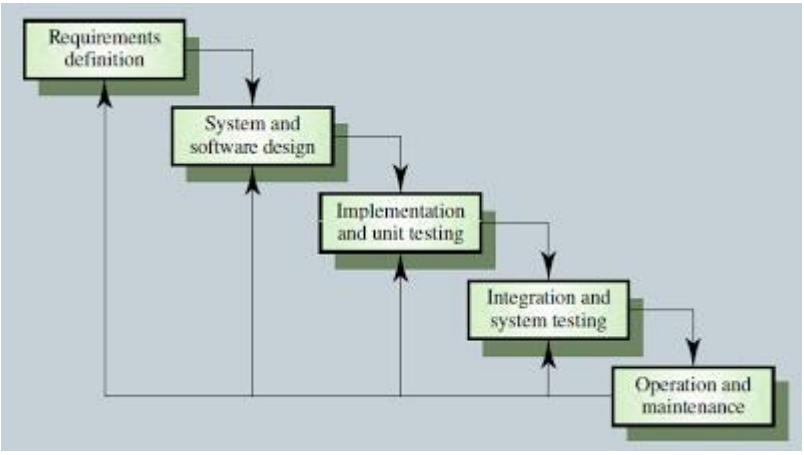
5. Keluaran dari proses pengembangan perangkat lunak harus bernilai ekonomis. Nilai dari sebuah perangkat lunak sebenarnya agak susah dirupiah- kan.

Namun efek dari penggunaan perangkat lunak yang telah dikembangkan haruslah memberi nilai tambah bagi organisasi. Hal ini dapat Setiap model yang dikembangkan mempunyai karakteristik sendiri-sendiri.

Model proses perangkat lunak masih menjadi object penelitian, tapi sekarang ada banyak model umum atau paradigma yang berbeda dari pengembangan perangkat lunak, antara lain:

A. Waterfall Model

Sebuah pendekatan pengembangan perangkat lunak sistematik dan sekuensial. Disebut juga "Classic Life Cycle". Disebut waterfall (berarti air terjun) karena memang diagram tahapan prosesnya mirip dengan air terjun yang bertingkat, berikut diagram tahapannya :



Aktivitas Waterfall Model

Requirements analysis and definition : mengumpulkan kebutuhan secara lengkap kemudian dianalisis dan didefinisikan kebutuhan yang hasrus dipenuhi oleh program yang akan dibangun.

System and software design : desain dikerjakan setelah kebutuhan selesai dikumpulkan secara lengkap.

Implementation and unit testing : desain program diterjemahkan ke dalam kode-kode dengan menggunakan bahasa pemrograman yang sudah ditentukan. Program yang dibangun langsung diuji.

Integration and system testing : penyatuan unit-unit program kemudian diuji secara keseluruhan (system testing)

Operation and maintenance : mengoperasikan program dilingkungannya dan melakukan pemeliharaan, seperti penyesuaian atau perubahan karena adaptasi dengan situasi sebenarnya.

Keunggulan dari waterfall:

1. Software yang dikembangkan dengan metode ini biasanya menghasilkan kualitas yang baik.

2. Dokumen pengembangan sistem sangat terorganisir, karena setiap fase harus terselesaikan dengan lengkap sebelum melangkah ke fase berikutnya.

Kekurangan dari waterfall:

1. Perubahan sulit dilakukan karena sifatnya yang kaku.
2. Karena sifat kakunya, model ini cocok ketika kebutuhan dikumpulkan secara lengkap sehingga perubahan bisa ditekan sekecil mungkin. Tapi pada kenyataannya jarang sekali konsumen/pengguna yang bisa memberikan kebutuhan secara lengkap, perubahan kebutuhan adalah sesuatu yang wajar terjadi.
3. Waterfall pada umumnya digunakan untuk rekayasa sistem yang besar dimana proyek dikerjakan di beberapa tempat berbeda, dan dibagi menjadi beberapa bagian sub-proyek.

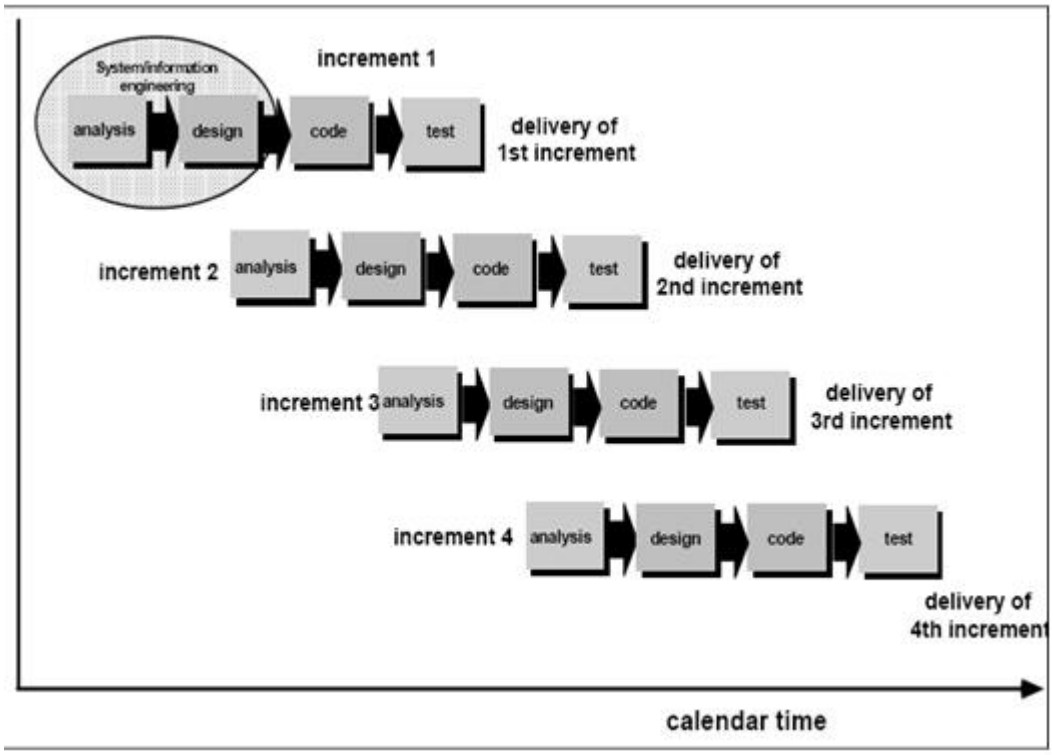
B. Evolutionary Software Process Models

Bersifat iteratif/ mengandung perulangan. Hasil proses berupa produk yang makin lama makin lengkap sampai versi terlengkap dihasilkan sebagai produk akhir dari

proses. Dua model dalam **evolutionary software process model** adalah:

1. Incremental Model

Incremental Model merupakan gabungan antara model linear sekuensial dan prototyping. Setiap linear sekuen menghasilkan produk yang deliveriables. Increment pertama merupakan produk inti yang mengandung persyaratan/kebutuhan dasar. Penambahan dilakukan pada increment-incremet berikutnya.



Keterangan:

1. Kombinasikan elemet-element dari waterfall dengan sifat iterasi/perulangan.
2. Element-element dalam waterfall dikerjakan dengan hasil berupa produk dengan spesifikasi tertentu, kemudian proses dimulai dari fase pertama hingga akhir dan menghasilkan produk dengan spesifikasi yang lebih lengkap dari yang sebelumnya. Demikian seterusnya hingga semua spesifikasi memenuhi kebutuhan yang ditetapkan oleh pengguna.
3. Produk hasil increment pertama biasanya produk inti (core product), yaitu produk yang memenuhi kebutuhan dasar. Produk tersebut digunakan oleh pengguna atau menjalani review/pengecekan detil. Hasil review tersebut menjadi bekal untuk pembangunan pada increment berikutnya. Hal ini terus dikerjakan sampai produk yang komplit dihasilkan.
4. Model ini cocok jika jumlah anggota tim pengembang/pembangun PL tidak cukup.
5. Mampu mengakomodasi perubahan secara fleksibel.

6. Produk yang dihasilkan pada increment pertama bukanlah prototype, tapi produk yang sudah bisa berfungsi dengan spesifikasi dasar.

Keunggulan dari Incremental Model :

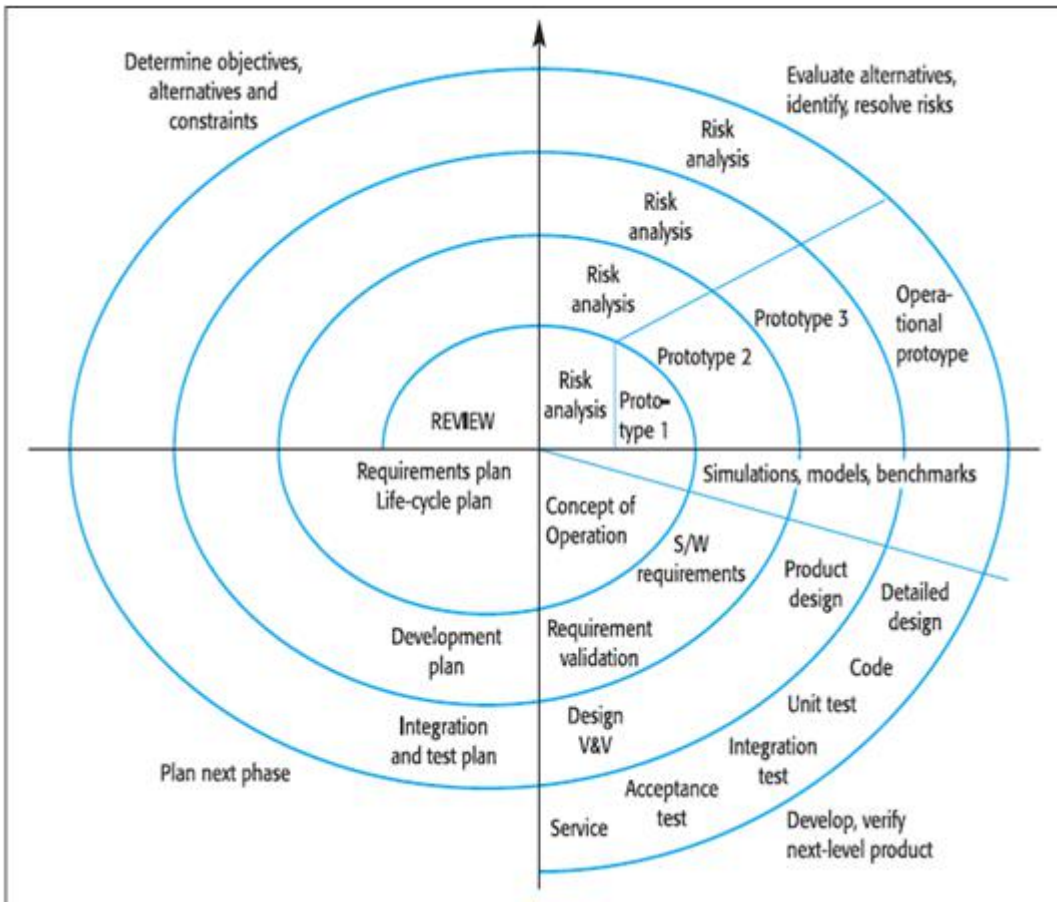
1. Personil bekerja optimal
2. Pihak konsumen dapat langsung menggunakan dahulu bagian-bagian yang telah selesai dibangun. Contohnya pemasukan data karyawan
3. Mengurangi trauma karena perubahan sistem. Klien dibiasakan perlahan-lahan menggunakan produknya bagian per bagian
4. Memaksimalkan pengembalian modal investasi konsumen

Kekurangan dari Incremental Model :

1. Cocok untuk proyek berukuran kecil (tidak lebih dari 200.000 baris coding)
2. Mungkin terjadi kesulitan untuk memetakan kebutuhan pengguna ke dalam rencana spesifikasi masing-masing hasil increment

3. Dapat menjadi build and Fix Model, karena kemampuannya untuk selalu mendapat perubahan selama proses rekayasa berlangsung.

2. Spiral Model

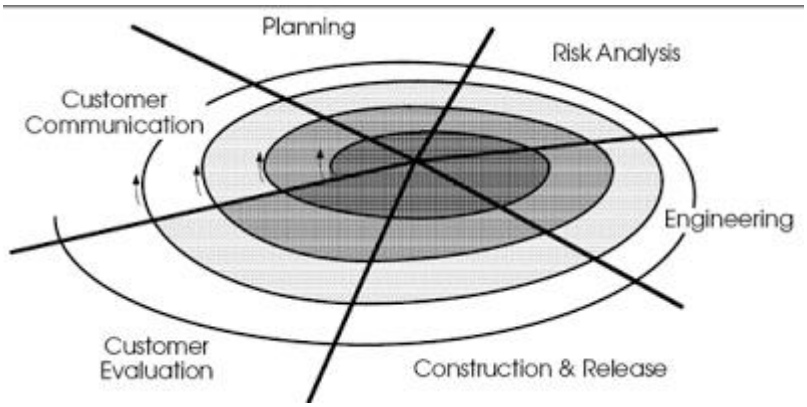


Proses digambarkan sebagai spiral. Setiap loop mewakili satu fase dari software process. Loop paling dalam berfokus pada kelayakan dari sistem, loop selanjutnya tentang definisi dari kebutuhan, loop berikutnya berkaitan dengan desain sistem dan seterusnya. Setiap Loop dibagi menjadi beberapa sektor :

1. Objective settings (menentukan tujuan): menentukan tujuan dari fase yang ditentukan. Batasan-batasan pada proses dan produk sudah diketahui. Perencanaan sudah disiapkan. Resiko dari proyek sudah diketahui. Alternatif strategi sudah disiapkan berdasarkan resiko-resiko yang diketahui, dan sudah direncanakan.
2. Risk assessment and reduction (Penanganan dan pengurangan resiko): setiap resiko dianalisis secara detil pada sektor ini. Langkahlangkah penanganan dilakukan, misalnya membuat prototype untuk mengetahui ketidakcocokan kebutuhan.
3. Development and Validation (Pembangunan dan pengujian): Setelah evaluasi resiko, maka model pengembangan sistem dipilih. Misalnya jika resiko user interface dominan, maka membuat prototype User Interface. Jika bagian keamanan yang bermasalah, maka menggunakan model formal dengan perhitungan matematis, dan jika masalahnya adalah integrasi sistem model waterfall lebih cocok.
4. Planning: Proyek dievaluasi atau ditinjau-ulang dan diputuskan untuk terus ke fase loop selanjutnya atau

tidak. Jika melanjutkan ke fase berikutnya rencana untuk loop selanjutnya.

Pembagian sektor tidak bisa saja dikembangkan seperti pada pembagian sektor berikut pada model variasi spiral di bawah ini:



Customer communication: membangun komunikasi yang baik dengan pengguna/customer.

Planning: mendefinisikan sesumber, batas waktu, informasi-informasi lain seputar proyek

Risk analysis: identifikasi resiko managemen dan teknis

Engineering: pembangunan contoh-contoh aplikasi, misalnya prototype

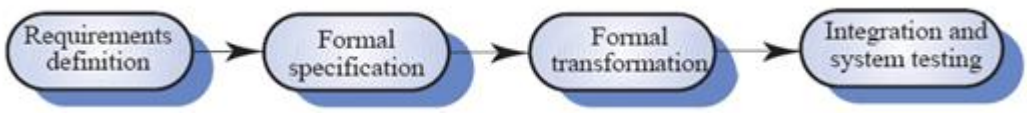
Construction and release : pembangunan, test, install dan support.

Customer evaluation: mendapatkan feedback dari pengguna berdasarkan evaluasi PL

Pada model spiral, resiko sangat dipertimbangkan. Resiko adalah sesuatu yang mungkin mengakibatkan kesalahan. Model spiral merupakan pendekatan yang realistik untuk PL berskala besar. Pengguna dan pembangun bisa memahami dengan baik software yang dibangun karena setiap kemajuan yang dicapai selama proses dapat diamati dengan baik. Namun demikian, waktu yang cukup panjang mungkin bukan pilihan bagi pengguna, karena waktu yang lama sama dengan biaya yang lebih besar.

C. Transformasi Formal

Metode ini berbasiskan pada transformasi spesifikasi secara matematik melalui representasi yang berbeda untuk suatu program yang dapat dieksekusi. Trasformasi menyatakan spesifikasi program Menggunakan pendekatan 'Cleanroom' untuk pengembangan PL.



Metode ini mempunyai keterbatasan dalam pemakaiannya. Keunggulannya adalah mengurangi jumlah kesalahan pada sistem sehingga penggunaan utamanya adalah pada sistem yang kritis. Hal ini menjadi efektif dari segi biaya.

Pemakaian model pengembangan formal memerlukan tingkat kerahasian sebelum digunakan.
Permasalahan dalam model pengembangan metode formal:

Memerlukan keahlian khusus dan pelatihan untuk mengaplikasikannya

Sulit menentukan beberapa aspek dari suatu sistem seperti user interface

D. Model Rapid Aplication Development (RAD)

Rapid Aplication Development (RAD) adalah sebuah model proses perkembangan software sekuensial linier yang menekankan siklus perkembangan yang sangat pendek. Model RAD ini merupakan sebuah adaptasi "kecepatan tinggi" dari model sekuensial linier di mana perkembangan cepat dicapai dengan menggunakan pendekatan kontruksi berbasis komponen. Jika kebutuhan dipahami dengan baik, proses RAD

memungkinkan tim pengembangan menciptakan “sistem fungsional yang utuh” dalam periode waktu yang sangat pendek (kira-kira 60 sampai 90 hari). Karena dipakai terutama pada aplikasi sistem konstruksi, pendekatan RAD melingkupi fase – fase sebagai berikut :

1. Bussiness modeling

Aliran informasi di antara fungsi – fungsi bisnis dimodelkan dengan suatu cara untuk menjawab pertanyaan – pertanyaan berikut : informasi apa yang mengendalikan proses bisnis? Informasi apa yang di munculkan? Siapa yang memunculkanya? Ke mana informasi itu pergi? Siapa yang memprosesnya?

2. Data modeling

Aliran informasi yang didefinisikan sebagai bagian dari fase bussiness modelling disaring ke dalam serangkaian objek data yang dibutuhkan untuk menopang bisnis tersebut. Karakteristik (disebut atribut) masing–masing objek diidentifikasi dan hubungan antara objek – objek tersebut didefinisikan.

3. Prosess modeling

Aliran informasi yang didefinisikan di dalam fase data modeling ditransformasikan untuk mencapai aliran

informasi yang perlu bagi implementasi sebuah fungsi bisnis. Gambaran pemrosesan diciptakan untuk menambah, memodifikasi, menghapus, atau mendapatkan kembali sebuah objek data.

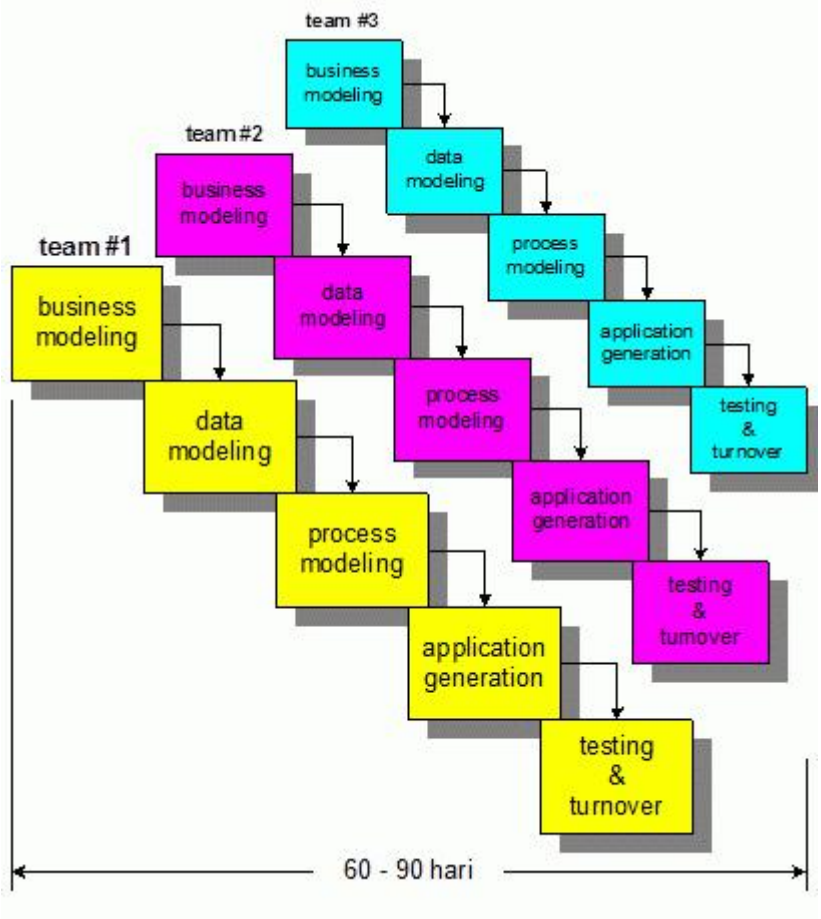
4. Aplication generation

RAD mengasumsikan pemakaian teknik generasi ke empat. Selain menciptakan perangkat lunak dengan menggunakan bahasa pemrograman generasi ketiga yang konvensional, RAD lebih banyak memproses kerja untuk memkai lagi komponen program yang ada (pada saat memungkinkan) atau menciptakan komponen yang bisa dipakai lagi (bila perlu). Pada sem

ua kasus, alat – alat bantu otomatis dipakai untuk memfasilitasi konstruksi perangkat lunak.

5. Testing and turnover

Karena proses RAD menekankan pada pemakaian kembali, banyak komponen program telah diuji. Hal ini mengurangi keseluruhan waktu pengujian. Tetapi komponen baru harus di uji dan semua interface harus dilatih secara penuh.



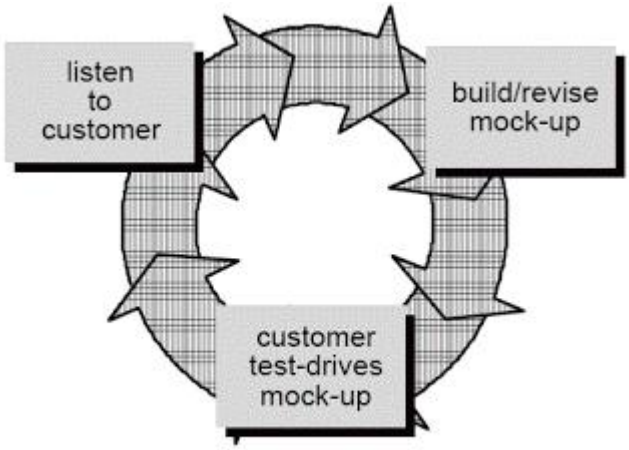
Keunggulan model RAD adalah :

1. Setiap fungsi mayor dapat dimodulkan dalam waktu tertentu kurang dari 3 bulan dan dapat dibicarakan oleh tim RAD yang terpisah dan kemudian diintegrasikan sehingga waktunya lebih efisien
2. RAD mengikuti tahap pengembangan sistem seperti umumnya, tetapi mempunyai kemampuan untuk menggunakan kembali komponen yang ada sehingga pengembang tidak perlu membuat dari awal lagi dan waktu yang lebih singkat

Kekurangan model RAD adalah :

1. Bagi proyek yang besar tetapi berskala, RAD memerlukan sumber daya manusia yang memadai untuk menciptakan jumlah tim RAD yang baik.
2. RAD menuntut pengembangan dan pelanggan memiliki komitmen di dalam aktivitas rapid-fire yang diperlukan untuk melengkapi sebuah sistem, di dalam kerangka waktu yang sangat diperpendek. Jika komitmen tersebut tidak ada, proyek RAD akan gagal.

E. Prototyping Model



Gambar : model Prototyping

Kadang-kadang klien hanya memberikan beberapa kebutuhan umum software tanpa detil input, proses atau detil output. Di lain waktu mungkin dimana tim pembangun (developer) tidak yakin terhadap efisiensi

dari algoritma yang digunakan, tingkat adaptasi terhadap sistem operasi atau rancangan form user interface. Ketika situasi seperti ini terjadi model prototyping sangat membantu proses pembangunan software.

Proses pada model prototyping yang digambarkan pada gambar model prototyping, bisa dijelaskan sebagai berikut:

Pengumpulan kebutuhan: developer dan klien bertemu dan menentukan tujuan umum, kebutuhan yang diketahui dan gambaran bagian-bagian yang akan dibutuhkan berikutnya. Detil kebutuhan mungkin tidak dibicarakan disini, pada awal pengumpulan kebutuhan.

Perancangan : perancangan dilakukan cepat dan rancangan mewakili semua aspek software yang diketahui, dan rancangan ini menjadi dasar pembuatan prototype.

Evaluasi prototype: klien mengevaluasi prototype yang dibuat dan digunakan untuk memperjelas kebutuhan software.

Perulangan ketiga proses ini terus berlangsung hingga semua kebutuhan terpenuhi. Prototype-prototype dibuat untuk memuaskan kebutuhan klien dan untuk memahami kebutuhan klien lebih baik. Prototype yang dibuat dapat dimanfaatkan kembali untuk membangun software lebih cepat, namun tidak semua prototype bisa dimanfaatkan. Sekalipun prototype memudahkan komunikasi antar developer dan klien, membuat klien mendapat gambaran awal dari prototype , membantu mendapatkan kebutuhan detil lebih baik namun demikian prototype juga menimbulkan masalah:

Dalam membuat prototype banyak hal yang diabaikan seperti efisiensi, kualitas, kemudahan dipelihara/dikembangkan, dan kecocokan dengan lingkungan yang sebenarnya. Jika klien merasa cocok dengan prototype yang disajikan dan berkeras terhadap produk tersebut, maka developer harus kerja keras untuk mewujudkan produk tersebut menjadi lebih baik, sesuai kualitas yang seharusnya.

Developer biasanya melakukan kompromi dalam beberapa hal karena harus membuat prototype dalam waktu singkat. Mungkin sistem operasi yang tidak

sesuai, bahasa pemrograman yang berbeda, atau algoritma yang lebih sederhana. Agar model ini bisa berjalan dengan baik, perlu disepakati bersama oleh klien dan developer bahwa prototype yang dibangun merupakan alat untuk mendefinisikan kebutuhan software.

F. Component-based Development Model

Component-based development sangat berkaitan dengan teknologi berorientasi objek. Pada pemrograman berorientasi objek, banyak class yang dibangun dan menjadi komponen dalam suatu software. Class-class tersebut bersifat reusable artinya bisa digunakan kembali. Model ini bersifat iteratif atau berulang-ulang prosesnya.

Secara umum proses yang terjadi dalam model ini adalah:

1. Identifikasi class-class yang akan digunakan kembali dengan menguji class tersebut dengan data yang akan dimanipulasi dengan aplikasi/software dan algoritma yang baru
2. Class yang dibuat pada proyek sebelumnya disimpan dalam class library, sehingga bisa langsung

diambil dari library yang sudah ada. Jika ternyata ada kebutuhan class baru, maka class baru dibuat dengan metode berorientasi objek.

3. Bangun software dengan class-class yang sudah ditentukan atau class baru yang dibuat, integrasikan.

Penggunaan kembali komponen software yang sudah ada menguntungkan dari segi:

►Siklus waktu pengembangan software, karena mampu

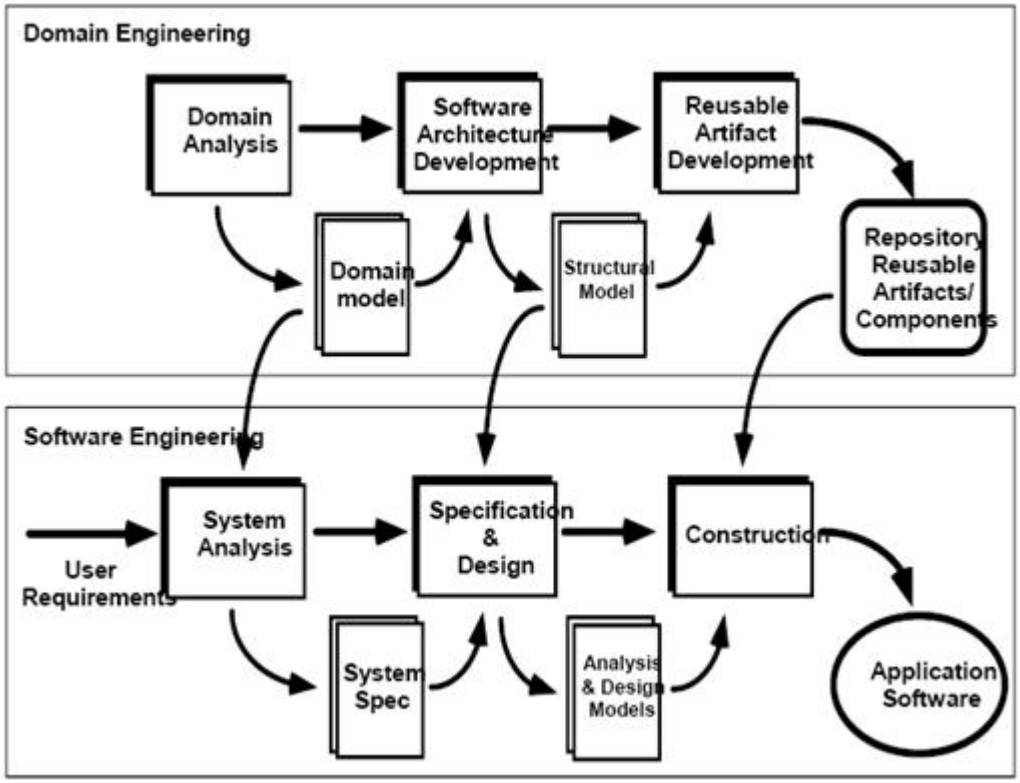
mengurangi waktu 70%

►Biaya produksi berkurang sampai 84% arena

pembangunan komponen berkurang

Pembangunan software dengan menggunakan komponen yang sudah tersedia dapat menggunakan komponen COTS (Commercial off-the-shelf) – yang bisa didapatkan dengan membeli atau komponen yang sudah dibangun sebelumnya secara internal. Component-Based Software Engineering (CBSE) adalah proses yang menekankan

perancangan dan pembangunan software dengan menggunakan komponen software yang sudah ada. CBSE terdiri dari dua bagian yang terjadi secara paralel yaitu software engineering (component-based development) dan domain engineering seperti yang digambarkan pada gambar dibawah ini,



1. Domain engineering menciptakan model domain bagi aplikasi yang akan digunakan untuk menganalisis kebutuhan pengguna. Identifikasi, pembangunan, pengelompokan dan pengalokasikan

komponen-komponen software supaya bisa digunakan pada sistem yang ada dan yang akan datang.

2. Software engineering (component-based development) melakukan analisis terhadap domain model yang sudah ditetapkan kemudian menentukan spesifikasi dan merancang berdasarkan model struktur dan spesifikasi sistem, kemudian melakukan pembangunan software dengan menggunakan komponen-komponen yang sudah ditetapkan berdasarkan analisis dan rancangan yang dihasilkan sebelumnya hingga akhirnya menghasilkan software.

G. Extreme Programming (XP) Model

Model proses ini diciptakan dan dikembangkan oleh Kent Beck. Model ini adalah model proses yang terbaru dalam dunia rekayasa perangkat lunak dan mencoba menjawab kesulitan dalam pengembangan software yang rumit dan sulit dalam implementasi. Menurut Kent Beck XP adalah : “A lightweight, efficient, low-risk, flexible,predictable, scientific and fun way to develop software”. Suatu model yang menekankan pada:

► keterlibatan user secara langsung

► pengujian

► pay-as-you-go design

Sebagai contoh :

Adapun empat nilai penting dari XP :

1. Communication/Komunikasi : komunikasi antara developer dan klien sering menjadi masalah. Karena itu komunikasi dalam XP dibangun dengan melakukan pemrograman berpasangan (pair programming). Developer didampingi oleh pihak klien dalam melakukan coding dan unit testing sehingga klien bisa terlibat langsung dalam pemrograman sambil berkomunikasi dengan developer. Selain itu perkiraan beban tugas jugadiperhitungkan.
2. Simplicity/ sederhana: Menekankan pada kesederhanaan dalam pengkodean: “What is the simplest thing that could possibly work?” Lebih baik melakukan hal yang sederhana dan mengembangkannya besok jika diperlukan. Komunikasi yang lebih banyak mempermudah, dan rancangan yang sederhana mengurangi penjelasan.
3. Feedback / Masukan/Tanggapan: Setiap feedback ditanggapi dengan melakukan tes, unit test atau system integration dan jangan menunda karena biaya akan membengkak (uang, tenaga, waktu).

4. Courage / Berani: Banyak ide baru dan berani mencobanya, berani mengerjakan kembali dan setiap kali kesalahan ditemukan, langsung diperbaiki.

BAB I
PENDAHULUAN

A. Latar Belakang Masalah

Dalam dunia teknologi sekarang pengembangan dalam bidang informatikan telah mengalami perkembangan yang sangat pesat. Dengan perkembangan ini, dalam bidang informatika tidak hanya menghasilkan hanya dalam pengembangan program perangkat lunak saja, melainkan pengambangan dalam bidang suatu permodelan yang bersifat komplek.

Dalam pembuatan sebuah perangkat lunak yang haruslah memiliki Teknik analisa kebutuhan dan teknik permodelan yang baik, supaya terwujudnya suatu perangkat lunak yang baik. Dengan hal tersebut maka perlulah suatu pengenalan mengenai permodelan dalam suatu pembangunan Perangkat Lunak *(Software)*. Berdasarkan tugas yang kami peroleh, kami hanya membatasi penjelasan mengenai permodelan ini.

B. Perumusan Masalah

a) Model proses pengembangan Perangkat lunak ?
b) Model air terjun (Model Waterfall) ?
c) Model Prototyping ?
d) Model RAD (Rapid Aplication Development) ?
e) Model 4GT ?

C. Tujuan Pembuatan

Adapun tujuan pembuatan makalah ini adalah :

a) Supaya kita bisa memahami model proses pengembangan perangkat lunak.
b) Supaya kita bisa menyajikan karakteristik berbagai metode pengembangan perangkat lunak.

BAB II
PEMBAHASAN

A. Pengertian Proses Pengembangan Perangkat Lunak

Proses pengembangan perangkat lunak (Software Process / Development Paradigm) adalah sekumpulan tahap, tugas dan aktivitas yang dibutuhkan untuk secara effisien mentransformasikan kebutuhan pemakai ke suatu solusi perangkat lunak yang efektif.

Pemodelan proses perangkat lunak (Software Process Modeling) bertujuan untuk merepresentasikan aktivitas yang terjadi selama pembuatan perangkat lunak dan perubahan-perubahannya (evolusi). Latar belakang penggunaan model-model tersebut adalah kebutuhan untuk menghasilkan suatu sistem yang

benar sedini mungkin didalam proses pengembangannya.

Alasan utama adalah biaya, Semakin dini suatu kesalahan bisa dideteksi dalam pengembangan sistem, biaya perbaikannya semakin rendah.

B. Model-Model Proses Pengembangan Perangkat Lunak.

1) Model air terjun (Model Waterfall) ?
2) Model Prototyping ?
3) Model RAD (Rapid Aplication Development) ?
4) Spiral Model/Spiral Boehm
5) Model 4GT ?

C. Pengertian Model Proses Pengembangan Perangkat Lunak.

1. Model air terjun (**Model Waterfall**).

Model ini adalah model klasik yang mengusung pengembangan perangkat lunak yang sistematis, berurutan/sekuensial dimulai pada tingkat dan kemajuan system pada seluruh persyaratan dalam analisis, perancangan (desain), pengkodean,

pengujian (testing), hingga ke tahap pemeliharaan dalam membangun software (perangkat lunak).

Berikut ini gambaran dari Linear Sequential Model / waterfall model.

Pada setiap tahapan dianalogikan bak air yang mengalir dari tempat tinggi ke tempat yang lebih rendah, artinya sebuah proses baru bias dilanjutkan setelah satu tahap awal selesai dengan sempurna.

Penjelasan tentang setiap tahapan dapat diringkas sebagai berikut:

a) Tahap analisis: pada tahap ini berlangsung proses pengumpulan kebutuhan secara lengkap untuk dianalisis dan didefinisikan kebutuhan apa saja yang harus dipenuhi oleh program yang akan dibuat, seperti memahami domain permasalahan,

tingkah laku, unjuk kerja dan interface (antar muka).

b) Tahap desain: proses ini melibatkan empat atribut sebuah program yaitu struktur data, arsitektur, perangkat lunak, representasi interface, dan detail (algoritma) prosedural.

c) Tahap pengkodean: proses penterjemahan desain ke dalam bentuk bahasa mesin yang dapat dilakukan secara mekanis.

d) Tahap pengujian: proses ini dikerjakan setelah kode dirancang dan difokuskan pada fungsi dan jumlah kesalahan untuk diperbaiki.

e) Tahap pemeliharaan: meliputi penyesuaian atau perubahan yang berkembang seiring dengan adaptasi perangkat lunak dengan kondisi atau situasi sebenarnya setelah disampaikan kepada konsumen atau pelanggan.

Kelebihan metode ini antara lain mudah diaplikasikan karena urutan-urutan pengerjaan sudah sering dipakai; selain itu juga cocok untuk software berskala besar dan yang bersifat umum; yang paling penting, karena langkah-langkahnya sangat sekuensial, pengerjaan proyek akan mudah dikontrol dan terjadwal dengan baik.

Namun, terdapat pula beberapa kelemahan yang menjadi kekurangan dari metode waterfall ini, seperti kurang fleksibel, dikarenakan rincian prosesnya harus benar-benar jelas dan tidak boleh diubah-ubah. Apabila dikerjakan dengan melampaui tahap yang seharusnya maka proses desain yang sebelumnya itu akan berubah total dan memakan waktu yang banyak jika harus mengulang proses.

Model waterfal ini sangat sesuai digunakan dalam pengembangan sistem perangkat lunak dan hardware yang luas

dan apabila kebutuhan pengguna telah dimengerti dengan baik. Selain itu, juga apabila waktu yang tersedia juga masih cukup banyak.

2. Prototyping Model

Metode ini menyajikan gambaran yang lengkap dari sistem, terdiri atas model kertas, model kerja dan program. Pihak pengembang akan melakukan identifikasi kebutuhan pemakai, menganalisa sistem dan melakukan studi kelayakan serta studi terhadap kebutuhan pemakai, meliputi model interface, teknik prosedural dan teknologi yang akan dimanfaatkan.

Secara ringkas, tahapan-tahapan dalam model prototyping adalah:

a) Tahap Pengumpulan kebutuhan: pada tahap ini, pelanggan dan pengembang saling bantu dalam mendefinisikan format seluruh perangkat lunak,

menentukan keperluan dan garis besar sistem yang akan dirancang.

b) Tahap Quick design: membangun rancangan global sebagai contoh bagi user.

c) Tahap Pembangunan Prototipe: proses perancangan sementara yang fokusnya kepada penyajian kepada pelanggan, termasuk pengujian dan penyempurnaan.

d) Tahap Evaluasi Pelanggan: di mana pelanggan melakukan pengujian terhadap prototipe yang ada dan pengembang memperhalus analisis kebutuhan pemakai.

e) Tahap Pembuatan dan Implementasi: tahap ini termasuk proses desain (rancang), pengkodean dan testing.

Keunggulan model ini adalah sifatnya yang sangat interaktif sehingga pengembang dan pengguna (pemakai)

dapat terus berinteraksi selama pengerjaan tahapan-tahapan tersebut. Peran aktif pemakai ini dapat menghemat waktu dalam pengembangan sistem dan bila terdapat kesalahan atau ketidaksesuaian keinginan, pemakai dapat segera memberitahukannya sehingga pengembang dapat secepatnya melakukan penyesuaian.

Kelemahan model ini antara lain, akibat adanya quick design, kadang pemakai tidak menyadari bahwa perangkat lunak yang ditunjukkan masih berupa blue print sehingga tidak ada jaminan terhadap kualitas secara keseluruhan dan pemeliharaan jangka panjangnya. Dari sisi pengembang, karena ingin menyegerakan selesainya proyek, sering menggunakan bahasa pemrograman yang sederhana dalam membuat prototipe tanpa memikirkan lebih lanjut

program yang lebih kompleks untuk membangun sistem yang sebenarnya.

Model Prototyping ini sangat sesuai diterapkan untuk kondisi yang beresiko tinggi di mana masalah-masalah tidak terstruktur dengan baik, terdapat fluktuasi kebutuhan pemakai yang berubah dari waktu ke waktu atau yang tidak terduga, bila interaksi dengan pemakai menjadi syarat mutlak dan waktu yang tersedia sangat terbatas sehingga butuh penyelesaian yang segera. Model ini juga dapat berjalan dengan maksimal pada situasi di mana sistem yang diharapkan adalah yang inovatif dan mutakhir sementara tahap penggunaan sistemnya relatif singkat.

3. Model RAD (Rapid Aplication Development) ?

RAD adalah proses pembangunan Perangkat Lunak yang menekankan pada siklus pengembangan yang pendek dan singkat. Model ini mengawinkan model

waterfall dan model component based construction.

Secara ringkas, tahapan-tahapan RAD adalah sebagai berikut.

a) Tahap Pemodelan Bisnis: dibuat agar dapat menjawab pertanyaan-pertanyaan berikut: informasi apa yang mengontrol proses bisnis? Informasi apa yang didapat? Siapa yang mendapatkannya? Untuk siapa informasi itu ditujukan? Siapa yang akan memprosesnya?
b) Tahap Pemodelan Data: informasi-informasi yang dipadu dari pemodelan bisnis dipilah-pilah ke menjadi sekumpulan objek data yang masing-masing objek diidentifikasikan dan ditentukan hubungan antara objek-objek tersebut.
c) Tahap Pemodelan Proses: aliran informasi yang didapat dalam proses

pemodelan data diolah sedemikian untuk dapat menopang fungsi-fungsi bisnis. Prosesnya dikreasikan untuk menambah, memodifikasi, menghapus dan atau mendapatkan kembali sebuah objek data.

d) Tahap Pembuatan Aplikasi: RAD dapat saja memakai kembali komponen program yang sudah ada bila dimungkinkan, atau membuat komponen yang dapat digunakan lagi bila diperlukan di masa mendatang. RAD juga diasumsikan menggunakan teknik generasi keempat (4GT).

e) Tahap Pengujian dan Pergantian: Proses RAD menekankan pada pemakaian kembali yang memungkinkan berkurangnya keseluruhan waktu pengujian, namun komponen harus diuji dan harus dilatih secara penuh dan terintegrasi.

Kelebihan model RAD: tahap-tahap RAD membuatnya mampu untuk menggunakan kembali komponen yang ada (reusable object), karena setiap komponen software dikerjakan secara terpisah dengan tim-tim tersendiri sehingga dapat digunakan juga untuk aplikasi lain yang pada akhirnya akan menghemat waktu. Penggunaan tim yang terpisah untuk mengerjakan pekerjaan yang berbeda membuat pekerjaan lebih cepat dalam proses integrasi dan efisien terhadap waktu tanpa mengacaukan aplikasi.

Kelemahan model RAD: Tidak begitu cocok untuk proyek dengan skala besar karena dibutuhkan sumber daya manusia yang semakin banyak seiring dengan semakin banyaknya komponen yang dikerjakan, selain itu, semakin besar proyek, semakin kompleks pula koordinasi yang dibutuhkan. Dalam

waktu yang singkat, rasanya sulit untuk pengembang dan pemakai berkomitmen untuk melaksanakan berbagai kegiatan untuk melengkapi sistem. Apalagi bila sistem ternyata tidak dapat dimodularisasi sementara sistem mempunyai resiko teknik yang tinggi.

Model RAD sangat tepat diterapkan untuk sistem yang telah jelas dan lengkap kebutuhannya, di mana terdapat komponen-komponen yang dapat dipakai kembali dalam proyek yang berskala kecil dengan waktu pengembangan perangkat lunak yang singkat.

4. Spiral Model/Spiral Boehm

Model ini mengadaptasi dua model perangkat lunak yang ada yaitu model prototyping dengan pengulangannya dan model waterfall dengan pengendalian dan sistematikanya.

Model ini dikenal dengan sebutan Spiral Boehm. Pengembang dalam model ini

memadupadankan beberapa model umum tersebut untuk menghasilkan produk khusus atau untuk menjawab persoalan-persoalan tertentu selama proses pengerjaan proyek.

Tahap-tahap model ini dapat dijelaskan secara ringkas sebagai berikut.

a) Tahap Liason: pada tahap ini dibangun komunikasi yang baik dengan calon pengguna/pemakai.
b) Tahap Planning (perencanaan): pada tahap ini ditentukan sumber-sumber informasi, batas waktu dan informasi-informasi yang dapat menjelaskan proyek.
c) Tahap Analisis Resiko: mendefinisikan resiko, menentukan apa saja yang menjadi resiko baik teknis maupun manajemen.

d) Tahap Rekayasa (engineering): pembuatan prototipe

e) Tahap Konstruksi dan Pelepasan (release): pada tahap ini dilakukan pembangunan perangkat lunak yang dimaksud, diuji, diinstal dan diberikan sokongan-sokongan tambahan untuk keberhasilan proyek.

f) Tahap Evaluasi: Pelanggan/pemakai/pengguna biasanya memberikan masukan berdasarkan hasil yang didapat dari tahap engineering dan instalasi.

Kelebihan model ini adalah sangat mempertimbangkan resiko kemungkinan munculnya kesalahan sehingga sangat dapat diandalkan untuk pengembangan perangkat lunak skala besar. Pendekatan model ini dilakukan melalui tahapan-tahapan yang sangat baik dengan menggabungkan model waterfall ditambah dengan

pengulangan-pengulangan sehingga lebih realistis untuk mencerminkan keadaan sebenarnya. Baik pengembang maupun pemakai dapat cepat mengetahui letak kekurangan dan kesalahan dari sistem karena proses-prosesnya dapat diamati dengan baik.

Kekurangan model ini adalah waktu yang dibutuhkan untuk mengembangkan perangkat lunak cukup panjang demikian juga biaya yang besar. Selain itu, sangat tergantung kepada tenaga ahli yang dapat memperkirakan resiko. Terdapat pula kesulitan untuk mengontrol proses. Sampai saat ini, karena masih relatif baru, belum ada bukti apakah metode ini cukup handal untuk diterapkan.

Model Boehm sangat cocok diterapkan untuk pengembangan sistem dan perangkat lunak skala besar di mana pengembang dan pemakai dapat lebih

mudah memahami kondisi pada setiap tahapan dan bereaksi terhadap kemungkinan terjadinya kesalahan. Selain itu, diharapkan juga waktu dan dana yang tersedia cukup memadai.

5. MODEL 4GT

Istilah Fourth Generation Techniques (4GT) mencakup seperangkat peralatan perangkat lunak yang berfungsi sebagai perangkat bantu yang memudahkan seorang pengembang software mengaplikasi beberapa karakteristik software pada tingkat yang tinggi, yang akan menghasilkan source code dan object code secara otomatis sesuai dengan spesifikasi (persyaratan khusus) yang dibuat oleh sang pengembang perangkat lunak.

Dewasa ini, 4GT tools dipakai sebagai 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.

Tahapan-tahapan model 4GT dapat diringkas sebagai berikut.

a) Tahap Pengumpulan Kebutuhan: tahap ini dimulai dengan mengumpulkan serangkaian kebutuhan yang nantinya akan diterjemahkan ke dalam prototipe. Namun, apabila pelanggan tidak yakin dengan apa yang diperlukan dan fakta-fakta tidak jelas diketahui maka prototipe tidak dapat dikerjakan oleh peralatan 4GT.

b) Tahap Merancang Strategi: tahap ini dibutuhkan untuk proyek besar yakni dengan menterjemahkan kebutuhan menjadi prototipe operasional agar tidak timbul masalah yang sama jika dibuat

dengan model konvensional. Namun, untuk proyek skala kecil tahap ini dapat dihilangkan dengan langsung melakukan implementasi dengan menggunakan bahasa generasi keempat (4GT).

c) Tahap Implementasi Menggunakan Bahasa Keempat: untuk skala kecil tahap ini dapat langsung dilakukan ketika kebutuhan telah jelas, dan untuk proyek besar tahapan ini dijalankan setelah dirancang prototipe operasional. Implementasi yang menggunakan 4GT memudahkan pengembang software untuk menjelaskan hasil yang diharapkan yang nantinya akan diterjemahkan ke dalam bentuk kode sumber dan kode objek.

d) Tahap Produksi: Tahap ini merupakan langkah terakhir yakni mengubah implementasi 4GT ke dalam hasil akhir berupa produk.

Kelebihan model ini adalah pengurangan waktu dan peningkatan produktivitas yang besar.

Kekurangan model ini adalah kemungkinan akan sulit memanfaatkan alat bantu/peralatan/tools 4GT dibandingkan dengan menggunakan bahasa pemrograman yang konvensional, selain itu terdapat juga masalah dalam hal kode sumber yang tidak efisien. Di samping itu, pemeliharaan sistem software besar yang dikembangkan oleh 4GT juga masih sedang dalam proses pengkajian.

Model ini diaplikasikan untuk mengembangkan perangkat lunak yang memakai bentuk bahasa khusus atau notasi grafik yang dieksekusi/diselesaikan dengan syarat atau ketentuan yang dipahami oleh pemakai/pengguna/kustomer.

BAB III
PENUTUP

A. Kesimpulan

Proses pengembangan perangkat lunak (Software Process / Development Paradigm) adalah sekumpulan tahap, tugas dan aktivitas yang dibutuhkan untuk secara effisien mentransformasikan kebutuhan pemakai ke suatu solusi perangkat lunak yang efektif.

Pemodelan proses perangkat lunak (Software Process Modeling) bertujuan untuk merepresentasikan aktivitas yang terjadi selama pembuatan perangkat lunak dan perubahan-perubahannya (evolusi).

Model-Model Proses Pengembangan Perangkat Lunak.

1) Model air terjun (Model Waterfall) ?
2) Model Prototyping ?
3) Model RAD (Rapid Aplication Development) ?
4) Spiral Model/Spiral Boehm
5) Model 4GT ?

B. Pesan

Jangan pernah bangga atas ilmu yang anda dapatkan sekarang, terus belajar dan cari ilmu sebanyak mungkin, penulis berharap pembaca bisa mengembangkan makalah ini.

DAFTAR PUSTAKA

A.S Rosa,Shalahuddin.M.2009.”Modul Rekayasa Perangkat Lunak (Terstruktur dan Berorientasi Objek).Bandung:Modula Bandung.

Davor Gornik , IBM Rational Unified Process : Best Practices for Software Development Teams.

www.agilemodelling.com

http://afridatriana.blogspot.com/2012/10/pemodelan-proses-pengembangan-perangkat.html

http://more-examples.blogspot.com/2012/11/macam-macam-model-proses-perangkat.html