**METODE PENGEMBANGAN PERANGKAT LUNAK**

(Tugas : Rekayasa Perangkat Lunak, pengampu : Bp. Aryawan Agung Nugorho, ST)

Disusun Oleh :

Kinanti hayuningtyas (10105241032)

Elsa Sugiarti (10105241008)

Wahyu Aji Prayogo (10105241029)

Edo Ihzandy (10105241002)

Priyo Dwi Prayogo (101052410)

METODE PENGEMBANGAN PERANGKAT LUNAK

Pengembangan perangkat lunak dapat diartikan sebagai proses membuat suatu perangkat lunak baru untuk menggantikan perangkat lunak lama secara keseluruhan atau memperbaiki perangkat lunak yang telah ada. Agar lebih cepat dan tepat dalam mendeskripsikan solusi dan mengembangkan perangkat lunak, juga hasilnya mudah dikembangkan dan dipelihara, maka pengembangan perangkat lunak memerlukan suatu metodologi khusus. Metodologi pengembangan perangkat lunak adalah suatu proses pengorganisasian kumpulan metode dan konvensi notasi yang telah didefinisikan untuk mengembangkan perangkat lunak. Secara prinsip bertujuan untuk membantu menghasilkan perangkat lunak yang berkualitas (Jaidan Jauhari .pdf). Berikut batu landasan yang menopang rekayasa perangkat lunak (Rogers S. Pressman, 2002:28):

Gb.1

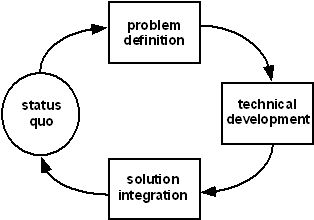
Metodologi pengembangan perangkat lunak (atau disebut juga model proses atau paradigma rekayasa perangkat lunak) adalah suatu strategi pengembangan yang memadukan proses, metode, dan perangkat (*tools*) (Modul RPL.pdf).Metode-metode rekayasa perangkat lunak, memberikan teknik untuk membangun perangkat lunak. Berkaitan dengan serangkaian tugas yang luas yang menyangkut analisis kebutuhan, konstruksi program, desain, pengujian, dan pemeliharaan (Rogers S. Pressman:2002).

Untuk menyelesaikan masalah di dalam pengembangan perangkat lunak, tim perekayasa harus menggabungkan strategi pengembangan yang melingkupi lapisan proses, metode, dan alat bantu. Model proses rekayasa perangkat lunak dipilih berdasarkan sifat aplikasi dan proyeknya, metode dan alat-alat bantu yang akan dipakai, dan control serta penyampaian yang dibutuhkan.

Berikut Metode-Metode Pengembangan Perangkat Lunak (Model Proses Pengembangan Perangkat Lunak) yaitu metode sekuensial linier, prototype, RAD, Fourth Generation Techniques (4GT) dan spiral (dalam Jaidan Jauhari.Pdf). Selain metode-metode tersebut Roger S. Pressman (2002:45), mengungkapkan dua metode lagi yaitu metode Proses Perangkat Lunak Evolusioner (pertambahan, spiral, konkruen), dan metode formal. Berikut Metode-metode tersebut :

1. **Model Sekuensial Linier**

Sering juga disebut dengan “siklus kehidupan klasik” atau “model air terjun.” Penggambaran model ini :



Gb. Fase lingkaran pemecahan masalah

tes

kode

desain

analisis

Gb. Model sekuensial linier

Metode pengembangan perangkat lunak dengan pendekatan pada perkembangan perangkat lunak yang sistematik dan sekuensial yang mulai pada tingkat dan kemajuan system pada seluruh analisis, desain, kode, pengujian, dan pemeliharaan. Model ini melingkupi aktivitas-aktivitas :

1. Rekayasa dan pemodelan system/informasi

Karena perangkat lunak adalah bagian dari sistem yang lebih besar, pekerjaan dimulai dari pembentukan kebutuhan-kebutuhan untuk seluruh elemen sistem dan kemudian memilah mana yang untuk pengembangan perangkat lunak. Hal ini penting, ketika perangkat lunak harus berkomunikasi dengan *hardware*, orang dan basis data

1. Analisis kebutuhan perangkat lunak

Pengumpulan kebutuhan dengan fokus pada perangkat lunak, yang meliputi : Domain informasi, fungsi yang dibutuhkan, unjuk kerja/performansi dan antarmuka. Hasilnya harus didokumentasi dan di*review* ke pelanggan

1. Desain

Ada 4 atribut untuk program yaitu : Struktur Data, Arsitektur perangkat lunak, Prosedur detil dan Karakteristik Antarmuka. Proses desain mengubah kebutuhan-kebutuhan menjadi bentuk karakteristik yang dimengerti perangkat lunak sebelum dimulai penulisan program. Desain ini harus terdokumentasi dengan baik dan menjadi bagian konfigurasi perangkat lunak.

1. Generasi kode

Penterjemahan perancangan ke bentuk yang dapat dimengerti oleh mesin, dengan menggunakan bahasa pemrograman

1. Pengujian

Setelah kode program selesai testing dapat dilakukan. Testing memfokuskan pada logika internal dari perangkat lunak, fungsi eksternal dan mencari segala kemungkinan kesalahan dan memriksa apakah sesuai dengan hasil yang diinginkan.

1. Pemeliharaan

Merupakan bagian paling akhir dari siklus pengembangan dan dilakukan setelah perangkat lunak dipergunakan. Pemeliharaan perangkat lunak mengaplikasikan lagi Setiap fase program sebelumnya dan tidak membuat yang baru. Kegiatan :

• *Corrective Maintenance* : Mengoreksi kesalahan pada perangkat lunak, yang baru terdeteksi pada saat perangkat lunak dipergunakan.

• *Adaptive Maintenance* : Penyesuaian dengan lingkungan baru, misalnya sistem operasi atau sebagai tuntutan atas perkembangan sistem komputer, misalnya penambahan *printer driver*

• *Perfektive Maintenance* : Bila perangkat lunak sukses dipergunakan oleh pemakai. Pemeliharaan ditujukan untuk menambah kemampuannya seperti memberikan fungsi-fungsi tambahan, peningkatan kinerja dan sebagainya

Model sekuensial adalah paradigm rekayasa perangkat lunak yang paling luas dipakai dan paling tua. Kelemahan model ini antara lain :

* Proyek yang sebenarnya jarang mengikuti alur sekuensial seperti diusulkan, sehingga perubahan yang terjadi dapat menyebabkan hasil yang sudah didapat tim harus diubah kembali/iterasi sering menyebabkan masalah baru.
* Linear sequential model mengharuskan semua kebutuhan pemakai sudah dinyatakan secara eksplisit di awal proses, tetapi kadang-kadang ini tidak dapat terlaksana karena kesulitan yang dialami pemakai saat akan mengungkapkan semua kebutuhannya tersebut.
* Pemakai harus bersabar karena versi dari program tidak akan didapat sampai akhir rentang waktu proyek.
* Adanya waktu menganggur bagi pengembang, karena harus menunggu anggota tim proyek lainnya menuntaskan pekerjaannya.

1. **Model Prototype**

Mendengarkan pelanggan

Uji pelanggan-mengendalikan market

Membangun, memperbaiki market

Gb. Prototype paradigma

Model ini dimulai dengan pengumpulan kebutuhan. Pendekatan *prototyping* model digunakan jika pemakai hanya mendefenisikan objektif umum dari perangkat lunak tanpa merinci kebutuhan *input*, pemrosesan dan *output*nya, sementara pengembang tidak begitu yakin akan efisiensi algoritma, adaptasi sistem operasi, atau bentuk antarmuka manusia-mesin yang harus diambil. Cakupan aktivitas dari *prototyping* model terdiri dari :

1. Mendefinisikan objektif secara keseluruhan dan mengidentifikasi kebutuhan yang sudah diketahui.
2. Melakukan perancangan secara cepat sebagai dasar untuk membuat prototype.
3. Menguji coba dan mengevaluasi prototype dan kemudian melakukan penambahan dan perbaikan-perbaikan terhadap prototype yang sudah dibuat.

secara ideal prototype berfungsi sebagai sebuah mekanisme untuk mengidentifikasi kebutuhan perangkat lunak. Bila prototype yang sedang bekerja dibangun, pengembang harus menggunakan fragmen-fragmen program yang ada atau mengaplikasikan alat-alat bantu (contoh: window manager, dsb) yang memungkinkan program yang bekerja agar dimunculkan secara cepat.

Kelemahan *prototyping* model :

* 1. Pelanggan yang melihat working version dari model yang dimintanya tidak menyadari, bahwa mungkin saja prototype dibuat terburu-buru dan rancangan tidak tersusun dengan baik
  2. Pengembang kadang-kadang membuat implementasi sembarang, karena ingin working version bekerja dengan cepat.

1. **Model RAD (*Rapid Application Development*)**

Merupakan model proses pengembangan perangkat lunak secara linear sequential yang menekankan pada siklus pengembangan yang sangat singkat/pendek. Jika kebutuhan dipahami dengan baik, proses RAD memungkinkan tim pengembangan menciptakan “sistem fungsional yang utuh” dalam periode waktu yang sangat pendek (kira-kira 60-90 hari). Pendekatan RAD model menekankan cakupan :

1. Pemodelan bisnis (*Bussiness Modelling*)

Aliran informasi diantara fungsi-fungsi bisnis dimodelkan dengan suatu cara untuk menjawab pertanyaan-pertanyaan berikut : Informasi apa yang mengendalikan proses bisnis ? Kemana informasi itu pergi? Siapa yang memprosesnya ?

1. Pemodelan data (*Data Modelling*)

Aliran informasi yang didefinisikan sebagai bagian dari fase pemodelan bisnis disaring ke dalam serangkaian objek data yang dibutuhkan untuk menopang bisnis tersebut. Karakteristik/atribut dari masing-masing objek diidentifikasi dan hubungan antara objek-objek tersebut didefinisikan.

1. Pemodelan proses (*Process Modelling*)

Aliran informasi yang didefinisikan dalam fase pemodelan data 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.

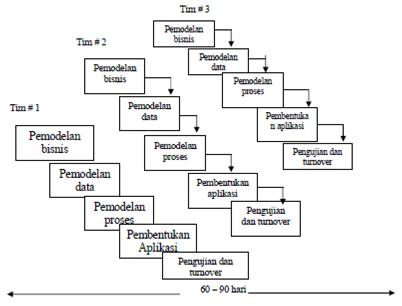
1. Pembuatan aplikasi (*Application generation*)

Selain menciftakan perangkat lunak dengan menggunakan bahasa pemrograman generasi ketiga yang konvensional, RAD lebih banyak memproses kerja untuk memakai lagi komponen program yang telah ada atau menciftakan komponen yang bias dipakai lagi. Pada semua kasus, alat-alat Bantu otomatis dipakai untuk memfasilitasi kontruksi perangkat lunak.

1. Pengujian dan pergantian (*Testing and turnover*)

Karena proses RAD menekankan pada pemakaian kembali, banyak komponen yang telah diuji. Hal ini mengurangi keseluruhan waktu pengujian. Tapi komponen baru harus diuji.

Gb. Model RAD



Kelemahan model RAD :

* 1. Untuk proyek dengan skala besar, RAD membutuhkan sumber daya manusia yang cukup untuk membentuk sejumlah tim RAD yang baik.
  2. RAD membutuhkan pengembang dan pemakai yang mempunyai komitmen dalam aktivitas *rapid-fire* untuk melaksanakan aktivitas melengkapi sistem dalam kerangka waktu yang singkat. Jika komitmen tersebut tidak ada, proyek RAD gagal.

Tidak semua aplikasi sesuai untuk RAD.bila system tidak dapat dimodulkan dengan teratur, pembangunan komponen penting pada RAD akan menjadi sangat problematic. RAD menjadi tidak sesuai jika risiko teknisnya tinggi.

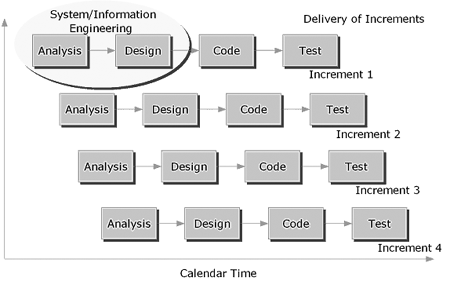
1. **Model Proses Perangkat Lunak Evolusioner**

Model evolusioner adalah model iterative, ditandai dengan tingkah laku yang memungkinkan perekayasa perangkat lunak mengembangkan versi perangkat lunak yang lebih lengkap sedikit demi sedikit. Kebutuhan produk dan bisnis kadang-kadang berubah seiring dengan laju perkembanganya. Dalam situasi tersebut maupun lainya, perekayasa perangkat lunak membutuhkan sebuah model proses yang sudah dirancang secara eksplisit untuk mengakomodasi produk perkembangan sepanjang waktu. Model ini bukan termasuk rekayasa perangkat lunak klasik. Model evolusioner meliputi :

1. Model pertambahan

Model incremental menggabungkan elemen-elemen model sekuensial linier (diaplikasikan secara berulang) dengan filosofi prototype iterative. Model ini memakai urutan-urutan linier di dalam model yang membingungkan, seiring dengan laju waktu kalender. Setiap urutan linier menghasilkan pertambahan, perangkat lunak “yang bisa disampaikan.” Contoh, perangkat lunak pengolah kata yang dikembangkan dengan menggunakan paradigm pertambahan akan menyampaikan manajemen file, editing, serta fungsi penghasilan dokumen pada pertambahan pertama, dan selanjutnya. Pertambahan pertama dapat disebut sebagai produk inti (*core product*).

Model ini berfokus pada penyampaian produk operasional dalam Setiap pertambahanya. Pertambahan awal ada di versi *stripped down* dari produk akhir, tetapi memberikan kemampuan untuk melayani pemakai dan juga menyediakan platform untuk evaluasi oleh pemakai.



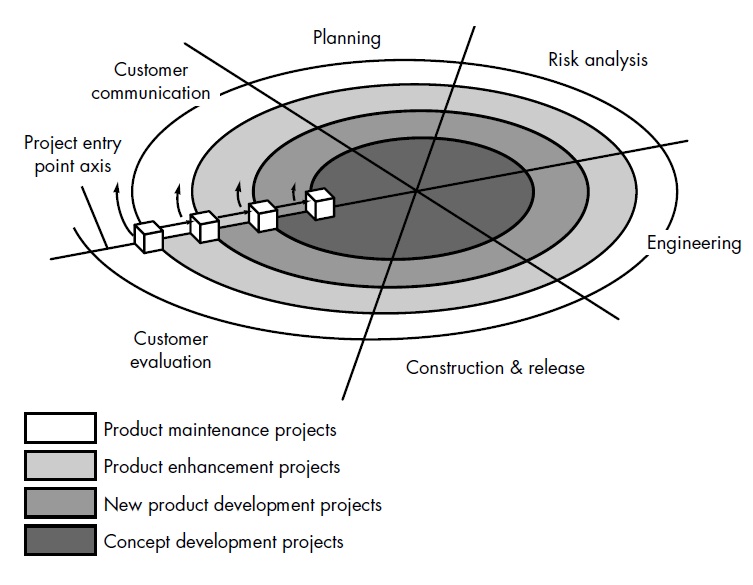
Gb. Model Incremental

Perkembangan pertambahan, khususnya berguna pada staffing, tidak bisa dilakukan menggunakan implementasi lengkap oleh batas waktu bisnisyang sudah disepakati untuk proyek tersebut. jika produk inti diterima dengan baik, maka staf tambahan bisa ditambahkan untuk mengimplementasi pertambahan selanjutnya.

1. Model spiral

Awalnya diusulkan oleh Boehm (BOE88), adalah model proses perangkat lunak yang evolusioner, merangkai sifat iterative dari prototype dengan cara control dan aspek sistematis dari model sekuensial linier. Model yang berpotensi untuk pengembangan versi pertambahan perangkat lunak secara cepat. Model spiral dibagi menjadi sejumlah aktifitas kerangka kerja atau wilayah tugas, antara lain :

* Komunikasi pelanggan, tugas-tugas yang dibutuhkan untuk membangun komunikasi yang efektif diantara pengembang dan pelanggan.
* Perencanaan, tugas-tugas yang dibutuhkan untuk mendefinisikan sumber-sumber daya, ketepatan waktu, dan proyek informasi lain yang berhubungan.
* Analisis resiko, tugas-tugas yang dibutuhkan untuk menaksir resiko-resiko, baik manajemen maupun teknis.
* Perekayasaan, tugas-tugas yang dibutuhkan untuk membangun satu atau lebih representasi dari aplikasi tersebut.
* Konstruksi dan peluncuran, tugas-tugas yang dibutuhkan untuk mengkonstruksi, menguji, memasang (instal), dan memberikan pelayanan kepada pemakai (contohnya pelatihan dan dokumentasi)
* Evaluasi pelanggan, tugas-tugas untuk memperoleh umpan balik dari pelanggan dengan didasarkan pada evaluasi representasi perangkat lunak, yang dibuat selama masa perekayasaan, dan dimplementasikan selama masa pemasangan.



Model spiral menjadi pendekatan yang realistis bagi perkembangan system dan perangkat lunak skala besar. Karena perangkat lunak terus bekerja selama proses bergerak, pengembang dan pemakai memahami, dan bereaksi lebih baik terhadap resiko dari Setiap tingkat evolusi. Model spiral menggunakan prototype sebagai mekanisme pengurangan resiko.

Model spiral membutuhkan keahlian penafsiran resiko yang masuk akal, dan sangat bertumpu pada keahlian ini untuk mencapai keberhasilan. Jika sebuah resiko tidak dapat ditemukan dan diatur, pasti akan terjadi masalah. Model ini membutuhkan waktu bertahun-tahun sampai kehandalannya bisa dipertimbangkan dengan kepastian absolute.

1. Model rakitan komponen

Model ini menggabungkan beberapa karakteristik model spiral. Bersifat evolusioner, sehingga membutuhkan pendekatan iterative untuk menciptakan perangkat lunak. Tetapi model ini merangkai aplikasi dari komponen perangkat lunak sebelum dipaketkan (kadang disebut kelas).

Aktivitas perangkat lunak dimulai dengan identifikasi calon kelas. Dipenuhi dengan mengamati data yang akan dimanipulasi oleh aplikasi dan algoritma-algoritma yang akan diaplikasikan. Data dan algoritma yang berhubungan dikemas ke dalam kelas. Kelas-kelas tersebut disimpan dalam *class library* (tempat penyimpanan).

Model ini membawa pada penggunaan kembali perangkat lunak, dan kegunaan kembali itu memberi sejumlah keuntungan yang bisa diukur pada rekayasa perangkat lunak.

1. Model perkembangan konkruen

Representasi aktivitas dalam model ini, meliputi aktivitas analisis, bisa berada dalam salah satu dari keadaan-keadaan yang dicatat pada saat tertentu. Dengan cara yang sama, aktivitas yang lain (desain atau komunikasi pelanggan) dapat direpresentasikan dalam sebuah sikap yang analog. Semua aktifitas ada secara konkruen tetapi dia tinggal didalam keadaan yang berbeda. Model ini sering digunakan sebagai paradigm bagi pengembangan aplikasi klien/server.

Kenyataanya model proses konkruen bisa diaplikasikan ke dalam semua tipe perkembangan perangkat lunak, dan memberikan gambaran akurat mengenai keadaan tertentu dari sebuah proyek. Selain membatasi ajtivitas perekayasa ke dalam sederetan kejadian, model proses juga mendefinisikan jaringan aktivitas.

1. **Model Formal**

Model metode formal mencangkup sekumpulan aktivitas yang membawa kepada spesifikasi matematis perangkat lunak computer. Metode ini memungkinkan perekayasa perangkat lunak untuk mengkhususkan, mengembangkan, dan memverifikasi system berbasis computer dengan menggunakan notasi matematis yang tepat.variasi dalam pendekatan ini, disebut clean-room rekayasa perangkat lunak, sedang diaplikasikan oleh banyak organisasi pengembang perangkat lunak.

Bila metode formal dipakai selama masa pengembangan, maka akan memberikan mekanisme untuk mengeliminasi banyak masalah yang sulit dipecahkan menggunakan paradigm perangkat lunak yang lain. Ambiguitas, ketidaklengkapan, dan ketidak-konsitenan bisa ditemukan dan diperbaiki secara mudah, tidak melalui kajian *ad hoc* tetapi melalui aplikasi analisis matematis. Jika metode ini dipakai selama proses perancangan, maka berfungsi sebagai dasar bagi verifikasi program sehingga memungkinkan perekayasa untuk menemukan dan memperbaiki kesalahan yang mungkin saja tidak terdeteksi.

Metode formal akan banyak memperoleh penganut diantara pengembang perangkat lunak yang harus membangun perangkat lunak yang kritis untuk keselamatan (missal : pengembangan perangkat medis, dan penerbangan pesawat), serta diantara yang harus menderita karena faktor ekonomis yang harus dialami oleh perangkat lunak.

1. **Fourth Generation Techniques (4GT)**

Istilah generasi ke empat, mengarah ke perangkat lunak yang umum yaitu tiap pengembang perangkat lunak menentukan beberapa karakteristik perangkat lunak pada level tinggi. Saat ini pengembangan perangkat lunak yang mendukung 4GT, berisi tool-tool berikut : i) Bahasa non prosedural untuk query basis data; ii) *Report generation; iii)* *Data manipulation ; iv)* Interaksi layar ; v) Kemampuan grafik level tinggi ; vi) Kemampuan spreadsheet . Tiap *tool* ini ada tapi hanya untuk sauatu aplikasi khusus.

Menggunakan perangkat bantu (*tools*) yang akan membuat kode sumber secara otomatis berdasarkan spesifikasi dari pengembang perangkat lunak. Hanya digunakan untuk menggunakan perangkat lunak yang menggunakan bahasa khusus atau notasi grafik yang diselesaikan dengan syarat yang dimengerti pemakai. Cakupan aktivitas 4GT :

1. Pengumpulan kebutuhan, idealnya pelanggan akan menjelaskan kebutuhan yang akan ditranslasikan ke prototype operasional.

2. Translasi kebutuhan menjadi prototype operasional, atau langsung melakukan implementasi secara langsung dengan menggunakan bahasa generasi keempat (4GL) jika aplikasi relatif kecil.

3. Untuk aplikasi yang cukup besar, dibutuhkan strategi perancangan sistem walaupun 4GL akan digunakan.

4. Pengujian.

5. Membuat dokumentasi.

6. Melaksanakan seluruh aktivitas untuk mengintegrasikan solusi-solusi yang membutuhkan paradigma rekayasa perangkat lunak lainnya.

Salah satu keuntungan penggunaan model 4GT adalah pengurangan waktu dan peningkatan produktivitas secara besar, sementara kekurangannya terletak pada kesulitan penggunaan perangkat bantu (tools) dibandingkan dengan bahsa pemrograman, dan juga kode sumber yang dihasilkannya tidak efisien.

Untuk aplikasi yang yang kecil, adalah mungkin untuk langsung berpindah dari pengumpulan kebutuhan ke implementasi dengan menggunakan 4GL. Tapi untuk aplikasi yang besar, dibutuhkan pengembangan strategi desain untuk sistem, walau digunakan 4GL. Penggunaan 4GT tanpa perencanaan yang matang (untuk proyek skala besar) akan meyebabkan kesulitan yang sama (kualitas dan pemeliharaan yang jelek, ketidakpuasan pelanggan) seperti dengan metode konvensional.

**DAFTAR PUSTAKA**

Jauhari, Jaidan. \_\_\_.“Modul Rekayasa Perangkat Lunak.”\_\_:\_\_.pdf. http://heckerlaye.files.wordpress.com/2009/11/modul-rekayasa-perangkat-lunak.pdf

Pressman, Roger S. 2002.”Rekayasa Perangkat Lunak (Pendekatan Praktis).” Yogyakarta : Andi.