BAB 1

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*) . 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.

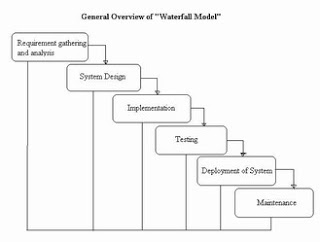
Berikut Metode-Metode Pengembangan Perangkat Lunak (Model Proses Pengembangan Perangkat Lunak) yaitu metode Waterfall, prototype, RAD dan Model Sekuensial Linier (dalam Jaidan Jauhari.Pdf). Selain metode-metode tersebut Roger S. Pressman (2002:45), Berikut Metode-metode tersebut :

1. **Model Waterfall**

          Model Waterfall merupakan salah satu model klasik bersifat Sisetematis. Kenapa di sebut klasik??!! karena model ini dikerjakan secara berurutan. Pengunaan model ini dalam penerapan kehidupan sehari-hari sangatlah memakan waktu dan sangat sedikit di pakai dalam membuat software.

masalah yang terdapat pada model ini :

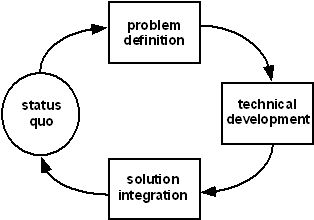
1. Perubahan suolit dilakukan karena sistemnya baku dan kaku.
2. Karena, sifatnya kaku dan baku maka kebutuhan harus di kumpulkan secara lengkap sehingga perubahan bisa ditekan seminimal mungkin.
3. Perlu diketahui penerapan model ini hanya untuk rekayasa sistem yang besar.

[](http://3.bp.blogspot.com/-aMd0SpEJbyA/TnVfw3W3T1I/AAAAAAAAAAg/I02TsonK49w/s1600/waterfall.jpg)

Gb. Metode waterfall

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.

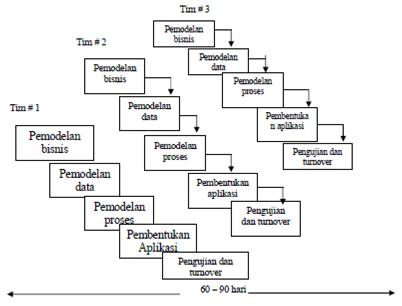
1. Pembuatan aplikasi (*Application generation*)

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.

BAB 2

PERBANDINGAN METODE PERANGKAT LUNAK DENGAN METODE TRADISIONAL

|  |  |
| --- | --- |
| Metode Tradisional | Metode RAD |
| * + 1. **Aspek**   Pada 1970an, Rapid Application Development muncul sebagai respon mengerikan untuk proses nontangkas, seperti model Waterfall. Pengembang perangkat lunak menghadapi masalah waktu dengan metodologi sebelumnya sebagai aplikasi begitu lama untuk membangun bahwa spesifikasi persyaratan diubah oleh sistem waktu itu selesai. Dengan demikian, metodologi tersebut sering mengakibatkan sistem tidak dapat digunakan. Dengan demikian, metodologi tersebut sering mengakibatkan sistem tidak dapat digunakan.   * + 1. **Tahap**   tahap tradisonal mengacu pada urutan tahaptahap  SDLC | * + 1. **Aspek**   pembangunan prototipe untuk tujuan membangkitkan kembali desain untuk kebutuhan pengguna. Tujuannya adalah untuk membangun sebuah fitur ringan yang hasil akhirnya dalam jumlah pendek dengan waktu yang memugkinkan. Prototipe awal berfungsi sebagai bukti konsep untuk klien, tetapi lebih penting berfungsi sebagai titik berbicara dan alat untuk kebutuhan pemurnian. Mengembangkan prototipe cepat dicapai dengan Computer Aided Engineering CASE tools Software yang berfokus pada menangkap persyaratan, mengkonversi mereka ke model data, mengubah model data ke database, dan  menghasilkan kode semua dalam satu alat.   * + 1. **Tahap**   Pada RAD Tahap pertama langsung membuat analisis dan design, lalu langsung ketahap siklus prototyping yaitu membangun, memperhalus dan mendemonstrasikannya. Itu akan mempercepat proses dalam pembuatan suatu project. RAD memang lebih cepat dari Waterfall. Jika kebutuhan dan batasan project sudah diketahui dengan baik. Juga jika proyek memungkinkan untuk dimodularisasi. |

Gambar 1 Perbandingan RAD dengan metode tradisional

|  |  |
| --- | --- |
| Metode tradisional | Metode Prototype |
| pengerjaan dari suatu sistem dilakukan secara berurutan atau secara linear. Jadi jika langkah satu belum dikerjakan maka tidak akan bisa melakukan pengerjaan langkah 2, 3 dan seterusnya.  Secara otomatis tahapan ke-3 akan bisa dilakukan jika tahap ke-1 dan ke-2 sudah dilakukan.   1. Tahapan   Pemilihan Fungsi., Penyusunan Sistem Informasi, Evaluasi. Penggunaan selanjutnya. | Metode ini sering digunakan pada dunia riil. Karena metode ini secara keseluruhan akan mengacu kepada kepuasan user. Bisa dikatakan bahwa metode ini merupakan metode waterfall yang dilakukan secara berulang-ulang.   1. Tahap   Analisa, Design, Code dan Testing, Penerapan dan Pemeliharaan. |