**B A B I**

**METODOLOGI PENGEMBANGAN PERANGKAT LUNAK**

* 1. **PENDAHULUAN**
		1. **Deskripsi Singkat**

Dalam bab ini dibahas tentang deskripsi umum mengenai perangkat lunak, rekayasa perangkat lunak, paradigma perangkat lunak, karakteristik perangkat lunak, komponen perangkat lunak, aplikasi perangkat lunak dan model proses perangkat lunak.

* + 1. **Manfaat dan Relevansi**

Pada bagian ini mahasiswa akan mempelajari tentang definisi perangkat lunak, definsi rekayasa perangkat lunak, pandangan umum rekayasa perangkat lunak, manfaat rekayasa perangkat lunak, model proses perangkat lunak, penggambaran model proses dan penggunaannya, penggambaran outline model proses perangkat lunak dan metode-metode perangkat lunak. Dengan dasar pemahaman ini akan menjadi landasan bagi mahasiswa dalam pengenalan terhadap rekayasa perangkat lunak, model proses perangkat lunak dan metodologi pengembangan perangkat lunak. Bagian akhir dari bab ini merupakan dasar bagi mahasiswa untuk mempelajari, mendalami serta memahami pentingnya rekayasa perangkat lunak dalam pembuatan sebuah aplikasi/program.

* + 1. **Kompetensi Khusus**

Setelah mempelajari materi ini, mahasiswa diharapkan mampu:

1. Menyimpulkan perbedaan perangkat lunak dengan rekayasa perangkat lunak
2. Menerapkan model proses perangkat lunak
3. Menjelaskan tujuan rekayasa perangkat lunak
	* 1. **Petunjuk Belajar**

Kegiatan pokok yang perlu dilakukan oleh mahasiswa agar mudah menguasai materi perkuliahan ini adalah:

1. Mengenal kompetensi khusus yang harus dicapai;
2. Membaca dan memahami bahan ajar yang disajikan baik dalam tulisan maupun perkuliahan;
3. Menjawab pertanyaan-pertanyaan yang terdapat pada akhir bab ini;
4. Menyelesaikan tes formatif.
	1. **PENYAJIAN**

Dalam penyajian materi ini, diawali dengan pemberian informasi umum tentang tujuan pembelajaran yang hendak dicapai. Selanjutnya dalam proses pembelajaran akan digunakan metode pembelajaran secara bergantian, yang meliputi: tanya jawab, diskusi, ceramah dan pemberian tugas.

Proses perangkat lunak telah menjadi perhatian yang serius selama decade terakhir. Tetapi apakah sebenarnya proses perangkat lunak itu? Proses perangkat lunak sebagai sebuah kerangka kerja untuk tugas-tugas yang dibutuhkan membangun perangkat lunak dengan kualitas yang baik.

Untuk membangun sebuah perangkat lunak yang baik, maka diharapkan pembangunan perangkat lunak dilakukan secara sistematis, sehingga dibuatlah paradigm-paradigma yang bervariasi untuk menangani pembangunan perangkat lunak yang berbeda-beda.

* **Perangkat Lunak**

Menurut Pressman (2002), perangkat lunak dapat didefinsikan sebagai berikut :

1. Perintah (program computer) yang bila dieksekusi memberikan fungsi dan unjuk kerja seperti yang diinginkan.
2. Struktur data yang memungkinkan program memanipulasi informasi secara proporsional.
3. Dokumen yang menggambarkan informasi dan kegunaan program
	* **Karakteristik Perangkat Lunak**

Untuk memperoleh pemahaman perangkat lunak (serta pemahaman tentang *software engineering),* penting juga untuk meneliti karakteristik perangkat lunak berbeda dari hal-hal lain yang dibangun manusia. Ketika perangkat lunak dibuat, proses kreatif manusia (analisis, desain, konstruksi, pengujian) diterjemahkan kedalam bentuk fisik. Jika kita membuat computer baru, sketsa dasar, penggambaran desain formal dan *prototype* berkembang kedalam suatu produk fisik.

Perangkat lunak lebih merupakan elemen logika dan bukan elemen fisik. Dengan demikian, perangkat lunak merupakan ciri yang berbeda dari perangkat keras :

1. Perangkat lunak dibangun dan dikembangkan, tidak dibuat dalam bentuk yang klasik

Meskipun banyak kesamaan diantara pabrik perangkat keras dan perangkat lunak, aktivitas keduanya secara mendasar sangat berbeda. Dalam keduanya tersebut, kualitas yang tinggi dicapai melalui perancangan yang baik, tetapi didalam fase pembuatan perangkat keras, selalu saja ditemukan masalah kualitas yang tidak mudah untuk disesuaikan dengan perangkat lunak. Kedua aktivitas itu tergantung pada manusia, tetapi hubungan antara penerapan yang dilakukan manusia dengan usaha yang diperoleh sangat berbeda. Kedua aktivitas itu membutuhkan konstruksi sebuah produk tetapi pendekatan yang dipakai berbeda. Biaya untuk perangkat lunak dikonsentrasikan pada pengembangan. Hal ini berarti proyek perangkat lunak tidak dapat diatur seperti pengaturan proyek pemanufakturan.

1. Perangkat lunak tidak pernah usang

 Perangkat lunak tidak rentan terhadap pengaruh lingkungan yang merusak sehingga menyebabkan perangkat keras menjadi using. Kesalahan-kesalahan yang tidak dapat ditemukan akan menyebabkan tingkat kegagalan menjadi sangat tinggi pada awal hidup program. Tetapi hal itu dapat diperbaiki dan diharapkan tidak lagi ditemukan kesalahan yang lain.

 Aspek lain dari keusangan menggambarkan perbedaan perangkat keras dengan perangkat lunak. Bila komponen suatu perangkat telah using, komponen dapat diganti suku cadangnya. Namun tidak ada suku cadang bagi perangkat lunak. Setiap kegagalan perangkat lunak menggambarkan kesalahan dalam perancangan atau proses dimana rancangan diterjemahkan kedalam kode mesin yang dapat dieksekusi. Demikianlan pemeliharaan perangkat lunak lebih kompleks daripada pemeliharaan perangkat keras.

1. Sebagian besar perangkat lunak dibuat dengan costum-built, serta tidak dapat dirakit dari komponen yang sudah ada

 Saat perangkat keras untuk produk berbasis mikroprosesor dirancang dan dibuat, pengembang desain menggambar sebuah skema sederhana dari rangkaian digital, melakukan serangkaian analisis dasar untuk memastikan bahwa fungsi yang tepat dicapai serta kemudian menyesuaikan ke katalog komponen digital. Setiap IC (chip) mempunyai nomor bagian tersendiri, sebuah fungsi yang sudah terdefinisi dan tervalidasi, interface yang terdefinsikan dengan baik, serta rangkaian standar tuntutan terintegrasi. Setelah masing-masing komponen diseleksi, perangkat keras dapat dipesan secara terpisah.

 Sayangnya para perancang perangkat lunak tidak diberi fasilitas seperti yang digambarkan diatas. Dengan sedikit pengecualian, tidak ada katalog komponen perangkat lunak. Memang memungkinkan untuk memesan perangkat lunak secara terpisah, tetapi tetap merupakan satu kesatuan yang lengkap, bukan sebagai komponen yang dapat dipasangkan kedalam program-program yang baru.

* **Pembagian Perangkat Lunak**

*Software* secara umum dapat di bagi dua yaitu :

*software* sistem dan *software* aplikasi.

*Software* sistem dapat di bagi lagi menjadi tiga macam yaitu :

**1. Bahasa pemrograman**

Bertugas mengkonversikan arsitektur dan algoritma yang di rancang manusia ke dalam format yang dapat di jalankan komputer, contoh bahasa pemrograman di antaranya : BASIC, COBOL, Pascal, C++, FORTRAN

**2. Sistem Operasi**

Saat komputer pertama kali di hidupkan, sistem operasilah yang pertama kali di jalankan, sistem operasi yang mengatur seluruh proses, menterjemahkan masukan, mengatur proses internal, memanejemen penggunaan memori dan DOS, Unix, Windows 95, IMB OS/2, Apple’s System 7

**3. Utility**

**Software** sistem dengan fungsi tertentu, misalnya pemeriksaan perangkat

keras (*hardware troubleshooting*), memeriksa disket yang rusak (bukan rusak fisik), mengatur ulang isi *harddisk* (partisi, defrag), contoh Utilty adalah Norton Utility.

*Software* aplikasi merupakan bagian *software* yang sangat banyak di jumpai dan terus berkembang. Sebelum tahun 1990-an aplikasi yang di kenal yaitu pemroses kata (Word Star, Chi Write), pemroses tabel (Lotus 123, Quatro Pro), database (DBASE), dan hiburan (game). Pada perkembangan pemroses kata, table dan database saat ini telah di bundel menjadi aplikasi *office* dengan tambahan aplikasi untuk pembuatan presentasi. Yang berkembang sangat banyak saat ini adalah aplikasi multimedia dan internet. Contoh aplikasi multimedia adalah Winamp untuk memutar musik berformat MP3 atau CD Audio, kemudian RealPlayer yang dapat digunakan untuk menonton film atau VCD. Aplikasi internet yang umum di gunakan adalah untuk browsing, e-mail, chatting dan messenger. Aplikasi yang bersifat khusus di antaranya untuk membantu pekerjaan Engineer seperti AutoCAD (gambar struktur), Protel (gambar rangkaian elektronik), dan Matlab (pemroses dan visualisasi persamaan matematis).

* **Rekayasa Perangkat Lunak**

Terdapat beberapa definisi rekayasa perangkat lunak, antara lain sebagai berikut

1. Aplikasi dari suatu pendekatan kuantifiabel, displin dan sistematis dalam pengembangan, operasi dan pemeliharaan sistem perangkat lunak

2. Studi tentang pedekatan-pendekatan dalam pengembangan, operasi dan pemeliharaan system perangkat lunak.

Pengertian Rekayasa Perangkat Lunak itu sendiri adalah Suatu disiplin ilmu yang membahas semua aspek produksi perangkat lunak, mulai dari tahap awal yaitu analisa kebutuhan pengguna, menentukan spesifikasi dari kebutuhan pengguna, disain, pengkodean, pengujian sampai pemeliharaan sistem setelah digunakan.

Jelaslah bahwa RPL tidak hanya berhubungan dengan cara pembuatan program komputer. Pernyataan “semua aspek produksi” pada pengertian di atas, mempunyai arti semua hal yang berhubungan dengan proses produksi seperti manajemen proyek, penentuan personil, anggaran biaya, metode, jadwal, kualitas sampai dengan pelatihan pengguna merupakan bagian dari RPL.

* **Tujuan Rekayasa Perangkat Lunak :**

a. Memperoleh biaya produksi perangkat lunak yang rendah.

b. Menghasilkan perangkat lunak yang kinerjanya tinggi, andal dan tepat waktu.

c. Menghasilkan perangkat lunak yang dapat bekerja pada berbagai jenis platform.

d. Menghasilkan perangkat lunak yang biaya perawatannya rendah.

* **Ruang Lingkup Rekayasa Perangkat Lunak**

Sesuai definisi yang telah disampaikan sebelumnya, maka ruang lingkup

RPL adalah sebagai berikut :

- Software requirements berhubungan dengan spesifikasi kebutuhan dan persyaratan perangkat lunak.

- Software design mencakup proses penentuan arsitektur, komponen, antarmuka, dan karakteristik lain dari perangkat lunak.

- Software construction berhubungan dengan detil pengembangan perangkat

lunak, termasuk algoritma, pengkodean, pengujian, dan pencarian kesalahan.

- Software testing meliputi pengujian pada keseluruhan perilaku perangkat lunak.

- Software maintenance mencakup upaya-upaya perawatan ketika perangkat lunak telah dioperasikan.

- Software configuration management berhubungan dengan usaha perubahan konfigurasi perangkat lunak untuk memenuhi kebutuhan tertentu.

- Software engineering management berkaitan dengan pengelolaan dan pengukuran RPL, termasuk perencanaan proyek perangkat lunak.

- Software engineering tools and methods mencakup kajian teoritis tentang alat bantu dan metode RPL.

- Software engineering process berhubungan dengan definisi, implementasi, pengukuran, pengelolaan, perubahan dan perbaikan proses RPL.

- Software quality menitikberatkan pada kualitas dan daur hidup perangkat

* Rekayasa Perangkat Lunak dan Pemecahan Masalah

Secara konsep, rekayasa perangkat lunak memiliki kedekatan dengan prinsip-prinsip pemecahan masalah. Pemahaman tentang masalah, strategi dan proses pemecahan masalah, serta pendekatan sistem pada pemecahan masalah akan sangat membantu proses rekayasa perangkat lunak.

* Masalah dan Gejala

Masalah adalah perbedaan antara kondisi yang terjadi dan kondisi yang diharapkan atau boleh juga diartikan sebagai perbedaan antara kondisi sekarang dengan tujuan yang diinginkan. Sebagai contoh seorang siswa berharap memperoleh nilai di atas 80 untuk ujian mata pelajaran Pemrograman C++, namun pada kenyataannya dia hanya memperoleh nilai 60. Adanya perbedaan ini menunjukkan adanya masalah.

Seringkali kita kesulitan membedakan antara gejala dan masalah. **Gejala** adalah tanda/petunjuk terjadinya suatu masalah. Perhatikan seorang yang berprofesi sebagai dokter. Seorang dokter dalam usaha mengobati penyakit pasien selalu bertanya dulu tentang gejala-gejala yang dirasakan pasien kemudian menyimpulkan bahwa pasien menderita penyakit tertentu dan menentukan obat yang tepat. Pusing, demam, batuk, dan pilek merupakan gejala atau tanda dari penyakit flu. Apabila dokter hanya member obat sakit kepala, maka penyakit flu tidak akan sembuh. Satu masalah mungkin memiliki satu gejala tetapi mungkin juga lebih. Perhatikan Gambar dibawah ini :



**Tipe-tipe Masalah**

Masalah dapat dikelompokkan seperti pada Gambar berikut :

Masalah pemenuhan standar

Tipe masalah dalam kelompok ini adalah masalah-masalah yang berhubungan dengan pencapaian standar yang telah ditentukan dalam sebuah organisasi. Biasanya tujuan seperti ini berlaku dalam jangka yang relative panjang.

* Masalah pemilihan alternative

Masalah dalam kelompok ini berhubungan dengan bagaimana memilih solusi terbaik dari berbagai alternative berdasarkan kriteria-kriteria tertentu. Permasalahan ini seringkali kita jumpai dalam kehidupan sehari-hari, seperti bagaimana memilih sekolah yang tepat, memilih lokasi tempat tinggal, memilih bidang pekerjaan. Masing-masing alternatif dan kriteria memiliki bobot yang telah disepakati.

- Masalah pemenuhan kepuasan konsumen

Pada organisasi-organisasi yang bersifat profit (mencari keuntungan), masalah-masalah pada kelompok ini merupakan tipe yang seringkali muncul. Konsumen memiliki berbagai macam keinginan yang satu sama lain berbeda. Memenuhi seluruh keinginan konsumen sangat tidak mungkin dan sangat memberatkan sebuah organisasi. Oleh karena itu perlu dicari pemecahan yang sama-sama menguntungkan, baik bagi konsumen maupun organisasi tersebut.

- Masalah pencapaian tujuan

Tipe ini mirip dengan tipe pertama (masalah pemenuhan standar). Yang berbeda adalah, pada tipe ini tujuan yang ingin dicapai dapat berubahubah dan bersifat jangka pendek.

* **Pemecahan Masalah**

**Pemecahan masalah** adalah sebuah proses dimana suatu situasi diamati kemudian bila ditemukan ada masalah dibuat penyelesaiannya dengan cara menentukan masalah, mengurangi atau menghilangkan masalah atau mencegah masalah tersebut terjadi. Ada banyak urutan proses pemecahan masalah yang diajukan oleh para ahli.

Pada gambar dibawah ini terlihat serangkaian tahapan proses yang berbeda yang dapat digunakan dalam berbagai tingkatan, tergantung dari tipe dan sifat masalahnya. Masalah yang berbeda membutuhkan penggunaan cara yang berbeda, bahkan mungkin urutan yang berbeda. Tahapan kritis dari proses pemecahan masalah adalah **Pendefinisian Masalah**. Apabila masalah tidak cukup jelas didefinisikan maka tahapan-tahapan berikut sulit untuk dijalankan. Bahkan apabila dipaksakan, kemungkinan besar penyelesaian yang tepat tidak akan diperoleh.

Secara umum proses pemecahan masalah dapat dilakukan dengan empat tahapan utama yaitu :

• Memahami dan mendefinisikan masalah

Bagian ini merupakan bagian yang sangat penting karena menjadi awal dari seluruh proses pemecahan masalah. Tujuan pada bagian ini adalah memahami masalah dengan baik dan menghilangkan bagian-bagian yang dirasa kurang penting.

• Membuat rencana untuk pemecahan masalah

Pada bagian ini ada dua kegiatan penting yaitu :

a) mencari berbagai cara penyelesaian yang mungkin diterapkan

b) membuat rencana pemecahan masalah

Penyelesaian suatu masalah biasanya tidak hanya satu tapi mungkin bias beberapa macam. Sebagai ilustrasi, apabila kita berada di kota Surabaya dan ingin pergi ke Jakarta, maka banyak cara yang mungkin bisa dilakukan, misalnya kita bisa menempuh dengan angkutan darat, laut atau udara. Dengan angkutan darat kita bisa menggunakan kereta api, bus atau angkutan yang lain. Jalurnya pun kita bisa lewat jalur utara, tengah atau selatan. Jadi banyak sekali cara penyelesaian yang bisa kita kembangkan. Masing-masing mempunyai karakteristik sendiri-sendiri. Dari sekian banyak penyelesaian ini kita harus memilih satu yang berdasarkan persyaratan tertentu merupakan cara yang paling baik untuk menyelesaikan permasalahan. Setelah terpilih, maka kita dapat membuat rencana kasar (outline) penyelesaian masalah dan membagi masalah dalam bagian-bagian yang lebih kecil. Rencana kasar (outline) penyelesaian masalah hanya berisi tahapan-tahapan utama penyelesaian masalah.

• Merancang dan menerapkan rencana untuk memperoleh cara penyelesaian Pada bagian ini rencana kasar penyelesaian masalah diperbaiki dan diperjelas dengan pembagian dan urutan rinci yang harus ditempuh dalam penyelesaian masalah.

• Memeriksa dan menyampaikan hasil dari pemecahan masalah

Bagian ini bertujuan untuk memeriksa apakah akurasi (ketepatan) hasil dari cara yang dipilih telah memenuhi tujuan yang diinginkan. Selain itu juga untuk melihat bagaimana daya guna dari cara yang dipilih yang dipilih.

Secara umum dalam rekayasa kita akan dihadapkan persoalan-persoalan berikut

¤ Persoalan yang akan dipecahkan

¤ Karakteristik entitas persoalan untuk mencari solusi

¤ Pendekatan dalam menemukan kesalahan-kesalahan yang dibuat didalam desain dan konstruksi

¤ Menyangga hasil rekayasa selama proses adaptasi, selama proses koreksi, serta ketika diperlukan perbaikan

Tedapat 3 fase umum dalam rekayasa perangkat lunak :

¤ Fase definisi

¤ Fase pengembangan

¤ Fase pemeliharaan

***Fase Definisi***

Berfokus pada “apa” (what); dimana pada definisi ini pengembang perangkat lunak harus mengidentifikasi informasi apa yang akan diproses, fungsi dan unjuk kerja apa yang dibutuhkan,tingkah laku sistem seperti apa yang diharapkan, interface apa yang akan dibangun, batasan desain apa yang ada, dan kriteria validasi apa yang dibutuhkan untuk mendefinisikan system yang sukses. Kebutuhan (requirement) kunci dari sistem dan perangkat lunak yang didefinisikan.

Metode yang diaplikasikan selama fase definisi berbeda, tergantung pada paradigma rekayasa perangkat lunak (atau kombinasi paradigma) yang diaplikasikan. Ada tiga tugas utama yang berada dalam bentuk yang sama; sistem atau rekayasa informasi, perencanaan proyek perangkat lunak, serta analisis kebutuhan.

***Fase Pengembangan***

Berfokus pada how (bagaimana), yaitu dimana selama masa pengembangan perangkat lunak, teknisi harus mendefinisikan bagaimana data dikunstruksikan, bagaimana fungsi-fungsi diimplementasikan sebagai sebuah arsitektur perangkat lunak, bagaimana detail prosedur akan diimplementasikan, bagaimana interface ditandai (dikarakterisasi), bagaimana rancangan akan diterjemahkan ke dalam bahasa pemrograman (atau bahasa non prosedural), serta bagaimana pengujian akan dilakukan. Metode-metode yang diaplikasikan selama masa pengembangan program

akan bervariasi, tetapi ada tiga tugas teknis yang khusus yang harus selalu ada; rancangan perangkat lunak, pemunculan kode,dan pengujian perangkat lunak.

***Fase Pemeliharaan.***

Berfokus pada perubahan (change), yang dihubungkan dengan koreksi kesalahan, penyesuaian yang dibutuhkan ketika lingkungan perangkat lunak berkembang, serta perubahan sehubungan dengan perkembangan yang disebabkan oleh perubahan kebutuhan pelanggan. Fase pemeliharaan mengaplikasikan lagi langkah-langkah pada fase definisi dan fase pengembangan, tetapi semuanya tetap tergantung pada konteks perangkat lunak yang ada. Ada empat tipe perubahan yang terjadi selama masa fase pengembangan, yaitu :

¤ Koreksi

¤ Adaptasi

¤ Perkembangan

¤ Pencegahan

***Koreksi***

Meskipun dengan jaminan kualitas yang terbaik, sepertinya pelanggan akan tetap menemukan cacat pada perangkat lunak. Pemeliharaan korektif (Corrective maintenance) mengubah perangkat lunak, membetulkan cacat atau kerusakan.

***Adaptasi***

Dari waktu ke waktu, lingkungan original ( contohnya CPU, sistem operasi, aturan-aturan bisnis, karakteristik produk eksternal) dimana perangkat lunak dikembangkan akan terus berubah. Pemeliharaan adaptif (Adaptif maintenance) menghasilkan modifikasi kepada perangkat lunak untuk mengakomodasi perubahan pada kebutuhan fungsional original.

***Perkembangan*** (Enhancement)

Ketika perangkat lunak dipakai, pemakai/pelanggan akan mengenali fungsi-fungsi tambahan yang memberi mereka keuntungan. Perfective maintenance memperluas perangkat lunak sehingga melampaui kebutuhan fungsi originalnya.

***Pencegahan***

Keadaan perangkat lunak semakin memburuk sehubungan dengan waktu, dan karena itu, preventive maintenance yang sering juga disebut Rekayasa perangkat lunak, harus dilakukan untuk memungkinkan perangkat lunak melayani kebutuhan para pemakainya. Pada dasarnya preventive maintenance melakukan perubahan pada program komputer sehingga bisa menjadi lebih mudah untuk dikoreksi,disesuaikan, dan dikembangkan. Sekarang, aging software plant memaksa banyak perusahaan untuk mengikuti strategi rekayasa perangkat lunak.

Secara global, rekayasa perangkat lunak sering dianggap sebagai bagian dari perekayasaan ulang proses bisnis. Fase dan langkah langkah yang berhubungan, seperti yang digambarkan pada padangan umum kita tentang rekayasa perangkat lunak, harus diimbangi dengan sejumlah aktifitas pelindung, yang meliputi :

1. kontrol dan pelacakan proyek perangkat lunak,

2. Review teknis formal,

3. Jaminan kualitas perangkat lunak,

4. Manajemen konfigurasi perangkat lunak,

5. Penghasilan dan penyiapan dokumen,

6. Manajemen reusabilitas,

7. Pengukuran,

8. Manajemen resiko.

* **Proses, Metode dan Alat Bantu**

Rekayasa perangkat lunak merupakan sebuah pempaparan rangkaian teknologi. Untuk mencapai suatu kualitas, banyak pendekatan keteknikan dan komitmen dasar yang mesti diterapkan. Misalnya Total Quality Management, atau pengembangan proses yang terus menerus.

 Fokus kepada kualitas merupakan penyangga dalam rekayasa perangkat lunak. Diatas penyangga tersebut, terdapat rangkaian proses yang menjadi dasar dalam rekayasa perangkat lunak. Prosesproses rekayasa perangkat lunak akan menjadi perekat yang menjaga rangkaian teknologi secara bersama-sama mendukung pengembangan perangkat lunak yang tepat waktu dan rasional. Proses proses tersebut membatasi kerangka kerja untuk serangkaian area proses kunci yang harus dibangun demi efektifitas pengembangan perangkat lunak.

Area proses kunci ini akan :

¤ membentuk dasar bagi kontrol manajemen proyek perangkat lunak

¤ membangun konteks dimana

- metoda teknis diaplikasikan

- produk dihasilkan

- fondasi dibangun

- kualitas dijamin

- dan perubahan diatur secara rapi.

Metode-metode rekayasa memberikan teknik untuk membangun perangkat lunak. Metode-metode tersebut meliputi serangkaian tugas yang luas yang menyangkut :

¤ analisis kebutuhan

¤ konstruksi program

¤ desain

¤ pengujian

¤ pemeliharaan

Tool-tool rekayasa perangkat lunak memberikan topangan yang otomatis ataupun semi otomatis pada proses-proses dan metode-metode yang ada.

* **Model – Model Proses Perangkat Lunak**

Untuk menyelesaikan masalah aktual dalam suatu setting industri, rekayasa perangkat lunak dan tim rekayasa mesti menggabungkan stratergi pengembangan dan fase-fase generik. Strategi sering direfer sebagai model proses atau paradigma rekayasa. Model proses yang akan dipilih dalam suatu proyek perangkat lunak dipilih berdasarkan sifat aplikasi dan sifat proyek itu sendiri.

Berikut model-model proses perangkat lunak yang populer :

1. Model Sekuensial Linear

2. Model Prototype

3. Model RAD

4. Model-model Evolusioner (Siklus hidup)

¤ Model inkremental

¤ Model Assembly

1. Teknik Generasi ke-empat
* Model Sekuensial Linier

Model sekuensial linier untuk *software engineering,* sering disebut juga

dengan **siklus kehidupan klasik** atau **model air terjun**. Model ini mengusulkan sebuah pendekatan kepada perkembangan software yang sistematik dan sekuensial yang mulai pada tingkat dan kemajuan sistem pada seluruh analisis, desain, kode, pengujian, dan pemeliharaan. Dimodelkan setelah siklus rekaysa konvensional, model sekuensial linier melingkupi aktivitas – aktivitas sebagai berikut :

**1. Rekayasa dan pemodelan sistem/informasi**

Karena sistem merupakan bagian dari sebuah sistem yang lebih besar, kerja dimulai dengan membangun syarat dari semua elemen sistem dan mengalokasikan beberapa subset dari kebutuhan ke software tersebut. Pandangan sistem ini penting ketika software harus berhubungan dengan elemen-elemen yang lain seperti software, manusia, dan database. Rekayasa dan anasisis system menyangkut pengumpulan kebutuhan pada tingkat sistem dengan sejumlah kecil analisis serta disain tingkat puncak. Rekayasa informasi mancakup juga pengumpulan kebutuhan pada tingkat bisnis strategis dan tingkat area bisnis.

**2. Analisis kebutuhan Software**

Proses pengumpulan kebutuhan diintensifkan dan difokuskan, khusunya pada software. Untuk memahami sifat program yang dibangun, analis harus memahami domain informasi, tingkah laku, unjuk kerja, dan interface yang diperlukan. Kebutuhan baik untuk sistem maupun software didokumentasikan dan dilihat lagi dengan pelanggan.

**3. Desain**

Desain software sebenarnya adalah proses multi langkah yang berfokus pada empat atribut sebuah program yang berbeda; struktur data, arsitektur software, representasi interface, dan detail (algoritma) prosedural. Proses desain menterjemahkan syarat/kebutuhan ke dalam sebuah representasi software yang dapat diperkirakan demi kualitas sebelum dimulai pemunculan kode. Sebagaimana persyaratan, desain didokumentasikan dan menjadi bagian dari konfigurasi software.

**4. Generasi Kode**

Desain harus diterjemahkan kedalam bentuk mesin yang bias dibaca. Langkah pembuatan kode melakukan tugas ini. Jika desain dilakukan dengan cara yang lengkap, pembuatan kode dapat diselesaikan secara mekanis.

1. **Pengujian**

Sekali program dibuat, pengujian program dimulai. Proses pengujian berfokus pada logika internal software, memastikan bahwa semua pernyataan sudah diuji, dan pada eksternal fungsional, yaitu mengarahkan pengujian untuk menemukan kesalahan – kesalahan dan memastikan bahwa input yang dibatasi akan memberikan hasil aktual yang sesuai dengan hasil yang dibutuhkan.

**6. Pemeliharaan**

Software akan mengalami perubahan setelah disampaikan kepada pelanggan (perkecualian yang mungkin adalah software yang dilekatkan). Perubahan akan terjadi karena kesalahan – kesalahan ditentukan, karena software harus disesuaikan untuk mengakomodasi perubahan – perubahan di dalam lingkungan eksternalnya (contohnya perubahan yang dibutuhkan sebagai akibat dari perangkat peripheral atau sistem operasi yang baru), atau karena pelanggan membutuhkan perkembangan fungsional atau unjuk kerja. Pemeliharaan software mengaplikasikan lagi setiap fase program sebelumnya dan tidak membuat yang baru lagi.

Masalah yang kadang terjadi ketika model sekuensial linier diaplikasikan adalah :

1. Jarang sekali proyek nyata mengikuti aliran sekuensial yang dianjurkan oleh model. Meskipun model linier bisa mengakomodasi iterasi, model ini melakukannya dengan cara tidak langsung. Sebagai hasilnya, perubahan – perubahan dapat menyebabkan keraguan pada saat tim proyek berjalan.

2. Kadang – kadang sulit bagi pelanggan untuk menyatakan semua kebutuhannya secara eksplisit. Model linier sekuensial memerlukan hal ini dan mengalami kesulitan untuk mengakomodasi ketidakpastian natural yang ada pada bagian awal beberapa proyek.

3. Pelanggan harus bersifat sabar. Sebuah versi kerja dari program – program kerja itu tidak akan diperoleh sampai akhir waktu proyek dilalui. Sebuah kesalahan besar, jika tidak terdeteksi sampai program yang bekerja tersebut dikaji ulang, bisa menjadi petaka.

4. Pengembang sering melakukan penundan yang tidak perlu. Sifat alami dari siklus kehidupan klasik membawa kepada *blocking state* di mana banyak anggota tim proyek harus menunggu tim yang lain untuk melengkapi tugas yang saling memiliki ketergantungan. Blocking state cenderung menjadi lebih lazim pada awal dan akhir sebuah proses sekuensial linier.

* Model Prototipe

*Prototyping paradigma* dimulai dengan pengumpulan kebutuhan. Pengembang dan pelanggan bertemu dan mendefinisikan obyektif keseluruhan dari software, mengidentifikasi segala kebutuhan yang diketahui, dan area garis besar diman definisi lebih jauh merupakan keharusan kemudian dilakukan “perancangan kilat”. Perancangan kilat berfokus pada penyajian dari aspek - aspek software tersebut yang akan nampak bagi pelanggan atau pemakai (contohnya pendekatan input dan format output). Perancangan kilat membawa kepada konstruksi sebuah prototipe. Prototipe tersebut dievaluasi oleh pelanggan/pemakai dan dipakai untuk menyaring kebutuhan pengembangan software. Iterasi terjadi pada saat prototipe disetel untuk memenuhi kebutuhan pelanggan, dan pada saat yang sama memungkinkan pengembang untuk secara lebih baik memahami apa yang harus dilakukannya.

Secara ideal prototipe berfungsi sebagai sebuah mekanisme untuk mengidentifikasi kebutuhan software. Bila prototipe yang sedang bekerja dibangun, pengembang harus mempergunakan fragmen – fragmen program yang ada atau mengaplikasikan alat –alat bantu (contohnya report generator, window manager, dll) yang memungkinkan program yang bekerja untuk dimunculkan secara cepat.

Prototipe bisa juga menjadi masalah karena alasan sebagai berikut:

1. Pelanggan melihat apa yang tampak sebagai versi software yang bekerja tanpa melihat bahwa prototipe itu dijalin bersama – sama “dengan permen karet dan *baling wire*”, tanpa melihat bahwa di dalam untuk membuatnya bekerja, kita belum menyantumkan kualitas software secara keseluruhan atau kemampuan pemeliharaan untuk jangka waktu yang panjang. Ketika diberi informasi bahwa produk harus dibangun lagi agar tingkat kualitas yang tinggi bisa dijaga, pelanggan akan meneriakan kecurangan dan permintaan agar dipakai “beberapa campuran” untuk membuat prototipe menjadi sebuah produk yang bekerja yang lebih sering

terjadi, sehingga manajemen pengembangan software menjadi penuh dengan belas kasihan.

2. Pengembang sering membuat kompromi – kompromi implementasi untuk membuat prototipe bekerja dengan cepat. Sistem operasi atau bahasa pemrograman yang tidak sesuai bisa dipakai secara sederhana karena mungkin diperoleh dan dikenal; algoritma yang tidak efisien secara sederhana bisa diimplementasikan untuk mendemonstrasikan kemampuan. Setelah selang waktu tertentu, pengembang mungkin mengenali pilihan – pilihan tersebut dan melupakan semua alasan mengapa mereka tidak cocok. Pilihan yang kurang ideal telah menjadi bagian integral dari sebuah sistem. Meskipun berbagai masalah bisa terjadi, prototipe bisa menjadi paradigm yang efektif bagi *Software Engineering.* Kuncinya adalah mendefinisikan aturan main pada saat awal; yaitu pelanggan dan pengembang keduanya harus setuju bahwa prototipe dibangun untuk berfungsi sebagai mekanisme pendefinisian kebutuhan. Prototipe kemudian disingkirkan (paling tidak sebagian), dan software aktual direkayasa dengan tertuju kepada kualitas dan kemampuan pemeliharaan.

* **MODEL RAPIN APLICATION DEVELOPMENT**

*Rapin 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?
1. **Data modeling**

Aliran informasi yang didefinisikan sebagai bagian dari fase business 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 modelling**

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 semua 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.

**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.

* **MODEL SPIRAL**

Model spiral (*spiral model*) adalah model proses software yang evolusioner yang merangkai sifat iteratif dari prototipe dengan cara kontrol dan aspek sistematis dari model sekuensial linier. Model ini berpotensi untuk pengembangan versi pertambahan software secara cepat. Di dalam model spiral, software dikembangkan di dalam suatu deretan pertambahan. Selama awal iterasi, rilis inkremental bisa merupakan sebuah model atau prototipe kertas. Selama iterasi berikutnya, sedikit demi sedikit dihasilkan versi sistem rekayasa yang lebih lengkap.

Model spiral dibagi menjadi sejumlah aktifitas kerangka kerja, disebut juga wilayah tugas, di antara tiga sampai enam wilayah tugas, yaitu :

1. Komunikasi Pelanggan

Tugas – tugas yang dibutuhkan untuk membangun komunikasi yang efektif di antara pengembangan dan pelanggan.

2. Perencanaan

Tugas – tugas yang dibutuhkan untuk mendefinisikan sumber – sumber daya, ketepatan waktu, dan proyek informasi lain yang berhubungan.

3. Analisis Risiko

Tugas – tugas yang dibutuhkan untuk menaksir risiko – risiko, baik manajemen maupun teknis.

4. Perekayasaan

Tugas – tugas yang dibutuhkan untuk membangun satu atau lebih representasi dari aplikasi tersebut.

5. Konstruksi dan peluncuran

Tugas – trugas yang dibutuhkan untuk mengkonstruksi, menguji, memasang (instal) dan memberikan pelayanan kepada pemakai (contohnya pelatihan dan dokumentasi).

6. Evaluasi pelanggan

Tugas – tugas yang dibutuhkan untuk memperoleh umpan balik dari pelnggan dengan didasarkan pada evaluasi representasi software, yang dibuat selama masa perekayasaan, dan diimplementasikan selama masa pemasangan.

Kekurangan model spiral adalah sulitnya untuk meyakinkan konsumen (khusunya dalam situasi kontrak) bahwa pendekatan evolusioner bisa dikontrol.

Model spiral memerlukan keahlian penaksiran risiko yang masuk akal , dan sangat

bertumpu pada keakhlian ini untuk mencapai keberhasilan. Jika resiko mayor tidak ditemukan dan diatur, pasti akan terjadi masalah. Akhirnya model itu sendiri masih baru dan belum dipergunakan secara luas seperti paradigma sekuensial dan prototipe.

* 1. **PENUTUP**
		1. **Rangkuman**
* **Perangkat lunak** adalah seluruh perintah yang digunakan untuk memproses informasi
* **Program** adalah kumpulan perintah yang dimengerti oleh computer
* **Prosedur** adalah perintah yang dibutuhkan oleh pengguna dalam memproses informasi
* RPL adalah suatu disiplin ilmu yang membahas semua aspek produksi perangkat lunak, mulai dari tahap awal yaitu analisa kebutuhan pengguna, menentukan spesifikasi dari kebutuhan pengguna, disain, pengkodean, pengujian sampai pemeliharaan sistem setelah digunakan.
* Tujuan RPL adalah menghasilkan perangkat lunak dengan kinerja tinggi, tepat waktu, berbiaya rendah, dan multiplatform.
* **Masalah (problem)** adalah perbedaan antara kondisi yang terjadi dan kondisi yang diharapkan dan **Gejala** adalah tanda/petunjuk terjadinya suatu masalah.
* Tipe-tipe masalah :

o Masalah pemenuhan standar

o Masalah pemilihan alternatif

o Masalah pemenuhan kepuasan konsumen

o Masalah pencapaian tujuan

* + **Pemecahan masalah** adalah sebuah proses dimana suatu situasi diamati kemudian bila ditemukan ada masalah dibuat penyelesaiannya dengan cara menentukan masalah, mengurangi atau menghilangkan masalah atau mencegah masalah tersebut terjadi.
	+ Tahapan utama pemecahan masalah :

o Memahami dan mendefinisikan masalah

o Membuat rencana untuk pemecahan masalah

o Merancang dan menerapkan rencana untuk memperoleh cara penyelesaian

o Memeriksa dan menyampaikan hasil dari pemecahan masalah

* + Model – Model Perangkat Lunak :

o Model sekuensial linier

* Model Prototipe
* Model Rapin Aplication Development
* Model Spiral
	+ 1. **Tes Formatif**
1. Tuliskan pengertian dari perangkat lunak dan rekayasa perangkat lunak.
2. Tuliskan persamaan karakteristik yang dimiliki oleh model-model pengembangan perangkat lunak.
3. Apakah gejala dan masalah itu?
4. Apakah yang menjadi tujuan rekayasa perangkat lunak?
	* 1. **Tindak Lanjut**

Sebagai tindak lanjut yang harus dilakukan setelah pelaksanaan tes formatif, adalah:

1. Jika hasil tes formatif telah mencapai kompetensi khusus yang diharapkan, maka akan dilanjutkan kepada materi pembelajaran selanjutnya. Namun, jika hasil tes belum mencapai standar yang diharapkan, maka akan dilakukan remedial terhadap kompetensi yang masih dianggap rendah dengan melakukan perbaikan pembelajaran atau memberikan tugas kepada mahasiswa yang belum mencapai kompetensi khusus;
2. Mengambil keputusan terhadap efektif tidaknya suatu metode pembelajaran untuk memperbaiki berbagai kelemahan;
3. Melakukan peta analisis terhadap pelaksanaan evaluasi yang dapat ditinjau dari segi bentuk tes yang digunakan, relevansi isi tes dengan bahan ajar, dan kesesuaian materi tes dengan kompetensi khusus yang diharapkan.
	* 1. **Kunci Tes Formatif**
4. - Perangkat lunak adalah Perintah (program computer) yang bila dieksekusi memberikan fungsi dan unjuk kerja seperti yang diinginkan.

 - Suatu disiplin ilmu yang membahas semua aspek produksi perangkat lunak, mulai dari tahap awal yaitu analisa kebutuhan pengguna, menentukan spesifikasi dari kebutuhan pengguna, disain, pengkodean, pengujian sampai pemeliharaan sistem setelah digunakan.

2. Persamaan Karakteristik yang dimiliki oleh model-model pengembangan perangkat lunak :

* Kebutuhan terhadap definisi masalah yang jelas.
* Tahapan-tahapan pengembangan yang teratur.
* Stakeholder berperan sangat penting dalam keseluruhan tahapan pengembangan.
* Dokumentasi merupakan bagian penting dari pengembangan perangkat lunak.
* Keluaran dari proses pengembangan perangkat lunak harus bernilai ekonomis.

3. **-**  Gejalaadalah tanda/petunjuk terjadinya suatu masalah.

 - Masalah adalah perbedaan antara kondisi yang terjadi dan kondisi yang diharapkan atau boleh juga diartikan sebagai perbedaan antara kondisi sekarang dengan tujuan yang diinginkan.

4.Tujuan Rekayasa Perangkat Lunak adalah untuk menghasilkan perangkat lunak dengan kinerja tinggi, tepat waktu, berbiaya rendah, dan multiplatform.

* + 1. **Daftar Pustaka**

 Simarmata, Janner, 2010. *Rekayasa Perangkat Lunak,* Yogyakarta : Andi Offset*.*

Tavri D. Mahyuzir, 1997, *Pengantar Rekayasa Perangkat Lunak*, Jakarta : Elexmedia Komputindo.

Al Bahra Bin Ladjamudin, 2006. *Rekayasa Perangkat Lunak,* Jakarta : Graha Ilmu.

Kristanto Andi, 2004. *Rekayasa Perangkat Lunak,* Yogyakarta : Gava Media

* + 1. **Senarai**

*Software,* Perangkat Lunak

*Hardware,* Perangkat Keras

*C++,* Salah satu bahasa pemrograman

*Rpl,* Rekayasa perangkat lunak

*Aplikasi,* Salah satu bagian perangkat lunak