Oleh : 5165-Kundang K Juman

Modul ke **3**

Proses Perangakat Lunak [lanjutan] :

Pendahuluan

Software atau perangkat lunak merupakan kumpulan intruksi yang dapat menjalankan suatu perintah Perangkat lunak komputer merupakan sebuah program komputer yang menjembatani pengguna komputer dan perangkat keras yang

digunakanya. Dengan kata lain perangkat lunak merupakan penerjemah antara manusia sebagai orang yang memberi instruksi dan komputer sebagai pihak yang menerima instruksi. Perangkat lunak adalah program komputer yang berfungsi sebagai sarana interaksi atau yang menjembatani pengguna dengan perankat keras dan juga sebagai penerjemah perintah-perintah yang dijalankan pengguna komputer untuk, diteruskanatau diproses oleh perangkat keras.Dari berbagai pengertian diatas dapat disimpulkan bahwa perangkat lunak membantu untuk menjalankan perintah dari pengguna yang akan diproses atau dijalankan oleh perangkat keras.

- Untuk memperkenalkan model proses perangkat lunak
- Untuk menggambarkan sejumlah model proses yang berbeda dan kapan mereka dapat digunakan
- Untuk menggambarkan model proses garis besar untuk rekayasa persyaratan, pengembangan perangkat lunak, pengujian dan evolusi

Model proses perangkat lunak merupakan suatu deskripsi yang disederhanakan dari proses perangkat lunak dan kemudian dipresentasikan dengan sudut pandang tertentu.Model proses perangkat lunak itu sendiri dibagi menjadi beberapa macam pengembangan, seperti dibawah ini :

Waterfall atau air terjun adalah model yang dikembangkan untuk pengembangan perangkat lunak, membuat perangkat lunak. model berkembang secara sistematis dari satu tahap ke tahap lain dalam mode seperti air terjun.
Model ini mengusulkan sebuah pendekatan kepada pengembangan software yang sistematik dan sekuensial yang mulai dari tingkat kemajuan sistem pada seluruh analisis, desain, kode, pengujian dan pemeliharaan. Model ini melingkupi aktivitas-aktivitas sebgai berikut : rekayasa dan pemodelan sistem informasi, analisis kebutuhan, desain, koding, mengujian dan pemeliharaan.Model pengembangan ini bersifat linear dari tahap awal pengembangan system yaitu tahap perencanaan sampai tahap akhir pengembangan system yaitu tahap pemeliharaan. Tahapan berikutnya tidak akan dilaksanakan sebelum tahapan sebelumnya selesai dilaksanakan dan tidak bisa kembali atau mengulang ke tahap sebelumnya.

Dalam model ini terdapat beberapa sifat-sifat yang menojol dan cenderung menjadi permasalahan pada model waterfall.

Dengan demikian, Waterfall dianggap pendekatan yang lebih cocok digunakan untuk proyek pembuatan sistem baru. Tetapi salah satu kelemahan paling dasar adalah menyamakan pengembangan perangkat keras dengan perangkat lunak dengan meniadakan perubahan saat pengembangan.

Tahapan Waterfall :

Tahap-Tahap dalam Model Proses Waterfall

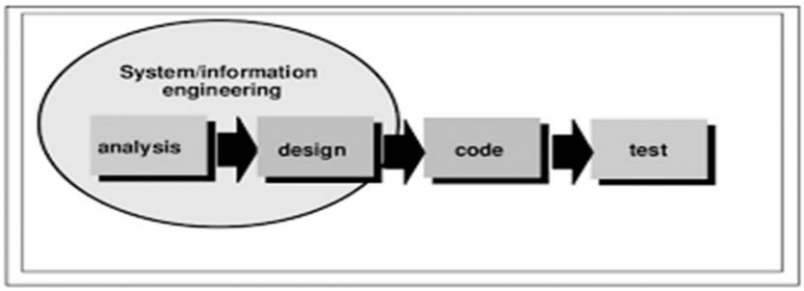
Pertama: Analisis dan Definisi Persyaratan melingkupin Pelayanan, batasan, dan tujuan sistem ditentukan melalui konsultasi dengan user sistem.

Kedua: Perancangan sistem dan Perangkat Lunak melingkupin Proses perancangan sistem membagi persyaratan dalam sistem perangkat keras atau perangkat lunak,lalu menentukan arsitektur sistem secara keseluruhan.

Ketiga: Implementasi dan pengujian unit : Perancangan perangkat lunak direalisasikan sebagai serangkaian program atau unit program. Pengujian unit melibatkan verifikasi bahwa setiap unit telah memenuhi spesifikasinya.

Keempat: Integrasi dan Pengujian Sistem dimana Unit program atau program individual diintegrasikan dan diuji sebagai sistem yang lengkap untuk menjamin bahwa persyaratan sistem telah dipenuhi.

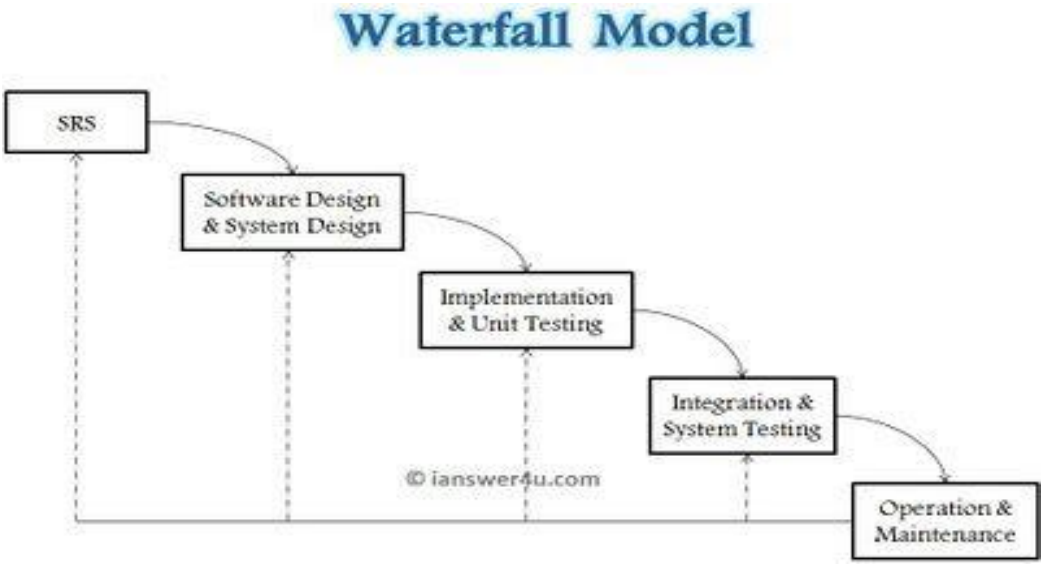
Kelima : Operasi dan Pemeliharaan Biasanya sih merupakan fase siklus yg paling lama (walaupun tidak seharusnya).dimana Sistem diinstall dan di pakai. Pemeliharaan pun mencakup koreksi dan berbagai error yg tdk ditemukan pada tahap-tahap sebelumnya, perbaikan atas implementasi unit sistem dan pengembangan pelayanan sistem.



Gambar 1 Tahapan Waterfall

Kelebihan dan kekurangan waterfall :

Waterfall Model adalah sebuah metode pengembangan software yang bersifat sekuensial. Metode ini dikenalkan oleh Royce pada tahun 1970 dan pada saat itu disebut sebaga isi klus klasik dan sekarang ini lebih dikenal dengan sekuensial linier. Selain itu Model ini merupakan model yang paling banyak dipakai oleh para pengembang software. Inti dari metode *waterfall* adalah pengerjaan dari suatu system dilakukan secara berurutan atau secara linear. Jadi jika langkah satu belum dikerjakan maka tidak akan bisa melanjutkan kelangkah 2, 3 dan seterusnya. Secara otomatis tahapan ke-3 akan bisa dilakukan jika tahap ke-1 dan ke-2 sudah dilakukan. Ada dua gambaran dari Waterfall Model, biarpun berbeda dalam menggunakan fase tapi intinya sama.



Gambar 2 Waterfall Model

Keterkaitan dan pengaruh antar tahap ini ada karena output sebuah tahap dalam Waterfall Model merupakan input bagi tahap berikutnya, dengan demikian ketidak sempurnaan hasil pelaksanaan tahap sebelumnya adalah awal ketidak sempurnaan tahap berikutnya. Memperhatikan karakteristik ini, sangat penting bagi tim pengembang dan perusahaan untuk secara bersama-sama melakukan analisa kebutuhan dan desain system sesempurna mungkin sebelum masuk kedalam tahap penulisan kode program. Secara garis besar metode*waterfall* mempunyai langkah-langkah sebagai berikut :

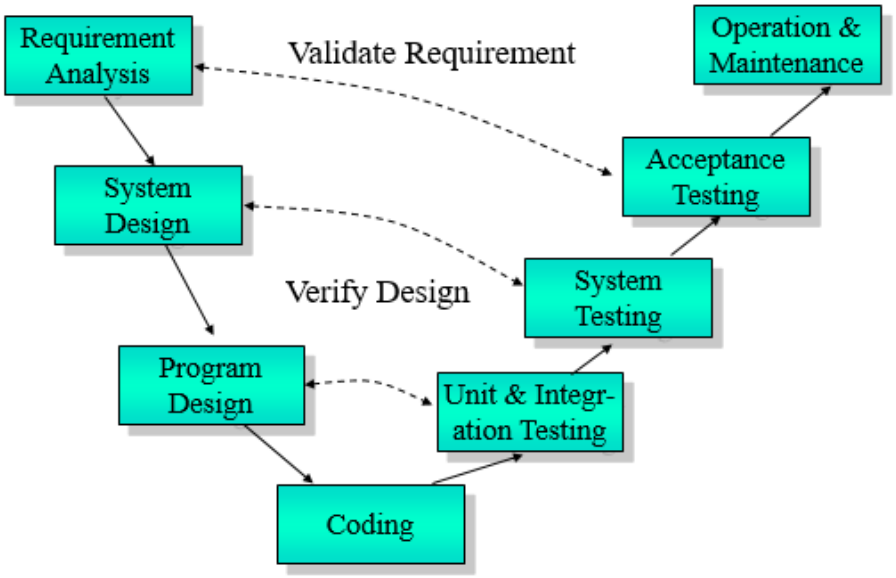
Kelemahan waterfall :

1. Diperlukan majemen yang baik, karena proses pengembangan tidak dapat dilakukan secara berulang sebelum terjadinya suatu produk.
2. Kesalahan kecil akan menjadi masalah besar jika tidak diketahui sejak awal pengembangan yang berakibat pada tahapan selanjutnya.
3. Pelanggan sulit menyatakan kebutuhan secara eksplisit sehingga tidak dapat mengakomodasi ketidak pastian pada saat awal pengembangan.
4. Pelanggan harus sabar, karena pembuatan perangkat lunak akan dimulai ketika tahap desain sudah selesai. Sedangkan pada tahap sebelum desain bisa memakan waktu yang lama.
5. Pada kenyataannya, jarang mengikuti urutan sekuensial seperti pada teori. Iterasi sering terjadi menyebabkan masalah baru.

The V-model

Merupakan Proses Pengembangan Perangkat Lunak (Juga Berlaku Untuk Pengembangan Hardware) Yang Dapat Dianggap Sebagai Perluasan Dari Model Air Terjun. Alih-alih Bergerak Turun Dengan Cara Yang Linear, Langkah-langkah Proses Yang Bengkok Ke Atas Setelah Fase Coding, Untuk Membentuk Bentuk V Yang Khas. The V-model Menunjukkan Hubungan Antara Setiap Fase Siklus Hidup Pengembangan Dan Fase Terkait Pengujian. Sumbu Horisontal Dan Vertikal Merupakan Kelengkapan Waktu Atau Proyek (Kiri Ke Kanan) Dan Tingkat Abstraksi (Yang Kasar-butiran Menonjol Abstraksi), Masing-masing.

:



Gambar 3 V Model

Dalam tahap ini analis sistem mulai merancang sistem dengan mengacu pada dokumentasi kebutuhan pengguna yang sudah dibuat pada tahap sebelumnya. Keluaran dari tahap ini adalah spesifikasi software yang meliputi organisasi sistem secara umum, struktur data, dan yang lain. Selain itu tahap ini juga menghasilkan contoh tampilan window dan juga dokumentasi teknik yang lain seperti Entity

Model Spiral :

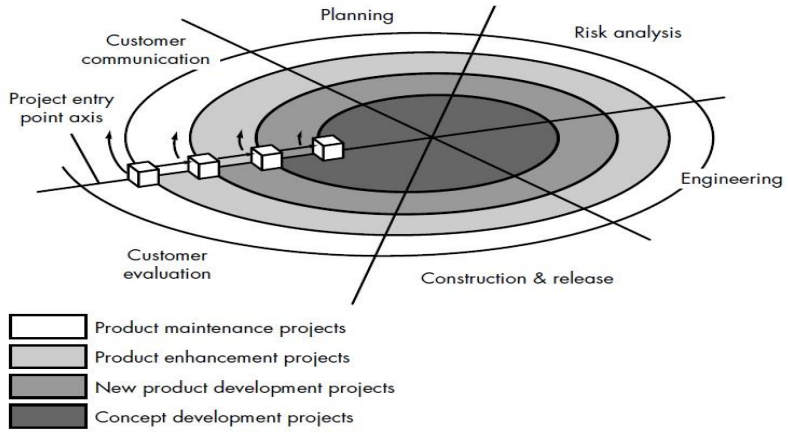
Model spiral

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 :

1. Komunikasi pelanggan, tugas-tugas yang dibutuhkan untuk membangun komunikasi yang efektif diantara pengembang dan pelanggan.
2. Perencanaan, tugas-tugas yang dibutuhkan untuk mendefinisikan sumber-sumber daya, ketepatan waktu, dan proyek informasi lain yang berhubungan.
3. Analisis resiko, tugas-tugas yang dibutuhkan untuk menaksir resiko-resiko, baik manajemen maupun teknis.

4. Perekayasaan, tugas-tugas yang dibutuhkan untuk membangun satu atau lebih representasi dari aplikasi tersebut.
5. 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.



Gambar 4 Spiral Model

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.

a. 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.

b. 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.

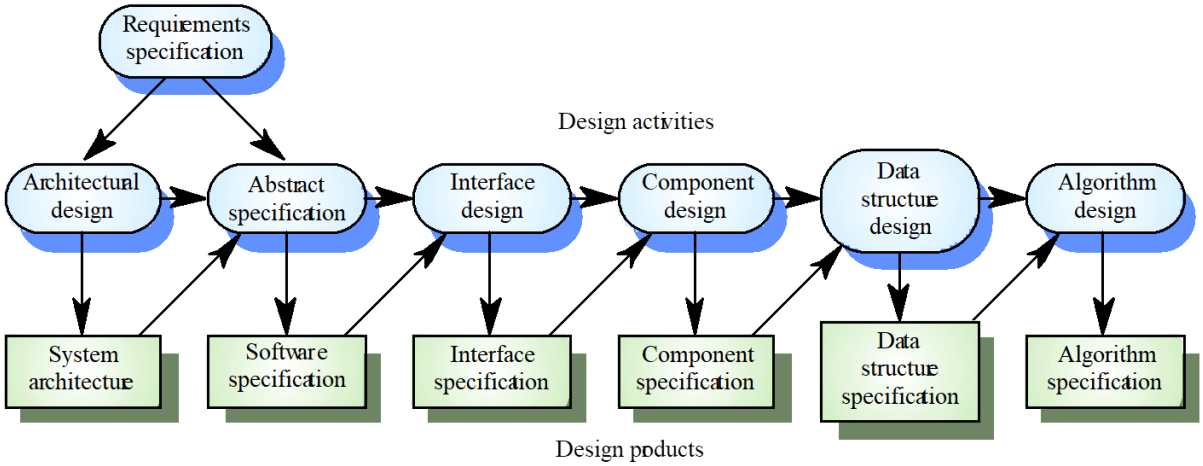
Software Engingeering :

Dalam rekayasa perangkat luna k tahap awal adalah pendefinisian tentang rekayasa system apa yang akan dibuat. Diperlukan proses perencanaan dan analisis kebutuhan. Setelah pendefinisian tahap selanjutnya adalah pengembangan, dalam tahap ini adalah bagaimana produk yang telah didefinisikan dengan jelas kemudian akan mulai diimplementasikan.Maka pada proses pengembangan ini akan dilakukan design software, kemudian mengenerate koding - koding pembangun program, hingga program siap dites kebenarannya. Proses rekayasa perangkat lunak adalah proses yang terus berulang, karena karakteristik perangkat lunak yang membutuhkan pemeliharaan dan continue development agar perangkat lunak tidak kadarluasa. Dalam proses pemelihataan kita melakukan koreksi kesalahan, adaptasi kebutuhan, pe

Software ningkatan kemampuan atau fungsi dan bentuk pencegahan lainnya agar perangkat lunak tersebut tidak kadarluasa.

Software design and implementation

Tahap implementasi pengembangan perangkat lunak merupakan proses pengubahan spesifikasi sistem menjadi sistem yang dapat dijalankan. Tahap ini selalu mencakup proses perancangan dan pemrograman perangkat lunak. Perancangan perangkat lunak merupakan deskripsi struktur perangkat lunak yang akan diimplementasikan, data yang merupakan bagian sistem, interface antara komponen - komponen sistem dan terkadang algoritma yang digunakan.



Gambar 5 Software Engineering

Tahap proses perancangan bersifat urut (sekuensial). Pada kenyataannya, kegiatan proses perancangan akan saling tumpang tindih. Spesifikasi untuk tahap berikutnya merupakan output dari setiap kegiatan perancangan. Spesifikasi ini bisa merupakan spesifikasi abstrak dan formal yang dihasilkan untuk menjelaskan persyaratan, tapi bisa juga merupakan spesifikasi mengenai bagaimana bagian sistem akan direalisasikan.

Design methods

Metode terhadap perancangan perangkat lunak adalah metode terstruktu yang merupakan serangkaian notasi dan panduan untuk perancangan perangkat lunak.

Metode terstruktur mencakup model proses perancangan, notasi untuk merepresentasikan desain tersebut, format laporan, aturan dan panduan perancangan. Walaupun ada banyak metode, mereka memiliki banyak kesamaan.

Metode sistem

- Metode aliran data Dimana sistem dimodelkan dengan menggunakan transformasi data yang terjadi pada saat pemrosesannya.
- Model relasi entitas Merupakan teknik normal yang dipakai untuk mendeskripsikan struktuk database.
- Model struktural Dimana komponen komponen sistem dan interaksinya didokumentasikan
- Metode orientasiObjek mencakup model inheritansi (pewarisan) sistem, model hubungan statis dan dinamis diantara objek, dan model bagaimana objek berinteraksi satu sama lain ketika sistem sedang dijalankan.

Software validation

Verifikasi adalah proses pemeriksaan apakah logika operasional model (program komputer) sesuai dengan logika diagram alur. Kalimat sederhananya, apakah ada kesalahan dalam program? (Hoover dan Perry, 1989); verifikasi adalah pemeriksaan apakah program komputer simulasi berjalan sesuai dengan yang diinginkan, dengan pemeriksaan program komputer. Verifikasi memeriksa penerjemahan model simulasi konseptual (diagram alur dan asumsi) ke dalam bahasa pemrograman secara benar (Law dan Kelton, 1991) .

Validasi adalah proses penentuan apakah model, sebagai konseptualisasi atau abstraksi, merupakan representasi berarti dan akurat dari sistem nyata? (Hoover dan Perry, 1989); validasi adalah penentuan apakah mode konseptual simulasi (sebagai tandingan program komputer) adalah representasi akurat dari sistem nyata yang sedang dimodelkan (Law dan Kelton, 1991).

etika membangun model simulasi sistem nyata, kita harus melewati beberapa tahapan atau level pemodelan. Seperti yang dapat dilihat pada Gambar diatas, pertama kita harus membangun model konseptual yang memuat elemen sistem

nyata. Dari model konseptual ini kita membangun model logika yang memuat relasi logis antara elemen sistem juga variabel

eksogenus yang mempengaruhi sistem. Model kedua ini sering disebut sebagai model diagram alur. Menggunakan model diagram alur ini, lalu dikembangkan program komputer, yang disebut juga sebagai model simulasi, yang akan mengeksekusi model diagram alur. Pengembangan model simulasi merupakan proses iteratif dengan beberapa perubahan kecil pada setiap tahap. Dasar iterasi antara model yang berbeda adalah kesuksesan atau kegagalan ketika verifikasi dan validasi setiap model. Ketika validasi model dilakukan, kita mengembangkan representasi kredibel sistem nyata, ketika verifikasi dilakukan kita memeriksa apakah logika model diimplementasikan dengan benar atau tidak. Karena verifikasi dan validasi berbeda, teknik yang digunakan untuk yang satu tidak selalu bermanfaat untuk yang lain.

Baik untuk verifikasi atau validasi model, kita harus membangun sekumpulan kriteria untuk menilai apakah diagram alur model dan logika internal adalah benar dan apakah model konseptual representasi valid dari sistem nyata. Bersamaan dengan kriteria evaluasi model, kita harus spesifikasikan siapa yang akan mengaplikasikan kriteria dan menilai seberapa dekat kriteria itu memenuhi apa yang sebenarnya.

a. Automated process support (CASE)

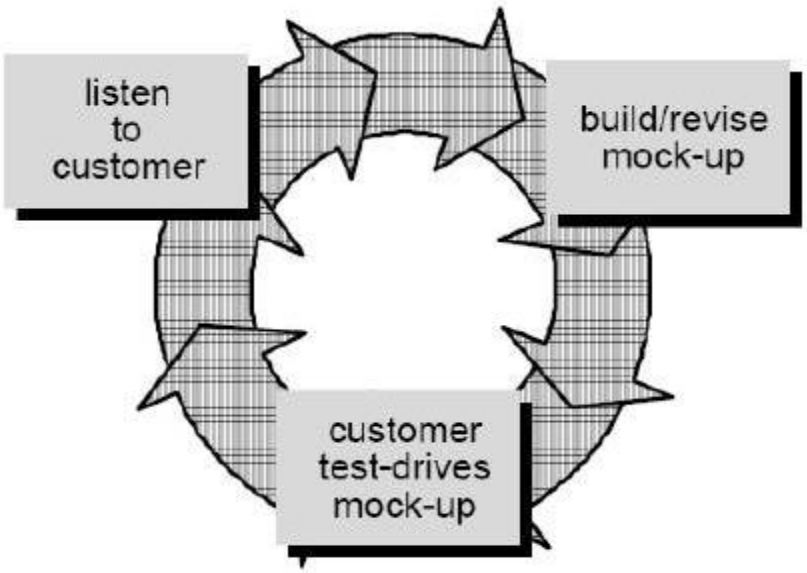
- CASE adalah perangkat lunak untuk mendukung pengembangan perangkat **lunak dan proses evolusi**
- Otomatisasi aktivitas
- Editor grafis untuk pengembangan model sistem
- Pembuat UI grafis untuk konstruksi antarmuka pengguna
- Debugger untuk mendukung penemuan kesalahan program

Model Pengembangan perangkat lunakprototyping

Dahulu, rancangan fisik merupakan proses yang menggunakan kertas dan pinsil. Seorang analis mengambarkan tata letak atau struktur dari *output*, *input*, basis data, dan aliran hubungan dan prosedur. Ini merupakan proses yang memakan waktu yang memiliki kemungkinan terjadinya kesalahan. Biasanya hasil dari rancangan kertas ini adalah tidak lengkap dan tidak akurat. Sekarang, banyak analis dan perancang memilih *Prototyping*, sebuah pendekatan berbasis rekayasa (*engineering*)

untuk merancang. Pendekatan *Prototyping* adalah proses iterative yang melibatkan hubunan kerja yang dekat antara perancang dan pengguna.

Pressman (2001) menyatakan bahwa seringkali seorang pelanggan mendefinisikan serangkaian sasaran umum bagi perangkat lunak, tetapi tidak mengidentifikasi kebutuhan *input*, pemrosesan, ataupun *output* detail. Pada kasus yang lain, pengembang mungkin tidak memiliki kepastian terhadap efisiensi algoritme, kemapuan penyesuaian dari sistem operasi, atau bentuk-bentuk yang harus dilakukan oleh interaksi manusia dan mesin. Dalam situasi seperti ini salah satu model yang cocok digunakan adalah model *prototype*(*Prototyping paradigm*). Model *Prototype* dapat dilihat pada gamba



Gambar 6 Prototype model

Pendekatan *Prototyping* melewati tiga proses, yaitu pengumpulan kebutuhan, perancangan, dan evaluasi *Prototype*. Proses-proses tersebut dapat dijelaskan sebagai berikut:

Pengumpulan kebutuhan: *developer* dan klien bertemu dan menentukan tujuan umum, kebutuhan yang diketahui dan gambaran bagian-bagian yang akan dibutuhkan berikutnya;

1. Perancangan: perancangan dilakukan cepat dan rancangan mewakili semua aspek *software* yang diketahui, dan rancangan ini menjadi dasar pembuatan *prototype*;
2. 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*. Pendekatan ini memiliki beberapa keuntungan :

1. Pemodelan membutuhkan partisipasi aktif dari *end-user*. Hal ini akan meningkatkan sikap dan dukungan pengguna untuk pengerjaan proyek. Sikap moral pengguna akan meningkat karena system berhubungan nyata dengan mereka.
2. Perubahan dan iterasi merupakan konsekuensi alami dari pengembangan system-sehingga *end user* memiliki keinginan untuk merubah pola pikirnya. *Prototyping* lebih baik menempatkan situasi alamiah ini karena mengasumsikan perubahan model melalui iterasi kedalam system yang dibutuhkan.
3. *Prototyping* mematahkan folosofi "*end user* tidak mengetahui secara detail apa yang dibutuhkan sampai mereka melihat implementasinya"
4. *Prototyping* adalah model aktif, tidak pasif, sehingga *end user* dapat melihat, merasakan, dan mengalaminya.
5. Kesalahan yang terjadi dalam *prototyping* dapat dideteksi lebih dini
6. *Prototyping* dapat meningkatkan kreatifitas karena membolehkan adanya *feedback* dari *end user*. Hal ini akan memberikan solusi yang lebih baik.
7. *Prototyping* mempercepat beberapa fase hidup dari *programmer*.
8. McLeod dan Schell (2001) mengemukakan bahwa alasan-alasan pemakai maupun

spesialis informasi menyukai model *prototype* adalah:

1. Komunikasi antara analis sistem dan pemakai membaik;
2. Analis dapat bekerja dengan lebih baik dalam menemukan kebutuhan pemakai;
3. Pemakai berperan lebih aktif dalam pengembangan sistem;
4. Spesialis informasi dan pemakai menghabiskan lebih sedikit waktu dan usaha dalam mengembangkan sistem;
5. Implementasi menjadi lebih mudah karena pemakai mengetahui sistem yang diharapkan.
6. Tetapi, terdapat beberapa kelemahan dari *prototyping*, kelemahan tersebut antara lain :
7. *Prototyping* memungkinkan terjadinya pengembalian terhadap kode, implementasi, dan perbaikan siklus hidup yang dugunakan untuk mendominasi sistem informasi.
8. *Prototyping* tidak menolak kebutuhan dari fase analisis sistem. *Prototype*hanya dapat memecahkan masalah yang salah dan memberi kesempatan sebagai sistem pengembangan konvensional.
9. Perancangan issu numerik tidak dialamaykan oleh *prototyping*. Isu tersebut dapat dilupakan jika pengguna tidak berhati-hati.
10. *Prototyping* dapat mengurangi kreatifitas perancangan.
11. *Prototyping* terkadang dapat memberikan performansi yang lambat, membantu mendapatkan kebutuhan detil lebih baik namun demikian *Prototype* juga menimbulkan masalah:

1. 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.

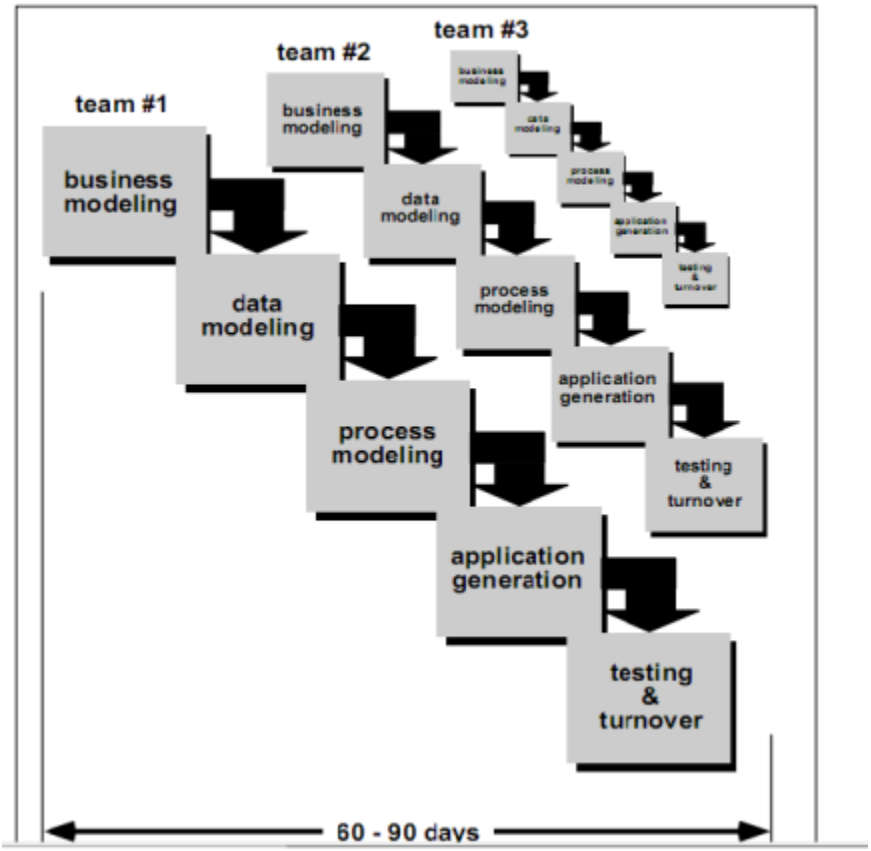
Agar model ini bisa berjalan dengan baik, perlu disepakati bersama oleh klien dan developer bahwa *prototype* yang dibangun merupakan alat untuk mendefinisikan kebutuhan software.

RAD (Rapid Application Development) :

Rapid application development (RAD) atau rapid prototyping adalah model proses pembangunan perangkat lunak yang tergolong dalam teknik incremental (bertingkat). RAD menekankan pada siklus pembangunan pendek, singkat, dan cepat. Waktu yang singkat adalah batasan yang penting untuk model ini. Rapid application development menggunakan metode iteratif (berulang) dalam mengembangkan sistem dimana working model (model bekerja) sistem dikonstruksikan di awal tahap pengembangan dengan tujuan menetapkan kebutuhan (requirement) user. RAD mengadopsi model waterfall dan pembangunan dalam waktu singkat dicapai dengan menerapkan component based construction.

Kelemahan dalam model RAD yaitu:

1. Model RAD membutuhkan sumber daya yang besar, terutama untuk proyek dengan skala besar.
2. proyek bisa gagal karena waktu yang disepakati tidak dipenuhi
3. sistem yang tidak bisa dimodularisasi tidak cocok untuk model RAD
4. resiko teknis yang tinggi juga kurang cocok untuk model RAD



Gambar 7 Rad Model

Secara umum fase-fase pada RAD adalah sebagai berikut

- **Bussines modeling**
- **Data modeling**
- **Proses modeling**
- **Application generation** : RAD mengasumsikan pemakaian teknik 4G (generasi keempat). Selain menciptakan Perangkat Lunak dengan bahasa pemrograman generasi ketiga yang konvensional, RAD lebih banyak memproses kerja untuk memakai lagi komponen program atau menciptakan komponen yang bisa dipakai lagi.
- **Testing and Turn Over** : karena menekankan pada reusability, banyak komponen program yang telah diuji sehingga mengurangi keseluruhan waktu pengujian. Tapi komponen baru harus diuji dan semua interface harus dilatih secara penuh.

AGILE SOFTWARE DEVELOPMENT

Agile merupakan adalah jenis pegembangan sistem jangka pendek yang memerlukan adaptasi cepat dan pengembang terhadap perubahan dalam bentuk apapun. Dalam Agile Software Development interaksi dan personel lebih penting dari pada proses dan alat, software yang berfungsi lebih penting daripada dokumentasi yang lengkap, kolaborasi dengan klien lebih penting dari pada negosiasi kontrak, dan sikap tanggap terhadap perubahan lebih penting daripada mengikuti rencana. Agile juga dapat diartikan sebagai sekelompok metodologi pengembangan software yang didasarkan pada prinsip-prinsip yang sama atau pengembangan system jangka pendek yang memerlukan adaptasi cepat dari pengembang terhadap perubahan dalam bentuk apapun. Menurut Agile Alliance, ada 12 prinsip yang mendorong keberhasilan dalam penerapan Agile Software Development, yaitu:

1. Kepuasan klien adalah prioritas utama dengan menghasilkan produk lebih awal dan terus menerus.
2. Menerima perubahan kebutuhan, sekalipun diakhir pengembangan.
3. Penyerahan hasil/software dalam hitungan waktu beberapa minggu sampai beberapa bulan.
4. Pihak bisnis dan pengembang harus bekerja sama setiap hari selama pengembangan berjalan.
5. Membangun proyek dilingkungan orang-orang yang bermotivasi tinggi yang bekerja dalam lingkungan yang mendukun dan yang dipercaya untuk dapat menyelesaikan proyek.
6. Komunikasi dengan berhadapan langsung adalah komunikasi yang efektif dan efisien
7. Software yang berfungsi adalah ukuran utama dari kemajuan proyek
8. Dukungan yang stabil dari sponsor, pembangun, dan pengguna diperlukan untuk menjaga perkembangan yang berkesinambungan
9. Perhatian kepada kehebatan teknis dan desain yang bagus meningkatkan sifat agile
10. Kesederhanaan penting
11. Arsitektur, kebutuhan dan desain yang bagus muncuk dari tim yang mengatur dirinya sendiri

12. Secara periodik tim evaluasi diri dan mencari cara untuk lebih efektif dan segera melakukannya.

Kelebihan dari Agile Software Development yaitu:

- Meningkatkan kepuasan kepada klien
- Pembangunan system dibuat lebih cepat
- Mengurangi resiko kegagalan implementasi software dari segi non-teknis
- Jika pada saat pembangunan system terjadi kegagalan,kerugian dar segi materi relative kecil.

DAFTAR PUSTAKA

1, Roger Presman, Software engineering
2.Ian Somovile Software engineering
3. Metode pengembangan perangkat lunak,
https://andisulaeman.weebly.com/.../metode__pengembangan__perangkat__lunak