

PERENCANAAN PROSES PERANGKAT LUNAK

Dosen: Imam Khairi, ST, MT

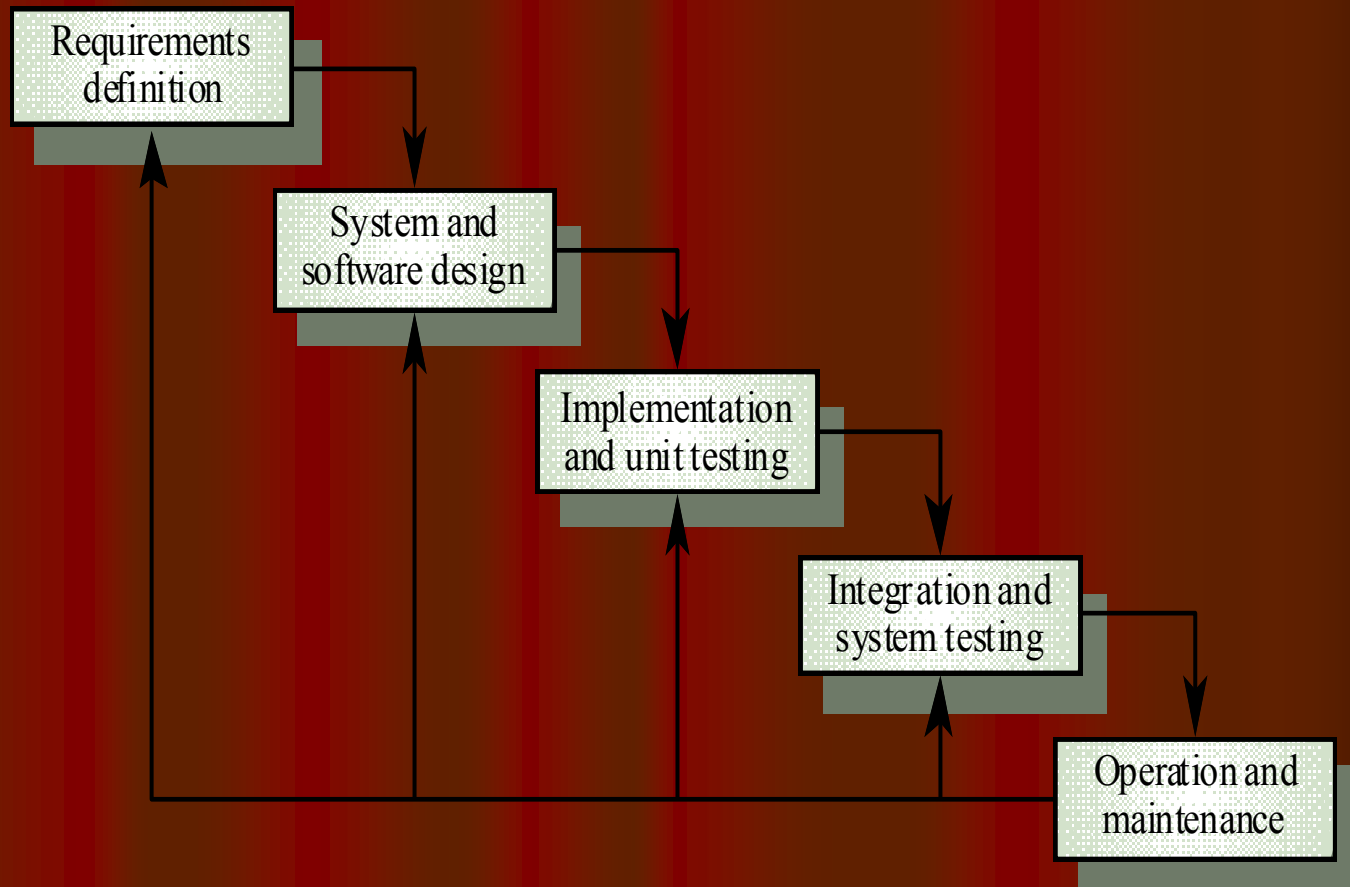
PROSES PERANGKAT LUNAK

- Serangkaian kegiatan dan hasil yang berhubungan dengannya, yang menuju pada dihasilkannya produk perangkat lunak.
- Kegiatan-kegiatan mendasar yg umum bagi semua proses Perangkat Lunak :
 1. Spesifikasi Perangkat Lunak → Fungsionalitas perangkat lunak dan batasan kemampuan operasinya harus didefinisikan.
 2. Pengembangan (Perancangan dan Implementasi) Perangkat Lunak → Perangkat lunak yang memenuhi spesifikasi harus di produksi
 3. Validasi Perangkat Lunak → Perangkat lunak harus divalidasi untuk menjamin bahwa perangkat lunak bekerja sesuai dengan apa yang diinginkan oleh pelanggan.
 4. Evolusi Perangkat Lunak → Perangkat lunak harus berkembang untuk memenuhi kebutuhan pelanggan.

Model Proses Perangkat Lunak

- A. Model air terjun (*waterfall*) → Biasa juga disebut siklus hidup perangkat lunak. Mengambil kegiatan dasar seperti spesifikasi, pengembangan, validasi, dan evolusi dan merepresentasikannya sebagai fase-fase proses yang berbeda seperti spesifikasi persyaratan, perancangan perangkat lunak, implementasi, pengujian dan seterusnya.

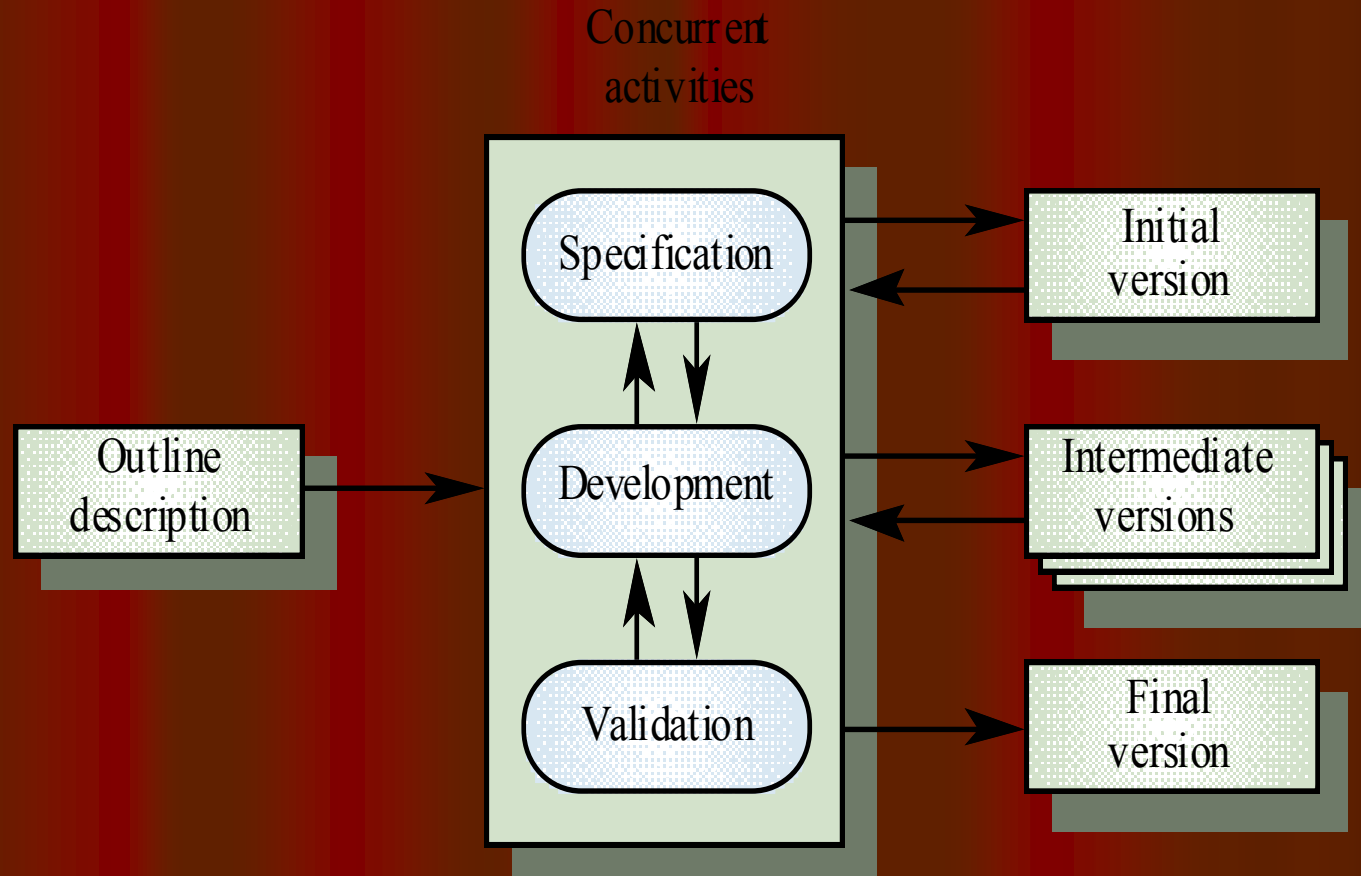
Gambar model *waterfall*



B. Pengembangan Evolusioner → Berdasarkan pada ide untuk mengembangkan implementasi awal, memperlihatkan kepada *user* untuk dikomentari, dan memperbaikinya versi demi versi sampai sistem yang memenuhi persyaratan diperoleh.

Tidak ada kegiatan spesifikasi, pengembangan, dan validasi yang terpisah. Kegiatan2 ini dilakukan pada saat yang bersamaan dengan umpan balik yang cepat untuk masing2 kegiatan.

Gambar model Pengembangan Evolusioner



Ada 2 jenis pengembangan evolusioner

1. Pengembangan Eksplotari → Tujuan proses ini adalah bekerja dengan pelanggan untuk menyelidiki persyaratan mereka dan mengirimkan sistem akhir. Harusnya diawali dengan kebutuhan yang sudah dimengerti.
2. Prototipe yang dapat dibuang (*throw-away*) → Berkonsentrasi pada eksperimen, dengan persyaratan pelanggan yang tidak dipahami dengan baik.

Kelebihan model Pengembangan Evolusioner

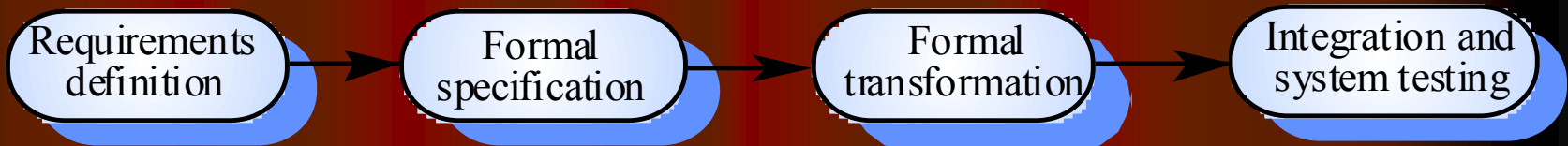
- Lebih efektif dari pendekatan air terjun dalam menghasilkan sistem yang memenuhi kebutuhan langsung dari pelanggan.
- Sementara user mendapat pemahaman yang lebih baik dari masalah mereka, sistem perangkat lunak dapat merefleksikannya.

Masalah pada model Pengembangan Evolusioner

- Kurangnya visibilitas proses → Jika sistem dikembangkan dengan cepat, tidaklah efektif dari segi biaya jika dihasilkan dokumen yang merefleksikan setiap versi sistem.
- Sistem seringkali memiliki struktur yang buruk → Perubahan yang terus-menerus cenderung merusak struktur perangkat lunak. Penyesuaian perubahan menjadi kian sulit dan mahal.
- Membutuhkan kemampuan khusus.

C. Model Pengembangan Sistem Formal

- Proses pengembangan Perangkat Lunak didasarkan pada transformasi matematis dari spesifikasi sistem menjadi program yang dapat dijalankan.



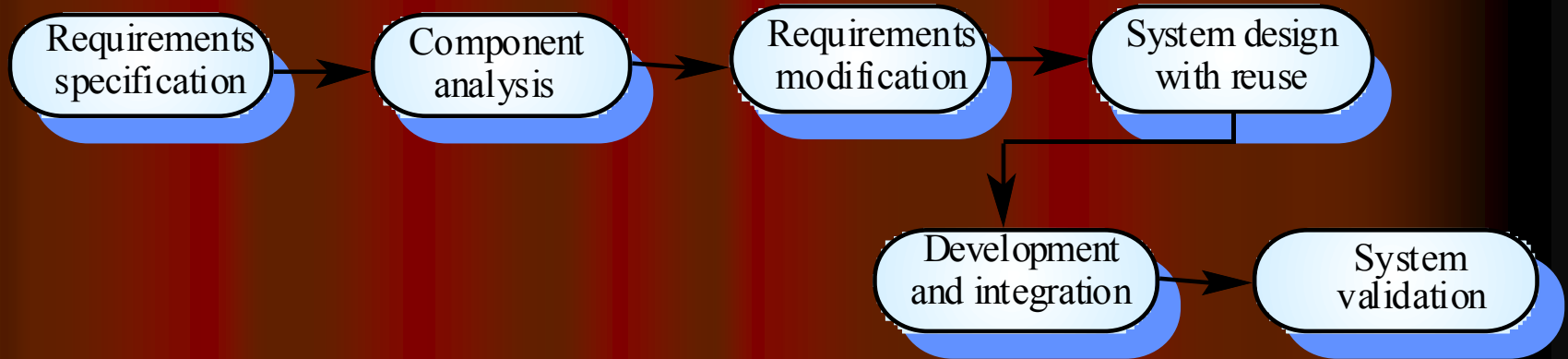
Masalah dalam Pengembangan Metode Formal

- Memerlukan keahlian khusus dan pelatihan untuk mengaplikasikannya
- Untuk sebagian besar sistem, metode ini tidak memberikan keuntungan biaya atau kualitas yang signifikan dibandingkan dengan pendekatan yang lain.

D. Model Pengembangan Berorientasi Pemakaian Ulang (*Re-Usable*)

- Bergantung pada sejumlah besar komponen perangkat lunak yang dapat dipakai ulang, yang bisa didapat, dan berapa kerangka kerja integrasi untuk komponen-komponen ini.
- Komponen-komponen ini dapat juga disebut sistem yang disebut COTS (*Commercial Off-The-Shelf Systems*/Sistem Siap Beli Komersial) yang dapat digunakan untuk memberikan fungsionalitas khusus seperti format teks, perhitungan numerik,dll.

Gambar Model Pengembangan Berorientasi Pemakaian Ulang (*Re-Usable*)



Tahap-tahap *Re-Usable*

1. Analisis Komponen → Spesifikasi persyaratan telah diketahui, komponen2 untuk implementasi spesifikasi tersebut akan dicari. Biasanya, tidak ada kesesuaian yang tepat dan komponen yang dapat dipakai hanya memberikan sebagian dari fungsionalitas yang dibutuhkan.
2. Modifikasi Persyaratan → Persyaratan dianalisis menggunakan informasi tentang komponen yang didapat, kemudian dimodifikasi untuk merefleksikan komponen yang ada. Jika modifikasi tidak mungkin dilakukan, maka kegiatan analisis komponen bisa diulang untuk mencari solusi alternatif.

3. Perancangan sistem dengan pemakaian ulang → Kerangka kerja sistem dirancang, atau kerangka kerja yang telah ada dipakai ulang.
4. Pengembangan dan Integrasi → Perangkat Lunak yang tidak dapat dibeli akan dikembangkan dan komponen dan sistem COTS diintegrasikan untuk membantu sistem.

Keuntungan Model *Re-Usable*

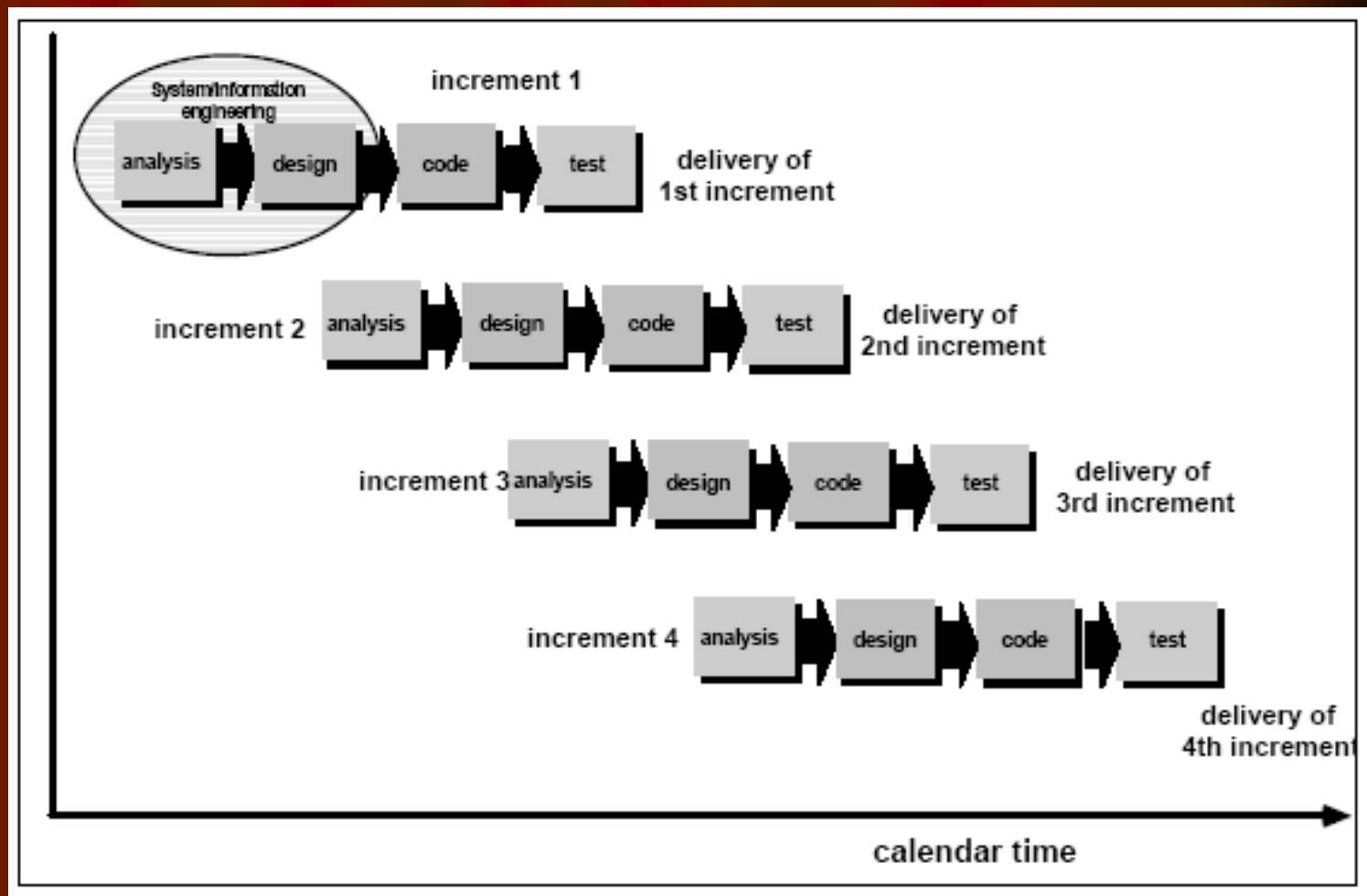
- Mengurangi besarnya perangkat lunak yang akan dikembangkan
- Memperkecil biaya dan resiko
- Memungkinkan penyelesaian perangkat lunak dengan cepat

ITERASI PROSES

- Digunakan untuk kebanyakan sistem besar
- Perlu digunakan berbagai pendekatan untuk berbagai bagian sistem, sehingga harus digunakan model HIBRID → bagian proses diulang, sementara persyaratan sistem berubah.
- Terdapat 2 model iterasi :
 - Pengembangan Inkremental
 - Pengembangan Spiral

A. Pengembangan Inkremental

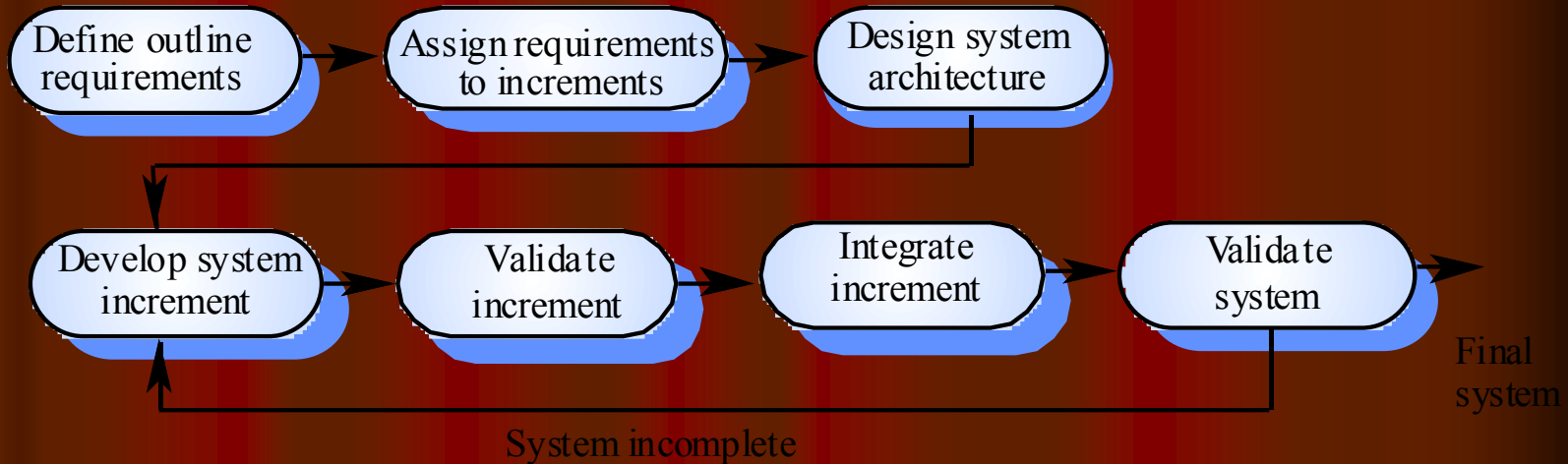
- Pengembangan sistem berdasarkan model sistem yang dipecah sehingga model pengembangannya secara increment/bertahap.
- Kebutuhan pengguna diprioritaskan dan prioritas tertinggi dimasukkan dalam awal increment



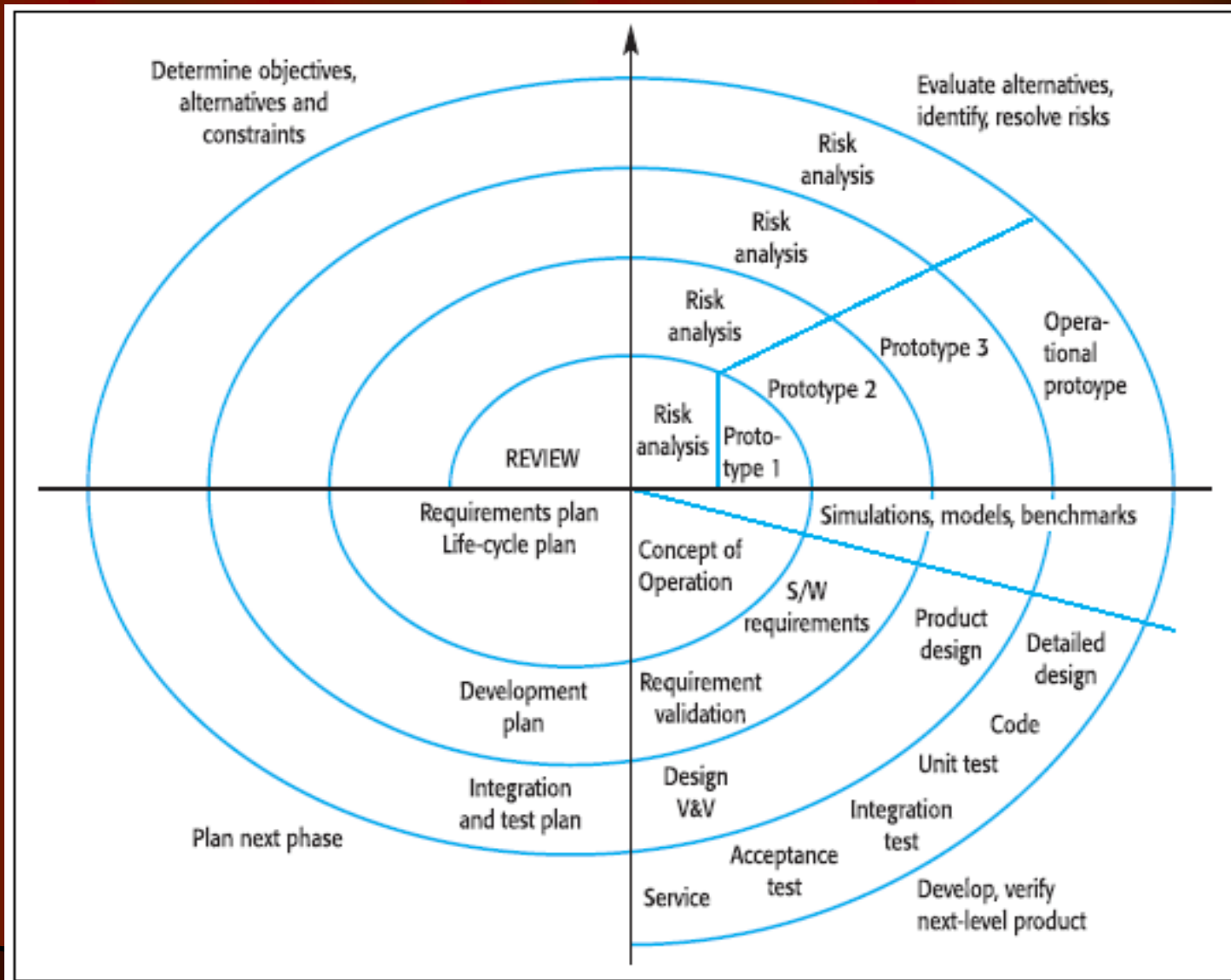
Penjelasan model Inkremen Pressman

- Kombinasikan elemet-element dari waterfall dengan sifat iterasi/perulangan.
- Element-element dalam waterfall dikerjakan dengan hasil berupa produk dengan spesifikasi tertentu, kemudian proses dimulai dari fase pertama hingga akhir dan menghasilkan produk dengan spesifikasi yang lebih lengkap dari yang sebelumnya. Demikian seterusnya hingga semua spesifikasi memenuhi kebutuhan yang ditetapkan oleh pengguna.
- Produk hasil increment pertama biasanya produk inti (core product), yaitu produk yang memenuhi kebutuhan dasar. Produk tersebut digunakan oleh pengguna atau menjalani review/pengecekan detil. Hasil review tersebut menjadi bekal untuk pembangunan pada increment berikutnya. Hal ini terus dikerjakan sampai produk yang komplit dihasilkan.
- Mampu mengakomodasi perubahan secara fleksibel.
- Produk yang dihasilkan pada increment pertama bukanlah prototype, tapi produk yang sudah bisa berfungsi dengan spesifikasi dasar.

Model Increment Sommerville



B. SPIRAL MODEL



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

Setiap Loop dibagi menjadi beberapa sektor :

1. Objective settings (menentukan tujuan): menentukan tujuan dari fase yang ditentukan. Batasan-batasan pada proses dan produk sudah diketahui. Perencanaan sudah disiapkan. Resiko dari proyek sudah diketahui. Alternatif strategi sudah disiapkan berdasarkan resiko-resiko yang diketahui, dan sudah direncanakan.
2. Risk assessment and reduction (Penanganan dan pengurangan resiko): setiap resiko dianalisis secara detil pada sektor ini. Langkah-langkah penanganan dilakukan, misalnya membuat prototype untuk mengetahui ketidakcocokan kebutuhan.

3. Development and Validation (Pembangunan dan pengujian):

Setelah evaluasi resiko, maka model pengembangan sistem dipilih.

- Misalnya jika resiko user interface dominan, maka membuat prototype User Interface.
- Jika bagian keamanan yang bermasalah, maka menggunakan model formal dengan perhitungan matematis,
- Jika masalahnya adalah integrasi sistem model waterfall lebih cocok.

4. Planning : Proyek dievaluasi atau ditinjau-ulang dan diputuskan untuk terus ke fase loop selanjutnya atau tidak. Jika melanjutkan ke fase berikutnya rencana untuk loop selanjutnya.
- Pada model spiral, resiko sangat dipertimbangkan.
 - Resiko adalah sesuatu yang mungkin mengakibatkan kesalahan.
 - Model spiral merupakan pendekatan yang realistik untuk PL berskala besar.
 - Pengguna dan pembangun (Perekayasa) bisa memahami dengan baik software yang dibangun karena setiap kemajuan yang dicapai selama proses dapat diamati dengan baik.
 - Namun demikian, waktu yang cukup panjang mungkin bukan pilihan bagi pengguna, karena waktu yang lama sama dengan biaya yang lebih besar.

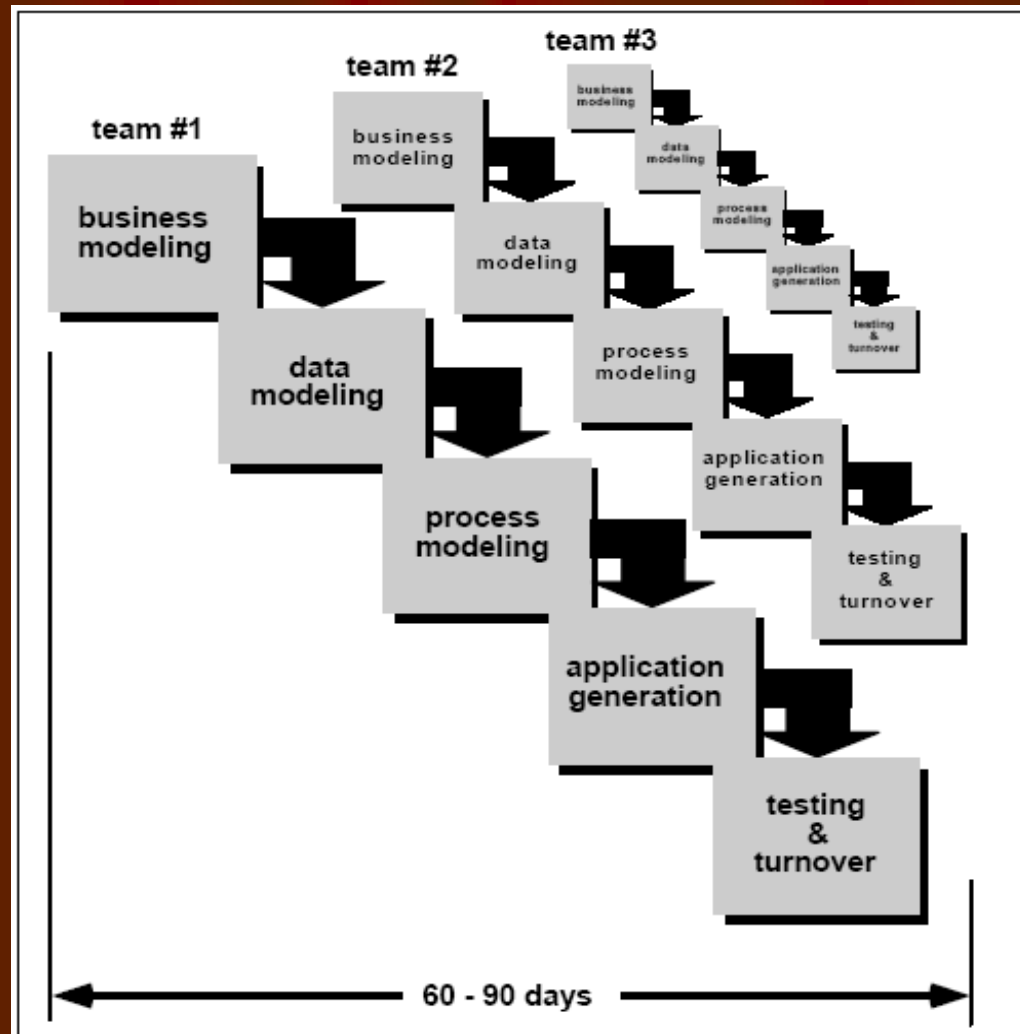
RAD (Rapid Application Development)

- RAD adalah model proses pembangunan PL yang incremental.
- RAD menekankan pada siklus pembangunan yang pendek/singkat.
- RAD mengadopsi model waterfall dan pembangunan dalam waktu singkat dicapai dengan menerapkan component based construction.
- Waktu yang singkat adalah batasan yang penting untuk model ini.
- Jika kebutuhan lengkap dan jelas maka waktu yang dibutuhkan untuk menyelesaikan secara komplit software yang dibuat adalah misalnya 60 sampai 90 hari.

Kelemahan dalam model ini:

- Tidak cocok untuk proyek skala besar
- Proyek bisa gagal karena waktu yang disepakati tidak dipenuhi
- Sistem yang tidak bisa dimodularisasi tidak cocok untuk model ini
- Resiko teknis yang tinggi juga kurang cocok untuk model ini

Gambar Model RAD



- Fase-fase di atas menggambarkan proses dalam model RAD.
 - Sistem dibagi-bagi menjadi beberapa modul dan dikerjakan dalam waktu yang hampir bersamaan dalam batasan waktu yang sudah ditentukan.
-
- Business modelling : menjawab pertanyaan-pertanyaan: informasi apa yang mengendalikan proses bisnis? Informasi apa yang dihasilkan? Siapa yang menghasilkan informasi? Kemana informasi itu diberikan? Siapa yang mengolah informasi? → kebutuhan dari sistem
 - Data modelling: aliran informasi yang sudah didefinisikan, disusun menjadi sekumpulan objek data. Ditentukan karakteristik/atribut dan hubungan antar objek-objek tersebut → analisis kebutuhan dan data
 - Process Modelling : objek data yang sudah didefinisikan diubah menjadi aliran informasi yang diperlukan untuk menjalankan fungsi-fungsi bisnis.
 - Application Generation: RAD menggunakan component program yang sudah ada atau membuat component yang bisa digunakan lagi, selama diperlukan.
 - Testing and Turnover: karena menggunakan component yang sudah ada, maka kebanyakan component sudah melalui uji atau testing. Namun component baru dan interface harus tetap diuji.

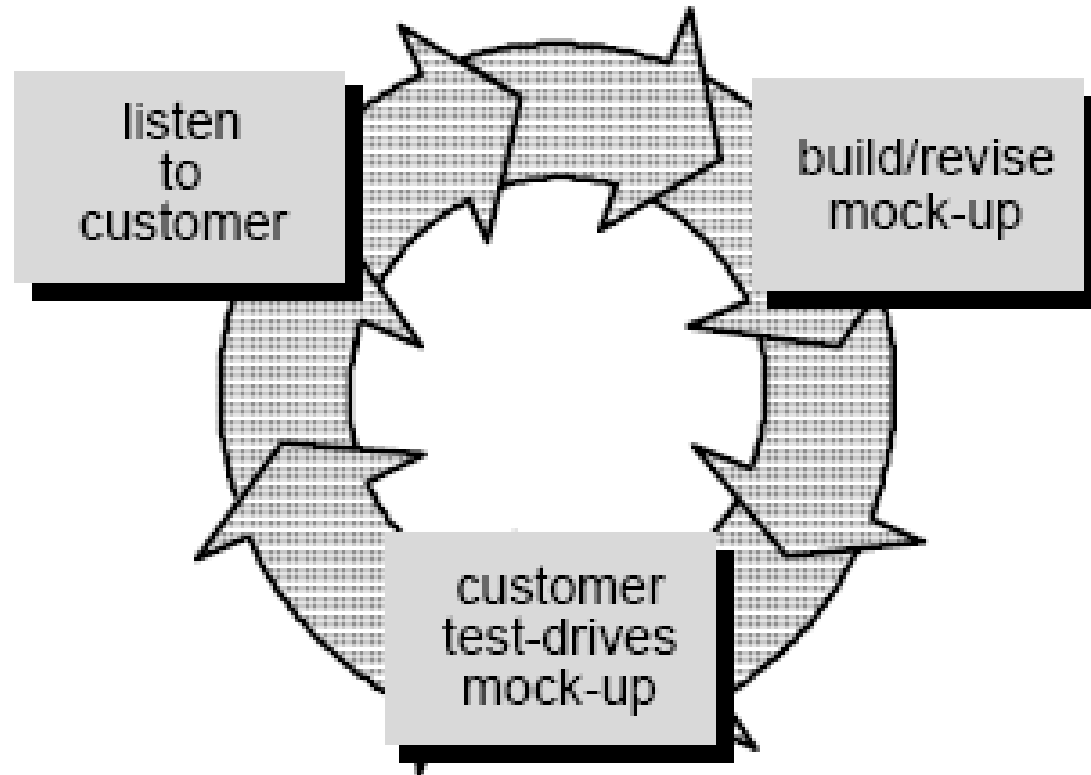
Prototyping Model

- Kadang-kadang klien hanya memberikan beberapa kebutuhan umum software tanpa detail input, proses atau detail output.
- Di lain waktu mungkin dimana tim pembangun (developer) tidak yakin terhadap efisiensi dari algoritma yang digunakan, tingkat adaptasi terhadap sistem operasi atau rancangan form user interface.
- Ketika situasi seperti ini terjadi model prototyping sangat membantu proses pembangunan software.

Proses pada model prototyping yang bisa dijelaskan sebagai berikut:

- Pengumpulan kebutuhan: developer dan klien bertemu dan menentukan tujuan umum, kebutuhan yang diketahui dan gambaran bagian-bagian yang akan dibutuhkan berikutnya. Detil kebutuhan mungkin tidak dibicarakan disini, pada awal pengumpulan kebutuhan
- Perancangan : perancangan dilakukan cepat dan rancangan mewakili semua aspek software yang diketahui, dan rancangan ini menjadi dasar pembuatan prototype.
- Evaluasi prototype: klien mengevaluasi prototype yang dibuat dan digunakan untuk memperjelas kebutuhan software.

Gambar model Prototype



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

Masalah2 yg ada pada Prototype Model :

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.
2. 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.