Bab 3

Perspektif Model Proses

Tujuan Pembelajaran Umum

Menjelaskan Perangkat Lunak Sebagai Proses

Tujuan Pembelajaran Khusus

Mampu Menjelaskan Perspektif Model Proses

Model Proses (*Process Model)*

Model proses merupakan gambaran (kerangka kerja) yang merepresentasikan proses dalam RPL agar mudah dipahami dan proses dapat dilakukan sesuai dengan aturannya. Berbagai macam model telah diajukan menyesuaikan dengan berbagai macam kondisi yang mungkin dalam pembangunan perangkat lunak. Beberapa model proses dalam RPL adalah sebagai berikut:

Model Proses Perangkat Lunak

Mencoba untuk mengatur siklus hidup perangkat lunak dengan terlibat

- kegiatan mendefinisikan dalam produksi perangkat lunak
- urutan kegiatan dan hubungan mereka

Tujuan dari proses perangkat lunak

- Menghasilkan standarisasi, prediktabilitas, produktivitas, kualitas produk yang tinggi, kemampuan untuk merencanakan waktu dan kebutuhan anggaran
- Mengkode program dan perbaikannya

Pendekatan awal

- Menulis kode
- Perbaiki untuk menghilangkan kesalahan yang telah terdeteksi, untuk meningkatkan fungsionalitas yang ada, atau untuk
    - menambahkan fitur baru
- Kesulitan sumber kesulitan dan kekurangan
    - kemungkinan untuk memprediksi
    - kemungkinan untuk mengelola

Model diperlukan dapat dilihat gejala kekurangan : krisis perangkat lunak

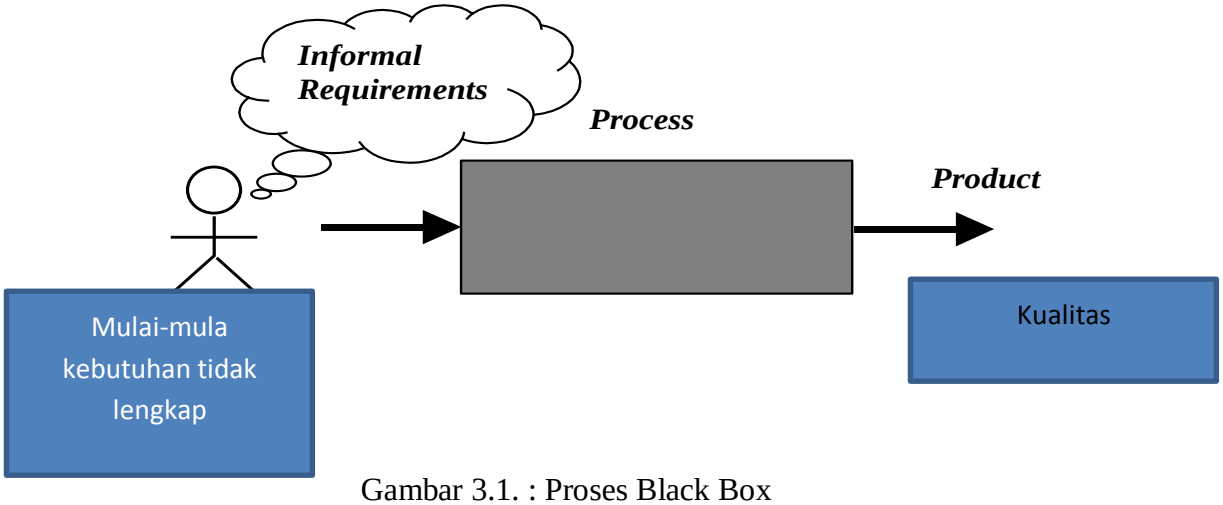
- waktu yang dijadwalkan dan biaya melebihi
- harapan pengguna tidak terpenuhi
- kualitas buruk

Tujuan Model Proses

A. Boehm 1988

"menentukan urutan tahapan yang terlibat dalam pengembangan perangkat lunak dan evolusi,
dan untuk menetapkan transisi kriteria untuk kemajuan dari satu tahap ke tahap berikutnya.
Ini termasuk kriteria penyelesaian untuk tahapan saat ini ditambah pilihan kriteria dan kriteria untuk masuk tahap berikutnya. Jadi model alamat proses proyek perangkat lunak pertanyaan berikut
Apa yang harus kita lakukan selanjutnya?
Berapa lama kita akan terus melakukannya? "

Sebuah Proses Black Box

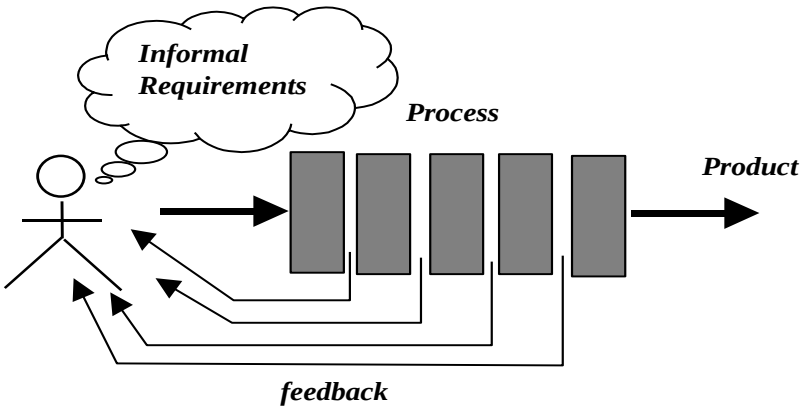


Gambar 3.1. : Proses Black Box

Permasalahan

- Asumsinya adalah bahwa persyaratan dapat sepenuhnya dipahami sebelum pembangunan
- Interaksi dengan pelanggan terjadi hanya pada awal (persyaratan) dan akhir (setelah melahirkan)
- Sayangnya asumsi hampir tidak pernah dipegang

Sebuah proses White Box



Gambar 3.2. : Proses White Box

Kelemahan

- Mengurangi risiko dengan meningkatkan visibilitas
- Memungkinkan perubahan proyek sebagai proyek berlangsung berdasarkan umpan balik dari pelanggan

Kegiatan utama didalam memproduksi perangkat lunak :

- Mereka harus dilakukan secara independen dari model
- Model ini hanya mempengaruhi aliran antara kegiatan
- Ukuran dan nilai ekonomi dari aplikasi perangkat lunak yang diperlukan "proses yang sesuai

Model-model perspektif

Model proses preskriptif menganjurkan pendekatan tertib rekayasa perangkat lunak

Yang mengarah ke beberapa pertanyaan...

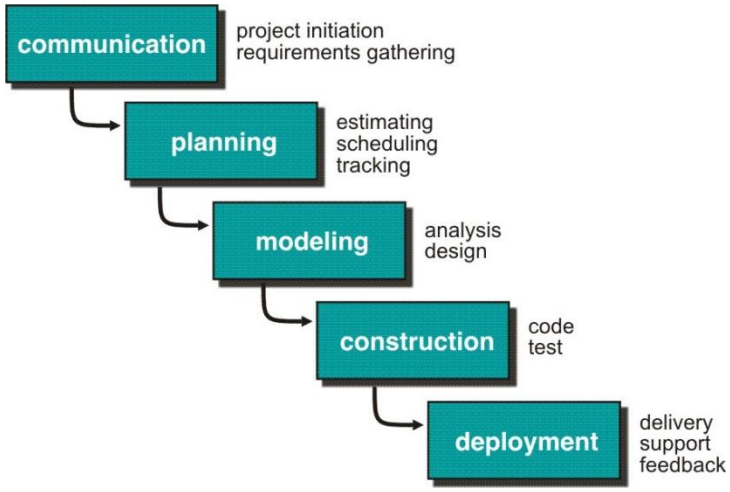
Jika model proses preskriptif berusaha untuk struktur dan ketertiban, mereka pantas untuk dunia perangkat lunak yang tumbuh subur pada perubahan ?.
Namun, jika kita menolak model proses tradisional (dan urutan mereka menyiratkan) dan penggantinya dengan sesuatu yang kurang terstruktur, kita membuat tidak mungkin untuk mencapai koordinasi dan koherensi dalam pekerjaan perangkat lunak?

Model Waterfall

Model Waterfall merupakan model dasar dari perangkat lunak.

Beberapa kelebihan model ini adalah :

1. Titik awal dan titik akhir yang eksplisit
2. Setiap tahapan didefinisikan dengan jelas
3. Setiap akhir suatu tahap, disesuikan kembali dengan tahap sebelumnya, sehingga kesalahan yang mungkin terjadi bisa ditemukan dan diselesaikan lebih dini.
4. Incremental release, lingkup kerja untuk tahapan-tahapan berikutnya menjadi lebih kecil, dan tugas yang lebih mudah. Jika tahap awal dilakukan dengan benar maka akan mempermudah tahap berikutnya.



Gambar 3.3. : "Model Air Terjun"

Aktivitas setiap tahapannya adalah :

1. *Communication* : Inisiasi proyek dan penyiapan kebutuhan seluruh elemen system
2. *Planning* : Merencanakan pekerjaan perangkat lunak (estimasi : memperkirakan pembiayaan, tenaga, waktu; penjadwalan proyek dan penentuan pelaksanaan.
3. *Modelling* : melakukan analisis dan merancang meliputi (rancang struktur data, arsitektur perangkat lunak, rincian procedural, karakteristik antar muka)
4. *Construction* : Mengkode program dan melakukan testing program
5. *Deployment* : implementasi program, dukungan dan umpan balik pengguna.

Asumsi-asumsi Model Waterfall

1. Persyaratan yang diketahui di awal pelaksanaan.
2. Persyaratan tidak terselesaikan, implikasi berisiko tinggi misalnya, risiko karena pilihan COTS, biaya, jadwal, kinerja, keselamatan, keamanan, user interface, dampak organisasi
3. Sifat dari persyaratan tidak akan berubah banyak
   * Selama pembangunan; selama evolusi

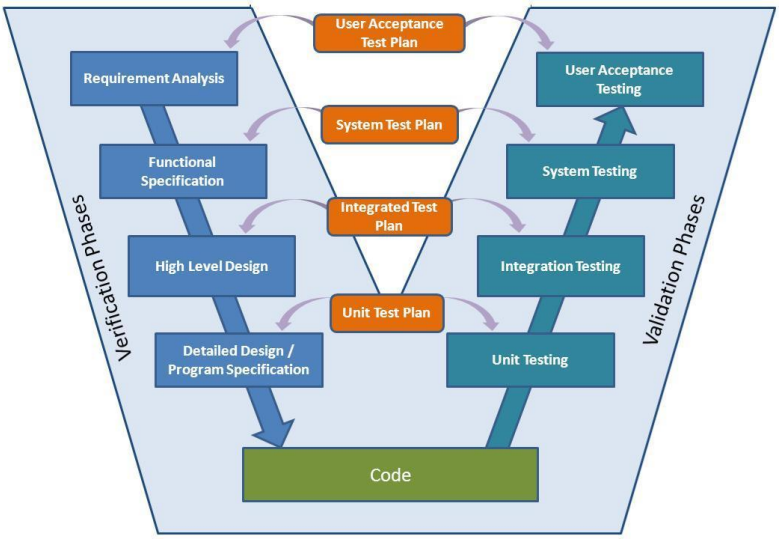
4. Persyaratan yang kompatibel dengan semua harapan sistem stakeholder kunci '
    * misalnya, pengguna, pelanggan, pengembang, pengelola, investor
5. Arsitektur tepat untuk menerapkan persyaratan dipahami dengan baik.
6. Ada waktu kalender yang cukup untuk melanjutkan berurutan.

Model air terjun adalah paradigma rekayasa perangkat lunak yang paling luas dipakai dan paling tua. Tetapi kritik terhadap paradigma tersebut menyebabkan banyak pihak yang mempertanyakan kehandalannya. Masalah-masalah yang timbul ketika model tersebut diterapkan adalah :

1. Meskipun dalam prosesnya model ini bisa mengakomodasi iterasi, tetapi tidak dilakukan secara langsung, akibatnya perubahan-perubahan dapat menyebabkan keraguan pada saat tim proyek berjalan.
2. Kadang-kadang pelanggan sulit untuk menyatakan semua kebutuhannya secara eksplisit, sehingga sulit untuk mengakomodasi ketidakpastian tersebut.
3. Pelanggan harus bersikap sabar, karena masa pakai dari program tidak akan diperoleh sampai akhir waktu proyrk berakhir. Akibatnya bisa saja terdapat kesalahan yang tidak tedeteksi sampai program tersebut tiba masanya untuk dikaji ulang.
4. Pengembang sering melakukan penundaan yang tidak perlu, karena seringnya beberapa anggota tim proyek harus menunggu anggota lain untuk melengkapi tugas yang saling ketergantungan.

Model V

Model V-[2] merupakan proses pengembangan perangkat lunak (juga berlaku untuk pengembangan perangkat keras) yang dapat dianggap sebagai perpanjangan dari model air terjun. Alih-alih bergerak turun secara linear, langkah-langkah proses yang ditekuk ke atas setelah fase coding, untuk membentuk V bentuk khas. V-Model menunjukkan hubungan antara setiap fase dari siklus hidup pengembangan dan fase terkait pengujian. Sumbu horisontal dan vertikal mewakili waktu atau proyek kelengkapan (kiri ke kanan) dan tingkat abstraksi (yang kasar-butiran abstraksi teratas), masing-masing.

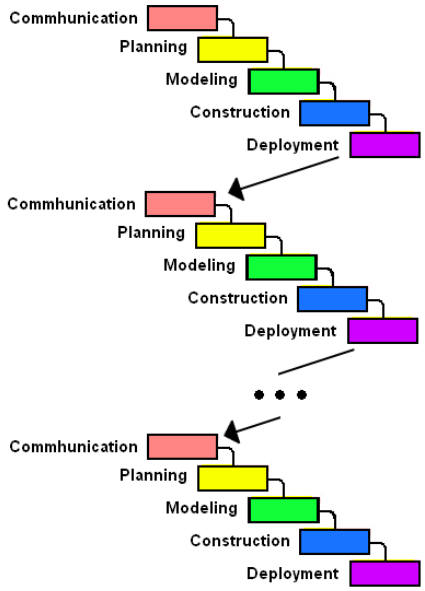


Gambar 3.4. : Model V

Model Incrimental adalah (Incremental Models, Rapid Application Development (RAD)); adanya tahapan-tahapan dalam pengembangan P/L.

Model Incrimental

Membangun model Incremental merupakan metode pengembangan perangkat lunak di mana model ini dirancang, diimplementasikan dan diuji secara bertahap (sedikit lebih ditambahkan setiap waktu) sampai produk selesai. Ini melibatkan baik pengembangan dan pemeliharaan. Produk didefinisikan selesai ketika memenuhi semua persyaratan. Model ini menggabungkan unsur-unsur dari model air terjun dengan iterasi filosofi prototyping.



Gambar 3.5. : Model Incrimental

Model RAD

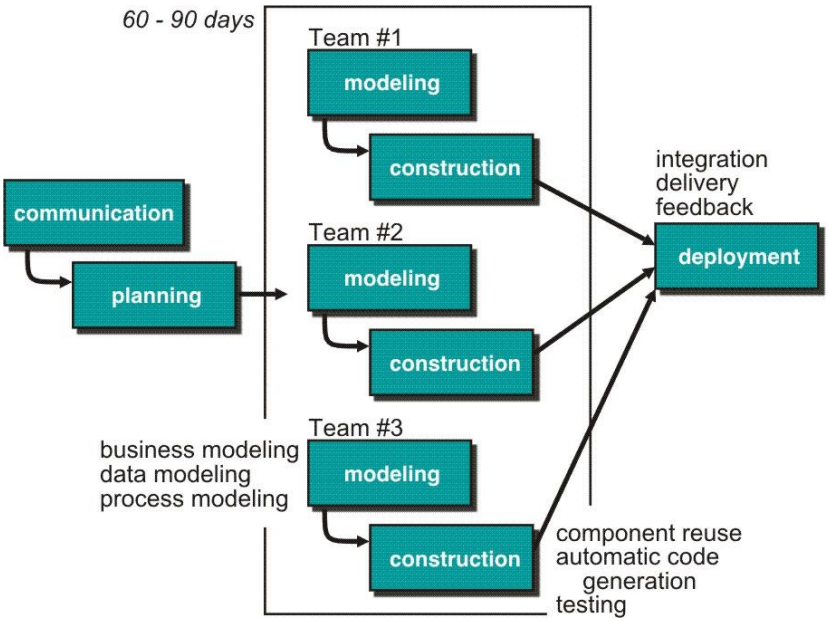
RAD adalah merupakan model proses pengembangan perangkat lunak adaptasi kecepatan tinggi dari model sekuensial linier yang menekankan siklus perkembangan yang sangat pendek. Perjalanan pengembangannya sangat cepat dengan menggunakan pendekatan konstruksi berbasis komponen, yang memungkinkan tim pengembang bisa menciptakan sistem fungsional secara utuh dalam waktu 60 s.d. 90 hari. Pada umumnya digunakan pada aplikasi sistem konstruksi.

Pendekatan RAD melingkupi fase-fase sebagai berikut :

1. *Businnes modelling*. Pemodelan dari aliran informasi diantara fungsi-fungsi bisnis.
2. *Data modelling*. Mengidentifikasi serangkaian objek data yang dibutuhkan dan karakteristik masing-masing objek tersebut, serta mendefinisikan hubungan antara objek-objek tersebut.
3. *Process modelling*. Mentransformasikan hasil data modelling untuk mencapai aliran informasi yang perlu bagi implementasi fungsi-fungsi prosesnya. Gambaran proses dibuat untuk menambah, memodifikasi, menghapus, atau mendapatkan kembali sebuah objek data.
4. Aplication generation. RAD mengasumsikan pemakaian teknik generasi keempat (4GL), lebih banyak memakai komponen program yang sudah ada, juga menciptakan komponen yang bisa dipakai lagi.
5. Testing and turnover. Karena proses RAD menekankan pada pemakaian kembali, maka setiap komponen baru harus diuji untuk mengurangi keseluruhan waktu pengujian

Kelemahan model RAD :

1. Bagi proyek yang besar tetapi berskala, RAD memerlukan sumber daya manusia yang memadai untuk menciptakan jumlah tim RAD yang baik.
2. RAD menuntut pengembang dan pelanggan memiliki komitmen tinggi di dalam aktivitas pengembangan.



Gambar 3.6. : Model RAD

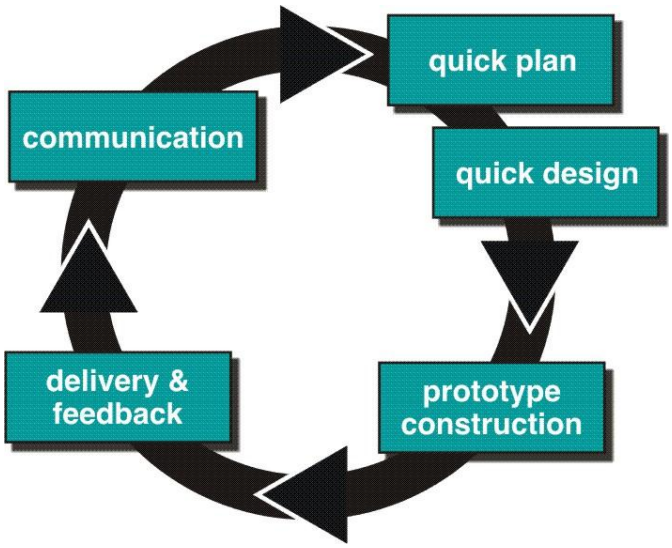
Model Proses Evolusioner (Prototyping, Spiral Model, Concurrent Development Model); adanya perputaran yang berulang dalam pengembangan P/L.

Model Prototyping

Sering kali seorang pelanggan mendefinisikan serangkaian umum bagi perangkat lunak yang dibutuhkan, tetapi tidak mengidentifikasi kebutuhan output, pemrosesan, maupun input secara detail. Pada kasus lain, pengembang tidak memiliki kepastian terhadap efisiensi algoritma, kemampuan penyesuaian dari sebuah sistem operasi, atau bentuk-bentuk yang harus dilakukan oleh interaksi manusia dan mesin. Dalam hal ini, paradigma prototipe menawarkan pendekatan yang terbak.

Paradigma ini dimulai dengan mengumpulkan kebutuhan. Pengembang dan pelanggan bertemu untuk mendefinisikan obyektif keseluruhan dari perangkat lunak. Kemudian dilakukan perancangan kilat yang berfokus pada penyajian dari aspek-aspek perangakt lunak yang akan nampak oleh pelanggan/pemakai (misal format input dan outputnya). Perancangan kilat tersebut membawa kepada konstruksi prototipe. Prototipe ini dievaluasi oleh pelanggan/pemakai dan dipakai untuk menyaring kebutuhan pengembangan perangkat lunak yang dibutuhkan. 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 dilakukan.

Prototipe bisa berfungsi sebagai "sistem awal". Tetapi pada beberapa proyek yang dibangun dengan prototipe, saat penggunaan pertama sistem awal yang baru dibangun tersebut, mungkin akan terasa terlalu pelan, terlalu besar, janggal dalam pemakaian, atau bahkan tiga hal tersebut semua terjadi. Jika terjadi demikian maka tidak ada pilihan lain kecuali memulai lagi untuk membangun versi yang baru dimana masalah yang muncul bisa diselesaikan.



Gambar 3.7. : Model Prototyping

Model Spiral

Model spiral adalah model proses pembangkit risiko didorong untuk proyek-proyek perangkat lunak. Berdasarkan pola risiko yang unik dari sebuah proyek tertentu,

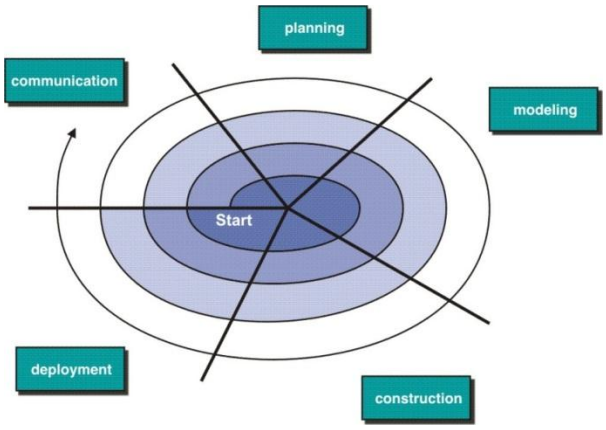
model spiral memandu tim untuk mengadopsi elemen dari satu atau lebih model proses, seperti tambahan, air terjun, atau prototipe evolusioner.

Bemtuk lebih sederhana dibanding model air terjun ditambah analisis resiko
Setiap tahap didahului dengan

- alternatif
- analisis risiko

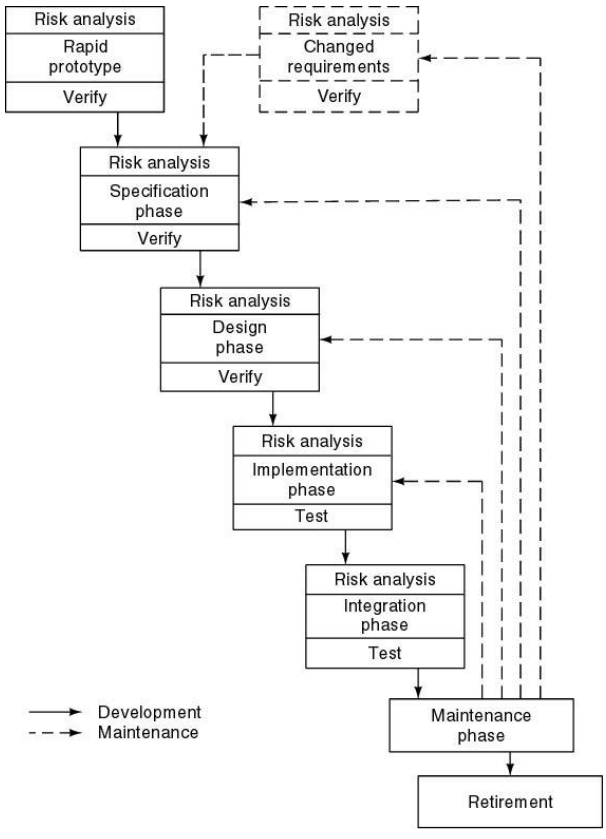
Ikuti setiap tahap oleh

- Evaluasi
- Perencanaan fase berikutnya



Gambar 3.8. : Model Spiral

Model Spiral :

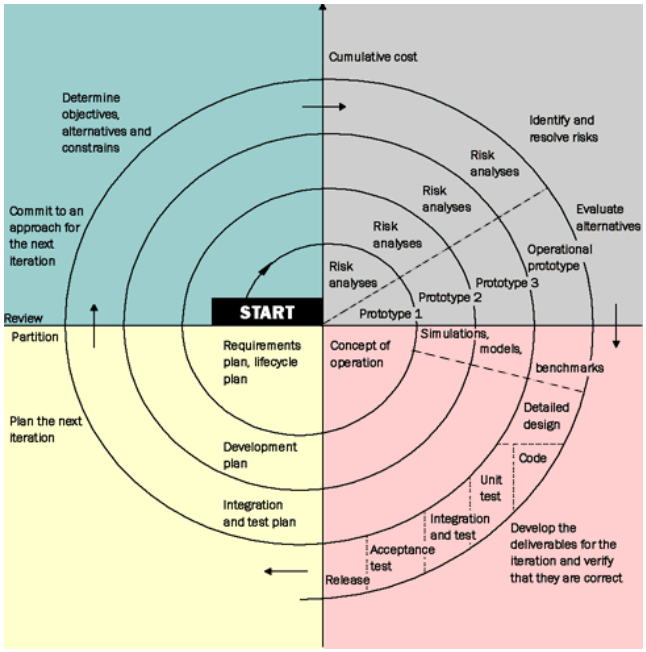


Gambar 3.9. : Model Spiral

Model Full Spiral

Dimensi radial: biaya kumulatif sampai saat ini

Dimensi sudut: kemajuan melalui spiral



Gambar 3.10 : Model Full Spiral

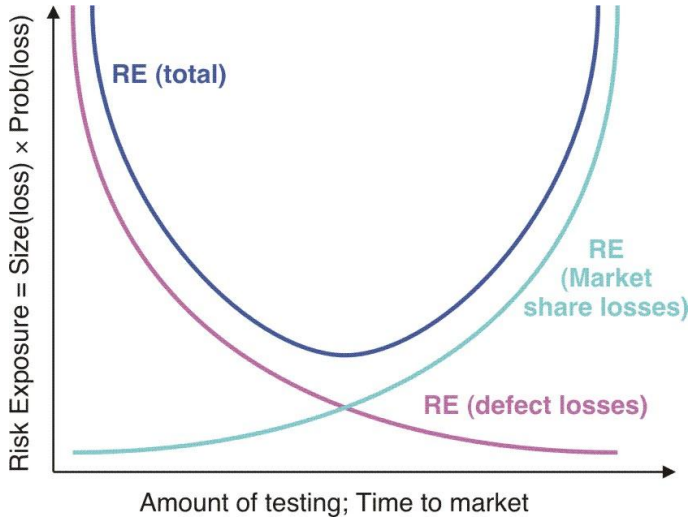
Analisis pada model spiral
kekuatan

- Mudah untuk menilai berapa banyak untuk menguji
- Tidak ada perbedaan antara pembangunan, pemeliharaan

kelemahan

- Hanya untuk perangkat lunak skala besar
- Hanya untuk perangkat lunak internal (in-house)

Risk Exposures



Gambar : 3.11

Model Proses Uniffied

Sebuah proses software yaitu:

- digunakan use case
- arsitektur terpusat
- iteratif dan incremental

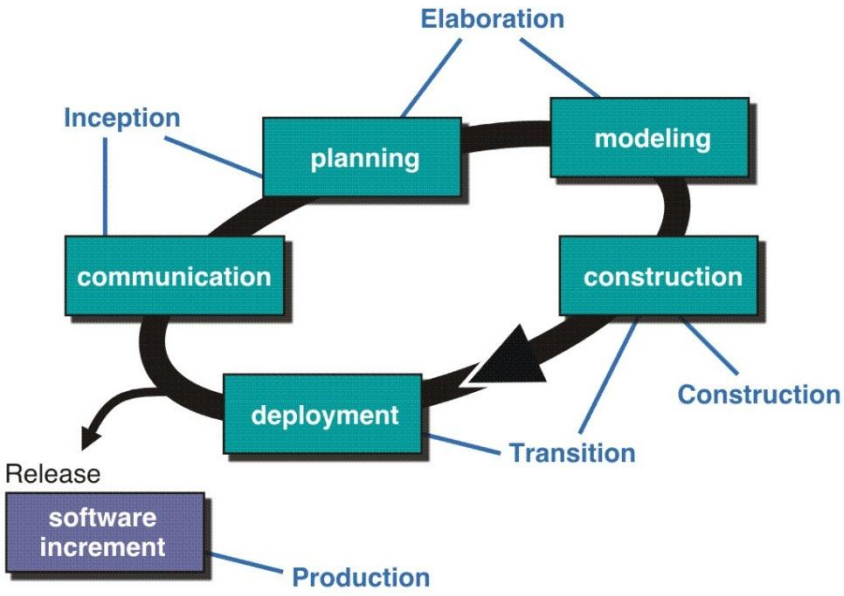
Berkaitan erat dengan

- Unified Modeling Language (UML)

Model Unified Proses

The Unified Process bukan hanya suatu proses, melainkan suatu kerangka extensible yang harus disesuaikan untuk organisasi atau proyek-proyek tertentu. Rational Unified Process adalah, sama, kerangka disesuaikan. Akibatnya sering tidak mungkin untuk mengatakan apakah merupakan penyempurnaan dari proses ini berasal dari UP atau dari RUP, sehingga nama-nama cenderung digunakan secara bergantian.

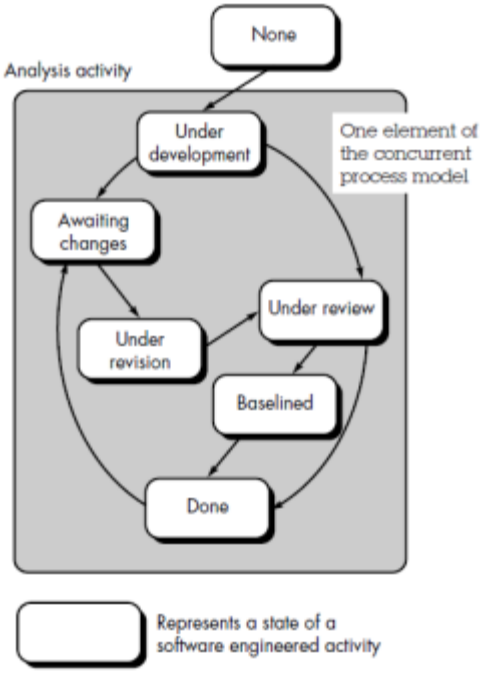
Nama Unified Process sebagai lawan Rational Unified Proses umumnya digunakan untuk menggambarkan proses generik, termasuk unsur-unsur yang paling umum untuk perbaikan. Unified Nama proses juga digunakan untuk menghindari masalah potensi pelanggaran merek dagang sejak Rational Unified Proses dan RUP adalah merek dagang dari IBM. Buku pertama untuk menggambarkan proses itu berjudul The Unified Proses Pengembangan Perangkat Lunak (ISBN 0-201-57169-2) dan diterbitkan pada tahun 1999 oleh Ivar Jacobson, Grady Booch dan James Rumbaugh. Sejak itu berbagai penulis tidak terafiliasi dengan Software Rasional telah menerbitkan buku dan artikel menggunakan nama Unified Process, sedangkan penulis berafiliasi dengan Software Rasional telah disukai nama Rational Unified Process.



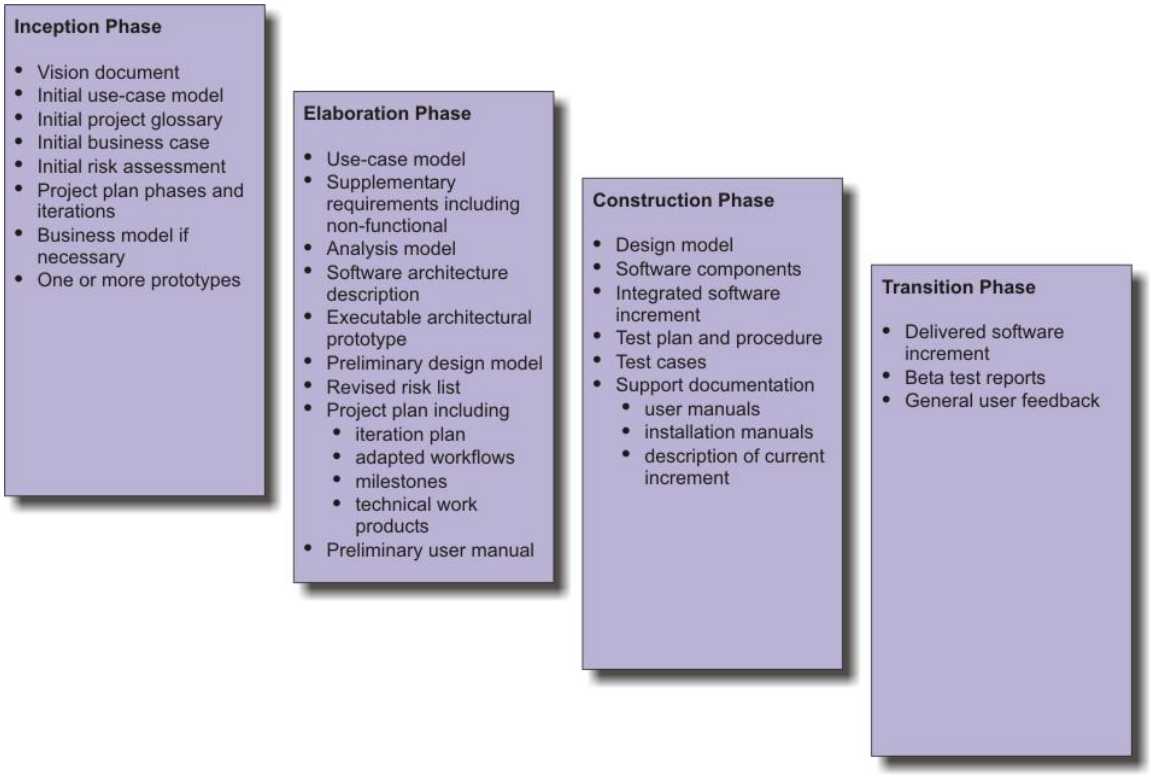
Gambar 3.12. : Model Unified Process

Model Concurent Development

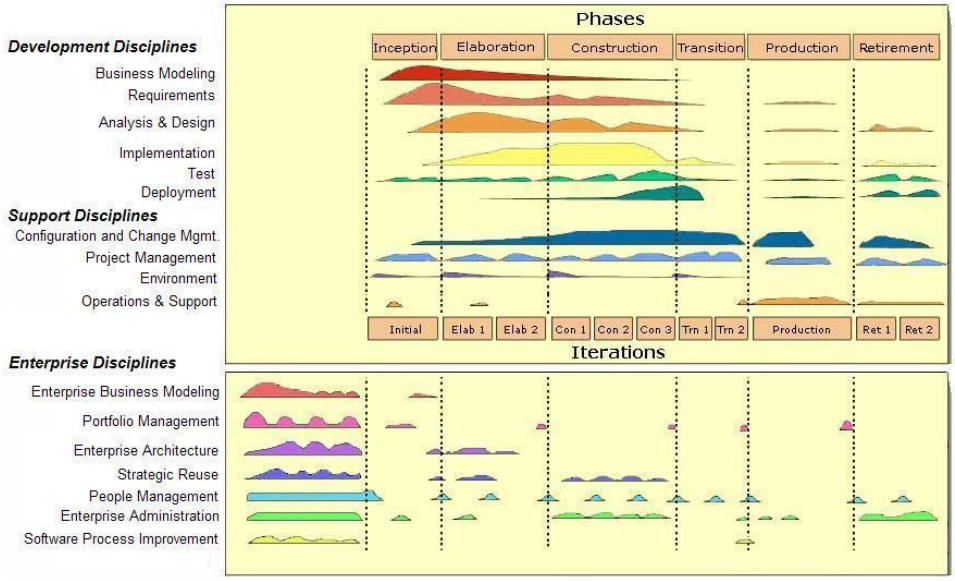
Concurrent engineering adalah metodologi kerja berdasarkan paralelisasi tugas (yaitu melakukan tugas secara bersamaan). Hal ini mengacu pada pendekatan yang digunakan dalam pengembangan produk di mana fungsi rekayasa desain, teknik manufaktur dan fungsi lainnya yang terintegrasi untuk mengurangi waktu yang telah berlalu diperlukan untuk membawa produk baru ke pasar.



Gambar 3.13. : Model Concurent Development



Gambar 3.14. : UP Work Products



Gambar 3.15. : Life Cycle Unified Process

Sinkronisasi Model dan Stabilisasi

- Pada akhir hari dilakukan sinkronisasi (test dan debug)
- Pada akhir pembangunan distabilkan (freeze membangun)
- Komponen selalu bekerja sama
    - Dapatkan wawasan awal ke pengoperasian produk

Siklus hidup model orientasi objek

Perlu untuk iterasi di dalam dan di antara fase

- Model Fountain
- Rekursif / siklus hidup paralel
- Round-trip gestalt
- Proses pengembangan perangkat lunak terpadu

Semua menggabungkan beberapa bentuk

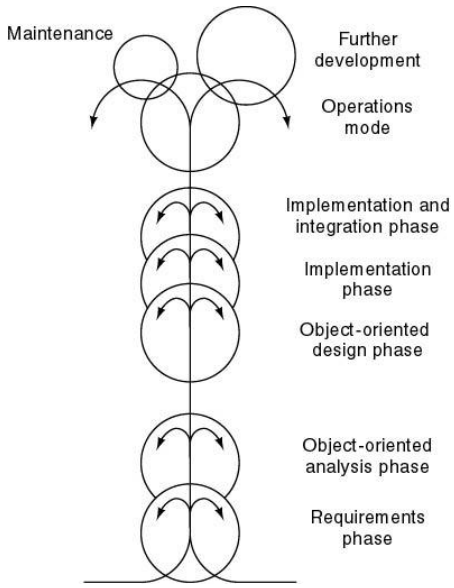
- perulangan
- paralelisme
- pengembangan Incremental

bahaya

- CABTAB

Model Fountain

- Fitur
- Tumpang tindih (paralelisme)
- Arrows (iterasi)
- Pemeliharaan Lingkaran yang lebih kecil

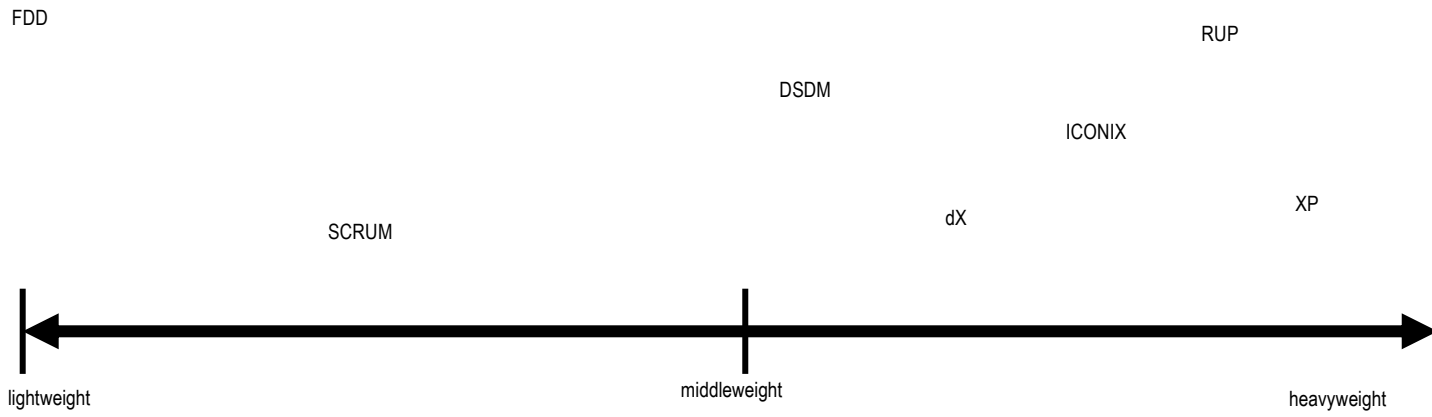


Gambar 3.16 : Model Fountain

Model-model proses spesialisasi

- Component Based Development; mengedepankan konsep reusability (dalam bentuk komponen)
- Formal Method; menggunakan model matematis untuk menghilangkan ambiguitas dan inkonsistensi
- Aspect Oriented Development; mengedepankan *separation of concern* untuk fungsi yang tersebar.

Spektrum PL



Latihan :

1. Apa yang dimaksud dengan model proses perangkat lunak ?
2. Coba jelaskan hubungan proses perangkat lunak dengan metodologi perangkat lunak ?
3. Apa kelebihan dan kekurangan antara proses black box dan white box ?
4. Coba jelaskan pengembangan model v ?
5. Kelebihan model incremental dengan waterfall model ?
6. Apa yang dimaksud dengan model full spiral model dan jelaskan langkah-langkahnya ?
7. Jelaskan metodologi pengembangan UML ?

Bab 4.

Pengembangan Agile

Tujuan Pembelajaran Umum

Menjelaskan Perangkat Lunak Sebagai Proses

Tujuan Pembelajaran Khusus

Mampu Menjelaskan Proses Agile

Pandangan umum model proses Agile

Jika pada model proses yang diajukan sebelumnya berfokus pada pengembangan yang mengikuti kerangka kerja (*framework*) yang terdefinisi pada model prosesnya, beberapa ilmuwan berpendapat bahwa perlunya*Agile View* dalam pendefinisian proses dalam RPL. Ide dasar dari *Agile View* adalah fleksibilitas proses terhadap perubahan. Dalam hal ini, model proses yang ditawarkan dari *Agile View* berasumsi bahwa selama proses pengembangan P/L akan terjadi banyak perubahan yang terjadi karena perubahan kebutuhan *customer*.

Agile Modeling adalah suatu chaordic, yaitu suatu proses berdasarkan praktek untuk memodelkan dan mendokumentasikan suatu sistem yang berbasiskan software secara efektif. Agile Modeling juga dikatakan sebagai suatu kumpulan dari kebiasaan-kebiasaan berdasarkan beberapa nilai dan prinsip-prinsip teknik software yang terpercaya. Selain itu Agile Modeling merupakan sebuah pendekatan light-weight yang mempertinggi usaha pemodelan dan pendokumentasian untuk proses software lainnya seperti XP dan RUP. Agile model lebih efektif daripada model tradisional yang tidak cukup bagus dan tidak sempurna. Kita bisa memakai pendekatan Agile modeling pada requirements, analysis, architecture, dan design. Agile modeling bukan suatu proses yang bersifat menentukan, dengan kata lain tidak mendefinisikan prosedur secara detil untuk bagaimana membuat suatu tipe model yang telah diberikan, meskipun terdapat cara bagaimana untuk menjadi suatu modeler yang efektif. Agile modeling mengenali bahwa meskipun semua factor yang ada adalh penting, tetapi jika kita menggunakan pendekatan agile untuk memodelkan sesuatu maka kita harus focus pada:

- ***Individual dan interaction*** **dari pada proses proses dan** tools
- ***Kinerja software*** daripada dokumentasi yang lengkap
- ***Kolaborasi customer*** **daripada negosiasi kontrak**
- ***Reaksi untuk mengubah*** **daripada mengikuti suatu rencana**

Apa itu lincah (agility) ?

- Respon yang efektif (cepat dan adaptif) untuk mengubah
- Komunikasi yang efektif di antara semua pemangku kepentingan
- Melibatkan konsumen pada tim

- Pengorganisasian tim sehingga mengendalikan pekerjaan yang dilakukan

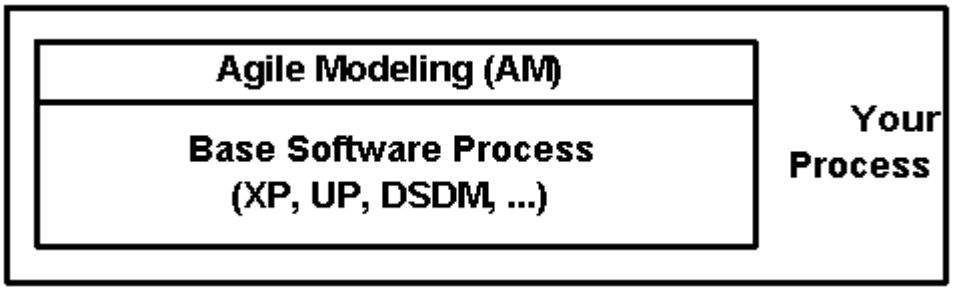
Tujuan dari Agile Modeling sendiri antara lain adalah:

- Untuk mendefinisikan dan menunjukkan bagaimana mempraktekkan, sekumpulan nilai, prinsip prinsip dan praktek yang lebih efektif, suatu modeling yang ringan. Rahasia dari Agile Modeling bukan teknik modelingnya sendiri, seperti use case model, class model, data model, or user interface model tetapi bagaimana mereka diaplikasikan.
- Selain itu adalah untuk menempatkan persoalan bagaimana cara untuk mengaplikasikan teknik modeling pada suatu proyek software yang menggunakan pendekatan agile seperti eXtreme Programming (XP), Dynamic Systems Development Method (DSDM), atau SCRUM. Mengambil komentar Kent Beck dari eXtreme Programming Explained (XPE) "kita akan terus menerus memperbaiki design dari sistem, mulai dari suatu awal yang sederhana. Kita akan menghapus fleksibilitas yang tidak terbukti berguna." Berkebalikan dengan klaim dari beberapa kritikus eXtreme Programming bahwa akan memakan banyak waktu untuk memodelkan pada kenyataanya ketika menggunakan suatu pendekatan eXtreme Programming, tetapi jika kita tidak mempunyai pilihan lain. Terkadang lebih produktif bagi developer untuk menggambar bubble dan garis daripada memikirkan suatu ide, atau untuk membandingkan beberapa pendekatan yang berbeda untuk mengatasi permasalahan, daripada memulai membajak kode yang sederhana.
- Untuk menempatkan permasalahan bagaimana memodelkan sesuatu dengan lebih efektif pada suatu proyek Unified Process (UP), instansiasi yang biasa yang meliputi Rational Unified Process (RUP) dan Enterprise Unified Process (EUP). EUP meliputi beberapa workflow yang berorientasi modeling pada requirement, Business Modeling, Analysis & Design, Infrastructure Management dimana ketika mengaplikasikan pada suatu nilai akan lebih memberatkan. Agile Modeling membantu untuk meningkatkan keefektifan dari usaha memodelkan pada suatu proyek Unified Process atau beberapa proyek lain yang terjadi.

Ruang lingkup Agile Modeling.

Konsep yang penting untuk dimengerti dari suatu Agile Modeling adalah bahwa ini bukanlah proses software yang lengkap. Agile Modeling berkonsentrasi pada modeling dan dokumentasi yang efektif. Tidak meliputi aktifitas pemrograman, meskipun mengatakan untuk membuktikan model kita dengan suatu code (coding). Tidak meliputi aktifitas testing juga, meskipun mengatakan untuk mempertimbangkan model yang bisa lolos testing. Tidak mencakup manajemen proyek, system deployment, system operasi, system support, dan sebagainya. Karena Agile Modeling fokus pada suatu porsi dari semua proses software yang harus memakai yang lain, proses full-fledged seperti

eXtreme Programming (XP), Dynamic Systems Development Method (DSDM), SCRUM atau Unified Process (UP) seperti yang telah disebutkan pada tujuan dari Agile Modeling. Gambar1 berikut menggambarkan konsep ini. Kita mulai dengan proses dasar, seperti XP atau UP atau mungkin proses kita sendiri yang sudah ada, kemudian dilanjutkan dengan agile Modeling (diharapkan menggunakan semua Agile Modelling) seperti teknik lain yang tepat untuk membentuk proses kita yang merefleksikan kebutuhan yang unik.



Gambar 4.1. AM enhances other software processes.

Nilai dari Agile Modeling meliputi nilai dari eXtreme Programming co*mmunication*, *simplicity*, *feedback* dan *courage*, dilengkapi dengan nilai kelima yaitu *humility*:

- **Communication**. Model meningkatkan komunikasi antara tim kita dengan stakeholder proyek kita sebaik antara developer dan tim kita.
- **Simplicity**. Penting bahwa developer mengerti bahwa model mudah untuk disederhanakan baik software dan proses software, lebih mudah untuk mengeksplorasi ide, dan meningkatkannya sesuai dengan peningkatan pemahaman, dengan menggambar diagram atau menulis puluhan bahkan ribuan baris kode.
- **Feedback**. Kent Beck mengatakan bahwa yang terbaik pada Extreme Programming Explained:"optimis adalah suatu resiko dalam pekerjaan pemrograman,dan feedback adalah pengobatannya." Dengan mengkomunikasikan ide kita melalui deigram, kita akan cepat mendapatkan feedback, sehingga kita bisa melakukan sesuai nasehat itu.
- **Courage**. Courage penting karena kita harus membuat keputusan yang penting dan bisa untukn merubah perintah dengan pembuangan atau pembuatan kembali pekerjaan kita ketika beberapa dari keputusan kita terbukti tidak mencukupi.
- **Humility**. Developer yang terbaik mempunyai kerendahan hati untuk mengenali bahwa mereka tidak tahu semuanya, bahwa developer bawahan mereka, customer, dan pada kenyataannya semua stakeholder proyek juga mempunyai daerah kemampuan masing masing dan mempunyai nilai untuk menambahkan sesuatu pada proyeknya. Suatu pendekatan yang efektif adalah untuk mengasumsikan bahwa semua orang terlibat dengan proyek kita mempunyai nilai sama dan harus dihormati.

Kesimpulannya, kunci untuk modeling yang sukses adalah dengan mempunyai komunikasi yang efektif antara semua yang terlibat dalam proyek, untuk

mengembangkan solusi yang paling sederhana yang mungkin sesuai dengan semua kebutuhan kita, untuk mendapatkan feedback sesuai dengan yang kita kerjakan dengan cepat dan sesering mungkin, untuk mendapatkan keberanian untuk membuat dan membuang keputusan, dan mempunyai kerendahan hati untuk mengatakan bahwa kita mungkin tidak mengetahui segalanya dan bahwa semua mempunyai nilai untuk berperan dalam proyek kita.

Prinsip prinsip Agile Modeling

Agile Modeling mendefinisikan sekumpulan prinsip prinsip yang utama dan pendukung yang jika diaplikasikan dalam proyek software development membangun tingkatan untuk kumpulan praktek pemodelan.

Prinsip prinsip utama:

- **Assume Simplicity**. Selama kita membangun kita harus mengasumsikan bahwa solusi yang sederhana adalah solusi yang terbaik. Jangan membuat software sampai overbuild, atau pada kasusnya AM tidak menggambarkan fitur fitur tambahanpada model kita yang tidak kita butuhkan sekarang. Mempunyai keberanian bahwa kita tidak membutuhkan untuk terlalu memodelkan system secara berlebihan sekarang, kita dapat memodelkan berdasarkan requirement yang ada sekarang dan membangun kembali system di masa depan jika requirementnya bertambah. Pertahankan model kita sesederhana mungkin.
- **Embrace Change**. Requirement meningkat sewaktu waktu. Stakeholder proyek dapat merubah proyek kedepannya, orang baru menambah dan yang sudah ada bias ditinggalkan. Stakeholder proyek dapat merubah sudut pandang, merubah tujuan dan menambahkan criteria pada peran kita. Maksudnya adalah bahwa project environment kita berubah sesuai dengan efforts progress, dan bahwa sebagai hasilnya pendekatan kita pada pengembangan ini harus merefleksikan realita ini.
- **Enabling The Next Effort Is Your Secondary Goal**. Proyek kita masih dapat mengalami kegagalan meskipun tim kita memberikan suatu working system pada user kita, bagian dari melengkapi kebutuhan dari stakeholder proyek adalah untuk memastikan bahwa sistem kita cukup kuat jadi bisa ditambahkan sewktu waktu. Usaha kita selanjutnya mungkin menjadi pengembangan dari dikeluarkannya rilis selanjutnya dari sistem kita atau bisa juga menyederhanakan operasi dan mendukung versi yang telah ada yang kita bangun. Untuk memfungsikannya kita tidak hanya ingin membangun kualitas software tapi juga membuat dokumentasi yang cukup dan materi pendukung sehingga orang selanjutnya yang mengerjakan bisa lebih efektif. Kesimpulannya, ketika kita bekerja pada sistem, kita harus memperhatikan masa depan pembuatan sistem ini.
- **Incremental Change**. Suatu konsep yang penting untuk mengerti suatu pemodelan adalah bahwa kita tidak harus membuat benar pada pertama kalinya, kenyataanya tidak seperti yang diinginkan meskipun kita sudah berusaha. Terlebih lagi kita tidak harus merekam setiap detil pada model kita, kita hanya

harus mengusahakan yang terbaik pada waktunya. Karena kita bisa melakukan perubahan secara bertahap jika dibutuhkan.

- **Maximize Stakeholder Investment**. Stakeholder proyek kita menanamkan resource resource seperti waktu, uang, fasilitas dan sebagainya untuk mendapatkan software yang dikembangkan sesuai dengan keinginannya. Karenanya kita harus bisa memanfaatkan resource tersebut secara maksimal untuk kepuasan semua pihak.
- **Model With A Purpose**. Beberapa pengembang khawatir mengenai artifact mereka seperti model, source code, atau dokumen apakah cukup detil atau terlalu detil, ataukah akurat atau tidak. Kemudian yang dilakukan adalah kembali kebelakang mempertanyakan mengapa membuat artifact ini dan untuk siapa dibuat. Karenanya untuk menghindari terjadinya hal tersebut maka kita harus terlebih dulu menentukan tujuan pembuatan dan sasaran dibuatnya model tersebut.
- **Multiple Models**. Kita mungkin saja butuh menggunakan model yang beragam untuk membangun software karena tiap model menggambarkan suatu aspek tunggal dari software. Dengan mempertimbangkan kompleksitas software sekarang ini maka kita harus mengetahui cakupan yang luas dari teknik pada peralatan intelektual modeling kita sehingga lebih efektif. Kita tidak harus membangun semua model yang diberikan tetapi harus dilihat cara yang paling efektif meskipun menggunakan beragam model.
- **Quality Work**. Tak ada orang yang menyukai pekerjaan yang sembarangan. Seseorang akan menyukai pekerjaan jika itu bisa dibanggakan dan ada hasil yang akan dicapai, tidak lemah dan sesuai dengan yang diinginkan.
- **Rapid Feedback**. Waktu antara aksi dan umpan balik dari aksi adalah penting. Bekerja dekat dengan customer, untuk mengetahui requirement, menganalisa atau membangun suatu user interface yang sesuai keinginan akan membuat umpan balik yang lebih cepat.
- **Software Is Your Primary Goal**. Tujuan dari pengembangan software adalah untuk menghasilkan software yang sesuai dengan keinginan dari stakeholder proyek dengan cara yang efektif, bukan dengan dokumentasi atau model model yang berlebihan.
- **Travel Light**. Setiap artifak yang dibuat, dan diputuskan untuk dipertahankan, akan membutuhkan untuk di maintain sewaktu waktu. Kompleksitas dan detil yang kita buat diawal akan memberatkan sehingga jika kita membuat model awal harus yang mudah dimengerti dan efektif sehingga akan meringankan perjalanan menuju akhir pembuatan software.

Prinsip prinsip pendukung

- **Content Is More Important Than Representation**. Model yang diberikan mempunyai banyak cara merepresentasikannya. Tetapi isi dari model dan software yang dikerjakan lebih penting daripada representasinya.
- **Everyone Can Learn From Everyone Else**. Kita tidak bisa benar benar mencontoh sesuatu, tetapi kita bisa belajar dari orang lain termasuk teknik

melaui berbagai cara misalnya training, education, mentoring, reading dan lain lain.

- **Know Your Models**. Karena kita mempunyai multiple model yang harus diaplikasikan maka kita harus tahu kekuatan dan kelemahan untuk penggunaan yang efektif.
- **Know Your Tools**. Kita harus memahami peralatan, software, modeling tools, diagramming tools untuk menggunakannya dengan baik.
- **Local Adaptation**. Pendekatan pada software development harus merefleksikan lingkungan kita, termasuk organisasi kita, keadaan stakeholder proyek dan lingkungan proyek itu sendiri.
- **Open And Honest Communication**. Kita harus terbuka dan berkomunikasi dengan jujur pada pengembang dan stakeholder proyek, pengguna dan semua yang terlibat pada pembuatan proyek.
- **Work With People's Instincts**. Bekerja pada bidang ini bukan berarti tidak mempertimbangkan perasaan dan insting. Kita juga harus bekerja dengan insting manusia.

Praktek Agile Modeling

Kebiasaan kebiasaan Agile Modeling juga mempunyai praktek utama dan praktek pendukung.

Praktek praktek utama:

- Partisipasi Aktif Stakeholder
- Menggunakan Barang-Barang yang Tepat
- Collective Ownership
- Mempertimbangkan Kemampuan Tes
- Membuat Beberapa Model Secara Paralel
- Membuat Kandungan Sederhana
- Menggambarkan Kesederhanaan Model
- Memperlihatkan Model di Depan Umum
- Mengiterasi Artefak lainnya
- Model in Small Increments
- Modelkanlah Dengan Yang Lainnya
- Percayakan Dengan Kode
- Gunakan Alat Paling Sederhana

Praktek praktek pendukung:

- Gunakan Permodelan Standar
- Gunakan Contoh dengan Hati-Hati
- Buang Model Sementara
- Bentuklah Kontrak Model
- Model Untuk Berkomunikasi

- Model Untuk Dimengerti
- Gunakan Kembali Artefak yang Ada
- Update Hanya Ketika Rusak

Dokumentasi Agile

Pemakaian dokumentasi pada Agile Modeling mempunyai beberapa alasan baik alas an yang bagus atau alas an buruk. Alasan bagusnya antara lain karena stakeholder kita membutuhkannya, untuk menentukan kontrak model, untuk mendukung komunikasi dengan kelompok luar, untuk memikirkan sesuatu dari awal sampai selesai. Sedangkan alasan yang dianggap buruk adalah karena pemohon ingin terlihat dalam kendali, pemohon ingin diketahui keberadaannya, pemohon tidak mengetahui yang lebih baik, proses anda mengatakan untuk membuatnya, seseorang ingin meyakinkan bahwa semuanya baik-baik saja, tugas spesial anda untuk kelompok lain, kontrak perusahaan anda secara rutin mengarah ke kompetisi ulang. Sedangkan tujuan adanya dokumentasi Agile adalah memaksimalkan investasi stakeholder yang berharga, memenuhi tujuan pembuatan model, menggambarkan informasi yang kemungkinan besar sedikit untuk dirubah, menggambarkan "barang bagus untuk diketahui", mempunyai pelanggan tertentu dan memfasilitasi usaha pelanggan tersebut, cermat, konsisten, and detail, mempunyai daftar pengerjaan semua proses.

Untuk menjaga Agile Modelling tetap sederhana yaitu dengan cara cara sebagai berikut yaitu:

- Tambahkan praktek AM hanya ketika mereka akan mempertinggi produktifitas anda
- Buatlah sebuah model hanya ketika model menambah nilai
- Buatlah dokumen hanya ketika model menambah nilai dan stakeholder anda berkeinginan untuk membayar dan stakeholder anda mengerti perdagangan

Kapan kita menggunakan Agile Modeling dalam pemodelan bisa dibatasi pada masalah masalah antara lain ketika kita menggunakan pendekatan Agile untuk pembangunan, jika bekerja secara iteratif dan increment, persyaratan yang tidak jelas atau berubah – ubah, tujuan utama anda adalah membangun software, kita mempunyai dukungan dan keterlibatan stakeholder aktif, kelompok pembangun berada didalam kontrol atas nasibnya, seorang pemenang sebenarnya untuk keberadaan AM , kita mempunyai tanggung jawab dan memotifasi pembangun, kita mempunyai sumber yang cukup untuk anda gunakan

Yang paling banyak digunakan proses tangkas, awalnya diusulkan oleh Kent Beck

- Perencanaan XP
- Dimulai dengan penciptaan cerita-cerita pengguna
- Tim memeriksa setiap keinginan dan menyebutkan biaya
- Cerita dikelompokkan ke untuk peningkatan penyampaian
- Komitmen dibuat sesuai tanggal penyajian

- Setelah proyek pertama mengalami peningkatan kecepatan proyek digunakan untuk membantu menentukan tanggal berikutnya bagi tahapan lain

Metodologi XP terbagi atas 3 tahapan antara lain :

XP Desain

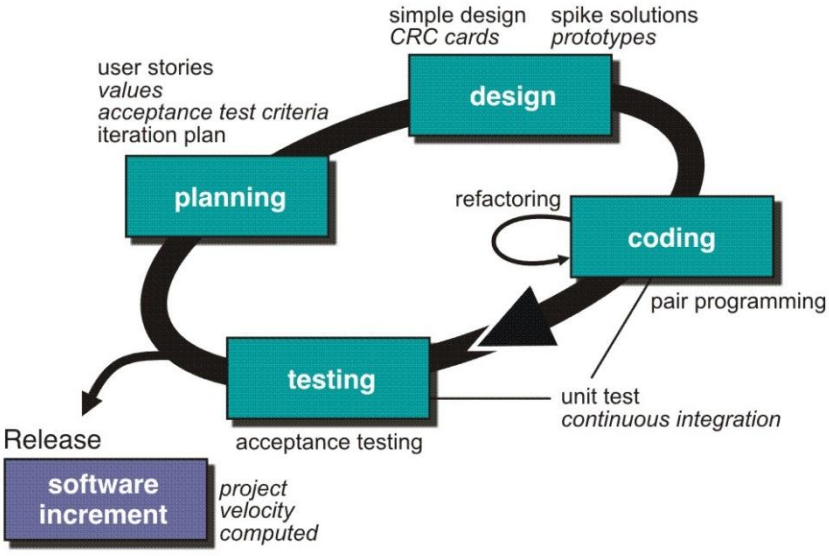
- Mengikuti prinsip KIS (Keep it simple)
- Mendorong penggunaan kartu CRC (lihat Bab 8)
- Untuk masalah desain yang sulit, menunjukkan penciptaan solusi lonjakan - desain prototipe
- Mendorong refactoring - sebuah perbaikan iteratif dari desain program internal

XP Coding

- Merekomendasikan konstruksi tes unit untuk toko sebelum coding dimulai
- Mendorong pasangan pemrograman

Pengujian XP

- Semua tes unit dieksekusi setiap hari
- Tes penerimaan ditentukan oleh konsumen dan dieksekusi untuk menilai pelanggan terlihat functionalit



Gambar 4.2.: Model XP

Model-model proses agile yang lain adalah :

- Adaptive Software Development (ASD)

- Dynamic Systems Development Method (DSDM)
- Scrum
- Crystal
- Feature Driven Development
- Agile Modeling (AM)

Latihan :

1. Apa yang dimaksud dengan model pengembangan proses agile ?
2. Fokus pengembangan agile ?
3. Ruang lingkup pengembangan agile ?
4. Prinsip-prinsip utama pengembangan agile ?
5. Prinsip-prinsip pendukung pengembangan agile ?
6. Dokumentasi agile ?
7. Dalam pengembangan agile terdapat beberapa model mana yang paling baik diantara beberapa pengembangan model tersebut ?