Agile Development Methods

Oleh : 5165-Kundang K Juman

Agile Development Methods :

PENDAHULUAN

Latar Belakang

Bidang pengembangan perangkat lunak berkembang sangat pesat dewasa ini. Dalam perkembangannya ini tidak terlepas dari peran metodologi yang digunakan untuk membangun. Jika kita lihat kembali ke belakang, berbagai metodologi untuk mengembangkan perangkat lunak telah cukup banyak diperkenalkan. Mulai dari model *waterfall* sampai dengan model-model *incremental*. Semua metodologi yang berkembang sebelumnya tidak mampu menangani kemungkinan perubahan atau

penambahan *requirements* yang terjadi pada saat pengembangan dilaksanakan atau bahkan pada waktu fase terakhir pengembangan. Saat bekerja dalam tim untuk mengerjakan suatu proyek sangatlah penting menentukan metodologi pengembangan perangkat lunak dan proses pengembangan perangkat lunak yang akan digunakan.

Metodologi pengembangan perangkat lunak sendiri adalah sebuah metodologi yang digunakan untuk membuat struktur, rencana, dan kontrol pengerjaan suatu proyek, sedangkan proses pengembangan perangkat lunak adalah model-model dan metodologi yang digunakan untuk mengembangkan suatu perangkat lunak. Hal ini yang mendorong munculnya metodologi atau model pengembangan perangkat lunak yang baru yang mampu mengatasi kekurangan tersebut. Untuk pembuatan atau pengembangan suatu perangkat lunak terdapat beberapa hal yang harus di perhatikan, mulai dari sumber daya manusia (*resources*) yang menangani proyek tersebut, sampai dengan metode apa yang harus diterapkan dalam proses pengembangan perangkat lunaknya. Pada era 2000-an mulai berkembang metodologi baru yang sangat fleksibel, yaitu *Agile Development Methods*. *Agile Development Methods* dikembangkan karena pada metodologi tradisional terdapat banyak hal yang membuat proses pengembangan tidak dapat berhasil dengan baik sesuai tuntutan *user*. Metodologi ini sudah cukup banyak berkembang, di antaranya adalah *Extreme Programmning* (XP), *Adaptive Software Development* (ASD), *Dynamic Systems Development Method* (DSDM), *Scrum Methodology*, *Crystal*, *Feature Driven Development* (FDD), *Agile Modeling* (AM), *Rational Unified Process* (RUP).

Agile Development Methods

Berdasarkan latar belakang tersebut maka kami rumuskan masalah dalam makalah ini adalah sebagai berikut :

1. Apa itu *Agile Development Methods ?*
2. Apa itu *Agile Manifesto ?*
3. Bagaimana prinsip *Agile Development Methods* ?
4. Apa tujuan dari *Agile Development Methods ?*
5. Apa saja dan bagaimana model serta proses dari *Agile Development Methods* itu ?

Tujuan Agile Development Methods

:Tujuan Agile Development Methods adalah sebagai berikut :

1. Untuk menjelaskan *Agile Development Methods* yang merupakan salah satu metodologi pengembangan perangkat lunak
2. Untuk menjelaskan bagaimana proses dan model-model pengembangan perangkat lunak *Agile Development Methods*

Pengertian Agile Development Methods

Agile Development Methods adalah sekelompok metodologi pengembangan perangkat lunak yang didasarkan pada prinsip-prinsip yang sama atau pengembangan sistem jangka pendek yang memerlukan adaptasi cepat dari pengembang terhadap perubahan dalam bentuk apapun. *Agile development methods* merupakan salah satu dari Metodologi pengembangan perangkat lunak yang digunakan dalam pengembangan perangkat lunak. *Agile* memiliki pengertian bersifat cepat, ringan, bebas bergerak, dan waspada. Sehingga saat membuat perangkat lunak dengan menggunakan *agile development methods* diperlukan inovasi dan responsibiliti yang baik antara tim pengembang dan klien agar kualitas dari perangkat lunak yang dihasilkan bagus dan kelincahan dari tim seimbang.

Agile Model merupakan proses pengembangan software yang berkembang pada tahun 1990. Metodologi yang dikenal sebagai *agile development methods* ini mengutamakan fleksibilitas terhadap perubahan-perubahan yang terjadi selama pengembangan. Model-model dari *agile* diantaranya *Rational Unified Process* (1994), *Scrum* (1995), *Crystal*, *Extreme Programming* (1996), dan *Adaptive Software Development*, *Feature Driven Development*, and *Dynamic Systems Development Method* (DSDM) (1995). Dan pada akhirnya terbentuklah pada tahun 2001 proses pengembangan *agile*.

Dalam proses pengembangan *agile* dilakukan secara iterasi atau perulangan. Singkatnya jika suatu proyek pengembangan *software* dikerjakan dengan menggunakan metode *agile*, maka selama waktu pengerjaannya akan selalu dijumpai proses pengembangan yang dilakukan berulang (dalam Imansyah, 2012).

Agile Manifesto

Agile Manifesto merupakan nilai-nilai yang digunakan dalam mendasari berlangsungnya *Agile Software Development*. *Agile Manifesto* muncul dilatarbelakangi oleh frustasi para pengembang *software* akibat metode yang digunakan pada saat itu, yaitu *Waterfall* Model. Salah satunya jeda waktu yang panjang antara penentuan *requirement* dan *product delivery* yang dapat berujung pada pembatalan project karena prosesnya yang lama. Kebutuhan klien juga berubah-ubah pada jangka waktu tersebut sehingga produk akhir tidak memenuhi *requirement*. Pada tahun 2000, Jon Kern, Kent Beck, Ward Cunningham, Arie van Bennekum, Alistair Cockburn, dan 12 orang lainnya mengadakan pertemuan di Oregon dan kemudian pada tahun 2001 di The Lodge, Utah. Pertemuan tersebut menghasilkan *Agile Manifesto* yang dituliskan secara formal. *Agile Manifesto* dapat dilihat melalui agilemanifesto.org.

Agile Manifesto terdiri dari 4 nilai utama. Berikut adalah 4 nilai *Agile Manifesto* yang dikutip dari website tersebut.

Individuals and Interactions Over Processes and Tools

Individu dan interaksi antar individu pelaku *development* lebih diutamakan dibandingkan prosesnya, karena yang merespon terhadap kebutuhan klien adalah individu, bukan proses, sehingga individu lah yang menyetir *developmen*t yang berlangsung. Dengan demikian, *development* akan lebih responsif terhadap

kebutuhan klien. Selain itu, jika individu sebagai penyetir *development*, komunikasi antar individu dapat berlangsung dengan fleksibel, misalnya ketika dibutuhkan sesuatu dan perlu dikomunikasikan. Sedangkan jika disetir oleh proses, bisa jadi komunikasi antar anggota harus terstruktur, terjadwal, dan kontennya spesifik. Selain individu itu sendiri, interaksi antar individu juga penting. Mungkin perkataan seperti *"loh, ini codenya kok udah berubah? sejak kapan? percuma dong gue udah bikin code sampe begadang!"* tidak akan terjadi ketika terdapat interaksi yang baik antar individu. Pentingnya interaksi juga menandakan bahwa developer tidak boleh "masa bodoh" dengan pekerjaan developer lain.

1. Working Software Over Comprehensive Documentation

Dulu, banyak waktu yang dialokasikan untuk membuat dokumentasi produk untuk melakukan *development* maupun *delivery*. Misalnya penjabaran berbagai spesifikasi produk, *requirement*, desain *interface*, dan sebagainya yang ditulis secara detail sehingga menghasilkan dokumen yang cukup banyak. Kemudian dokumen-dokumen tersebut juga memerlukan persetujuan pihak terkait agar *development* dapat dilaksanakan. Hal ini menyebabkan waktu *development* semakin panjang (menambah *delay* untuk *delivery*). Pada *agile*, lebih diutamakan *working* produk yang ter*deliver* ke klien dengan cepat, sehingga hal-hal terkait dokumentasi tersebut dikurangi. Pada *Agile*, dokumentasi-dokumentasi *requirement* diwujudkan sebagai *user stories*, dan itu sudah cukup untuk developer untuk memulai megimplementasikan fungsi yang diinginkan. Namun, agile tetap memperhatikan dokumentasi meskipun *working software* lebih diutamakan.

2. Customer Collaboration Over Contract Negotiation

Pada *agile*, iterasi yang dilengkapi dengan kolaborasi dengan klien akan lebih efektif untuk *development*. Klien dapat memberikan *feedback* terhadap hasil yang telah dibuat pada setiap iterasi sehingga dapat diperbaiki maupun ditambahkan. Ini menguntungkan kedua pihak, developer dan klien. Developer diuntungkan karena spesifikasi produk menjadi lebih jelas dan mengeliminasi adanya fitur yang kurang *feasible* dan klien juga diuntungkan karena mendapatkan produk yang lebih sesuai dengan keinginannya. Sedangkan pada *Waterfall*, klien bernegosiasi dengan *development* terkait *requirement* produk secara detail sebelum *development* dimulai. Klien hanya terlibat sebelum proses dimulai, bukan ketika proses berjalan. Ini akan lebih menyulitkan developer untuk melakukan implementasi sesuai kebutuhan klien. *Agile* berusaha memastikan produk sesuai atau sedekat mungkin dengan apa yang klien inginkan sebenarnya.

3. Responding to Change Over Following a Plan

Model *development* tradisional seperti *Waterfall* tidak fleksibel terhadap perubahan, sehingga sekarang banyak dihindari. Model tersebut menekankan proses *development* yang sesuai dengan perencanaan dan secara detail serta sesuai urutan yang telah ditetapkan dengan fitur-fitur yang sudah pasti, tidak dapat berubah-ubah. Pada *Agile*, *development* lebih fleksibel terhadap perubahan. Agile menggunakan iterasi yang pendek dan fitur baru dapat ditambahkan pada iterasi berikutnya jika diperlukan fitur baru. Menurut *Agile*, fleksibilitas terhadap perubahan

dapat meningkatkan kualitas *project* sehingga diperoleh nilai lebih pada hasil akhirnya.

C. Prinsip Agile Development Methods

Dari keempat nilai tersebut, dapat dijabarkan ke dalam 12 prinsip sebagai *Agile Principles*. Berikut adalah prinsip-prinsip tersebut yang dikutip dari *website agile manifesto*.

1. Prioritas utama adalah memuaskan klien dengan menghasilkan perangkat lunak yang bernilai secara cepat dan rutin.

2. Siap terhadap perubahan kebutuhan. Proses Agile memanfaatkan perubahan untuk keuntungan klien.

3. Menghasilkan perangkat lunak yang bekerja secara rutin, dari jangka waktu beberapa minggu sampai beberapa bulan, dengan mengutamakan jangka waktu yang pendek.

4. Rekan bisnis dan pengembang perangkat lunak harus bekerjasama sepanjang proyek.

5. Lingkungan pengembang proyek memiliki suasana yang motivatif. Berikan mereka lingkungan dan dukungan yang dibutuhkan, dan percayai mereka untuk dapat menyelesaikan pekerjaan dengan baik.

6. Metode yang paling efisien dan efektif untuk bertukar informasi dari dan dalam tim pengembang adalah dengan komunikasi secara langsung.

7. Perangkat lunak yang bekerja adalah ukuran utama kemajuan suatu tim.

8. Proses *Agile* mendukung pengembangan yang berkelanjutan dengan kecepatan pengembangan yang konsisten.

9. Perhatian terhadap detail-detail teknis dan desain akan meningkatkan *agility*.

10. Kesederhanaan (memaksimalkan jumlah pekerjaan yang belum dilakukan) adalah hal yang sangat penting.

11. *Self-organizing team* mendukung arsitektur, kebutuhan, dan rancangan perangkat lunak yang baik.

12. Secara berkala, tim pengembang berefleksi tentang bagaimana agar pengembangan lebih efektif, kemudian menyesuaikan cara bekerja mereka.

Dua belas prinsip tersebut menjadi suatu dasar bagi tim agar sukses *menerapkan agile development methods*. Dengan prinsip-prinsip tersebut *agile* berusaha untuk menyiasati tiga masalah yang biasanya dihadapi saat proses pembuatan perangkat lunak, yaitu :

- Kebutuhan perangkat lunak sulit diprediksi dari awal dan selalu akan berubah. Selain itu, prioritas klien juga sering berubah seiring berjalannya proyek.
- Desain dan pembangunan sering tumpang tindih. Sulit diperkirakan seberapa jauh desain yang diperlukan sebelum pembangunan.

- Analisis, desain, pembangunan dan testing tidak dapat diperkirakan seperti yang diinginkan.

B. Tujuan Agile Development Methods

Secara garis besar tujuan dirumuskannya *agile development methods*, yaitu :

1. ***High-value & working App system***, diharapkan dengan memakai *agile development methods* dapat dihasilkan perangkat lunak yang mempunyai nilai jual yang tinggi, biaya pembuatan bisa di tekan dan perangkat lunak bisa berjalan dengan baik.

2. ***Iterative, incremental, evolutionary***, *agile* adalah metode pengembangan perangkat lunak yang iteratif, selalu mengalami perubahan, dan evolusioner. Tim harus bekerja dalam waktu yang singkat(biasanya 1-3 minggu) dan juga selalu menambah fungsionalitas dari perangkat lunak sesuai dengan kebutuhan klien. *Agile* dapat dianalogikan ketika seseorang ingin pergi ke suatu kota dan dia tidak tahu jalannya. Lalu bagaimana dia bisa sampai tujuan? Dengan sering bertanya kepada orang yang dia temui di jalan hingga dia sampai di tempat tujuan.

3. ***Cost control & value-driven development***, salah satu tujuan dari *agile* yaitu pengembangan perangkat lunak disesuaikan dengan kebutuhan pengguna, tim bisa dengan cepat merespon kebutuhan yang diinginkan pengguna sehingga waktu dan biaya pembuatan perangkat lunak bisa dikontrol.

4. ***High-quality production***, walaupun biaya pembuatan perangkat lunak bisa ditekan dan proses pembuatan bisa dipercepat , tetapi kualitas dari perangkat lunak yang dibuat harus tetap dijaga. Dengan melakukan tes setiap fungsionalitas perangkat lunak setelah selesei dibuat berarti *agile* juga mengakomodir kebutuhan ini.

5. ***Flexible & risk management***, jika kita menggunakan metode pembuatan yang biasanya dipakai, jika ingin mengubah fungsionalitas dari *wireframe* yang telah dibuat dibutuhkan proses yang rumit. Mulai dari pertemuan dengan sistem analis untuk mengubah sistem perangkat lunak, perubahan rencana rilis produk hingga perubahan biaya produksi. Pertemuan dengan klien untuk melakukan tes perangkat lunak juga sering dilakukan sehingga fungsionalitas perangkat lunak mudah diubah dan akhirnya kegagalan perangkat lunakpun bisa diminimalisir.

6. ***Collaboration***, dengan menggunakan *agile*, tim pengembang diharuskan sering bertemu untuk membahas perkembangan proyek dan *feedback* dari klien yang nantinya akan ditambahkan dalam perangkat lunak, sehingga tim bisa berkolaborasi dengan maksimal.

7. ***Self-organizing, self-managing teams***, rekrut orang terbaik, beri dan dukung kebutuhan mereka lalu biarkan mereka bekerja. Itulah perbedaan *agile* dan SDM lainnya. Dengan *agile*, *developer* dapat memanajemen dirinya sendiri, sedangkan manajer tim hanya bertugas mengkolaborasikan *developer* perangkat lunak dengan klien. Sehingga terciptalah tim yang solid.

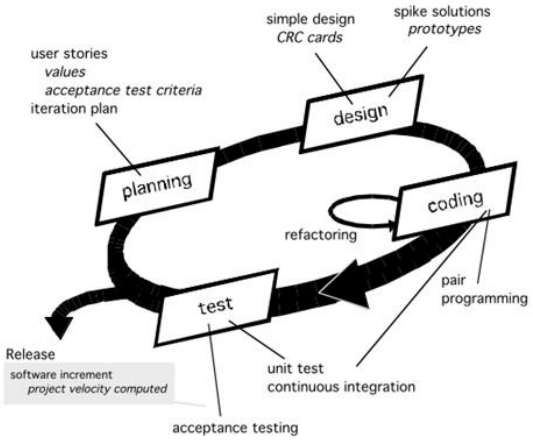
Model Agile Development Methods

a. Extreme Programming

Filosofi *eXtreme Programming* (XP) adalah metode pengembangan perangkat lunak yang ringan dan termasuk salah satu *agile methods* yang dipelopori oleh Kent Beck, Ron Jeffries, dan Ward Cunningham. XP merupakan *agile methods* yang paling banyak digunakan dan menjadi sebuah pendekatan yang sangat terkenal. Sasaran XP adalah tim yang dibentuk berukuran antara kecil sampai medium saja, tidak perlu menggunakan sebuah tim yang besar (Widodo dan Massus Subekti, 2006:96).

eXtreme Programming sebagai sebuah metode yang dinamis diperlihatkan dalam empat *values* yang dimilikinya dan keempatnya merupakan dasar-dasar yang diperlukan dalam XP. XP juga digunakan untuk mengatasi masalah *requirements* yang tidak jelas dan sering berubah-ubah (*vague and volatile requirements*). Kent Beck menyatakan bahwa tujuan jangka pendek individu sering berbenturan dengan tujuan sosial jangka panjang. Karena itu dibuatlah *values* yang menjadi aturan, hukuman, dan juga penghargaan. Keempat *values* tersebut adalah komunikasi (*communication*), kesederhanaan (*simplicity*), umpan balik (*feedback*), dan keberanian (*courage*). XP dimunculkan untuk menangani perubahan-perubahan yang biasanya sering terjadi pada saat pengembangan berlangsung bahkan pada saat proses pengembangan sudah hampir berakhir. (Widodo, 2008).

Langkah Operasional



Gambar 1. Tahapan *eXtreme Programming* (Sumber: www.catatandestra.blogspot.com)

Dalam *eXtreme Programming* terdapat 4 tahapan aktivitas utama dalam pengerjaannya yang dapat dilihat dari diagram di atas, yaitu :

Planning

Planning atau perencanaan adalah proses metodis yang dirancang untuk mencapai tujuan tertentu dan pengambilan keputusan untuk mencapai hasil yang diinginkan. Kebutuhan yang dibutuhkan pada tahap ini, yaitu :

Teknik pengumpulan data

a) Analisis kebutuhan sistem
b) Identifikasi *actor*
c) Identifikasi *use case*

Aktivitas *planning* pada model proses XP berfokus pada mendapatkan gambaran fitur serta fungsi dari perangkat lunak yang akan dibangun. Pada aktivitas ini dimulai dengan membuat kumpulan cerita atau gambaran yang diberikan klien yang kemudian akan menjadi gambaran dasar dari perangkat lunak.

Kumpulan tersebut nantinya dikumpulkan dalam sebuah indeks cerita dimana setiap poin dari indeks tersebut ditentukan prioritasnya untuk dibangun. Anggota tim dari pengembang aplikasi nantinya akan menentukan perkiraan waktu dan biaya yang dibutuhkan untuk setiap indeks tersebut. Setelah menentukan kebutuhan tersebut, tim XP akan menentukan alur pengembangan aplikasi dengan terlebih dahulu memulai mengembangkan tugas dengan resiko dan nilai prioritas yang tinggi terlebih dahulu, dan seluruh tugas akan selesai dalam tenggat waktu dua minggu.

Selama proses pengembangan, klien dapat mengubah, memperkecil, membagi, dan membuang setiap rencana dari aplikasi. Tim XP akan mempertimbangkan setiap perubahan yang diajukan klien yang berikutnya akan mengubah setiap rencana dari pengembangan perangkat lunak tersebut juga.

Design

Aktivitas *design* dalam pengembangan aplikasi bertujuan untuk mengatur pola logika dalam sistem. Sebuah *design* yang baik, dapat mengurangi ketergantungan antar setiap proses pada sebuah sistem. Dengan begitu, jika salah satu fitur pada sistem mengalami kerusakan, tidak akan mempengaruhi sistem secara keseluruhan.

Design pada model proses XP menjadi panduan dalam membangun perangkat lunak yang didasari dari cerita klien sebelumnya. Dalam XP, proses *design* terjadi sebelum dan sesudah aktivitas *coding* berlangsung. Dimana aktivitas *design* terjadi secara terus-menerus selama proses pengembangan aplikasi berlangsung.

Coding

Setelah menyelesaikan pengumpulan cerita dan menyelesaikan *design* untuk aplikasi secara keseluruhan, XP lebih merekomendasikan tim untuk terlebih dahulu membuat modul unit tes yang bertujuan untuk melakukan uji coba setiap cerita yang didapat dari klien. Setelah berbagai unit tes selesai dibangun, tim barulah melanjutkan aktivitasnya ke penulisan *coding* aplikasi. XP menerapkan konsep pair *programming* dimana setiap tugas sebuah modul dikembangkan oleh dua orang *programmer.* XP beranggapan, 2 orang akan lebih cepat dan baik dalam menyelesaikan sebuah masalah. Selanjutnya, modul aplikasi yang sudah selesai dibangun akan digabungkan dengan aplikasi utama.

Testing

Tahapan uji coba pada XP sudah dilakukan juga pada saat tahapan sebelumnya yaitu *coding*. Pengujian perangkat lunak dimaksudkan untuk menguji semua elemen-elemen perangkat lunak yang dibuat apakah sudah sesuai dengan yang diharapkan. XP menerapkan perbaikan masalah kecil dengan sesegera mungkin akan lebih baik

dibandingkan menyelesaikan masalah pada saat akan mencapai tenggat akhir. Oleh karena itu, setiap modul yang sedang dikembangkan akan terlebih dahulu mengalami pengujian dengan modul unit tes yang telah dibuat sebelumnya.

Setelah semua modul telah dikumpulkan dalam sebuah sistem yang sempurna, barulah pengujian penerimaan (*acceptance test*) dilakukan. Pada tahapan pengujian ini aplikasi langsung diuji coba oleh pengguna atau klien dan mendapat tanggapan langsung mengenai penerapan cerita yang telah digambarkan sebelumnya.

Keunggulan

Keunggulan yang dimiliki *eXtreme Programming*, yaitu :

1) Semua masalah dan pekerjaan harus dioptimalkan pada waktu jam kerja (*On-site customer*), karena XP memerlukan satu orang dari pihak bisnis yang akan dibawa dalam proses pengembangan dari awal sampai berakhir.

2) XP menjadi sebuah metodologi yang semi formal karena semua dilakukan dengan *practice* yang sederhana. Beberapa hal yang dapat menjadikan XP masuk kategori metodologi semi formal adalah (Widodo, 2008) :

    d) Komunikasi yang selalu bersifat oral tanpa dokumentasi formal.
    e) Umpan balik yang segera setelah mendapat respon dari *user*.
    f) *Collective ownership* yang tidak menggantungkan pekerjaan pada satu orang saja.
    g) Perubahan dan penambahan *requirements* dapat direspon meskipun proses pengembangan sudah hampir selesai.
    h) Proses pengembangan yang menyertakan satu orang dari pihak *user* menjadi *on-site customer* memudahkan komunikasi selama proses pengembangan.

    Keunggulan lain yang dimiliki *eXtreme Programming*, yaitu (Dennislouis. 2014)

    1) Metode yang populer karena lebih santai dan non-restriktif.
    2) Biaya lebih murah.
    3) Mampu mengotomatiskan tes.
    4) Setiap *feedback* ditanggapi dengan melakukan tes, unit tes atau *system integration* dan jangan menunda karena biaya akan membengkak (uang, tenaga, waktu).
    5) Banyak ide baru dan berani mencobanya, berani mengerjakan kembali dan setiap kali kesalahan ditemukan, langsung diperbaiki.

Kelemahan

Kelemahan yang dimiliki *eXtreme Programming*, yaitu tanpa dokumentasi formal maka proses pengembangan ini akan kembali seperti proses yang tidak terpola dan primitif. Selain itu kelemahan XP, yaitu (Dennislouis. 2014)

1) Developer harus selalu siap dengan perubahan karena perubahan akan selalu diterima.
2) Tidak bisa membuat kode yang detail di awal (prinsip *simplicity* dan juga anjuran untuk melakukan apa yang diperlukan hari itu juga).
3) XP tidak memiliki dokumentasi formal yang dibuat selama pengembangan. Satu-satunya dokumentasi adalah dokumentasi awal yang dilakukan oleh *user*.

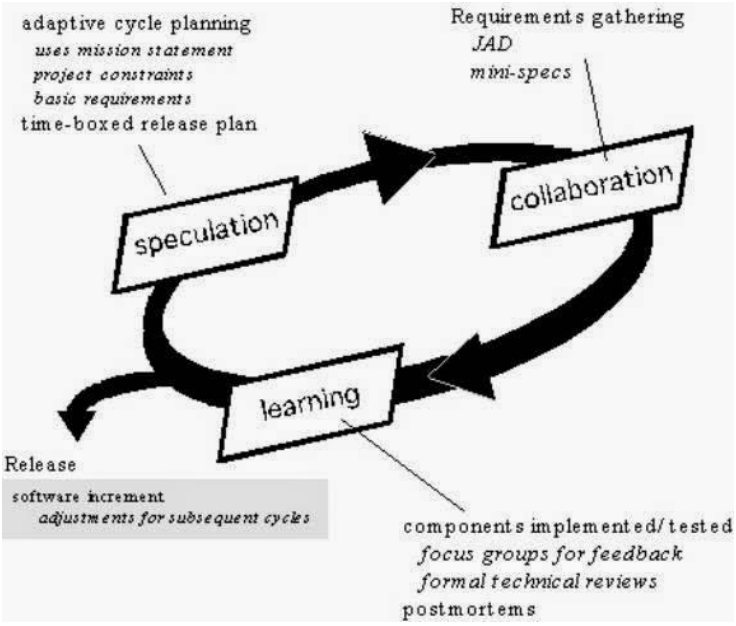
Kondisi Ideal

Penerapan dari *Extreme Programing* cocok untuk teknologi atau prototipe baru, dimana persyaratan berubah dengan cepat, atau beberapa pengembangan diperlukan untuk menemukan masalah pelaksanaan yang tak terduga (Ningsih, 2013). Contoh penerapan ideal yaitu untuk pembuatan aplikasi sistem perusahaan yang sedang melakukan transisi dari cara manual ke sistem yang terkomputerisasi. Atau dengan kata lain untuk user yang sudah terbiasa dengan sistem manual namun ingin meminta *programmer* untuk membuat sistem komputer yang sesuai dengan sistem manual yang telah ada.

Adaptive Software Development (ASD)

Adaptive Software Development biasa di singkat ASD adalah pengembangan *software* yang mewujudkan prinsip bahwa adaptasi yang berdekatan dari proses kerja dalam keadaan normal. Metode ini bisa dikatakan menggantikan metode Pengembangan *Software Waterfall* dengan serangkaian perulangan berspekulasi, berkolaborasi dan siklus. Siklus dinamis memberikan pembelajaran dan adaptasi kepada proyek. Karakteristik dari siklus hidup ASD adalah mengacu kepada misi fokus, berbasis fitur, perulangan, *timeboxed*, risiko dan tolerasi yang berubah-ubah. ASD ini di kembangkan oleh **Jim Highsmith** sebagai teknik untuk membangun *software* dan sistem yang kompleks. Filosofi yang mendasari adalah kolaborasi manusia dan tim yang mengatur diri sendiri.

3 aktifitas yang di lakukan ASD yaitu sebgai berikut :



Gambar 2. Proses *Adaptive Software Development* (ASD) (sumber : http://www.materi-it.com)

1. ***Speculation*** adalah aktivitas *adaptive cycle planning* yaitu menggunakan informasi awal seperti misi dari klien, batasan proyek dan kebutuhan dasar untuk definisikan rangkaian *software increment* (produk *software* yang secara berkala diserahkan).

2. ***Collaboration*** adalah aktifitas orang-orang yang bermotivasi tinggi bekerja sama : saling melengkapi, rela membantu, kerja keras, terampil dibidangnya dan komunikasikan masalah untuk hasilkan penyelesaian yang efektif.

3. ***Learning*** adalah aktivitas tim pembangun sering merasa sudah tahu semua hal tentang proyek. Proses pembelajaran proyek ini bisa dilakukan 3 cara yaitu sebagai berikut :

    - ***Focus Group*** adalah klien dan pengguna memberi masukan terhadap *software*.
    - ***Formal Technique Reviews*** adalah tim ASD lengkap melakukan *review*.
    - ***Postmortems*** adalah tim ASD lakukan introspeksi pada kinerja dan proses.

Bisa diambil kesimpulan bahwa Metodologi ASD ini merupakan aktivitas tim pengembangan *software* yang pertama ditekankan adalah adaptasi atau melakukan pendekatan kepada proyek yang sedang dikerjakan, sama halnya seperti 3 aktifitas ASD yang sebelumnya dijelaskan.

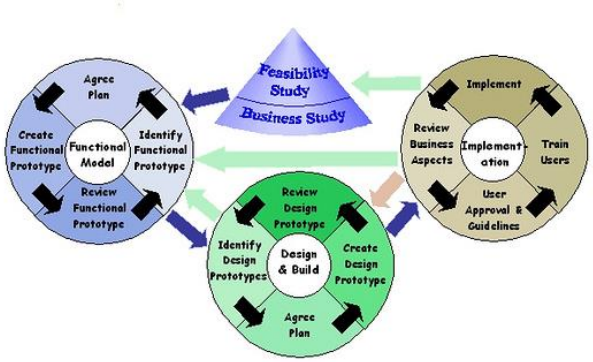
b. *Dynamic Systems Development Method* (DSDM)

Pada *Dynamic System Development Method* menyajikan kerangka kerja (*framework*) untuk membangun dan memelihara sistem dalam waktu yang terbatas melalui penggunaan *prototyping* yang *incremental* dalam lingkungan yang

terkondisikan. Metode ini akan membangun *software* dengan cepat : 80% dari proyek diserahkan dalam 20% dari waktu total untuk menyerahkan proyek secara utuh.

Dynamic System Development Method dapat dikombinasikan dengan *Extreme Programmning* menghasilkan kombinasi model proses yang mengikuti *Dynamic System Development Method* dan praktek yang sejalan dengan *Extreme Programmning*.

Dynamic System Development Method memiliki beberapa aktifitas seperti :



1. Feasibility Study

Kesesuaian proyek awal dinilai dalam fase ini. Fase ini membantu untuk mengidentifikasi jawaban untuk beberapa pertanyaan seperti :

- Apakah DSDM berlaku untuk proyek ini ?
- Apa saja ketergantungan yang muncul dalam proyek ini?
- Apakah ada tantangan teknis?
- Apakah ada keterbatasan sumber daya?
- Apakah ada masalah organisasi yang berdampak dalam proyek?
- Apakah ada risiko yang muncul, Jika demikian apa saja resiko tersebut?
- Bagaimana perkiraan tingkat tinggi dari skala waktu dan biaya ?

Ruang lingkup dari studi kelayakan adalah untuk mengumpulkan rincian yang diperlukan tentang apakah terdapat solusi yang layak atau tidak. Analisis rinci dilakukan pada tahap selanjutnya. Laporan kelayakan (*Feasibility report*) adalah laporan tingkat tinggi yang memungkinkan komite pengarah proyek untuk memutuskan masa depan proyek, dan studi kelayakan lebih lanjut.

Business Study

Setelah melakukan analisis kelayakan pada langkah 1, langkah selanjutnya adalah menganalisis karakteristik bisnis dan teknologi. Studi Bisnis memberikan dasar untuk semua karya – karya berikutnya. Fase ini mengarah pada proses bisnis yang terkena dampak secara rinci dan informasi-informasi yang mereka butuhkan.

Fase ini mencoba menjawab beberapa pertanyaan kunci, seperti:

- Apa Ruang Lingkup Proyek?
- Adakah risiko yang harus ditinjau kembali?
- Apa saja kebutuhan non-fungsionalnya (kinerja, kendala, subyektif)?
- Bagaimana *Prototype* produk di masa yang akan datang ?
- Bagaimana dasar perkembangan teknologi ?
- Apa saja prioritas persyaratan yang diidentifikasi pada langkah 1 dan langkah 2 ?

Fase ini menghasilkan beberapa definisi, antara lain ; definisi area bisnis, kebutuhan prioritas, definisi arsitektur sistem dan rencana pengembangan.

2. Functional Model Iteration

Fase ini memiliki tujuan untuk memberikan model fungsional yang terdiri dari kedua prototipe perangkat lunak yang bekerja dan model statis. Fase ini menghasilkan pengolahan informasi yang diperoleh dalam penelitian bisnis.

Fase ini menghasilkan model fungsional, non fungsional, *time box plan*, dan fungsional model *review records*.

3. Design and Build Iteration

Fase ini menyempurnakan *prototype* fungsional yang dikembangkan pada langkah 3 untuk memenuhi kebutuhan fungsional. Pada fase ini secara utama mengembangkan sistem untuk memenuhi kebutuhan pengguna. Sebuah produk uji coba adalah hasil utama dari fase ini. Iterasi desain dan *build* ini terdiri dari empat kegiatan.

- Mengidentifikasi persyaratan modul.
- Merencanakan dan melakukan rencana sesuai dengan kebutuhan.
- Mengembangkan modul, dan
- Validasi fungsi modul.

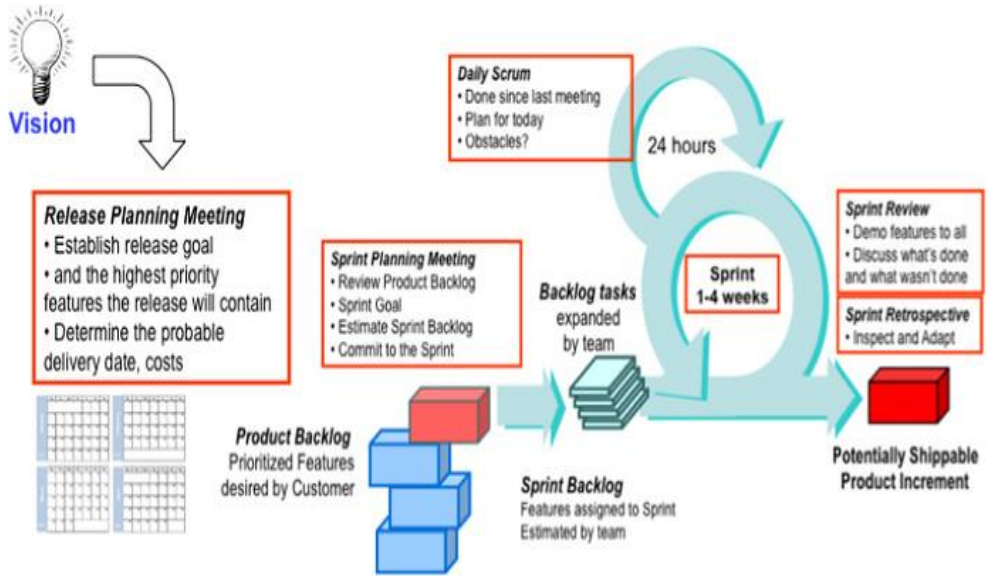
Design and Iteration Build tercapai dalam sebuah rencana kotak waktu (*time box plan*), sistem yang diuji, prototipe desain, dan catatan pengujian.

4. Implementation Phase

Fase ini meliputi transisi dari lingkungan pengembangan untuk lingkungan operasional. Tujuan utama dari tahap ini adalah untuk menempatkan sistem yang diuji ke dalam lingkungan pengguna dan melatih individu – individu untuk menggunakannya.

c. Scrum Methodology

Menurut (Schwaber & Sutherland, 2011), Scrum adalah suatu metodologi atau kerangka kerja yang terstruktur untuk pengembangan produk yang kompleks. *Scrum* terdiri dari sebuah tim yang memiliki peran dan tugas masing-masing. Setiap komponen dalam kerangka melayani tujuan tertentu dan sangat penting untuk kesuksesan penggunaan *scrum*.



Langkah-langkah aktifitas dengan menggunakan metodologi *Scrum* adalah sebagai berikut :

- ***Product Backlog***

  Bagian pertama yang perlu dilakukan adalah membuat kumpulan hal-hal yang diperlukan dan harus tersedia dari produk atau dalam hal ini adalah sistem yang akan dibangun.

- ***Sprint Backlog***

  Langkah ini adalah membuat perencanaan dengan dilakukan pertemuan antara developer dan user, yang akan berkolaborasi untuk memilih *product backlog* untuk dimasukkan ke dalam proses *Sprint*. Hasil pertemuan tersebut di sebut *Sprint Backlog*.

- ***Sprint***

  Dalam Scrum, Sprint adalah sebuah kerangka waktu yang berdurasi maksimal 1 bulan untuk mengembangkan produk yang berpotensi untuk dirilis. Dalam Sprint terdapat 2 pekerjaan, yaitu :

  1. Pertemuan Harian (*Daily Standup Meeting*)

     Merupakan pertemuan dimana setiap 24 jam (1 hari), tim pengembang bertemu untuk membahas perkembangan produk. Hal ini kami lakukan tidak secara langsung, tetapi via dunia maya melalui Skype ataupun Whatsapp.

2. *Refleksi Sprint*

   Merupakan pertemuan yang dilakukan setiap bulannya yang bertujuan untuk membahas hal dari *Sprint Backlog* yang telah berjalan dan telah berhasil dikerjakan, serta dapat memperbaiki dan meningkatkan kualitas produk Sprint yang berikutnya.

- ***Increment***

  Increment merupakan hasil dari seluruh hal dalam *Product Backlog* yang telah selesai dikerjakan pada seluruh *Sprint*. Pada akhir Sprint, *Increment* harus sudah benar-benar selesai, yang berarti harus dalam keadaan yang *useable*.

Scrum sendiri memiliki prinsip :

- ✓ Ukuran tim yang kecil melancarkan komunikasi, mengurangi biaya, dan memberdayakan satu sama lain.
- ✓ Proses dapat beradaptasi terhadap perubahan dan bisnis.
- ✓ Proses menghasilkan beberapa *software increment*.
- ✓ Pembangunan dan orang yang membangun dibagi dalam tim yang kecil.
- ✓ Dokumentasi dan pengujian terus menerus dilakukan setelah *software* dibangun.
- ✓ Proses *scrum* mampu menyatakan bahwa produk selesai kapanpun diperlukan.

d. Crystal

Crystal merupakan sekumpulan dari setiap proses yang diterapkan pada berbagai jenis projek. Metode *Crystal* dikembangkan oleh Alistair Cockburn pada pertengahan 1998. Metode Crystal dikembangkan untuk mengatasi variabilitas lingkungan dan karakteristik khusus dari suatu proyek karena proyek dan orang-orang berkembang dari waktu ke waktu, metodologi demikian juga harus disetel dan berevolusi selama proyek. Inti metode Kristal ini sendiri adalah tujuannya yang memungkinkan tim dari proyek untuk memilih anggota dari *Crystal Family* yang paling sesuai dengan proyek dan lingkungan mereka.

Metode Crystal memfokuskan pada

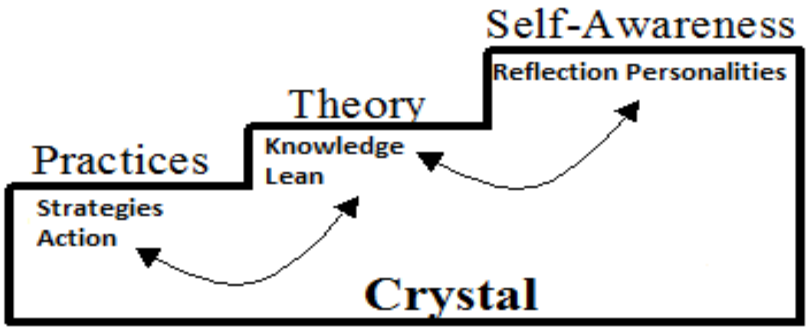
1) *People* (Orang)
2) *Interaction* (Interaksi)
3) *Community* (Perkumpulan)
4) *Skill* (Kemampuan)
5) *Talent* (Bakat)
6) *Communication* (Komunikasi)

Metode Crystal mempunyai prioritas tujuan yaitu dalam suatu proyek dapat terus berlanjut atau dapat dikembangkan, efisiensi dari proyek, dan

penyesuaian ataupun kesabaran dalam mengerjakan proyek. Dalam metode Crystal menekankan pada nilai komunikasi *face to face*, penyesuaian metodologi dengan jenis proyek yang dikerjakan, dan mengatasi inti kesulitan dari proyek yang dibuat.

Sifat – Sifat

Metode *Crystal* mempunyai 7 sifat yang harus dipenuhi agar projek yang dikerjakan dapat berhasil.



Berikut ini beberapa sifat yang dimiliki oleh metode *Crystal* yaitu :

1. *Frequent Delivery*

   Frequent Delivery merupakan iterasi dari perilisan yang reguler dari program *software*. Waktu perilisan tergantung pada panjangnya atau lamanya proyek yang dikerjakan.

   Dengan merilis iterasi dari program, *stakeholder* dapat terlebih dulu menemukan masalah yang ada dalam proyek yang selanjutnya akan mengurangi banyak kerumitan di kemudian hari. Poin yang lain adalah apabila pengguna memutuskan bahwa proyek tersebut tidak berjalan sesuai dengan cara-cara yang dinginkan dalam menyelesaikan proyek tersebut, maka terdapat langkah yang bisa diambil untuk menyelesaikan masalah .

2. *Reflective Improvement*

   Reflective Improvement melibatkan pengembang mengambil istirahat dari suatu pembangunan proyek dan mencoba menemukan cara yang lebih baik dari proses suatu proyek yang dikerjakan. Iterasi membantu memberikan umpan balik pada apakah atau tidak proses dari suatu bekerja dengan lancar.

   Dengan metode *Crystal*, gagasan agar tim mengadakan *workshop* setiap beberapa minggu sangat dianjurkan. *Workshop* ini digunakan untuk membantu menemukan proses yang tidak bekerja dengan baik dan membantu tim untuk melakukan modifikasi suatu proses sehingga sebuah strategi dapat dikembangkan dan bekerja dengan baik untuk tim.

3. *Osmotic Communication* (Komunikasi Tertutup)

   Komunikasi osmotik atau komunikasi tertutup melibatkan seluruh anggota tim dari suatu proyek untuk bersama-sama dalam sebuah ruang lingkup dimana informasi akan menyebar secara merata di sekitar ruang lingkup tersebut. Dengan mendengarkan pendapat orang lain dalam tim, pengembang akan dapat memahami apa yang orang lain lakukan, mendapatkan pengalaman dan mengembangkan ide-ide baru untuk proyek selanjutnya. Pengembang yang bekerja dekat satu sama lainnya dapat membantu menyelesaikan suatu masalah yang dihadapi.

4. *Personal Safety*

   Maksud dari keselamatan pribadi berkaitan dengan masalah kebebasan berbicara dalam sekelompok orang. Orang – orang dalam suatu tim harus mampu saling percaya dan merasa bebas untuk berbicara dan mengemukakan pendapat.

5. *Focus*

   Focus yang ada dalam metode Crystal mengacu pada dua hal, pertama berfokus pada tugas yang diberikan untuk setiap individu dalam sebuah proyek untuk waktu yang cukup, dimana kemajuan dari proyek akan dibuat. Dan kedua, mengacu pada arah dimana proyek merupakan tujuan utama yang ingin dikehendaki.

6. *Easy Access to Expert Users*

   Easy Access to Expert Users melibatkan para pengembang yang bekerja dengan seseorang yang ahli yang berhubungan dengan proyek tersebut sehingga ahli bisa menjawab pertanyaan, menyarankan solusi untuk masalah, dan lainnya. Pengguna ahli harus menjadi seorang pengguna yang aktual atau nyata dan bukan hanya sebagai *tester* dari pembangunan tim. Semakin melibatkan pengguna ahli maka akan semakin baik proyek yang dihasilkan.

7. *Technical Environment with Automateeed Test, Configuration Management & Frequent*

   Ide di balik ini adalah bahwa harus terdapat integrasi yang berkesinambungan dan pengujian dari sebuah proyek, sehingga jika ada perubahan yang dibuat, maka kesalahan dan lain-lain dapat terlihat. Karena hal ini dilakukan secara teratur, maka masalah cenderung tidak muncul dan tumbuh karena mereka dapat diselesaikan lebih awal dalam proyek.

Pengkategorian Projek

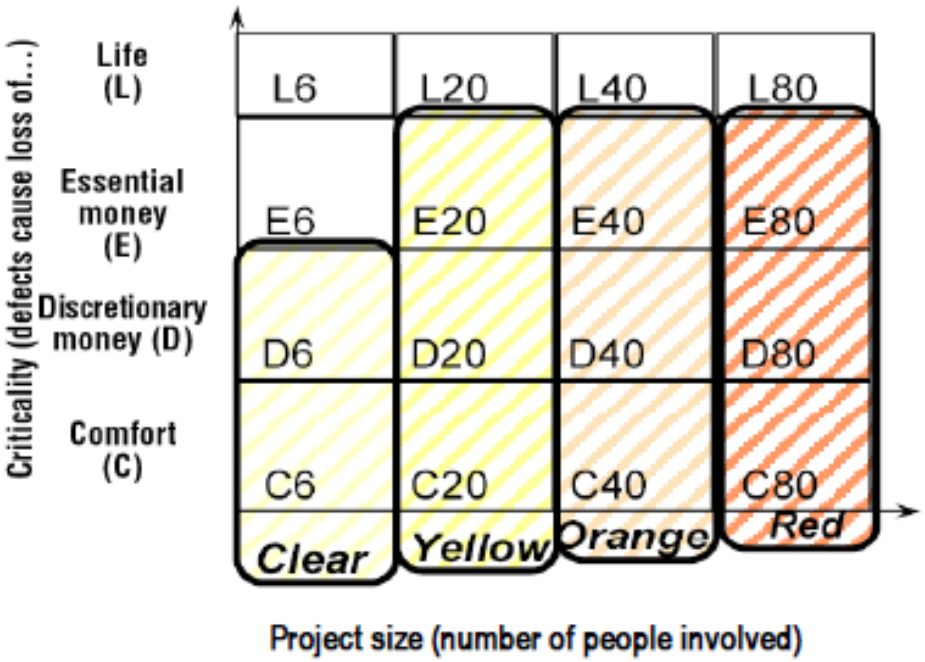
Suatu projek pada metode Crystal dikategorikan berdasarkan dengan kekritisan dari sistem yang dihasilkan dan ukuran dari suatu projek. Terdapat 4 tingkat kekritisan yang ditetapkan, berdasarkan hal yang mungkin hilang karena kegagalan dari sistem yang dihasilkan yakni :

1) *Comfort (C)*

2) *Discretionary Money (D)*

3) *Essential Money (E)*

4) *Life (L)*



Jumlah maksimum orang yang mungkin harus terlibat dalam proyek dianggap sebagai ukuran ukuran proyek.

Kompleksitas

Metode Crystal menempatkan penekanan pada komunikasi diantara pihak yang terlibat dalam suatu projek, yaitu :

- Proyek dengan ukuran yang lebih besar membutuhkan metodologi yang lebih berat (lebih kompleks), karena mereka melibatkan lebih banyak orang, dan karenanya perlu koordinasi yang lebih baik.
- Proyek dengan kekritisan tinggi menggunakan pendekatan yang lebih ketat, yang mungkin diakomodasi oleh metodologi yang mengatur proyek yang kurang penting.

Crystal dikategorikan sesuai dengan ukuran proyek yang dikerjakan. Setiap anggota Crystal telah ditetapkan yang ditandai dengan warnanya masing-masing yang menunjukkan suatu kompleksitas relatif dari proyek: lebih berat metodologi, maka semakin gelap warna diperuntukan untuk mengerjakan suatu proyek.

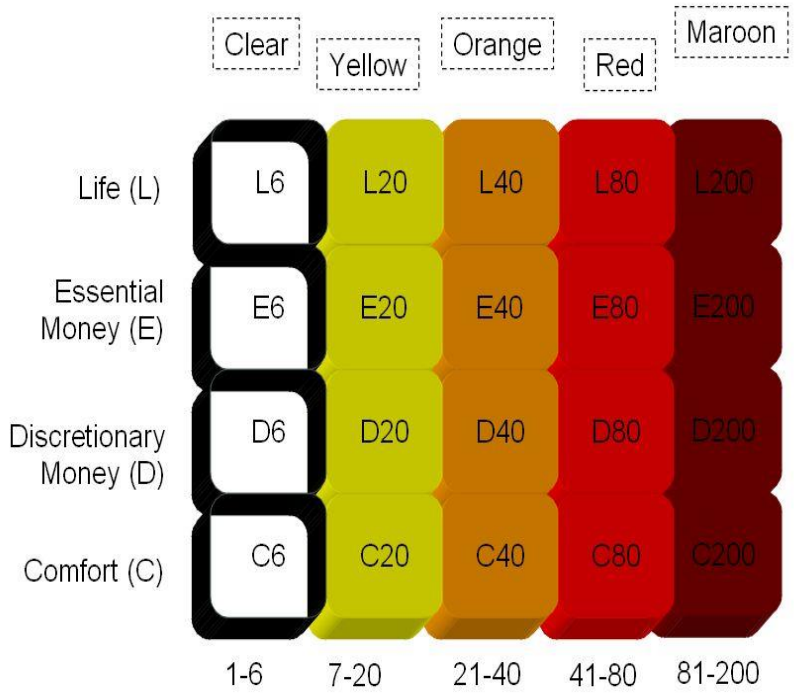
Crystal Family

Cockburn mengembangkan metode yang berbeda dalam suatu metodologi yang sesuai dengan tim dari berbagai ukuran yang perlu strategi yang berbeda untuk memecahkan masalah yang beragam.

Crystal keluarga metodologi menggunakan warna yang berbeda untuk menunjukkan "bobot" yang metodologi untuk memecahkan masalah yang beragam.

The family is divided into different colours. Some examples:

1. *Crystal Clear*
2. *Crystal Yellow*
3. *Crystal Orange*
4. *Crystal Orange Web*
5. *Crystal Red*
6. *Crystal Maroon*
7. *Crystal Diamond*
8. *Crystal Sapphire*



Kelebihan dan Kekurangan

Kelebihan dari metode Crystal yaitu :

1) Selalu lebih murah dan cepat karena berkomunikasi secara langsung
2) Proyek berkembang sesuai ukuran *team* dan berkembang menjadi lebih luas dan metodologi akan menjadi lebih tinggi.

3) Untuk melakukan modifikasi pada proyek yang dikerjakan oleh suatu tim dapat dilakukan dengan mudah karena setiap anggota tim berkomunikasi antara satu sama lain.

4) Proyek yang dihasilkan dapat dikembangkan oleh pengembang dari waktu ke waktu

Kelemahan dari metode Crystal yaitu :

1) Konsistensi dari anggota tim dalam mengerjakan suatu proyek dapat berubah – ubah tergantung pada situasi dan keadaan.

2) Kedisiplinan yang kurang dalam mengerjakan suatu proyek.

3) Setiap instruksi yang diberikan untuk mengerjakan suatu proyek tidak selalu berjalan dengan baik.

e. Feature Driven Development

Fitur-driven development (FDD) adalah iteratif dan proses pengembangan perangkat lunak tambahan. Ini adalah salah satu dari sejumlah metode *Agile* untuk mengembangkan perangkat lunak dan bagian bentuk dari Aliansi *Agile*. FDD memadukan sejumlah industri-praktek terbaik yang diakui menjadi satu kesatuan yang kohesif. Praktek-praktek semua didorong dari klien-nilai perspektif (fitur) fungsi. Tujuan utamanya adalah untuk memberikan nyata, *software* yang bekerja berulang-ulang pada waktu yang tepat.

Sejarah

FDD awalnya dirancang oleh Jeff De Luca, untuk memenuhi kebutuhan spesifik dari satu bulan 15-, 50-orang proyek pengembangan perangkat lunak pada sebuah bank besar di Singapura 1997. Jeff De Luca menyampaikan satu set lima proses yang menutupi pengembangan model secara keseluruhan dan daftar, perencanaan, desain dan bangunan fitur. Proses pertama sangat dipengaruhi oleh pendekatan Peter Coad untuk pemodelan obyek. Proses kedua menggabungkan ide-ide Peter Coad tentang menggunakan daftar fitur untuk mengelola kebutuhan fungsional dan tugas-tugas pembangunan. Proses lainnya dan campuran proses menjadi suatu kesatuan kohesif merupakan hasil dari pengalaman Jeff De Luca. Sejak penggunaan sukses pada proyek Singapura, ada beberapa implementasi dari FDD.

Gambaran FDD pertama kali diperkenalkan kepada dunia dalam Bab 6 dari buku Modeling Jawa Warna dengan UML oleh Peter Coad, Lefebvre Eric dan Jeff De Luca pada tahun 1999. Kemudian, di Stephen Palmer dan buku Mac Felsing A Panduan Praktis untuk Pengembangan *Fitur-Driven* (diterbitkan pada 2002), deskripsi yang lebih umum FDD diberikan, seperti yang dipisahkan dari pemodelan Jawa dalam warna.

FDD asli dan terbaru proses dapat ditemukan di situs web Jeff De Luca di bawah wilayah 'Pasal'. Ada juga web komunitas yang tersedia di mana orang dapat mempelajari lebih lanjut tentang FDD, pertanyaan bisa ditanyakan, dan pengalaman, dan proses itu sendiri dibahas.

Karakteristik

penggunaan FDD sebaiknya digunakan jika; mempekerjakan 10 – 250 developer yang memiliki kemampuan teknis lebih dari rata-rata, dan jangan digunakan jika; jumlah tim kurang dari 10, tim sedang belajar menguasai pekerjaan dan jika kurang dukungan dari sistem. FDD lebih terhirarki daripada *Extreme Programming*, memiliki *class owenership* yang terpisah-pisah, sukses jika dalam rentang jumlah developer diatas rata-rata, klien tidak dilibatkan dalam seluruh urutan proses (hanya dalam proses 1, 2 dan 4) dan menghargai developer sebagai individu manusia yang menentukan sukses atau tidaknya pengembangan.

Kelebihan dan kekurangan

1) Kelebihan

- *User* dapat menggambarkan dengan mudah bentuk sistem.
- Dapat di organisasikan atau diatur ke dalam kelompok bisnis yang hirarki.
- Desain dan kode lebih mudah diperiksa secara efektif.
- Merancang proyek, penjadwalan dan jalur diarahkan oleh *feature*.

2) Kekurangan

Tidak ada

f. Agile Modeling

Agile Modeling adalah suatu metodologi yang praktis untuk proses dokumentasi dan pemodelan sistem perangkat lunak. *Agile Modeling* sendiri adalah kumpulan nilai-nilai, prinsip dan praktek-praktek untuk memodelkan perangkat lunak agar dapat diaplikasikan pada proyek pengembangan perangkat lunak secara efektif dan efisien.

Prinsip dalam *Agile Modeling* antara lain :

- Membuat model dengan tujuan: tentukan tujuan sebelum membuat model
- Menggunakan *multiple models*: tiap model mewakili aspek yang berbeda dari model lain.
- *Travel light*: simpan model-model yang bersifat jangka panjang saja
- Isi lebih penting dari pada penampilan: modeling menyajikan informasi kepada *audiens*
- yang tepat.
- Memahami model dan alat yang digunakan untuk membuat *software*
- Adaptasi secara lokal

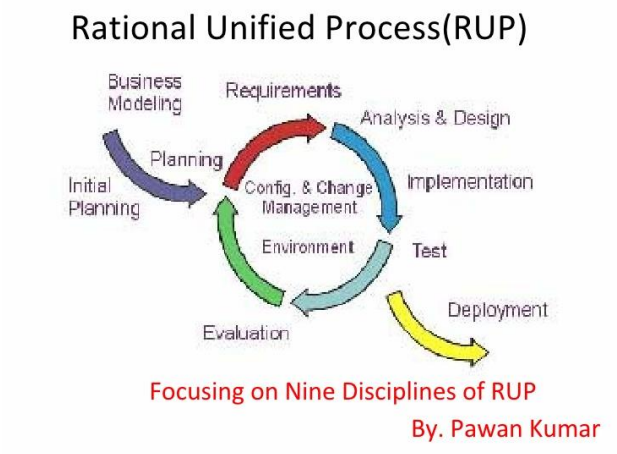
Dengan menggunakan *Agile Modeling*, diharapkan tim pengembang (developer) dapat fokus pada prinsip dan praktek pemodelan yang efektif.

g. Rational Unified Process

Rational Unified Process, adalah suatu kerangka kerja proses pengembangan perangkat lunak iteratif yang dibuat oleh *Rational Software*, suatu divisi

dari IBM sejak 2003. RUP bukanlah suatu proses tunggal dengan aturan yang konkret, melainkan suatu kerangka proses yang dapat diadaptasi dan dimaksudkan untuk disesuaikan oleh organisasi pengembang dan tim proyek perangkat lunak yang akan memilih elemen proses sesuai dengan kebutuhan mereka.

Model ini membagi suatu sistem aplikasi menjadi beberapa komponen sistem dan memungkinkan para developer aplikasi untuk menerapkan metode *iterative* (analisis, desain, implementasi dan pengujian) pada tiap komponen. Dengan menggunakan model ini, RUP membagi tahapan pengembangan perangkat lunaknya ke dalam 4 fase sebagai berikut.



- ***Inception***, merupakan tahap untuk mengidentifikasi sistem yang akan dikembangkan. Aktivitas yang dilakukan pada tahap ini antara lain mencakup analisis sistem *eksisting*, perumusan sistem target, penentuan arsitektur global target, identifikasi kebutuhan, perumusan persyaratan perumusan kebutuhan pengujian, pemodelan diagram UML, dan pembuatan dokumentasi.
- ***Elaboration***, merupakan tahap untuk melakukan desain secara lengkap berdasarkan hasil analisis di tahap *inception*. Aktivitas yang dilakukan pada tahap ini antara lain mencakup pembuatan desain arsitektur subsistem, desain komponen sistem, desain format data desain *database*, disain antarmuka/tampilan, desain peta aliran tampilan, penentuan *design pattern* yang digunakan, pemodelan diagram UML, dan pembuatan dokumentasi.
- ***Construction***, merupakan tahap untuk mengimplementasikan hasil desain dan melakukan pengujian hasil implementasi. Pada tahap awal *construction*, ada baiknya dilakukan pemeriksaan ulang hasil analisis dan desain, terutama desain pada domain perilaku (diagram *sequence*) dan domain struktural (diagram *class*, *component*, *deployment*). Apabila desain yang dibuat telah sesuai dengan analisis sistem, maka implementasi dengan bahasa pemrograman tertentu dapat dilakukan. Aktivitas yang dilakukan pada tahap ini antara lain mencakup pengujian hasil analisis dan desain (misal menggunakan *Class Responsibility Collaborator* untuk kasus pemrograman berorientasi obyek), pendataan kebutuhan implementasi lengkap (berpedoman pada identifikasi

kebutuhan di tahap analisis), penentuan *coding pattern* yang digunakan, pembuatan program, pengujian, optimasi program, pendataan berbagai kemungkinan pengembangan / perbaikan lebih lanjut, dan pembuatan dokumentasi.

- ***Transition,*** merupakan tahap untuk menyerahkan sistem aplikasi ke konsumen (*roll-out*), yang umumnya mencakup pelaksanaan pelatihan kepada pengguna dan testing beta aplikasi terhadap ekspetasi pengguna.

Kesimpulan

Agile Development Methods adalah sekelompok metodologi pengembangan perangkat lunak yang didasarkan pada prinsip-prinsip yang sama atau pengembangan sistem jangka pendek yang memerlukan adaptasi cepat dari pengembang terhadap perubahan dalam bentuk apapun. *Agile development methods* merupakan salah satu dari Metodologi pengembangan perangkat lunak yang digunakan dalam pengembangan perangkat lunak. *Agile* memiliki pengertian bersifat cepat, ringan, bebas bergerak. Sehingga saat membuat perangkat lunak dengan menggunakan *agile development methods* diperlukan inovasi dan responsibiliti yang baik antara tim pengembang dan klien agar kualitas dari perangkat lunak yang dihasilkan bagus dan kelincahan dari tim seimbang. *Agile Manifesto* merupakan nilai-nilai yang digunakan dalam mendasari berlangsungnya *Agile Software Development*. Dimana 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 Development Methods* dikembangkan karena pada metodologi tradisional terdapat banyak hal yang membuat proses pengembangan tidak dapat berhasil dengan baik sesuai tuntutan *user*. Metodologi ini sudah cukup banyak berkembang, di antaranya adalah *Extreme Programmning* (XP), *Adaptive Software Development* (ASD), *Dynamic Systems Development Method* (DSDM), *Scrum Methodology*, *Crystal*, *Feature Driven Development* (FDD), *Agile Modeling* (AM),

DAFTAR PUSTAKA

- https://www.dictio.id/t/apa-yang-dimaksud-dengan-adaptive-software-development-asd/15200
- https://www.dictio.id/t/apa-yang-dimaksud-dengan-dynamic-system-development-method-dsdm/15201/2
- https://www.dictio.id/t/apakah-yang-dimaksud-dengan-metode-scrum/302/4
- https://www.dictio.id/t/apa-yang-dimaksud-dengan-agile-modeling-am/15203/2
- http://www.materi-it.com/2014/11/mengenal-adaptive-software-development.html?id-