MAKALAH

REKAYASA PERANGKAT LUNAK

Disusun untuk memenuhi Tugas Mata Kuliah

Rekayasa Perangkat Lunak 1

Disusun Oleh :

Maria Agata	10110552
Arinten Dewi Hidayat	10110557
Andriano D. P	10110576
Toni Hartono	10110577
Herdi Julianto	10110578

JURUSAN TEKNIK INFORMATIKA

FAKULTAS TEKNIK DAN ILMU KOMPUTER

UNIVERSITAS KOMPUTER INDONESIA

2012

Daftar Isi

Daftar Isi

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Dalam dunia teknologi sekarang pengembangan dalam bidang informatikan telah mengalami perkembangan yang sangat pesat. Dengan perkembangan ini, dalam bidang informatika tidak hanya menghasilkan hanya dalam pengembangan program perangkat lunak saja, melainkan pengambangan dalam bidang suatu permodelan yang bersifat komplek.

Dalam pembuatan sebuah perangkat lunak yang haruslah memiliki Teknik analisa kebutuhan dan teknim permodelan yang baik, supaya terwujudnya suatu perangkat lunak yang baik. Dengan hal tersebut maka perlulah suatu pengenalan mengenai permodelan dalam suatu pembangunan suatu Perangkat Lunak *(Software)*. Terdapat banyak permodelan mengenai pembangunan suatu Perangkat lunak seperti SDLC dan Agile Model. Yang dimana dari setiap model ini memiliki macam macam model lainnya.

Berdasarkan tugas yang kami peroleh, kami hanya membatasi penjelasan mengenai permodelan ini, hanya memberikan konsep mengenai kekurangan, kelebihan dari *V model, Increment Model, Prototyping Model, Spiral Model* dan *Concurrent Development Model.* Memberikan penjelasan menganai *Agile Process* yang mencakupi *Extreme Programming (XP)* dan *SCRUM,* serta membandingkan antara RUP *(Rational Unified Proccess).*

1.2 Perumusan Masalah

Berdasarkan penjelasan yang terdapat pada latar belakang masalah diatas, kami dihadapkan untuk menganalisa mengenai kekurangan serta kelebihan dari permodelan perangkat lunak yakni *System Development Life Cycle* (SDLC) yang mencakupi *V model, Increment Model, Prototyping Model, Spiral Model* dan *Concurrent Development Model.* Memberikan penjelasan menganai *Agile Process* yang mencakupi *Extreme Programming (XP)* dan *SCRUM,* serta membandingkan antara RUP *(Rational Unified Proccess).*

1.3 Tujuan Pembuatan

Adapun tujuan pembuatan makalah ini adalah :

a. Untuk memenuhi tugas makalah dari mata kuliah Rekayasa Perangkat Lunak I (Teknik Berorientasi Objek) semester 5.
b. Memahami lebih mendalam akan konsep permodelan SDLC dan Agile Proccess baik dalam hal, penjelasa, kekurangan, kelebihan dan perbandingan diantara RUP dan SCRUM.
c. Memahami lebih mendalam akan konsep analisa dan pemodelan pada Agile Modeling.

1.4 Ruang Lingkup Pembahasan

Dalam makalah yang kami buat ini, kami hanya akan membahas secara khusus mengenai kekurangan dan kelebihan dari setiap macam model dari SDLC. Menjelaskan konsep dari Agile Proccess yakni Extreme Programming (XP) dan SCRUM, serta membandingkan model agile RUP dan SCRUM.

1.5 Metodologi Pemecahan Masalah

Dalam membahas permasalahan yang kami mengenai Kekurangan, kelebihan, penjelasan mengenai konsep XP dan SCRUM serta perbandingan antara RUP dan SCRUM ini, kami mendapatkan referensi berdasakan pencarian melalui internet. Serta melalui studi literatur yakni pengambilan informasi melalui pencarian dari buku.

BAB II

PEMBAHASAN

2.1 Model SDLC

2.1.1 V Model

V Model memiliki beberapa kelebihan. Kelebihan-kelebihan tersebut secara garis besar dapat dijelaskan seperti berikut:

- V Model sangat fleksibel. V Model mendukung *project tailoring* dan penambahan dan pengurangan *method* dan *tool* secara dinamik. Akibatnya sangat mudah untuk melakukan *tailoring* pada V Model agar sesuai dengan suatu proyek tertentu dan sangat mudah untuk menambahkan *method* dan *tool* baru atau menghilangkan *method* dan *tool* yang dianggap sudah *obsolete*.
- V Model dikembangkan dan di-*maintain* oleh publik. *User* dari V Model berpartisipasi dalam *change control board* yang memproses semua *change request* terhadap V Model.

V Model juga memiliki beberapa kekurangan. Kekurangan-kekurangan tersebut yaitu:

- V Model adalah model yang *project oriented* sehingga hanya bisa digunakan sekali dalam suatu proyek.
- V Model terlalu fleksibel dalam arti ada beberapa *activity* dalam V Model yang digambarkan terlalu abstrak sehingga tidak bisa diketahui dengan jelas apa yang termasuk dalam *activity* tersebut dan apa yang tidak.

2.1.2 Incremental Model

2.1.3 Model *Rapid Application Development* (RAD)

Rapid Application Development (RAD) adalah model proses pengembangan perangkat lunak yang bersifat inkrementalterutama untuk waktu pengerjaan yang pendek. Dodep RAD merupakan adaptasi dari permodelan *waterfall* versi kecepatan tinggi dan menggunakan model *waterfall* untuk pengembangan setiap komponen perangkat lunak.

Kelemahan dan Kelebihan

Model RAD memiliki kelemahan sebagai berikut :

- Untuk pembuatan sistem perangkat lunak dengan skala besar makamodel RAD akan memerlukan sumber daya manusia yang cukup besar untuk membentuk tim-tim yang mengembangkan komponen-komponen;
- Jika ada persetujuan untk mengembangkan perangkat lunak dengan cara cepat (*rapid)* maka proyek dengan model ini akangagal, karena akan membingungkan ketika mendefinisikan kebutuhan pelanggan;
- Jika sistem perangkat lunak yang akan dibuat tidak bisa dimodulkan (dibagi – bagi menjadi beberapa komponen) maka model RAD tidak dapat digunakan untuk membuat sistem perangkat lunak ini karena terlalu banyak campur tangan antar tim;
- Model RAD tidak cocok digunakan untuk sistem perangkat lunak yang memiliki resiko teknis sangat tinggi, misalnya manggunakan teknologi baru yang belum banyak dikenal dan dikuasai pengembang.

Selain itu, model RAD memiliki kelebihan sebagai berikut :

- Setiap fungsi mayor dapat dimodulkan dalam waktu tertentu kurang dari 3 bulan dandapat dibicarakan oleh tim RAD yang terpisah dan kemudian diintegrasikan sehinngawaktunya lebih efesien.
- RAD mengikuti tahapan pengembangan sistem sepeti umumnya, tetapi mempunyaikemampuan untuk menggunakan kembali komponen yang ada (reusable object) sehingga pengembang pengembang tidak perlu membuat dari awal lagi dan waktulebih singkat .

2.1.4 Prototype Model

Prototyping merupakan salah satu metode pengembangan perangat lunak yang banyak digunakan. Dengan metode prototyping ini pengembang dan pelanggan dapat salingberinteraksi selama proses pembuatan sistem.Seing terjadi seorang pelanggan hanya mendefinisikan secara umum apa yangdikehendakinya tanpa menyebutkan secara detal output apa saja yang dibutuhkan,pemrosesan dan data-data apa saja yang dibutuhkan. Sebaliknya disisi pengembang kurangmemperhatikan efesiensi algoritma, kemampuan sistem operasi dan interface yangmenghubungkan manusia dan komputer. Untuk mengatasi ketidakserasian antara pelanggan

dan pengembang , maka harusdibutuhakan kerjasama yanga baik diantara keduanya sehingga pengembang akanmengetahui dengan benar apa yang diinginkan pelanggan dengan tidak mengesampingkansegi-segi teknis dan pelanggan akan mengetahui proses-proses dalm menyelasaikan sistemyang diinginkan. Dengan demikian akan menghasilkan sistem sesuai dengan jadwal waktupenyelesaian yang telah ditentukan. Kunci agar model prototype ini berhasil dengan baik adalah dengan mendefinisikanaturan-aturan main pada saat awal, yaitu pelanggan dan pengembang harus setuju bahwaprototype dibangun untuk mendefinisikan kebutuhan. Prototype akan dihilangkan sebagianatau seluruhnya dan perangkat lunak aktual aktual direkayasa dengan kualitas danimplementasi yang sudah ditentukan.

Tahapan-tahapan Prototyping

Tahapan-tahapan dalam Prototyping adalah sebagai berikut:

1. Pengumpulan kebutuhanPelanggan dan pengembang bersama-sama mendefinisikan format seluruh perangkatlunak, mengidentifikasikan semua kebutuhan, dan garis besar sistem yang akan dibuat.
2. Membangun prototyping

   Membangun prototyping dengan membuat perancangan sementara yang berfokuspada penyajian kepada pelanggan (misalnya dengan membuat input dan formatoutput)
3. Evaluasi protoptyping

   Evaluasi ini dilakukan oleh pelanggan apakah prototyping yang sudah dibangun sudahsesuai dengan keinginann pelanggan. Jika sudah sesuai maka langkah 4 akan diambil.Jika tidak prototyping direvisi dengan mengulangu langkah 1, 2 , dan 3.
4. Mengkodekan sistem

   Dalam tahap ini prototyping yang sudah di sepakati diterjemahkan ke dalam bahasapemrograman yang sesuai
5. Menguji sistem

   Setelah sistem sudah menjadi suatu perangkat lunak yang siap pakai, harus ditesdahulu sebelum digunakan. Pengujian ini dilakukan dengan White Box, Black Box,Basis Path, pengujian arsitektur dan lain-lain
6. Evaluasi SistemPelanggan mengevaluasi apakah sistem yang sudah jadi sudah sesuai dengan yangdiharapkan . Juka ya, langkah 7 dilakukan; jika tidak, ulangi langkah 4 dan 5.
7. Menggunakan sistemPerangkat lunak yang telah diuji dan diterima pelanggan siap untuk digunakan .

Keunggulan dan Kelemahan Prototyping

Keunggulan prototyping adalah:

1. Adanya komunikasi yang baik antara pengembang dan pelanggan.
2. Pengembang dapat bekerja lebih baik dalam menentukan kebutuhan pelanggan
3. Pelanggan berperan aktif dalam pengembangan sistem
4. Lebih menghemat waktu dalam pengembangan sistem
5. Penerapan menjadi lebih mudah karena pemakai mengetahui apa yang diharapkannya.

Kelemahan prototyping adalah :

1. Pelanggan kadang tidak melihat atau menyadari bahwa perangkat lunak yang ada belum mencantumkan kualitas perangkat lunak secara keseluruhan dan juga belummemikirkan kemampuan pemeliharaan untuk jangja waktu lama.
2. pengembang biasanya ingin cepat menyelesaikan proyek. Sehingga menggunakanalgoritma dan bahasa pemrograman yang sederhana untuk membuat prototypinglebih cepat selesai tanpa memikirkan lebih lanjut bahwa program tersebut hanyamerupakan cetak biru sistem .
3. Hubungan pelanggan dengan komputer yang disediakan mungkin tidak mencerminkan teknik perancangan yang baik Prototyping bekerja dengan baik pada penerapan-penerapan yang berciri sebagai berikut:

    1. Resiko tinggi Yaitu untuk maslaha-masalah yang tidak terstruktur dengan baik, adaperubahan yang besar dari waktu ke waktu, dan adanya persyaratan data yang tidak menentu.
    2. Interaksi pemakai penting . Sistem harus menyediakan dialog on-line antarapelanggan dan komputer.
    3. Perlunya penyelesaian yang cepat
    4. Perilaku pemakai yang sulit ditebak
    5. Sitem yang inovatif. Sistem tersebut membutuhkan cara penyelesaian masalah dan penggunaan perangkat keras yang mutakhir.
    6. Perkiraan tahap penggunaan sistem yang pendek

2.1.5 Model Spiral

Model spiral pada awalnya diusulkan oleh Boehm, adalah model proses perangkatlunak evolusioner yang merangkai sifat iteratif dari prototype dengan cara kontrol dan aspek sistematis

model sequensial linier.Model iteratif ditandai dengan tingkah laku yang memungkinkan pengembangmengembangkan versi perangkat lunak yang lebih lengkap secara bertahap. Perangkat lunak dikembangkan dalam deretan pertambahan. Selama awal iterasi, rilis inkremantal bisaberupa model/prototype kertas, kemudian sedikit demi sedikit dihasilkan versi sistem yanglebih lengkap.

Tahapan-Tahapan Model Spiral

Model spiral dibagi menjadi enam wilayah tugas yaitu:

1. Komunikasi pelanggan

   Yaitu tugas-tugas untuk membangun komunikasi antara pelanggan dan kebutuhan-kebutuhan yang diinginkan oleh pelanggan.

2. Perencanaan

   Yaitu tugas-tugas untuk mendefinisikan sumber daya, ketepatan waktu, dan proyek informasi lain yg berhubungan.

3. Analisis Resiko

   Yaitu tugas-tugas yang dibutuhkan untuk menaksir resikomanajemen dan teknis.

4. Perekayasaan

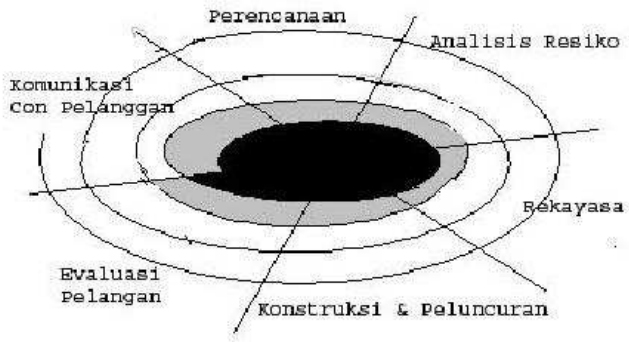
   Yaitu tugas yang dibutuhkan untuk membangun satu atau lebih representasi dari apikasi tersebut.

5. Konstruksi dan peluncuran

   Yaitu tugas-tugas yang dibutuhkan untuk mengkonstruksi, menguji, memasang , danmemberi pelayanan kepada pemakai.

6. Evaluasi Pelanggan

   Yaitu tugas-tugas untuk mendapatkan umpan balik dari pelanggan.



Gambar 2. Model Spiral

Dari gambar tersebut, proses dimulai dari inti bergerak searah dengan jarum jam mengelilingi spiral. Lintasan pertama putaran menghasilkan perkembangan spesifikasiproduk. Putaran selanjutnya digunakan untuk mengembangkan sebuah prototype, dan secaraprogresif mengembangkan versi perangkat lunak yang lebih canggih. Masing-masinglintasan yang melalui daerah perencanaan menghasilkan penyesuaian pada rencanan proyek.Biaya dan jadwal disesuaikan berdasarkan umpan balik yang disimpulakan dari evaluasi pelanggan. Manajer proyek akan menambah jumlah iterasi sesuai dengan yang dibutuhkan.

Kelebihan dan Kelemahan Model Spiral

a. Kelebihan model Spiral :

- Dapat disesuaikan agar perangkat lunak bisa dipakai selama hidup perangkat lunak komputer.
- Lebih cocok untuk pengembangan sistem dan perangkat lunak skala besar.
- Pengembang dan pemakai dapat lebih mudah memahami dan bereaksi terhadapresiko setiap tingkat evolusi karena perangkat lunak terus bekerja selama proses .
- Menggunakan prototipe sebagai mekanisme pengurangan resiko dan pada setiapkeadaan di dalam evolusi produk.
- Tetap mengikuti langkah-langkah dalam siklus kehidupan klasik dan memasukkannyake dalam kerangka kerja iteratif .
- Membutuhkan pertimbangan langsung terhadp resiko teknis sehingga mengurangiresiko sebelum menjadi permaslahan yang serius.

b. Kelemahan model Spiral:

- Sulit untuk menyakinkan pelanggan bahwa pendekatan evolusioner ini bisa dikontrol.
- Memerlukan penaksiran resiko yang masuk akal dan akan menjadi masalah yangserius jika resiko mayor tidak ditemukan dan diatur.
- Butuh waktu lama untuk menerapkan paradigma ini menuju kepastian yang absolutD.

2.1.6 Concurrent Development Model

Concurrent Development Model merupakan

Kelebihan dan Kelemahan Concurrent Development Model

Kelebihan yang dimiliki oleh model ini adalah :

- Proses Concurrent Development Model ini berlaku untuk semua jenis pengembangan perangkat lunak dan memberikan gambaran yang akurat tentang keadaan sekarang dari suatu proyek.

Kekurangan yang dimiliki oleh model ini adalah :

- statenya sangat banyak sehingga membutuhkan waktu lebih banyak.

2.2 Agile Proccces Model

Agile Software development adalah salaha satu metodologi dalam pengembangan sebuah perangkat lunak. Kata *Agile* berarti bersifat cepat, ringan, bebas bergerak, waspada.Kata ini digunakan sebagai kata yang menggambarkan konsep model proses yang berbeda dari konsep model – model proses yang sudah ada. Konsep agile software developoment dicetuskan oleh Kent Beck dan 16 rekannya dengan mengatakan bahwa Agile Software Development adalah cara membangun software dengan melakukannya dan membantu orang lain membangunnya sekaligus.

Dalam hal ini Agile Process model terdiri dari 5 macam model, yakni :

1. **Extreme Programming (XP)**
2. Adaptive Software Development (ASD)
3. Dinamic System Development Method
4. **SCRUM**
5. Agile Model

2.2.1 Extreme Programming (XP)

Extreme Programming (XP) merupakan salah satu metodologi dalam rekayasa perangkat lunak dan juga merupakan satu dari beberapa *agile software development methotodogies* yang berfokus pada *coding* sebagai aktivitas utama di semua tahap pada siklus pengembangan yang lebih responsive terhadap kebutuhan costumer (“agile”) di bandingkan dengan metode – metode tradisional sambil membangun suatu software dengan kualitas yang lebih baik, selain itu *extreme programming* meliputi seluruh area pengembangan perangkat lunak.

Model agile process ini dikembangkan oleh Kent Beck dan Ward Cunningham pada bulan maret 1996. Model Extreme Programming (XP) ini merupakan yang terpopuler dari beberapa metodologi pengembangan software yang dipakai untuk mengimplementasikan proyek pengembangan perangkat lunak.

Tujuan Extreme Programming

Tujuan utama yang ada pada extreme programming adalah untuk menurunkan biaya dari adanya perubahan pembangunan *software.* Dalam pengembangan sistem tradisional, kebutuhan sistem di tentukan awal pengembangan proyek dan bersifat *fixed.* Extreme programming diarahkan untuk menurunkan biaya dari adanya perubahan dengan memperkenalkan nilai-nilai basis dasar, prinsip dan praktis. Dengan menerapkan extreme xp, pembangunan suatu sistem haruslah lebih fleksibel terhadap terjadinga suatu perubahan.

Nilai-nilai Dasar XP

Berikut adalah nilai-nilai mendasar yang menjadi roh dari XP pada setiap tahapan proses pengembangan perangkat lunak:

1. Communication (Komunikasi)

Tugas utama developer dalam membangun suatu sistem perangkat lunak adalah mengkomunikasikan kebutuhan sistem kepada pengembang perangkat lunak. Komunikasi dalam XP dibangun dengan melakukan pemrograman berpasangan (pair programming). Developer didampingi oleh pihak klien dalam melakukan coding dan unit testing sehingga klien bisa terlibat langsung dalam pemrograman sambil berkomunikasi dengan developer. Tujuannya untuk memberikan pandangan pengembang sesuai dengan pandangan pengguna sistem.

2. Simplicity (Kesederhanaan)

Extreme Programming XP mencoba untuk mencari solusi paling sederhana dan praktis. Perbedaan metode ini dengan metodologi pengembangan sistem konvensional lainnya terletak pada proses desain dan coding yang terfokus pada kebutuhan saat ini daripada kebutuhan besok, seminggu lagi atau sebulan lagi. Lebih baik melakukan hal yang sederhana dan mengembangkannya besok jika diperlukan.

3. Feedback (Masukan)

Hal ini diperlukan untuk mengetahui kemajuan dari proses dan kualitas dari aplikasi yang dibangun. Informasi ini harus dikumpulkan setiap interval waktu yang singkat secara konsisten. Ini dimaksudkan agar hal-hal yang menjadi masalah dalam proses pengembangan dapat diketahui sedini mungkin. Setiap feed back ditanggapi dengan melakukan tes, unit test atau system integration dan jangan menunda karena biaya akan membengkak (uang, tenaga, waktu).

4. Courage (Keberanian)

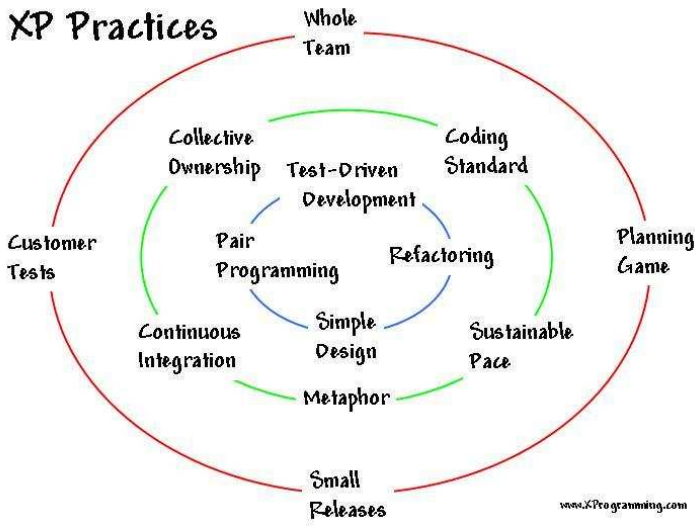
Berani mencoba ide baru. Berani mengerjakan kembali dan setiap kali kesalahan ditemukan, langsung diperbaiki. Contoh dari courage adalah komitmen untuk selalu melakukan design dan coding untuk saat ini dan bukan untuk esok. Ketika ada kode yang terlalu rumit, sulit dibaca dan dipahami, tidak sesuai dengan kemauan pelanggan, dll maka seharusnya kode program seperti itu di refactor (kalau perlu dibangun ulang). Hal ini menjadikan pengembang merasa nyaman dengan *refactoring* program ketika diperlukan.

5. Respect (Menghormati)

Pentingnya respect terhadap anggota team lainnya karena dengan siklus pendek dan integrasi continue, programmer tidak boleh melakukan perubahan yang dapat merusak kompilasi dan menyebabkan keberadaan unit uji gagal atau memperlambat kerja team. Respects tiap individu akan selalu menghasilkan kualitas tinggi.

Aspek Dasar XP

Aspek dasar XP terdiri dari berbagai teknik atau metode yang diterapkan Beck dan Jeffries pada *C3 Project*. Teknik-teknik tersebut dapat diamati pada gambar berikut ini:



Gambar 2 : Exterema Programming (XP) practices

1. The Planning Game

Pendekatan XP dalam perencanaan sangat mirip dengan metode yang diterapkan pada *RAD (Rapid Application Development)*. Proses pendek dan cepat, mengutamakan aspek teknik, memisahkan unsur bisnis dengan unsur teknis dan pertemuan intensif antara klien dengan *developer.* Pada XP proses ini menggunakan terminologi "*game*" karena Beck menyarankan untuk menggunakan teknik *score card* dalam menentukan *requirements*. Semakin sulit aspek teknis yang dibutuhkan semakin tinggi pula skor pada kartu rencana tersebut.

2. Small Releases

Setiap *release* dilakukan dalam lingkup sekecil mungkin pada XP. Setiap *developer* menyelesaikan sebuah unit atau bagian dari perangkat lunak maka hasil tersebut harus segera dipresentasikan dan didiskusikan dengan klien. Jika memungkinkan untuk menerapkan unit tersebut pada perusahaan, hal itu juga dapat dilakukan sekaligus sebagai tes awal dari penerapan keseluruhan sistem. Kendati demikian hal ini tidak selalu perlu dilakukan karena harus dihitung terlebih dahulu sumberdaya yang dibutuhkan. Apakah lebih menguntungkan langsung melakukan tes terhadap unit tersebut atau melakukan tes setelah unit tersebut terintegrasi secara sempurna pada sistem.

3. Metaphor

Metaphor pada dasarnya sama dengan arsitektur perangkat lunak. Keduanya menggambarkan visi yang luas terhadap tujuan dari pengembangan perangkat lunak. Beck sendiri seperti para penandatangan Agile Manifesto lainnya bercita-cita menyederhanakan proses pengembangan perangkat lunak yang saat ini sudah dianggap terlalu rumit. Arsitektur yang saat ini banyak berisi diagram dan kode semacam UML dianggap terlalu rumit untuk dimengerti, terutama oleh klien. *Metaphor*, walaupun mirip dengan arsitektur lebih bersifat naratif dan deskriptif. Dengan demikian diharapkan komunikasi antara klien dengan *developer* akan berlangsung lebih baik dan lancar dengan penggunaan *metaphor*.

4. Simple Design

Sebagai salah seorang penandatangan Agile Manifesto, Beck adalah seorang yang tidak menyukai desain yang rumit dalam sebuah pengembangan perangkat lunak. Tidak heran jika dia memasukkan *Simple Design* sebagai salah satu unsur XP. Pada XP desain dibuat dalam lingkup kecil dan sederhana. Tidak perlu melakukan antisipasi terhadap berbagai perubahan di kemudian hari. Dengan desain yang simpel apabila terjadi perubahan maka membuat desain baru untuk mengatasi perubahan tersebut dapat dengan mudah dilakukan dan resiko kegagalan desain dapat diperkecil.

5. Refactoring

Refactoring adalah salah satu aspek paling khas dari XP. *Refactoring* seperti didefinisikan oleh Martin Fowler adalah ”Melakukan perubahan pada kode program dari perangkat lunak dengan tujuan meningkatkan kualitas dari struktur program tersebut tanpa mengubah cara program tersebut bekerja”. *Refactoring* sendiri sangat sesuai untuk menjadi bagian XP karena *Refactoring* mengusung konsep penyederhanaan dari proses desain maupun struktur baris kode program. Dengan *Refactoring* tim pengembang dapat melakukan berbagai usaha untuk meningkatkan kualitas program tanpa kembali mengulang-ulang proses desain. Fowler adalah salah satu kolega dekat dari Kent Beck karena itu tidak mengherankan bahwa cara berpikir mereka terhadap proses pengembangan perangkat lunak sangat mirip satu dengan lainnya.

6. Testing

XP menganut paradigma berbeda dalam hal tes dengan model pengembangan perangkat lunak lainnya. Jika pada pengembangan perangkat lunak lainnya tes baru dikembangkan setelah perangkat lunak selesai menjalani proses *coding* maka pada XP tim pengembang harus membuat terlebih dahulu tes yang hendak dijalani oleh perangkat lunak. Berbagai model tes yang mengantisipasi penerapan perangkat lunak pada sistem dikembangkan terlebih dahulu.

Saat proses *coding* selesai dilakukan maka perangkat lunak diuji dengan model tes yang telah dibuat tersebut. Pengetesan akan jauh lebih baik apabila dilakukan pada setiap unit perangkat lunak dalam lingkup sekecil mungkin daripada menunggu sampai seluruh perangkat lunak selesai dibuat. Dengan memahami tahap ini kita dapat melihat bahwa siklus pada XP adalah *requirement analysis* → *test* → *code* → *design*. Sekilas terlihat hal ini tidak mungkin dilakukan tetapi pada kenyataannya memang gambaran inilah yang paling dapat menjelaskan tentang XP.

7. Pair Programming

Pair programming adalah melakukan proses menulis program dengan berpasangan. Dua orang programer saling bekerjasama di komputer yang sama untuk menyelesaikan sebuah unit. Dengan melakukan ini maka keduanya selalu dapat berdiskusi dan saling melakukan koreksi apabila ada kesalahan dalam penulisan program. Aspek ini mungkin akan sulit dijalankan oleh para programer yang memiliki ego tinggi dan sering tidak nyaman untuk berbagi komputer bersama rekannnya.

8. Collective Ownership

Tidak ada satupun baris kode program yang hanya dipahami oleh satu orang programer. XP menuntut para programer untuk berbagi pengetahuan untuk tiap baris program bahkan beserta hak untuk mengubahnya. Dengan pemahaman yang sama terhadap keseluruhan program, ketergantungan pada programer tertentu ataupun berbagai hambatan akibat perbedaan gaya menulis program dapat diperkecil. Pada level yang lebih tinggi bahkan dimungkinkan para programer dapat bertukar unit yang dibangunnya.

9. Coding Standards

Pair programming dan *collective ownership* hanya akan dapat berjalan dengan baik apabila para programer memiliki pemahaman yang sama terhadap penulisan kode program. Dengan adanya *coding standards* yang telah disepakati terlebih dahulu maka pemahaman terhadap program akan menjadi mudah untuk semua programer dalam tim. Hal ini dapat diterapkan sebagai contoh pada penamaan variabel dan penggunaan tipe data yang sama untuk tiap elemen semua *record* atau *array* pada program.

10. Continous Integration

Melakukan *build* setiap hari kerja menjadi sebuah model yang disukai oleh berbagai tim pengembang perangkat lunak. Hal ini terutama didorong oleh keberhasilan penerapan sistem ini oleh Microsoft dan telah sering dipublikasikan. Dengan melakukan *build* sesering mungkin berbagai kesalahan pada program dapat dideteksi dan diperbaiki secepat mungkin. Apabila

banyak tim pengembang perangkat lunak meyakini bahwa *build* sekali sehari adalah minimum maka pada XP hal tersebut adalah maksimum. Pada XP tim disarankan untuk melakukan *build* sesering mungkin misalnya setiap 4 jam atau bahkan lebih cepat lagi.

11. 40-hours Week

Beck berpendapat bekerja 8 jam sehari dan 5 hari seminggu adalah maksimal untuk tiap programer. Lebih dari itu programer akan cenderung membuat berbagai *error* pada baris-baris kode programnya karena kelelahan.

12. On-Site Customer

Sebuah pendekatan klasik, di mana XP menganjurkan bahwa ada anggota dari klien yang terlibat pada proses pengembangan perangkat lunak. Yang lebih penting lagi ia harus ada di tempat pemrogaman dan turut serta dalam proses build dan test yang dilakukan. Apabila ada kesalahan dalam pengembangan diharapkan klien dapat segera memberikan masukan untuk koreksinya.

Keunggulan dan Kekurangan *Extreme Programming*

Keunggulan yang dimiliki adalah :

a. Menjalin komunikasi yang baik dengan client.
b. Meningkatkan komunikasi dan sifat saling menghargai antar developer.

Kelemahan yang dimiliki adalah :

a. Developer harus selalu siap dengan perubahan karena perubahan akan selalu diterima.
b. Tidak bisa membuat kode yang detail di awal (prinsip simplicity dan juga anjuran untuk melakukan apa yang diperlukan hari itu juga).

2.2.2 SCRUM

Scrum adalah sebuah pendekatan tangkas untuk pengembangan perangkat lunak. Scrum merupakan suatu kerangka kerja. Jadi, bukannya menyediakan deskripsi rinci tentang bagaimana segala sesuatu yang harus dilakukan pada proyek seperti diserahkan kepada tim pengembangan perangkat lunak pada umumnya. Hal ini dilakukan supaya tim akan tahu bagaimana cara terbaik untuk memecahkan masalah yang mereka disajikan. Inilah sebabnya mengapa, misalnya, rapat perencanaan sprint digambarkan dalam bentuk hasil yang diinginkan (komitmen

untuk mengatur fitur yang akan dikembangkan di sprint berikutnya), bukan definisi Tugas, kriteria Validasi, dllseperti yang akan disediakan dalam metodologi yang lain.

Scrum bergantung pada pengorganisasian-diri, tim lintas fungsional. Tim scrum adalahmengorganisir diri dalam bentuk tidak ada pemimpin tim secara keseluruhan yang memutuskanmana orang yang akan melakukan tugas atau bagaimana suatu masalah akan dipecahkan. Mereka adalah isu-isu yang ditentukan oleh tim secara keseluruhan. Tim ini lintas fungsional sehingga setiap orang perlu untuk mengambil fitur dari ide untuk diimplementasi.

Tim-tim pengembangan scrum yang didukung oleh dua orang tertentu: *Scrum Master* dan pemilik produk (*product owner*). Scrum Master dapat dianggap sebagai pelatih bagi tim, membantuanggota tim menggunakan kerangka Scrum untuk tampil di tingkat tertinggi. Pemilik produk mewakili bisnis, pelanggan atau pengguna dan memandu tim ke arah pegembangan produk yang tepat.

Proyek Scrum membuat kemajuan dalam serangkaian sprint, yang timeboxed iterasi tidak lebih dari sebulan panjang. Pada awal sprint, anggota tim berkomitmen untuk memberikan beberapa nomor fitur yang terdaftar di product backlog proyek. Pada akhir sprint, fitur inidilakukan – mereka diimplementasikan, diuji, dan diintegrasikan ke dalam produk berkembangatau sistem. Pada akhir sprint tinjauan sprint dilakukan selama tim menunjukkan fungsi barukepada pemilik produk dan pemangku kepentingan lain yang memberikan umpan balik yang dapat mempengaruhi sprint berikutnya.

Dalam hal ini Scrum memiliki prinsip sebagai berikut :

- Ukuran tim yang kecil melancarkan komunikasi, mengurangi biaya,dan memberdayakan satu sama lain.
- Proses dapat beradaptasi terhadap perubahan teknis 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 kepanpun di perlukan.

2.3 RUP dan SCRUM

Perbandingan RUP antara SCRUM sebagai berikut :

No	Faktor	RUP	SCRUM
1	Kompleksitas, dan Kelengakapan	Lebih besar, lebih kompleks dan lebih lengkap	Lebih sederhana, lebih simple
2	Penyesuaian	Perlu down sized atau penyederhanaan penerapan RUP	Perlu down size atau penyederhanaan penerapan RUP
3	Ketersediaan	Komersial, didukung Rational Software dan IBM	Freeware, dimaintain oleh community
4	Fokus	Manajemen Kelengkapan proses, artefak , activity dan role	Berfokus pada pengerjaan proyek pengembangan software
5	Pendekatan	Top-down solution	Bottom-up approach
6	Team	Ada pembagian responsibility dan tugas	Team work sangat dipertahnakan
7	Siklus	Siklus sudah pasti dengan 4 tahap, tapi beberapa “workflow” bisa konkuren	Setiap iterasi adalah sebuah siklus yang komplit
8	Perencanaan	Perencaan proyek sudah terencana dan sesauai dengan iterasi yang digunakan. Memiliki dead-line.	Tidak memiliki dead-line. Setiap perencanaan selanjutnya tergantunga pada iterasi saat ini.
9.	Target	Target sudah pasti, jelas, dan tercatat.	Tidak memiliki target melainkan membandingkannya dengan iterasi terdahulu.
10.	Tipe Proyek/Produk	Direkomendasikan untuk proyek besar dan memiliki jangka waktu yang panjang.	Direkomendasikan untuk pengembangan yang cepat dan untuk organisasi yang tidak memperdulikan deadline.

BAB III

PENUTUP

3.1 Kesimpulan

SCRUM dan RUP merupakan metodologi pengembangan software yang memiliki persamaan dan perbedaan. Keduanya memiliki kelebihan dan kekurangan masing-masing. RUP dan SCRUM memiliki tipe proyek yang berbeda sehingga keduanya akan tepat apabila diterapkan pada proyek yang sesuai dengan tipikal masing-masing metodologi.

DAFTAR PUSTAKA

A.S Rosa,Shalahuddin.M.2009.”Modul Rekayasa Perangkat Lunak (Terstruktur dan Berorientasi Objek).Bandung:Modula Bandung.

Davor Gornik , IBM Rational Unified Process : Best Practices for Software Development Teams.

www.agilemodelling.com