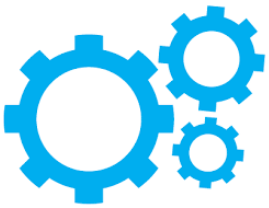
PERBANDINGAN ANTARA TIGA SDLC *METHODOLOGY*, PARALLEL, ITERATIVE DAN *AGILE* DEVELOPMENT

Nanny Raras Setyoningrum
Program Studi Magister Teknik Informatika
STMIK Amikom Yogyakarta
(ra2s_ukhti@yahoo.co.id)



ABSTRAK

Perkembangan ilmu pengetahuan dan teknologi telah membuat komputer menjadi sesuatu yang sangat penting dalam kehidupan saat ini. Pemanfaatan teknologi informasi terkomputerisasi secara luas digunakan dalam berbagai bidang kehidupan misalnya perdagangan, pendidikan, industri dan lain-lain. Komputer dapat menghemat waktu dalam membantu memecahkan persoalan yang kompleks. Sebuah program perangkat lunak diperlukan dalam mendukung pengembangan sistem informasi, banyak perusahaan memproduksi program perangkat lunak untuk memfasilitasi pekerjaan baik di pemerintahan, bank, kantor dan lain-lain. SDLC (Sistem Development Life Cycle) dalam rekayasa sistem dan rekayasa perangkat lunak adalah proses pembuatan dan pengubahan sistem serta model dan metodologi yang digunakan untuk mengembangkan sistem tersebut. Ada berbagai model telah banyak digunakan untuk mengembangkan perangkat lunak. Dalam paper ini akan dijelaskan tiga methodology yaitu methodology parallel, methodology iterative dan methodology agile development.

Kata Kunci: SDLC Models, rekayasa perangkat lunak, Methodology Parallel, Iterative dan Agile Development.

1. PENDAHULUAN

1.1 Metodologi Pengembangan Sistem

Sistem *Development Life Cycle* atau SDLC dalam rekayasa sistem dan rekayasa perangkat lunak adalah sebuah proses pembuatan dan pengubahan sistem serta model dan metodologi yang digunakan untuk mengembangkan sistem. Konsep ini umumnya merujuk pada sistem komputer atau informasi.

Dalam rekayasa perangkat lunak, konsep SDLC mendasari berbagai jenis metodologi pengembangan perangkat lunak. Metodologi–metodologi ini membentuk suatu kerangka kerja untuk perencanaan dan pengendalian pembuatan sistem informasi, yaitu proses pengembangan perangkat lunak.

Dapat dikatakan SDLC merupakan usaha bagaimana sebuah sistem informasi dapat mendukung kebutuhan bisnis, rancangan dan pembangunan sistem serta *delivering*-nya kepada pengguna.

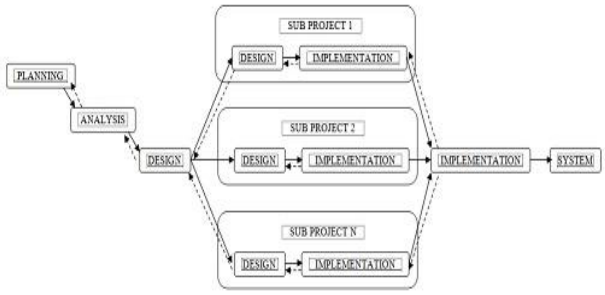
1.2 Tahapan Pengembangan Sistem

Secara umum, tahapan SDLC meliputi proses perencanaan, analisis, desain dan implementasi. Proses perencanaan biasanya lebih menekankan pada alasan mengapa sebuah sistem harus dibuat. Tahapan perencanaan ini kemudian dilanjutkan dengan proses analisis yang lebih menekankan pada siapa, apa, kapan dan dimana sebuah sistem akan dibuat. Proses desain lebih menekankan kepada bagaimana sistem akan berjalan. Pada tahap akhir dilanjutkan dengan fase implementasi yaitu proses *delivery*-nya kepada pengguna.

Pendekatan formal tahapan pengembangan sistem ini disebut metodologi. Saat ini terdapat berbagai macam metodologi dalam pengembangan sistem. Kita dapat memilih metodologi yang tepat sesuai dengan kebutuhan proses bisnis dan data yang mendukungnya.

2. *METHODOLOGY* PARALLEL

Methodology parallel termasuk salah satu *methodology* tradisional dalam pengembangan sistem informasi atau rekayasa perangkat lunak. *Methodology* ini mencoba mengatasi masalah lamanya waktu antara fase analisis dengan implementasi sistem. Desain secara umum akan ditampilkan terlebih dahulu kemudian proyek dibagi-bagi menjadi beberapa sub proyek yang dikerjakan secara bersama-sama secara parallel. Jadi, parallel development memungkinkan beberapa fase dilakukan secara bersama-sama untuk mempersingkat waktu.



Gambar 1. *Parallel methodology*

Kelebihan :

Jadwal waktu yang dibutuhkan untuk mengerjakan proyek menjadi lebih cepat, hanya ada sedikit kesempatan jika terjadi perubahan rencana pada lingkungan bisnis yang cepat berubah dapat menyebabkan pengerjaan kembali.

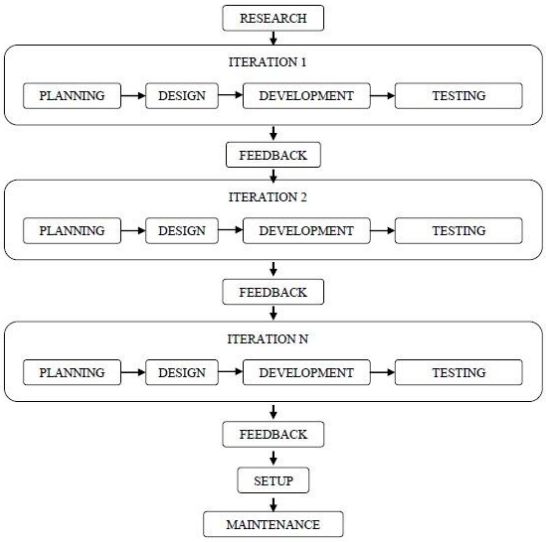
Kekurangan :

1. Seperti *methodology waterfall*, *methodology* ini akan mengalami masalah pada lamanya pekerjaan jika ada subproyek yang bermasalah yang berimbas pada sub proyek yang lain.
2. Pada akhir proyek, integrasi hasil kerja dari sub proyek juga dapat menimbulkan masalah.
3. Tidak cocok untuk proyek-proyek perangkat lunak yang kompleks.

3. METHODOLOGY ITERATIVE

Permasalahan pada model *waterfall* menciptakan suatu permintaan akan suatu *methodology* baru dari sistem yang dapat memberikan hasil yang lebih cepat dan menawarkan fleksibilitas yang lebih besar. Dengan *methodology* iterative, proyek dibagi menjadi subproyek-subproyek [1]. Hal ini memungkinkan tim development untuk menunjukkan hasil sebelumnya dan mendapatkan *feedback* yang berharga dari pengguna

sistem. Setiap iterasi sebenarnya adalah sebuah proses mini-*Waterfall* dengan *feedback* dari satu fase yang memberikan informasi penting untuk desain tahap berikutnya. *Methodology* ini membangun sebuah implementasi parsial dari keseluruhan sistem. Kemudian perlahan-lahan menambah fungsionalitas.



Gambar 2. *Iterative methodology*

4. METHODOLOGY AGILE DEVELOPMENT

Perkembangan dalam *methodology* pengembangan perangkat lunak saat ini telah mengalami banyak perubahan untuk menutupi kelemahan dari *methodology-methodology* sebelumnya. Jika kita melihat ke belakang, *methodology* yang digunakan tidak mampu menangani kemungkinan perubahan atau penambahan requirement pada saat proses pengembangan perangkat lunak berlangsung. Hal inilah yang menjadi pendorong munculnya *methodology* baru dalam pengembangan perangkat lunak. Untuk menangani kelemahan tersebut diperkenalkan *methodology* baru pada dekade 90-an, yakni *agile* methods.

Saat ini hasil dari perkembangan dari *agile* methods sudah cukup banyak, diantaranya:

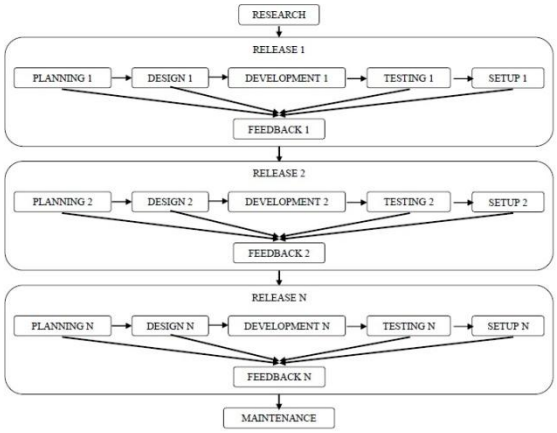
1. *eXtreme Programming*
2. *Scrum Methodology*
3. *Crystal Family*
4. *Dynamic* Sistem *Development Method*
5. *Adaptive Software Development*
6. *Feature Driven Development*

Arti kata *agile* sendiri berarti tangkas, cepat, atau ringan. Agility merupakan *methodology* yang ringan dan cepat dalam pengembangan perangkat lunak. *Agile Alliance* mendefinisikan 12 prinsip untuk mencapai proses yang termasuk dalam *agility*:

1. Prioritas tertinggi adalah memuaskan pelanggan melalui penyerahan awal dan berkelanjutan perangkat lunak yang bernilai.
2. Menerima perubahan *requirements* meskipun perubahan tersebut diminta pada akhir pengembangan.
3. Memberikan perangkat lunak yang sedang dikerjakan dengan sering, beberapa minggu atau beberapa bulan, dengan pilihan waktu yang paling singkat.
4. Pihak bisnis dan pengembang harus bekerja sama setiap hari selama pengembangan berjalan.
5. Bangun proyek dengan individu-individu yang bermotivasi tinggi dengan memberikan lingkungan dan dukungan yang diperlukan, dan mempercayai mereka sepenuhnya untuk menyelesaikan pekerjaannya.
6. *Methodology* yang paling efektif dan efisien dalam menyampaikan informasi kepada tim pengembangan adalah dengan komunikasi langsung *face-to-face*.
7. Perangkat lunak yang dikerjakan merupakan pengukur utama kemajuan.
8. Proses *agile* memberikan proses pengembangan yang bisa ditopang. Sponsor, pengembang, dan *user* harus bisa menjaga ke-konstanan langkah yang tidak pasti.
9. Perhatian yang terus menerus terhadap rancangan dan teknik yang baik meningkatkan *agility*.
10. Kesederhanaan –seni untuk meminimalkan jumlah pekerjaan– adalah penting.
11. Arsitektur, *requirements*, dan rancangan terbaik muncul dari tim yang mengatur Sendiri [2].

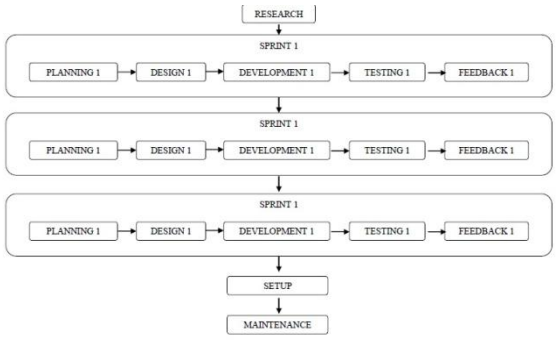
12. Pada interval reguler tertentu, tim merefleksikan bagaimana menjadi lebih efektif, kemudian menyesuaikannya.

Methodology yang berkembang telah beralih ke *methodology* yang lebih sederhana dengan mengutamakan fleksibilitas dan dengan lingkungan yang cepat berubah-ubah. *Methodology agile* yang sering digunakan adalah *methodology extreme programming* dan *scrum*. *Extreme programming* didasarkan pada *planning game* dan panduan dalam pengkodean program.



Gambar 3. *Extreme programming methodology*

Pendekatan dengan *scrum* telah dikembangkan untuk mengelola proses pengembangan perangkat lunak di lingkungan yang berubah-ubah berdasarkan fleksibiltas, kemampuan beradaptasi, dan produktivitas. *Scrum* adalah sebuah proses kerangka kerja yang disusun untuk menunjang pengembangan produk yang kompleks.



Gambar 4. *Scrum methodology*

Scrum terdiri dari tim *scrum* beserta peran-peran yang diperlukan, acara, artefak dan aturan main. Setiap komponen di dalam kerangka kerja ini memiliki tujuan tertentu dan peran penting terhadap keberhasilan dari jalannya proses *Scrum* [3]. *Scrum* fokus ke praktek manajement dan organisasi sedangkan XP sebagian besar fokus kepada praktek pemrograman. Oleh karena itulah keduanya bisa digabungkan dengan baik, karena keduanya mengatasi area yang berbeda dan saling melengkapi.

Mari kita memeriksa *agile manifesto*, yang menyatakan dengan tegas bahwa empat nilai utama dari *methodology agile* adalah:

1. Individu dan interaksi lebih dari proses dan sarana perangkat lunak.
2. Perangkat lunak yang bekerja lebih dari dokumentasi yang menyeluruh.
3. Kolaborasi dengan klien lebih dari negosiasi kontrak.
4. Tanggap terhadap perubahan lebih dari mengikuti rencana.

Pendukung *methodology* tradisional mengklaim bahwa dokumentasi perangkat lunak yang baik merupakan hal terpenting dalam siklus hidup dari *methodology* tradisional, persyaratan dan desain, kualitas perangkat lunak, dan pemeliharaan perangkat lunak. Jadi tanpa dokumentasi perangkat lunak yang baik, perangkat lunak tidak dapat ditentukan, dirancang, atau dipertahankan, dan pada akhirnya menghasilkan kualitas perangkat lunak

yang rendah. Untuk memperburuk situasi, pencipta *methodology agile* membuat kebalikan dari apa yang biasa atau diterima dari kebijaksanaan konvensional lama dengan mendefinisikan nilai yang mengatakan bahwa "perangkat lunak bekerja lebih dari dokumentasi yang komprehensif."

Kebanyakan pihak memiliki persepsi bahwa dalam *methodology agile*, dokumentasi tidak terlalu dipentingkan sehingga banyak organisasi pengembang IT yang tidak ingin untuk beralih ke *methodology agile*. Sebab telah diamati melalui literatur bahwa dokumentasi yang baik memainkan peran yang sangat penting dalam pengembangan perangkat lunak. Ini membantu dalam meningkatkan kecepatan pengembangan dan juga membantu dalam mempertahankan komunikasi dan hubungan di antara para *developer* dan *stakeholder* lainnya [4].Namun persepsi yang mengatakan bahwa dalam *methodology agile* tidak terlalu mementingkan dokumentasi belum tentu benar sebab *agile* manifesto sifatnya lebih ke preferensi, bukan keharusan atau mandat. Sedangkan dalam *scrum* sendiri, penggunaan kerangka kerjanya sangat beragam [5]. Jadi selama dokumentasi itu memiliki nilai, maka dokumentasi tersebut harus dibuat. Jadi sering terjadi kesalahpahaman dari pemahaman terhadap *methodology agile*.

5. PERBANDINGAN TIGA SDLC, *METHODOLOGY* PARALLEL, ITERATIVE DAN *AGILE* DEVELOPMENT

Dalam pengembangan sistem dan rekayasa perangkat lunak, setiap *methodology* memiliki karakteristik masing-masing. Serta memiliki kelebihan dan kelemahan, jadi kita dapat memilih sesuai dengan kondisi rancangan yang akan dibangun.

Tabel 1. Perbandingan *methodology parallel, iterative* dan *agile development*

Methodology	**Karakteristik**	**Kelebihan**	**Kelemahan**
Parallel	Desain secara umum akan ditampilkan terlebih dahulu kemudian proyek dibagi-bagi menjadi beberapa sub proyek yang dikerjakan secara bersama-sama secara parallel.	Efisiensi waktu yang dibutuhkan untuk mengerjakan proyek menjadi lebih cepat	Akan mengalami masalah pada lamanya pekerjaan jika ada subproyek yang bermasalah; tidak cocok untuk proyek-proyek perangkat lunak yang kompleks.
Iterative	Membangun sebuah model yang dimaksudkan untuk diperpanjang di iterasi yang berurutan; menekankan desain lebih dokumentasi.	Adanya feedback yang terus menerus; kode kerja disampaikan awal perancangan.	Setiap iterasi baku menyerupai struktur mini *waterfall*.
Agile development	Pairing program; Unit testing; lebih cepat; owner dapat memutuskan prioritas tugas.	Aplikasi akan sangat cepat dalam lingkungan produksi; berkurangnya jumlah bug; integrasi kode mulus; feedback terus menerus dari owner; cepat adaptasi untuk perubahan	Kurangnya dokumentasi; kurangnya komitmen untuk mendefinisikan proses produksi dengan baik

6. KESIMPULAN

Dari ketiga *methodology* tersebut, *parallel dan iterative* termasuk tradisional *methodology* sedangkan *agile development* termasuk *methodology* baru era 90-an. *Parallel development* memungkinkan beberapa fase dilakukan bersama-sama untuk mempersingkat waktu. Pada i*terative methodology* terdapat pengulangan proses sehingga adanya *feedback* terus menerus digunakan untuk perbaikan sistem. *Methodology* yang berkembang telah beralih ke *methodology* yang lebih sederhana dengan mengutamakan fleksibilitas dan dengan lingkungan yang cepat berubah-ubah yaitu *Agile development*. *Methodology agile* yang sering digunakan adalah *methodology extreme programming* dan *scrum*. *Extreme programming* didasarkan pada *planning game* dan panduan dalam pengkodean program sedangkan *scrum* telah dikembangkan untuk mengelola proses pengembangan perangkat lunak di lingkungan yang berubah-ubah berdasarkan fleksibiltas, kemampuan beradaptasi, dan produktivitas.

DAFTAR PUSTAKA

Alshamrani, Adel & Bahattab, Abdullah. (2015). A Comparison Between Three SDLC Models *Waterfall* Model, Spiral Model, and Incremental/Iterative Model. *International Journal of Computer Science Issues.* 106-111.

Ashima, & Aggarwal, H. (2010). Multidimensionality in *Agile* Software Development. *International Journal of Computer Science and Information Security,* 234-237.

Schwaber, K, & Sutherland, J. (2011, Oktober). Panduan *Scrum* : Aturan Dalam Bermain.

Uikey, N., Suman, U., & Ramani, A. (2011). A Documented Approach in *Agile* Software Development. *International Journal of Software Engineering,* 13–20.

Aitken, Ashley & Ilango, Vishnu. (2013). A Comparative Analysis of Traditional Software Engineering and *Agile* Software Development. *Hawaii International Conference on* Sistems *Sciences.* 4751-4760.