View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by IKADO E-Journal (Institut Informatika Indonesia)

Analisis Pemilihan Penerapan Proyek Metodologi Pengembangan Rekayasa Perangkat Lunak

Darmawan Setiya Budi
Magister Teknik Informatika
STIMIK Amikom
darmawan.setiyabudi@gmail.com

Taghfirul Azhima Yoga Siswa
Magister Teknik Informatika
STIMIK Amikom
taghfirul.yoga@yahoo.co.id

Heri Abijono
Magister Teknik Informatika
STIMIK Amikom
ahabijono@gmail.com

Abstrak - Metodologi merupakan kerangka pijakan utama dalam perancangan dan pengembangan perangkat lunak profesional untuk menghasilkan sebuah sistem informasi yang sesuai dengan kebutuhan bisnis sebuah organisasi. Keberhasilan pengembangan perangkat lunak bergantung pada pengelolaan proyek perangkat lunak secara keseluruhan. Tidak ada metodologi yang benar-benar sesuai dengan semua jenis organsasi, sehingga dibutuhkan pendekatan lebih lanjut untuk memilih metodologi mana yang paling sesuai untuk dapat diterapkan pada organisasi tertentu. *Paper* ini menjelaskan dan menganalisa metodologi pengembangan perangkat lunak yang meliputi: *Linear Sequential Model* atau *Waterfall, Parallel Model, Iterative Model, Prototyping Model, RAD (Rapid Application Development) Model, Spiral Model, V-Shaped Model* dan *Agile Development* untuk membuat perbandingan yang menunjukan kelebihan dan kelemahan masing-masing. Hasil paper ini menunjukan pertimbangan pemilihan metodologi yang didasarkan pada faktor-faktor kriteria penilaian yang terdiri dari kejelasan persyaratan pengguna, keakraban dengan teknologi, kompleksitas sistem, sistem keandalan, jadwal waktu singkat dan visibility jadwal hingga mereferensi beberapa pendapat dari jurnal ilmiah.

Kata Kunci : metodologi pengembangan rekayasa perangkat lunak, *Linear Sequential Model, Waterfall, Parallel Model, Iterative Model, Prototyping Model, RAD (Rapid Application Development) Model, Spiral Model, V-Shaped Model, Agile Development*

I. PENDAHULUAN

Keberhasilan pengembangan perangkat lunak bergantung pada pengelolaan proyek perangkat lunak secara keseluruhan. Menetapkan sebuah metodologi memiliki dinamisasi yang tinggi dalam tahap-tahap perancangan model yang menggambarkan tahap-tahap aktivitas dan daur hidup suatu sistem.

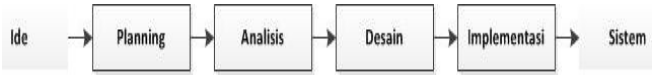
Metodologi merupakan kerangka pijakan utama dalam perancangan dan pengembangan perangkat lunak profesional untuk menghasilkan sebuah sistem informasi yang sesuai dengan kebutuhan bisnis sebuah organisasi. Memilih sebuah metodologi bukanlah hal yang mudah dilakukan karena tidak satupun metodologi yang dapat dikatakan terbaik. Setiap organisasi biasanya memiliki standarisasi tertentu. Sehingga hal ini menjadi alasan paper ini dapat mejawab tuntutan tersebut

Metodologi Pengembangan Perangkat Lunak dapat diartikan sebagai proses membuat suatu perangkat lunak baru untuk menggantikan perangkat lunak lama secara keseluruhan atau memperbaiki perangkat lunak yang telah ada. Metodologi pengembangan perangkat lunak ini sangat diperlukan agar dapat lebih cepat dan tepat dalam mendeskripsikan solusi dan mengembangkan perangkat lunak. Dengan metodologi pengembangan ini nantinya juga dapat membantu untuk menghasilkan perangkat lunak yang berkualitas.

II. METODOLOGI PENGEMBANGAN PERANGKAT LUNAK

Menurut Azhar Susanto menyatakan bahwa SDLC (*System Development Life Cycle*) adalah salah satu metode pengembangan sistem informasi yang popular pada saat sistem informasi pertama kali dibuat [1].

Terdapat empat tahapan dalam membangun atau mengembangkan sistem informasi dengan menggunakan SDLC, yaitu: *planning*, *analysis*, *design*, dan *implementation*. Adapun dalam implementasi SDLC terdapat berbagai metodologi yang dapat dipergunakan. Penggunaan metodologi akan bervariasi tergantung kepada penekanannya, apakah terhadap bisnis proses ataukah pada data pendukung bisnis.



Gambar 1. *System Development Life Cycle*.

Berdasarkan pengertian tersebut, secara umum dapat dikatakan bahwa proses pengembangan perangkat lunak mengikuti tahap-tahap**:**

1) Menentukan APA yang harus dikerjakan oleh perangkat lunak dalam satu rentang waktu tertentu.
2) Mendefinisikan BAGAIMANA perangkat lunak dibuat, mencakup arsitektur perangkat lunaknya, antar muka internal, algoritma, dan lain-lain.
3) Penerapan (penulisan program) dan pengujian unit-unit program.
4) Integrasi dan pengujian modul-modul program.

5) Validasi perangkat lunak secara keseluruhan (pengujian sistem).

A. Komponen dan Karakteristik Proyek Metodologi Pengembangan Perangkat Lunak

Menurut Pressman bahwa komponen metodologi pengembangan perangkat lunak dapat dibagi ke dalam tiga unit, yaitu [2]:

1) **Metode,** yaitu suatu cara atau teknik pendekatan yang sistematik yang dipergunakan untuk mengembangkan perangkat lunak. Metode ini mencakup**:** Perencanaan proyek dan perkiraan, analisis keperluan sistem dan perangkat lunak, perancangan struktur data, arsitektur program, prosedur algoritma, penulisan kode program (*coding*), uji coba, dan pemeliharaan.
2) **Alat Bantu (*Tools*),** yaitu alat-alat (manual maupun otomatis) yang mendukung pengembangan perangkat lunak. Terdapat dua alat bantu yang dapat digunakan yaitu**:** alat bantu manual dan alat bantu otomatis.
3) **Prosedur,** yang dipergunakan untuk mendefinisikan urut-urutan pekerjaan (daur) dari metode dan alat bantu tersebut.

Menurut Despa bahwa karakteristik proyek pengembangan perangkat lunak terdiri dari**:** sering berubah spesifikasi, dinamika tinggi teknologi dan standard, tenaga kerja terampil, dan tim didistribusikan secara global [3]. Hal ini dapat dilihat pada tabel 1.

Tabel 1. Karakteristik Proyek Pengembangan Perangkat Lunak.

Karakteristik	Dampak Positif	Dampak Negatif
Sering berubah spesifikasi		Membahayakan tenggat waktu
		Hasil melebihi anggaran proyek
		Menyebabkan stres dan ketidakpuasan bagi tim pengembangan
Dinamika tinggi teknologi dan standar	Menghasilkan peluang baru dalam dari segi desain dan *codding*	Perangkat lunak dapat menjadi usang pada saat marak di pasaran
		Pengembang software harus menginvestasikan banyak waktu dalam meneliti teknologi baru
Tenaga kerja terampil	Meningkatkan kemungkinan mencapai hasil yang inovatif	Biaya tinggi yang dihasilkan oleh sumber daya manusia
Tim didistribusikan secara global	Bekerja dapat dilakukan sekitar jam	Monitoring dan kontrol menjadi lebih sulit
	Keragaman budaya memelihara kreativitas	Mengintegrasikan kode baru yang lebih menantang

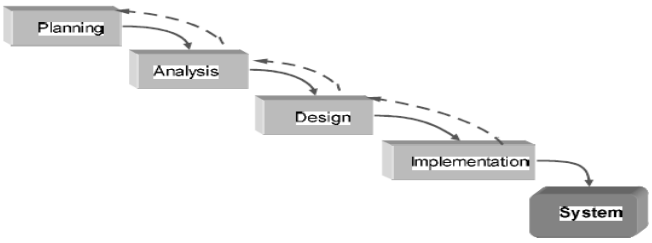
B. Macam-Macam Metodologi Pengembangan Perangkat Lunak

Macam-macam klasifikasi metodologi pengembangan perangkat lunak terdiri dari dua pendapat antara lain:

Pertama, menurut Ian Sommerville bahwa model proses pengembangan perangkat lunak terbagi menjadi empat, yaitu**:** Model Pengembangan *Prototyping* (*Evolusioner*), Model Pengembangan Sistem Formal, Model Pengembangan Berorientasi Pemakaian Ulang (*Reuse-oriented software engineering*), dan Model Pengembangan *Waterfall* [4].

Kedua, menurut Pressman yang juga mejadi fokus pembahasan dalam paper ini – menyebutkan bahwa model proses pengembangan perangkat lunak terbagi menjadi 5 metode yaitu**:** *Linear Sequential Model* atau *waterfall*, *Incremental Process Model*, *Evolutionary Process Model*, *RAD (Rapid Application Development) Model*, dan *Concurrent Model* [2].

1) *Linear Sequential Model*



Gambar 2. *Linear Sequential Model*.

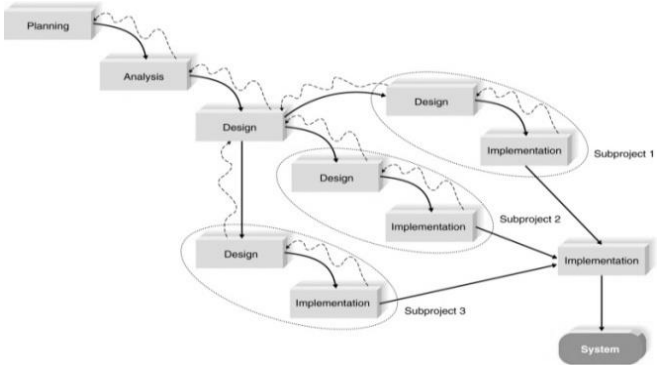
Linear sequential model ("*classic life cycle*" atau "*waterfall model*") adalah metode pengembangan perangkat lunak dengan pendekatan sekuensial dengan cakupan aktivitas**:**

a) **Rekayasa Sistem dan Analisis (*System Engineering and Analysis*).** Karena perangkat lunak adalah bagian dari sistem yang lebih besar, pekerjaan dimulai dari pembentukan kebutuhan-kebutuhan untuk seluruh elemen sistem dan kemudian memilah mana yang untuk pengembangan perangkat lunak. Hal ini penting, ketika perangkat lunak harus berkomunikasi dengan hardware, orang, dan basis data.
b) **Analisis Kebutuhan Perangkat Lunak (*Software Requirements Analysis*).** Pengumpulan kebutuhan dengan fokus pada perangkat lunak, yang meliputi**:** domain informasi, fungsi yang dibutuhkan, unjuk kerja/performansi dan antarmuka. Hasilnya harus didokumentasi dan di-*review* ke pelanggan.
c) **Perancangan (*Design*).** Ada empat atribut untuk program, yaitu**:** Struktur Data, Arsitektur perangkat lunak, Prosedur detil, dan Karakteristik Antarmuka. Proses desain mengubah kebutuhan-kebutuhan menjadi bentuk karakteristik yang dimengerti perangkat lunak sebelum dimulai penulisan program. Desain ini harus terdokumentasi dengan baik dan menjadi bagian konfigurasi perangkat lunak.

d) **Pembuatan Kode (*Coding*).** Penterjemahan perancangan ke bentuk yang dapat dimengerti oleh mesin, dengan menggunakan bahasa pemrograman.
e) **Pengujian (*Testing*).** Setelah kode program selesai testing dapat dilakukan. Testing memfokuskan pada logika internal dari perangkat lunak, fungsi eksternal dan mencari segala kemungkinan kesalahan dan memriksa apakah sesuai dengan hasil yang diinginkan.
f) **Pemeliharaan (*Maintenance*).** Merupakan bagian paling akhir dari siklus pengembangan dan dilakukan setelah perangkat lunak dipergunakan, meliputi kegiatan-kegiatan**:**
   i) ***Corrective Maintenance*:** Mengoreksi kesalahan pada perangkat lunak, yang baru terdeteksi pada saat perangkat lunak dipergunakan.
   ii) ***Adaptive Maintenance*:** Penyesuaian dengan lingkungan baru, misalnya sistem operasi atau sebagai tuntutan atas perkembangan sistem komputer, misalnya penambahan printer driver.
   iii) ***Perfektive Maintenance*:** Bila perangkat lunak sukses dipergunakan oleh pemakai. Pemeliharaan ditujukan untuk menambah kemampuannya seperti memberikan fungsi-fungsi tambahan, peningkatan kinerja dan sebagainya.

2) *Parallel Model*

Menurut Dennis, Parallel Model merupakan metodologi yang mencoba untuk mengatasi interval waktu yang lama antara tahap analisis dan pengiriman sistem [5]. Metodologi ini mencoba untuk memperbaiki kelemahan dari metodologi *waterfall*, melakukan desain umum dan implementasi secara berurutan untuk seluruh sistem dan kemudian proyek ini dibagi menjadi serangkaian subproyek yang berbeda yang dapat dirancang dan dilaksanakan secara paralel. Setelah semua subproyek sempurna, maka dilakukan integrasi akhir sehingga dilakukan *delivery* pada sistem.

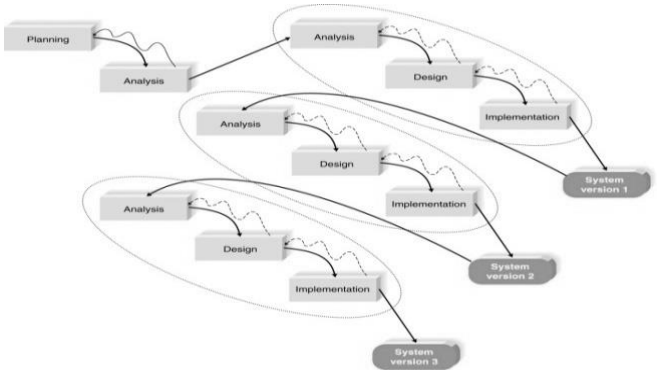


Gambar 3. Metodologi *Parallel*.

3) *Iterative Model*

Metologi ini berkembang didasari oleh masalah pada model *waterfall* yang menciptakan permintaan untuk metode baru dari sistem yang berkembang agar dapat memberikan hasil yang lebih cepat, membutuhkan lebih sedikit informasi yang mutakhir, dan menawarkan fleksibilitas yang lebih besar.

Menurut Larman, *Iterative Model* merupakan metodologi yang mengandalkan pembangunan aplikasi perangkat lunak satu langkah pada satu waktu dalam bentuk memperluas model [6]. Metodologi ini didasarkan pada spesifikasi awal model dasar dari aplikasi yang dibangun. Setelah model diuji dan umpan balik diterima dari spesifikasi proyek, maka selanjutnya disesuaikan dengan model yang akan dikembangkan. Proses ini diulang sampai model menjadi aplikasi yang berfungsi penuh untuk memenuhi semua kebutuhan pemilik proyek.

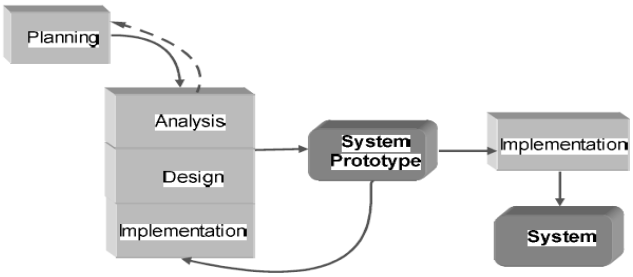


Gambar 4. Metodologi *Iterative*.

Model ini diimplementasi dengan cara perulangan, sehingga proyek pada model ini dibagi menjadi bagian-bagian kecil. Hal ini memungkinkan tim pengembangan untuk menunjukkan hasil sebelumnya dapat di proses dan mendapatkan umpan balik yang berharga dari pengguna sistem.

Seringkali, setiap perulangan sebenarnya adalah sebuah proses mini-*Waterfall* dengan umpan balik dari satu fase yang menyediakan informasi penting untuk desain tahap berikutnya. Dalam variasi model ini, produk-produk perangkat lunak, yang diproduksi pada akhir setiap langkah (atau serangkaian langkah-langkah), dapat masuk ke produksi langsung sebagai temuan tambahan.

4) *Prototyping Model*



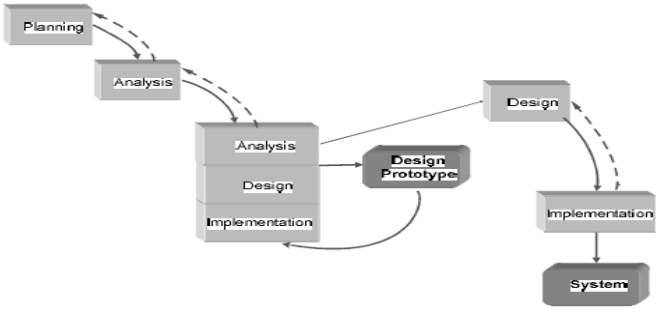
Gambar 5. *Prototyping Model*.

Pendekatan *prototyping* model digunakan jika pemakai hanya mendefenisikan objektif umum dari perangkat lunak tanpa memerinci kebutuhan input, pemrosesan dan outputnya, sementara pengembang tidak begitu yakin akan efesiensi algoritma, adaptasi sistem operasi, atau bentuk

antarmuka manusia-mesin yang harus diambil. Cakupan aktivitas dari *prototyping* model terdiri dari**:**

a) Mendefinisikan objektif secara keseluruhan dan mengidentifikasi kebutuhan yang sudah diketahui.
b) Melakukan perancangan secara cepat sebagai dasar untuk membuat *prototype*.
c) Menguji coba dan mengevaluasi *prototype* dan kemudian melakukan penambahan dan perbaikan-perbaikan terhadap *prototype* yang sudah dibuat.

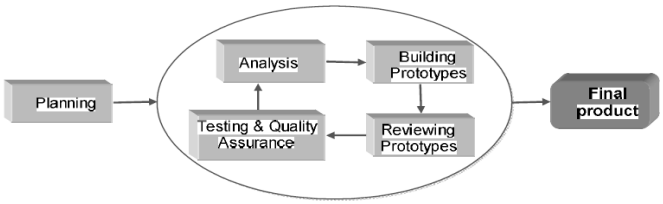
5) *Threwaway Prototyping Model*



Gambar 6. *Threwaway Prototyping Model*.

Metodologi ini mirip dengan metodologi berdasarkan *prototyping*. Perbedaan utama adalah bahwa lembaran prototipe selesai selama titik yang berbeda dalam SDLC. Fokus pembangunan adalah untuk menguji fitur yang tidak dipahami dengan menganalisis, merancang, dan membangun prototipe desain. Prototipe desain merupakan bagian dari sistem yang perlu perbaikan tambahan, dan itu hanya cukup rinci untuk memungkinkan pengguna untuk memahami isu-isu yang sedang dipertimbangkan. Setelah masalah diselesaikan, proyek bergerak ke dalam desain dan implementasi. Pada titik ini, desain prototipe dibuang, yang merupakan perbedaan penting antara *Threwaway Prototyping* dan *Prototyping*, di mana prototipe berkembang menjadi sistem final. Pendekatan ini menghasilkan lebih stabil dan dapat diandalkan sistem.

6) *RAD (Rapid Application Development) Model*

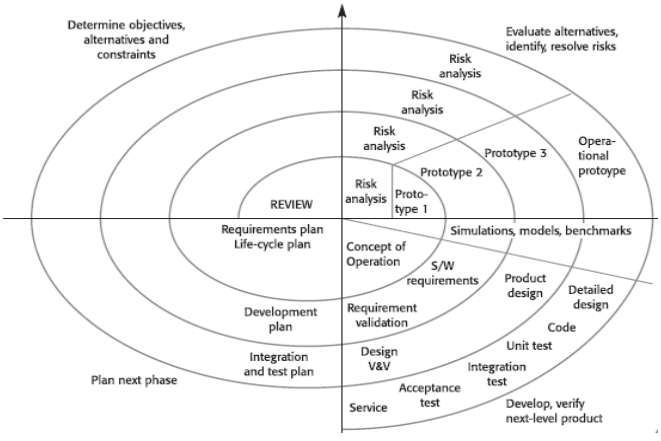


Gambar 7. *RAD (Rapid Application Development) Model*.

Merupakan model proses pengembangan perangkat lunak secara *linear sequential* yang menekankan pada siklus pengembangan yang sangat singkat. Jika kebutuhan dipahami dengan baik, proses RAD memungkinkan tim pengembangan menciptakan "sistem fungsional yang utuh" dalam periode waktu yang sangat pendek (kira-kira 60-90 hari). Cakupan aktivitas dari RAD model ini terdiri dari**:**

a) **Pemodelan Bisnis (*Bussiness Modelling*).** Aliran informasi diantara fungsi-fungsi bisnis dimodelkan dengan suatu cara untuk menjawab pertanyaan-pertanyaan berikut**:** Informasi apa yang mengendalikan proses bisnis? Ke mana informasi itu pergi? Siapa yang memprosesnya?
b) **Pemodelan Data (*Data Modelling*).** Aliran informasi yang didefinisikan sebagai bagian dari fase pemodelan bisnis disaring ke dalam serangkaian objek data yang dibutuhkan untuk menopang bisnis tersebut. Karakteristik/atribut dari masing-masing objek diidentifikasi dan hubungan antara objek-objek tersebut didefinisikan.
c) **Pemodelan Proses (*Process Modelling*).** Aliran informasi yang didefinisikan dalam fase pemodelan data ditransformasikan untuk mencapai aliran informasi yang perlu bagi implementasi sebuah fungsi bisnis. Gambaran pemrosesan diciptakan untuk menambah, memodifikasi, menghapus atau mendapatkan kembali sebuah objek data.
d) **Pembuatan Aplikasi (*Application Generation*).** Selain menciptakan perangkat lunak dengan menggunakan bahasa pemrograman generasi ketiga yang konvensional, RAD lebih banyak memproses kerja untuk memakai lagi komponen program yang telah ada atau menciptakan komponen yang bias dipakai lagi. Pada semua kasus, alat-alat bantu otomatis dipakai untuk memfasilitasi kontruksi perangkat lunak.
e) **Pengujian dan Pergantian (*Testing and Turnover*).** Karena proses RAD menekankan pada pemakaian kembali, banyak komponen yang telah diuji. Hal ini mengurangi keseluruhan waktu pengujian. Tapi komponen baru harus diuji.

7) *Spiral Model*



Gambar 8. *Spiral Model.*

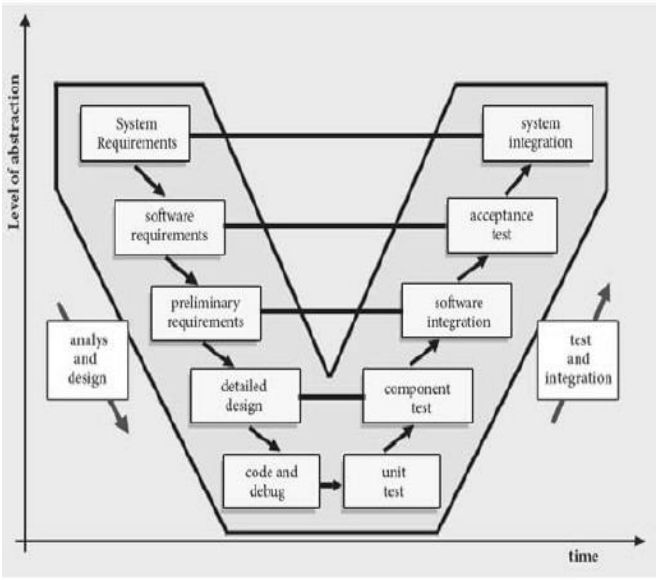
Merupakan model proses perangkat lunak yang memadukan wujud pengulangan dari model *prototyping* dengan aspek pengendalian dan sistematika dari linear sequential model, dengan penambahan elemen baru yaitu analisis resiko.

Model ini memiliki empat aktivitas penting, yaitu**:**

a) **Perencanaan (*Planning*).** penentuan tujuan, alternatif, dan batasan.
b) **Analisis resiko (*Risk Analysis*).** analisis alternatif dan identifikasi/pemecahan resiko.
c) **Rekayasa (*Engineering*).** pengembangan level berikutnya dari produk.
d) **Evaluasi Pemakai (*Customer Evaluation*).** penilaian terhadap hasil rekayasa Bentuk spiral memberikan gambaran bahwa semakin besar iterasinya, maka menunjukkan makin lengkap versi dari perangkat lunak yang dibuat. Selama awal sirkuit, objektif, alternatif dan batasan didefinisikan serta resiko diidentifikasikan dan dianalisa.

Jika resiko menunjukkan ada ketidakpastian terhadap kebutuhan, maka *prototyping* harus dibuat pada kuadran rekayasa. Simulasi dan pemodelan lain dapat digunakan untuk mendefinisikan masalah dan memperbaiki kebutuhan. Pelanggan mengevaluasi hasil rekayasa (kuadran evaluasi pelanggan) dan membuat usulan untuk perbaikan. Berdasarkan masukan dari pelanggan, fase berikutnya adalah perencanaan dan analisis resiko. Setelah analisis resiko selalu diperiksa apakah proyek diteruskan atau tidak, jika resiko terlalu besar, maka proyek dapat dihentikan. Model spiral ini adalah pendekatan yang paling realistic untuk sistem sekala besar.

8) *V-Shaped Model*



Gambar 9. *V-Shaped Model*.

Sama seperti model air terjun, V- yang siklus hidup berbentuk jalan berurutan dari pelaksanaan proses. Setiap fase harus diselesaikan sebelum tahap berikutnya dimulai. Pengujian ditekankan dalam model ini lebih dari model air terjun. Prosedur pengujian yang dikembangkan di awal

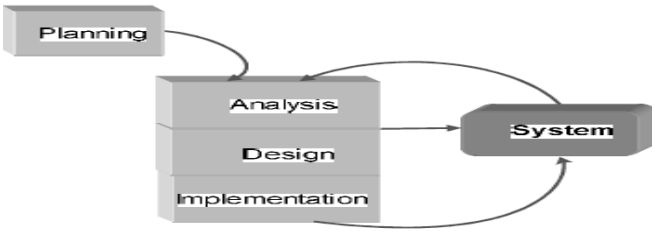
siklus hidup sebelum coding dilakukan, masing-masing selama fase sebelumnya implementasi. Persyaratan mulai model siklus hidup seperti model air terjun. Sebelum pembangunan dimulai, rencana uji sistem dibuat. Rencana uji sistem berfokus pada pemenuhan fungsi yang ditetapkan dalam persyaratan pengumpulan.

Tahap desain tingkat tinggi berfokus pada arsitektur sistem dan desain. Sebuah rencana uji integrasi dibuat dalam fase ini dalam rangka untuk menguji potongan kemampuan sistem perangkat lunak untuk bekerja sama. Namun, tahap desain tingkat rendah terletak di mana komponen perangkat lunak yang sebenarnya dirancang, dan tes unit yang dibuat dalam fase ini juga.

9) *Agile Development l*

Kategori ini berfokus pada perampingan SDLC dengan menghilangkan banyak pemodelan dan dokumentasi overhead dan waktu yang dihabiskan untuk tugas-tugas. Proyek menekankan sederhana, pengembangan aplikasi berulang. Kategori ini menggunakan pemrograman ekstrim (XP), yang dijelaskan sebagai berikut**:**

Prinsip-prinsip Kunci XP meliputi pengujian terus menerus, coding sederhana dan interaksi yang dekat dengan pengguna akhir untuk membangun sistem yang sangat cepat. Setelah proses perencanaan yang dangkal, tim proyek melakukan analisis, desain, dan fase implementasi *iterative*.



Gambar 10. *Agile Development*.

III. MEMILIH METODOLOGI PENGEMBANGAN YANG TEPAT

Beberapa pertimbangan pemilihan metodologi pengembangan perangkat lunak yang tepat menurut Dennis terdiri dari beberapa keriteria meliputi**:** kejelasan kebutuhan pengguna (*clarity user requirement*), penguasaan teknologi (*familiarity with technology*), tingkat kerumitan sistem (*system complexity*), tingkat kehandalan sistem (*system realibility*), waktu pelaksanaan (*short time schedules*), dan visibilitas jadwal pelaksanaan (*schedule visibility*) [5]. Hal ini dijelaskan pada tabel 2.

Tabel 2. Kriteria Pemilihan Metodologi.

Kriteria Pengembangan Sistem	*Waterfall*	*Parallel*	*V-Model*	*Iterative*	*System Prototyping*	*Threwaway Prototyping*	*Agile Development*
Kejelasan kebutuhan pengguna	Buruk	Buruk	Buruk	Baik	Baik Sekali	Baik Sekali	Baik Sekali
Penguasaan teknologi	Buruk	Buruk	Buruk	Baik	Buruk	Baik Sekali	Buruk
Tingkat kerumitan sistem	Baik	Baik	Baik	Baik	Buruk	Baik Sekali	Buruk
Tingkat kehandalan sistem	Baik	Baik	Baik Sekali	Baik	Buruk	Baik Sekali	Baik
Waktu pelaksanaan	Buruk	Baik	Buruk	Baik Sekali	Baik Sekali	Baik	Baik Sekali
Visibilitas jadwal pelaksanaan	Buruk	Buruk	Buruk	Baik Sekali	Baik Sekali	Baik	Baik

Tabel 3. Kelebihan dan Kelemahan Metodologi Pengembangan Perangkat Lunak.

No	Metodologi	Kelebihan	Kelemahan
A	*Linear Sequential Model*	1) Mudah dalam pengelolaan karena hampir seluruh requirements telah diidentifikasikan dan didokumentasikan, 2) Tahapan yang berurutan secara linier, identifikasi dan dokumentasi yang lengkap, menyebabkan proses mudah dipahami oleh seluruh tim yang terlibat ataupun project owner.	1) Tahapan yang berurutan secara linier tidak memungkinkan untuk kembali pada tahapan selanjutnya, 2) Tidak fleksibel terhadap perubahan kebutuhan yang terjadi dalam tahap pengembangan sistem, 3) Hampir tidak ada toleransi kesalahan, terutama pada tahapan planning dan design.
B	*Parallel Model*	Waktu pengembangan sistem yang lebih singkat dibandingkan *waterfall*, karena beberapa tahapan diakselerasikan dengan membagi menjadi beberapa *sub project.*	1) Integrasi sistem memiliki kesulitan tersendiri. Kegagalan atau keterlambatan pada salah satu sub project memberikan dampak pada proses mengintegrasikan seluruh sistem, 2) Terdapat kemungkinan kesulitan dalam penanganan jika terjadi permasalah pada sub project secara bersamaan.
C	*Iterative Model*	1) Umpan balik terus menerus dari pemilik proyek, 2) Beberapa revisi pada seluruh aplikasi dan fungsi spesifik, 3) Pekerjaan disampaikan di awal proyek.	Setiap perulangan adalah struktur kaku yang menyerupai project kecil *waterfall.*
D	*Prototyping Model*	1) *Requirements identification* yang akurat karena dilakukan evaluasi secara berkala dan mendapatkan masukan dari *project owner* terhadap purwa rupa yang dihasilkan, 2) *User experience* yang meningkat, karena secara terus menerus melakukan uji coba dan evaluasi terhadap purwa rupa, 3) Kesalahan dan redudansi dapat diminimalkan karena proses identifikasi yang baik terhadap purwa rupa.	1) Setiap evaluasi dan masukan terhadap purwa rupa, maka akan membutuhkan penyesuaian terhadap purwa rupa tersebut. Dan setiap penyesuaian akan meningkatkan kompleksitas sistem yang dikembangkan, 2) Memberikan beban tambahan kepada programmer, 3) Terdapat kebutuhan biaya tambahan terkait dengan pembuatan purwa rupadan dapat dilakukan penyesuaian versi purwa rupa sesuai kebutuhan, hingga purwa rupa dapat disetujui oleh project owner.
E	*RAD (Rapid Application Development) Model*	1) Efisiensi waktu pengiriman, 2) Perubahan kebutuhan dapat ditampung, 3) Waktu siklus dapat pendek dengan penggunaan alat-alat RAD yang kuat, 4) Produktivitas dengan lebih sedikit orang dalam waktu singkat, 5) Penggunaan alat-alat dan kerangka kerja.	1) Kompleksitas manajemen, 2) Cocok untuk sistem yang berbasis komponen dan terukur, 3) Membutuhkan keterlibatan pengguna di seluruh siklus hidup, 4) Membutuhkan personal yang sangat terampil, 5) Ketergantungan tinggi pada kemampuan modeling, 6) Tidak berlaku untuk proyek-proyek yang lebih murah sebagai biaya pemodelan dan otomatis generasi kode sangat tinggi untuk proyek-proyek yang dianggarkan lebih murah untuk pantas.
F	*Threwaway Prototyping Model*		
G	*Spiral Model*	1) Jumlah analisis risiko yang tinggi, 2) Baik untuk proyek-proyek besar dan *mission-critical*, 3) Software diproduksi di awal siklus hidup perangkat lunak.	1) Dapat menjadi model mahal untuk digunakan, 2) Analisis risiko membutuhkan keahlian yang sangat spesifik, 3) Keberhasilan proyek sangat tergantung pada tahap analisis risiko, 4) Tidak bekerja dengan baik untuk proyek-proyek yang lebih kecil.
H	*V-Shaped Model*	1) Sederhana dan mudah digunakan, 2) Setiap fase memiliki *delivery* tertentu, 3) Kesempatan keberhasilan yang lebih tinggi atas model *waterfall* karena perkembangan awal dari rencana pengujian selama siklus hidup, 4) Bekerja dengan baik untuk proyek-proyek kecil di mana persyaratan yang mudah dipahami.	1) Sangat kaku seperti model *waterfall*, 2) Sedikit fleksibilitas dan ruang lingkup menyesuaikan sulit dan mahal, 3) Software dikembangkan selama tahap implementasi, sehingga tidak ada prototipe awal dari perangkat lunak yang dihasilkan, 4) Model ini tidak memberikan jalan yang jelas untuk masalah yang ditemukan selama pengujian tahap.
I	*Agile Development*	1) Metode ringan sesuai proyek ukuran kecil-menengah, 2) Menghasilkan kohesi tim yang baik, 3) Menekankan produk akhir, 4) Berulang, 5) Pendekatan berbasis tes untuk persyaratan dan jaminan kualitas.	1) Tidak cocok untuk menangani dependensi yang kompleks, 2). Lebih risiko keberlanjutan, rawatan dan diperpanjang, 3) Sebuah rencana keseluruhan, pemimpin lincah danmanajemen proyek tangkas praktekadalah suatu keharusan tanpa yang tidak akan bekerja.

IV. PEMBAHASAN

Metodologi pengembangan perangkat lunak yang terdiri dari *Linear Sequential Model atau waterfall, Parallel Model, Iterative Model, Prototyping Model, RAD (Rapid Application Development) Model, Spiral Model, V-Shaped Model* dan *Agile Development* memiliki perbandingan yang menunjukkan fitur kelebihan dan kelemahan masing-masing. Pertimbangan pemilihan metodologi yang tepat sesuai dengan kebutuhan dapat didasarkan pada kriteria penilaian yang terdiri dari kejelasan persyaratan pengguna, keakraban dengan teknologi, sistem kompleksitas, sistem keandalan, jadwal waktu singkat dan visibility jadwal. Tidak ada metodologi yang benar-benar sesuai dengan semua jenis organsasi, sehingga diperlukan pendekatan lebih lanjut untuk memilih metodologi mana yang paling sesuai untuk dapat diterapkan pada organisasi tertentu. Beberapa pendapat tentang pemilihan metodologi pengembangan sistem dari beberapa hasil literatur jurnal ilmiah antara lain**:**

A. Menurut Munassar, terdapat banyak model yang digunakan untuk mengembangkan sistem dengan ukuran yang berbeda dilihat dari project dan kebutuhannya [7]. Umumnya menggunakan model *waterfall* dan *spiral* yang dibangun pada tahun 1970 dan tahun 1999. Setiap model memiliki kelebihan dan kekurangan, sehingga masing-masing model mencoba untuk menghilangkan kekurangan dari model sebelumnya.
B. Menurut Ajah, bahwa pilihan metodologi dipengaruhi oleh kejelasan kebutuhan pengguna (*clarity user requirement*), penguasaan teknologi (*familiarity with technology*), tingkat kerumitan sistem (*system complexity*), tingkat kehandalan sistem (*system realibility*), waktu pelaksanaan (*short time schedules*), dan visibilitas jadwal pelaksanaan (*schedule visibility*) [8]. Untuk menjadi sukses dalam proyek perangkat lunak, pemegang saham harus kritis terhadap metode yang berbeda, sehingga secara efektif dapat menggabungkan metode yang akan membantu mencapai tujuan perangkat lunak.
C. Menurut Taya, semua model pengembangan perangkat lunak memiliki kelebihan dan kekurangan [9]. Namun pengaturan dan pemilihan waktu sangat penting dalam pengembangan perangkat lunak. Jika penundaan terjadi dalam tahap pengembangan, pasar dapat diambil alih oleh pesaing. Jika produk yang dilincurkan lebih cepat dari pada pesaing ternyata berisi "*bug*", hal ini dapat mempengaruhi reputasi perusahaan. Sehingga, diperlukan komitmen antara waktu pengembangan dan kualitas produk. Pelanggan tidak mengharapkan produk gratis yang berisi "*bug*" tetapi produk yang *user-friendly* yang menghasilkan gairah atau kegembiraan.
D. Menurut Despa, metodologi pengembangan software mengikuti dua filosofi utama: kelas berat dan kelas ringan [3]. Metodologi kelas berat cocok untuk proyek-proyek di mana kebutuhan tidak mungkin diubah dan kompleksitas software digunakan untuk perencanaan secara rinci. Dengan metodologi kelas berat manajer proyek dapat dengan mudah melakukan pelacakan, evaluasi dan pelaporan. Pemilik proyek jauh terlibat hanya dalam tahap penelitian dan perencanaan. Metodologi kelas ringan cocok untuk proyek-proyek dengan spesifikasi tidak jelas atau mungkin berubah karena faktor internal atau eksternal. Metodologi kelas ringan didasarkan pada pendekatan bertahap, software disampaikan dalam beberapa pengulangan berturut-turut dan semua menjadi versi fungsional dari aplikasi. Metodologi kelas ringan memberikan fleksibilitas yang besar dan dapat dengan mudah beradaptasi dengan perubahan.

VI. KESIMPULAN

Dapat disimpulkan bahwa keberhasilan pengembangan perangkat lunak bergantung pada pengelolaan proyek perangkat lunak secara keseluruhan. Komponen metodologi pengembangan perangkat lunak terdiri dari metode, alat bantu (*Tools*), dan prosedur. Tidak ada metodologi yang benar-benar sesuai dengan semua jenis organsasi, sehingga dibutuhkan pendekatan lebih lanjut untuk memilih metodologi mana yang paling sesuai untuk dapat diterapkan pada organisasi tertentu. Metodologi pengembangan perangkat lunak yang terdiri dari *Linear Sequential Model* atau *waterfall, Parallel Model, Iterative Model, Prototyping Model, RAD (Rapid Application Development) Model, Spiral Model,* V-*Shaped Model* dan *Agile Development* memiliki perbandingan yang menunjukkan fitur kelebihan dan kelemahan masing-masing. Pertimbangan pemilihan metodologi yang tepat sesuai dengan kebutuhan dapat didasarkan pada kriteria penilaian yang terdiri dari kejelasan persyaratan pengguna, keakraban dengan teknologi, sistem kompleksitas, sistem keandalan, jadwal waktu singkat dan *visibility* jadwal hingga mereferensi beberapa pendapat dari penelitian atau jurnal ilmiah. Disarankan untuk menganalisis metodologi yang lain dengan pendekatan yang berbeda untuk mensimulasi dan membandingkan karakteristik dalam rangka mewujudkan keberhasilan untuk memilih sebuah metodologi yang akan diimplementasikan dalam sebuah organisasi.

REFERENSI

[1] Susanto, A. (2004). *Sistem Informasi Manajemen*. Bandung : Lingga Jaya.
[2] Pressman, R. S. (2005). *Software Engineering: a Practitioner's Approach*. Seventh Edition.
[3] Despa, M. L. (2014). *Comparative Study on Software Development Methodologies*. Database Systems Journal vol. V, no. 3.
[4] Ian, S. (2004). *Software Engineering 7th Edition*, Addison Wesley.

[5] Dennis. A, Wixom. B, and Roth. R. (2006). *System Analysis and Design.* John Wiley and Sons, Inc pp. 171-209.
[6] Larman. C, Basili. V. R, (2003). *Iterative and Incremental Development: A Brief History*. Computer, vol. 36, no. 6, pg. 47-56, doi:10.1109/ MC.2003.1204375.
[7] Munassar, N. M. A. and Govardhan, A. (2010). *Comparison Between Five Models Of Software Engineering*. IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 5, September.
[8] Ajah, I. A. and Ugah, J. O. (2013). *Comparative Analysis of Software Development Methodologies*. Volume 3, Issue 6, June. www.ijarcsse.com.
[9] Taya, S. and Gupta, S. (2011). *A Comparison Between Five Models Of Software Engineering*. IJCST Vol. 2, Issue 4, Oct.–Dec.