

ANALISA PENGGUNAAN METODOLOGI PENGEMBANGAN PERANGKAT LUNAK

Maikel Bolung¹⁾, Henry Ronald Karunia Tampangela²⁾

^{1,2)} Teknik Informatika Universitas Amikom Yogyakarta
e-mail: bolungmaikel@gmail.com¹⁾, henry.tampangela@gmail.com²⁾

ABSTRACT

The methodology is a formal approach or sequence of actions to implement System development life cycle (SDLC) which is a process of understanding how an information system can support business needs, design a system, build it and present it to an organization. The methodology is also the main foothold in the design and development of software to produce an information system that suits the business needs of the organization. This paper describes the analysis of software development methodology selection, such as: Linear Sequential Model or Waterfall, Parallel Model, Iterative Model, Prototyping Model, RAD (Rapid Application Development) Model, Spiral Model, V-Shaped Model and Agile Development. The results of this paper can provide consideration for choosing and using appropriate methodologies based on needs, strengths and weaknesses, as well as other assessment factors such as familiarity with technology, system complexity, system reliability, short and precise time, also referencing multiple journals scientific.

Keywords: Software development methodology, Linear Sequential Model or waterfall, Parallel Model, Iterative Model, Prototype Model, RAD (Rapid Application Development) Model, Spiral Model, V-Shaped Model, Agile Development.

ABSTRAK

Metodologi adalah sebuah pendekatan formal atau rangkaian dari tindakan untuk mengimplementasikan System development life cycle (SDLC) yang merupakan suatu proses pemahaman tentang bagaimana sebuah sistem informasi dapat mendukung kebutuhan bisnis, mendisain sistem, membangun dan menyajikannya kepada sebuah organisasi. Metodologi juga merupakan kerangka pijakan utama dalam perancangan dan pengembangan perangkat lunak untuk menghasilkan sebuah sistem informasi yang sesuai dengan kebutuhan bisnis suatu organisasi. Paper ini menjelaskan analisa pemilihan metodologi pengembangan perangkat lunak yaitu: Linear Sequential Model atau waterfall, Parallel Model, Iterative Model, Prototyping Model, RAD (Rapid Application Development) Model, Spiral Model, V-Shaped Model dan Agile Development. Hasil dari paper ini dapat memberikan pertimbangan untuk melakukan pemilihan dan penggunaan metodologi yang tepat berdasarkan kebutuhan, kelebihan dan kelemahan, juga faktor-faktor penilaian yang lain seperti keakraban dengan teknologi, kompleksitas sistem, keandalan sistem, waktu yang singkat dan tepat, hingga mereferensi beberapa jurnal ilmiah.

Kata Kunci: Metodologi pengembangan rekayasa perangkat lunak, Linear Sequential Model atau waterfall, Parallel Model, Iterative Model, Prototyping Model, RAD (Rapid Application Development) Model, Spiral Model, V-Shaped Model, Agile Development.

I. PENDAHULUAN

Kesuksesan pengembangan suatu perangkat lunak bergantung pada pengelolaan proyek perangkat lunak secara keseluruhan. Metodologi menyusun sebuah pendekatan untuk melaksanakan *System Development Life Cycle (SDLC)*. Metodologi ialah bagian utama dalam perencanaan dan pengembangan perangkat lunak yang bertujuan untuk menghasilkan sistem informasi sesuai kebutuhan bisnis suatu organisasi.

Memilih dan menentukan suatu metodologi dalam merancang dan membangun perangkat lunak sistem informasi bukanlah hal yang mudah dilakukan seperti yang dibayangkan karena semua metodologi memiliki kelebihan dan kekurangan. Setiap organisasi biasanya memiliki standarisasi tertentu, sehingga hal inilah yang menjadi alasan paper ini, yaitu dapat menjawab kebutuhan tersebut.

Metodologi pengembangan perangkat lunak dapat diartikan sebagai suatu proses membuat perangkat lunak baru atau hanya memperbaiki perangkat lunak yang sudah ada Mengembangkan perangkat lunak

dengan menggunakan metodologi SDLC sangat membantu untuk menganalisa, mempercepat dan menghasilkan ketepatan dalam menggambarkan sebuah solusi yang diperlukan untuk menjadikan perangkat lunak yang berkualitas.

II. METODOLOGI PENGEMBANGAN SISTEM PERANGKAT LUNAK

Menurut Azhar Susanto^[11] menyatakan bahwa SDLC (*System Development Life Cycle*) adalah salah satu metode pengembangan sistem informasi yang populer pada saat sistem informasi pertama kali dibuat.

Terdapat empat fase dalam SDLC untuk mengembangkan system informasi, yaitu : *planning*, *analysis*, *design*, dan *implementation*. Setiap fase terdiri dari sebuah rangkaian langkah atau tindakan yang mengandalkan teknik-teknik yang menghasilkan dokumen spesifik yang menjelaskan berbagai elemen sebuah system. Setiap proyek yang berbeda akan menekankan bagian-bagian yang berbeda dari SDLC. Untuk Mengimplementasikan SDLC terdapat berbagai metodologi yang dapat dipergunakan bervariasi tergantung kebutuhannya.



Gambar 1. System Development Life Cycle^[12]

Siklus hidup perangkat lunak adalah urutan dari kegiatan yang ada di dalam sebuah pengembangan perangkat lunak^[2]. Berdasarkan pengertian tersebut, secara umum dapat dikatakan bahwa proses pengembangan perangkat lunak mengikuti tahap-tahap:

a. *Planning*

Sebuah perencanaan yang biasanya lebih menonjolkan pada alasan apa sebuah system dibuat dan dikerjakan oleh perangkat lunak dalam satu rentang waktu tertentu.

b. *Analysis*

Tingkatan perencanaan yang dilanjutkan dengan proses analisis yang menekankan bagaimana, siapa, dan dimana system dibuat, mencakup arsitektur, antar muka internal, dan algoritma perangkat lunaknya.

c. *Design*

Proses desain lebih menekankan pada bagaimana system ini akan berjalan, penerapan dan pengujian unit-unit program.

d. *Implementation*

Tahap ini mengenai proses penyampaiannya pada pengguna, integrasi dan pengujian modul-modul program.

Setelah semua tahapan dilakukan maka perlu juga dilakukan pengujian sistem sebagai validasi perangkat lunak secara keseluruhan.

III. KOMPONEN DAN KARAKTERISTIK PENGEMBANGAN PERANGKAT LUNAK

Menurut Pressman^[8] bahwa komponen metodologi pengembangan perangkat lunak dapat dibagi ke dalam tiga unit, yaitu:

- A. Metode**, yaitu suatu cara atau teknik pendekatan yang sistematis 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.
- B. 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.
- C. Prosedur**, yang dipergunakan untuk mendefinisikan urutan-urutan pekerjaan (daur) dari metode dan alat bantu tersebut.

Menurut Despa^[6] bahwa karakteristik proyek pengembangan perangkat lunak terdiri dari: sering berubah spesifikasi, dinamika tinggi teknologi dan standard, tenaga kerja terampil, dan tim

didistribusikan secara global. Hal ini dapat dilihat pada Tabel 1.

TABEL 1
KARAKTERISTIK PROYEK PENGEMBANGAN PERANGKAT LUNAK [6]

Karakteristik	Dampak Positif	Dampak Negatif
Sering berubah spesifikasi		Membahayakan tenggat waktu Hasil melebihi anggaran proyek Menyebabkan stres dan ketidakpuasan bagi tim pengembangan
Dinamika tinggi dan standar teknologi	Menghasilkan peluang baru dalam dari segi desain dan coding	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

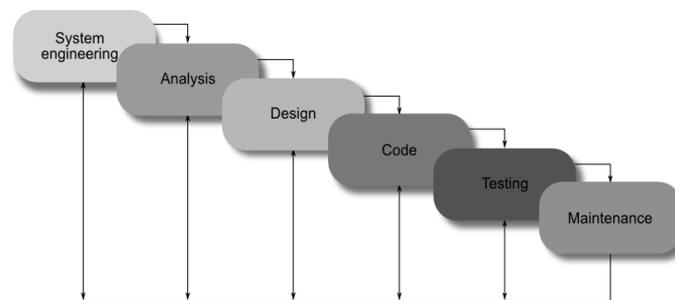
IV. MACAM-MACAM METODOLOGI PENGEMBANGAN PERANGKAT LUNAK

Metodologi pengembangan sistem perangkat lunak dapat diklasifikasikan dalam dua pendapat ahli, yaitu:

Pertama, menurut Ian Sommerville ^[10] bahwa model proses pengembangan perangkat lunak terbagi atas empat : *Prototyping, Formal System, Reuse-oriented software engineering, Waterfall*.

Kedua, menurut Pressman ^[8] – yang juga mejadi fokus pembahasan dalam paper ini – menyebutkan bahwa model proses pengembangan perangkat lunak terbagi atas lima metode : *Linear sequential model* atau *Waterfall, Incremental process model, Evolutionary process model, Rapid application development (RAD)*, dan *Concurrent model*.

A. Linear Sequential Model / Waterfall Model



Gambar 2. Linear Sequential Model

Model ini sering disebut juga dengan “*classic life cycle*” atau “*waterfall model*”. Model ini termasuk ke dalam model generic pada rekayasa perangkat lunak dan pertama kali diperkenalkan oleh Winston Royce sekitar tahun 1970 ^[5]. Linear sequential model adalah metode pengembangan perangkat lunak dengan pendekatan sekuensial dengan cakupan aktivitas:

1) 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.

2) 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.

3) 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.

4) Pembuatan Kode (Coding).

Penterjemahan perancangan ke bentuk yang dapat dimengerti oleh mesin, dengan menggunakan bahasa pemrograman.

5) 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 memeriksa apakah sesuai dengan hasil yang diinginkan.

6) 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:**

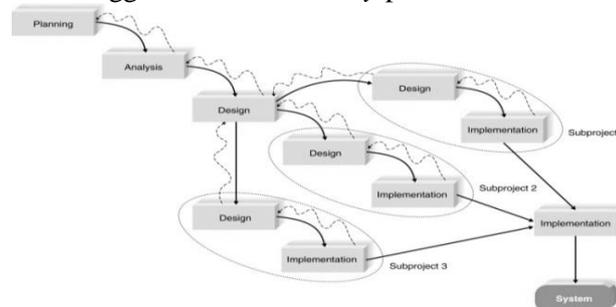
Penyesuaian dengan lingkungan baru, misalnya sistem operasi atau sebagai tuntutan atas perkembangan sistem komputer, misalnya penambahan printer driver.

iii) **Perfektive Maintenance:**

Bila perangkat lunak sukses dipergunakan oleh pemakai. Pemeliharaan ditujukan untuk menambah kemampuannya seperti memberikan fungsi-fungsi tambahan, peningkatan kinerja dan sebagainya.

B. Parallel Model

Menurut Dennis ^[1] Parallel Model merupakan metodologi yang mencoba untuk mengatasi interval waktu yang lama antara tahap analisis dan pengiriman sistem. 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.

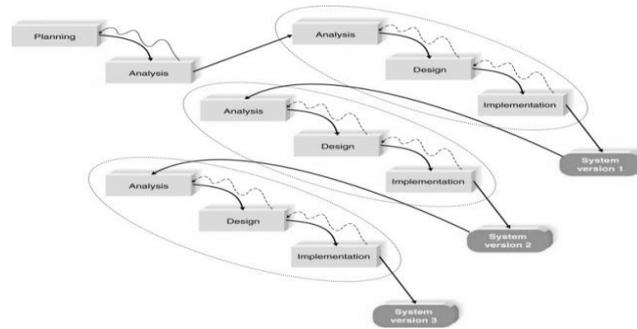


Gambar 3. Parallel Model [1]

C. 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 [4] Iterative Model merupakan metodologi yang mengandalkan pembangunan aplikasi perangkat lunak satu langkah pada satu waktu dalam bentuk memperluas model. 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.

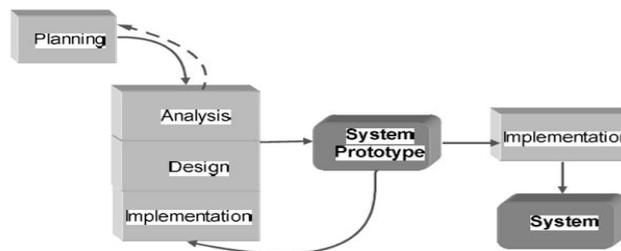


Gambar 4. Iterative Model

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.

D. Prototyping Model

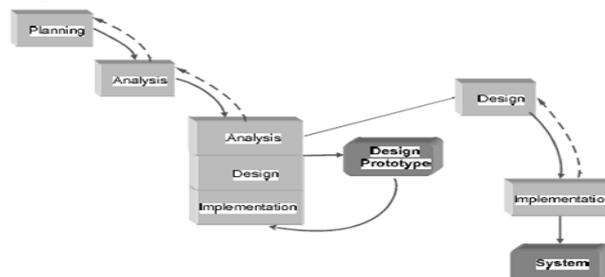


Gambar 5. Prototyping Model

Pendekatan prototyping model digunakan jika pemakai hanya mendefinisikan objektif umum dari perangkat lunak tanpa memerinci kebutuhan input, pemrosesan dan outputnya, sementara pengembang tidak begitu yakin akan efisiensi algoritma, adaptasi sistem operasi, atau bentuk antarmuka manusia-mesin yang harus diambil. Cakupan aktivitas dari prototyping model terdiri dari:

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

1) Throw away Prototyping Model

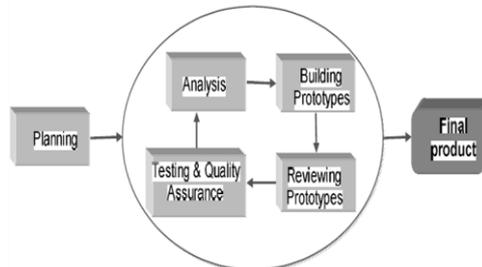


Gambar 6. Throw away Prototyping Model

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 *Throwaway Prototyping* dan *Prototyping*, di mana prototipe berkembang menjadi sistem final. Pendekatan ini menghasilkan lebih stabil dan dapat diandalkan sistem.

E. RAD (Rapid Application Development) Model

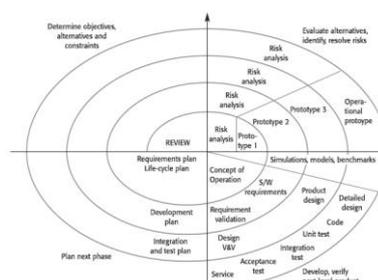


Gambar 7. RAD (Rapid Application Development) Model

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:

- 1) **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?
- 2) **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.
- 3) **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.
- 4) **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.
- 5) **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.

F. *Spiral Model*



Gambar 8. Spiral Model

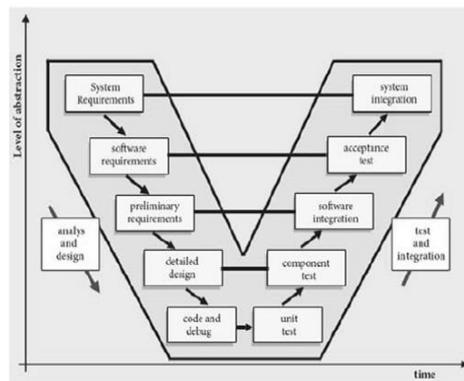
Merupakan model proses perangkat lunak yang memadukan wujud pengulangan dari model prototyping dengan aspek pengendalian dan sistematika dari linear sequential model, dengan penambahan elemen baru yaitu analisis resiko.

Model ini memiliki empat aktivitas penting, yaitu:

- 1) **Perencanaan (*Planning*)**. penentuan tujuan, alternatif, dan batasan.
- 2) **Analisis resiko (*Risk Analysis*)**. analisis alternatif dan identifikasi/pemecahan resiko.
- 3) **Rekayasa (*Engineering*)**. pengembangan level berikutnya dari produk.
- 4) **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 diidentifikasi 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.

G. V-Shaped Model



Gambar 9. V-Shaped Model

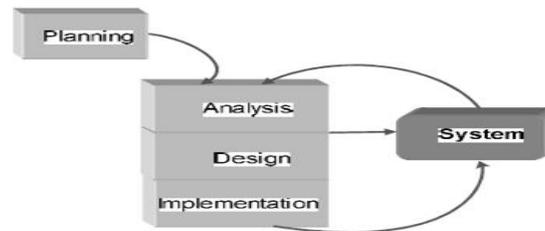
Sama seperti model air terjun, V- yang siklus hidup berbentuk jalan berurutan dari pelaksanaan proses. Setiap fase harus diselesaikan sebelum tahap berikutnya dimulai. Pengujian ditekankan dalam model ini lebih dari model air terjun. Prosedur pengujian yang dikembangkan di awal siklus hidup sebelum coding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

H. Agile Development

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:

Extreme Programming. 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

V. MEMILIH METODOLOGI PENGEMBANGAN YANG TEPAT

Beberapa pertimbangan pemilihan metodologi pengembangan perangkat lunak yang tepat menurut Dennis ^[1] terdiri dari beberapa kriteria meliputi: kejelasan kebutuhan pengguna (*clarity user requirement*), penguasaan teknologi (*familiarity with technology*), tingkat kerumitan sistem (*system complexity*), tingkat kehandalan sistem (*system reliability*), waktu pelaksanaan (*short time schedules*), dan visibilitas jadwal pelaksanaan (*schedule visibility*). Hal ini dijelaskan pada Tabel 2.

TABEL 2
KRITERIA PEMILIHAN METODOLOGI

Kriteria pengembangan sistem	Waterfall	Parallel	V-Model	Iterative	System Prototyping	Throwaway 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

VI. KELEBIHAN DAN KELEMAHAN METODOLOGI

TABEL 3
KELEBIHAN DAN KELEMAHAN METODOLOGI PENGEMBANGAN PERANGKAT LUNAK

No	Metodologi	Kelebihan	Kelemahan
1	Linear Sequential Model	1) Mudah dalam pengelolaan karena hampir seluruh requirements telah diidentifikasi 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.
2	Parallel Model	Waktu pengembangan sistem yang lebih singkat dibandingkan <i>waterfall</i> , karena beberapa tahapan diakselerasikan dengan membagi menjadi beberapa <i>sub project</i> .	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 permasalahan pada sub project secara bersamaan.
3	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 <i>waterfall</i> .
4	Prototyping Model	1) <i>Requirements identification</i> yang akurat karena dilakukan evaluasi secara berkala dan mendapatkan masukan dari <i>project owner</i> terhadap purwa rupa yang dihasilkan, 2) <i>User experience</i> yang meningkat, karena secara terus menerus melakukan uji coba dan evaluasi terhadap purwa rupa, 3) Kesalahan dan redundansi 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 rupan dapat dilakukan penyesuaian versi purwa rupa sesuai kebutuhan, hingga purwa rupa dapat disetujui oleh project owner.
5	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	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)

6	<i>Throwaway Prototyping Model</i>	orang dalam waktu singkat, 5) Penggunaan alat-alat dan kerangka kerja.	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.
7	<i>Spiral Model</i>	1) Jumlah analisis risiko yang tinggi, 2) Baik untuk proyek-proyek besar dan <i>mission-critical</i> , 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.
8	<i>V-Shaped Model</i>	1) Sederhana dan mudah digunakan, 2) Setiap fase memiliki <i>delivery</i> tertentu, 3) Kesempatan keberhasilan yang lebih tinggi atas model <i>waterfall</i> 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 <i>waterfall</i> , 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.
9	<i>Agile Development</i>	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 dan manajemen proyek tangkas praktek adalah suatu keharusan tanpa yang tidak akan bekerja.

VII. 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 organisasi, 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 ^[7] terdapat banyak model yang digunakan untuk mengembangkan sistem dengan ukuran yang berbeda dilihat dari project dan kebutuhannya. 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 Angela ^[3] 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 reliability*), waktu pelaksanaan (*short time schedules*), dan visibilitas jadwal pelaksanaan (*schedule visibility*). 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 ^[9] semua model pengembangan perangkat lunak memiliki kelebihan dan kekurangan. 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 diluncurkan 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 ^[6] metodologi pengembangan software mengikuti dua filosofi utama: kelas berat dan kelas ringan. 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.

VIII. 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 organisasi, 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.

DAFTAR PUSTAKA

- [1] Dennis. A, Wixom. B, and Roth. R. (2006). *System Analysis and Design*. John Wiley and Sons, Inc pp. 171-209.
- [2] Gustafson. K. L; Branch. R. M. (2002). *Survey of Instructional Development Models, Fourth Edition*
- [3] Ifeyinwa Angela Ajah, John Otozi Ugah. (2013). *Comparative Analysis of Software Development Methodologies*. Volume 3, Issue 6, June. www.ijarcsse.com.
- [4] Larman. C, Basili. V. R, "Iterative and Incremental Development: A Brief History". (2003). *Computer*, vol. 36, no. 6, pg. 47-56, doi:10.1109/MC.2003.1204375.
- [5] library.binus.ac.id/eColls/eThesisc/Bab1/2014-2-01054-MTIF%20Bab1001.pdf
- [6] Mihai Liviu Despa. (2014). *Comparative study on software development methodologies*. *Database Systems Journal* vol. V, no. 3.
- [7] Nabil Mohammed Ali Munassar and A. Govardhan. *A Comparison Between Five Models Of Software Engineering*. (2010). *IJCSI International Journal of Computer Science Issues*, Vol. 7, Issue 5, September.
- [8] Pressman, S. Roger. 2005. *Software Engineering: a Practitioner's Approach*. Seventh Edition.
- [9] Sanjana Taya, Shaveta Gupta, (2011). *A Comparison Between Five Models Of Software Engineering*. *IJCST* Vol. 2, Issue 4, Oct.–Dec.
- [10] Sommerville Ian. (2004). *Software Engineering*, Addison Wesley, 7th edition.
- [11] Susanto, Azhar. (2004). *Sistem Informasi Manajemen*. Bandung : Lingga Jaya.
- [12] Susanto, Azhar. (2004). *Sistem Informasi Manajemen*. Bandung : Lingga Jaya. *International Journal of Information Management*, 21 (5), 349-364. 2001.