**Merkurius : Jurnal Riset Sistem Informasi dan Teknik Informatika**
**Volume. 2 No. 4 Juli 2024**
e-ISSN: 3031-8912; dan p-ISSN: 3031-8904; Hal. 122-134
DOI: https://doi.org/10.61132/merkurius.v2i4.148

# Analisa Perbandingan Penggunaan Metodologi Pengembangan Perangkat Lunak (Waterfall, Prototype, Iterative, Spiral, Rapid Application Development (RAD))

**Az Zahra Dwi Nur Adiya**
Universitas Amikom Purwokerto

**Dea Lili Anggraeni**
Universitas Amikom Purwokerto

**Ilham Albana**
Universitas Amikom Purwokerto

Alamat: Jl. Letjend Pol. Soemarto No. 126, Watumas, Purwanegara, Kec. Purwokerto Utara, Banyumas, Jawa Tengah, 53127
*Korespondensi penulis: azzahradwinuradiya@gmail.com*

***Abstract***. *Methodology is a very important basic framework in the design and development of professional software, aimed at creating an information system that meets the business needs of an organization. Success in software development relies heavily on managing the software project as a whole. Establishing a methodology includes high dynamics in the design stages of a model that describes the activities and life cycle of a system. SDLC (Systems Development Life Cycle) or Systems Development Life Cycle, in the context of systems and software engineering, is a process that includes the creation and modification of systems, as well as the models and methodologies used to develop them. This discussion will explain the analysis of selecting software development methodologies, including waterfall, prototype, iterative, spiral, and rapid application development (RAD). It is hoped that the results of this discussion can provide considerations in choosing and using the right methodology based on each individual's needs, strengths and weaknesses, as well as other factors such as familiarity with technology, system complexity, system reliability, and short and precise time requirements, as well as referencing several scientific journals.*

***Keywords***: *Waterfall, Prototype, Iterative, Spiral, Rapid Application Development (RAD).*

**Abstrak**. Metodologi merupakan kerangka dasar yang sangat penting dalam perancangan dan pengembangan perangkat lunak profesional, bertujuan untuk menciptakan sistem informasi yang memenuhi kebutuhan bisnis suatu organisasi. Keberhasilan dalam pengembangan perangkat lunak sangat bergantung pada pengelolaan proyek perangkat lunak secara keseluruhan. Menetapkan sebuah metodologi mencakup dinamika tinggi dalam tahap-tahap perancangan model yang menggambarkan aktivitas dan siklus hidup suatu sistem. SDLC (*Systems Development Life Cycle*) atau Siklus Hidup Pengembangan Sistem, dalam konteks rekayasa sistem dan perangkat lunak, adalah proses yang mencakup pembuatan dan modifikasi sistem, serta model dan metodologi yang digunakan untuk mengembangkannya. Pembahasan ini akan menjelaskan analisis pemilihan metodologi pengembangan perangkat lunak, termasuk *waterfall, prototype, iterative, spiral, dan rapid application development* (RAD). Hasil dari pembahasan ini diharapkan dapat memberikan pertimbangan dalam memilih dan menggunakan metodologi yang tepat berdasarkan kebutuhan, kelebihan dan kelemahan masing-masing, serta faktor-faktor lain seperti familiaritas dengan teknologi, kompleksitas sistem, keandalan sistem, dan kebutuhan waktu yang singkat dan tepat, serta mereferensikan beberapa jurnal ilmiah.

**Kata kunci**: Waterfall, Prototype, Iterative, Spiral, Rapid Application Development (RAD).

*Received: Mei 30, 2024; Accepted: Juni 21, 2024; Published: Juli 30, 2024;*
** Az Zahra Dwi Nur Adiya , azzahradwinuradiya@gmail.com*

## LATAR BELAKANG

Saat ini, teknologi komputer berkembang dengan sangat cepat, baik dalam hal perangkat keras maupun perangkat lunak. Keberhasilan dalam pengembangan perangkat lunak sangat bergantung pada manajemen keseluruhan proyek perangkat lunak. Menentukan sebuah metodologi mencakup dinamika yang tinggi dalam berbagai tahap perancangan model yang menggambarkan aktivitas dan siklus 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 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 (Setiya Budi et al., 2016).

SDLC (*Systems Development Life Cycle*) atau Siklus Hidup Sistem, dalam konteks rekayasa sistem dan perangkat lunak, adalah proses untuk menciptakan dan memodifikasi sistem, serta model dan metodologi yang digunakan untuk mengembangkannya. Konsep ini biasanya merujuk pada sistem komputer atau informasi. SDLC juga merupakan pola yang diikuti dalam pengembangan sistem perangkat lunak, yang mencakup tahap-tahap perencanaan (*planning*), analisis (*analysis*), desain (*design*), implementasi (*implementation*), pengujian (*testing*), dan pemeliharaan (*maintenance*) (Pricillia & Zulfachmi, 2021).

Dalam rekayasa perangkat lunak, konsep SDLC menjadi dasar bagi berbagai metodologi pengembangan perangkat lunak. Metodologi-metodologi ini membentuk kerangka kerja untuk perencanaan dan pengendalian dalam pembuatan sistem informasi, yang merupakan proses pengembangan perangkat lunak. Terdapat 3 jenis metode siklus hidup sistem yang paling banyak digunakan, yakni siklus hidup sistem tradisional (*traditional system life cycle*), siklus hidup menggunakan prototyping (*life cycle using prototyping*), dan siklus hidup sistem orientasi objek (*object-oriented system life cycle*) (Pricillia & Zulfachmi, 2021).

Pemilihan metodologi pengembangan perangkat lunak yang tepat sangat penting untuk memastikan keberhasilan proyek pengembangan sistem informasi. Masing-masing metodologi, seperti *Waterfall, Prototype, Iterative, Spiral, dan Rapid Application Development* (RAD), menawarkan pendekatan yang unik dengan kelebihan dan kekurangannya sendiri. Artikel ini bertujuan untuk menganalisis dan membandingkan penggunaan berbagai metodologi tersebut guna memberikan panduan bagi organisasi dalam memilih metodologi

yang paling sesuai dengan kebutuhan mereka. Analisis ini akan mempertimbangkan faktor-faktor seperti kompleksitas sistem, kebutuhan bisnis, keandalan, waktu pengembangan, dan standarisasi organisasi, sehingga dapat membantu dalam pengambilan keputusan yang lebih informasional dan strategis dalam pengembangan perangkat lunak.

## KAJIAN TEORITIS

### Analisis

Menurut Kamus Besar Bahasa Indonesia (KBBI), analisis adalah penyelidikan terhadap suatu peristiwa (karangan, perbuatan, dan sebagainya) untuk mengetahui keadaan yang sebenarnya. Analisis merupakan kegiatan memfokuskan, mengabstraksikan, mengorganisasikan data secara sistematis dan rasional untuk memberikan bahan jawaban terhadap permasalahan (Suryana, 2015).

### Perbandingan

Menurut Kamus Lengkap Bahasa Indonesia, perbandingan berasal dari kata "banding" yang artinya persamaan. Secara khusus, "membandingkan" berarti mengadu dua hal untuk mengetahui persamaannya. Dalam konteks ini, perbandingan dijelaskan sebagai selisih atau persamaan antara dua hal atau lebih.

Perbandingan adalah metode pengkajian atau penyelidikan yang melibatkan pembandingan antara dua atau lebih objek kajian untuk memperluas dan memperdalam pengetahuan tentang objek yang sedang diteliti. Dalam perbandingan ini, objek yang dibandingkan sudah dikenal sebelumnya, namun pengetahuan tentang mereka belum sepenuhnya jelas dan tegas (Basah, 1994).

### Pengembangan Sistem

Berdasarkan Kamus Besar Bahasa Indonesia (KBBI), pengembangan adalah proses, cara, atau perbuatan untuk mengembangkan. Pengembangan dapat merujuk pada proses, produk, atau rancangan yang sedang dikembangkan. Pengembangan sistem mencakup pembangunan sistem baru yang bertujuan untuk menggantikan, memperbaiki, atau meningkatkan fungsi dari sistem yang sudah ada (Kusrini, 2007).

### Perangkat Lunak

(Ariani Sukamto & Shalahuddin, 2015) menjelaskan bahwa perangkat lunak merupakan program komputer yang terkait dengan dokumentasi seperti kebutuhan, desain model, dan panduan penggunaan. (Ladjamudin, 2013) juga mengungkapkan bahwa perangkat lunak terdiri dari kumpulan perintah atau fungsi yang ditulis dengan aturan tertentu untuk menginstruksikan komputer dalam menjalankan tugas tertentu.

Dari beberapa definisi di atas, dapat disimpulkan bahwa perangkat lunak adalah program komputer yang terdiri dari serangkaian perintah atau fungsi yang dirancang untuk menjalankan tugas-tugas spesifik.

***System Development Life Cycle* (SDLC)**

SDLC atau Software Development Life Cycle atau sering disebut juga System Development Life Cycle adalah proses mengembangkan atau mengubah suatu sistem perangkat lunak dengan menggunakan model-model dan metodologi yang digunakan orang untuk mengembangkan sistem-sistem perangkat lunak sebelumnya (berdasarkan best practice atau cara - cara yang sudah teruji baik) (Ariani Sukamto & Shalahuddin, 2015).

SDLC juga berfungsi untuk membagi peran dan tanggung secara jelas antara designer, business analyst dan project manager. Fungsi lain dari SDLC juga dapat memberikan gambaran jelas tentang input dan output dari satu tahap ke tahap berikutnya. Secara umum ada enam tahapan dalam SDLC yaitu perencanaan sistem, analisis sistem, perancangan sistem, implementasi sistem, pengujian sistem, dan pemeliharaan sistem (Paksi et al., 2023).

## METODE PENELITIAN

Metodologi penelitian merupakan suatu pendekatan ilmiah yang diterapkan dalam usaha untuk memperoleh atau mengumpulkan data dengan tujuan atau kegunaan tertentu (Sugiyono, 2019).

Dalam penelitian ini, penulis memanfaatkan data yang diperoleh dari tinjauan beberapa jurnal dan melakukan perbandingan di antaranya. Dengan kata lain, penelitian ini mengadopsi metode penelitian deskriptif. Metode deskriptif merupakan suatu metode penelitian dalam meneliti status dari kelompok manusia, suatu obyek, suatu sistem pemikiran, suatu set kondisi, ataupun suatu kelas peristiwa pada saat ini (Widiyanto, 2018). Tujuan dari penelitian deskriptif adalah untuk memberikan gambaran yang jelas dan detail mengenai situasi atau kondisi yang diamati, termasuk mencatat variasi, frekuensi, dan distribusi dari berbagai variabel yang terlibat. Dengan demikian, tujuan utama dari penelitian deskriptif adalah untuk menyajikan data secara sistematis sehingga memungkinkan pemahaman yang lebih baik tentang fenomena yang diteliti.

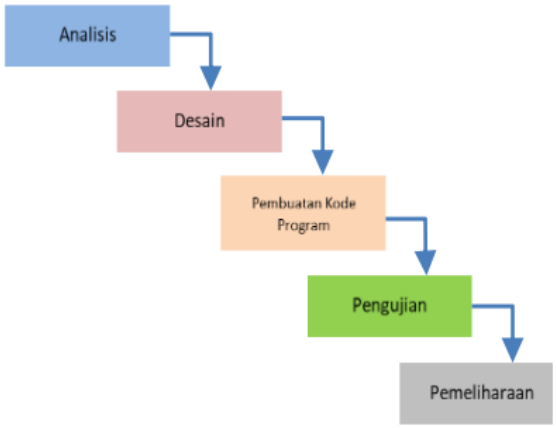
Langkah awal pengumpulan data menggunakan studi pustaka. Studi pustaka merupakan metode pengumpulan data yang diarahkan kepada pencarian data dan informasi melalui dokumen baik tertulis, foto, gambar, maupun dokumen elektronik yang mendukung dalam proses penulisan (Abdul Wahid, 2020). Sebagai objek penelitian dan perbandingan, penulis mengambil beberapa metode pengembangan perangkat lunak yaitu metode *Waterfall*,

*Prototype, Iterative, Spiral, dan Rapid Application Development* (RAD).

## HASIL DAN PEMBAHASAN

### Model *Waterfall*

Metode air terjun atau yang disebut metode waterfall sering dinamakan siklus hidup klasik (classic life cycle), nama model ini sebenarnya adalah "Linear Sequential Model" dimana hal ini menggambarkan pendekatan yang sistematis dan juga berurutan pada pengembangan perangkat lunak, dimulai dengan spesifikasi kebutuhan pengguna lalu berlanjut melalui tahapan-tahapan perencanaan (planning), permodelan (modelling), konstruksi (contruction), serta penyerahan sistem ke para pengguna (deployment), yang diakhiri dengan dukungan pada perangkat lunak lengkap yang dihasilkan (Abdul Wahid, 2020).




**Gambar 1. Model Waterfall**

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 dengan menentukan kebutuhan untuk semua elemen sistem, lalu memisahkan kebutuhan yang khusus untuk pengembangan perangkat lunak. Hal ini penting terutama ketika perangkat lunak harus berinteraksi dengan perangkat keras, pengguna, dan basis data.

2. **Analisis Kebutuhan Perangkat Lunak (*Software Requirements Analysis*)**

   Pengumpulan kebutuhan difokuskan pada perangkat lunak, yang mencakup: domain informasi, fungsi yang diperlukan, performa, dan antarmuka. Hasil pengumpulan ini harus didokumentasikan dan direview bersama pelanggan.

**3. Perancangan (*Design*)**

Ada empat atribut penting untuk sebuah program, yaitu: Struktur Data, Arsitektur Perangkat Lunak, Prosedur Detil, dan Karakteristik Antarmuka. Proses desain bertujuan mengubah kebutuhan-kebutuhan ini menjadi karakteristik yang dapat dipahami perangkat lunak sebelum penulisan kode dimulai. Desain ini harus terdokumentasi dengan baik dan menjadi bagian dari konfigurasi perangkat lunak.

**4. Pembuatan Kode (*Coding*)**

Penerjemahan perancangan ke dalam bentuk yang dapat dimengerti oleh mesin dilakukan dengan menggunakan bahasa pemrograman.

**5. Pengujian (*Testing*)**

Setelah kode program selesai, pengujian dapat dilakukan. Pengujian ini berfokus pada logika internal perangkat lunak, fungsi eksternal, serta mencari dan memperbaiki segala kemungkinan kesalahan untuk memastikan hasil sesuai dengan yang diinginkan.

**6. Pemeliharaan (Maintenance)**

*Maintenance* merupakan tahap terakhir dari siklus pengembangan dan dilakukan setelah perangkat lunak digunakan secara operasional. Kegiatan-kegiatan dalam maintenance meliputi:

a) *Corrective Maintenance*

Mengoreksi kesalahan pada perangkat lunak, yang baru terdeteksi pada saat perangkat lunak dipergunakan.

b) *Adaptive Maintenance*

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

c) *Perfektive Maintenance*

Bila perangkat lunak sukses dipergunakan oleh pemakai. Pemeliharaan ditujukan untuk menambah kemampuannya seperti memberikan fungsi-fungsi tambahan, peningkatan kinerja dan sebagainya (Bolung & Tampangela, 2017).

**Model *Prototype***

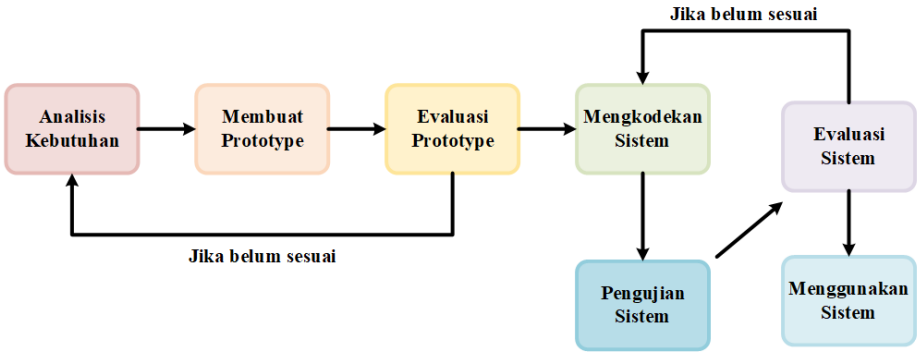
*Prototyping* merupakan proses yang digunakan untuk membantu pengembangan perangkat lunak dalam membentuk model perangkat lunak. *Prototype* ini adalah versi awal dari sebuah tahapan sistem perangkat lunak yang digunakan untuk mempresentasikan gambaran dari ide, mengeksperimenkan sebuah rancangan, mencari masalah yang ada sebanyak mungkin serta mencari solusi terhadap penyelesaian masalah tersebut. Model *prototype* yang

dipergunakan oleh sistem akan mengijinkan pengguna mengetahui seperti apa tahapan sistem yang dibuat sehingga sistem dapat mampu beroperasi secara baik (Fridayanthie et al., 2021).

Ada 4 metodologi prototyping yang paling utama yaitu:

1. *Illustrative*, menghasilkan contoh laporan dan tampilan layar.
2. *Simulated*, mensimulasikan beberapa alur kerja sistem tetapi tidak menggunakan data real.
3. *Functional*, mensimulasikan beberapa alaur sistem yang sebenarnya dan menggunakan data real.
4. *Evolutionary*, menghasilkan model yang menjadi bagian dari operasional sistem.

Adapun tujuan sebuah prototyping adalah untuk mengumpulkan informasi dari pengguna sehingga pengguna dapat berinteraksi dengan model *prototype* yang dikembangkan, sebab *prototype* menggambarkan versi awal dari sistem untuk kelanjutan sistem sesungguhnya yang lebih besar (Purnomo, 2017).




**Gambar 2. Model Prototype**

Model pengembangan perangkat lunak berbasis *prototype* diilustrasikan pada Gambar 1 dimulai dengan tahap analisis kebutuhan, di mana pengembang mengidentifikasi dan mendefinisikan semua kebutuhan perangkat lunak yang akan dibuat. Tahap berikutnya adalah pembuatan *prototype*, di mana sebuah rancangan sementara sistem dibuat dengan fokus pada alur perangkat lunak, serta metode dan algoritma yang digunakan.

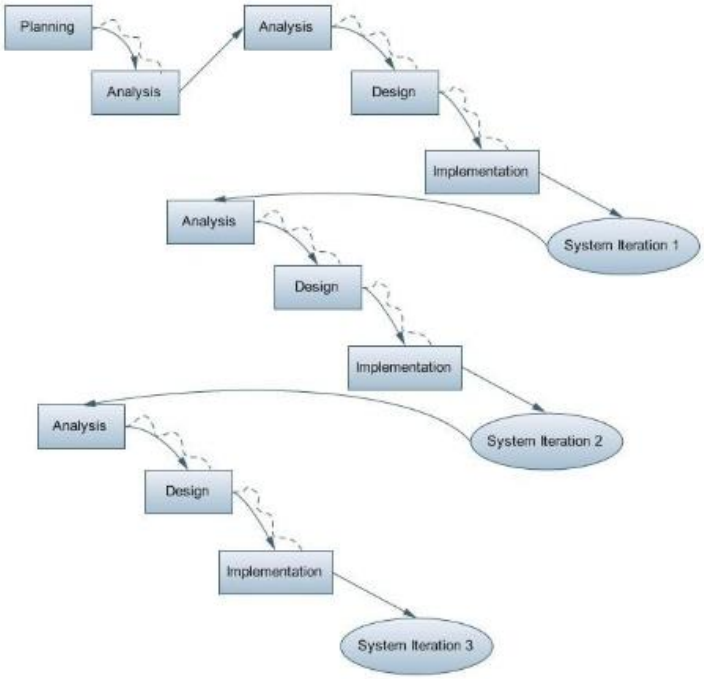
Setelah pembuatan *prototype*, tahap evaluasi *prototype* dilakukan untuk meninjau awal sistem dan memastikan apakah prototype tersebut sudah mencerminkan produk akhir dan sesuai dengan harapan klien. Jika hasil evaluasi *prototype* dinyatakan memenuhi persyaratan, proses dapat dilanjutkan dengan tahap pengkodean sistem menggunakan bahasa pemrograman yang sesuai.

Selanjutnya, dilakukan pengujian sistem untuk menguji kinerja perangkat lunak berdasarkan fungsinya. Pengujian ini melibatkan metode seperti *whitebox testing*, *blackbox testing*, dan metode lainnya untuk memastikan bahwa perangkat lunak beroperasi dengan baik.

Setelah selesai pengujian sistem, dilakukan evaluasi sistem oleh pengguna untuk meninjau apakah perangkat lunak sudah sesuai dengan harapan. Jika belum memenuhi persyaratan, tahapan pengkodean ulang sistem dan pengujian sistem diulang. Jika semua tahapan telah diselesaikan dengan memuaskan, tahap terakhir adalah implementasi atau penggunaan sistem, di mana perangkat lunak yang telah diuji dan disetujui siap untuk digunakan oleh pengguna.

**Model *Iterative***

Iterative Model merupakan metodologi yang mengandalkan pembangunan aplikasi perangkat lunak satu langkah pada satu waktu dalam bentuk memperluas model (Larman & Basili, 2003). 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 3. Model Iterative**

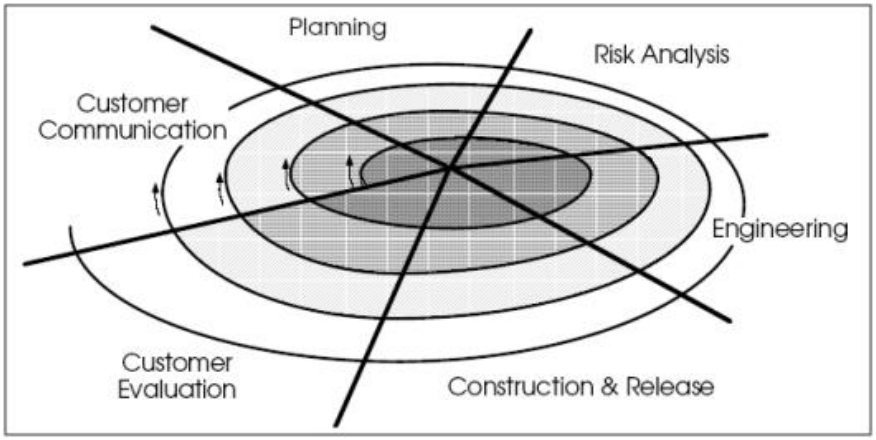
Model ini diimplementasi dengan cara perulangan, di mana proyek pengembangan perangkat lunak dibagi menjadi bagian-bagian kecil yang disebut iterasi (perulangan). Dengan pendekatan ini, tim pengembang bisa menghasilkan bagian-bagian perangkat lunak yang dapat diuji dan mendapatkan umpan balik dari pengguna secara berulang.

Setiap iterasi dalam pendekatan ini mirip dengan proses mini-Waterfall, di mana setiap tahapan memberikan umpan balik penting untuk tahapan desain berikutnya. Artinya, kita mengulang proses desain, pengkodean, pengujian, dan evaluasi untuk setiap bagian kecil atau iterasi dari proyek.

Dalam model ini, produk perangkat lunak yang dihasilkan pada akhir setiap iterasi atau serangkaian iterasi dapat langsung dimasukkan ke dalam produksi. Hal ini memungkinkan perangkat lunak untuk terus diperbaiki dan ditingkatkan secara bertahap berdasarkan umpan balik dari pengguna dan stakeholder lainnya. Pendekatan ini membantu dalam mengakomodasi perubahan kebutuhan serta memastikan bahwa produk yang dihasilkan sesuai dengan harapan pengguna.

**Model *Spiral***

Model *spiral* (*spiral model*) adalah model proses *software* yang evolusioner yang merangkai sifat iteratif dari prototipe dengan cara kontrol dan aspek sistematis dari model sekuensial linier. Model ini berpotensi untuk pengembangan versi pertambahan *software* secara cepat. Di dalam model spiral, *software* dikembangkan di dalam suatu deretan pertambahan. Selama awal iterasi, rilis inkremental bisa merupakan sebuah model atau prototipe kertas. Selama iterasi berikutnya, sedikit demi sedikit dihasilkan versi sistem rekayasa yang lebih lengkap (Aryo Nur Utomo & Muhammad Alfaridzi, 2018).




**Gambar 4. Model Spiral**

Model ini memiliki empat aktivitas penting, yaitu:

1. **Perencanaan (*Planning*)**

   Tahap ini melibatkan penentuan tujuan proyek, identifikasi alternatif solusi, dan menetapkan batasan-batasan yang relevan. Ini adalah fase di mana tujuan proyek ditetapkan dengan jelas serta batasan-batasan yang ada dipertimbangkan.

**2. Analisis Risiko (*Risk Analysis*)**

Pada tahap ini, dilakukan analisis mendalam terhadap risiko-risiko yang mungkin terjadi selama pengembangan perangkat lunak. Risiko-risiko ini diidentifikasi, dievaluasi, dan strategi mitigasi diputuskan untuk mengurangi atau mengelola risiko tersebut.

**3. Rekayasa (*Engineering*)**

Tahap ini melibatkan pengembangan level berikutnya dari produk perangkat lunak. Setiap iterasi dalam model Spiral menghasilkan versi yang semakin lengkap dari perangkat lunak, dimana setiap iterasi berfungsi sebagai siklus lengkap dalam dirinya sendiri yang mencakup analisis, desain, implementasi, pengujian, dan evaluasi.

**4. Evaluasi Pemakai (*Customer Evaluation*)**

Pada tahap ini, hasil rekayasa dievaluasi oleh pengguna atau pemangku kepentingan. Umpan balik dari evaluasi ini digunakan untuk memutuskan langkah selanjutnya dalam pengembangan produk.

***Rapid Application Development* (RAD)**

*Rapid Application Development* (RAD) adalah sebuah model proses pengembangan perangkat lunak yang menggunakan pendekatan sekuensial linier dengan penekanan pada siklus pengembangan yang sangat singkat, biasanya antara 60 hingga 90 hari. Model RAD ini menerapkan konsep "kecepatan tinggi" dari model sekuensial linier dengan cara memanfaatkan konstruksi berbasis komponen untuk mencapai pengembangan yang cepat.

*Rapid Application Development* (RAD) adalah model proses pengembangan perangkat lunak yang bersifat incremental terutama untuk waktu pengerjaan yang pendek. RAD merupakan model proses perangkat lunak yang menekankan pada daur pengembangan hidup yang singkat, dan versi adaptasi cepat dari metode *Waterfall* dengan menggunakan kontruksi komponen (Hariyanto et al., 2021).

Tahapan RAD terdiri dari 3 tahap yang terstruktur dan saling bergantung disetiap tahap, yaitu:

**1. *Requirements Planning* (Perencanaan Persyaratan)**

*Requirements Planning* dalam *Rapid Application Development* (RAD) dimulai dengan pertemuan antara pengguna dan analis untuk mengidentifikasi tujuan dari aplikasi atau sistem yang akan dikembangkan. Proses ini sangat berorientasi pada pemecahan masalah bisnis, di mana tujuan utama adalah untuk memahami dengan jelas kebutuhan bisnis yang akan dipenuhi oleh perangkat lunak yang dikembangkan.

**2. *Design Workshop***

*Design Workshop* merupakan tahap yang melibatkan serangkaian aktivitas untuk merancang dan menyempurnakan aplikasi atau sistem yang akan dikembangkan. Tim melibatkan kelompok pendukung keputusan sistem untuk membantu pengguna dalam menyetujui desain yang diusulkan. Programmer dan analis bekerja sama untuk membangun serta menampilkan tampilan visual dari desain dan alur kerja pengguna. Setelah itu, pengguna memberikan tanggapan terhadap prototipe kerja yang telah dibuat. Analis kemudian menggunakan umpan balik dari pengguna untuk menyempurnakan modul-modul yang dirancang, sehingga memastikan bahwa sistem memenuhi kebutuhan dan harapan pengguna dengan lebih baik.

**3. *Implementation* (Penerapan)**

Sebagai sistem yang baru dibangun, implementasi melibatkan pengujian dan pengenalan sistem baru atau bagian-bagian tertentu kepada organisasi. Proses ini memungkinkan untuk memastikan bahwa sistem berfungsi sesuai dengan yang diharapkan sebelum digunakan secara luas.

Dalam konteks RAD, tidak perlu menjalankan sistem lama secara paralel ketika sistem baru diperkenalkan. Pendekatan ini memungkinkan fokus penuh terhadap sistem baru yang telah dikembangkan, tanpa adanya kompleksitas atau biaya tambahan yang terkait dengan menjaga dan mengelola sistem lama secara bersamaan.

## KESIMPULAN DAN SARAN

### Kesimpulan

Dari hasil analisis, dapat disimpulkan bahwa:

1. Metodologi *Waterfall* cocok untuk proyek-proyek dengan persyaratan yang stabil dan jelas, namun kurang fleksibel terhadap perubahan.
2. Metodologi *Prototype* efektif dalam mempercepat pengembangan perangkat lunak dan memungkinkan pengguna untuk melihat dan memberikan umpan balik terhadap produk yang sedang dikembangkan.
3. Pendekatan *Iterative* memungkinkan adaptasi terhadap perubahan kebutuhan pengguna selama pengembangan, tetapi membutuhkan pengelolaan yang cermat untuk menghindari masalah kelalaian desain.
4. Metodologi *Spiral* menawarkan pendekatan yang berkelanjutan untuk mitigasi risiko dan evolusi proyek, tetapi memerlukan sumber daya yang lebih besar.

5. *Rapid Application Development* (RAD) mengutamakan kecepatan dalam pengembangan perangkat lunak dengan prototipe yang dapat diimplementasikan secara cepat, namun dapat menyebabkan pengorbanan dalam hal dokumentasi dan kualitas.

Dengan memahami karakteristik dan *trade-off* dari masing-masing metodologi, organisasi dapat membuat keputusan yang lebih informasional dalam memilih pendekatan yang paling sesuai dengan proyek mereka. Selain itu, penting untuk mempertimbangkan konteks proyek, sumber daya yang tersedia, dan kebutuhan bisnis secara menyeluruh dalam mengadopsi suatu metodologi pengembangan perangkat lunak. Dengan demikian, kesimpulan ini menjadi landasan yang kuat bagi pengambilan keputusan yang efektif dalam mengelola proyek pengembangan perangkat lunak secara sukses.

**Saran**

Untuk memilih metodologi pengembangan perangkat lunak yang tepat:

1. Lakukan evaluasi menyeluruh terhadap karakteristik proyek sebelum memilih metodologi.
2. Pilih metodologi yang memungkinkan adaptasi terhadap perubahan dengan efisien.
3. Pastikan tim pengembang memiliki keterampilan yang sesuai dengan metodologi yang dipilih.
4. Evaluasi secara berkala kinerja metodologi yang digunakan dan terapkan perbaikan berkelanjutan.

## DAFTAR REFERENSI

Abdul Wahid, A. (2020). Analisis Metode Waterfall Untuk Pengembangan Sistem Informasi. *Jurnal Ilmu-Ilmu Informatika Dan Manajemen STMIK*, *November*, 1–5.

Ariani Sukamto, R., & Shalahuddin, M. (2015). *Rekayasa Perangkat Lunak : Terstruktur dan Berorientasi Objek*. Informatika Bandung.

Aryo Nur Utomo, & Muhammad Alfaridzi. (2018). Perancangan Sistem Informasi Pada Percetakan CV Citra Kencana Jakarta Timur Berbasis Web. *Jurnal Rekayasa Informasi*, *7*(1), 43–47. https://ejournal.istn.ac.id/index.php/rekayasainformasi/article/view/273

Basah, S. (1994). *Perbandingan dan Persamaan*. Pustaka Pelajar.

Bolung, M., & Tampangela, H. R. K. (2017). Analisa Penggunaan Metodologi Pengembangan Perangkat Lunak. *Jurnal ELTIKOM*, *1*(1), 1–10. https://doi.org/10.1007/978-981-15-7530-3_9

Fridayanthie, E. W., Haryanto, H., & Tsabitah, T. (2021). Penerapan Metode Prototype Pada Perancangan Sistem Informasi Penggajian Karyawan (Persis Gawan) Berbasis Web. *Paradigma - Jurnal Komputer Dan Informatika*, *23*(2), 151–157. https://doi.org/10.31294/p.v23i2.10998

Hariyanto, D., Sastra, R., Putri, F. E., Informasi, S., Kota Bogor, K., & Komputer, T. (2021). Implementasi Metode Rapid Application Development Pada Sistem Informasi Perpustakaan. *Jurnal JUPITER*, *13*(1), 110–117.

Kusrini. (2007). *Konsep dan Aplikasi Sistem Pendukung Keputusan*. Penerbit Andi.

Ladjamudin, A.-B. Bin. (2013). *Analisis dan Desain Sistem Informasi*. Graha Ilmu.

Larman, C., & Basili, V. R. (2003). Iterative and incremental development: A brief history. *Computer*, *36*(6), 47–56. https://doi.org/10.1109/MC.2003.1204375

Paksi, A. B., Hafidhoh, N., & Bimonugroho, S. K. (2023). Perbandingan Model Pengembangan Perangkat Lunak Untuk Proyek Tugas Akhir Program Vokasi. *Jurnal Masyarakat Informatika*, *14*(1), 70–79. https://doi.org/10.14710/jmasif.14.1.52752

Pricillia, T., & Zulfachmi. (2021). Perbandingan Metode Pengembangan Perangkat Lunak (Waterfall, Prototype, RAD). *Jurnal Bangkit Indonesia*, *10*(1), 6–12. https://doi.org/10.52771/bangkitindonesia.v10i1.153

Purnomo, D. (2017). Model Prototyping Pada Pengembangan Sistem Informasi. *J I M P - Jurnal Informatika Merdeka Pasuruan*, *2*(2), 54–61. https://doi.org/10.37438/jimp.v2i2.67

Setiya Budi, D., Azhima Yoga Siswa, T., & Abijono, H. (2016). Analisis Pemilihan Penerapan Proyek Metodologi Pengembangan Rekayasa Perangkat Lunak. *24 TEKNIKA*, *5*(1).

Sugiyono, S. (2019). *Metodologi Penelitian Kualitatif, Kuantitatif, dan RD*. CV. Alfabeta.

Suryana. (2015). *Metode Penelitian*. CV. Pustaka Setia.

Widiyanto, W. W. (2018). Analisa Metodologi Pengembangan Sistem Dengan Perbandingan Model Perangkat Lunak Sistem Informasi Kepegawaian Menggunakan Waterfall Development Model, Model Prototype, Dan Model Rapid Application Development (Rad). *Jurnal Informa Politeknik Indonusa Surakarta ISSN*, *4*(1), 34–40. http://www.informa.poltekindonusa.ac.id/index.php/informa/article/view/34