Journal Of Informatics And Busisnes
E-ISSN:2988-4853
Vol. 01 No. 04 Januari – Maret 2024 Hal. 229-232
Journal homepage: https://jurnal.ittc.web.id/index.php/jibs/index
Published by: ITTC INDONESIA

# Pengembangan Software Menggunakan Model SDLC Guna Mencapai Keselarasan dengan Kebutuhan Pengguna

**Fitria Anisa, Fauzi Syahputra Harahap, Harits Al Khosyi, Intan Permata Sari, Yahfizham**
Sistem Informasi, Universitas Islam Negeri Sumatera Utara
E-mail: yahfizham@uinsu.ac.id

**Abstrak**
Software Development Life Cycle (SDLC) merinci kegiatan yang terjadi pada setiap tahap proses pengembangan perangkat lunak. Model SDLC menjadi krusial dalam menyusun perangkat lunak secara sistematis, memastikan pengiriman tepat waktu dan kualitas yang diinginkan. Model SDLC memberikan kerangka kerja teoritis untuk pengembangan perangkat lunak, dan karena variasi model yang ada, setiap SDLC memiliki kelebihan dan kelemahan yang digunakan sesuai dengan kebutuhan pengguna. Oleh karena itu, penting untuk memahami kebutuhan di mana suatu model SDLC akan diimplementasikan. Dalam penelitian ini, berbagai model SDLC terkemuka, seperti model waterfall, prototipe, RAD, model spiral, dan model agile, akan diulas. Dengan demikian, pembaca dapat menggunakan penelitian ini sebagai panduan untuk menemukan model yang paling sesuai dengan kebutuhan spesifik mereka.
**Kata Kunci:** SDLC, Waterfall, Prototype, RAD, Spiral, Agile

**Abstract**
*The Software Development Life Cycle (SDLC) details the activities that occur at each stage of the software development process. SDLC models are crucial in systematically structuring software, ensuring timely delivery and the desired quality. SDLC models provide a theoretical framework for software development, and due to the variety of models that exist, each SDLC has advantages and disadvantages that are used according to user requirements. Therefore, it is important to understand the needs in which an SDLC model will be implemented. In this research, various prominent SDLC models, such as the waterfall model, prototype, RAD, Spiral model, and Agile model, will be reviewed. As such, readers can use this research as a guide to find the model that best suits their specific needs.*
***Keywords:*** *SDLC, Waterfall, Prototype, RAD, Spiral, Agile*

## PENDAHULUAN

Model Siklus Hidup Pengembangan Perangkat Lunak (SDLC) digunakan sebagai pendekatan sistematis untuk implementasi dan modifikasi sistem. Metodologi ini dapat diterapkan pada sistem informasi, penulisan ulang sistem, dan penulisan ulang perangkat lunak. Menurut prinsip dasarnya, SDLC mengukur upaya kerja konseptual yang diterapkan oleh berbagai metodologi yang digunakan di pasar saat ini untuk produk perangkat lunak atau di industri untuk mengembangkan produk perangkat lunak. Hal ini menyoroti kebutuhan pekerja prosedural yang memfasilitasi produksi, administrasi, dan analisis sistem informasi. Dalam kondisi lanskap perangkat lunak saat ini, ada banyak model rekayasa perangkat lunak yang dapat diakses berdasarkan SDLC. Model-model rekayasa perangkat lunak ini memberikan kemampuan untuk mengelola dan mengawasi berbagai proyek perangkat lunak, tergantung pada konteks spesifiknya. Setiap metodologi atau model memiliki ambang batas risiko dan manfaat yang berbeda untuk mengatasi hambatan proyek, penundaan, dan kemungkinan keterlambatan pengiriman. Proyek-proyek besar dapat memperoleh manfaat dari sejumlah model yang menekankan pada proses yang fleksibel yang memungkinkan penyesuaian cepat pada setiap tahap siklus pengembangan perangkat lunak. Berdasarkan hal ini, beberapa peneliti telah mengevaluasi SDLC dan berbagai model telah dikembangkan dimana kekuatan dan kelemahannya dibahas. Ada beberapa model yang dapat dirujuk sebagai model SDLC yang sukses, diantaranya adalah *Waterfall*, Spiral, *Rapid Application Development* (RAD), Prototipe, dan Agile. Selain itu, semua model SDLC yang telah disetujui memiliki karakteristik dasar yang sama. Namun, dalam penelitian ini, kekuatan dan kelemahan model *Waterfall*, Spiral, Prototipe, RAD dan Agile yang akan dibahas bersama dengan perbandingan yang dekat dengan model lainnya.

## METODE

Eksplorasi literatur yang komprehensif diperlukan untuk metodologi penelitian. Metodologi studi literatur mencakup banyak prosedur yang berkaitan dengan ekstraksi data dari sumber-sumber perpustakaan, entri data yang tepat, pengumpulan data, dan pengurutan bahan penelitian secara sistematis. Peran studi literatur sangat penting dalam upaya penelitian, terutama dalam penelitian akademis, di mana tujuan utamanya adalah konstruksi

Journal Of Informatics And Busisnes
E-ISSN:2988-4853
Vol. 01 No. 04 Januari – Maret 2024 Hal. 229-232
Journal homepage: https://jurnal.ittc.web.id/index.php/jibs/index
Published by: ITTC INDONESIA

teoritis dan aplikasi praktis. Setiap peneliti harus melakukan studi literatur, dengan tujuan utama untuk mengidentifikasi literatur yang relevan untuk pengembangan kerangka teori, pengujian kerangka konseptual, dan temuan penelitian sekunder, yang secara umum disebut sebagai hipotesis penelitian. Hal ini memudahkan para peneliti untuk mengkategorikan, mengelompokkan, mengorganisasikan, dan menggunakan berbagai literatur yang sesuai dengan bidangnya. Melalui analisis literatur, para peneliti mengembangkan pemahaman yang lebih komprehensif dan mendalam tentang subjek yang mereka pelajari.

## HASIL DAN PEMBAHASAN

### A. *Waterfall* (air terjun)

Winston W. Royce adalah orang yang pertama kali mempresentasikan model *waterfall* pada tahun 1970. Model siklus hidup linier-sekuensial adalah nama lain untuk model *waterfall*. Paradigma ini menghasilkan kerangka kerja yang mudah dipahami dan digunakan. Tidak ada tumpang tindih antara tahapan dalam model *waterfall*; masing-masing harus selesai sepenuhnya sebelum pindah ke tahap berikutnya. Model *waterfall*, juga dikenal sebagai model siklus hidup urutan linier, menggambarkan proses pengembangan perangkat lunak sebagai aliran berurutan linier. Dalam fase ini, analisis kebutuhan, desain, pengkodean, pengujian, dan implementasi adalah langkah selanjutnya. Setiap fase berurutan, dengan fokus pada non-pengulangan, yang berarti bahwa suatu fase tidak diinisiasi kembali dan progres pengembangan tidak bergeser ke tahap berikutnya hingga fase sebelumnya sepenuhnya terselesaikan. Akibatnya, ada kelemahan untuk strategi ini, terutama jika kebutuhan proyek berubah.

**Kelebihan Metode *Waterfall***

1. Persyaratan jelas sebelum pengembangan dimulai
2. Refleksi keterpaduan rekayasa yang menjaga keberlanjutan kualitas perangkat lunak.
3. Struktur logis model ini membawa keuntungan dalam menghindari kesalahan konseptual.
4. Estimasi biaya total dapat dilakukan dengan tingkat akurasi yang relatif tinggi dalam kondisi tanpa konflik.
5. Proses ini mendefinisikan titik awal dan akhir yang pasti dari sebuah proyek.

**Kekurangan Metode *Waterfall***

1. Ketidakmampuan untuk melakukan retret. Setelah suatu tahap diselesaikan, hal tersebut mengindikasikan bahwa tahap tersebut telah terkunci dan tidak dapat dikembalikan.
2. Apabila modifikasi diperlukan, proyek harus diinisiasi kembali dari tahap awal. Situasi semacam ini dapat menimbulkan biaya yang substansial bagi sejumlah organisasi.
3. Kualitas dapat dinilai oleh pengguna secara eksklusif pada tahap akhir.
4. Produk yang dikirim tidak dihasilkan dalam periode waktu yang singkat.
5. Dalam proyek *outsourcing*, hal ini dapat menjadi kerugian yang krusial, karena penundaan pada tanggal rilis dapat terjadi dan kondisi pasar mungkin telah mengalami perubahan selama periode tersebut.

### B. Prototype

Model ini diterapkan untuk mengatasi keterbatasan model *waterfall*. Sebaliknya, daripada membekukan persyaratan sebelumnya dalam tahap pengkodean, pengujian, atau desain, suatu prototipe dibangun dengan tujuan memahami persyaratan dengan lebih jelas. Prototipe ini dikonstruksi berdasarkan persyaratan yang berlaku pada saat itu. Melalui evaluasi prototipe ini, klien memperoleh pemahaman yang lebih mendalam mengenai fitur dari produk akhir. Prototipe itu sendiri merupakan representasi model dari suatu produk yang mungkin belum melibatkan seluruh fitur produk yang sebenarnya, namun telah mencakup fitur-fitur inti dari produk yang sebenarnya. Prototipe sering digunakan untuk keperluan pengujian sebelum memasuki tahap pembuatan produk sebenarnya. Dengan penerapan metode prototype ini, para pengembang dan pelanggan dapat berinteraksi secara timbal balik selama proses pembuatan suatu produk.

**Kelebihan Metode Prototipe**

1. Saat model prototipe disajikan kepada pengguna, pihak tersebut memperoleh pemahaman yang lebih akurat mengenai persyaratannya. Pengguna juga dapat merasakan fungsionalitas perangkat lunak sehingga dapat memberikan saran terkait perubahan dan modifikasi yang diperlukan.
2. Tindakan ini mengurangi risiko kegagalan, karena kemungkinan risiko dapat terdeteksi pada tahap awal dan tindakan yang diperlukan dapat diambil untuk mengeliminasi risiko tersebut.
3. Tanggapan pengguna yang lebih cepat dapat diakses, membawa konsekuensi pada perbaikan solusi yang lebih optimal.

**Kekurangan Metode Prototipe**

Journal Of Informatics And Busisnes
E-ISSN:2988-4853
Vol. 01 No. 04 Januari – Maret 2024 Hal. 229-232
Journal homepage: https://jurnal.ittc.web.id/index.php/jibs/index
Published by: ITTC INDONESIA

1. Meskipun pengguna dapat mengamati peningkatan dari setiap iterasi prototipe, namun mungkin tidak disadari bahwa pembuatan versi tersebut dilakukan tanpa mempertimbangkan aspek kualitas dan pemeliharaan jangka panjang.
2. Risiko yang signifikan terkait dengan masalah-masalah yang kurang terstruktur, perubahan yang substansial seiring waktu, dan ketidakpastian terkait persyaratan data.
3. Analisis masalah yang tidak memadai.

## C. RAD

Metode *Rapid Application Development* (RAD) merujuk pada suatu model pengembangan perangkat lunak yang tergolong dalam teknik bertingkat. Dalam prosesnya, model ini menerapkan metode iteratif atau berulang di mana sebuah model sistem yang berfungsi dibangun pada tahap awal pengembangan untuk menetapkan kebutuhan pengguna. Pendekatan RAD memusatkan perhatian pada pengumpulan kebutuhan pelanggan melalui lokakarya atau kelompok fokus, pelaksanaan pengujian awal pada prototipe oleh pelanggan dengan menggunakan konsep berulang, pemanfaatan kembali prototipe yang sudah ada (komponen), integrasi yang berkelanjutan, dan penyampaian solusi secara cepat. Dalam kerangka model ini, prototipe dikonstruksi, sementara fitur-fitur secara progresif dikembangkan dan diserahkan kepada pelanggan. Sebaliknya dengan model prototipe, prototipe yang telah ada tidak diabaikan, melainkan ditingkatkan dan dimanfaatkan dalam proses konstruksi perangkat lunak.

**Kelebihan Metode RAD**

1. Penerapan Metode RAD dapat diselesaikan dalam jangka waktu yang relatif singkat, antara 30 hingga 90 hari.
2. Kejelasan proses mempermudah pemahaman pengguna terhadap sistem yang sedang dikembangkan.
3. Fleksibilitas meningkat karena pengembang dapat melakukan perancangan ulang secara simultan.
4. Penggunaan alat bantu (*case tools*) membantu mengurangi kemungkinan kesalahan.
5. Adanya RAD menghasilkan sistem dengan biaya yang lebih efisien dan menghasilkan sistem dengan cepat.

**Kekurangan Metode RAD**

1. Peningkatan kecepatan dan penurunan biaya dapat berdampak pada kualitas sistem secara menyeluruh.
2. Pengurangan fasilitas-fasilitas yang ditawarkan seringkali terjadi akibat keterbatasan waktu yang tersedia.
3. Kesulitan dengan penggunaan kembali modul untuk sistem masa depan.

## D. Spiral

Model ini diberi nama berdasarkan representasi diagram yang menyerupai spiral dengan lilitan yang berkelanjutan. Jumlah pasti putaran spiral tidak ditetapkan dan dapat bervariasi antara satu proyek dengan proyek lainnya. Setiap putaran spiral disebut sebagai fase dalam proses pengembangan perangkat lunak. Jumlah pasti fase di mana produk dikembangkan dapat disesuaikan oleh manajer proyek, bergantung pada tingkat risiko proyek. Salah satu ciri khas yang mencolok dari model spiral ini adalah kemampuannya untuk mengatasi risiko yang tidak terduga yang mungkin muncul dalam tahap yang jauh setelah proyek dimulai. Model Spiral adalah suatu model proses perangkat lunak yang bersifat evolusioner, menggabungkan sifat iteratif dari prototipe dengan kontrol dan aspek sistematis dari model sekuensial linier.

**Kelebihan Metode Spiral**

1. Dikonseptualisasikan untuk mencakup karakteristik terbaik dari model *waterfall* dan Prototype
2. Pemantauan proyek menjadi suatu proses yang mudah dan efisien saat dilakukan pada setiap tahap iterasi.
3. Manajemen risiko menjadi fitur utama dalam model ini, menjadikannya lebih menarik dibandingkan dengan model lainnya.
4. Dikarenakan prototipe dibuat dalam segmen kecil, estimasi biaya menjadi lebih terukur, dan pelanggan dapat mengendalikan aspek administrasi dari sistem yang baru dikembangkan.

**Kekurangan Metode Spiral**

1. Metode ini mahal untuk digunakan
2. Kesuksesan suatu proyek sangat bergantung pada fase analisis risiko

Journal Of Informatics And Busisnes
E-ISSN:2988-4853
Vol. 01 No. 04 Januari – Maret 2024 Hal. 229-232
Journal homepage: https://jurnal.ittc.web.id/index.php/jibs/index
Published by: ITTC INDONESIA

3. Investasi waktu yang dialokasikan untuk perencanaan, penyesuaian tujuan, pelaksanaan analisis risiko, dan prototyping mungkin berlebihan.

**E. Agile**

Agile adalah kumpulan metode pengembangan perangkat lunak yang mengadopsi pendekatan berulang dan inkremental, di mana evolusi persyaratan dan solusi terjadi melalui kerjasama tim lintas fungsional yang memiliki otonomi. Pendekatan ini mendorong perencanaan yang dapat beradaptasi, pengembangan dan pengiriman secara evolusioner, penerapan siklus iteratif dalam kerangka waktu tertentu, serta mempromosikan respons yang cepat dan fleksibel terhadap perubahan. Metode ini merupakan suatu konseptualisasi kerangka kerja yang mengadvokasi interaksi yang diantisipasi sepanjang proses pengembangan. Komunikasi, fleksibilitas, dan analisis yang cermat menjadi elemen kunci bagi keberhasilan proyek perangkat lunak. Oleh karena itu, metode agile diterapkan dalam pengembangan perangkat lunak, menunjukkan pentingnya fleksibilitas untuk beradaptasi saat kondisi berubah.

**Kelebihan Metode Agile**

1. Pengintegrasian perubahan dapat dilakukan dengan sangat mudah.
2. Tidak ada ketidakpastian yang muncul antara tim pengembangan dan pelanggan, sebab terdapat komunikasi tatap muka dan umpan balik berkelanjutan dari klien.

**Kekurangan Metode Agile**

1. Lebih sering terdapat kendala dalam susunan tim dibandingkan kendala dalam proses perencanaan.
2. Hanya para pengembang berpengalaman yang bisa membuat keputusan yang diperlukan dalam konteks pengembangan agile. Model ini hampir tidak memberikan ruang bagi programmer pemula, kecuali jika mereka bekerja sama dengan sumber daya yang lebih berpengalaman.

## PENUTUP

### Simpulan

Terdapat berbagai model Siklus Hidup Pengembangan Perangkat Lunak (SDLC), seperti *waterfall*, prototype, RAD, spiral, agile, dan sebagainya yang digunakan di berbagai organisasi sesuai dengan kondisi yang berlaku di tempat tersebut. Setiap model SDLC ini memiliki kelebihan dan kekurangan masing-masing. Dalam konteks ini, kami melakukan perbandingan antara berbagai model siklus hidup pengembangan perangkat lunak, dengan fokus pada fitur-fitur tertentu seperti spesifikasi persyaratan, keterlibatan risiko, keterlibatan pengguna, dan biaya. Berdasarkan fitur-fitur ini, untuk suatu proyek perangkat lunak tertentu, seseorang dapat memilih model SDLC yang paling sesuai. Pengambilan keputusan mengenai model siklus hidup yang tepat memiliki signifikansi yang besar dalam industri perangkat lunak, mengingat perlunya pengiriman perangkat lunak dalam batas waktu yang ditentukan dan dengan kualitas yang diharapkan. Penelitian ini diharapkan dapat menyederhanakan proses pemilihan model SDLC.

## DAFTAR PUSTAKA

Alshamrani, A., & Bahattab, A. (n.d.). *A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model*. www.IJCSI.org

Amlani, R. D. (2012). Advantages and Limitations of Different SDLC Models. In *International Journal of Computer Applications & Information Technology: Vol. I*. www.ijcait.com

Arora, R., & Arora, N. (2016). International Journal of Current Engineering and Technology Analysis of SDLC Models. In *268| International Journal of Current Engineering and Technology* (Vol. 6, Issue 1). http://inpressco.com/category/ijcet

Hasanah, F. N., & Untari, R. S. (2020). Buku Ajar Rekayasa Perangkat Lunak. *Umsida Press*, 1–119.

Iqbal, S. Z., & Idrees, M. (2017). Z-SDLC model: a new model for software development life cycle (SDLC). *International Journal of Engineering and Advanced Research Technology (IJEART)*, *3*(2), 8.

Kumar Dora, S., & Dubey, P. (n.d.). *SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC) ANALYTICAL COMPARISON AND SURVEY ON TRADITIONAL AND AGILE METHODOLOGY*. www.abhinavjournal.com

Kumar, N., Zadgaonkar, A. S., & Shukla, A. (2013). Evolving a new software development life cycle model SDLC-2013 with client satisfaction. *International Journal of Soft Computing and Engineering (IJSCE)*, *3*(1), 2231–2307.

Kute, S. S., & Thorat, S. D. (2014). A review on various software development life cycle (SDLC) models. *International Journal of Research in Computer and Communication Technology*, *3*(7), 778–779.

Migunani, S., Kom, M., & Kom. (n.d.). *Rekayasa Perangkat Lunak*.

Mishra, A., & Dubey, D. (2013). A Comparative Study of Different Software Development Life Cycle Models in Different Scenarios. *International Journal of Advance Research in Computer Science and Management Studies*, *1*(5). www.ijarcsms.com

Mujumdar, A., Masiwal, G., & Chawan, P. M. (n.d.). Analysis of various Software Process Models. In *International Journal of Engineering Research and Applications (IJERA)* (Vol. 2). www.ijera.com

Murugaiyan, D. (2012). *International Journal of Information Technology and Business Management WATEERFALLVs V-MODEL Vs AGILE: A COMPARATIVE STUDY ON SDLC*. *2*(1). www.jitbm.com

Verma, S. (2014). Analysis of Strengths and Weakness of SDLC Models. *International Journal of Advance Research in Computer Science and Management Studies*, *2*(3). www.ijarcsms.com