# Apa itu SDLC (Software Development Life Cycle)?

Oleh Anisha Verma | Terakhir diperbarui pada 8 Agustus 2025 | 90038 Tayangan


Sebelumnya Berikutnya

Daftar Putar Tutorial

Sumber Daya Pemrograman Terbaik

Apa itu Algoritma: Definisi, Jenis, Karakteristik

Apa itu Array? Panduan Lengkap Beserta Contoh

Apa itu BIOS (Basic Input/Output System)?

Apa itu Struktur Data?

Apa itu FastAPI? Fitur dan Manfaatnya

Apa itu Gradle? Panduan untuk Pemula

Apa itu Hash Table? – Penjelasan Komprehensif (Diperbarui 2026)

Apa itu MATLAB?

Apa itu Maven?

Apa itu Middleware?

Apa itu Sistem Operasi?

Tujuan Kelas Abstrak di Java

Apa itu PyCharm?

Pengantar Pygame di Python: Sebu[illegible]taka Pengembangan Game

## Daftar isi



Industri teknologi saat ini memiliki permintaan yang tinggi untuk pengembang perangkat lunak. Menurut Indeed , gaji rata-rata di India adalah 8,4 LPA dan $120.112 di AS. Akibatnya, sangat penting untuk memahami siklus hidup pengembangan perangkat lunak (SDLC). Menurut TechReport , metodologi SDLC seperti Waterfall, Agile, dan DevOps digunakan oleh lebih dari 71% tim pengembangan di seluruh dunia, menjadikannya aspek fundamental manajemen proyek yang mengawasi setiap tahap, dari inisiasi hingga implementasi.

Dalam blog ini, kita akan membahas SDLC dan konsep-konsep terkait lainnya secara detail.

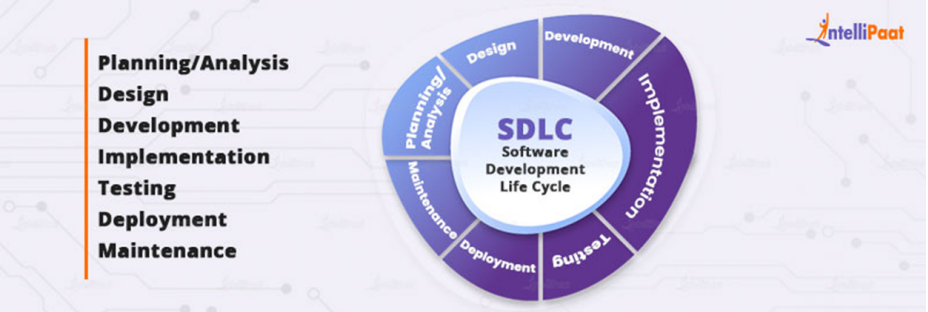
***Jika Anda ingin mengetahui semua yang ditawarkan SDLC, silakan tonton*** **video ini.**


## Apa Itu Pengembangan Perangkat Lunak?


Pengembangan perangkat lunak dapat didefinisikan sebagai proses yang melibatkan pembuatan, perancangan, pengembangan, pengujian, penyebaran, dan pemeliharaan aplikasi perangkat lunak. Namun tunggu, definisi ini tidak terbatas di sini. Pengembangan perangkat lunak juga mencakup penulisan kode dan pengembangan alat serta algoritma perangkat lunak untuk aplikasi tersebut.

Bidang ini juga mencakup aktivitas lain seperti manajemen proyek, desain UI/UX , dan jaminan kualitas . Karena semua faktor ini, permintaan dan pertumbuhan pesat sektor pengembangan perangkat lunak tidak lagi mengejutkan siapa pun. Bahasa pemrograman yang paling populer yang digunakan oleh SDE (Software Development Engineer) adalah Python .

## Apa itu SDLC?




Siklus hidup pengembangan perangkat lunak (SDLC) merujuk pada proses yang digunakan tim TI untuk merancang, mengembangkan, menguji, menyebarkan, dan memelihara aplikasi dan sistem perangkat lunak berkualitas tinggi. Memahami fase-fase SDLC sangat penting untuk mengelola proyek perangkat lunak yang besar atau kompleks.

Berikut gambaran singkat tentang siklus hidup pengembangan perangkat lunak:

1. Perencanaan Proyek
2. Analisis Proyek
3. Desain Proyek
4. Pelaksanaan Proyek
5. Pengujian Proyek
6. Implementasi Proyek

## Pentingnya SDLC

SDLC penting karena menyediakan kerangka kerja untuk mengembangkan aplikasi perangkat lunak berkualitas secara tepat waktu dan hemat biaya.

Hal ini juga membantu memastikan bahwa aplikasi memenuhi kebutuhan dan persyaratan pengguna. Dengan mematuhinya, organisasi dapat mengurangi risiko dan mendapatkan visibilitas ke dalam proses pengembangan.

Organisasi sering menggunakan Siklus Hidup Pengembangan Sistem (System Development Life Cycle/SDLC) untuk membuat dan memelihara aplikasi perangkat lunak dan sistem terkait lainnya. Perencanaan, perancangan, pembuatan, pengujian, dan penyebaran aplikasi perangkat lunak merupakan proses sistematis.

## Bagaimana Cara Kerja SDLC?

Proses yang dikenal sebagai Siklus Hidup Pengembangan Perangkat Lunak (SDLC) umumnya digunakan oleh tim pengembangan perangkat lunak untuk merencanakan , membuat, menguji, dan meluncurkan produk perangkat lunak. Berikut adalah uraian bagaimana proses ini berlangsung:

1. **Perencanaan:** Dimulai dengan fase awal ini di mana ruang lingkup proyek, tujuan, persyaratan, dan sumber daya didefinisikan. Para pemain kunci harus berkumpul untuk menyusun cetak biru proyek dan menetapkan jadwal waktu.

2. **Analisis:** Tahap ini adalah tentang pemeriksaan persyaratan oleh anggota tim termasuk pemangku kepentingan saat mereka berupaya memahami kebutuhan pengguna, tujuan bisnis , dan kendala teknis.

3. **Desain:** Pada tahap ini, tim menyusun desain perangkat lunak berdasarkan persyaratan tersebut. Ini akan melibatkan pembuatan rencana dan pembuatan struktur basis data serta antarmuka pengguna.

4. **Implementasi:** Menulis kode sesuai dengan spesifikasi desain selama fase pengkodean pengembang atau yang juga dapat disebut implementasi. Ini mencakup pembangunan komponen perangkat lunak dan integrasi di antara komponen-komponen tersebut.

5. **Pengujian:** Pengujian aplikasi perangkat lunak merupakan langkah penting yang bertujuan untuk mengidentifikasi kesalahan yang mungkin terjadi selama pengembangan. Pengujian tersebut meliputi pengujian unit , pengujian integrasi, pengujian sistem , dan pengujian penerimaan yang akan menjamin bahwa perangkat lunak sesuai dengan standar kualitas yang ditetapkan dan berfungsi dengan baik.

6. **Implementasi:** Langkah selanjutnya setelah disetujui dan diuji adalah meluncurkan perangkat lunak ini ke lingkungan produksi. Proses ini meliputi instalasi perangkat lunak, konfigurasi yang tepat, dan membuatnya dapat diakses oleh pengguna.

Setelah diimplementasikan, perangkat lunak beralih ke fase pemeliharaan. Pada tahap ini, fokusnya adalah menggabungkan masukan pengguna, menyelesaikan masalah, memperkenalkan fungsionalitas, dan memastikan bahwa perangkat lunak berfungsi dengan lancar untuk mengakomodasi kebutuhan yang terus berkembang.

## Manfaat SDLC

1. **Peningkatan Visibilitas dan Kolaborasi**

Kerangka kerja SDLC memfasilitasi penyelarasan di antara para pemangku kepentingan dan anggota tim dengan menetapkan tujuan, hasil yang diharapkan, dan tonggak pencapaian untuk setiap fase. Peningkatan transparansi ini memungkinkan para manajer , pengembang, dan pelanggan untuk berkolaborasi dalam meluncurkan perangkat lunak berkualitas tinggi.

2. **Peningkatan Estimasi, Perencanaan, dan Pelacakan**

Dengan memecah proyek ke dalam fase SDLC, Anda dapat meningkatkan akurasi estimasi, manajemen jadwal, alokasi sumber daya , dan alur kerja proyek secara keseluruhan. Pemantauan kemajuan memastikan bahwa proses pengembangan tetap sesuai rencana baik dari segi waktu maupun anggaran.

3. **Risiko yang Dimitigasi**

Pengumpulan persyaratan di awal dan pengujian rutin membantu mencegah masalah penentuan ruang lingkup yang mahal dan kegagalan pengiriman. Jika perlu, perubahan dikelola secara sistematis melalui model Agile.

## Keuntungan dan Kerugian SDLC

### Keunggulan SDLC

- **Pendekatan Terstruktur:** SDLC menawarkan kerangka kerja yang terstruktur dan teratur untuk pengembangan perangkat lunak guna memastikan bahwa semua tahapan proses pengembangan direncanakan dan dilaksanakan.
- **Peningkatan Jaminan Mutu:** Dengan mengikuti beberapa langkah yang telah disiapkan, SDLC membantu mengidentifikasi dan menyelesaikan cacat pada tahap awal siklus pengembangan sehingga meningkatkan kualitas perangkat lunak.
- **Manajemen Proyek yang Lebih Baik:** Manajemen proyek yang lebih baik difasilitasi oleh SDLC karena memecah setiap fase proses pengembangan menjadi tonggak pencapaian, hasil yang diharapkan, dan tenggat waktu yang jelas. Hal ini mempermudah pelacakan kemajuan serta manajemen sumber daya yang efektif.
- **Komunikasi yang Lebih Baik:** Di satu sisi, para pemangku kepentingan seperti pengembang, penguji, manajer proyek, dan pelanggan didorong untuk berkolaborasi melalui SDLC sehingga menghasilkan pemahaman yang lebih baik tentang kebutuhan dan persyaratan.
- **Manajemen Risiko:** Selain itu, tim dapat menggunakan teknik manajemen risiko yang membantu mereka mengidentifikasi potensi risiko atau ancaman sehingga mereka dapat merumuskan langkah-langkah yang diperlukan untuk mengurangi dampaknya terhadap proyek.

### Kelemahan SDLC

- **Kekakuan:** SDLC mungkin kaku dan tidak fleksibel, sehingga sulit untuk memasukkan perubahan persyaratan atau beradaptasi dengan perubahan kebutuhan proyek, terutama dalam lingkungan yang serba cepat.
- **Memakan Waktu:** Proses SDLC (Software Development Life Cycle) yang mencakup pengumpulan persyaratan, desain, pengembangan, pengujian, dan peluncuran produk akhir membutuhkan waktu yang lama.
- **Biaya Tinggi:** Dibandingkan dengan metode agile lainnya, SDLC seringkali memiliki investasi awal yang tinggi untuk perencanaan, dokumentasi, dan sumber daya, yang mengakibatkan peningkatan biaya pengembangan.
- **Keterlibatan Pemangku Kepentingan yang Terbatas:** Terkadang keterlibatan pemangku kepentingan mungkin terbatas pada tahapan tertentu dalam Siklus Pengembangan Perangkat Lunak (SDLC), yang menyebabkan kurangnya umpan balik dan kolaborasi berkelanjutan.
- **Penekanan Berlebihan pada Dokumentasi:** Terkadang terjadi penekanan berlebihan pada dokumentasi dalam SDLC yang mengakibatkan terlalu banyak pekerjaan administrasi yang mengalihkan sumber daya dari aktivitas pengembangan yang sebenarnya.

## Metodologi SDLC Utama

Pendekatan Waterfall , Agile , dan Spiral memiliki kesamaan dalam hal fase pengembangan inti. Ketiganya mencakup perencanaan, desain, pembangunan, pengujian, penerapan, dan pemeliharaan/dukungan pasca-penerapan. Perbedaan utamanya terletak pada bagaimana fase-fase ini diorganisasikan dan diadaptasi dalam siklus perangkat lunak.

## Jenis-jenis Model SDLC

Tim pengembang perangkat lunak dapat memilih dari berbagai kerangka kerja pengembangan terstruktur untuk membangun aplikasi secara efisien. Empat model yang paling banyak diadopsi meliputi:

### 1. Air Terjun: Linier & Berurutan

Metodologi waterfall menjalankan proses pengiriman perangkat lunak dalam fase-fase yang teratur dan berurutan tanpa tumpang tindih. Tahapannya meliputi pengumpulan persyaratan, desain sistem, pengkodean, pengujian, rilis, dan pemeliharaan. Jalur yang terdefinisi ini membawa disiplin tetapi membatasi fleksibilitas dalam perubahan kebutuhan pasca-peluncuran.

### 2. Agile: Iteratif & Adaptif

Agile menggunakan siklus kerja pendek dan berulang yang disebut "sprint" untuk mengembangkan fitur secara bertahap. Persyaratan disempurnakan secara progresif, bukan ditentukan sebelumnya. Umpan balik dan penyesuaian pengguna yang berkelanjutan menjaga tim tetap selaras dengan kebutuhan yang berubah. Kemampuan beradaptasi ini memprioritaskan respons terhadap tuntutan pasar yang baru.

### 3. Model Iteratif:

Model Iteratif adalah model di mana proses pengembangan dibagi menjadi beberapa siklus atau iterasi. Pada setiap iterasi, persyaratan dianalisis, dirancang, dan dikodekan. Umpan balik dari iterasi sebelumnya digunakan untuk memodifikasi proyek.

Model ini digunakan ketika persyaratan produk akhir tidak dipahami dengan baik atau diperkirakan akan berubah seiring waktu.

Model Iteratif adalah proses pengembangan perangkat lunak siklik yang mengulang siklus pengembangan beberapa kali hingga hasil yang diinginkan tercapai. Ini berbeda dengan Model Waterfall yang merupakan pendekatan linier untuk pengembangan perangkat lunak dengan serangkaian langkah yang telah ditentukan.

Model Iteratif memiliki empat fase umum:

1. **Perencanaan:** Tim proyek mengembangkan rencana untuk proyek tersebut. Ini termasuk mendefinisikan tujuan dan ruang lingkup proyek, serta menentukan sumber daya dan jangka waktu.
2. **Desain:** Pada fase ini, tim proyek mendesain arsitektur dan antarmuka pengguna perangkat lunak.
3. **Pengembangan:** Tim proyek membangun perangkat lunak sesuai dengan spesifikasi yang telah ditentukan pada fase sebelumnya.
4. **Pengujian:** Tim proyek menguji perangkat lunak untuk memastikan bahwa perangkat lunak tersebut memenuhi persyaratan.

### 4. Spiral: Siklus dengan Analisis Risiko


**Sumber Daya Pemrograman Terbaik**

Apa itu Algoritma: Definisi, Jenis, Karakteristik

Apa itu Array? Panduan Lengkap Beserta Contoh

Apa itu BIOS (Basic Input/Output System)?

Apa itu Struktur Data?

Apa itu FastAPI? Fitur dan Manfaatnya

Apa itu Gradle? Panduan untuk Pemula

Apa itu Hash Table? – Penjelasan Komprehensif (Diperbarui 2026)


Tahapan inti pembangunan model spiral adalah perencanaan, desain, rekayasa, dan evaluasi pada setiap rotasi iteratif. Membuat versi kecil produk dan membiarkan pengguna memberikan umpan balik sejak dini membantu menemukan masalah. Analisis risiko berkelanjutan memastikan kita menangani potensi masalah sejak dini.

Semua model tersebut menerapkan prinsip-prinsip SDLC yang serupa ke dalam kerangka kerja yang terstruktur berbeda, dengan memprioritaskan keteraturan, disiplin, atau fleksibilitas. Memahami pertimbangan-pertimbangan ini membantu tim perangkat lunak memilih siklus pengembangan optimal yang selaras dengan kebutuhan proyek dan bisnis.

**Pelajari apa itu Model V dan bagaimana cara kerjanya melalui blog ini.**

## Bagaimana SDLC Menangani Keamanan?

Dalam iklim ancaman siber yang kompleks saat ini , organisasi harus menjadikan keamanan sebagai prioritas di seluruh siklus hidup perangkat lunak. DevSecOps telah menjadi metodologi utama untuk mengintegrasikan praktik terbaik keamanan ke dalam setiap fase SDLC.

DevSecOps menghilangkan pemisahan antara pengembang, insinyur keamanan , dan operator TI melalui alur kerja kolaboratif untuk menciptakan pengerasan sistem. Dengan menyatukan perspektif lintas fungsi ini selama fase desain, arsitektur dan kode yang lebih tangguh dapat dihasilkan.

Proses otomatis seperti integrasi berkelanjutan/penyebaran berkelanjutan (CI/CD) dalam DevSecOps membantu tim merilis pembaruan keamanan dengan cepat ketika masalah muncul setelah peluncuran produk. Memperbaiki masalah bukan lagi proses yang panjang ketika kode baru dapat disebarkan sesuai permintaan.

Aktivitas kunci yang perlu diintegrasikan dalam keamanan meliputi pemodelan ancaman selama perencanaan arsitektur, melakukan analisis statis/dinamis pada kode, menjalankan kasus uji yang berfokus pada keamanan, dan pemantauan komprehensif sistem produksi. Menerapkan DevSecOps pada akhirnya menciptakan aplikasi yang secara proaktif tahan terhadap risiko siber modern. Memprioritaskan keamanan sejak awal mengurangi kerentanan organisasi melalui SDLC yang aman.

## Membandingkan SDLC dan Manajemen Siklus Hidup Aplikasi

Meskipun terkadang digunakan secara bergantian, siklus hidup pengembangan perangkat lunak (SDLC) dan manajemen siklus hidup aplikasi (ALM) merujuk pada disiplin manajemen teknologi yang terkait namun berbeda.

SDLC secara khusus berfokus pada tahap pengembangan perangkat lunak yang telah dikodekan. Tahapan ini mencakup fase-fase kunci seperti perencanaan awal, pengumpulan persyaratan, pengembangan iteratif, pengujian, penerapan, dan pemeliharaan pasca-peluncuran.

ALM (Application Lifecycle Management) mengamati seluruh siklus hidup produksi suatu perangkat lunak, mulai dari saat perangkat lunak tersebut aktif digunakan hingga saat dihentikan penggunaannya. ALM mencakup seluruh proses, menghubungkan berbagai tahapan pengembangan untuk melacak perubahan dan peningkatan dari waktu ke waktu.

## Bagaimana Komputasi Awan Dapat Membantu Anda Memenuhi Persyaratan SDLC Anda?

Penggunaan komputasi awan memberikan manfaat yang mendukung pemenuhan kebutuhan SDLC (Software Development Life Cycle). Beberapa contoh penyedia layanan awan adalah Amazon Web Service (AWS) , Microsoft Azure , dan Google Cloud Platform (GCP) . Komputasi awan meningkatkan efektivitas SDLC dengan cara-cara berikut:

1. **Skalabilitas:** Platform cloud memungkinkan penyesuaian infrastruktur untuk memenuhi perubahan kebutuhan proyek selama siklus pengembangan perangkat lunak.

2. **Efektivitas Biaya:** Layanan cloud mengikuti pendekatan bayar sesuai penggunaan, sehingga mengurangi pengeluaran dan menghilangkan kebutuhan untuk berinvestasi pada perangkat keras fisik.

3. **Kolaborasi Global:** Alat berbasis cloud memfasilitasi kerja tim secara real-time di antara tim yang tersebar, sehingga meningkatkan produktivitas dan efektivitas.

4. **Mempercepat Waktu Peluncuran Produk:** Lingkungan cloud memberikan akses ke sumber daya saat dibutuhkan, mempercepat proses pengembangan dan peluncuran produk.

5. **Optimalisasi Sumber Daya:** Platform cloud menawarkan berbagai layanan yang dapat membantu menyederhanakan operasi, mengotomatiskan tugas , dan meningkatkan efisiensi dalam organisasi Anda.

6. **Keamanan dan Kepatuhan:** Penyedia layanan cloud memiliki protokol keamanan yang ketat . Pastikan bahwa ketentuan pencadangan data yang memadai telah dibuat untuk memastikan pencadangan data secara teratur dan meminimalkan waktu henti selama keadaan darurat.

7. **Pemulihan Bencana dan Kelangsungan Bisnis:** Fitur redundansi seperti pencadangan otomatis disertakan dalam platform cloud untuk memastikan integritas data serta meminimalkan waktu henti selama bencana.

## Kesimpulan

Pengembangan perangkat lunak adalah industri yang terus berkembang dan masa depannya akan sukses. Industri ini berorientasi ke masa depan karena kebutuhan akan perangkat lunak terus meningkat dan teknologi baru terus bermunculan.

Untuk tetap kompetitif, pengembang perangkat lunak harus mengikuti tren dan teknologi terbaru. Terlebih lagi, metodologi pengembangan perangkat lunak yang tangkas akan terus diadopsi lebih sering karena menekankan kolaborasi, kecepatan, dan lain sebagainya.

### Tentang Penulis

Anisha Verma
Pemimpin Konten Teknis | Pengembang Perangkat Lunak

Anisha adalah seorang Pengembang Perangkat Lunak dan Pemimpin Konten Teknis berpengalaman dengan lebih dari 6,5 tahun keahlian dalam Pengembangan Full Stack. Ia unggul dalam menyusun konten yang jelas, akurat, dan menarik yang menerjemahkan konsep teknis yang kompleks menjadi wawasan praktis. Dengan minat yang kuat pada teknologi dan

pendidikan, Anisha menulis tentang berbagai topik TI, memberdayakan pelajar dan profesional untuk tetap unggul di lanskap digital yang berkembang pesat saat ini.

## Video yang Direkomendasikan


Bahasa Pemrograman Terbaik untuk Dipelajari


Tutorial Java untuk Pemula

## Program yang Direkomendasikan


Kursus Rekayasa Pengembangan Perangkat Lunak

5 **(23421)**


Rekayasa Perangkat Lunak dan Pengembangan Aplikasi

5 **(6315)**


Program Akselerator di bidang Rekayasa Perangkat Lunak

5 **(6058)**



**Alamat:** Lantai 6, Menara Primeco, Persimpangan Gerbang Arekere, Jalan Utama Bannerghatta, Bengaluru, Karnataka 560076, India.