

# MODUL PEMBELAJARAN

## SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

**Program Keahlian:** Pengembangan Perangkat Lunak dan Gim (PPLG)

**Kelas:** X

**Materi:** Software Development Life Cycle (SDLC)

---

### A. Tujuan Pembelajaran

Setelah mempelajari materi ini, peserta didik diharapkan mampu:

1. Menjelaskan pengertian SDLC.
  2. Menjelaskan tujuan dan manfaat SDLC.
  3. Menjelaskan tahapan umum pengembangan perangkat lunak.
  4. Memahami berbagai metode SDLC.
  5. Menjelaskan karakteristik masing-masing metode SDLC.
  6. Mengetahui kelebihan dan kekurangan setiap metode.
  7. Membandingkan metode SDLC berdasarkan kebutuhan proyek.
  8. Menentukan metode SDLC yang sesuai dengan suatu proyek.
  9. Menjelaskan perbedaan metode tradisional dan metode Agile.
  10. Menerapkan konsep SDLC sederhana dalam sebuah proyek perangkat lunak.
- 

### B. Apa Itu SDLC?

**SDLC (Software Development Life Cycle)** adalah sebuah proses atau siklus yang digunakan untuk mengembangkan perangkat lunak secara sistematis, mulai dari tahap perencanaan sampai perangkat lunak digunakan dan dipelihara.

Secara sederhana:

**SDLC adalah tahapan yang dilakukan untuk membuat sebuah perangkat lunak dari ide sampai perangkat lunak tersebut digunakan dan dipelihara.**

Contohnya, sekolah ingin membuat aplikasi **Sistem Informasi Perpustakaan**.

Sebelum programmer langsung membuat program, perlu dilakukan beberapa kegiatan:

1. Menentukan kebutuhan aplikasi.
2. Menganalisis masalah.

3. Membuat desain aplikasi.
4. Membuat program.
5. Melakukan pengujian.
6. Menginstal aplikasi.
7. Melakukan pemeliharaan.

Keseluruhan proses tersebut merupakan bagian dari SDLC.

---

## C. Mengapa SDLC Diperlukan?

Membuat perangkat lunak tidak hanya berarti menulis kode program.

Jika programmer langsung membuat program tanpa perencanaan, kemungkinan yang terjadi adalah:

- kebutuhan pengguna tidak terpenuhi;
- program sering mengalami perubahan;
- biaya pengembangan membengkak;
- waktu pengerjaan menjadi lebih lama;
- banyak kesalahan;
- program sulit dipelihara;
- pengguna tidak puas dengan hasil aplikasi.

Dengan SDLC, proses pengembangan menjadi lebih terstruktur.

### Manfaat SDLC

| No | Manfaat               | Penjelasan                              |
|----|-----------------------|-----------------------------------------|
| 1  | Perencanaan           | Proyek direncanakan sebelum dikerjakan  |
| 2  | Terstruktur           | Pekerjaan memiliki tahapan yang jelas   |
| 3  | Mengurangi kesalahan  | Masalah dapat ditemukan lebih awal      |
| 4  | Menghemat biaya       | Penggunaan sumber daya lebih terkontrol |
| 5  | Menghemat waktu       | Pekerjaan memiliki target yang jelas    |
| 6  | Meningkatkan kualitas | Software diuji sebelum digunakan        |
| 7  | Dokumentasi           | Setiap proses dapat didokumentasikan    |
| 8  | Pemeliharaan          | Software lebih mudah dikembangkan       |

---

## D. Tahapan Umum SDLC

Walaupun terdapat banyak metode SDLC, secara umum pengembangan perangkat lunak memiliki aktivitas berikut:

**Planning → Analysis → Design → Development → Testing → Deployment → Maintenance**

### 1. Planning — Perencanaan

Pada tahap ini ditentukan:

- masalah yang ingin diselesaikan;
- tujuan aplikasi;
- pengguna aplikasi;
- sumber daya;
- waktu pengerjaan;
- biaya;
- risiko proyek.

#### Contoh

Sekolah mengalami kesulitan dalam mengelola data perpustakaan.

Solusi yang direncanakan:

Membuat aplikasi perpustakaan berbasis web.

---

### 2. Analysis — Analisis

Pada tahap analisis, tim mencari tahu kebutuhan pengguna.

Pertanyaan yang dapat diajukan:

- Siapa yang akan menggunakan aplikasi?
- Apa saja fitur yang diperlukan?
- Data apa yang harus disimpan?
- Bagaimana proses bisnisnya?
- Apa masalah pada sistem lama?

#### Contoh kebutuhan aplikasi perpustakaan

Admin membutuhkan:

- login;

- data buku;
  - data anggota;
  - transaksi peminjaman;
  - transaksi pengembalian;
  - laporan.
- 

### 3. Design — Perancangan

Setelah kebutuhan diketahui, dibuat rancangan aplikasi.

Rancangan dapat berupa:

- desain tampilan;
- database;
- ERD;
- UML;
- flowchart;
- struktur navigasi;
- arsitektur aplikasi.

Contoh:

```
User
|
v
Login
|
v
Dashboard
|
+---- Data Buku
|
+---- Data Anggota
|
+---- Peminjaman
|
+---- Pengembalian
|
+---- Laporan
```

---

## 4. Development / Implementation — Pengembangan

Pada tahap ini programmer mulai membuat aplikasi berdasarkan desain.

Aktivitas dapat meliputi:

- membuat database;
- membuat frontend;
- membuat backend;
- membuat API;
- menulis kode;
- mengintegrasikan sistem.

Contoh teknologi:

- HTML
- CSS
- JavaScript
- PHP
- Laravel
- MySQL
- SQLite
- Python
- Java
- C#
- dan lain-lain.

---

## 5. Testing — Pengujian

Aplikasi yang sudah dibuat harus diuji.

Tujuannya untuk menemukan kesalahan atau **bug**.

Contoh pengujian:

| Pengujian                   | Hasil yang Diharapkan  |
|-----------------------------|------------------------|
| Login dengan password benar | Berhasil masuk         |
| Login password salah        | Muncul pesan kesalahan |
| Menambah buku               | Data tersimpan         |

| Pengujian          | Hasil yang Diharapkan |
|--------------------|-----------------------|
| Menghapus buku     | Data terhapus         |
| Meminjam buku      | Transaksi tersimpan   |
| Mengembalikan buku | Status berubah        |

---

## 6. Deployment — Implementasi

Setelah aplikasi selesai dan lolos pengujian, aplikasi mulai digunakan.

Contohnya:

- aplikasi dipasang pada server;
- domain dikonfigurasi;
- database dipindahkan;
- pengguna diberikan akses;
- dilakukan pelatihan pengguna.

---

## 7. Maintenance — Pemeliharaan

Setelah aplikasi digunakan, pekerjaan belum selesai.

Software perlu dipelihara.

Pemeliharaan dapat berupa:

- memperbaiki bug;
- meningkatkan keamanan;
- menambah fitur;
- meningkatkan performa;
- menyesuaikan dengan kebutuhan baru.

---

## E. 15 METODE SDLC

Terdapat berbagai pendekatan yang dapat digunakan untuk mengembangkan perangkat lunak.

Dalam materi ini kita akan mempelajari **15 metode SDLC**, yaitu:

1. Waterfall

2. V-Model
  3. Iterative Model
  4. Incremental Model
  5. Prototyping Model
  6. Spiral Model
  7. RAD (Rapid Application Development)
  8. Agile Model
  9. Scrum
  10. Extreme Programming (XP)
  11. Kanban
  12. DevOps
  13. Big Bang Model
  14. Fountain Model
  15. RUP (Rational Unified Process)
- 

# 1. WATERFALL MODEL

## Pengertian

**Waterfall** adalah metode pengembangan perangkat lunak yang menggunakan tahapan secara berurutan.

Setelah satu tahap selesai, proses dilanjutkan ke tahap berikutnya.

## Alur

```
graph TD;
    Planning --> Analysis;
    Analysis --> Design;
    Design --> Development;
    Development --> Testing;
    Testing --> Deployment;
    Deployment --> Maintenance;
```

## Karakteristik

- Berurutan.
- Setiap tahap memiliki dokumentasi.

- Perubahan setelah tahap selesai relatif sulit.
- Kebutuhan sebaiknya sudah jelas sejak awal.

### Kelebihan

- Mudah dipahami.
- Tahapannya jelas.
- Dokumentasi lengkap.
- Cocok untuk proyek dengan kebutuhan stabil.

### Kekurangan

- Sulit menghadapi perubahan kebutuhan.
- Pengguna baru melihat hasil pada tahap akhir.
- Kesalahan pada tahap awal dapat berdampak ke tahap berikutnya.

### Cocok digunakan untuk

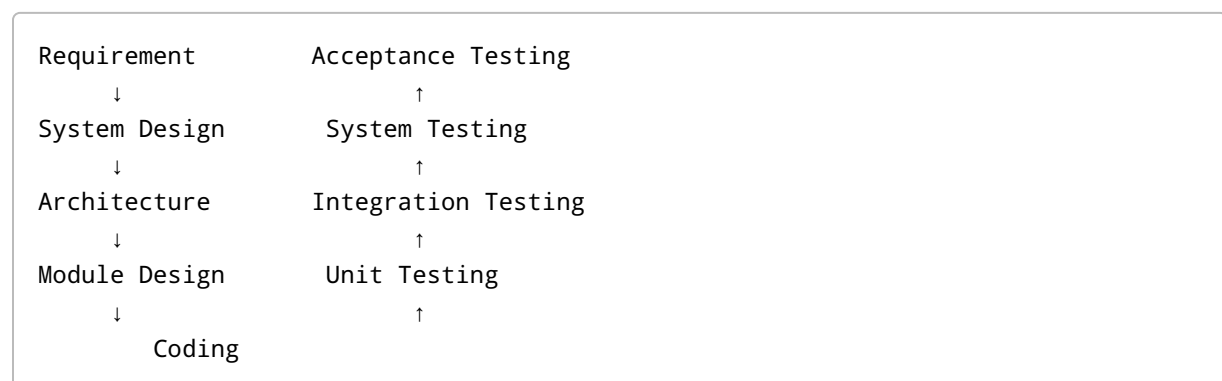
- Sistem dengan kebutuhan yang jelas.
- Proyek pemerintahan tertentu.
- Sistem yang memiliki spesifikasi stabil.

## 2. V-MODEL

### Pengertian

V-Model merupakan pengembangan dari Waterfall yang menekankan hubungan antara proses pengembangan dan proses pengujian.

Disebut V-Model karena bentuk alurnya menyerupai huruf **V**.



### Kelebihan

- Pengujian direncanakan sejak awal.



- Kualitas software lebih diperhatikan.
- Tahapan jelas.

### Kekurangan

- Kurang fleksibel terhadap perubahan.
- Membutuhkan perencanaan yang matang.

### Cocok untuk

Proyek yang membutuhkan proses pengujian ketat.

---

## 3. ITERATIVE MODEL

### Pengertian

Iterative Model mengembangkan perangkat lunak secara berulang.

Software dibuat dalam beberapa **iterasi**.

```
Analisis
↓
Desain
↓
Coding
↓
Testing
↓
Evaluasi
↓
Iterasi berikutnya
```

Pada setiap iterasi, software diperbaiki.

### Contoh

Versi 1:

Login

Versi 2:

Login + Data Buku

Versi 3:

Login + Data Buku + Peminjaman

Versi 4:

Login + Data Buku + Peminjaman + Laporan

### Kelebihan

- Perbaikan dapat dilakukan secara bertahap.
- Masalah dapat ditemukan lebih awal.
- Pengguna dapat memberikan masukan.

### Kekurangan

- Membutuhkan pengelolaan iterasi yang baik.
  - Dapat terjadi perubahan yang terus-menerus.
- 

## 4. INCREMENTAL MODEL

### Pengertian

Incremental Model mengembangkan software menjadi beberapa bagian atau **increment**.

Setiap increment menghasilkan bagian software yang dapat digunakan.

### Contoh aplikasi sekolah

#### Increment 1

Login

#### Increment 2

Login  
+  
Data Siswa

#### Increment 3

Login  
+  
Data Siswa  
+  
Data Guru

#### Increment 4

Login  
+  
Data Siswa  
+  
Data Guru  
+  
Nilai

#### Kelebihan

- Produk dapat digunakan lebih cepat.
- Pengembangan bertahap.
- Risiko dapat dibagi.

#### Kekurangan

- Perencanaan arsitektur harus baik.
- Integrasi antar-increment dapat menjadi kompleks.

---

## 5. PROTOTYPING MODEL

### Pengertian

Prototyping digunakan untuk membuat model awal aplikasi sebelum aplikasi final dibuat.

Tujuan utamanya adalah mendapatkan gambaran kebutuhan pengguna.

#### Alur

Kebutuhan Awal  
↓  
Prototype  
↓  
Evaluasi Pengguna

↓  
Perbaiki  
↓  
Prototype Baru  
↓  
Software Final

## Contoh

Sekolah ingin membuat aplikasi pembayaran.

Developer membuat prototype:

- halaman login;
- dashboard;
- halaman pembayaran;
- laporan.

Kemudian prototype diperlihatkan kepada bendahara.

Bendahara memberikan masukan:

"Tambahkan filter berdasarkan kelas."

Prototype diperbaiki.

## Kelebihan

- Pengguna dapat melihat gambaran aplikasi.
- Kesalahpahaman kebutuhan dapat dikurangi.
- Cocok untuk kebutuhan yang belum jelas.

## Kekurangan

- Pengguna dapat mengira prototype sudah menjadi aplikasi final.
- Prototype yang terlalu sering berubah dapat menghabiskan waktu.

---

# 6. SPIRAL MODEL

## Pengertian

Spiral Model menggabungkan konsep iterasi dengan **manajemen risiko**.

Setiap putaran spiral melakukan:

1. Perencanaan.
2. Analisis risiko.
3. Pengembangan.
4. Evaluasi.



### Fokus utama

**Risk Management**

### Kelebihan

- Risiko diperhatikan sejak awal.
- Cocok untuk proyek besar.
- Fleksibel terhadap perubahan.

### Kekurangan

- Lebih kompleks.
- Biaya pengembangan dapat tinggi.
- Membutuhkan tenaga ahli.

---

## 7. RAD — RAPID APPLICATION DEVELOPMENT

### Pengertian

RAD merupakan metode yang berfokus pada **pengembangan aplikasi secara cepat**.

RAD banyak menggunakan:

- prototype;
- reusable components;
- feedback pengguna;

- pengembangan cepat.

### Tahapan umum

```
Requirement Planning
    ↓
User Design
    ↓
Construction
    ↓
Cutover
```

### Kelebihan

- Pengembangan cepat.
- Pengguna banyak dilibatkan.
- Prototype dapat dibuat dengan cepat.

### Kekurangan

- Membutuhkan tim yang kompeten.
- Tidak cocok untuk semua jenis proyek.
- Membutuhkan keterlibatan pengguna.

---

## 8. AGILE MODEL

### Pengertian

Agile adalah pendekatan pengembangan perangkat lunak yang menekankan:

- fleksibilitas;
- kolaborasi;
- feedback pengguna;
- pengembangan bertahap;
- respons terhadap perubahan.

Prinsip sederhananya:

**Membuat software sedikit demi sedikit, mendapatkan feedback, kemudian memperbaikinya.**

### Contoh

Daripada membuat seluruh aplikasi selama 6 bulan lalu diberikan kepada pengguna, tim membuat:

Sprint 1 → Login  
Sprint 2 → Produk  
Sprint 3 → Transaksi  
Sprint 4 → Laporan

Setiap bagian mendapatkan evaluasi.

### **Kelebihan**

- Fleksibel.
- Cepat menerima perubahan.
- Pengguna terlibat.
- Feedback diperoleh lebih cepat.

### **Kekurangan**

- Membutuhkan komunikasi yang baik.
- Dokumentasi dapat lebih ringan dibanding metode tradisional.
- Kurang cocok jika tim tidak disiplin.

---

## **9. SCRUM**

### **Pengertian**

Scrum adalah framework Agile yang menggunakan pekerjaan dalam periode tertentu yang disebut **Sprint**.

Biasanya satu Sprint berlangsung beberapa minggu.

### **Elemen Scrum**

#### **Product Owner**

Bertanggung jawab terhadap kebutuhan dan prioritas produk.

#### **Scrum Master**

Membantu tim menerapkan Scrum dan mengatasi hambatan proses.

#### **Developers**

Mengerjakan pengembangan produk.

## Istilah penting

- Product Backlog
- Sprint Backlog
- Sprint
- Daily Scrum
- Sprint Review
- Sprint Retrospective
- Increment

## Contoh

Product Backlog:

```
[ ] Login
[ ] Registrasi
[ ] Data siswa
[ ] Data guru
[ ] Input nilai
[ ] Cetak laporan
```

Sprint 1:

```
[✓] Login
[✓] Registrasi
```

Sprint 2:

```
[✓] Data siswa
[✓] Data guru
```

---

## 10. EXTREME PROGRAMMING (XP)

### Pengertian

Extreme Programming atau **XP** merupakan metode Agile yang sangat menekankan kualitas kode dan feedback yang cepat.

### Praktik XP

- Pair Programming



- Test-Driven Development
- Continuous Integration
- Refactoring
- Simple Design
- Small Releases
- Customer Feedback

## Pair Programming

Dua programmer bekerja bersama:

Programmer A → menulis kode  
 Programmer B → melakukan review

Kemudian mereka bertukar peran.

### Kelebihan

- Kualitas kode dapat meningkat.
- Bug dapat ditemukan lebih cepat.
- Feedback cepat.

### Kekurangan

- Membutuhkan komunikasi tinggi.
- Pair programming membutuhkan sumber daya dua programmer.
- Tidak semua tim cocok dengan cara kerja ini.

# 11. KANBAN

## Pengertian

Kanban adalah metode visualisasi pekerjaan.

Pekerjaan ditampilkan dalam papan.

Contoh:

| TO DO | DOING    | DONE |
|-------|----------|------|
| ----- |          |      |
| Login | Database | ERD  |

Dashboard      API      UI Login  
Laporan

Ketika pekerjaan dimulai:

TO DO → DOING

Setelah selesai:

DOING → DONE

### Kelebihan

- Sederhana.
- Visual.
- Mudah mengetahui pekerjaan yang sedang berjalan.
- Fleksibel.

### Kekurangan

- Tidak memiliki struktur Sprint seperti Scrum.
- Membutuhkan disiplin dalam memperbarui status pekerjaan.

---

## 12. DEVOPS

### Pengertian

DevOps menggabungkan aktivitas **Development** dan **Operations**.

Tujuannya agar proses pengembangan sampai deployment menjadi lebih cepat dan otomatis.

### Alur sederhana

```
Plan
↓
Code
↓
Build
↓
Test
↓
```

```
Release
↓
Deploy
↓
Operate
↓
Monitor
↓
Feedback
```

DevOps banyak menggunakan:

- Version Control;
- CI/CD;
- Automated Testing;
- Automated Deployment;
- Monitoring.

## Contoh

Programmer melakukan:

```
git push
```

Kemudian sistem otomatis:

```
Build
↓
Testing
↓
Deploy
```

## Kelebihan

- Deployment lebih cepat.
- Otomatisasi.
- Kolaborasi Development dan Operations.
- Feedback lebih cepat.

## Kekurangan

- Membutuhkan tools dan keahlian.
- Infrastruktur lebih kompleks.

## 13. BIG BANG MODEL

### Pengertian

Big Bang Model merupakan pendekatan yang relatif minim perencanaan formal.

Developer langsung mengembangkan software berdasarkan ide atau kebutuhan yang tersedia.

### Gambaran

```
graph TD; A[Ide] --> B[Coding]; B --> C[Testing]; C --> D[Software];
```

Ide  
↓  
Coding  
↓  
Testing  
↓  
Software

### Kelebihan

- Sederhana.
- Tidak membutuhkan banyak dokumentasi.
- Cocok untuk eksperimen kecil.

### Kekurangan

- Risiko tinggi.
- Sulit memperkirakan waktu dan biaya.
- Sulit digunakan untuk proyek besar.
- Hasil dapat tidak sesuai kebutuhan.

### Cocok untuk

- Eksperimen.
  - Proyek kecil.
  - Pembelajaran.
  - Proof of Concept.
-

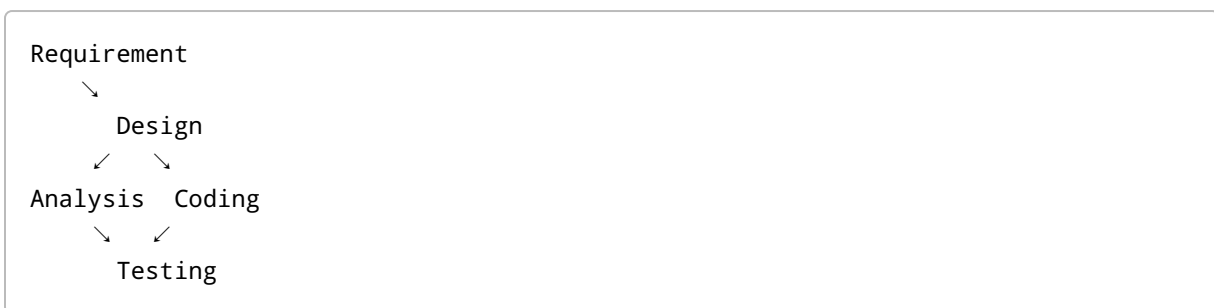
## 14. FOUNTAIN MODEL

### Pengertian

Fountain Model merupakan model pengembangan yang memungkinkan aktivitas pengembangan saling tumpang tindih.

Berbeda dengan Waterfall yang cenderung berurutan, pada Fountain Model beberapa aktivitas dapat berjalan secara bersamaan atau kembali ke aktivitas sebelumnya.

Contoh:



### Kelebihan

- Lebih fleksibel dibanding model berurutan.
- Memungkinkan aktivitas saling berhubungan.
- Dapat mengakomodasi perubahan.

### Kekurangan

- Lebih sulit dikelola.
- Membutuhkan koordinasi yang baik.
- Dokumentasi dan pengendalian proses harus baik.

---

## 15. RUP — RATIONAL UNIFIED PROCESS

### Pengertian

RUP adalah proses pengembangan perangkat lunak yang menggunakan pendekatan iteratif dan berorientasi pada kebutuhan serta arsitektur sistem.

RUP memiliki empat fase utama:

## 1. Inception

Menentukan:

- tujuan proyek;
- ruang lingkup;
- kebutuhan awal;
- kelayakan.

## 2. Elaboration

Memperjelas:

- kebutuhan;
- arsitektur;
- risiko.

## 3. Construction

Tahap pembangunan software.

## 4. Transition

Software diberikan kepada pengguna.

```
graph TD;
    A[Inception] --> B[Elaboration];
    B --> C[Construction];
    C --> D[Transition];
```

### Kelebihan

- Struktur proses jelas.
- Manajemen risiko diperhatikan.
- Cocok untuk proyek kompleks.

### Kekurangan

- Cukup kompleks.
- Membutuhkan dokumentasi.
- Membutuhkan tim yang memahami proses RUP.

## F. PERBANDINGAN 15 METODE SDLC

| No | Metode      | Fokus Utama                   | Fleksibilitas |
|----|-------------|-------------------------------|---------------|
| 1  | Waterfall   | Tahapan berurutan             | Rendah        |
| 2  | V-Model     | Pengembangan + testing        | Rendah        |
| 3  | Iterative   | Pengulangan                   | Sedang        |
| 4  | Incremental | Pengembangan bertahap         | Sedang        |
| 5  | Prototyping | Validasi kebutuhan            | Tinggi        |
| 6  | Spiral      | Manajemen risiko              | Tinggi        |
| 7  | RAD         | Kecepatan                     | Tinggi        |
| 8  | Agile       | Adaptasi perubahan            | Tinggi        |
| 9  | Scrum       | Sprint dan kolaborasi         | Tinggi        |
| 10 | XP          | Kualitas kode                 | Tinggi        |
| 11 | Kanban      | Visualisasi pekerjaan         | Tinggi        |
| 12 | DevOps      | Otomasi dan delivery          | Tinggi        |
| 13 | Big Bang    | Eksperimen cepat              | Sangat tinggi |
| 14 | Fountain    | Aktivitas yang tumpang tindih | Tinggi        |
| 15 | RUP         | Iterasi dan arsitektur        | Sedang-Tinggi |

## G. PERBEDAAN WATERFALL DAN AGILE

Ini merupakan salah satu perbedaan paling penting yang harus dipahami siswa.

| Waterfall                            | Agile                                     |
|--------------------------------------|-------------------------------------------|
| Berurutan                            | Iteratif                                  |
| Perubahan sulit                      | Perubahan relatif mudah                   |
| Perencanaan banyak dilakukan di awal | Perencanaan dilakukan secara bertahap     |
| Feedback cenderung lebih akhir       | Feedback dilakukan terus-menerus          |
| Dokumentasi biasanya lebih formal    | Dokumentasi disesuaikan kebutuhan         |
| Cocok untuk kebutuhan stabil         | Cocok untuk kebutuhan yang sering berubah |

## Ilustrasi

## Waterfall

Plan → Analyze → Design → Code → Test → Deploy

## Agile

Plan → Code → Test → Feedback

↑                      ↓  
← — Perbaikan

## H. CONTOH KASUS

## Kasus 1 — Aplikasi Perpustakaan Sekolah

Sekolah ingin membuat aplikasi perpustakaan.

Kebutuhan sudah sangat jelas:

- data buku;
- data anggota;
- peminjaman;
- pengembalian;
- laporan.

Kebutuhan tidak diperkirakan banyak berubah.

### Metode yang dapat dipilih

## Waterfall

Alasannya:

- kebutuhan jelas;
- sistem relatif stabil;
- tahapan dapat direncanakan sejak awal.



## Kasus 2 — Aplikasi Marketplace

Sebuah perusahaan ingin membuat marketplace.

Namun kebutuhan pengguna dapat berubah berdasarkan hasil uji coba.

### Metode yang sesuai

#### Agile/Scrum

Karena:

- kebutuhan dapat berubah;
  - feedback pengguna penting;
  - fitur dapat dibuat bertahap.
- 

## Kasus 3 — Sistem yang Memiliki Risiko Tinggi

Sebuah perusahaan ingin membuat sistem yang kompleks dan memiliki risiko tinggi.

### Metode yang sesuai

#### Spiral

Karena metode ini sangat memperhatikan identifikasi dan pengelolaan risiko.

---

## Kasus 4 — Pengguna Belum Mengetahui Aplikasi yang Diinginkan

Sebuah sekolah ingin membuat aplikasi baru, tetapi pengguna belum dapat menjelaskan secara detail bentuk aplikasi yang mereka inginkan.

### Metode yang sesuai

#### Prototyping

Developer dapat membuat prototype terlebih dahulu sehingga pengguna dapat memberikan masukan.

---

# I. BAGAIMANA MEMILIH METODE SDLC?

Tidak ada satu metode SDLC yang selalu paling baik.

Pemilihan metode bergantung pada kondisi proyek.

Pertimbangkan:

## 1. Apakah kebutuhan sudah jelas?

Jika **ya**, Waterfall dapat dipertimbangkan.

Jika **belum**, Prototyping atau Agile dapat dipertimbangkan.

## 2. Apakah kebutuhan sering berubah?

Jika **ya**, Agile dapat menjadi pilihan.

## 3. Apakah proyek memiliki risiko tinggi?

Jika **ya**, Spiral dapat dipertimbangkan.

## 4. Apakah software harus dibuat dengan cepat?

Jika **ya**, RAD dapat dipertimbangkan.

## 5. Apakah pekerjaan perlu divisualisasikan?

Kanban dapat digunakan.

## 6. Apakah deployment harus cepat dan otomatis?

DevOps dapat digunakan bersama pendekatan pengembangan lainnya.

---

# J. SDLC DALAM KEHIDUPAN SEHARI-HARI

Konsep SDLC sebenarnya dapat ditemukan dalam kehidupan sehari-hari.

Misalnya kita ingin membuat **aplikasi kantin sekolah**.

## 1. Planning

Masalah:

Antrean kantin terlalu panjang.

Solusi:

Membuat aplikasi pemesanan makanan.

## 2. Analysis

Kebutuhan:

- daftar makanan;
- pemesanan;
- pembayaran;
- status pesanan.

## 3. Design

Membuat:

- desain halaman;
- database;
- alur sistem.

## 4. Development

Programmer membuat aplikasi.

## 5. Testing

Menguji:

Apakah siswa dapat melakukan pemesanan?

## 6. Deployment

Aplikasi mulai digunakan.

## 7. Maintenance

Jika terjadi bug:

Bug diperbaiki.

Jika diperlukan fitur baru:

Fitur ditambahkan.

---

## K. ISTILAH PENTING

Peserta didik perlu memahami beberapa istilah berikut.

| Istilah     | Pengertian                                                        |
|-------------|-------------------------------------------------------------------|
| SDLC        | Siklus hidup pengembangan perangkat lunak                         |
| Requirement | Kebutuhan sistem                                                  |
| Developer   | Orang yang mengembangkan software                                 |
| Programmer  | Orang yang menulis program                                        |
| User        | Pengguna software                                                 |
| Stakeholder | Pihak yang berkepentingan                                         |
| Prototype   | Model awal aplikasi                                               |
| Iteration   | Pengulangan proses pengembangan                                   |
| Increment   | Bagian software yang dikembangkan                                 |
| Bug         | Kesalahan pada software                                           |
| Testing     | Pengujian software                                                |
| Deployment  | Proses membuat software siap digunakan                            |
| Maintenance | Pemeliharaan software                                             |
| Sprint      | Periode kerja dalam Scrum                                         |
| Backlog     | Daftar pekerjaan/kebutuhan                                        |
| CI/CD       | Praktik otomatisasi integrasi, pengujian, dan delivery/deployment |

---

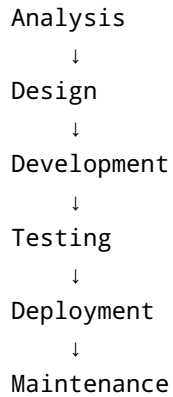
## L. RANGKUMAN

SDLC merupakan proses yang digunakan untuk mengembangkan perangkat lunak secara sistematis.

Tahapan umum SDLC meliputi:

Planning

↓



Terdapat berbagai metode SDLC.

15 metode yang dipelajari:

1. Waterfall
2. V-Model
3. Iterative
4. Incremental
5. Prototyping
6. Spiral
7. RAD
8. Agile
9. Scrum
10. Extreme Programming
11. Kanban
12. DevOps
13. Big Bang
14. Fountain
15. RUP

Setiap metode memiliki tujuan, karakteristik, kelebihan, dan kekurangan yang berbeda.

**Tidak ada metode yang selalu paling baik.**

Metode harus dipilih berdasarkan:

- kebutuhan proyek;
- ukuran proyek;
- tingkat risiko;
- kemampuan tim;
- waktu;
- biaya;
- tingkat perubahan kebutuhan;
- keterlibatan pengguna.

# M. LATIHAN PEMAHAMAN

## Pilihan Ganda

### 1. Apa yang dimaksud dengan SDLC?

- A. Bahasa pemrograman
- B. Siklus pengembangan perangkat lunak
- C. Sistem operasi
- D. Database
- E. Jaringan komputer

**Jawaban: B**

---

### 2. Tahap SDLC yang digunakan untuk mengetahui kebutuhan pengguna adalah...

- A. Testing
- B. Deployment
- C. Analysis
- D. Maintenance
- E. Coding

**Jawaban: C**

---

### 3. Metode yang memiliki tahapan berurutan adalah...

- A. Scrum
- B. Kanban
- C. Waterfall
- D. XP
- E. DevOps

**Jawaban: C**

---

### 4. Metode yang sangat memperhatikan risiko adalah...

- A. Spiral
- B. Waterfall
- C. Big Bang
- D. Kanban
- E. XP

**Jawaban: A**

---

**5. Metode yang menggunakan prototype untuk memahami kebutuhan pengguna adalah...**

- A. V-Model
- B. Prototyping
- C. Waterfall
- D. Big Bang
- E. Fountain

**Jawaban: B**

---

**6. Dalam Scrum, periode pengerjaan disebut...**

- A. Increment
- B. Sprint
- C. Cycle
- D. Release
- E. Phase

**Jawaban: B**

---

**7. Kanban menggunakan pendekatan utama berupa...**

- A. Diagram ERD
- B. Papan visual pekerjaan
- C. Database
- D. Flowchart
- E. Source code

**Jawaban: B**

---

**8. Metode yang berfokus pada pengembangan software secara cepat adalah...**

- A. RAD
- B. Waterfall
- C. V-Model
- D. Fountain
- E. RUP

**Jawaban: A**

---

### 9. DevOps menggabungkan...

- A. Design dan Analysis
- B. Development dan Operations
- C. Database dan Network
- D. Testing dan Design
- E. User dan Developer

**Jawaban: B**

---

### 10. Extreme Programming biasa disingkat...

- A. EP
- B. XP
- C. EX
- D. XPR
- E. PM

**Jawaban: B**

---

## N. SOAL URAIAN

1. Jelaskan pengertian SDLC dengan bahasa kalian sendiri!
  2. Mengapa SDLC diperlukan dalam pengembangan perangkat lunak?
  3. Sebutkan tahapan umum SDLC!
  4. Jelaskan perbedaan Waterfall dan Agile!
  5. Apa yang dimaksud dengan prototype?
  6. Mengapa Spiral cocok untuk proyek dengan risiko tinggi?
  7. Jelaskan pengertian Incremental Model!
  8. Apa yang dimaksud dengan Sprint dalam Scrum?
  9. Jelaskan fungsi Kanban!
  10. Apa hubungan antara Development dan Operations dalam DevOps?
  11. Sebutkan empat fase utama RUP!
  12. Jelaskan kelebihan dan kekurangan Waterfall!
  13. Mengapa Agile cocok untuk kebutuhan yang sering berubah?
  14. Jelaskan perbedaan Iterative dan Incremental!
  15. Jika sekolah ingin membuat aplikasi yang kebutuhannya belum jelas, metode apa yang sebaiknya digunakan? Jelaskan alasannya.
-



## O. TUGAS KELOMPOK

### Studi Kasus: Sistem Informasi Sekolah

Buatlah rancangan sederhana sebuah aplikasi untuk sekolah.

Contoh:

#### **Sistem Informasi Akademik**

Aplikasi minimal memiliki:

- Login
- Data siswa
- Data guru
- Data kelas
- Data mata pelajaran
- Data nilai
- Laporan

Setiap kelompok harus menentukan **satu metode SDLC**.

#### **Tugas kelompok:**

1. Tentukan masalah yang ingin diselesaikan.
2. Tentukan nama aplikasi.
3. Tentukan pengguna aplikasi.
4. Tentukan minimal 5 kebutuhan/fungsi.
5. Pilih satu metode SDLC.
6. Jelaskan alasan pemilihan metode.
7. Buat diagram tahapan metode.
8. Buat rancangan sederhana aplikasi.
9. Presentasikan hasilnya di depan kelas.

---

## P. PROYEK PRAKTIK

### Mini Project SDLC

Peserta didik dibagi menjadi kelompok beranggotakan 3–5 orang.

Setiap kelompok membuat rancangan proyek perangkat lunak sederhana.

## Pilihan proyek

- Sistem Informasi Perpustakaan
- Sistem Informasi Kantin
- Sistem Absensi Siswa
- Sistem Peminjaman Laboratorium
- Sistem Pengelolaan Ekstrakurikuler
- Sistem Kas Kelas
- Sistem Pengaduan Siswa
- Sistem Booking Ruang
- Sistem Inventaris Sekolah
- Sistem Penjualan Sederhana

## Produk akhir

Setiap kelompok menghasilkan:

1. Judul proyek
2. Latar belakang
3. Rumusan masalah
4. Tujuan aplikasi
5. Analisis kebutuhan
6. Pemilihan metode SDLC
7. Tahapan SDLC
8. Flowchart
9. Use Case sederhana
10. Wireframe/interface
11. Kesimpulan

## Q. RUBRIK PENILAIAN

| Komponen           | Bobot |
|--------------------|-------|
| Pemahaman SDLC     | 20%   |
| Analisis masalah   | 15%   |
| Analisis kebutuhan | 20%   |
| Pemilihan metode   | 15%   |
| Rancangan aplikasi | 15%   |
| Presentasi         | 10%   |
| Kerja sama         | 5%    |

| Komponen     | Bobot       |
|--------------|-------------|
| <b>Total</b> | <b>100%</b> |

---

## R. KESIMPULAN AKHIR

SDLC merupakan salah satu konsep dasar yang sangat penting bagi siswa PPLG.

Seorang programmer tidak hanya dituntut untuk mampu membuat kode, tetapi juga harus memahami bagaimana sebuah software direncanakan, dianalisis, dirancang, dikembangkan, diuji, digunakan, dan dipelihara.

Dengan memahami 15 metode SDLC, siswa diharapkan tidak hanya mengetahui nama-nama metode, tetapi juga mampu menjawab pertanyaan:

**"Metode SDLC apa yang paling sesuai untuk sebuah proyek dan mengapa?"**

Kemampuan memilih metode berdasarkan kondisi proyek merupakan salah satu keterampilan penting dalam pengembangan perangkat lunak.