SHARE IT 2025

REKAYASA PERANGKAT LUNAK

Penalaran Intelektual

Himpunan Mahasiswa
Teknik Informatika
Universitas Teknologi Bandung

Apa yang Akan Kita Pelajari?

01. Pemahaman Dasar
Memahami konsep dasar dan pentingnya RPL

02. Software Development Life Cycle (SDLC)
Memahami fase-fase dalam SDLC

03. Model Pengembangan
Mengenal berbagai Model Proses Pengembangan RPL

04. Data Flow Diagram (DFD)
Pengenalan dan Fungsi Data Flow Diagram (DFD)

05. Pengujian & Perawatan
Memahami prinsip desain, pengujian, dan perawatan perangkat lunak

Apa Itu RPL?

Definisi (IEEE): Penerapan pendekatan yang sistematis, disiplin, dan terkuantifikasi terhadap pengembangan, penggunaan, dan pemeliharaan perangkat lunak.

Fokus:
Kualitas, Ketepatan Waktu , Kesesuaian Anggaran

Programmer
fokus pada kode

Software Engineer
fokus pada seluruh siklus hidup proyek

Mengapa RPL Itu Penting?

Problem tanpa RPL (Software crisis)

Proyek gagal deliver

Melebihi budget yang dialokasikan

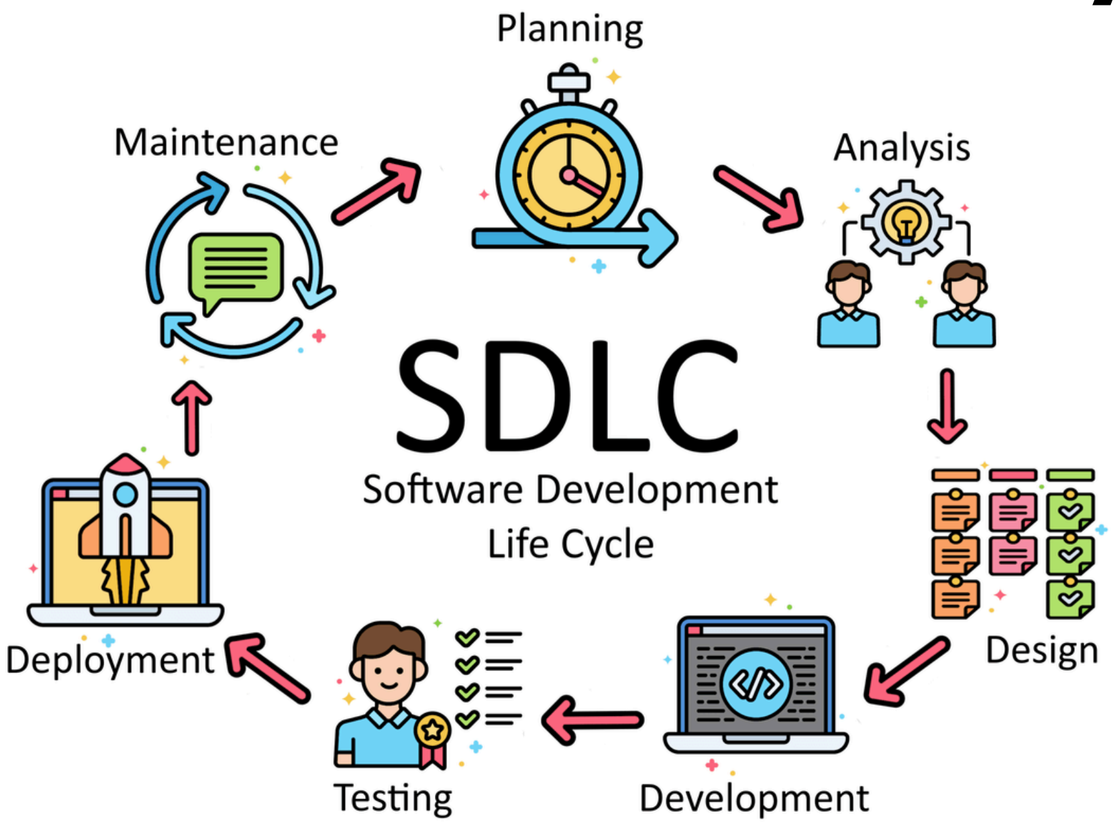
Tidak sesuai dengan kebutuhan user

Kualitas rendah dan penuh bug

Pendekatan terstruktur mengurangi risiko tersebut.

KONSEP INTI

Software Development Life Cycle (SDLC)



Serangkaian fase yang membentuk proses pengembangan suatu perangkat lunak dari lahir hingga "pensiun"

Planning -> Analysis -> Design -> Implementation -> Testing -> Deployment -> Maintenance.

RPL adalah tentang mengelola seluruh siklus ini dengan baik.

Software Development Method

Waterfall

Metode pengembangan perangkat lunak berurutan (tahap demi tahap).

Tahapan: Analisis → Desain → Implementasi → Testing → Deployment → Maintenance.

Cocok untuk proyek dengan kebutuhan yang jelas dan stabil.

Agile

Metode pengembangan perangkat lunak yang fleksibel dan iteratif.

Fokus pada kolaborasi tim, perubahan cepat, dan hasil bertahap.

Cocok untuk proyek dengan kebutuhan yang sering berubah.

Scrum

Salah satu kerangka kerja Agile.

Menggunakan Sprint (periode waktu singkat, biasanya 2-4 minggu) untuk menghasilkan fitur.

Ada peran: Product Owner, Scrum Master, Development Team.

Rekayasa Kebutuhan (Requirement Engineering)

Proses menemukan, menganalisis, mendokumentasikan, dan memvalidasi kebutuhan sistem.

Fungsional: Apa yang sistem harus lakukan.
(Contoh: "Sistem harus bisa login")

Non-Fungsional: Bagaimana sistem beroperasi.
(Contoh: "Proses login harus <2 detik")

Software Requirements Specification (SRS)

Dokumen kontrak yang berisi deskripsi lengkap tentang perilaku sistem yang akan dikembangkan.

SRS adalah dasar dari segala desain dan pengujian.

Isi Penting / struktur dokumen SRS

1. **PENDAHULUAN** (Tujuan Penulisan Dokumen, Lingkup Masalah, Definisi, Istilah dan Singkatan, Referensi, Deskripsi Umum Dokumen)
2. **DESKRIPSI UMUM PERANGKAT LUNAK**(Deskripsi Umum Sistem, Fungsi Produk, Karakteristik Pengguna, Batasan, Lingkungan Operasi)
3. **DESKRIPSI UMUM KEBUTUHAN (**Kebutuhan antarmuka eksternal, Deskripsi Fngsional, Data Requirement, Non Functional Requiremen, Ringkasan Kebutuhan)
4. **IMPLEMENTASI DAN PENGUJIAN**(Pengujian Sistem)

Data Flow Diagram
DFD

DFD (Data Flow Diagram) adalah diagram yang menggambarkan alur perpindahan data dalam sebuah sistem, mulai dari input, proses pengolahan, hingga menjadi output.
"Blueprint" untuk perangkat lunak.

- Memvisualisasikan bagaimana data bergerak di dalam sistem.
- Mempermudah analisis kebutuhan dan perancangan sistem.
- Menjelaskan hubungan antara pengguna, proses, dan penyimpanan data.

Pengujian Perangkat Lunak (Testing)

Mencari Cacat Sebelum Digunakan User

Black Box Testing

- Menguji fungsi sistem dari luar tanpa melihat kode program.
- Fokus pada input dan output → apakah hasil sesuai yang diharapkan.
- Contoh: menguji form login dengan data benar/salah.

White Box Testing

- Menguji bagian dalam sistem dengan melihat kode program.
- Fokus pada alur logika, struktur, dan jalur eksekusi kode.
- Contoh: memastikan setiap percabangan if-else dijalankan dengan benar

Perawatan Perangkat Lunak (Maintenance)

Hidup Setelah Rilis

Fakta: Biaya maintenance bisa 60-70% dari total biaya siklus hidup.

Jenis Maintenance:

- Corrective: Perbaikan bug.
- Adaptive: Penyesuaian dengan lingkungan baru.
- Perfective: Penambahan fitur baru.
- Preventive: Meningkatkan kode untuk masa depan (refactoring).

Kesimpulan

- RPL adalah disiplin engineering untuk membangun perangkat lunak yang berkualitas.
- Memilih model proses yang tepat sangat kritis untuk kesuksesan proyek.
- Rekayasa Kebutuhan adalah fase paling kritis; kesalahan di sini sangat mahal untuk diperbaiki.

Any Question?

Terima Kasih