RESUME CHAPTER 6-7 DPPL

Kelompok 13 :

- Gusti Panji Widodo (2407113145)
- Muhammad Nabil Nadif (2407112714)
- Qorri Aquina Adisty (2107111517)

RESUME
CHAPTER 6
PRINCIPLES THAT GUIDE PROCESS

- **Prinsip yang memandu proses**
    - **Be agile -** apa pun model proses yang digunakan, prinsip agile tetap menjadi dasar.
    - **Fokus pada kualitas di setiap tahap** – setiap aktivitas harus menghasilkan produk kerja yang berkualitas.
    - **Siap beradaptasi** – tidak kaku pada metode; sesuaikan dengan kebutuhan proyek, tim, dan masalah.
    - **Bangun tim yang efektif** – keberhasilan bergantung pada orang, bukan hanya proses.
    - **Komunikasi dan koordinasi yang jelas** – banyak proyek gagal karena miskomunikasi.
    - **Kelola perubahan** – siapkan mekanisme formal/informal untuk menangani perubahan.
    - **Lakukan penilaian risiko** – identifikasi hal yang berpotensi salah dan buat rencana cadangan.
    - **Hasilkan produk kerja yang bernilai** – jangan membuat dokumen atau artefak yang tidak berguna.

- **Prinsip yang memandu praktik**
    - **Divide and conquer –** gunakan *separation of concerns* agar masalah lebih mudah dipecah.
    - **Abstraksi –** sederhanakan sistem kompleks agar mudah dipahami.
    - **Konsistensi** – desain yang familiar lebih mudah digunakan.
    - **Fokus pada transfer informasi** – antarmuka harus dirancang dan diuji dengan cermat.
    - **Modularitas** – bangun perangkat lunak dalam modul terpisah.
    - **Gunakan pola (patterns)** – manfaatkan solusi untuk masalah berulang.
    - **Multiple viewpoints** – lihat masalah dari berbagai perspektif.
    - **Ingat pemeliharaan** – orang lain akan mengelola hasil kerja Anda.

- **Prinsip Komunikasi**
    - Dengarkan dengan fokus, jangan hanya menunggu giliran bicara.
    - Persiapkan diri sebelum berdiskusi.
    - Setiap pertemuan butuh fasilitator agar tetap produktif.
    - Komunikasi tatap muka lebih efektif.
    - Catat keputusan penting.
    - Upayakan kolaborasi, bukan dominasi.
    - Tetap fokus, hindari diskusi melompat-lompat.
    - Jika sulit dipahami, gunakan gambar/diagram.
    - Jika sudah sepakat tau buntu, segera lanjut ke topik lain.
    - Negosiasi bukan permainan menang-kalah, tapi mencari solusi bersama.

- **Prinsip Perencanaan**
    - Pahami lingkup proyek.
    - Libatkan pelanggan dalam penyusunan rencana.
    - Rencana bersifat iteratif dan dapat berubah.
    - Estimasi berdasar informasi yang tersedia.
    - Pertimbangkan risiko sejak awal.
    - Sesuaikan detail rencana dengan kebutuhan.
    - Tentukan bagaimana kualitas akan dijaga.
    - Buat mekanisme menghadapi perubahan.
    - Lacak rencana secara rutin dan lakukan penyesuaian.

- **Prinsip Pemodelan (Agile Modeling)**
    - Tujuan utama adalah membangun perangkat lunak, bukan sekadar model.
    - Buat model secukupnya (*travel light*).
    - Utamakan model sederhana yang mudah dipahami.
    - Bangun model yang mudah berubah.
    - Setiap model harus punya tujuan jelas.
    - Sesuaikan model dengan sistem yang dibangun.
    - Cari model yang bermanfaat, bukan sempurna.
    - Jangan terlalu kaku dengan aturan model.
    - Ikuti intuisi—jika model tidak bekerja, periksa kembali.
    - Dapatkan umpan balik secepatnya.

- **Prinsip Konstruksi (Coding)**
    - **Sebelum menulis kode**: pahami masalah, prinsip desain, pilih bahasa & lingkungan pemrograman yang sesuai, serta siapkan unit test.
    - **Saat menulis kode**: gunakan *structured programming*, pertimbangkan pair programming, pilih struktur data yang tepat, dan buat antarmuka sesuai arsitektur.
    - **Setelah menulis kode**: lakukan code walkthrough, jalankan unit test, perbaiki kesalahan, dan lakukan refactoring untuk meningkatkan kualitas.

- **Prinsip Pengujian**
    - Semua tes harus bisa ditelusuri ke kebutuhan pelanggan.
    - Rencanakan tes jauh sebelum fase testing.
    - Gunakan prinsip Pareto (80% bug ditemukan di 20% modul).
    - Mulai dari pengujian kecil (unit) ke besar (sistem).
    - Tidak mungkin melakukan tes yang benar-benar menyeluruh.
    - Alokasikan usaha testing sesuai potensi kerusakan modul.
    - Gunakan static testing untuk hasil cepat.
    - Catat dan analisis pola kesalahan.
    - Sertakan tes untuk memastikan software berjalan benar.

- **Prinsip Deployment**
    - Kelola ekspektasi pelanggan sejak awal.
    - Paket perangkat lunak harus lengkap dan teruji.
    - Dukungan (support) harus disiapkan sebelum software dirilis.
    - Sediakan dokumentasi dan instruksi bagi pengguna akhir.
    - Perbaiki bug dulu, baru rilis produk.

RESUME
CHAPTER 7
UNDERSTANDING REQUIREMENTS

- **Rekayasa Kebutuhan (Requirements Engineering)**

Rekayasa kebutuhan adalah tahap paling awal dan paling penting dalam pengembangan perangkat lunak. Ibarat membangun rumah, tahap ini sama dengan menggambar blueprint/denah. Jika blueprint salah, rumah pasti salah juga.
Rekayasa Kebutuhan ini memiliki beberapa tahapan, yaitu:

1. Inception (Permulaan)

   Tahap awal untuk membangun pemahaman dasar tentang masalah, orang-orang yang menginginkan solusi, dan sifat solusi yang diinginkan. Tahap ini juga krusial untuk membangun komunikasi yang efektif antara pelanggan dan pengembang. Pada intinya tahap ini akan menggali masalah inti dan siapa yang terlibat. Contoh: aplikasi kasir dibuat karena pemilik toko ingin mempercepat transaksi.
2. Elicitation (Pengumpulan Kebutuhan)

   Proses untuk menggali persyaratan dan tujuan bisnis dari semua pemangku kepentingan (stakeholder). Contoh: kasir butuh fitur scan barcode, pemilik butuh laporan harian.
3. Elaboration (Perincian)

   Tahap ini berfokus pada pengembangan model persyaratan yang lebih rinci, yang mengidentifikasi aspek fungsi, perilaku, dan informasi dari perangkat lunak.
4. Negotiation (Negosiasi)

   Kesepakatan dicapai mengenai cakupan sistem yang dapat dikirimkan, yang realistis bagi pengembang dan pelanggan. Negosiasi bertujuan untuk mencapai hasil "win-win", di mana pelanggan mendapatkan produk yang memenuhi sebagian besar kebutuhan mereka dan pengembang mendapatkan tenggat waktu yang dapat dicapai. Salah satu metode untuk mencapai ini adalah handshaking, di mana pengembang mengusulkan solusi dan pelanggan meninjaunya.
5. Specification (Spesifikasi)

   Persyaratan dapat dispesifikasikan dalam berbagai bentuk, termasuk dokumen tertulis, model grafis, model matematis, skenario penggunaan, dan prototipe.
6. Validation (Validasi)

   Produk kerja rekayasa persyaratan dievaluasi untuk kualitas dan konsistensinya. Ini termasuk memeriksa apakah setiap persyaratan konsisten dengan tujuan sistem secara keseluruhan, berada pada tingkat abstraksi yang tepat, dan benar-benar diperlukan. Selain itu, setiap persyaratan harus terikat, tidak ambigu, dan dapat diuji setelah diimplementasikan.
7. Requirements Management (Manajemen Persyaratan)

Serangkaian aktivitas untuk membantu tim proyek mengidentifikasi, mengontrol, dan melacak persyaratan serta perubahannya seiring berjalannya proyek.

- **Persyaratan Non-Fungsional (Non-Functional Requirements – NFR)**

NFR adalah atribut kualitas, kinerja, atau keamanan, serta batasan umum sistem. Untuk menentukan NFR mana yang kompatibel, digunakan proses dua fase, yaitu:

1. Membuat matriks dengan setiap NFR sebagai kolom dan pedoman sistem sebagai baris.
2. Tim memprioritaskan setiap NFR dengan mengklasifikasikan pasangan NFR dan pedoman sebagai komplementer, tumpang tindih, bertentangan, atau independen.

- **Membangun Landasan (Establishing the Groundwork)**

Pertanyaan awal untuk stakeholder agar dapat memahami semua kebutuhan dari segala sudut pandang yang ada, seperti:

1. Siapa pengguna utama?
2. Apa manfaat bisnisnya?
3. Apakah sudah ada solusi lain?

- **Pengumpulan Persyaratan Kolaboratif (Collaborative Requirements Gathering)**

Pertemuan diadakan dengan insinyur perangkat lunak dan pemangku kepentingan lainnya. Pertemuan ini dipimpin oleh seorang "fasilitator" dan menggunakan mekanisme seperti lembar kerja atau bagan untuk memfasilitasi aliran ide. Tujuannya adalah untuk mengidentifikasi masalah, mengusulkan elemen solusi, dan menegosiasikan berbagai pendekatan.

- **Produk Kerja Elicitation (Elicitation Work Products)**

Hasil dari tahap penggalian persyaratan, meliputi:

1. Pernyataan kebutuhan & kelayakan.
2. Lingkup sistem yang jelas.
3. Daftar stakeholder.

4. Deskripsi lingkungan teknis.
5. Daftar persyaratan (fungsional & non-fungsional).
6. Kumpulan skenario penggunaan yang ditulis dengan kata-kata pemangku kepentingan.

- **Kasus Penggunaan (Use Case):**

Kasus penggunaan adalah kumpulan skenario pengguna yang menjelaskan cara penggunaan sistem. Setiap skenario dijelaskan dari sudut pandang "aktor" (orang atau perangkat) yang berinteraksi dengan perangkat lunak. Skema ini menjawab pertanyaan seperti: siapa aktor utama dan sekunder, apa tujuan aktor, dan apa tugas atau fungsi utama yang dilakukan.

- **Model Analisis (Analysis Model)**

Model analisis memberikan deskripsi domain informasi, fungsional, dan perilaku yang dibutuhkan oleh sistem. Elemen-elemen ini meliputi:

1. Berbasis Skenario

   Deskripsi fungsional yang diungkapkan dalam kata-kata pelanggan dan interaksi aktor dengan sistem menggunakan diagram kasus penggunaan UML. Contoh diagram kasus penggunaan menampilkan interaksi antara "Homeowner" (Pemilik Rumah) dan "System Administrator" (Administrator Sistem) dengan sistem dan sensor.

2. Berbasis Kelas

   Kumpulan atribut dan perilaku yang tersirat dari cerita pengguna dan diekspresikan menggunakan diagram kelas UML. Contoh diagram kelas menunjukkan kelas "Sensor" dengan atribut seperti nama dan lokasi, serta operasi seperti identify() dan enable().

3. Perilaku

   Dapat diekspresikan menggunakan diagram status UML, yang menunjukkan bagaimana input menyebabkan perubahan status. Contoh diagram status menggambarkan transisi dari status "Reading Commands" ke "System Status = 'off'".

- **Pola Analisis (Analysis Patterns)**

  Pola analisis adalah deskriptor yang menangkap esensi pola, dengan elemen-elemen seperti nama, tujuan, motivasi, konteks, solusi, konsekuensi, desain, kegunaan yang diketahui, dan pola terkait. Pemecahan pemecahan masalah umum dengan pola yang umum. Contoh: pola "Login" yang hampir ada di semua aplikasi.

- **Negosiasi Kebutuhan (Negotiating Requirements)**

  Negosiasi bertujuan untuk mencapai hasil "win-win", di mana:

  1. Pemangku kepentingan menang dengan mendapatkan produk yang memenuhi sebagian besar kebutuhan mereka.
  2. Pengembang menang dengan mendapatkan tenggat waktu yang dapat dicapai.

  Salah satu cara untuk mencapai "win-win" adalah melalui *handshaking*:

  1. Pengembang mengusulkan solusi terhadap kebutuhan, menjelaskan dampaknya, dan menyampaikan niat mereka kepada pelanggan.
  2. Pelanggan meninjau solusi yang diusulkan, mencari fitur yang hilang, dan meminta klarifikasi terhadap kebutuhan baru.
  3. Suatu kebutuhan dianggap cukup baik jika pelanggan menerima solusi yang diusulkan.

  Handshaking membantu meningkatkan identifikasi, analisis, dan pemilihan variasi solusi.

- **Pemantauan Persyaratan (Requirements Monitoring)**

  Berguna untuk pengembangan inkremental dan mencakup:

  1. Distributed debugging (debugging terdistribusi) untuk mengungkap kesalahan.
  2. Run-time verification (verifikasi saat runtime) untuk menentukan apakah perangkat lunak sesuai dengan spesifikasinya.
  3. Run-time validation (validasi saat runtime) untuk menilai apakah perangkat lunak memenuhi tujuan pengguna.
  4. Business activity monitoring untuk mengevaluasi apakah sistem memenuhi tujuan bisnis.
  5. Evolution and codesign untuk memberikan informasi kepada pemangku kepentingan saat sistem berkembang.

- **Validasi Kebutuhan (Validating Requirements)**

  Validasi kebutuhan bertujuan untuk memastikan bahwa setiap kebutuhan yang ditentukan benar-benar relevan, jelas, dan sesuai dengan tujuan sistem atau produk. Proses ini mencakup:

  1. Konsistensi dengan tujuan keseluruhan sistem.
  2. Tingkat detail yang sesuai dengan tahap pengembangan.
  3. Penilaian apakah kebutuhan tersebut esensial atau hanya tambahan.
  4. Kejelasan dan batasan yang tidak ambigu.
  5. Identifikasi sumber atau asal dari setiap kebutuhan.
  6. Deteksi potensi konflik antar kebutuhan.

  Dengan melakukan validasi ini, tim pengembang dapat menghindari kesalahan, mengurangi risiko, dan memastikan bahwa sistem yang dibangun benar-benar memenuhi harapan pengguna dan pemangku kepentingan