BAB 2
LANDASAN TEORI

2.1 Inventaris

Inventaris adalah stok barang atau bahan yang digunakan oleh suatu organisasi atau perusahaan untuk menjalankan operasionalnya. Jika perusahaan menghasilkan produk atau jasa, bahan tersebut digunakan untuk mendukung atau memenuhi kebutuhan produksi [10].

2.2 Agile

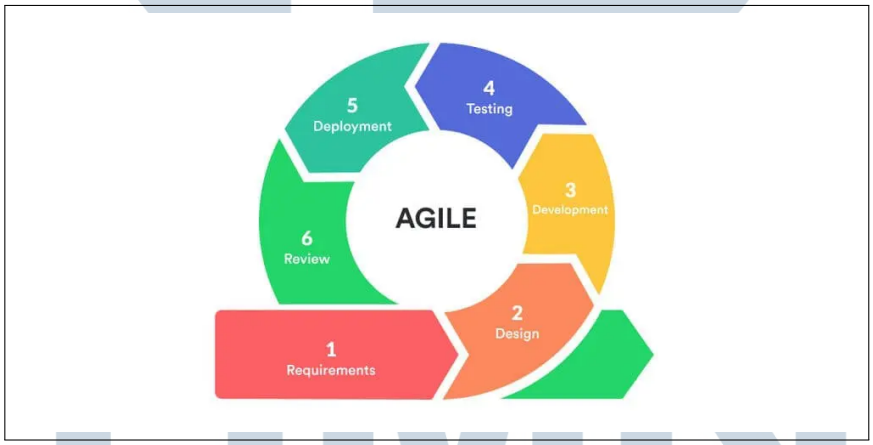
Agile adalah pendekatan modern dalam pengembangan perangkat lunak yang memungkinkan pengembang menciptakan *software* secara lebih fleksibel, cepat, dan responsif terhadap perubahan kebutuhan pengguna. Pendekatan ini muncul sebagai solusi dari keterbatasan metode tradisional seperti *Waterfall*, yang bersifat kaku, linier, dan sulit menyesuaikan perubahan. Agile memungkinkan siklus pengembangan yang bersifat iteratif dan inkremental, di mana setiap tahapan pengembangan menghasilkan bagian fungsional dari perangkat lunak yang dapat segera diuji dan dievaluasi. Dengan demikian, Agile mendukung kolaborasi yang lebih intensif antara pengembang dan pengguna, serta fokus pada kepuasan pelanggan melalui pengiriman perangkat lunak berkualitas tinggi yang dapat diubah atau ditingkatkan sesuai kebutuhan pasar [11].

Siklus pengembangan dalam metode Agile terdiri dari beberapa tahapan yang dilakukan secara berulang dan disesuaikan berdasarkan hasil evaluasi setiap iterasi. Tahapan utama dalam Agile meliputi:

- **Requirement**: Tahap awal ini berfokus pada pengumpulan kebutuhan pengguna atau stakeholder secara berkala. Tidak seperti metode tradisional yang mengumpulkan semua kebutuhan di awal proyek, Agile memungkinkan perubahan kebutuhan pada setiap iterasi agar produk tetap relevan.
- **Design**: Pada tahap ini, pengembang merancang solusi awal yang fleksibel dan mudah diadaptasi. Desain tidak dibuat secara mendetail di awal, melainkan berkembang seiring iterasi berikutnya.
- **Development**: Pengembangan dilakukan dalam *sprint* pendek, biasanya

berkisar antara 1 hingga 4 minggu. Pada setiap *sprint*, pengembang fokus menciptakan fitur atau bagian kecil dari produk yang dapat langsung diuji.

- **Testing**: Setiap *increment* atau bagian hasil pengembangan diuji secara menyeluruh untuk memastikan fungsionalitas dan kualitasnya. Pengujian ini dilakukan secara terus-menerus untuk meminimalisir bug atau kesalahan.
- **Deployment**: Produk atau fitur yang telah selesai dikembangkan akan dirilis secara bertahap kepada pengguna. Rilis dilakukan dalam skala kecil agar pengguna dapat memberikan masukan dengan cepat.
- **Review**: Tahap ini melibatkan evaluasi hasil pengembangan bersama pengguna atau stakeholder. Masukan yang diperoleh digunakan sebagai acuan untuk iterasi berikutnya, sehingga pengembangan dapat lebih selaras dengan ekspektasi pengguna.



Gambar 2.1. Metode Agile

Dengan adanya tahapan tersebut, Agile menciptakan lingkungan pengembangan yang dinamis dan kolaboratif. Pengembang dapat merespons perubahan dengan cepat, sedangkan pengguna memiliki kesempatan untuk memberikan masukan secara langsung dalam setiap siklus iterasi. Prinsip-prinsip ini diadopsi dari *Agile Manifesto* yang menekankan empat nilai utama, yaitu: individu dan interaksi lebih penting daripada proses dan alat, perangkat lunak yang berfungsi lebih diutamakan daripada dokumentasi yang menyeluruh, kolaborasi dengan pelanggan lebih penting dibandingkan negosiasi kontrak, serta respons terhadap perubahan lebih diutamakan daripada mengikuti rencana secara kaku [12].

Pendekatan Agile menawarkan banyak keuntungan, seperti percepatan waktu pengembangan, peningkatan kualitas perangkat lunak, serta kemampuan untuk merespons perubahan kebutuhan dengan cepat. Selain itu, metode ini membantu meminimalisir risiko kegagalan proyek karena produk dievaluasi secara berkala dan dapat dikoreksi segera jika terdapat kesalahan atau ketidaksesuaian. Implementasi metode Agile yang digunakan dalam penelitian ini dapat dilihat pada Gambar 2.1.

2.3 Diagram UML

2.3.1 Pengertian dan Penjelasan Diagram UML

Unified Modeling Language (UML) adalah bahasa pemodelan standar yang digunakan untuk mendeskripsikan, merancang, dan mendokumentasikan sistem perangkat lunak. UML memungkinkan pengembang untuk menggambarkan struktur dan perilaku sistem secara visual, mempermudah pemahaman dan komunikasi antara pemangku kepentingan dalam proyek perangkat lunak.

UML digunakan pada berbagai tahap pengembangan perangkat lunak, mulai dari analisis kebutuhan hingga implementasi dan pengujian. Diagram UML dapat digunakan untuk menggambarkan sistem secara konseptual maupun detail teknis.

2.3.2 Jenis-Jenis Diagram UML

Berikut adalah empat jenis diagram UML yang sering digunakan dalam pengembangan perangkat lunak:

A Activity Diagram

Diagram ini menggambarkan alur aktivitas atau proses dalam sistem, termasuk aliran kerja yang bersifat paralel atau bersyarat. Activity diagram sering digunakan untuk memodelkan logika bisnis atau langkah-langkah operasional yang kompleks.

B Use Case Diagram

Diagram ini menggambarkan interaksi antara aktor (pengguna atau sistem eksternal) dengan sistem yang dirancang. Use case diagram berfokus pada

bagaimana sistem digunakan untuk mencapai tujuan tertentu.

C Class Diagram

Diagram ini menggambarkan struktur statis dari sistem, termasuk kelas, atribut, metode, dan hubungan antar kelas. Class diagram sering digunakan untuk mendeskripsikan komponen utama dalam sistem dan relasinya.

D Sequence Diagram

Diagram ini mengilustrasikan interaksi antara objek dalam urutan waktu tertentu. Sequence diagram menggambarkan pesan yang dikirimkan antara objek untuk menyelesaikan suatu proses.

2.3.3 Manfaat Diagram UML

Penggunaan diagram UML memberikan beberapa manfaat utama, seperti:

- **Mempermudah komunikasi:** Memberikan visualisasi sistem yang mudah dipahami oleh pengembang dan pemangku kepentingan.
- **Dokumentasi terstruktur:** Membantu dalam mendokumentasikan sistem secara konsisten untuk digunakan kembali.
- **Meningkatkan efisiensi:** Membantu mengidentifikasi masalah sejak awal dan mendukung pengembangan modular.

2.4 User Acceptance Test (UAT)

User Acceptance Test (UAT) adalah proses yang dilakukan oleh pengguna untuk memastikan bahwa sistem yang dikembangkan sudah sesuai dengan kebutuhan mereka dan dapat diterima. Pengujian UAT bertujuan untuk mengevaluasi tingkat kepuasan pengguna dalam menggunakan aplikasi. Pengujian ini sangat penting untuk mendeteksi potensi kesalahan dalam sistem yang telah dibuat, sehingga jika terdapat masalah, perbaikan dapat segera dilakukan [13].

2.5 Skala Likert

Skala Likert adalah metode pengukuran yang digunakan untuk menilai tanggapan individu terhadap sejumlah pernyataan atau pertanyaan terkait perilaku, sikap, atau opini mereka. Penggunaannya melibatkan serangkaian pernyataan yang harus dijawab oleh responden dengan memilih salah satu dari lima opsi respons yang telah ditentukan. Lima pilihan tersebut terdiri dari: sangat setuju, setuju, netral (tidak memutuskan), tidak setuju, dan sangat tidak setuju. Setiap pilihan mencerminkan tingkat persetujuan atau ketidaksetujuan terhadap pernyataan yang diberikan. Skala ini sangat berguna dalam penelitian sosial, psikologi, dan bisnis untuk mengukur sikap dan pendapat dengan cara yang lebih kuantitatif [14], [15].

1. Menghitung masing-masing pertanyaan kuesioner.

$$SkorTotal = (1 \times P1) + (2 \times P2) + (3 \times P3) + (4 \times P4) + (5 \times P5) \quad (2.1)$$

   - P1 = Total responden menjawab ”Sangat Tidak Setuju”
   - P2 = Total responden menjawab ”Tidak Setuju”
   - P3 = Total responden menjawab ”Netral”
   - P4 = Total responden menjawab ”Setuju”
   - P5 = Total responden menjawab ”Sangat Setuju”

   Skor Total adalah hasil skor dari setiap aspek.

2. Menghitung skor yang dilakukan untuk setiap pertanyaan agar mendapatkan hasil perhitungan.

$$\text{Interpretasi}(\%) = \frac{\text{SkorTotal}}{Y} \times 100 \quad (2.2)$$

   Keterangan:

   - Y= Skor maksimum setiap aspek
   - Skor Total = Hasil skor setiap aspek

3. Menghitung Interval Skor.

Mengkalkulasi interval dan persentase penting untuk memahami evaluasi melalui teknik penentuan interval skor persentase.

$$I = \frac{50}{\text{Jumlah skor likert}} \tag{2.3}$$

Karena jumlah skor yang digunakan ada 5, maka perhitungan dapat dikerjakan.

$$I = \frac{50}{5} \tag{2.4}$$

$$I = 10 \tag{2.5}$$

(10 merupakan jarak dari terendah 0% hingga tertinggi 100%)

Tabel 2.1. Kriteria Perhitungan skor

Keterangan	Persentase
Sangat Tidak Setuju	0% - 19,99%
Tidak Setuju	20% - 39,99%
Netral	40% - 59,99%
Setuju	60% - 79,99%
Sangat Setuju	80% - 100%

2.6 Black Box Testing

Black Box Testing adalah metode pengujian perangkat lunak yang berfokus pada evaluasi fungsionalitas perangkat lunak tanpa memperhatikan detail internal atau implementasi kode sumbernya. Metode ini digunakan untuk memvalidasi apakah perangkat lunak bekerja sesuai dengan spesifikasi yang telah ditentukan,

dengan memeriksa keluaran (output) berdasarkan serangkaian masukan (input) tertentu. Pengujian ini sering digunakan untuk mendeteksi kesalahan yang berkaitan dengan:

- Fungsi yang tidak berjalan dengan benar atau fungsi yang hilang.
- Kesalahan pada antarmuka pengguna, seperti kesalahan navigasi atau tampilan.
- Masalah pada pengelolaan dan akses terhadap struktur data serta basis data.
- Ketidakoptimalan performa, seperti waktu respons yang terlalu lama atau kegagalan dalam pengelolaan sumber daya.
- Kesalahan selama proses inisialisasi atau terminasi aplikasi.

Pengujian Black Box Testing biasanya dilakukan oleh tim pengujian (testers) yang tidak memiliki pengetahuan tentang detail kode sumber. Hal ini memungkinkan penguji untuk menempatkan diri mereka pada perspektif pengguna akhir (end-user), sehingga pengujian berfokus pada aspek-aspek eksternal perangkat lunak yang dapat diakses oleh pengguna. Jenis pengujian ini mencakup berbagai teknik, seperti:

- **Equivalence Partitioning**: Membagi ruang input menjadi kelas-kelas ekivalen untuk mengurangi jumlah kasus uji yang diperlukan.
- **Boundary Value Analysis**: Menguji nilai-nilai batas, karena kesalahan sering ditemukan pada area ini.
- **Decision Table Testing**: Menggunakan tabel keputusan untuk mengidentifikasi kombinasi masukan dan keluaran yang mungkin.
- **State Transition Testing**: Memeriksa perilaku perangkat lunak berdasarkan perubahan status.
- **Error Guessing**: Bergantung pada pengalaman penguji untuk mengantisipasi kesalahan yang mungkin terjadi.

Metode ini dilakukan oleh berbagai pihak yang terlibat dalam pengembangan perangkat lunak, termasuk:

- **Tim QA (Quality Assurance)**: Bertanggung jawab untuk memastikan bahwa perangkat lunak memenuhi standar kualitas.
- **Penguji Independen (Independent Testers)**: Profesional eksternal yang tidak terlibat dalam proses pengembangan untuk memberikan evaluasi yang objektif.
- **Pengguna Akhir (End-Users)**: Kadang-kadang diundang untuk melakukan pengujian dalam bentuk uji coba beta (beta testing) guna mendapatkan umpan balik langsung.

Dengan pendekatan ini, Black Box Testing melengkapi metode pengujian lainnya, seperti White Box Testing, untuk memberikan cakupan pengujian yang lebih komprehensif. Proses ini bertujuan untuk memastikan bahwa perangkat lunak tidak hanya berfungsi sesuai dengan desainnya, tetapi juga memberikan pengalaman pengguna yang optimal. [16].