BAB II
KAJIAN LITERATUR

2.1 Konsep Sistem Informasi

2.1.1 Sistem

Sistem merupakan suatu kesatuan yang dibentuk oleh berbagai elemen yang saling berinteraksi dan bekerja sama untuk mencapai tujuan tertentu. Elemen-elemen tersebut dapat bersifat nyata maupun abstrak, namun membentuk satu kesatuan yang terstruktur. Proses kerja suatu sistem dipengaruhi oleh pola hubungan antar elemen serta mekanisme interaksi yang berlangsung di dalamnya [2].

Setiap sistem memiliki karakteristik utama, yaitu adanya masukan (*input*), proses, keluaran (*output*), dan umpan balik (*feedback*). Masukan merupakan data atau sumber daya yang masuk ke dalam sistem, kemudian diolah melalui suatu proses untuk menghasilkan keluaran berupa informasi atau hasil tertentu. Umpan balik digunakan untuk mengendalikan dan memperbaiki proses sistem agar tetap sesuai dengan tujuan yang telah ditetapkan. Selain itu, sistem memiliki batasan yang memisahkannya dari lingkungan luar, namun tetap dipengaruhi oleh lingkungan tersebut [2], [3].

Keberadaan tujuan menjadi aspek penting dalam sistem karena tujuan tersebut menentukan arah kerja sistem. Tanpa tujuan yang jelas, sistem tidak dapat berfungsi secara optimal dan tidak mampu menghasilkan keluaran yang bernilai bagi penggunanya.

2.1.2 Informasi

Informasi merupakan hasil pengolahan data yang telah diberi makna sehingga dapat digunakan untuk memahami suatu kondisi atau peristiwa. Data yang masih bersifat mentah belum memiliki nilai guna yang signifikan sebelum diproses dan dikontekstualisasikan. Setelah melalui proses pengolahan, data tersebut berubah menjadi informasi yang dapat digunakan untuk menjawab pertanyaan dan mendukung aktivitas organisasi [2].

Nilai informasi sangat ditentukan oleh kualitasnya. Informasi yang berkualitas harus akurat, relevan, dan tersedia pada waktu yang tepat. Informasi yang akurat mencerminkan kondisi yang sebenarnya, sedangkan relevansi menunjukkan kesesuaian informasi dengan kebutuhan pengguna. Ketepatan waktu menjadi penting karena informasi yang terlambat dapat kehilangan nilainya dalam mendukung pengambilan keputusan [3].

Dalam pengambilan keputusan, informasi berfungsi untuk mengurangi ketidakpastian dan membantu manajemen dalam memilih alternatif tindakan yang paling tepat. Informasi yang baik memungkinkan organisasi memahami kondisi internal dan eksternal secara lebih menyeluruh, sehingga keputusan yang diambil dapat memberikan dampak positif bagi pencapaian tujuan organisasi [2].

2.1.3 Sistem Informasi

Sistem informasi didefinisikan sebagai sekumpulan komponen yang saling berhubungan dan bekerja sama untuk mengumpulkan, mengolah, menyimpan, serta mendistribusikan informasi guna mendukung pengendalian, koordinasi, dan pengambilan keputusan dalam organisasi [2]. Sistem informasi dirancang untuk memastikan bahwa informasi yang dihasilkan memiliki kualitas yang baik dan dapat diakses oleh pihak yang membutuhkan.

Komponen utama sistem informasi meliputi teknologi, manusia, tugas, dan struktur organisasi. Teknologi mencakup perangkat keras, perangkat lunak, serta sarana komunikasi yang digunakan untuk memproses data. Manusia berperan sebagai pengguna dan pengelola sistem, sedangkan tugas dan struktur organisasi mengatur bagaimana sistem informasi digunakan dalam mendukung proses bisnis. Keterpaduan antar komponen tersebut menentukan keberhasilan sistem informasi dalam mencapai tujuannya [2], [3].

Dalam operasional organisasi, sistem informasi berperan dalam meningkatkan efisiensi proses kerja, mempercepat aliran informasi, serta meningkatkan akurasi data. Selain itu, sistem informasi juga mendukung manajemen dalam perencanaan, pengendalian, dan evaluasi kinerja organisasi. Dengan demikian, sistem informasi menjadi salah satu elemen strategis yang berkontribusi langsung terhadap keberhasilan dan daya saing organisasi.

2.2 Jenis Sistem Informasi

2.2.1 Sistem Informasi Pembelian

Sistem pembelian merupakan rangkaian aktivitas yang berkaitan dengan pengadaan barang dari *supplier* untuk memenuhi kebutuhan operasional organisasi. Proses pembelian umumnya dimulai dari identifikasi kebutuhan barang, pembuatan pesanan pembelian, penerimaan barang, hingga pencatatan transaksi pembelian. Ketepatan dalam proses pembelian sangat berpengaruh terhadap kelancaran operasional, terutama dalam memastikan ketersediaan barang sesuai dengan kebutuhan.

Fungsi utama sistem pembelian adalah menjamin tersedianya barang dengan kualitas, jumlah, dan waktu yang tepat, serta dengan biaya yang efisien. Selain itu, sistem pembelian juga berfungsi sebagai alat pengendalian terhadap pengeluaran perusahaan dan sebagai sarana dokumentasi transaksi yang terjadi dengan *supplier*. Pencatatan yang tidak terstruktur dalam proses pembelian dapat menyebabkan kesulitan dalam memantau hutang, harga barang, serta histori transaksi pembelian.

Peran sistem informasi dalam mendukung proses pembelian sangat signifikan, terutama dalam mengintegrasikan data *supplier*, data barang, dan transaksi pembelian ke dalam satu sistem yang terpusat. Dengan adanya sistem informasi pembelian, proses pencatatan menjadi lebih terstruktur, pembuatan laporan pembelian dapat dilakukan secara cepat, serta informasi terkait pemesanan dan penerimaan barang dapat diakses dengan mudah. Sistem informasi pembelian juga membantu manajemen dalam mengambil keputusan yang lebih tepat terkait waktu dan jumlah pemesanan barang [4].

2.2.2 Sistem Informasi Persediaan

Persediaan merupakan aset penting bagi organisasi karena berfungsi sebagai penunjang utama dalam proses penjualan dan operasional. Persediaan mencakup seluruh barang yang disimpan untuk dijual kembali atau digunakan dalam aktivitas bisnis. Pengelolaan persediaan yang tidak baik dapat menyebabkan terjadinya kelebihan stok, kekurangan barang, maupun kerugian akibat barang rusak atau kedaluwarsa.

Tujuan utama pengelolaan persediaan adalah menjaga keseimbangan antara ketersediaan barang dan biaya penyimpanan. Organisasi perlu memastikan bahwa jumlah persediaan selalu berada pada tingkat yang optimal agar mampu memenuhi permintaan tanpa menimbulkan pemborosan. Selain itu, pengelolaan persediaan juga bertujuan untuk menyediakan informasi yang akurat mengenai jumlah barang masuk, barang keluar, serta sisa stok yang tersedia.

Sistem informasi persediaan berperan dalam mencatat dan mengelola seluruh transaksi yang berkaitan dengan pergerakan barang, baik barang masuk dari hasil pembelian maupun barang keluar akibat penjualan. Dengan sistem informasi persediaan yang terintegrasi, data stok dapat diperbarui secara otomatis setiap kali terjadi transaksi, sehingga informasi yang dihasilkan menjadi lebih akurat dan *up-to-date*. Sistem ini juga membantu dalam pengendalian stok barang, pembuatan laporan persediaan, serta mendukung pengambilan keputusan terkait pengadaan barang dan pengendalian aset perusahaan [4].

2.3 Situs Web Sebagai Media Sistem Informasi

2.3.1 Situs Web

Situs web merupakan sekumpulan halaman yang saling terhubung dan dapat diakses melalui jaringan internet. Situs web mampu memuat berbagai jenis konten, seperti teks, gambar, dan multimedia, yang disusun secara terstruktur sehingga memudahkan pengguna dalam memperoleh informasi. Pemanfaatan situs web banyak digunakan sebagai media informasi dan promosi karena mampu menjangkau pengguna secara luas tanpa batasan ruang dan waktu.

Dalam konteks sistem informasi, situs web berperan sebagai antarmuka yang menghubungkan pengguna dengan sistem yang berjalan di sisi *server*. Melalui situs web, pengguna dapat melakukan input data, melihat hasil pengolahan data, serta memantau informasi yang disediakan oleh sistem secara terpusat. Peran ini menjadikan situs web sebagai komponen penting dalam mendukung proses penyampaian informasi dan pengelolaan data berbasis teknologi [5].

2.3.2 Jenis-Jenis Situs Web

Situs web dapat diklasifikasikan berdasarkan sifat pengelolaan konten serta tujuan penggunaannya. Klasifikasi ini menunjukkan perbedaan karakteristik dan fungsi situs web dalam mendukung kebutuhan pengguna.

Situs web statis merupakan situs web yang memiliki konten bersifat tetap dan jarang mengalami perubahan. Perubahan informasi pada situs web statis dilakukan secara manual oleh pengelola. Situs web jenis ini umumnya digunakan untuk menampilkan informasi sederhana yang tidak memerlukan interaksi pengguna secara kompleks.

Situs web dinamis merupakan situs web yang kontennya dapat berubah secara otomatis dan bersifat interaktif. Situs web ini memungkinkan adanya interaksi antara pengguna dan sistem, seperti penginputan data dan penyajian informasi hasil pengolahan sistem. Situs web dinamis banyak digunakan dalam penerapan sistem informasi karena mendukung pengelolaan data yang terintegrasi.

Selain itu, situs web juga dapat dibedakan berdasarkan tujuan penggunaannya. Situs web pribadi digunakan untuk menampilkan informasi, portofolio, atau karya yang dimiliki oleh individu. Situs web bisnis atau perusahaan digunakan untuk menyampaikan informasi terkait perusahaan, produk, atau layanan yang ditawarkan. Situs web komunitas dimanfaatkan sebagai media berbagi informasi dan komunikasi antar anggota komunitas.

Situs web *e-commerce* digunakan untuk mendukung kegiatan jual beli barang atau jasa secara *online* serta menampilkan informasi produk dan aktivitas bisnis yang dijalankan [6].

2.4 Basis Data

2.4.1 Konsep Basis Data

Basis data atau *database* merupakan sekumpulan data atau informasi yang saling berhubungan, disimpan, dan disusun di dalam komputer secara metodis. Implementasi basis data bertujuan untuk menyediakan lingkungan terstruktur yang memudahkan manajemen, penyimpanan, dan akses data secara efisien [7], [8]. Adapun adalah aspek-aspek fundamental mengenai basis data adalah sebagai berikut:

1. Skema dan Struktur: Basis data memiliki penjelasan terorganisir mengenai jenis fakta yang tersimpan di dalamnya yang disebut sebagai skema. Skema menggambarkan objek yang diwakili serta hubungan fungsional di antara objek tersebut.
2. Model Data: Terdapat berbagai metodologi untuk mengatur skema, seperti model relasional yang merepresentasikan informasi dalam bentuk tabel (baris dan kolom), serta model hierarki dan jaringan yang menggunakan pendekatan lebih eksplisit dalam mengekspresikan hubungan antartabel.
3. Tujuan Operasional: Tujuan utama basis data adalah untuk mengelola data secara efektif dan efisien, mengatasi masalah redundansi data (duplikasi), serta menghindari isolasi data akibat perbedaan format *file*.
4. Fungsi Strategis: Dalam lingkungan organisasi, basis data merupakan instrumen krusial untuk menghasilkan analisis data yang mendalam, sehingga memfasilitasi pengambilan keputusan strategis berdasarkan data yang akurat dan *real-time* [7], [8], [9].

2.4.2 Sistem Manajemen Basis Data

Sistem Manajemen Basis Data atau *Database Management System* (DBMS) adalah perangkat lunak khusus yang dirancang untuk membangun, mengelola, memelihara, dan mengontrol akses terhadap basis data. DBMS berfungsi sebagai perantara (*interface*) antara pengguna dengan basis data fungsional guna memastikan integritas, keamanan, dan kinerja sistem yang optimal. Secara deskriptif, fungsi-fungsi utama DBMS meliputi:

1. Integritas dan Konsistensi: DBMS meminimalkan redundansi data guna memaksimalkan konsistensi informasi agar data yang ditampilkan selalu sesuai dengan data aslinya.

2. Manajemen Penyimpanan: Perangkat lunak ini mengelola prosedur penyimpanan fisik dan manipulasi data, termasuk tipe data kompleks seperti gambar dan video, tanpa mengharuskan pengguna mengetahui detail teknis penyimpanannya.
3. Keamanan dan Hak Akses: DBMS menyediakan sistem izin akses yang terdefinisi dengan baik, membatasi akses ke informasi sensitif, dan mengatur wewenang pengguna sesuai dengan perannya masing-masing.
4. Manajemen Transaksi dan Pemulihan: Sistem ini menjamin keakuratan melalui pengelolaan transaksi yang terjadi bersamaan (konkurensi) serta menyediakan fitur *backup* dan *restore* untuk pemulihan data jika terjadi kegagalan sistem atau bencana.
5. Manipulasi Data: DBMS menyediakan bahasa generasi keempat (4GL) seperti SQL (*Structured Query Language*) yang mencakup *Data Definition Language* (DDL) untuk membuat skema dan *Data Manipulation Language* (DML) untuk operasi input, modifikasi, serta penghapusan data [8], [9].

2.4.3 PostgreSQL

PostgreSQL adalah sistem manajemen basis data relasional (*Relational Database Management System*/RDBMS) berbasis *opensource* yang menekankan pada keandalan, kepatuhan terhadap standar SQL, serta dukungan fitur lanjutan. PostgreSQL dirancang khusus untuk menangani kebutuhan pengembangan aplikasi skala besar dengan karakteristik teknis sebagai berikut:

1. Keandalan Transaksi: PostgreSQL mendukung kepatuhan ACID (*Atomicity*, *Consistency*, *Isolation*, *Durability*) secara penuh, memastikan setiap transaksi diproses secara andal tanpa risiko korupsi data [10].
2. Mekanisme Konkurensi: Menggunakan metode *Multi-Version Concurrency Control* (MVCC) yang memungkinkan banyak pengguna mengakses sumber daya yang sama secara bersamaan tanpa adanya hambatan penguncian data yang signifikan [10], [11].
3. Fleksibilitas Tipe Data: Mendukung berbagai tipe data kompleks seperti JSONB, *array*, *hstore*, dan *range*, yang sangat esensial untuk kebutuhan analisis data intensif dan pengembangan aplikasi modern [10].
4. Optimasi Kueri Kompleks: Hasil pengujian menunjukkan bahwa PostgreSQL memiliki stabilitas performa yang lebih tinggi dibandingkan DBMS lain dalam menangani kueri yang melibatkan penggabungan banyak tabel (*complex join clauses*) dan fungsi agregasi pada volume data besar [10], [12].

5. *Information Schema*: Sistem ini menyediakan metadata skema internal yang menyimpan detail mengenai tabel, kolom, batasan (*constraints*), dan prosedur. Pemanfaatan *information schema* ini memungkinkan pengembang untuk memetakan kolom *database* ke dalam bentuk aplikasi secara dinamis, sehingga mempercepat proses migrasi dan pengembangan formulir tanpa pengulangan kode [11].

2.5 Metode Pengembangan Sistem

2.5.1 *System Development Life Cycle* (SDLC)

System Development Life Cycle (SDLC) merupakan sebuah kerangka kerja konseptual yang digunakan untuk merencanakan, menganalisis, merancang, membangun, menguji, hingga menyebarkan sebuah sistem informasi. Secara fundamental, SDLC berfungsi sebagai metodologi sistematis bagi pengembang untuk memastikan bahwa perangkat lunak yang dihasilkan memiliki kualitas tinggi, andal, serta mampu memenuhi kebutuhan pengguna dan pemangku kepentingan dalam batasan waktu dan anggaran yang telah ditentukan.

Struktur SDLC mencakup konstelasi tahapan yang diatur secara logis untuk mengelola risiko serta memberikan kendali penuh terhadap proses pengembangan sistem. Pemilihan metodologi SDLC yang tepat sangat bergantung pada berbagai faktor, seperti kompleksitas proyek, stabilitas kebutuhan (*requirements*), ketersediaan sumber daya, serta tingkat keterlibatan pelanggan yang diharapkan [13].

Secara garis besar, pendekatan SDLC terbagi menjadi dua paradigma utama:

1. Pendekatan Tradisional (*Heavy-weight*): Bersifat prediktif, linier, dan sangat menekankan pada dokumentasi serta perencanaan detail di awal proyek, seperti model *Waterfall* dan V-Model.
2. Pendekatan *Agile* (*Light-weight*): Bersifat adaptif, iteratif, dan inkremental, yang memprioritaskan kolaborasi langsung dengan pelanggan serta fleksibilitas terhadap perubahan kebutuhan [14].

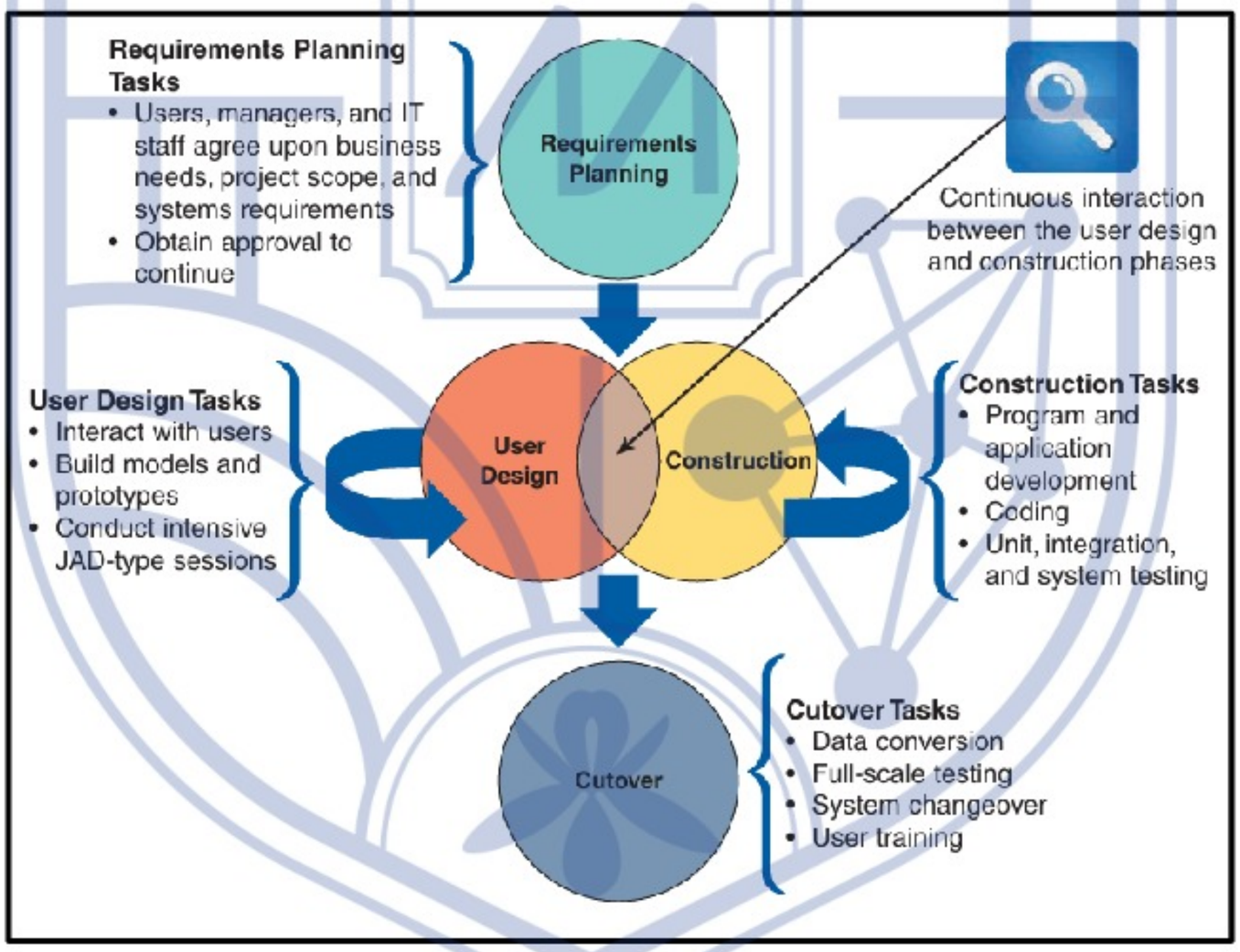
2.5.2 Metode *Rapid Application Development*

Rapid Application Development (RAD) merupakan sebuah metodologi pengembangan sistem yang memprioritaskan kecepatan dan fleksibilitas melalui proses yang bersifat iteratif dan partisipatif. Berbeda dengan model tradisional yang mengharuskan setiap tahapan selesai terlebih dahulu sebelum berpindah ke tahap berikutnya, RAD memungkinkan tim pengembang untuk bergerak cepat dari tahap analisis kebutuhan menuju

pembuatan prototipe fungsional yang dapat langsung diuji oleh pengguna [15]. Strategi ini bertujuan untuk mengurangi waktu yang sering kali terbuang dalam dokumentasi formal yang berlebihan, sehingga sistem dapat diserahkan kepada perusahaan dalam durasi yang lebih singkat namun tetap memiliki kualitas yang tinggi [16].

Keunggulan utama metodologi ini terletak pada penggunaan alat bantu pengembangan perangkat lunak modern, seperti *Computer-Aided Software Engineering* (CASE) dan teknik *prototyping*, yang memungkinkan pengembang untuk membangun elemen-elemen sistem secara dinamis. Melalui pendekatan ini, pengguna tidak perlu menunggu hingga akhir proyek untuk melihat hasil kerja, sebaliknya, mereka dapat memberikan umpan balik langsung pada model sistem yang sedang dikembangkan [15]. Hal ini secara signifikan dapat meminimalisasi risiko kegagalan proyek akibat adanya perbedaan antara persepsi pengembang dan kebutuhan nyata pengguna di lapangan [16].

Adapun siklus hidup pengembangan dalam metode RAD secara sistematis dikelompokkan ke dalam empat fase utama yang saling berhubungan, yaitu:



Gambar 2.1 Tahapan Metode *Rapid Application Developme*nt

1. Perencanaan Kebutuhan (*Requirements Planning*)

   Fase ini merupakan titik awal di mana para pemangku kepentingan, manajer, dan tim IT berkumpul untuk menetapkan tujuan strategis, batasan proyek, serta kebutuhan sistem secara garis besar. Fokusnya bukan pada detail teknis yang mendalam, melainkan pada pemahaman bersama mengenai masalah bisnis yang ingin diselesaikan [15], [16].

2. Desain Pengguna (*User Design*)

Pada tahap ini, interaksi antara pengguna dan analis sistem terjadi secara intensif. Pengguna akan meninjau, mencoba, dan memberikan masukan terhadap prototipe yang dikembangkan. Proses ini dilakukan secara berulang (iteratif) hingga model sistem benar-benar mencerminkan seluruh proses bisnis, baik dari sisi input, pengolahan data, maupun output yang dihasilkan [16].

3. Konstruksi (*Construction*)

   Berbeda dengan tahap pengkodean tradisional, fase konstruksi dalam RAD melibatkan pengembangan aplikasi secara berkelanjutan di mana pengguna tetap memberikan validasi terhadap fitur-fitur yang dibangun. Pengembang memanfaatkan umpan balik dari fase desain untuk menyempurnakan kode program dan mengintegrasikannya ke dalam fungsi sistem yang utuh [15], [16].

4. Perpindahan (*Cutover*)

   Merupakan tahap perpindahan yang meliputi konversi data dari sistem lama ke sistem baru, serta pelatihan bagi pengguna. Karena pengguna telah terlibat sejak awal pengembangan, fase transisi ini biasanya berlangsung lebih lancar karena tingkat familiaritas pengguna terhadap sistem yang baru sudah sangat tinggi [16].

Secara keseluruhan, implementasi RAD memberikan nilai tambah bagi organisasi dalam bentuk efisiensi biaya dan waktu pengembangan. Namun, kesuksesan metode ini sangat bergantung pada keterlibatan aktif dan komitmen dari pengguna serta ketersediaan tim pengembang yang memiliki keahlian teknis tinggi untuk menjalankan alat bantu pengembangan secara optimal [15]. Dengan koordinasi yang baik, RAD mampu menghasilkan solusi teknologi yang tepat sasaran dan adaptif terhadap perubahan lingkungan bisnis yang cepat [15], [16].

2.6 Teknik Pengembangan Sistem

2.6.1 Diagram Use-Case

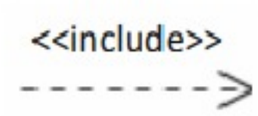
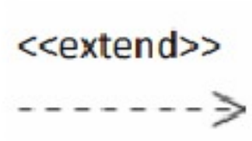
Use case diagram merupakan salah satu instrumen pemodelan yang digunakan pada tahap analisis dalam *System Development Life Cycle* (SDLC) untuk menggambarkan interaksi antara sistem dengan lingkungan eksternalnya. Diagram ini memberikan pandangan fungsional terhadap suatu proses bisnis dengan memperlihatkan bagaimana pengguna memandang proses tersebut, bukan pada mekanisme internal yang menggerakkannya. Secara fundamental, *use case* mewakili cara sistem berinteraksi dengan lingkungannya melalui ilustrasi aktivitas pengguna beserta respons yang diberikan oleh sistem sebagai timbal balik.

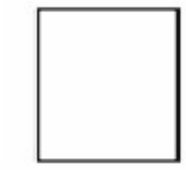
Adapun tujuan utama dari pembuatan *use case diagram* adalah untuk menciptakan sekumpulan *use case* yang mendeskripsikan seluruh tugas yang perlu dilakukan pengguna menggunakan sistem. Operasional sistem dalam pemodelan ini bersifat *event-driven*, yang berarti setiap aktivitas dalam sistem dipandang sebagai respons terhadap suatu pemicu (*trigger event*). Sehingga dapat dikatakan bahwa sistem berada dalam kondisi istirahat (*idle*) saat tidak ada kejadian yang muncul. Begitu sebuah pemicu hadir, sistem akan merespons dengan menjalankan aksi-aksi yang telah didefinisikan dalam *use case*, lalu kembali ke kondisi menunggu.

Pendekatan ini sangat berharga dalam pengembangan aplikasi sistem bisnis dan situs web karena kedua jenis sistem tersebut umumnya melibatkan interaksi pengguna yang intensif. Dengan menerapkan *use case diagram*, pengembang dapat mendefinisikan kebutuhan fungsional secara lengkap dan terstruktur, sehingga perilaku sekumpulan aksi yang dihasilkan benar-benar memberikan nilai nyata bagi pengguna atau pemangku kepentingan [15].

Adapun rincian simbol-simbol dalam *use-case diagram* berserta penjelasannya adalah sebagai berikut:

Tabel 2.1 Simbol Diagram *Use Case*

Simbol	Nama Simbol	Keterangan
	*Actor*	Pihak di luar sistem informasi yang berinteraksi dengan sistem yang dikembangkan.
	*Use-case*	Rangkaian aksi sistem yang menghasilkan keluaran tertentu bagi aktor.
	*Include*	Hubungan *use-case* tambahan yang bergantung pada *use-case* lain untuk berjalan.
	*Extend*	Hubungan *use-case* tambahan yang dapat dijalankan secara mandiri.

	Association	Interaksi antara aktor dan *use-case* dalam suatu proses.
	*System*	Paket yang menampilkan bagian sistem secara terbatas.
	*Dependency*	Hubungan ketergantungan di mana perubahan satu elemen memengaruhi elemen lainnya.
	*Generalization*	Hubungan umum–khusus antara dua *use-case*.
	*Collaboration*	Kerja sama elemen dan aturan yang membentuk perilaku sistem secara menyeluruh.
	*Note*	Elemen fisik yang merepresentasikan sumber daya komputasi saat sistem berjalan.

1. *Actor*: Menspesifikasikan jenis peran yang dimainkan oleh pengguna atau sistem lain saat berinteraksi dengan *use case*.
2. *Use Case*: Deskripsi urutan aksi yang dijalankan oleh sistem sehingga menghasilkan hasil yang terukur bagi aktor.
3. *Include*: Relasi yang menunjukkan bahwa perilaku sebuah *use case* secara eksplisit dimasukkan ke dalam perilaku *use case* lain untuk menjalankan fungsinya. Relasi ini mendukung penggunaan kembali (*reuse*) bagian perilaku yang bersifat umum.
4. *Extend*: Relasi yang menentukan bagaimana perilaku tambahan dapat ditambahkan ke dalam *use case* dasar, biasanya dilakukan secara bersyarat pada titik tertentu.
5. *Association*: Garis penghubung yang menunjukkan interaksi atau komunikasi antara aktor dan *use case*.
6. *System*: Batasan fisik atau logika yang menampilkan paket sistem secara terbatas.
7. *Dependency*: Hubungan di mana perubahan pada satu elemen mandiri (*independent*) akan berdampak pada elemen lain yang bergantung padanya.

8. *Generalization*: Hubungan yang menunjukkan pola umum dan khusus antara dua *use-case* di mana objek anak (*descendent*) mewarisi struktur data dan perilaku dari objek induk (*ancestor*).
9. *Collaboration*: Representasi dari aturan dan elemen yang bekerja sama secara sinergis untuk menyediakan perilaku sistem yang lebih kompleks.
10. *Note*: Elemen pendukung yang mencerminkan sumber daya komputasi atau informasi tambahan yang eksis saat aplikasi dijalankan [17].

Implementasi *use case diagram* sebagai gambaran visual tingkat tinggi memerlukan dokumentasi pelengkap agar fungsionalitas sistem dapat dipahami secara utuh. *Use case diagram* dan narasi *use case* merupakan dua dokumen yang saling melengkapi dan tidak dapat dipisahkan dalam proses analisis sistem. Apabila *use case diagram* berfungsi menjawab pertanyaan mengenai "apa" yang dapat dilakukan sistem beserta pihak yang terlibat, maka narasi *use case* berperan untuk menguraikan secara rinci mengenai "bagaimana" setiap interaksi tersebut berlangsung langkah demi langkah antara aktor dan sistem.

Prinsip mendasar yang menghubungkan kedua dokumen ini adalah konsistensi dokumentasi, di mana setiap satu elips yang terdapat dalam *use case diagram* wajib memiliki satu dokumen narasi *use case* tersendiri sebagai padanannya. Prinsip keselarasan jumlah ini sangat penting untuk menjaga kelengkapan dan ketertelusuran (*traceability*) dokumentasi analisis sistem. Secara umum, sebuah narasi *use case* dalam format elaboratif memuat komponen-komponen terstruktur sebagai berikut:

1. Informasi Dasar: Mencakup nama *use case*, ID, prioritas, aktor, deskripsi singkat, pemicu (*trigger*), serta tipe kejadian baik bersifat eksternal maupun temporal.
2. Kondisi Prasyarat (*Preconditions*): Kondisi-kondisi yang harus terpenuhi sebelum sebuah *use case* dapat dimulai.
3. Alur Normal (*Normal Course*): Urutan langkah interaksi antara aktor dan sistem ketika seluruh proses berjalan lancar tanpa hambatan.
4. Kondisi Pasca-kejadian (*Postconditions*): Mendefinisikan keadaan akhir sistem setelah *use case* selesai dieksekusi.
5. Kondisi Pengecualian (*Exception Conditions*): Menggambarkan kondisi error atau situasi tidak normal yang dapat menghentikan jalannya *use case*.
6. Ringkasan Input dan Output: Tabel yang merekapitulasi seluruh data masuk beserta sumbernya, serta seluruh data keluar beserta tujuannya.

Use Case Name: Select Winning Flight Bid	**ID:** UC-2C	**Priority:** High

Actor: Flight Operations Manager

Description: This use case describes how the Flight Operations Manager finalizes the selection of the winning pilot bid on a flight

Trigger: A flight bid window has closed

Type: ☐ External ☑ Temporal

Preconditions:
1. Flight Operations Manager is authenticated
2. Bidding window on a flight request has closed

Normal Course:	**Information for Steps**
1.0 Select Winning Flight Bid	
1. System posts "closed to bid" message on pilot's dashboard	→ Pilot dashboard update
2. Flight Operations manager requests list of all bids for the flight	→ Flight Bid List Request
3. System displays list of all bids	← Flight Bid records
4. For each bid,	← Pilot's drone details
a. The flight manager verifies the drone's capability to perform the flight	← Flight Request details
b. Bids based on drone without required equipment are marked "Not Qualified," updated, and removed from bid list	→ Updated Flight Bid record
5. System sorts and ranks remaining bids based on flight completion time and bid price	← Qualified Flight Bid records
6. Flight Operations Manager selects bid that optimizes flight completion time and bid price	→ Updated Flight Bid record
7. System changes status of all unselected flight bids	→ Updated Flight Bid records
8. System sends notification message to selected bid pilot	→ Bid Award Notification
9. System posts flight details on selected pilot's dashboard	→ Awarded flight details
10. System notifies all other bidding pilots of the final selection	→ Unsuccessful bid notice
11. System notifies customer that flight has been assigned to a pilot	→ Flight Assigned notice
12. System updates Flight Request with flight assignment details.	→ Updated Flight Request

Postconditions:
1. Pilot of selected bid is notified
2. Selected pilot's dashboard is updated with flight details
3. Pilots' bids not selected are notified
4. Flight bid records selection status is updated
5. Client notified of flight assignment
6. Flight Request record updated with flight assignment

Exceptions:
E1: No bids submitted (occurs at Step 2)
1. Flight operations manager modifies flight proximity area and/or price guidelines
2. Flight operations manager performs Use Case 2A to re-post modified Flight Request
3. If Flight Request has been posted twice with no bids, the Flight Operations manager will contact the client; exit the use case

Summary Inputs	**Source**	**Summary Outputs**	**Destination**
Flight Bid records	Flight Bid datastore	Bidding window closed	Pilot dashboards
Pilot drone details	Pilot drone datastore	Updated Flight Bids	Flight Bid datastore
Flight Request details	Flight Request datastore	Bid Award Notification	Selected Pilot
Qualified Flight Bids	Flight Bid datastore	Awarded Flight details	Selected Pilot dashboard
		Unsuccessful bid notice	Unselected pilots
		Assigned flight notice	Flight Client
		Assigned Flight Request	Flight Request datastore

Gambar 2.2 Contoh Penggunaan Narasi *Use Case*

Secara keseluruhan, sekumpulan narasi *use case* yang terstruktur tidak hanya memperjelas kebutuhan fungsional sistem secara mendalam, tetapi juga membantu tim pengembang dalam mengidentifikasi kasus pengecualian dan kondisi *error* yang sering kali tidak terlihat pada level diagram. Integrasi antara diagram dan dokumentasi tertulis ini membentuk fondasi yang kokoh bagi pengembangan model proses dan model data pada tahap analisis sistem selanjutnya.

2.6.2 Metode PIECES

Metode PIECES merupakan sebuah kerangka kerja (*framework*) yang digunakan untuk melakukan analisis kinerja serta mengevaluasi tingkat kepuasan pengguna terhadap suatu sistem informasi. Kerangka kerja ini sangat krusial dalam mengidentifikasi berbagai masalah yang ada pada sistem saat ini guna memberikan dasar yang kuat bagi pengembangan dan perbaikan sistem di masa mendatang [18]. Analisis PIECES mengevaluasi sistem melalui enam variabel utama, yaitu *Performance* (Kinerja), *Information* (Informasi),

Economics (Ekonomi), *Control and Security* (Kontrol dan Keamanan), *Efficiency* (Efisiensi), dan *Service* (Layanan) [19].

Uraian mengenai keenam variabel dalam metode PIECES adalah sebagai berikut:

1. *Performance* (Kinerja):

   Variabel ini digunakan untuk menilai kemampuan sistem dalam menyelesaikan tugas-tugasnya secara cepat dan tepat. Fokus utamanya adalah pada *throughput*, yaitu jumlah pekerjaan yang dapat diselesaikan dalam waktu tertentu, serta *response time*, yakni durasi yang dibutuhkan sistem untuk merespons permintaan pengguna [18], [19].

2. *Information* (Informasi):

   Evaluasi pada aspek ini berkaitan dengan kualitas luaran sistem. Informasi yang dihasilkan harus memenuhi kriteria relevan, akurat, tepat waktu, dan mudah dipahami agar dapat mendukung proses pengambilan keputusan yang efektif bagi pengguna [18].

3. *Economics* (Ekonomi):

   Aspek ekonomi menilai keterkaitan antara biaya yang dikeluarkan dengan manfaat yang diperoleh dari penggunaan sistem. Analisis ini mempertimbangkan efektivitas biaya operasional serta potensi keuntungan atau penghematan yang dapat dihasilkan oleh sistem bagi organisasi [19].

4. *Control and Security* (Kontrol dan Keamanan):

   Variabel ini berfokus pada mekanisme perlindungan data dan pengendalian sistem. Tujuannya adalah untuk memastikan bahwa sistem memiliki keamanan yang cukup dari akses ilegal atau kerusakan data, serta memiliki kontrol internal yang mampu mendeteksi dan memperbaiki kesalahan dalam pemrosesan data [18], [19].

5. *Efficiency* (Efisiensi):

   Efisiensi mengukur sejauh mana sistem dapat beroperasi dengan menggunakan sumber daya seminimal mungkin tanpa mengurangi fungsionalitasnya. Hal ini mencakup minimalisasi pemborosan waktu, tenaga manusia, maupun sumber daya komputasi lainnya dalam menjalankan prosedur sistem [19].

6. *Service* (Layanan):

   Variabel terakhir ini menilai kualitas dukungan dan kenyamanan yang diberikan kepada pengguna. Layanan yang baik mencakup antarmuka yang ramah pengguna (*user-friendly*), kemudahan dalam mempelajari sistem, serta reliabilitas sistem dalam melayani kebutuhan pengguna secara konsisten [18].

2.6.3 *Blackbox Testing*

Blackbox testing merupakan metode pengujian perangkat lunak yang berfokus pada evaluasi fungsionalitas sistem tanpa harus mengetahui detail struktur internal, desain, atau kode program yang mendasarinya [20]. Pengujian ini dilakukan dengan cara memberikan *input* tertentu ke dalam sistem dan mengamati *output* yang dihasilkan untuk memastikan apakah sistem beroperasi sesuai dengan spesifikasi kebutuhan pengguna (*requirements*) [21]. Fokus utama dari metode ini adalah untuk mengidentifikasi kesalahan fungsional, masalah pada antarmuka, kesalahan dalam struktur data atau akses basis data eksternal, serta kesalahan performa [22].

Dalam konteks siklus pengembangan perangkat lunak, *blackbox testing* memberikan beberapa keuntungan strategis karena memungkinkan penguji melihat sistem dari perspektif pengguna akhir, sehingga pengujian tidak bias oleh pengetahuan teknis mengenai implementasi kode [22]. Selain itu, metode ini sangat efektif untuk mendeteksi ketidakkonsistenan dalam alur logika sistem dari sisi luar [20]. Beberapa teknik yang sering digunakan dalam metode ini antara lain adalah *Equivalence Partitioning*, *Boundary Value Analysis*, dan *Decision Table Testing* [21], [22].

Implementasi *blackbox testing* menjadi krusial untuk menjamin kualitas perangkat lunak sebelum dirilis. Dengan memastikan setiap fungsi pada antarmuka aplikasi berjalan dengan benar, pengembang dapat meminimalisir risiko kegagalan fungsional yang dapat mengganggu pengalaman pengguna [20], [21]. Keunggulan lain dari metode ini adalah fleksibilitasnya yang memungkinkan pengujian dilakukan oleh pihak independen di luar tim pengembang inti, sehingga objektivitas hasil pengujian tetap terjaga [22].