

BAB II TINJAUAN PUSTAKA

1.1 Pengertian Sistem Informasi

Mengumpulkan, memproses, menyimpan, menganalisis, dan berbagi data untuk suatu tujuan adalah bagaimana suatu sistem dikarakterisasi. Suatu sistem informasi, seperti sistem lainnya, memerlukan masukan (informasi, perintah) dan menghasilkan keluaran (laporan, perhitungan) (Hakim dan Anshori, 2019).

1.2 Implementasi Society System di Perumahan

Implementasi Society System di perumahan dapat meningkatkan komunikasi antarwarga serta memperkuat ikatan sosial di antara komunitas. Studi ini memberikan pandangan yang mendalam mengenai pentingnya membangun kerangka kerja yang terintegrasi untuk meningkatkan partisipasi masyarakat dalam kegiatan sosial dan kemanusiaan di dalam perumahan (Wang et al, 2018).

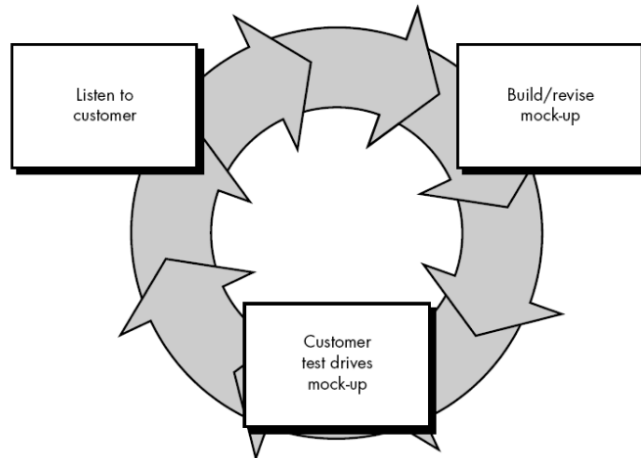
1.3 Metode Pengembang Sistem

Sebagai alur proses pengembangan, pendekatan pengembang sistem memungkinkan terjadinya pengembangan penelitian sesuai dengan tahapan pengembangannya (Rosa dan Shalahuddin, 2019).

1.3.1 *Prototype*

Untuk lebih memahami kebutuhan teknologi dari audiens target, *prototype* adalah metode yang efektif untuk digunakan. Menurut Rosa dan Saladin (2019), pendekatan prototipe dapat membantu pengguna memahami masalah teknologi dan memberikan spesifikasi yang jelas kepada pengembang perangkat lunak tentang kebutuhan yang dimaksudkan.

Mengumpulkan kebutuhan pengguna untuk program yang akan dikembangkan adalah langkah pertama dalam membuat prototipe. Kemudian untuk memudahkan evaluasi pengguna, prototipe dapat dirancang menggunakan model *prototype* dalam bentuk mockup. Agar produk akhir dapat memenuhi harapan dan permintaan khalayak sasaran.



Gambar 2. 1 Metode Prototype
 Sumber : (Rosa dan Shalahuddin, 2019)

1. Kelebihan Prototype

- a. Meminimalisir tenaga dan uang untuk pembangunan
- b. Untuk mengurangi kemungkinan terjadinya masalah sistem sejak awal, pemilik sistem dilibatkan.
- c. Memfasilitasi komunikasi yang lebih baik di antara anggota tim.
- d. Karena mereka memiliki konsep sistem yang akan dikembangkan, klien dapat merasakan kepuasan pribadi.
- e. Karena pelanggan sudah familiar dengan deskripsi sistem, penerapan atau penggunaan sistem menjadi lebih mudah.
- f. Kesederhanaan dalam memproyeksikan peningkatan sistem di masa depan
- g. Memungkinkan pelanggan menyiapkan perangkat lunak mereka untuk pembuatan sistem.

2. Kelemahan Prototype

- a. Jika pelanggan tidak puas dengan tahap awal prototipe, prosesnya bisa memakan waktu lama.
- b. Menjadi lebih sulit untuk membuat sistem karena kebutuhan terus bertambah oleh klien yang menginginkan hal serupa.
- c. Masalah dengan komunikasi dua arah yang efisien akan memperlambat sistem.

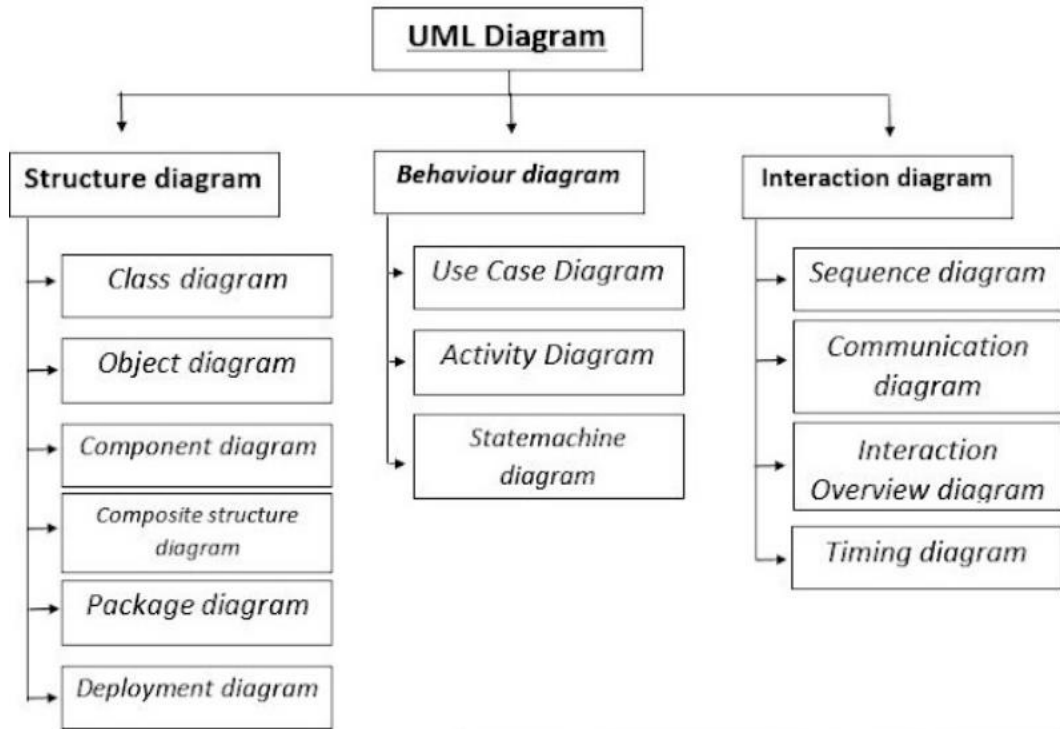
2.3.2 Tahapan Penelitian

Berikut ini adalah proses yang terlibat dalam melakukan penelitian dengan prototype yang dituangkan dalam proses penelitian :

1. Mendengarkan pelanggan
Struktur perangkat lunak, semua persyaratan, dan cetak biru sistem ditentukan bersama oleh klien dan pengembang.
2. Membangun atau memperbaiki mockup
Langkah pertama adalah membuat prototipe, yaitu desain *mockup* yang menampilkan sistem kepada pengguna potensial (seperti desain input dan output).
3. Pelanggan melihat atau menguji mockup
Pengujian sistem yang dibuat untuk mengetahui apakah sudah sesuai dengan fungsi sistem dilanjutkan dengan pengujian penggambaran sistem (seperti maket) pada konsumen untuk mendapatkan permintaan yang sesuai dengan keinginannya.

1.1 *Unified Modelling Language (UML)*

Konsep dalam desain sistem yang memungkinkan representasi visual dari sistem dikenal sebagai alat pengembangan sistem. Metode pengembangan, yang mungkin termasuk penggunaan *Unified Modeling Language*, menentukan alat mana yang perlu disesuaikan. *Unified Modeling Language* adalah bahasa visual untuk merepresentasikan dan mendiskusikan sistem melalui penggunaan diagram dan teks tambahan (Rosa dan Shalahuddin, 2019). Setiap diagram *Unified Modeling Language* didefinisikan dan dijelaskan di bawah ini.



Gambar 2. 2 Bagan UML
 Sumber : (Rosa dan Shalahuddin, 2019)

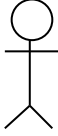

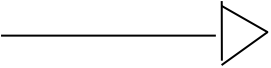
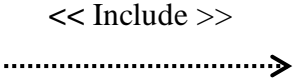
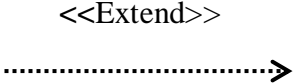
1.1.1 Use Case Diagram

Use Case Diagram adalah skenario yang menggambarkan bagaimana pengguna yang dituju akan terlibat dengan sistem informasi di masa depan. Untuk menentukan pengguna mana yang memiliki izin untuk mengakses bagian mana dari sistem TI, digunakan kasus penggunaan (Rosa dan Salahuddin, 2019). *Use Case Diagram* akan dijelaskan menggunakan simbol-simbol yang tertera pada Tabel 2.1.

Tabel 2. 1 Simbol *Use Case Diagram*

No	Simbol	Deskripsi
1.		<i>Usecase</i> : Kasus penggunaan sering kali dimulai dengan kata kerja untuk menggambarkan operasi sistem sebagai unit yang bertukar pesan dengan unit atau aktor lain.


Tabel 2.1 (Lanjutan)

No	Simbol	Deskripsi
2.		Aktor: sesuatu atau seseorang yang berhubungan dengan benda yang akan dibuat. eksternal ke database. "Biasanya" adalah frase kata benda
3.		Asosiasi(<i>association</i>): terjadi ketika aktor berkomunikasi dengan use case, baik karena use case berinteraksi dengan aktor atau karena aktor terlibat dalam use case.
4.		Generalisasi (<i>generalization</i>) : ada sebagai koneksi (umum - spesifik) antara dua aplikasi, dengan satu aplikasi melayani tujuan yang lebih umum.
5.		Include berarti menunjukkan bahwa <i>use case</i> tambahan akan dipanggil selama eksekusi <i>use case</i> tambahan.
6.		Ekstensi (<i>extend</i>) adalah <i>use case</i> yang memberi nilai tambah pada <i>use case</i> lain dan dapat digunakan secara mandiri.

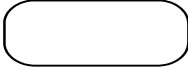
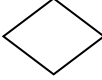

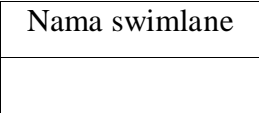

1.1.2 Activity Diagram

Menurut Rosa dan Salahuddin (2019), *activity diagram* adalah cara yang bagus untuk menunjukkan bagaimana suatu sistem atau proses bisnis bekerja. Mereka fokus pada kemampuan sistem daripada tindakan individu pemain. Tabel 2.2 menampilkan simbol-simbol yang akan digunakan untuk menggambarkan diagram aktivitas.

Tabel 2. 2 Simbol *Activity Diagram*

No.	Simbol	Keterangan
1.		Keadaan awal aktivitas sistem, diagram aktivitas mempunyai status keadaan awal.

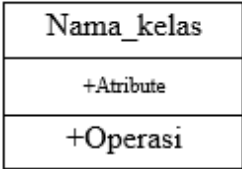
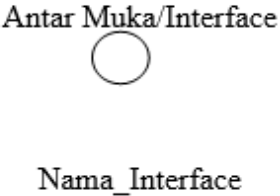
Tabel 2. 2(Lanjutan)

No.	Simbol	Keterangan
2.		Tugas yang dilakukan oleh sistem, yang sering kali dimulai dengan kata kerja.
3.		Ketika lebih dari satu pilihan tindakan disajikan, asosiasi percabangan yang dihasilkan disebut keputusan (<i>decision</i>).
4.		Penggabungan (<i>join</i>) adalah merger yang menggabungkan dua atau lebih aktivitas menjadi satu.
5.		Swimlane Membentuk entitas perusahaan yang berbeda untuk menangani berbagai tugas.
6.		Setiap sistem mempunyai keadaan akhir, dan diagram aktivitas tidak terkecuali.

1.1.3 Class Diagram

Class diagram membuat rencana arsitektur sistem dengan menguraikan kelas-kelas yang akan digunakan untuk membangunnya (Rosa dan Salahuddin, 2019). Untuk memahami Diagram Kelas, lihat Tabel 2.3 untuk simbol-simbol yang akan digunakan.

Tabel 2. 3 Simbol *Class Diagram*

No.	Simbol	Deskripsi
1.		Kelas pada struktur sistem.
2.		Sama dengan konsep interface dalam pemrograman berorientasi objek.

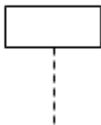

Tabel 2.3 (Lanjutan)

No.	Simbol	Deskripsi
3.	Asosiasi / Asociation 	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan <i>symbol</i>
4.	Asosiasi Berarah / <i>Digunakan Association</i> 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan <i>symbol</i> .
5.	Generalisasi 	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum khusus)
6.	Ketergantungan / dependency 	Relasi antar kelas dengan makna ketergantungan antar kelas.
7.	Agregasi / <i>aggregation</i> 	Relasi antar kelas dengan makna semua bagian (<i>whole-part</i>)


1.1.4 *Sequence Diagram*

Diagram *Sequence* menggambarkan langkah-langkah yang terlibat dalam melaksanakan suatu operasi, pesan yang dikirim, dan waktu pelaksanaannya (Rosa A.S. dan Shalahuddin, 2019). Berikut simbol *sequence diagram* pada Tabel 2.4.

Tabel 2. 4 Simbol *Sequence Diagram*

No	Simbol	Deskripsi
1.	<i>Object lifeline</i> 	Menggambarkan panjang kehidupan suatu objek selama scenario sedang di buat
2.	<i>Activation</i> 	Dimana proses sedang dilakukan oleh <i>object</i> atau <i>class</i> untuk memenuhi pesan atau perintah

Tabel 2.4 (Lanjutan)

No	Simbol	Deskripsi
3.	<p style="text-align: center;"><i>Message</i></p> 	Sebuah anak panah yang mengindikasikan pesan diantara objek. Dan objek dapat mengirimkan pesan ke dirinya sendiri

1.2 Node JS

Node.js adalah perangkat lunak yang dibangun dalam sintaks bahasa pemrograman JavaScript dan dimaksudkan untuk digunakan dalam membuat aplikasi yang berjalan di web. Jika selama ini JavaScript hanya dikenal sebagai bahasa sisi klien, *Node.js* mengisi kekosongan tersebut dengan menjadikannya bahasa sisi server juga, sehingga bersaing dengan PHP, Ruby, Perl, dan lainnya (Node.js.org).

1.3 TypeScript

TypeScript adalah bahasa untuk mengembangkan aplikasi Javascript modern. Untuk menulis Javascript yang ringkas dan mudah dipahami, Anda dapat menggunakan bahasa yang dikompilasi secara statis ini. Jika browser Anda mendukung ECMAScript atau versi yang lebih baru, Anda dapat menjalankannya di NodeJS. TypeScript menawarkan kelas, antarmuka, dan pengetikan statis opsional.

TypeScript berguna untuk lebih dari sekedar membangun aplikasi Angular. Selain Angular, TypeScript juga menemukan jalannya ke kerangka kerja dan perpustakaan lain. Karena TypeScript dapat mendeteksi kesalahan umum yang disebabkan oleh penulis JavaScript karena kebebasan menulis kode JavaScript, hal ini sangat bermanfaat untuk merancang sistem ini. (typescriptlang.org)

1.4 Pengujian *Black Box Testing*

Black box testing khususnya, mengevaluasi perangkat lunak berdasarkan persyaratan fungsionalnya daripada desain dan kode sumbernya. Menurut Rosa dan Salahuddin (2019), tujuan penilaian perangkat lunak adalah untuk menentukan apakah input, output, dan fungsionalitas program memenuhi persyaratan yang ditentukan.

Perangkat lunak diuji dengan membuat kasus uji yang menjalankan semua fiturnya untuk menentukan apakah memenuhi spesifikasi. Penting untuk menggunakan benar dan salah dalam kasus uji, seperti proses *login* “Jika user memasukan *username* dan *password* yang benar maka dapat *login* ?”.