BAB 2
TINJAUAN PUSTAKA

Pada bab ini, secara garis besar akan dijelaskan pengertian-pengertian dan konsep-konsep yang akan digunakan dalam perancangan sistem yang dibuat dalam tugas akhir ini.

2.1 Profil Perusahaan

Circle Trust Travel adalah Perusahaan yang bergerak di bidang jasa Travel yang melayani dan memfasilitasi perjalanan bisnis perorangan ataupun perusahaan untuk menunjang aktivitas bisnis Jasa Travel yang melayani dan memfasilitasi perjalanan dan membantu *Cashflow* perusahaan dengan pembayaran berjangka.

Circle Trust Travel memiliki Visi dan Misi dalam pengembangan perusahaan. Ada pun Visi Perusahaan yaitu Menjadi penyedia jasa Travel yang paling menarik bagi perorangan atau perusahaan - perusahaan di Indonesia dengan kualitas pelayanan terbaik. Sedangkan misi dari perusahaan ini dibagi menjadi beberapa, yaitu:

1. Meningkatkan tata kelola manajemen dan sistem informasi perjalanan melalui ketersediaan sumber daya yang memadai.
2. Meningkatkan pelayanan *client* dengan kualitas mutu yang memuaskan.
3. Menjalin hubungan baik dengan *client*.

2.2 Landasan Teori

2.2.1 Teori Dasar Analisis

Analisis adalah kajian yang dilaksanakan terhadap sebuah bahasa guna meneliti struktur bahasa tersebut secara mendalam. Salah satu tujuan dari analisis adalah untuk benar-benar mengerti mengenai hal-hal yang diperlukan oleh sistem dan mengembangkan sebuah sistem yang mampu memenuhi semua persyaratan tersebut[3].

2.2.2 Jenis Persyaratan Sistem

Pada dokumen persyaratan sistem, dikenal dua macam persyaratan, yaitu fungsional (*functional requirement*), dan persyaratan non-fungsional (*nonfunctional requirement*), yaitu sebagai berikut:

a. Persyaratan Fungsional (*Functional Requirement*).

Persyaratan fungsional ini berhubungan dengan proses yang harus dikerjakan atau informasi yang harus dimuat oleh sistem. Persyaratan fungsional ini merupakan fungsi dasar dari sebuah sistem. Dapat dikatakan, jika persyaratan fungsional dari sistem tidak berfungsi, maka hilanglah manfaat dari sistem[3].

b. Persyaratan Non-Fungsional (*Nonfunctional Requirement*).

Persyaratan non-fungsional mengacu pada perilaku yang harus dimiliki oleh sistem, seperti: kinerja dan kemudahan penggunaan. Berbeda dengan persyaratan fungsional, persyaratan non-fungsional ini diperhatikan jika fungsi dasar dari sistem sudah terpenuhi. Yang menjadi tinjauan dari persyaratan non-fungsional ini adalah dalam hal kualitas.

2.2.3 HCI (*Human Computer Interaction*)

Human Computer Interaction atau Interaksi Manusia dengan Komputer merupakan ilmu yang mempelajari tentang perencanaan dan desain tentang bagaimana pengguna dan komputer dapat bekerja sama sehingga kebutuhan pengguna dapat terpenuhi dengan cara yang paling efektif.[4]

1. UI (*User Interface*)

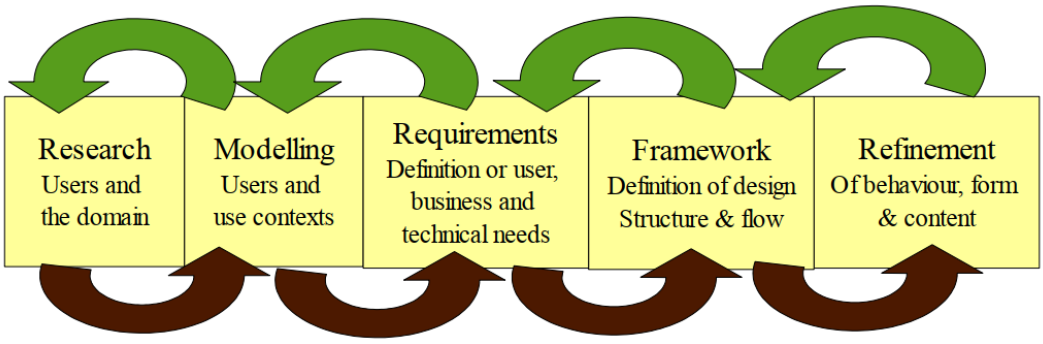
Antarmuka pengguna (*User Interface*) merupakan bagian dari komputer dan perangkat lunaknya yang dapat dilihat, didengar, disentuh, dan diajak bicara, baik secara langsung maupun dengan proses pemahaman tertentu.

2. UX (*User Experience*)

User Experience (UX) yang merupakan proses meningkatkan kepuasan pengguna website atau aplikasi tertentu melalui kegunaan dan kesenangan yang diberikan dalam interaksi antara pengguna dan produk. UX Design inilah yang memungkinkan suatu website bisa digunakan dengan mudah, sehingga tidak membingungkan pengguna. UX mencakup keseluruhan elemen dari suatu website. Termasuk di dalamnya, memastikan website tersusun dengan baik, dan pengguna mudah berpindah dari satu halaman ke halaman lain.

2.2.4 *Goal Directed Design*

Goal Directed Design adalah metode desain interaksi yang dibuat untuk mengidentifikasi tujuan dan sikap pengguna, tujuan sebuah bisnis, serta menerjemahkannya langsung menjadi desain. *Goal Directed Design* memiliki beberapa tahapan sebagai berikut:



Gambar 2.1 Tahapan Goal Directed Design

1. *Research*

Pada tahapan ini bertujuan untuk mengumpulkan data dengan cara observasi, interview, dan cara pengumpulan data lain untuk mendapatkan data kualitatif tentang calon pengguna atau pengguna sesungguhnya dari sebuah produk.

a. Menentukan *Scope*

Menentukan *Scope* adalah tahapan untuk menentukan tujuan dari proyek yang akan dibangun dalam penggunaan metode *Goal Directed Design* [5].

b. Audit (Evaluasi Produk)

Audit merupakan tahapan untuk meninjau produk yang sedang berjalan untuk mengetahui letak kekurangan yang perlu diperbaiki. Untuk mengetahui apakah produk yang sedang berjalan masih memiliki kekurangan adalah dengan menggunakan *Usability Testing* [5].

1. *Usability Testing*

Usability Testing adalah kegiatan dimana seseorang mencoba untuk menggunakan sesuatu (baik itu sebuah *website*, prototipe, atau sketsa dari sebuah desain yang baru) untuk mengerjakan tugas tertentu, lalu memperbaiki sesuatu yang membingungkan atau memusingkan *User*[6].

Usability Testing mengurangi risiko membangun hal yang salah. Menghemat uang, waktu, dan sumber daya berharga lainnya. Itu menemukan masalah ketika mereka masih mudah dan murah untuk memperbaikinya. *Usability Testing* adalah proses berulang, ini bukanlah proses yang dapat diselesaikan dalam sekali percobaan, penguji perlu mengulangi proses sampai desain tidak membingungkan lagi dan pengguna dapat mencapai skenario yang diusulkan [7].

Tujuan dari *usability testing* adalah mencari permasalahan yang berkaitan dengan kegunaan, mengumpulkan data kualitatif dan kuantitatif, serta menentukan kepuasan pengguna dengan produk tersebut. *Usability testing* dapat dikelompokkan berdasarkan uji kegunaan seperti *formative testing* dan *summative testing* Dalam penelitian ini, *tools* yang digunakan dalam melakukan *Usability Testing* adalah dengan menggunakan *Thinking Aloud*.

i. *Tools Thinking Aloud*

Thinking Aloud merupakan metode pengujian berbasis pengguna yang melibatkan *End User* untuk melakukan verbalisasi secara kontinyu terhadap apa yang dipikirkan saat menggunakan sistem. Dengan melakukan verbalisasi, memungkinkan pengamat untuk menginterpretasikan pada bagian antarmuka mana yang memiliki masalah. Pada saat pengguna melakukan verbalisasi, seluruh komentar direkam, sehingga semua yang dipikirkan oleh pengguna dapat ditangkap dan poin – poin penting tidak terlewat pada saat proses analisis.

Tools ini memiliki sejumlah keunggulan. Yang paling penting, *Tools* ini memungkinkan penguji menemukan apa yang benar-benar dipikirkan pengguna tentang produk yang sedang diuji. Secara khusus, penguji mendengar kesalahpahaman mereka, yang biasanya berubah menjadi rekomendasi desain ulang yang dapat ditindaklanjuti: ketika pengguna salah mengartikan elemen desain, pihak pengembang perlu mengubahnya. Bahkan lebih baik, pihak pengembang biasanya belajar mengapa pengguna

salah menebak tentang beberapa bagian UI dan mengapa mereka menemukan yang lain mudah digunakan.

Adapun beberapa keuntungan lain dari *Tools Thinking Aloud* menurut Jakob Nielsen, yakni :

a. Cheap

Tidak diperlukan peralatan khusus. Penguji cukup duduk di sebelah pengguna dan membuat catatan saat pengguna berbicara. Diperlukan sekitar satu hari untuk mengumpulkan data dari segelintir pengguna, yang mana semuanya diperlukan untuk informasi paling penting.

b. Robust

Kebanyakan orang adalah fasilitator yang buruk dan tidak menjalankan studi dengan tepat sesuai dengan metodologi yang tepat. Tetapi, kecuali penguji secara terang-terangan membiasakan pengguna dengan memasukkan kata-kata ke mulut mereka, penguji masih akan mendapatkan temuan yang cukup baik, bahkan dari penelitian yang berjalan buruk. Sebaliknya, studi kegunaan kuantitatif (statistik) sudah matang dengan masalah metodologi dan kesalahan terkecil dapat merusak studi dan membuat temuan langsung menyesatkan. Studi kuantitas juga jauh lebih mahal.

c. Flexible

Penguji dapat menggunakan metode ini pada setiap tahap dalam siklus hidup pengembangan, dari prototipe kertas awal hingga sistem berjalan yang sepenuhnya diimplementasikan. *Thinking Aloud* sangat cocok untuk proyek Agile. Penguji dapat menggunakan metode ini untuk mengevaluasi semua jenis antarmuka pengguna dengan segala bentuk teknologi. Situs web, aplikasi perangkat lunak, intranet, produk konsumen, perangkat lunak perusahaan, desain.

d. Convincing

Pengembang yang paling ulet, perancang yang sombong, dan eksekutif yang keras kepala biasanya melunak ketika mereka mendapatkan paparan langsung tentang bagaimana pelanggan berpikir tentang pekerjaan mereka. Membuat sisa tim Anda (dan manajemen) duduk dalam beberapa sesi dengan berpikir keras tidak membutuhkan banyak waktu dan merupakan cara terbaik untuk memotivasi mereka agar memperhatikan kegunaan.

e. Ease to Learn

Tentunya bahwa *Tools* ini sangat mudah untuk dipelajari.

c. Wawancara

Wawancara merupakan tahapan selanjutnya untuk mengetahui kebutuhan pengguna dalam menggunakan produk yang akan dibuat secara detail.

2. Modelling

Pada fase ini perancang menggunakan bermacam alat untuk mensintesa, membandingkan, membuat prioritas personel dalam skenario tersebut, mengeksplorasi tujuan-tujuan yang berbeda, dan memetakan berbagai perilaku pengguna untuk menghilangkan kesenjangan atau duplikasi.

3. *Requirements*

Fokus pada personal yang sudah ditentukan dalam fase sebelumnya untuk membangun suatu skenario yang didasarkan pada tujuan dan kebutuhan spesifik dari pengguna. Pengguna memberikan gambaran tugas-tugas mana yang penting

dan mengapa, ini akan mengarah kepada desain antarmuka yang minimalis dalam tugas tapi memberikan hasil yang maksimal.

a. Membuat *Problem Statements* dan *Vision Statements*

i. *Problem Statements*

Problem Statements menentukan tujuan inisiatif desain. *Problem Statements* secara ringkas mencerminkan situasi yang perlu diubah, baik untuk persona ataupun bisnis yang menyediakan produk kepada persona.

ii. *Vision Statements*

Vision Statements sendiri merupakan kebalikan dari *Problem Statements* yang berupa pernyataan sebagai acuan untuk merancang desain berdasarkan tujuan pengguna dan bagaimana cara mereka mencapai tujuan tersebut dari desain yang dirancang.

b. *Explore and Brainstorming*

c. *Construct Context Scenario*

Construct Context Scenario tahapan yang menggambarkan pola konteks penggunaan sebuah produk diperlihatkan, yang mencakup pertimbangan lingkungan atau organisasi. Skenario konteks tidak boleh mewakili perilaku sistem pada produk yang sedang berjalan. Skenario ini mewakili produk yang akan dibuat atau dikembangkan

d. *Identifying Requirements*

Identifying Requirements merupakan tahapan yang terdiri dari objek, tindakan, dan konteks untuk mengidentifikasi kebutuhan pengguna dari produk yang akan dibuat atau dikembangkan. Untuk mempermudah dalam mendapatkan yang kebutuhan yang diperlukan, dapat dilakukan dengan cara berikut:

i. *Data Requirements*

Dalam tahap ini, mendefinisikan data yang diperlukan bersama dengan kebutuhan yang akan ditampilkan nantinya pada perancangan *website*

berdasarkan skenario yang dibuat. Sehingga data yang dibutuhkan beserta kebutuhan yang diperlukan untuk mencapai tujuan.

ii. *Functional Requirements*

Tahap ini adalah menentukan fungsionalitas pada sistem (*website*) yang digunakan untuk membantu dalam memperoleh kebutuhan yang diperlukan untuk mencapai tujuan oleh pengguna *website*.

iii. *Contextual Requirements*

Tahap ini memetakan hubungan fungsionalitas pada *website* produk masa depan yang sesuai dengan tujuan (*Goal*) dari *persona*.

4. *Framework*

Pada fase ini perancang menganalisis Interaction Framework dengan menggunakan alat-alat visual.

a. Form Factor

i. *Posture*

ii. *Input Methods*

b. Functional dan Data Elements

c. Menentukan *Functional Groups & Hierarchy*

Pada tahap ini dilakukan suatu pengelompokan elemen fungsionalitas ke dalam satuan unit lalu ditentukan bentuk hierarkinya. Dalam menentukan bentuk hierarkinya, dilakukan dengan menggunakan *Hierarchical Task Analysis* (HTA), dimana HTA merupakan sebuah metode yang didasarkan kepada konsep kinerja dan bagaimana mengatur prinsip kerja. Pada banyak penyelidikan,

metode HTA digunakan untuk meninjau efektifitas kerja dan kegiatan yang tidak tepat dilakukan sehingga dapat diperoleh produktivitas yang diinginkan [6].

i. *Sketch the Interaction Framework*
ii. Anatomi Website
iii. *Wireframing*
iv. Penentuan Warna dan Tipografi

5. *Refinement*

Fase ini melanjutkan fase Framework Definition dengan fokus lebih pada menggabungkan tugas-tugas, menjalani langkah-langkah, dan validasi skenario berdasarkan cerita atau suatu situasi yang ditetapkan.

2.2.5 *Reengineering*

Reengineering adalah pemeriksaan, analisis dan perubahan sistem perangkat lunak yang ada untuk menyusun kembali dalam bentuk baru, dan implementasi selanjutnya dari bentuk baru. Tujuannya adalah untuk memahami perangkat lunak yang ada (spesifikasi, desain, implementasi) dan kemudian mengimplementasikannya kembali untuk meningkatkan fungsionalitas, kinerja, atau implementasi sistem[8].

Reengineering dapat digunakan untuk mengekstraksi informasi-informasi yang ada dalam perangkat lunak. Informasi yang diekstraksi dapat digunakan untuk membangun kembali perangkat lunak ataupun membuat dokumentasi yang lebih ter *update* dan akurat terhadap perangkat lunak.

Saat melakukan *reengineering* ada beberapa faktor yang menjadi penyebab sistem untuk di lakukan *reengineering*. Faktor-faktor yang menyebabkan *reengineering* yaitu sebagai berikut [8]:

a. Hilangnya atau tidak lengkapnya desain/spesifikasi.
b. Kedaluwarsa, tidak sesuai atau hilangnya dokumentasi.
c. Kurang terstruktur nya *Source Code*.
d. Meningkatnya kompleksitas program.
e. Kebutuhan untuk menerjemahkan program ke dalam bahasa program yang berbeda.

2.2.5.1 Proses *Reengineering*

Dalam melakukan *reengineering* pada perangkat lunak memiliki tiga tahapan atau proses utama yang perlu dilakukan. Proses *reengineering* meliputi *reverse engineering, restructuring*, dan *forward engineering*.

a. *Reverse Engineering*

Reverse engineering adalah proses menganalisis sistem subjek untuk mengidentifikasi komponen sistem dan keterkaitannya serta membuat representasi sistem dalam bentuk lain atau pada tingkat abstraksi yang lebih tinggi. *Reverse engineering* dapat digunakan untuk mengekstraksi artefak-artefak yang ada dalam peranti lunak. Artefak-artefak tersebut dapat digunakan untuk membangun kembali peranti lunak ataupun membuat dokumentasi yang *up-to-date* dan akurat terhadap peranti lunak. Tahapan dalam *reverse engineering* terdiri dari beberapa tahapan sebagai berikut:

1. *Implementation* (*code*)
2. *Design*
3. *Requirement*

b. *Forward Engineering*

Forward Engineering adalah sebuah proses pengubahan dari abstraksi level yang paling tinggi (*Requirement*) dan *logic* ke level *design* sampai ke level fisik (Code)dari sistem. Tahapan dalam *forward engineering* terdiri dari beberapa tahapan sebagai berikut:

1. *Requirement*
2. *Design*
3. *Implementation* (*code*)

2.2.6 Konsep Aplikasi Berbasis *Web*

1. Aplikasi

Aplikasi merupakan program siap pakai. Program yang direka untuk melaksanakan suatu fungsi bagi pengguna atau aplikasi yang lain.

2. Web

Web merupakan suatu sistem di *internet* yang memungkinkan siapa pun agar bisa menyediakan informasi. Dengan menggunakan teknologi tersebut, informasi dapat diakses selama 24 jam dalam satu hari. Untuk mengakses informasi yang disediakan *web* ini, diperlukan berbagai perangkat lunak yang disebut dengan *web browser*[9].

3. Aplikasi Berbasis Web

Aplikasi berbasis *web* adalah aplikasi yang dapat dijalankan langsung melalui *web browser* bisa menggunakan *internet* ataupun *intranet* dan tidak tergantung pada sistem operasi yang digunakan. Unsur-unsur dalam *web* adalah sebagai berikut:

a. *Internet*

Internet merupakan kepanjangan dari *Interconnection Networking*. *Internet* merupakan rangkaian jaringan terbesar di dunia dimana semua jaringan yang berada pada semua organisasi dihubungkan dengan suatu jaringan terbesar melalui telepon, satelit dan sistem-sistem komunikasi yang lain sehingga dapat saling berkomunikasi[10].

b. *Web Service*

Web Service merupakan suatu aplikasi yang mendeskripsikan sekumpulan informasi yang dapat diakses dalam sebuah jaringan melalui pesan yang telah distandarkan[10].

c. *Web Browser*

Web browser digunakan untuk memperoleh informasi dengan format *hypertext*. *Web browser* akan mengirimkan *request* ke *web server* dan menampilkan hasilnya ke pengguna[10].

d. *Web* Server

Web Server merupakan program aplikasi yang berjalan di server, berfungsi untuk menjalankan aplikasi web sehingga bisa diakses oleh klien baik melalui jaringan *intranet* maupun *internet*[10].

e. *Web Hosting*

Web Hosting yaitu sebagai ruangan yang terdapat dalam hard disk tempat menyimpan berbagai data, *file-file*, gambar, dan lain-lain yang akan ditampilkan di *website*[10].

4. Tiket.com

Tiket.com adalah situs yang memberikan layanan pemesanan tiket secara *Online*. Tiket.com dapat diakses melalui Komputer desktop dengan menggunakan *browser* seperti Google Chrome, Firefox, Safari. Tiket.com juga dapat diakses dari *Smart phone* dengan mengunduh aplikasinya di Google Play Store bagi pengguna Sistem operasi Android dan di APP Store bagi pengguna sistem operasi iOS (http://tiket.com).[11]

2.2.7 Application Programming Interface (API)

API memungkinkan *developer* untuk mengintegrasikan dua bagian dari aplikasi atau dengan aplikasi yang berbeda secara bersamaan. *API* terdiri dari berbagai elemen seperti *function, protocols, dan tools* lainnya yang memungkinkan *developer* untuk membuat aplikasi. Tujuan penggunaan *API* adalah untuk mempercepat proses *Development* dengan menyediakan *function* secara terpisah sehingga *developer* tidak perlu membuat fitur yang serupa[12]. Penerapan *API* akan sangat terasa jika fitur yang diinginkan sudah sangat kompleks, tentu membutuhkan waktu untuk membuat yang serupa dengannya. API yang akan digunakan adalah API tiket.com, dimana *User* mengirim *Request* berupa URL yang sudah ditentukan oleh tiket.com dan tiket.com merespons dengan memberikan data sesuai dengan *Request* yang dikirim oleh *Developer*, seperti yang ditunjukkan oleh gambar berikut:

2.2.8 Perangkat Lunak Aplikasi

Perangkat lunak aplikasi adalah suat subkelas perangkat lunak computer yang memanfaatkan kemampuan computer langsung untuk melakukan suatu tugas yang diinginkan pengguna. Biasanya dibandingkan dengan perangkat lunak sistem yang mengintegrasikan berbagai kemampuan computer, tapi tidak secara langsung

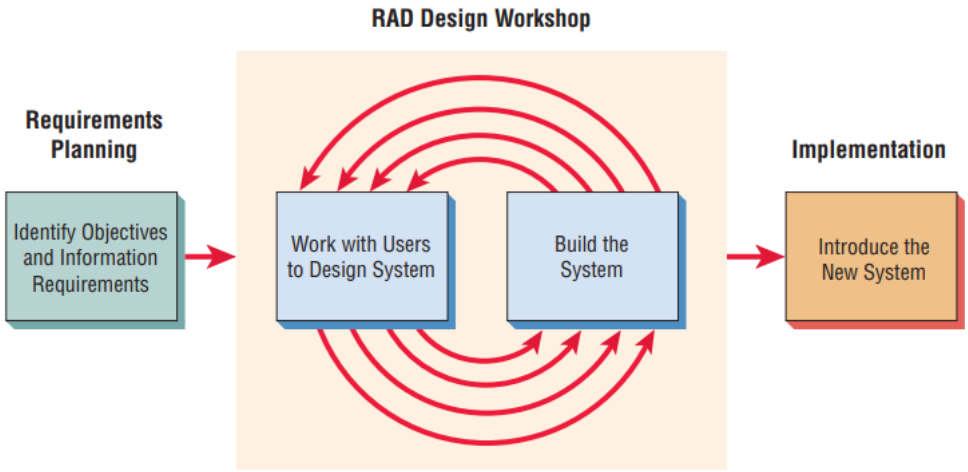
menerapkan kemampuan tersebut untuk mengerjakan suatu tugas yang menguntungkan pengguna.

2.2.9 Basis Data

Basis data adalah kumpulan data yang umumnya menjabarkan aktivitas-aktivitas dari satu atau lebih organisasi yang terkait. Data yang saling berhubungan biasanya ditunjukkan dengan kunci dari tiap file yang ada. Dalam satu file terdapat *record-record* yang sejenis, sama besar, dan sama bentuk yang merupakan satu kumpulan entitas yang seragam.

2.2.10 Metodologi Pengembangan Sistem

Model *Rapid Application Development* adalah metode pengembangan perangkat lunak *Incremental* yang menekankan pada jangka waktu pengembangan yang pendek. Model RAD ini merupakan adaptasi "berkecepatan tinggi" dari model *Waterfall,* dimana kecepatan pengembangan dicapai melalui pendekatan berbasis komponen. Jika kebutuhan dan cakupan sistem dapat dimengerti pada awal pengembangan, model RAD ini memungkinkan pengembangan untuk membuat sistem yang fungsional dalam jangka waktu yang pendek. Tujuan utama dari semua metode pengembangan sistem adalah memberikan suatu sistem yang dapat memenuhi harapan dari para pemakai. Tapi terkadang para pemakai tidak dilibatkan langsung dalam melakukan pengembangan sistem sehingga hal ini menyebabkan sistem informasi yang dibuat jauh dari yang diharapkan. RAD memiliki tahap-tahap adalah sebagai berikut:[13]



Gambar 2.2 Tahapan RAD

Model pengembangan RAD memiliki tiga fase yaitu fase perencanaan syarat-syarat, fase workshop design, dan fase implementasi. Berikut adalah penjelasan masing-masing fase dalam penelitian ini:

1. **Fase Perencanaan Syarat-syarat (*Requirement Planning*)**

   Pada tahap ini dilakukan identifikasi tujuan aplikasi atau sistem, serta untuk mengidentifikasi syarat-syarat informasi yang ditimbulkan dari tujuan-tujuan tersebut.

2. **Fase *Workshop Design***

   Pada tahap ini adalah melakukan proses desain dan melakukan perbaikan-perbaikan apabila masih terdapat ketidak sesuaian desain antara User dan analyst. Untuk tahap ini maka keaktifan User yang terlibat sengat menentukan untuk mencapai tujuan, karena User bisa langsung memberikan komentar apabila terdapat ketidak sesuaian pada desain.

3. **Fase Implementasi (*Implementation*)**

   Setelah desain dari sistem yang akan dibuat sudah disetujui baik itu oleh User dan analyst, maka pada tahap ini programmer mengembangkan desain menjadi suatu program. Hal terpenting adalah keterlibatan User sangat diperlukan supaya sistem yang dikembangkan dapat memberi kepuasan kepada User. Dan pada tahap ini pula dilakukan pengujian sistem, dengan

melakukan pengujian mandiri yang akan dilakukan pengembang dan pengujian yang akan dilakukan oleh User.

2.2.11 *Unified Modelling Language*

UML (*Unified Modelling Language*) adalah salah satu alat bantu yang sangat andal didunia pengembangan sistem. Hal ini disebabkan karena UML menyediakan bahasa pemodelan visual yang memungkinkan bagi pengembangan sistem untuk membuat cetak biru atas visi mereka dalam bentuk baku, mudah dimengerti serta dilengkapi dengan mekanisme efektif untuk berbagi dan mengkomunikasikan rancangan mereka dengan yang lain[3].

Ada 3 (tiga) karakter penting yang melekat di UML, yaitu sketsa, cetak program dan bahasa pemrograman. Sebagai sebuah sketsa, UML bisa berfungsi sebagai jembatan dalam mengkomunikasikan beberapa aspek dari sistem, sehingga semua anggota tim akan memiliki gambaran yang sama tentang suatu sistem. Sebagai cetak biru, UML dapat Member informasi detail tentang *Coding* program dan menginterpretasikannya kembali dalam sebuah diagram. Sedangkan cetak program, UML dapat menerjemahkan diagram yang ada di UML menjadi program yang siap untuk dijalankan[3].

Sebagai sebuah alat bantu modeling dalam suatu pengembangan sistem. UML memiliki beberapa diagram yang mampu membantu pengembang mengkomunikasikan sistem yang akan mereka buat, diagram-diagram tersebut antara lain adalah *Use Case*, *Activity Diagram*, *Class Diagram*, dan *Sequence Diagram*[3].

1. ***Use Case***

Use Case merupakan penjelasan fungsi dari sebuah sistem melalui perspektif pengguna. *Use Case* bekerja dengan cara mendeskripsikan jenis interaksi antara *User* (aktor) dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Urutan langkah-langkah yang menerangkan hubungan antar aktor dengan sistem disebut dengan skenario. Secara singkat, *Use Case* dapat

dikatakan sebagai rangkaian skenario yang digabungkan bersama-sama oleh tujuan umum pengguna[3].

Setidaknya, ada empat aspek dalam diagram *Use Case*, antara lain adalah *Actor*, *Use Case System/Sub system*, *Relationship*, dan *Boundary*.

a. *Actor* merupakan sebuah peran yang bisa dimainkan oleh pengguna dalam interaksinya dengan sistem. Aktor dapat berupa orang, peralatan atau sistem lain yang berinteraksidengan sistem.
b. *Use Case* sistem atau subsistem menjelaskan fungsi interaksi yang dapat dimainkan Actor dalam sebuah sistem.
c. *Relationship* menjelaskan hubungan yang terjadi antara Aktor dengan *Use Case* ataupun menjelaskan hubungan antara suatu *Use Case* dengan *Use Case* lain dalam sebuah sistem. Berikut ini adalah beberapa jenis relasi yang ada pada sebuah diagram *Use Case*.
    a. *Association*

       Association merupakan *relationship* antara aktor dengan *Use Case* dimana terjadi interaksi diantara mereka.
    b. *Extends*

       Extends Use Case merupakan *Use Case* yang terdiri dari langkah yang ter ekstraksi dari *User-User* yang lebih kompleks untuk menyederhanakan masalah dan memperluas fungsinya.
    c. *Uses* (*Include*)

       Hubungan *Use* atau *Includes* menggambarkan bahwa satu *Use Case* seluruhnya meliputi fungsionalitas dari *Use Case* lainnya.
    d. *Depends On*

       Suatu *Use Case* terkadang memiliki ketergantungan dengan *Use Case* lainnya. Ketergantungan ini dimodelkan dengan menggunakan *Depends On Relationship*. Hubungan *Depends On* sangat membantu untuk mengetahui *Use Case* mana yang memiliki ketergantungan pada *Use Case* lainnya yang bertujuan untuk menentukan urutan dalam pengembangan *Use Case*.

e. *Inheritance*

Hubungan *Inheritance* terjadi ketika dua atau lebih aktor menggunakan *Use Case* yang sama.

d. *Boundary* menjelaskan batasan antara *Use Case* dengan aktor.

2. *Activity Diagram*

Activity Diagram merupakan representasi grafis yang memodelkan alur kerja (*Work Flow*) sebuah bisnis dan urutan aktivitas pada suatu proses. Diagram ini dibuat untuk menggambarkan aktivitas dari aktor. Selain itu, diagram ini juga bisa dilakukan untuk mewakili secara grafis aliran kejadian (*Flow Event*) dari suatu *Use Case*[3].

Activity diagram sangat bermanfaat dalam menggambarkan perilaku *pararel* atau menjelaskan bagaimana perilaku dari *Use Case* saling berinteraksi.

3. *Class Diagram*

Class Diagram merupakan representasi sebuah gambar yang memperlihatkan *Attribute* atau *Property* serta operasi yang dimiliki oleh suatu objek dan menggambarkan hubungan dengan objek lainnya. *Class* biasanya digunakan untuk mendefinisikan objek-objek bisnis. *Class* seperti ini biasanya mendefinisikan model *Database* dari suatu aplikasi[3]. Adapun hubungan struktur yang dapat terjadi antara objek dalam suatu kelas diagram meliputi:

a. *Aggregation*

Sebuah *Aggregation* sering dideskripsikan sebagai kelas yang memiliki arti relasi “memiliki”. Hubungan ini menunjukkan bahwa suatu objek dapat di susun dari bagian objek lain. Ini merupakan sebuah relasi yang lemah, sebagai contoh “suatu departemen memiliki sebuah kursus dan kursus untuk sebuah departemen”. Pada kasus ini departemen diperbolehkan untuk mengubah atau menghapus kursus tersebut dalam daftar kepemilikan mereka tetapi kursus tersebut mungkin masih tetap ada [13].

b. *Collection*

Sebuah *collection* dapat dideskripsikan sebagai kelas atau objek yang terdiri dari sejumlah bagian objek lain dan membentuknya sebagai bagian objek yang

utuh. Jenis hubungan ini termasuk dalam kategori hubungan yang lemah dan dapat dideskripsikan seperti hubungan antara perpustakaan dengan buku-buku. Jumlah buku dan katalog dalam perpustakaan bisa saja berubah, namun tetap kedua objek tersebut mempertahankan identitasnya sebagai buku dan perpustakaan [13].

c. *Composition*

Sebuah *composition* dapat diartikan sebagai sebuah hubungan antar objek maupun hubungan antar objek dengan kelas, dimana objek atau kelas yang satu memiliki tanggung jawab terhadap kelas atau objek lainnya. Jenis hubungan ini termasuk dalam katagori hubungan yang kuat, jika suatu objek dihapus maka seluruh objek atau kelas yang terhubungan menjadi *composition* objek akan dihapus pula [13].

d. *Generalization*

Generalization dapat diartikan sebagai sebuah hubungan yang menggambarkan antar jenis umum dari suatu benda atau objek kepada jenis yang lebih spesifik lagi dari benda atau objek tersebut. Jenis hubungan ini sering digambarkan sebagai hubungan “adalah”. Sebagai contoh, mobil “adalah” kendaraan dan truk “adalah” kendaraan. Dalam hal ini, “kendaraan” adalah objek umum mengingat “mobil” dan “truk” adalah objek yang lebih spesifik lagi [13].

4. *Sequence Diagram*

Sequence Diagram digunakan untuk menggambarkan perilaku pada sebuah skenario. Diagram ini menunjukkan sejumlah contoh objek dan *Message* (pesan) yang diletakkan diantara objek-objek ini di dalam *Use Case*. Komponen utama *Sequence Diagram* terdiri atas objek yang dituliskan dalam kotak segiempat bernama. *Message* diwakili oleh garis dengan tanda panah dan waktu yang ditunjukkan dengan progress *Vertical*[3].