BAB II
LANDASAN TEORI

2.1 Sistem Informasi

Menyangkut pemahaman tentang pengertian sistem informasi ini, dalam buku nya yang berjudul Sistem Informasi Konsep dan Aplikasi, (Mulyanto, 2009) mengutipkan beberapa pendapat dari para ahli, diantaranya:

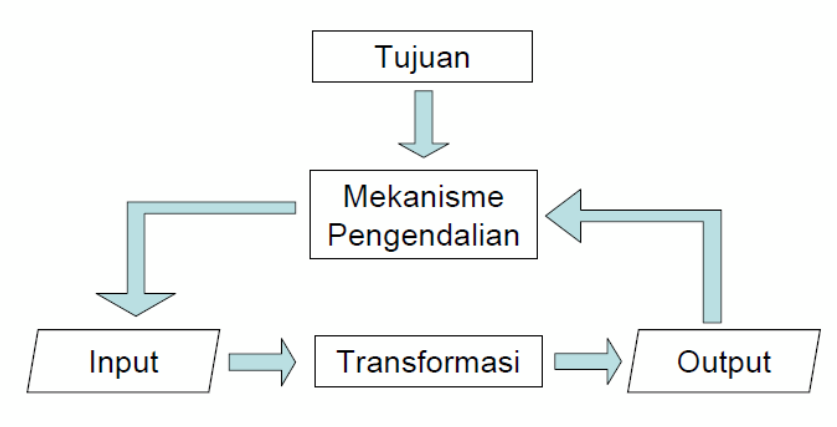
1. Menurut James Alter, sistem informasi adalah “Kombinasi antar prosedur kerja, informasi, orang dan teknologi informasi yang diorganisasikan untuk mencapai tujuan dalam sebuah organisasi.”

2. Menurut Bodnar dan Hopwood, sistem informasi adalah “Kumpulan perangkat keras dan perangkat lunak yang dirancang untuk mentransformasikan data ke dalam bentuk informasi yang berguna.”

3. Menurut Gelinas, Oram dan Wiggins, sistem informasi adalah “Suatu sistem buatan manusia yang secara umum terdiri atas sekumpulan komponen berbasis komputer dan manual yang dibuat untuk menghimpun, menyimpan, dan mengelola data serta menyediakan informasi keluaran kepada pemakai.”

4. Menurut Turban, Mc Lean dan Waterbe, sistem informasi adalah “Sistem yang mengumpulkan, memproses, menyimpan, menganalisis, dan menyebarkan informasi untuk tujuan spesifik.”

5. Menurut Joseph Wilkinson, sistem informasi adalah “Kerangka kerja yang mengkoordinasikan sumber daya (manusia, komputer) untuk mengubah masukan (input) menjadi keluaran (informasi), guna mencapai sasaransasaran perusahaan.”



Gambar 2.1 Karakteristik Sistem

2.2 PT. Tosari Utama

PT. Tosari Utama adalah badan usaha penyedia jasa pengamanan yang berada di Kota Jayapura dan didirikan sejak tahun 2009 oleh Hamid Ambarak. Pengetahuan tentang standar kelayakan jasa keamanan di dapat oleh pendiri saat masih bekerja di Perusahaan penyedia jasa keamanan.

2.3 Website

World Wide Web biasa dikenal dengan *Web* adalah layanan yang menyajikan informasi menggunakan konsep *hyperlink* (tautan) memfasilitasi pekerjaan pengguna Internet (istilah ini merujuk pada pengguna komputer yang menjelajah atau mencari informasi di Internet). Fungsionalitas ini telah menjadikan *web* sebagai layanan yang tubmuh paling cepat. *Web* memungkinkan kita untuk menyorot (menggarisbawahi) kata atau gambar dalam dokumen untuk ditautkan atau diarahkan ke media lain seperti dokumen frasa klip video atau file audio. *Web* dapat menautkan dari mana saja dalam dokumen atau gambar ke mana saja di dokumen lain. Dengan browser yang memiliki *Graphical User Interface* (GUI) tautan afiliasi dapat ditautkan ke tujuan mereka dengan mengarahkan mouse ke tautan dan mengkliknya (Susilo, 2018).

2.3.1. Sifat Website

1. *Website* Statis

Website Statis adalah *website* dimana pengguna tidak dapat merubah kontendari *web* tersebut secara langsung menggunkaan *browser*. Interaksi yang terjadi antara pengguna dan *server* hanyalah seputar pemrosesan link saja. Halaman – halaman *web* tersebut tidak memiliki database, data dan informasi yang ada pada *web* statis tidak berubah – ubah kecuali diubah sintaksnya. Dokumen *web* yang dikirim client akan sama isinya seperti apa yang ada di *web server*.

2. *Website* Dinamis

Dalam *website* dinamis, interaksi yang terjadi antara pengguna dan *server* sangat kompleks. Pengguna *website* dapat mengubah konten dari halaman tertentu dengan menggunakan *browser*. Permintaan dari pengguna dapat diproses oleh *server* yang kemudian ditampilkan dalam isi yang berbeda – beda menurut alur programnya. Halaman – halaman *web* tersebut memiliki database. *Website* dinamis memiliki data dan informasi yang berbeda – beda tergantung *input* yang disampaikan *client*. Dokumen yang sampai di client akan berbeda dengan dokumen yang ada di *web server*. Perbedaan *web statis* dan *web* dinamis (Huizingh, 2000):

a. Interaksi antar pengunjung dan pemilik *web* dalam *web statis* tidak dimungkinkan terjadinya interaksi antar pengunjung dengan pemilik web. Sementara dalam *web* dinamis terdapat interaksi antar pengunjung dengan pemilik *web*.
b. *Web* statis hanya menggunakan satu bahasa *script* yaitu HTML, atau setidaknya ditambahkan dengan CSS. Sedangkan web dinamis menggunakan ahasa pemrograman *web* yang leih kompleks seperti PHP, ASP, serta Javascript.
c. *Website* statis tidak menggunakan database karena tidak ada data untuk disimpan dan diproses. Sedangkan *web* dinamis menggunakan database seperti MYSQL Oracle dll. untuk menyimpan dan mengolah data.

d. Konten *web* statis hanya disediakan oleh pemilik situs *web* dan jarang diupdate, sedangkan konten di *web* dinamis dapat berasal dari pengunjung dan lebih sering di perbarui. Kemudian konten di *web* dinamis dapat diambil dari database sehingga konten mungkin berbeda meskipun kita membuka halaman *web* yang sama

2.4 Gaji

Menurut (Veithzal Rivai Zainal, 2011), "Gaji adalah balas jasa dalam bentuk uang yang diterima karyawan sebagai konsekuensi dari statusnya sebagai seorang karyawan yang memberikan kontribusi dalam mencapai tujuan perusahaan". Sedangkan menurut (Sujarweni, 2015), "Gaji adalah pembayaran atas jasa-jasa yang dilakukan oleh karyawan yang dilakukan perusahaan setiap bulan". Menurut (Suparyadi, 2015), "Gaji adalah uang yang diberikan kepada karyawan secara tetap sebagai balas jasa atas kontribusinya kepada organisasi atau perusahaan, yaitu dengan melakukan pekerjaan yang menjadi tanggung jawabnya". Sedangkan menurut (Hasibuan, 2017), "Gaji adalah balas jasa yang dibayar secara periodik kepada karyawan tetap serta mempunyai jaminan yang pasti. Maksudnya, gaji akan tetap dibayarkan walaupun pekerja tersebut tidak masuk kerja".

2.5 Basis Data

Basis data adalah kumpulan data atau informasi yang menjadi satu dan saling memiliki hubungan dengan data lainnya. Basis data tersimpan secara sistematik di dalam sebuah komputer dan dapat diperiksa menggunakan program dari basis data tersebut (Danaru, 2018). Basis data merupakan komponen utama dalam sebuah system informasi dan system informasi tidak dapat berjalan tanpa adanya basis data. Adapun beberapa istilah terkait basis data, sebagai berikut :

1. *Entity*

   Entitas merupakan objek berupa yang terekam dalam informasi basis data.

2. *Database*

   Basis data merupakan kumpulan data yang saling terhubung dan membentuk bangunan data.

3. *File*

   File adalah kumpulan *record*, elemen, dan atribut yang sama, namun berbeda isi datanya

4. *Record*

   Record adalah kumpulan informasi dalam bentuk elemen atau *field* yang tersusun dalam table berkaitan dengan suatu topik atau kategori.

5. *Field*

   Field adalah sekumpulan nilai yang tersusun dalam suatu tabel yang memiliki tipe data yang sama

6. *Key*

   Key merupakan tanda pengenal untuk mengidentifikasi suatu tabel dalam basis data.

Ada beberapa tipe *key* dalam basis data yang paling sering digunakan, yaitu:

a. *Primary Key*

   Primary Key adalah suatu atribut atau kumpulan atribut dalam sebuah tabel yang memiliki nilai unik untuk setiap baris.

b. *Foreign Key*

   Foreign Key adalah atribut atau kumpulan atribut dalam suatu tabel yang menunjuk pada *Primary Key* dari tabel lain. Fungsi utama *Foreign Key* adalah untuk membangun keterkaitan antar tabel.

c. *Candidate Key*

   Candidate Key adalah salah satu atau sekelompok atribut dalam tabel yang memiliki karakteristik untuk menjadi *Primary Key*. Ini berarti bahwa *Candidate Key* juga memiliki sifat unik dan dapat digunakan untuk mengidentifikasi entitas.

7. *Byte*

   Byte merupakan bagian dari *field* berupa huruf yang membentuk suatu nilai.

8. *Bit*

   Bit adalah satuan terkecil dari secara keseluruhan berupa karakter ASCII

2.5.1 Operasi Dasar Basis Data

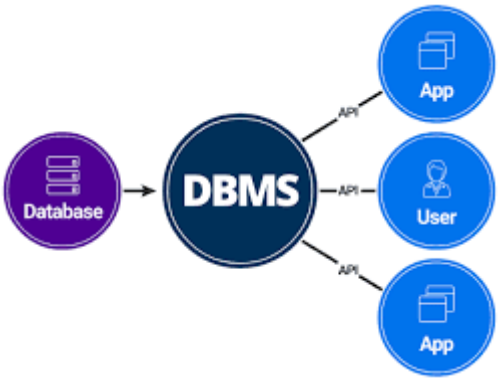
Basis data dapat di kelola dengan cara memberikan perintah. Menambah data, menghapus data, mengambil data, dan membaca data melalui perangkat lunak manajemen basis data (Setiawan, E. 2019). Beberapa perintah dasar dalam pengooperasian basis data meliputi :

1. Pembuatan basis data baru (*Create database*)
2. Penghapusan basis data (*Drop database*)
3. Pembuatan tabel baru dalam suatu basis data (*Create table*)
4. Penghapusan tabel dari suatu basis data (*Drop table*)
5. Penambahan atau pengisian data dalam sebuah tabel dalamuatu basis data (*Insert*)
6. Pengambilan data dari sebuah tabel dalam suatu basis data (*Query*)
7. Mengubah data dari suatu tabel (*Update*)
8. Menghapus data dari sebuah tabel (*Delete*)

2.6 Database Management Systems (DBMS)

Database Management Systems (DBMS) merupakan Perangkat lunak yang digunakan untuk mengelola dan memanggil *query* basis data. DBMS memiliki karakteristik sebagai berikut :

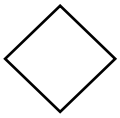
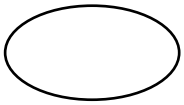
1. Sebuah program perangkat lunak
2. Program tambahan pada system operasi
3. Manajemen data
4. Memanggil data dan menghasilkan laporan
5. Keamanan Data



Gambar 2.2 *DBMS*

2.7 ERD (Entity Relationship Diagram)

Entity Relationship Diagram merupakan diagram yang menggambarkan *hubungan* antara entitas yang relevan dari sebuah *system interest*. *Entity Relationship Diagram* juga merupakan bentuk paling awal dalam memulai sebuah perancangan basis data relasional. Biasanya *Entity Relationship Diagram* dipakai untuk berkomunikasi dengan pemakai eksekutif tingkat tinggi dalam suatu organisasi.

Simbol	Keterangan
	Entitas, yaitu kumpulan dari objek yang dapat diidentifikasikan secara unik.
	Relasi, yaitu hubungan yang terjadi antara satu atau lebih entitas.
	Atribut, yaitu karakteristik dari *entity* atau relasi yang merupakan penjelasan detail tentang entitas.
	Hubungan antara *entity* dengan atributnya dan himpunan entitas dengan himpunan relasinya

2.8 MySQL

MySQL (My Structured Query Language) adalah Sebuah program database server yang mampu menerima dan mengirimkan datanya sangat cepat, multi user serta menggunakan peintah dasar SQL *(Structured Query Language*). MySQL merupakan *Relational Database Management System* (RDBMS) atau sistem bekerja dengan database dengan cara menyimpan data dalam tabel yang berada dan dihubungkan berdasarkan relasinya dengan menghubungkan primary key dan foreign key dengan cepat dan mudah digunakan. serta sudah banyak digunakan untuk berbagai kebutuhan. MySQL dikembangkan oleh MySQL AB Swedia. (Enterprise, 2018). Berikut ini hal-hal yang menyebabkan MySQL menjadi popular:

1. Berlisensi open source, sehingga anda dapat menggunakannya secara gratis.
2. Program yang powerful dan menyediakan fitur yang lengkap.
3. Menggunakan bentuk standar bahasa data SQL.
4. Dapat bekerja dengan banyak sistem operasi dan dengan bahasa- Bahasa pemrograman seperti PHP, PERL, C, C++, JAVA, dan lain- lain.
5. Sangat mudah digunakan dengan PHP untuk pengembangan aplikasi web.
6. Bekerja dengan cepat dan baik, bahkan dengan data set yang banyak.
7. Mendukung banyak database, sampai 50 juta baris atau lebih dalam suatu tabel.
8. Dapat dikostumasikan sesuai keinginan. (Enterprise, 2018).

2.9 PHP

PHP atau yang memiliki kepanjangan Hypertext PreProcessor, adalah suatu bahasa pemrograman yang difungsikan untuk membangun suatu website dinamis. PHP menyatu dalam kode HTMLyang digunakan untuk membangun atau pondasi dari kerangka layout web, sedangkan PHP difungsikan pada bagian prosesnya, sehingga dengan adanya PHP, sebuah web akan sangat mudah di-*maintenance*. (Magdalena & Rachman, 2017)

Hypertext Preprocessor (PHP) adalah sebuah bahasa pemrograman disisi server. Ketika kita mengakses sebuah URL, maka peramban web akan melakukan

request ke sebuah web server. Pengembangan bahasa PHP dimulai pada tahun 1994 oleh Rasmus Lerdorf. Dia mengembangkan perkakas yang digunakan sebagai engine parsing sebagai penterjemah interpreter beberapa macro. Pada saat itu engine digunakan untuk pembuatan buku tamu, counter dan beberapa homepage. Ia menamai engine parser tersebut dengan nama PHP/FI.

2.10 Framework CodeIgniter

CodeIgniter adalah sebuah *framework* dengan basis Bahasa pemrograman PHP yang dapat membantu mempersingkat pengembang web dalam pengembangan aplikasi web berbasis PHP dibanding jika menulis semua kode program dari awal.

Codeigniter berisi kumpulan kode berupa pustaka *(library*) dan alat (*tools*) yang dipadukan sedemikian rupa menjadi suatu kerangka kerja (*framework*). *Codeigniter* adalah *framework* web untuk bahasa pemrograman PHP yang diciptakan dan dikembangkan oleh Rick Ellis pada tahun 2006, penemu dan pendiri EllisLab.

Codeigniter menerapkan pola desain atau arsitektur *Model-View-Controller (MVC)* yang memisahkan bagian kode untuk penanganan proses bisnis dengan bagian kode untuk keperluan tampilan. Dengan menggunakan pola *MVC* , memungkinkan pengembang web untuk mengerjakan aplikasi berbasis web secara bersama (*teamwork*). Dengan demikian pengembang web dapat lebih berfokus pada bagiannya masing-masing tanpa mengganggu bagian yang lain. Sehingga aplikasi yang dibangun akan selesai lebih cepat.

Adapun beberapa keuntungan menggunakan CodeIgniter, diantaranya : (Sahi, 2020)

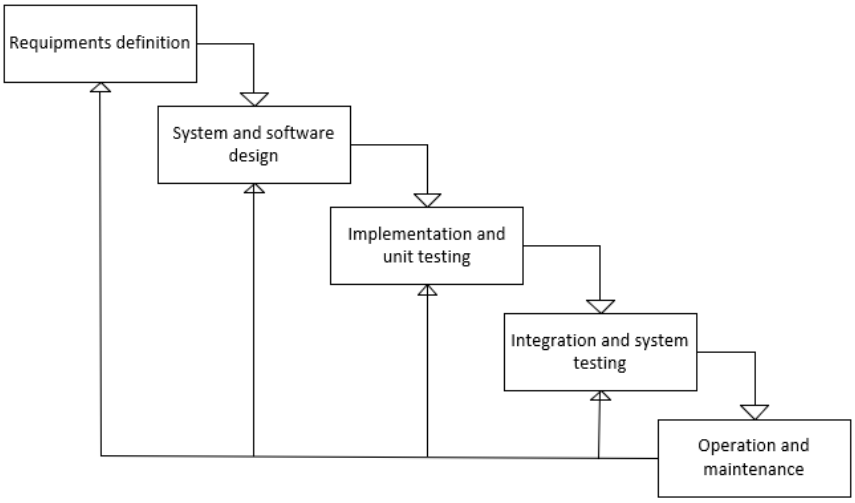
1. Gratis, CodeIgniter berlisensi dibawah Apache/BSD opensorce.
2. Ditulis Menggunakan PHP 4, Meskipun CodeIgniter dapat berjalan di PHP 5, namun sampai saat ini kode program CodeIgniter masih dibuat dengan menggunakan PHP 4.
3. Berukuran Kecil, Ukuran CodeIgniter yang kecil merupakan keunggulan tersendiri. Dibanding dengan *framework* lain yang berukuran besar.

4. Menggunakan Konsep MVC, CodeIgniter menggunakan konsep MVC yang memungkinkan pemisahan *layer application-logic* dan *presentation.*
5. URL yang Sederhana, Secara default, URL yang dihasilkan CodeIgniter sangat bersih dan *Serach Engine Friendly* (SEF).
6. Memiliki Paket *Library* yang Lengkap, CodeIgniter mempunyai *library* yang lengkap untuk mengerjakan operasioperasi yang umum dibutuhkan oleh sebuah aplikasi berbasis web, misalnya mengakses *database*, mengirim email, memvalidasi form, menangani *session* dan sebagainya.

2.11 Model pengembangan system

a. *Waterfall*

Metode *Waterfall* adalah suatu proses pengembangan perangkat lunak bertahap, di mana kemajuan dipandang sebagai terus mengalir ke bawah (seperti air terjun) melewati tahap-tahap perencanaan, pemodelan, implementasi (konstruksi), dan pengujian. Dalam pengembangannya metode *waterfall* memiliki beberapa tahapan yang runtut: *requirement* (analisi kebutuhan), design sistem (*system design*), *coding & testing*, penerapan program, pemeliharaan. (Sonata, 2019)



Gambar 2.3 Metode *Waterfall*

1) *Requirement* (analisis kebutuhan)

Dalam langkah ini merupakan analisa terhadap kebutuhan sistem. Pengumpulan data dalam tahap ini bisa melakukan sebuah penelitian, wawancara atau *study literatur*. Seseorang *system* analisis akan menggali informasi sebanyak-banyaknya dari user sehingga akan tercipta sebuah sistem komputer yang bisa melakukan tugas-tugas yang diinginkan oleh *user* tersebut. Tahapan ini akan menghasilkan *document user requipment* atau bisa dikatakan sebagai data yang berhubungan dengan keinginan user dalam pembuatan sistem. Dokumen inilah yang akan menjadi acuan *system* analisis untuk menerjemahkan kedalam bahasa pemrograman.

2) *Design System* (design sistem)

Proses design akan diterjemahkan sebagai syarat kebutuhan ke sebuah perancangan perangkat lunak yang dapat diperkirakan sebelum dibuat koding. Proses ini terfokus pada: struktur data, arsitektur perangkat lunak, representasi *interface*, dan *detail* (algoritma) *procedural*. Tahapan ini akan mengasilkan dokumen yang disebut *software requipment.* Dokumen inilah yang akan digunakan programmer untuk melakukan aktivitas pembuatan sistemnya.

3) *Coding & Testing* (penulisan sinkode program/*implemention*)

Coding merupakan penerjemahan design dalam bahasa yang bisa dikenali oleh komputer. Dilakukan oleh programmer yang akan menterjemahkan transaksi yang diminta oleh *user*. Tahapan inilah yang merupakan tahapan secara nyata dalam mengerjakan suatu sistem. Dalam artian penggunaan *computer* akan dimaksimalkan dalam tahapan ini. Setelah pengkodean selesai maka akan dilakukan *testing* terhadap sistem yang telah dibuat tadi. Tujuan *testing* adalah menemukan kesalahankesalahan terhadap system tersebut dan kemudian bisa diperbaiki.

4) Penerapan/pengujian program (*integration & testing*)

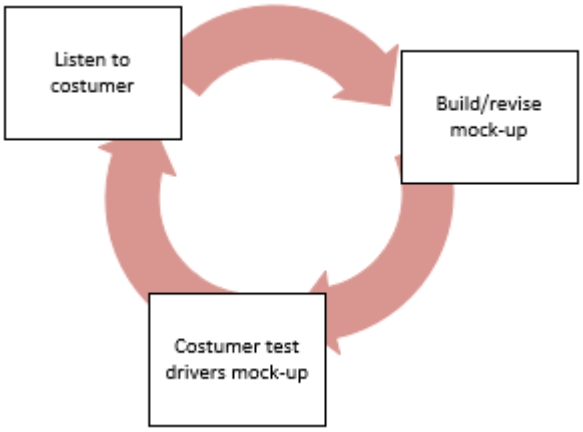
Tahapan ini bisa dikatakan final dalam pembuatan sebuah sistem. Setelah melakukan analisa, *design* dan pengkodean maka sistem yang sudah jadikan digunakan oleh *user*.

5) Pemeliharaan (*operation & maintenance*)

Perangkat lunak yang susah disampaikan kepada pelanggan pasti akan mengalami perubahan. Perubahan tersebut bisa karena mengalami kesalahan karena perangkat lunak harus menyesuaikan dengan lingkungan (*peripheral* atau system operasi baru) baru, atau karena pelanggan membutuhkan perkembangan fungsional.

b. *Prototyping*

Sebuah *prototype* adalah versi awal dari sistem perangkat lunak yang digunakan untuk mendemonstrasikan konsep-konsep, percobaan rancangan, dan menemukan lebih banyak masalah dan solusi yang memungkinkan. Sistem prototype memperbolehkan pengguna untuk mengetahui bagaimana sistem berjalan dengan baik. Penggunaan metode *prototyping* di dalam penelitian ini bertujuan agar peneliti mendapatkan gambaran aplikasi yang akan dibangun melalui tahap pembangunan aplikasi prototype terlebih dahulu yang akan dievaluasi oleh user. Aplikasi *prototype* yang telah dievaluasi oleh user selanjutnya akan dijadikan acuan untuk membuat aplikasi yang dijadikan produk akhir sebagai output dari penelitian ini.



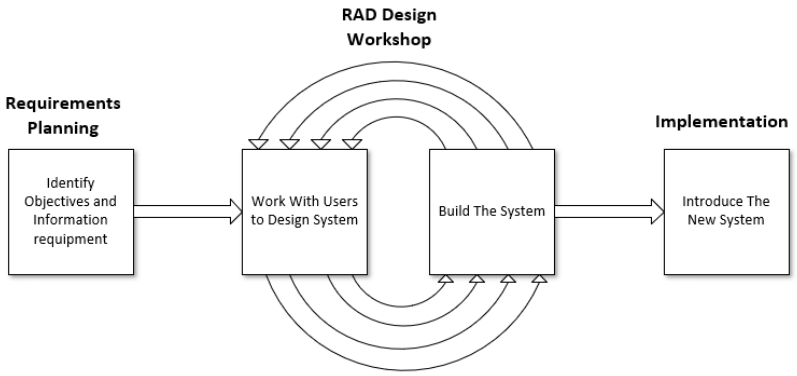
Gambar 2.4 Langkah-langkah *Prototyping*

Gambar 2.4 menjelaskan bahwa metode *prototyping* dimulai dengan mendengarkan kebutuhan dan masukan dari pengguna. Pengembang dan pengguna bertemu dan bersama-sama menentukan tujuan keseluruhan untuk perangkat lunak dan mengidentifikasi apapun persyaratan yang diperlukan. Lalu pengembang membuat sebuah gambaran tentang aplikasi yang selanjutnya dapat dipresentasikan kepada pelanggan. Gambaran tersebut berfokus pada representasi aspek-aspek aplikasi yang akan terlihat oleh pelanggan/pengguna. Beberapa keunggulan dalam menggunakan metode *prototyping*:

- Pengembang sistem dan pengguna saling berkomunikasi khususnya dalam hal penyamaan persepsi terhadap pemodelan sistem yang akan menjadi dasar pengembangan sistem operasionalnya,
- Pelanggan/pengguna ikut terlibat secara aktif dan berpartisipasi dalam menentukan model sistem dan sistem operasionalnya sehingga pelanggan/pengguna akan puas karena sistem yang dibuat sesuai dengan keinginan dan harapannya,
- Sistem yang dibangun memiliki kualitas yang diinginkan karena sesuai dengan kebutuhan yang ada

c. *Rapid Aplication Development (RAD)*

Sebuah proses perkembangan perangkat lunak *sekuensial linier* yang menekankan siklus perkembangan dalam waktu yang singkat. RAD menggunakan metode *iteratif* (berulang) dalam mengembangkan sistem dimana *working model* (model bekerja) sistem dikonstruksikan di awal tahap pengembangan dengan tujuan menetapkan kebutuhan (*requirement*) pengguna dan selanjutnya disingkirkan. Dalam pengembangan sistem informasi normal, memerlukan waktu minimal 180 hari, namun dengan menggunakan metode RAD, sistem dapat diselesaikan dalam waktu 30-90 hari. Model RAD memiliki 3 tahapan sebagai berikut:



Gambar 2.5 Tahapan Model RAD

a. *Requirements Planning*

Secara garis besar dalam fase Rencana Kebutuhan (*Requirement Planning*): *User* dan *analyst* melakukan pertemuan untuk mengidentifikasi tujuan dari sistem dan kebutuhan informasi untuk mencapai tujuan. Tahap Rencana Kebutuhan juga dilakukan pengumpulan dan analisis alat, bahan, sumber daya, biaya dan data Pada tahap ini merupakan hal terpenting yaitu adanya keterlibatan dari kedua belah pihak.

b. *RAD Design Workshop*

Secara garis besar pada tahap ini keaktifan *user* yang terlibat menentukan untuk mencapai tujuan karena pada proses ini melakukan proses desain dan melakukan perbaikanperbaikan apabila masih terdapat ketidaksesuaian desain antara *user* dan *analis*. Seorang *user* dapat langsung memberikan komentar apabila terdapat ketidaksesuaian pada desain, merancang sistem dengan mengacu pada dokumentasi kebutuhan *user* yang sudah dibuat pada tahap sebelumnya. Keluaran dari tahapan ini adalah spesifikasi *software* yang meliputi organisasi sistem secara umum, struktur data dan yang lain

c. *Implementation*

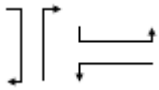
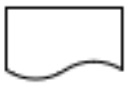
Tahap Implementasi merupakan tugas dari programmer meneruskan dalam bentuk *coding* melalui tinjauan pemrograman berdasarkan

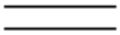
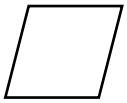
rancangan sistem yang telah dibuat oleh desainer sistem. Tinjauan pemrograman yang dipakai tergantung dari permintaan *user*.

2.12 Flowmap

Flowmap adalah sebuah diagram yang menunjukan aliran data berupa formulir-formulir ataupun keterangan berupa dokumentasi yang mengalir atau beredar dalam suatu sistem. *Flowmap* juga biasa disebut sebagai gambaran secara grafik dari tahapan-tahapan dan urut-urutan prosedur dari suatu program. Berikut adalah simbol-simbol yang terdapat pada *Flowmap*, yaitu: (Maryani, 2016)

Tabel 2. 1 Simbol *Flowmap*

SIMBOL	NAMA	KETERANGAN
	Arah aliran	Menunjukan arah aliran dokumen antar bagian yang terkait pada suatu sistem, baik dari sistem atau keluar sistem
	Dokumen	Menunjukan dokumen *input/output* pada proses manual maupun proses berbasis komputer
	Proses manual	Menunjukan proses yang dilakukan secara manual
	Proses komputer	Menunjukan proses yang dilakukan secara komputerisasi
	Penghubung	Menunjukan aliran dokumen yang terputus atau terpisah pada *flowmap* yang sama
	Penghubung antar *flowmap*	Menunjukan aliran dokumen yang saling berhubungan pada *flowmap* yang berbeda
	Pengarsipan	Menunjukan simpanan data non komputer/informasi berupa file pada proses manual. (Dokumen yang disimpan pada lemari arsip, map dll)

	Penyimpanan magnetic	Media penyimpanan yang dilakukan untuk proses terkomputerisasi
	Kondisi	Keputusan menunjukan pilihan iya atau tidak
	Arsip digital	Menunjukan simpanan data terkomputerisasi
	Catatan	Digunakan untuk menggambarkan catatan akuntansi yang digunakan untuk mencatat data yang direkam sebelumnya di dalam dokumen atau formulir

2.13 Unifed Modelling Language (UML)

UML (*Unified Modeling Language*) merupakan kesatuan struktur dan cara bagi pemodelan desain program berorientasi objek (OOP) serta aplikasinya. UML adalah metodologi untuk mengembangkan sistem OOP dan sekelompok perangkat *tool* untuk mendukung pengembangan sistem tersebut. UML mulai diperkenalkan oleh *Object Management Group*, sebuah organisasi yang telah mengembangkan model, teknologi, dan standar OOP sejak tahun 1980-an. UML adalah suatu bahasa yang digunakan untuk menentukan, memvisualisasikan, membangun, dan mendokumentasikan suatu sistem informasi (Pakaya, Tapate, & Suleman, 2020)

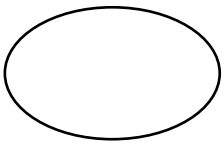
Unified Modeling Language (UML) adalah bahasa umum/standar yang digunakan untuk mendeskripsikan, menspesifikasikan, mendokumentasikan dan membangun perangkat lunak. UML merupakan metodologi dalam mengembangkan suatu sistem berorientasi objek dan sebagai alat untuk mendukung pengembangan sistem. Alat bantu yang digunakan dalam perancangan berbasiskan UML adalah sebagai berikut:

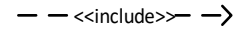
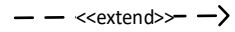
1. *Use Case Diagram*

Use case diagram merupakan pemodelan untuk tindakan atau kelakuan (*behavior*) sistem informasi yang akan dibuat atau dibangun. *Use case* digunakan

untuk mengetahui fungsi yang ada di dalam sistem informasi dan siapa yang memiliki hak menggunakan fungsi-fungsi tersebut

Tabel 2. 2 Simbol *Use Case Diagram*

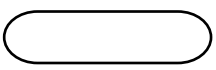
Simbol	Keterangan
	*Use Case* menggambarkan fungsionalitas atau tindakan yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktor, yang dinyatakan dengan menggunakan kata kerja.
	*Actor*/Aktor adalah penggambaran dari orang atau sistem yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan aktor, harus dilakukan pembagian pekerjaan dan tugas- tugas yang berkaitan dengan peran berdasarkan pada target sistem. Orang atau sistem bisa memiliki beberapa peran. Aktor berinteraksi dengan *use case*, tetapi tidak memiliki kendali terhadap *use case.*
.	*Association* berupa asosiasi hubungkan antara aktor dan *usecase*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengindikasikan data.
	Generalization berupa asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengindikasikan bila actor berinteraksi secara pasif dengan sistem.

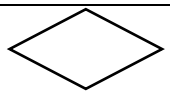
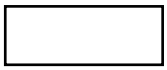
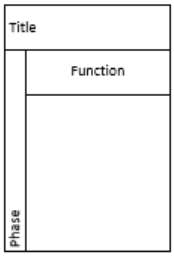
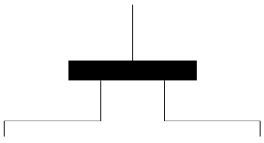
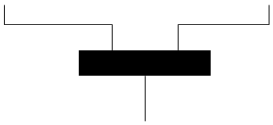
	*Include*, merupakan pemanggilan *use case* oleh *use case* lain menggambarkan relasi *use case* dengan *use case* tambahan dimana *use case* yang tambahan memerlukan *use case* utama untuk menjalankan fungsinya atau sebagai syarat dijalankannya *use case* tambahan.
	*Extend*, merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi, menggambarkan relasi *use case* dengan *use case* turunannya, dimana *use case* utama dapat berdiri sendiri tanpa *use case* tambahan (turunan)

2. Activity Diagram

Diagram aktivitas (*activity diagram*) menggambarkan workflow (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis

Tabel 2. 3 Simbol *Activity Diagram*

Simbol	Keterangan
	*Initial Node/ Start Point*, diletakkan pada pojok kiri atas dan menunjukkan awal aktivitas.
	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.

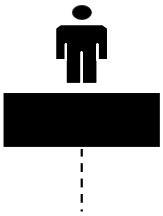
	*Decision Points*, menggambarkan pilihan untuk pengambilan keputusan, *true* atau *false.*
	*Fork node*, menggambarkan pemisahan di mana terdapat lebih dari satu aktivitas dipisahkan menjadi beberapa aktivitas. *Join node* menggambarkan penggabungan di mana terdapat lebih dari satu aktivitas digabungkan menjadi satu.
	*Final Node/ End Point*, akhir aktivitas.
	*Swimlane*, pembagian *activity diagram* untuk menunjukkan siapa pelaku dan apa yang dilakukan.
	*Fork*/ percabangan, digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel atau untuk menggabungkan dua kegiatan paralel menjadi satu.
	*Join* (penggabungan), digunakan untuk menunjukkan adanya dekomposisi.

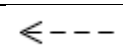
3. Sequence Diagram

Diagram urutan (*sequence diagram*) menunjukkan & menggambarkan tingkah laku suatu objek pada *use case* dengan mendeskripsikan waktu hidup objek

dan pesan yang dikirimkan dan diterima oleh objek (Hendini, 2016).

Tabel 2. 4 Simbol *Sequence Diagram*

Simbol	Keterangan
Aktor 	*Actor Lifeline*, *Actor* menggambarkan objek diluar sistem atau bukan hanya pengguna sistem yang saling berinteraksi dengan sistem yang dibangun atau dikembangkan dan *lifeline* menyatakan kehidupan suatu objek.
Garis hidup/*lifeline* 	*Object*, menyatakan objek yang berinteraksi dengan pesan.
Waktu aktif 	*Activation*, menunjukkan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu yang aktif ini adalah sebuah tahapan yang dilakukan. *Activation*, mewakili sebuah eksekusi operasi dari sebuah objek, panjang kotak ini berbanding lurus dengan waktu/ durasi aktivasi sebuah operasi.
Pesan tipe *call* 	*Self message*, menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri.
Pesan tipe *send*	*Message*, menyatakan bahwa suatu objek mengirimkan pesan berupa data atau informasi ke objek lainnya dengan arah panah menunjukkan arah objek yang dituju.
Pesan tipe *return*	*Return message*, menyatakan bahwa suatu objek yang

	telah menjalankan suatu fungsi atau metode akan menghasilkan suatu pesan kembali ke objek tertentu, arah panah mengarah pada suatu objek yang menerima pesan kembali.

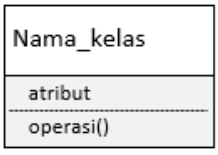
4. Class Diagram

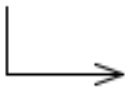
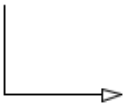
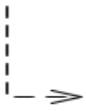
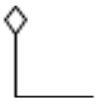
Diagram kelas (*class diagram*) merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan tugas-tugas, aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem.

Class diagram menggambarkan atribut-atribut dan operasi-operasi dari sebuah kelas dan constraint yang berhubungan dengan suatu objek yang dihubungkan atau dikoneksikan.

Class diagram meliputi: kelas (*class*), relasi (*relation*) *association*, generalization dan aggregation, atribut (*attributes*), operasi (*operation/ method*) dan *visibility*, tingkat akses objek eksternal kepada suatu operasi atau atribut. Hubungan antar kelas memiliki keterangan yang biasa disebut dengan *multiplicity* atau *cardinality*.

Tabel 2. 5 Simbol *Class Diagram*

Simbol	Keterangan
Kelas 	Kelas pada struktur sistem.
Asosiasi 	Relasi antar kelas dengan makna umum.

Asosiasi berarah/*directed association* 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain.
Generalisasi 	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum-khusus).
Ketergantungan/*dependency* 	Kebergantungan antar kelas.
Agresi/*aggregation* 	Relasi antar kelas dengan makna semua-bagian (*whole-part*)

2.14 Pengujian Sistem

Pengujian adalah suatu proses pelaksanaan suatu sistem dengan tujuan menemukan suatu kesalahan. Suatu kasus test yang baik adalah apabila test tersebut mempunyai kemungkinan menemukan sebuah kesalahan yang tidak terungkap. Suatu test yang sukses adalah bila test tersebut membongkar suatu kesalahan yang awalnya tidak ditemukan. Salah satu dari jenis pengujian yang ada adalah *Black Box Testing*

2.14.1 *White Box Testing* (Pengujian Kotak Putih)

White-Box Testing (pengujian kotak putih) yaitu menguji perangkat lunak dari segi desain dan kode program apakah mampu menghasilkan fungsi-fungsi, masukan, dan keluaran yang sesuai dengan spesifikasi kebutuhan. Pengujian kotak putih dilakukan dengan memeriksa logik dari kode program. Pegujian mengikuti

stdanar pengujian dari stdanar pemrograman yang seharusnya. Contohnya menguji alur (dengan menelusuri) pengulangan (looping) pada logika pemrograman.

2.14.2 *Black Box Testing* (Pengujian Kotak Hitam)

Black Box, yaitu menguji sistem dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian dimaksudkan untuk mengetahui apakah fungsi–fungsi, masukan, dan keluaran dari sistem sesuai dengan spesifikasi yang dibutuhkan