BAB II
LANDASAN TEORI

2.1 Sistem Informasi

Sesungguhnya yang dimaksud sistem informasi tidak harus melibatkan komputer. Sistem informasi yang menggunakan komputer biasa disebut sistem informasi berbasis komputer (*Computer Based Information System* atau CBIS). Dalam praktik, istilah sistem informasi lebih sering dipakai tanpa embel-embel berbasis komputer, walaupun dalam kenyataannya komputer merupakan bagian yang penting. Di buku ini, yang dimaksudkan dengan sistem informasi adalah sistem informasi berbasis komputer.

Ada beragam definisi sistem informasi, sebagaimana tercantum pada Tabel 2.1. Berdasarkan berbagai definisi tersebut, dapat disimpulkan bahwa sistem informasi mencangkup sejumlah komponen (manusia, komputer, teknologi informasi dan prosedur kerja), ada sesuatu yang diproses (data menjadi informasi) dan dimaksudkan untuk mencapai suatu sasaran atau tujuan (Kadir, 2014).

Tabel 2.1 Definisi Sistem Informasi

Sumber	**Definisi**
Alter (1992)	Sistem informasi adalah kombinasi antar prosedur kerja, informasi, orang dan teknologi informasi yang diorganisasikan untuk mencapai tujuan dalam sebuah organisasi.
Bodnar dan Hopwood (1993)	Sistem informasi adalah kumpulan perangkat keras dan peragkat lunak yang dirancang untuk mentransformasikan data ke dalam bentuk informasi yang berguna.
Gelinas, Oram	Sistem informasi adalah suatu sistem buatan

dan Wiggins (1990)	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 para pemakai.
Hall (2001)	Sistem informasi adalah sebuah rangkaian prosedur formal, dimana data dikelompokkan, diproses menjadi informasi dan didistribusikan kepada para pemakai.
Turban, McLean dan Wetherbe (1999)	Sebuah sistem informasi mengumpulkan, memproses, mentimpan, menganalisis dan menyebarkan informasi untuk tujuam yang spesifik.
Wilkinson (1992)	Sistem informasi adalah kerangka kerja yang mengoordinasikan sumber daya (manusia dan komputer) untuk mengubah masukan (*input*) menjadi keluaran (informasi) guna mencapai sasaran-sasaran perusahaan.

2.2 Pemograman Berorientasi Objek

Metodologi berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai komponen objek yang berisi data dan operasi yang diberlakukan terhadapnya (Rosa, 2011). Metodologi berorientasi objek merupakan suatu cara bagaimana sistem perangkat lunka dibanggun melalui pendekatan objek secara sistematis. Metodologi berorientasi objek didasarkan pada penerapan prinsip-prinsip pengelolaan kompleksitas, yang meliputi rangkaian aktivitas analisis berorientasi objek, perancangan berorientasi objek, pemograman berorientasi objek dan pengujian berorientasi objek. Keuntungan menggunakan metodologi pemograman berorientasi objek adalah sebagai berikut :

a. Meningkatkan produktivitas, karena kelas dan objek yang ditemukan dalam suatu masalah masih dapat dipakai ulang untuk masalah lainnya yang melibatkan objek tersebut (*reusable*).
b. Kecepatan pengembangan, karena sistem yang dibangun baik dan benar pada saat analisis dan perancangan akan menyebabkan berkurangnya kesalahan pada saat pengodean.
c. Kemudahan pemeliharaan, pola-pola yang cenderung tetap dan stabil dapat dipisahkan dan pola-pola yang mungkin sering berubah-ubah.
d. Adanya konsistensi, karena sifat pewarisan dan pengurangan notasi yang sama pada saat analisis, perancangan maupun pengodean.
e. Meningkatkan kualitas perangkat lunak, karena pendekatan pengembangan lebih dekat dengan dunia nyata dan adanya konsistensi pada saat pengembangannya, perangkat lunak yang dihasilkan akan mampu memenuhi kebutuhan pemakai serta mempunyai sedikit kesalahan.

2.3 Konsep Dasar Berorientasi Objek

Dalam rekayasa perangkat lunak, konsep pendekatan berorientasi objek dapat diterapkan pada tahap analisis, perancangan, pemograman dan pengujian perangkat lunak (Rosa, 2011). Beberapa konsep dasar yang harus dipahami tentang metodologi berorientasi objek adalah sebagai berikut :

a. Kelas (*class*)

   Kelas adalah kumpulan objek-objek dengan karakteristik yang sama dan memiliki sifat (atribut). Secara teknis, kelas adalah sebuah struktur tertentu dalam pembuatan perangkat lunak. Kelas merupakan bentuk struktur pada kode program yang menggunakan metodologi berorientasi objek.

b. Objek (*object*)

   Objek adalah abstraksi dan sesuatu yang mewakili dunia nyata seperti benda, manusia, satuan organisasi, tempat, kejadian, struktur dan hal-hal lainnya yang bersifat abstrak. Objek merupakan suatu entitas yang mampu menyimpan informasi (status) dan mempunyai operasi (kelakuan) yang dapat berpengaruh pada status objeknya.

c. Metode (*method*)

Operasi atau metode sebuah kelas hampir sama dengan fungsi atau prosedur pada metodologi struktural. Sebuah kelas boleh memiliki lebih dari satu metode atau operasi.

d. Atribut (*attribute*)

Atribut dapat berupa nilai atau elemen-elemen data yang dimiliki oleh objek dalam kelas.

e. Abstraksi (*abstraction*)

Prinsip untuk merepresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan.

f. Enkapulasi (*encapulation*)

Pembungkusan atribut dan layanan (operasi-operasi) yang dipunyai objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerjanya.

g. Pewarisan (*inheritance*)

Mekasnisme yang memungkinkan satu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dari dirinya.

h. Antarmuka (*interface*)

Antarmuka sangat mirip dengan kelas, akan tetapi tanpa atribut kelas dan memiliki metode yang dideklarasikan tanpa isi. Deklarasi metode pada sebuah *interface* dapat diimplementasikan oleh kelas lain.

i. Reusability

Pemanfaatan kembali objek yang sudah didefinisikan untuk suatu permasalahan pada permasalahan lainnya yang melibatkan objek tersebut.

j. Generalisasi dan Spealisasi

Menunjukkan hubungan antara kelas dan objek yang umum dengan kelas dan objek yang khusus.

k. Komunikasi Antar Objek

Komunikasi antar objek dilakukan lewat pesan yang dikirim dari satu objek ke objek lainnya.

l. Poliformisme (*polymorphism*)

Kemampuan suatu objek untuk digunakan dibanyak tujuan yang berbeda dengan nama yang sama sehingga menghemat baris program.

m. Package

Merupakan sebuah kontainer atau kemasan yang dapat digunakan untuk mengelompokkan kelas-kelas sehingga memungkinkan beberapa kelas yang bernama sama disimpan dalam *package* yang bebeda.

2.4 Analisa dan Desain Berorientasi Objek

2.4.1 Analisa Berorientasi Objek

Analisa berorientasi objek atau *Object Oriented Analysis* (OOA) adalah tahapan untuk menganalisis spesifikasi atau kebutuhan akan sistem yang akan dibangun dengan konsep berorientasi objek. OOA biasanya menggunakan kartu CRC *(Component, Responsibility, Collabolator)* untuk membangun kelas-kelas yang akan digunakan atau menggunakan UML *(Unified Modeling Language)* pada bagian diagram *use case,* diagram kelas dan diagram objek (Rosa, 2011).

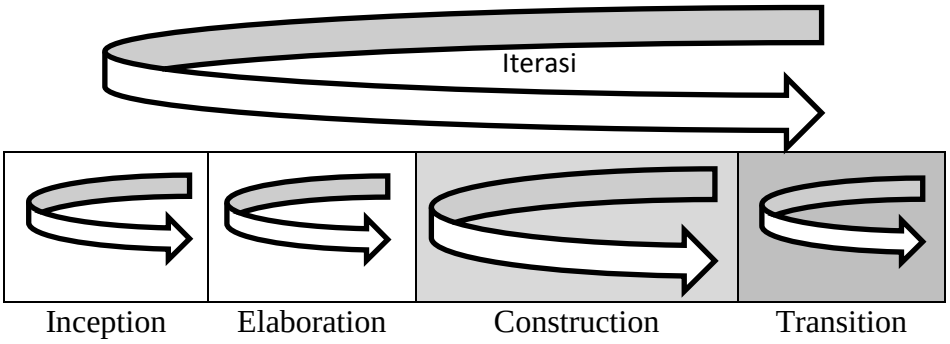
2.4.2 Desain Berorientasi Objek

Desain berorientasi objek atau *Objek Oriented Design* (OOD) adalah tahapan perantara untuk memetakan spesifikasi atau kebutuhan sistem yang akan dibangun dengan konsep berorientasi objek ke desain pemodelan agar lebih mudah diimplementasikan dengan pemograman berorientasi objek. Pemodelan beroientasi objek biasanya dituangkan dalam dokumentasi perangkat lunka yang menggunakan perangkat pemodelan berorientasi objek, diantaranya adalah UML (*Unified Modeling Language*). OOA dan OOD dalam proses yang berulang-ulang seringnya memiliki batasan yang samar, sehingga ke dua tahapan ini sering juga disebut OOAD (*Object Oriented Analysis and Design*) atau dalam bahasa Indonesia berarti Analisis dan desai Berorientasi Objek (Rosa, 2011).

2.5 Metode Pengembangan Perangkat Lunak

Unified Process atau dikenal juga dengan proses iteratif dan inkremental merupakan sebuah proses pengembangan perangkat lunak yang dilakukan secara iteratif (berulang) dan inkremental (bertahap dengan proses menaik). Iteratif bisa dilakukan di dalam setiap tahap atau iteratif tahap pada proses pengembangan perangkat lunak untuk menghasilkan perbaikan fungsi yang inkremental, dimana setiap iterasi akan memperbaiki iterasi berikutnya (Rosa, 2011). Salah satu *Unified Process* yang terkenal adalah RUP (*Rational Unified Process*).

RUP adalah pendekatan pengembangan perangkat lunak yang dilakukan berulang-ulang , fokus pada arsitektur, lebih diarahkan berdasarkan penggunaan kasus (*use case driven*). RUP merupakan proses rekayasa perangkat lunak dengan pendefinisian yang baik dan penstrukturan yang baik. RUP memiliki empat buah tahap fase, yaitu seperti pada Gambar 2.1.



Gambar 2.1 Alur Hidup RUP

a. *Inception* (permulaan)

Tahap ini leih pada memodelkan bisnis yang dibutuhkan dan mendefinisikan kebutuhan akan sistem yang akan dibuat. Tahap yang dibutuhkan pada permulaan ini adalah :

1. Memahami ruang lingkup dari proyek (termasuk biaya, waktu, kebutuhan, resiko dan lainnya).
2. Membangun kasus bisnis yang dibutuhkan.

Hasil yang diharapkan pada tahap ini adalah memenuhi *lifecycle objective milestobe* (batas/tonggak objektif dari siklus) dengan kriteria berikut :

1. Umpan balik dari pendefinisian ruang lingkup, perkiraan biaya dan perkikraan jadwal.
2. Kebutuhan dimengerti dengan pasti dan sejalan dengan kasus primer yang dibutuhkan.
3. Kredibilitas dari perkiraan biaya, perkiraan jadwal, penentuan sekala prioritas, risiko dan proses pengembangan.
4. Ruang lingkup purwarupa (prototype) yang akan dikembangkan.
5. Membangun garis dasar dengan membandingkan perencanaan aktual dengan perencanaan yang direncanakan.

Jika pada akhir tahap ini target yang diinginkan tidak dicapai maka dapat dibatalkan atau diulang kembali setelah dirancang ulang agar kriteria yang diinginkan dapat dicapai.

b. *Elaboration* (perluasan atau perencanaan)

Tahap ini lebih difkuskan pada perencanaan arsitektur sistem. tahap ini lebih pada analisis dan desain sistem serta implementasi sistem yang fokus pada purwarupa sistem (*prototype*). Hasil yang diharapkan pada tahap ini adalah memenuhi *lifecycle objective milestobe* (batas/tonggak objektif dari siklus) dengan kriteria berikut :

1. Model kasus yang digunakan (*use case*) dimana kasus dan aktor yang terlihat telah didefinisikan dan sebagian besar kasus harus dikembangkan.

2. Deskripsi dari arsitektur perangkat lunak telah dibuat.
3. Rancangan aarsitektur yang dapat diimplementasikan dan mengimplementasikan *use case*.
4. Kasus bisnis atau proses bisnis dan daftar resiko yang sudah mengalami perbaikan.
5. Rencana pengembangan untuk seluruh proyek telah dibuat.
6. Purwarupa (*prototype*) yang dapat didemonstrasikan untuk mengurangi setiap resiko teknis yang diidentifikasi.

Jika pada akhir tahap ini target yang diinginkan tidak dicapai, maka dapat dibatalkan atau diulang kembali.

c. *Construction* (konstruksi)

Tahap ini fokus pada pengembangan komponen dan fitur-fitur sistem. tahap ini lebih pada implementasi dan pengujian sistem yang fokus pada implementasi perangkat lunak atau kode program. Tahap ini menghasilkan produk perangkat lunak dimana menjadi syarat dari *Initial Operational Capability Milestone* atau batas/tonggak kemampuan operasional awal.

d. *Transition* (transisi)

Tahap ini lebih pada *deployment* atau inisialisasi sistem agar dapat dimengerti *oleh user*. Tahap ini menghasilkan prosuk perangkat lunak dimana menjadi syarat dari *Initial Operational Capability Milestone*atau batas/tonggak kemampuan operasional awal. Aktivitas pada tahap ini termasuk pada pelatihan *user*, pemeliharaan dan pengujian sistem.

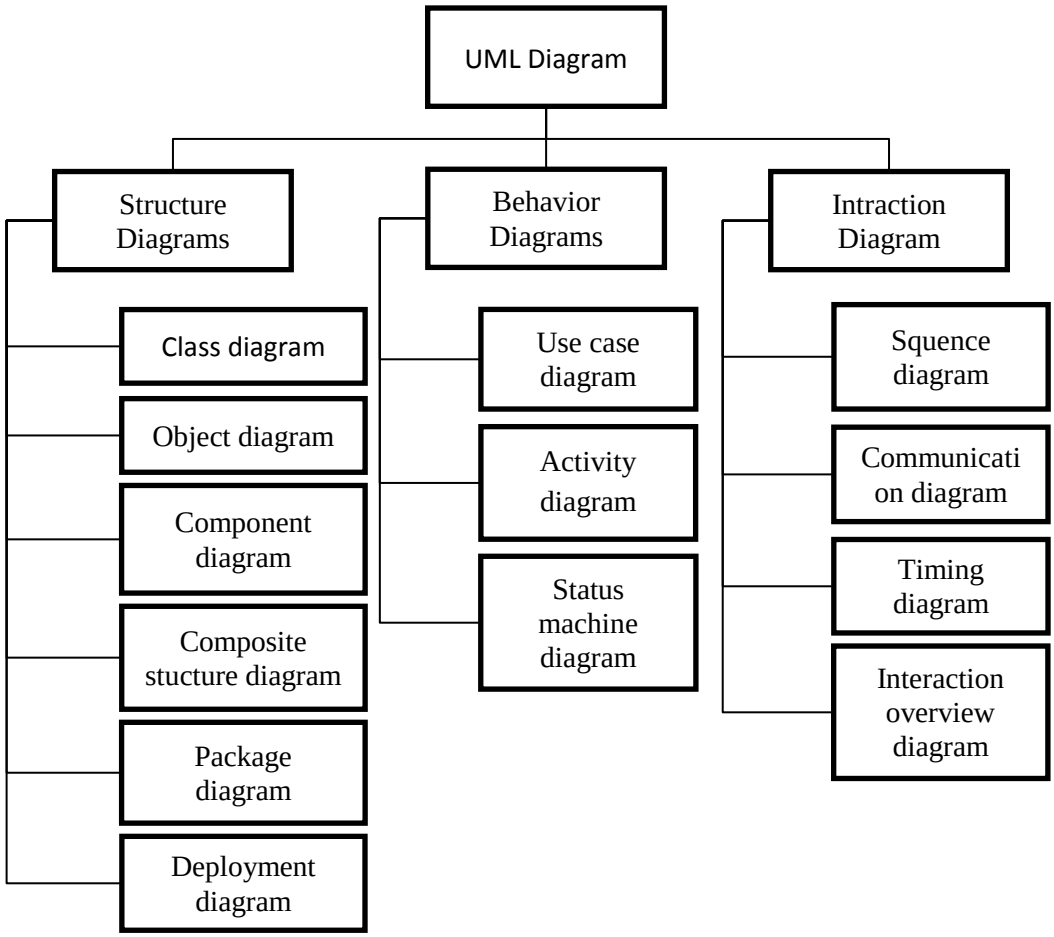
2.6 Alat Bantu Perancangan Sistem

2.6.1 UML

Banyak orang yang telah membuat bahasa pemodelan pembangunan perangkat lunak yang sesuai dengan teknologi pemograman yang berkembangpada saat itu, misalnya yang sempat berkembang dan diguakan oleh banyak pihak adalah *Data*

Flow Diagram (DFD) untuk memodelkan perangkat lunak yang menggunakan pemograman prosedural ataustruktural, kemudian juga ada *State Transition Diagram* (STD) yang digunakan untuk memodelkan sistem *real time* (wakyu nyata).

Pada perkembangan teknik pemograman berorientasi objek, munculah sebuah standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakan teknik pemograman berorientasi objek, yaitu *Unified Modeling Language* (UML). UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun dan mendokumentasi dari sistem perangkat lunak. UML terdiri dari 13 macam diagram yang dikelompokkan dalam tiga kategori, yaitu seperti pada Gambar 2.2 (Rosa, 2011).



Gambar 2.2 Diagram UML

Penjelasan dari pembagian kategori tersebut adalah :

a. *Structure diagram*, yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
b. *Behavior diagram*, yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.
c. *Interaction diagram*, yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antar sub sistem pada suatu sistem.

2.6.1.1 Use Case

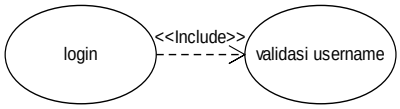
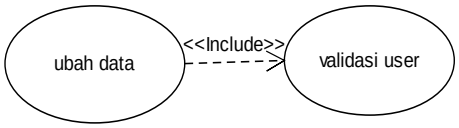
Use case mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu. Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami (Rosa, 2011). Ada dua hal utama pada *use case* yaitu pendefinisian apa yang dibuat aktor dan *use case*.

a. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi, walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
b. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

Tabel 2.2 Simbol *Use Case* Diagram

No.	Keterangan	Simbol	Deskripsi
1.	*Use Case*		Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja diawal-awal frase nama *use case*
2.	Aktor		Orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar itu sendiri. Jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda diawal frase nama aktor.

3.	Asosiasi	———	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor.
4.	Ekstensi	«extends» ——▷	Relasi use case tambahan ke sebuah *use case*, dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu; mirip dengan prinsip *inheritance* pada pemograman berorientasi objek; biasanya *use case* tambahan memiliki nama depan yang sama dengan use case yang ditambahkan, misal  Arah panah mengarah pada *use case* yang ditambahkan.
5.	Generalisasi	——→	Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya, misalnya :  Arah panah mengarah pada *use case* yang menjadi generalisasinya (umum).
6.	Menggunakan/*include/uses*	<<Include>> - - - - - - >	Ada dua sudut pandang yang cukup besar mengenai *include* di *use case* : a. Include berarti use case yang

		«uses»	ditambahkan akan selalu dipanggil saat use case tambahan dijalankan, misal pada kasus berikut :  b. Include berarti use case yang tambahan akan selalu melakukan pengecekan apakah use case yang ditambahkan telah dijalankan sebelum use case tambahan dijalankan, misal pada kasus berikut :  Ke dua interpretasi di atas dapat dianut salah satu atau keduanya tergantung pada pertimbangan dan interpretasi yang dibutuhkan.

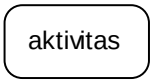
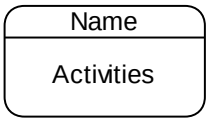
2.6.1.2 Activity Diagram

Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapta dilakukan oleh sistem. Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut :

a. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.

b. Urutan atau pengelompokan tampilan dari sistem/*user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
c. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.

Tabel 2.3 Simbol Diagram Aktivitas

No.	Keterangan	Simbol	Deskripsi
1.	Status awal		Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
2.	Aktivitas		Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
3.	Percabangan		Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
4.	Penggabungan		Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
5.	*Swimlane*		Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.
6.	Status akhir		Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.

2.6.1.3 Class Diagram

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. kelas memiliki apa yang disebut atribut dan metode atau operasi. Atribut merupakan

variabel-variabel yang dimiliki suatu kelas, sedangkan operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas (Rosa, 2011).

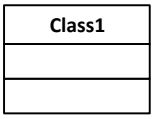
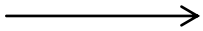
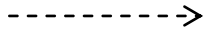
Kelas-kelas yang ada pada struktur sistem, harus dapat melakukan fungsi-fungsi sesuai dengan kebutuhan sistem. Susunan struktur kelas yang baik pada diagram kelas sebaiknya memiliki jenis-jenis kelas berikut :

a. Kelas main
   Kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.
b. Kelas yang menangani tampilan sistem
   Kelas yang mendefinisikan dan mengatur tampilan ke pemakai.
c. Kelas yang diambil dari pendefinisian *use case*
   Kelas yang menangani fungsi-fungsi yang baru ada diambil dari pendefinisian *use case.*
d. Kelas yang diambil dari pendefinisian data
   Kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data.

Jenis-jenis kelas tersebut juga dapat digabungkan satu sama lain sesuai dengan pertimbangan yang dianggap baik asalkan fungsi-fungsi yang sebaiknya ada pada struktur kelas tetap ada. Susunan kelas juga dapat ditambahkan kelas utilitas seperti koneksi ke basis data, membaca *file* teks dan lainnya.

Dalam mengidentifikasikan metode yang ada di dalam kelas perlu memperhatikan apa yang disebut dengan *cohesion* dan *coupling*. *Cohesion* adalah ukuran seberapa dekat keterkaitan instruksi di dalam sebuah metode terkait satu sama lain, sedangkan *coupling* adalah ukuran seberapa dekat keterkaitan instruksi antara metode yang satu dengan metode yang lain dalam sebuah kelas. Sebagai aturan secara umum, maka sebuah metode yang dibuat harus memiliki kadar *cohesion* yang kuat dan kadar *coupling* yang lemah. Simbol-simbol yang ada pada diagram kelas adalah seperti pada Tabel 2.4.

Tabel 2.4 Simbol *Class Diagram*

Simbol	Deskripsi
Kelas 	Kelas pada struktur sistem.
Natarmuka/*interface*	Sama dengan konsep *interface* dalam pemograman berorientasi objek.
Asosiasi	Relasi antar kelas dalam makna umum, asosiasi biasanya juga disertai dengan *multiplicity*.
Asosiasi berarah 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*.
Generalisasi	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum-khusus).
Kebergantungan 	Relasi antar kelas dengan makna kebergantungan antar kelas.
Agregasi 	Relasi antar kelas dengan maknasemua bagian (*whole-part*).

2.6.2 Basis Data

Basis data (*database*) adalah suatu pengorganisasian sekumpulan data yang saling terkait sehingga memudahkan aktivitas untuk memperoleh informasi. Basis data di maksudkan untuk mengatasi problem pada sistem yang memakai pendekatan berbasis berkas.

Untuk mengelola basis data diperlukan perangkat lunak yang disebut *Database Management System* (DBMS). DBMS adalah perangkat lunak sistem yang memungkinkan para pemakai membuat, memelihara, mengontrol, dan mengakses basis data dengan cara yang praktis dan efisien. DBMS dapat

digunakan untuk mengakomodasikan berbagai macam pemakai yang memiliki kebutuhan akses yang berbeda-beda.

Gambar di atas menunjukan bahwa suatu aplikasi dapat berkomunikasi melalui DBMS untuk mengakses basis data dan kemudian membetuk laporan-laporan Selain itu, pemakai juga bisa berinyeraksi secara langsung dengan DBMS untuk mengakses basis data, baik untuk keperluan memintaataupun untuk melakukan perubahan data. Interaksi secara langsung denganbasis data memungkinkan pemakai untuk memperoleh informasi-informasi yang dibutuhkan sewaktu-waktu dan bersifat sementara, tanpa memerlukanbantuan pemrogram.

Umumnya DBMS menyediakan fitur-fitur sebagai berikut :

a. Independensi data program

   Karena basia data ditangani oleh DBMS, program dapat ditulis sehingga tidak tergantung pada stuktur data dalam basis data. Dengan perkataan lain, program tidak akan terpengaruh sekiranya bentuk fisik data diubah.

b. Keamanan

   Keamanan dimaksudkan untuk mencegah pengaksesan data oleh orang yang tidak berwewenang.

c. Integritas

   Hal ini ditujukan untuk menjaga agar data selalu dalam keadaan yang valid dan konsisten.

d. Konkurensi

   Konkurensi memungkinkandata dapat diakses oleh banyak pemakai tanpa menimbulkan masalah.

e. Pemulihan (*recovery*)

   DBMS menyediakan mekanisme untuk mengembalikan basis data ke keadaansemula yang konsisten sekiranya terjadi gangguan perangkat keras atau kegagalan perangkat lunak.

f. Katalog sistem

   Katalog sistem adalah deskripsi tentang data yang terkandung dalam basis data yang dapat diakses oleh pemakai.

g. Perangkat produktivitas

Untuk menyediakan kemudahan bagi pemakai dan meningkatkan produktivitas, DBMS menyediakan sejumlah perangkakat produktivitas seperti pembangkit *query* dan pembangkit laporan.

Komponen-komponen yang menyusun lingkungan DBMS terdiri atas:

a. Perankat keras

   Perangkat keras digunakan untuk menjalankan DBMS beserta aplikasi-aplikasinya. Perangkat keras berupa komputer dan periferal pendukungnya. Komputer dapat berupa PC, minikomputer, mainframe, dan lain-lain.

b. Perangkat lunak

   Komponen perangkat lunak mencakup DBMS itu sendiri, program aplikasi, serta perangkat lunak pendukung untuk komputer dan jaringan. Program aplikasi dapat dibangun dengan menggunakan bahasa pemrograman seperti C++, Pascal, Delphi, atau Visual BASIC.

c. Data

   Bagi sisi pemakai, komponen terpenting dalam DBMS adalah data karena dari data inilah pemakai dapat memperoleh informasi yang sesuai dengan kebutuhan masing-masing.

d. Prosedur

   Prosedur adalah petujuk tertulis yang berisi cara merancang hingga menggunakan basis data. Beberapa hal yang dimasukkan dalam prosedur:

   1. Cara masuk ke DBMS (login).
   2. Cara memekai fasilitas-fasilitas tertentu dalam DBMS maupun cara menggunakan aplikasi.
   3. Cara mengaktifkan dan menhentikan DBMS.
   4. Cara membuat cadangan basis data dan cara mengembalikan cadangan ke DBMS.

e. Orang

   Komponen orang dapat dibagi menjadi tiga kelompok, yaitu :

   1. Pemakai akhir (end-user).
   2. Pemogram aplikasi.
   3. Administrator basis data

Tedapat beberapa elemen basis data, yaitu :

a. Database

Database atau basis data adalah kumpilan tabel yang mempunyai kaitan antara suatu tabel dengan tabel lainya sehingga membentuk suatu bangunan data.

b. Tabel

Tabel adalah kumpulan *record-record* yang mempunyai panjang elemen yang sama dan atribut yang sama namun berbeda data *value*nya.

c. Entitas

Entitas adalah sekumpulan objek yang terdefinisikan yang mempunyai karakteristik sama dan bisa dibedakan satu dengan lainnya. Objek dapat berupa barang, orang, tempat atau suatu kejadian.

d. Atribut

Atribut adalah deskripsi data yang bisa mengidentifikasi entitas yang membedakan entitas tersebut dengan entitas yang lain. Seluruh atribut harus cukup untuk menyatakan identitas objek atau dengan kata lain, kumpulan atribut dari setiap entitas dapat mengidentifikasi keunikan suatu individu.

e. *Data Value* (Nilai Data)

Data value adalah data aktual atau informasi yang disimpan pada tiap data, elemen atau atribut. Atribut nama pegawai menunjukan tempat dimana informasi nama karyawan disimpan, nilai datanya misalnya adalah Anjang, Arif, Suryo dan lain-lain yang merupakan isi data nama pegawai tersebut.

f. File

File adalah kumpulan *record* sejenis yang mempunyai panjang elemen yang sama, atribut yang sama namun berbeda nilai datanya.

g. Record/Tuple

Kumpulan elemen-elemen yang saling berkaitan menginformasikan tentang suatu entitas secara lengkap. Satu *record* mewakili satu data atau informasi.

2.7 Kebutuhan Perangkat Lunak

2.7.1 PHP (*Hypertext Prepocessor*)

PHP yaitu bahasa pemrograman *web server-side* yang bersifat *open source*. PHP merupakan *script* yang terintegrasi dengan HTML dan berada pada *server* (*server side HTML embedded scripting).* PHP adalah *script* yang digunakan untuk membuat halaman *website* yang dinamis. Dinamis berarti halaman yang akan di tampilkan dibuat saat halaman itu diminta oleh *client.* Mekanisme ini menyebabkan informasi yang di terima client selalu yang terbaru/*up to date*. Semua *script* PHP dieksekusi pada *server* dimana *script* tersebut dijalankan (Rudyanto, 2011).

2.7.2 MySQL

MySQL adalah salah satu jenis *database server* yang terkenal dan banyak digunakan untuk membangun aplikasi web yang menggunakan *database* sebagai sumber dan pengelolaan data. Kepopuleran *MySQL* antara lain karena *MySQL* menggunakan *SQL* sebagai bahasa dasarnya untuk mengakses *database*nya sehingga mudah untuk digunakan.

MySQL termasuk RDBMS (*Relational Database Management System*). Pada *MySQL*, sebuah d*atabase* mengandung satu atau sejumlah tabel. Tabel terdiri atas sejumlah kolom dan baris, dimana setiap kolom berisi sekumpulan data, dan baris merupakan sekumpulan data yang saling berkaitan dan membentuk informasi. Kolom biasanya disebut sebagai field dan informasi yang tersimpan dalam setiap baris disebut *record* (Rudyanto, 2011).

2.7.3 Xampp

Xampp adalah sebuah *software* yang berfungsi untuk menjalankan *website* berbasis PHP dan menggunakan pengolahan data *MySQL* di komputer lokal. *Xampp* berperan sebagai *server* web pada kumputer. *Xampp* juga dapat disebut sebuah *Cpanel server virtual*, yang dapat membantu melakukan *preview* sehingga dapat memodifikasi *website* tanpa harus *online* atau terakses dengan internet (Yogi, 2008).

2.7.4 Bootsrap

Bootrstrap adalah sebuah alat yang membantu pengembang *website* dalam mendesain sebuahtampilan *website*. *Bootstrap* juga bisa disebut sebagai *Framework CSS* pembuatan *website* gratis dari *twitter*. *Bootstrap* menggunakan *Lisensi Apache 2.0*, sebuah lisensi terbuka yang membebaskan kita menggunakannya dengan mudah. Kelebihan yang dimiliki *bootstrap* adalan sebagai berikut :

a. Cepat, memiliki banyak *library* yang menyediakan potongan kode yang siap digunakan.
b. Fleksibel, dapat menyesuaikan sesuai kebutuhan *website*.

The Grid, menggunakan *grid* menjadikan pekerjaan menjadi lebih mudah. Penataan sudah diselesaikan dalam mode otomatis. *Bootstrap* dapat berjalan sempurna pada *browser-browser* seperti *Opera, Firefox, Safari, Chrome,* dan *Internet Explorer*. *Bootstrap* diciptakan pada Tahun 2011 oleh Mark Otto dan Jacob Thomton yang menjadi *programmer* di twitter.Pertama kali diluncurkan pada Agustus 2011.*Bootstrap* hanya sebuah proyek berbasis CSS, namun telat berevolusi menjadi sebuah *framework* yang berisi*Javascript Plugin, Icon, Forms* dan *Button* (Ariskevin, 2014).

2.7.5 CodeIgniter

CodeIgniter adalah aplikasi *open source* yang berupa *framework* dengan model MVC (*Model, View, Controller*) untuk membangun *website* dinamis dengan menggunakan PHP. *CodeIgniter* memudahkan *developer* untuk membuat aplikasi web dengan cepat dan mudah dibandingkan dengan membuatnya dari awal. *CodeIgniter* dirilis pertama kali pada 28 Februari 2006.

Framework secara sederhana dapat diartikan kumpulan dari fungsi-fungsi atau prosedur-prosedur dan *class-class* untuk tujuan tertentu yang sudah siap digunakan sehingga bisa lebih mempermudah dan mempercepat pekerjaan

seorang pemrograman, tanpa harus membuat fungsi atau *class* dari awal. Ada beberapa alasan mengapa menggunakan *framework*, yaitu :

a. Mempercepat dan mempermudah pembangunan sebuah aplikasi web.
b. Relatif memudahkan dalam proses maintenance karena sudah ada pola tertentu dalam sebuah *framework* (dengan syarat programmer mengikuti pola standar yang ada).
c. Umumnya *framework* menyediakan fasilitas-fasilitas yang umum dipakai sehingga kita tidak perlu membangun dari awal, misalnya validasi, ORM, *pagination*, *multiple database*, *scaffolding*, pengaturan *session*, *error handling* dan lainnya.
d. Lebih bebas dalam pengembangan jika dibandingkan CMS.

Model *View Controller* merupakan suatu konsep yang cukup populer dalam pembangunan aplikasi web, berawal pada bahasa pemrograman Small Talk, MVC memisahkan pengembangan aplikasi berdasarkan komponen utama yang membangun sebuah aplikasi seperti manipulasi data, user interface dan bagian yang menjadi kontrol aplikasi. Terdapat 3 jenis komponen yang membangun suatu MVC pattern dalam suatu aplikasi yaitu :

a. *View*, merupakan bagian yang menangani *presentation logic*. Pada suatu aplikasi web bagian ini biasanya berupa *file template* HTML yang diatur oleh *controller*. *View* berfungsi untuk menerima dan merepresentasikan data kepada user. Bagian ini tidak memiliki akses langsung terhadap bagian model.
b. Model, biasanya berhubungan langsung dengan *database* untuk memanipulasi data (*insert, update, delete, search*), menangani validasi dari bagian *controller*, namun tidak dapat berhubungan langsung dengan bagian *view*.
c. *Controller*, merupakan bagian yang mengatur hubungan antara bagian model dan bagian view. *Controller* berfungsi untuk menerima request dan data dari *user* kemudian menentukan apa yang akan diproses oleh aplikasi.

Dengan menggunakan prinsip MVC suatu aplikasi dapat dikembangkan sesuai dengan kemampuan developernya, yaitu *programmer* yang menangani bagian model dan *controller*, sedangkan *designer* yang menangani bagian *view*, sehingga penggunaan arsitektur MVC dapat meningkatkan *maintanability* dan organisasi kode. Walaupun demikian dibutuhkan komunikasi yang baik antara *programmer* dan *design*er dalam menangani variabel-variabel yang akan ditampilkan. Ada beberapa kelebihan *CodeIgniter* (CI) dibandingkan dengan *framework* PHP lain, yaitu :

a. Performa sangat cepat

   Salah satu alasan tidak menggunakan *framework* adalah karena eksekusinya yang lebih lambat daripada PHP *from the scracth*, tapi *CodeIgniter* sangat cepat bahkan mungkin bisa dibilang *CodeIgniter* merupakan *framework* yang paling cepat dibanding *framework* yang lain.

b. Konfigurasi yang sangat minim (*nearly zero configuration*)

   Tentu saja untuk menyesuaikan dengan *database* dan keleluasaan *routing* tetap diizinkan melakukan konfigurasi dengan mengubah beberapa *file* konfigurasi seperti *database.php* atau *autoload.php*, namun untuk menggunakan *CodeIgniter* dengan *setting standard*, anda hanya perlu merubah sedikit saja *file* pada *folder config*.

c. Banyak komunitas

   Dengan banyaknya komunitas CI ini, memudahkan kita untuk berinteraksi dengan yang lain, baik itu bertanya atau teknologi terbaru.

d. Dokumentasi yang sangat lengkap

   Setiap paket instalasi codeigniter sudah disertai *user guide* yang sangat bagus dan lengkap untuk dijadikan permulaan, bahasanya pun mudah dipahami (Dewi, 2011).

2.8 Penelitian Terkait

Penelitian yang terkait dengan penelitian yang dilakukan sebelumnya adalah sebagai berikut :

a. Menurut Firzaldy dalam “Rancang Bangun Sistem Informasi Bengkel Mobil Berbasis *Web*” menyimpulkan bahwa aplikasi yang dibuat dapat mendukung bagian gudang untuk membuat bukti pengeluaran barang, membuat daftar pemesanan barang dan mencatat transaksi pembelian barang dengan baik dan sesuai.
b. Menurut Mochammad dalam “Rancang Bangun Sistem Layanan *Service* Mobil Pada Bengkel Tejo Motor Semarang” menyimpulkan bahwa pembuatan aplikasi ini bisa menjadi solusi dalam memperkenalkan dan memudahkan pelayanan penjualan dan perawatan mobil lebih mudah karena dapat dilakukan pada satu bagian saja yaitu bagian administrasi. Memudahkan pimpinan untuk memperoleh berbagai macam laporan yang berhubungan dengan proses penjualan, perawatan mobil karena semua laporan sudah tersedia dalam sistem informasi salon ini dan memudahkan dalam pencarian barang tanpa harus mengecek ke gudang. Membantu pimpinan untuk mengambil keputusan berhubungan dengan penambahan pegawai, pembelian barang dan keputusan lainnya dari laporanlaporan yang dihasilkan sistem yang dapat diperoleh setiap saat.