BAB II

LANDASAN TEORI

II.1. Konsep Dasar

II.1.1. Definisi Sistem

Menurut Sutabri (2012: 10), Sistem adalah suatu kumpulan atau himpunan dari unsur, komponen, atau variabel yang terorganisir, saling berinteraksi, saling tergantung satu sama lain, dan terpadu. Teori sistem secara umum yang pertama kali diuraikan oleh Kenneth Boulding, terutama menekan pentingnya perhatian terhadap setiap bagian yang membentuk sebuah sistem.

Menurut Mulyadi (2010 : 2), Setiap sistem pasti terdiri dari struktur dan proses, struktur sistem merupakan unsur-unsur yang membentuk sistem tersebut, sedangkan proses sistem menjelaskan cara kerja dari setiap unsur sistem tersebut dalam mencapai tujuan sistem. Dan dapat disimpulkan bahwa suatu sistem pada dasarnya adalah sekelompok unsur yang erat berhubungan satu dengan yang lainnya, yang berfungsi bersama-sama untuk mencapai tujuan tertentu (Rosana Junita Sirait, et al., 2015 : 2).

II.1.2. Definisi Informasi

Sumber informasi adalah data. Data merupakan kenyataan yang menggambarkan suatu kejadian serta merupakan suatu bentuk yang masih mentah

yang belum dapat bercerita banyak sehingga perlu diolah lebih lanjut melalui suatu model untuk menghasilkan informasi. Fungsi utama informasi adalah menambah pengetahuan. Informasi yang disampaikan kepada pemakai mungkin merupakan hasil data yang sudah diolah menjadi sebuah keputusan (Rosana Junita Sirait, et al., 2015 : 2).

Informasi adalah data yang telah diklasifikasikan atau diolah atau diinterpretasikan untuk digunakan dalam proses pengambilan keputusan. Sistem pengolahan informasi akan mengolah data menjadi informasi atau mengolah data dari bentuk tak berguna menjadi berguna bagi yang menerimanya. Nilai informasi berhubungan dengan keputusan. Bila tidak ada pilihan atau keputusan maka informasi tidak diperlukan. Keputusan dapat berkisar dari keputusan berulang sederhana sampai keputusan strategis jangka panjang. Nilai informasi dilukiskan paling berarti dalam konteks pengambilan keputusan (Tata Sutabri, 2012: 21-22).

II.1.3. Definisi Sistem Informasi

Menurut Sutabri (2012 : 46), Sistem informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian yang mendukung fungsi operasi organisasi yang bersifat manajerial dengan kegiatan strategi dari suatu organisasi untuk dapat menyediakan kepada pihak luar tertentu dengan laporan-laporan yang diperlukan.

Menurut Anatasia Diana dan Lilis Setiawati (2010 : 4), sistem informasi adalah sistem buatan manusia yang biasanya terdiri dari sekumpulan komponen

baik manual ataupun berbasis komputer yang terintegrasi untuk mengumpulkan, menyimpan, dan mengelola data serta menyediakan informasi kepada pihak-pihak yang berkepentingan sebagai pemakai informasi tersebut (Rosana Junita Sirait, et al., 2015 : 2).

Spesialis informasi digolongkan menjadi lima (5) macam, yaitu (Oktafiansyah, 2012 : 2) :

1. Analis sistem, adalah pakar dalam mendefinisikan masalah dan menyiapkan dokumentasi tertulis mengenai cara komputer membantu pemecahan masalah.
2. Pengelola basis data (*Data Base Administrator*/DBA), bekerjasama dengan pemakai dan analis sistem menciptakan basis data yang berisi data yang diperlukan untuk menghasilkan informasi bagi pemakai.
3. Spesialis jaringan (*Network Specialist*), adalah orang yang ahli dalam bidang komputer dan telekomunikasi.
4. Pemrogram (*Programmer*), bekerja dengan menggunakan dokumentasi yang disiapkan oleh analis sistem untuk membuat kode program dalam bahasa tertentu untuk memproses data masukan yang tersedia menjadi keluaran yang berupa informasi bagi para pemakai.
5. Operator, mengoperasikan peralatan komputer berskala besar (misal: *mainframe,* mini), memantau layar komputer, mengganti ukuran kertas di printer, mengelola perpustakaan *disk storage*, dan lain-lain.

II.1.4. Definisi Akuntansi

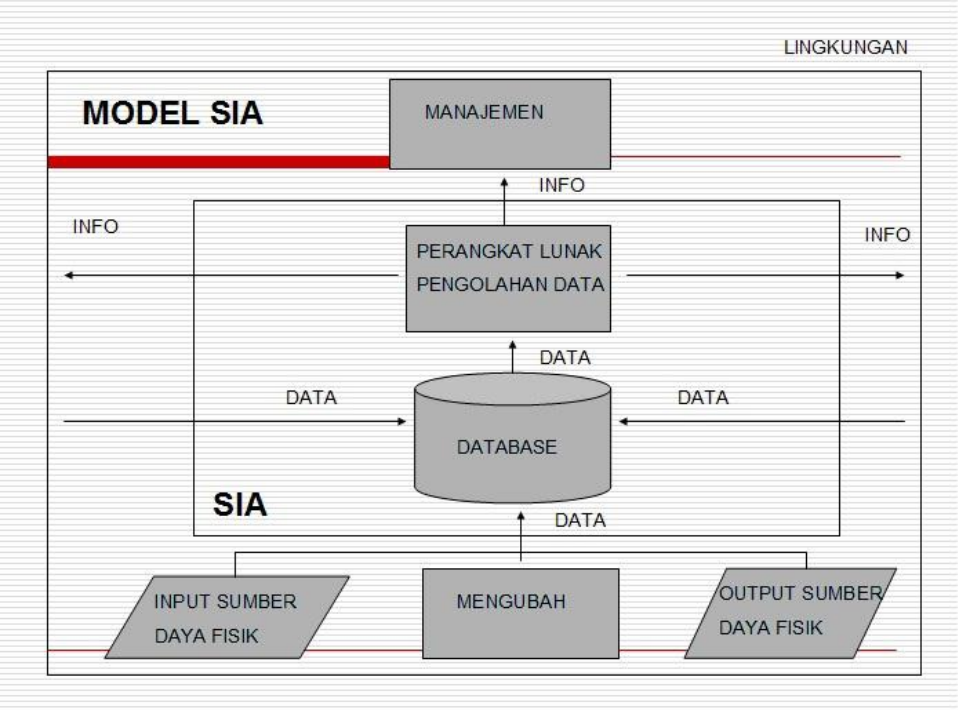
Menurut Herry (2012 : 8), akuntansi adalah melakukan pencatatan atas transaksi harian dan menyiapkan laporan keuangan. Akuntansi adalah kegiatan atau pencatatan (*record*), penggolongan (*classifying*), peringkasan (*summarizing*) transaksi-transaksi keuangan yang terjadi pada suatu organisasi dan melaporkan atau menyajikan serta menafsirkan hasilnya (Rosana Junita Sirait, et al., 2015 : 2).

II.2. Sistem Informasi Akuntansi

Menurut Ajeng Wind (2014 : 39), sistem informasi akuntansi adalah suatu sistem untuk mengumpulkan dan memproses data yang dipergunakan untuk pembuatan laporan keuangan yang nantinya digunakan oleh pihak-pihak yang berkaitan. Menurut Donald E. Kieso dan kawan-kawan (2008 : 72), sistem informasi akuntansi adalah sistem pengumpulan dan pemrosesan data transaksi serta penyebaran informasi keuangan kepada pihak-pihak yang berkepentingan. Sistem informasi akuntansi sangat bervariasi dari satu bisnis ke bisnis lainnya (Rosana Junita Sirait, et al., 2015 : 2).

Sistem Informasi Akuntansi (SIA) adalah sebuah sistem informasi yang menangani segala sesuatu yang berkenaan dengan akuntansi. Akuntansi sendiri sebenarnya adalah sebuah sistem informasi. Faktor-faktor yang dipertimbangkan dalam penyusunan sistem informasi akuntansi: Sistem informasi akuntansi yang disusun harus memenuhi prinsip (1) cepat yaitu sistem informasi akuntansi harus menyediakan informasi yang diperlukan dengan cepat dan tepat waktu serta dapat

memenuhi kebutuhan dan kualitas yang sesuai, (2) aman yaitu sistem informasi harus dapat membantu menjaga keamanan harta milik perusahaan. (3) murah yang berarti bahwa biaya untuk menyelenggarakan sistem informasi akuntansi tersebut harus dapat ditekan sehingga relatif tidak mahal (Rochmawati Daud, 2014 : 19).



Gambar II.1. Model SIA

(Sumber : Alita Erma, 2014)

Gambar diatas adalah suatu model SIA. Elemen input, transformasi dan output dari sistem fisisk perusahaan berada pada bagian bawah. Data dikumpulkan dari seluruh sistem fisik dan lingkungan, lalu dimasukkan ke dalam database. Perangkat lunak data mengubah data menjadi informasi untuk manajemen perusahaan serta untuk perorangan dan organisasi di lingkungan perusahaan.

Arus informasi kelingkungan penting untuk dipahami. SIA adalah satu–satunya CBIS yang bertanggung jawab memenuhi kebutuhan informasi di luar perusahaan. SIA bertanggung jawab untuk menyediakan informasi bagi tiap elemen lingkungan kecuali pesaing (Alita Erma, 2014).

II.3. Metode Fluktuasi

Suatu metode pencatatan dan pengendalian kas kecil, di mana jumlah kas kecil akan selalu berubah karena pengisian kembali kas kecil selalu sama dari waktu ke waktu. Setiap pengeluaran yang menggunakan kas kecil harus selalu dicatat (dijurnal) berdasarkan bukti transaksi yang ada satu per satu. (Rudianto, 2012 : 188).

Ilustrasi berikut ini dapat memperjelas keterangan mengenai pencatatan kas kecil metode fluktuasi.

Pada awal bulan Februari 2012, Manajer Keuangan PT. Mitra Lestari membentuk dana kas kecil yang akan digunakan untuk membayar pengeluaran-pengeluaran tunai yang tidak besar jumlahnya dan sering terjadi. Disepakati bahwa dana kas kecil yang dibentuk sebesar Rp. 1.500.000, yang akan diisi kembali setiap tanggal 1 dan 16 setiap bulannya.

Selama bulan Februari 2012, transaksi PT. Mitra Lestari yang menggunakan kas kecil adalah sebagai berikut :

4/2/2012 Membeli materai dan perangko sebesar Rp. 225.000.

10/2/2012 Membayar beban perbaikan kendaraan sebesar Rp. 600.000.

12/2/2012 Membeli bensin, solar, dan minyak sebesar Rp. 275.000.

17/2/2012 Membayar beban perbaikan gedung kantor sebesar Rp. 850.000.

25/2/2012 Membeli perlengkapan kantor sebesar Rp. 450.000.

Jurnal umum yang diperlukan menyangkut penggunaan dana kas kecil tersebut, dengan metode fluktuasi adalah sebagai berikut (Rudianto, 2012 : 189) :

Tabel II.1. Jurnal Umum Metode Fluktuasi

Tanggal		**Fluktuasi**	
2012			
Feb	1	Kas kecil	1.500.000
		Kas	1.500.000
		(mencatat pembentukan kas kecil)	
	4	Beban materai & perangko	225.000
		Kas kecil	225.0000
		(mencatat pembelian perangko dan materai)	
	10	Beban perbaikan kendaraan	600.000
		Kas kecil	600.000
		(mencatat beban reparasi kendaraan)	
	12	Beban bahan bakar	275.000

		Kas kecil 275.000 (mencatat pembelian bensin dan solar)
	16	Kas kecil 1.500.000 Kas 1.500.000 (mencatat pengisian kembali kas kecil)
	17	Beban perbaikan gedung 850.000 Kas kecil 850.000 (mencatat beban perbaikan bangunan)
	25	Perlengkapan kantor 450.000 Kas kecil 450.000 (mencatat pembelian perlengkapan kantor)
Mar	1	Kas kecil 1.500.000 Kas 1.500.000 (mencatat pengisian kembali kas kecil)

(Sumber : Rudianto, 2012 : 189)

II.4. Metode Imprest

Suatu metode pengisian dan pengendalian kas kecil di mana jumlah kas kecil selalu tetap dari waktu ke waktu, karena pengisian kembali kas kecil akan selalu sama dengan jumlah yang telah dikeluarkan. Penggunaan kas kecil yang

dicatat dengan metode imprest tidak memerlukan pencatatan (jurnal) atas setiap transaksi yang terjadi. Bukti-bukti transaksi dikumpulkan, dan pada saat pengisian kembali, kas kecil diisi kembali berdasarkan jumlah dari keseluruhan bukti transaksi tersebut. (Rudianto, 2012 : 188).

Pada awal bulan Februari 2012, Manajer Keuangan PT. Mitra Lestari membentuk dana kas kecil yang akan digunakan untuk membayar pengeluaran-pengeluaran tunai yang tidak besar jumlahnya dan sering terjadi. Disepakati bahwa dana kas kecil yang dibentuk sebesar Rp. 1.500.000, yang akan diisi kembali setiap tanggal 1 dan 16 setiap bulannya.

Selama bulan Februari 2012, transaksi PT. Mitra Lestari yang menggunakan kas kecil adalah sebagai berikut :

4/2/2012 Membeli materai dan perangko sebesar Rp. 225.000.

10/2/2012 Membayar beban perbaikan kendaraan sebesar Rp. 600.000.

12/2/2012 Membeli bensin, solar, dan minyak sebesar Rp. 275.000.

17/2/2012 Membayar beban perbaikan gedung kantor sebesar Rp. 850.000.

25/2/2012 Membeli perlengkapan kantor sebesar Rp. 450.000.

Jurnal umum yang diperlukan menyangkut penggunaan dana kas kecil tersebut, dengan metode imprest adalah sebagai berikut (Rudianto, 2012 : 189) :

Tabel II.2. Jurnal Umum Metode Imprest

Tanggal		Imprest	
2012			
Feb	1	Kas kecil	1.500.000
		Kas	1.500.000
		(mencatat pembentukan kas kecil)	
	4		
	10		
	12		
	16	Macam-macam beban	1.100.000
		Kas	1.100.000
		(mencatat pengisian kembali kas kecil)	
	17		
	25		
Mar	1	Macam-macam beban	1.300.000
		Kas	1.300.000
		(mencatat pengisian kembali kas kecil)	

(Sumber : Rudianto, 2012 : 189)

II.5. *Visual Basic* .NET

Pada tahun 1991 Microsoft mengeluarkan Visual Basic, pengembangan dari Basic yang berubah dari sisi pembuatan antarmukanya. Visual Basic sampai sekarang masih menjadi salah satu bahasa pemrograman terpopuler di dunia. Pada akhir tahun 1999, Teknologi .NET diumumkan. Microsoft memosisikan teknologi tersebut sebagai *platform* untuk membangun XML Web *Services*. XML Web

services memungkinkan aplikasi tipe manapun dan dapat mengambil data yang tersimpan pada server dengan tipe apapun melalui internet.

Visual Basic.NET adalah Visual Basic yang direkayasa kembali untuk digunakan pada *platform* .NET sehingga aplikasi yang dibuat menggunakan Visual Basic .NET dapat berjalan pada sistem komputer apa pun, dan dapat mengambil data dari server dengan tipe apa pun asalkan terinstal .NET Framework. (Priyanto Hidayatullah, 2012 : 5).

Visual Basic .NET layak untuk dijadikan pilihan karena mempunyai cukup banyak kelebihan. Beberapa kelebihan Visual Basic .NET antara lain (Priyanto Hidayatullah, 2012 : 7-8) :

1. Sederhana dan mudah dipahami

   Seperti pada VB, bahasa yang digunakan pada VB .NET sangat sederhana sehingga lebih mudah dipahami bagi mereka yang masih awam terhadap dunia pemrograman.

2. Mendukung GUI

   VB .NET bisa membuat *software* dengan antarmuka grafis yang lebih *user friendly.*

3. Menyederhanakan *deployment*

   VB .NET mengatasi masalah *deployment* dari aplikasi berbasis Windows yaitu DLL Hell dan registrasi COM (*Component Object Model*). Selain itu tersedia wizard yang memudahkan dalam pembuatan *file setup.*

4. Menyedarhanakan pengembangan perangkat lunak

Ketika terjadi kesalahan penulisan kode dari sisi sintaks (bahasa), maka VB .NET langsung menuliskan kesalahannya pada bagian *Message* Windows sehingga *Programmer* dapat memperbaiki kode dengan lebih cepat.

5. Mendukung penuh OOP

   Memiliki fitur bahasa pemrograman berorientasi objek seperti *inheritence* (pewarisan), *encapsulation* (pembungkusan), dan *polymorphism* (banyak bentuk).

II.6. *SQL Server 2008*

SQL *(Structured Query Language)* adalah sebuah bahasa yang dipergunakan umtuk mengakses data dalam basis data relasional. Bahasa ini secara *de facto* merupakan bahasa standar yang dipergunakan dalam manajemen basis data relasional. Saat ini hampir semua server basis data yang ada mendukung bahasa ini untuk melakukan manajemen datanya.

SQL terdiri dari dua bahasa, yaitu *Data Definition Language Manipulation Language (DML)*. Implementasi DDL dan DML sistem manajemen basis data *(SMBD)*, namun secara umum implemen bahasa ini memiliki bentuk standar yang ditetapkan oleh *ANSI*. (Adelia dan Jimmy Setiawan, 2011 : 115).

1. *Data Defenition Language (DDL)*

   DDL digunakan untuk mendefinisikan, mengubah, serta menghapus basis data dan objek-objek yang diperlukan dalam basis data, misalnya

tabel, *view*, *user*, dan sebagainya. *DDL* biasanya digunakan oleh administrator basis data dalam pembuatan sebuah aplikasi basis data. Secara umum *DDL* yang digunakan adalah:

a. *CREATE* untuk membuat objek baru.
b. *USE* untuk menggunakan objek yang sudah ada.
c. *ALTER* untuk mengubah objek yang sudah ada.
d. *DROP* untuk menghapus objek.

2. *Data Manipulation Language (DML)*

DML digunakan untuk memanipulasi data yang ada dalam suatu tabel. Perintah-perintah yang umum dilakukan adalah:

a. *SELECT* untuk menampilkan data.
b. *INSERT* untuk menambahkan data baru.
c. *UPDATE* untuk mengubah data yang sudah ada.
d. *DELETE* untuk menghapus data.

II.7. Pengertian Basis Data

Basis data adalah kumpulan data (arsip, atau file) yang saling berhubungan yang disimpan dalam media penyimpanan elektronis agar dapat dimanfaatkan kembali dangan cepat, dan mudah. Sedangkan sistem basis data adalah kumpulan file, atau tabel yang saling berhubungan yang memungkinkan beberapa pemakai, atau program lain untuk mengakses, dan memanipulasi file-file (tabel) tersebut (Eka Kurniawan, 2015 : 4).

Sistem basis data mempunyai beberapa elemen penting yaitu (Oktafiansyah, 2012 : 3) :

1. Basis data sebagai inti dari sistem basis data.
2. Perangkat lunak (*software*) untuk perancangan dan pengelolaan basis data.
3. Perangkat keras (*hardware*) sebagai pendukung operasi pengolahan data.
4. Manusia (*brainware*) sebagai pemakai atau para spesialis informasi yang mempunyai fungsi sebagai perancang atau pengelola.

II.8. Normalisasi

Menurut Martin (1975), Normalisasi diartikan sebagai suatu teknik yang menstrukurkan/ mendekomposisi data dalam cara-cara tertentu untuk mencegah timbulnya permasalahan pengolahan data dalam basis data. Permasalahan yang dimaksud adalah berkaitan dengan penyimpangan-penyimpangan *(anomalies)* yang terjadi akibat adanya kerangkapan data dalam relasi dan in-efisiensi pengolahan (Edy Sutanta ; 2011 : 174).

Proses normalisasi menghasilkan relasi yang optimal, yaitu (Martin, 1975) : (Edy Sutanta ; 2011 : 175)

1. Memiliki struktur *record* yang konsisten secara logik;
2. Memiliki struktur *record* yang mudah untuk dimengerti;
3. Memiliki struktur *record* yang sederhana dalam pemeliharaan;

4. Memiliki struktur *record* yang mudah ditampilkan kembali untuk memenuhi kebutuhan pengguna;
5. Minimalisasi kerangkapan data guna meningkatkan kinerja sistem.

Secara berturut-turut masing-masing level normal tersebut dibahas berikut ini, dimulai dari bentuk tidak normal. (Edy Sutanta ; 2011 : 176-179)

1. Relasi bentuk tidak normal (*Un Normalized Form* / UNF)

   Relasi-relasi yang dirancang tanpa mengindahkan batasan dalam defisi basis data dan karakteristik *Relational Database Management System* (RDBM) menghasilkan relasi *Un Normalized Form* (UNF). Bentuk ini harus di hindari dalam perancangan relasi dalam basis data. Relasi *Un Normalized Form* (UNF) mempunyai kriteria sebagai berikut.

   a. Jika relasi mempunyai bentuk *non flat file* (dapat terjadi akibat data disimpan sesuai dengan kedatangannya, tidak memiliki struktur tertentu, terjadi duplikasi atau tidak lengkap)
   b. Jika relasi membuat *set atribut* berulang *(non single values)*
   c. Jika relasi membuat *atribut non atomic value*

2. Relasi bentuk normal pertama (*First Norm Form* / 1NF)

   Relasi disebut juga *First Norm Form* (1NF) jika memenuhi kriteria sebagai berikut.

   a. Jika seluruh atribut dalam relasi bernilai *atomic (atomic value)*
   b. Jika seluruh atribut dalam relasi bernilai tunggal *(single value)*
   c. Jika relasi tidak memuat set atribut berulang

d. Jika semua record mempunyai sejumlah atribut yang sama.

Permasalahan dalam *First Norm Form* (1NF) adalah sebagai berikut.

a. Tidak dapat menyisipkan informasi parsial
b. Terhapusnya informasi ketika menghapus sebuah *record*

3. Bentuk normal kedua *(Second Normal Form /* 2NF)

Relasi disebut sebagai *Second Normal Form (*2NF) jika memenuhi kriteria sebagai berikut

a. Jika memenuhi kriteria *First Norm Form (*1NF)
b. Jika semua atribut nonkunci *Functional Dependence* (FD) pada *Primary Key* (PK)

Permasalahan dalam *Second Normal Form /* 2NF adalah sebagai berikut:

a. Kerangkapan data *(data redundancy)*
b. Pembaharuan yang tidak benar dapat menimbulkan inkonsistensi data *(data inconsistency)*
c. Proses pembaharuan data tidak efisien

Kriteria tersebut mengidentifikasikan bahwa antara atribut dalam *Second Normal Form* masih mungkin mengalami *Third Norm Form*. Selain itu, relasi *Second Normal Form* (2NF) menuntut telah didefinisikan atribut *Primary Key* (PK) dalam relasi. Mengubah relasi *First Norm Form* (1NF) menjadi bentuk *Second Normal Form* (2NF) dapat dilakukan dengan mengubah struktur relasi dengan cara :

a. Identifikasikan *Functional Dependence* (FD) relasi *First Norm Form* (1NF)
b. Berdasarkan informasi tersebut, dekomposisi relasi *First Norm Form* (1NF) menjadi relasi-relasi baru sesuai *Functional Dependence* nya. Jika menggunakan diagram maka simpul-simpul yang berada pada puncak diagram ketergantungan data bertindak *Primary Key* (PK) pada relasi baru

4. Bentuk normal ketiga (*Third Norm Form* / 3NF)

   Suatu relasi disebut sebagai *Third Norm Form* jika memenuhi kriteria sebagai berikut.

   a. Jika memenuhi kriteria *Second Normal Form* (2NF)
   b. Jika setiap atribut nonkunci tidak (TDF) *(Non Transitive Dependeny)* terhadap *Primary Key* (PK)

   Permasalahan dalam *Third Norm Form* (3NF) adalah keberadaan penentu yang tidak merupakan bagian dari *Primary Key* (PK) menghasilkan duplikasi rinci data pada atribut yang berfungi sebagai *Foreign Key* (FK) (duplikasi berbeda dengan keterangan data).

   Mengubah relasi *Second Normal Form* (2NF) menjadi bentuk *Third Norm Form* (3NF) dapat dilakukan dengan mengubah struktur relasi dengan cara :

   a. Identifikasi TDF relasi *Second Normal Form* (2NF)
   b. Berdasarkan informasi tersebut, dekomposisi relasi *Second Normal Form* (2NF) menjadi relasi-relasi baru sesuai TDF-nya.

5. Bentuk normal *Boyce-Cood (Boyce-Codd Norm Form* / BCNF)

   Bentuk normal *Boyce-Codd Norm Form* (BCNF) dikemukakan oleh R.F. Boyce dan E.F. Codd. Suatu relasi disebut sebagai *Boyce-Codd Norm Form* (BCNF) jika memenuhi kriteria sebagai berikut.

   a. Jika memenuhi kriteria *Third Norm Form* (3NF)

   b. Jika semua atribut penentu (determinan) merupakan CK

6. Bentuk normal keempat (*Forth Norm Form* / 4NF)

   Relasi disebut sebagi *Forth Norm Form* (4NF) jika memenuhi kriteria sebagai berikut.

   b. Jika memenuhi kriteria *Boyce-Codd Norm Form*.

   c. Jika setiap atribut didalamnya tidak mengalami ketergantungan pada banyak nilai.

7. Bentuk normal kelima (*Fifth Norm Form* / 5NF)

   Suatu relasi memenuhi kriteria *Fifth Norm Form* (5NF) jika kerelasian antar data dalam relasi tersebut tidak dapat direkonstruksi dari struktur relasi yang sederhana.

8. Bentuk normal kunci domain (*Domain Key Norm Form* / DKNF)

   a. Relasi disebut sebagai *Domain Key Norm Form* (DKNF) jika setiap batasan dapat disimpulkan secara sederhana dengan mengetahui sekumpulan nama atribut dan domainnya selama menggunkan sekumpulan atribut pada kuncinya.

II.9. *Unified Modeling Language* (UML)

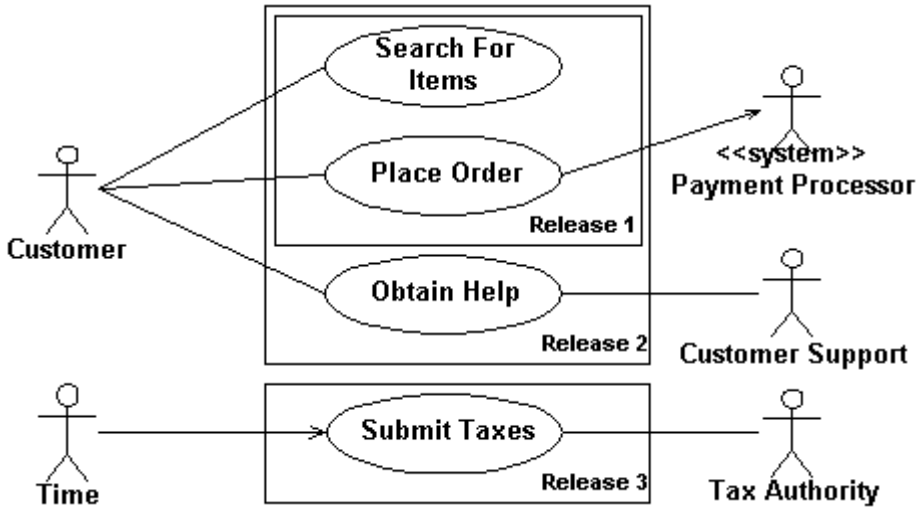
Unified Modeling Language (UML) adalah sebuah bahasa yang diterima dan digunakan oleh *software developer* dan *software analyst* sebagai suatu bahasa yang cocok untuk merepresentasikan grafik dari suatu relasi antar entitas-entitas software (Gornik, 2003). Dengan menggunakan UML, tim pengembang *software* akan mempunyai banyak keuntungan, seperti memudahkan komunikasi dengan sesama anggota tim tentang *software* apa yang akan dibuat, memudahkan integrasi ke dalam area pengerjaan *software* karena bahasa ini berbasiskan *meta-models* dimana *meta-models* bisa mendefinisikan proses-proses untuk mengkonstruksikan konsep-konsep yang ada. UML juga menggunakan formatur *input* dan *output* yang sudah mempunyai bentuk standar yaitu *XML Metadata Interchange* (XMI), menggunakan aplikasi dan pemodelan data yang universal, merepresentasikan dari tahap analisis ke implementasi lalu ke *deployment* yang terpadu, dan mendeskripsikan keutuhan tentang spesifikasi software.

UML menyediakan kumpulan alat yang sudah terstandarisasi, yang digunakan untuk mendokumentasikan analisis dan perancangan sebuah sistem perangkat lunak. (Kendall & Kendall, 2005, p663) Peralatan utama UML adalah diagram-diagram yang digunakan untuk membantu manusia dalam memvisualisasikan proses pengembangan sebuah sistem perangkat lunak, sama seperti penggunaan denah (*blueprint*) dalam pembuatan bangunan (Edgar Winata dan Johan Setiawan, 2013 : 37).

II.9.1. *Use Case* Diagram

Diagram yang menggambarkan *actor*, *use case* dan relasinya sebagai suatu urutan tindakan yang memberikan nilai terukur untuk aktor. Sebuah *use case* digambarkan sebagai *elips horizontal* dalam suatu diagram UML *use case*. *Use Case* memiliki dua istilah, yaitu (Haviluddin, 2011 : 4) :

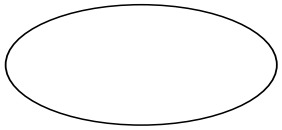
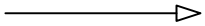
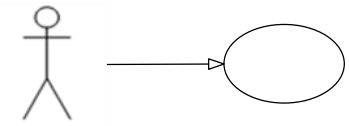
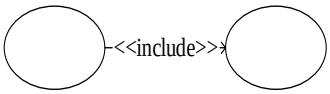
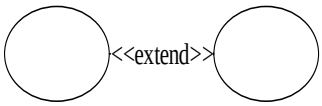
1. *System use case*; interaksi dengan sistem.
2. *Business use case*; interaksi bisnis dengan konsumen atau kejadian nyata.



Gambar II.2. Notasi *Use Case* Diagram
(Sumber : Haviluddin, 2011 : 4)

Simbol-simbol yang digunakan dalam membuat *use case* diagram dapat dilihat pada tabel II.3.

Tabel II.3. Simbol-simbol *Use Case* Diagram

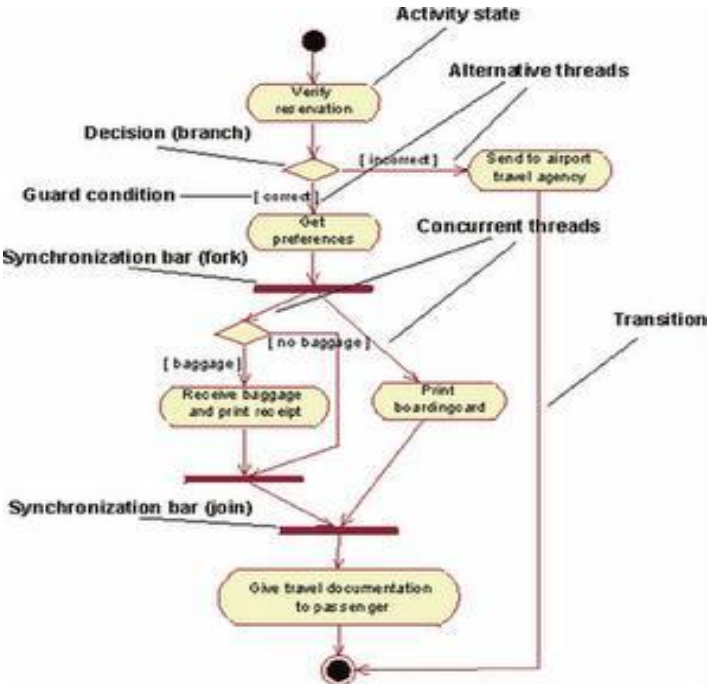
Simbol	Keterangan
	*Use Case* : menggambarkan bagaimana seseorang akan menggunakan/memanfaatkan sistem.
	Aktor : seseorang/sesuatu yang berinteraksi dengan sistem yang sedang kita kembangkan
	Relasi : sebagai penghubung antara aktor-*use case*, *use case* - *use case* dll.
	Relasi Asosiasi : relasi terjadi antara aktor dengan *use case* biasanya berupa garis lurus dengan kepala panah di salah satu ujungnya.
	*Include Relationship* (Relasi Cakupan) : memungkinkan suatu *use case* untuk menggunakan fungsionalitas yang disediakan oleh *use case* lainnya.
	*Extend Relathionship* : memungkinkan *use case* memiliki kemungkinan untuk memperluas fungsionalitas yang disediakan oleh *use case* lainnya.

(Sumber : Oktafiansyah, 2012 : 5)

II.9.2. *Activity* Diagram

Activity diagram menggambarkan aktifitas-aktifitas, objek, *state*, *transisi state* dan *event*. Dengan kata lain kegiatan diagram alur kerja menggambarkan perilaku sistem untuk aktivitas (Haviluddin, 2011 : 4).

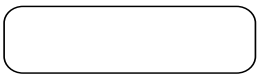
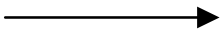
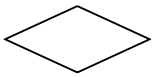
Teknik untuk menjelaskan *business process*, *procedural logic*, dan *work flow*. Bisa dipakai untuk menjelaskan teks *use case* dalam notasi grafis dengan menggunakan notasi yang mirip *flow chart*, meskipun terdapat sedikit perbedaan notasi (Edgar Winata dan Johan Setiawan, 2013 : 39).



Gambar II.3. Notasi *Activity* Diagram
(Sumber : Haviluddin, 2011 : 4)

Simbol-simbol yang digunakan dalam membuat *activity* diagram dapat dilihat pada tabel II.4.

Tabel II.4. Simbol-simbol *Activity* Diagram

Simbol	Keterangan
	Titik Awal
	Titik AKhir
	*Activity*
	Garis tidak putus. Urutan dari suatu kejadian atau aktivitas ke yang berikutnya.
	*Fork* : digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel atau untuk menggabungkan dua kegiatan paralel menjadi satu.
	*Rake* : menunjukkan adanya dekomposisi.
	Tanda pengiriman.
	Tanda waktu.
	Pilihan untuk pengambilan keputusan.
	Tanda penerimaan
	Aliran Akhir (*Flow Final*)

(Sumber : Oktafiansyah, 2012 : 5)

II.9.3. *Class* Diagram

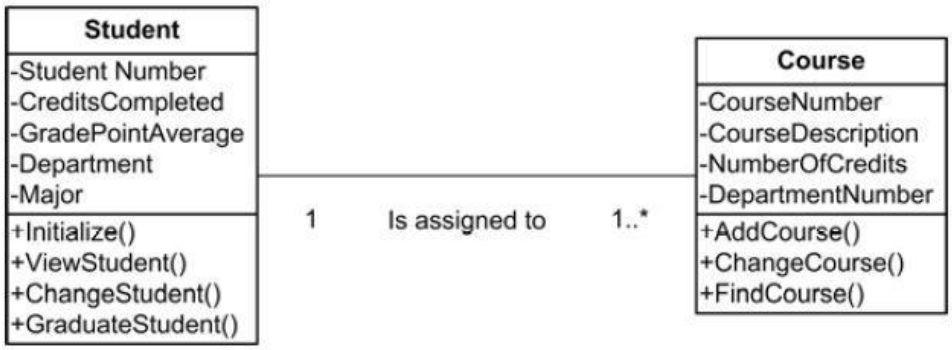
Class diagram merupakan diagram paling umum yang dijumpai dalam pemodelan berbasis UML. Didalam *Class* diagram terdapat *class* dan *interface* beserta atribut-atribut dan operasinya, relasi yang terjadi antar objek, *constraint* terhadap objek-objek yang saling berhubungan dan *inheritance* untuk organisasi *class* yang lebih baik. *Class* diagram juga terdapat *static view* dari elemen pembangun sistem. Pada intinya *Class* diagram mampu membantu proses pembuatan sistem dengan memanfaatkan konsep *forward* ataupun *reverse engineering* (Edgar Winata dan Johan Setiawan, 2013 : 38).

Berbagai simbol yang hadir didalam *class* diagram antara lain adalah:

1. *Class*, yang berfungsi untuk merepresentasikan tipe dari data yang dimilikinya. *Class* diagram dapat ditampilkan dengan menunjukkan atribut dan operasi yang dimilikinya atau hanya menunjukkan nama *class*-nya saja. Dapat juga kita tuliskan nama *class* dengan atributnya saja atau nama *class* dengan operasinya.
2. *Attribute*, merupakan data yang terdapat didalam *class* dan *instance*-nya dengan operator.
3. *Operation*, berfungsi untuk merepresentasikan fungsi-fungsi yang ditampilkan oleh *class* dan *instance*-nya dengan operator.
4. *Association*, digunakan untuk menunjukkan bagaimana dua *class* berhubungan satu sama lainnya. *Association* ditunjukkan dengan sebuah garis yang terletak diantara dua *class*. Didalam setiap *association* terdapat *multiplicity*, yaitu simbol yang mengindikasikan

berapa banyak *instance* dari *class* pada ujung *association* yang satu dengan *instance class* di ujung *association* lainnya.

5. *Generalizations*, berfungsi untuk mengelompokkan *class* ke dalam hirarki *inheritance*.
6. *Aggregation*, merupakan bentuk khusus dari *association* yang merepresentasikan hubungan "*part-whole*". Bagian "*whole*" dari hubungan ini sering disebut dengan *assembly* atau *aggregate*. *Class* yang satu dapat dikatakan merupakan bagian dari *class* yang lain yang ikut membentuk *class* tersebut.
7. *Composition*, merupakan jenis *aggregation* yang lebih kuat diantara dua *class* yang memiliki *association* dimana jika *whole* ditiadakan, maka *part*-nya juga ikut ditiadakan. Berbeda dengan *aggregation*, *part* akan tetap bisa berdiri sendiri meskipun bagian *whole*-nya ditiadakan.
8. Penggunaan operator (+) dalam *class* diagram diartikan dengan *public*, operator (-) diartikan *private*, dan operator (#) diartikan *protected*.

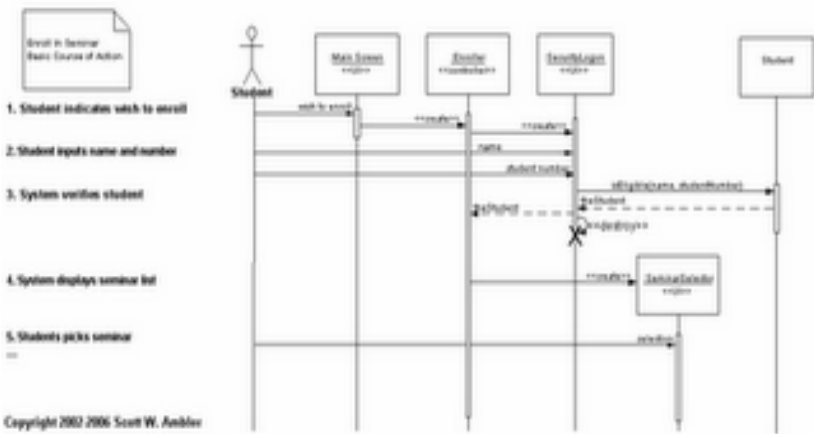


Gambar II.4. Contoh *Class* Diagram
(Sumber : Edgar Winata dan Johan Setiawan, 2013 : 38)

II.9.4. *Sequence* Diagram

Menjelaskan interaksi obyek-obyek yang saling berkolaborasi (berhubungan), mirip dengan *activity* diagram yaitu menggambarkan alur kejadian sebuah aktivitas tetapi lebih detil dalam menggambarkan aliran data termasuk data atau behaviour yang dikirimkan atau diterima namun kurang mampu menjelaskan detil dari sebuah algoritma (Edgar Winata dan Johan Setiawan, 2013 : 39).

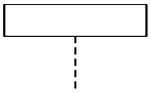
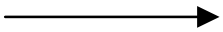
Secara mudahnya *sequence diagram* adalah gambaran tahap demi tahap, termasuk kronologi (urutan) perubahan secara logis yang seharusnya dilakukan untuk menghasilkan sesuatu sesuai dengan *use case diagram* (Haviluddin, 2011 : 5).



Gambar II.5. Notasi *Sequence* Diagram
(Sumber : Haviluddin, 2011 : 5)

Simbol-simbol yang digunakan dalam membuat *sequence* diagram dapat dilihat pada tabel II.5.

Tabel II.5. Simbol-simbol *Sequence* Diagram

Simbol	Keterangan
	Aktor : seseorang/sesuatu yang berinteraksi dengan sistem yang sedang kita kembangkan.
	Objek : menambah objek baru pada diagram.
	Aktivasi : menggambarkan langkah-langkah dalam aliran kerja.
	Pesan : menggambarkan pesan antara dua objek.
	Pengulangan : menggambarkan pesan yang menuju dirinya sendiri.

(Sumber : Oktafiansyah, 2012 : 6)