BAB II

LANDASAN TEORI

II.1. Sistem Pendukung Keputusan

Pada dasarnya sistem pendukung keputusan merupakan pengembangan lebih lanjut dari sistem informasi manajemen terkomputerisasi yang dirancang sedemikian rupa sehingga bersifat interaktif dengan pemakainya. Sifat interaktif dimaksudkan untuk memudahkan integrasi antara berbagai komponen dalam proses pengambilan keputusan seperti prosedur, kebijakan, teknik analisis, serta pengalaman dan wawasan manajerial guna membentuk suatu kerangka keputusan bersifat fleksibel. Konsep Sistem Pendukung Keputusan (SPK)/*Decision Support Sistem* (DSS) pertama kali diungkapkan pada awal tahun 1970-an oleh Michael S. Scott Morton dengan istilah *Management Decision Sistem*. Sistem tersebut adalah suatu sistem yang berbasis komputer yang ditujukan untuk membantu pengambil keputusan dengan memanfaatkan data dan model tertentu untuk memecahkan berbagai persoalan yang tidak terstruktur.(Desi leha kurniasih ; 2013 : 7)

Ciri-ciri Sistem Pendukung Keputusan (SPK) Menurut Kosasi dan Kusrini (2007), adapun ciriciri

sebuah SPK seperti yang dirumuskan oleh Alters Keen adalah sebagai berikut:

1. SPK ditujukan untuk membantu pengambilan keputusan-keputusan yang kurang terstruktur dan umumnya dihadapi oleh para manajer yang berada di tingkat puncak.

2. SPK merupakan gabungan antara kumpulan model kualitatif dan kumpulan data.
3. SPK memiliki fasilitas interaktif yang dapat mempermudah hubungan antara manusia dengan komputer.
4. SPK bersifat luwes dan dapat menyesuaikan dengan perubahan-perubahan yang terjadi.

Adapun Karakteristik, Kemampuan, dan Keterbatasan SPK adalah Sehubungan banyaknya definisi yang dikemukakan mengenai pengertian dan penerapan dari sebuah SPK, sehingga menyebabkan terdapat banyak sekali pandangan mengenai sistem tersebut. Selanjutnya Turban (1996), menjelaskan terdapat sejumlah karakteristik dan kemampuan dari SPK yaitu:

a. Karakteristik SPK
    1. Mendukung seluruh kegiatan organisasi.
    2. Mendukung beberapa keputusan yang saling berinteraksi.
    3. Dapat digunakan berulang kali dan bersifat konstan
    4. Terdapat dua komponen utama, yaitu data dan model Menggunakan baik data eksternal dan internal
    5. Memiliki kemampuan *what-if analysis* dan *goal seeking analysis* Menggunakan beberapa model kuantitatif
b. Kemampuan SPK
    1. Menunjang pembuatan keputusan manajemen dalam menangani masalah semi terstruktur dan tidak terstruktur.

2. Membantu manajer pada berbagai tingkatan manajemen, mulai dari manajemen tingkat atas sampai manajemen tingkat bawah.
3. Menunjang pembuatan keputusan secara kelompok maupun perorangan.
4. Menunjang pembuatan keputusan yang saling bergantung dan berurutan.
5. Menunjang tahap-tahap pembuatan keputusan antara lain *intelligensi, desain, choice, dan implementation*.
6. Kemampuan untuk melakukan adaptasi setiap saat dan bersifat fleksibel.
7. Kemudahan melakukan interaksi system.
8. Meningkatkan efektivitas dalam pembuatan.
9. keputusan daripada efisiensi.
10. Mudah dikembangkan oleh pemakai ahli.
11. Kemampuan pemodelan dan analisis pembuatan keputusan.
12. Kemudahan melakukan pengaksesan berbagai sumber dan format data.

c. Keterbatasan SPK

1. Ada beberapa kemampuan manajemen dan bakat manusia yang tidak dapat dimodelkan, sehingga model yang ada dalam sistem tidak semuanya mencerminkan persoalan sebenarnya.

2. Kemampuan suatu SPK terbatas pada pembendaharaan pengetahuan yang dimilikinya (pengetahuan dasar serta model dasar).
3. Proses-proses yang dapat dilakukan oleh SPK biasanya tergantung juga pada kemampuan perangkat lunak yang digunakannya. SPK tidak memiliki kemampuan intuisi seperti yang dimiliki oleh manusia. Karena walau bagaimanapun canggihnya suatu SPK, hanyalah sautu kumpulan perangkat keras, perangakat lunak dan sistem operasi yang tidak dilengkapi dengan kemampuan berpikir.

II.2. Pengertian Metode *AHP*.

Proses-proses yang dapat dilakukan oleh SPK biasanya tergantung juga pada kemampuan perangkat lunak yang digunakannya. SPK tidak memiliki kemampuan intuisi seperti yang dimiliki oleh manusia. Karena walau bagaimana pun canggihnya suatu SPK, hanyalah sautu kumpulan
perangkat keras, perangakat lunak dan sistem operasi yang tidak dilengkapi dengan kemampuan berpikir.

AHP adalah sebuah metode memecah permasalahan yang komplek/ rumit dalam situasi yang tidak terstruktur menjadi bagian-bagian komponen. Mengatur bagian atau variabel ini menjadi suatu bentuk susunan hierarki, kemudian memberikan nilai numerik untuk penilaian subjektif terhadap kepentingan relatif dari setiap variabel dan mensintesis penilaian untuk variabel mana yang memiliki prioritas tertinggi yang akan mempengaruhi penyelesaian dari situasi tersebut.

AHP menggabungkan pertimbangan dan penilaian pribadi dengan cara yang logis dan dipengaruhi imajinasi, pengalaman, dan pengetahuan untuk menyusun hierarki dari suatu masalah yang berdasarkan logika, intuisi dan juga pengalaman untuk memberikan pertimbangan. AHP merupakan suatu proses mengidentifikasi, dan memberikan perkiraan interaksi sistem secara keseluruhan.

Prosedur dalam metode AHP terdiri dari beberapa tahap (Suryadi dan Ramdhani, 1998), yaitu:

1. Menyusun hirarki dari permasalahan yang dihadapi.

Penyusunan hirarki yaitu dengan menentukan tujuan yang merupakan sasaran sistem secara keseluruhan pada level teratas. Level berikutnya terdiri dari kriteria-kriteria untuk menilai atau mempertimbangkan alternatif-alternatif yang ada dan menentukan alternatif-alternatif tersebut. Setiap kriteria dapat memiliki subkriteria dibawahnya dan setiap kriteria dapat memiliki nilai intensitas masing-masing.

2. Menentukan prioritas elemen.

a. Langkah pertama dalam menentukan prioritas elemen adalah membuat perbandingan berpasangan, yaitu membandingkan elemen secara berpasangan sesuai kriteria yang di berikan dengan menggunakan bentuk matriks. Matriks bersifat sederhana, berkedudukan kuat yang menawarkan kerangka untuk memeriksa konsistensi, memperoleh informasi tambahan dengan membuat semua perbandingan yang mungkin dan menganalisis kepekaan prioritas secara keseluruhan untuk merubah pertimbangan. Proses perbandingan berpasangan dimulai dari level paling atas hirarki untuk memilih kriteria, misalnya C,

kemudian dari level dibawahnya diambil elemen-elemen yang akan dibandingkan, misal A1, A2, A3, A4,

A5, maka susunan elemen-elemen pada sebuah matrik berikut:

Matrix perbandingan berpasangan

C	A1	A2	A3	A4	A5
A1	1				
A2		1			
A3			1		
A4				1	
A5					1

b. Mengisi matrik perbandingan berpasangan yaitu dengan menggunakan bilangan untuk merepresentasikan kepentingan relatif dari satu elemen terhadap elemen lainnya yang dimaksud dalam bentuk skala dari 1 sampai dengan 9. Skala ini mendefinisikan dan menjelaskan nilai 1 sampai 9 untuk pertimbangan dalam perbandingan berpasangan elemen pada setiap level hirarki terhadap suatu kreteria di level yang lebih tinggi. Apabila suatu elemen dalam matrik dan dibandingkan dengan dirinya sendiri, maka diberi nilai 1. Jika i dibanding j mendapatkan nilai tertentu, maka j dibanding i merupakan kebalikkannya. Berikut ini skala kuantitatif 1 sampai dengan 9 untuk menilai tingkat kepentingan suatu elemen dengan elemen lainnya.

c. Sintesis.

Pertimbangan-pertimbangan terhadap perbandingan berpasangan di sintesis untuk memperoleh keseluruhan prioritas.

1) Menjumlahkan nilai-nilai dari setiap kolom pada matriks.
2) Membagi setiap nilai dari kolom dengan total kolom yang bersangkutan untuk memperoleh normalisasi matriks.
3) Menjumlahkan nilai dari setiap matriks dan membaginya dengan jumlah elemen untuk mendapatkan nilai rata-rata.
4) Mengukur konsistensi.

Konsistensi penting untuk mendapatkan hasil yang valid dalam dunia nyata. AHP mengukur konsistensi pertimbangan dengan rasio konsistensi (consitency ratio). Nilai Konsistensi rasio harus kurang dari 5% untuk matriks 3x3, 9% untuk matriks 4x4 dan 10% untuk matriks yang lebih besar. Jika lebih dari rasio dari batas tersebut maka nilai perbandingan matriks di lakukan kembali. Langkah-langkah menghitung nilai rasio konsistensi yaitu:

a) Mengkalikan nilai pada kolom pertama dengan prioritas relatif elemen pertama, nilai pada kolom kedua dengan prioritas relatif elemen kedua, dan seterusnya.
b) Menjumlahkan setiap baris.
c) Hasil dari penjumlahan baris dibagikan dengan elemen prioritas relatif yang bersangkutan.
d) Membagi hasil diatas dengan banyak elemen yang ada, hasilnya disebut eigen value (□max).
e) Menghitung indeks konsistensi (*consistency index*) dengan rumus :

CI = (⅄ max-n)/n

Dimana CI : *Consistensi Index*

乁max : Eigen Value

n : Banyak elemen.

f) Menghitung konsistensi ratio (CR) dengan rumus :

CR=CI/RC

Dimana : CR : *Consistency Ratio*

CI : *Consistency Index*

RC : *Random Consistency*

Matriks random dengan skala penilaian 1 sampai 9 beserta kebalikkannya sebagai *random consistency* (RC). Berdasarkan perhitungan *saaty* menggunakan 500 sampel, jika pertimbangan memilih secara acak dari skala 1/9, 1/8, … , 1, 2, … , 9 akan diperoleh rata-rata konsistensi untuk matriks yang berbeda.(Tominanto : 2012 ; 2)

II.3. Pengertian Database

Database adalah suatu aplikasi yang menyimpan sekumpulan data. Setiap database mempunyai API tertentu untuk membuat, mengakses, mengatur, mencari, adan menyalin data yang ada di dalamnya.

Untuk menampung dan mengatur data yang begitu banyak, Anda dapat menggunakan *Relational database Management Systems(RDBMS)*. Hal ini disebut relational database karena semua data disimpan dalam tabel-tabel yang berbeda dan dihubungkan berdasarkan relasinya dengan menggunakan primary key dan foreign key.

Relational Database Management Systems(RDBMS) adalah software yang:

a. Memungkinkan anda untuk mengimplimentasikan sebuah database dengan tabel-tabel, kolom-kolom, dan indeks-indeks.
b. Menjamin integritas referensi di antara baris-baris pada berbagai tabel.
c. Mengupdate indeks-indekas secara otomatis.
d. Menginterpretasikan query SQL dan menggabungkan informasi dari berbagai tabel.

Terminologi RDBMS

Berikut ini adalah istilah-istilah yang digunakan dalam database:

a. Database: merupakan kumpulan tabel-tabel yang berisi data-data yang saling berkaitan.
b. Tabel: merupakan matriks berisi data. Tabel dalam database terlihat seperti *spreadsheet* sederhana.
c. Kolom: satu kolom (elemen data) mengandung data dengan satu jenis yang sama.
d. Baris: sebuah baris (masukan atau rekaman data) merupakan sekumpulan data yang berhubungan.
e. *Redudancy:* menyimpan data dua kali secara redudan untuk membuat sistem berjalan lebih cepat.
f. *Primary key:* yang bersifat unik. Sebuah nilai key tidak dapat digunakan dua kali dalam satu tabel.
g. *Foreign key:* merupakan penghubung antara dua tabel.

h. *Compound key:* atau disebut juga composite key merupakan key yang terdiri dari beberapa kolom.

i. *Indeks:* merupakan indeks dalam database yang menyerupai indeks buku.

j. *Integritas referensial:* digunakan untuk memastikan nilai foreign selalu mengacu pada suatu baris yang ada.(Jubilee Enterprise ; 2014 :7)

III.4. . *Entity Relationship Diagram* (ERD)

II.4.1. *Entity relationship* (ER)

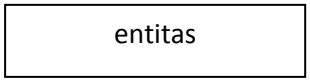
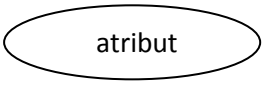
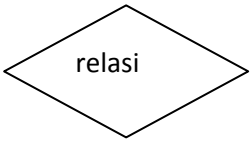
Entity relationship (ER) data model didasarkan pada persepsi terhadap dunia nyata tersusun atas kumpulan objek-objek dasar yang disebut entitas dan hubungan antar objek. Entitas adalah sesuatu atau objek dalam dunia nyata yang dapat dibedakan dari objek lain. Sebagai contoh, masing-masing mahasiswa adalah entitas dan mata kuliah dapat pula dianggap sebagai entitas.

Entitas digambarkan dalam basisdata dengan kumpulan atribut. Misalnya atribut nim, nama, alamat, dan kota bisa menggambarkan data mahasiswa tertentu dalam suatu universitas. Atribut-atribut membentuk entitas mahasiswa. Demikian pula, atribut kodeMK, namaMK, dan SKS mendeskripsikan entitas mata kuliah.

Relasi adalah hubungan antara beberapa entitas. Sebagai contoh, relasi menghubungkan mahasiswa dengan mata kuliah yang diambilnya. Kumpulan semua entitas bertipe sama disebut kumpulan entitas *(entity set)*, sedangkan kumpulan semua relasi bertipe sama disebut relasi (*relationship set).*

Struktur logis (skema database) dapat ditunjukkan secara grafis dengan diagram ER yang dibentuk dari komponen-komponen berikut:

Tabel II.1 ER

Gambar	**keterangan**
	Persegi panjang mewakili kumpulan entitas
	Elips mewakili atribut
	Belah ketupat mewakili relasi
	Garis menghubungkan atribut dengan kumpulan entitas dan kumpulan entitas dengan relasi.

(Sumber : JANNER SIMARMATA ; 2010 : 59)

II.4.2. *Entity Relationship Diagram*

Entity Relationship Diagram adalah alat pemodelan data utama dan akan membantu mengorganisasi data dalam suatu proyek ke dalam entitas-entitas dan menentukan hubungan antar entitas. Proses memungkinkan analis menghasilkan struktur basis data yang baik sehingga data dapat disimpan dan diambil secara efisien.

1. **Entitas *(Entity)***

   Entitas adalah sesuatu yang nyata atau abstrak dimana akan menyimpan data.

2. **Relasi *(Relationship)***

   Relasi adalah hubungan alamiah yang terjadi antara satu atau lebih entitas.

3. **Atribut *(Attribute)***

   Atribut adalah ciri umum semua atau sebagian besar instansi pada entitas tertentu. Sebutan lain atribut adalah properti, elemen data, dan field.

II.5. Kamus Data

Kamus data adalah suatu ensiklopedik dari informasi yang berkaitan dengan data perusahaan. Atau dapat juga kita katakan bahwa kamus data adalah katalog atau *directory* yang berbasis komputer *(computer-based catalog or directory)* yang berisi data perubahan (metadata). Yang berkenaan dengan tahapan penjelasan data ini adalah sistem kamus data *(data dictionary system/DDS)* dan bahasa pendeskripsian data *(data description language/DDL).*

Sistem kamus data berbentuk perangkat lunak yang fungsinya adalah penciptaan dan pemeliharaan serta penyediaan kamus data agar dapat digunakan. Kamus data dapat berbentuk kertas ataupun arsip *(file)* komputer. DDS dapat kita peroleh dalam paket perangkat lunak terpisah ataupun dalam bentuk modul seperti yang ada dalam DBMS *(database management systems)* dan *CASE* (teknik perangkat lunak tambahan komputer/ *computer-aided software engineering).*

(IAN SOMMERVILLE ; 2010 : 344).

II.6. Normalisasi

Normalisasi adalah teknik perancangan yang banyak digunakan sebagai pemandu dalam merancang basisdata relasional. Pada dasarnya, normalisasi adalah proses dua langkah yang meletakkan data dalam bentuk tabulasi dengan menghilangkan kelompok berulang lalu menghilangkan data yang terduplikasi dari tabel relasional(www.utexas.edu).

Tujuan normalisasi adalah membuat kumpulan tabel relasional yang bebas dari data berulang dan dapat dimodifikasi secara benar dan konsisten. Berikut adalah aturan- aturan normalisasi:

1) Hilangkan kelompok berulang: buat tabel terpisah untuk setiap himpunan atribut yang berhubungan dan tentukan kunci utama pada masing-masing tabel.
2) Hilangkan data berulang: jika sebuah atribut hanya tergantung pada sebagian kunci utama gabungan, pindahkan atribut ke tabel lain.
3) Hilangkan kolom yang tidak tergantung pada kunci : jika atribut tidak tergantung pada kunci, pindahkan atribut ke tabel lain.
4) Pisahkan relasi majemuk : tidak ada yang bisa mengandung dua atau lebih relasi 1:n atau n:m yang tidak berhubungan langsung.
5) Pisahkan relasi majemuk yang berhubungan secara semantik : ada batasan pada informasi yang memperbolehkan pemisahanrelasi many-to-many yang berhubungan secara logis.(Janner Simarmata & Iman Praryudi ; 2010 : 62)

Ada beberapa bentuk normalisasi yaitu :

1. Bentuk Normal Pertama (1NF)

   Relasi disebut sebagai 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 1NF adalah sebagai berikut :

   a. Tidak dapat menyisipkan informasi parsial.
   b. Terhapusnya informasi ketika menghapus sebuah *record.*
   c. Pembaruan atribut nonkunci mengakibatkan sejumlah *record* harus diperbarui.

2. Bentuk normal kedua (2NF)

   Relasi disebut sebagai 2NF jika memenuhi kriteria sebagai berikut:

   a. Jika memenuhi kriteria 1NF.
   b. Jika semua atribut nonkunci FD pada PK.

Permasalahan dalam 2NF adalah sebagai berikut.

   a. Kerangkapan data (*data redundacy).*
   b. Pembaruan yang tidak benar dapat menimbulkan inkonsistensi data (*data inconsistency).*
   c. Proses pembaruan data tidak efisien.
   d. Penyimpangan pada saat penyisipan, penghapusan, dan pembaruan.

Kriteria tersebut mengindikasikan bahwa diantara atribut dalam 2NF masih munkin mengalami TDP. Selain itu, relasi 2NF menuntut telah didefinisikan atribut PK dalam relasi. Mengubah relasi 1NF menjadi bentuk 2NF dapat dilakukan dengan mengubah struktur relasi dengan cara :

a. Identifikasikan FD relasi 1NF (jika perlu gambarkan diagram ketergatungan datanya).
b. Berdasarkan informasi tersebut, dekompusi relasi 1NF menjadi relasi-relasi baru sesuai FD-nya. Jika menggunakan diagram maka simpul-simpul yang berada pada puncak diagram ketergantungan data bertindak sebagai PK pada relasi baru.

3. Bentuk normal ketiga (3NF)

Suatu relasi disebut sebagai 3NF jika memenuhi kriteria sebagai berikut :

a. Jika memenuhi kriteria 2NF.
b. Jika setiap atribut nonkunci tidak TDF (*non transitive dependency)* terhadap PK.

Permaslahan dalam 3NF adalah keberadaan penentu yang tidak merupakan bagian dari PK menghasilkan duplikasi rinci data pada atribut yang berfungsi sebgaia FK (duplikai berbeda dengan kerangkapan data).

Mengubah relasi 2NF menjadi bentuk 3NF dapat dilakukan dengan memgubah struktur relasi dengan cara:

a. Mengidentifikasi TDF relasi 2NF.

b.Berdasrakan informasi tersebut, dekomposisi relasi 2Nf menjadi relasi-relasi baru sesuai TDF-nya.

4. Bentuk normal keempat (4NF)

Relasi disebut sebagai 4NF jika memenuhi kriteria sebagai berikut:

a.Jika memenuhi kriteria BCNF.

Jika setiap atribut didalamnya tidak mengalami ketergantungan pada banyak nilai. Atau dengan kalimat lain, bahwa semua atribut yang mengfalami ketergantungan pada banyak nilai adalah bergantung normal BCNF dikemukakan oleh R.F. Boyce dan E.F.Codd. suatu relasi disebut sebagaib BCNF jika memenuhi keriteria sebagai berikut:

b.Jika memenuhi kriteria 3NF.

5. Bentuk normal *Boyce-Cood (BCNF)*

a. Bentuk

6. Bentuk normal kelima (5NF)

a. Suatu relasi memenuhi

Jika semua atribut penentu *(detrminan)* merupakan CK.secara fungsioanl. kriteria 5NF jika kerelaisan antardatadalam relasi tersebut tidak dapat direkonstruksi dari struktur relasi yang sederhana.

II.7. *MySQL*

MySQL adalah RDBMS yang cepat dan mudah digunakan, serta sudah banyak digunakan untuk berbagai kebutuhan . *MySQL* dikembangkan oleh *MySQL* AB Swedia.

Berikut ini hal-hal yang menyebabkan *MySQL* menjadi populer:

1. Berlisensi *open-source,* sehingga anda dapat menggunakannya secar gratis.
2. Merupakan 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 pemrogram seperti PHP, PERL, C, C++, JAVa,dan lain-lain.
5. Bekerja dengan dan baik bahkan dengan data set yang banyak.
6. Sangat mudah digunakan dengan PHP untuk pengembangan aplikasi web.
7. Mendukung banyak database, sampai 50 juta baris atau lebih dalam suatu tabel.
8. Dapat dikostumisasi sesuai dengan keinginan anda. Menurut(Jubilee enterprise ; 2010 : 8).

II.8. Java

Bahasa pemrograman Java dikembangkan oleh *Sun Microsystem* yang dimulai oleh James Gosling dan dirilis pada tahun 1995. Saat ini *Sun Microsystem* telah diakuisisi oleh *Oracle Corporation.*

Java bersifat write once, Run Anywhere (program yang ditulis satu kali dan dapat berjalan pada banyak platform).

a. Konsep PBO (pemrograman berbasis objek)

Java merupakan bahasa pemrograman yang berbasis objek. Berikut ini konsep-konsep dasar pemrograman berbasis objek.

1. Objek
2. Kelas
3. Abstraksi Data
4. Enkapsulasi Data
5. Pewarisan
6. Polimorfisma

b. Fitur-fitur Java

Berikut ini fitur-fitur Java:

a. Berorientasi objek: dalam Java, semua adalah objek.
b. Bersifat Platform Independent: Java di-compile dalam bit kode platform independen dan bukan pada mesin platform spesifik seperti pada C dan C++.
c. Sederhana: Java didesain untuk dapat dengan mudah dipelajari.
d. Aman: Dengan fitur keamanan Java, Anda dapat membuat sistem yang bebas virus dan prowerful.
e. Bersifat *Architectural-neutral*: Compiler Java membuat format fileobjek yang architectural-neutral, yang membuat kode yang decompile dapat dieksekusi pada berbagai prosesor yang memiliki sistem runtime *Java.*
f. *Portable: Java* bersifat portable karena adanya fitur *platform independent* dan *architectural-neutral.*
g. kuat dan powerful: *Java* mengeliminasui error dengan menjalankan pengecekan pada waktu compile dan runtime.

h. Multithreaded: dengan fitur multithread java, anda dapat membuat program yang dapat mengerjakan banyak tugas sekaligus.
i. Terinterprestasi: kode bit java ditranslasi secara langsung pada intruksi mesin dan tidak disimpan.
j. Performa tinggi: Java memilki performa yang tinggi karena menggunakan compiler langsung.
k. Terdistribusi: Java didesain untuk lingkungan ditribusi internet.
l. Dinamis: *Java* lebih dinamis dari C dan C++ karena java didesain untuk beradaptasi dengan lingkungan pengembangan.(Jubilee Enterprise ; 2014 : 1)

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

Hasil pemodelan pada OOAD terdokumentasikan dalam bentuk *unified Modeling Language* (UML). UML adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasi dan membangun perangkat lunak.

UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem.

UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodalan umum dalam industri perangkat lunak dan pengembangan sistem.(Windu dan Grace Gata ; 2013 :4)

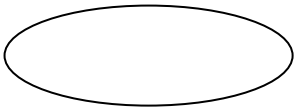
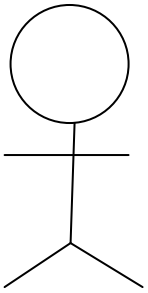
Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut:

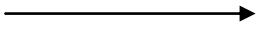
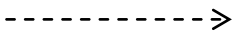
II.9.1. Use Case Diagram

Use Case Diagram merupakan permodelan untuk kelakuan(*behavior*) sistem informasi yang akan dibuat. *Use Case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dapat dikatakan *Use Case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut.

Simbol-simbol yang digunakan dalam *Use Case Diagram* yaitu:

Tabel II.2. Diagram Use Case

Gambar	**Keterangan**
	Use Case menggambarkan fungsional yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktor, biasanya diyatakan dengan menggunakan kata kerja di awal nama Use Case.
	Actor atau aktor adalah abstraction dari orang atau sistem yang lain yang mengatifkan fungsi dari target sistem. Untuk mengidentifikasikan aktor, harus ditentukan pembagian tenaga kerja dan

	tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa muncul dalam beberapa peran. Perlu dicatat bahwa aktor berinteraksi dengan use case, tetapi tidak memiliki kontrol terhadap use case.
	Asosiasi antara aktor dan use case, di gambarkan dngan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengindikasikan aliran data.
	Asosiasi antara aktor dan usecase yang menggunakan panah terbuka untuk mengindikasikan bila aktor berinteraksi secara pasif dengan sistem.
 <<include>>	Include, merupakan di dalam use case lain (required) atau pemanggilan use case oleh use case lain, contohnya adalah pemanggilan sebuah fungsi program.

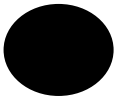
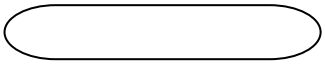
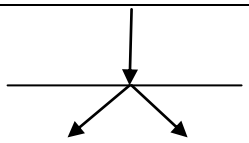
<---------- <<extend>>	Extend merupakan perluasan dari use case lain kondisi atau syarat terpenuhi.

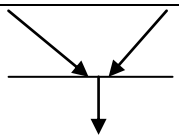
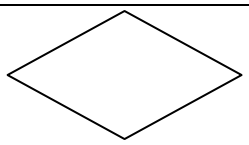
(Sumber : Windu Gata ; 2013 : 4)

II.9.2. Diagram Aktivitas (Activity Diagram)

Activity diagram mengambarkan *workflow* (aliran kerja) aatu aktivitas dari sebuah sistem atau proses bisnis. Simbol-simbol yang digunakan dalam *activity diagram*, yaitu:

Tabel II.3. Diagram Aktivitas

Gambar	**Keterangan**
	Start point diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	End point, akhir aktifitas
	Activities, menggambarkan suatu proses/kegiatan bisnis.
	Fork (percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel atau untuk menggabungkan dua kegiatan paralel menjadi satu.

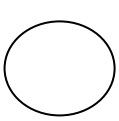
	join (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi.
	Decision points, menggambarkan pilihan untuk pengambilan keputusan, true atau false.
	Swimlame, pembagian activity diagram untuk menunjukkan siapa melakukan apa.

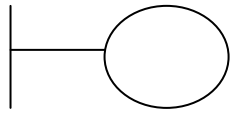
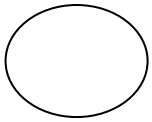
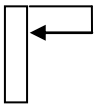
(Sumber : Windu Gata ; 2013 : 6)

II.9.3. Diagram Urutan (*Squence Diagram*)

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Simbol-simbol yang digunakan dalam S*equence Diagram*, yaitu:

Tabel II.4. Diagram Urutan

Gambar	**Keterangan**
	Entity Class, merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data.

	Boundary Class, berisi kumpulan kelas yang menjadi interface atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan formentry dan form cetak.
< 	Control Class, suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek. Control objek mengkoordinir pesan antar boundary dengan entitas.
	Message, simbol mengirim pesan antar Class.
	Recursive, menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri.
	Activation, activation mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi kotak aktivasi sebuah operasi
| |	Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang lifeline

| |	terdapat activation.

(Sumber : Windu Gata ; 2013 : 7)

II.9.4. *Class Diagram (*Diagram Kelas)

Merupakan hubungan antar kelas dan penjelasn detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem.

Class diagram juga menunjukan atribut-atribut dan operasi-operasi dari sebuah kelas dan constraint yang berhubungan dengan objek yang dikoneksikan.

Class diagram secara khas meliputi: kelas (*Class*), relasi, *associations*, *generalization* dan *aggregation*, atribut *(attributes*), operasi (*operations/method*), dan *visibility*, tingkat akses objek eksternal kepada suatu operasi atribut. Hubungan antar kelas mempunyai keterangan yang disebut dengan *Multiplicity* atau kardinaliti

Tabel II.5. *multiplicity Class Diagram*

Multiplicity	**Penjelasan**
1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1 atau lebih
1..*	1 atau lebih
0..1	Boleh tidak ada, maksimal 1
n..n	Batasan antara contoh: 2..4 mempunyai arti minimal 2 maksimum 4

(Sumber : Windu Gata ; 2013 : 9)