

## **BAB II**

### **TINJAUAN PUSTAKA**

#### **II.1. Sistem Pendukung Keputusan**

Sistem pendukung keputusan adalah sebuah alternatif solusi atau alternatif tindakan dari sejumlah alternatif solusi dan tindakan guna menyelesaikan suatu masalah, sehingga masalah tersebut dapat diselesaikan secara efektif dan efisien. Sistem pendukung keputusan berfungsi untuk beberapa hal antara lain, sebagai pemahaman secara komprehensif terhadap masalah, sebagai pemberian kerangka berfikir secara sistematis, dapat membimbing dalam penerapan teknik-teknik pengambilan keputusan, dan meningkatkan kualitas suatu keputusan. (Sylvia Hartati Saragih, 2013).

*Decision Support System* didefinisikan sebagai sistem computer yang mampu memberikan kemampuan baik kemampuan pemecahan masalah maupun kemampuan pengkomunikasian untuk masalah semi terstruktur. Secara khusus, DSS didefinisikan sebagai sebuah sistem yang mendukung kerja seorang manager maupun sekelompok manager dalam memecahkan masalah semi terstruktur dengan cara memberikan informasi ataupun usulan menuju pada keputusan tertentu. DSS ini bisa berbentuk sistem manual maupun sistem terkomputerisasi. Namun dalam buku ini ditekankan pada sistem penunjang keputusan yang pelaksanaannya berbasis pada komputer.

Dari definisi diatas bisa disimpulkan bahwa tujuan DSS dalam proses pengambilan keputusan adalah :

1. Membantu menjawab masalah semi-terstruktur
2. Membantu manajer dalam mengambil keputusan, bukan menggantikannya
3. Meningkatkan efektifitas pengambilan keputusan seorang manajer dari pada efisiensinya.

### **II.1.1. Karakteristik Sistem Pendukung Keputusan**

Karakteristik dari sistem pendukung keputusan (*Decision Support System*) adalah sebagai berikut :

1. Tujuan utama dari Sistem Pendukung Keputusan adalah untuk memperbaiki mutu keputusan serta performance. Sistem Pendukung Keputusan tidak hanya sekedar menyajikan informasi yang lebih banyak, lebih baik dan lebih akurat pada waktu yang tepat saja.
2. Sistem Pendukung Keputusan ditujukan untuk environment yang kompleks, kurang terstruktur dan bahkan politis sifatnya.
3. Sistem Pendukung Keputusan bertumpu pada laporan perkecualian dan macamnya untuk menunjang proses identifikasi masalah.
4. Sistem Pendukung Keputusan berkombinasi “*modelling*” dan teknik-teknik analisa yang lain dengan fungsi penyajian kembali data.
5. Sistem Pendukung Keputusan berfokus pada prinsip “mudah dipakai” dan “fleksibel” dalam berhadapan dengan pemakai tertentu atau sekelompok pemakai.
6. Proses Pengambilan Keputusan.

### II.1.2. Konfigurasi Sistem Pendukung Keputusan

Terdapat 3 komponen utama, yaitu data *management*, model *management*, dan *user interface*.

Aliran kerja dari DSS ini adalah sebagai berikut :

1. Data management melakukan pengambilan data yang diperlukan baik dari *database* yang berisi data internal maupun *database* yang berisi data eksternal. Jadi, fungsi komponen data disini jelas sebagai penyedia data yang diperlukan oleh sistem DSS.
2. Model management melalui model base management melakukan interaksi baik dengan *user interface* untuk mendapatkan perintah maupun data management untuk mendapatkan data yang akan diolah. Model base management akan menggunakan model base yang berisi model-model yang digunakan dalam DSS yang dengan bantuan perangkat lunak atau engine yang ada di dalam modeling tools melakukan pengolahan data yang kemudian hasilnya dikembalikan lewat model base management untuk dikirim ke *user interface*.
3. *User interface* digunakan untuk berinteraksi antara user dengan DSS, baik untuk memasukkan informasi ke sistem maupun menampilkan informasi ke user. Karena begitu pentingnya komponen *user interface* bagi suatu sistem DSS, maka kita harus bisa merancang suatu *user interface* yang bisa mudah dipelajari dan digunakan user dan laporan yang bisa secara mudah dimengerti oleh pengguna. (Edo, 2013 : 4).

## **II.2. Metode *Analytic Hierarchy Process* (AHP)**

Metode *Analytic Hierarchy Process* (AHP) merupakan suatu model pendukung keputusan yang dikembangkan oleh Thomas L. Saaty. Model pendukung keputusan ini akan menguraikan masalah multi faktor atau multi kriteria yang kompleks menjadi suatu hirarki. Hirarki didefinisikan sebagai suatu representasi dari sebuah permasalahan yang kompleks dalam suatu struktur multi level. (Sylvia Hartati Saragih, 2013 : 83).

### **II.2.1. Tahapan-Tahapan Dalam *Analytic Hierarchy Process* (AHP)**

Berikut ini adalah tahapan-tahapan yang terdapat pada metode *Analytic Hierarchy Process* (AHP) :

1. Mendefinisikan masalah dan menentukan solusi yang diinginkan.
2. Membuat struktur hierarki yang diawali dengan tujuan umum, dilanjutkan dengan kriteria-kriteria dan alternatif- alternatif pilihan.
3. Membuat matrik perbandingan berpasangan yang menggambarkan kontribusi relatif atau pengaruh setiap elemen terhadap tujuan atau kriteria yang setingkat di atasnya. Perbandingan dilakukan berdasarkan pilihan atau *judgement* dari pembuat keputusan dengan menilai tingkat kepentingan suatu elemen dibandingkan elemen lainnya.
4. Menormalkan data yaitu dengan membagi nilai dari setiap elemen di dalam matrik yang berpasangan dengan nilai total dari setiap kolom.
5. Menghitung nilai *eigen vector* dan menguji konsistensinya, jika tidak konsisten maka pengambilan data (preferensi) perlu diulangi. Nilai *eigen*

*vector* yang dimaksud adalah nilai *eigen vector* maksimum yang diperoleh.

6. Mengulangi langkah 3, 4 dan 5 untuk seluruh tingkat hirarki.
7. Menghitung *eigen vector* dari setiap matriks perbandingan berpasangan. Nilai *eigen vector* merupakan bobot setiap elemen.
8. Menguji konsistensi hirarki. Jika tidak memenuhi dengan  $CR < 0,100$  maka penilaian harus diulangi kembali. (Darmanto, dkk, 2014 : 78).

### **II.3. Visual Basic 2010**

*Visual Basic 2010* merupakan versi perbaikan dan pengembangan dari versi pendahulunya, yaitu *Visual Basic 2008*. Beberapa pengembangan yang terdapat didalamnya antara lain dukungan terhadap *library* terbaru dari *Microsoft*, yaitu *.Net Framework 4.0*, dukungan terhadap aplikasi berbasis *Cloud Computing*, serta perluasan dukungan terhadap *database-database*, baik *standalone* maupun *database server*. (Sari, dkk, 2015 : 2).

### **II.4. SQL Server 2008**

*SQL Server 2008* adalah sebuah terobosan baru dari *Microsoft* dalam bidang *database*. *SQL Server* adalah *DBMS (Database Management System)* yang dibuat oleh *Microsoft* untuk ikut berkecimpung dalam persaingan dunia pengolahan data menyusul pendahulunya seperti *IBM* dan *Oracle*. *SQL Server 2008* dibuat pada saat kemajuan dalam bidang *hardware* sedemikian pesat. Oleh karena itu sudah dapat dipastikan bahwa *SQL Server 2008* membawa

beberapa terobosan dalam bidang pengolahan dan penyimpanan data. (Widya dan Zulkarnaen, 2012 : 3).

## II.5. Normalisasi

Normalisasi merupakan parameter digunakan untuk menghindari duplikasi terhadap tabel dalam basi data dan juga merupakan proses mendekomposisikan sebuah tabel yang masih memiliki beberapa anomali atau ketidakwajaran sehingga menghasilkan tabel yang lebih sederhana dan struktur yang bagus, yaitu sebuah tabel yang tidak memiliki *data redundancy* dan memungkinkan *user* untuk melakukan *insert*, *delete*, dan *update* pada baris(*record*) tanpa menyebabkan inkonsistensi data. (Triyono, 2012 : 19).

### 1. First Normal Form (1 NF)

Sudah tidak ada *repeating group* yaitu pengulangan yang terjadi pada beberapa atribut atau kolom dalam sebuah tabel, dan juga setiap atribut harus bernilai tunggal. Atribut *multivalued*, *composite*, *derive* tidak tunggal. Setiap nilai dari atribut hanya mempunyai nilai tunggal.

### 2. Second Normal Form (2 NF)

Untuk menjadikan tabel normal tingkat ke 2 maka sudah 1NF dan setiap atribut yang bukan *primary key* sepenuhnya secara *funksional* tergantung pada semua atribut pembentuk *primary key*.

### 3. Third Normal Form (3 NF)

Tabel sudah 2NF dan tidak memiliki *transitive dependencies*, *Transitive dependency* adalah ketika ada atribut yang secara tidak langsung

tergantungan pada *primary key* dan atribut tersebut juga tergantung pada atribut lain yang bukan *primary key*.

#### 4. *Boyce-codd Normal Form (BCNF)*

Tabel dalam BCNF jika sudah 3NF dan semua *determinants* adalah *candidate keys*. Perbedaan 3NF dan BCNF adalah untuk *functional dependency*  $A \rightarrow B$ , 3NF memperbolehkan ketergantungan ada dalam relasi jika B adalah *Primary Key* dan A bukan merupakan *candidate key*. Sedangkan BCNF menuntut untuk ketergantungan tetap ada dalam relasi, A harus menjadi *candidate key*.

#### 5. *Fourth Normal Form (4 NF)*

Relasi berada pada bentuk normal keempat apabila memenuhi syarat BCNF dan tidak mempunyai *multivalued dependency*.

#### 6. *Fifth Normal Form (5 NF)*

Tabel bentuk normal kelima sering disebut PJNF (*Projection Join Normal Form*), penyebutan PJNF karena untuk suatu relasi akan berbentuk normal kelima jika tabel tersebut dapat dipecah atau diproyeksikan menjadi beberapa tabel dan dari proyeksi-proyeksi itu dapat disusun kembali (*join*) menjadi tabel yang sama dengan keadaan semula. Jika penyusunan ini tidak mungkin dilakukan dikatakan pada relasi itu terdapat *join dependencies* dan dikatakan bersifat *lossy join*. (Triyono, 2012 : 20).

## II.6. *Unified Modeling Language (UML)*

Menurut Windu Gata (2013) Hasil pemodelan pada OOAD terdokumentasikan dalam bentuk *Unified Modeling Language (UML)*. UML adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak.


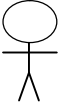

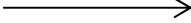
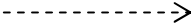
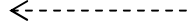
UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodelan umum dalam industri perangkat lunak dan pengembangan sistem. (Urva dan Siregar, 2015, Hal : 93).

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

### 1. *Use case Diagram*

*Use case* diagram merupakan pemodelan 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 dapat dilihat pada tabel II.1 dibawah ini:

Tabel II.1. Simbol *Use Case*




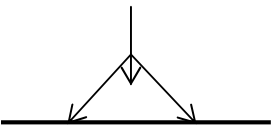
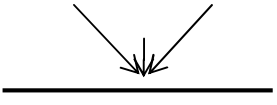
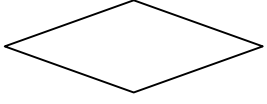

Gambar	Keterangan
	<i>Use case</i> menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama <i>use case</i> .
	Aktor adalah <i>abstraction</i> dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasi 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 <i>use case</i> , tetapi tidak memiliki control terhadap <i>use case</i> .
	Asosiasi antara aktor dan <i>use case</i> , digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengindikasikan aliran data.
	Asosiasi antara aktor dan <i>use case</i> yang menggunakan panah terbuka untuk mengindikasikan bila aktor berinteraksi secara pasif dengan sistem.
	<i>Include</i> , merupakan di dalam <i>use case</i> lain ( <i>required</i> ) atau pemanggilan <i>use case</i> oleh <i>use case</i> lain, contohnya adalah pemanggilan sebuah fungsi program.
	<i>Extend</i> , merupakan perluasan dari <i>use case</i> lain jika kondisi atau syarat terpenuhi.

(Sumber:Urva dan Siregar; 2015, Hal : 94)

## 2. Diagram Aktivitas (*Activity Diagram*)

*Activity Diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Simbol-simbol yang digunakan dalam *activity diagram* dapat dilihat pada tabel II.2 dibawah ini :

Tabel II.2. Simbol *Activity Diagram*

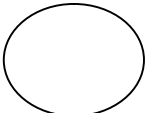
Gambar	Keterangan
	<i>Start point</i> , diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	<i>End point</i> , akhir aktifitas.
	<i>Activites</i> , menggambarkan suatu proses/kegiatan bisnis.
	<i>Fork</i> (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan pararel menjadi satu.
	<i>Join</i> (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi.
	<i>Decision Points</i> , menggambarkan pilihan untuk pengambilan keputusan, <i>true</i> , <i>false</i> .
	<i>Swimlane</i> , pembagian <i>activity diagram</i> untuk menunjukkan siapa melakukan apa.

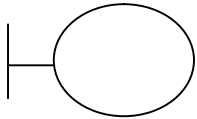
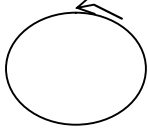

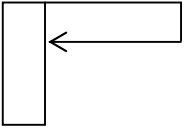


(Sumber : Urva dan Siregar; 2015, Hal : 94)

### 3. Diagram Urutan (*Sequence 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 *sequence diagram* dapat dilihat pada tabel II.3 dibawah ini :

Tabel II.3. Simbol *Sequence Diagram*

Gambar	Keterangan
	<i>EntityClass</i> , merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data.

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

(Sumber : Urva dan Siregar; 2015, Hal : 95)

#### 4. *Class Diagram* (Diagram Kelas)

*Class diagram* merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem. *Class diagram* juga menunjukkan 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*), *Visibility*, tingkat akses objek eksternal kepada suatu

operasi atau atribut. Hubungan antar kelas mempunyai keterangan yang disebut dengan *multiplicity* atau kardinaliti yang dapat dilihat pada tabel II.4 dibawah ini :

**Tabel II.4. *Multiplicity Class Diagram***

<b>Multiplicity</b>	<b>Penjelasan</b>
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 : Urva dan Siregar; 2015, Hal : 95)**