BAB II

TINJAUAN PUSTAKA

II.1. Sistem

Sistem adalah sekumpulan elemen yang saling terkait atau terpadu yang dimaksudkan untuk mencapai suatu tujuan. Sebagai gambaran, jika dalam sebuah sistem terdapat elemen yang tidak memberikan manfaat dalam mencapai tujuan yang sama, maka elemen tersebut dapat dipastikan bukanlah bagian dari sistem. (Abdul Kadir, 2014, Hal : 61).

II.2. Sistem Pendukung Keputusan

Sistem pendukung keputusan (SPK) atau *Decision Support Systems (DSS)* adalah sistem informasi interaktif yang menyediakan informasi, pemodelan, dan pemanipulasian data yang digunakan untuk membantu pengambilan keputusan pada situasi yang semiterstruktur dan situasi yang tidak terstruktur di mana tek seorang pun tahu secara pasti bagaimana keputusan seharusnya dibuat. Konsep *DSS* dikemukakan pertama kali oleh Scott-Morton pada tahun 1971. Beliau mendefenisikan cikal bakal *DSS* tersebut sebagai ”Sistem berbasis komputer interaktif, yang membantu pengambil keputusan menggunakan data dan model untuk memecahkan persoalan-persoalan tidak terstruktur”.

DSS dibuat sebagai reaksi atas ketidakpuasan terhadap TPS dan MIS. Sebagaimana diketahui, TPS lebih memfokuskan diri dari pada perekaman dan pengendalian transaksi yang merupakan kegiatan yang bersifat berulang dan terdefenisi dengan baik, sedangkan MIS lebih berorientasi pada penyediaan

laporan bagi manajemen yang sifatnya tidak fleksibel. *DSS* lebih ditujukan untuk mendukung manajemen dalam melakukan pekerjaan yang bersifat analitis, dalam situasi yang kurang terstruktur dan dengan kriteria yang kurang jelas. *DSS* tidak dimaksudkan untuk mengotomasikan pengambilan keputusan, tetapi memberikan perangkat interaktif yang memungkinkan pengambil keputusan dapat melakukan berbagai analisis dengan menggunakan model-model yang tersedia. (Abdul Kadir, 2014, Hal : 108).

Adapun beberapa karakteristik yang terdapat pada sistem pendukung keputusan adalah sebagai berikut :

- Menawarkan keluasan, kemudahan beradaptasi, dan tanggapan yang cepat.
- Memungkinkan pemakai memulai dan mengendalikan masukan dan keluaran.
- Dapat dioperasikan dengan sedikit atau tanpa bantuan pemrogram professional.
- Menyediakan dukungan untuk keputusan dan permasalahan yang solusinya tak dapat ditentukan di depan.
- Menggunakan analisis data dan perangkat pemodelan yang canggih. (Abdul Kadir, 2014, Hal : 108).

II.2.1. Komponen Sistem Pendukung Keputusan

Sistem Pendukung Keputusan mempunyai 3 komponen utama yaitu dialog manajemen, model manajemen dan data manajemen.Ketiga komponen ini merupakan komponen utama dari Sistem Pendukung Keputusan. Komponen pertama adalah *dialog management* atau *user interface* yaitu komponen untuk berdialog dengan pemakai sistem. Komponen ini didalah sistem informasi

merupakan komponen input dan komponen *output*. Komponen kedua dari SPK adalah model *management*, yaitu komponen yang merubah data menjadi informasi yang relevan. Model-model yang banyak digunakan di Sistem Pendukung Keputusan adalah model matematik optimisasi seperti *linier programming*, *dynamic programming* dan lain sebagainya. Komponen ketiga adalah *data management*, yaitu komponen basis data yang terdiri dari semua basis data yang dapat diakses. Seperti halnya sistem informasi pada umumnya, sistem pendukung keputusan juga mempunyai komponen teknologi dan kontrol. Komponen teknologi terdiri dari perangkat keras dan perangkat lunak. Perangkat lunak spesifik yang digunakan oleh SPK misalnya spreadsheet, DBMS, bahasa query. (Fahmi Maulana, 2013, Hal : 50).

II.3. Metode TOPSIS

TOPSIS adalah salah satu metode pengambilan keputusan multikriteria yang pertama kali diperkenalkan oleh Yoon dan Hwang. TOPSIS menggunakan prinsip bahwa alternative yang terpilih harus mempunyai jarak dekat dari solusi ideal positif dan terjauh dari solusi ideal negatif dari sudut pandang geometris dengan menggunakan jarak *Euclidean* untuk menentukan kedekatan relative dari suatu alternatif dengan solusi optimal. Solusi ideal positif didefinisikan sebagai jumlah dari seluruh nilai terbaik yang dapat dicapai untuk setiap atribut, sedangkan solusi negatif-ideal terdiri dari seluruh nilai terburuk yang dicapai untuk setiap atribut. TOPSIS mempertimbangkan keduanya, jarak terhadap solusi ideal positif dan jarak terhadap solusi ideal negatif dengan mengambil kedekatan relatif terhadap solusi ideal positif. Berdasarkan perbandingan terhadap jarak

relatifnya, susunan prioritas alternatif bisa dicapai. Metode ini banyak digunakan untuk menyelesaikan pengambilan keputusan. Hal ini disebabkan konsepnya sederhana, mudah dipahami, komputasinya efisien, dan memiliki kemampuan mengukur kinerja relatif dari alternatifalternatif keputusan. (Desi Leha Kurniasih, 2013, Hal : 8).

Adapun langkah-langkah metode TOPSIS adalah sebagai berikut :

1. Membangun *normalized decision matrix* Elemen *rij* hasil dari normalisasi *decision matrix R* dengan metode *Euclidean length of a vector* adalah :

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i}^{m} = 1^{x_{ij}^2}}}$$

Dimana :

Rij = hasil dari normalisasi matriks keputusan R

i = 1,2,3,...,m;

j = 1,2,3,...,n;

2. Membangun *weighted normalized decision matrix* Dengan bobot *W* = (*w1,w2,....,wn*), maka normalisasi bobot matriks *V* adalah :

$$V = \begin{bmatrix} w_{11}r_{11} & \cdots & w_{1n}r_{1n} \\ \vdots & \ddots & \vdots \\ w_{m1}r_{m1} & \cdots & w_{nm}r_{nm} \end{bmatrix}$$

3. Menentukan solusi ideal positif dan solusi ideal negatif

Solusi ideal positif dinotasikan dengan A+ dan solusi ideal negatif dinotasikan dengan A-, sebagai berikut :

Menentukan solusi ideal (+) dan (-)

$$A^{+} = \left\{\left(\max v_{ij} \middle| j \in J\right)\left(\min v_{ij} \middle| j \in J'\right), i = 1,2,3,...m\right\} = \left\{v_1^{+}, v_2^{+}, \cdots v_m^{+}\right\}$$
$$A^{-} = \left\{\left(\max v_{ij} \middle| j \in J\right)\left(\min v_{ij} \middle| j \in J'\right), i = 1,2,3,...m\right\} = \left\{v_1^{-}, v_2^{-}, \cdots v_m^{-}\right\}$$

Dimana :

$v = ij$

elemen matriks *V* baris ke-*i* dan kolom ke*j*

J = {j=1,2,3,...,n dan j berhubungan dengan

benefit criteria}

{j=1,2,3,...,n dan j berhubungan dengan

cost criteria}

4. Menghitung separasi

Separation measure ini merupakan pengukuran jarak dari suatu alternatif ke solusi ideal positif dan solusi ideal negatif. Perhitungan matematisnya adalah sebagai berikut :

Separation measure untuk solusi ideal positif, dengan *i* = 1,2,3,...,m

$$S_i^{+} = \sqrt{\sum_{j=1}^{n}\left(v_{ij} - v_j^{+}\right)^2}\ , \text{dengan } i = 1,2,3,...,\text{m}$$

Dimana :

J = {j=1,2,3,...,n dan j merupakan *benefit criteria*}

J' = {j=1,2,3,...,n dan j merupakan *cost criteria*}

Separation measure untuk solusi ideal negatif

$$S_i^- = \sqrt{\sum_{j=1}^{n} \left(v_{ij} - v_j^-\right)^2}$$, dengan $i = 1,2,3,...,n$

Dimana :

J = {j=1,2,3,...,n dan j merupakan *benefit criteria*}

J' = {j=1,2,3,...,n dan j merupakan *cost criteria*}

5. Menghitung kedekatan relatif terhadap solusi ideal

Kedekatan relatif dari alternatif A+ dengan solusi ideal A- direpresentasikan dengan :

$$C_i = \frac{S_i^-}{S_i^- + S_i^+}$$, dengan $0 < C_i^+ < 1$ dan $i = 1,2,3,...,m$

6. Merangking alternatif

Alternatif dapat diranking berdasarkan urutan *Ci* *. Maka dari itu, alternatif terbaik adalah salah satu yang berjarak terpendek terhadap solusi ideal dan berjarak terjauh dengan solusi ideal negatif. (Desi Leha Kurniasih, 2013, Hal : 89).

II.4. Basis Data

Secara harifah *database* merupakan sekumpulan data yang tersusun dengan aturan tertentu dalam bentuk tabel. Adapun secara fungsi, *database* merupakan suatu tempat yang dipergunakan untuk menyimpan sekumpulan data dalam format tertentu. Saat ini terdapat puluhan jenis *database* yang secara aktif dikembangkan dan dipergunakan dalam aplikasi. Dilihat dari lokasi penyimpanan

data, *database* dibagi menjadi dua, yaitu *database* lokal dan *database server*. Contoh *database* yang termasuk dalam *database* lokal diantaranya *SQLite*, *Dbase*, *Firebird*, dan *Paradox*. Adapun contoh *database* yang termasuk dalam *database server* di antaranya adalah *MySQL*, *Oracle*, *SQL Server*, *Interbase*, dan *PostgreSQL*. (Wahana Komputer, 2011).

Basis data dapat didefenisikan sebagai koleksi dari data-data yang terorganisasi sedemikian rupa sehingga data mudah disimpan dan dimanipulasi (diperbarui, dicari, diolah dengan perhitungan-perhitungan tertentu, serta dihapus). Secara teoritis, basis data tidak harus berurusan dengan komputer (misalnya, catatan belanja hari ini yang dibuat oleh seorang ibu rumah tangga juga merupakan basis data dalam bentuk yang sangat sederhana). (Adi Nugroho, 2011).

Menurut Abdul Kadir (2014) basis data (*database)* adalah suatu pengorganisasian sekumpulan data yang saling terkait sehingga memudahkan aktifitas untuk memperoleh informasi. Basis data dimaksudkan 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. (Abdul Kadir, 2014, Hal : 218).

Umumnya *DBMS* menyediakan fitur-fitur sebagai berikut :

- Independensi data-program

  Karena basis data ditangani oleh *DBMS*, program dapat dipilih sehingga tidak tergantung pada struktur data dalam basis data. Dengan perkataan lain, program tidak akan terpenaruh sekiranya bentuk fisik data diubah.

- Keamanan

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

- Integritas

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

- Konkurensi

  Konkurensi memungkinkan data dapat diakses oleh banyak pemakai tanpa menimbulkan masalah.

- Pemulihan (*recovery*)

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

- Katalog Sistem

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

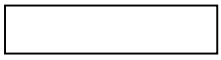
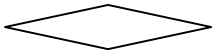
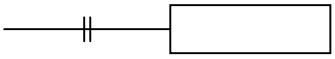
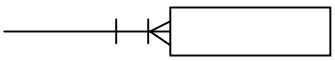
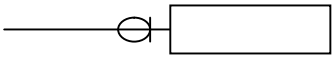
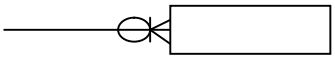
- Perangkat Produktivitas

Untuk menyediakan kemudahan bagi pemakai dan meningkatkan produktivitas, *DBMS* menyediakan sejumlah perangkat produktivitas seperti pembangkit *query* dan pembangkit laporan. (Abdul Kadir, 2014, Hal : 219).

II.5. *Entity Relationship Diagram*

Entity Relationship Diagram (ERD) adalah bagian yang menunjukkan hubungan antara entity yang ada dalam sistem. Simbol-simbol yang digunakan dapat dilihat dari tabel II.6. (Yuhendra, M.T, Dr. Eng dan Riza Eko Yulianto, 2015, Hal : 70).

Tabel II.6. Simbol Yang Digunakan Pada *Entity Relationship Diagram*

SIMBOL	KETERANGAN
	*Entity*
	Atribut Dan *Entity*
	Atribut Dan *Entity* Dengan *Key* (Kunci)
	Relasi Atau Aktifitas Antar *Entity*
	Hubungan Satu Dan Pasti
	Hubungan Banyak Dan Pasti
	Hubungan Satu Tapi Tidak Pasti
	Hubungan Banyak Tapi Tidak Pasti

(Sumber : Yuhendra, M.T, Dr. Eng dan Riza Eko Yulianto; 2015, Hal : 70)

II.6. Kamus Data

Kamus data merupakan sebuah daftar yang terorganisasi dari elemen data yang berhubungan dengan sistem, dengan definisi yang tegar dan teliti sehingga

pemakai dan analis sistem akan memiliki pemahaman yang umum mengenai *input*, *output*, komponen penyimpanan. (Yusi Ardi Binarso, 2012).

II.7. Tahapan Normalisasi

Normalisasi dapat dipahami sebagai tahapan-tahapan yang masing-masing berhubungan dengan bentuk normal. Bentuk normal adalah keadaan relasi yang dihasilkan dengan menerapkan aturan sederhana berkaitan dengan konsep kebergantungan fungsional pada relasi yang bersangkutan. Kita akan menggambarkannya secara garis besar sebagai berikut :

1. Bentuk Normal Pertama (1NF/ *First Normal Form*)

   Bentuk normal pertama adalah suatu bentuk relasi dimana atribut bernilai banyak (*multivalues attribute*) telah dihilangkan sehingga kita akan menjumpai nilai tunggal (mungkin saja nilai *null*) pada perpotongan setiap baris dan kolom.

2. Bentuk Normal Kedua (2ND/ *Second Normal Form*)

   Semua kebergantungan fungsional yang bersifat sebagian (*particial functional dependency*) telah dihilangkan.

3. Bentuk Normal Ketiga (3RD/ *Thrid Normal Form*)

   Semua kebergantungan transitif (*transitive dependency*) telah dihilangkan.

4. Bentuk Normal *Boyce-Codd* (BCNF/ *Boyce-Codd Normal Form*)

   Semua anomaly yang tersisa dari hasil penyempurnaan kebergantungan fungsional sebelumnya telah dihilangkan.s

5. Bentuk Normal Keempat (4NF/ *Fourth Normal Form*)

   Semua kebergantungan bernilai banyak telah dihilangkan.

6. Bentuk Normal Kelima (5NF/ *Fifth Normal Form*)

   Semua anomaly yang tertinggi telah dihilangkan. (Adi Nugroho, 2011, Hal : 199).

II.8. *SQL Server 2008*

SQL Server 2008 adalah sebuah *RDBMS (Relational Database Management System)* yang sangat powerful dan telah terbukti kekuatannya dalam mengolah data. Dalam versi terbarunya ini, *SQL Server* 2008 memiliki banyak fitur yang bisa dihandalkan untuk meningkatkan performa *database. SQL Server* 2008 memiliki suatu *GUI (Graphic User Interface)* yang kita gunakan untuk melakukan aktivitas sehari hari berkaitan dengan database,seperti menulis *T-SQL*, melakukan *backup* dan *restore database*, melakukan *security database* terhadap aplikasi, dan sebagainya. Pada *GUI* tersebut kita bisa melakukan settingan terhadap *SQL Serve*r untuk berkerja lebih optimal.Settingan juga bisa dilakukan menggunakan *script* untuk memudahkan developer mengubah *Setting Options* pada *SQL Server* 2008. (Ruslan, 2013, Hal : 39).

II.9. *Microsoft Visual Basic 2010*

Visual Basic 2010 merupakan salah satu bagian dari produk pemrograman terbaru yang dikeluarkan oleh *Microsoft*, yaitu *Microsoft Visual Studio 2010. Visual Studio* merupakan produk pemrograman andalan dari *mocrosoft corporation*, dimana di dalamnya berisi beberapa jenis *IDE* pemrograman seperti *Visual Basic*, *Visual C++*, *Visual Web Developer*, *Visual C#,* dan *Visual F#.*

Semua *IDE* pemrograman tersebut sudah mendukung penuh implementasi *.Net Framework* terbaru, yaitu *.Net Framework 4.0* yang merupakan pengembangan dari *.Net Framework 3.5*. Adapun database standar yang disertakan adalah *Microfot SQL Server 2008 express*.

Visual Basic 2010 merupakan versi perbaikan dan pengembangan dari versi pendahulunya yaitu *visual basic 2008*. Beberapa pengembangan yang terdapat di dalamnya antara lain dukungan terhadap *library* terbaru dari *Microsoft*, yaitu *.Net Framework 4.0,* dukungan terhadap pengembangan aplikasi menggunakan *Microsoft SilverLight*, dukungan terhaddap aplikasi berbasis *cloud computing*, serta perluasan dukungan terhadap *database-database*, baik *standalone* maupun *database server.* (Wahana Komputer, 2011, Hal : 2).

II.10. *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.

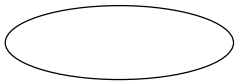
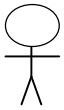
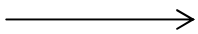
UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. *UML* saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodelan umum dalam industri perangkat lunak dan pengembangan sistem. (Gellysa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

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

1. *Use case* Diagram

Use case diagram merupakan pemodelan untuk kelakukan (*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.2 dibawah ini :

Tabel II.10.1. Simbol *Use Case*

Gambar	Keterangan
	*Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama *use case*.
	Aktor adalah *abstraction* dari orang atau sistem yang lain yang mengaktifkan 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 control terhadap *use case*.
	Asosiasi antara aktor dan *use case*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengidikasikan aliran data.
	Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengidinkasikan bila aktor berinteraksi secara pasif dengan sistem.

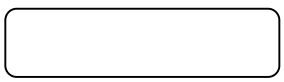
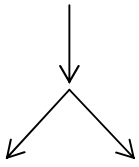
	Include, merupakan di dalam *use case* lain (*required*) atau pemanggilan *use case* oleh *use case* lain, contohnya adalah pemanggilan sebuah fungsi program.
	Extend, merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.

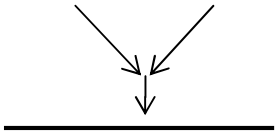
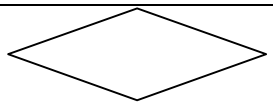
(Sumber : Gellysa Urva dan Helmi Fauzi Siregar; 2015, Hal : 94)

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.10.2. Simbol *Activity Diagram*

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

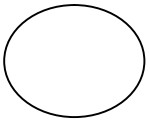
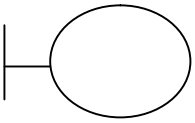
	*Join* (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi.
	*Decision Points,* menggambarkan pilihan untuk pengambilan keputusan, *true, false.*
	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa.

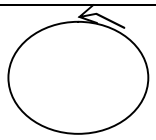
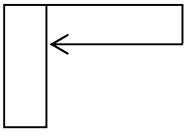
(Sumber : Gellysa Urva dan Helmi Fauzi Siregar; 2015, Hal : 94)

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.4 dibawah ini :

Tabel II.10.3. Simbol *Sequence Diagram*

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.
	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 aktivitas sebuah operasi.
	Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation*.

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

4. *Class Diagram* (Diagram Kelas)

Merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggng 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.5 dibawah ini :

Tabel II.10.4. *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 : Gellysa Urva dan Helmi Fauzi Siregar; 2015, Hal : 95)