BAB II

TINJAUAN PUSTAKA

II.1. Sistem

Secara sederhana, suatu sistem dapat diartikan sebagai 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 Kannet Boulding,terutama menekan pentingnya perhatian terhadap setiap bagian yang membentuk sebuah sistem. Kecenderungan manusia yang dapat tugas memimpin suatu organisasi adalah terlalu memusatkan perhatian pada salah satu komponen saja dari sistem organisasi. (Siti Kholijah Ritonga, 2013:2)

II.2. Keputusan

Keputusan merupakan suatu reaksi terhadap beberapa solusi alternatif yang dilakukan secara sadar dengan cara menganalisa kemungkinan-kemungkinan dari alternatif tersebut bersama konsekuensinya. Pengambilan keputusan merupakan suatu pendekatan sistematis terhadap hakikat suatu masalah, pengumpulan fakta-fakta, penentu yang matang dari alternatif yang dihadapi, dan pengambilan tindakan yang menurut perhitungan (Putu Angga Septiana Putra dkk : 2016).

II.3. Sistem Pendukung Keputusan

Sistem pendukung keputusan merupakan sistem informasi interaktif yang menyediakan informasi, pemodelan dan memanipulasi data. Sistem itu digunakan untuk membantu pengambilan keputusan dalam situasi yang semiterstruktural dan situasi yang tidak terstruktur dimana tak seorang pun tahu sacara pasti bagaimana keputusan seharusnya dibuat. Sistem pendukung keputusan biasanya dibangun untuk mendukung solusi atas suatu masalah atau untuk mengevaluasi suatu peluang. Sistem pendukung keputusan yang seperti itu disebut aplikasi Sistem pendukung keputusan. Aplikasi Sistem pendukung keputusan digunakan dalam pengambilan keputusan. Aplikasi menggunakan CBIS (*Computer Based Information System*) yang fleksibel, interaktif, dan dapat diadaptasi,yang dikembangkan untuk mendukung solusi atas masalah menajemen yang tidak terstruktur.

Aplikasi Sistem pendukung keputusan menggunakan data, memberikan antar muka pengguna yang mudah,dan dapat menggabungkan pemikiran pengambilan keputusan. Sistem pendukung keputusan lebih ditujukan untuk mendukung manajemen dalam melakukan pekerjaan yang bersifat analitis dalam situasi yang kurang terstruktur dan dengan kriteria yang kurang jelas. Sistem pendukung keputusan tidak dimaksudkan untuk mengotomatisasikan pengambilan keputusan, tetapi memberikan perangkat interaktif yang memungkinkan pengambilan keputusan untuk melakukan berbagai analisis menggunakan model-model yang tersedia.

1. Membantu manajer dalam pengambilan keputusan atas masalah semistuktur.

2. Memberikan dukungan atas pertimbangan manajer dan bukannya dimaksudkan untuk menggantikan fungsi manajer.
3. Meningkatkan efektifitas keputusan yang diambil lebih daripada perbaikan efisiensinya.
4. Kecepatan komputasi. Komputer memungkinkan para pengambil keputusan untuk melakukan banyak komputansi secara cepat dengan biaya rendah.
5. Peningkatan produktivitas.
6. Dukungan kualitas.
7. Berdaya saing.
8. Mengatasi keterbatasan kognitif dalam pemrosesan dan penyimpanan. (Sylvia Hartati Saragih, 2013:2)

II.3.1. Karakteristik SPK (Sistem Pendukung Keputusan)

Beberapa karakteristik yang membedakan Sistem Pendukung Keputusan dengan sistem informasi yaitu :

1. Sistem Pendukung Keputusan dirancang untuk membantu pengambilan keputusan dalam memecahkan masalah yang sifatnya semi terstruktur ataupun tidak terstruktur.
2. Dalam proses pengolahannya, sistem pendukung keputusan mengkombinasikan model-model analisis dengan teknik pemasukan dan konvensional secara fungsi fungsi pencarian informasi.
3. Sistem Pendukung Keputusan dirancang sedemikian rupa sehingga dapat digunakan atau dioperasikan dengan mudah oleh orang-orang yang tidak

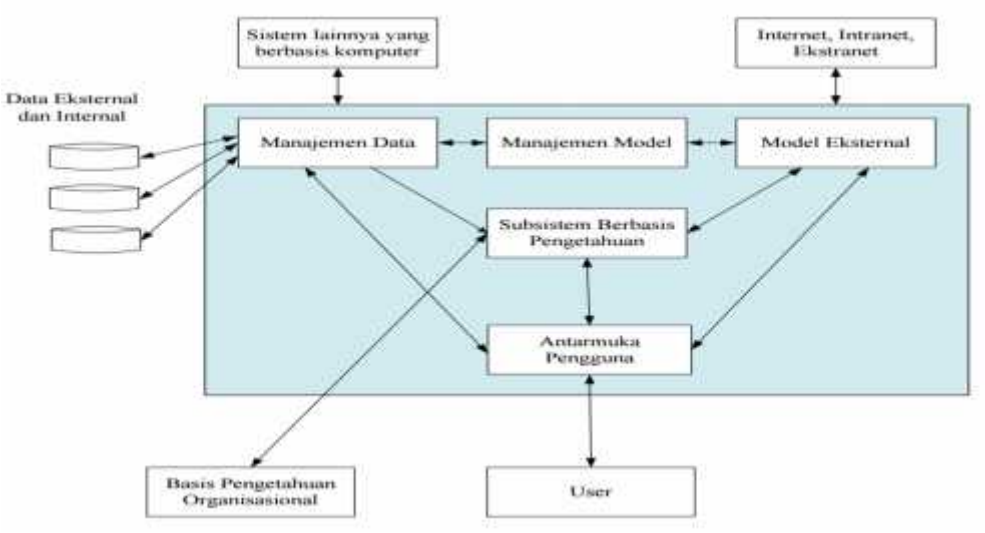
memiliki dasar kemampuan pengoperasian komputer yang tinggi. Oleh karena itu pendekatan yang digunakan biasanya model interaktif.

4. Sistem Pendukung Keputusan dirancang dengan menekankan pada aspek fleksibilitas serta kemampuan adaptasi yang tinggi. Sehingga mudah disesuaikan dengan beragai perubahan lingkungan yang terjadi pada kebutuhan pemakai. (Dwi Citra Hartini dkk, 2013:546)

II.3.2. Komponen-Komponen Sistem Pendukung Keputusan

Komponen-komponen dari Sistem Pendukung Keputusan adalah sebagai berikut :

1. Manajemen Data, mencakup *database* yang mengandung data yang relevan dan diatur oleh sistem yang disebut *Database Management System* (DBMS).
2. Manajemen Model, merupakan paket perangkat lunak yang memasukkan model-model finansial, statistik, ilmu manajemen, atau model kuantitatif yang lain yang menyediakan kemampuan analisis sistem dan management *software* yang terkait.
3. Antarmuka Pengguna, media interaksi antara sistem dengan pengguna, sehingga pengguna dapat berkomunikasi dan memberikan perintah pada SPK melalui subsistem ini.
4. Subsistem Berbasis Pengetahuan, subsistem yang dapat mendukung subsistem lain atau bertindak sebagai komponen yang berdiri sendiri. (Nana Yulia Fitri, Nurhadi, 2017 : 321)



Gambar II.1. Komponen Sistem Pendukung Keputusan

(Sumber : Ma'ruf, 2016 : 291)

II.3.3. Keuntungan Sistem Pendukung Keputusan

Sistem pendukung keputusan dapat memberikan berbagai manfaat atau keuntungan bagi pemakainya, antara lain :

1. Memperluas kemampuan pengambilan keputusan dalam memproses data/informasi bagi pemakainya.
2. Membantu pengambilan keputusan dalam hal penghematan waktu yang dibutuhkan untuk memecahkan masalah terutama berbagai masalah yang sangat kompleks dan tidak terstruktur.
3. Dapat menghasilkan solusi dengan lebih cepat serta hasilnya dapat diandalkan.
4. Walaupun suatu Sistem Pendukung Keputusan, mungkin saja tidak mampu memecahkan masalah yang dihadapi oleh pengambil keputusan, namun dapat

menjadi stimulan bagi pengambil keputusan dalam memahami persoalannya, karena sistem pendukung keputusan mampu menyajikan berbagai alternatif.

5. Dapat menyediakan bukti tambahan untuk memberikan bukti tambahan untuk memberikan pembenaran sehingga posisi pengambil keputusan. (Nana Yulia Fitri, Nurhadi, 2017 : 321)

II.4. Donor Darah

Menurut Dorland dalam jurnal Nur Yuli Dewi Hapsari (2012) Donor adalah penderma darah. Donor adalah organisme yang memberikan jaringan hidup untuk dapat digunakan pada tubuh yang lain, seperti orang yang memberikan darahnya untuk transfusi, atau organ untuk ditransplantasikan

Keberadaan donor sangat penting karena donor merupakan satu-satunya sumber pasokan darah. Menurut Antonio Fernandez-Montoya dalam jurnal Nur Yuli Dewi Hapsari (2012) juga menyatakan donor sukarela dan tidak dibayar tetap menjadi sumber donasi terbaik. Donor sukarela tak dibayar adalah landasan persediaan darah yang aman dan tetap terpelihara. Tanpa sebuah sistem yang berdasarkan donor sukarela tanpa bayaran, khususnya donor sukarela reguler, tak satupun negara yang dapat menyediakan cukup darah untuk semua pasien yang membutuhkan transfusi. Selain itu, donor sukarela dapat dipandang sebagai aset nasional yang berharga. Donor reguler sukarela juga dapat memainkan peran penting dalam mengidentifikasi faktor-faktor yang memotivasi donor reguler dan dalam merancang strategi dan materi-materi untuk memelihara donor. Selain itu, donor sukarela cocok sebagai pendidik donor, perekrut dan promoter kesehatan

yang efektif. Namun demikian, terdapat teori yang menyatakan bahwa donor sebenarnya memiliki makna *self-esteem* yang lebih rendah dan mereka berdonor dalam upaya untuk meningkatkan konsep dirinya.

Sementara itu, donor darah adalah menyumbangkan darah untuk menolong orang lain yang memerlukan darah. Donor darah merupakan suatu prosedur yang sangat aman dan sangat penting untuk transfusi darah. Transfusi darah merupakan suatu komponen esensial bagi pelayanan kesehatan. Transfusi darah berkontribusi menyelamatkan jutaan nyawa setiap tahun dalam situasi normal maupun darurat, mengijinkan intervensi medis kompleks dan operasi yang kian bertambah serta peningkatan harapan hidup dan kualitas hidup pasien-pasien dengan berbagai kondisi akut dan kronis.

II.5. Metode *Simple Additive Weighting* (SAW)

Metode *Simple Additive Weighting* sering juga di kenal dengan istilah metode penjumlahan berbobot. Konsep dasar metode *Simple Additive Weighting* adalah mencari penjumlahan terbobot dari rating kinerja pada setiap alternatif pada semua atribut. Metode *Simple Additive Weighting* disarankan untuk menyelesaikan masalah penyeleksian dalam sistem pengambilan keputusan multi proses. Metode *Simple Additive Weighting* merupakan metode yang banyak digunakan dalam pengambilan keputusan yang memiliki banyak atribut. Metode *Simple Additive Weighting* membutuhkan proses normalisasi matriks keputusan (x) ke suatu skala yang dapat diperbandingkan dengan semua rating alternatif yang ada.

Formula untuk melakukan normalisasi tersebut adalah sebagai berikut :

$$r_{ij} = \begin{cases} \dfrac{X}{Max\ X} & benefit \\ & \text{..... (1)} \\ \dfrac{Min\ X}{X} & cost \end{cases}$$

Keterangan:

Max x_{ij} = Nilai terbesar dari setiap kriteria *i.*

Min x_{ij} = Nilai terkecil dari setiap kriteria *i.*

x_{ij} = Nilai atribut yang dimiliki dari setiap kriteria.

Benefit = Jika nilai terbesar adalah yang terbaik.

Cost = Jika nilai terkecil adalah yang terbaik.

Dimana r_{ij} adalah rating kinerja ternormalisasi dari alternatif A_i pada atribut C_{ij} i=1,2,…,n. Nilai preferensi untuk setiap alternatif (V_i) di berikan sebagai:

$$Vi = \sum_{j=1}^{n} wj\ r\ ij \quad \text{..... (2)}$$

Keterangan :

V_i = Rangking untuk setiap alternatif.

W_j = Nilai bobot rangking (dari setiap kriteria).

r_{ij} = Nilai rating kinerja ternormalisasi.

Nilai V_i yang lebih besar mengidentifikasikan bahwa alternatif A_i lebih terpilih.

Langkah-langkah pemecahan masalah dalam penelitian ini adalah sebagai berikut :

1. Menentukan kriteria-kriteria yang akan dijadikan acuan dalam menentukan pengambilan keputusan C_j.
2. Memberikan nilai setiap alternatif (A_i) pada setiap kriteria (C_j) yang sudah ditentukan, dimana nilai *i*=1,2,……n.
3. Menentukan rating kecocokan setiap alternatif pada setiap kriteria kemudian memodelkannya ke dalam bilangan *fuzzy* setelah itu dikonversikan kebilangan *crisp*.
4. Memberikan nilai bobot (W) yang juga didapatkan berdasarkan nilai *crisp.*
5. Melakukan normalisasi matriks dengan cara menghitung nilai rating kinerja ternormalisasi (r_{ij}) dari alternatif A_i pada atribut C_j berdasarkan persamaan yang disesuaikan dengan jenis atribut (atribut keuntungan/*benefit* = *MAXIMUM* atau atribut biaya/*cost* = *MINIMUM)*. Apabila berupa atribut keuntungan maka *crisp (*$x_{i\,j}$) dari setiap kolom atribut dibagi dengan nilai *crisp MAX (MAX* $x_{i\,j}$) dari setiap kolom, sedangkan untuk atribut biaya, nilai *crisp MIN (MIN* $x_{i\,j}$) dari setiap kolom atribut dibagi dengan nilai *crisp (*$x_{i\,j}$) setiap kolom.
6. Melakukan proses perangkingan untuk setiap alternatif (V_i) dengan cara mengalikan nilai (W_i) dengan nilai rating kinerja ternormalisasi (r_{ij}).
7. Menentukan nilai prefensi untuk setiap alternatif (V_i) dengan cara menjumlahkan hasil kali antara matriks ternormalisasi (*R*) dengan nilai bobot (*W*). Nilai V_i yang lebih besar mengindikasikan bahwa alternatif A_i lebih terpilih. (Harold Situmorang, 2015 : 25-26)

II.6. Normalisasi

Normalisasi diartikan sebagai suatu teknik yang menstrukturkan mendekompsisi data dalam cara-cara tertentu untuk mencegah timbulnya permasalahan pengolahan data dalam basis data. Permasalahan yang dimaksud adalah berkaitan dengan penyimpangan-penyimpangan (*anomallies)* yang terjadi akibat adanya kerangkapan data dalam relasi dan in-efisiensi pengolahan (Martin, 1975) : (Edy Sutanta, 2011 : 174-175).

Proses normalisasi menghasilkan relasi yang optimal, yaitu

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.

   a. Jika memenuhi kriteria *Boyce-Codd Norm Form*;
   b. 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)

   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.7. Basis Data

Pada dasarnya basis data bukanlah sistem yang selalu terkait dengan komputer. Adapun beberapa penjelasan terkait dengan basis data adalah pengertian data, operasi dasar basis data, dan pengertian sistem informasi itu sensistem menegemen basis data.

Basis data terdiri dari 2 kata, yaitu basis dan data, basis dapat di artikan sebagai maskas atau gudang tempat bersarang atau berkumpul. Sedangkan data adalah representasi fakta dunia nyata yang mewakili suatu objek seperti manusia (pegawai, siswa, pembeli dan lain-lain), barang hewan, peristiwa, konsep keadaan dan sebagainya yang direkam dalam bentuk angka, huruf, simbol, teks, gambar, bunyi, atau kombinasinya. (Syaifudin Ramadhani, 2014:2)

II.8. SQL Server

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 Server* untuk berkerja lebih optimal. Settingan juga bisa dilakukan menggunakan *script* untuk memudahkan *developer* mengubah *Setting Options* pada *SQL Server* 2008. (Ruslan, 2013 : 39)

II.9. Visual Studio 2010

Visual Basic 2010 adalah inkarnasi dari bahasa *visual basic* yang sangat populer dan telah dilengkapi dengan fitur serta fungsi yang setara dengan bahasa tingkat lainnya seperti C++. Anda dapat menggunakan *visual basic* 2010 untuk membuat aplikasi *windows, mobile, web,* dan *office* atau kode yang telah ditulis oleh orang lain dan kemudian dimasukkan ke program lainnya. *Visual basic* menyediakan berbagai *tools* dan *fitur* canggih yang memungkinkan dapat menulis kode, menguji dan menjalankan program tunggal ataau terkadang serangkaian program yang terkait dengan satu aplikasi. (Christopher Lee, 2014:1)

II.10. *Unified Modeling Language* (UML)

UML merupakan bahasa visual dalam pemodelan yang memungkinkan pengembang sistem membuat sebuah *blueprint* yang dapat menggambarkan visi mereka tentang sebuah sistem dalam format yang standar, mudah dimengerti, dan menyediakan mekanisme untuk mudah dikomunikasikan dengan pihak lain (Yosua P. W Simaremare, et al., 2013 : 471).

UML adalah bahasa untuk mengspesifikasi, menvisualisasi, membangun dan mendokumentasikan *artefact* (bagian dari informasi yang digunakan atau dihasilkan oleh proses pembuatan perangkat lunak, seperti pada pemodelan bisnis dan sistem non perangkat lunak lainnya). UML juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan peranti lunak dalam bahasa-bahasa berorientasi objek seperti C++, Java, C#, atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C (I Made Budi Adnyana, 2016 : 51-52).

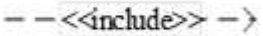
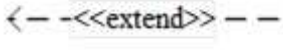
II.10.1. *Use Case* Diagram

Use Case Diagram adalah gambaran fungsionalitas dari suatu sistem, sehingga *customer* atau pengguna sistem paham dan mengerti mengenai kegunaan sistem yang akan dibangun (I Made Budi Adnyana, 2016 : 51).

Use case diagram merupakan pemodelan untuk kelakuakn (*behavior*) sistem yang akan dibuat. *Use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut (Ade Hendini, 2016 : 108).

Simbol-simbol yang digunakan dalam *use case* diagram dapat dilihat pada tabel II.1.

Tabel II.1. Simbol *Use Case* Diagram

Gambar	Keterangan
	*Use Case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktor, yang dinyatakan dengan menggunakan kata kerja.
	*Actor* atau 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 kontrol 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 mengindikasikan data.
	Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengindikasikan 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 : Ade Hendini, 2016 : 108-109)

II.10.2. *Class* Diagram

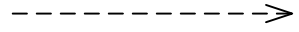
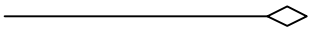
Class diagram menggambarkan struktur dan deskripsi *class*, *package* dan *object* beserta hubungan satu sama lain seperti pewarisan, asosiasi, dan lain-lain (I Made Budi Adnyana, 2016 : 52).

Class diagram menggambarkan struktur sistem dari segi pendifinisian kelas-kelas yang akan di buat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi (Winda Aprianti dan Umi Maliha, 2016 : 22).

Simbol-simbol yang digunakan pada *class* diagram dapat dilihat pada tabel II.2 sebagai berikut.

Tabel II.2. Simbol *Class* Diagram

Gambar	Keterangan
	Kelas. Kelas pada struktur sistem
	Antarmuka/*Interface.* Sama dengan konsep *interface* dalam pemrograman berorientasi objek
	Asosiasi/*Association* Relasi antar kelas dengan makna umum, asosiasi biasanya disertai dengan *multiplicity*
	Asosiasi berarah/*Directed association.* Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya disertai dengan *multiplicity*
	Generalisasi. Relasi antar kelas dengan makna generalisasi-spesialisasi (umum-khusus)

	Kebergantungan/*Dependency*. Relasi antar kelas dengan makna kebergantungan antar kelas
	Agregasi/*Aggregation.* Relasi antar kelas dengan makna semua bagian.

(Sumber : Winda Aprianti dan Umi Maliha, 2016 : 22)

Hubungan antar kelas mempunyai keterangan yang disebut *multiplicity* atau *cardinality* yang dapat dilihat pada tabel II.3.

Tabel II.3. *Multiplicity Class* Diagram

Multiplicity	**Keterangan**
1	Satu dean 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 maksimal 4

(Sumber : Ade Hendini, 2016 : 111)

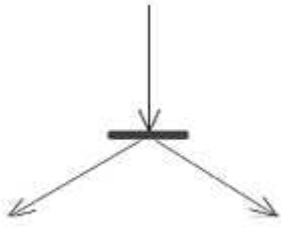
II.10.3. *Activity* Diagram

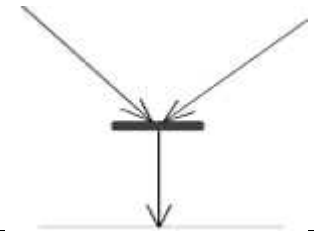
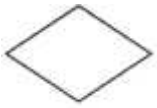
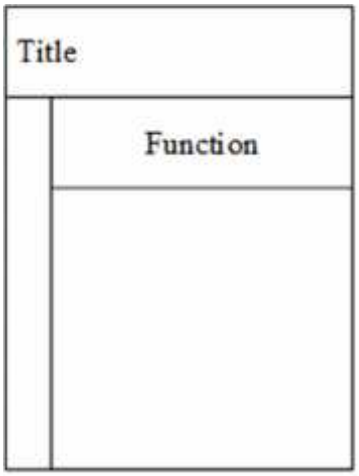
Activity diagram menggambarkan sebagai alur aktifitas dalam sistem yang sedang dirancang. Bagaimana masing-masing alur berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. Sebuah aktifitas dapat direalisasikan oleh satu *use case* atau lebih. Aktifitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana *actor* menggunakan sistem untuk melakukan aktivitas. *Decision* digunakan untuk menggambarkan

behaviour pada kondisi teretentu. Untuk mengilustrasikan proses-proses *parallel* (*fork* dan join) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertical. *Activity* diagram dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktifitas tertentu (I Made Budi Adnyana, 2016 : 52).

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

Tabel II.4. Simbol *Activity* Diagram

Gambar	Keterangan
	*Start Point*, diletakkan pada pojok kiri atas dan merupakan awal aktivitas
	*End Point*, akhir aktivitas
	*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*
	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa

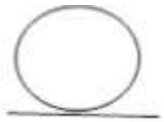
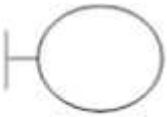
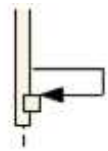
(Sumber : Ade Hendini, 2016 : 109-110)

II.10.4. *Sequence* Diagram

Sequence Diagram menggambarkan interaksi atar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence* Diagram terdiri atas dimensi vertical (waktu) dan dimensi horizontal (obyek– obyek yang terkait). *Sequence* Diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu (I Made Budi Adnyana, 2016 : 52).

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

Tabel II.5. Simbol *Sequence* Diagram

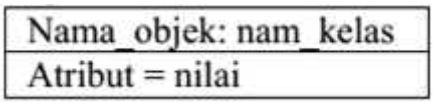
Gambar	Keterangan
	*Entity Class*, merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data
	*Boundary Class*, berisi kumpulan kelas yang menjadi *interfaces* atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan *form entry* 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*, mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivasi sebuah operasi
	*Lifeline*, garis titik-titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation*

(Sumber : Ade Hendini, 2016 : 110)

II.10.5. *Object* Diagram

Object diagram menggambarkan struktur sistem dari segi penamaan objek dan jalannya objek dalam sistem. *Object* diagram memastikan bahwa semua kelas yang sudah didefinisikan pada *Class* diagram harus dipakai objeknya, karena jika tidak, pendefinisian kelas itu tidak dapat dipertanggung jawabkan. Simbol-simbol *object* diagram ditunjukkan pada tabel II.6 (Winda Aprianti dan Umi Maliha, 2016 : 22).

Tabel II.6. Simbol *Object* Diagram

Gambar	Keterangan
	Objek. Objek dari kelas yang berjalan saat sistem dijalankan
	Link. Relasi antar objek

(Sumber : Winda Aprianti dan Umi Maliha, 2016 : 22)

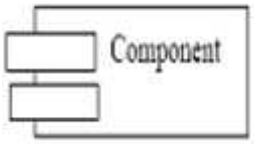
II.10.6. *Deployment* Diagram

Deployment diagram menampilkan rancangan fisik jaringan dimana berbagai komponen akan terdapat di sana. *Deployment* diagram juga dapat menunjukkan perangkat-perangkat *nodes* diantara hubungan yang dimilikinya antar komponen dan menunjukkan tata letak sebuah sistem secara fisik,

menampakkan bagian-bagian *software* yang berjalan pada bagian-bagian *hardware* (Marsha Sevin Aldilla, et al., 2015 : 6).

Simbol-simbol yang digunakan dalam *deployment* diagram dapat dilihat pada tabel II.7.

Tabel II.7. Simbol *Deployment* Diagram

Gambar	Keterangan
	Pada *deployment* diagram, komponen-komponen yang ada diletakkan didalam *node* untuk memastikan keberadaan posisi mereka
	*Node* menggambarkan bagian-bagian *hardware* dalam sebuah sistem. Notasi untuk *node* digambarkan sebagai sebuah kubus 3 dimensi
	Sebuah *association* digambarkan sebagai sebuah garis yang menghubungkan dua *node* yang mengindikasikan jalur komunikasi antara elemen-elemen *hardware*

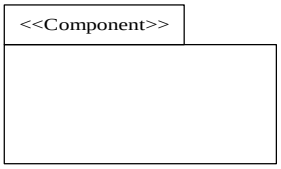
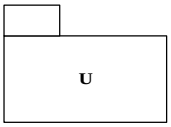
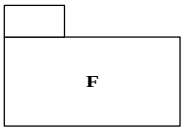
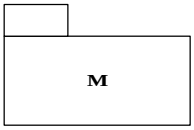
(Sumber : Ade Hendini, 2016 : 111)

II.10.7. *Component* Diagram

Component diagram menunjukkan model secara fisik komponen perangkat lunak pada sistem dan hubungannya antar mereka. Komponen pada piranti lunak adalah berupa modul-modul yang berisikan kode. Umumnya komponen yang terbentuk dari beberapa kelas atau juga terbentuk dari komponen komponen yang lebih kecil (Marsha Sevin Aldilla, et al., 2015 : 6).

Simbol-simbol yang digunakan dalam *component* diagram dapat dilihat pada tabel II.8.

Tabel II.8. Simbol *Component* Diagram

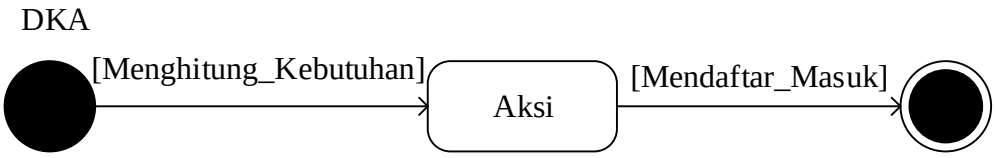
Gambar	Keterangan
	*Peckage* merupakan tempat komponen-komponen di mana komponen tersebut untuk memodelkan sesuatu
	Komponen *user interface* untuk mengatur interaksi antara aktor dan fungsi
	Komponen *function* : memberikan fungsi untuk model
	Komponen model/*database* : digunakan untuk menyimpan objek-objek yang tergambar dalam problem domain
	Dependency merupakan penghubung antara komponen di antara *client-server*

(Sumber : Indrajani, 2015 : 48)

II.10.8. *Statechart* Diagram

Statechart diagram digunakan untuk membuat model bagaimana suatu objek mengalami perubahan *state*, menggambarkan *behaviour* dari sub-sistem, membuat model interaksi antara *class-class* dan model dari tampilan sistem. *Statechart* diagram banyak digunakan pada saat peralihan antara analisis dan fase desain. Umumnya digunakan untuk menggambarkan sistem interaktif yang *"real tima"*. Diagram ini bersifat optional dalam suatu perancangan sistem. Kumpulan

sub-*state* yang dikelompokkan ke dalam sebuah *State* disebut sebagai *composite state* (Indrajani, 2015 : 46).



Gambar II.2. *Statechart* Diagram

(Sumber : Indrajani, 2015 : 47)