BAB II

TINJAUAN PUSTAKA

II.1. Sistem

Menurut Asbon hendra (2012 : 157) Sistem merupakan kumpulan dari unsure atau elemen-elemen yang saling berkaitan/berinteraksi dan saling memengaruhi dalam melakukan kegiatan bersama untuk mencapai suatu tujuan tertentu.

II.1.1. Karakteristik Sistem

Ada beberapa karakteristik sistem adalah sebagai berikut :

1. Komponen (*Component*)

   Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, bekerja sama membentuk satu kesatuan. Komponen-komponen sistem dapat berupa suatu subsistem atau bagian-bagian dari sistem.

2. Batas Sistem (*Boundary*)

   Batas sistem merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lainnya atau dengan lingkungan lainnya.

3. Lingkungan Luar Sistem (*Environment*)

   Environment merupakan segala sesuatu di luar batas sistem yang mempengaruhi operasi dari suatu sistem.

4. Penghubung Sistem (*Interface*)

   Merupakan media penghubung antara satu subsistem dengan subsistem yang lainnya untuk membentuk satu kesatuan sebagai sumber-sumber daya mengalir dari subsistem yang satu ke subsistem yang lainnya.

5. Masukan Sistem (*Input*)

   Merupakan energi yang dimasukkan ke dalam sistem. Masukan dapat berupa masukan perawatan (*Maintenance Input*) adalah energi yang dimasukkan supaya sistem tersebut dapat beroperasi.

6. Keluaran Sistem (*Output*)

   Merupakan dari hasil energi yang diolah oleh sistem meliputi *output* yang berguna.

7. Pengolah Sistem (*Process*)

   Merupakan bagian yang memproses masukan untuk menjadi keluaran yang di inginkan.

8. Tujuan Sistem (*Goal*)

   Suatu sistem dikatakan berhasil kalau pengoperasian sistem itu mengenai sasaran atau tujuannya. (Asbon Hendra 2012 : 158-160)

II.1.2. Infomasi

Menurut Asbon hendra (2012 : 167) Informasi merupakan data yang telah diproses menjadi bentuk yang memiliki arti bagi penerima dan dapat berupa fakta, suatu nilai yang bermanfaat.

II.1.3. Sistem Informasi

Sistem infomasi adalah sekumpulan prosedur manual atau terkomputerisasi yang mengumpulkan/mengambil, mengolah, menyimpan, dan menyebarkan informasi dalam mendukung pengambilan dan kendali keputusan. (Asbon hendra : 2012 : 168-169)

II.2. Sistem Pendukung Keputusan (SPK)

Sistem pendukung keputusan menurut para ahli mengatakan bahwa :

Menurut Asfi dan Sari (2010:132) Sistem Penunjang Keputusan (SPK) adalah bagian dari sistem informasi berbasis komputer termasuk sistem berbasis pengetahuan untuk mendukung pengambilan keputusan dalam suatu organisasi atau perusahaan. SPK juga dapat merupakan sistem komputer yang mengolah data menjadi informasi untuk mengambil keputusan dari masalah semi-terstruktur yang spesifik. SPK dapat menjadi alat bantu bagi para pengambil keputusan untuk memperluas kapabilitas mereka, namun tidak untuk menggantikan penilaian mereka. SPK ditujukan untuk keputusan-keputusan yang memerlukan penilaian atau pada keputusan-keputusan yang sama sekali tidak dapat didukung oleh algoritma.

Menurut Jahanshahloo dalam Masruro (2013:07-39),“sistem pendukung keputusan adalah sebuah sistem yang dimaksudkan untuk mendukung para pengambil keputusan manajerial dalam situasi keputusan semi terstruktur”.

Menurut Sarwindah (2013),“sistem pendukung keputusan merupakan pasangan intelektual dari sumber daya manusia dengan kemampuan komputer

untuk memperbaiki keputusan, yaitu sistem pendukung keputusan berbasis komputer bagi pembuat keputusan manajemen yang menghadapi masalah semi terstruktur ”.

Dari ketiga definisi di atas, maka dapat disimpulkan sistem pendukung keputusan adalah sistem yang membantu para pengambil keputusan manajerial dalam situasi semi terstruktur yaitu dengan kegiatan intelijen, kegiatan merancang, menemukan, mengembangkan dan menganalisis berbagai alternatif tindakan yang mungkin dan tidak terstruktur yaitu kegiatan menelaah menilai pilihan-pilihan yang lalu dengan berbasis komputer interaktif.

II.2.1. Karakteristik Sistem Pendukung Keputusan

Beberapa karakteristik dari Sistem Pendukung Keputusan menurut (Turban : 2010) adalah sebagai berikut :

1. Sistem Pendukung Keputusan dirancang untuk membantu pengambil keputusan dalam memecahkan masalah yang sifatnya semi terstruktur ataupun tidak terstruktur.
2. Dalam proses pengolahannya, sistem pendukung keputusan mengkombinasikan penggunaan model-model / teknik-teknik analisis dengan teknik pemasukan data konvensional serta fungsi-fungsi pencari / interogasi informasi.
3. Sistem Pendukung Keputusan, dirancang sedemikian rupa sehingga dapat digunakan / dioperasikan dengan mudah oleh orang-orang yang tidak memiliki dasar kemampuan 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 berbagai perubahan lingkungan yang terjadi dan kebutuhan pemakai.

Dengan berbagai karakter khusus seperti yang dikemukakan di atas, sistem pendukung keputusan dapat memberikan berbagai manfaat atau keuntungan bagi pemakainya. Keuntungan yang dimaksud di antaranya meliputi :

1. Sistem Pendukung Keputusan memperluas kemampuan pengambil keputusan dalam memproses data / informasi bagi pemakainya.
2. Sistem Pendukung Keputusan membantu pengambil keputusan dalam hal penghematan waktu yang dibutuhkan untuk memecahkan masalah terutama berbagai masalah yang sangat kompleks dan tidak terstruktur.
3. Sistem Pendukung Keputusan 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 disajikan kesimpulan bagi pengambil keputusan dalam memahami persoalannya. Karena sistem ini mampu menyajikan berbagai alternatif.
5. Sistem Pendukung Keputusan dapat menyediakan bukti tambahan untuk memberikan pembenaran sehingga dapat memperkuat posisi pengambil keputusan. (Nana dan Nurhadi ; 2017)

Di samping berbagai keuntungan dan manfaat yang dikemukakan di atas,

Sistem Pendukung Keputusan juga memiliki keterbatasan menurut (Katen dan Novriyeni ; 2013) adalah sebagai berikut :

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 sistem pendukung keputusan terbatas pada pengetahuan dasar serta model dasar yang dimilikinya
3. Proses yang dapat dilakukan oleh sistem pendukung keputusan biasanya tergantung juga pada kemampuan perangkat lunak yang digunakannya.

II.2.2. Komponen-komponen Sistem Pendukung Keputusan

Menurut Heny Pratiwi (2014:96-97) Komponen-komponen dalam SPK meliputi 8 (delapan) bagian, yaitu :

1. Perangkat Keras
2. Perangkat Lunak
3. Sumber Data
4. Sumber Model
5. Sumber Daya Manusia
6. Model Sistem Pendukung Keputusan
7. Lembar Kerja Elektronik
8. Sistem Pendukung Keputusan Kelompok

II.2.3. Tujuan Sistem Pendukung Keputusan

Menurut Heny Pratiwi (2014:96), Tujuan dari sistem pendukung keputusan adalah :

1. Membantu manajer dalam pengambilan keputusan atas masalah semi-terstruktur.
2. Memberikan dukungan atas pertimbangan manajer dan bukannya dimaksudkan untuk menggantikan fungsi manajer.
3. Meningkatkan efektivitas keputusan yang diambil manajer lebih daripada perbaikan efisiensinya.
4. Kecepatan komputasi. Komputer memungkinkan para pengambil keputusan untuk melakukan banyak komputasi secara cepat dengan biaya yang rendah.
5. Peningkatan produktivitas. Membangun satu kelompok pengambil keputusan, terutama para pakar, bisa sangat mahal. Pendukung terkomputerisasi bisa mengurangi ukuran kelompok dan memungkinkan para anggotanya untuk berada di berbagai lokasi yang berbeda beda (menghemat biaya perjalanan). Selain itu, produktivitas staf pendukung (misalnya analisis keuangan dan hukum) bisa ditingkatkan. Produktivitas juga bisa ditingkatkan menggunakan peralatan optimalisasi yang menentukan cara terbaik untuk menjalankan sebuah bisnis.

II.3. Metode Topsis

TOPSIS (*Technique for Order Performance of Similarity to Ideal Solution)* adalah metode multi kriteria yang digunakan untuk mengidentifikasi

solusi dari himpunan alternatif berdasarkan minimalisasi simultan dari jarak titik ideal dan memaksimalkan jarak dari titik terendah. TOPSIS dapat menggabungkan bobot relatif dari kriteria penting. Metode TOPSIS merupakan metode penilaian yang ditafsirkan dapat memberikan setiap objek untuk dievaluasi nilainya secara spesifik. Metode TOPSIS pertama kali disampaikan oleh Hwang dan Yoon, merupakan metode beberapa kriteria sederhana dan efisien untuk mengidentifikasi solusi dari himpunan beberapa alternatif. (Ahmad Abdul Chamid; 2016)

TOPSIS mempunyai prinsip bahwa alternatif yang terpilih harus mempunyai jarak terdekat dari solusi ideal positif dan mempunyai jarak terjauh dari solusi ideal negatif dari sudut pandang geometris dengan menggunkana jarak *Euclidean* (jarak antara dua titik) untuk menentukan kedekatan relatif dari suatu alternatif.

Metode TOPSIS memiliki keuntungan sebagai berikut:

1. Metode Topsis merupakan salah satu metode yang *simple* dan konsep rasional yang mudah dipahami.
2. Metode Topsis mampu untuk mengukur kinerja relatif dalam membentuk form matematika sederhana.

Adapun tahapan-tahapan metode Topsis sebagai berikut:

1. Membuat matriks keputusan yang ternormalisasi.
2. Membuat matriks keputusan yang ternormalisasi terbobot.
3. Menentukan matriks solusi ideal positif dan matriks solusi ideal negatif.
4. Menentukan jarak antara nilai setiap alternatif dengan matriks solusi ideal

positif dan negatif.

5. Menentukan nilai preferensi untuk setiap alternatif .

TOPSIS didasarkan pada konsep dimana alternatif terpilih yang terbaik tidak hanya memiliki jarak terpendek dari solusi ideal positif, namun juga memiliki jarak terpanjang dari solusi ideal negatif. Konsep ini banyak digunakan pada beberapa model MADM untuk menyelesaikan masalah keputusan secara praktis. Hal ini disebabkan konsepnya sederhana dan mudah dipahami, komputasinya efisien, dan memiliki kemampuan untuk mengukur kinerja relatif dari alternatif-alternatif keputusan dalam bentuk matematis yang sederhana. Secara umum, prosedur TOPSIS mengikuti langkah-langkah sebagai berikut :

a. Membuat matriks keputusan yang ternormalisasi;

b. Membuat matriks keputusan yang ternormalisasi terbobot;

c. Menentukan matriks solusi ideal positif dan matriks solusi ideal negatif;

d. Menentukan jarak antara nilai setiap alternatif dengan matriks solusi ideal positif dan matriks solusi ideal negatif;

e. Menentukan nilai preferensi untuk setiap alternatif.

TOPSIS membutuhkan rating kerja setiap alternatif A_i pada setiap kriteria C_j yang ternormalisasi

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{m} x_{ij}^{\,2}}} \; ; \qquad \text{.....(1)}$$

dengan i=1,2,...,m; dan j=1,2,...,n

dimana :

r_{ij} = matriks ternormalisasi [i][j]

x_{ij} = matriks keputusan [i][j]

Solusi ideal positif A^+ dan solusi ideal negatif A^- dapat ditentukan berdasarkan rating bobot ternormalisasi (y_{ij}) sebagai :

y_{ij} = wi.rij ; dengan i=1,2,...,m; dan j=1,2,...,n

$A^+ = (y_1^+, y_2^+, ..., y_n^+)$;

$A^- = (y_1^-, y_2^-, ..., y_n^-)$;

dimana :

y_{ij} = matriks ternormalisasi terbobot [i][j]

w_i = vektor bobot[i] dari proses AHP

y_j^+ = max y_{ij}, jika j adalah atribut keuntungan

min y_{ij}, jika j adalah atribut biaya

y_j^- = min y_{ij}, jika j adalah atribut keuntungan

max y_{ij}, jika j adalah atribut biaya

j = 1,2,...,n

Jarak antara alternatif A_i dengan solusi ideal positif :

$$D_i^+ = \sqrt{\sum_{i=1}^{n} (y_i^+ - y_{ij})^2} \quad ; \quad i=1,2,...,m \qquad .....(2)$$

dimana :

D_i^+ = jarak alternatif Ai dengan solusi ideal positif

y_i^+ = solusi ideal positif[i]

y_{ij} = matriks normalisasi terbobot[i][j]

Jarak antara alternatif A_i dengan solusi ideal negatif :

$$D_i^- = \sqrt{\sum_{j=1}^{n} (y_{ij} - y_i^-)^2} \quad ; \quad i=1,2,...,m \qquad \text{.....(3)}$$

dimana :

D_i^- = jarak alternatif Ai dengan solusi ideal negatif

y_i^- = solusi ideal positif[i]

y_{ij} = matriks normalisasi terbobot[i][j]

Nilai preferensi untuk setiap alternatif (V_i) dapat dilihat pada rumus (4).

$$Vi = \frac{D_i^-}{D_i^- + D_i^+} \quad ; \quad i=1,2,...,m \qquad \text{.....(4)}$$

dimana :

V_i = kedekatan tiap alternatif terhadap solusi ideal

D_i^+ = jarak alternatif Ai dengan solusi ideal positif

D_i^- = jarak alternatif Ai dengan solusi ideal negatif

Nilai V_i yang lebih besar menunjukkan bahwa alternatif A_i lebih dipilih. (Agus Perdana Windarto;2017)

II.4. Visual Basic .Net

Visual basic 2010 adalah salah satu bagian dari microsoft visual studio 2010. Sebuat alat yang digunakan oleh pengembang windows dari berbagai level untuk mengembangkan dan membangun aplikasi yang bergerak diatas sistem

.NET Framework, dengan menggunakan bahasa BASIC. Visual Basic menyediakan cara cepat dan mudah untuk membuat aplikasi (Aswan 2012 : 1).

Pada perkembangan nantinya, mungkin untuk membuat program dengan teknologi *.NET*, memungkinkan para pengembang perangkat lunak akan dapat menggunakan lintas sistem operasi, yaitu dapat dikembangkan di sistem operasi *Windows* juga dapat dijalankan pada sistem operasi lain, misalkan pada sistem operasi *Linux*, seperti yang telah dilakukan pada pemograman *Java* oleh *Sun Microsystem*. Pada saat ini perusahaan-perusahaan sudah banyak meng-*update* aplikasi yang lama yang dibuat dengan *Microsoft Visual Basic 6.0* ke teknologi *.NET* karena kelebihan-kelebihan yang ditawarkan. "*Visual basic* adalah bahasa pemrograman komputer" bahasa pemrograman adalah perintah-perintah yang dimengerti oleh komputer untuk melakukan tugas-tugas tertentu, *visual basic* adalah salah satu *development tool*, yaitu alat untuk membantu berbagai macam program komputer yang khususnya mengunakan sistem operasi *windows*. (Agus Tinus Setiawan, Rika Putri Permadani : 2016).

II. 5. *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. *Microsoft* merilis *SQL Server 2008* dalam beberapa versi yang disesuaikan dengan segment-segment pasar yang dituju. Versi-versi tersebut adalah sebagai berikut. Menurut cara pemrosesan data pada prosesor maka *Microsoft* mengelompokkan produk ini berdasarkan 2 jenis yaitu :

a. Versi 32-bit(x86), yang biasanya digunakan untuk komputer dengan *single prosesor* (Pentium 4) atau lebih tepatnya *prosesor* 32 bit dan sistem operasi *Windows XP*.
b. Versi 64-bit(x64), yang biasanya digunakan untuk komputer dengan lebih dari satu *prosesor* (Misalnya *Core 2 Duo*) dan sistem operasi 64 bit seperti *Windows XP 64*, *Vista*, dan *Windows 7*. Sedangkan secara keseluruhan terdapat versi-versi seperti berikut ini:
c. *Versi Compact*, ini adalah versi "Tipis" dari semua versi yang ada. Versi ini seperti versi *desktop* pada *SQL Server 2000*. Versi ini juga digunakan pada *handled drvice* seperti *Pocket PC, PDA, SmartPhone, Tablet PC*.
d. *Versi Express*, ini adalah versi "Ringan" dari semua versi yang ada (tetapi versi ini berbeda dengan versi *compact*) dan paling cocok untuk latihan, *Express Manager* standar, integrasi dengan *CLR* dan *XML*.

Microsoft SQL Server merupakan salah satu *database relational* yang banyak digunakan oleh dunia usaha. (Agus Tinus Setiawan, Rika Putri Permadani: 2016).

II.6. Pemasok

Pemasok merupakan suatu perusahaan atau individu yang mampu untuk menyediakan sumber daya, baik dalam bentuk barang atau jasa yang dibutuhkan oleh perusahaan lainnya. Pemasok merupakan komponen penting dibidang logistik dan manajemen produksi. Untuk memperoleh pemasok yang mampu memenuhi barang atau jasa sesuai permintaan, diperlukan proses pemilihan pemasok yang baik. Tujuan dari pemilihan pemasok yaitu untuk memperoleh pemasok yang tepat sehingga dapat mengurangi biaya pembelian barang atau jasa. Pemilihan pemasok yang salah, dapat merugikan perusahaan. Untuk itu pemilihan pemasok merupakan komponen penting yang harus dilakukan dalam suatu perusahaan. (Suci Oktri Viarani; 2015)

II.7. UML *(Unified Modelling Language)*

Unified Modeling Language (UML) adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement*, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek. *UML* merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung.

UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak. *UML* hanya berfungsi untuk melakukan pemodelan. Jadi penggunaan *UML* tidak terbatas pada metodologi tertentu, meskipun pada

kenyataannya *UML* paling banyak digunakan pada metodologi berorientasi objek (Rosa A.S dan M. Shalahudin, 2015:133).

II.7.1. Sejarah *UML*

Bahasa pemrograman berorientasi objek yang pertama dikembangkan dikenal dengan nama Simula-67 yang dikembangkan pada tahun 1967. Perkembangan aktif dari pemrograman berorientasi objek mulai menggeliat ketika berkembangnya bahasa pemrograman Smalltalk pada awal 1980-an yang kemudian diikuti dengan perkembangan bahasa pemrograman berorientasi objek yang lainnya seperti C objek, C++, Eiffel, dan CLOS.

Sekitar lima tahun setelah Smalltalk berkembang, maka berkembang pula metode pengembangan berorientasi objek. Karena banyaknya metodologi-metodologi yang berkembang pesat saat itu, maka muncullah ide untuk membuat sebuah bahasa yang dapat dimengerti semua orang. Maka dibuat bahasa yang merupakan gabungan dari beberapa konsep, seperti konsep *Object Modeling Technique (OMT)* dari Rumbaugh dan Booch (1991), konsep *The Classes, Responsibilities, Collaborators (CRC)* dari Rebecca Wirfs-Brock (1990), konsep pemikiran Ivar Jacobson, dan beberapa konsep lainnya dimana James R. Rumbaigh, Grady Booch, dan Ivar Jacobson bergabung dalam sebuah perusahaan yang bernama *Rational Software Corporation* menghasilkan bahasa yang disebut dengan *Unified Modeling Language (UML).*

Pada tahun 1996, *Object Management Group (OMG)* mengajukan proposal agar adanya standarisasi pemodelan berorientasi objek dan pada bulan September 1997 *UML* diakomodasi oleh *OMG* sehingga sampai saat ini *UML*

telah memberikan kontribusinya yang cukup besar di dalam metodologi berorientasi objek dan hal-hal yang terkait di dalamnya (Rosa A.S dan M. Shalahudin, 2015:138).

II.7.2. Diagram *UML*

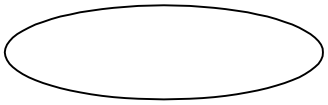
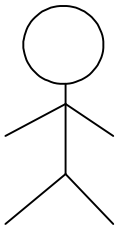
Rosa A.S dan M. Shalahudin (2015:140), pada *UML* terdiri dari 13 macam diagram yang dikelompokkan dalam 3 kategori. Berikut ini penjelasan singkat dari pembagian kategori tersebut.

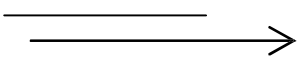
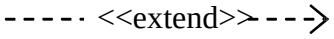
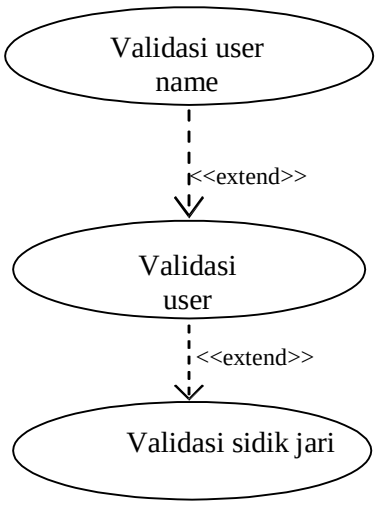
1. *Structure diagram*, yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan. *Structure diagram* terdiri dari *class diagram, object diagram, component diagram, composite structure diagram, package diagram dan deployment diagram.*
2. *Behavior diagram* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem. Behavior diagram terdiri dari *Use case diagram, Activity diagram, State Machine System.*
3. *Interaction diagram* yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antar subsistem pada suatu sistem. *Interaction diagram* terdiri dari *Sequence Diagram, Communication Diagram, Timing Diagram, Interaction Overview Diagram.*

II.7.3. *Use Case Diagram*

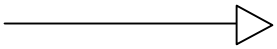
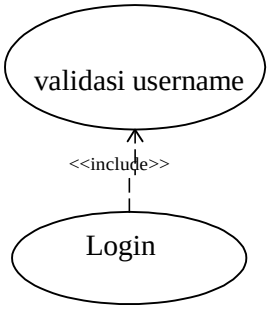
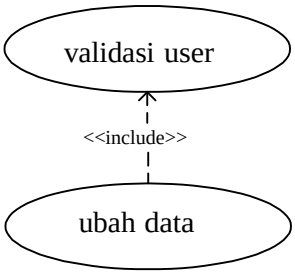
Rosa dan M. Shalahudin (2015:155), *use case* atau diagram *use case* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu. Berikut adalah simbol-simbol yang ada pada diagram *use case* yang dapat dilihat pada Tabel II.1 :

Tabel II.1. Simbol-simbol *Use Case Diagram*

No.	Simbol	Deskripsi
1.	*Use case*  Nama use case	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor, biasanya dinyatakan dengan menggunakan kata kerja diawal frase nama *use case*.
2.	Aktor/*actor* 	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang, biasanya dinyatakan menggunakan kata benda di awal frase nama aktor.

3.	Assosiasi/association 	Komunikasi antara aktor dan *usecase* yang berpartisilpasi pada *usecase* atau *usecase* memiliki interaksi dengan aktor.
4.	Exstensi/extend 	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu, mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek, biasanya *usecase* tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan, misal  Arah panah mengarah pada *use case* yang ditambahkan, biasanya *use case* yang menjadi *extend*-nya merupakan jenis yang sama dengan *use case* yang menjadi induknya.

6.	Menggunakan / *include* / *uses* <<include>>	Relasi *use case* tambahan ke sebuah *usecase* dimana *use case* yang ditambahkan memerlukan *usecase* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini Ada dua sudut pandang yang cukup besar mengenai *include* di *use case* : - *Include* berarti *use case* yang ditambahkan akan selalu di panggil saat *use case* tambahan dijalankan, misal pada kasus berikut :

	<<uses>> 	 *Include* berarti *use case* yang tambahan akan selalu melakukan pengecekan apakah *use case* yang ditambahkan telah dijalankan sebelum *use case* tambahan di jalankan, misal pada kasus berikut :  Kedua interpretasi di atas dapat dianut salah satu atau keduanya tergantung pada pertimbangan dan interpretasi yang dibutuhkan.

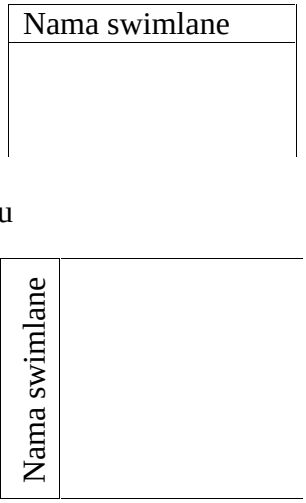
Sumber : Rosa A.S dan M. Shalahudin (2015:156)

4. *Activity Diagram*

Rosa dan M. Shalahudin (2015:161), diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem. Berikut adalah simbol-simbol yang ada pada diagram aktivitas yang dapat dilihat pada tabel II.2:

Tabel II.2. Simbol-simbol *Activity Diagram*

No.	Simbol	Deskripsi
1.	Status awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
2.	Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
3.	Percabangan/*decision* 	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
4.	Penggabungan/join 	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
5.	Status akhir 	Status akhir yang dilakukan oleh sistem, sebuah diagram aktivitas memiliki sebuah status akhir.

6.	Swimlane  	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

Sumber : Rosa A.S dan M. Shalahudin (2015:162)

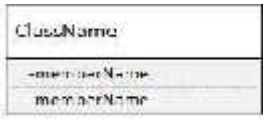
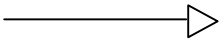
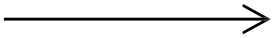
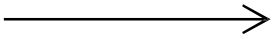
5. *Class Diagram*

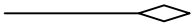
Rosa dan M. Shalahudin (2015:141), diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan *method* atau operasi. Berikut penjelasan atribut dan *method* :

1. Atribut merupakan variable-variabel yang dimiliki oleh suatu kelas.
2. Operasi atau *method* adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

Berikut adalah simbol-simbol yang ada pada diagram kelas yang dapat dilihat pada tabel II.3 :

Tabel II.3. Simbol-simbol *Class Diagram*

No.	Simbol	Deskripsi
1.	Kelas 	Kelas pada struktur sistem
2.	Antarmuka/*interface* 	Sama dengan konsep *interface* dalam pemrograman berorientasi objek
3.	Asosiasi/*association*	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*
4.	Asosiasi berarah/ *directed Association* 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*
5.	Generalisasi 	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum-khusus)
6.	Kebergantungan/*dependensi* 	Relasi antar kelas dengan makna kebergantungan antar kelas

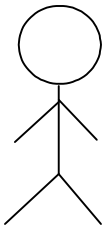
7.	Agregasi/*aggregation* 	Relasi antar kelas dengan makna semua-bagian (*whole-part*)

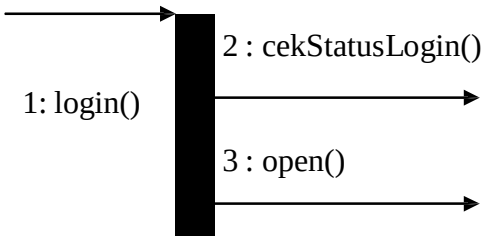
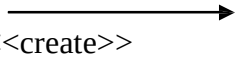
Sumber : Rosa A.S dan M. Shalahudin (2015:146)

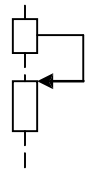
II.7.4. *Sequence Diagram*

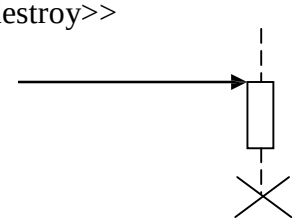
Rosa dan M. Shalahudin (2015:165), diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dengan *massage* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu. Membuat diagram sekuen juga dibutuhkan untuk melihat skenario yang ada pada *use case*. Banyaknya diagram sekuen yang harus digambar adalah minimal sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup dalam diagram sekuen sehingga semakin banyak *use case* yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak. Berikut adalah simbol-simbol yang ada pada diagram sekuen yang dapat dilihat pada tabel II.4 :

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

No.	Simbol	Deskripsi
1.	Aktor  Atau Nama aktor Tanpa waktu aktif	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang, biasanya dinyatakan dalam menggunakan kata benda diawal frase nama aktor.
2.	Garis hidup/*lifeline* 	Menyatakan kehidupan suatu objek
3.	Objek Nama objek : nama kelas	Menyatakan objek yang berinteraksi Pesan

4.	Waktu aktif 	Menyatakan objek dalam keadaan aktif dan berinteraksi, semuanya yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan di dalamnya, misalnya  Maka cek StatusLogin() dan open() dilakukan didalam metode login(). Aktor tidak memiliki waktu aktif
5.	Pesan tipe create  	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat.

6.	Pesan tipe call 1 : nama_metode() Arah panah mengarah pada objek yang memiliki operasi/metode, karena ini memanggil operasi/metode maka operasi/metode yang dipanggil harus ada pada diagram kelas sesuai dengan kelas objek yang berinteraksi.	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri, 1 : nama_metode() 
7.	Pesan tipe send 1 : masukkan	Menyatakan bahwa suatu objek mengirimkan data/masukkan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirim
8.	Pesan tipe return 1 : keluaran	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian

9.	Pesan tipe destroy <<destroy>> 	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaliknya jika ada *create* maka ada *destroy*

Sumber : Rosa A.S dan M. Shalahudin (2015:165)