BAB II

TINJAUAN PUSTAKA

II.1. Penelitian Terdahulu

Adapun penelitian yang pernah dilakukan oleh Riska Hanifa dalam judulnya “Implementasi Metode Promethee Dalam Penentuan Penerima Kredit Usaha Rakyat (KUR)” yang mana pada penelitian tersebut penulis menjelaskan tentang penerapan metode PROMETHEE terhadap penerima kredit usaha rakyat yang layak untuk diberikan kredit.

Di dalam jurnalnya, Riska Hanifa menjelaskan tentang penerapan metode berdasarkan beberapa kriteria yang dijadikan sebagai penelitian yaitu data KTP, dokumen legalitas kepemilikan usaha dan beberapa hal yang lain yang berkaitan dengan kredit usaha rakyat. Adapun hasil yang didapat pada penelitian tersebut adalah nama pemilik usaha yang sudah memenuhi kriteria sebagai penerima kredit usaha.

II.2. Konsep SistemInformasi

II.2.1. Pengertian Sistem

Sistem adalah sekelompok unsur yang erat hubungannya satu dengan yang lain, yang berfungsi bersama-sama untuk mencapai tujuan tertentu. Strukur sistem merupakan unsur-unsur yang membentuk sistem tersebut.Sedangkan proses sistem menjelaskan cara kerja setiap unsur sistem tersebut dalam mencapai tujuan

sistem. Dari definisi ini dapat dirinci lebih lanjut pengertian sistem secara umum, yaitu sebagai berikut :

1. Setiap sistem terdiri dari berbagai unsur.
2. Unsur-unsur tersebut merupakan bagian yang tak terpisahkan dari sistem yang bersangkutan.
3. Unsur-unsur didalam sistem tersebut bekerja sama untuk mencapai tujuan sistem.
4. Suatu sistem merupakan bagian dari sistem lain yang lebih besar

Gordon B.Davis dalam bukunya menyatakan bahwa sistem bisa berupa abstrak atau fisik. Sistem yang abstrak adalah susunan gagasan-gagasan atau konsepsi yang teratur yang saling bergantung. Sedangkan sistem yang bersifat fisik adalah serangkaian unsur yang bekerja sama untuk mencapai suatu tujuan. (Tata Sutabri ; 2012 : 5-6).

Suatu sistem dapat terdiri dari bagian-bagian sistem atau subsistem. Subistem-subsistem tersebut berinteraksi sedemikan rupa sehingga tercapai satu kesatuan yang terpadu dan terintegrasi *(integrated)*.

Norman L.Enger mengatakan dalam bukunya bahwa subsistem adalah serangkaian kegiatan yang dapat ditentukan identitasnya yang berhubungan dalam suatu sistem.

Sedangkan Gordon B. Davis dalam bukunya mengatakan bahwa sistem terbagi atas beberapa faktor atau unsur ke dalam beberapa subsistem. Batasan dan penghubung atau *interace* di dalam suatu sistem ditelaah secara cermat untuk

menjamin bahwa hubungan antar subsistem didefinisikan secara jelas bahwa jumlah semua subsistem merupakan keseluruhan sistem (Tata Sutabri ;2012 : 10).

II.2.2. Karakteristik Sistem

Model umum sebuah sistem terdiri dari *input*, proses, dan *output*. Hal ini merupakan konsep sebuah sistem yang sangat sederhana mengingat sebuah sistem dapat mempunyai beberapa masukan dan keluaran sekaligus. Adapun karakteristik yang dimaksud adalah sebagai berikut: (Tata Sutabri ; 2012 : 13-14).

1. Komponen Sistem *(Components)*

   Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, yang bekerja sama membentuk satu kesatuan. Komponen-komponen sistem tersebut dapat berupa satu bentuk sistem. Setiap subsistem memiliki sifat-sifat sistem yang menjalankan suatu fungsi tertentu dan mempengaruhi proses sistem secara keseluruhan.

2. Batasan Sistem *(Boundary)*

   Ruang lingkup sistem merupakan daerah yang membatasi antara sistem dengan sistem lainnya atau sistem dengan lingkungan luarnya.

3. Lingkungan Luar Sistem *(Environtment)*

   Bentuk apapun yang ada di luar ruang lingkup atau batasan sistem yang mempengaruhi operasi sistem tersebut disebut dengan lingkungan luar sistem.

4. Penghubung Sistem *(Interface)*

   Media yang menghubungkan sistem dengan subsistem yang lain dengan penghubung sistem atau *interface.*

5. Masukan Sistem *(Input)*

Energi yang dimasukkan ke dalam sistem disebut masukan sistem, yang dapat berupa pemeliharaan *(maintenance input)* dan sinyal *(signal input).* Sebagai contoh didalam suatu unit sistem komputer, *"program"* adalah *maintenance input* yang digunakan untuk mengoperasikan komputer. Sementara data adalah signal input yang akan diolah menjadi informasi.

6. Keluaran Sistem *(Output)*

Hasil dari energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna. Keluaran ini merupakan masukan bagi subsistem yang lain.

7. Pengolah Sistem *(Process)*

Suatu sistem dapat mempunyai suatu proses yang akan mengubah masukan menjadi keluaran.

8. Sasaran Sistem *(Objective)*

Suatu sistem memiliki tujuan dan sasaran yang pasti dan bersifat deterministik.

II.2.3. Defenisi Sistem Informasi

Suatu sistem didalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial, dan merupakan kegiatan strategi dari suatu organisasi, serta menyediakan laporan-laporan yang diperlukan oleh pihak luar.

Berdasarkan dukungan kepada pemakainya, sistem informasi dibagi menjadi :

1. Sistem Pemrosesan Transaksi (*Transaction Processing System* atau TPS)
2. Sistem Informasi Manajemen (*Management Information System* atau MIS)
3. Sistem Otomasi Perkantoran (*Office Automation System* / OAS)
4. Sistem Pendukung Keputusan (*Decision Support System* atau DSS)
5. Sistem Informasi Eksekutif (*Executive Information System* atau EIS)
6. Sistem Pendukung Kelompok (*Group Support System* atau GSS)
7. Sistem Pendukung Cerdas (*Intelligent Support System* atau ISS).

Mengingat bahwa EIS, DSS, dan MIS digunakan untuk mendukung keputusan manajemen, maka ketiga sistem tersebut sering disebut Sistem Pendukung Manajemen (*Management Support System* atau MSS). (Kusrini ; 2007 : 11-12).

II.3. Konsep Sistem Pendukung Keputusan

II.3.1. Sistem Pendukung Keputusan / *Decision Support System* (DSS)

Decision Support System merupakan sistem informasi interaktif yang menyediakan informasi, pemodelan, dan pemanipulasian data. Sistem ini digunakan untuk membantu pengambilan keputusan dalam situasi yang semiterstruktur dan situasi yang tidak terstruktur, dimana tak seorangpun tahu secara pasti bagaimana keputusan seharusnya dibuat (Alter, 2002).

Decision Support System biasanya dibangun untuk mendukung solusi atas suatu masalah atau untuk mengevaluasi suatu peluang. *Decision Support System* yang seperti itu disebut aplikasi *Decision Support System*. Aplikasi *Decision Support System*digunakan dalam pengambilan keputusan. Aplikasi *Decision*

*Support System*menggunakan CBIS *(Computer Based Information Systems)* yang fleksibel, interaktif, dan dapat diadaptasi, yang dikembangkan untuk mendukung solusi atas masalah manajemen spesifik yang tidak tersruktur.(Kusrini ; 2007 : 15-16).

II.3.2. Tujuan *Decision Support System* (DSS)

Tujuan dari *Decision Support System* (DSS) adalah (Turban, 2005) : (Kusrini ; 2007 : 16-17)

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 dari pada perbaikan efisiensinya.
4. Kecepatan komputasi. Komputer memungkinkan para pengambil keputusan untuk melakukan banyak komputasi secara cepat dengan biaya yang rendah.
5. Peningkatan produktifitas. Membangun satu kelompok pengambilan keputusan, terutama para pakar, bisa sangat mahal. Pendukung terkomputerisasi bisa mengurangi ukuran kelompok dan memungkinkan para anggotanya untuk berada diberbagai lokasi yang berbeda-beda (menghemat biaya perjalanan).

6. Dukungan kualitas. Komputer bisa meningkatkan kualitas keputusan yang dibuat. Sebagai contoh, semakin banyak data yang diakses, makin banyak juga alternatif yang bisa dievaluasi.
7. Berdaya saing. Manajemen dan pemberdayaan sumber daya perusahaan. Tekanan persaingan menyebabkan tugas pengambilan keputusan menjadi sulit. Persaingan didasarkan tidak hanya pada harga, tetapi juga pada kualitas, kecepatan, kustomasi produk, dan dukungan pelanggan. Teknologi pengambilan keputusan bisa menciptakan pemberdayaan yang signifikan dengan cara memperbolehkan seseorang untuk membuat keputusan yang baik secara cepat, bahkan jika mereka memiliki pengetahuan yang kurang.
8. Mengatasi keterbatasan kognitif dalam pemrosesan dan penyimpanan. Menurut Simon (1977), otak manusia memiliki kemampuan yang terbatas untuk memproses dan menyimpan informasi. Orang-orang kadang sulit mengingat dan menggunakan sebuah informai dengan cara bebas dari kesalahan.

Ditinjau dari tingkat teknologinya, *Decision Support System* (DSS) dibagi menjadi 3, yaitu : (Kusrini ; 2007: 18).

1. Sistem Pendukung Keputusan (SPK) Spesifik

   Sistem Pendukung Keputusan (SPK) spesifik bertujuan membantu memecahkan suatu masalah dengan karakteristik tertentu. Misalnya, Sistem Pendukung Keputusan penentuan harga satuan barang.

2. Pembangkit Sistem Pendukung Keputusan (SPK)

   Suatu *software* yang khusus digunakan untuk membangun dan mengembangkan Sistem Pendukung Keputusan (SPK).

3. Perlengkapan Sistem Pendukung Keputusan (SPK)

   Berupa *software* dan *hardware* yang digunakan atau mendukung pembangunan SPK spesifik maupun pembangkit Sistem Pendukung Keputusan (SPK).

Keputusan yang diambil untuk menyelesaikan suatu masalah dilihat dari keterstrukturannya yang bisa dibagi menjadi : (Kusrini ; 2007 : 19-20).

1. Keputusan Terstruktur *(Structured Decison)*

   Keputusan terstruktur adalah keputusan yang dilakukan secara berulang-ulang dan bersifat rutin. Prosedur pengambilan keputusan sangatlah jelas.

2. Keputusan Semiterstruktur *(Semistructured Decision)*

   Keputusan semiterstruktur adalah keputusan yang memiliki dua sifat. Sebagian keputusan bisa ditangani oleh komputer dan yang lain tetap harus dilakukan oleh pengambilan keputusan. Contoh keputusan jenis ini adalah pengevaluasian kredit, penjadwalan produksi, dan pengendalian sediaan.

3. Keputusan Tak Terstruktur *(Unstructured Decision)*

   Keputusan tak terstruktur adalah keputusan yang penanganannya rumit karena tidak terjadi berulang-ulang atau tidak selalu terjadi. Keputusan terstruktur menuntut pengalaman dan berbagai sumber yang bersifat eksternal. Keputusan tersebut umumnya terjadi pada manajemen tingkat atas.

Contohnya adalah keputusan untuk pengembangan teknologi baru, keputusan untuk bergabung dengan perusahaan lain, dan perekrutan eksekutif.

II.3.3. Karakteristik *Decision Support System* (DSS)

Decision Support System memiliki karakteristik dasar sebagai berikut (PowerD. J., 2002) : (Jurnal SEMNASTEKNOMEDIA 2015, STMIK AMIKOM Yogyakarta, 2015).

1. *Decision Support System* dirancang secara khusus untuk memfasilitasi proses pembuatan keputusan.
2. *Decision Support System* seharusnya lebih bersifat membantu, bukan menghasilkan keputusan.
3. *Decision Support System* harus mampu untuk menangani perubahan kebutuhan pembuat keputusan secara cepat. *Decision Support System* seperti tipe sistem informasi lainnya, pada dasarnya terdiri atas tiga bagian utama yaitu masukan, proses, serta keluaran. Yang membedakan *Decision Support System* dengan tipe sistem informasi lainnya adalah jenis masukan dan keluaran serta proses yang dilakukannya.

II.3.4. Komponen-komponen *Decision Support System* (DSS)

Komponen *Decision Support System* dapat berupa : (Sumber : Jurnal SEMNASTEKNOMEDIA 2015 STMIK AMIKOM Yogyakarta ; Adil Setiawan dan Surya Darma ; Februari 2015 : 2).

a. *Data Management,*

   Termasuk *database,* yang mengandung data yang relevan untuk pelbagai situasi dan diatur oleh *software* yang disebut *Database Management Systems* (DBMS).

b. *Model Management,*

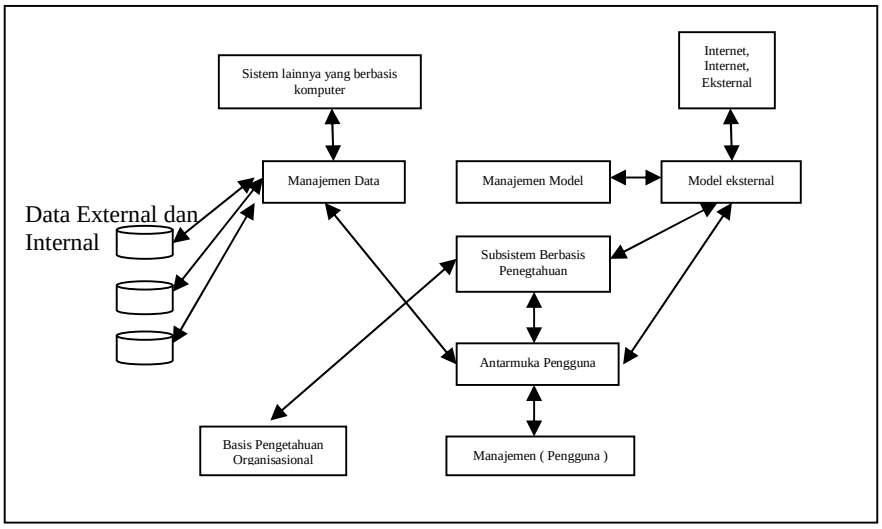
   Melibatkan model *finansial, statistikal, management science,* atau pelbagai model kuantitatif lainnya, sehingga dapat memberikan kesistem suatu kemampuan analitis, dan manajemen *software* yang diperlukan.

c. *Communication (Dialog Subsystem),*

   User dapat berkomunikasi dan memberikan perintah pada *Decision Support System* melalui subsistem ini. Ini berarti menyediakan antarmuka.

d. *Knowledge Management,*

   Subsistem optional ini dapat mendukung subsistem lain atau bertindak sebagai komponen yang berdiri sendiri. Gambar II.1 menunjukkan model konseptual *Decision Support System*.



Gambar II.1.Model Konseptual *Decision Support System* (DSS)
(Sumber : Jurnal SEMNASTEKNOMEDIA 2015 STMIK AMIKOM Yogyakarta ; Adil Setiawan dan Surya Darma ; Februari 2015 : 2)

II.3.5. Langkah-langkah Pemodelan dalam *Decision Support System* (DSS)

Saat melakukan pemodelan dalam pembangunan *Decision Support System* dilakukan langkah-langkah sebagai berikut : (Kusrini ; 2007 : 30-31).

1. Studi Kelayakan (*Intelligence*)

   Pada tahap ini, sasaran ditentukan dan dilakukan pencarian prosedur, pengumpulan data, identifikasi masalah, identifikasi kepemilikan masalah, klasifikasi masalah, hingga akhirnya terbentuk sebuah pernyataan masalah.

   Kepemilikan masalah berkaitan dengan bagian apa yang akan dibangun oleh *Decision Support System* dan apa tugas dari bagian tersebut sehingga model tersebut bisa relevan dengan kebutuhan si pemilik masalah.

2. Perancangan *(Design)*

   Pada tahapan ini akan diformulasikan model yang akan digunakan dan kriteria-kriteria yang ditentukan. Setelah itu, dicari alternatif model yang bisa

menyelesaikan permasalahan tersebut. Langkah selanjutnya adalah memprediksi keluaran yang mungkin. Kemudian, ditentukan variabel-variabel model.

3. Pemilihan *(Choice)*

   Setelah pada tahap *design* ditentukan berbagai alternatif model beserta variabel-variabelnya, pada tahapan ini akan dilakukan pemilihan modelnya, termasuk solusi dari model tersebut. Selanjutnya dilakukan analisis sensitivitas, yakni dengan mengganti beberapa variabel.

4. Membuat *Decision Support System*

   Setelah menentukan modelnya, berikutnya adalah mengimplementasikan dalam aplikasi *Decision Support System*.

II.4. Metode *Preference Ranking Organization For Enrichment Evaluation (PROMETHEE).*

Metode *Promethee* termasuk kedalam kelompok pemecahan masalah *Multi Criteria Decision Making (MCDM)* atau pengambilan keputusan kriteria majemuk yang merupakan disiplin ilmu yang sangat penting dalam pengambilan keputusan atau suatu masalah yang memiliki lebih dari satu kriteria *(multikriteria).*

Menurut Brans dan Marcschal (1999;15), *Promethee* yang merupakan singkatan dari *Preference Ranking Organization For Enrichment Evaluation* adalah metode *outranking* yang menawarkan cara yang fleksibel dan sederhana kepada user (pembuat keputusan) untuk menganalisis masalah-masalah

multikriteria. Prinsip yang digunakan adalah penetapan prioritas alternatif yang telah ditetapkan berdasarkan pertimbangan kaidah dasar :

$$\text{Max } \{f_1(x),f_2(x),f_3(x),.....,f_i(x),.....f_k(x)\ I_x$$

Dimana k adalah sejumlah kumpulan alternatif dan f_i (i=1,2,...., k) merupakan nilai/ukuran relatif kriteria untuk masing-masing alternatif.

Termasuk dalam keluarga dari metode *outranking* yang dikembangkan oleh B.Roy (Brans et al, 1999) dan meliputi dua fase :

1. Membangun hubungan *outranking* dari K, dimana K adalah sejumlah kumpulan alternatif
2. Eksploitasi dari hubungan ini memberikan jawaban optimasi kriteria dalam paradigma permasalahan multikriteria.

Dalam fase pertama, nilai hubungan *outranking* berdasarkan pertimbangan dominasi masing-masing kriteria. Indeks preferensi ditentukan dan nilai *outranking* secara grafis disajikan berdasarkan preferensi dari pembuat keputusan. (Makalah Ilmiah Informasi dan Teknologi Ilmiah (INTI), Vol. 1, No. 1 ; Dewi Safitri Hutabarat ; 2013 : 14).

Tujuan utama dari pendekatan *PROMETHEE* ini adalah untuk mempermudah proses pengambilan keputusan dengan cara mengelompokkan tipe keputusan menjadi 6 fungsi kriteria yang cukup dapat mewakili semua jenis keputusan untuk menyelesaikan kasus-kasus sehari-hari dan melakukan kuantifikasi derajat preferensi dengan menggunakan maksimum 2 parameter yang

memiliki karakteristik ekonomi yang signifikan. (Jurnal Itenas Rekayasa, Vol. 13, No. 4 ; Ambar Harsono, et al ; Oktober-Desember 2009 : 187)

II.4.1 Nilai Hubungan *Outranking* dalam *Promethee*

1. Dominasi Kriteria

Nilai *f* merupakannilai nyata dari suatu kriteria : (Jurnal PelitaInformatika Budi Darma, Vol. 4, No. 2 ; Reizha Arsita ; Agustus 2013 : 107)

$$F:\mathrm{K} \rightarrow \mathrm{R}$$

Untuk setiap alternatif a € K, f (a) merupakan evaluasi dari alternatif tersebut untuk suatu kriteria. Pada saat dua alternatif dibandingkan, a, b € K, harus dapat ditentukan perbandingan preferensinya. Penyampaian intesitas (P) dari preferensi alternatif a terhadap b sedemikian rupa sehingga :

a. P (a,b) = 0, berarti tidak ada *(indefferent)* antara a dan b, atau tidak ada preferensi daria lebih baikdarib.
b. P (a,b) ~ 0, berartilemah preferensi darialebih baikdarib.
c. P (a,b) ~ 1, berartikuat preferensi darialebih baik dari b.
d. P (a,b) = 1, berarti mutlak preferensi dari a lebih baikdarib.

Dalammetodeini, fungsi preferensi seringkali menghasilkan nilai fungsi yangberbeda antara dua evaluasi, sehingga :

$$P(a,b) = P\,(f(a) - f(b)) \quad \text{..... (1)}$$

2. Rekomendasi Fungsi Preferensi Untuk Keperluan Aplikasi

Dalam*Promethee* disajikan enam bentuk fungsi preferensi kriteria. Enampreferensi tersebut adalah sebagai berikut : (Jurnal Pelita Informatika Budi Darma, Vol. 4, No. 2 ; Reizha Arsita ;Agustus 2013 : 107)

a. Kriteria Biasa *(Usual Criterian)*

$$H(d) = \begin{cases} 0 \; jika \; d \; \leq 0 \\ 1 \; jika \; d \; > 0 \end{cases} \text{..... (2)}$$

Dimana :

H (d) = fungsi selisih kriteria antar alternatif

d = selisih nilai kriteria $\{d = f(a) - f(b)\}$

Pada kasus ini, tidak ada beda (sama penting) antara a dan b jika dan hanya jika kriteria *f*(a) = *f*(b); apabila kriteria pada masing-masing alternatif memiliki nilai berbeda, pembuat keputusan membuat preferensi mutlak untuk alternatif memiliki nilai yang baik.

b. Kriteria Quasi *(Quasi Criterian)*

$$H(d) = \begin{cases} 0 \; jika \; d \; \leq q \\ 1 \; jika \; d \; > q \end{cases} \text{..... (3)}$$

Dimana :

H (d) = fungsi selisih kriteria antar alternatif

d = selisih nilai kriteria $\{d = f(a) - f(b)\}$

q = harus merupakan nilai tetap

Dua alternatif memiliki preferensi yang sama penting selama selisih atau nilai H(d) dari masing-masing alternatif untuk kriteria tertentu tidak melebihi

Nilai q maka terjadi bentuk preferensi mutlak. Jika pembuat keputusan menggunakan kriteriakuasi, maka harus menentukan nilai q, dimana nilai ini dapat menjelaskan pengaruh yang signifikan dari suatu kriteria.

c. Kriteria Dengan Preferensi Linier

$$H(d) = \begin{cases} 0 \; jika \; d \leq q \\ \frac{d}{p} \; jika \; 0 \leq d \leq p \\ 1 \; jika \; d > p \end{cases} \quad \text{..... (4)}$$

Dimana :

H (d) = fungsi selisih kriteria antar alternatif

d = selisih nilai kriteria $\{d = f(a) - f(b)\}$

q = nilai kecendrungan atas

Kriteria preferensi linier dapat menjelaskan bahwa selama nilai selisih memiliki nilai yang lebih rendah dari p, preferensi dari pembuat keputusan meningkat secara linier dengan nilai d. Jika nilai d lebih besar dibandingkan dengan nilaip, maka terjadi preferensi mutlak. Pada saat pembuat keputusan mengidentifikasi beberapa kriteria untuk tipe ini, harus ditentukan nilai dari kecenderungan atas (nilai p).

d. Kriteria Level *(Level Criterion)*

$$H(d) = \begin{cases} 0 \; jika \; d \leq q \\ 0{,}5 \; jika \; q < d \leq p \\ 1 \; jika \; d > p \end{cases} \quad \text{..... (5)}$$

Dimana :

H (d) = fungsi selisih kriteria antar alternatif

d = selisih nilai kriteria $\{d = f(a) - f(b)\}$

p = nilai kecendrungan atas

q = harus merupakan nilai yang tetap

Dalamkasus ini, kecenderungan tidak berbeda q dan kecenderungan preferensi adalah ditentukan secara simultan. Jika d berada diantara nilai q dan p, hal ini berarti situasi preferensi yang lemah(H(d) = 0,5).

e. Kriteria dengan Preferensi Linier dan Area yang Tidak Berbeda

$$H(d) = \begin{cases} 0 \; jika \; d \leq q \\ \frac{d-q}{p-q} \; jika \; q \leq d \leq p \\ 1 \; jika \; d > p \end{cases} \quad \text{..... (6)}$$

Dimana :

H (d) = fungsi selisih kriteria antar alternatif

d = selisih nilai kriteria $\{d = f(a) - f(b)\}$

p = nilai kecendrungan atas

q = harus merupakan nilai yang tetap

Pada kasus ini, pengambil keputusan mempertimbangkan peningkatan preferensi secara linier dari tidak berbeda hingga preferensi mutlak dalam area antara dua kecenderungan q dan p.

f. Kriteria Gaussian *(Gaussian Criterion)*

$$H(d) = \begin{cases} 0 & jika \; d \leq 0 \\ 1 - e - \frac{d^2}{2a^2} \; jika \; d > 0 \end{cases} \quad \text{..... (7)}$$

Dimana :

H (d) = fungsi selisih kriteria antar alternatif

d = selisih nilai kriteria $\{d = f(a) - f(b)\}$

II.4.2. Indeks Preferensi Multikriteria

Indeks preferensi multikriteria ditentukan berdasarkan rata-rata bobot dari fungsi preferensi Pi.

$$\varphi(a,b) = \sum_{i=1}^{n} \pi_i P_i(a,b) : \forall a,b \in A \quad \text{..... (8)}$$

$\varphi(a,b)$ merupakan intensitas preferensi pembuat keputusan yang menyatakan bahwa alternatif a lebih baik dari alternatif b dengan pertimbangan secara simultan dari keseluruhan kriteria. Hal ini dapat disajikan dengan nilai antara nilai 0 dan 1, dengan ketentuan sebagai berikut :

a. $\varphi(a,b) = 0$ menunjukkan preferensi yang lemah untuk alternatif a > alternatif b berdasarkan semua kriteria.
b. $\varphi(a,b) = 1$ menunjukkan preferensi yang kuat untuk alternatif a > alternatif b berdasarkan semua kriteria.

Indeks preferensi ditentukan berdasarkan nilai hubungan outranking pada sejumlah kriteria dari masing-masing alternatif. Hubungan ini dapat disajikan sebagai grafik nilai outranking, node-nodenya merupakan alternatif berdasarkan penilaian kriteria tertentu.(Jurnal Pelita Informatika Budi Darma, Vol. 4, No. 2 ; Reizha Arsita ; Agustus 2013 : 108)

II.4.3.*Promethee Ranking*

Perhitungan arah preferensi dipertimbangkan berdasarkan nilai indeks : (Jurnal Pelita Informatika Budi Darma, Vol. 4, No. 2 ; Reizha Arsita ; Agustus 2013 : 108)

1. *Leaving Flow*

$$\varphi^{+}(a) = \frac{1}{n-1}\sum_{x \in A} \varphi(a, x) \quad \text{..... (9)}$$

2. *Entering Flow*

$$\varphi^{-}(a) = \frac{1}{n-1}\sum_{x \in A} \varphi(a, x) \quad \text{..... (10)}$$

3. *Net Flow*

$$\varphi(a) = \varphi^{+}(a) - \varphi^{-}(a) \quad \text{..... (11)}$$

Keterangan :

$\varphi(a,x)$ = menunjukkan preferensi bahwa alternatif lebih baik dari alternatif x.

$\varphi(a,x)$ = menunjukkan preferensi bahwa alternatif x lebih baik dari alternatif.

$\varphi^{+}(a)$ = *Leaving Flow,* digunakan untuk menentukan urutan prioritas pada proses *Promethee* I yang menggunakan urutan persial

$\varphi^{-}(a)$ = *Entering Flow,* digunakan untuk menentukan urutan prioritas pada proses *Promethee* I yang mengunakan urutan parsial.

$\varphi(a)$ = *Net Flow,* digunakan untuk menghasilkan keputusan akhir penentuan urutan dalam menyelesaikan masalah sehingga menghasilkan urutan lengkap.

II.5. Tumbuhan Obat

Tumbuhan obat adalah semua tumbuhan yang dapat digunakan sebagai obat, berkisar dari yang terlihat oleh mata hingga yang nampak dibawah mikroskop. Menurut Zuhud dalam (Kartikawati, 2004), tumbuhan obat adalah seluruh jenis tumbuhan obat yang diketahui atau dipercaya mempunyai khasiat obat yang dikelompokkan menjadi :

1. Tumbuhan obat tradisional, yaitu: jenis tumbuhan obat yang diketahui atau dipercayaoleh masyarakat mempunyai khasiat obat dan telah digunakan sebagai bahan baku obattradisional.
2. Tumbuhan obat modern, yaitu: jenis tumbuhan yang secara ilmiah telah dibuktikanmengandung senyawa atau bahan bioaktif yang berkhasiat obat dan penggunaannyadapat dipertanggungjawabkan secara medis.
3. Tumbuhan obat potensial, yaitu: jenis tumbuhan obat yang diduga mengandung senyawaatau bahan aktif yang berkhasiat obat, tetapi belum dibuktikan secara ilmiah ataupenggunaannya sebagai obat tradisional sulit ditelusuri.

II.6. Pengertian Basis Data

Sebuah basis data adalah sebuah kumpulan data yang saling berhubungan secara logis dan merupakan sebuah penjelasan dari data tersebut, yang didesain untuk menemukan data yang dibutuhkan oleh sebuah organisasi. Basis data juga merupakan sekumpulan data elemen data yang saling terintegrasi (Indrajani; 2014 : 70).

Berdasarkan tingkat kompleksitas nilai data, tingkatan data dapat disusun dalam sebuah hierarki, mulai dari yang paling sederhana hingga paling sederhana hingga paling kompleks (Edy Sutanta ; 2011 : 35-36).

1. Sistem basis data, merupakan sekumpulan subsistem yang terdiri atas basis data dengan para pemakai yang menggunakan basis data secara bersama-sama, personal-personal yang merancang dan mengelola basis data, teknik-

teknik untuk merancang dan mengelola basis data, serta sistem komputer untuk mendukungnya.

2. Basis data, merupakan sekumpulan dari bermacam-macam tipe *record* yang memiliki hubungan antar-*record* dan rincian data terhadap obyek tertentu.
3. File, merupakan sekumpulan *record* sejenis secara relasi yang tersimpan dalam media penyimpanan sekunder.
4. *Record,* merupakan *field*/atribut/data item yang saling berhubungan terhadap obyek tertentu.
5. *Data item/field/*atribut, merupakan unit terkecil yang disebut data, sekumpulan *byte* yang mempunyai makna.
6. *Data agregate,* merupakan sekumpulan data item/*field*/atribut dengan ciri tertentu dan diberi nama.
7. *Byte,* adalah bagian terkecil yang dialamatkan dalam memori. *Byte* merupakan sekumpulan *bit* yang secara konvensional terdiri atas kombinasi 8 *bit* biner yang menyatakan sebuah karakter dalam memori (1 *byte* = 1 karakter).
8. *Bit,* adalah sistem biner yang terdiri atas dua macam nilai, yaitu 0 dan 1. Sistem biner merupakan dasar yang dapat digunakan untuk komunikasi antara manusia dan mesin (komputer).

II.7. Normalisasi

Normalisasi diartikan sebagai suatu teknik yang menstrukurkan/ mendekomposisi data dalam cara-cara tertentu untuk mencegah timbulnya

permasalahan pengolahan data dalam basis data. Permasalahan yang dimaksud adalah berkaitan dengan penyimpangan-penyimpangan *(anomalies)* yang terjadi akibat adanya kerangkapan data dalam relasi dan in-efisiensi pengolahan (Martin, 1975). (Edy Sutanta ; 2011 : 174)

Proses normalisasi menghasilkan relasi yang optimal, yaitu (Martin, 1975): (Edy Sutanta ; 2011 : 175)

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 antardata 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.8. *Unified Modelling Language (UML)*

Pada perkembangan teknologi perangkat lunak, diperlukan adanya bahasa yang digunakan untuk memodelkan perangkat lunak yang akan dibuat dan perlu adanya standarisasi agar orang diberbagai negara dapat mengerti pemodelan perangkat lunak. Seperti yang kita ketahui bahwa menyatukan banyak kepala untuk menceritakan sebuah ide dengan tujuan untuk memahami hal yang sama tidaklah mudah, oleh karena itu diperlukan sebuah bahasa pemodelan perangkat lunak yang dapat dimengerti oleh banyak orang.

Banyak orang yang telah membuat bahasa pemodelan pembangunan perangkat lunak sesuai dengan teknologi pemrograman yang berkembang pada saat itu, misalnya yang sempat berkembang dan digunakan oleh banyak pihak adalah *Data Flow Diagram* (DFD) untuk memodelkan perangkat lunak yang menggunakan pemrograman prosedural atau struktural, kemudian juga ada *State Transition Diagram* (STD) yang digunakan untuk memodelkan sistem *real time* (waktu nyata).

Pada perkembangan teknik pemrograman berorientasi objek, muncullah sebuah standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang

dibangun dengan menggunakan teknik pemrograman berorientasi objek, yaitu *Unified Modelling Language* (UML). UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasi, menggambarkan, membangun dan dokumentasi dari sistem perangkat lunak.

"UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung" (Shalahuddin, M. dan Rosa A.S, 2014:137).

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. Seperti yang kita ketahui bahwa banyak hal di dunia sistem informasi yang tidak dapat dibakukan, semua tergantung kebutuhan, lingkungan dan konteksnya. Begitu juga dengan perkembangan penggunaan UML bergantung pada *level* abstraksi penggunaannya. Jadi belum tentu pandangan yang berbeda dalam penggunaan UML adalah suatu yang salah, tapi perlu ditelaah dimanakah UML digunakan dan hal apa yang ingin divisualkan.

II.9. *Use Case Diagram*

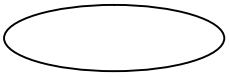
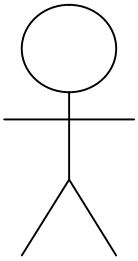
Use case atau diagram use case merupakan pemodelan untuk melakukan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu.

Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut aktor dan *use case.*

1. Aktor merupakan 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.
2. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

Berikut adalah simbol-simbol yang ada pada diagram *use case* :

Tabel II.1 Simbol-Simbol *Use CaseDiagram*

Simbol	Nama	Keterangan
	*Use Case*	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama *use case*.
	*Actor*	Orang, proses, atau sistem lain yang berorientasi 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 menggunakan kata benda di awal frase nama aktor.

————————	*Association*	Komunikasi antar aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor.
<<extend>> ----------►	*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 *use case* tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan.
————————►	*Generalization*	Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.
<<include>> ----------►	*Include*	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini.

(Sumber : Shalahuddin, M. dan Rosa A.S, 2014:156, Rekayasa Perangkat Lunak)

II.10. *Activity Diagram*

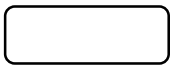
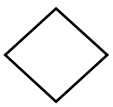
Diagram aktivitas atau *activity diagram* menggambarkan *work flow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan *actor*, jadi aktivitas yang dapat dilakukan oleh sistem.

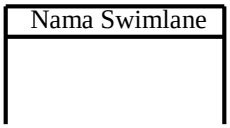
Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut :

1. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
2. Urutan atau pengelompokan tampilan dari sistem/ *user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
3. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.
4. Rancangan menu yang ditampilkan pada perangkat lunak.

Berikut adalah simbol-simbol yang ada pada diagram aktivitas :

Tabel II.2 Simbol *ActivityDiagram*

Simbol	Nama	Keterangan
	Status Awal	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
	Aktivitas	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
	Percabangan/ *Decision*	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
	Penggabungan/ *Join*	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
	Status Akhir	Status akhir yang dilakukan sistem, sebuah diagram aktivitas

		memiliki sebuah status akhir.
	Swimlane	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

(Sumber : Shalahuddin, M. dan Rosa A.S, 2014:162, Rekayasa Perangkat Lunak)

II.11. *Class Diagram*

Diagram kelas atau *class diagram* menggambarkan struktur system dari segi pendefenisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi.

a. Atribut merupakan variable-variabel yang dimiliki oleh suatu kelas
b. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

Diagram kelas dibuat agar pembuat program atau *programmer* membuat kelas-kelas sesuai rancangan di dalam diagram kelas agar Antara dokumentasi perancangan dan perangkat lunak sinkron. Banyak berbagai kasu, perancangan kelas yang dibuat tidak sesuaidengan kelas-kelas yang dibuat pada perangkat lunak, sehingga tidaklah ada gunanya lagi sebuah perancangan karena apa yang dirancang dan hasil jadinya tidak sesuai.

Kelas-kelas yang ada pada struktur system harus dapat melakukan fungsi-fungsi-fungsi sesuai dengan kebutuhan system sehingga pembuat perangkat lunak dapat membuat kelas-kelas didalam program perangkat lunak sesuai dengan perancangan diagram kelas. Susunan struktur kelas yang baik pada diagram kelas sebaiknya memiliki jenis-jenis kelas berikut :

a. Kelas main : kelas yang memiliki fungsi awal dieksekusi ketika system dijalankan.
b. Kelas yang menangani tampilan system (*view*) : kelas yang mendefenisikan dan mengatur tampilan ke pemakai.
c. Kelas yang diambil dari pendefenisian *use case* (*controller*) : kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefenisian *use case*, ,kelas ini biasanya disebut dengan kelas proses yang menangani proses bisnis pada perangkat lunak.
d. Kelas yang diambil dari pendefenisian data (*model)* : kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data. Semua table yang dibuat di basis data dapat dijadikan kelas, namun untuk table dari hasil ralasi atau atribut mutivalue pada ERD dapat dijadikan kelas tersendiri dapat juga tidak asalkan pengaksesannya dapat dipertanggungjawabkan atau tetap ada didalam perancangan kelas.

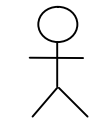
II.12. *Sequence Diagram*

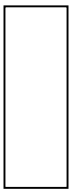
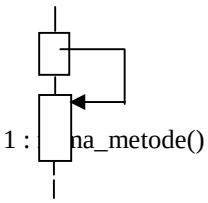
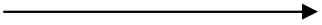
Diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirmkan dan diterima antar objek. Oleh karean itu untuk menggambar diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimilik kelas yang diinstansiasi menjadi objek itu. Membuat diagram sekuen juga dibutuhkan untuk melihat scenario yang ada pada *use case*.

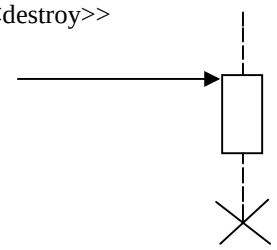
Banyaknya diagram sekuen yang harus digambar adalah minimal sebanyak pendefenisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefenisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak *use case* yang didefenisikan maka diagram sekuen yang harus dibuat juga semakin banyak.

Berikut adalah simbol-simbol yang ada pada diagram sekuen:

Tabel II.3. Simbol *Sequence Diagram*

Simbol	**Deskripsi**
 Nama aktor Atau Nama aktor Tanpa waktu aktif	Orang, proses, atau sistem lain yanng 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 menggunakan kata benda di awal frase nama aktor
Garis hidup / *lifeline* 	Menyatakan kehidupan suatu objek
Objek Nama objek ; nama kelas	Menyatakan objek yang berinteraksi pesan

Waktu aktif 	Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan didalamnya.
Pesan tipe create <<create>> 	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat.
Pesan tipe call 1 : nama_metode() 	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri,  Arah panah mengarah pada objek yang memiliki operasi/metode, karena ini memanggil operasi/metode maka operasi/metode yanng dipanggil harus ada pada diagram kelas sesuai dengan kelas objek yang berinteraksi.
Pesan tipe send 1 : masukan 	Menyatakan bahwa suatu objek mengirimkan data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi.
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

Pesan tipe destroy <<destroy>> 	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhri, sebaiknya jika ada create maka ada destroy.

(Sumber : Shalahuddin, M. dan Rosa A.S, 2014:165-167, Rekayasa Perangkat Lunak)

II.13. *Microsoft Visual Basic 2010*

Visual Basic 2010 merupakan salah satu pemrograman terbaru yang dikeluarkan oleh *Microsoft,* yaitu *Microsoft Visual Studio 2010. Visual Studio* merupakan produk pemrograman andalan dari *Microsoft Corporation*, dimana didalamnya 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 standart yang disertaka adalah *SQL Server 2008 Express*.

Visual Basic 2010 merupakan versi perbaikan dan pengembangan dari versi pendahulunya, yaitu *Visual Basic 2008*. Beberapa pengembangan yang terdapat didalamnya antara lain dukungan terhadap *library* terbaru dari *Microsoft*, yaitu *.Net FrameWork 4.0,* dukungan terhadap aplikasi berbasis *Cloud Computing,* serta perluasan dukungan terhadap database-database, baik *standalone* maupun *database server.* (Wahana Komputer ; 2011 : 2).

II.14. *Microsoft 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 :

1. 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.
2. 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:

1. 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.

2. Versi Express, ini adalah versi “Ringan” dari semua versi yang ada(tetapi versi ini berbeda dengan versi compact) dan paling cocok untuk latihan para pengembang aplikasi. Versi ini memuat *Express Manager standar,* integrasi dengan CLR dan XML. (Jurnal Sistem Informasi ; Wenny Widya, at al ; 2009 : 3-4)