BAB 2
TINJAUAN PUSTAKA

2.1 Sistem Pendukung Pegambilan Keputusan

Sistem pendukung keputusan (SPK) dirancang sedemikian rupa sehingga bersifat interaktif dengan pemakainya. Maksud dan tujuan dari adanya SPK, yaitu untuk mendukung pengambil keputusan memilih alternatif keputusan yang merupakan hasil pengolahan informasi - informasi yang diperoleh/tersedia dengan menggunakan model-model pengambil keputusan serta untuk menyelesaikan masalah-masalah bersifat terstruktur, semi terstruktur dan tidak terstruktur [10].

2.1.1 Karakteristik Sistem Pendukung Keputusan

Konsep Sistem Pendukung Keputusan pertama kali diperkenalkan pada tahun 1970-an oleh Michael S. Scott Morton dengan istilah Management Decision Model [9]. Konsep sistem pendukung keputusan ditandai dengan sistem interaktif berbasis komputer yang membantu pengambil keputusan memanfaatkan data dan model untuk menyelesaikan masalah-masalah yang tidak restruktur [11].

Pada dasarnya sistem pendukung keputusan dirancang untuk mendukung seluruh tahap pengambilan keputusan mulai dari mengidentifikasi masalah, memilih data yang relevan, menentukan pendekatan yang digunakan dalam proses pengambilan keputusan, sampai mengevaluasi pemilihan interaktif. Peranan sistem pendukung keputusan dalam konteks keseluruhan sistem informasi ditujukan untuk memperbaiki kinerja melalui aplikasi teknologi informasi. Terdapat sepuluh karakteristik dasar sistem pendukung keputusan yang efektif, yaitu [11]:

1. Mendukung proses pengambilan keputusan, menitikberatkan pada *management by perception*
2. Adanya interface manusia/ mesin dimana manusia (*user*) tetap mengontrol proses pengambilan keputusan
3. Mendukung pengambilan keputusan untuk membahas masalah-masalah terstruktur, semiterstruktur, dan tidak restruktur
4. Menggunakan model-model matematis dan statistik yang sesuai.

5. Memiliki kapabilitas dialog untuk memperoleh informasi sesuai dengan kebutuhan – model interaktif
6. Output ditunjukkan untuk personil organisasi dalam semua tingkatan.
7. Memiliki subsistem-subsistem yang terintegrasi sedemikian rupa sehingga dapat berfungsi sebagai kesatuan sistem
8. Membutuhkan struktur data komprehensif yang dapat melayani kebutuhan informasi keseluruhan tingkatan manajemen
9. Pendekatan *easy to use*. Ciri suatu sistem pendukung keputusan yang efektif adalah kemudahan untuk digunakan, dan memungkinkan keleluasaan pemakai untuk memilih atau mengembangkan pendekatan-pendekatan baru dalam membahas masalah yang dihadapi
10. Kemampuan sistem beradaptasi secara tepat, dimana pengambil keputusan dapat menghadapi masalah-masalah baru, dan pada saat yang sama dapat menangani dengan cara mengadaptasi sistem terhadap kondisi-kondisi perubahan yang terjadi.

2.1.2 Komponen-komponen Sistem Pendukung Keputusan

Suatu sistem pendukung keputusan memiliki tiga subsistem utama yang menentukan kapabilitas teknis sistem pendukung keputusan tersebut, yaitu [11]:

1. Subsistem Manajemen Basis Data (*Database Management Subsystem*)

Sistem pendukung keputusan membutuhkan proses ekstraksi dan DataBase Management Subsystem (DBMS) yang dalam pengelolaannya harus cukup fleksibel untuk memungkinkan penambahan dan pengurangan secara cepat. Dalam hal ini, kemampuan yang dibutuhkan dari manajemen database dapat diringkas, sebagai berikut :

a. Kemampuan untuk mengkombinasikan berbagai variasi data melalui pengambilan dan ekstraksi data
b. Kemampuan untuk menambahkan sumber data secara cepat dan mudah
c. Kemampuan untuk menggambarkan struktur data logikal sesuai dengan pengertian pemakai sehingga pemakai mengetahui apa yang tersedia dan dapat menentukan kebutuhan penambahan dan pengurangan.

d. Kemampuan untuk menangani data secara personil sehingga pemakai dapat mencoba berbagai alternatif pertimbangan personil
e. Kemampuan untuk mengelola berbagai variasi data

2. Subsistem Manajemen Basis Model (*Model Base management Subsystem*)

Salah satu keunggulan dalam sistem pendukung keputusan adalah kemampuan untuk mengintegrasikan akses data dan model-model keputusan. Hal ini dapat dilakukan dengan menambahkan model-model keputusan ke dalam sistem informasi yang menggunakan database sebagai mekanisme integrasi dan komunikasi di antara model-model. Kemampuan yang dimiliki subsistem basis model meliputi :

a. Kemampuan untuk menciptakan model-model baru secara cepat dan mudah
b. Kemampuan untuk mengakses dan mengintegrasikan model-model keputusan
c. Kemampuan untuk mengelola basis data dengan fungsi manajemen yang analog dan manajemen basis data (seperti mekanisme untuk menyimpan, membuat dialog, menghubungkan, dan mengakses model).

3. Subsistem Perangkat Lunak Penyelenggara Dialog (*Dialog Generation and Management Software*)

Fleksibilitas dan kekuatan karakteristik sistem pendukung keputusan timbul dari kemampuan interaksi antara sistem dan pemakai,yang dinamakan subsistem dialog. Bennet mendefinisikan pemakai,terminal, dan sistem perangkat lunak sebagai komponen-komponen dari sistem dialog. Ia membagi subsistem dialog menjadi tiga bagian, yaitu [12]:

a. Bahasa aksi, meliputi apa yang dapat digunakan pemakai dalam berkomunikas dengan sistem.
b. Bahasa tampilan atau presentasi, meliputi apa yang harus diketahui oleh pemakai.
c. Basis Pengetahuan, meliputi apa yang harus diketahui oleh pemakai.

2.2 Metode Soft Multiset

Soft Multiset merupakan *tool* matematika yang mampu menyelesaikan kasus ketidakpastian pada bidang teknik, *medical science*, ekonomi, *enviroments*, dan masalah ketidakpastian pada bidang lainnya. *Soft Multiset* merupakan generalisasi dari *tool* matematika *Soft Set*, sehingga konsep dasar yang digunakan oleh kedua metode ini sama. *Soft Set* pertama kali diperkenalkan oleh Molodtsov[13]. Molodtsov mendefinisikan *soft set* seperti berikut, misalkan U merupakan inisial dari himpunan semesta dan E merupakan himpunan parameter, P(U) menunjukkan himpunan kuasa dari U dan A⊂ E sehingga (F,A) merupakan *soft set* U dimana F : A ⟶ P(U). *Soft set* U merupakan himpunan bagian dari himpunan semesta U.

Soft set ditunjukkan sebagai berikut, misalkan U merupakan himpunan rumah idaman dan E merupakan himpunan paremater dalam pemilihan rumah seperti murah, modern, indah, dll. Dalam kasus ini *soft set* berarti menunjukkan rumah murah, rumah modern, dan rumah indah. Sehingga *soft set* (F,E) mendeskripsikan rumah idaman yang diinginkan.

Hal yang membedakan antara *soft set* dengan *soft multiset* adalah *soft set* hanya mampu menangani kasus yang memiliki satu himpunan semesta (U), sedangkan *soft multiset* mampu menangani kasus yang memiliki lebih dari satu himpunan semesta [14]. Misalkan $\{U_i : i \in I\}$ merupakan kumpulan dari himpunan semesta dan misalkan $\{E_{U_i} : i \in I\}$ merupakan kumpulan himpunan parameter, maka

$$U = \prod_{i=1}^{n} P(U_i), \qquad E = \prod_{i=1}^{n} E_U \quad , di \quad A \subseteq E \tag{2.1}$$

Keterangan :

n = banyak himpunan semesta alternatif

$P(U_i)$ = himpunan kuasa alternatif

E_U = himpunan kriteria yang berhubungan dengan himpunan semesta

Algoritma penyelesaian kasus pendukung keputusan menggunakan metode *soft multiset* adalah sebagai berikut [15]:

1. Buat himpunan soft multiuset dan lakukan operasi *Union* pada himpunan *soft multiset* alternatif
2. Hasil *union* dari penilaian *decision maker* disimpan dalam tabel *resultan*
3. Buat tabel perbandingan untuk *soft multiset* berdasarkan nilai pada tabel *resultan*. Tabel perbandingan merupakan tabel persegi dimana jumlah baris dan kolomnya sama. Baris dan kolom diberi label yang sama sesuai dengan nama objek atau alternatif dari himpunan semesta. Isi table perbandingan Cij diperoleh dari jumlah kriteria yang nilai keanggotaan xi lebih besar atau sama dengan nilai keanggotaan xj
4. Hitung nilai r_i dengan menggunakan persamaan berikut

$$r_i = \sum_{j-1}^{n} c_i \qquad (2.2)$$

   Keterangan :

   j = jumlah objek (kolom)

   c_i = nilai bobot dari tabel perbandingan

5. Hitung nilai t_j dengan menggunakan persamaan berikut

$$t_j = \sum_{i-1}^{n} c_i \qquad (2.3)$$

   Keterangan :

   i = jumlah objek (baris)

   c_i = nilai bobot dari tabel perbandingan

6. Hitung nilai S_i dengan menggunakan persamaan berikut

$$S_i = r_i - t_j \qquad (2.4)$$

   Keterangan :

   r_i = jumlah nilai baris

   t_j = jumlah nilai kolom

7. Gunakan kembali himpunan hasil operasi *union* pada perhitungan objek sebelumnya dan mengganti nilai himpunan objek lain dengan 0 jika nilai hasil alternatif pada proses sebelumnya merupakan bukan nilai maksimal. Jika nilai

alternatif sebelumnya merupakan nilai maksimal maka, nilai objek lain tidak diganti.

8. Lakukan langkah 3 sampai 6, sesuai dengan jumlah objek yang dimiliki oleh kasus.

2.3 Metode *Entropy*

Metode *entropy* merupakan metode yang dapat digunakan untuk menentukan suatu bobot pada kriteria dengan variasi nilai tertinggi akan mendapatkan bobot tertinggi. Metode *entropy* bisa digunakan untuk menghitung bobot dalam pengambilan keputusan berkelompok (*Group Decision Making*). Salah satu kelebihan dari pendekatan *entropy* adalah kemampuannya dalam mengakomodasi nilai bobot yang berasal dari beberapa pembuat keputusan.

Adapun langkah-langkah pembobotan dengan menggunakan metode *entropy* adalah sebagai berikut [4]:

1. Semua pengambil keputusan harus memberikan nilai yang menunjukkan kepentingan suatu kriteria tertentu terhadap pengambilan keputusan. Tiap pengambil keputusan boleh menilai sesuai preferensinya masing-masing.
2. Kurangkan tiap angka tersebut dengan nilai paling ideal, hasil pengurangan tersebut dinyatakan dengan k_{ij}.
3. Bagi tiap nilai (k_{ij}) dengan jumlah total nilai dalam semua kriteria

$$a = \frac{k_i}{\sum_{i-1}^{m}\sum_{i-1}^{n} k_i} \tag{2.5}$$

Keterangan :

m = jumlah pengambil keputusan

n = jumlah kriteria

k_i = nilai normalisasi data awal

4. Menghitung nilai entropy untuk tiap kriteria dengan rumus berikut :

(2.6)

$$E = \left[\frac{-1}{\ln(m)}\right]\sum_{i=1}^{n}\left[a_i \ln(a_i)\right]$$

Keterangan :

m = jumlah pengambil keputusan

n = jumlah pengambil keputusan

a_i = nilai perhitungan matriks

E_j = nilai entropy

5. Menghitung dispersi tiap kriteria dengan rumus berikut:

$$D_j = 1 - E_j \quad (2.7)$$

Keterangan :

E_j = nilai perhitungan entropy

D_j = nilai Dispersi

6. Karena diasumsikan total bobot adalah 1, maka untuk mendapatkan bobot tiap kriteria, nilai dispersi harus dinormalisasikan dahulu, sehingga:

$$W = \frac{D_j}{\sum D_j} \quad (2.8)$$

Keterangan :

D_j = hasil perhitungan dispersi

2.4 Metode MCDM (*Multi Criteria Decision Making*)

Zimmermann mengemukakan bahwa *Multi criteria decision making* (MCDM) adalah suatu metode pengambilan keutusan untuk menetapkan alternatif terbaik dari sejumlah alternatif berdasarkan beberapa kriteria tertentu [16]. Kriteria biasanya berupa ukuran-ukuran, aturan-aturan atau standar yang digunakan dalam pengambilan keputusan. Berdasarkan tujuannya, MCDM dapat dibagi dua model: *Multi Attribute Decision Making* (MADM) dan *Multi Objective Decision Making* (MODM) [16].

Seringkali MADM dan MODM digunakan untuk menerangkan kelas atau kategori yang sama. MADM digunakan untuk menyelesaikan masalah-masalah dalam ruang diskrit. Oleh karena itu, pada MADM biasanya digunakan untuk melakukan penilaian atau seleksi terhadap beberapa alternatif dalam jumlah yang terbatas. Sedangkan MODM digunakan untuk menyelesaikan masalah-masalah pada ruang kontinyu. Secara umum dapat dikatakan bahwa, MADM menyeleksi alternatif terbaik dari sejumlah alternatif sedangkan MODM merancang alternatif terbaik [16].

2.4.1 Klasifikasi Metode MCDM

Cara dalam mengklasifikasi metode MCDM yaitu menurut jumlah pengambil keputusan yang terlibat dalam proses pengambil keputusan . MCDM dapat dibagi berdasarkan pengambil keputusan satu orang, atau pengambil keputusan dalam bentuk grup (kelompok) [16].

2.4.2 Klasifikasi Solusi MCDM

Masalah MDCM tidak selalu memberikan solusi unik, perbedaan tipe bisa jadi akan memberikan perbedaan solusi [11].

1. Solusi ideal, kriteria atau atribut dapat dibagi menjadi dua kategori,yaitu kriteria yang nilainya akan dimaksimumkan (kategori nilai keuntungan), dan kriteria yang nilainya kan diminimumkan (kategori kriteria biaya). Solusi ideal kan memaksimumkan semua kriteria keuntungan dan meminimumkan semua kriteria biaya.
2. Solusi *non-dominated*, solusi ini sering juga dikenal dengan nama solusi pareto-optimal. Solusi feasible MDCM dikatakan *nondominated* jika tidak ada solusi *feasible* yang lain akan menghasilkan perbaikan terhadap suatu atribut tanpa menyebabkan degenerasi pada atribut lainnya.
3. Solusi yang memuaskan, solusi yang memuaskan adalah himpunan bagian dari solusi-solusi *feasible* dimana setiap alternatif melampaui semua kriteria yang diharapkan.

4. Solusi yang lebih disukai, solusi yang disukai adalah solusi *nondominated* yang paling banyak memuaskan pengambil keputusan.

2.4.3 Metode MADM (Multi Atribute Decision Making)

Pada dasarnya proses *Multi Attribute Decision Making* dilakukan melalui tiga tahap, yaitu penyusunan komponen-komponen situasi, analisis, dan sintetis informasi [12]. Sebagian besar pendekatan *Multi Attribute Decision Making* dilakukan melalui dua langkah, yaitu pertama melakukan agregasi terhadap keputusan-keputusan yang tanggap terhadap semua tujuan pada setiap alternatif dan kedua melakukan perankingan alternatif-alternatif keputusan tersebut berdasarkan hasil agregasi keputusan.

Dengan demikian, bisa dikatakan bahwa masalah *multi attribute decision making* adalah mengevaluasi m alternatif Ai (i=1,2,…,m) terhadap sekumpulan atribut atau kriteria Cj (j=1,2,…,n), dimana setiap atribut saling tidak tergantung satu dengan yang lainnya. Matriks keputusan setiap alternatif terhadap setiap atribut X diberikan sebagai :

$$X = \begin{Bmatrix} x11 & x12 \dots & x1n \\ x21 & x22 \dots & x2n \\ x\ \ 1 & x\ \ 2 \dots & x \end{Bmatrix} \qquad (2.9)$$

Dimana xij merupakan rating kinerja alternatif ke-i terhadap atribut ke-j. Nilai bobot yang menunjukkan tingkat kepentingan relatif setiap atribut, diberikan sebagai:

$$W = \{w1, w2, \dots w\ \} \qquad (2.10)$$

Rating kinerja (X) dan nilai bobot (W) merupakan nilai utama yang merepresentasikan preferensi absolut dari pengambil keputusan. Masalah *multi atribut decision making* diakhiri dengan proses perankingan untuk mendapatkan alternatif terbaik diperoleh berdasarkan nilai keseluruhan preferensi yang diberikan.

2.5 Metode TOPSIS (*Technique For Order Preference by Similarity to Ideal Solution*)

TOPSIS adalah salah satu metode pengambilan keputusan multikriteria yang pertama kali diperkenalkan oleh Yoon dan Hwang tahun 1981. TOPSIS didasarkan pada konsep dimana alternatif yang terpilih atau terbaik tidak hanya mempunyai jarak terdekat (terpendek) dari solusi ideal positif, namun juga memiliki jarak terjauh (terpanjang) dari solusi ideal negatif dari sudut pandang geometris dengan menggunakan jarak *Euclidean* untuk menentukan kedekatan relatif dari suatu alternatif dengan solusi optimal. Solusi ideal positif didefinisikan sebagai jumlah dari seluruh nilai terbaik yang dapat dicapai untuk setiap atribut, sedangkan solusi negatif-ideal terdiri dari seluruh nilai terburuk yang dicapai untuk setiap atribut. [18]

TOPSIS mempertimbangkan keduanya, jarak terhadap solusi ideal positif dan jarak terhadap solusi ideal negatif dengan mengambil kedekatan relatif terhadap solusi ideal positif. Berdasarkan perbandingan terhadap jarak relatifnya, susunan prioritas alternatif bisa dicapai. Metode ini banyak digunakan pada beberapa model MADM untuk menyelesaikan masalah pengambilan keputusan secara praktis. Hal ini disebabkan konsepnya sederhana dan mudah dipahami, komputasinya efisien, dan memiliki kemampuan mengukur kinerja relatif dari alternatif-alternatif keputusan dalam bentuk matematis yang sederhana. [18]

2.5.1 Langkah-langkah Metode TOPSIS

Prosedur perhitungan metode TOPSIS dalam pendukung keputusan adalah sebagai berikut [18]:

1. Membangun sebuah matriks keputusan

   Matriks keputusan X mengacu terhadap m alternatif yang akan dievaluasi berdasarkan n kriteria. Matriks keputusan X dapat dilihat sebagai berikut:

$$X = \begin{array}{c} \\ a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_m \end{array} \begin{array}{c} \begin{array}{cccccc} x_1 & x_2 & x_3 & . & . & x_n \end{array} \\ \begin{bmatrix} x_{11} & x_{12} & x_{13} & . & . & x_{1n} \\ x_{21} & x_{22} & x_{23} & . & . & x_{2n} \\ x_{31} & x_{32} & x_{33} & . & . & x_{3n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{m1} & x_{m2} & x_{m3} & . & . & x_{mn} \end{bmatrix} \end{array} \quad (2.11)$$

Keterangan :

a_i = alternatif-alternatif yang mungkin

x_j = atribut dimana performansi alternatif diukur

x_{ij} = performansi alternatif a_i dengan acuan atribut x_j

2. Membangun matriks keputusan yang ternormalisasi. Persamaan yang digunakan untuk mentransformasikan setiap elemen x_{ij} adalah

$$r_i = \frac{x_i}{\sqrt{\sum_{i=1}^{m} x_i^2}} \quad (2.12)$$

Dengan i = 1, 2, 3, ..., m; dan j = 1, 2, 3, ..., n;

Keterangan :

r_{ij} = elemen dari matriks keputusan yang ternormalisasi R

x_{ij} = elemen dari matriks keputusan X

3. Membuat matriks keputusan yang ternormalisasi terbobot. Dengan bobot $w_j = (w_1, w_2, w_3, \ldots, w_n)$, dimana w_j adalah bobot dari kriteria ke-j dan $\sum_{j=1}^{n} w_j = 1$, maka normalisasi bobot matriks Y adalah :

$$y_i = w_j r_i \quad (2.13)$$

Dengan i= 1, 2, 3,..., m; dan j = 1, 2, 3,..., n;

Keterangan:

y_{ij} = elemen dari matriks keputusan yang ternormalisasi terbobot Y

w_j = bobot kriteria ke-j

r_{ij} = elemen dari matriks keputusan yang ternormalisasi R

4. Menentukan matriks solusi ideal dan matriks solusi ideal negatif. Solusi ideal positif dinotasikan A^+ , sedangkan solusi ideal negatif dinotasikan A^- . Berikut ini adalah persamaan dari A^+ dan A^-

   a. $A^+ = \{(\max y_i \mid j \in J), (\min y_i \mid j \in J'), i = 1,2,3 \dots m\}$
   $= (y_1^+, y_2^+, y_3^+, \dots, y_n^+)$ (2.14)

   b. $A^- = \{(\min y_i \mid j \in J), (\max y_i \mid j \in J'), i = 1,2,3 \dots m\}$
   $= (y_1^-, y_2^-, \dots, y_n^-)$ (2.15)

   Keterangan:

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

   A^+ = elemen matriks solusi ideal positif

   A^- = elemen matriks solusi ideal negatif

5. Menentukan jarak antara nilai setiap alternatif dengan matriks solusi ideal positif dan matrik ideal negatif.

   a. Jarak antara alternatif A_i dengan solusi ideal positif dirumuskan sebagai:

   $$D_i^+ = \sqrt{\sum_{j=1}^{m} (y_i^+ - y_i)^2} \quad (2.16)$$

   Keterangan :

   D_i^+ = jarak alternatif Ai dengan solusi ideal positif

   y^+ = solusi ideal positif

   y_i = matriks normalisasi terbobot

   b. Jarak antara alternatif A_i dengan solusi ideal negatif dirumuskan sebagai:

   $$D_i^- = \sqrt{\sum_{j=1}^{m} (y_i - y_i^-)^2} \quad (2.17)$$

   Keterangan :

   D_i^- = jarak alternatif Ai dengan solusi ideal negatif

   y^- = solusi ideal negatif

   y_{ij} = matriks normalisasi terbobot

6. Menghitung kedekatan terhadap solusi ideal positif. Kedekatan setiap alternatif terhadap solusi ideal dihitung berdasarkan rumus:

$$V_i = \frac{D_i^-}{D_i^- + D_i^+} \quad (2.18)$$

Keterangan :

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

7. Alternatif yang memiliki nilai kedekatan terhadap solusi ideal positif (V_i) tertinggi merupakan alternatif terbaik berdasarkan perhitungan TOPSIS.

2.6 Perancangan Berorientasi Objek

Teknologi objek menganalogikan sistem aplikasi seperti kehidupan nyata yang didominasi oleh objek. Didalam membangun sistem berorientasi objek akan menjadi lebih baik apabila langkah awalnya didahului dengan proses analisis dan perancangan yang berorientasi objek. Tujuannya adalah mempermudah programmer didalam mendesain program dalam bentuk objek-objek dan hubungan antar objek tersebut untuk kemudian dimodelkan dalam sistem nyata.Suatu perusahaan *software* yaitu *Rational Software*, telah membentuk konsarium dengan berbagai organisasi untuk meresmikan pemakaian *Unified Modelling Language* (UML) sebagai bahasa standar dalam *Object Oriented Analysist Design* (OOAD).

2.7 Unified Modelling Language (UML)

UML dalam sebuah bahasa untuk menentukan visualisasi, konstruksi, dan mendokumentasikan artifacts dari sistem *software*, untuk memodelkan bisnis, dan sistem *non-software* lainnya. UML merupakan sistem arsitektur yang bekerja dalam OOAD dengan satu bahasa yang konsisten untuk menentukan, visualisasi, konstruksi dan mendokumentasikan *artifact* yang terdapat dalam sistem. *Artifact* adalah sepotong informasi yang digunakan atau dihasilkan dalam suatu proses rekayasa *software*. *Artifact* dapat berupa model, deskripsi atau *software*. [19]

2.7.1 Use Case Diagram

Use Case Diagram menjelaskan manfaat sistem jika dilihat menurut pandangan orang yang berada diluar sistem (Aktor). Diagram ini menunjukan fungsionalitas suatu sistem yang berinteraksi dengan dunia luar. *Use Case* Diagram dapat digunakan selama proses analisis untuk memahami bagaimana sistem bekerja [20]. Komponen yang digunakan dalam *Use Case* diagram adalah sebagai berikut:

1. Aktor adalah segala sesuatu yang berinteraksi dengan *use case.* Aktor digambarkan menyerupai orang



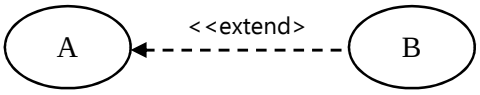
Gambar 2.1 Aktor

2. *Use Case* adalah deskripsi dari sekumpulan urutan tindakan. *Use case* digambarkan menggunakan elips.



Gambar 2.2 Use Case

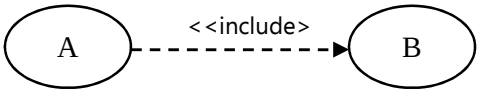
3. *Asociation* adalah garis yang menghubungkan aktor dengan *use case*. *Asociation* terdiri atas beberapa jenis:
   a. *Extend* digunakan untuk menetapkan bahwa *use case* yang dituju memperpanjang perilaku dari sumber *use case*.



Gambar 2.3 Extend

Jika use case A di-*extend* oleh *use case* B maka antara A dan B dapat diimplementasikan dan digunakan secara bebas.

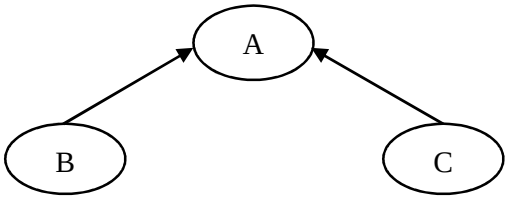
b. *Include* digunakan untuk menetapkan bahwa *use case* awal secara eksplisit menggabungkan prilaku dari *use case* yang ditunjuk oleh *use case* awal.



Gambar 2.4 Include

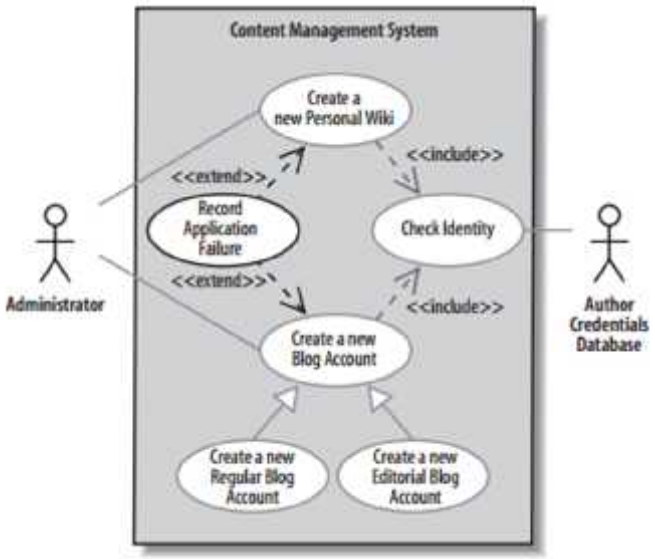
Jika use Case A meng-include *use case* B maka *use case* B akan diimplementasikan setiap kali *use case* A diimplementasikan.

c. Generalisasi digunakan untuk membuat aktor atau *use case* yang lebih spesifik dari suatu aktor atau *use case*.



Gambar 2.5 Generalisasi

Contoh pengimplementasian *use case* diagram secara keseluruhan dapat dilihat pada gambar 2.6



Gambar 2.6 *Use Case* Diagram Pembuatan Akun Blog
(Sumber : *Learning* UML 2.0, 2006)

2.7.2 Class Diagram

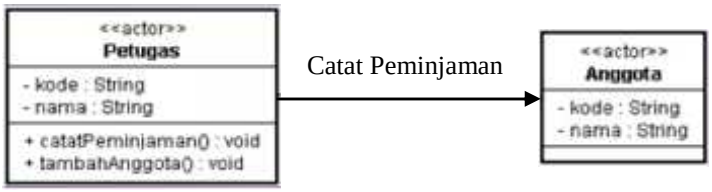
Class Diagram menjelaskan dalam visualisasi struktur kelas-kelas dari suatu sistem dan merupakan tipe diagram yang paling banyak dipakai. *Class Diagram* memperlihatakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas dalam model desain dari suatu sistem. Selama proses analisis, *class diagram* memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan prilaku sistem. Selama tahap desain, *class diagram* berperan dalam menagkap struktur dari semua kelas yang membentuk arsitektur sistem yang dibuat [20].

Diagram Class mempunyai 3 macam relationalships (hubungan), sebagai berikut :

1. Relasi *Association*

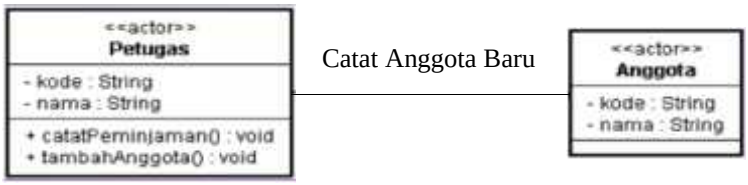
Suatu hubungan antara bagian dari dua kelas. Terjadi *association* antara dua kelas jika salah satu bagian dari kelas mengetahui yang lainnya dalam melakukan suatu kegiatan. Di dalam diagram, sebuah *association* adalah penghubung yang menghubungkan dua kelas. Relasi *association* terbagi atas dua jenis yaitu:

a. *Association* satu arah menggambarkan bahwa pesan atau urutan kejadian terjadi dari hanya salah satu kelas sedangkan kelas lain pasif.



Gambar 2.7 *Association* Satu Arah

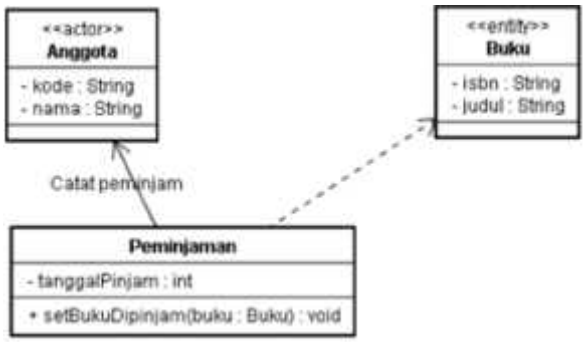
b. *Association* dua arah terjadi ketika salah satu kelas mengirimkan pesan kepada kelas yang lain kemudian kelas yang lain mengirimkan pesan kepada kelas yang mengirimnya pesan.



Gambar 2.8 *Association* Dua Arah

2. Relasi *Depedency*

   Relasi jenis ini menunjukkan bahwa sebuah kelas mengacu kepada kelas lainnya. Oleh sebab itu perubahan pada kelas yang diacu akan sangat berpengaruh pada kelas yang mengacu.



Gambar 2.9 Relasi *Depedency*

3. Relasi *Agregation*

   Suatu bentuk relasi yang jauh lebih kuat daripada asosiasi. *Aggregation* dapat diartikan bahwa suatu kelas merupakan bagian dari kelas yang lain namun bersifat tidak wajib.



Gambar 2.10 Relasi Aggregation

4. Relasi *Compositon*

   Merupakan relasi yang paling kuat dibandingkan dengan *assosiation* dan *aggregation*. Pada komposisi diartikan bahwa suatu kelas merupakan bagian yang wajib dari kelas yang lain.

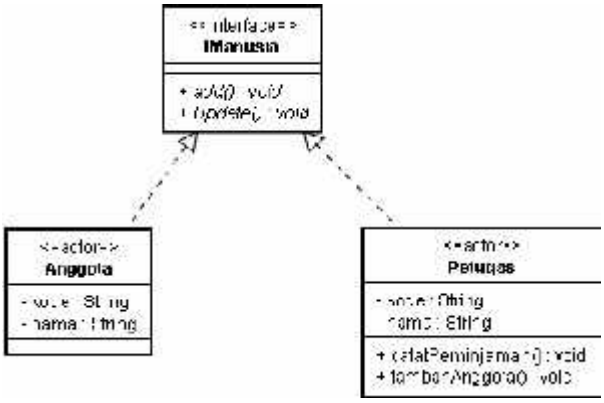


Gambar 2.11 Relasi *Compositon*

5. Relasi *Realization*

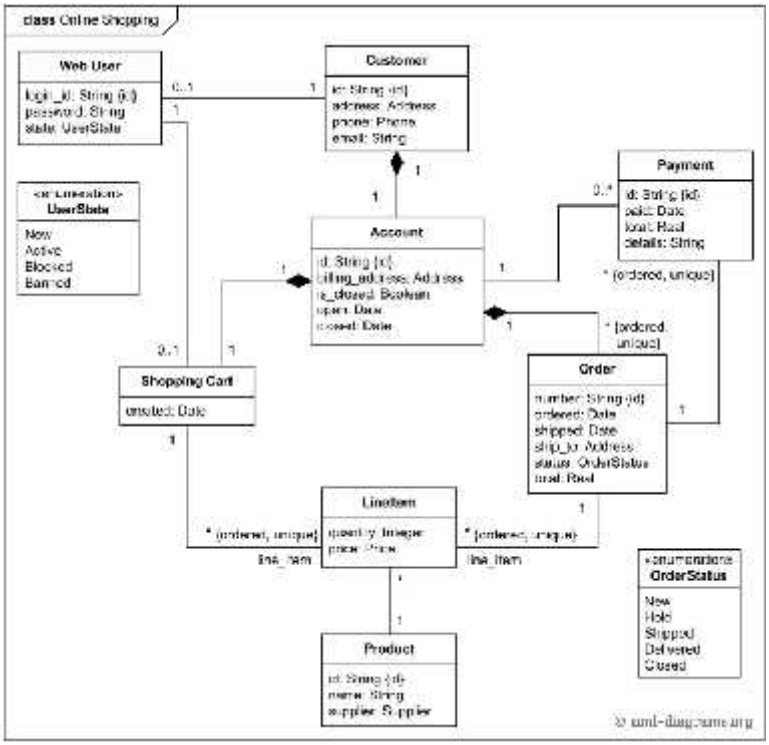
   bisa disebut juga implementasi merupakan suatu relasi yang menunjukkan penerapan terhadap suatu *interface* kepada sebuah *Class*. Relasi *realization*

biasanya digunakan untuk mewajibkan suatu kelas memiliki suatu *Method* yang sudah didefinisikan bentuk kerangkanya dalam suatu *interface*.



Gambar 2.12 Relasi Realization

Contoh pembuatan *Class* diagram pada kasus transaksi belanja *online* dapat dilihat pada gambar 2.13



Gambar 2.13 Class Diagram Belanja *Online*
(Sumber: http://www.uml-diagrams.org/class-diagrams-examples.html)

2.7.3 Behaviour Diagram

Behavior diagram dapat dikelompokan menjadi tiga diagram, yaitu [20] :

1. *Statechart Diagram*

   Statechart Diagram berfungsi untuk memodelkan prilaku dinamis satu kelas satu objek.

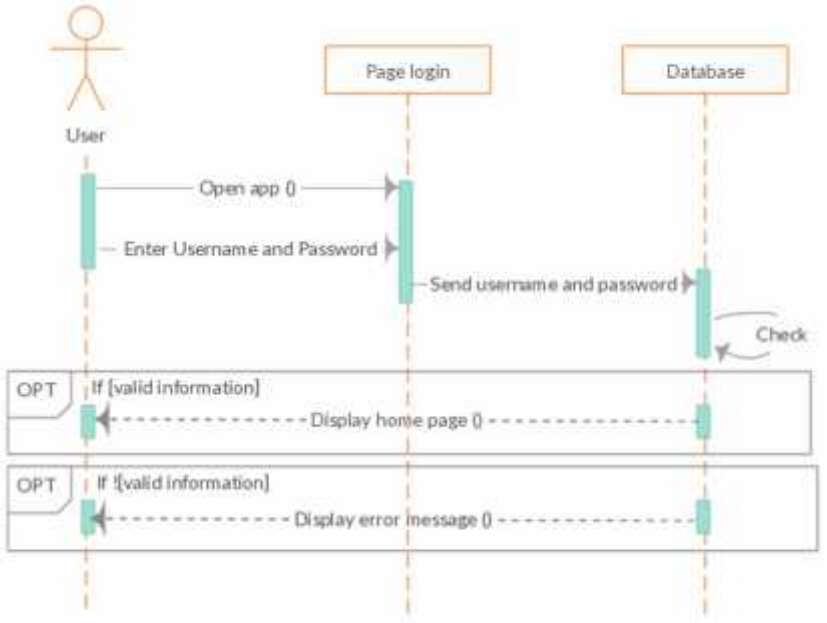
2. *Activity Diagram*

   Activity Diagram memodelkan alur kerja (work flow) sebuah proses bisnis dan urutan aktifitas dalam suatu proses.

3. *Interaction Diagram*

   Interaction Diagram dibagi menjadi dua model diagram yaitu :

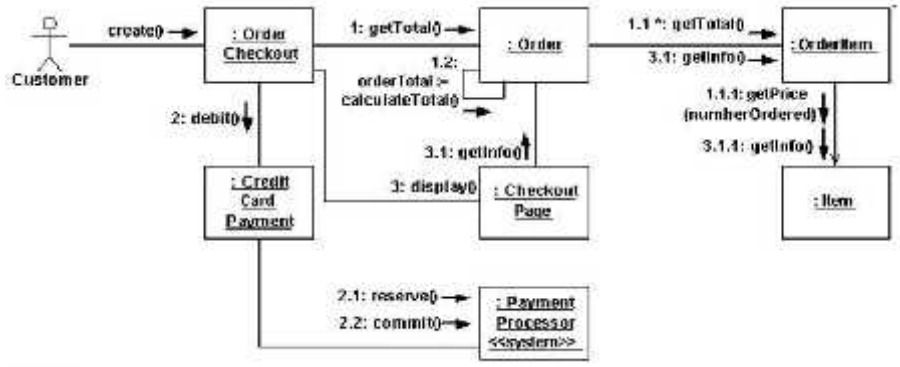
   a. *Sequence Diagram* menjelaskan interaksi objek yang disusun dalam suatu urutan waktu. Diagram ini secara khusus bersosialisasi dengan use case. Sequence diagram, memperlihatkan tahap demi tahap apa yang seharusnya terjadi untuk menghasilkan sesuatu dalam *use case.* Contoh pembuatan sequence diagram pada kasus *login* dapat dilihat pada gambar 2.14



Gambar 2.14 Sequence Diagram
(Sumber : http://creately.com/diagram/example/iggvfnqj2/login+)

   b. *Collaboration Diagram* melihat pada interaksi dan hubungan terstruktur antar objek. Tipe diagram ini menekankan pada hubungan (*relationship*) antar objek, sedangkan *sequence diagram* menekankan pada urutan kejadian. Dalam

collaboration diagram terdapat beberapa objek, link, dan *message*. Gambar 2.15 menunjukan *collaboration* diagram untuk kasus belanja *Online*.



Gambar 2.15 Collaboration Diagram
(Sumber : http://brailleatm.com/)

2.8 Perangkat Lunak Pendukung

Adapun perangkat lunak yang akan digunakan dalam pembangunan *prototype* pada penelitian ini adalah sebagai berikut:

2.8.1 Java

Java adalah bahasa pemrograman tingkat tinggi yang berorientasi objek dan program java tersusun dari bagian yang disebut kelas. Kelas terdiri atas metode-metode yang melakukan pekerjaan dan mengembalikan informasi setelah melakukan tugasnya. Para pemrogram Java banyak mengambil keuntungan dari kumpulan kelas di pustaka kelas Java, yang disebut dengann(API). Kelaskelas ini diorganisasikan menjadi sekelompok yang disebut paket Java API telah menyediakan fungsionalitas yang memadai untuk menciptakan dan aplikasi canggih. Jadi ada dua hal yang harus dipelajari dalam Java, yaitu mempelajari bahasa Java dan bagaimana mempergunakan kelas pada Java API. Kelas merupakan satu-satunya cara menyatakan bagian eksekusi program, tidak ada cara lain. Pada Java program javac untuk mengkompilasi file kode sumber Java menjadi kelas-kelas. File kode sumber mempunyai ekstensi *.java. Kompilator javac menghasilkan filekelas dengan ekstensi *.class. Interpreter merupakan modul utama sistem Java yang digunakan aplikasi Java dan menjalankan program Java.

Beberapa keunggulan java yaitu java merupakan bahasa yang sederhana. Java dirancang agar mudah dipelajari dan digunakan secara efektif. Java tidak menyediakan fitur-fitur rumit bahasa pemrograman tingkat tinggi, serta banyak pekerjaan pemrograman yang mulanya harus dilakukan manual, sekarang digantikan dikerjakan Java secara otomatis seperti dealokasi memori. Bagi pemrogram yang sudah mengenal bahasa C++ akan cepat belajar susunan bahasa Java namun harus waspada karena mungkin Java mengambil arah (semantiks) yang berbeda dibanding C++ [21].

2.8.2 Netbeans IDE

NetBeans adalah *Integrated Development Environment* (IDE) berbasiskan Java dari Sun Microsystems yang berjalan di atas Swing. Swing sebuah teknologi Java untuk pengembangan aplikasi Desktop yang dapat bejalan di berbagai macam platforms seperti Windows, Linux, Mac OS X and Solaris. Netbeans merupakan software development yang Open Source, dengan kata lain software ini di bawah pengembangan bersama, bebas biaya NetBeans merupakan sebuah proyek kode terbuka yang sukses dengan pengguna yang sangat luas, komunitas yang terus tumbuh, dan memiliki hampir 100 mitra. Sun Microsystems mendirikan proyek kode terbuka NetBeans pada bulan Juni 2000 dan terus menjadi sponsor utama.

Suatu IDE adalah lingkup pemrograman yang diintegrasikan kedalam suatu aplikasi perangkat lunak yang menyediakan pembangun *Graphic User Interface* (GUI), suatu text atau kode editor, suatu compiler atau interpreter dan suatu debugger. The NetBeans IDE adalah sebuah lingkungan pengembangan - sebuah kakas untuk pemrogram menulis, mengompilasi, mencari kesalahan dan menyebarkan program. Netbeans IDE ditulis dalam Java - namun dapat mendukung bahasa pemrograman lain. Terdapat banyak modul untuk memperluas Netbeans IDE. Netbeans IDE adalah sebuah produk bebas dengan tanpa batasan bagaimana digunakan. NetBeans IDE mendukung pengembangan semua tipe aplikasi Java (J2SE, web, EJB, dan aplikasi mobile). Fitur lainnya adalah sistem proyek berbasis Ant, kontrol versi, dan refactoring [21].

2.8.3 MySQL

MySQL merupakan salah satu database server yang berkembang di lingkungan open source dan didistribusikan secara free (gratis) dibawah lisensi GPL. MySQL merupakan RDBMS (*Relational Database Management System*) server. RDBMS adalah program yang memungkinkan pengguna database untuk membuat, mengelola, dan menggunakan data pada suatu model relational. Dengan demikian, tabel-tabel yang ada pada database memiliki relasi antara satu tabel dengan tabel lainnya. Beberapa keunggulan dari MySQL yaitu [22]:

1. Cepat, handal dan mudah dalam penggunaannya MySQL lebih cepat tiga sampai empat kali dari pada database server komersial yang beredar saat ini, mudah diatur dan tidak memerlukan seseorang yang ahli untuk mengatur administrasi pemasangan MySQL.
2. Didukung oleh berbagai bahasa database server MySQL dapat memberikan pesan error dalam berbagai bahasa seperti Belanda, Portugis, Spanyol, Inggris, Perancis, Jerman, dan Italia.
3. Mampu membuat tabel berukuran sangat besar ukuran maksimal dari setiap tabel yang dapat dibuat dengan MySQL adalah 4 GB sampai dengan ukuran file yang dapat ditangani oleh sistem operasi yang dipakai.
4. Lebih Murah, MySQL bersifat open source dan didistribusikan dengan gratis tanpa biaya untuk UNIX platform, OS/2 dan Windows platform.

2.9 SQL (*Structure Query Language*)

SQL adalah sebuah bahasa permintaan database yang terstruktur. Bahasa SQL dibuat sebagai bahasa yang dapat merelasikan beberapa tabel dalam database maupun merelasikan antar database. Bahasa SQL ditulis langsung dalam sebuah program databases sehingga pengguna dapat melihat langsung permintaan yang diinginkan sekaligus melihat hasilnya. SQL dibagi menjadi 2 bentuk query, yaitu [22].

1. DDL (*Data Definition Language*)

   DDL adalah sebuah metode *query* sql yang berguna untuk mendefinisikan data pada sebuah database, adapun *query* yang dimiliki adalah

   a. CREATE : Digunakan untuk melakukan pembuatan tabel dan database
   b. DROP : Digunakan untuk melakukan penghapusan tabel maupun database
   c. ALTER : Digunakan untuk melakukan perubahan struktur tabel yang telah dibuat, baik menambah *field* (*add*), mengganti nama *field* (*change*) ataupun menamakannya kembali (*rename*) serta menghapus (*drop*)

2. DML (*Data Manipulation Language*)

   DML adalah sebuah metode *query* yang dapat digunakan apabila DDL telah terjadi, sehingga fungsi dari *query* ini adalah untuk melaukan pemanipulasian *database* yang telah ada atau telah dibuat sebelumnya. Adapun *query* yang termasuk di dalamnya adalah:

   a. INSERT : Digunakan untuk melakukan penginputan pada tabel didatabase
   b. UPDATE : Digunakan untuk melakukan pengubahan atau peremajaan terhadap data yang ada pada tabel
   c. DELETE : Digunakan untuk melakukan penghapusan data pada tabel. Penghapusan ini dapat dilakukan secara sekaligus (seluruh isi tabel) maupun hanya beberapa *Recordset*.