BAB II

LANDASAN TEORI

2.1. Pengambilan Keputusan

Sistem pendukung keputusan atau *decision support systems* adalah bagian dari sistem informasi berbasis komputer (termasuk sistem berbasis pengetahuan (manajemen pengetahuan) yang dipakai untuk mendukung pengambilan keputusan dalam suatu organisasi atau perusahaan. Dapat juga dikatakan sebagai sistem komputer yang mengolah data menjadi informasi untuk mengambil keputusan dari masalah semi-terstruktur yang spesifik. Sistem Pendukung Keputusan (*Decision Support System*) membantu pengambil keputusan memilih berbagai alternatif keputusan yang merupakan hasil pengolahan informasi-informasi yang diperoleh/tersedia dengan menggunakan model-model pengambilan keputusan.

(Holsapple, 2009) mendeskripsikan SPK sebagai teknologi mendapatkan pengetahuan bagi pengambil keputusan secara tepat, pada waktu yang tepat dalam representasi yang tepat dengan biaya yang tepat. Perkembangan teknik-teknik pemrosesan informasi dan teknologi digital dalam mendukung kegiatan penyelesaian masalah dan pengambilan keputusan makin mendorong munculnya sistem pendukung keputusan yang cerdas. Empat tahapan dalam pembuatan keputusan adalah : *Intelligenc*e (menemukan apa yang harus diperbaiki); *Design* (menemukan pilihan perbaikan); *Choice* (mengambil keputusan pilihan perbaikan); dan *Implementation* (mengimplementasikan perbaikan).

2.2. *Technique for Order Preference by Similarty to Ideal Solution* (TOPSIS)

TOPSIS diperkenalkan pertama kali oleh Yoon dan Hwang pada tahun 1981 untuk digunakan sebagai salah satu metode dalam memecahkan masalah multikriteria (Sachdeva, 2009). TOPSIS memberikan sebuah solusi dari sejumlah alternatif yang mungkin dengan cara membandingkan setiap alternatif dengan alternatif terbaik dan alternatif terburuk yang ada diantara alternatif-alternatif masalah. Metode ini menggunakan jarak untuk melakukan perbandingan tersebut. TOPSIS telah digunakan dalam banyak aplikasi termasuk keputusan investasi keuangan, perbandingan performansi dari perusahaan, perbandingan performansi dalam suatu industri khusus, pemilihan sistem operasi, evaluasi pelanggan, dan perancangan robot.

Yoon dan Hwang mengembangkan metode TOPSIS berdasarkan intuisi yaitu alternatif pilihan merupakan alternatif yang mempunyai jarak terkecil dari solusi ideal positif dan jarak terbesar dari solusi ideal negatif dari sudut pandang geometris dengan menggunakan jarak *Euclidean* (Sachdeva, 2009). Namun, alternatif yang mempunyai jarak terkecil dari solusi ideal positif, tidak harus mempunyai jarak terbesar dari solusi ideal negatif. Maka dari itu, TOPSIS mempertimbangkan keduanya, jarak terhadap solusi ideal positif dan jarak terhadap solusi ideal negatif secara bersamaan. Solusi optimal dalam metode TOPSIS didapat dengan menentukan kedekatan relatif suatu alternatif terhadap solusi ideal positif. TOPSIS akan merangking alternative berdasarkan prioritas nilai kedekatan relatif suatu alternatif terhadap solusi ideal positif. Alternatif-alternatif yang telah dirangking kemudian dijadikan

sebagai referensi bagi pengambil keputusan untuk memilih solusi terbaik yang diinginkan.

Metode ini banyak digunakan untuk menyelesaikan pengambilan keputusan secara praktis. Hal ini disebabkan metode Topsis memiliki keunggulan yaitu konsepnya sederhana dan mudah dipahami, komputasinya efisien, dan memiliki kemampuan mengukur kinerja relatif dari alternatif-alternatif keputusan. Topsis memiliki karakteristik yaitu bahwa alternatif yang dipilih harus memiliki jarak geometris terpendek dari solusi ideal positif dan jarak geometris terpanjang dari solusi ideal negatif. Ini adalah agregasi kompensasi yang membandingkan satu set alternatif dengan mengidentifikasi bobot untuk setiap kriteria, normalisasi bobot untuk setiap kriteria dan menghitung jarak geometris antara masing-masing alternatif dan alternatif yang ideal, yang merupakan nilai terbaik di setiap criteria (1).

Langkah-langkah dari metode dimulai dengan membangun sebuah matriks keputusan.

$$D = \begin{bmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{bmatrix} \quad \text{.....(1)}$$

Dimana:

D = matriks

m = alternatif

n = kriteria

x_{ij} = alternatif ke – i dan kriteria ke – j

2.2.1. Menghitung Matriks Ternormalisasi

Setiap elemen pada matriks *D* dinormalisasikan untuk mendapatkan matriks normalisasi *R*. Setiap normalisasi dari nilai r_{ij} (2) dapat dilakukan dengan perhitungan sebagai berikut:

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{m} x^2_{ij}}} \quad \text{.....(2)}$$

Untuk i =1,2,3,…,m;

j =1,2,3,…,n

2.2.2. Menghitung Matriks Ternormalisasi Terbobot

Persamaan digunakan untuk menghitung matriks ternormalisasi terbobot (4), maka harus ditentukan terlebih dahulu nilai bobot yang akan mempresentasikan prefensi absolute dari pengambilan keputusan. Nilai bobot prefensi menunjukan tingkat kepentingan relative setiap keriteria atau sub keriteria pada persamaan (3).

$$W = \{W_1, W_2, W_3, \ldots, W_n\} \quad \text{.....(3)}$$

$$\gamma = W_i \, \gamma_{ij} \quad \text{.....(4)}$$

2.2.3. Menentukan Solusi Ideal Positif Dan Ideal Negatif

Solusi ideal positif (5) dinotasikan dengan A^+dan solusi ideal negatif (6) dinotasikan dengan A^-sebagi berikut :

Menentukan Solusi Ideal (+) dan Solusi Ideal (-)

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

Dimana :

γ_{ij} = elemen matriks ɣ baris ke-*i* dan kolom ke- *j*

J ={j=1,2,3,…,n dan j berhubung dengan *benefit criteria*}

J' ={j=1,2,3,…,n dan j berhubung dengan *cost criteria*}

2.2.4. Menghitung Jarak Antara Nilai Setiap Alternatif Dengan Matriks Solusi Ideal Positif Dan Solusi Ideal Negatif

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

untuk solusi ideal positif :

$$D_i^+ = \sqrt{\sum_{j=1}^{n} (\gamma_{ij} - \gamma_j^+)^2} \quad \text{.....(7)}$$

dengan *i* =1,2,3,…,n untuk solusi ideal negative :

$$D_i^- = \sqrt{\sum_{j=1}^{n} (\gamma_{ij} - \gamma_j^-)^2} \quad \text{.....(8)}$$

dengan *i* =1,2,3,…,n

2.2.5. Menghitung Nilai Preferensi Untuk Setiap Alternatif

Kedekatan relative dari alternatif D^+ dengan solusi ideal D^- direpresentasikan dengan:

$$V_i = \frac{D_i^-}{D_i^- + D_i^+}, \quad \text{.....(10)}$$

dengan $0 < V_i < 1$ dan *i* = 1,2,3…..m

2.3.MySQL

Menurut (Kustiyaningsih, 2010) Basis data adalah sekumpulan informasi yang diatur agar mudah dicari. Dalam arti umum basis data adalah sekumpulan data yang diproses dengan bantuan komputer yang memungkinkan data dapat diakses dengan mudah dan tepat, yang dapat digambarkan sebagai aktivitas dari satu atau lebih organisasi yang berelasi.

MySQL merupakan suatu database. MySQL dapat juga dikatakan sebagai database yang sangat cocok bila dipadukan dengan PHP. Secara umum, database berfungsi sebagai tempat atau wadah untuk menyimpan. Mengklarifikasikan data secara profesional. MySQL bekerja menggunakan SQL Language (Structure Query Language). Itu dapat diartikan bahwa MySQL merupakan standar penggunaan database di dunia untuk pengolahan data.

MySQL termasuk jenis RDBMS (Relational Database Management System). Sedangkan RDBMS sendiri akan lebih banyak mengenal istilah seperti tabel, baris, dan kolom digunakan dalam perintah-perintah di MySQL. MySQL merupakan sebuah basis data yang mengandung satu atau sejumlah tabel. Tabel terdiri dari atas sejumlah baris dan setiap baris mengandung satu atau sejumlah tabel. Tabel ini terdiri atas sejumlah bari dan setiap baris mengandung satu atau beberapa kolom. Di dalam PHP telah menyediakan fungsi untuk koneksi ke basis data dengan sejumlah fungsi untuk pengaturan baik menghubungkan maupun memutuskan koneksi dengan server database MySQL sebagai sarana untuk mengumpulkan informasi.

Pada umumnya, perintah yang paling sering digunakan dalam MySQL adalah *Select* (mengambil), *insert* (menambah), *update* (mengubah), dan *delete* (menghapus). Selain itu, SQL juga menyediakan perintah untuk membuat database, field, ataupun index guna menambah atau menghapus data.

2.4. *Pretext Hyper Processor* (PHP)

Menurut (Saputra, 2011) PHP atau yang memiliki kepanjangan PHP *Hypertext Preprocessor* merupakan suatu bahasa pemrograman yang difungsikan untuk membangun suatu website dinamis. PHP menyatu dengan kode HTML, maksudnya adalah beda kondisi. HTML digunakan sebagai pembangun atau pondasi dari kerangka layout web, sedangkan PHP difungsikan sebagai prosesnya sehingga dengan adanya PHP tersebut, web akan sangat mudah di-*maintenance*.

PHP berjalan pada sisi server sehingga PHP disebut juga sebagai bahasa *Server Side Scripting*. Artinya bahwa dalam setiap/untuk menjalankan PHP, wajib adanya web server. PHP ini bersifat open source sehingga dapat dipakai secara cuma-cuma dan mampu lintas platform, yaitu dapat berjalan pada sistem operasi Windows maupun Linux. PHP juga dibangun sebagai modul pada web server apache dan sebagai binary yang dapat berjalan sebagai CGI.

2.5. HTML

HTML kependekan dari *HyperText Markup Language*. Dokumen HTML adalah file teks murni yang dapat dibuat dengan editor teks sembarang. Dokumen ini dikenal sebagai *web page*. Dokumen HTML

merupakan dokumen yang disajikan dalam *browser web surfer*. Dokumen ini umumnya berisi informasi atau interface aplikasi di dalam Internet. Ada dua cara untuk membuat sebuah *web page*: dengan HTML editor atau dengan editor teks biasa (misalnya notepad) (Pohan. 2009).

Bermula dari sebuah bahasa yang sebelumnya banyak digunakan di dunia penerbitan dan percetakan yang disebut dengan SGML (*Standard Generalized Markup Language*), HTML adalah sebuah standar yang digunakan secara luas untuk menampilkan halaman web. HTML saat ini merupakan standar Internet yang didefinisikan dan dikendalikan penggunaannya oleh *World Wide Web Consortium* (W3C). Versi terakhir dari HTML adalah HTML 5.01, meskipun saat ini telah berkembang XHTML yang merupakan pengembangan dari HTML.

HTMLberupa kode-kode tag yang menginstruksikan browser untuk menghasilkan tampilan sesuai dengan yang diinginkan. Sebuah file yang merupakan file HTML dapat dibuka dengan menggunakan *web browser* seperti *Mozilla Firefox* atau *Microsoft Internet Explorer*. HTML juga dapat dikenali oleh aplikasi pembuka email ataupun dari PDA dan program lain yang memiliki kemampuan *browser*.

2.6. CSS

CSS merupakan singkatan dari *Cascading Style Sheet.* Kegunaannya adalah untuk mengatur tampilan dokumen HTML, contohnya seperti pengaturan jarak antar baris, teks, warna dan format border bahkan penampilan file gambar. CSS dikembangkan oleh W3C, organisasi yang mengembangkan teknologi internet. Tujuannya tak lain untuk mempermudah

proses penataan web. Perlu diingat, CSS hanyalah berupa kumpulan script yang tujuannya bukan untuk mengantikan HTML, melainkan pelengkap agar dokumen HTML bisa tampil lebih cantik dan dinamis.

Sejak ditemukannya CSS pada awal dekade 90an, CSS terus dikembangkan dan diserap oleh *web developer*. Hingga sekarang telah mencapai versi ke-3. Kode CSS bersifat lintas *platform*, yang berarti script ini dapat dibaca oleh berbagai macam sistem operasi dan browser. Hanya saja browser seperti Internet Explorer, seringkali salah mengartikan script CSS yang menyebabkan ketidaksempurnaannya tampilan dokumen HTML. Script CSS perlu dioptimalkan agar tampil maksimal pada *browser internet explorer* (Jayan, 2010).

2.7. UML

Unified Modeling Language (UML) merupakan alat merancang perangkat lunak, sarana komunikasi antara perangkat lunak dengan, menjabarkan sistem secara rinci untuk analisa dan mencari apa yang diperlukan sistem, mendokumentasikan sistem yang ada, proses-proses dan organisasinya (Herlawati, 2011).

Unified Modeling Language (UML) adalah salah satu bentuk language atau bahasa, menurut pencetusnya UML di definisikan sebagai bahasa visual untuk menjelaskan, memberikan spesifikasi, merancang, membuat model, dan mendokumentasikan aspek-aspek dari sebuah sistem.

Salah satu cara untuk mengatur diagram UML adalah dengan menggunakan *view*. *View* adalah kumpulan diagram yang menggambarkan

aspek yang sama dari proyek. View mempunyai 3 pelengkap, yaitu *Static View*, *Dynamic View*, dan *Functional View*.

2.7.1. *Static View*

Static View termasuk diagram yang memberikan gambaran dari unsur-unsur dari sistem tetapi tidak memberitahu bagaimana elemen akan berperilaku. Hal ini sangat mirip *Blueprint*. *Blueprint* itu komprehensif, tetapi mereka hanya menunjukkan apa yang tetap diam, maka disebut *Static View*. *Static View* dibentuk oleh dua diagram, yaitu *Class Diagram* dan *Object Diagram*.

2.7.2. *Dynamic View*

Pada *Dynamic View* meliputi diagram yang mengungkapkan bagaimana benda berinteraksi dengan satu sama lain dalam respon terhadap lingkungan. Ini termasuk *Sequence Diagram* dan *Collaboration Diagram*, yang kolektif disebut sebagai diagram interaksi. Mereka secara khusus dirancang untuk menjelaskan bagaimana benda berbicara satu sama lain. Ini juga mencakup *Statechart Diagram*, yang menunjukkan bagaimana dan mengapa perubahan objek dari waktu ke waktu dalam menanggapi lingkungan.

2.7.3. *Functional View*

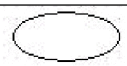
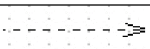
Functional View terbentuk oleh *Use Case Diagram* dan *Activity Diagram*.

2.7.4. *Use Case Diagram*

Use case diagram 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. *Use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah

sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Syarat penamaan pada *use case* adalah nama didefinisikaan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut aktor dan *use case*. Simbol-simbol yang ada pada diagram *use case* :

Tabel 2.7.4. Simbol *Use Case Diagram* (Shalahuddin dan Rosa, 2011)

Simbol	Keterangan
	*Actor* : Sebuah peran yang dimainkan oleh seseorang, sistem, atau perangkat yang memiliki saham dalam keberhasilan operasi dari sistem.
	*Use Case* : Untuk mengungkapkan tujuan bahwa sistem harus dicapai.
	Association : Mengidentifikasi interaksi antara aktor dan *Use Case*
	*Dependency* : Mengidentifikasi hubungan komunikasi antara dua *Use Case*

2.7.5. Activity Diagram

Diagram aktivitas atau *activity diagram* menggambarkan *workflow* atau aktivitas dari sebuah sistem atau proses bisnis. Yang perlu diperhatikan disini bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan 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 pengelompokkan 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.

Berikut simbol-simbol yang ada pada diagram aktivitas :

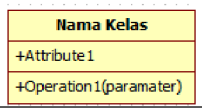
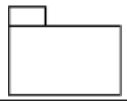
Tabel 2.7.5. Simbol *Activity Diagram* (Shalahuddin dan Rosa , 2011)

Simbol	Keterangan
	Titik Awal
	Titik Akhir
	Activity
	Pilihan untuk pengambilan keputusan
	Fork : Untuk menunjukkan kekuatan yang dilakukan secara parallel
	Rake : Menunjukkan adanya dekomposisi
	Tanda Waktu
	Tanda Penerimaan
	Aliran Akhir (Flow Final)

2.7.6. Class Diagram

Kelas Diagram terdiri dari tiga kompartemen (ruang persegi panjang) yang mengandung informasi yang berbeda diperlukan untuk menjelaskan sifat-sifat satu jenis objek. Simbol-simbol yang digunakan dalam class diagram. Simbol-simbol yang digunakan dalam *class diagram :*

Tabel 2.7.6. Simbol *Class Diagram* (Rosa, 2011)

Simbol	Deskripsi
Kelas 	Kelas pada struktur system
Paket/*package* 	Paket/*package* merupakan sebuah bungkusan dari satu atau lebih kelas (kumpulan kelas)
Assosiasi	Asosiasi merupakan hubungan antar kelas dengan makna umum, asosiasi biasanya jua disertai dengan *multiplicity*
Generalisasi 	Generalisasi merupakan hubungan generalsasi dan spesialisasi (umum khusus) antara dua kelas dimana fungsi yang satu adalah fungsi yang lebih umum dari fungsi yang lainnya
Dependency 	Dependency merupakan hubungan antarkelas yang saling bergantung, membutuhkan satu sama lain.
Agregasi 	Agregasi merupakan hubungan antar kelas dimana satu kelas merupakan semua bagian dari kelas-kelas yang lain.

2.7.7. *Sequence Diagram*

Sequence Diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* 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.

Banyaknya diagram sekuen yang harus digambar adalah sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada 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 *Sequence Diagram*:

Tabel 2.7.7. Simbol *Sequence Diagram* (Shalahuddin dan Rosa, 2011).

Nama	Simbol	Keterangan
Objek dan kelas		Objects; mewakili peserta
Lifeline		*Lifeline* merupakan siklus hidup sebuah pesan/message berdasarkan waktu
Message		Pesan/*message* merupakan komunikasi antara objek yang satu dengan objek yang lainnya
Return message		Return *message* merupakan balasan/hasil yang berisi nilai dari sebuah objek yang meminta (mengirim pesan)
Self message		*Self message* merupakan pesan dari sebuah objek kepada objek itu sendiri untuk melakukan suatu aksi
Return self message		Balasan/hasil dari *self message* yang berisi suatu nilai kepada objek itu sendiri