BAB II

LANDASAN TEORI

II.1. Sistem

Sistem berasal dari bahasa latin (*systema*) dan bahasa Yunani (*sustema*) adalah suatu kesatuan yang terdiri komponen atau elemen yang dihubungkan bersama untuk memudahkan aliran informasi, materi atau energi. Istilah ini sering dipergunakan untuk menggambarkan suatu set entitas yang berinteraksi, di mana suatu model matematika seringkali bisa dibuat.

Sistem juga merupakan kesatuan bagian - bagian yang saling berhubungan yang berada dalam suatu wilayah serta memiliki bagian - bagian penggerak, contoh umum misalnya seperti negara. Negara merupakan suatu kumpulan dari beberapa elemen kesatuan lain seperti provinsi yang saling berhubungan sehingga membentuk suatu negara dimana yang berperan sebagai penggeraknya yaitu rakyat yang berada dinegara tersebut.

Kata "sistem" banyak sekali digunakan dalam percakapan sehari - hari, dalam forum diskusi maupun dokumen ilmiah. Kata ini digunakan untuk banyak hal, dan pada banyak bidang pula, sehingga maknanya menjadi beragam. Dalam pengertian yang paling umum, sebuah sistem adalah sekumpulan benda yang memiliki hubungan di antara mereka.

Ada beberapa elemen yang membentuk sebuah sistem yaitu :

1. Tujuan.

   Setiap sistem memiliki tujuan (*Goal)*, entah hanya satu atau mungkin banyak. Tujuan inilah yang menjadi pemotivasi yang mengarahkan sistem. Tanpa tujuan, sistem menjadi tak terarah dan tak terkendali. Tentu saja, tujuan antara satu sistem dengan sistem yang lain berbeda.

2. Masukan.

   Masukan (input) sistem adalah segala sesuatu yang masuk ke dalam sistem dan selanjutnya menjadi bahan yang diproses. Masukan dapat berupa hal - hal yang berwujud (tampak secara fisik) maupun yang tidak tampak. Contoh masukan yang berwujud adalah bahan mentah, sedangkan contoh yang tidak berwujud adalah informasi (misalnya permintaan jasa pelanggan).

3. Proses.

   Proses merupakan bagian yang melakukan perubahan atau transformasi dari masukan menjadi keluaran yang berguna dan lebih bernilai, misalnya berupa informasi dan produk, tetapi juga bisa berupa hal - hal yang tidak berguna, misalnya saja sisa pembuangan atau limbah. Pada pabrik kimia, proses dapat berupa bahan mentah. Pada rumah sakit, proses dapat berupa aktivitas pembedahan pasien.

4. Keluaran.

   Keluaran (*output)* merupakan hasil dari pemrosesan. Pada sistem informasi, keluaran bisa berupa suatu informasi, saran, cetakan laporan, dan sebagainya.

5. Batas.

   Yang disebut batas (*boundary)* sistem adalah pemisah antara sistem dan daerah di luar sistem (lingkungan). Batas sistem menentukan konfigurasi, ruang lingkup, atau kemampuan sistem. Misalnya pertumbuhan sebuah toko kelontong dipengaruhi oleh pembelian pelanggan, gerakan pesaing dan keterbatasan dana dari bank. Tentu saja batas sebuah sistem dapat dikurangi atau dimodifikasi sehingga akan mengubah perilaku sistem. Sebagai contoh, dengan menjual saham ke publik, sebuah perusahaan dapat mengurangi keterbasatan dana.

6. Lingkungan.

   Lingkungan adalah segala sesuatu yang berada diluar sistem. Lingkungan bisa berpengaruh terhadap operasi sistem dalam arti bisa merugikan atau menguntungkan sistem itu sendiri. Lingkungan yang merugikan tentu saja harus ditahan dan dikendalikan supaya tidak mengganggu kelangsungan operasi sistem, sedangkan yang menguntungkan tetap harus terus dijaga, karena akan memacu terhadap kelangsungan hidup sistem.

Pada prinsipnya, setiap sistem selalu terdiri atas tiga elemen :

1. Objek, yang dapat berupa bagian, elemen, ataupun variabel. Ia dapat benda fisik, abstrak, ataupun keduanya, tergantung kepada sifat sistem tersebut.
2. Atribut, yang menentukan kualitas atau sifat kepemilikan sistem dan objeknya.
3. Hubungan internal, di antara objek - objek di dalamnya. (Janner Simamarta ; 2007)

II.2. Sistem Pendukung Keputusan

II.2.1. Pengertian Sistem Pendukung Keputusan

Pada umumnya para penulis sependapat bahwa kata keputusan (*decision*) berarti pilihan (*choice*), yaitu pilihan dari dua atau lebih kemungkinan. Keputusan dapat dilihat pada kaitannya dengan proses, yaitu bahwa suatu keputusan ialah keadaan akhir dari suatu proses yang lebih dinamis yang disebut pengambilan keputusan. Dengan kata lain, keputusan merupakan sebuah definisi *Decision Support System* atau Sistem Pendukung Keputusan atau yang selanjutnya kita singkat dalam skripsi ini menjadi SPK. (Kusrini ; 2007)

II.2.2. Pengertian Pengambilan Keputusan

Pengambilan keputusan didalam suatu organisasi merupakan hasil suatu proses komunikasi dan partisipasi yang terus menerus dari keseluruhan organisasi. Persoalan pengambilan keputusan, pada dasarnya adalah bentuk pemilihan dari

berbagai alternatif tindakan yang mungkin dipilih yang prosesnya melalui mekanisme tertentu, dengan harapan akan menghasilkan sebuah keputusan yang terbaik. Penyusunan model keputusan adalah suatu cara untuk mengembangkan hubungan - hubungan logis yang mendasari persoalan keputusan ke dalam suatu model matematis, yang mencerminkan hubungan diantara faktor - faktor yang terlibat. (Kusrini ; 2007)

II.2.3. Konsep Dasar Sistem Pendukung Keputusan

Konsep adalah suatu abstraksi yang menggambarkan cirri - ciri umum sekelompok objek, peristiwa atau fenomena lainnya. *Woodruff* (Kusrini, 2007), mendefinisikan konsep sebagai suatu gagasan / ide yang relatif sempurna dan bermakna, suatu pengertian tentang suatu objek, produk subjektif yang berasal dari cara seseorang membuat pengertian terhadap objek - objek atau benda - benda melalui pengalamannya (setelah melakukan persepsi terhadap objek / benda). Pada tingkat konkrit, konsep merupakan suatu gambaran mental dari beberapa objek atau kejadian yang sesungguhnya. Pada tingkat abstrak dan komplek, konsep merupakan sintesis sejumlah kesimpulan yang telah ditarik dari pengalaman dengan objek atau kejadian tertentu.

Kegiatan pembuatan keputusan meliputi pengidentifikasian masalah, pencarian alternatif penyelesaian masalah, evaluasi dari alternatif - alternatif tersebut dan pemilihan alternatif keputusan yang terbaik. Kemampuan seorang manajer dalam membuat keputusan dapat ditingkatkan apabila mengetahui dan menguasai teori dan teknik pembuatan keputusan. Dengan peningkatan

kemampuan manajer dalam pembuatan keputusan diharapkan dapat ditingkatkan kualitas keputusan yang dibuatnya, dan hal ini tentu akan meningkatkan efisiensi kerja manajer yang bersangkutan. (Kusrini ; 2007)

II.2.4. Keuntungan Sistem Pendukung Keputusan

Sistem pendukung keputusan dapat memerikan manfaat atau keuntungan bagi pemakainya. Keuntungan dimaksud diantaranya meliputi :

1. Sistem pendukung keputusan memperluas kemampuan pengambil keputusan dalam memproses data / informasi bagi pemakainya.
2. Sistem pendukung keputusan membantu pengambil keputusan dalam hal penghematan waktu yang dibutuhkan untuk memecahkan masalah terutama berbagai masalah yang sangat kompleks dan tidak terstruktur.
3. Sistem pendukung keputusan dapat menghasilkan solusi dengan lebih cepat dan hasilnya dapat diandalkan.
4. Walaupun suatu sistem pendukung keputusan, mungkin saja tidak mampu memecahkan masalah yang dihadapi oleh pengambil keputusan, namun ia dapat menjadi stimulasi bagi pengambil keputusan dalam memahami persoalannya. Karena sistem pendukung keputusan mampu menyajikan berbagai alternatif.
5. Sistem pendukung keputusan dapat menyediakan bukti tambahan untuk memberikan pembenaran sehingga dapat memperkuat posisi pengambil keputusan.

Disamping berbagai keuntungan dan manfaat seperti diungkapkan diatas. SPK juga memiliki beberapa keterbatasan, diantaranya adalah :

1. Ada beberapa kemampuan manajemen dan bakat manusia yang tidak dapat dimodelkan, sehingga model yang ada dalam sistem tidak mencerminkan persoalan sebenarnya.
2. Kemampuan suatu SPK terbatas pada pembendaharaan pengetahuan yang dimilikinya (pengetahuan dasar serta model dasar).
3. Proses - proses yang dapat dilakukan oleh SPK biasanya tergantung juga pada kemampuan perangkat lunak yang digunakannya.
4. SPK tidak memiliki kemampuan intuisi seperti yang dimiliki oleh manusia. Karena walau bagaimanapun canggihnya suatu SPK, dia hanyalah suatu kumpulan perangkat keras, perangkat lunak dan sistem operasi yang tidak dilengkapi dengan kemampuan berpikir. (Kusrini ; 2007)

II.2.5. Komponen Sistem Pendukung Keputusan

Adapun komponen - komponen dari SPK adalah sebagai berikut :

1. Subsistem manajemen data, mencakup satu basis data (*database)* yang berisi data yang relevan dan dikelola oleh perangkat lunak yang disebut *Database Management System* (DBMS).
2. Subsistem manajemen model, menggunakan perangkat lunak yang berkaitan dengan bidang - bidang seperti keuangan, statistik, manajemen, atau model - model kuantitatif yang memiliki kemampuan

untuk melakukan analisa sistem. Perangkat lunak ini dikenal dengan *Model Base Management System* (MBMS). Subsistem ini memiliki komponen yang dapat dikoneksikan ke penyimpanan eksternal yang ada pada model.

3. Subsistem antarmuka pengguna, digunakan sebagai media interaksi antara sistem dengan pengguna. Pengguna dapat berkomunikasi dengan SPK dan memerintahkan SPK melalui subsistem ini.
4. Subsistem manajemen berbasis pengetahuan, dapat mendukung subsistem lain atau bertindak sebagai komponen yang berdiri sendiri yang tidak terkait dengan komponen lain.

Manfaat yang dapat diambil dari sistem pendukung keputusan adalah :

1. Memperluas kemampuan pengambilan keputusan dalam memproses data / informasi bagi pemakainya.
2. Membantu pengambil keputusan untuk memecahkan masalah terutama berbagai masalah yang sangat kompleks dan tidak terstruktur.
3. Dapat menghasilkan solusi dengan lebih cepat serta hasilnya dapat diandalkan. (Kusrini ; 2007)

II.3. Metode Promethee

Promethee (*Preference Ranking Organization Method For Enrichment Evaluation*) adalah salah satu motodologi dalam pengambilan keputusan dengan multi kriteria (*Multi Criteria Decision Making*).

Promethee adalah salah satu dari beberapa metode yang termasuk MCDM yang berarti penentuan urutan atau prioritas dalam analisis multikriteria. Metode ini lebih efisien dan simple, selain itu metode ini juga mudah diterapkan dibanding dengan metode lain untuk menyelesaikan masalah yang berhubungan dengan multikriteria. Metode ini mampu mengakomodir kriteria pemilihan yang bersifat kuantitatif dan kualitatif. Masalah utamanya adalah kesederhanaan, kejelasan dan kestabilan. Dugaan dari dominasi kriteria yang digunakan dalam *Promethee* adalah penggunaan nilai dalam hubungan outranking. Kriteria adalah definisi masalah dalam bentuk yang konkret dan kadang - kadang dianggap sebagai sasaran yang akan dicapai. Semua parameter yang dinyatakan mempunyai pengaruh nyata menurut pandangan ekonomi.

Langkah - langkah perhitungan dengan metode *promethee* adalah sebagai berikut :

1. Penentuan alternatif - alternatif nilai dari data.
2. Menentukan tipe fungsi preferensi dan nilai preferensi.
3. Perhitungan indeks preferensi.
4. Perhitungan arah preferensi dipertimbangkan berdasarkan nilai *indeks leaving flow, enterflow* dan *net flow*.

Perhitungan nilai preferensi (P) antar alternatif ditentukan berdasarkan penyampaian intensitas (P) dari preferensi alternatif *a* terhadap alternatif *b* sedemikian rupa sehingga :

1. P (a, b) = 0, berarti tidak ada beda (*indefferent*) antara a dan b, atau tidak ada preferensi dari a lebih baik dari b.
2. P (a, b) ~ 0, berarti lemah preferensi dari a lebih baik dari b.
3. P (a, b) ~ 1, berarti kuat preferensi dari a lebih baik dari b.
4. P (a, b) = 1, berarti mutlak preferensi dari a lebih baik dari b.

Dalam metode Promethee ini fungsi preferensi seringkali menghasilkan nilai fungsi yang berbeda antara dua evaluasi sehingga :

$$P(a,b) = P(f(a) - f(b))$$

Dimana : P = Preferensi

f = alternatif

Indeks preferensi multikriteria di hitung untuk masing - masing pasangan kriteria dengan rumus dibawah ini :

$$¢(a,b) = \sum_{i=1}^{n} ® Pi(a,b); \forall a,b £ A$$

δ (a,b) adalah intensitas preferensi pembuat keputusan yang menyatakan bahwa alternatif a lebih baik lebih baik dari alternatif b dengan pertimbangan secara simultan dari seluruh kriteria, sehingga :

1. δ (a,b) = 0, menunjukan preferensi yang lemah untuk alternatif a lebih dari alternatif b berdasarkan semua kriteria.
2. δ (a,b) = 1, menunjukan preferensi yang kuat untuk alternatif a lebih dari alternatif b berdasarkan semua kriteria. (Eka Hendra Setyawan ; 2002)

II.3.1. Kelebihan Metode Promethee

Adapun terdapat kelebihan dari metode *promethee* adalah sebagai berikut :

1. Lebih jelas dan lebih sederhana / mudah dipahami oleh para praktisi.
2. Memperhitungkan data kualitatif sebaik data kuantitatif.
3. Menyediakan enam tipe preferensi terhadap kriteria.
4. Memperhitungkan kriteria berbeda pada saat yang sama, yang tidak mungkin dengan keputusan berbasis proses yang didasarkan hanya pada satu kriteria.
5. Dapat menggunakan kriteria yang berbeda untuk setiap dimensi.
6. Perangkingan alternatif dapat dilakukan secara parsial maupun lengkap. (Eka Hendra Setyawan ; 2002)

II.3.2. Kelemahan Metode Promethee

Disamping kelebihan diatas terdapat juga beberapa kekurangan dari metode *promethee* yaitu :

1. Membutuhkan informasi tambahan berupa fungsi preferensi tertentu yang harus didefenisikan / dijelaskan.
2. Tidak mampu menangani masalah optimasi terhadap kendala yang sangat mungkin ada dalam permasalahan pemilihan alternatif optimal. (Eka Hendra Setyawan ; 2002)

II.4. Visual Basic 2010

Visual basic berasal dari bahasa *BASIC* yang dikembangkan mulai dari tahun 1963. *Visual Basic* merupakan bahasa pemrograman yang dikembangkan dari bahasa pemrograman *basic*. *Akronim dari BASIC* adalah *Beginner's All Purpose Symbolic Instruction Code.* Dengan akronim tersebut dapat kita mengerti bahasa *BASIC* merupakan bahasa bagi pemula, mudah, dan andal untuk semua tujuan. Dengan kata lain bahasa *basic* dibuat dengan tujuan memudahkan *user* agar dapat dengan mudah mempelajari, membuat, dan mengembangkan program komputer. *Visual basic* merupakan bahasa yang dikembangkan dari *basic* yang ditujukan untuk membuat program cepat dengan tampilan *GUI (Graphical User Interface).* Istilah ini sering disebut dengan *RAD (Rapid Aplication Development).*

Bahasa pemrograman *visual basic* merupakan bahasa pemrograman utama dari perusahaan *Microsoft Inc* yang paling suskes hingga 12 tahun. Bahasa pemrograman ini menjadi contoh semua pemrograman *RAD*. Hingga tahun ini kepopuleran bahasa pemrograman *visual basic* masih bertahan kuat karena kemudahan, ringan dan andal. *Visual Studio 2010* merupkan *IDE* bahasa pemrograman *visual basic* menggunakan teknologi *.Net versi 3.5. Visual studio 2010* hadir dengan edisi *Team System, Professional Edition, Standard Edition*, Dan *Express Edition*.

Adapun perkembangan bahasa pemrograman *visual basic*, yaitu :

1. Bahasa *basic* dikembangkan dari tahun 1963.
2. *Visual Basic 1.0* versi 1 dirilis tahun 1991 digunakan untuk sistem operasi *Microsoft DOS / MSDOS*.

3. *Visual Basic 2.0* versi 2 tahun 1992.
4. *Visual Basic 3.0* versi 2 tahun 1993.
5. *Visual Basic 4.0* versi 2 tahun 1995.
6. *Visual Basic 5.0* versi 2 tahun 1997.
7. *Visual Basic 6.0* versi 6 tahun 1998. Versi ini sangat popular sehingga bertahan lama, dikarenakan versi ini tidak lagi didukung oleh *Microsost Inc* pada bulan Maret 1998.
8. *Visual Basic .Net* dirilis pada Februari 2002 digunakan untuk platform *.Net*.
9. *IDE (Integrated Development Environment)* versi 2002 menggunakan teknologi *.Net* versi 1.
10. *IDE* versi 2003 menggunakan teknologi *.Net* versi 1.1.
11. *IDE* versi 2005 berada pada *visual studio 2005* menggunakan teknologi *.Net* versi 2.0.
12. *IDE* versi 2008 berada pada *visual studio 2008* menggunakan teknologi *.Net* versi 3.5.
13. *IDE 2010 Visual Studio 2010*.

Visual basic 2010 merupakan salah satu bagian dari produk pemograman terbaru yang dikeluarkan oleh Microsoft, yaitu Microsoft Visual Studio 2010. Visual studio merupakan produk pemograman andalan dari Microsoft corporation, yang 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 standar yang disertakan adalah Microsoft SQL Server 2008 Express. Visual Basic 2010 merupakan versi perbaikan dan pengembangan dari versi pendahulunya, yaitu Visual Bassic 2008. (R.H Sianipar ; 2014)

II.5. My SQL Server

Microsoft SQL Server adalah sebuah sistem manajemen basis data relasional (RDBMS) produk Microsoft. Bahasa kueri utamanya adalah *Transact-SQL* yang merupakan implementasi dari SQL standar ANSI/ISO yang digunakan oleh *Microsoft* dan *Sybase*. Umumnya SQL Server digunakan di dunia bisnis yang memiliki basis data berskala kecil sampai dengan menengah, tetapi kemudian berkembang dengan digunakannya SQL Server pada basis data besar. (R.H Sianipar ; 2014)

II.6. UML (Unified Modeling Language)

UML *(Unified Modelling Language)* adalah salah satu alat bantu yang sangat handal di dunia pengembangan sistem yang berorientasi objek. Hal ini disebabkan karena UML menyediakan bahasa pemodelan visual yang memungkinkan bagi pengembang sistem untuk membuat cetak biru atas visi mereka dalam bentuk yang baku, mudah dimengerti serta dilengkapi dengan

mekanisme yang efektif untuk berbagi *(sharing)* dan mengkomunikasikan rancangan mereka dengan yang lain.

UML merupakan kesatuan dari bahasa pemodelan yang dikembangkan oleh *Booch*, *Object Modeling Technique* (OMT) dan *Object Oriented Software Engineering* (OOSE). Metode *Booch* dari *Grady Booch* sangat terkenal dengan nama metode *Designe Object Oriented*. Metode ini menjadikan proses analisis dan desain ke dalam empat tahapan interatif, yaitu identifikasi kelas - kelas dan obyek - obyek, identifikasi semantik dari hubungan obyek dan kelas tersebut, perincian *interface* dan implementasi. Keunggulan metode *Booch* adalah pada detil dan kayanya dengan notasi dan elemen. Pemodelan OMT yang dikembangkan oleh *Rumbaugh* didasarkan pada analisis terstruktur dan pemodelan *entity-relationship*. Tahapan utama dalam metodologi ini adalah analisis, desain sistem, desain obyek dan implementasi. Keunggulan metode ini adalah dalam penotasian yang mendukung semua konsep OO. Metode OOSE dari *Jacobson* lebih memberi penekanan pada *use case*. OOSE memiliki tiga tahapan, yaitu membuat membuat model *requirement* dan analisis, desain dan implementasi, dan model pengujian (tes model). Keunggulan metode ini adalah mudah dipelajari karena memiliki notasi yang sederhana namun mencakup seluruh tahapan dalam rekayasa perangkat lunak. (Munawar ; 2005)

II.6.1. Diagram UML

Umumnya sebuah sistem mempunyai sejumlah *stakeholder* (orang yang mempunyai ketertarikan pada suatu sistem namun dari sudut pandang yang

berbeda). Sebagai contoh ketika kita merancang sebuah sistem untuk seorang pelanggan, maka akan sangat berbeda ketika sistem tersebut kita terapkan untuk banyak pelanggan. Dari sini jelas kita sangat butuh banyak diagram dari berbagai sudut pandang. Dengan demikian tujuan utama dari banyaknya diagram ini adalah untuk memuaskan semua *stakeholder*. (Munawar ; 2005)

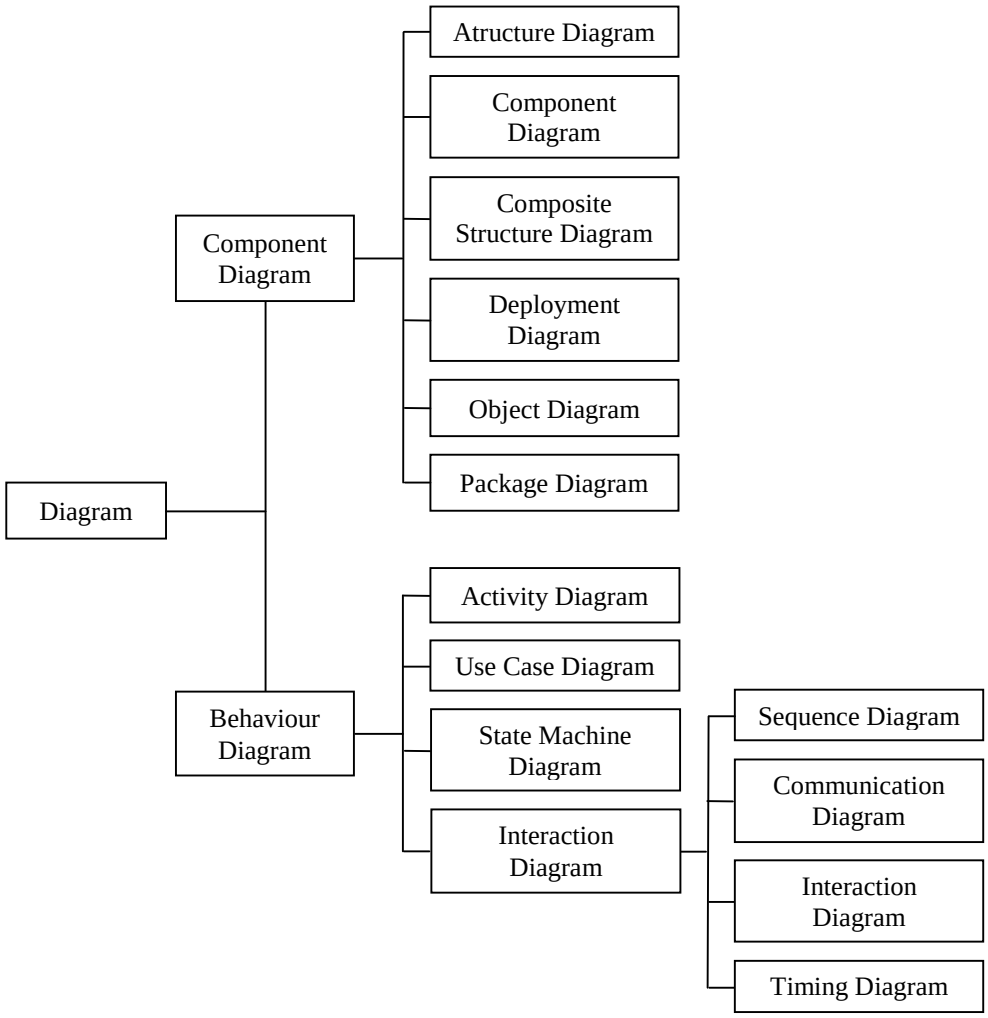
Adapun Tipe diagram UML dapat dilihat seperti pada tabel dibawah ini :

Tabel. II.1. Tipe Diagram UML

Diagram	Tujuan	Keterangan
Activity	Prilaku prosedural dan parallel	Sudah ada di UML 1
Class	Class, fitur dan relasinya	Sudah ada di UML 1
Communication	Interaksi diantara objek. Lebih menekankan kepada link	Di UML 1 disebut collaboration
Component	Struktur dan koneksi dari komponen	Sudah ada di UML 1
Composite Structure	Dekomposisi sebuah class saat runtime	Baru untuk UML 2
Deployment	Penyebaran/instalasi ke klien	Sudah ada di UML 1
Interaction Overview	Gabungan dari activity dan sequence diagram	Baru untuk UML 1
Object	Contoh konfigurasi instance	Tidak resmi ada di UML 1
Package	Struktur hierarki saat kompilasi	Tidak resmi ada di UML 1
Sequence	Interaksi antara objek. Lebih menekankan pada urutan.	Sudah ada di UML 1
State Machine	Bagaimana event mengubah sebuah objek	Sudah ada di UML 1
Timing	Interaksi antar objek. Lebih menekankan pada waktu	Sudah ada di UML 1
Use Case	Bagaimana user berinteraksi dengan sebuah sistem	Sudah ada di UML 1

(Sumber : Munawar ; 2005)

Adapun klasifikasi diagram UML seperti gambar dibawah ini :



Gambar II.1. Diagram UML
(Sumber : Munawar ; 2005)

II.6.2. Class Diagram

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

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

Kelas - kelas yang ada pada struktur sistem harus dapat melakukan fungsi - fungsi sesuai dengan kebutuhan sistem. Susunan struktur kelas yang baik pada diagram kelas sebaiknya memiliki jenis - jenis kelas berikut :

1. Kelas main.

   Kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.
2. Kelas yang menangani tampilan sistem.

   Kelas yang mendefinisikan dan mengatur tampilan ke pemakai.
3. Kelas yang diambil dari pendefinisian *use case*.

   Kelas yang menangani fungsi - fungsi yang harus ada diambil dari pendefinisian *use case*.
4. Kelas yang diambil dari pendefinisian data.

   Kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data. Jenis - jenis kelas di atas juga dapat digabungkan satu sama lain sesuai dengan pada struktur kelas tetap ada. Susunan kelas juga dapat ditambahkan kelas utilitas seperti koneksi ke basis data, membaca file teks, dan lain sebagainya pertimbangan yang dianggap baik asalkan fungsi - fungsi yang sebaiknya ada sesuai kebutuhan.

Berikut ini adalah simbol - simbol yang ada pada diagram kelas :

Tabel II.2. Simbol - Simbol Diagram Kelas

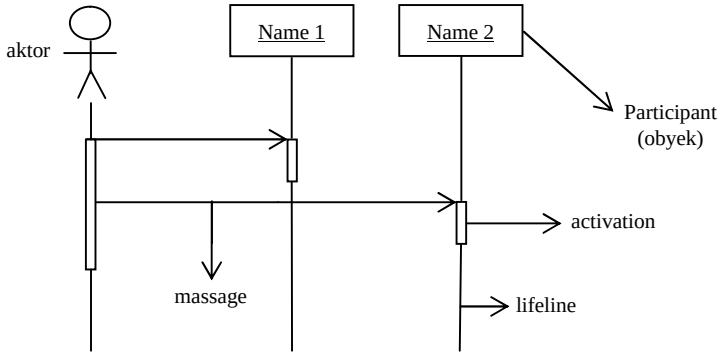
Simbol	Deskripsi
Kelas nama_kelas +atribut +operasi()	Kelas pada struktur sistem.
Antarmuka / *Interface* nama_interface	Sama dengan konsep *interface* dalam pemrograman berorientasi objek.
Asosiasi / *Association*	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*.
Asosiasi berarah / *directed association*	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*.
Generalisasi	Relasi antar kelas dengan makna generalisasi - spesialisasi (umum khusus).
Kebergantungan / *dependency*	Relasi antar kelas dengan makna kebergantungan antar kelas.
Agregasi / aggregation	Semua - bagian (*whole - part)*

(Sumber : Munawar ; 2005)

II.6.3. Sequence Diagram

Sequence Diagram digunakan untuk menggambarkan perilaku pada sebuah skenario. Diagram ini menunjukkan jumlah contoh objek dan *messege* (pesan) yang diletakkan diantara objek - objek ini dalam *use case*.

Komponen utama sequence diagram terdiri ats obyek yang dituliskan dengan kotak segiempat bernama. *Messege* diwakili oleh garis dengan tanda panah dan waktu yang ditunjukkan dengan *progress vertical*.



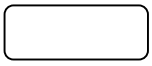
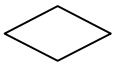
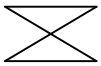
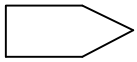
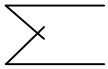
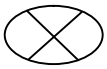
Gambar II.2. Contoh Sequence Diagram
(Sumber : Munawar ; 2005)

II.6.4. Activity Diagram

Actifity Diargram adalah teknik untuk mendiskripsikan logika prosedural. Proses bisnis dan aliran dalam banyak kasus. *Actifity Diagram* mempunyai simbol seperti halnya *flowchart*, akan tetapi perbedaannya dengan *flowchart* adalah *actifity diagram* bisa mendukung prilaku pararel sedangkan *flowchart* tidak bisa.

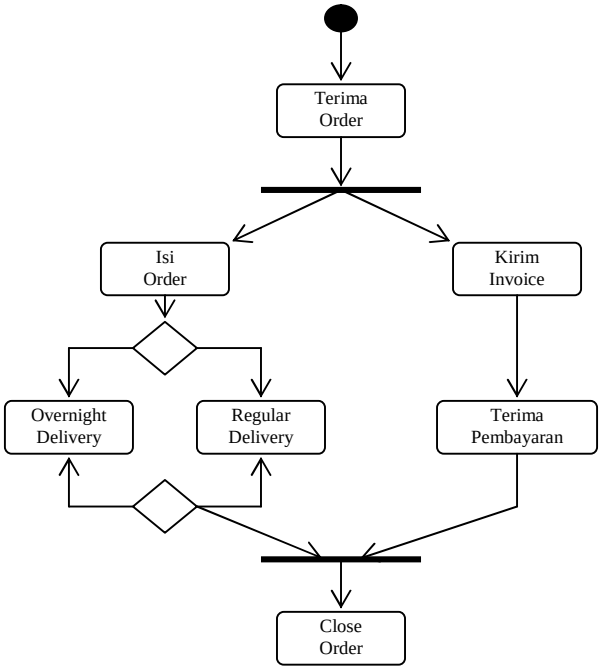
Simbol - simbol yang sering digunakan pada saat pembuatan *actifity diagram* dapat dilihat pada tabel II.3 seperti di bawah ini :

Tabel II.3. Simbol - Simbol Yang Sering Dipakai Pada Activity Diagram

Simbol	Keterangan
	Titik awal
	Titik awal
	Activity
	Pilihan untuk pengambilan keputusan
	Fork ; digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel atau untuk menggabungkan dua kegiatan paralel menjadi satu
	Rake; menunjukkan adanya dekomposisi
	Tanda waktu
	Tanda pengiriman
	Tanda penerimaan
	Aliran akhir (Flow Final)

(Sumber : Munawar ; 2005)

Adapun contoh dari *Activity Diagram* dapat di lihat pada gambar II.3 di bawah ini :



Gambar II.3. Contoh Activity Diagram Sederhana
(Sumber : Munawar 2005)

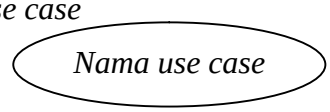
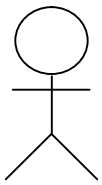
II.6.5. Use Case Diagram

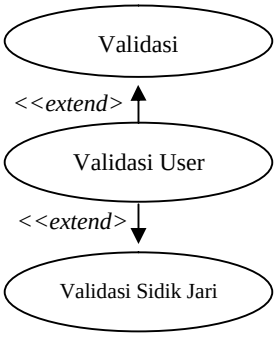
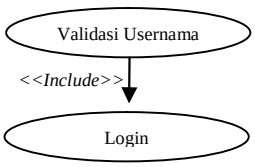
Use case atau *diagram use case* merupakan pemodelan untuk kelakukan (*behavior)* sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang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

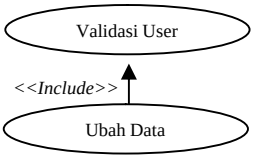
Syarat penamaan pada *use case* adalah nama didefinisikan semudah mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut aktor dan *use case*, yaitu :

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.4. Simbol – Simbol Diagram Use Case

Use case 	Fungsionalitas yang disediakan sistem sebagai unit - unit yang saling bertukaran pesan antar unit atau aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal di awal *frase* nama *use case*.
Aktor / Actor 	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang,tapi aktor belum tentu merupakan orang ; biasanya dinyatakan menggunakan kata benda di awal *frase* nama *actor*.
Asosiasi / Association _____	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan *actor*.

Ekstensi / *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, misalkan arah panah mengarah pada use case yang ditambahkan. 
Menggunakan / *Include* / *Uses* <<*include*>> <<*uses*>>	Relasi *use case* tambahkan ke sebuah *use case* dimana *use case* ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat di jalankan *use case* ini. Ada dua sudut pandang yang cukup besar mengenai *include* di *use case* : a. *Include* berarti *use case* yang ditambahkan akan selalu dipanggil saat *use case* tambahan dijalankan, misal pada kasus berikut :  b. *Include* berarti *use case* yang tambahan akan selalu melakukan pengecekan apakah *use case* yang ditambahkan telah dijalankan sebelum *use case* tambahan dijalankan, misal pada kasus berikut :

	 Kedua interpretasi di atas dapat dianut salah satu atau keduanya tergantung pada pertimbangan dan interpretasi yang dibutuhkan.

(Sumber : Munawar ; 2005)

II.7. Pengertian Database

Database merupakan komponen terpenting dalam pembangunan sistem informasi, karena menjadi tempat untuk menampung dan mengorganisasikan seluruh data yang ada dalam sistem, sehingga dapat di eksplorasi untuk menyusun informasi - informasi dalam berbagai bentuk. Database tersebut diorganisasikan sedemikian rupa agar tidak terjadi duplikasi yang tidak perlu, sehingga dapat diolah atau dieksporasi secara cepat dan mudah untuk menghasilkan informasi. (Budi Sutedji Dharma Oetomo ; 2006 : 99)

II.7.1. Hierarki Data Dalam Database

Data dalam sebuah database disusun berdasarkan sistem hierarki yang unik yaitu :

1. Database, merupakan kumpulan file yang saling terkait satu sama lain, misalnya file data induk karyawan, file jabatan, file penggajian dan lain sebagainya. Kumpulan file yang tidak saling terkait satu sama lain

tidak dapat disebut database, misalanya file data induk karyawan, file tamu undangan perkawinan, file barang retail pasar swalayan.

2. File, yaitu kumpulan dari record yang saling terkait dan memiliki format *field* yang sama dan sejenis.
3. Record, yaitu kumpulan *field* yang menggambarkan suatu unit data individu tertentu.
4. Field, yaitu atribut dari *field* yang menunjukkan suatu item dari data, seperti nama,alamat, dan lain sebagainya.
5. Byte, yaitu atribut dari *field* yang berupa huruf yang membentuk nilai dari sebuah *field*. Huruf tersebut dapat berupa numerik maupun abjad atau karakter khusus.
6. Bit, yaitu bagian terkecil dari data secara keseluruhan, yaitu berupa karakter ASCII nol atau satu yang merupakan komponen pembentuk *byte*. (Budi Sutedjo Dhrama Oetomo ; 2006 : 102).

II.8. Normalisasi

Normalisasi adalah teknik perancangan yang banyak digunakan sebagai pemandu dalam merancang basis data relasional. Pada dasarnya, normalisasi adalah proses dua langkah yang meletakkan data dalam bentuk tabulasi dengan menghilangkan kelompok berulang lalu menghilangkan data yang terduplikasi dari tabel relational.

Teori normalisasi didasarkan pada konsep bentuk normal. Sebuah tabel relasional dikatakan berada pada bentuk normal tertentu jika tabel memenuhi

himpunan batasan tertentu. Ada lima bentuk normal yang telah ditemukan. (Janner Simarmata & Imam Prayudi ; 2006 : 76)

1. Bentuk Normal Pertama (1NF / First Normal Form).

   Bentuk normal pertama adalah suatu bentuk relasi dimana atribut bernilai banyak *(multivalues atribute)* telah dihilangkan sehingga kita akan menjumpai nilai tunggal (mungkin saja nilai *null*) pada perpotongan setiap baris dan kolom satu nilai untuk irisan baris dan kolom pada tabel.

2. Bentuk Normal Kedua (2NF / Second Normal Form).

   Semua kebergantungan fungsional *(functional dependeny)* yang bersifat sebagian *(partial functional dependeny)* telah dihilangkan.

3. Bentuk Normal Ketiga (3NF / Third Normal Form).

   Semua kebergantungan transitif *(transitive dependeny)* telah dihilangkan.

4. Boyce-Codd Normal Form (BCNF / Boyce-Codd Normal Form).

   Semua anomali yang tersisa dari hasil penyempurnaan kebergantungan fungsional *(functional dependeny)* diatas telah dihilangkan.

5. Bentuk Normal Keempat (4NF / Fifth Normal Form).

   Semua anomali yang berasal dari kebergantungan banyak-nilai *(multivalues dependeny)* telah dihilangkan. (Adi Nugroho ; 2010 : 34)

Tujuan normalisasi adalah membuat kumpulan tabel relasional yang bebas dari data berulang yang dapat dimodifikasi secara benar dan konsisten. Ini berarti

bahwa semua tabel pada basis data relasional harus berada pada bentuk normal ketiga (3NF). Sebuah tabel relasional berada pada 3NF jika dan hanya jika semua kolom bukan kunci adalah (a) saling independen dan (b) sepenuhnya tergantung pada kunci utama. Saling independen berarti bahwa tidak ada kolom bukan kunci yang tergantung pada senbarang kombinasi kolom lainnya. Dua bentuk normal pertama adalah langkah antara untuk mencapai tujuan, yaitu mempunyai semua tabel dalam 3NF (Stephens and Plew, 2000). (Janner Simarmata & Imam Prayudi ; 2006 : 77)