BAB II

LANDASAN TEORI

II.1. Konsep Dasar

II.1.1. Sistem

Prof. Dr. Mr. S. Prajudi Atmosudirdjo menyatakan bahwa suatu sistem terdiri atas objek-objek atau unsur-unsur atau komponen-komponen yang berkaitan dan berhubungan satu sama lainnya sedemikian rupa sehingga unsur-unsur tersebut merupakan suatu kesatuan pemrosesan atau pengolahan yang tertentu. Sedangkan Norman L. Enger menyatakan bahwa suatu sistem dapat terdiri atas kegiatan-kegiatan yang berhubungan guna mencapai tujuan-tujuan perusahaan seperti pengendalian inventaris atau penjadwalan produksi.

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.

Secara sederhana sistem dapat diartikan sebagai suatu kumpulan atau himpunan dari unsur, komponen, atau variabel yang terorganisasi, saling berinteraksi, saling tergantung satu sama lain dan terpadu (Tata Sutabri, 2012: 3-7).

II.1.2. Karakteristik Sistem

Untuk memahami atau mengembangkan suatu sistem, maka perlu membedakan unsur-unsur dari sistem yang membentuknya. Berikut adalah karakteristik sistem yang dapat membedakan suatu sistem dengan sistem lainnya yaitu:

1. Batasan (*boundary*): Penggambaran dari suatu elemen atau unsur mana yang termasuk di dalam sistem dan mana yang di luar sistem.
2. Lingkungan (*environment*): Segala sesuatu di luar sistem, lingkungan yang menyediakan asumsi, kendala, dan input terhadap suatu sistem.
3. Masukan (*input*): Sumber daya (data, bahan baku, peralatan, energi) dari lingkungan yang dikonsumsi dan dimanipulasi oleh suatu sistem.
4. Keluaran (*output*): Sumber daya atau produk (informasi, laporan, dokumen, tampilan *layer computer*, barang jadi) yang disediakan untuk lingkungan sistem oleh kegiatan dalam suatu sistem.
5. Komponen (*component*): Kegiatan-kegiatan atau proses dalam sistem yang mentransformasikan *input* menjadi bentuk setengah jadi (*output*). Komponen ini bisa merupakan subsistem dari sebuah sistem.
6. Penghubung (*interface*): Tempat di mana komponen atau sistem dan lingkungannya bertemu atau berinteraksi.
7. Penyimpanan (*storage*): Area yang dikuasai dan digunakan untuk penyimpanan sementara dan tetap dari informasi, energi, bahan baku, dan sebagainya. Penyimpanan merupakan suatu media penyangga di antara komponen tersebut bekerja dengan berbagai tingkatan yang ada

dan memungkinkan komponen yang berbeda dari berbagai data yang sama (Aris, et al., 2015).

II.1.3. Pengertian Sistem

Sistem adalah suatu kesatuan usaha yang terdiri dari bagian-bagian yang berkaitan satu sama lain yang berusaha mencapai suatu tujuan dalam suatu lingkungan kompleks.

Kata sistem berasal dari bahasa Yunani yaitu “Sistema” yang berarti suatu kesatuan yang saling bergantung dan saling bekerja sama untuk mencapai tujuan tertentu. Suatu sistem dapat terdiri dari sistem-sistem bagian lainnya atau sering disebut subsistem. Sistem suatu jaringan kerja dari beberapa prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau menyelesaikan suatu tujuan tertentu untuk menerima input lalu memprosesnya dan akhirnya menghasilkan *output* (Achmad Hamzah Nasrullah, Dadang Sudrajat, 2015 : 2).

II.1.4. Informasi

Informasi merupakan proses lebih lanjut dari data yang sudah memiliki nilai tambah, informasi dapat dikelompokkan menjadi 3 bagian, yaitu:

1. Informasi Strategis. Informasi ini digunakan untuk mengambil keputusan jangka panjang.
2. Informasi Taktis. Informasi ini dibutuhkan untuk mengambil keputusan jangka menengah.

3. Informasi Teknis. Informasi ini dibutuhkan untuk keperluan operasional sehari-hari.

Informasi adalah data yang telah diklasifikasikan atau diolah atau diinterpretasikan untuk digunakan dalam proses pengambilan keputusan. Sistem pengolahan informasi akan mengolah data menjadi informasi atau mengolah data dari bentuk tak berguna menjadi berguna bagi yang menerimanya. Nilai informasi berhubungan dengan keputusan. Bila tidak ada pilihan atau keputusan maka informasi tidak diperlukan. Keputusan dapat berkisar dari keputusan berulang sederhana sampai keputusan strategis jangka panjang. Nilai informasi dilukiskan paling berarti dalam konteks pengambilan keputusan (Tata Sutabri, 2012: 21-22).

II.1.6. Sistem Informasi

Sistem informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian yang mendukung fungsi operasi organisasi yang bersifat manajerial dengan kegiatan strategi dari suatu organisasi untuk dapat menyediakan laporan-laporan yang diperlukan oleh pihak luar tertentu.

Tipe sistem informasi adalah sebagai berikut:

1. Sistem informasi Akuntansi
2. Sistem informasi Pemasaran
3. Sistem informasi Manajemen Persediaan
4. Sistem informasi Personalia
5. Sistem informasi Distribusi
6. Sistem informasi Pembelian

7. Sistem informasi Kekayaan
8. Sistem informasi Analisis Kredit
9. Sistem informasi Penelitian dan Pengembangan
10. Sistem informasi Teknik

Semua sistem informasi tersebut dimaksudkan untuk memberikan informasi kepada semua tingkat manajemen, mulai manajemen tingkat bawah, manajemen tingkat menengah, hingga manajemen tingkat atas (Tata Sutabri, 2012: 38-41).

II.2. Sistem Pakar *(Expert System)*

Secara umum, Sistem Pakar *(expert system)* adalah sistem yang berusaha mengadopsi pengetahuan manusia ke komputer, agar komputer dapat menyelesaikan masalah seperti yang biasa dilakukan oleh para ahli. Sistem Pakar yang baik dirancang agar dapat menyelesaikan suatu permasalahan tertentu dengan meniru kerja para ahli. Dengan Sistem Pakar ini, orang awam juga dapat menyelesaikan masalah yang cukup rumit yang sebenarnya hanya dapat diselesaikan dengan bantuan para ahli. Bagi para ahli, Sistem Pakar ini juga akan membantu aktivitasnya sebagai asisten yang sangat berpengalaman (Sri Rahayu, 2013 : 130).

Sistem pakar adalah suatu sistem yang dirancang untuk dapat menirukan keahlian seorang pakar dalam menjawab pertanyaan dan memecahkan masalah. Sistem pakar akan memberikan pemecahan suatu masalah yang didapat dari dialog dengan pengguna. Dengan bantuan Sistem Pakar seseorang yang bukan

pakar/ahli dapat menjawab pertanyaan, menyelesaikan masalah serta mengambil keputusan yang biasanya dilakukan oleh seorang pakar (Sutojo, T., 2011 : 13).

II.3. Karakteristik Sistem Pakar (Expert System)

II.3.1. Sistem Pakar

Ada beberapa definisi tentang Sistem Pakar, antara lain sebagai berikut :

1. Menurut Durkin : Sistem Pakar adalah suatu program komputer yang dirancang untuk memodelkan kemampuan penyelesaian masalah yang dilakukan seorang pakar.
2. Menurut Ignizo : Sistem Pakar adalah suatu model dan prosedur yang berkaitan, dalam suatu domain tertentu, yang mana tingkat keahliannya dapat dibandingkan dengan keahlian seorang pakar.
3. Menurut Giarratano dan Riley : Sistem Pakar adalah suatu sistem komputer yang bisa menyamai atau meniru kemampuan seorang pakar (Achmad Hamzah Nasrullah, Dadang Sudrajat, 2013 : 2-3).

II.3.2. Bentuk Sistem Pakar

Ada 4 bentuk Sistem Pakar, yaitu sebagai berikut : 1. Berdiri sendiri, Sistem Pakar jenis ini merupakan *Softwere* yang berdiri sendiri, tidak tergantung dengan *softwere* yang lain. 2. Tergabung. Sistem Pakar jenis ini merupakan bagian dari program yang terkandung di dalam suatu algoritma (konvensional) atau merupakan program dimana di dalamnya memanggil algoritma subrutin lain (konvensional). 3. Menghubungkan dengan *softwere* yang lain. Bentuk ini

biasanya menghubungkan Sistem Pakar yang menghubungkan suatu paket program tertentu, misalnya DBMS. 4. Sistem Mengabdi. Sistem Pakar merupakan bagian dari komputer khusus yang dihubungkan dengan suatu fungsi tertentu, misalnya Sistem Pakar yang digunakan untuk membantu menganalisis data radar (Achmad Hamzah Nasrullah, Dadang Sudrajat, 2013 : 3)

Ada beberapa alasan mendasar mengapa sistem pakar dikembangkan untuk menggantikan seorang pakar, di antaranya :

Dapat menyediakan kepakaran setiap waktu dan di berbagai lokasi.

1. Secara otomatis mengerjakan tugas-tugas rutin yang membutuhkan seorang pakar. Seorang pakar akan pensiun atau pergi.
2. Seorang pakar adalah mahal.
3. Kepakaran juga dibutuhkan pada lingkungan yang tidak bersahabat (*hostile environtment*) (Achmad Hamzah Nasrullah, Dadang Sudrajat, 2013 : 3).

II.4. Metode *Certainty Factor*

Teori Certainty Factor (CF) diusulkan oleh Shortlife dan Buchanan pada 1975 untuk mengakomodasi ketidakpastian pemikiran (*inexact reasoning*) seorang pakar. Seorang pakar, (misalnya dokter) sering kali menganalisis informasi yang ada dengan ungkapan seperti “mungkin”, “kemungkinan besar”, hampir pasti”. Untuk mangakomodasi hal ini kita menggunakan *certainty factor* (CF) guna menggambarkan tingkat keyakinan pakar terhadap masalah yang sedang dihadapi(T.Sutojo, dkk ; 2011 : 194).

Ada dua cara dalam mendapatkan tingkat keyakinan (CF) dari sebuah rule yaitu (T.Sutojo, dkk ; 2011 : 194-196):

1. Metode *'Net Belief'* yang diusulkan oleh E. H. Shortliffe B. G. Buchanan

   CF(Rule) = MB[H,E] – MD[H,E] .....(II.1)

$$MB(H,E) = \begin{cases} 1 & P(H) = 1 \\ \dfrac{\max[P(H|E),P(H)] - P(H)}{\max[1,0] - P(H)} & \text{lainnya} \end{cases} \quad \text{..... (II. 2)}$$

$$MD(H,E) = \begin{cases} 1 & P(H) = 0 \\ \dfrac{\min[P(H|E),P(H)] - P(H)}{\min[1,0] - P(H)} & \text{lainnya} \end{cases} \quad \text{..... (II. 3)}$$

   Dimana :

   CF(Rule) = Faktor kepastian

   MB(H,E) = *measure of belief* (ukuran kepercayaan) terhadap hipotesis H,

   jika diberikan *evidence* E (antara 0 dan 1)

   MD(H,E) = *measure of disbelief* (ukuran ketidakpercayaan) terhadap *Evidenc*e H, jika diberikan *evidence*E (antara 0 dan 1)

   P(H) = Probabilitas kebenaran hipotesis H

   P(H|E) = Probabilitas bahwa H benar karena fakta E

2. Dengan cara mewawancarai seorang pakar

   Nilai CF (*Rule*) didapat dari interpretasi "term" dari pakar, yang diubah menjadi nilai CF tertentu sesuai tabel berikut.

Tabel II.1.Nilai CF

Uncertain Term	CF
Definitely not (pasti tidak)	-1.0
Almost certainly not (hampir pasti tidak)	-0.8
Probably not (kemungkinan besar tidak)	-0.6
Maybe not (mungkin tidak)	-0.4
Unknown (tidak tahu)	-0.2 to 0.2
Maybe (mungkin)	0.4
Probably (kemungkinan besar)	0.6
Almost certainly (hampir pasti)	0.8
Definitely (pasti)	1.0

Sumber : T.Sutojo, dkk (2011 : 195-196)

II.4.1. Perhitungan *Certainty Factor* Gabungan

Secara umum, *rule* direpresentasikan dalam bentuk sebagai berikut (T.Sutojo, dkk ;2011 : 196-198) :

IF E_1 AND E_2 .....AND E_n THEN H (CF Rule)

Atau

IF E_1 OR E_2 .....OR E_n THEN H (CF Rule)

Di mana :

E_1 ... E_n : Fakta-fakta (evidence) yang ada

H : Hipotesis atau konklusi yang dihasilkan

CF (Rule) : Tingkat keyakinan terjadinya hipotesis H akibat adanya fakta-fakta E_1 ... E_n

1. *Rule* dengan *evidence* E tunggal dan Hipotesis H tunggal

   IF E THEN H (CF rule)

   CF(H,E) = CF(E) x CF (rule).....(II.4)

Catatan :

Secara praktik, nilai CF *rule* ditentukan oleh pakar, sedangkan nilai CF(E) ditentukan oleh pengguna saat berkonsultasi dengan sistem pakar.

2. *Rule* dengan *evidence* E ganda dan Hipotesis H tunggal

   IF E_1 AND E_2 .....AND E_n THEN H (CF Rule)

   CF(H,E) = min[CF(E_1), CF (E_2),....., CF(E_n)] x CF (rule).....(II.5)

   IF E_1 OR E_2 .....OR E_n THEN H (CF Rule)

   CF(H,E) = max[CF(E_1), CF (E_2),....., CF(E_n)] x CF (rule).....(II.6)

3. Kombinasi dua buah *rule* dengan *evidence* berbeda (E_1 dan E_2), tetapi hipotesis sama.

   IF E_1 THEN H Rule 1 $CF(H,E_1) = CF_1 = C(E_1)$ x CF (Rule1)

   IF E_2 THEN H Rule 2 $CF(H,E_2) = CF_2 = C(E_2)$ x CF (Rule2)

$$CF(CF_1, CF_2) = \begin{cases} CF_1 + CF_2(1 - CF_1) \text{ jika } CF_1 > 0 \; dan CF_2 > 0 \\ \dfrac{CF_1 + CF_2}{1 - \min[|CF_1|, |CF_2|]} \text{ jika } CF_1 < 0 \; atau CF_2 < 0 \\ CF_1 + CF_2 \, x \, (1 + CF_1) \text{ jika } CF_1 < 0 \; dan CF_2 < 0 \end{cases} \quad \ldots (II.7)$$

II.4.2. Kelebihan dan Kekurangan Metode *Certainty Factor*

Kelebihan metode *Certainty Factor* adalah (T.Sutojo, dkk ;2011 : 204) :

1. Metode ini cocok dipakai dalam sistem pakar yang mengandung ketidakpastian.
2. Dalam sekali proses perhitungan hanya dapat mengolah 2 data saja sehingga kekurangan data dapat terjaga.

Sedangkan kekurangan metode *Certainty Factor* adalah:

1. Pemodelan ketidakpastian yang menggunakan perhitungan metode *Certainty Factor* biasanya masih diperdebatkan.
2. Untuk data lebih dari 2 buah, harus dilakukan beberapa kali pengolahan data.

II.5. *Microsoft Visual Studio 2010*

Microsoft Visual Studio 2010 merupakan sebuah perangkat lunak yang dapat digunakan untuk melakukan pengembangan aplikasi, yang didalamnya terdapat beberapa kumpulan *tools* pemrograman seperti *Visual Basic .NET, Visual C++ .NET, Visual C# .NET,* dan *Visual J# .NET*. Namun yang paling umum digunakan yaitu *Visual Basic .NET*. *Visual Basic .NET* adalah *Visual Basic* yang direkayasa kembali untuk digunakan pada *platform .NET* sehingga aplikasi yang dibuat menggunakan *Visual Basic .NET* dapat berjalan pada sistem komputer apapun, dan dapat mengambil data dari *server* dengan tipe apapun asalkan terinstal *.NET Framework* (Priyanto, 2012 : 5).

II.6. *Microsoft SQL Server*

Microsoft SQL Server merupakan produk *Relational Database Management System* (RDBMS) yang dibuat oleh *Microsoft*. Pada tahun 2008 *Microsoft* mengeluarkan *SQL Server 2008 R2* yang merupakan versi yang banyak digunakan. *SQL* (*Structured Query Language*) adalah sebuah bahasa yang dipergunakan untuk mengakses data dalam basis data relasional. Bahasa ini secara *de facto* merupakan bahasa standar yang digunakan dalam manajemen basis data

relasional. Saat ini hampir semua *server* basis data yang ada mendukung bahasa ini untuk melakukan manajemen datanya. *SQL* terdiri dari dua bahasa, yaitu *Data Definition Language* (DDL) dan *Data Manipulation Language* (DML). Implementasi DDL dan DML berbeda untuk tiap Sistem Manajemen Basis Data (SMBD), namun secara umum implementasi setiap bahasa ini memiliki bentuk standar yang ditetapkan oleh ANSI (Adelia, 2011 : 115).

II.7. Pengertian *Database*

Database sering didefinisikan sebagai kumpulan data yang terkait. Secara teknis, yang berada dalam sebuah database adalah sekumpulan tabel atau objek lain (indeks, *view*, dan lain-lain). Tujuan utama pembuatan *database* adalah untuk memudahkan dalam mengakses data (Eka Choliviana, Lies Yulianto, 2013: 55).

Basis data sendiri dapat didefinisikan dalam sejumlah sudut pandang seperti (Akhmad Sholikhin, 2013 : 51) :

1. Himpunan kelompok data (arsip) yang saling berhubungan yang diorganisasikan sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah.
2. Kumpulan data yang saling berhubungan disimpan secara bersama sedemikian rupa dan tanpa pengulangan yang yang perlu untuk memenuhi berbagai kebutuhan.
3. Kumpulan *file* atau tabel atau arsip yang saling berhubungan yang disimpan dalam media penyimpanan elektronis.

Berdasarkan tingkat kompleksitas nilai data, tingkatan data dapat disusun dalam sebuah hierarki, mulai dari yang paling sederhana hingga paling kompleks.

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). (Edy Sutanta, 2011 : 35-36).

II.8. Normalisasi

Normalisasi sendiri merupakan cara pendekatan lain dalam membangun desain lojik basis data relasional yang tidak secara langsung berkaitan dengan model data, tetapi dengan menerapkan sejumlah aturan dan kriteria standar untuk menghasilkan struktur tabel yang normal (Elena Monica, et al., 2015 : 69).

Proses normalisasi merupakan proses pengelompokan elemen data menjadi tabel-tabel yang menunjukkan entitas dan relasinya. Proses ini selalu diuji pada beberapa kondisi. Apakah ada kesulitan pada saat menambah *(insert),* menghapus *(delete),* mengubah *(update),* atau membaca *(retrieve)* pada suatu *database*. Bila ada kesulitan pada pengujian tersebut maka relasi dapat dipecah dalam beberapa tabel lagi (Tata Sutabri, 2012: 138).

Sebuah tabel dapat dikatagorikan baik (efisien) atau normal, jika telah memenuhi 3 (tiga) kriteria berikut:

1. Jika ada dekomposisi (penguraian) tabel, maka dekomposisinya harus dijamin aman (*Lossless-Join Decomposition*).
2. Terpeliharanya ketergantungan fungsional pada saat perubahan data (*Dependency Preservation*).
3. Tidak melanggar *Boyce-Code Normal Form* (BCNF).

Jika kriteria ketiga (BCNF) tidak dapat terpenuhi, maka paling tidak harus diupayakan agar tabel tersebut tidak melanggar bentuk normal tahap ketiga (*3rdNormal Form/ 3NF*).

Bentuk-bentuk Normal (*Normal Form*) yang lain adalah:

1. Bentuk Normal tahap Pertama (*1st Normal Form/1NF*)

   Bentuk Normal tahap Pertama (1NF) terpenuhi jika sebuah tabel tidak memiliki Atribut Bernilai Banyak (*Multivalued Attribute*) atau lebih dari satu atribut dengan domain nilai yang sama.

2. Bentuk Normal tahap Kedua (*2nd Normal Form/2NF*) Bentuk Normal tahap Kedua (2NF) terpenuhi jika pada sebuah tabel, semua atribut yang tidak termasuk dalam *key primer* memiliki Ketergantungan Fungsional (KF) pada *keyprimer* secara utuh. Sebuah tabel dikatakan tidak memenuhi 2NF, jika ketergantungannya hanya bersifat parsial (hanya tergantung pada sebagian dari *key primer*).

3. Bentuk Normal tahap Keempat (*4th Normal Form*) dan Bentuk Normal tahap Kelima (*5thNormal Form*).

Penerapan aturan Normalisasi sampai dengan tahap ketiga sesungguhnya sudah sangat memadai untuk menghasilkan tabel-tabel yang berkualitas baik. Namun demikian, dari sejumlah *literature* dapat pula dijumpai adanya pembahasan tentang Bentuk Normal tahap Keempat (4NF) dan Bentuk Normal tahap Kelima (5NF). Bentuk Normal tahap Keempat berkaitan dengan sifat Ketergantungan Banyak Nilai (*Multivalued Dependency*) pada suatu tabel yang merupakan pengembangan dari Ketergantungan Fungsional. Sedangkan Bentuk

Normal tahap Kelima (merupakan nama lain dari *Project-Join Normal Form*/ PJNF) berkenaan dengan Ketergantungan Relasi antar tabel (*Join Dependency*). Pembahasan kedua bentuk normal yang terakhir ini cukup kompleks, tetapi manfaatnya sendiri tidak begitu besar (Elena Monica, et al., 2015 : 69-70).

II.9. *Unified Modeling Language* (UML)

Menurut Henderi (2012:152), *Unified Modelling Language* (UML) merupakan satu kumpulan konvensi pemodelan yang digunakan untuk menentukan atau menggambarkan sebuah sistem *software* yang terkait dengan objek. UML merupakan salah satu alat bantu yang sangat handal dalam bidang pengembangan sistem berorientasi objek karena UML menyediakan bahasa pemodelan visual yang memungkinkan pengembang sistem membuat *blue print* atas visinya dalam bentuk yang baku. UML berfungsi sebagai jembatan dalam mengkomunikasikan beberapa aspek dalam sistem melalui jumlah elemen grafis yang bisa dikombinasikan menjadi diagram (Rosana Junita Sirait, et al., 2015).

Menurut Rama (2008:111), "*Unified Modeling Language* (UML) adalah suatu bahasa permodelan untuk menyebutkan, memvisualisasikan, membuat dan mendokumentasikan sistem informasi. "Menurut Henderi (2007:4), "*Unified Modeling Language* (UML) adalah sebuah bahasa pemodelan yang telah menjadi standar dalam industri *software* untuk visualisasi, merancang, dan mendokumentasikan sistem perangkat lunak. "Bahasa pemodelan UML lebih cocok untuk pembuatan perangkat lunak dalam bahasa pemrograman berorientasi

objek (C+,Java,VB.NET), namun demikian tetap dapat digunakan pada bahasa pemrograman prosedural.

Berdasarkan beberapa pendapat dikemukakan diatas dapat ditarik kesimpulan bahwa "*Unified Modeling Language* (UML) adalah sebuah bahasa yang berdasarkan grafik atau gambar untuk menvisualisasikan, menspesifikasikan, membangun dan pendokumentasian dari sebuah sistem pengembangan perangkat lunak berbasis OO (*Object Oriented*) (Aris, et al., 2015).

II.9.1. *Use Case* Diagram

Use case adalah deskripsi dari sebuah sistem dari perspektif pengguna. *Use case* bekerja dengan cara mendeskripsikan tipikal interaksi antara *user* (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai.

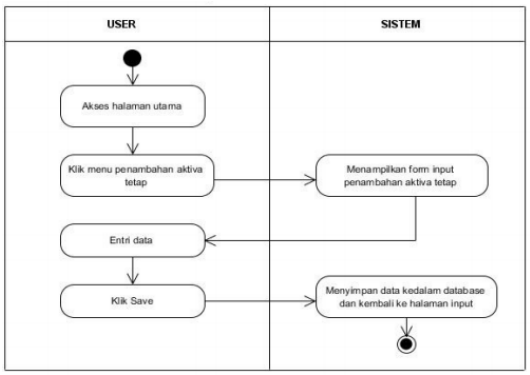
Use Case memiliki dua istilah, yaitu:

1. *System use case*; interaksi dengan sistem.
2. *Business use case*; interaksi bisnis dengan konsumen atau kejadian nyata.

Gambar II.1. Notasi *Use Case* Diagram
(Sumber :Haviluddin, Vol. 6, No. 1, 2011)

II.9.2. *Activity* Diagram

Activity diagram menggambarkan aktifitas-aktifitas, objek, *state*, *transisi state* dan *event*. Dengan kata lain kegiatan diagram alur kerja menggambarkan perilaku sistem untuk aktivitas. *Activity* diagram adalah teknik untuk mendeskripsikan logika prosedural, proses bisnis dan aliran kerja dalam banyak kasus (Haviluddin, Vol. 6, No. 1, 2011).



Gambar II.2. Notasi *Activity* Diagram
(Sumber : Haviluddin, Vol. 6, No. 1, 2011)

II.9.3. *Class* Diagram

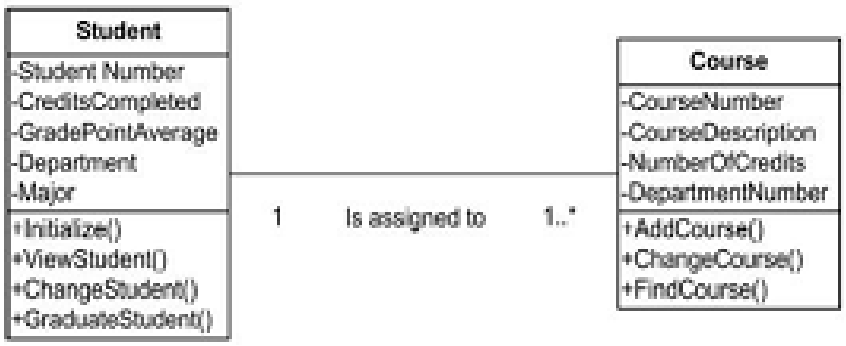
Class diagram merupakan diagram paling umum yang dijumpai dalam pemodelan berbasis UML. Didalam *Class* diagram terdapat *class* dan *interface* beserta atribut-atribut dan operasinya, relasi yang terjadi antar objek, *constraint* terhadap objek-objek yang saling berhubungan dan *inheritance* untuk organisasi *class* yang lebih baik. *Class* diagram juga terdapat *static view* dari elemen pembangun sistem. Pada intinya *Class* diagram mampu membantu proses pembuatan sistem dengan memanfaatkan konsep *forward* ataupun *reverse engineering* (Edgar Winata, Johan Setiawan, 2013 : 38).

Berbagai simbol yang hadir didalam *class* diagram antara lain adalah (Edgar Winata, Johan Setiawan, 2013 : 38) :

1. *Class*, yang berfungsi untuk merepresentasikan tipe dari data yang dimilikinya. *Class* diagram dapat ditampilkan dengan menunjukkan atribut dan operasi yang dimilikinya atau hanya menunjukkan nama *class*-nya saja. Dapat juga kita tuliskan nama *class* dengan atributnya saja atau nama *class* dengan operasinya.
2. *Attribute*, merupakan data yang terdapat didalam *class* dan *instance*-nya dengan operator.
3. *Operation*, berfungsi untuk merepresentasikan fungsi-fungsi yang ditampilkan oleh *class* dan *instance*-nya dengan operator.
4. *Association*, digunakan untuk menunjukkan bagaimana dua *class* berhubungan satu sama lainnya. *Association* ditunjukkan dengan sebuah garis yang terletak diantara dua *class*. Didalam setiap

association terdapat *multiplicity*, yaitu simbol yang mengindikasikan berapa banyak *instance* dari *class* pada ujung *association* yang satu dengan *instance class* di ujung *association* lainnya.

5. *Generalizations*, berfungsi untuk mengelompokkan *class* ke dalam hirarki *inheritance*.
6. *Aggregation*, merupakan bentuk khusus dari *association* yang merepresentasikan hubungan "*part-whole*". Bagian "*whole*" dari hubungan ini sering disebut dengan *assembly* atau *aggregate*. *Class* yang satu dapat dikatakan merupakan bagian dari *class* yang lain yang ikut membentuk *class* tersebut.
7. *Composition*, merupakan jenis *aggregation* yang lebih kuat diantara dua *class* yang memiliki *association* dimana jika *whole* ditiadakan, maka *part*-nya juga ikut ditiadakan. Berbeda dengan *aggregation*, *part* akan tetap bisa berdiri sendiri meskipun bagian *whole*-nya ditiadakan.
8. Penggunaan operator (+) dalam *class* diagram diartikan dengan *public*, operator (-) diartikan *private*, dan operator (#) diartikan *protected*.

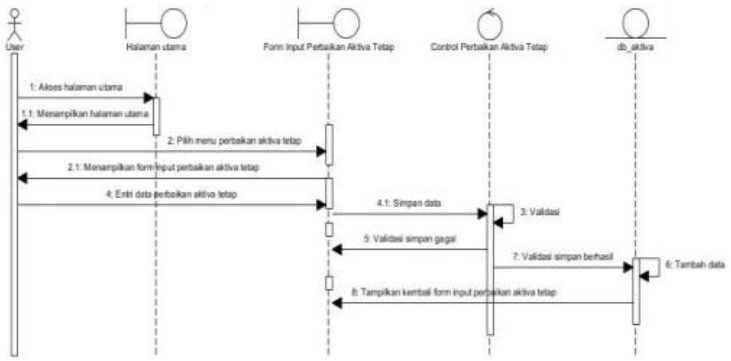


Gambar II.3. Contoh *Class* Diagram
(Sumber :Edgar Winata, Johan Setiawan, 2013 : 38)

II.9.4. *Sequence* Diagram

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

Secara mudahnya *sequence* diagram adalah gambaran tahap demi tahap, termasuk kronologi (urutan) perubahan secara logis yang seharusnya dilakukan untuk menghasilkan sesuatu sesuai dengan *use case diagram* (Haviluddin, Vol. 6, No. 1, 2011).



Gambar II.4. Notasi *Sequence* Diagram
(Sumber : Haviluddin, Vol 6, No. 1, 2011)