BAB II

LANDASAN TEORI

II.1. Konsep Dasar

II.1.1. Sistem

Sistem merupakan kumpulan dari unsur atau elemen-elemen yang saling berkaitan/berinteraksi dan saling mempengaruhi dalam melakukan kegiatan bersama untuk mencapai suatu tujuan tertentu (Asbon Hendra, 2011 : 157).

Berikut ini merupakan pengertian sistem dari beberapa ahli:

1. Jerry FithGerald “Sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan dan berkumpul bersama-sama untuk melakukan suatu kegiatan atau menyelesaikan suatu sasaran tertentu.
2. Ludwig Von Bartalanfy “Sistem merupakan seperangkat unsur yang saling terikat dalam suatu antar relasi di antara unsur-unsur tersebut dengan lingkungan.
3. Anatol Raporot “Sistem adalah suatu kumpulan kesatuan dan perangkat hubungan satu sama lain.
4. L. Ackof “Sistem adalah setiap kesatuan secara konseptual atau fisik yang terdiri dari bagian-bagian dalam keadaan saling tergantung satu sama lainnya.

Dari uraian di atas, sehingga dapat disimpulkan bahwa sistem adalah sekumpulan elemen yang saling terkait atau terpadu untuk mencapai tujuan tertentu.

II.1.2. Karakteristik Sistem

Model umum sebuah sistem terdiri dari input, proses, dan output. Selain itu sebuah sistem juga memiliki karakteristik atau sifat-sifat tertentu, yang mencirikan bahwa hal tersebut bisa dikatakan sebagai suatu sistem. Adapun karakteristik yang dimaksud adalah sebagai berikut (Asbon Hendra, 2012 : 158-160) :

1. Komponen Sistem *(Components)*

   Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, yang bekerja sama membentuk satu kesatuan. Komponen-komponen sistem dapat berupa suatu subsistem atau bagian-bagian dari sistem.

2. Lingkungan Luar Sistem *(Environtment)*

   Environment merupakan segala sesuatu diluar batas sistem yang memengaruhi operasi dari suatu sistem. Lingkungan luar sistem ini dapat bersifat menguntungkan atau merugikan. Lingkungan luar yang menguntungkan harus dipelihara dan dijaga agar tidak hilang pengaruhnya, sedangkan lingkungan luar yang merugikan harus dimusnahkan atau dikendalikan agar tidak mengganggu operasi sistem.

3. Batasan Sistem *(Boundary)*

   Batasan sistem merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lainnya atau dengan lingkungan luarnya. Batas suatu sistem menunjukkan ruang lingkup (*scope*) dari sistem tersebut.

4. Penghubung Sistem *(interface)*

   Media penghubung antara suatu subsistem dengan subsistem yang lainnya untuk membentuk satu kesatuan sehingga sumber-sumber daya mengalir dari subsistem yang satu ke subsistem yang lainnya. Dengan kata lain, *output* dari suatu subsistem akan menjadi *input* dari subsistem yang lainnya.

5. Masukan Sistem *(Input)*

   Energi yang dimasukan ke dalam sistem disebut masukan sistem. Masukan dapat berupa Masukan Perawatan (*Maintenance Input*) adalah energi yang dimasukkan supaya sistem tersebt dapat beroperasi.

6. Keluaran Sistem *(Output)*

   Merupakan hasil dari energi yang diolah oleh sistem, meliputi *output* yang berguna, contohnya informasi yang dikeluarkan oleh komputer. Dan *output* yang tidak berguna dikenal sebagai sisa pembuangan, contohnya panas yang dikeluarkan oleh komputer.

7. Pengolahan Sistem (*Process*)

   Bagian yang memproses masukan untuk menjadi keluaran yang diinginkan.

8. Tujuan Sistem *(Goal)*

   Setiap sistem pasti mempunyai tujuan ataupun sasaran yang memengaruhi *input* yang dibutuhkan dan *output* yang dihasilkan. Dengan kata lain, suatu sistem akan dikatakan berhasil kalau pengoperasian sistem itu mengenai sasaran atau tujuannya.

II.2. Data Mining

II.2.1. Defenisi Data Mining

Data mining merupakan suatu proses pendukung pengambil keputusan dimana kita mencari pola informasi dalam data. Pencarian ini dapat dilakukan oleh pengguna, misalnya dengan menggunakan query atau dapat dibantu dengan suatu aplikasi yang secara otomatis mencari pola informasi pada basis data. Pencarian ini disebut discovery (Sri Rahayu Siregar, 2014: 153).

Data mining merupakan suatu proses pendukung pengambil keputusan dimana kita mencari pola informasi dalam data. Pencarian ini dapat dilakukan oleh pengguna, misalnya dengan menggunakan query atau dapat dibantu dengan suatu aplikasi yang secara otomatis mencari pola informasi pada basis data. Pencarian ini disebut discovery. Discovery adalah proses pencarian dalam basis data untuk menemukan pola yang tersembunyi tanpa ide yang didapatkan sebelumnya atau hipotesa tentang pola yang ada. Dengan kata lain aplikasi mengambil inisiatif untuk menemukan pola dalam data tanpa pengguna berpikir mengenai pertanyaan yang relevan terlebih dulu (Eka Novita Sari, 2013 : 35-36).

Data mining adalah proses yang menggunakan teknik statistik, matematika, kecerdasan buatan, dan *machine learning* untuk mengekstraksi dan mengidentifikasi informasi yang bermanfaat dan pengetahuan yang terkait dari berbagai basis data besar. Data mining, sering juga disebut *Knowledge Discovery in Database* atau disingkat menjadi KDD, adalah kegiatan yang meliputi pengumpulan, pemakaian data historis untuk menemukan keteraturan, pola atau hubungan dalam set data berukuran besar (Hapsari Dita Anggraeni, 2013 : 2).

II.2.2. Pengelompokkan Data Mining

Data mining dibagi menjadi beberapa kelompok berdasarkan tugas yang dapat dilakukan, yaitu :

1. Deskripsi

   Terkadang peneliti dan analis secara sederhana ingin mencoba mencari data untuk menggambarkan pola dan kecenderungan yang terdapat dalam data. Sebagai contoh, petugas pengumpulan suara mungkin tidak dapat menentukan keterangan atau fakta bahwa siapa yang tidak cukup professional akan sedikit didukung dalam pemilihan presiden. Deskripsi dari pola dan kecenderungan sering memberikan kemungkinan penjelesan untuk suatu pola atau kecenderungan.

2. Estimasi

   Estimasi hampir sama dengan klasifikasi, kecuali variable target estimasi lebih kearah numerik dari pada kearah kategori. Model dibangun menggunakan record lengkap yang menyediakan nilai dari

variabel target sebagai prediksi. Selanjutnya, pada peninjauan berikutnya estimasi nilai dari variabel target dibuat berdasarkan nilai variabel predikasi. Sebagai contoh akan dilakukan estimasi tekanan darah sistolik pada pasien rumah sakit berdasarkan umur pasien, jenis kelamin, indeks berat badan, dan level sodium darah. Hubungan antara tekanan darah sistolik dan nilai variabel prediksi dalam proses pembelajaran akan menghasilkan model estimasi. Model estimasi yang dihasilkan dapat digunakan untuk kasus baru lainnya.

3. Prediksi

   Prediksi hampir sama dengan klasifikasi dan estimasi, kecuali bahwa dalam predikasi nilai dari hasil akan ada dimasa mendatang.

Contoh prediksi bisnis dan penelitian adalah:

a. Prediksi harga beras dalam tiga bulan yang akan datang.
b. Prediksi persentasi kenaikan kecelakaan lalu lintas tahun depan jika batas bawah kecepatan dinaikkan.

Beberapa metode dan teknik yang digunakan dalam klasifikasi dan estimasi dapat pula digunakan (untuk keadaan yang tepat) untuk prediksi.

4. Klasifikasi

   Dalam klasifikasi, terdapat target variabel kategori. Sebagai contoh, penggolongan pendapatan dapat dipisahkan dalam tiga kategori, yaitu pendapatan tinggi, pendapatan sedang, dan pendapatan rendah.

Contoh lain klasifikasi dalam bisnis dan penelitian adalah:

a. Menentukan apakah suatu transaksi kartu kredit merupakan transaksi yang curang atau tidak.
b. Memperkirakan apakah suatu pengajuan hipotek oleh nasabah merupakan suatu kredit yang baik atau buruk.
c. Mendiagnosis penyakit seorang pasien untuk mendapatkan termasuk kategori penyakit apa.

5. Pengklusteran (clustering)

Pengklusteran merupakan pengelompokan record, pengamatan, atau memperhatikan dan membentuk kelas objek-objek yang memiliki kemiripan. Kluster adalah kumpulan record yang memiliki kemiripan satu dengan yang lainnya dan memiliki ketidakmiripan dengan record-record dalam kluster lain. Pengklusteran berbeda dengan klasifikasi yaitu tidak adanya variabel target dalam pengklusteran. Pengklusteran tidak mencoba untuk melakukan klasifikasi, mengestimasi, atau memprediksi nilai dari variabel target. Akan tetapi, algoritma pengklusteran mencoba untuk melakukan pembagian terhadap keseluruhan data menjadi kelompok-kelompok yang memiliki kemiripan (homogeny), yang mana kemiripan dalam satu kelompok akan bernilai maksimal, sedangkan kemiripan dengan record dalam kelompok lain akan bernilai minimal.

Contoh pengklusteran dalam bisnis dan penelitian adalah:

a. Mendapatkan kelompok-kelompok konsumen untuk target pemasaran dari satu suatu produk bagi perusahaan yang tidak memiliki dana pemesaran yang besar.
b. Untuk tujuan audit akuntansi, yaitu melakukan pemisahan terhadap perilaku financial dalam baik dan mencurigakan.
c. Melakukan pengklusteran terhadap ekspresi dari gen, untuk mendapatkan kemiripan perilaku dari gen dalam jumlah besar.

6. Asosiasi

Tugas asosiasi dalam data mining adalah menemukan *attribut* yang muncul dalam satu waktu. Dalam dunia bisnis lebih umum disebut analisis keranjang belanja.

Contoh asosiasi dalam bisnis dan penelitian adalah :

a. Meneliti jumlah pelanggan dari perusahaan telekomunikasi seluler yang diharapkan untuk memberikan respon positif terhadap penawaran *upgrade* layanan yang diberikan.
b. Menentukan barang dalam supermarket yang dibeli secara bersamaan dan yang tidak pernah dibeli secara bersamaan (Kennedi Tampubolon, 2013 : 97).

II.2.3. Fungsi dan Tugas Data Mining

Data mining menganalisis data menggunakan tool untuk menemukan pola dan aturan dalam himpunan data. Perangkat lunak bertugas untuk menemukan pola dengan mengidentifikasi aturan dan fitur pada data. Tool data

mining diharapkan mampu mengenal pola ini dalam data dengan input minimal dari user (Kennedi Tampubolon, 2013 : 97).

II.3. Algoritma Apriori

Algoritma Apriori adalah suatu algoritma dasar yang diusulkan oleh Agrawal & Srikant pada tahun 1994 untuk penentuan *frequent itemsets* untuk aturan asosiasi *boolean*. Algoritma apriori termasuk jenis aturan asosiasi pada data mining. Aturan yang menyatakan asosiasi antara beberapa *attribut* sering disebut *affinity analysis* atau *market basket analysis.* Analisis asosiasi atau *association rule mining* adalah teknik data mining untuk menemukan aturan suatu kombinasi *item.* Salah satu tahap analisis asosiasi yang menarik perhatian banyak peneliti untuk menghasilkan algoritma yang efisien adalah analisis pola frequensi tinggi(frequent pattern mining). Penting tidaknya suatu asosiasi dapat diketahui dengan dua tolak ukur, yaitu : *support* dan *confidence*. Support (nilai penunjang) adalah persentase kombinasi *item* tersebut dalam database, sedangkan *confidence* (nilai kepastian) adalah kuatnya hubungan antar-*item* dalam aturan asosiasi (Sri Rahayu Siregar, 2014 : 153).

Metodologi dasar analisis asosiasi terbagi menjadi dua tahap yaitu (Kennedi Tampubolon, 2013 : 98).

1. Analisis pola frekuensi tinggi

   Tahap ini mencari kombinasi item yang memenuhi syarat minimum dari nilai *support* dalam database. Nilai *support* sebuah *item* diperoleh dengan memakai rumus berikut :

Support (A) =

$$\frac{\text{Jumlah Transaksi Mengandung A}}{\text{Total Transaksi}} \quad \text{.....(1)}$$

Sedangkan nilai dari *support* dua *item* diperoleh dari rumus berikut :

Support (A,B) = (A∩B)

$$\frac{\text{Jumlah Transaksi Mengandung A dan B}}{\text{Total Transaksi}} \quad \text{.....(2)}$$

2. Pembentukan Aturan Asosiasi

   Setelah semua pola frekuensi tinggi ditemukan, barulah dicari aturan assosiatif yang memenuhi syarat minimum untuk *confidence* dengan menghitung *confidence* aturan *asosiasi* ” jika A maka B ”. Nilai *confidence* dari aturan ” jika A maka B ” diperoleh dari rumus berikut :

   Confidence = P(B|A) =

$$\frac{\text{Jumlah Transaksi Mengandung A dan B}}{\text{Jumlah Transaksi Mengandung A}} \quad \text{.....(3)}$$

II.4. *Microsoft Visual Basic* 2010

Pada akhir tahun 1999, Teknologi .NET diumumkan. Microsoft memosisikan teknologi tersebut sebagai *platform* untuk membangun XML Web *Services*. XML Web *services* memungkinkan aplikasi tipe manapun dan dapat mengambil data yang tersimpan pada server dengan tipe apapun melalui internet.

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 apa pun, dan dapat

mengambil data dari server dengan tipe apa pun asalkan terinstal .NET Framework. (Priyanto Hidayatullah, 2012 ; 5).

II.5. *SQL Server* 2008

SQL *(Structured Query Language)* adalah sebuah bahasa yang dipergunakan umtuk mengakses data dalam basis data relasional. Bahasa ini secara *de facto* merupakan bahasa standar yang dipergunakan 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 Manipulation Language (DML)*. Implementasi DDL dan DML sistem manajemen basis data *(SMBD)*, namun secara umum implemen bahasa ini memiliki bentuk standar yang ditetapkan oleh *ANSI*. (Adelia, Jimmy Setiawan : 2011 ; 115).

II.6. Pengertian Database

Basis data dapat dipahami sebagai suatu kumpulan data terhubung (*interrelated data*) yang disimpan secara bersama-sama pada suatu media, tanpa *mengatap* satu sama lain atau tidak perlu suatu kerangkapan data (kalaupun ada maka kerangkapan data tersebut harus seminimal mungkin dan terkontrol [*controlled redudancy*]), data disimpan dengan cara-cara tertentu sehingga mudah digunakan/atau ditampilkan kembali; data dapat digunakan oleh satu atau lebih program-program aplikasi secara optimal; data disimpan tanpa mengalami ketergantungan dengan program yang akan menggunakannya; data disimpan

sedemikian rupa sehingga proses penambahan, pengambilan, dan modifikasi data dapat dilakukan dengan mudah dan terkontrol (Edy Sutanta, 2011 : 29-30).

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

1. Sisem basis data, merupakan sekumpulan subsistem yang terdiri atas basis data dengan para pemakai yang menggunakan basis data secara bersama-sama, personal-personal yang merancang dan mengelola basis data, teknik-teknik untuk merancang dan mengelola basis data, serta sistem komputer untuk mendukungnya.
2. Basis data, merupakan sekumpulan dari bermacam-macam tipe *record* yang memiliki hubungan antar-*record* dan rincian data terhadap obyek tertentu.
3. File, merupakan sekumpulan *record* sejenis secara relasi yang tersimpan dalam media penyimpanan sekunder.
4. *Record*, merupakan *field*/atribut/data item yang saling berhubungan terhadap obyek tertentu.
5. *Data item/field*/atribut, merupakan unit terkecil yang disebut data, sekumpulan *byte* yang mempunyai makna.
6. *Data agregate*, merupakan sekumpulan data item/*field*/atribut dengan ciri tertentu dan diberi nama.
7. *Byte*, adalah bagian terkecil yang dialamatkan dalam memori. *Byte* merupakan sekumpulan *bit* yang secara konvensional terdiri atas

kombinasi 8 *bit* biner yang menyatakan sebuah karakter dalam memori (1 *byte* = 1 karakter).

8. *Bit*, adalah sistem biner yang terdiri atas dua macam nilai, yaitu 0 dan 1. Sistem biner merupakan dasar yang dapat digunakan untuk komunikasi antara manusia dan mesin (komputer).

II.7. Normalisasi

Normalisasi diartikan sebagai suatu teknik yang menstrukurkan/ mendekomposisi data dalam cara-cara tertentu untuk mencegah timbulnya permasalahan pengolahan data dalam basis data. Permasalahan yang dimaksud adalah berkaitan dengan penyimpangan-penyimpangan *(anomalies)* yang terjadi akibat adanya kerangkapan data dalam relasi dan in-efisiensi pengolahan (Edy Sutanta ; 2011 : 174).

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

1. Memiliki struktur *record* yang konsisten secara logik;
2. Memiliki struktur *record* yang mudah untuk dimengerti;
3. Memiliki struktur *record* yang sederhana dalam pemeliharaan;
4. Memiliki struktur *record* yang mudah ditampilkan kembali untuk memenuhi kebutuhan pengguna;
5. Minimalisasi kerangkapan data guna meningkatkan kinerja sistem.

Secara berturut-turut masing-masing level normal tersebut dibahas berikut ini, dimulai dari bentuk tidak normal. (Edy Sutanta ; 2011 : 176-179)

1. Relasi bentuk tidak normal (*Un Normalized Form* / UNF)

   Relasi-relasi yang dirancang tanpa mengindahkan batasan dalam defisi basis data dan karakteristik *Relational Database Management System* (RDBM) menghasilkan relasi *Un Normalized Form* (UNF). Bentuk ini harus di hindari dalam perancangan relasi dalam basis data. Relasi *Un Normalized Form* (UNF) mempunyai kriteria sebagai berikut.

   a. Jika relasi mempunyai bentuk *non flat file* (dapat terjadi akibat data disimpan sesuai dengan kedatangannya, tidak memiliki struktur tertentu, terjadi duplikasi atau tidak lengkap)
   b. Jika relasi membuat *set atribut* berulang *(non single values)*
   c. Jika relasi membuat *atribut non atomic value*

2. Relasi bentuk normal pertama (*First Norm Form* / 1NF)

   Relasi disebut juga *First Norm Form* (1NF) jika memenuhi kriteria sebagai berikut.

   a. Jika seluruh atribut dalam relasi bernilai *atomic (atomic value)*
   b. Jika seluruh atribut dalam relasi bernilai tunggal *(single value)*
   c. Jika relasi tidak memuat set atribut berulang
   d. Jika semua record mempunyai sejumlah atribut yang sama.

   Permasalahan dalam *First Norm Form* (1NF) adalah sebagai berikut.

   a. Tidak dapat menyisipkan informasi parsial
   b. Terhapusnya informasi ketika menghapus sebuah *record*

3. Bentuk normal kedua *(Second Normal Form /* 2NF)

Relasi disebut sebagai *Second Normal Form (*2NF) jika memenuhi kriteria sebagai berikut

a. Jika memenuhi kriteria *First Norm Form (*1NF)
b. Jika semua atribut nonkunci *Functional Dependence* (FD) pada *Primary Key* (PK)

Permasalahan dalam *Second Normal Form* / 2NF adalah sebagai berikut:

a. Kerangkapan data *(data redundancy)*
b. Pembaharuan yang tidak benar dapat menimbulkan inkonsistensi data *(data inconsistency)*
c. Proses pembaharuan data tidak efisien

Kriteria tersebut mengidentifikasikan bahwa antara atribut dalam *Second Normal Form* masih mungkin mengalami *Third Norm Form*. Selain itu, relasi *Second Normal Form* (2NF) menuntut telah didefinisikan atribut *Primary Key* (PK) dalam relasi. Mengubah relasi *First Norm Form* (1NF) menjadi bentuk *Second Normal Form* (2NF) dapat dilakukan dengan mengubah struktur relasi dengan cara :

a. Identifikasikan *Functional Dependence* (FD) relasi *First Norm Form* (1NF)
b. Berdasarkan informasi tersebut, dekomposisi relasi *First Norm Form* (1NF) menjadi relasi-relasi baru sesuai *Functional Dependence* nya. Jika menggunakan diagram maka simpul-simpul

yang berada pada puncak diagram ketergantungan data bertindak *Primary Key* (PK) pada relasi baru

4. Bentuk normal ketiga (*Third Norm Form* / 3NF)

   Suatu relasi disebut sebagai *Third Norm Form* jika memenuhi kriteria sebagai berikut.

   a. Jika memenuhi kriteria *Second Normal Form* (2NF)
   b. Jika setiap atribut nonkunci tidak (TDF) *(Non Transitive Dependeny)* terhadap *Primary Key* (PK)

   Permasalahan dalam *Third Norm Form* (3NF) adalah keberadaan penentu yang tidak merupakan bagian dari *Primary Key* (PK) menghasilkan duplikasi rinci data pada atribut yang berfungi sebagai *Foreign Key* (FK) (duplikasi berbeda dengan keterangan data).

   Mengubah relasi *Second Normal Form* (2NF) menjadi bentuk *Third Norm Form* (3NF) dapat dilakukan dengan mengubah struktur relasi dengan cara :

   a. Identifikasi TDF relasi *Second Normal Form* (2NF)
   b. Berdasarkan informasi tersebut, dekomposisi relasi *Second Normal Form* (2NF) menjadi relasi-relasi baru sesuai TDF-nya.

5. Bentuk normal *Boyce-Cood (Boyce-Codd Norm Form* / BCNF)

   Bentuk normal *Boyce-Codd Norm Form* (BCNF) dikemukakan oleh R.F. Boyce dan E.F. Codd. Suatu relasi disebut sebagai *Boyce-Codd Norm Form* (BCNF) jika memenuhi kriteria sebagai berikut.

   a. Jika memenuhi kriteria *Third Norm Form* (3NF)

b. Jika semua atribut penentu (determinan) merupakan CK

6. Bentuk normal keempat (*Forth Norm Form* / 4NF)

   Relasi disebut sebagi *Forth Norm Form* (4NF) jika memenuhi kriteria sebagai berikut.

   a. Jika memenuhi kriteria *Boyce-Codd Norm Form*.

   b. Jika setiap atribut didalamnya tidak mengalami ketergantungan pada banyak nilai.

7. Bentuk normal kelima (*Fifth Norm Form* / 5NF)

   Suatu relasi memenuhi kriteria *Fifth Norm Form* (5NF) jika kerelasian antar data dalam relasi tersebut tidak dapat direkonstruksi dari struktur relasi yang sederhana.

8. Bentuk normal kunci domain (*Domain Key Norm Form* / DKNF)

a. Relasi disebut sebagai *Domain Key Norm Form* (DKNF) jika setiap batasan dapat disimpulkan secara sederhana dengan mengetahui sekumpulan nama atribut dan domainnya selama menggunkan sekumpulan atribut pada kuncinya..

II.8. *Unified Language Model* (UML)

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).

UML menyediakan kumpulan alat yang sudah terstandarisasi, yang digunakan untuk mendokumentasikan analisis dan perancangan sebuah sistem perangkat lunak. (Kendall & Kendall, 2005, p663) Peralatan utama UML adalah diagram-diagram yang digunakan untuk membantu manusia dalam memvisualisasikan proses pengembangan sebuah sistem perangkat lunak, sama seperti penggunaan denah (*blueprint*) dalam pembuatan bangunan (Edgar Winata dan Johan Setiawan, 2013).

II.8.1. Fungsi *Unified Modeling Language (UML)*

Unified Modeling Language (UML) biasa digunakan untuk (Aris, et al., 2015):

a. Menggambarkan batasan sitem dan fungsi -fungsi sistem secara umum, dibuat dengan *use case* dan *actor*.
b. Menggambarkan kegiatan atau proses bisnis yang dilaksanakan secara umum, dibuat dengan *interaction diagrams*.
c. Menggambarkan representasi struktur *static* sebuah sistem dalam betuk *class diagrams*.
d. Membuat model behavior “yang menggambarkan kebiasaan atau sifat sebuah sistem” dengan *state transition diagrams*.

e. Menyatakan arsitektur implementasi fisik menggunakan *component and development.*

f. Menyampaikan atau memperluas *fungsionality* dengan *stereotypes*.

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 sejumlah elemen grafis yang bisa dikombinasikan menjadi diagram. UML mempunyai banyak diagram yang dapat mengakomodasikan berbagai sudut pandang dari suatu perangkat lunak yang akan dibangun. Diagram-diagram tersebut digunakan untuk :

a. Mengkomunikasikan ide.

b. Melahirkan ide-ide baru dan peluang-peluang baru.

c. Menguji ide dan membuat prediksi.

d. Memahami struktur dan relasi-relasinya

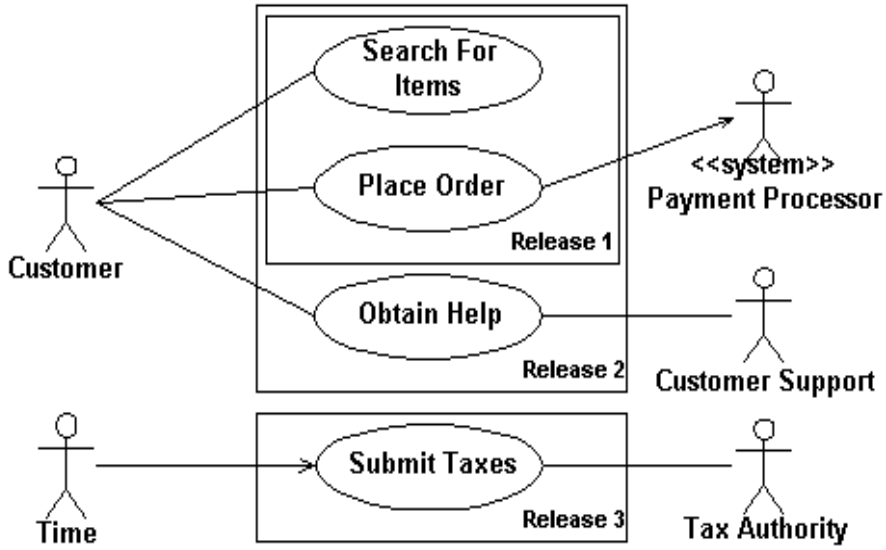
II.8.2. Diagram-Diagram *Unified Modeling Language* (UML)

Adapun jenis-jenis dari diagram UML adalah sebagai berikut :

1. *Use Case* Diagram

Diagram yang menggambarkan *actor*, *use case* dan relasinya sebagai suatu urutan tindakan yang memberikan nilai terukur untuk aktor. Sebuah *use case* digambarkan sebagai *elips horizontal* dalam suatu diagram UML *use case*. *Use Case* memiliki dua istilah, yaitu (Haviluddin, 2011) :

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



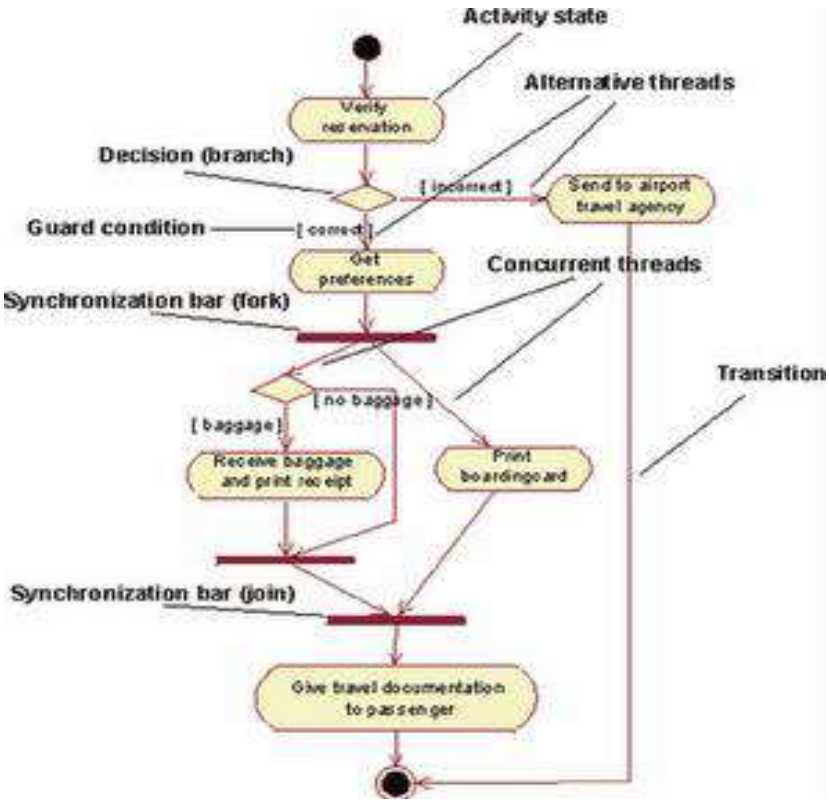
Gambar II.1. Notasi *Use Case* Diagram

(Sumber : Haviluddin, 2011)

2. *Activity* Diagram

Activity diagram memperlihatkan alur langkah demi langkah dalam suatu proses. Suatu aktivitas menunjukkan sekumpulan aksi (secara sekuensial atau bercabang dari satu aksi ke aksi lain), dan nilai yang dihasilkan atau digunakan oleh aksi-aksi yang terjadi. Activity diagram ditunjukkan untuk memodelkan fungsi dari suatu sistem dan menekankan pada alur dari kontrol didalam pelaksanaan dari suatu tindakan (Achmad Hamzah Nasrullah dan Dadang Sudrajat, 2015).

Activity diagram menggambarkan aktifitas-aktifitas, objek, *state*, *transisi state* dan *event*. Dengan kata lain kegiatan diagram alur kerja menggambarkan perilaku sistem untuk aktivitas (Haviluddin, 2011).



Gambar II.2. Notasi Activity Diagram

(Sumber : Haviluddin, 2011)

3. Class Diagram

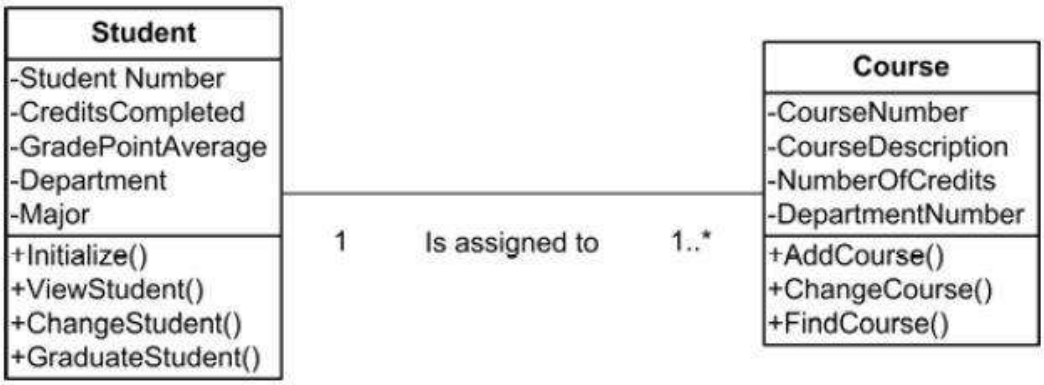
Class diagram merupakan diagram paling umum yang dijumpai dalam pemodelan berbasis UML. Didalam *Class* diagram terdapat *class* dan *interface* beserta atribut-atribut dan operasinya, relasi yang terjadi antar objek, *constraint* terhadap objek-objek yang saling berhubungan dan *inheritance* untuk organisasi *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 dan Johan Setiawan, 2013).

Berbagai simbol yang hadir didalam *class* diagram antara lain adalah:

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*.

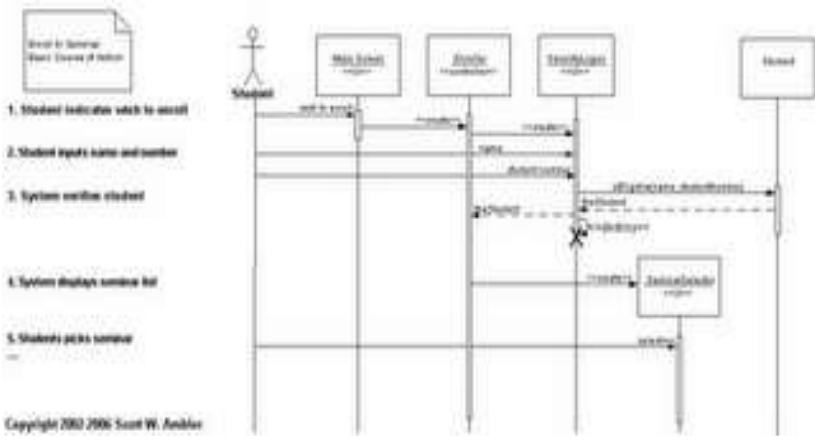


Gambar II.3. Contoh *Class* Diagram
(Sumber : Edgar Winata dan Johan Setiawan, 2013)

4. *Sequence* Diagram

Menjelaskan interaksi obyek-obyek yang saling berkolaborasi (berhubungan), mirip dengan *activity* diagram yaitu menggambarkan alur kejadian sebuah aktivitas tetapi lebih detil dalam menggambarkan aliran data termasuk data atau behaviour yang dikirimkan atau diterima namun kurang mampu menjelaskan detil dari sebuah algoritma (Edgar Winata dan Johan Setiawan, 2013).

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, 2011).



Gambar II.4. Notasi *Sequence* Diagram
(Sumber : Haviluddin, 2011)