BAB II

TINJAUAN PUSTAKA

II.1. Konsep Sistem Informasi

II.1.1. Pengertian 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)

1. Menurut 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. Menurut Ludwig Von Bartalanfy, sistem merupakan seperangkat unsur yang saling terikat dalam suatu antar relasi di antara unsur – unsur tersebut dengan lingkungan.
3. Menurut Anatol Raporot, sistem adalah suatu kumpulan kesatuan dan perangkat hubungan satu sama lain.
4. Menurut L. Ackof, sistem adalah setiap kesatuan secara konseptual atau fisik yang terdiri dari bagian – bagian dalam keadaan saling tergantung satu sama lain.

II.1.2. Syarat – Syarat Sistem

1. Sistem harus dibentuk untuk menyelesaikan tujuan.
2. Elemen system harus mempunyai rencana yang ditetapkan.

3. Adanya hubungan di antara elemen sistem.
4. Unsur dasar dari pross (arus informasi, energy, dan material) lebih penting daripada elemen sistem.
5. Tujuan Organisasi lebih penting dari tujuan elemen.

II.1.3. Karakteristik Sistem

Model Umum sebuah sistem terdiri dari input, proses, output. Hal ini merupakan konsep sebuah sistem yang sangat sederhana mengingat sebuah sistem dapat mempunyai beberapa masukan dan keluaran sekaligus. 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:

1. Komponen Sistem (*Components*)

   Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, yang bekerja sama membentuk satu kesatuan. Komponen-komponen tersebut dapat berupa suatu bentuk subsistem. Setiap subsistem memiliki sifat-sifat sistem yang menjalankan suatu fungsi tertentu dan mempengaruhi proses sistem secara keseluruhan. Suatu sistem dapat mempunyai sistem lebih besar yang disebut dengan supra sistem.

2. Batasan Sistem (*Boundary*)

   Batasan sistem merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lainnya atau dengan lingkungan luarnya. Batasan sistem ini memungkinkan suatu sistem dipandang sebagai suatu kesatuan, karena dengan batas sistem ini fungsi dan tugas dari subsistem yang satu dengan

lainnya berbeda tetapi tetap saling berinteraksi. Batas suatu sistem menunjukkan ruang lingkup (*scope*) dari sistem tersebut.

3. Lingkungan Luar Sistem (*Environtment)*

Environment merupakan segala sesuatu di luar batas system yang mempengaruhi 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.

4. Penghubung Sistem (*Interface*)

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

5. Masukan Sistem (*input*)

   Merupakan energy yang dimasukkan ke dalam system. Masukan dapat berupa Masukan Perawatan (*Maintenance Input*) adalah energy yang dimasukkan supaya system tersebut dapat beroperasi

6. Keluaran Sistem (*Output*)

   Merupakan hasil dari energy yang diolah oleh system, meliputi *output* yang berguna, contohnya informasi yang dikeluarkan oleh computer.

7. Pengolahan Sistem (*Process*)

   Merupakan bagian yang memperoses masukan untuk menjadi keluaran yang diinginkan. Contoh CPU pada computer, bagian produksi yang mengubah bahan baku menjadi barang jadi, serta bagian akuntansi yang mengelolah data transaksi menjadi laporan keuangan.

8. Tujuan Sistem (*Goal*)

   Setiap system pasti mempunyai tujuan ataupun sasaran yang mempengaruhi *input* yang dibutuhkan dan *output* yang dihasilkan (Asbon Hendra; 2011:158).

II.1.4. Klasifikasi Sistem

Sistem dapat diklasifikasikan dari beberapa sudut pandang, diantaranya adalah sebagai berikut :

1. Sistem diklasifikasikan sebagai system abstrak (*abstrak system*) dan system fisik (*physical system*).
2. Sistem diklasifikasikan sebagai system alamiah (*natural system*) dan system buatan manusia (*human made system*).
3. Sistem diklasifikasikan sebagai system tertentu (*deterministic system*) dan system tidak tentu (*probabilistic system*).
4. Sistem diklasifikasikan sebagai system tertutup (*closed system*) dan system terbuka (iopen system) (Asbon Hendra; 2011:160).

II.2. Konsep Sistem Pendukung Keputusan

II.2.1. Sistem Pendukung Keputusan / *Decision Support System* (DSS)

Sistem Pendukung Keputusan (SPK) merupakan suatu pendekatan atau metodelogi untuk mendukung keputusan. SPK menggunakan CBIS (Computer Based Information System) yang fleksibel, interaktif dan dapat diadaptasi, yang dikembangkan untuk mendukung solusi untuk masalah manajemen spesifik yang tidak terstruktur. SPK menggunakan data, memberikan antarmuka pengguna yang mudah dan dapat menggabungkan pemikiran pengambil keputusan.

Sebagai tambahan, SPK biasanya menggunakan berbagai model dan dibangun oleh suatu proses interaktif dan iterative. Ia mendukung semua fase pengambilan keputusan dan dapat memasukkan suatu komponen pengetahuan.
SPK dapat digunakan oleh pengguna tunggal pada suatu PC atau bias menjadi Web untuk digunakan oleh banyak orang pada beberapa lokasi (Perdana & Widodo; 2013).

II.2.2. Tahapan Sistem Pengambilan Keputusan

Didalam proses pengambilan keputusan memiliki 4 tahap yang harus dilalui Menurut Herbert A. Simon (Adiwisanghagni; 2015).

1. Penelusuran (*intelligence*)

   Tahap ini merupakan tahap pendefenisian masalah serta identifikasi informasi yang dibutuhkan yang berkaitan dengan persoalan yang dihadapi serta keputusan yang akan diambil.

2. Perancangan (*disign*)

Tahap ini merupakan tahap analisa dalam kaitan mencari atau merumuskan alternatif-alternatif pemecahan masalah.

3. Pemilihan (*choice*)

Yaitu memilih alternatif solusi yang diperkirakan paling sesuai.

4. Implementasi (*implementation*)

Tahap ini merupakan tahap pelaksanaan dari keputusan yang telah diambil.

II.2.3. Karakteristik *Decision Support System* (DSS) / Sistem Pendukung Keputusan

Menurut Rini (2013), Sistem pendukung keputusan memiliki karakteristik sebagai berikut :

1. Sistem pendukung keputusan dirancang untuk membantu pengambilan keputusan dalam memecahkan masalah yang bersifat semi terstruktur dengan menambahkan kebijaksanaan manusia dan informasi komputerisasi.
2. Dalam proses pengolahannya, sistem pendukung keputusan mengkombinasikan pengguna model-model analisi dengan teknik pemasukkan data konvesional serta fungsi-fungsi interogasi informasi.
3. Sistem Pendukung keputusan, dirancang sedemikian rupa sehingga dapat digunakan / dioperasikan dengan mudah.
4. Sistem Pendukung Keputusan dirancang dengan menentukan pada aspek fleksibilitas serta kemampuan beradaptasi yang tinggi.

II.2.4. Keterbatasan Sistem Pendukung Keputusan

Suatu sistem pendukung keputusan memiliki 2 keterbatasan yaitu :

a. Adanya gambaran bahwa SPK seakan – akan hanya dibutuhkan pada tingkat manajemen puncak. Pada kenyataannya, dukungan bagi pengambilan keputusan dibutuhkan pada semua tingkatan manajemen dalam suatu organisasi.

b. Pengambilan keputusan yang terjadi pada beberapa level harus dikoordinasikan. Jadi, dimensi dan pendukung keputusan adalah komunikasi dan koordinasi diantara pengambilan keputusan antar level organisasi yang berbeda maupun pada level organisasi yang sama.

II.3. Metode *Technique For Order Preference by Similarity to Ideal Solution* (TOPSIS).

TOPSIS adalah salah satu metode pengambilan keputusan multi criteria yang pertama kali diperkenalkan oleh Yonn dan Hwang (1981) dengan ide dasarnya adalah bahwa alternatif yang dipilih memiliki jarak terdekat dengan solusi ideal positif dan memiliki jarak terjauh dasi solusi ideal negatif. Berikut ini adalah contoh sebuah matriks dengan alternatif dan *criteria*.

Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) didasarkan pada konsep dimana alternatif terpilih yang terbaik tidak hanya memiliki jarak terpendek dari solusi ideal positif, namun juga memiliki jarak terpanjang dari solusi ideal negatif (Freklin, 2011).

II.3.1. Tahapan Dalam Metode TOPSIS

Metode TOPSIS memiliki beberapa tahapan yaitu :

1. Membuat matriks keputusan yang ternormalisasi Elemen rij hasil dari normalisasi decision matrix R dengan metode Euclidean length of a vector adalah:

$$r_{ij} \frac{ij}{\sqrt{\sum_i^m = 1 x_i^2 j}} \cdots (1)$$

Dimana :

rij = hasil dari normalisasi matriks keputusan R

i = 1,2,3,...,m;

j = 1,2,3,...,n;

xij= elemen dari matriks keputusan, i = 1,2,3, ..., m,

j = 1, 2, 3, ..., n.

2. Membuat matriks keputusan yang ternormalisasi terbobot

Dengan bobot W = (w1,w2,....,wn), maka normalisasi bobot

matriks V adalah :

$$V = \begin{bmatrix} w_{11} r_{11} & \cdots & w_{n1} r_{1n} \\ \vdots & \ddots & \vdots \\ w_m r_{m1} & \cdots & w_{nm} r_{nm} \end{bmatrix} \cdots (2)$$

3. Menentukan matriks solusi ideal positif dan matriks solusi ideal negatif

Solusi ideal positif dinotasikan dengan A+ dan solusi ideal negatif dinotasikan dengan A sebagai berikut :

Menentukan solusi ideal (+) dan (-)

$$A^{+} = \left\{\left(max_{V_{ij}}\right)\left(min_{V_{ij}}\middle| j \in J'\right), i = 1,2,3,\ldots m\right\} = \{V_1^{+}, V_2^{+}, \ldots V_m^{+}\} \ldots (3)$$
$$A^{-} = \left\{\left(max_{V_{ij}}\right)\left(min_{V_{ij}}\middle| j \in J'\right), i = 1,2,3,\ldots m\right\} = \{V_1^{-}, V_2^{-}, \ldots V_m^{-}\} \ldots (4)$$

Dimana :

v ij = elemen matriks V baris ke-i dan kolom ke-j

J = {j=1,2,3,...,n dan j berhubungan dengan benefit

criteria}

J '={j=1,2,3,...,n dan j berhubungan dengancost criteria}

4. Menentukan *S e p a r a s i*

Seaparation measure merupakan pengukuran jarak antara nilai setiap alternatif dengan matriks solusi ideal positif dan negative

Perhitungan Jarak untuk solusi ideal positif :

$$S_I^{+} = \sum_{J=l}^{n}\left(v_{ij-v_j^{+}}\right)^2 \text{dengan i = 1,2,3…, } m\ldots(5)$$

Dimana :

J = {j=1,2,3,...,n dan j merupakan benefit criteria}

J’ = {j=1,2,3,...,n dan j merupakan cost criteria}

Perhitungan Jarak untuk solusi ideal negatif :

$$S_I^{-} = \sum_{j=l}^{n}\left(v_{ij-v_j^{-}}\right)^2, \text{dengan i= 1,2,3…, } n\ldots(6)$$

Dimana :

J = {j=1,2,3,...,n dan j merupakan benefit criteria}

J’ = {j=1,2,3,...,n dan j merupakan cost criteria}

5. Menentukan kedekatan relatif terhadap solusi ideal yang akan diambil Kedekatan relatif dari alternatif A+ dengan solusi ideal A- direpresentasikan dengan :

$$C_1 = \frac{S_I^-}{S_I^- + s_I^+} \ldots (7)$$

dengan 0 < < 1 dan I = 1,2,3,…..m

6. Menentukan ranking alternative Alternatif dapat diranking berdasarkan urutan Ci*. Maka dari itu, alternatif terbaik adalah salah satu yang berjarak terpendek terhadap solusi ideal dan berjarak terjauh dengan solusi ideal negatif.

II.3.2. 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 (Martin, 1975). (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)

   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.5. Mengenal *Unified Modelling Language*(UML)

Unified Modeling Language (UML) adalah salah satu perkakas (tool) yang sangat bermanfaat untuk melakukan analisis dan perancangan sistem program aplikasi dalam konteks pemrograman berorientasi objek. UML pertama kali diperkenalkan pada tahun 1990 ketika Grady booch dan ivan Jacobson dan James Rumbaugh mulai mengadopsi ide-ide serta kemampuan-kemampuan tambahan dari masing-masing metodenya dan berusaha membuat metodologi terpadu yang kemudian dinamakan UML (Unified Modelling Language). UML digunakan sebagai suatu cara untuk mengkomunikasikan idenya kepada para pemrograman Nugroho adi, Rekayasa perangkat lunak menggunakan UML dan JAVA (Aris, dkk; 2015).

II.5.1. Fungsi *Unified Modelling Language* (UML)

Unified Modeling Language (UML) biasa digunakan untuk :

1. Menggambarkan batasan sitem dan fungsi-fungsi sistem secar umum, dibuat dengan *usecase* dan *actor.*
2. Menggambarkan kegiatan atau proses bisnis yang dilaksanakan secara umum, dibuat dengan *interaction diagrams*.
3. Menggambarkan representasi struktur *static* sebuah sistem dalam betuk *class diagrams*.
4. Membuat model *behavior* "yang menggambarkan kebiasaan atau sifat sebuah sistem" dengan *state transition diagrams*.

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

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

UML merupakan salah satu alat bantu yang sangat handal dalam bidang pengembangan sistem berorientas objek. Karena UML menyediakan bahasa pemodelan *visual* yang memungkinkan pengembang *system* membuat *blue print* atas visinya dalam bentuk yang baku. UML berfungsi sebagai jembatan dalam mengkomunikasikan beberap aspek dalam system melalui sejumlah elmen grafis yang bias dikombinasikan menjadi diagram. UML mempunyai banyak diagram yang dapat mengakomodasikan bebargai sudut pandang dari suatu perangkat lunak yang akan dibangun (Aris, dkk; 2015).

II.5.2. Tujuan Pemanfaatan *Unified Modelling Language* (UML)

Sejauh ini para pakar merasa lebih mudah dalam menganalisa dan mendesain atau memodelkan suatu sistem karena UML memiliki seperangkat aturan dan notasi dalam bentuk grafis yang cukup spesifik (Sugrue J. 2009).

Komponen atau notasi UML diturunkan dari 3(tiga) notasi yang telah ada sebelumnya yaitu *Grady Booch, OOD (Object-Oriented Design), Jim Rumbaugh, OMT (ObjectModellingTechnique),* dan *Ivar Jacobson OOSE (Object-Oriented Software Engineering).*

Pada UML terdiri atas tiga kategori, diantaranya : (Jurnal Informatika MulawarmanVol 6 No. 1 Febuari 2011)

1. Struktur Diagram

Menggambarkan elemen dari spesifikasi dimulai dengan kelas, obyek, dan hubungan mereka, dan beralih kedokumen arsitektur logis dari suatu sistem. Beberapa struktur diagram dalam UML terdiri atas :

a. *Class diagram*

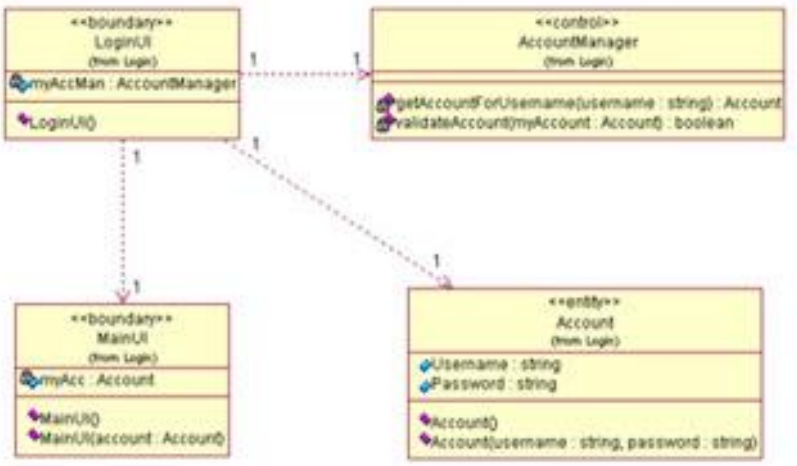
Class diagram menggambarkan struktur statis dari kelas dalam system anda dan menggambarkan atribut, operasi dan hubungan antara kelas. *Classdiagram* membantu dalam memvisualisasikan struktur kelas-kelas dari suatu sistem dan merupakan tipe diagram yang paling banyak dipakai. Selama tahap desain, *class diagram* berperan dalam menangkap struktur dari semua kelas yang membentuk arsitektur system yang dibuat.

Class memiliki tiga area pokok :

1). Nama (dan *stereotype*)

2). Atribut

3). Metoda



Gambar II.2. Notasi Class Diagram

(Sumber : Jurnal Informatika Mulawarman Vol 6 No. 1 Febuari 2011)

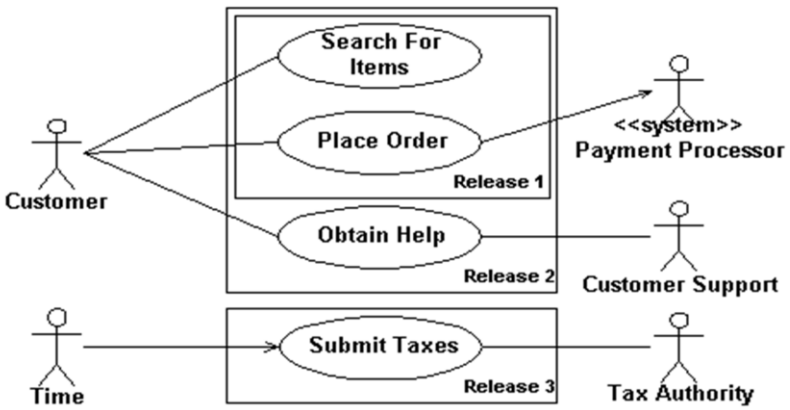
2. *Behavior Diagram*

Menggambarkan ciri-ciri behavior/metode/fungsi dari sebuah system atau *business process*.*Behavior diagram* dalam UML diantaranya terdiri atas :

a. *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 :

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

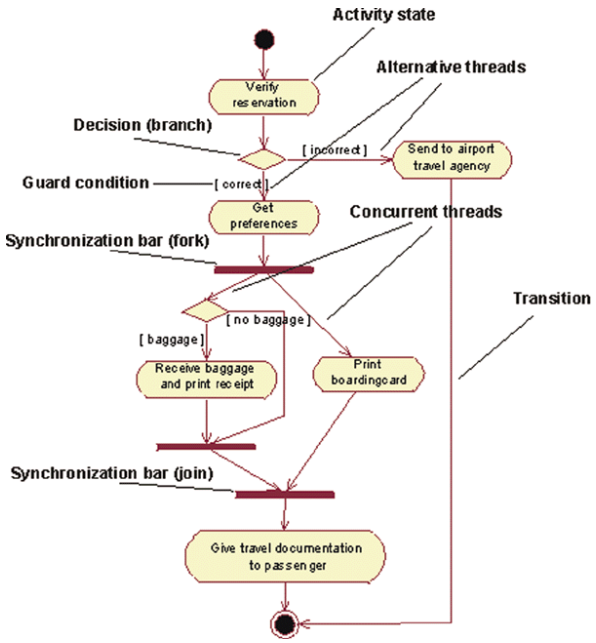


Gambar II.3. Notasi Use Case Diagram

(Sumber : Jurnal Informatika Mulawarman Vol 6 No. 1 Febuari 2011)

b. Activity diagram

Menggambarkan aktifitas-aktifitas, objek, *state*, *transisistate* dan *event*. Dengan kata lain kegiatan diagram alur kerja menggambarkan perilaku sistem untuk aktivitas.Berikut notasi *object diagram* dapat dilihat pada Gambar II.4.di bawah ini.



Gambar II.4. Notasi Activity Diagram

(Sumber : Jurnal Informatika Mulawarman Vol 6 No. 1 Febuari 2011)

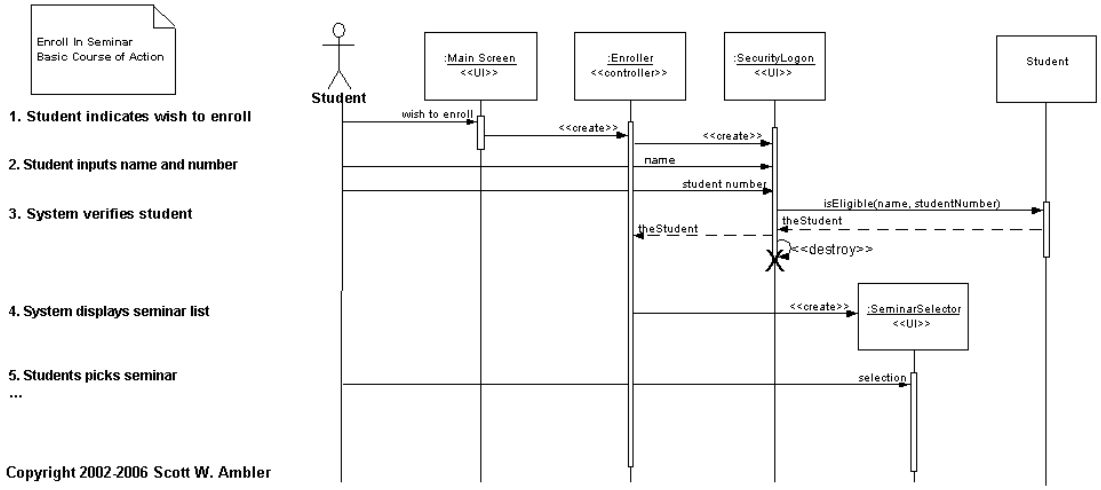
3. *Interaction Diagram*

Bagian dari *behavior diagram* yang menggambarkan interaksi objek. *Interaction diagram* dalam UML salah satunya adalah :

a. *Sequence diagram*

Sequence diagram menjelaskan interaksi objek yang disusun berdasarkan urutan waktu. 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.*



Gambar II.5.Notasi Sequence Diagram

(Sumber : Jurnal Informatika Mulawarman Vol 6 No. 1 Febuari 2011)

Untuk menggambarkan analisa dan desain diagram, UML memiliki seperangkat notasi yang akan digunakan ke dalam tiga kategori diatas yaitu struktur diagram, *behavior diagram*, dan interaction diagram. Berikut beberapa notasi dalam UML diantaranya :

1) *Actor,* menentukan peran yang dimainkan oleh useratau sistem lain yang berinteraksi dengan subjek.*Actor*adalah segala sesuatu yang berinteraksi langsung dengan system aplikasi komputer, sepertiorang, benda atau lainnya. Tugas actor adalah memberikan informasi kepada

sistem dan dapat memerintahkan sistem untuk melakukan sesuatu tugas.

2) *Class diagram,* Notasi utama dan yang paling mendasar pada diagram UML adalah notasi untuk mempresentasikan suatu class beserta dengan atribut dan operasinya. Class adalah pembentuk utama dari system berorientasi objek.
3) *Use Case* dan *use case specification, Use case* adalah deskripsi fungsi dari sebuah sistem perspektif pengguna. Use case bekerja dengan cara mendeskripsikan tipikal interaksi antara user (pengguna) sebuah system dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah system dipakai. Urutan langkah-langkah yang menerangkan antara pengguna dan sistem disebut skenario. Use case merupakan awal yang sangat baik untuk setiap fase pengembangan berbasis objek, design, testing, dan dokumentasi yang menggambarkan kebutuhan system dari sudut pandang di luar sistem. Perlu diingat bahwa use casehanya menetapkan apa yang seharusnya dikerjakan oleh sistem, yaitu kebutuhan fungsional sistem dan tidak untuk menentukan kebutuhan non fungsional, misalnya: sasaran kinerja, bahasa pemrograman dan lain sebagainya.
4) *Interaction, Interaction* digunakan untuk menunjukkan baik aliran pesan atau informasi antar obyek maupun hubungan antar obyek.
5) *Association, Association* menggambarkan navigasi antar *class (navigation*), berapa banyak obyek lain yang bias berhubungan

dengan satu obyek *(multiplicity antarclass)* dan apakah suatu class menjadi bagian dari class lainnya *(aggregation).*

II.6. *Microsoft Visual Basic 2010*

Visual Basic 2010 merupakan salah satu pemrograman terbaru yang dikeluarkan oleh *Microsoft,* yaitu *Microsoft Visual Studio 2010. Visual Studio* merupakan produk pemrograman andalan dari *Microsoft Corporation*, dimana 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 standart yang disertaka adalah *SQL Server 2008 Express*.

Visual Basic 2010 merupakan versi perbaikan dan pengembangan dari versi pendahulunya, yaitu *Visual Basic 2008*. Beberapa pengembangan yang terdapat didalamnya antara lain dukungan terhadap *library* terbaru dari *Microsoft*, yaitu *.Net FrameWork 4.0*, dukungan terhadap aplikasi berbasis *Cloud Computing*, serta perluasan dukungan terhadap database-database, baik *standalone* maupun *database server* (Wahana Komputer, 2011 : 2).

II.7. *Microsoft SQL Server 2008*

SQL Server 2008 adalah sebuah terobosan baru dari *Microsoft* dalam bidang database. *SQL Server* adalah DBMS *(Database Management System)* yang dibuat oleh *Microsoft* untuk ikut ber kecimpung dalam persaingan dunia

pengolahan data menyusul pendahulunya seperti IBM dan Oracle. *SQL Server 2008* dibuat pada saat kemajuan dalam bidang hardware sedemikian pesat. Oleh karena itu sudah dapat dipastikan bahwa *SQL Server 2008* membawa beberapa terobosan dalam bidang pengolahan dan penyimpanan data. *Microsoft* merilis *SQL Server 2008* dalam beberapa versi yang disesuaikan dengan segment-segment pasar yang dituju. Versi-versi tersebut adalah sebagai berikut.

Menurut cara pemrosesan data pada prosesor maka *Microsoft* mengelompokkan produk ini berdasarkan 2 jenis yaitu :

1. Versi 32-bit (x86), yang biasanya digunakan untuk komputer dengan *single prosesor* (Pentium 4) atau lebih tepatnya prosesor 32 bit dan sistem operasi Windows XP.
2. Versi 64-bit (x64), yang biasanya digunakan untuk komputer dengan lebih dari satu prosesor (Misalnya Core 2 Duo) dan sistem operasi 64 bit seperti Windows XP 64, Vista, dan Windows 7.

Sedangkan secara keseluruhan terdapat versi-versi seperti berikut ini:

1. Versi Compact, ini adalah versi "Tipis" dari semua versi yang ada. Versi ini seperti versi desktop pada *SQL Server 2000.* Versi ini juga digunakan pada handled drvice seperti Pocket PC, PDA, SmartPhone, Tablet PC.
2. Versi Express, ini adalah versi "Ringan" dari semua versi yang ada(tetapi versi ini berbeda dengan versi compact) dan paling cocok untuk latihan para pengembang aplikasi. Versi ini memuat *Express Manager standar*, integrasi dengan CLR dan XML (Widya, dkk; 2013).