BAB II
TINJAUAN PUSTAKA

II.1. Sistem

II.1.1. Konsep Dasar Sistem

Sistem merupakan kumpulan dari unsur atau elemen – elemen yang saling berkaitan / berinteraksi dan saling memengaruhi dalam melakukan kegiatan bersama untuk mencapai suatu tujuan tertentu (Asbon Hendra ; 2012 : 157-160).

Adapun pengertian sistem menurut para ahli :

1. **Menurut Jerry FithGerald**, sistem adalah suatu jaringan kerja dari prosedur – prosedur yang saling berhubungan dan berkumpul bersama – sama untuk melakukan kegiatan atau menyelesaikan suatu sasaran tertentu.
2. **Menurut Ludwig Von Bartalanfy**, sistem merupakan seperangkat unsur yang saling terikat dalam suatu antarrelasi 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 lainnya.

II.1.2. Syarat – Syarat Sistem

1. Sistem harus dibentuk untuk menyelesaikan tugas.
2. Elemen sistem harus mempunyai rencana yang ditetapkan.
3. Adanya hubungan di antara elemen sistem.
4. Unsur dasar dari proses (arus informasi, energi dan material) lebih penting daripada elemen sistem.
5. Tujuan organisasi lebih penting dari pada tujuan elemen.

II.1.3. Karakteristik Sistem

1. Komponen (*Component*)

   Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, bekerja sama membentuk satu kesatuan. Komponen – komponen sistem dapat berupa suatu subsistem atau bagian – bagian dari sistem. Setiap sistem, tidak peduli betapa pun kecilnya, selalu mengandung komponen – komponen atau subsistem – subsistem. Setiap subsistem mempunyai sifat – sifat dari sistem untuk menjalankan suatu fungsi tertentu dan mempengaruhi proses sistem secara keseluhan. Suatu sistem dapat mempunyai suatu sistem yang lebih besar yang disebut ***supra sistem***.

2. Batas Sistem (*Boundary*)

   Batas sistem merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lainnya atau dengan lingkungan luarnya. Batas 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 (*Environment*)

   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 menganggu operasi sistem.

4. Penghubung Sistem (*Interface*)

   Merupakan suatu penghubung antara satu 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*)

   Merupakan energi ysng dimasukkan ke dalam sistem. Masukan dapat berupak Masukan Perawatan (*Maintenance*) adalah energi yang dimasukkan supaya sistem tersebut dapat beroperasi. Maukan Sinyal (*Signal Input*) adalah energi yang diproses untuk didapatkan keluaran.

6. Keluaran Sistem (*Output*)

   Merupakan hasil dari energi yag 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. Pengelola Sistem (*Process*)

   Merupakan bagian yang memproses masukan untuk menjadi keluaran yang diinginkan.

8. Tujuan Sistem (*Goal*)

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

II.2. Sistem Pakar

II.2.1. Pengertian Sistem Pakar

Bidang sistem pakar merupakan penyelesain pendekatan yang sangat berhasil dan bagus untunk permasalahan *Aritificial Intelligence* (AI) klasik dari pemograman *intelligent* (cerdas). Sistem pakar (*expert system*) merupakan solusi *Aritificial Intelligence* (AI) bagi masalah pemograman pintar (*intelligent*). Profesor Edward Feigenbaum dari *Standard University* yang merupakan pionir dalam teknologi sistem pakar mendefinisikan sistem pakar sebagai sebuah program komputer pintar

(*intelligent computer program*) yang memanfaatkan pengetahuan (*knowledge*) dan prosedur inferensi (*inference procedure*) untuk memecahkan masalah yang cukup sulit hingga membutuhkan keahlian khusus dari manusia.

Dengan kata lain, sistem pakar adalah sistem komputer yang ditujukan untuk meniru semua aspek (*emulates*) kemampuan pengambilan keputusan (*decision making*) seorang pakar. Sistem pakar memanfaatkan secara maksimal pengetahuan khusus selayaknya seorang pakar untuk memecahkan masalah (Rika Rosnelly ; 2012 : 2).

II.2.2. Stuktur Sistem Pakar

Komponen yang terdapat dalam struktur sistem pakar adalah *knowledge base (rules), inference engine, working memory, explanation facility, knowledge acqusition facility, user interface* (Rika Rosnelly ; 2012 : 13-15).

1. *Knowledge Base* (Basis Pengetahuan)

   Basis pengetahuan mengandung pengetahuan untuk pemahaman, formulasi dan penyelesaian masalah. Komponen sistem pakar disusun atas dua elemen dasar, yaitu fakta dan aturan. Fakta merupakan informasi tentang objek dalam area permasalahan tertentu, sedangkan aturan merupakan informasi tentang bagaimana memperoleh fakta baru dari fakta yang telah diketahui.

2. *Inference Engine* (Mesin Inferensi)

   Mesin inferensi merupakan otak dari sebuah sistem pakar dan dikenal juga dengan sebutan *control structure* (struktur control) atau *rule interpreter*.

Komponen ini mengandung mekanisme pola pikir dan penalaran yang digunakan oleh pakar dalam menyelesaikan masalah.

3. *Working Memory*

   Bergunakan untuk menyimpan fakta yang dihasilkan oleh *inference engine* dengan penambahan *parameter* berupa derajat kepercayaan atau dapat juga dikatakan sebagail *global database* dari fakta yang digunakan *rule – rule* yang ada.

4. *Explanation Facility*

   Menyediakan kebenaran dari solusi yang dihasilkan kepada *user* (*reasoning chain*).

5. *Knowledge Acqusition Facility*

   Meliputi proses pengumpulan, pemindahan dan perubahan dari kemampuan pemecahan masalah seorang pakar atau sumber pengetahuan terdokumentasi ke program komputer, yang bertujuan untuk memperbaiki atau mengembangkan basis pengetahuan

6. *User Interface*

   Mekanisme untuk memberikan kesempatan kepada user dan sistem pakar untuk berkomunikasi. Antar muka menerima informasi dari pemakai dan mengubahnya ke dalam bentuk yang dapat diterima oleh sistem.

II.3. Narkoba

NAPZA (*Narkotika, Alkohol, Psikotropika, dan Zat Adiktif*) adalah bahan / zat / obat yang bila masuk kedalam tubuh manusia akan mempengaruhi tubuh terutama otak / susunan saraf pusat, sehingga menyebabkan gangguan kesehatan fisik, psikis dan fungsi sosialnya karena terjadi kebiasaan, ketagihan (*adiksi*) serta ketergantungan (*dependensi*) terhadap NAPZA. Istilah NAPZA umumnya digunakan oleh sektor pelayanan kesehatan yang menitik beratkan pada upaya penanggulangan dari sudut kesehatan fisik, psikis dan sosial. NAPZA sering disebut zat psikoaktif, yaitu zat yang bekerja pada otak, sehingga menimbulkan perubahan perilaku, perasaan dan pikiran. Dengan kenyataan yang demikian peredaran NAPZA di Indonesia semakin mudah dan murah untuk mendapatkannya oleh setiap kalangan masyarakat mulai dari anak-anak, pejabat, artis, mahasiswa bahkan oleh aparat penegak hukum, hal ini di sebabkan oleh keuntungan besar yang di janjikan dalam waktu yang singkat di balik bisnis haram ini.

Walaupun melanggar hokum dengan resiko sanksi yang berat seperti pidana mati, akan tetapi masih banyak orang yang bersedia menerima resiko ini demi keuntungan dari bisnis ini, sehingga pasokan barang-barang ini tidak hanya pada kota-kota besar di Indonesia, namun peredarannya juga sudah sampai ke kota-kota kecil bahkan sudah sampai di kecamatan dan desa-desa terpencil yang pendistribusiannya melalui jalur-jalur baik darat, laut maupun udara yang

terorganisasi sangat rapi dan rahasia, yang tanpa memperhatikan kepentingan moral, agama dan nasional (Dewi Anggreni, 2015).

II.4. Logika *Fuzzy*

II.4.1. Pengertian Logika *Fuzzy*

Logika *fuzzy* adalah suatu cara yang tepat untuk memetakan suatu ruang *input* ke dalam suatu ruang *output* (Rika Rosnelly, 2012 : 64). Istilah logika *fuzzy* yang didasarkan pada logika *Boolean* yang umum digunakan dalam komputasi. Secara ringkas, teorema *fuzzy* memungkinkan komputer “berpikir” tidak hanya dalam skala hitam putih (0 dan 1, mati atau hidup) tetapi juga dalam skala abu-abu. Dalam Logika *Fuzzy* suatu preposisi dapat direpresentasikan dalam derajat kebenaran (*truthfulness*) atau kesalahan (*falsehood*) tertentu (Miftahus Solihin, dkk, 2013).

II.3.2. Alasan Digunakannya Logika *Fuzzy*

Ada beberapa alasan mengapa orang menggunakan logika *fuzzy*, antara lain : (Rika Rosnelly, 2012 : 65).

1. Konsep logika *fuzzy* mudah dimengerti. Konsep *matematis* yang mendasari penalaran *fuzzy* sangat sederhana dan mudah dimengerti.
2. Logika *fuzzy* sangat fleksibel.
3. Logika *fuzzy* memiliki toleransi terhadap data – data yang tidak tepat.
4. Logika *fuzzy* mampu memodelkan fungsi – fungsi *nonlinear* yang sangat *kompleks*.

5. Logika *fuzzy* dapat membangun dan mengaplikasikan pengalaman – pengalaman para pakar secara langsung tanpa harus memulai proses pelatihan.
6. Logika *fuzzy* dapat bekerjasama dengan teknik – teknik kendali secara *konvensional.*
7. Logika *fuzzy* didasarkan pada bahasa alami.

II.4.3. Himpunan *Fuzzy*

Pada himpunan tegas (crisp), nilai keanggotaan suatu item x dalam suatu himpunan A, yang sering ditulis dengan $\mu_a[x]$, memiliki 2 kemungkinan, yaitu : (Rika Rosnelly, 2012 : 66-67).

1. Satu (1), yang berarti bahwa suatu item menjadi anggota dalam suatu himpunan, atau
2. Nol (0), yang berarti bahwa suatu item tidak menjadi anggota dala suatu himpunan.

II.4.4. Fungsi Keanggotaan

Fungsi keanggotaan adalah suatu kurva yang menunjukkan pemetaan titik – titik input data kedalam nilai keanggotaannya yang memiliki *interval* antara 0 sampai 1. Salah satu cara dapat digunakan untuk mendapatkan nilai keanggotaan adalah dengan melalui pendekatan fungsi (Rika Rosnelly, 2012 : 69-71).

II.4.4.1. Aturan *Rule* If Then *Fuzzy*

1. Aturan IF-THEN *fuzzy* adalah pernyataan IF-THEN dimana beberapa kata – kata dalam pernyataan tersebut ditentukan oleh fungsi keanggotaan.

2. Aturan produksi *fuzzy* adalah relasi *fuzzy* antara dua proposisi. Aturan tersebut dinyatakan dalam bentuk :

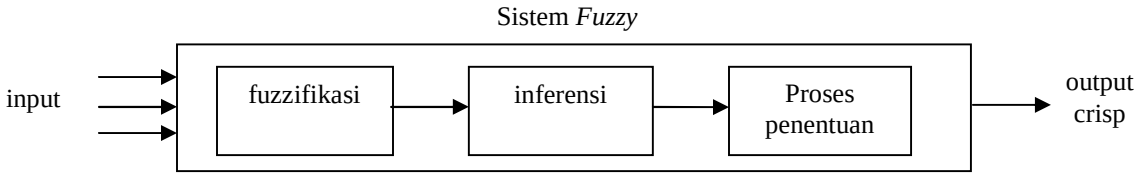
IF (proposisi *fuzzy* 1) THEN (proposisi *fuzzy* 2)

Disebut *anteceden*/ premis Disebut *consequent*/kesimpulan

3. Proposisi *fuzzy* adalah memiliki derajat kebenaran yang dinyatakan dalam suatu bilangan dalam bentuk *interval* [0,1], dimana benar dinyatakan oleh nilai 1 dan salah dinyatakan oleh 0.

II.4.4.2. Tahapan Membangun Sistem *Fuzzy*

Tahapan membangun sistem *fuzzy* tergantung metode yang digunakan, karena banyak teori/metode untuk membangun sistem *fuzzy*. Namun secara garis besar dapat disimpulkan sebagai berikut : (Rika Rosnelly, 2012 : 71-72).



Gambar II.1. Tahapan Membangun Sistem *Fuzzy*
(Sumber : Rika Rosnelly, 2012)

Keterangan :

1. ***Fuzzifikasi***

   Mengambil masukan nilai *crisp* dan menentukan derajat dimana nilai – nilai tersebut menjadi anggota dari setiap himpunan *fuzzy* yang sesuai membuat fungsi keanggotaan.

2. ***Inferensi***

    a. Mengaplikasikan aturan pada masukan *fuzzy* yang dihasilkan pada proses *fuzzifikasi.*

    b. Mengevaluasi tiap aturan dengan masukan yang dihasilkan dari proses *fuzzifikasi* dengan mengevaluasi hubungan atau derahat keanggotaan *anteceden*/premis setiap aturan.

    c. Derajat keanggotaan/ nilai kebenaran dari premis digunakan untuk menentukan nilai kebenaran bagi *consequent*/kesimpulan.

3. **Proses Penentuan *Output Crisp***

    Tergantung teori/metode yang digunakan.

II.5. Metode Tsukamoto

Pada metode Tsukamoto, setiap aturan direpresentasikan menggunakan himpunan-himpunan *fuzzy*, dengan fungsi keanggotaan yang monoton. Untuk menentukan nilai *output crisp* atau hasil yang tegas (Z) dicari dengan cara mengubah input (berupa himpunan *fuzzy* yang diperoleh dari komposisi aturan-aturan *fuzzy*) menjadi suatu bilangan pada domain himpunan *fuzzy* tersebut. Cara ini disebut dengan metode defuzzifikasi (penegasan). Metode *defuzzifikasi* yang digunakan dalam metode Tsukamoto adalah metode *defuzzifikasi* rata-rata terpusat (*Center Average Defuzzyfier*) (Miftahus Solihin, dkk, 2013).

Misal ada 2 variabel input, var-1(x) dan var-2(y) serta 1 variabel output var-3(z), dimana var-1 terbagi atas 2 himpunan yaitu A1 dan A2 dan var-2 terbagi atas himpunan B1 dan B2. Sedangkan var-3 juga terbagi atas 2 himpunan yaitu C1 dan C2

Ada dua aturan yang digunakan yaitu:

R1] IF (x is A1) and (y is B2) THEN (z is C1)

[R2] IF (x is A2) and (y is B1) THEN (z is C2)

II.6. Basis Data (*Database*)

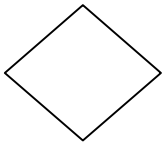
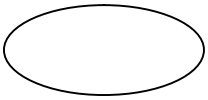
Database atau sering juga disebut basis data adalah sekumpulan informasi yang disimpan dalam komputer secara sistematik dan merupakan sumber informasi yang dapat diperiksa menggunakan suatu program komputer.*Database* berfungsi untuk menyimpan informasi atau data.Untuk mengelola *database* diperlukan *software* yang sering disebut dengan DBMS (*Database Management System*). Dengan DBMS pengguna atau *user* dapat membuat, mengelola, mengontrol, dan mengakses *database* dengan mudah, praktis dan efisien.

II.7. Entity Relationship Diagram

Entity Relationship Diagram (ERD) untuk mendokumentasikandat perusahaan dengan mengindentifikasi jenis entitas (*entity*) dan hubungannya. ERD merupakan suatu model jaringan yang menggunakan susunan data yang disimpan pada sistem secara abstrak. ERD juga menggambarkan hubungan antara sutu entitas dengan yang memiliki sejumlah atribut dengan entitas yanglain dalam suatu sistem

yang terintegrasi. ERD digunakan oleh perancang sistem untuk memodelkan data yang nantinya akan dikembangkan menjadi basis data (*database*). Model data ini juga dapat membantu pada saat melakukan analisis dan perancangan bsis data, karena model data ini akan menunjukkan bermacam-macam data yang dibutuhkan dan hubungan antar data. ERD ini juga merupakan model konseptual yang dapat mendeskripsikan hubungan antara file yang digunakan untuk memodelkan struktur data serta hubungan antar data (Yakub, 2012 : 60).

Tabel II.1. Simbol – simbol *Entity Relation Diagram*

Simbol	Keterangan
	Etitas, yaitu kumpulan dari objek yang dapat di identifikasikan secara unik
	Relasi, yaitu hubungan yang terjadi antara satu atau lebih entitas, jenis hubungan antara lain; satu ke satu, satu ke banyak, banyak ke banyak
	Atribut, yaitu karakteristik dari entity atau relasi yang merupakan penjelasan detail tentang entitas
	Hubungan antara entity dengan atributnya dan himpunan entitas dengan himpunan relasinya

(Sumber : Yakub, 2012)

II.8. Normalisasi

II.8.1. Pengertian Normalisasi

Normalisasi merupakan salah satu cara pendekatan atau teknik yang digunakan dalam membangun desain lojik basis data *relation* dengan menerapkan sejumlah aturan dan kriteria standar. Tujuan dari normalisasi adalah untuk menghasilkan struktur tabel yang normal atau baik. Teknik normalisasi adalah upaya agar desain lojik tabel – tabel berada dalam ”normal form” (bentuk normal) yang dapat didefinisikan dengan menggunakan ketergantungan fungsi (Yakub, 2012 ;70).

II.8.2. Bentuk Normalisasi

Bentuk normal adalah suatu aturan yang dikenakan pada relasi-relasi atau tabel-tabel dalam basis data dan harus dipenuhi oleh relasi atau tabel tersebut pada level-level normalisasi. Suatu relasi dikatakan dalam bentuk normalisasi diantaranya adalah bentuk tidak normal (*unnormalized*), normalisasi pertama (*1st normal form*), normalisasi ke dua (*2st normal form*), dan normalisasi ke tiga (*3rd normal form*) (Yakub, 2012 ;71).

II.8.3. Tidak Normal

Bentuk tidak normal merupakan kumpulan data yang direkam dan tidak ada keharusan dengan mengikuti suatu format tertentu. Pada bentuk tidak normal terdapat *repeating group* sehingga pada kondisi seperti ini akan menjadi permasalahan dalam melakukan manipulasi data (*insert, update,* dan *delete anomalies*). *Update anomalies* terjadi apabila ada perubahan pada sejumlah data yang mubazir pada suatu tabel

tetapi tidak seluruhnya diubah. *Insert anomalies,* terjadi apabila pada saat penambahan hendak dilakukan, ternyata ada elemen data yang masih kosong, dan elemen data tersebut justru menjadi kunci. *Delete, anomalies* terjadi apabila suatu baris (*record*) yang tidak terpakai dihapus, dan sebagainya akibat data lainnya yang hilang (Yakub, 2012 ;71).

II.8.4. Nomalisasi pertama

Dalam *relational database* tidak dipernankan adanya *repeating group* karena dapat berdampak terjadinya *anomalies*. Oleh karena itu tahap unormalisasi akan menghasilkan bentuk normal pertama. Normalisasi ke satu, suatu relasi atau tabel memenuhi normal ke satu jika dan hanya jika setiap atribut dari relasi tersebut hanya memiliki nilai tunggal (*scalar value*) dalam satu baris atau r*ecord*. Tiap *field* hanya satu pengertian, bukan merupakan kumpulan data yang mempunyai arti mendua dan tidak ada set atribut yang berulang-ulang atau atribut bernilai ganda. Pada saat tabel sebelumnya, contoh data belum ternormalisasi sehingga dapat diubah kedalam bentuk normal pertama dengan cara membuat setiap baris berisi kolom dengan jumlah yang sama dan setiap kolom hanya mengandung satu nilai. Bentuk normalisasi pertama, bentuk normal pertama ini mempunyai ciri yaitu setiap data dibentuk file datar atau rata (*flat file*), data dibentuk dalam satu *record* demi satu *record* dan nilai dari *field-field* berupa "*atomik value*" artinya berupa nilai yang tidak dapat dibagi – bagi lagi (Yakub, 2012 ;71-72).

II.8.5. Normalisasi Ke dua

Dalam perancangan basis data relational tidak diperkenankan adalah *partial functional dependency* kepada *primary key*, karena dapat berdampak terjadi *anomalies*. Oleh karena itu tahap normalisasi pertama akan menghasilkan bentuk normal ke dua yang dapat didefinisikan sebagai berikut :

Normalisasi ke dua, suatu relasi memenuhi relasi ke dua jika dan hanya jika relasi tersebut memenuhi normal pertama dan setiap atribut yang bukan kunci (*non key*) bergantung secara fungsional (FD) secara utuh kepada kunci utama (*primary key*). Bentuk normal ke dua ini mempunyai syarat yaitu bentuk data yang telah memenuhi kriteria bentuk normal ke satu. Atribut bukan kunci haruslah bergantung secara fungsi pada kunci utama (*primary key*) sehingga untuk membentuk normal ke dua haruslah sudah ditentukan kunci-kunci *field* (Yakub, 2012 ;72).

II.8.6. Normalisasi Ke tiga

Dalam perancangan basis data relational tidak diperkenankan adanya *transitive dependency* karena dapat berdampak terjadinya *anomalies.* Oleh karena itu harus dilakukan normalisasi tahap ke tiga yang dapat didefinisikan sebagai berikut :

Normalisasi ke tiga, suatu relasi memenuhi normal ketiga jika dan hanya jika relasi tersebut memenuhi normal ke dua dan setiap atribut yang bukan kunci (*non key*) tidak mempunyai *transitive functional dependency* kepada kunci utama (*primary key*). Bentuk normal ke tiga ini relasi haruslah dalam bentuk normal ke dua dan semua atribut bukan kunci utama tidak punya hubungan transitif. Artinya setiap

atribut bukan kunci harus bergantung hanya pada *primary key* secara keseluruhan, dan bentuk normalisasi ke tiga sudah didapat tabel yang optimal (Yakub, 2012 ;72).

II.9. UML (*Unified Modelling Langguage)*

II.9.1. Pengertian UML (*Unified Modelling Langguage)*

Menurut Chonoles dalam Widodo dan Herlawati (2011:6)" Sebagai bahasa, berarti UML memiliki sintaks dan semantic". Ketika kita membuat model menggunakan konsep UML ada aturan – aturan yang harus diikuti. Bagaimana elemen pada model – model yang kita buat berhubungan satu dengan lainnya harus mengikuti standar yang ada. UML bukan hanya sekedar diagram, tetapi juga menceritakan konteksnya.

Beberapa litelatur menyebutkan bahwa UML menyediakan Sembilan jenis diagram, yang lain menyebutkan delapan karena ada beberapa diagram yang digabung, misal diagram komunikasi, diagram urutan dan diagram pewaktuan digabung menjadi diagram interaksi. Namun demikian model-model itu dapat dikelompokan bedasarkan sifatnya yaitu statis atau dinamis (Mia Rosmiati, 2015).

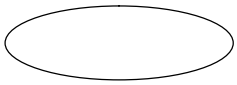
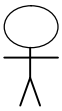
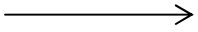
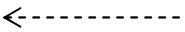
II.9.2. Diagram UML (*Unified Modelling Langguage)*

1. *Use case* Diagram

Use case diagram 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. Dapat dikatakan *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem informasi dan

siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Simbol-simbol yang digunakan dalam *use case* diagram, yaitu : (Windu Gata & Grace Gate 2013 : 4-5).

Tabel II.2. Simbol *Use Case*

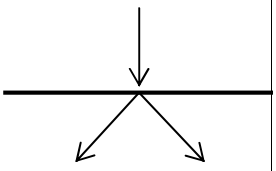
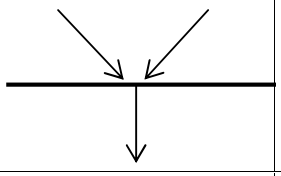
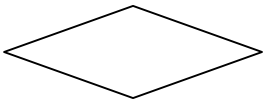
Gambar	Keterangan
	*Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukan pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama *use case*.
	Aktor adalah *abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan aktor, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa muncul dalam beberapa peran. Perlu dicatat bahwa aktor berinteraksi dengan *use case*, tetapi tidak memiliki control terhadap *use case*.
	Asosiasi antara aktor dan *use case*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengidikasikan aliran data.
	Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengidinkasikan bila aktor berinteraksi secara pasif dengan sistem.
	*Include*, merupakan di dalam *use case* lain (*required*) atau pemanggilan *use case* oleh *use case* lain, contohnya adalah pemanggilan sebuah fungsi program.
	*Extend*, merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.

(Sumber : Windu Gata & Grace Gata, 2013)

2. Diagram Aktivitas (*Activity Diagram*)

Activity Diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Simbol-simbol yang digunakan dalam *activity diagram,* yaitu : (Windu Gata & Grace Gate 2013 : 6-7).

Tabel II.3. Simbol *Activity Diagram*

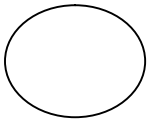
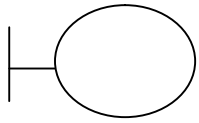
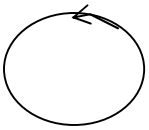
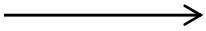
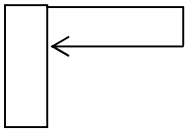
Gambar	Keterangan
	*Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	*End point*, akhir aktifitas.
	*Activites*, menggambarkan suatu proses/kegiatan bisnis.
	*Fork* (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan pararel menjadi satu.
	*Join* (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi.
	*Decision Points*, menggambarkan pilihan untuk pengambilan keputusan, *true*, *false.*
	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa.

(Sumber : Windu Gata & Grace Gata, 2013)

3. Diagram Urutan (*Sequence Diagram*)

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Simbol-simbol yang digunakan dalam *sequence diagram*, yaitu : (Windu Gata & Grace Gate 2013 : 7-8).

Tabel II.4. Simbol *Sequence Diagram*

Gambar	Keterangan
	*EntityClass*, merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data.
	*Boundary Class*, berisi kumpulan kelas yang menjadi *interface* atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan formentry dan *form* cetak.
	*Control class*, suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek.
	*Message*, simbol mengirim pesan antar *class*.
	*Recursive*, menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri.
	*Activation, activation* mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivitas sebuah operasi.

	Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation*.

(Sumber : Windu Gata & Grace Gata, 2013)

4. *Class Diagram* (Diagram Kelas)

Merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggng jawab entitas yang menentukan perilaku sistem.

Class diagram juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan. *Class diagram* secara khas meliputi: Kelas (*Class*), Relasi, *Associations*, *Generalization* dan *Aggregation*, Atribut (*Attributes*), Operasi (*Operations/Method*), *Visibility,* tingkat akses objek eksternal kepada suatu operasi atau atribut. Hubungan antar kelas mempunyai keterangan yang disebut dengan *multiplicity* atau kardinaliti (Windu Gata & Grace Gate 2013 : 8-9).

Tabel II.5. *Multiplicity Class Diagram*

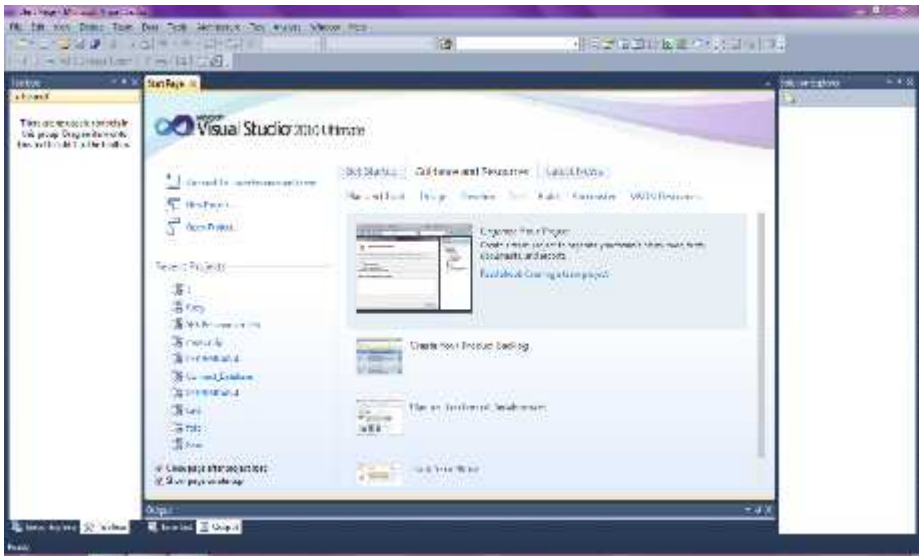
Multiplicity	**Penjelasan**
1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1 atau lebih
1..*	1 atau lebih
0..1	Boleh tidak ada, maksimal 1
n..n	Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimum 4

(Sumber : Windu Gata & Grace Gata, 2013)

II.10. *Microsoft Visual Studio* 2010

II.10.1. Pengertian *Microsoft Visual Studio* 2010

Visual Studio 2010 merupakan sebuah *Integrated Development Environtment* (IDE) atau lingkungan kerja yang digunakan untuk membangun aplikasi .NET dengan muah. *Visual Studio Profesional* 2010 menyediakan berbagai *tool* yang lengkap bari para pengembang untuk membangun aplikasi yang berjalan di .Net Framework. Berbagai tool, antara lain *tool Toolbox* yang berisi komponen visual, sehingga anda tinggal *drag and drop* komponen *Visual Studio* 2010 akan menuliskan kode untuk anda (wahana computer, 2012 ; 7).



Gambar II.2. *Microsoft Visual Studio* 2010

II.11. *SQL Server* 2008

SQL Server 2008 adalah sebua RDBMS (*Relational Database Management System*) yang sangat powerful dan telah terbukti kekuatannya dalam mengolah data.

Dalam versi terbarunya ini, *SQL Server* 2008 memiliki banyak fitur yang bisa dihandalkan untuk meningkatkan performa database. *SQL Server* 2008 memiliki suatu GUI (*Graphic User Interface*) yang kita gunakan untuk melakukan aktivitas sehari hari berkaitan dengan database,seperti menulis T-SQL, melakukan beckup dan restore database, melakukan security database terhadap aplikasi, dan sebagainya.

Pada GUI tersebut kita bisa melakukan settingan terhadap *SQL Server* untuk berkerja lebih optimal.Settingan juga bisa dilakukan menggunakan script untuk memudahkan developer mengubah *Setting Opsions* pada *SQL Server* 2008 (Ruslan, 2013).