BAB II
TINJAUAN PUSTAKA

II.1. Pengertian Sistem

Sistem merupakan sekumpulan elemen-elemen yang saling terintegrasi serta melaksanakan fungsinya masing-masing untuk mencapai tujuan yang telah ditetapkan. (Sulindawati dan Fathoni; 2010 : 375)

Karekteristik sistem terdiri dari :

1. Komponen sistem

   Suatu sistem terdiri dari sejumlah komponen yang saling berintegrasi, yang artinya saling berkerja sama membentuk suatu kesatuan.

2. Batasan sistem

   Batasan merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lainnya atau lingkungan luarnya.

3. Lingkungan luar sistem

   Lingkungan luar dari suatu sistem adalah apapun diluar batas dari sistem yang mempengaruhi operasi sistem.

4. Penghubung sistem

   Penghubung merupakan media penghubung antara satu subsistem dengan subsistem lainnya.

5. Masukan sistem

   Masukan sistem adalah energi yang dimasukkan ke dalam sistem. Masukan dapat berupa masukan perawatan *(maintenance input)* dan masukan sinyal *(signal input).*

6. Keluaran sistem

   Keluaran sistem adalah hasil energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna dan sisa pembuangan.

7. Pengolah sistem

   Suatu sistem dapat mempunyai suatu bagian pengolah atau sistem itu sendiri sebagai pengolahnya.

8. Sasaran sistem

   Suatu sistem mempunyai tujuan *(goal)* atau sasaran (*objective*). Kalau suatu sistem tidak mempunyai sasaran, maka operasi sistem tidak ada gunanya.

II.1.1. Pengertian informasi

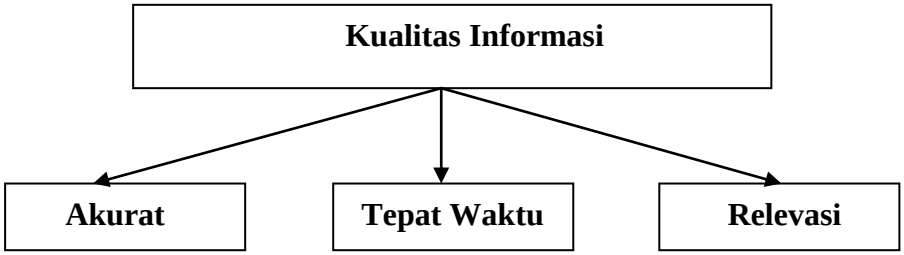
Informasi adalah data yang diolah menjadi suatu bentuk yang berarti bagi penerimanya dan bermanfaat dalam pengambilan keputusan pada saat sekarang atau yang akan datang. Informasi juga merupakan fakta-fakta atau data yang telah diproses sedemikian rupa atau mengalami proses transformasi data sehingga berubah bentuk menjadi informasi. (Sulindawati dan Fathoni; 2010 : 3)

Kualitas dari suatu informasi tergantung pada tiga hal yaitu :

1. Akurat, berarti informasi harus bebas dari kesalahan dan tidak bias atau menyesatkan.
2. Tepat pada waktunya, berarti informasi yang pada penerima tidak boleh terlambat. Informasi yang sudah usang tidak akan mempunyai nilai lagi. Karena informasi merupakan landasan dalam pengmbilan keputusan.

3. Relevan, berarti informasi tersebut mempunyai mamfaat untuk pemakainya, relevasi informasi untuk tiap-tiap orang satu dengan yang lainnya berbeda.

Adapun Pilar kualitas informasi dapat dilihat pada Gambar II.1.



Gambar II.1. Pilar Kualitas Informasi

(Sumber : Sulindawati, Fathoni; 2010 : 3)

II.1.2. Pengertian Sistem Informasi

Sistem informasi dapat diartikan sebagai suatu sistem didalam organisasi yang merupakan kombinasi dari orang-orang, fasilitas, teknologi, media, prosedur-prosedur, dan pengendalian yang ditunjukkan untuk mendapatkan jalur kombinasi yang penting. (Sulindawati dan Fathoni; 2010 : 4)

Di dalam suatu sistem informasi terdapat beberapa komponen-komponen, yaitu :

1. Perangkat keras *(hardware)* : mencangkup piranti-piranti seperti monitor, printer, scanner, keyboard, dan mouse.
2. Pengkat lunak *(software)* atau program : sekumpulan intruksi yang memungkinkan perangkat keras untuk dapat memperoses data.
3. Prosedur : sekumpulan aturan yang dipakai untuk mewujudkan pemrosesan data dan pembangkitan keluaran yang dikehendaki.

4. Orang : semua pihak yang bertanggung jawab dalam pengembangan sistem informasi, pemrosesa, dan penggunaan sistem informasi.
5. Basis data *(database)* : sekumpulan tabel, hubungan, dan lain-lain yang berhubungan dengan penyimpanan data.
6. Jaringan komputer dan komunikasi data : sistem penghubung yang memungkinkan suatu sumber dipakai secara bersama atau diakses oleh sejumlah pemakai.

II.2. Sistem Pakar

Sistem pakar adalah sistem komputer yang ditunjukan 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; 2012 : 2)

Sistem pakar *(Expert System)* adalah program berbasis komputer yang menyediakan solusi-solusi dengan kualitas pakar untuk problema-problema dalam suatu *domain* yang spesifik. Sistem pakar merupakan program komputer yang meniru proses pemikiran dan pengetahuan pakar dalam menyelesaikan suatu masalah tertentu. (Helmi, Iwan; 2012 : 188)

II.2.1. Defenisi dan Tujuan Sistem Pakar

Pengembangan kecerdasan buatan adalah sistem pakar yang menggabungkan pengetahuan dan penelusuran data untuk memecahkan masalah yang secara normal memerlukan keahlian manusia. Tujuan sistem pakar sebenarnya bukan untuk menggantikan peran manusia, tetapi untuk

mensubtitusikan pengetahuan manusia ke dalam sistem, sehingga dapat digunakan oleh banyak orang. (Rika; 2012 : 3)

Ada banyak manfaat yang dapat diperoleh dengan memanfaatkan keahlian didalam sistem pakar, antara lain :

1. Orang awam yang bukan pakar bisa meningkatkan kemampuan mereka dalam memecahkan masalah.
2. Sistem pakar sebagai asisten yang berpengetahuan.
3. Penghematan waktu dalam menyelesaikan masalah yang kompleks.
4. Memberikan penyederhanaan solusi kasus-kasus yang kompleks dan berulang-ulang.

II.2.2. Ciri dan Karakteristik Sistem Pakar

Ada berbagai ciri-ciri dan karakteristik yang membedakan sistem pakar dengan sistem yang lain. Ciri-ciri dan karakteristik ini menjadi pedoman utama dalam pengembangan sistem pakar. Ciri-ciri dan karakteristik yang dimaksud adalah sebagai berikut :

1. Pengetahuan sistem pakar merupakan suatu konsep, bukan berbentuk numeris.
2. Informasi dalam sistem pakar tidak selalu lengkap, subyektif, tidak konsisten sehingga keputusan diambil bersifat tidak pasti.
3. Kemungkinan solusi sistem pakar terhadap suatu permasalahan adalah bervariasi.
4. Perubahan atau pengembangan pengetahuan dalam sistem pakar terjadi setiap saat bahkan sepanjang waktu.

5. Pandangan dan pendapat setiap pakar tidaklah selalu sama.
6. Keputusan merupakan bagian terpenting dalam sistem pakar. Sistem pakar harus memberikan solusi yang akurat berdasarkan masukan pengetahuan meski solusinya sulit.

Berikut ini merupakan hubungan antara pengguna dan fungsi sistem pakar, seperti pada tabel II.1 berikut :

Tabel II.1 Hubungan antara pengguna dan fungsi sistem pakar

Pengguna	Kepentingan	Fungsi Sistem Pakar
Klien bukan pakar	Mancari saran/nasehat	Konsultan atau penasehat
Mahasiswa	Belajar	Instruktur
Pembangun sistem	Memperbaiki/menambah Basis pengetahuan	Rekan (partner)
Pakar	Membantu analisis rutin atau proses komputasi, Mencari (mengklasifikasi) informasi, alat bantu diagnosa	Rekan kerja atau asisten

(Sumber : Andi Offset; 2012 : 12)

II.2.3. Kelebihan Sistem Pakar

Sistem pakar memiliki beberapa fitur menarik yang merupakan kelebihannya, seperti :

- Meningkatkan ketersediaan *(increased availability)*. Kepakaran atau keahlian menjadi tersedia dalam suatu komputer.
- Mengurangi biaya *(reduce cost)*. Biaya yang diperlukan untuk

menyediakan keahlian per satu orang user menjadi berkurang.

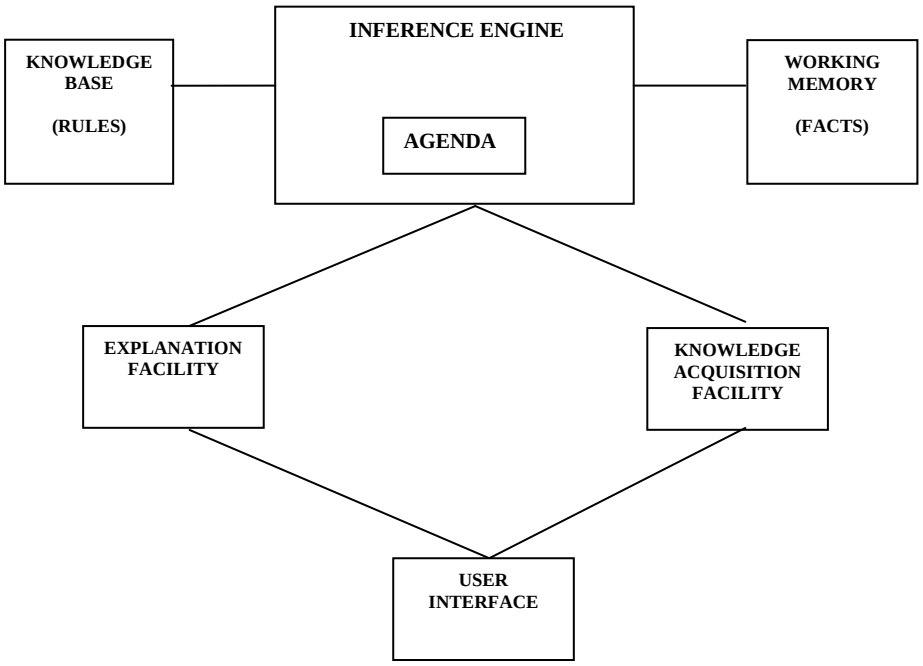
- Mengurangi bahaya *(reduce danger).* Sistem pakar dapat digunakan dilingkungan yang mungkin berbahaya bagi manusia.
- Permanen *(permanence).* Sistem pakar dan pengetahuan yang terdapat di dalamnya bersifat lebih permanen dibandingkan manusia.
- Keahlian mutipel *(multiple expertise).* Pengetahuan dari beberapa pakar dapat dimuat ke dalam sistem dan bekerja secara simultandan kontinyu menyelesaikan suatu masalah setiap saat.
- Meningkatkan kehandalan *(increased reliability).* Sistem pakar meningkatkan kepercayaan dengan memberikan hasil yang benar
- Penjelasan *(explanation).* Sistem pakar dapat menjelaskan detail proses penalaran yang dilakukan sehingga mencapai suatu kesimpulan.
- Respon yang cepat *(fast response).* Respon yang cepat atau *real time* diperlukan pada beberapa aplikasi.
- Stabil tidak emosional dan memberikan respon yang lengkap setiap saat.
- Pembimbing pintar *(intelligent tutor).*
- Basis data cerdas *(intelligent database).*

II.2.4. Kelemahan Sistem Pakar

1. Biaya yang diperlukan untuk membuat dan memeliharanya sangat mahal.
2. Sulit dikembangkan
3. Sistem pakar tidak 100% bernilai benar.

II.2.5. Struktur Sistem Pakar

Adapun struktur sistem pakar dapat dilihat pada Gambar II.2 berikut :



Gambar II.2 Struktur Sistem Pakar

(Sumber : Andi Offset; 2012 : 13)

Adapun penjelasan dari gambar II.2 sebagai berikut :

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 cara bagaimana memperoleh fakta baru dari fakta yang telah diketahui. Pada struktur sistem pakar diatas, *knowledge base* disini untuk menyimpan

pengetahuan dari pakar berupa rule / aturan (if <kondisi> then <aksi> atau dapat juga disebut condition-action rules).

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* (dalam sistem pakar berbasis kaidah). Komponen ini mengandung mekanisme pola pikir dan penalaran yang digunakan oleh pakar dalam menyelesaikan suatu masalah. Mesin inferensi disini adalah *procesor* pada sistem pakar yang mencocokkan bagian kondisi dari rule yang tersimpan di dalam *knowledge base* dengan fakta yang tersimpan di *working memory.*

3. ***Working Memory***

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

4. Explanation Facility

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

5. Knowledge Acquistion Facility

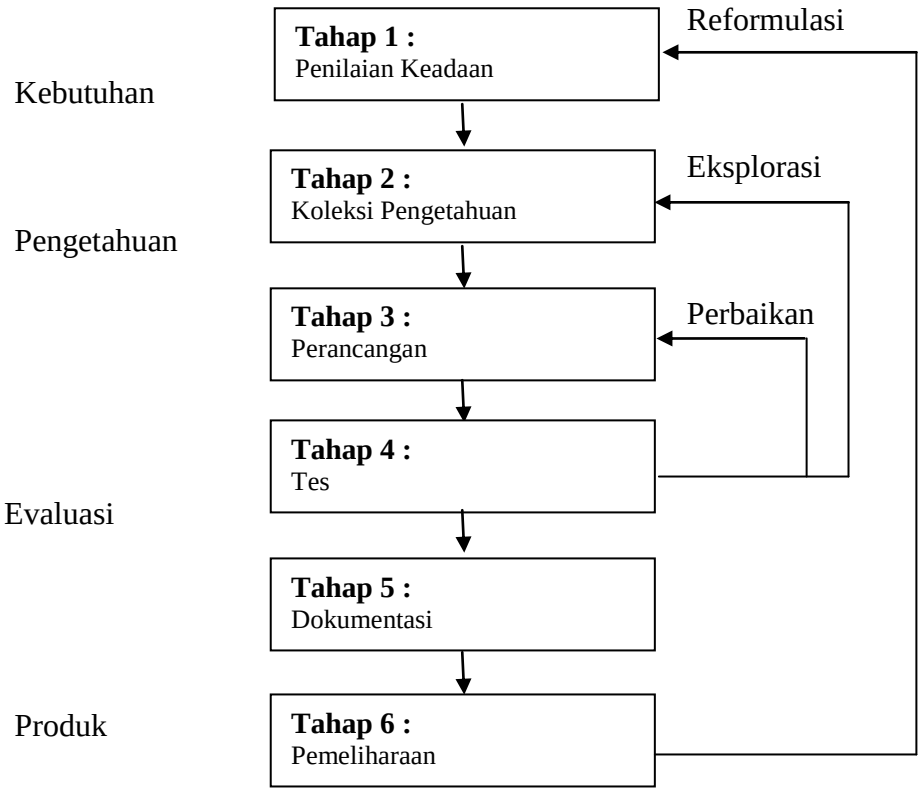
Meliputi proses pengumpulan, pemindahan dan perubahan dari kemampuan pemecahan masalah seorang pakar atau sumber pengetahuan terdokumentasi ke program komputer, yang bertujuan untuk memperbaiki atau mengembangkan basis pengetahuan.

6. User Interface

Mekanisme untuk memberi kesempatan kepada user dan sistem pakar untuk berkomunikasi. Antar muka menerima informasi dari pemakai dan mengubahnya ke dalam bentuk yang dapat diterima oleh sistem. Selain itu antarmuka menerima informasi dari sistem dan menyajikan ke dalam bentuk yang dapat dimengerti oleh pemakai.

II.2.6. Tahapan Pengembangan Sistem Pakar

Seperti layaknya pengembangan perangkat lunak, pada pengembangan sistem pakar inipun diperlukan beberapa tahap seperti terlihat pada gambar II.3



Gambar II.3 Tahapan Pengembangan Sistem Pakar
(Sumber : Andi Offset; 2012 : 95)

Secara garis besar pengembangan sistem pakar pada gambar II.2 adalah sebagai berikut :

1. Mengidentifikasi masalah dan kebutuhan. Mengkaji situasi dan memutuskan dengan pasti tentang masalah yang akan dikomputerisasi dan apakah dengan sistem pakar bisa lebih membantu atau tidak.
2. Menetukan masalah yang cocok. Menghasilkan solusi mental bukan fisik, artinya hanya memberikan anjuran tidak bisa melakukan aktivitas fisik seperti merasakan.
3. Mempertimbangkan alternatif. Menggunakan 2 alternatif, yaitu menggunakan sistem pakar atau komputer tradisional.
4. Menghitung pengambilan investasi.termasuk diantaranya biaya pembuatan sistem pakar, biaya pemeliharaan, dan biaya training.
5. Memilih alat pengembangan. Bisa digunakan software pembuat sistem pakar (seperti : SHELL) atau dirancang dengan bahasa pemograman sendiri.
6. Rekayasa pengetahuan. Perlu dilakukan penyempurnaan terhadap aturan-aturan yang sesuai.
7. Merancang sistem. Bagian ini termasuk pembuatan *prototype* apabila sistem yang telah ada sudah sesuai dengan keinginan.
8. Melengkapi pengembangan. Termasuk pengembangan *prototype* apabila sistem yang telah ada sudah sesuai dengan keinginan.
9. Menguji dan mencari kesalahan sistem.

10. Memelihara sistem. Dalam hal ini harus dilakukan pembaharuan pengetahuan, mengganti pengetahuan yang sudah ketinggalan agar bisa lebih baik lagi dalam menyelesaikan masalah.

II.2.7. Kaidah Produksi

Kaidah produksi dituliskan dalam bentuk pernyataan IF-THEN (Jika-Maka). Pernyataan ini menghubungkan bagian premis (IF) dan bagian kesimpulan (THEN) yang dituliskan dalam bentuk :

IF [premis] THEN [konklusi]

Beberapa struktur kaidah IF-THEN yang menghubungkan obyek atau atribut sebagai berikut :

IF Premis THEN Konklusi

IF Masukan THEN Keluaran

IF Kondisi THEN Tindakan

IF Antesenden THEN Konsekuen

IF Data THEN Hasil

IF Tindakan THEN Tujuan

Premis mengacu pada fakta yang harus benar sebelum konklusi tertentu dapat diperoleh. Masukan mengacu pada data yang harus tersedia sebelum keluaran dapat diperoleh. Kondisi mengacu pada informasi yang harus tersedia sehingga sebuah hasil dapat diperoleh. Tindakan mengacu kegiatan yang harus dilakukan sebelum hasil dapat diharapkan.

II.3. Faktor Ketidakpastian

Salah satu karakteristik umum dari suatu informasi yang tersedia untuk seorang ahli adalah ketidaksempurnaan atau kecacatan. Informasi bisa jadi tidak lengkap, tidak konsisten, tidak sesuai untuk menyelesaikan suatu permasalahan, akan tetapi seorang pakar dapat mengatasi kerusakan dan biasanya dapat membuat suatu pertimbangan benar dan keputusan yang benar. (Rika; 2012 : 85)

II.3.1. Faktor Kepastian *(Certainty Factor)*

Certainty Factor (CF) merupakan metode penggabungan dari kepercayaan dan ketidak percayaan untuk suatu permasalahan, dengan menunjukkan pendekatan nilai bobot dari probabilitas hasil akhir dari permasalahan. (Helmi, Iwan; 2011 : 201)

Certainty Factor (CF) menunjukkan ukuran kepastian terhadap suatu fakta atau aturan. Notasi Faktor Kepastian :

CF(H,E)=MB(H,E)-MD(H,E)

Dengan :

CF (H,E) : Certainty Factor dari hipotesis H yang dipengaruhi oleh gejala (*evidence*) E. Besarnya CF berkisar antara -1 sampai dengan 1. Nilai -1 menunjukkan ketidakpercayaan mutlak dan nilai 1 menunjukkan kepercayaan mutlak.

MB (H,E) : Ukuran kenaikan kepercayaan (*measure of increased belief*) terhadap hipotesis H yang dipengaruhi oleh gejala E.

MD (H,E) : Ukuran kenaikan ketidakpercayaan (*measure of increased*

disbelief) terhadap hipotesis H yang dipengaruhi oleh gejala E.

II.3.2. Metode Kuantifikasi Pertanyaan

Metode kualifikasi pertanyaan merupakan metode untuk mendapatkan nilai faktor kepaastian dari pengguna terhadap suatu evidence dengan mengkuantifikasi pertanyaan. Sebagai contoh, diinginkan untuk mengetahui derajat kepercayaan bisul seorang pasien. Nilai derajat kepercayaan adalah antara -1 s/d 1. Nilai -1 artinya tidak nyeri sama sekali pada kulit yang terinfeksi, dan nilai 1 berarti sangat byeri pada kulit. Untuk mendapatkan nilai derajat kepercayaan terhadap influensa yang dialami pasien, maka pertanyaan yang diberikan oleh sistem adalah *" seberapa nyeri yang terjadi pada infeksi".*

Dari jawaban pengguna, besarnya nilai kepercayaan pengguna akan dihitung oleh sistem.

II.4. Visual Basic 2010

Visual Basic adalah bahasa pemrograman klasik, legendaries, dan tiada duanya yang paling banyak dipakai oleh programmer di dunia. Dari zaman pemrograman visual di computer berbasis Windows 3.x hingga kini di zaman web. Bahasa pemrograman ini dipakai oleh jutaan programmer, dan tercatat sebagai progam yang paling dikuasai oleh mayoritas orang.

Visual digunakan dari mulai *programmer professional* yang mencari nafkah dari pembuatan program dan *coding*, hingga para hobi dan para mahasiswa yang membuat program untuk tugas kulia dan tugas akhir.

Selain digunakan untuk melakukan pemrograman desktop (yang merupakan asal mula kegunaan Visual Basic), kini Visual Basic 2010 juga sudah lazim dipakai untuk mengembangkan aplikasi web. Ini karena Visual Studio juga member fasilitas Visual web *designer* yang memungkinkan Visual Basic 2010 digunakan untuk membuat web dengan bahasa pemrograman ASP.NET. (Edy, Ali dkk, 2011 : 7)

II.5. Database SQL Server

SQL Server adalah sebuah terobosan baru dari Microsoft dalam bidang database. SQL Server adalah sebuah DBMS (*Database Management System*) yang dibuat oleh Microsoft untuk ikut berkecimpung dalam persaingan dunia pengolahan data menyusul pendahulunya seperti IBM dan Oracle. SQL Server dibuat pada saat kemajuan dalam bidang hardware sedemikian pesat. Oleh karena itu sudah dapat dipastikan bahwa SQL Server membawa beberapa terobosan dalam bidang pengolahan dan penyimpanan data. (Wahana Komputer, 2010 : 2)

II.6. Pengenalan UML (*Unified Modeling Languange)*

UML *(Unified Modeling Language)* merupakan metodologi koloborasi antara metoda – metoda *booch, OMT (Object Modeling Technique)* serta OOSE (*Object oriented Engineering)* dan beberapa metoda lainnya merupakan metodololgi yang paling sering digunakan saat ini untuk mengadaptasi marak nya penggunaan bahasa “pemograman berorientasi objek” (OOP) (Adi Nogroho ;2009 : 4)

II.6.1. Diagram – Diagram UML

Beberapa literature menyebutkan bahwa UML menyediakan Sembilan jenis diagram, yang lain menyebutkan delapan karena ada beberapa diagram yang digabung menjadi diagram interaksi. Namun demikian model – model itu dapat dikelompokkan berdasarkan sifatnya yaitu statis atau dinamis. Jenis diagram itu antara lain:

1. Diagram Kelas (*Class Diagram).* bersifat statis. Diagram ini memperlihatkan himpunan kelas – kelas, Antarmuka – antarmuka, koloborasi – koloborasi, serta relasi. Diagram ini umum dijumpai pada pemodelan sistem berientasi objek.
2. Diagram paket *(Package Diagram).* Bersifat statis. Diagram ini memperlihatkan kumpulan kelas – kelas , merupakan bagian dari diagram komponen.
3. Diagram *Use Case.* Bersifat statis. Diagram ini memperlihatkan himpunan *use-case* dan aktor – aktor (suatu jenis khusus dari kelas). Diagram ini terutama sangat penting untuk mengorganisasi dan memodelkan perilaku suatu sistem yang dibutuhkan serta diharapkan pengguna.
4. Diagram interaksi dan *Sequence* (urutan). Bersifat dinamis. Diagram urutan adalah diagram interaksi yang menekankan pada pengiriman pesan dalam suatu waktu tertentu.
5. Diagram Komunikasi *(Communication Diagram).* Bersifat dinamis. Diagram koloborasi UML yang menekankan organisasi structural dari objek – objek yang menerima serta mengirim pesan.

6. Diagram *Statechart (Statechart Diagram). Bersifat dinamis.* Diagram status memperlihatkan keadaan – keadaan pada sistem, memuat status *(state),*transisi, kejadian serta aktifitas. Diagram ini terutama penting untuk memperlihatkan sifat dinamis dari antar muka *(interface),* kelas, koloborasi dan terutama penting pada pemodelalan sistem – sistem yang reaktif.
7. Diagram Aktivitas *(Activity Diagram).* Barsifat dinamis. Diagram aktivitas adalah tipe khusus dari diagram status yang memperlihat aliran dari suatu aktivitas ke aktivitas lainnya dalam suatu sistem. Diagram ini terutama penting dalam pemodelan fungsi – fungsi suatu sistem dan member tekanan pada aliran kendali antar objek
8. Diagram Komponen *(Component Diagram).* Bersifat statis. Diagram komponen ini memperlihatkan hubungan dengan supplier dan pelanggan yang bersifat eksternal harus diperhatikan.

II.6.2. Jenis – jenis 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.

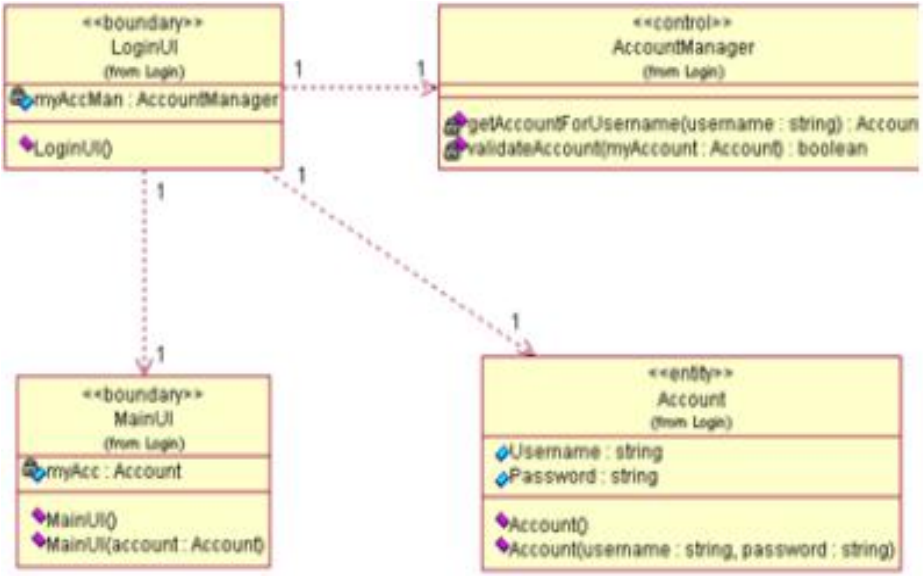
Pada UML terdiri atas tiga kategori dan memiliki 13 jenis diagram yaitu sebagai berikut:

1. *Class diagram*

Class diagram menggambarkan struktur statis dari kelas dalam sistem anda dan menggambarkan atribut, operasi dan hubungan antara kelas. Class diagram membantu dalam memvisualisasikan struktur kelas-kelas dari suatu

sistem dan merupakan tipe diagram yang paling banyak dipakai. Class memiliki tiga area pokok : Nama (dan stereotype) , Atribut, Metoda.

Adapun gambar notasi *Class diagram* dapat dilihat pada gambar II.4.



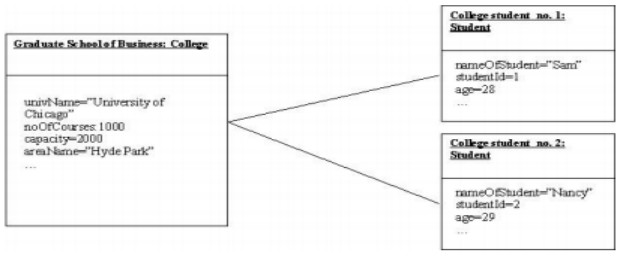
Gambar II.4 Notasi *Class diagram*

(Sumber: Mulawarman; 2011: 6)

2. *Object diagram*

Object diagram menggambarkan kejelasan kelas dan warisan dan kadang-kadang diambil ketika merencanakan kelas, atau untuk membantu pemangku kepentingan non-program yang mungkin menemukan diagram kelas terlalu abstrak.

Adapun gambar notasi *Objek diagram* dapat dilihat pada gambar II.5.



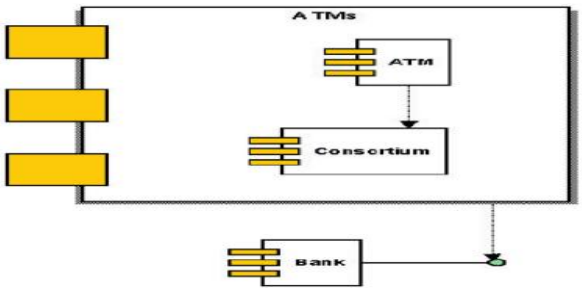
Gambar II.5 Notasi *Object diagram*

(Sumber: Mulawarman; 2011: 6)

3. *Component diagram*

Component diagram menggambarkan struktur fisik dari kode, pemetaan pandangan logis dari kelas proyek untuk kode aktual di mana logika ini dilaksanakan.

Adapun gambar notasi *Component diagram* dapat dilihat pada gambar II.6.



Gambar II.6 Notasi *Component diagram*

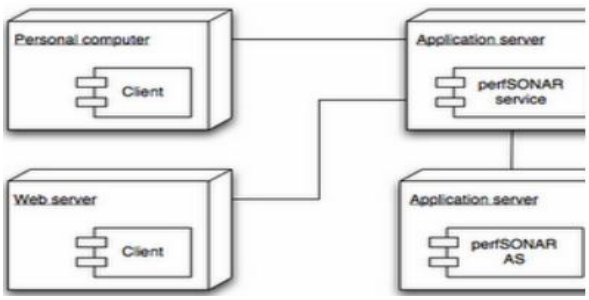
(Sumber: Mulawarman; 2011: 6)

4. *Deployment diagram*

Deployment diagram memberikan gambaran dari arsitektur fisik perangkat lunak, perangkat keras, dan artefak dari sistem. Deployment diagram dapat dianggap sebagai ujung spektrum dari kasus penggunaan

menggambarkan bentuk fisik dari sistem yang bertentangan dengan gambar konseptual dari pengguna dan perangkat berinteraksi dengan sistem.

Adapun gambar notasi *deployment diagram* dapat dilihat pada gambar II.7.



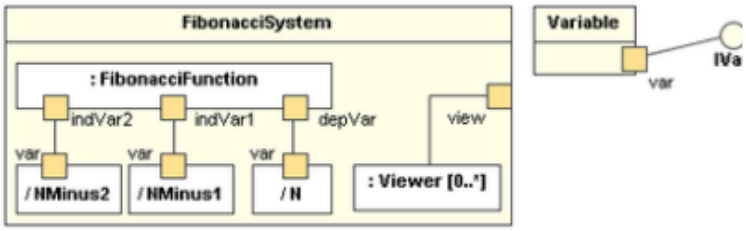
Gambar II.7 Notasi *deployment diagram*

(Sumber: Mulawarman; 2011: 6)

5. *Composite structure diagram*

Composite structure diagram Sebuah diagram struktur komposit mirip dengan diagram kelas, tetapi menggambarkan bagian individu, bukan seluruh kelas. Kita dapat menambahkan konektor untuk menghubungkan dua atau lebih bagian dalam atau ketergantungan hubungan asosiasi.

Adapun gambar notasi *Composite structur* dapat dilihat pada gambar II.8.



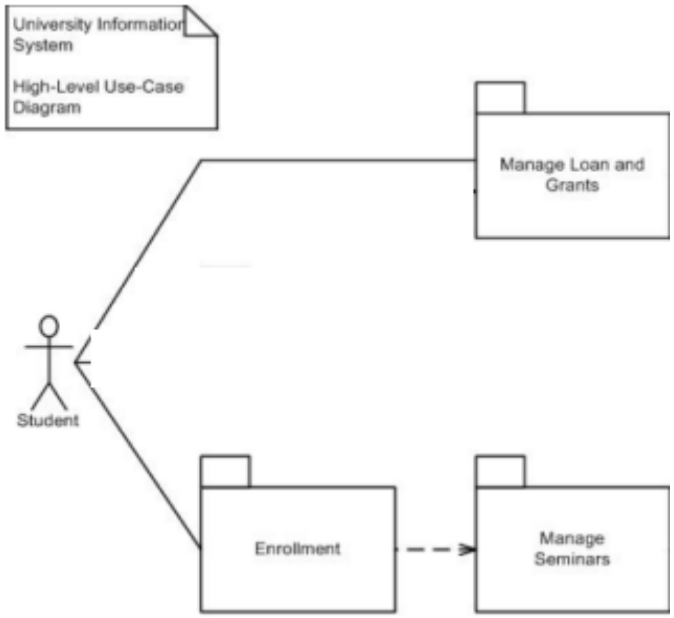
Gambar II.8 Notasi *Composite structure diagram*

(Sumber: Mulawarman; 2011: 6)

6. *Package diagram*

Paket diagram biasanya digunakan untuk menggambarkan tingkat organisasi yang tinggi dari suatu proyek software. Atau dengan kata lain untuk menghasilkan diagram ketergantungan paket untuk setiap paket dalam Pohon Model.

Adapun contoh gambar notasi *package diagram* dapat dilihat pada gambar II.9.



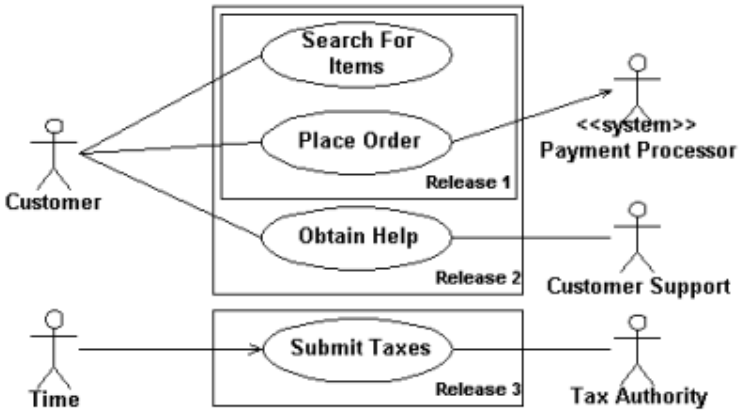
Gambar II.9 Notasi *Package diagram*

(Sumber: Mulawarman; 2011: 4)

7. *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.

Adapun gambar notasi *use case diagram* dapat dilihat pada gambar II.10.



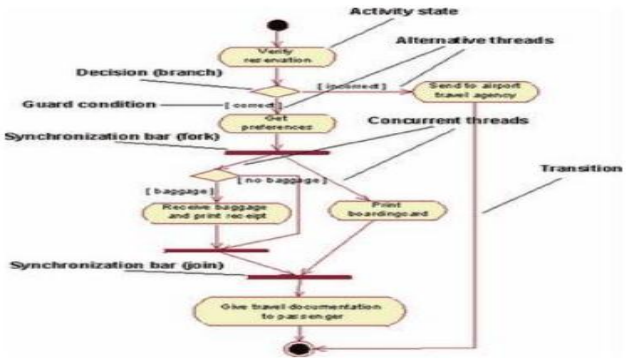
Gambar II.10 Notasi *use case diagram*

(Sumber: Mulawarman; 2011: 4)

8. *Activity diagram*

Menggambarkan aktifitas-aktifitas, objek, state, transisi state dan event. Dengan kata lain kegiatan diagram alur kerja menggambarkan perilaku sistem untuk aktivitas.

Adapun gambar notasi *Activity diagram* dapat dilihat pada gambar II.11.



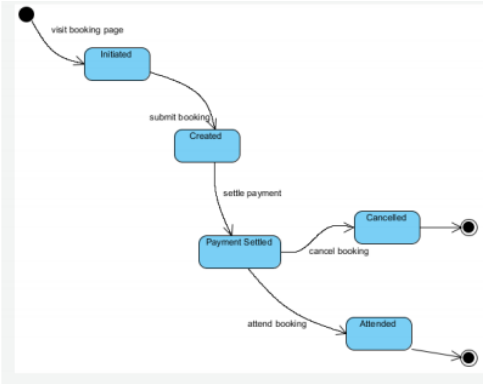
Gambar II.11 Notasi *Activity diagram*

(Sumber: Mulawarman; 2011: 4)

9. State Machine diagram

Menggambarkan state, transisi state dan event.

Adapun gambar notasi *State Mchine diagram* dapat dilihat pada gambar II.12.



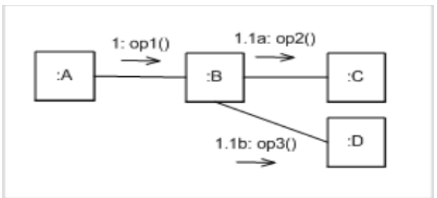
Gambar II.12 *State Machine diagram*

(Sumber: Mulawarman; 2011: 6)

10. *Communication diagram*

Serupa dengan sequence diagram, tetapi diagram komunikasi juga digunakan untuk memodelkan perilaku dinamis dari use case. Bila dibandingkan dengan Sequence diagram, diagram komunikasi lebih terfokus pada menampilkan kolaborasi benda daripada urutan waktu.

Adapun gambar notasi *Communicatin diagram* dapat dilihat pada gambar II.13.



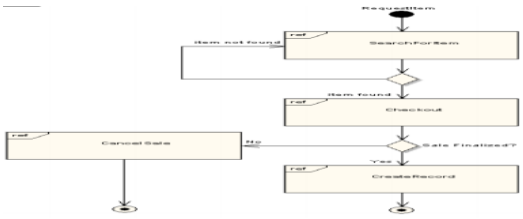
Gambar II.13 Notasi *Communication diagram*

(Sumber: Mulawarman; 2011: 4)

11. *Interaction Overview diagram*

Interaksi overview diagram berfokus pada gambaran aliran kendali interaksi dimana node adalah interaksi atau kejadian interaksi.

Adapun gambar notasi *Interaction OverView diagram* dapat dilihat pada gambar II.14.



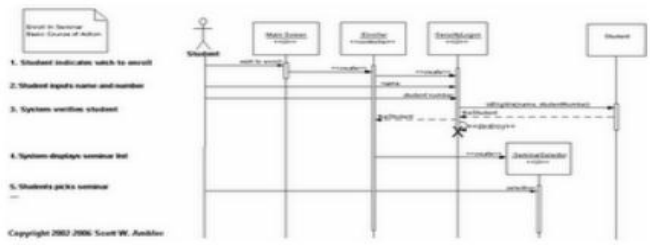
Gambar II.14 Notasi *Interaction Overview diagram*

(Sumber: Mulawarman; 2011: 5)

12. *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.*

Adapun gambar notasi *Sequence diagram* dapat dilihat pada gambar II.15.



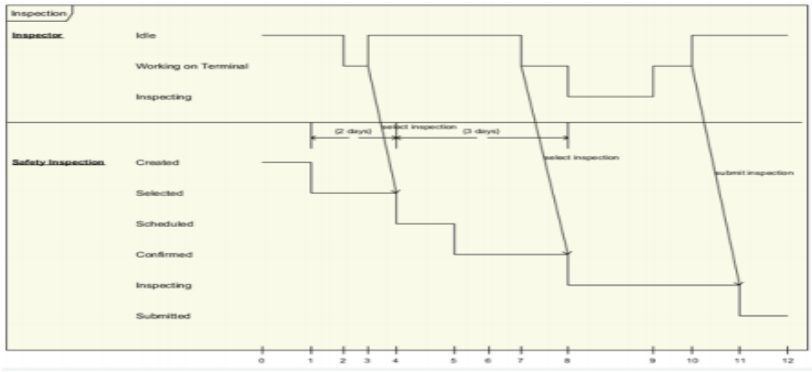
Gambar II.15 Notasi *Sequence diagram*

(Sumber: Mulawarman; 2011: 5)

13. *Timing diagram*

Timing diagram di UML didasarkan pada diagram waktu *hardware* awalnya dikembangkan oleh para insinyur listrik.

Adapun gambar notasi *Timing diagram* dapat dilihat pada gambar II.16.



Gambar II.16 Notasi *Timing diagram*

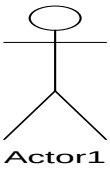
(Sumber: Mulawarman; 2011: 6)

II.5.3. Notasi – Notasi UML

UML memiliki seperangkat notasi yang akan digunakan ke dalam tiga kategori diatas yaitu struktur diagram, behaviour diagram dan interaction diagram. Berikut beberapa notasi dalam UML diantaranya :

1. *Actor;* menentukan peran yang dimainkan oleh user atau sistem lain yang berinteraksi dengan subjek. *Actor* adalah segala sesuatu yang berinteraksi langsung dengan sistem aplikasi komputer, seperti orang, benda atau lainnya.

   Adapun gambar *Actor* dapat dilihat pada gambar II.17.

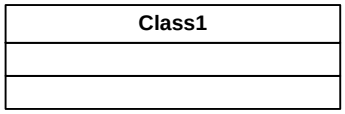


Gambar II.17. Notasi *actor* pada UML

(Sumber: Mulawarman; 2011: 6)

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 sistem berorientasi objek.

   Adapun gambar *Class diagram* dapat dilihat pada gambar II.18.



Gambar II.18 Notasi *class* pada UML

(Sumber: Mulawarman; 2011: 6)

3. *Use case* adalah deskripsi fungsi dari sebuah sistem 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.

   Adapun gambar *use case* dapat dilihat pada gambar II.19.

Gambar II.19. Notasi *Use case* pada UML

(Sumber: Mulawarman; 2011: 6)

4. *Realization* menunjukkan hubungan bahwa elemen yang ada di bagian tanpa panah akan merealisasikan apa yang dinyatakan oleh elemen yang ada di bagian dengan panah.

   Adapun gambar *Realization* dapat dilihat pada gambar II.20.

Gambar II.20 Notasi *Realization* pada UML

(Sumber: Mulawarman; 2011: 6)

5. *Interaction* digunakan untuk menunjukkan baik aliran pesan atau informasi antar obyek maupun hubungan antar obyek.

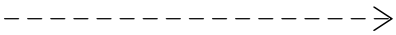
   Adapun *Interaction* gambar dapat dilihat pada gambar II.21.

Gambar II.21 Notasi *Interaction* pada UML

(Sumber: Mulawarman; 2011: 6)

6. *Dependency* merupakan relasi yang menunjukan bahwa perubahan pada salah satu elemen memberi pengaruh pada elemen lain. Terdapat 2 *stereotype* dari dependency, yaitu include dan extend. Include menunjukkan bahwa suatu bagian dari elemen (yang ada digaris tanpa panah) memicu eksekusi bagian dari elemen lain (yang ada di garis dengan panah).

   Adapun *Dependency* gambar dapat dilihat pada gambar II.22.



Gambar II.22 Notasi *Dependency* pada UML

(Sumber: Mulawarman; 2011: 6)

7. *Note* digunakan untuk memberikan keterangan atau komentar tambahan dari suatu elemen sehingga bisa langsung terlampir dalam model. *Note* ini bisa disertakan ke semua elemen notasi yang lain.

   Adapun gambar *Note* dapat dilihat pada gambar II.23.

Gambar II.23 Notasi *Note* pada UML

(Sumber: Mulawarman; 2011: 6)

8. *Association* menggambarkan navigasi antar class (navigation), berapa banyak obyek lain yang bisa berhubungan dengan satu obyek (*multiplicity* antar *class*) dan apakah suatu class menjadi bagian dari class lainnya (aggregation).

   Adapun gambar *Association* dapat dilihat pada gambar II.24.

Gambar II.24 Notasi *Association* pada UML

(Sumber: Mulawarman; 2011: 7)

9. *Generalization* menunjukkan hubungan antara elemen yang lebih umum ke elemen yang lebih spesifik.

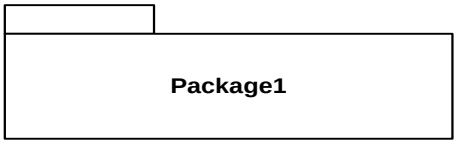
Adapun gambar *Association* dapat dilihat pada gambar II.25.

Gambar II.25 Notasi *Generalization* pada UML

(Sumber: Mulawarman; 2011: 7)

10. *Package* adalah mekanisme pengelompokkan yang digunakan untuk menandakan pengelompokkan elemen-elemen model.

    Adapun gambar *Package* dapat dilihat pada gambar II.26.



Gambar II.26 Notasi *Package* pada UML

(Sumber: Mulawarman; 2011: 7)

11. *Interface* merupakan kumpulan operasi berupa implementasi dari suatu *class*. Atau dengan kata lain implementasi operasi dalam *interface* dijabarkan oleh operasi di dalam *class*.

    Adapun gambar *Association* dapat dilihat pada gambar II.27.

Gambar II.27 Notasi *Interface* pada UML

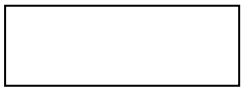
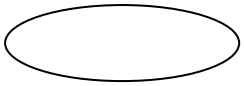
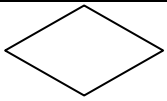
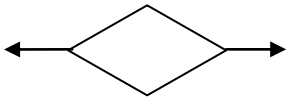
(Sumber: Mulawarman; 2011: 7)

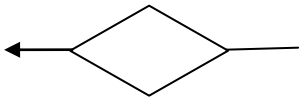
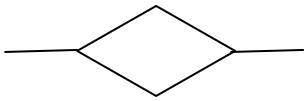
II.7. Pengenalan ERD (*Entity Relationship Diagram*)

Entity Relationship Diagram (ERD) adalah sekumpulan cara atau peralatan untuk mendeskripsikan data-data atau objek-objek yang dibuat berdasarkan dan berasal dari dunia nyata yang disebut entitas (*entity*) serta hubungan (*relationship*) antar entitas-entitas tersebut dengan menggunakan beberapa notasi.

Komponen-komponen pembentuk ERD dapat di lihat pada tabel II.2. di bawah ini.

Tabel II.2 : Komponen-Komponen ERD

Notasi	Komponen	Keterangan
	Entitas	Individu yang mewakili suatu objek dan dapat dibedakan dengan objek yang lain.
	Atribut	Properti yang dimiliki oleh suatu entitas, dimana dapat mendeskripsikan karakteristik dari entitas tersebut.
	Relasi	Menunjukkan hubungan diantara sejumlah entitas yang berbeda.
	Relasi 1 : 1	Relasi yang menunjukkan bahwa setiap entitas pada himpunan entitas pertama berhubungan dengan paling

		banyak satu entitas pada himpunan entitas kedua
	Relasi 1 : N	Relasi yang menunjukkan bahwa hubungan antara entitas pertama dengan entitas kedua adalah satu banding banyak atau sebaliknya. Setiap entitas dapat berelasi dengan banyak entitas pada himpunan entias yang lain
	Relasi N : N	Hubungan ini menunjukkan bahwa setiap entitas pada himpunan entitas yang pertama dapat berhubungan dengan banyak entitas pada himpunan entitas yang kedua, demikian juga sebaliknya

(Sumber : Doro, Stevalin; 2009 : 76)

II.7.1. Pemetaan Kardinalitas

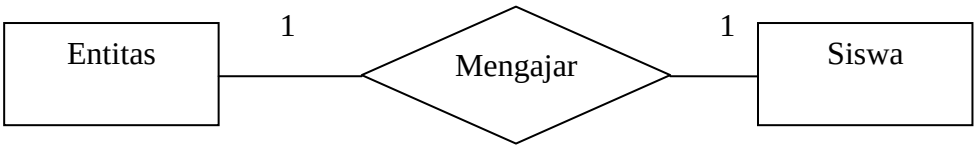
Pemetaan kardinalitas menyatakan jumlah entitas di mana entitas lain dapat dihubungkan ke entitas tersebut melalui sebuah himpunan relasi. (Tri Octafian; 2011:150-151).

1. *One to One*

Sebuah entitas pada A berhubungan dengan paling banyak satu entitas pada B dan sebuah entitas pada B berhubungan dengan paling banyak satu entitas pada A.

Contoh:

Pada pengajaran privat, satu guru satu siswa. Seorang guru mengajar seorang siswa, seorang siswa diajar oleh seorang guru.



Gambar II.28. Hubungan *One To One.*

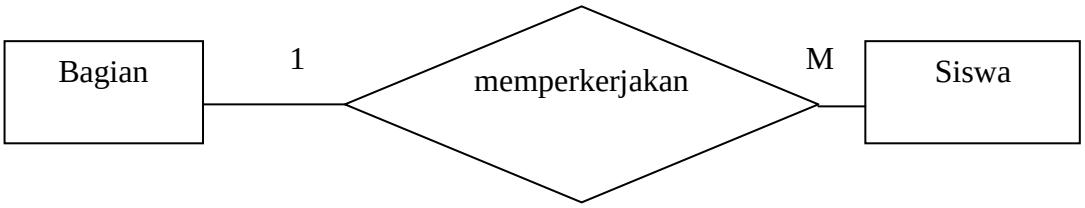
(Sumber: Tri Octafian; 2011:150)

2. *One to Many/ Many to One*

Sebuah entitas pada A berhubungan dengan lebih dari satu entitas pada B dansebuah entitas pada B berhubungan dengan paling banyak satu entitas pada A,atau sebaliknya (Many to One).

Contoh:

Dalam satu perusahaan, satu bagian mempekerjakan banyak pegawai. Satu bagian mempekerjakan banyak pegawai, satu pegawai kerja dalam satu bagian.



Gambar II.29. *One To Many*

(Sumber: Tri Octafian; 2011:151)

3. *Many To Many*

Sebuah entitas pada A berhubungan dengan lebih dari satu entitas pada B dan sebuah entitas pada B berhubungan dengan lebih dari satu entitas pada A.

Contoh:

Dalam universitas, seorang mahasiswa dapat mengambil banyak mata kuliah.Satu mahasiswa mengambil banyak mata kuliah dan satu mata kuliah diambil banyak mahasiswa.

Gambar II.30. Hubungan *Many To Many*

(Sumber: Tri Octafian ; 2011:152)