BAB II

LANDASAN TEORI

II.1. Sistem Pakar

Secara umum, sistem pakar *(expert system)* adalah sistem yang berusaha mengadopsi pengetahuan manusia ke komputer, agar komputer dapat menyelesaikan masalah seperti yang biasa dilakukan oleh para ahli. Sistem pakar yang dirancang agar dapat menyelesaikan suatu permasalahan tertentu dengan meniru kerja dari para ahli. Dengan sistem pakar ini, orang awampun dapat menyelesaikan masalah yang cukup rumit yang sebenarnya hanya dapat diselesaikan dengan bantuan para ahli. Bagi para ahli, sistem pakar ini juga akan membantu aktivitasnya sebagai asisten yang sangat berpengalaman. Ada beberapa defenisi tentang sistem pakar, antara lain :

1. Menurut Darkin sistem pakar adalah suatu program komputer yang dirancang untuk memodelkan kemampuan penyelesaian masalah yang dilakukan oleh seorang pakar.
2. Menurut Ignizio sistem pakar adalah suatu model dan prosedur yang berkaitan, dalam suatu domain tertentu, yang mana tingkat keahliannya dapat dibandingkan dengan keahlian seorang pakar.
3. Menurut Giarratano dan Riley sistem pakar adalah suatu sistem komputer yang bisa menyamai atau meniru kemampuan seorang pakar.

Sistem Pakar ini mengalami kegagalan pengetahuan terlalu luas sehingga terkadang justru meninggalkan pengetahuan-pengetahuan penting yang seharusnya disediakan. Sampai saat ini sudah banyak sistem pakar yang dibuat, beberapa contoh diantaranya terlihat pada Tabel II.1. (Sri Kusumadewi, 2003 : 109).

Tabel II.1. Sistem Pakar yang terkenal

Sistem Pakar	Kegunaan
MYCIN	Diagnosa Penyakit
DENDRAL	Mengidentifikasi struktur molekular campuran yang tak dikenal
XCON & XSEL	Membantu konfigurasi sistem komputer besar
SOPHIE	Analisis sirkit elektronik
Prospector	Digunakan didalam geologi untuk membantu mencari dan menemukan deposit.
FOLIO	Membantu memberikan keputusan bagi seorang manajer dalam hal stok *broker* dan investasi.
DELTA	Pemeliharaan lokomotif listrik disel

(Sumber : Sri Kusumadewi, 2003 : 110)

Salah satu teknik kecerdasan buatan yang menirukan proses penalaran manusia adalah Sistem Pakar. Secara umum, Sistem Pakar (*expert system*) adalah sistem yang berusaha mengadopsi pengetahuan manusia ke komputer, agar komputer dapat menyelesaikan masalah seperti yang biasa dilakukan oleh para ahli Sistem Pakar yang baik dirancang agar dapat menyelesaikan suatu permasalahan tertentu dengan meniru kerja para ahli. Dengan Sistem Pakar ini, orang awam juga dapat menyelesaikan masalah yang cukup rumit yang sebenarnya hanya dapat diselesaikan dengan bantuan para ahli. Bagi para ahli, Sistem Pakar ini juga akan membantu aktivitasnya sebagai asisten yang sangat berpengalaman (Jurnal Rahayu : 3).

II.2. Konsep Dasar Sistem Pakar

Menurut Efraim Turban (2001), konsep dasar sistem pakar mengandung keahlian, ahli, pengalihan keahlian, inferensi, aturan dan kemampuan menjelaskan.

Keahlian adalah suatu kelebihan penguasaan pengetahuan dibidang tertentu yang diperoleh dari pelatihan, membaca atau pengalaman. Contoh bentuk pengetahuan yang termasuk keahlian adalah :

1. Fakta-fakta pada lingkup permasalahan tertentu.
2. Teori-teori pada lingkup permasalahan tertentu.
3. Prosedur-prosedur dan aturan-aturan berkenaan dengan lingkup permasalahan tertentu.
4. Strategi-strategi global untuk menyelesaikan masalah.
5. *Meta-knowlegde* (pengetahuan tentang pengetahuan).

Bentuk-bentuk ini memungkinkan para ahli untuk dapat mengambil keputusan lebih cepat dan lebih baik daripada seseorang yang bukan ahli. Seorang ahli adalah seorang yang mampu menjelaskan suatu tanggapan, mempelajari hal-hal baru seputar topik permasalahan (domain), menyusun kembali pengetahuan jika dipandang perlu, memecah aturan-aturan jika dibutuhkan, dan menentukan releven tidaknya keahlian mereka.

Pengalihan keahlian dari para ahli ke komputer untuk kemudian dialihkan lagi ke orang lain yang bukan ahli, merupakan tujuan utama dari sistem pakar. Proses ini membutuhkan 4 aktivitas yaitu tambahan pengetahuan (dari para ahli atau sumber-sumber lainnya) representasi pengetahuan (ke komputer), inferensi pengetahuan, dan pengalihan pengetahuan ke user. Pengetahuan yang disimpan di

komputer disebut dengan nama basis pengetahuan. Ada 2 tipe pegetahuan yaitu dakta dan prosedur (biasanya berupa aturan).

Salah satu fitur yang harus dimiliki oleh sistem pakar adalah kemampuan untuk menalar. Jika keahlian-keahlian sudah tersimpan sebagai basis pengetahuan dan sudah tersedia program yang mampu mengakses basisdata, maka komputer harus dapat diprogram untuk membuat inferensi. Proses inferensi ini dikemas dalam bentuk motor inferensi *(inference engine).*

Sebagian besar sistem pakar komersial dibuat dalam bentuk *rule based systems*, yang mana pengetahuan disimpan dalam bentuk aturan-aturan. Aturan tersebut biasanya berbentuk IF-THEN.

Fitur lainnya dari sistem pakar adalah kemampuan untuk merekomendasi. Kemampuan inilah yang membedakan sistem pakar dengan sistem konvensional dilihat pada Tabel II.2 (Kusumadewi, 2003:112).

Tabel II.2 Sistem Konvensional Vs. Sistem Pakar

Sistem Konvensional	Sistem Pakar
Informasi dan pemprosesannya biasanya jadi satu dengan program.	Basis pengetahuan merupakan bagian terpisah dari mekanisme inferensi.
Biasanya tidak bisa menjelaskan mengapa suatu input data itu dibutuhkan, atau bagaimana *output* itu di peroleh.	Penjelasan adalah bagian terpenting dari sistem pakar.
Pengubahan program cukup sulit dan membosankan	Perubahan aturan dapat dilakukan dengan mudah.
Sistem hanya akan beroperasi jika sistem tersebut sudah lengkap.	Sistem dapat beroperasi hanya dengan beberapa aturan.
Eksekusi dilakukan langkah demi langkah.	Eksekusi dilakukan pada keseluruhan basis pengetahuan.
Menggunakan data.	Menggunakan pengetahuan.
Tujuannya utamanya adalah efisiensi	Tujuan utamanya adalah efektivitas.

(Sumber : Kusumadewi, 2003:112)

II.3. Struktur Sistem Pakar

Sistem pakar disusun oleh dua bagian utama, yaiut lingkungan pengembangan *(development environment)* dan lingkungan konsultasi *(consultation enviroment).* Lingkungan pengembangan sistem pakar digunakan untuk memasukkan pengetahuan pakar kedalam lingkungan sistem pakar, sedangkan lingkungan konsultasi digunakan oleh pengguna yang bukan pakar dalam memperoleh pengetahuan pakar. Komponen-komponen sistem pakar dalam kedua bagian tersebut yaitu *User Interfa*ce (antarmuka pengguna), basis pengetahuan, akuisisi pengetahuan, mesin inferensi, *workplace*, fasilitas, penjelasan, perbaikan pengetahuan (Kusumadewi, 2003:113).

II.3.1. Antarmuka Pengguna

Antarmuka pengguna *(user interface)* merupakan mekanisme yang digunakan oleh pengguna dan sistem pakar untuk berkomunikasi. Antarmuka menerima informasi dari pemakai dan mengubahnya ke dalam bentuk yang dapat diterima oleh sistem. Selain itu, antarmuka menerima informasi dari sistem dan menyajikannya ke dalam bentuk yang dapat dimengerti oleh pemakai. Menurut McLeod (1995), pada bagian ini terjadi dialog antara program dari pemakai,yang memungkinkan sistem pakar menerima instruksi dan informasi *(input)* dari pemakai, juga memberika informasi *(output)* kepada pemakai.

II.3.2. Basis Pengetahuan

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

Dalam studi kasus pada sistem berbasis pengetahuan, terdapat beberapa karakteristik dibangun yang akan membantu kita dalam membentuk serangkaian prinsip-prinsip arsitekturnya. Prinsip tersebut meliput :

1. Pengetahuan merupakan kunci kekuatan sistem pakar.
2. Pengetahuan sering tidak pasti dan tidak lengkap.
3. Pengetahuan sering miskin spesifikasi
4. Amatir menjadi ahli secara bertahap
5. Sistem pakar harus fleksibel
6. Sistem pakar harus transparan

Sejarah penelitian di bidang AI telah menunjukkan berulang kali bahwa basis pengetahuan adalah kunci untuk setiap sistem cerdas *(intelligence system).*

II.3.3. Akuisisi Pengetahuan

Akuisisi pengetahuan *(knowledge acquistion)* adalah akumulasi, transfer dan transformasi keahlian dalam menyelesaikan masalah dari sumber pengetahuan ke dalam program komputer. Dalam tahap ini, *knowledge engineer* berusaha menyerap pengetahuan untuk selanjutnya ditransfer ke dalam basis pengetahuan.

Pengetahuan diperoleh dari pakar, dilengkapi bukum basis data, laporan penelitian dan pengalaman pemakai. Menurut Turban (2001), terdapat tiga metode utama dalam akuisisi pengetahuan, yaitu :

1. Wawancara

   Wawancara adalah metode akuisisi yang paling banyak digunakan. Metode ini melibatkan pembicaraan dengan pakar secara langsung dalam suatu wawancara. Terdapat beberapa bentuk wawancara yang dapat digunakan. Masing-masing bentuk wawancara tersebut mempunyai tujuan yang berbeda.

2. Analisa protokol

   Dalam metode ini akuisisi ini, pakar diminta untuk melakukan suatu pekerjaan dan mengungkapkan proses pemikirannya dengan menggunakan kata-kata. Pekerjaan tersebut direkam, dituliskan, dan dianalisis.

3. Observasi pada pekerjaan pakar

   Dalam metode ini, pekerjaan dalam bidang tertentu yang dilakukan pakar direkam dan diobservasi.

4. Induksi aturan dari contoh

   Metode ini dibatasi untuk sistem berbasis aturan. Induksi adalah suatu proses penalaran dari khusus ke umum. Suatu sistem induksi aturan diberi contoh-contoh daru suatu masalah yang hasilnya telah diketahui. Setelah diberikan beberapa contoh, sistem induksi aturan tersebut dapat membuat aturan yang benar untuk kasus-kasus contoh. Selanjutnya,

aturan dapat digunakan untuk menilai kasus lain yang hasilnya tidak diketahui.

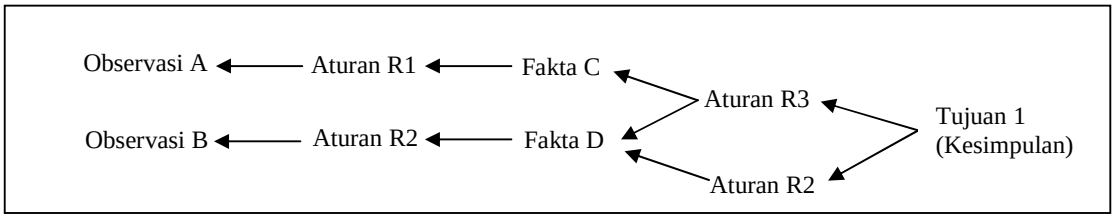
II.3.4. Mesin Inferensi

Komponen ini mengandung mekanisme pola pikir dan penalaran yang digunakan oleh pakar dalam menyelesaikan suatu masalah. Mesin inferensi adalah program komputer yang memberikan metodologi untuk penalaran tentang informasi yang ada dalam basis pengetahuan dan dalam *workplace*, dan untuk memformulasikan kesimpulan (Turban,1995).

Kebanyakan sistem pakar berbasis aturan menggunakan strategi inferensi yang dinamakan modus ponen. Berdasarkan strategi ini, jika terdapat aturan “IF A THEN B”, dan jika diketahui bahwa A benar, maka dapat disimpulkan bahwa B juga benar. Strategi inferensi modus ponen dinyatakan dalam bentuk : [A AND (A→B)]→B dengan A dan A→B adalah proposisi-proposisi dalam basis pengetahuan.

Terdapat dua pendekatan untuk mengontrol inferensi dalam sistem pakar berbasis aturan, yaitu pelacakan ke belakang (backward chaining) dan pelacakan ke depan (forward chaining). Pelacakan ke belakang adalah pendekatan yang dimotori tujuan (goal-driven). Dalam pendekatan ini pelacakan dimulai dari tujuan, selanjutnya dicari aturan yang memiliki tujuan tersebut untuk kesimpulannya. Selanjutnya proses pelacakan menggunakan premis untuk aturan tersebut sebagai tujuan baru sebagai tujuan baru dan mencari antara lain dengan

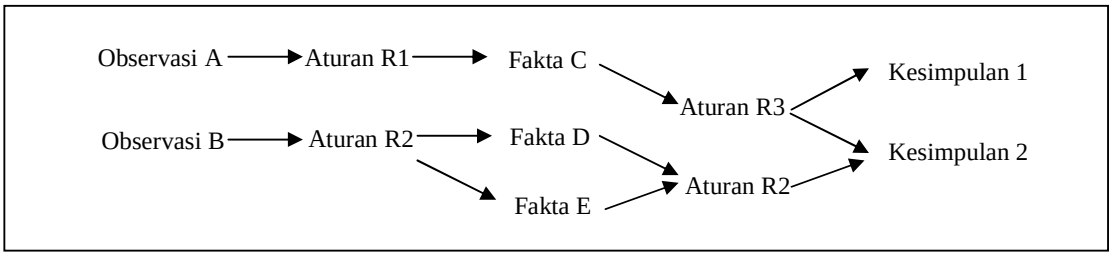
tujuan baru sebagai kesimpulannya. Proses berlanjut sampai semua kemungkinan ditemukan. Gambar II.1 menunjukkkan proses *backward chaining.*



Gambar. II.1. Proses *Bacward Chaining*

(Sumber : Arhami, 2005:19)

Pelacakan kedepan adalah pendekatan yang dimotori data(data-driven). Dalam pendekatan ini pelacakan dimulai dari informasi masukkan, dan selanjutnya mencoba menggambarkan kesimpulan. Pelacakan ke depan mencari fakta yang sesuai dengan bagian IF dari aturan IF-THEN. Gambar II.2 menunjukkan proses *forward chaining*.



Gambar II.2. Proses *Forward Chaining*

(Sumber: Arhami, 2005:20)

Kode metode inferensi tersebut dipengaruhi oleh tiga macam penelusuran, yaitu *Depth-First search, Breadth-First search* dan *Best-First search.*

a. *Depth-first search* melakukan penelusuran kaidah secara mendalam dari simpul akar bergerak menurun ke tingkat dalam yang berurutan.

b. *Breadth-First search*, bergerak dari simpul akar, simpul yang ada pada setiap tingkat diuji sebelum pindah ke tingkat selanjutnya.
c. *Best-First search*, bekerja berdasarkan kombinasi kedua metode sebelumnya.

II.3.5. *Workplace*

Workplace merupakan area dari sekumpulan memori kerja *(working memory). Workplace* digunakan untuk merekam hasil-hasil antara kesimpulan yang dicapai. Ada 3 tipe keputusan yang dapat direkam yaitu (Turban 2001) :

1. Rencana yaitu bagaimana menghadapi masalah
2. Agenda yaitu aksi-aksi yang potensial yang sedang menunggu untuk dieksekusi.
3. Solusi yaitu calon aksi yang akan dibangkitkan.

II.3.6. Fasilitas Penjelasan

Fasilitas penjelasan adalah komponen tambahan yang akan meningkatkan kemampuan sistem pakar. Komponen ini menggambarkan penalaran sistem kepada pemakai. Fasilitas penjelasan dapat menjelaskan perilaku sistem pakar dengan menjawab pertanyaan-pertanyaan sebagai berikut (Turban, 2001) :

1. Mengapa pertanyaan tertentu ditanyakan oleh sistem pakar ?
2. Bagaimana kesimpulan tertentu diperoleh ?
3. Mengapa alternatif tertentu ditolak ?
4. Apa rencana untuk memperoleh penyelesaian ?

II.3.7. Perbaikan Pengetahuan

Pakar memiliki kemampuan untuk menganalisis dan meningkatkan kinerjanya serta kemampuan untuk belajar dari kinerjanya. Kemampuan tersebut adalah penting dalam pembelajaran terkomputerisasi sehingga program akan mampu menganalisis penyebab kesuksesan dan kegagalan yang dialaminya.

II.4. Metode *Teorema Bayes*

Probabilitas *Bayes* merupakan salah satu cara yang baik untuk mengatasi ketidakpastian data dengan menggunakan formula bayes yang dinyatakan dengan rumus :

$$P(H|E) = \frac{P(E|H).P(H)}{P(E)}$$

Keterangan :

P(H | E) : probabilitas hipotesis H jika diberikan *evidence* E

P(E | H) : probabilitas munculnya *evidence* apapun

P(E) : probabilitas *evidence* E

Dalam bidang kedokteran teorema *Bayes* sudah dikenal tapi teorema ini lebih banyak diterapkan dalam logika kedokteran modern (Cutler:1991). Teorema ini lebih banyak diterapkan pada halhal yang berkenaan dengan probabilitas serta kemungkinan dari penyakit dan gejala-gejala yang berkaitan.

Secara umum teorema *Bayes* dengan E kejadian dan Hipotesis H dapat dituliskan dalam bentuk :

$$P(Hi|E)=\frac{P(E\cap Hi).}{\sum_j P(E\cap Hj)}$$

$$=\frac{P(E|Hi)\,P(Hi)}{\sum_j P(E|Hj)\,P(Hj)}$$

$$=\frac{P(E|Hi)\,P(Hi)}{P(E)}$$

Teorema Bayes dapat dikembangkan jika setelah dilakukan pengujian terhadap hipotesis kemudian muncul lebih daris satu *evidence*. Dalam hal ini maka persamaan nya akan menjadi :

$$P(H|E,e) = P(H) \mid E = \frac{P(e|E,H)}{P(e|E)}$$

Keterangan :

e : *evidence* lama

E : *evidence* baru

P(H | E,e) : probabilitas hipotesis H benar jika muncul *evidence* baru E dari evidence baru E dari *evidence* lama e.

P(H | E) : probabilitas hipotesis H benar jika diberikan *evidence* E.

P(e | E,H) : kaitan antar e dan E jika hipotesis H benar.

P(e | E) : kaitan antara e dan E tanpamemandang hipotesis apapun.

Berikut ini adalah contoh penghitungan probabilitas menggunakan probabilitas *Bayes* yang diambil dari Iswati (2004) dan Kusumadewi (2002) :

1. Probabilitas terkena penyakit *bronkhitis khronika* apabila mengalami batuk lebih dari 4 minggu. P(*bronchitis khronika* | batuk lebih dari 4 minggu) = 0,13 Terdapat gejala baru, yaitu batuk berdarah dalam 3 bulan terkahir,

probabilitas terkena penyakit *bronchitis khronika* apabila mengalami batuk berdarah dalam 3 bulan terakhir. P(*bronchitis khronika* | batuk darah dalam 3 bulan terakhir) = 0,4 Keterkaitan antara adanya gejala batuk lebih dari 4 minggu dan batuk darah dalam 3 bulan terkahir apabila seseorang menderita *bronchitis khronika* adalah 0,33. Keterkaitan antara adanya gejala batuk lebih dari 3 minggu dan batuk darah dalam 3 bulan terakhir tanpa memperhatikan penyakit yang diderita adalah 0,15, maka :

A = batuk darah dalam 3 bulan terakhir

B = batuk lebih dari 4 minggu

H = *bronchitis khronika*

$$P(H|A,B) = P(H|A) \times \frac{P(B|A,H)}{P(B,A)}$$

$$= 0,4 \times \frac{0,33}{0,15} = 0,88$$

(Arhami, 2005: 144)

2. Seorang dokter mengetahui bahwa penyakit maningitis menyebabkan “stiff neck” adalah 50%. Probabilitas pasien menderita maningitis adalah 1/50000 dan probabilitas pasien menderita stiff neck adalah 1/20 dari nilai-nilai tersebut. Didapatkan :

   P(stiff neck | maningitis) = 50%=0,5

   P(stiff neck) = 1/20

   Maka :

$$P(\text{maningitis} \mid \text{stiffneck} = \frac{P(\text{stiff neck} \mid \text{maningitis}) \times P(\text{maningitis})}{P(\text{stiff neck})}$$

$$\frac{\frac{5}{10} \times \frac{}{50000}}{\frac{1}{100}} = 0{,}0002$$

$$\frac{1}{5000} =$$

Hasil diatas menunjukkan bahwa hanya 1 di antara 5000 pasien yang mengalami *stiff neck*.

II.5. Kerusakan Mesin Automatic Floor Scrubbers

Mesin *Automatic Floor Scrubbers* merupakan perangkat pembersih lantai. Hal ini dapat menjadi alat sederhana seperti pel lantai dan sikat lantai, atau dalam bentuk berjalan-balik atau naik-*on*. Mesin ini untuk membersihkan daerah lantai yang lebih besar dengan menyuntikkan air dengan larutan pembersih, menggosok dan mengangkat kotoran dari lantai. Mesin ini merupakan mesin yang pengoperasian yang sangat mudah, aman digunakan, integrasi menggosok dan lebih efisiensi pembersihan dalam skala yang tinggi dan efek pembersihan luar biasa dan merupakan pilihan ideal untuk *replancing*, mesin ini berlaku untuk tempat-tempat umum besar dan menengah seperti *wineshops,* hotel, gedung perkantoran dan ruang pameran. Karakteristik mesin *automatic floor scrubbers* yang membedakan dengan mesin *automatic floor scrubbers* lainnya adalah metode penggunaan mesin yang sangat mudah. Mesin *automatic floor scrubbers* ini menggunakan *control handle* yang mudah digunakan ke arah mana saja.(Tatum, 2014 :12)

II.6. *SQL Server 2008*

SQL Server adalah sebuah sistem manejemen bisnis data relational (RDBMS) produk *Microsoft*. Bahasa query utamanya adalah *Transact-SQL* yang merupakan implementasi dari *SQL* standar ANSI/ISO yang digunakan oleh *Microsoft* dan *Sybase*. Umumnya *SQL Server* digunakan di dunia bisnis yang memiliki basis data berskala kecil sampai dengan menengah, tetapi kemudian berkembang dengan digunakannya *SQL Server* pada basis besar.

SQL Server 2008 adalah sebuah terobosan baru dari *Microsof*t dalam bidang database. *SQL Server* adalah sebuah DBMS *(Database Management System)* yang di buat oleh Microsoft untuk ikut berkecimpung dalam persaingan dunia pengolahan data menyusul pendahulunya seperti IBM dan *Oracle*. *SQL Server 2008* membawa terobosan dalam bidang pengolahan dan penyimpanan data.

II.7. *Unified Modelling Language* (UML)

UML yang merupakan singkatan dari *Unified Modelling Language* adalah sekumpulan pemodelan konvensi yang digunakan untuk menentukan atau menggambarkan sebuah sistem perangkat lunak dalam kaitannya dengan objek (Whitten, 2004).

Unified Modelling Language (UML) menurut Martin Fowler (2005 : 1) adalah keluarga notasi grafik yang didukung oleh meta-model tunggal, yang membantu pendeskripsian dan desain sistem perangkat lunak, khususnya sistem yang dibagun.

UML dapat juga diartikan sebuah bahasa grafik standar yang digunakan untuk memodelkan perangkat lunak berbasis objek. UML pertama kali dikembangkan pada pertengahan tahun 1990an dengan kerjasama antara James Rumbaugh, Grady Booch dan Ivar Jacobson, yang masing-masing telah mengembangkan notasi mereka sendiri diawal tahun 1990an. (Lethbride dan Leganiere, 2002).

UML singkatan dari *Unified Modelling Language* yang berarti bahasa pemodelan standar. Chonoles mengatakan 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 konteks.

Blok pembangunan utama UML adalah diagram. Beberapa diagram ada yang rinci (jenis *timing diagram*) dan lainnya yang bersifat umum (misalnya diagram kelas). Para pengembang sistem beorientasi objek menggunakan bahasa model untuk menggambar, membangun dan mendokumentasikan sistem yang mereka rancang. UML memungkinkan para anggota *team* untuk bekerja sama dengan bahasa model yang sama dengan mengaplikasikan beragam sistem. Intinya, UML merupakan alat komunikasi yang konsisten dalam men*support* para pengembang sistem saat ini. (Herlawati; 2011 : 6-7)

Beberapa leteratur menyebutkan bahwa UML menyediakan Sembilan jenis diagram, yang lain menyebutkan delapan karena ada beberapa yang digabung, misalnya diagram komunikasi, diagram urutan, dan diagram pewaktuan 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, bersifat statis. Diagram ini memperlihatkan himpunan kelas-kelas, antarmuka-antarmuka, kolaborasi-kolaborasi, serta relasi-relasi. Diagram ini umum dijumpai pada pemodelan sistem berorientasi objek. Meskipun bersifat statis, sering pula diagramkelas memuat kelas-kelas aktif.
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 himpunann *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 interksi yang menekankan pada pengiriman pesan dalam suatu waktu tertentu.
5. Diagram komunikasi (*Communication Diagram*), bersifat dinamis. Diagram sebagai pengganti diagram kolaborasi UML 14 yang menekankan organisasi structural objek-objek yang menerima mengirim pesan.

6. *Diagram Statechart (Statechart Diagram*), bersifat dinamis. Diagram state memperlihatkan keadaan-keadaan pada sistem, memuat status *(state),* transisi, kejadian serta aktifitas. Diagram ini terutama penting untuk memperlihatkan sifat dinamis dari antarmuka (interface), kelas, kolaborasi, dan terutama penting pada pemodelan sistem-sistem yang reaktif.
7. Diagram aktivitas *(Activity Diagram),* bersifat dinamis. Diagram aktivitasi adalah tipe khusus dari diagram status yang memperlihatkan aliran dari suatu aktivitas lainnya dalam suatu sistem. Diagram ini terutama penting dalam pemodelan fungsi-fungsi sistem dan memberi tekanan pada aliran kendali antar objek.
8. Diagram komponen *(Component Diagram),* bersifat statis. Diagram komponen ini memperlihatkan organisasi serta keberuntungan sistem/ perangkat lunak pada komponen-komponen yang telah ada sebelumnya. Diagram ini berhubungan dengan diagram kelas dimana komponen secara tipikal dipetakan ke dalam satu atau lebih kelas-kelas, antarmuka-antarmuka serta kolaborasi-kolaborasi.
9. Diagram *Deployment (Deployement Diagram),* bersifat statis. Diagram ini memperlihatkan konfigurasi saat aplikasi dijalankan *(run-time).* Memuat simpul-simpul beserta komponen-komponen yang ada di dalamnya. Diagram *deployment* berhubungan erat dengan diagram komponen dimana diagram ini memuat satu atau lebih komponen-komponen. Diagram ini sangat berguna saat aplikasi kita berlaku

sebagai aplikasi yang dijalankan pada banyak mesin *(Distributed Computing)*

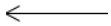
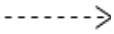
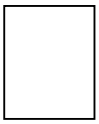
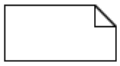
Kesembilan diagram ini tidak mutlak harus digunakan dalam pengembangan perangkat lunak, semuanya dibuat sesuai dengan kebutuhan. Pada UML memungkinkan kita menggunakan diagram-diagram lainnya (misalnya *Data Flow Diagram, Entity Relationship Diagram* dan sebagainya). (Prabowo Pudjo Widodo, Herlawati; 2011 : 10-12)

II.7.1. Diagram *Use Case (Use Case Diagram)*

Use Case menurut Martin Fowler (2005 : 141) adalah teknik untuk merekam persyaratan fungsional sebuah sistem. *Use Case* mendeskripsikan interaksi tipikal antara para pengguna sistem dengan sistem itu sendiri, dengan memberi sebuah narasi tentang bagaimana sistem tersebut digunakan. *Use Case* Diagram menampilkan aktor nama yang menggunakan *Use Case* mana, *Use Case* mana yang memasukkan *Use Case* lain dan hubungan antara aktor dan *Use Case* pada Tabel III.3.

Tabel II.3. Simbol Diagram *Use Case*

NO	GAMBAR	NAMA	KETERANGAN
1		*Actor*	Menspesifikasikan himpuan peran yang pengguna mainkan ketika berinteraksi dengan *use case*.
2		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (*independent*).

3		*Generalization*	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
4		*Include*	Menspesifikasikan bahwa *use case* sumber secara *eksplisit*.
5		*Extend*	Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan.
6		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.
7		*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
8		*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor
9		*Collaboration*	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan prilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
10		*Note*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi

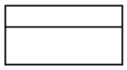
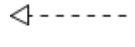
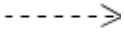
(Sumber : Martin Fowler, 2005 : 141)

II.7.2. Diagram Kelas *(Class Diagram)*

Class Diagram menurut Munawar (2005 : 28) merupakan himpunan dari objek-objek yang sejenis. Sebuah objek memiliki keadaan sesaat (state) dan perilaku (behavior). State sebuah objek adalah kondisi objek tersebut yang

dinyatakan dalam atribute/properties. Sedangkan perilaku suatu objek mendefenisikan bagaimana sebuah objek bertindak/berinteraksi dan memberikan reaksi.

Tabel II.4. Simbol *Class Diagram*

NO	GAMBAR	NAMA	KETERANGAN
1		*Generalization*	Hubungan dimana objek anak *(descendent)* berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
2		*Nary Association*	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
3		*Class*	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
4		*Collaboration*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu actor
5		*Realization*	Operasi yang benar-benar dilakukan oleh suatu objek.
6		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempegaruhi elemen yang bergantung padanya elemen yang tidak mandiri
7		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya

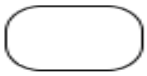
(Sumber : Munawar, 2005 : 28)

II.7.3. Diagram Aktivitas *(Activity Diagram)*

Activity Diagram menurut Martin Fowler (2005 : 163) adalah teknik untuk menggambarkan logika prosedural, proses bisnis, dan alur kerja. Dalam beberapa hal, activity diagram dimainkan peran mirip diagram alir, tetapi perbedaan prinsip antara notasi diagram alir adalah activity diagram mendukung behavior pararel.

Node pada sebuah activity diagram disebut sebagai action, sehingga diagram tersebut menampilkan sebuah activity yang tersusun dari action.

Tabel II.5. Simbol *Activity Diagram*

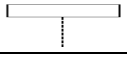
NO	GAMBAR	NAMA	KETERANGAN
1		*Actifity*	Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain
2		*Action*	State dari sistem yang mencerminkan eksekusi dari suatu aksi
3		*Initial Node*	Bagaimana objek dibentuk atau diawali.
4		*Actifity Final Node*	Bagaimana objek dibentuk dan dihancurkan
5		*Fork Node*	Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran

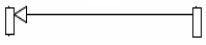
(Sumber : Martin Fowler, 2005 : 163)

II.7.4. Sequence Diagram

Sequence diagram menurut Munawar (2005 : 187) adalah grafik dua dimensi dimana objek ditunjukkan dalam dimensi horizontal, sedangkan *lifeline* ditunjukkan dalam dimensi vertikal.

Tabel II.6. Simbol *Sequence Diagram*

NO	GAMBAR	NAMA	KETERANGAN
1		*LifeLine*	Objek *entity*, antarmuka yang saling berinteraksi.
2		*Message*	Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi

3		*Message*	Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi

(Sumber : Munawar, 2005 : 187)

II.8. *Entity Relationship Diargam* (ERD)

Menurut Brady dan Loonam (2010), Entity Relationship diagram (ERD) merupakan teknik yang digunakan untuk memodelkan kebutuhan data dari suatu organisasi, biasanya oleh System Analys dalam tahap analisis persyaratan proyek pengembangan system. Sementara seolah-olah teknik diagram atau alat peraga memberikan dasar untuk desain database relasional yang mendasari sistem informasi yang dikembangkan. ERD bersama-sama dengan detail pendukung merupakan model data yang pada gilirannya digunakan sebagai spesifikasi untuk database.

II.8.1. Simbol-Simbol ERD *(Entity Relationship Diagram)*

Menurut Dolly Indra; 2010 : 3, *ERD* merupakan suatu cara untuk menjelaskan kepada para pemakai tentang hubungan antar data dalam basis data secara *logic* dengan *persepsi* bahwa *real world* terdiri dari objek-objek dasar yang saling berhubungan dengan cara menvisualisasikan ke dalam bentuk simbol-simbol grafis".

Pengertian *Entity Relationship Diagram (ERD)* menurut James A. Hall adalah : "*Entity Relationship Diagram* adalah Suatu teknik dokumentasi yang digunakan untuk menyajikan relasi antar entitas dalam sebuah sistem".

Dari kutipan di atas penulis menarik simpulan *Entity Relationship Diagram (ERD) adalah* suatu cara untuk menjelaskan kepada para pemakai tentang dokumentasi yang digunakan untuk menyajikan relasi, dan tentang hubungan antar data secara *logic*.

Pada model *Data Relation* hubungan antara *file* direlasikan dengan *relation key* yang merupakan kunci utama dari masing-masing *file*, adapun komponen utama dari *Entity Relationship Diagram* adalah:

1. *Entitas*

   Kumpulan dari objek antara objek yang satu dengan objek yang lain dapat dibedakan

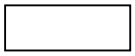
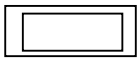
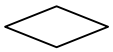
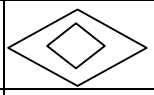
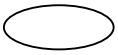
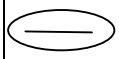
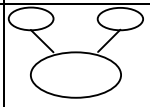
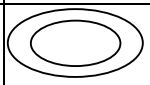
2. *Relationship*

   Hubungan yang terjadi antara satu entity atau lebih. *Entity Relationship* adalah relasi antara dua *file* atau dua *table* dapat dikategorikan menjadikan tiga macam. Yaitu, *One to One* (I : I), *One to Many* (I : M , M : I), *Many to Many* (M : M).

3. *Atribut*

   Kumpulan elemen-elemen data yang membentuk suatu *entity* yang menyediakan penjelasan *detail* dalam *entity*

Tabel II.7. Simbol *Entity Relationship Diagram*

No	Notasi	Nama	Arti
1		*Entity*	Objek yang dapat dibedakan dalam dunia nyata
2		*Week Entity*	Suatu *entity* dimana keberadaan dari *entity* tersebut tergantung dari keberadaan *entity* yang lain
3		*Relationship*	Hubungan yang terjadi antara satu atau lebih *entity*
4		*Identifying Relationship*	Hubungan yang terjadi antara satu atau lebih *weak entity*
5		*Atribut Simple*	*Atribut* yang bernilai tunggal atau atribut *atomic* yang tidak dapat dipilah-pilah lagi
6		*Atribut Primary Key*	Satu atau gabungan dari beberapa *atribut* yang membedakan semua baris data *(row)* dalam *table* secara unik
7		*Atribut Composite*	*Atribut* yang masih dapat diuraikan lagi menjadi sub-sub *atribut* yang masing-masing memiliki makna
8		*Atribut Multivalue*	Suatu *atribut* yang memiliki sekelompok nilai untuk setiap *instant entity*

(Sumber : Dolly Indra, 2010 : 3)

II.9. Normalisasi

Menurut Abdul Kadir (2009 : 116), Normalisasi adalah suatu proses yang digunakan untuk menentukan pengelompokkan atribut-atribut dalam sebuah relasi sehingga diperoleh relasi yang berstruktur baik. Dalam hal ini yang dimaksud dengan relasi yang berstruktur baik adalah relasi yang memenuhi dua kondisi yaitu sebagai berikut :

1. Mengandung redudansi yaitu data disimpan berkali-kali sesedikit mungkin.

2. Memungkinkan baris-baris dalam relasi disisipkan, dimodifikasi dan dihapus tanpa menimbulkan kesalahan atau ketidak konsistenan.

Normalisasi sendiri dilakukan melalui sejumlah langkah, setiap langkah berhubungan dengan bentuk normal *(Normal Form)* tertentu. Dalam hal ini yang disebut bentuk normal adalah suatu keadaan relasi yang dihasilkan oleh penerapan aturan-aturan sederhana yang berhubungan dengan *dependensi fungsional* terhadap relasi tersebut. (Abdul Kadir (2009 : 116). Yang dimaksud dengan aturan-aturan tersebut dan juga istilah dependensi fungsional yang dibahas.

II.10. *Microsoft Visual Studio 2010*

Microsoft Visual Studio 2010 merupakan kelanjutan dari *Microsoft Visual Studio* sebelumnya, yaitu *Visual Studio. Net* 2003 yang diproduksi oleh Microsoft. Pada bulan Februari 2002 *Microsoft* memproduksi teknologi. *Net Framework* versi 1.0, teknologi. *Net* ini didasarkan atas susunan berupa *Net Framework*, sehingga setiap produk baru yang terkait dengan teknologi. *Net* akan selalu berkembang mengikuti perkembangan. *Net Frameworknya*. Pada perkembangannya nantinya mungkin untuk membuat program dengan teknologi. *Net* memungkinkan para pengembang perangkat lunak akan dapat menggunakan lintas sistem operasi, yaitu dapat dikembangkan di sistem operasi *windows* juga dapat dijalankan pada sistem operasi lain, misalkan pada sistem operasi *Linux*, seperti yang telah dilakukan pada pemograman *Java* oleh *Sun Microsystem*. Pada saat ini perusahaan-perusahaan sudah banyak meng*update* aplikasi lama yang dibuat *Microsoft Visual Basic* 6.0 ke teknologi. *Net* karena kelebihan-kelebihan

yang ditawarkan, terutama memungkinkan pengembang perngkat lunak secara cepat mampu membuat program *robust*, serta berbasiskan integrasi ke internet yang dikenal dengan *XsML Web Service* (Erick Kurniawan, 2011).