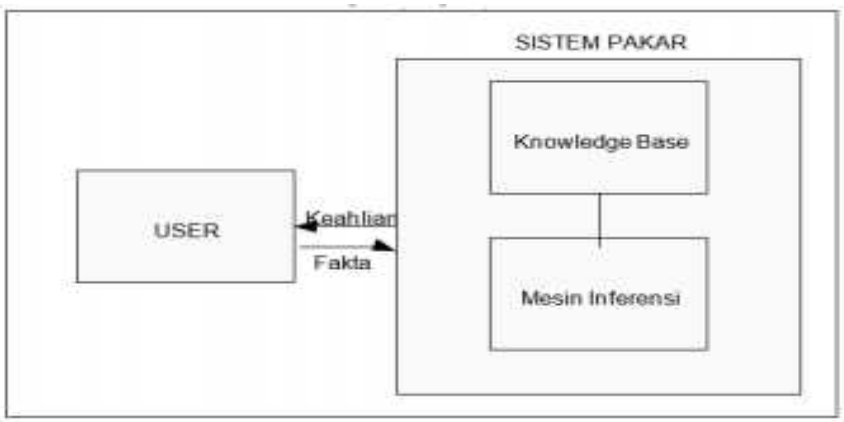
BAB II

TINJAUAN PUSTAKA

II.1. Sistem Pakar

Sistem pakar adalah cabang kecerdasan buatan yang menggunakan pengetahuan/*knowledge* khusus untuk memecahkan masalah pada level *human expert*/pakar. Sistem pakar banyak dikembangkan dalam berbagai ilmu, salah satu diantaranya dalam bidang kedokteran untuk melakukan diagnosa penyakit. Sistem pakar digunakan untuk menentukan diagnosa penyakit akan membantu mengkonfirmasi diagnosa dan menentukan saran dan terapinya. Menurut Turban (2005), sistem pakar adalah sistem informasi berbasis komputer yang menggunakan pengetahuan pakar untuk mencapai performa keputusan tingkat tinggi dalam domain persoalan sempit. Bagian dalam sistem pakar terdiri dari 2 komponen utama yakni berisi *knowledge base* yang berisi basis pengetahuan dan mesin *inferensi* yang menggambarkan kesimpulan. Kesimpulan tersebut merupakan *respons* dari sistem pakar atas permintaan pengguna.

Berikut menggambarkan konsep dasar suatu sistem pakar *knowledge based* pada gambar II.1 (Arief Kelik Nugroho ; 248 : 2013):



Gambar II.1. Konsep Sistem Pakar
Sumber : Arief Kelik Nugroho ; 248 : 2013

Komponen yang harus dimiliki dalam membangun sistem pakar adalah sebagai berikut :

(i) Antar Muka pengguna,

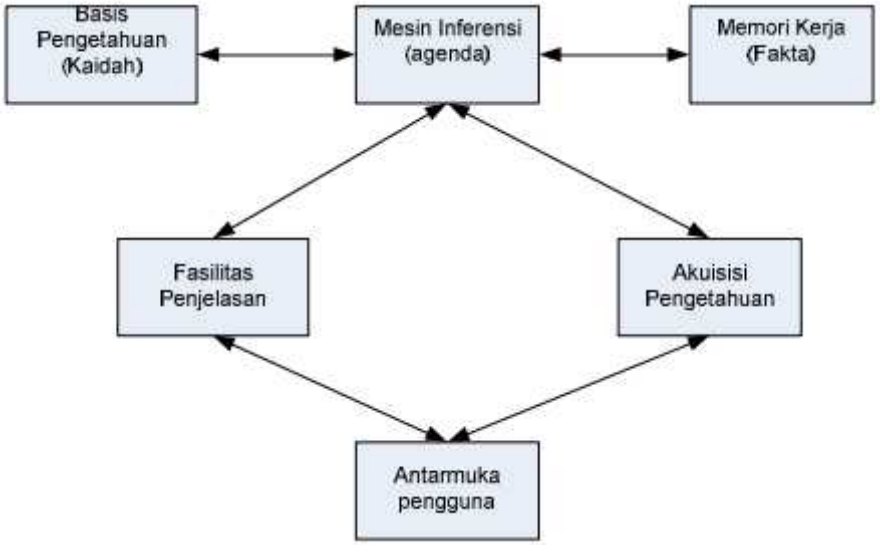
(ii) Basis Pengetahuan;

(iii) Mesin Inferensi;

(iv) Memori Kerja.

Sedangkan untuk menjadikan sistem pakar menjadi lebih menyerupai seorang pakar yang berinteraksi dengan pemakai, maka dilengkapi dengan fasilitas berikut : Fasilitas Penjelasan (*Explanation Facility*), Fasilitas Akuisisi Pengetahuan (*Knowledge Acquisition Faciility*).

Hal ini terlihat dalam struktur sistem pakar pada Gambar II.2 (Arief Kelik Nugroho ; 249 : 2013):



Gambar II.2. Struktur Sistem Pakar
Sumber : Arief Kelik Nugroho ; 249 : 2013

II.2. Metode *Teorema Bayes*

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

$$P(H|E)=\frac{P(E|H).P(H)}{P(E)}$$ ..... (1)

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. Teorema ini lebih

banyak diterapkan pada hal-hal yang berkenaan dengan probabilitas serta kemungkinan dari penyakit dan gejala-gejala yang berkaitan.

Misalnya seseorang menjalani tes klinik tersebut dan mendapatkan hasil positif, berapakah peluang bahwa ia benar-benar menderita penyakit langka tersebut? Dengan kata lain, kita mencoba untuk mencari peluang dari A, dimana B atau P (A | B).

Dari tabel di atas, dapat kita lihat bahwa P (A | B) adalah peluang dari positif yang benar dibagi dengan peluang positif (benar maupun salah), yaitu 0,0194 / (0,0194 + 0,0882) = 0,1803. Kita dapat juga mendapatkan hasil yang sama dengan menggunakan rumus *Teorema Bayes* di atas :

$$\boldsymbol{P}(A \mid B) = \frac{\boldsymbol{P}(B \cap A)}{\boldsymbol{P}(B)} \quad \text{.....(2)}$$

$$= \frac{\boldsymbol{P}(B \mid A) \times \boldsymbol{P}(A)}{\boldsymbol{P}(B \mid A)\boldsymbol{P}(A) + \boldsymbol{P}(B \mid A)\boldsymbol{P}(A)}$$

$$= \frac{97\% \times 2\%}{(97\% \times 2\%) + (9\% \times 98\%)}$$

$$= \frac{0.0194}{0.0194 + 0.0882}$$

$$= \frac{0.0194}{0.1076}$$

$$\boldsymbol{P}(A \mid B) = 0.1803$$

Hasil perhitungan ini sangat berbeda dengan intuisi kita di atas. Peluang bahwa orang yang mendapat hasil tes positif itu benar-benar menderita penyakit langka tidak sebesar yang kita bayangkan. Cuma ada sekitar 18% kemungkinan bahwa dia benar-benar menderita penyakit itu (Sri Rahayu ; 2013 : 131).

II.2.1. Langkah - Langkah Metode *Teorema Bayes*

Dalam melakukan perhitungan metode *Teorema Bayes*, maka diperlukan beberapa langkah – langkah dalam perhitungan metode *Teorema Bayes* seperti berikut:

1. Menentukan kriteria yang akan dinilai terlebih dahulu.
2. Kemudian menentukan data training.
3. Setelah data kriteria dan data training telah ditentukan, maka menentukan probabilitas dari setiap variabel.
4. Dari penilaian probabilitas maka akan dapat ditentukan nilai dari setiap karakter dan kriteria yang akan dinilai tersebut.
5. Kemudian dari nilai kriteria dapat diteruskan dengan mencari likelihood dan probabilitas dari masing – masing kriteria.
6. Adapun perhitungan mencari lokelihood dan probablitas dengan menggunakan rumus $P(H|E)=\frac{P(E|H).P(H)}{P(E)}$
7. Kemudian dapat ditentukan nilai tertinggi dari setiap data (Sri Rahayu ; 2013 : 131).

II.3. ERD

ERD adalah suatu diagram untuk menggambarkan desain konseptual dari model konseptual suatu basis data relasional. ERD juga merupakan gambaran yang merelasikan antara objek yang satu dengan objek yang lain dari objek di dunia nyata yang sering dikenal dengan hubungan antar entitas. Sebagai contoh jika membuat ERD dari sistem perpustakaan maka bahan sebagai objek ERD bisa berupa anggota, buku, peminjam, pengembalian dan sebagainya (Robi Yanto ; 2016 : 32).

II.4. Kamus Data

Kamus data umumnya berguna pada pengembangan model sistem dan dapat digunakan untuk menangani semua informasi dari semua tipe model sistem. Kamus data sederhana adalah daftar alfabetis dari nama-nama termasuk pada berbagai model sistem. Seperti namanya, kamus harus mencakup deskripsi yang berhubungan dengan entitas bernama tersebut dan, jika nama itu merepresentasikan objek komposit, mungkin saja ada deskripsi mengenai komposisinya. Informasi lain seperti tanggal pembuatan, pembuatnya dan representsi entitas juga dapat dimasukkan, tergantung pada tipe model yang sedang dikembangkan. Keuntungan penggunaan kamus data adalah :

1. Kamus data merupakan mekanisme untuk manajemen nama. Banyak orang yang harus menciptakan nama untuk entitas dan relasi ketika mengembangkan model sistem yang besar. Nama-nama ini harus dipakai secara konsisten dan tidak boleh bentrok. Perangkat lunak kamus data

dapat memeriksa keunikan nama dan memberitahu analisis persyaratan sekiranya terjadi duplikasi nama.

2. Kamus data berfungsi sebagai tempat penyimpanan informasi organisasional yang dapat menghubungkan analisis, desain, implementasi dan evolusi. Sementara sistem dikembangkan, informasi diambil untuk memberitahukan perkembangan informasi baru ditambahkan pada sistem. Semua informasi mengenai entitas berada pada satu tempat (Ian Sommerville ; 2012 : 151).

II.5. Normalisasi

Normalisasi adalah suatu proses untuk membuat data yang tidak normal menjadi data yang normal. Bentuk data yang tidak normal / data mentah biasa disebut juga *unnormalized form*. Masing – masing level normalisasi mempunyai aturan tersendiri.

1. First Normal Form

Suatu tabel dikatakan dalam keadaan *First Normal Form* (1NF) jika :

a. Tidak ada perulangan record data dalam tabel.
b. Setiap sel memiliki satu nilai saja. Artinya tidak ada perulangan group dan array.
c. Data yang diinputkan memiliki tipe data yang sama dengan tipe data kolom dalam tabel.

2. *Second Normal Form*

   Suatu tabel dikatakan dalam keadaan *Second Normal Form* (2NF) jika tabel tersebut sudah dalam keadaan *First Normal Form* (1NF) dan jika semua atribut yang bukan kunci tabel, baik *primary key* maupun *foreign key* tergantung pada semua kunci dalam tabel.

3. *Third Normal Form*

   Suatu tabel dikatakan dalam keadaan *Third Normal Form* (3NF) jika tabel tersebut sudah dalam keadaan *Second Normal Form* (2NF) dan jika tidak terdapat ketergantungan yang transitif. Artinya, data-data yang mungkin diisi berulang-ulang dapat dibuat sebuah tabel baru.

4. *Boyce-Codd Normal Form* (BCNF)

   Tabel dikatakan dalam keadaan *Boyce-Codd Normal Form* (BCNF) jika tabel tersebut dalam keadaan *Third Normal Form* (3NF) dan setiap determinan adalah kunci kandidat.

5. *Fourth Normal Form* (4NF)

   Suatu tabel dikatakan dalam keadaan *Fourth Normal Form* (4NF) jika tabel tersebut dalam keadaan *Boyce-Codd Normal Form* (BCNF) dan jika tidak terdapat ketergantungan nilai ganda.

6. *Fiveth Normal Form* (5NF)

   Tabel dikatakan dalam keadaan *Fiveth Normal Form* (5NF) jika tabel tersebut dalam keadaan *Fourth Normal Form* (4NF) dan jika setiap ketergantungan dalam join ada pada tabel sudah konsekuen dengan kunci kandidat pada tabel tersebut (Ema Utami ; 2012 : 73-76).

II.6. Database

Secara sederhana *database* (basis data/pangkalan data) dapat diungkapkan sebagai suatu pengorganisasian data dengan bantuan komputer yang memungkinkan data dapat diakses dengan mudah dan cepat. Pengertian akses dapat mencakup pemerolehan data maupun pemanipulasian data seperti menambah serta menghapus data. Dengan memanfaatkan komputer, data dapat disimpan dalam media pengingat yang disebut *harddisk.* Dengan menggunakan media ini, keperluan kertas untuk menyimpan data dapat dikurangi. Selain itu, data menjadi lebih cepat untuk diakses terutama jika dikemas dalam bentuk *database* (Agustinus Mujilan ; 2012 : 23).

II.7. SQL Server

SQL Server 2008 adalah sebuah terobosan baru dari Microsoft dalam bidang *database*. *SQL Server* adalah DBMS (*Database Management System*) yang dibuat oleh *Microsoft* untuk ikut berkecimpung dalam persaingan dunia pengolahan data menyusul pendahulunya seperti IBM dan Oracle. *SQl Server* 2008 dibuat pada saat kemajuan dalam bidang *hardware* sedemikian pesat. Oleh karena itu sudah dapat dipastikan bahwa *SQL Server* 2008 membawa beberapa terobosan dalam bidang pengolahan dan penyimpanan data. *Microsoft* merilis *SQL Server* 2008 dalam beberapa versi yang disesuaikan dengan segment-segment pasar yang dituju (Wenny Widya ; 2012 : 3).

II.8. Visual Basic 2008

VB.NET adalah salah satu bahasa pemrograman tingkat tinggi yang mendekati bahasa manusia. Kemunculan bahasa VB.NET ini sebagai jawaban untuk menyederhanakan bahasa pemrograman pada platform .NET yang diluncurkan tahun 2002 dan untuk menjembatani programmer Visual Basic. Bahasa VB.NET secara teknis mengadopsi sintak bahasa Visual Basic. Konsistensi API membuat bahasa VB.NET menjadi pilihan dalam membuat kode program diatas platform Windows. Fitur baru bahasa VB.NET dibandingkan Visual Basic bahwa bahasa VB.NET mendukung object-oriented dan juga dynamics programming. Ini menambah daftar kemudahan untuk belajar bahasa VB.NET.

Ibaratnya seperti ikan dan air yang tidak dipisahkan, ini sama halnya pada VB.NET dan .NET *Framework*. Bahasa VB.NET memerlukan .NET *Framework* agar dapat dikompilasi dan dijalankan. .NET *Framework* merupakan *Framework* yang membungkus kompleksitas OS *Windows* sehingga konsisten API dapat diperoleh dan tidak dipusingkan dengan beragam API tiap OS *Windows*. Buku ini tidak akan membahas .NET *Framework*. Pembaca dapat mempelajari buku yang khusus belajar mengenai .NET *Framework*. Pembaca juga dapat mengunjungi website resminya yaitu http://www.*microsoft*.com/net (Agus Kurniawan ; 2013 : 10).

II.9. UML (*Unified Modeling Language*)

Menurut Windu Gata (2013 : 4) Hasil pemodelan pada OOAD terdokumentasikan dalam bentuk *Unified Modeling Language* (UML). UML adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak.

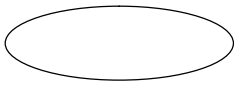
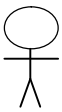
UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodelan umum dalam industri perangkat lunak dan pengembangan sistem.

Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah *use case* diagram, *class* diagram, *activity* diagram dan *sequence* diagram.

1. *Use case* Diagram

   Use case diagram merupakan pemodelan untuk melakukan (*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 :

Tabel II.1. 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 ; 2013 : 4)

2. *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 tanggung 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.

3. Tabel II.2. *Multiplicity Class Diagram*

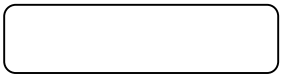
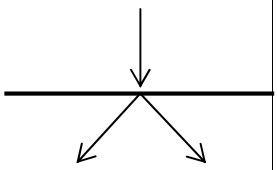
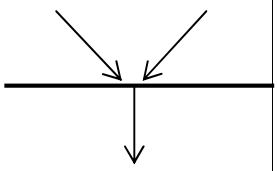
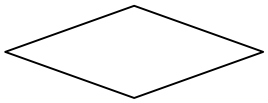
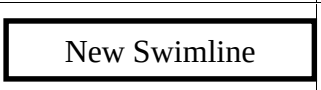
Multiplicity	**Penjelasan**
1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1 atau lebih
1..*	1 atau lebih
0..1	Boleh tidak ada, maksimal 1
n..n	Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimum 4

(Sumber : Windu Gata ; 2013 : 9)

3. 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 :

Tabel II.3. Simbol *Activity Diagram*

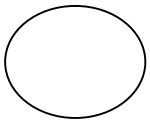
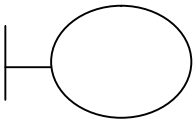
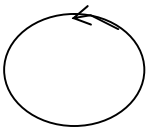
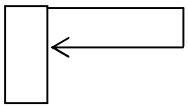
Gambar	Keterangan
	*Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	*End point*, akhir aktifitas.
	*Activites*, menggambarkan suatu proses/kegiatan bisnis.
	*Fork* (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan 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 ; 2013 : 6)

4. 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 :

Tabel II.4. Simbol *Sequence Diagram*

Gambar	Keterangan
	*Entity Class*, 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 ; 2013 : 7)