BAB II

LANDASAN TEORI

2.1 Sistem Pakar

Sistem Pakar (*expert system)* merupakan solusi AI (*Artificial Intelligence)* bagi masalah pemrograman pintar. Menurut Proffesor Edward Feigenbaum dari Standford University yang merupakan pionir dalam teknologi sistem pakar mendefinisikan sistem pakar sebagai sebuah program komputer pintar yang memanfaatkan pengetahuan dan prosedur inferensi untuk memecahkan masalah yang cukup sulit hingga membutuhkan keahlian khusus dari manusia. Dengan kata lain, sistem pakar adalah sistem komputer yang ditujukan untuk meniru semua aspek kemampuan pengambilan keputusan seorang pakar (Rosnelly, 2012). Menurut (Hayadi, 2018), sebuah program terdiri atas beberapa komponen yang mutlak harus ada.Komponen itu adalah sebagai berikut:

1. Basis Pengetahuan (*Knowledge Base)*

   Basis pengetahuan merupakan inti program sistem pakar karena basis pengetahuan ini merupakan representasi pengetahuan (*Knowledge Representation)* dari seorang pakar.

2. Basis Data

   Basis data adalah bagian yang mengandung semua fakta, baik fakta awal pada saat sistem mulai beroperasi maupun fakta yang didapatkan pada saat pengambilan kesimpulan sedang dilaksanakan

3. Mesin Inferensi

   Mesin *inferensi* adalah bagian yang mengandung mekanisme fungsi berfikir dan pola penalaran sistem yang digunakan oleh seorang pakar. Mekanisme ini akan menganalisa suatu masalah tertentu dan selanjutnya akan mencari jawaban atau kesimpulan yang terbaik. Mesin *inferensi* memulai pelacakannya dengan mencocokan kaidah dalam basis pengetahuan dengan fakta yang ada dalam basis data. Ada dua teknik *inferensi* yang ada yaitu pelacakan ke belakang (*Backward Chaining)* yang memulai penalaran dari kesimpulan hipotesa menuju fakta yang mengandung hipotesa tersebut. Teknik kedua yakni pelacakan ke depan (*Forward Chaining)* yang merupakan kebalikan dari pelacakan kebelakang yaitu memulai dari sekumpulan data menuju kesimpulan.

4. Antar Muka Pemakai (*User Interface)*

   Antar muka pemakai adalah bagian penghubung antara program sistem pakar dengan pemakainnya. Pada bagian ini akan terjadi dialog antara program dengan pemakai. Program akan mengajukan pertanyaan berbentuk “ya/tidak” (*yes or no question)* atau berbentuk menu pilihan. Melalui jawaban yang diberikan oleh pemakai, sistem pakar akan mengambil kesimpulan yang berupa informasi ataupun anjuran sesuai dengan sifat dari sistem pakar.

2.2 *Certainty Factor*

Teori Certainty Factor diusulkan oleh Shortliffe dan Buchanan pada 1975 untuk mengadopsi permasalahan ketidakpastian oleh seorang pakar. Metode Certainty Factor ini dipilih ketika menghadapi suatu permasalahan atau kejadian

yang tidak pasti dalam jawaban. Berikut rumus yang berlaku pada *Certainty Factor* (Ramadhan & Pane, 2018):

$$CF\ (P,E) = MB\ (P,E) - MD(P,E) \quad \text{..... (1)}$$

Rumus 2. 1 Persamaan 1

Keterangan:

CF = *certainty factor*

MB = *measure of belief*

MD = *measure of disbelief*

P = *probability*

E = *evidence or event*

Dalam metode *Certainty Factor* terdapat jenis perhitungan yang dapat digabungkan dalam sistem basis aturan MYCIN, berikut merupakan salah satu contoh perhitungan *Certainty Factor* menggunakan kombinasi:

Certainty Factor untuk kaidah dengan gejala premis/tunggal:

$$CF_{gejala} = CF[user] * CF[pakar] \quad \text{..... (2)}$$

Rumus 2. 2 Persamaan 2

Apabila terdapat kaidah dengan kesimpulan lebih dari satu gejala, maka CF selanjutnya dihitung dengan persamaan:

$$CF_{combine} = CF_{old} + CF_{gejala} * (1 - CF_{old}) \text{..... (3)}$$

Rumus 2. 3 Persamaan 3

Tabel 2. 1 Tabel Contoh Daftar Gejala

No	Daftar Gejala	Inflamasi Kulit		
		Eksim	Psoriasis	Atopik
1	Rasa panas bagian kulit yang terkena eksim(G01)	0.6		
2	Gatal (G02)	0.2		0.4
3	Kulit kering (G03)	0.6		
4	Bintil kecil (G04)	0.4		
5	Kulit bersisik (G05)	0.2		0.4
6	Bintik merah (G06)		0.6	
7	Sakit Sendi (G07)		0.4	
8	Bernanah (G08)		0.4	
9	Badan mengigil (G09)		0.4	
10	Kulit pecah-pecah (G10)			0.4

Terdapat sebuah pasien yang menderita gejala Gatal (G02), Kulit Kering(G03), Bintil kecil(G04). Bagaimana hasil diagnosa pasien tersebut?

CF(Gatal, Kulit kering) = 0.2 + 0,6(1-0,2) = 0,68 (CF Kombinasi)

CF(Kombinasi,Bintil Kecil) = 0,68 + 0,4(1-0,68) = 0,80 (Hasil CF)

CF(Gatal,0) = 0,4 + 0(1-0,4) = 0,4(Hasil CF)

Dari proses perhitungan yang telah dilakukan, maka pasien tersebut menderita penyakit Eksim dengan nilai kemungkinan 0,8 = 80%.

2.3 Android Studio

Android Studio adalah sebuah *Integrated Development Environment (IDE)* untuk pengembangan aplikasi android berdasarkan dari IntelliJ IDEA (Android Developer, 2018).

2.4 Java

Java adalah bahasa pemrograman tingkat tinggi yang memungkinkan pengembang untuk menulis program yang tidak tergantung pada jenis komputer tertentu. Bahasa tingkat tinggi lebih mudah dibaca, ditulis, dan dipelihara. Tetapi kode mereka harus diterjemahkan oleh kompiler atau ditafsirkan ke dalam bahasa mesin (tidak dapat dibaca oleh manusia karena terdiri dari angka) untuk dieksekusi, karena itu adalah satu-satunya bahasa yang dimengerti komputer (Cosmina, 2019).

2.5 PHP

PHP (akronim rekursif untuk PHP: Hypertext Preprocessor) adalah bahasa *scripting open source* yang banyak digunakan yang sangat cocok untuk pengembangan web dan dapat dimasukkan ke dalam HTML.

Hal terbaik dalam menggunakan PHP adalah sangat sederhana bagi pendatang baru, tetapi menawarkan banyak fitur canggih untuk programmer profesional (PHP, 2020).

2.6 *Rapid Application Development* (RAD)

Sistem yang semakin komplek dan waktu pengembangan yang dibutuhkan semakin cepat, membuat para pengembang sistem berfikir keras dan berusaha untuk mencari solusi teknik pengembangan yang cepat tanpa mengurangi kualitas sistem

yang dihasilkan. Dengan kondisi ini, dikembangkanlah *Rapid Application Development. Rapid Application Development* (RAD) merupakan metode yang memfokuskan pada kecepatan dalam pengembangan sistem untuk memenuhi kebutuhan pengguna atau pemilik sistem seperti *prototyping* namun mempunyai cangkupan yang lebih luas (Mulyani, 2016).

2.7 *The Eight Golden Rules of Interface Design*

Bagian ini memfokuskan perhatian pada delapan prinsip, yang disebut *golden rule,* yang berlaku di sebagian besar sistem interaktif dan lingkungan yang diperkaya. Prinsip-prinsip ini, berasal dari pengalaman dan disempurnakan selama tiga dekade, membutuhkan validasi dan penyetelan untuk domain desain tertentu. Menurut (Shneiderman, et al., 2016) *Eight golden rules* meliputi:

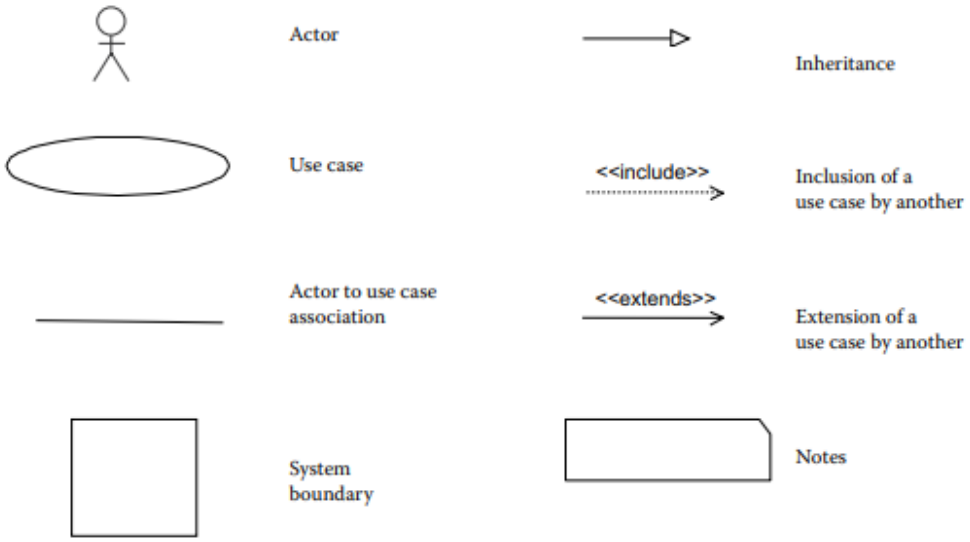
1. *Strive for consistency.* Urutan tindakan yang konsisten harus diminta serupa. Terminologi yang identik harus digunakan pada prompt, menu, dan layar bantuan; dan warna, tata letak, huruf besar, font, dan sebagainya yang konsisten, harus diberitahukan secara menyeluruh.
2. *Seek universal usability.* Kenali kebutuhan beragam pengguna dan desain untuk plastisitas, memfasilitasi transformasi konten. Perbedaan pemula hingga pakar, rentang usia, kecacatan, internasional, ariasi, dan keanekaragaman teknologi masing-masing memperkaya spektrum persyaratan yang memandu desain. Menambahkan fitur untuk pemula, seperti penjelasan, dan fitur untuk para ahli, seperti pintasan dan langkah lebih cepat, memperkaya desain antarmuka dan meningkatkan kualitas yang dirasakan.

3. *Offer informative feedback.* Untuk setiap tindakan pengguna, harus ada *feedback* antarmuka. Untuk tindakan yang sering dan kecil, responsnya bisa sederhana, sedangkan untuk tindakan yang jarang dan besar, responsnya harus lebih substansial. Presentasi visual dari objek yang menarik menyediakan lingkungan yang nyaman untuk menunjukkan perubahan secara eksplisit.
4. *Design dialogs to yield closure.* Urutan tindakan harus diatur ke dalam kelompok dengan awal, tengah, dan akhir. *Feedback* yang informatif pada penyelesaian sekelompok tindakan memberikan pengguna kepuasan atas pencapaian, rasa lega, sinyal untuk menjatuhkan rencana yang sesuai dari pikiran mereka, dan indikator untuk mempersiapkan kelompok tindakan selanjutnya. Misalnya, situs web e-commerce memindahkan pengguna dari memilih produk ke checkout, diakhiri dengan halaman konfirmasi yang jelas yang menyelesaikan transaksi.
5. *Prevent errors.* Sebisa mungkin, rancang antarmuka agar pengguna tidak dapat melakukan kesalahan serius; misalnya, hapus abu-abu item menu yang tidak sesuai dan tidak memungkinkan karakter alfabet dalam bidang entri numerik.
6. *Permit easy reversal of actions.* Sebisa mungkin, tindakan harus dapat dibalik. Fitur ini mengurangi kecemasan, karena pengguna tahu bahwa kesalahan dapat diurungkan, dan menghapus eksplorasi opsi yang tidak dikenal.

7. *Keep users in control.* Pengguna berpengalaman sangat menginginkan perasaan bahwa mereka bertanggung jawab atas antarmuka dan bahwa antarmuka merespons tindakan mereka. Mereka tidak ingin kejutan atau perubahan dalam perilaku yang akrab, dan mereka terganggu oleh urutan entri data yang membosankan, kesulitan dalam mendapatkan informasi yang diperlukan, dan ketidakmampuan untuk menghasilkan hasil yang diinginkan.
8. *Reduce short-term memory load.* Kapasitas manusia yang terbatas untuk pemrosesan informasi dalam memori jangka pendek (aturan praktisnya adalah bahwa orang dapat mengingat "tujuh plus atau minus dua potongan" informasi) mengharuskan perancang menghindari antarmuka di mana pengguna harus mengingat informasi dari satu layar dan kemudian menggunakan informasi itu pada tampilan lain. Ini berarti bahwa ponsel tidak perlu memasukkan kembali nomor telepon, lokasi situs web harus tetap terlihat, dan formulir yang panjang harus dipadatkan agar sesuai dengan satu tampilan.

2.8 *Use Case Diagram*

Diagram *Use Case* adalah salah satu model persyaratan sistem pada level tinggi. Diagram *use case* tertutama digunakan untuk memvisualisasikan *use case*, terkait sektor, dan interaksinya. *Use case* diagram juga bisa mempunyai definisi yaitu memberikan tinjauan umum fungsionalitas sistem atau proses bisnis dan perspektif pengguna. *Use case* diagram idealnya untuk melibatkan pengguna dan



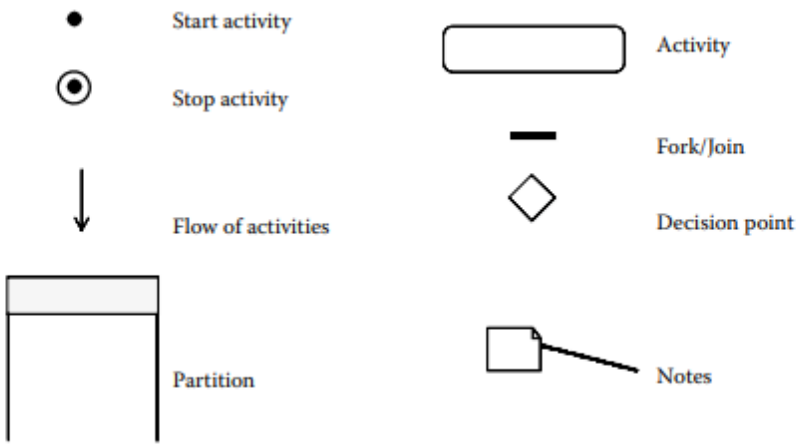
Gambar 2. 1 Notasi *Use Case Diagram*

memastikan partisipasi mereka dalam persyaratan pemodelan. Berikut merupakan notasi dari *use case* diagram.

Pada gambar 2.1, terlihat beberapa notasi yang sering dipakai untuk memvisualisasikan suatu *use case*. *Actor* bisa disebut sebagai *user*, *use case* adalah sebuah deskripsi tentang apa yang dikerjakan sistem. *association* menggambarkan interaksi antara *actor* dan *use case*, *system boundary* merupakan tempat aktivitas sistem dijalankan, *inheritance* disebut juga relasi antar *use case*, *include* dipakai ketika bagian dari perilaku yang didokumentasikan dalam *use case* digunakan kembali oleh sistem lainnya, *extends* berguna untuk menambah fungsionalitas ke *use case* yang ada, dan *notes* digunakan untuk menjelaskan informasi mengenai hubungan antara *use case* (Unhelkar, 2018).

2.9 *Activity Diagram*

Activity diagram memodelkan aliran, atau proses, dalam suatu sistem. Karena itu, *activity diagram* seperti bagan alur. Pemodelan aliran ini dapat dilakukan pada tingkat proses bisnis, dalam kasus penggunaan, dan kadang-kadang di antara kasus penggunaan. Aktivitas-aktivitas tersebut ada pada tingkat teknis terperinci atau pada tingkat bisnis. *Activity diagram* mendokumentasikan perilaku internal dalam *use case*, antara *use case*, atau bisnis secara keseluruhan. Sebuah diagram aktivitas tingkat tinggi digunakan sebagai diagram konteks yang menunjukkan bagaimana berbagai proses bisnis saling terkait

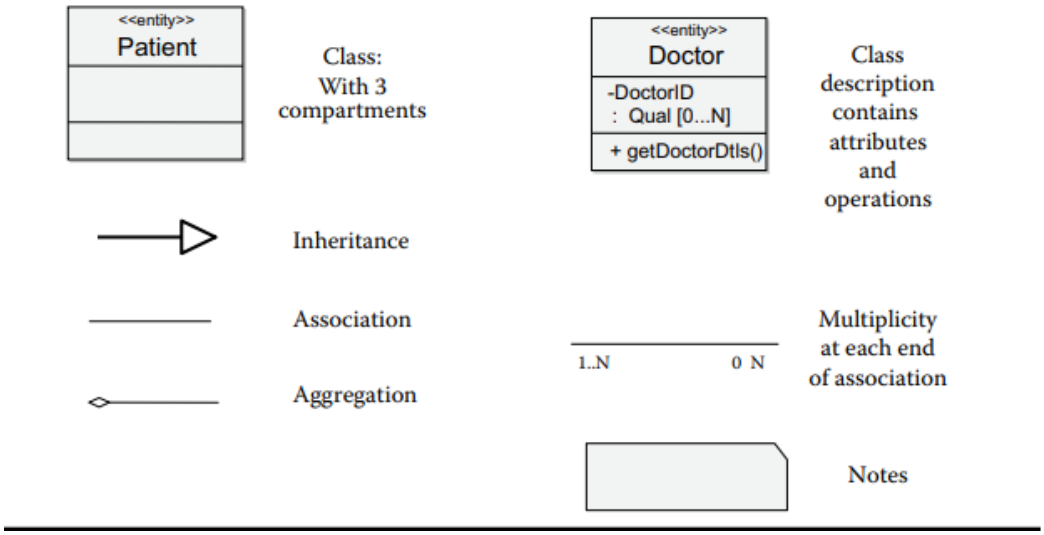


Gambar 2. 2 Notasi *Activity Diagram*

Pada gambar 2.2, terlihat notasi yang sering dipakai dalam suatu *activity diagram. Start activity* dan *stop activity* adalah sebagai penanda suatu aktivitas dimulai (*start)* dan aktivitas selesai (*stop*), *flow of activites* adalah hubungan antar aktivitas, *partition* juga dikenal sebagai *swimlanes* di versi UML sebelumny, *activity* menggambarkan langkah-langkah pada aktivitas, *fork/join* menunjukkan kegiatan yang tidak saling bergantung tetapi yang perlu dilakukan secara bersamaan, *decision point* memungkinkan untuk perabangan kegiatan berdasarkan kondisi yang ditentukan, *notes* digunakan untuk menjelaskan informasi mengenai *activity* (Unhelkar, 2018).

2.10 *Class Diagram*

Class diagram mewakili entitas utama dalam bisnis serta domain teknis. *Class diagram* sangat struktural dan statis. *Class diagram dapat* menampilkan kelas tingkat bisnis, serta kelas teknis yang berasal dari bahasa implementasi (seperti Java, C, C++). *Class diagram* juga menunjukkan hubungan antar kelas. Seluruh deskripsi kelas atau entitas dan hubungan yang akan mereka miliki satu sama lain adalah statis. Tidak ada ketergantungan yang ditunjukkan dalam diagram ini dan tidak ada konsep waktu



Gambar 2. 3 Notasi pada *Class Diagram*

Pada gambar 2.3 ada beberapa notasi yang digunakan pada *class diagram*. *Class* berisi deskripsi mengenai atribut dan operasi yang dilakukan oleh *class* tersebut, *inheritance* adalah hubungan antar *class* yang mempunyai level lebih tinggi diwarisi oleh *class* yang lebih rendah, *Association* adalah hubungan antar *class* dengan makna umum, *notes* digunakan untuk menjelaskan informasi mengenai *class* tersebut (Unhelkar, 2018).

2.11 *Mean Opinion Score*

Mean Opinion Score (MOS) adalah satuan kualitas suara yang biasa digunakan. Metode MOS adalah hasil survey dari percakapan dimana nilai rata-rata kualitas suara antara 1 sampai 5, dimana 1 berarti buruk dan 5 adalah yang paling baik. Sebelum adanya teknologi *VoIP*, metode MOS sudah lebih dulu ada dan digunakan dalam pengukuran kualitas komunikasi telepon analog (FITRIYANTI, LINDAWATI, & ARYANTI, 2018)

2.12 Penelitian Terdahulu

Pada tabel 2.2 menunjukkan hasil dari penelitian-penelitian terdahulu untuk mendukung penelitian ini. Adapun penelitian terdahulu tersebut yaitu:

Tabel 2. 2 Hasil Penelitian Terdahulu

No	Penulis	Judul	Nama Jurnal	Hasil
1	Raharjo, Joko S Dwi; Damiyana, Damdam; Hidayatullah, Miftach	*Sistem Pakar Diagnosa Penyakit Lambung dengan Metode Forward Chaining Berbasis Android* (Raharjo, Damiyana, & Hidayatullah, 2016)	JURNAL SISFOTEK GLOBAL	Penyakit lambung yang dibahas yaitu maag, dispesia, GERD, kanker lambung, gastroparesis, dan tukak lambung. Pada penelitian ini, metode yang dipakai yaitu *Forward Chaining* yang akan diimplementasi pada aplikasi android. Hasil akhir dari penelitian ini yaitu mengetahui apakah *user* mengalami penyakit lambung atau tidak.

2	Pratama, Vincentius Andrew; Natalia, Friska	*A Dempster-Shafer Approach to an Expert System Design in Diagnosis of Febrile Disease* (Pratama & Natalia, 2017)	2017 4th International Conference on New Media Studies	Penelitian ini dirancang dalam bentuk aplikasi yang dapat mendiagnosa penyakit demam. Penyakit demam yang termasuk dalam penelitian yaitu *Typhus, Measles, Dengue Fever*. Metode yang digunakan yaitu *Dempster Shaffer*. Hasil akhir dari penelitian merupakan hasil dari diagnosa 3 penyakit demam tersebut, informasi mengenai 3 penyakit demam, serta informasi mengenai jadwal praktik dokter.
3	Halim, Stephanie; Hansun, Seng	*Penerapan Metode Certainty Factor dalam Sistem Pakar Pendeteksi Resiko Osteoporosis dan Osteoarthritis* (Halim & Hansun, 2016)	Jurnal ULTIMA Computing	Penelitian ini dirancang untuk membuat suatu aplikasi yang dapat mendiagnosa penyakit osteoporosis dan osteoarthritis. Metode yang digunakan adalah *Certainty Factor* ,hasil akhirnya berupa diagnosa pasien tersebut dengan keakuratan presentase sebesar 80% menggunakan metode *Certainty Factor*.