BAB II

TINJAUAN PUSTAKA

II.1 Kecerdasan Buatan

II.1.1. Definisi Kecerdasan Buatan

Kecerdasan buatan berasal dari bahasa Inggris "*Artificial Intelligence"* atau disingkat AI, yaitu *intelligence* adalah kata sifat yang berarti cerdas, sedangkan *artificial* artinya buatan. Kecerdasan buatan yang dimaksud di sini menunjuk pada mesin yang mampu berfikir, menimbang tindakan yang akan diambil, dan mampu mengambil keputusan seperti yang dilakukan oleh manusia (T.Sutojo, dkk ; 2011 :1). Berikut adalah beberapa definisi kecerdasan buatan yang telah didefinisikan oleh beberapa ahli :

- Hebert Alexander Simon (June 15, 1916 February 9, 2001) :

  "Kecerdasan buatan (*Artificial Intelligence*) merupakan kawasan penelitian, aplikasi, dan instruksi yang terkait dengan pemograman komputer untuk melakukan sesuatu hal yang dalam pandangan adalah cerdas".

- Rich and Knight (1991) :

  "Kecerdasaan buatan (AI) merupakan sebuah studi tentang bagaimana membuat computer melakukan hal-hal yang pada saat ini dapat dilakukan lebih baik oleh manusia".

- Encyclopedia Britannica

  "Kecerdasan buatan (AI) merupakan cabang dari ilmu komputer yang dalam merepresentasi pengetahuan lebih banyak menggunakan simbol-simbol

daripada bilangan dan memproses informasi berdasarkan metode heuristic atau dengan berdasarkan sejumlah aturan".

- Menurut Winston dan Prendergast (1984), tujuan dari kecerdasan buatan adalah :

  1. Membuat mesin menjadi lebih pintar (tujuan utama)
  2. Memahami apa itu kecerdasan (tujuan ilmiah)
  3. Membuat mesin lebih bermanfaat (tujuan *entrepreneurial*)

II.2. Sistem Pakar

II.2.1. Definisi Sistem Pakar

Sistem adalah sekelompok unsur yang erat hubungannya satu dengan yang lain, yang berfungsi bersama-sama untuk mencapai tujuan tertentu (Tata Sutabri ; 2005 : 8).

Pakar adalah seseorang yang mempunyai pengetahuan dan metode khusus serta mampu menerapkannya untuk memecahkan masalah atau memberi nasehat (T.Sutojo, dkk ; 2011:163).

Sistem pakar merupakan cabang dari *Artificial Inteliligence(AI)* yang cukup tua karana sistem ini mulai dikembangkan pada pertengahan 1960. Sistem pakar yang muncul pertama kali adalah *General-purpose problem solver* (GPS) yang di kembangkan oleh Newel dan Simon (T.Sutojo, dkk ;2011:159).

Menurut beberapa ahli sistem pakar adalah

- Sistem pakar adalah sebuah sistem yang menggunakan pengetahuan manusia dimana pengetahuan tersebut dimasukkan ke dalam sebuah komputer dan

kemudian digunakan untuk menyelesaikan masalah-masalah yang biasanya membutuhkan kepakaran atau keahlian manusia (Turban (2001,p402)).

- Sistem pakar adalah program komputer yang mempresentasikan dan melakukan penalaran dengan pengetahuan beberapa pakar untuk memecahkan masalah atau memberikan saran (Jackson (1993,p3)).
- Sistem pakar adalah program yang berbasiskan pengetahuan yang menyediakan solusi 'kualitas pakar' kepada masalah-masalah dalam bidang (domain) yang spesifik (Luger dan Stubblefield (1993, p308)).

Sistem Pakar (*Expert System*) adalah sistem yang dirancang untuk dapat menirukan keahlian seorang pakar dalam menjawab pernyataan dan memecahkan suatu masalah. Sistem pakar akan memberikan pemecahan suatu masalah yang didapat dari dialog dengan pengguna(T. Sutojo, dkk;2010:13).

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

II.2.2. Manfaat Sistem Pakar

Sistem pakar menjadi sangat populer karena sangat banyak kemampuan dan manfaat yang diberikan, diantaranya:

1. Meningkatkan produktivitas, karena Sistem Pakar dapat bekerja lebih cepat daripada manusia.
2. Membuat seorang yang awan bekerja seperti layaknya seorang pakar.

3. Meningkatkan kualitas, dengan memberikan nasehat yang konsisten dan mengurangi kesalahan.
4. Mampu menangkap pengetahuan dan kepakaran seseorang.
5. Memudahkan akses pengetahuan seorang pakar.
6. Meningkatkan kapabilitas sistem komputer. Integritas Sistem Pakar dengan sistem komputer lain membuat sistem lebih efektif dan mencangkup lebih banyak sistem.
7. Mampu bekerja dengan informasi yang tidak lengkap atau tidak pasti. Berbeda dengan sistem komputer konvensional, Sistem Pakar dapat bekerja dengan informasi yang tidak lengkap.
8. Bisa digunakan sebagai media pelengkap dalam pelatihan. Pengguna pemula bekrja dengan Sistem Pakar akan menjadi lebih berpengalaman karena adanya fasilitas penjelas yang berfungsi sebagai guru.
9. Meningkatkan kemampuan untuk menyelesaikan masalah karena Sistem Pakar mengambil sumber pengetahuan dari banyak pakar.

II.2.2. Kekurangan Sistem Pakar

Selain manfaat, sistem pakar juga memiliki beberapa kelemahan, diantaranya:

1. Memerlukan biaya yang sangat mahal untuk membuat dan melihatnya.
2. Sulit dikembangkan karena keterbatasan keahlian dan ketersediaan pakar.
3. Sistem Pakar tidak selamanya 100% bernilai benar (T.Sutojo, dkk; 2011:161).

II.2.3. Karakteristik Sistem Pakar

Sistem Pakar pada umumnya dirancang untuk memenuhi beberapa karakteristik umum berikut ini :

- **Kinerja sangat baik** (*high performance*). Sistem harus mampu memberikan respon berupa saran (*advice*) dengan tingkat kualitas yang sama dengan seorang pakar atau melebihinya.
- **Waktu respon yang baik** (*adequate respon time*). Sistem juga harus mampu bekerja dalam waktu yang sama baiknya (*reasonable*) atau lebih cepat dibandingkan dengan seorang pakar dalam menghasilkan keputusan.
- **Dapat diandalkan** (*good reliability*). Sistem harus dapat diandalkan dan tidak mudah rusak/ crash.
- **Dapat dipahami** (*understandable*). Sistem harus mampu menjelaskan lngkah-langkah penalaran yang dilakukannya seperti seorang pakar.
- **Fleksibel** (*fleksibility*). Sistem harus menyediakan mekanisme untuk menambah, mengubah, dan menghapus pengetahuan (Rika Rosnelly; 2012 :20-21).

II.2.4. Konsep Dasar Sistem Pakar

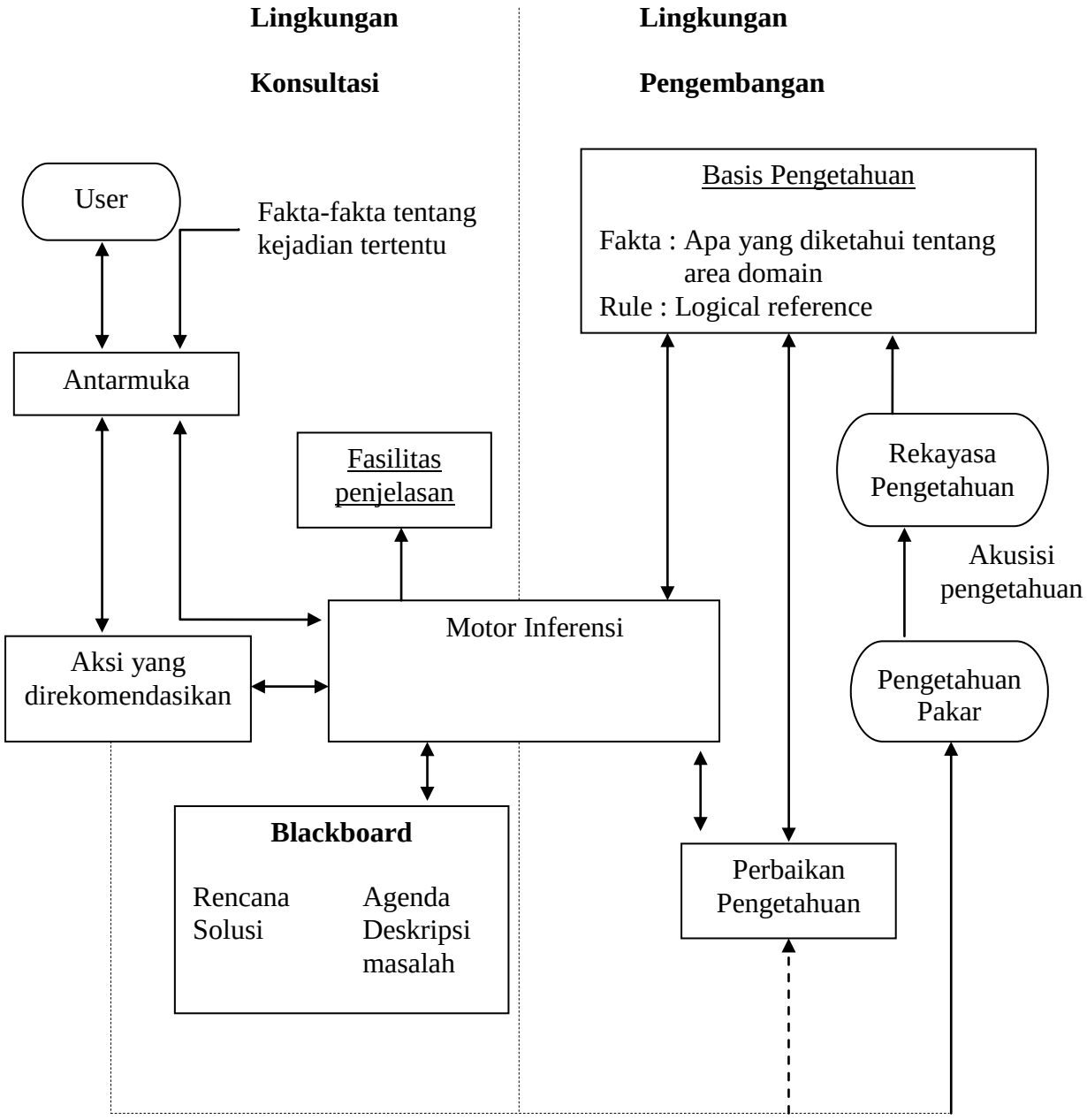
Konsep dasar Sistem Pakar meliputi enam hal berikut ini.

1. Kepakaran (*Expertise*), yaitu suatu pengetahuan yang diperoleh dari pelatihan, membaca dan pengalaman. Kepakaran inilah yang memungkinkan para ahli dapat mengambil keputusan lebih cepat dan tepat daripada seseorang yang bukan pakar.

2. Pakar (*Expert*), yaitu seseorang yang mempunyai pengetahuan, pengalaman dan metode khusus serta mampu menerapkannya untuk memecahkan masalah atau memberi nasehat.
3. Pemindahan Kepakaran (*Transferring Expertise*), yaitu proses pemindahan kepakaran dari seorang pakar ke dalam komputer kemudian mentransferkannya kepada orang lain yang bukan pakar.
4. Inferensi (*Inferencing*), yaitu sebuah prosedure (Program) yang mempunyai kemampuan dalam melakukan penalaran.
5. Aturan-aturan (*Rule*), yaitu pengetahuan disimpan dalam bentuk *rule*, sebagai prosedur-prosedur pemecahan masalah.
6. Kemampuan Menjelaskan (*Explanation Capability*) (T.Sutojo;2010:163-165).

II.2.5. Struktur Sistem Pakar

Ada dua bagian penting dari sistem pakar, yaitu lingkungan pengembangan (*development environment*) dan lingkungan konsultasi (*consultation enviroment*). Lingkungan pengembangan digunakan oleh pembuat sistem pakar untuk memnbangun komponen-komponennya dan memperkenalkan pengetahuan ke dalam *knowledge base* (basis pengetahuan). Lingkungan konsultasi digunakan pengguna untuk berkonsultasi sehingga pengguna mendapatkan pengetahuan dan nasihat dari sistem pakar layaknya berkonsultasi dengan seorang pakar. Gambar II.1 menunjukkan komponen-komponen yang penting dalam sebuah sistem pakar.



Gambar II.1 Struktur sistem pakar
Sumber : (T.Sutojo,dkk ;2011:167)

1. **Akuisisi Pengetahuan**

   Subsistem ini digunakan untuk memasukkan pengetahuan dari seorang pakar dengan cara merekayasa pengetahuan agar bisa diproses oleh komputer dan menaruhnya ke dalam basis pengetahuan dengan format tertentu (dalam

bentuk representasi pengetahuan). Sumber-sumber pengetahuan bisa diperoleh dari pakar,buku,dokumen,multimedia,basis data, laporan riset khusus, dan informasi yang terdapat di Web.

2. **Basis Pengetahuan (*Knowledge Base*)**

   basis pengetahuan mengandung pengetahuan yang diperlukan untuk memahami, memformulasikan, dan menyelesaikan masalah. Basis pengetahuan terdiri dari dua elemen dasar, yaitu:

   a. Fakta, misalnya situasi, kondisi, atau permasalahan yang ada
   b. Rule (Aturan), untuk mengarahkan pengguna pengetahuan dalam memecahkan masalah.

3. **Mesin Inferensi (*Inference Engine*)**

   Mesin inferensi adalah sebuah program yang berfungsi untuk memandu proses penalaran terhadap suatu kondisi berdasarkan pada basis pengetahuan yang ada, memanipulasi dan mengarahkan kaidah, model, dan fakta yang disimpan dalam basis pengetahuan untuk mencapai solusi atau kesimpulan. Dalam prosesnya, mesin inferensi menggunakan strategi pengendalian, yaitu strategi yang berfungsi sebagai panduan arah dalam melakukan proses penalaran. Ada tiga teknik pengendalian yang digunakan, yaitu *forward chaining*, *backward chaining*, dan gabungan dari kedua teknik tersebut.

   - **Forward Chaining**

     Forward chaining adalah teknik pencarian yang dimulai dengan fakta yang diketahui, kemudian mencocokan fakta-fakta tersebut dengan bagian IF dari *rules* IF-THEN. Bila ada fakta yang cocok dengan bagian IF, maka

rule tersebut dieksekusi. bila sebuah rule dieksekusi, maka sebuah fakta baru (bagian THEN) ditambahkan ke dalam database. setiap kali pencocokan, di mulai dari rule teratas. Setiap rule hanya boleh dieksekusi sekali saja. proses pencocokan berhenti bila tidak ada lagi rule yang bisa dieksekusi.

- **Backward Chaining**

  Backward chaining adalah metode inferensi yang bekerja mundur ke arah kondisi awal. proses diawali dari goal (yang berada dibagian THEN dari rule IF-THEN), kemudian pencarian mulai dijalankan untuk mencocokan apakah fakta-fakta yang ada cocok dengan premis-premis di bagian IF. Jika cocok, rule dieksekusi, kemudian hipotesis di bagian THEN ditempatkan di basis data sebagai fakta baru.

4. Daerah Kerja

Untuk merekam hasil sementara yang akan dijadikan sebagai keputusan dan untuk menjelaskan sebuah masalah yang sedang terjadi, sistem pakar membutuhkan *Blackboard*, yaitu area pada memori yang berfungsi sebagai basis data. Tiga tipe keputusan yang dapat direkam pada *blackboard,* yaitu:

a. rencana : bagaimana menghadapi masalah
b. agenda : aksi-aksi potensial yang sedang menunggu untu dieksekusi
c. solusi : calon aksi yang akan dibangkitkan

5. Antarmuka Pengguna

Digunakan sebagai media komunikasi antara pengguna dan sistem pakar. Komunikasi ini paling bagus bila disajikan dalam bahasa alami (*natural*

language) dan dilengkapi dengan grafik, menu, dan formulir elektronk. Pada bagian ini akan terjadi dialog antara sistem pakar dan pengguna.

6. **Subsistem Penjelasan**

   Berfungsi memberi penjelasan kepada pengguna, bagaimana suatu kesimpulan dapat diambil. Kemampuan seperti ini sangat penting bagi pengguna untuk mengetahui proses pemindahan keahlian pakar maupun dalam pemecahan masalah.

7. **Sistem Perbaikan pengetahuan (*Knowledge Refining System*)**

   Kemampuan memperbaiki pengetahuan (*knowledge refining system*) dari seorang pakar diperlukan untuk menganalisis pengetahuan. belajar dari kesalahan masa lalu, kemudian memperbaiki pengetahuannya sehingga dapat dipakai pada masa mendatang. Kemampuan evaluasi diri seperti itu diperlukan oleh program agar dapat menganalisis alasan-alasan kesuksesan dan kegagalannya dalam mengambil kesimpulan. dengan cara ini basis pengetahuan yang lebih baik dan penalaran yang lebih efektif akan dihasilkan.

8. **Pengguna (*User*)**

   Pada umumnya pengguna sistem pakar bukanlah seorang pakar (*non-expert*) yang membutuhkan solusi, saran, atau pelatihan (*training*) dari berbagai permasalahan yang ada.

II.3. Penyakit Cacingan

Penyakit cacingan pada manusia disertai oleh hampir 80%, penduduk Indonesia baik anak-anak maupun dewasa. Penyakit ini sering mengenai anak usia

balita dan sekolah dasar, serta orang dewasa yang bekerja didaerah pertambangan atau pertanian. Cacing dewasa hidup di dalam rongga usus kecil dan mulutnya selalu melekat diselaput lendir usus.

Cacingan merupakan penyakit khas daerah tropis yang penularan paling parahnya terjadi saat musim hujan, dengan sanitasi lingkungan yang masih buruk atau disebabkan oleh banjir, meluapnya sungai dan selokan, akhirnya larva-larva cacing menyebar ke berbagai tempat yang sangat mungkin bersentuhan dengan manusia.

Larva cacing masuk ke tubuh manusia melalui kontak langsung, seperti anak-anak yang bermain tanpa menggunakan alas kaki di daerah-daerah bekas aliran banjir atau pun saluran air yang meluap. Selain itu penularan penyakit cacingan juga bisa melalui makanan yang terkontaminasi oleh larva cacing.

Larva cacing biasanya tidak akan menetas selama tidak masuk ke tubuh manusia, larva ini dapat hidup selama berminggu-minggu dimana saja meski belum menemukan inangnya. Baru setelah larva tersebut masuk ke tubuh manusia, ia akan menetas di usus dan memakan makanan yang dicerna oleh manusia. Sehingga usus tidak bisa menyerap makanan tersebut karena sudah dibajak lebih dulu oleh cacing, hal inilah yang mengakibatkan tubuh manusia menjadi lemah, kurus, dan menurunkan daya tahan tubuh sehingga penyakit-penyakit lain mulai menyerang. Selain itu, jika seseorang dibiarkan terlalu lama terjangkit penyakit cacingan, hal tersebut dikhawatirkan dapat mengalami kelemahan fisik dan intelektualitas. Selain itu, cacingan juga dapat menyebabkan gangguan gizi serta anemia (Dr.Vanny Bernaddus, dkk : 2014).

Berberapa jenis cacing yang sangat potensial untuk menimbulkan infeksi pada anak-anak. Dan untuk selanjutnya mereka akan menjadi sumber penularan bagi infeksi berikutnya yang sangat potensial. Keadaan yang demikian inilah yang menyebabkan infeksi akibat parasit cacing sukar diatasi secara tuntas. Penderita yang tidak mendapatkan pengobatan yang tepat, merupakan sumber penularan bagi orang-orang dekat di sekitarnya (Qorry ‘Aina Abata ;2013:190).

II.3.1. Cacing gelang

Cacing betinanya yang panjangnya kira-kira 20-30 cm ini mampu bertelur 200.000 telur per harinya. Dalam waktu lebih kurang 3 minggu telur ini akan berisi larva yang bersifat infektif, yang dapat menjadi sumber penularan jika secara tidak sengaja mencemari makanan/ minuman yang kita konsimsi. Cacing ini hidup sebagai parasit dalam usus halus, sehingga akan mengambil nutrisi yang bermanfaat bagi tubuh kita dan menimbulkan kerusakan pada lapisan usus tersebut. Akhirnya timbullah diare dan gangguan penyerapan sari-sari makanan tersebut. Bahkan pada keadaan yang berat, larva dapat masuk ke paru-paru sehingga membutuhkan tindakan khusus (Qorry ‘Aina Abata ;2013:190-191).

II.3.2. Cacing cambuk (*Trichuris trichiura*)

Cacing ini juga menghisap sari makanan yang kita makan. Dia menghisap darah dan hidup didalam usus besar. Cacing betinanya dapat beratelur 5 ribu - 10 ribu butir per hari. Biadanya infeksi cacing ini menyerang pada usus besar. Infeksinya sering menimbulkan perlakaan usus, karen kepala cacing dimasukkan ke dalam permukaan usus penderita. Pada infeksi ringan biasanya hanya timbul diare saja. Tetapi pada infeksi yang berat, hampir pada sebagian pemukaan usus

bbesar dapat ditemukan cacing jenis ini. Akibatnya diare yang terjadi relative berat dan dapat belangsung terus menerus. Karena juga dapat menyebabkan perlukaan usus, maka anemia sebagai komplikasi pendarahan merupakan akibat yang tidak begitu saja dapat dianggap ringan (Qorry ‘Aina Abata ;2013:191).

II.3.3. Cacing tambang (*Necator americanus* dan *Ancylostoma duodenale*)

Cacing tambang (*Necator americanus* dan *Ancylostoma duodenale*) merupakan cacing yang paling ganas karena ia menghisap darah. Cacing ini betinanya biasa bertelur 15 ribu-20 ribu butir per hari. Penularannya cepat karena larva cacing tambang, sanggup menembus kulit kaki dan selanjutnya terbawa oleh pembuluh darah ke dalam usus. Cacing dewasa bertahan hidup 2-10 tahun. Cacing tambang ini menimbulkan luka pada permukaan usus, sehingga pendarahan dapat terjadi secara lebih berat dibandingkan dengan infeksi cacing jenis lainnya. Pendarahan yang lebih berat disebabkan karena mulut (*stoma*) cacing mengerat permukaan usus. Bahkan satu ekor cacing saja dapat menyebabkan kehilangan darah sebanyak 0,005-0,34 cc sehari. Mengingat itu semua, maka infeksi cacing tambang merupakan penyebab anemia yang paling sering ditemukan pada anak-anak, sehinggak dapat mempengaruhi daya tahan tubuhnya dan menurunkan prestasi belajarnya(Qorry ‘Aina Abata ;2013:191-192).

II.3.4. Cacing kremi

Cacing ini mirip kelapa parut, kecil-kecil dan berwarna putih. Awalnya, cacing ini akan bersarang diusus besar. Saat dewasa, cacing kremi betina akan pindah ke anus untuk bertelur. Telur-telur ini yang menimbulkan rasa gatal. Bila balita menggaruk anus yang gatal, telur akan pecah dan larva masuk ke dalam

dubur. Saat digaruk, telur-telur akan bersembunyi di jari dan kuku, sebagian lagi menempel di sprei, bantal atau pakaian. Lewat kontak langsung, telur cacing menular ke orang lain. Lalu siklus cacing dimulai lagi(Qorry ‘Aina Abata ;2013:192).

II.4. Teori Dempster-Shafer

Ada berbagai macam penalaran dengan model yang lengkap dan sangat konsisten, tetapi pada kenyataannya banyak permasalahan yang tidak dapat terselesaikan secara lengkap dan konsisten. Ketidak konsistenan yang tersebut adalah akibat adanya penambahan fakta baru. Penalaran yang seperti itu disebut dengan penalaran *non monotonis*. Untuk mengatasi ketidak konsistenan tersebut maka dapat menggunakan penalaran dengan teori *Dempster-Shafer*. *Dempster-Shafer* adalah suatu teori matematika untuk pembuktian berdasarkan *belief functions and plausible reasoning*(fungsi kepercayaan dan pemikiran yang masuk akal), yang digunakan untuk mengkombinasikan potongan informasi yang terpisah (bukti) untuk mengkalkulasi kemungkinan dari suatu peristiwa. Teori ini dikembangkan oleh Arthur P. Dempster dan Glenn Shafer. Secara umum teori *Dempster-Shafer* ditulis dalam suatu interval :

- [*Belief,Plausibility*]

Belief (*Bel*) adalah ukuran kekuatan evidence dalam mendukung suatu himpunan proposisi. Jika bernilai 0 maka mengindikasikan bahwa tidak ada evidence, dan jika bernilai 1 menunjukkan adanya kepastian. Dimana nilai bel yaitu (0-0.9).

- *Plausibility* (Pl) dinotasikan sebagai :

Pl(s) = 1 – Bel (-s)

Plausibility juga bernilai 0 sampai 1. Jika yakin akan -s, maka dapat dikatakan bahwa Bel(-s)=1, dan Pl(-s)=0. Pada teori *Dempster-Shafer* dikenal adanya *frame of discrement* yang dinotasikan dengan θ. Frame ini merupakan semesta pembicaraan dari sekumpulan hipotesis. Tujuannya adalah mengaitkan ukuran kepercayaan elemen-elemen θ. Tidak semua evidence secara langsung mendukung tiap-tiap elemen. Untuk itu perlu adanya probabilitas fungsi densitas (m). Nilai m tidak hanya mendefinisikan elemen-elemen θ saja, namun juga semua subsetnya. Sehingga jika θ berisi n elemen, maka subset θ adalah 2^n . Jumlah semua m dalam subset θ sama dengan 1. Apabila tidak ada informasi apapun untuk memilih hipotesis, maka nilai : m{θ} = 1,0.

Apabila diketahui X adalah subset dari θ, dengan m1 sebagai fungsi densitasnya, dan Y juga merupakan subset dari θ dengan m2 sebagai fungsi densitasnya, maka dapat dibentuk fungsi kombinasi m1 dan m2 sebagai m3, yaitu:

$$m3(Z) = \frac{\sum x \cap y = z \; m1(X).m2(Y)}{1-\sum x \cap y = \theta m1(X).m2(Y)}$$

II.5. Unified Modelling Language (UML)

II.5.1. Definisi Unified Modelling Language (UML)

Unified Modelling Language (UML) adalah “bahasa” yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan

sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

Dengan menggunakan UML, kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemograman apapun. Tetapi karena UML juga menggunakan dan operation dalam konssep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa-bahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C.

Seperti bahasa-bahasa lainnya, UML mengidentifikasikan notasi dan syntax/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML syntax mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (*Object-Oriented Software Design*), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*) (Yuni Sugiarti;2013:34).

Unified Modeling Language (UML) biasa digunakan untuk :

1. Menggambarkan batasan sistem dan fungsi-fungsi secara umum, dibuat dengan use case.
2. Menggambarkan kegiatan atau proses bisnis yang dilaksanakan secara umum, dibuat dengan interaction diagrams.

3. Menggambarkan representasi struktur static sebuah sistem dalam bentuk class diagrams.
4. Membuat model behavior “yang menggambarkan kebiasaan atau sifat sebuah sistem” dengan static transition diagrams.
5. Menyatakan arsitektur implementasi fisik menggunakan component and development diagrams.
6. menyampaikan atau memperluas fungsionally dengan stereotype.

UML merupakan salah satu alat bantu yang sangat handal dalam bidang pengembangan sistem berorientasi objek karena UML menyediakan bahasa pemodelan visual yang memungkinkan pengembang sistem membuat *blue print* atas visinya dalam bentuk yang baku. UML berfungsi sebagai jembatan dalam berkomunikasikan beberapa aspek dalam sistem melalui sejumlah elemen grafis yang bisa dikombinasikan menjadi diagram. UML mempunyai banyak diagram yang dapat mengakomodasi berbagai sudut pandang dari suatu perangkat lunak yang akan dibangun.(Yuni Sugiarti; 2013:36-37)

II.5.2. Use Case Diagram

Dalam membuat sebuah sistem, langkah awal yang perlu dilakukan adalah menentukan kebutuhan. Terdapat dua jenis kebutuhan, yaitu kebutuhan fungsional dan kebutuhan nonfungsional. Kebutuhan fungsional adalah kebutuhan pengguna dan stakeholder sehari-hari yang akan dimiliki oleh sistem, dimana kebutuhan ini akan digunakan oleh pengguna stakeholder. Sedangkan kebutuhan nonfungsional adalah kebutuhan yang memperhatikan hal-hal berikut yaitu performansi,

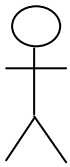
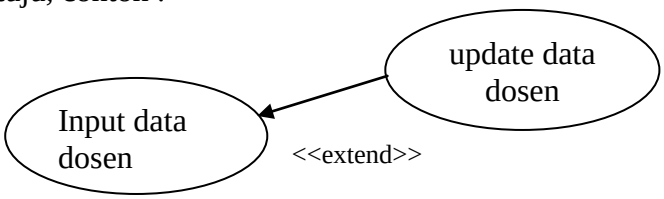
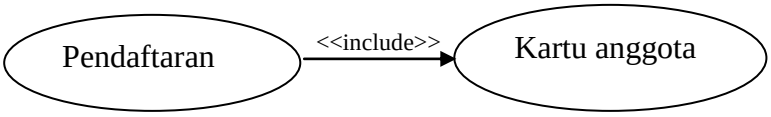
kemudahan dalam menggunakan sistem, kehandalan sistem, keamanan sistem, keuangan, legalitas, dan operasional.

Kebutuhan fungsi akan digambarkan melalui sebuah diagram yang dinamakan diagram use case. Use case diagram atau diagram use case merupakan pemodelan untuk menggambarkan kelakuan (behavior) sistem yang akan dibuat. Diagram use case mendeskripsikan sebuah interaksi antar satu atau lebih actor dengan sistem yang akan dibuat. Dengan pengertian yang cepat, diagram use case digunakan untuk mengetahui fungsi apa saja yang ada didalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Terdapat beberapa simbol dalam menggambarkan diagram use case, yaitu use case, actor dan relasi. Hal perlu diingat mengenai diagram use case adalah diagram use case bukan menggambarkan tampilan antarmuka (use interface), arsitektur dari sistem, kebutuhan nonfungsional, dan tujuan performansi. Sedangkan untuk penamaan use case adalah nama didefinisikan sesimpel mungkin, dapat dipahami dan menggunakan kata kerja (Yuni Sugiarti; 2013:41)

Berikut adalah simbol-simbol yang ada pada diagram use case :

Tabel II.1. Simbol – Simbol Use Case Diagram

Simbol	Deskripsi
Use Case 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit dan aktor; biasanya dinyatakan dengan menggunakan kata kerja di awalfrase nama use case

Aktor 	Orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor tentu merupakam orang; biasanya dinyatakan menggunakan kata benda diawal frase nama aktor.
Asosiasi / Assocition	Komunikasi antara actor dan use case yang berpatrisipasi pada usecase atau use case memiliki interaksi dengan aktor
Extend <<extend>>	Relasi use case tambahan kesebuah use case dimana use case yang ditambahkan dapat berdiri sendiri tanpa use case tambahan inti; mirip dengan prinsip inheritance pada pemograman berorientasi objek; biasanya use case tambahan memiliki nama depan yang sama dengan use case ditambahkan, arah panah menunjukan use case yang dituju, contoh : 
Include << include >>	Relasi use case tambahan kesebuah use case dimana use case yang ditambahkan memerlukan use case ini untuk menjalankan fungsinya atau sebagai syarat dijalankan use case ini. Ada dua sudut pandang yang cukup besar mengenai include di use case, include berarti use case yang ditambahkan akan selalu di panggil saat use case tambahan di jalankan. contoh : 

Sumber : (Yuni Sugiarti;2013:42)

II.5.3.Class Diagram (Diagram Kelas)

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. kelas memiliki apa yang disebut atribut dan metode atau operasi.

- Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas.
- Atribut mendeskripsikan property dengan sebaris teks didalam kotak kelas tersebut.
- Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

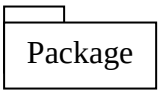
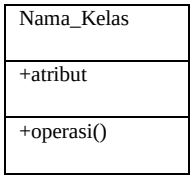
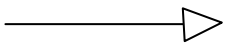
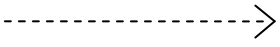
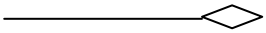
Diagram kelas mendeskripsikan jenis-jenis objek dalam sistem dan berbagai hubungan statis yang terdapat diantara mereka. Diagram kelas juga menunjukkan properti dan operasi sebuah kelas dan batasan-batasan yang terdapat dalam hubungan-hubungan objek. Diagram kelas menggambarkan struktur dan deskripsi class, package dan objek beserta hubungan satu sama lain seperti containment, perwarisan, asosiasi, dan lain-lain (Yuni Sugiarti;2013:57).

Kelas memiliki tiga area pokok :

1. Nama
2. Atribut
3. Operasi

Berikut adalah simbol-simbol yang ada pada diagram kelas :

Tabel II.2. Simbol-Simbol Diagram Kelas

Simbol	Deskripsi
Package 	Package merupakan bungkusan dari satu kelas atau lebih
Operasi 	Kelas pada struktur sistem
Antarmuka / *interface* 	Sama dengan konsep *interface* dalam pemrograman berorientasi objek
Asosiasi / *association*	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*
Asosiasi berarah / *Directed association* 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*
Generalisasi 	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum khusus)
Kebergantungan 	Relasi antar kelas dengan makna Kebergantungan antar kelas
Agregasi / *aggregation* 	Relasi antar kelas dengan makna Semua bagian (*whole part*)

Sumber : (Yuni Sugiarti;2013:59)

II.5.4. Activity Diagram (Aktivitas)

Diagram aktivitas atau *activity diagram* menggambarkan *workflow*(aliran kerja) atau aktivias dari sebuah sistem atau proses bisnis. *Activity diagram* menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, decision yang mungkin terjadi, dan bagaimana mereka berakhir.

Activity Diagran merupakan state diagram khusus, dimana sebagian besar state adalah action dan sebagian besar transisi di-trigger oleh selesainya state sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tdak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarka proses- proses dan jalur-jalur aktivitas dari level atas secara umum (Yuni Sugiarti;2013:75).

II.5.5. Squence Diagram

Diagram sekuen menggambarkan kelakuan/perilaku objek pada use case dengan mendeskripsikan waktu hidup objek dan message yang dikirimkan dan diterima antar objek.

Banyaknya diagram Sekuence yang digambarkan adalah sebanyak pendefenisian use case yang memiliki proses sendiri atau yang penting semua use case yang telah didefenisikan interaksi jalanya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak use case didefenisikan maka diagram sekuen yang harus dibuat juga semakin banyak (Yuni Sugiarti;2013:69).

II.6. Visual Basic 2010

Visual Basic merupakan salah satu bahasa pemrograman yang handal dalam lingkungan Windows. Visual Basic telah merajai pasar pembuatan *software* / perangkat lunak sampai beberapa dekade tanpa ada yang menyaingi. Visual Basic 2010 merupakan teknologi terbaru yang masuk ke dalam Visual Studio bersama dengan C#, C++, dan yang lainnya (Wahana Komputer ;2010).

Visual Basic adalah salah satu development tools untuk membangun aplikasi dalam lingkungan Windows. Visual Basic menggunakan pendekatan visual untuk merancang user interface dalam bentuk form, sedangkan untuk codingnya menggunakan dialek bahasa BASIC yang cenderung mudah dipelajari. Pada pemrograman visual, pengembangan aplikasi dimulai dengan pembentukan user interface, kemudian mengatur properties dari objek-objek yang digunakan dalam user interface, dan baru dilakukan penulisan kode program untuk menangani ke jadian-kejadian (event) (Wahana Komputer ; 2010).

Visual Basic 2010 merupakan salah satu bagian dari produk pemograman terbaru yang dikeluarkan oleh Microsoft, yaitu Microsoft Visual Studio 2010. Sebagai produk lingkunan pengembangan terintegrasi atau IDE andalan yang dikeluarkan oleh Microsoft (Wahana Komputer ;2010:2).

II.7. SQL Server 2008

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

mengolah data. Dalam versi terbaru ini, SQL Server 2008 memiliki banyak fitur yang bisa dihandalkan untuk meningkatkan performa database.

SQL Server 2008 memiliki suatu GUI (*Graphic User Interface*) yang dapat digunakan untuk melakukan aktivitas sehari-hari berkaitan dengan database, seperti menulis T-SQL, melakukan backup dan restore database, melakukan security database terhadap aplikasi dan sebagainya. Pada GUI tersebut terdapat setingan terhadap SQL Server untuk bekerja lebih optimal. Settingannya juga bisa dilakukan dengan menggunakan script untuk memudahkan developer mengubah setting pada SQL Server 2008.

II.8. Basis Data

Basis data adalah sekumpulan fakta berupa representasi tabel yang saling berhubungan dan disimpan dalam media penyimpanan secara digital. Suatu basis data terdiri dari sekumpulan tabel yang saling berelasi ataupun tidak berelasi. semua tabel tersebut merupakan representasi tempat untuk penyimpanan data yang mendukung fungsi dari basis data tersebut pada suatu sistem (Yudi Priadi ; 2014:1-2).

II.8.1. Basis Data Relational

Basis data *relational* diciptakan oleh seorang peneliti dari IBM yang bernama Dr. E. F. Codd, kemudian dikembangkan terus oleh peneliti lainnya hingga saat ini. Prinsipnya, model basis data *relational* digunakan sebagai suatu cara untuk mengelompokan data dari sebuah kumpulan data yang besar. hal ini dapat dilakukan dengan cara menghapus duplikasi dari suatu data melalui proses

yang disebut normalisasi. Proses ini terdiri dari beberapa langkah yang akan menjadi suatu bentuk normal. Hasilnya berupa bahasa umum untuk melakukan akses terhadap suatu basis data, yang disebut dengan *Strucutured Query Language* (SQL), sehingga dimungkinkan untuk melakukan *query* terhadp organisasi struktur datanya. SQL sebagai bahasa pada basis data telah menjadi standar untuk semua perusahaaan pengembang basis data hingga saat ini (Yudi Priadi ; 2014: 14).

II.9. Normalisasi Data

Normalisasi merupakan tahapan proses sistematis yang dilakukan pada struktur tabel basis data menjadi struktur tabel yang memiliki integritas data, sehingga tidak memiliki data anomali pada saat melakukan insert, delete, dan *update.* Kegiatan normalisasi adalah melakukan dekomposisi atau penguraian tabel beserta datanya, menjadi tabel yang normal menurut konsep RDBMS. Dekomposisi diawali dengan melakukan analisis pada suatu tabel atau beberapa contoh formulir yang sudah memiliki data lengkap utuk suatu basis data, tetapi masih dalam bentuk yang tidak normal (UNF). Oleh karena itu, agar dapat memenuhi syarat bentuk normal pertama (1NF), pada setiap barisnya diisikan suatu *value* dengan kelompok data yang sama, berdasarkan suatu atribut key. Dengan demikian, kelompok pengulangan dalam suatu baris dapat dihilangkan, karena sudah tidak terdapat *value* yang kosong untuk setiap *field* dan *record*-nya (Yudi Priadi ; 2014: 67- 68).

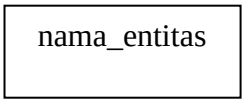
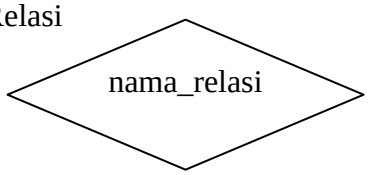
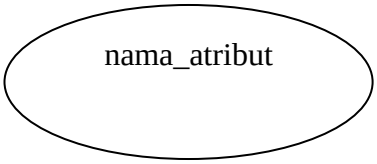
Setelah memenuhi syarat bentuk normal pertama (1NF), proses berikutnya adalah menghilangkan ketergantungan secara persial, yaitu dengan cara melakukan dekomposisi tabel menjadi beberapa kelompok berdasarkan *field* yang memiliki status sebagai *key*. Hal in dapat dilakukan oleh salah satu *field* saja, dengan tetap tidak mengubah arti relasi dan ketergantungannya. oleh sebab itu, disebut ketergantungan fungsional sebagian (*partially functional*), sehingga syarat bentuk normal kedeu (2NF) sudah tercapai (Yudi Priadi ; 2014:68).

Bentuk normal kedua (2NF) merupakan syarat yang harus dimiliki untuk menuju bentuk normal ketiga (3NF). Pada proses ini , dilakukan dengan menghilangkan ketergantungan secara transistif, yaitu konsep untuk tabel dari hasil relasi yang didalamnya terdapat ketergantungan secara tidak langsung pada beberapa atributnya. Pada umunya, proses normalisai suda dapat tercapai pada bentuk normal ketiga (3NF), yaitu dengan menghasilkan tabel yang tidak mengalami anomali basis data pada saat proses *insert*, *delete*, dan *update* (Yudi Priadi ; 2014:68).

II.10. ERD (Entity Relationship Diagram)

Pemodelan basis data dengan menggunakan diagram relasi antar entitas dapat dilakukan dengan menggunakan suatu pemodelan basis data yang bernama *Diagram Entity-Relationship*. Adapun simbol/notasi dasar yang digunakan pada diagram E-R yaitu :

Tabel II.3. Simbol-Simbol ERD

Simbol	Deskripsi
Entitas/*entity* 	Entitas merupakan notasi untuk mewakili suatu objek dengan karakteristik sama, yang dilengkapi oleh atribut, sehingga pada suatu lingkungan nyata setiap objek akan berbeda dengan objek yang lainnya. pada umumnya, objek dapat berupa benda,pekerjaan,tempat, dan orang.
Relasi 	Relasi merupakan notasi yang digunakan untuk menghubungkan beberapa entitas berdasarkan fakta pada suatu lingkungan.
Atribut 	Atribut merupakan notasi yang menjelaskan karakteristik suatu entitas dan juga relasinya. Atribut dapat sebagai key yang bersifat unik, yaitu primary key atau foreign key. atribut sebagai *key* pembeda dengan menggunakan garis bawah.
Garis penghubung ———————	Garis penghubung merupakan notasi untuk merangkaikan keterkaitan antara notasi-notasi yang digunakan dalam Diagram E-R yaitu entitas, relasi, dan atribut.

Sumber: (Yudi Priyadi, M.T ;2014:20-21)

Ada beberapa relasi yang dapat terjadi, yaitu :

- Satu ke satu (1:1), mempunyai arti bahwa setiap satu entitas pada himpunan entitas A hanya dapat berelasi dengan satu saja pada himpunan entitas B.
- Satu ke banyak (1:N), mempunyai arti bahwa setiap satu entitas pada himpunan entitas A dapat berelasi dengan banyak entitas pada himpunan entitas B.

- Banyak ke satu (N:1), mempunyai arti bahwa beberapa entitas pada himpunan entitas A hanya dapat berelasi dengan satu entitas saja pada himpunan entitas B.
- Banyak ke banyak (N:N), mempunyai arti bahwa beberapa entitas pada himpunan entitas A dapat berelasi dengan banyak entitas pada himpunan entitas B.