BAB II
LANDASAN TEORI

2.1. Teori Dasar

2.1.1 Kecerdasan Buatan (*Artificial Intelligence*)

Kecerdasan buatan berasal dari kata *Artificial Intelligence* yang mengandung arti tiruan atau kecerdasan. Secara harfiah *Artificial Intelligence* adalah kecerdasan buatan. Kecerdasan buatan adalah salah satu bidang dalam ilmu komputer yang membuat komputer agar dapat bertindak seperti manusia (menirukan kerja otak manusia).(Supartini & Hindarto, 2016)

2.1.1.1. Sistem Pakar (*Expert System)*

Sistem pakar merupakan cabang dari AI (*Artificial Inteligent*) yang membuat ekstensi khusus untuk spesialisasi pengetahuan guna memecahkan suatu permasalahan pada *Human Expert*. *Human Expert* merupakan seseorang yang ahli dalam suatu bidang ilmu pengetahuan tertentu, ini berarti bahwa expert memiliki suatu pengetahuan atau *skill* khusus yang dimiliki oleh orang lain. *Expert* dapat memecahkan suatu permasalahan yang tidak dapat dipecahkan oleh orang lain dengan cara efisien.

Pengetahuan di dalam *Expert system* berasal dari orang atau *knowledge* yang berasal dari buku-buku referensi, surat kabar atau karya ilmiah orang lain,

pengetahuan manusia ke dalam komputer, agar komputer dapat menyelesaikan masalah seperti yang biasa dilakukan oleh para ahli. Atau dengan kata lain sistem pakar adalah sistem yang didesain dan diimplementasikan dengan bantuan bahasa pemrograman tertentu untuk dapat menyelesaikan masalah seperti yang dilakukan oleh para pakar dalam hal ini adalah dokter.

Proses inferensi dilakukan dalam suatu modul yang disebut *inference engine* (mesin inferensi). Ketika representasi pengetahuan pada bagian *knowledge base* telah lengkap, atau paling tidak telah berada pada *level* cukup akurat, maka referensi pengetahuan tersebut telah siap digunakan. Sedangkan inferensi *engine* merupakan modul yang berisi program tentang bagaimana mengendalikan proses *reasoning*. Terdapat dua metode umum penalaran yang dapat digunakan apabila pengetahuan dipresentasikan untuk mengikuti aturan-aturan sistem pakar yaitu metode *forward chaining* dan metode *Backward Chaining* .(Supartha & Sari, 2014)

Sistem pakar adalah program komputer yang merupakan cabang dari penelitian dari ilmu komputer yaitu kecerdasan buatan. Sistem pakar berusaha mengadopsi pengetahuan manusia ke komputer, agar komputer dapat menyelesaikan masalah seperti biasa yang dilakukan oleh para ahli. Adapun komponen sistem pakar meliputi: (Syah & Ananta, 2015)

a. Antar muka pengguna, mekanisme yang digunakan oleh pengguna dan sistem pakar untuk berkomunikasi. Antar muka menerima informasi dari pengguna, lalu menampilkan keluarkan sebagai respon dari sistem pakar.

b. Basis pengetahuan, pengetahuan untuk pemahaman, formulasi, dan penyelesaian masalah. Komponen sistem pakar disusun atas dua elemen dasar, yaitu fakta dan

aturan. Fakta merupakan informasi tentang objek permasalahan, sedangkan aturan merupakan informasi tentang cara memperoleh fakta baru dari fakta yang telah diketahui.

c. Akuisisi pengetahuan, proses akumulasi, transfer dan transformasi keahlian dalam menyelesaikan masalah dari sumber pengetahuan ke dalam program komputer. Pengetahuan diperoleh dari pakar, dilengkapi dengan buku, laporan penelitian dan pengalaman pengguna.

d. Mesin inferensi otak dari sebuah sistem pakar dalam sistem berbasis kaidah. Komponen ini mengandung mekanisme pola pikir dan penalaran yang digunakan oleh pakar untuk menyelesaikan suatu masalah. Mesin inferensi adalah model yang memberikan metodologi untuk penalaran dalam memformulasikan kesimpulan.(Syah & Ananta, 2015)

Beberapa metode yang sering terlihat dari mesin inferensi, yaitu:

1. Forward Chaining

Forward chaining merupakan fakta untuk mendapatkan kesimpulan *(conclusion)* dari fakta tersebut. Penalaran ini berdasarkan fakta yang ada *(data driven),* metode ini adalah kebalikan dari metode *backward chaining*, dimana metode ini dijalankan dengan mengumpulkan fakta-fakta yang ada untuk menarik kesimpulan. Dengan kata lain, prosesnya dimulai dari *facts* (fakta-fakta yang ada) melalui proses *inference fact* (penalaran fakta-fakta) menuju suatu *goal* (suatu tujuan). Metode ini bisa juga disebut menggunakan aturan IF-THEN dimana *premise* (IF) menuju *conclusion* (THEN). (Dewi, Lestari, & Lestari, 2015)

2. Backward Chaining

Penalaran berdasarkan tujuan *(goal-driven)*, metode ini dimulai dengan membuat perkiraan dari apa yang akan terjadi, kemudian mencari fakta-fakta *(evidence)* yang mendukung (atau membantah) hipotesa tersebut. *Backward chaining* adalah suatu alasan yang berkebalikan dengan *hypothesis*, potensial konklusinya mungkin akan terjadi atau terbukti, karena adanya fakta yang mendukung akan *hypothesis* tersebut . Dengan kata lain, prosesnya dimulai dari initial *Hyphotesis or goal* (Hipotesa awal atau tujuan) melalui Intermediet *Hipotheses or sub goals* (hipotesa lanjutan atau bagian dari tujuan) yang akan memeriksa semua hipotesa yang ada apakah hipotesa itu benar atau salah sehingga akhirnya akan menuju suatu *Evidence* (fakta).

Sebagai contoh akan diuraikan sebagai berikut, jika suatu masalah mempunyai sederetan kaidah seperti tertulis dibawah ini:

R1 : A and C, THEN E

R2 : IF D and C, THEN F

R3 : IF B and E, Then F

R4: IF B THEN C

R5 : IF F THEN G

Dimana sebagai acuan diketahui bahwa fakta A dan B adalah *true* (benar) dan G adalah *GOAL* (tujuan). Berikut ini langkah-langkah yang digunakan dalam metode *backward chaining*: (Nur, Ikhsan, Ariadi, Rosyid, & Ridwan, 2017)

1. Langkah 1 : Mencari kebenaran dasar dari tujuan berdasarkan fakta yang ada, dimana sebagai acuannya kita sudah mengetahuinya.

2. Langkah 2 : R5 menunjukkan bahwa jika F benar maka G benar. Untuk itu, maka kita akan melihat R2 dan R3.

3. Langkah 3 : R2 menunjukkan bahwa D belum tentu benar sebab D tidak termasuk dalam fakta acuan, sehingga R2 tidak bisa digunakan, maka kita akan melihat ke kaidah yang lainnya yaitu kaidah R3.

4. Langkah 4 : Pada kaidah R3, kita ketahui sesuai fakta acuan yang ada bahwa B adalah benar, selanjutnya kita akan melihat apakah E benar.

5. Langkah 5 : Pada kaidah R1 sangat tergantung dengan kebenaran A dan C

6. Langkah 6 : Karena A diketahui sebagai fakta acuan adalah benar, selanjutnya kita akan melihat apakah C benar, dengan melihat R4.

7. Langkah 7: R4 menunjukkan bahwa C adalah benar karena B adalah benar Dari langkah diatas dapat diambil kesimpulan bahwa G adalah benar.

2.1.1.2. Jaringan Sistem Syaraf

Jaringan Saraf Tiruan (JST) merupakan salah satu representasi buatan dari otak manusia yang selalu mencoba untuk mensimulasikan proses pembelajaran otak manusia tersebut. Untuk JST tercipta sebagai suatu generalisasi model matematika dari pemahaman manusia *(human cognition)* yang didasarkan atas asumsi pemrosesan informasi terjadi pada elemen sederhana yang disebut neuron. Isyarat mengalir diantara sel saraf melalui suatu sambungan penghubung, setiap sambungan penghubung memiliki bobot yang bersesuaian dan setiap sel saraf akan merupakan fungsi aktivasi terhadap isyarat hasil penjumlahan berbobot yang masuk

kepadanya untuk menentukan isyarat keluarannya. (Lesnussa, Latuconsina, & Persulessy, 2015)

Jaringan Syaraf Tiruan merupakan pemodelan data yang kuat yang mampu menangkap dan mewakili hubungan *Input-Output* yang komplek, karena kemampuannya untuk memecahkan beberapa masalah relatif mudah digunakan, ketahnan untuk mengimput data kecepatan untuk eksekusi, dan menginisialisasikan sistem yang rumit. (Norhamreeza Abdul Hamid, 2011)

Jaringan Syaraf Tiruan (JST) merupakan suatu sistem pemrosesan informasi yang mempunyai karakteristik menyerupai jaringan syaraf biologis (JSB). Jaringan Syaraf Tiruan tercipta sebagai suatu generalisasi model matematis dari pemahaman manusia *(human cognition).* (Maharani Dessy Wuryandari, 2012)

2.1.1.3. Logika *Fuzzy*

Logika *fuzzy* pertama di kenalkan oleh Prof. Lotfi A. Zadeh pada tahun 1965. Logika *fuzzy* merupakan suatu metode pengambilan keputusan berbasis aturan yang digunakan untuk memecahkan keabu-abuan masalah pada sistem yang sulit dimodelkan atau memiliki ambiguitas. Dasar logika *fuzzy* adalah teori himpunan *fuzzy*. (Kinanti, Yamin, & Aksara, 2016)

Menurut (Kusumadewi & Hari , 2004), logika *fuzzy* adalah suatu cara yang tepat untuk memetakan suatu ruang input kedalam suatu ruang *output*, mempunyai nilai kontinyu. *Fuzzy* dinyatakan dalam derajat dari suatu keanggotaan dan derajat dari kebenaran. Oleh sebab itu sesuatu dapat dikatakan sebagian benar dan sebagian salah pada waktu yang sama. Sistem logika *fuzzy* terdiri dari himpunan *fuzzy* dan

aturan *fuzzy*. *Subset fuzzy* merupakan himpunan bagian yang berbeda dari variabel *input* dan *output*.

Sebelum munculnya logika *fuzzy*, dikenal sebuah logika tegas *(Crisp Logic)* yang memiliki nilai benar atau salah secara tegas. Sebaliknya Logika *Fuzzy* merupakan sebuah logika yang memiliki nilai kekaburan atau kesamaran *(fuzzyness)* antara benar dan salah. Dalam teori logika *fuzzy* sebuah nilai bisa bernilai benar dan salah secara bersamaan namun berapa besar kebenaran dan kesalahan suatu nilai tergantung kepada bobot keanggotaan yang dimilikinya. Logika *fuzzy* adalah suatu cara yang tepat untuk memetakan ruang *input* ke dalam suatu ruang *output*. (Kusumadewi, 2003)

Ada 4 parameter yang perlu diketahui dalam memahami sistem *fuzzy*, yaitu: (Syamsul, 2017)

1. Variabel Fuzzy Variabel fuzzy merupakan variabel yang hendak dibahas dalam suatu sistem fuzzy. Contoh: Kualitas Air, Debit Air, Harga Air.

2. Himpunan Fuzzy Himpunan fuzzy merupakan suatu grup yang mewakili suatu kondisi atau keadaan tertentu dalam suatu variabel fuzzy. Contoh: Variabel golongan pelanggan , terbagi menjadi 3 himpunan, yaitu: sosial, rumah tangga dan bisnis Himpunan fuzzy memiliki 2 atribut, yaitu:

1. Linguistik, yaitu penamaan grup yang mewakili suatu keadaan atau kondisi tertentu dengan menggunakan bahasaalami, seperti: Bersih, Standar dan keruh

2. Numeric, yaitu suatu nilai (angka) yang menunjukkan ukuran dari suatu variabel seperti: 5, 20, 25, 40, dan sebagainya.

2.1.2 *Web*

World Wide Web (*WWW*) adalah aplikasi yang digunakan dalam internet yang berfungsi sebagai transportasi data yang diterima sebagai start untuk menyimpan, menerima dan *formatting* dan menampilkan informasi melalui *client-server architecture*. *Web* dibagi menjadi 2 yaitu *web* statis dan *web* dinamis. (Pratama, Jusak, & Sudarmaningtyas, 2013)

1. Web statis

Web statis adalah *web* yang *content*-nya dikirimkan ke *user* sama dengan yang disimpan di *server*. Pada *web* ini sama sekali tidak ada perubahan, berbanding terbalik dengan *web* dinamis yang dihasilkan dari aplikasi *web server*.

2. *Web* dinamis

Web dinamis adalah *web* yang *content*-nya dihasilkan dari hasil *output* dari *web server*. Tidak seperti *web* statis yang contentnya tidak dapat berubah-ubah, *web* dinamis dapat berubah-ubah sesuai dengan informasi terakhir yang ada di *server*. *Web* dinamis dibagi menjadi dua yaitu :

a. *Server side*

Web dinamis dengan metode *server side* berjalan dengan kode program berjalan di *server*. Contoh : PHP, ASP, JSP, dan lain-lain. *Server side* memiliki kelebihan yaitu kode program yang tidak diketahui oleh pengguna. Sedangkan kelemahannya adalah kinerja *server* yang berat.

b. *Client Side*

Web dinamis dengan metode *client side* berjalan dengan kode program berjalan di *client*. Contoh : Javascript. *Client side* memiliki kelebihan yaitu kode

program dieksekusi di komputer pengguna sehingga mengurangi beban kerja *server*. Sedangkan kelemahannya adalah kode program dapat dibaca oleh pengguna.

2.1.3 Basis Data (*Database*)

Basis data terdiri atas semua fakta yang diperlukan, dimana fakta-fakta tersebut digunakan untuk memenuhi kondisi dari kaidah-kaidah dalam sistem. Basis data menyimpan semua fakta, baik fakta awal pada saat sistem mulai beroperasi, maupun fakta- fakta yang diperoleh pada saat proses penarikan kesimpulan sedang dilaksanakan. Basis data digunakan untuk menyimpan data hasil observasi dan data lain yang dibutuhkan selama pemrosesan. (Dewi et al., 2015)

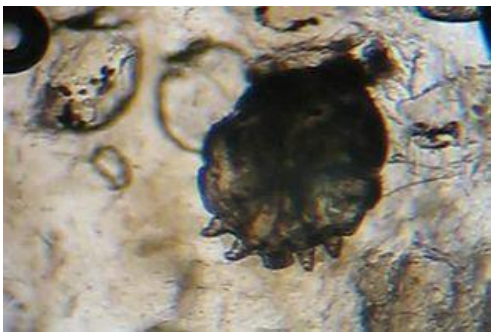
Basis data tersimpan di perangkat keras, serta dimanipulasi dengan menggunakan perangkat lunak. Pendefinisian basis data meliputi spesifikasi dari tipe data, struktur dan batasan dari data atau informasi yang akan disimpan. Istilah-istilah dalam basis data: (Mutammimul Ula, 2014)

a. *Enterprise*: suatu bentuk organisasi seperti Bank, Sekolah, Rumah Sakit, Pabrik, Kantor dan sebagainya.
b. Entitas: suatu objek yang dapat di bedakan dari lainnya yang dapat di wujudkan dalam basis data. Kumpulan dari entitas disebut himpunan entitas.
c. Atribut dan elemen data: karakteristik dari suatu entitas.
d. *Record* data: kumpulan suatu elemen data yang saling berhubungan.
e. Tabel: kumpulan data atau informasi.

2.2. Variabel Penelitian

Pada penelitian ini yang menjadi variabel adalah penyakit pada anjing jenis Alaskan Malamute yang dapat didiagnosa dari fisik yaitu :

1. Scabiosis



Gambar 2. 1 *Scabiosis*

Penyakit *Scabiosis* disebabkan oleh tungau atau kutu golongan *Sarcoptes Scabiei Canis* yang merupakan parasit yang sangat kecil, sulit dilihat dengan mata telanjang. Kutu atau tungau ini berkembang dengan bertelur di dalam poripori kulit atau dengan membuat terowongan di dalam kulit.

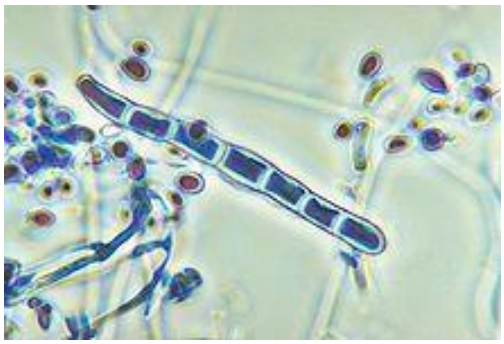
2. *Demodexcosis*

Gambar 2. 2 *Demodexcosis*

Penyakit *Demodexcosis* disebabkan oleh *Demodectic Mange* (Tungau *Demodex folliculorum*) atau disebut *Demodex* yang hanya dapat dilihat melalui

mikroskop saja karena sangat kecil. Parasit ini menyerang sampai dibawah kulit dan terutama di akar rambut.

3. *RingWorm*



Gambar 2. 3 *Ringworm*

Ringworm merupakan penyakit yang disebabkan oleh infeksi jamur. Biasanya daerah luka biasanya berada di sekitar lipatan leher, lipatan mata, lipatan paha, ekor dan daerah kuku. Daerah luka biasanya berbentuk seperti cincin melingkar. Penyakit ini bersifat *zoonosis* bisa menular ke manusia.

2.3. *Software* Pendukung

2.3.1 *XAMPP*

Menurut Putra Yoka (2015: 25), *server web* merupakan komputer yang berfungsi menyimpan dokumen berkaitan *web* yang melayani permintaan dokumen *web* dari *client*-nya. *XAMPP* merupakan salah satu perangkat lunak dengan *web server apache* yang berarti sudah tersedia *database server MySQL* didalamnya yang dapat mendukung proses pemrograman *PHP*. *XAMPP* sangat mudah didapatkan

dan digunakan baik *Windows* maupun *Linux*. *XAMPP* merupakan *software* yang mudah digunakan , gratis, dan mendukung instalasi di *Linux* dan *Windows*.

2.3.2 *PHP*

PHP adalah bahasa pemrograman yang digunakan secara luas untuk penanganan pembuatan dan pengembangan sebuah *web* dan bias digunakan pada *HTML*. *PHP* merupakan singkatan dari "*PHP : Hypertext Preprocessor*", dan merupakan bahasa yang disertakan dalam dokumen HTML, sekaligus bekerja di sisi *server* (*server-side HTML-embedded scripting*). Artinya sintaks dan perintah yang diberikan akan sepenuhnya dijalankan di *server* tetapi disertakan pada halaman *HTML* biasa, sehingga script-nya tak tampak disisi *client*. *PHP* dirancangan untuk dapat bekerja sama dengan *database server* dan dibuat sedemikian rupa sehingga pembuatan dokumen *HTML* yang dapat mengakses *database* menjadi begitu mudah. Tujuan dari bahasa scripting ini adalah untuk membuat aplikasi di mana aplikasi tersebut yang dibangun oleh *PHP* pada umumnya akan memberikan hasil pada *web* browser, tetapi prosesnya secara keseluruhan dijalankan di *server*. (Palit, Rindengan, & Lumenta, 2015)

2.3.3 *MySQL*

MySQL merupakan salah satu *database* kelas dunia yang sangat cocok bila dipadukan dengan bahasa pemrograman *PHP*. *MySQL* bekerja dengan bahasa

Structure Query Language yang berupa standar yang digunakan dalam memanipulasi *database*.

2.3.4 *Notepad++*

Notepad++ adalah sebuah aplikasi penyunting teks dan penyunting kode sumber yang berjalan di sistem operasi *Windows*. *Notepad++* menggunakan komponen *Scintilla* untuk dapat menampilkan dan menyuntingan teks dan berkas kode sumber berbagai bahasa pemrograman.

2.3.5 *UML (Unified Modeling Language)*

"*The UML defines a diagrammatic notation for describing the artefacts of an OOAD. Through the UML we can visualize, specify, construct and document our software application*". (Rahmawati & Mulyono, 2016)

UML adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement*, membantu analisis & desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek. Banyak orang telah membuat bahasa pemodelan pembangunan perangkat lunak sesuai dengan teknologi pemrograman yang berkembang pada saat itu, misalnya yang sempat berkembang dan digunakan banyak pihak adalah *Data Flow Diagram* (DFD) untuk memodelkan perangkat lunak yang menggunakan pemrograman *procedural* atau

structural, kemudian juga ada *State Transition Diagram* (STD) yang digunakan untuk memodelkan sistem *real time* (waktu nyata).

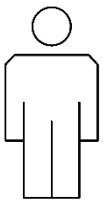
Pada perkembangan teknik pemrograman berorientasi objek, muncullah sebuah standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakan teknik pemrograman berorintasi objek, yaitu *Unified Modeling Language (UML)*. *UML* muncul karena adanya kebutuhan model visual untuk menspesifikasikan, menggambarkan, membangun dan dokumentasi dari sistem perangkat lunak. UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung.(Rosa A.S; M.Shalahuddin, 2011 :113).

2.3.5.1. *Use Case Diagram*

Use case atau diagram *use case* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut *actor* dan *use case*.(Rosa A.S; M.Shalahuddin, 2011)

1. *Actor* merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat itu sendiri, jadi walaupun *symbol* dari *actor* adalah gambar orang, tapi *actor* belum tentu merupakan orang.

2. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau *actor*.

Simbol	**Deskripsi**
Use Case	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal di awal frase nama *use case*
Aktor / *actor* 	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang, biasanya dinyatakan menggunakan kata benda di awal frase nama aktor.
Asosiasi / a*ssociation* _____	Komunikasi antara aktor dan *use case* yang berpatisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor.

Ekstensi / *extend* – – – –>	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu, mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek, biasanya *use case* tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan.
Generalisasi / *generalization* ——▷	Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.
Menggunakan / *include* – – – –>	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini.

Tabel 2. 1 *Simbol Use Case*

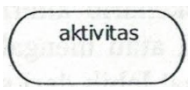
Sumber: (Rosa A.S; M.Shalahuddin, 2011)

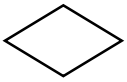
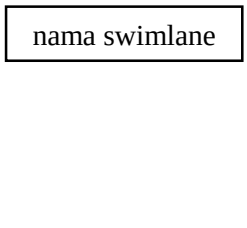
2.3.5.2. *Activity Diagram*

Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan *actor*, jadi aktivitas yang dapat dilakukan oleh sistem. Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut: (Rosa A.S; M.Shalahuddin, 2011)

1. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
2. Urutan atau pengelompokan tampilan dari sistem/*user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
3. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.
4. Rancangan menu yang ditampilkan pada perangkat lunak.

Berikut adalah simbol-simbol yang ada pada diagram aktivitas:

Simbol	Deskripsi
Status awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal
Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja

Percabangan / *decision* 	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu
Penggabungan / *join* 	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu
Status akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir
Swimlane 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi

Tabel 2. 2 *Simbol Activity Diagram*

Sumber: (Rosa A.S; M.Shalahuddin, 2011)

2.3.5.3. *Class Diagram*

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.

1. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas
2. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

Kelas-kelas yang ada pada struktur sistem harus dapat melakukan fungsi-fungsi sesuai dengan kebutuhan sistem sehingga pembuat perangkat lunak atau

programmer dapat membuat kelas-kelas di dalam program perangkat lunak sesuai dengan perancangan diagram kelas. Susunan struktur kelas yang baik pada diagram kelas sebaiknya memiliki jenis-jenis kelas berikut: (Rosa A.S; M.Shalahuddin, 2011)

1. Kelas main

   Kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.

2. Kelas yang menangani tampilan sistem (*View*)

   Kelas yang mendefinisikan dan mengatur tampilan ke pemakai.

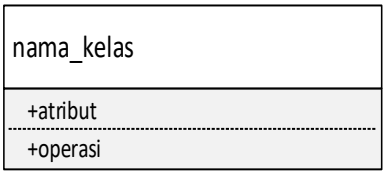
3. Kelas yang diambil dari pendefinisian *use case* (*controller*)

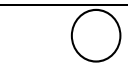
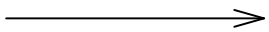
   Kelas ini biasanya disebut dengan kelas proses yang menangani proses bisnis pada perangkat lunak.

4. Kelas yang diambil dari pendefinisian data (*model*)

   Kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data.

Berikut adalah simbol-simbol yang ada pada diagram kelas:

Simbol	Deskripsi
Kelas 	Kelas pada struktur sistem
Antarmuka / *interface*	Sama dengan konsep interface dalam pemrograman berorientasi objek

	
Asosiasi berarah / *direct association* 	Relasi antarkelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*
Generalisasi 	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum-khusus)
Kebergantungan / *dependency* 	Relasi antar kelas dengan makna kebergantungan kelas
Agresi / aggregation 	Relasi antar kelas dengan makna semua-bagian (*whole-part*)

Tabel 2. 3 *Simbol Class Diagram*

Sumber: (Rosa A.S; M.Shalahuddin, 2011)

2.4. Penelitian Terdahulu

1. Nama Jurnal : Jurnal Informatika Polinema

   Judul Jurnal : Pembuatan Sistem Pakar Diagnosa Penyakit Pada Burung Puyuh Dengan Menggunakan Metode Forward Chaining

   Nama Peneliti : Alfian Karunyan Syah, Ahmadi Yuli Ananta

   Volume/Tahun/ISSN : 2/2015/2407-070X

Kesimpulan : Metode Forward Chaining dapat mendiagnosa penyakit pada burung puyuh berserta solusi penanganannya dengan ketepatan 85% dan presisi 100%.

2. Nama Jurnal : Jurnal Sarjana Teknik Informatika

Judul Jurnal : Sistem Pakar Mendiagnosa Penyakit Pada Ikan Konsumsi Air Tawar Berbasis *Web*site

Nama Peneliti : Elfani , Ardi Pujiyanta

Volume/Tahun/ISSN : 1/2013/2338-5197

Kesimpulan : Menghasilkan sebuah perangkat lunak untuk mengidentifikasi tentang penyakit pada ikan konsumsi air tawar berbasis *web*site yang didukung dengan Theorema Bayes yang hanya dengan memasukkan gejala serta memberikan solusi seperti layaknya seorang pakar.

3 Nama Jurnal : JSIKA

Judul Jurnal : Sistem Pakar Identifikasi Penyakit Kulit Anjing Menggunakan Metode Certainty Factor

Nama Peneliti : Arnaz Malikul Hakim, Jusak, Erwin Sutomo

Volume/Tahun/ISSN : 4/2015/2338-137X

Kesimpulan : Sistem Pakar ini mememiliki ketepatan sebesar 91,67% dimana 11 data dari 12 data mampu memberikan identifikasi penyakit kulit pada anjing berdasarkan gejala serta memberikan saran tindakan awal dari hasil sistem pakar.

4 Nama Jurnal : Jurnal Sarjana Teknik Informatika

Judul Jurnal : Sistem Pakar Untuk Mengidentifikasi Penyakit Udang Galah Dengan Metode Theorema Bayes

Nama Peneliti : Muhammad Johan Wahyudi, Abdul Fadlil

Volume/Tahun/ISSN : 1/2013/2338-5197

Kesimpulan : Menghasilkan sebuah perangkat lunak tentang sistem pakar berbasis dekstop untuk mendiagnosa penyakit udang galah dengan perpaduan metode Forward Chaining sebagai penarik kesimpulan dengan Theorema Bayes sebagai alat kepastian. Mampu mengidentifikasi penyakit udang galah berdasarkan gejala yang dimasukkan serta memberikan solusi seperti layaknya seorang pakar.

5 Nama Jurnal : Jurnal Ilmiah Komputer dan Informatika

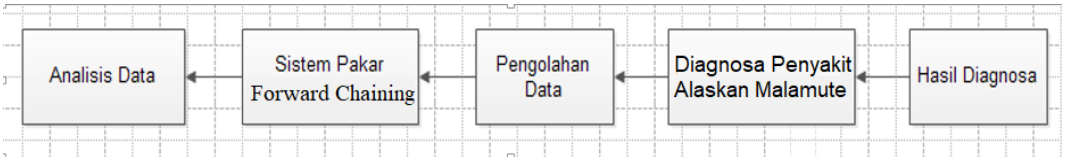
Judul Jurnal : Sistem Pakar Diagnosis Penyakit Ikan Koi Dengan Metode Bayes

Nama Peneliti : Puput Shinta Dewi, Ryana Dwi Lestari, Ryani Tri Lestari

Volume/Tahun/ISSN : 4/2015/2089-9033

Kesimpulan : Sistem pakar ini mampu mendiagnosa penyakit dari pertanyaan-pertanyaan yang ditanyakan oleh sistem setelah itu mendiagnosa penyakit dan cara perawatan, berdasarkan gejala-gejala yang diinputkan user dan membantu user dalam pemeliharaan ikan Koi.

2.5 Kerangka Pemikiran



Gambar 2. 4 Kerangka Pemikiran

(**Sumber :** Data Penelitian, 2018)

a. Analisis Data

   Menganalisis data yang telah didapatkan dari pakar dan memasukan kedalam sistem yang telah dibangun.

b. Sistem Pakar Forward Chaining

   User memasukan data untuk dianalisis oleh sistem pakar yang telah dibangun.

c. Pengolahan Data

   Pengolahan data yang sudah didapatkan dari user dengan metode Forward Chaining.

d. Diagnosa Penyakit Alaskan Malamute

   Melalui tahapan tahapan yang sudah dilalui akan menghasilkan diagnosa penyakit.

e. Hasil Diagnosa

   Hasil Diagnosa berupa diagnosa penyakit dan saran solusi atas diagnose penyakit.