BAB II
LANDASAN TEORI

2.1. Teori Dasar

Landasan teori perlu ditegakkan agar penelitian mempunyai dasar yang kokoh, dan bukan sekedar perbuatan coba-coba (*trial and error*). Adanya landasan teori merupakan ciri bahwa penelitian merupakan cara ilmiah untuk mendapatkan data. Kerlinger mengatakan bahwa teori adalah seperangkat konstruk (konsep), definisi, dan proposisi yang berfungsi untuk melihat fenomena secara sistematis melalui spesifikasi hubungan antarvariabel sehingga dapat berguna untuk menjelaskan dan meramalkan fenomena. Dengan kata lain, teori adalah generalisasi atau kumpulan generalisasi yang dapat digunakan untuk menjelaskan berbagai fenomena secara sistematis. (Sudaryono, 2015:13)

Teori dasar ini bertujuan untuk mengkaji tentang teori-teori yang berhubungan dengan penelitian yang sedang dikerjakan. Teori adalah serangkaian bagian atau variabel, definisi, dan dalil yang saling berhubungan yang menghadirkan sebuah pandangan sistematis mengenai fenomena dengan menentukan hubungan antar variabel, dengan menentukan hubungan antar variabel, dengan maksud menjelaskan fenomena alamiah.

2.1.1 Kecerdasan Buatan (*Artificial Intelligence*)

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 disini merujuk pada mesin yang mampu berpikir, menimbang tindakan yang diambil dan mampu mengambil keputusan seperti yang dilakukan oleh manusia.

Kecerdasan Buatan atau *Artificial Intelligence* (AI) merupakan bidang ilmu komputer yang mempunyai peran penting di era kini dan masa akan datang. Bidang ini telah berkembang sangat pesat di 20 tahun terakhir seiring dengan pertumbuhan kebutuhan akan perangkat cerdas pada industri dan rumah tangga. AI mencakup bidang yang cukup besar. Mulai dari yang paling umum hingga yang khusus. Dari *learning* atau *perception* hingga pada permainan catur, pembuktian teori matekmatika, menulis puisi, mengemudikan mobil, dan melakukan diagnosis penyakit. AI relevan dengan berbagai amcam *task* kecerdasan, AI merupakan sebuah ilmu yang universal (Budiharto & Suhartono, 2014:2). Adapun metode-metode *Artificial Intelligence* (AI) sebagai berikut:

2.1.1.1. Sistem Pakar atau *Expert System*

Sistem pakar adalah aplikasi berbasis komputer yang digunakan untuk menyelesaikan masalah sebagaimana yang dipikirkan oleh pakar. Pakar yang dimaksud di sini adalah orang yang mmepunyai keahlian khusus yang dapat menyelesaikan masalah yang tidak dapat diselesaikan oleh orang awam (Kusrini, 2008: 3).

Sistem pakar adalah sistem komputer yang ditujukan untuk meniru semua aspek kemampuan pengambilan keputusan seorang pakar. Sistem pakar memanfaatkan secara maksimal pengetahuan khusus selayaknya seorang pakar untuk memecahkan masalah(Rosnelly, 2012:2).

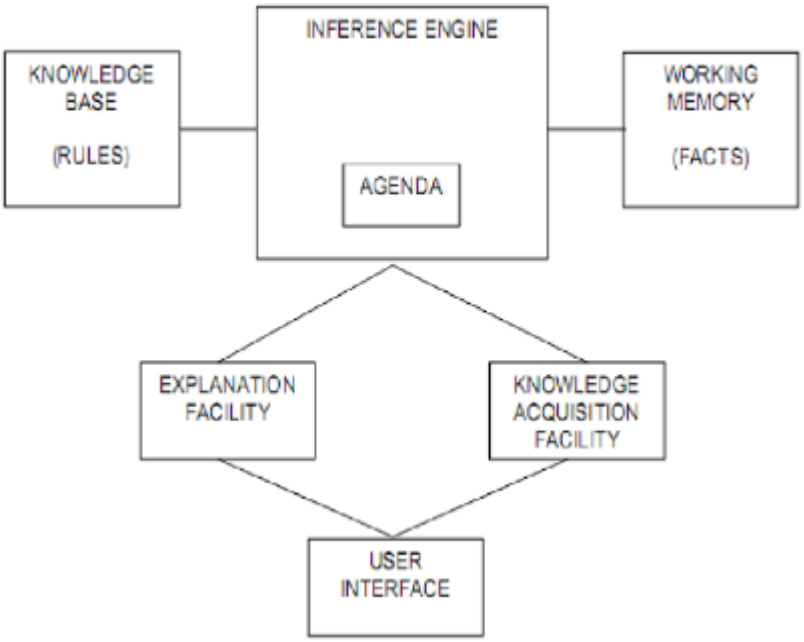
Pakar atau ahli didefinisikan sebagai seorang yang memiliki pengetahuan atau keahlian khusus yang tidak dimiliki oleh kebanyakan orang. Seorang pakar dapat memecahkan masalah yang tidak mampu dipecahkan kebanyakan orang. Dengan kata lain, dapat memecahkan suatu masalah dengan lebih efisien namun bukan berarti lebih murah. Pengetahuan yang dimuat ke dalam sistem pakar dapat berasal dari seorang pakar atau pun pengetahuan yang berasal dari buku, jurnal, majalah, dan dokumentasi yang dipublikasi lainnya, serta orang yang memiliki pengetahuan meskipun bukan ahli. Istilah sistem pakar sering disinonimkan dengan sistem berbasis pengetahuan.

a. **Kelebihan Sistem Pakar**

Sistem pakar memiliki beberapa fitur menarik yang merupakan kelebihannya, seperti Meningkatkan ketersediaan, Mengurangi Biaya, Permanen, Keahlian multipel, Meningkatkan kehandalan, Penjelasan, Respon yang cepat, Stabil, Pembimbing pintar dan Basis data yang cerdas.

b. **Struktur Sistem Pakar**

Adapun struktur sistem pakar dapat dilihat pada gambar berikut (Rosnelly, 2012:13).



Gambar 2.1. Gambar Struktur Sistem Pakar
Sumber: Rosnelly (2012)

Komponen yang terdapat dalam struktur sistem pakar ini adalah *knowledge base (rules), inference engine, working memory, explanation facility, knowledge acquisition facility, user interface.*

1. *Knowledge Base* (Base Pengetahuan)

Basis pengetahuan mengandung pengetahuan untuk pemahaman, formulasi dan dua elemen dasar, yaitu fakta dan aturan. Fakta merupakan informasi tentang objek dalam area permasalahan tertentu, sedangkan aturan merupakan informasi

tentang cara bagaimana memperoleh fakta baru dari fakta yang telah diketahui. Pada struktur sistem pakar diatas, *knowledge base* disini untuk menyimpan pengetahuan dari pakar berupa *rule*/aturan.

2. *Inference Engine* (Mesin Inferensi)

Mesin inferensi merupakan otak dari sebuah sistem pakar dan dikenal juga dengan sebutan *control structure* atau *rule interpreter*. Komponen ini mengandung mekanisme pola pikir dan penalaran yang digunakan oleh pakar dalam menyelesaikan suatu masalah. Mesin inferensi disini adalah processor pada sistem pakar yang mencocokkan bagian kondisi dari rule yang tersimpan didalam *knowledge base* dengan fakta yang tersimpan di *working memory*.

3. *Working Memory*

Berguna untuk menyimpan fakta yang dihasilkan oleh *inference engine* dengan penambahan parameter berupa derajat kepercayaan atau dapat juga dikatakan sebagai global database dari fakta yang digunakan oleh *rule-rule* yang ada.

4. *Explanation Facility*

Menyediakan kebenaran dari solusi yang dihasilkan kepada user (reasoning chain).

5. *Knowledge Acquisition Facility*

Meliputi proses pengumpulan, pemindahan dan perubahan dari kemampuan pemecahan masalah seorang pakar atau sumber pengetahuan terdokumentasi ke

program computer, yang bertujuan untuk memperbaiki atau mengembangkan basis pengetahuan.

6. *User Interface*

Mekanisme untuk memberi kesepakatan kepada *user* dan sistem pakar untuk berkomunikasi. Antar muka menerima informasi dari pemakai dan mengubahnya ke dalam bentuk yang dapat diterima oleh sistem. Selain itu antarmuka menerima informasi dari sistem dan menyajikannya ke dalam bentuk yang dapat dimengerti oleh pemakai.

c. ***Interference Engine* atau Mesin Interferensi**

Inferensi merupakan proses untuk menghasilkan informasi dari fakta yang diketahui atau diasumsikan. Inferensi adalah konklusi logis (*logical conclusion*) atau implikasi berdasarkan informasi yang tersedia (Kusrini, 2008:8). Dalam sistem pakar proses inferensi dilakukan dalam suatu modul yang disebut *Inference Engine* (Mesin Inferensi).

Pada sistem pakar berbasis *rule*, domain pengetahuan direpresentasikan dalam sebuah kumpulan *rule* berbentuk IF-THEN, sedangkan data direpresentasikan dalam sebuah kumpulan fakta-fakta tentang kejadian saat ini. Mesin interferensi membandingkan masing-masing *rule* yang tersimpan dalam basis pengetahuan dengan fakta-fakta yang terdapat dalam *database*. Jika bagian IF (kondisi) dari *rule* cocok dengan fakta, maka *rule* dieksekusi dan bagian THEN (aksi) diletakkan dalam *database* sebagai fakta baru yang ditambahkan (Sutojo, 2011:171)

Ada dua metode inferensi yang penting dalam sistem pakar yaitu:

1. Runut Maju (*Forward Chaining*)

Runut maju berarti menggunakan himpunan aturan kondisi-aksi. Dalam metode ini, data digunakan untuk menentukan aturan mana yang akan dijalankan, kemudian aturan tersebut dijalankan. Mungkin proses menambahkan data ke memori kerja. Proses diulang sampai ditemukan suatu hasil (Wilson, 1998).

Metode inferensi runut maju cocok digunakan untuk menangani masalah pengendalian (*controlling*) dan peramalan (*prognosis*) (Giarattano dan Riley, 1994).

2. Runut Balik (*Backward Chaining*)

Runut balik merupakan metode penalaran kebalikan dari runut maju. Dalam runut balik penalaran dimulai dengan tujuan kemudian merunut balik ke jalur yang akan mengarahkan ke tujuan tersebut (Giarattano dan Riley, 1994). Runut balik disebut juga sebagai *goal-driven reasoning*, merupakan cara yang efisien untuk memecahkan masalah yang dimodelkan sebagai masalah pemilihan terstruktur.

Tujuan inferensi adalah mengambil pilihan terbaik dari banyak kemungkinan. Metode inferensi runut balik ini cocok digunakan untuk memecahkan masalah diagnosi (Schnupp, 1989).

d. **Karakteristik Sistem Pakar.**

Sistem pakar umumnya dirancang untuk memenuhi beberapa karakteristik umum yaitu Kinerja sangat baik, Waktu respon yang baik, Dapat diandalkan, Dapat dipahami dan Fleksibel.

2.1.1.2. Jaringan Sistem Syaraf

Jaringan syaraf tiruan (JST) sistem pemrosesn informasi yang memiliki karakteristik serupa dengan jaringan syaraf biologis dan terdiri dari sejumlah besar elemen pemrosesan sederhana yang disebut neuron, unit, sel, atau node (Irwansyah & Faisal, 2015:42). Ada 3 hal yang diadopsi dari sistem syaraf otak manusia oleh JST adalah Neuron, Topologi, Toleransi Kesalahan (*Fault Tolerant*). Menurut Russel dan Norvig (2003), JST memiliki proses belajar yang berbeda dengan metode yang lainnya, sehingga *Nerual Network* memiliki struktur yang unik dimana metode pelatihannya dibagi ke dalam dua kelompok, yakni sebagai berikut:

a. *Feed-Forward Neural Network* (FFNN)

FFNN adalah jaringan syaraf tiruan dimana hubungan antara sinyal informasinya bergerak hanya satu arah saja dalam mengasosiasikan *input* dengan *output* yang ekstensif digunakan dalam pengenalan pola (Irwansyah & Faisal, 2015:47).

b. *Feed-Back Neural Network* (FBNN)

FBNN merupakan jaringan syaraf tiruan yang memiliki sinyal informasi yang bergerak dari kedua arah pada *layer* jarignannya (Irwansyah & Faisal, 2015:49).

2.1.1.3. Logika *Fuzzy*

Logika *Fuzzy* merupakan salah satu cabang dari bidang *soft computing*. Logika *Fuzzy* merupakan suatu teori himpunan logika yang dikembangkan untuk mengatasi konsep nilai yang terdapat diantara kebenaran (*true*) dan kesalahan (*false*) (Irwansyah & Faisal, 2015:31).

Terdapat 3 metode dalam logika *fuzzy*, yaitu:

a. Metode Mamdani

Sering dikenal sebagai metode max-min. Untuk mendapatkan output diperlukan berbagai tahapan (Pusadan, 2014:14) :

1. Pembentukan himpunan *fuzzy*
2. Aplikasi fungsi implikasi (aturan)
3. Komposisi aturan

b. Metode Sugeno

Metode Sugeno adalah penalaran dengan metode output (konsekuen) sistem tidak berupa himpunan *fuzzy*, melainkan berupa konstanta atau persamaan linear (Rofiq, 2013:6).

c. Metode Tsukamoto

Pada metode ini, setiap aturan direpresentasikan menggunakan himpunan-himpunan *fuzzy*, dengan fungsi keanggotaan yang monoton. Untuk menentukan nilai *output crisp* atau hasil yang tegas, dicari dengan cara mengubah *input* menjadi sebuah bilangan pada domain himpunan *fuzzy* tersebut (Sholihin, Fuad, & Khamiliyah, 2013:501).

2.1.2 Tabel Keputusan

Dalam proses pengambilan keputusan, semua informasi yang diperlukan disusun dalam bentuk ringkasan hasil yang disebut sebagai tabel hasil atau tabel keputusan. Tabel ini merupakan suatu matriks yang terdiri dari baris yang menunjukkan berbagai alternative pilihan/keputusan dan kolom yang menunjukkan nilai harapan untuk setiap alternative pilihan/keputusan pada berbagai keadaan atau situasi yang mungkin terjadi. (Herjanto & Herfan, 2008:26).

Tabel 2.1 Contoh Tabel Keputusan

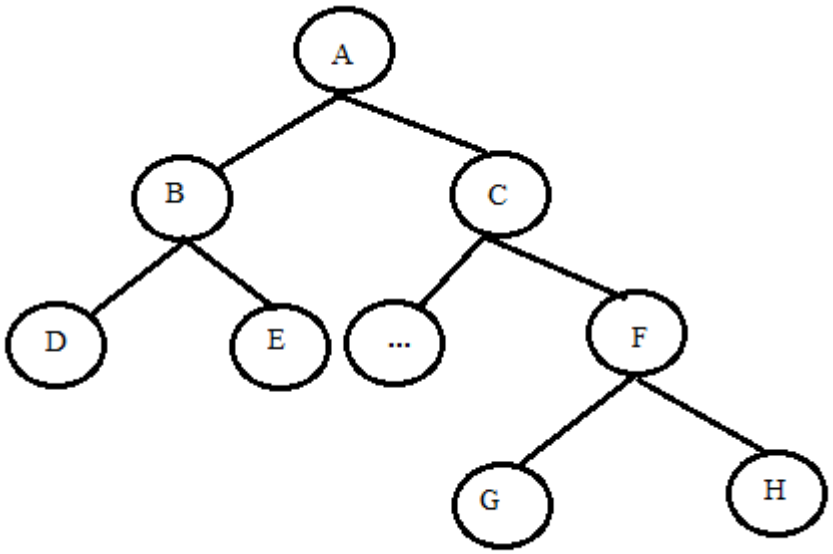
Hipotesa / **Indikator**	**Hipotesa1**	**Hipotesa2**	**Hipotesa3**	**Hipotesa4**
Indikator A	Ya	Ya	Tidak	Ya
Indikator B	Ya	Tidak	Ya	Tidak
Indikator C	Ya	Tidak	Ya	Tidak
Indikator D	Tidak	Ya	Tidak	Ya

Sumber: Data Penelitian (2017)

2.1.3 Pohon Keputusan

Sebuah pohon keputusan adalah sebuah struktur yang dapat digunakan untuk membagi kumpulan data yang besar menjadi himpunan-himpunan *record* yang lebih kecil dengan menerapkan serangkaian aturan keputusan. Dengan masing-masing rangkaian pembagian, anggota himpulan hasil menjadi mirip satu dengan yang lain (Berry & Linoff, 2004 dalam buku (Kusrini & Luthfi, 2009:14)).

Sebuah pohon keputusan mungkin dibangun dengan saksama secara manual atau dapat tumbuh secara otomatis dengan menerapkan salah satu atau beberapa algortima pohon keputusan untuk memodelkan himpunan data yang belum terklasifikasi.



Gambar 2.2 Contoh Pohon Keputusan
Sumber: Data Penelitian (2017)

2.2. Variabel Penelitian

Bendatu (2015: 42) Variabel penelitian juga menjadi suatu bagian penting dalam penelitian. Kemampuan peneliti untuk memahami variable penelitian sangat tergantung pada penguasaan konsep tentang penelitian terutama variable penelitian. Selain itu, pengalaman peneliti dalam melaksanakan penelitian maupun menyusun proposal peneltian juga dapat menambah pemahaman tentang kemampuan

mengidentifikasi variable penelitian. Variabel adalah sebuah konsep yang dioperasionalkan. Lebih tepatnya, operasional properti dari sebuah objek agar dapat dioperasionalkan, diaplikasikan, dan menjadi property dari objek (Swarjana, 2015: 42).

Variabel yang akan digunakan oleh peneliti dalam penelitiannya ada sebagai berikut:

2.2.1 Mesin Manufaktur

Mesin Manufaktur adalah tulang punggung dari setiap negara industri. Staff teknikal dan manufaktur dalam industri harus tau berbagai macam proses manufaktur, bahan yang di proses, alat yang diperlukan untuk manufaktur komponen berbeda atau produk dengan proses rencana keamanan yang optimal untuk menghindari kecelakaan. (Singh, 2006:1)

Indikator kerusakan yang terdapat pada mesin manufaktur adalah sebagai berikut:

1. Pompa air pendingin

   Merupakan sebuah tipe pompa yang digunakan untuk mensirkulasi pendingin, biasanya dalam bentuk air. (Singh, 2006:148)

2. Minyak pelumas

   Merupakan sebuah zat yang digunakan untuk mengurangi gesekan antara dua permukaan yang saling bersentuhan. (Singh, 2006:290)

3. Sparepart

Merupakan sebuah bagian yang dapat ditukarkan yang disimpan disebuah gudang dan digunakan untuk memperbaiki atau menggantikan unit yang gagal. (Singh, 2006:467)

2.3. *Software* Pendukung

Software pemograman biasanya dipakai untuk memudahkan para pembuat program (*programmer*) untuk menulis program yang kemudian dibentuk menjadi sebuah objek yang bisa diakses oleh *system software* dalam bentuk aplikasi. *Software* pendukung yang digunakan untuk merancang sistem aplikasi ini adalah:

2.3.1 UML (*Unified Modeling Language*)

Salah satu pemodelan yang saat ini paling banyak digunakan adalah UML. UML (*Unified Modeling Language*) adalah salah standar bahasa yang banyak digunakan di dunia industry untuk mendefinisikan *requirement*, membuat analis & desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek (Rosa & Shalahuddin, 2011, p. 113)

Berikut adalah beberapa jenis diagram UML:

2.3.1.1. Use Case Diagram

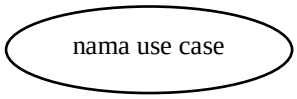
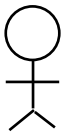
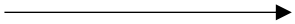
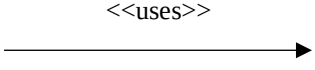
Use case diagram merupakan pemodelan untuk kelakukan sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih actor dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu.

Use case atau diagram *use case* merupakan permodelan 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 (2) hal utama pada use case yaitu pendefinisian apa yang disebut *actor* dan *use case* (Rosa & Shalahuddin, 2015:155)

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

Tabel 2.1 Simbol *Use Case Diagram*

Simbol	Deskripsi
Use Case  	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan meggunakan kata kerja di awal frase nama *use case*
Aktor / *actor*  nama aktor	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 actor
Asosiasi / *association* 	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* yang 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 generalisai dan spesialisasi (umum-khusu) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya
Menggunakan / *include* / *uses*   	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsina atau sebagai syarat dijalankan *use case* ini.

Sumber: Rosa, A. & Shalahuddin, M. (2011)

2.3.1.2. Activity Diagram

Activity diagram menggambarkan aliran kerja atau aktivitas dari sebuah sistem atau proses bisnis. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan actor, jadi aktivitas yang dapat dilakukan oleh sistem.

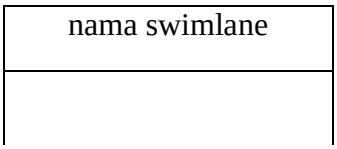
Diagram aktivitas atau *activity* diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan *actor*, jadi aktivitas yang dapat dilakukan oleh sistem. Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut: (Rosa & Shalahuddin, 2015:161)

a. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
b. Urutan atau pengelompokkan tampilan dari sistem/user interface dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
c. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.

   Rancangan menu yang ditampilkan pada perangkat lunak

   Berikut adalah simbol-simbol yang ada pada diagram aktivitas:

Tabel 2.2 Simbol *Activity Diagram*

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  Atau 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi

Sumber: Rosa, A. & Shalahuddin, M. (2011)

2.3.1.3. Sequence Diagram

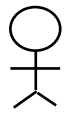
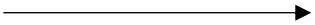
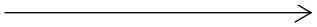
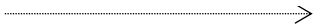
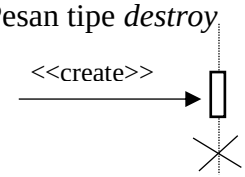
Diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan message yang dikirimkan dan diterima

antara objek. Membuat diagram sekuen juga dibutuhkan untuk melihat skenario yang ada pada *use case*.

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

Berikut adalah simbol-simbol yang ada pada diagram aktivitas:

Tabel *2*.3 Simbol *Sequence Diagram*

Simbol	Deskripsi
Aktor / *actor*  nama aktor	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 actor
Garis hidup / *lifeline*	Menyatakan kehidupan suatu objek
Objek Nama objek : nama kelas	Menyatakan objek yang berinteraksi pesan
Waktu aktif 	Menyatakan objek dalam keadaan aktif dan berinteraksi dengan pesan
Pesan tipe *create* <<create>> 	Objek yang lain, arah panah mengarah pada objek yang dibuat
Pesan tipe *call* 1 : nama_metode() 	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri, arah panah mengarah pada objek yang memiliki operasi/metode, karena ini memanggil operasi/metode maka operasi/metode yang dipanggil harus ada pada diagram kelas sesuai dengan kelas objek yang berinteraksi
Pesan tipe *send* 1: masukan 	Menyatakan bahwa suatu objek mengirimkan data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirim
Pesan tipe *return* 1: keluaran 	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian
Pesan tipe *destroy* <<create>> 	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada *create* maka ada *destroy*

Sumber: Rosa, A. & Shalahuddin, M. (2011)

2.3.1.4. 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.

a. Atribut merupakan variabel-variabel yang dimiliki suatu kelas.
b. 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 & Shalahuddin, 2015: 165):

1. Kelas main

   Kelas yang memiliki fungsi awal di eksekusi 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 yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisian *use case*, 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 disimpan ke basis data.

Berikut adalah simbol-simbol yang ada pada diagram kelas:

Tabel 2.4 Simbol *Class Diagram*

Simbol	Deskripsi
Kelas nama_kelas +atribut +operasi()	Kelas pada struktur sistem
Antarmuka / *interface* nama_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 / *dependency*	Relasi antar kelas dengan makna kebergantungan antar kelas
Agregasi / *aggregation*	Relasi antar kelas dengan makna semua bagian (*whole part*)

Sumber: Rosa, A. & Shalahuddin, M. (2011)

Untuk membuat diagram-diagram diatas, penulis menggunakan aplikasi yang bernama StarUML. StarUML adalah *platform* pemmodelan perangkat lunak yang mendukung UML (*Unified Modeling Language*). StarUML yang berbasiskan pada UML versi 1.4, menyediakan sebelas jenis *Diagram* yang berbeda, dan mendukung notasi UML 2.0 StarUML juga secara aktif mendukung pedekatan MDA (*Model Driven Architercture*) dengan mendukung konsep UML. StarUML unggul dalam hal kustomisasi lingkungan kerja pengguna, dan memiliki ekstensibilitas tinggi pada fungsionalitasnya (Triandini & Suardika, 2012: 1).

Gambar 2.3 Logo StarUML
Sumber: Data Penelitian (2017)

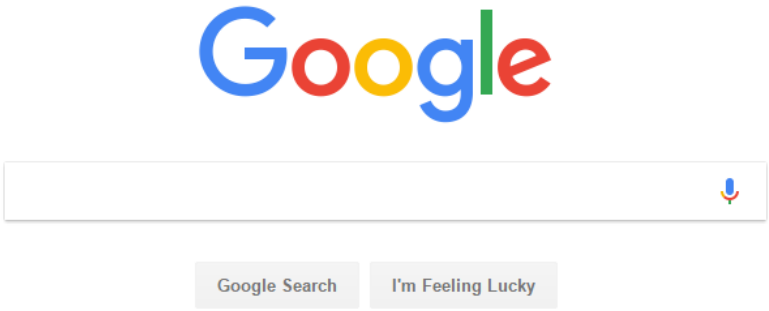
2.3.2 WEB

Web merupakan media informasi berbasis jaringan komputer yang dapat diakses dimana saja dengan biaya yang relatif murah. *Web* merupakan bentuk implementasi dari bahasa pemograman *web* (*web programming*). Sejarah perkembangan bahasa pemograman web diawali dengan munculnya HTML (*Hypertext Markup Language*), yang kemudian dikembangkan dengan munculnya CSS (*Cascading Style Sheet*) yang bertujuan untuk memperindah tampilan *website*. Pada kenyataannya, bahasa tersebut masih memiliki kelemahan, yaitu *script* program dapat dilihat secara utuh, sehingga anda dapat mengetahui kerangka situs tersebut dan memberi peluang *hacker* untuk mengubahnya dengan mudah.

Bahasa pemograman saat ini sudah sangat berkembang dengan berbagai kemudahaan dalam penyajian dan *interface* yang lebih *user friendly*. Penyajian yang baik dari suatu bahasa pemograman akan menghasilkan sebuah *web* dinamis sehingga pengunjung akan lebih mudah mendapatkan informasi yang dibutuhkan.

Beberapa bahasa pemograman *web* berbasis *server* (*server side*) mulai muncul dan dikembangkan oleh beberapa perusahaan perangkat lunak, seperti: ASP (*Active Server Pages*) oleh *Microsoft*, JSP (*Java Server Pages*) oleh *Sun Microsystem*, dan CGI (*Common Gateway Interface*).

Bahasa pemograman tersebut memiliki beberapa kendala, misalnya: *Active Server Pages* keluaran *Microsoft* hanya dapat dijalankan melalui sistem operasi buatan *Microsoft*, dan *Common Gateway Interface* yang hanya berjalan pada sistem operasi berbasis UNIX. Untuk mengatasi keterbatasan tersebut, muncullah bahasa pemograman web yang dapat dijalankan pada sistem operasi *Microsoft*, sekaligus sistem operasi berbasis UNIX. Yaitu: bahasa pemograman PHP (*PHP Hypertext Preprocessor*). (Andi, 2009: 2)



Gambar 2.4 Tampilan WEB Google
Sumber: Data Penelitian (2017)

2.3.3 Visual Studio 2010

Visual Studio adalah *Integrated Development Environment* (IDE) dari untuk membangun aplikasi console dan *Graphical User Interface* (GUI) dengan menggunakan bahasa yang didukung pada *.NET Framework*. Aplikasi GUI yang dapat dibangun diantaranya adalah *Windows Form, Website, Web Application, Windows Mobile*.

Gambar 2.5 Logo Visual Studio 2008
Sumber: Data Penelitian (2017)

Dengan menggunakan Visual Studio 2010, penulis menuliskan *coding* aplikasi menggunakan bahasa *programming* sebagai berikut.

a. Bahasa Pemograman C#

C# merupakan sebuah bahasa pemrograman yang berorientasi objek yang dikembangkan oleh Microsoft sebagia bagian dari inisiatif kerangka .NET *Framework*. Bahasa pemograman ini dibuat berbasiskan bahasa C++ yang telah dipengaruhi oleh aspek-aspek ataupun fitur bahasa yang terdapat pada bahasa-bahasa pemrograman lainnya seperti Java, Delphi, Visual Basic dan lain-lain dengan beberapa penyederhanaan.

Bahasa Pemograman C# (baca: *C-Sharp*) dirancang oleh *Microsoft Corp* sebagai bahasa pemograman yang sangat berdaya-guna, aman (*secure*), serta mudah digunakan. Bahasa pemograman C# juga dapat digunakan untuk mengembangkan aplikasi sarana bergerak (*mobile application*), aplikasi berbasis *Web* (*Web-based applications*), serta aplikasi berskala besar (*enterprise*).

b. Bahasa Pemograman ASP.NET

Active Server Pages .NET (sering disingkat ASP.NET) adalah kumpulan teknologi dalam *Framework* .NET untuk membangun aplikasi web dinamik dan XML *Web Service* (Layanan Web XML). Halaman ASP.NET dijalankan di *server* kemudian akan dibuat halaman markup (penanda) seperti HTML (*Hypertext Markup Language*), WML (*Wireless Markup Language*), atau XML (*Extensible Markup Language*) yang dikirim ke *browser desktop* atau *mobile*.

ASP.NET merupakan teknologi *Microsoft* yang dikhusukan untuk pengembangan aplikasi berbasis web dinamis berbasis platform (Kurniawan 2010:

1).*NET Framework*. ASP.NET didesain untuk memberikan kemudahan pada pengembang web untuk membuat aplikasi berbasis web dengan cepat, mudah, dan efisien karena meminimalkan penulisan kode program dengan bantuan komponen-komponen yang tersedia, sehingga dapat meningkatkan produktivitas.

2.3.4 Database SQL *Server*

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

Gambar 2.6 Logo SQL Server
Sumber: Data Penelitian (2017)

2.4. Penelitian Terdahulu

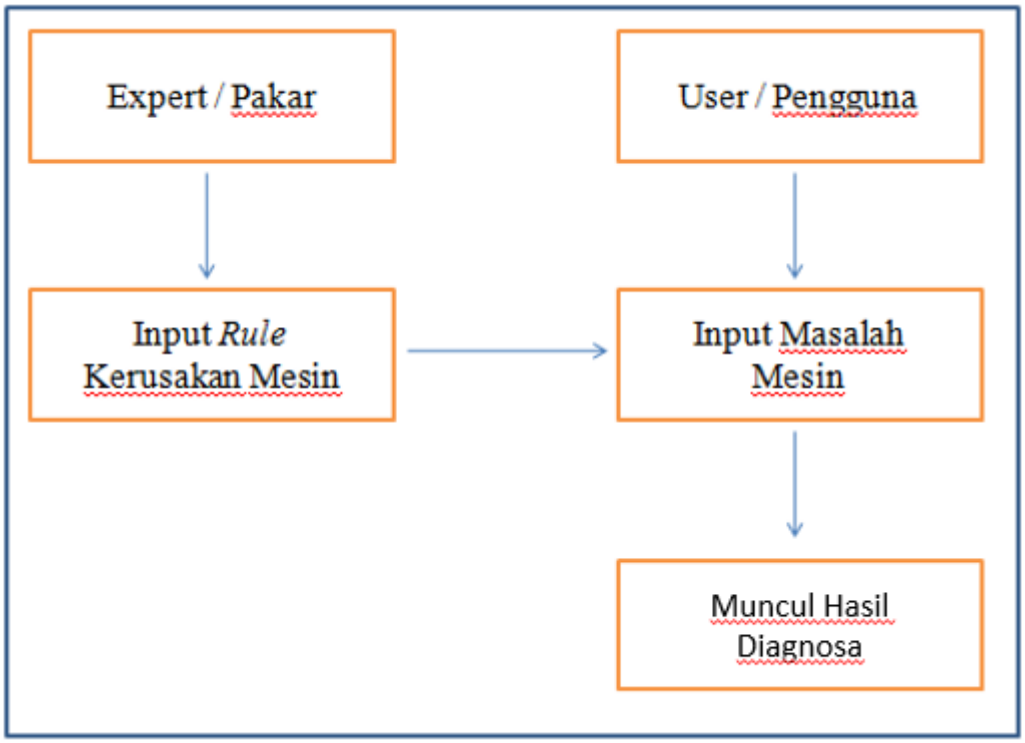
Penulis	Judul	Metode	Hasil
Nency Extise Putri (2016)	Sistem Pakar Kerusakan *Hardware* Komputer Dengan Metode *Forward Chaining*	*Forward Chaining*	Kesimpulan dari Hasil Penelitian adalah metode tersebut layak digunakan.
Purwanto dan Putra (2016)	Sistem Pakar Diagnosa Kerusakan Pada Televisi LED Dengan Menggunakan Metode *Forward Chaining*	*Forward Chaining*	Kesimpulan dari Hasil Penelitian adalah metode tersebut layak digunakan.
Harison dan Alexyusanderia (2014)	Sistem Pakar Perawatan dan Perbaikan Ringan Mobil Bensin Menggunakan Video Tutorial Berbasis Web	*Forward Chaining*	Kesimpulan dari Hasil Penelitian adalah metode tersebut layak digunakan.
Rosmawati Tamin (2015)	Sistem Pakar untuk Diagnosa Kerusakan Pada Printer Menggunakan Metode *Forward Chaining*	*Forward Chaining*	Kesimpulan dari Hasil Penelitian adalah metode tersebut layak digunakan.
Guntur dan Nita Merlina (2016)	Sistem Pakar Diagnosa Kerusakan Pada Mesin Pendingin Ruangan dengan Metode *Forward Chaining*	*Forward Chaining*	Kesimpulan dari Hasil Penelitian adalah metode tersebut layak digunakan.

2.5. Kerangka Pemikiran

Identifikasi masalah dalam penelitian ini adalah banyaknya pengguna mesin manufaktur ini yang tidak dapat memperbaiki sendiri dengan alasan kekurangan ilmu pengetahuan dan belum adanya sebuah sistem yang mampu membantu pengguna untuk memperbaiki kerusakan pada mesin manufaktur tersebut, lambatnya kehadiran teknisi dapat mengakibatkan perusahaan mengalami kerugian yang besar karena mesin manufaktur tidak dapat bekerja selagi rusak.

Kerangka berpikir ini disusun dengan berdasarkan pada tinjauan pustaka dan hasil penelitian yang relevan atau terkait.

Adapun kerangka pemikiran sebagai berikut



Gambar 2.7 Kerangka Pemikiran Penelitian
Sumber: Data Olahan Peneliti (2016)

Adapun uraian dari setiap point diatas adalah sebagai berikut:

2.5.1 *Expert* / Pakar

Merupakan seorang pakar yang memiliki akses seluruh halaman yang terdapat di dalam sistem. Termasuk halaman untuk mengisi *rule*, mau pun penambahan *user* / pengguna.

2.5.2 *User* / Pengguna

Merupakan pengguna sistem, biasanya hanya melakukan login kedalam sistem untuk melaporkan kerusakan yang ada di sistem. Tidak dapat mengisi *rule* ataupun menambah *user* lain.

2.5.3 Input *Rule* / Mesin

Penambahan *Rule* atau Mesin kedalam sistem, sehingga para User dapat melaporkan mesin yang terjadi kerusakan.

2.5.4 Input Masalah Mesin

Apabila pengguna mesin mengalami masalah saat menggunakan mesin, maka dapat ke langkah ini untuk memberitahukan kepada mesin bahwa terjadi kerusakan pada mesin yang mereka gunakan, serta gejala-gejala yang terjadi.

2.5.5 Muncul Hasil Diagnosa

Dari gejala-gejala yang telah di input oleh pengguna, maka sistem akan menampilkan hasil diagnosa kerusakan mesin nya.