BAB II

LANDASAN TEORI

2.1. Tinjauan Pustaka

A. Sistem Pakar *(Expert System)*

Menurut Mudrick dan Rios dalam Al-Fatta (2008:3) mendifinisikan “sistem sebagai seperangkat elemen yang digabungkan satu dengan yang lainnya untuk satu tujuan bersama”.

Menurut Kusrini (2009:17) “ sistem pakar adalah cabang dari kecerdasan buatan dan juga merupakan bidang ilmu yang muncul seiring perkembangan ilmu komputer saat ini. Sistem pakar adalah sistem berbasis komputer yang menggunakan pengetahuan, fakta dan teknik penalaran dalam memecahkan masalah yang biasanya hanya dapat dipecahkan oleh seorang pakar dalam bidang tersebut”. Sistem pakar yang baik dirancang agar dapat menyelesaikan suatu permasalahan tertentu dengan meniru kerja dari para ahli. Dengan sistem pakar ini, orang awam juga dapat menyelesaikan masalah yang cukup rumit yang sebenarnya hanya dapat diselesaikan dengan bantuan para ahli.

Ada tiga orang yang terlibat dalam lingkungan sistem pakar, Menurut Turban dalam Arhami (2008:12), yaitu :

1. Pakar

   Pakar adalah orang yang memiliki pengetahuan khusus, pendapat, pengalaman dan metode, serta kemampuan untuk mengaplikasikan keahliannya guna menyelesaikan masalah.

2. *Knowledge engineer* (Perekayasa Sistem)

   Knowledge engineer adalah orang yang membentu pakar dalam menyusun area permasalahan dengan menginterprestasikan dan mengintergrasikan jawaban-jawaban pakar atas pertanyaan yang diajukan, mengambarkan analogi, mengajukan *counter example* dan menerangkan kesulitan-kesulitan konseptual.

3. Pemakai

   Sistem pakar memiliki beberapa pemakai, yaitu pemakai bukan pakar, pelajar, pembangun sistem pakar yang ingin meningkatkan dan menambah basis pengetahuan dan pakar.

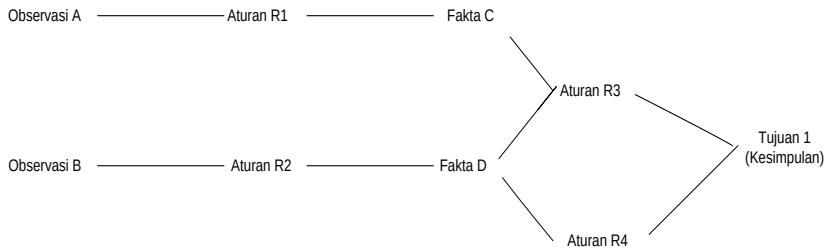
B. Mesin Inferensi *(Inference Engine)*

Menurut Kusrini (2009:19) “Inferensi merupakan proses untuk menghasilkan informasi dari fakta yang diketahui atau diasumsikan, dalam sistem pakar proses inferensi dilakukan pada suatu modul yang berisi program tentang bagaimana mengendalikan proses *reasoning*”. Terdapat dua pendekatan untuk mengontrol inferensi dalam sistem pakar berbasis aturan, yaitu pelacakan ke belakang *(backward chaining*) dan pelacakan ke depan *(foward chaining).*

1. Pelacakan ke Belakang *(Backward Chaining)*

   Menurut Arhami (2008:19) “Pendekatan ke belakang adalah pendekatan yang dimotori tujuan *(goal driven)*”. Pelacakan dimulai dari tujuan, selanjutnya dicari aturan yang memiliki tujuan tersebut untuk kesimpulannya. Selanjutnya proses pelacakan menggunakan premis untuk aturan tersebut

sebagai tujuan baru dan mencari aturan lain dengan tujuan baru sebagai kesimpulannya.



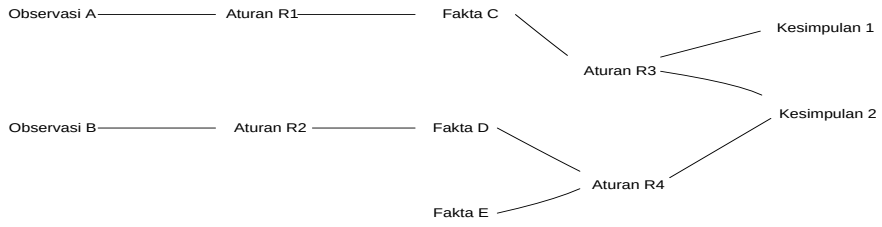
Sumber (Arhami, 2008:19)

Gambar II.1.

Pelacakan Mundur *(Backward Chaining)*

2. Pelacakan ke Depan *(Foward Chaining)*

Menurut Arhami (2008:20) “Pendekatan ke depan adalah pendekatan yang dimotori data *(data driven*)”. Pelacakan dimulai dari informasi masukan, dan selanjutnya mencoba menggambarkan kesimpulan. Pelacakan ke depan mencari fakta yang sesuai dengan bagian *IF* dari aturan *IF-THEN*.



Sumber (Arhami, 2008:19)

Gambar II.2.

Pelacakan Maju (*Forward Chaining*)

C. Metode Pengembangan Sistem

Menurut Kusrini dan Koniyo (2009:44) “Metode pengembangan sistem yang digunakan didasarkan pada pendekatan-pendekatan sistem terstruktur, moduler dan berkembang”.

Adapun pendekatan-pendekatan tersebut adalah sebagai berikut:

1. Pendekatan Sistem (*System Approach*)

   Pendekatan sistem memperhatikan sistem informasi sebagai satu kesatuan yang terintegrasi untuk masing-masing kegiatan atau aplikasinya. Pendekatan sistem ini juga menekankan pada pencapaian sasaran keseluruhan organisasi, tidak hanya pada sasaran sistem informasi itu saja.

2. Pendekatan Terstruktur (*Structure Approach*)

   Pendekatan terstruktur dilengkapi dengan alat-alat (*tools*) dan teknik-teknik (*techniques*) yang dibutuhkan dalam pengembangan sistem sehingga hasil akhir dari sistem yang dikembangkan adalah sistem yang strukturnya didefinisikan dengan baik dan jelas.

3. Pendekatan Moduler (*Moduler Approach*)

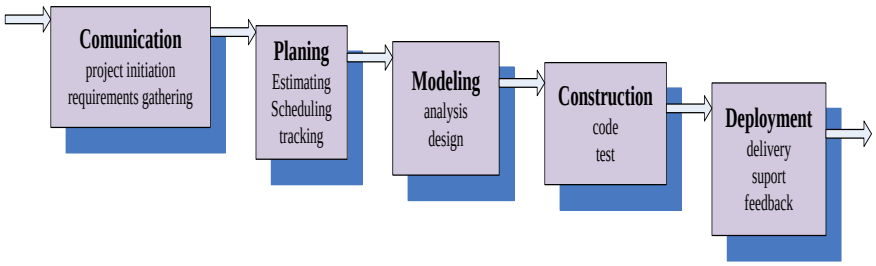
   Pendekatan moduler berusaha memecah sistem yang rumit menjadi beberapa bagian atau modul yang sederhana sehingga sistem lebih mudah dipahami dan dikembangkan.

4. Pendekatan Berkembang (*Evolutionary Approach*) Pendekatan berkembang menerapkan teknologi canggih hanya untuk aplikasi-aplikasi yang diperlukan

pada saat itu saja dan akan terus dikembangkan pada periode-periode berikutnya, memenuhi kebutuhan sesuai perkembangan teknologi.

D. Model *Waterfal*

Dalam penulisan skripsi ini, penulis menggunakan proses *Waterfall Model* sebagai pola pengembangan sistem. Menurut Pressman (2010:39) "*Waterfall Model* adalah model klasik yang bersifat sistematis, berurutan dalam membangun *software*". Adapun tahapannya sebagai berikut:



Sumber (Pressman, 2010:39)

Gambar II.3.

The Waterfall Model

1. *Communication*

   Pada tahap ini akan dilakukan inisiasi proyek, seperti menganalisis masalah yang ada dan tujuan yang akan dicapai. Selain itu dilakukan juga *requirements gathering*, dimana akan dikumpulkan *requirement* dari *user* melalui analisis kuesioner.

2. *Planning*

Tahap ini merupakan tahap dimana akan dilakukan estimasi mengenai kebutuhan-kebutuhan yang diperlukan untuk membuat sebuah sistem. Selain itu, penjadwalan dalam proses pengerjaan juga ditentukan pada tahap ini.

3. *Modeling*

   Kemudian mulai masuk pada tahap perancangan dimana perancang menerjemahkan kebutuhan sistem kedalam representasi untuk menilai kualitas sebelum tahap selanjutnya dikerjakan. Tahap ini lebih difokuskan pada atribut program, seperti struktur data, arsitektur perangkat lunak dan detail prosedur.

4. *Construction*

   Tahap ini merupakan tahap dimana perancangan diterjemahkan ke dalam bahasa yang dimengerti oleh mesin. Setelah itu dilakukan *testing* atau pengujian terhadap sistem yang telah dibuat.

5. *Deployment*

   Setelah proses pengkodean dan pengujian selesai, dilakukan pengiriman yang artinya implementasi kepada masyarakat luas. Pada tahap ini juga dilakukan pemeliharaan, perbaikan dan pengembangan agar sistem tersebut tetap dapat berjalan sebagaimana fungsinya.

E. Konsep Dasar Pemrograman

Program adalah kata, ekspresi, pernyataan atau kombinasinya yang disusun dan dirangkai menjadi satu kesatuan prosedur yang berupa urutan langkah untuk menyelesaikan masalah yang diimplementasikan dengan menggunakan bahasa pemrograman sehingga dapat dieksekusi oleh komputer. Bahasa pemrograman

merupakan prosedur atau tata cara penulisan program. Pemrograman merupakan proses mengimplementasikan urutan langkah untuk menyelesaikan suatu masalah dengan menggunakan suatu bahasa pemrograman.

Menurut Kusrini dan Andri (2009:281) “Pemrograman Terstruktur yaitu suatu bentuk pemrograman yang terorganisir, yang programnya mudah dipahami dan dapat dimodifikasi secara benar”. Ciri Teknik Pemrograman Terstruktur antara lain:

1. Mengandung algoritma pemecahan masalah yang tepat, benar, sederhana, standar dan efektif.
2. Memiliki struktur logika dan struktur program yang benar dan mudah dipahami serta menghindari penggunaan instruksi *GOTO.*
3. Membutuhkan biaya testing, pemeliharaan dan pengembangan yang rendah.
4. Memiliki dokumentasi yang baik.

   Sedangkan standar program yang baik antara lain:

   a. Standar Teknik Pemecahan Masalah Standar teknik dalam pemecahan masalah diantaranya dengan:

      1. Teknik *Top-Down*

         Suatu masalah yang kompleks dibagi-bagi kedalam beberapa tingkatan kelompok masalah hingga sub bagian yang paling kecil, kemudian disusun langkah-langkah untuk menyelesaikannya secara detail.

      2. Teknik *Bottom-Up*

Teknik pemecahan masalah yang sudah mulai ditinggalkan karena sulit melakukan standarisasi proses dari prosedur-prosedur yang sudah ada untuk digabungkan menjadi satu kesatuan.

b. Standar Penyusunan Program

Terdapat beberapa standar dalam penyusunan program, yaitu:

1. Kebenaran logika dan penulisan
2. Waktu minimum untuk penulisan program
3. Kecepatan maksimum eksekusi program
4. Ekspresi penggunaan memori
5. Kemudahan merawat dan mengembangkan program
6. *User friendly*
7. Portabilitas
8. Pemrograman modular.

c. Standar Perawatan Progam

1. Dokumentasi
2. Penulisan instruksi

F. Peralatan Pendukung Sistem (*Tools System*)

Merupakan alat yang digunakan untuk menggambarkan bentuk logika model dari suatu sistem dengan menggunakan simbol-simbol, lambang-lambang, diagram-diagram yang menunjukan secara tepat arti dan fungsinya. Adapun

peralatan pendukung sistem *(tools system)* yang dijelaskan sebagai model sistem yang akan dirancang adalah sebagai berikut:

1. UML (*Unified Modeling Language*)

Menurut Fowler (2011:1) *The Unified Modeling Language* (UML) adalah keluarga dari notasi grafis, yang didukung oleh meta-model tunggal, yang membantu dalam menjelaskan dan merancang sistem perangkat lunak, khususnya sistem perangkat lunak yang dibangun dengan menggunakan gaya yang berorientasi pada objek (*Object-Oriented*).

Itu merupakan definisi yang agak sederhana. Bahkan, UML adalah hal yang berbeda bagi beberapa orang yang berbeda. Hal ini berasal baik dari sejarahnya sendiri dan dari pandangan yang berbeda bahwa orang memiliki tentang apa yang membuat suatu proses rekayasa perangkat lunak yang efektif.

Karena program yang dibuat merupakan pemrograman terstruktur, maka pada tahapan ini penulis akan menjelaskan tentang diagram-diagramnya yaitu:

a. *Use Case Diagram*

Menurut Fowler (2011:99) “adalah suatu teknik untuk menangkap kebutuhan fungsional sistem. *Use Case* bekerja dengan menggambarkan interaksi khas antara pengguna sistem dan sistem itu sendiri, memberikan narasi tentang bagaimana sebuah sistem yang digunakan”.

b. *Activity Diagram*

Menurut Fowler (2011:116) “*Activity Diagram* adalah teknik untuk mendeskripsikan logika prosedural, proses bisnis dan aliran kerja dalam banyak kasus”. *Activity Diagram* mempunyai peran seperti halnya

flowchart, akan tetapi perbedaannya dengan *flowchart* adalah *activity diagram* bisa mendukung perilaku paralel sedangkan *flowchart* tidak bisa.

c. *Component Diagram*

Menurut Fowler (2011:141) "*Component Diagram* digunakan ketika ingin membagi sistem menjadi komponen dan ingin menampilkan hubungan timbal balik mereka melalui antarmuka atau rincian dari komponen ke dalam struktur-tingkat yang lebih rendah".

d. *Deployment diagram*

Menurut Fowler (2011:97) "*Deployment Diagram* menunjukkan tata letak sebuah sistem secara fisik, menampakkan bagian-bagian *software* yang berjalan pada bagian-bagian *hardware*".

Ada empat macam relasi dalam *Unified Modeling Language* (UML) yaitu:

1. Pengklasifikasian (*Classifier*)

Pengklasifikasi (*Classifier*) pada prinsipnya merupakan konsep diskret dalam model yang memiliki identitas (*identity*), *state*, perilaku (*behavior*), serta relasi dengan mengklasifikasi yang lainnya (*relationship*)

2. Asosiasi (*Association*)

Asosiasi (*Association*) pada dasarnya mendeskripsikan koneksi diskret antara objek atau antar *instance* lain dalam sistem atau perangkat lunak yang sedang dikembangkan.

3. Generalisasi (*Generalization*)

   Menggambarkan hubungan antara *use case* yang bersifat umum dengan *use case-use case* yang bersifat lebih spesifik.

4. Realisasi (*Realization*)

   Relasi realisasi (*realization*) menghubungkan elemen-elemen model, misalnya kelas ke elemen-elemen model lainnya, seperti suatu antarmuka yang menyediakan spesifikasi perilaku tetapi bukan strukturnya atau implementasinya.

2. ERD (*Entity Relationship Diagram*)

   a. Pengertian ERD (*Entity Relationship Diagram*)

      Menurut Yuhefizard (2008:17) "*Entity Relationship Diagram* digunakan untuk menggambarkan secara sistematis hubungan antar *entity-entity* yang ada dalam suatu sistem *database* menggunakan simbol-simbol sehingga lebih mudah dipahami". Simbol-simbol yang boleh digunakan adalah:

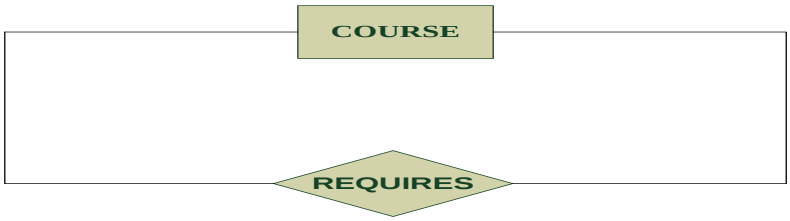
      1. Persegi panjang, berfungsi untuk menyatakan suatu *entity*.
      2. Elips, berfungsi untuk menyatakan *attribute*, jika diberi garis bawah menandakan bahwa *attribute* tersebut merupakan *attribute*/*field* kunci.
      3. Belah ketupat, menyatakan jenis relasi.
      4. Garis penghubungan antara relasi dengan *entity* dan antara *entity* dengan *attribute*.

   b. Derajat *Relationship*

      Terdapat tiga macam derajat dari *relationship*, yaitu:

      1. Derajat Satu (*Unary Degree*)

Bila satu *entity* mempunyai relasi terhadap dirinya sendiri.



Sumber (Mittal dan Jain, 2010:20.11)

Gambar II.4.

Contoh Derajat Satu (*Unary Degree*)

2. Derajat Dua (*Binary Degree*)

   Bila satu relasi menghubungkan dua *entity*.

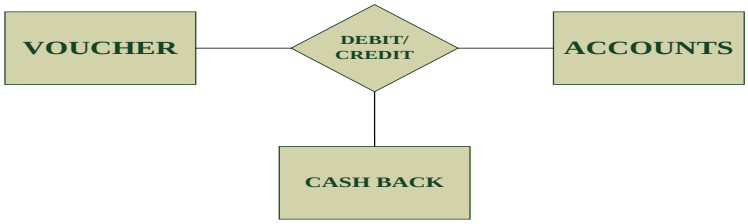
   Sumber (Mittal dan Jain, 2010:20.11)

Gambar II.5.

Contoh Derajat Dua (*Binary Degree*)

3. Derajat Tiga (*Ternary Degree*)

   Bila satu *entity* menghubungkan lebih dari dua *entity*



Sumber (Mittal dan Jain, 2010:20.11)

Gambar II.6.

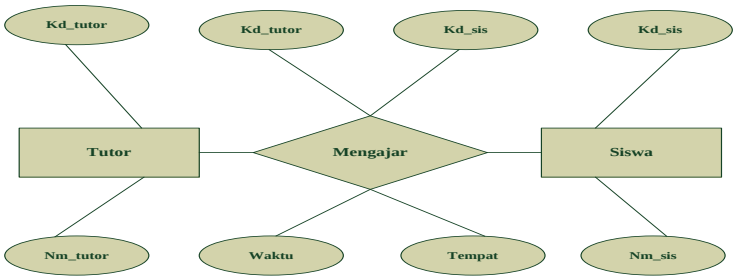
Contoh Derajat Tiga (*Ternary Degree*)

c. *Cardinality Ratio*

Terdapat empat jenis *Cardinality Ratio*, yaitu:

1. 1 : 1 (*One-To-One*)

   Sebuah *entity* A diasosiasikan pada sebuah *entity* B, dan sebuah *entity* B diasosiasikan dengan paling banyak sebuah *entity* A.



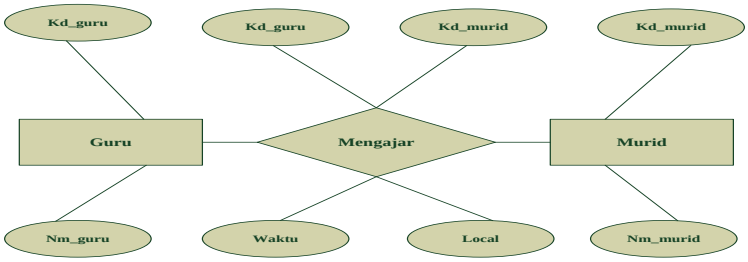
Sumber (Yuhefizard, 2008:18)

Gambar II.7.

Contoh relasi 1 : 1 (*One-To-One*)

2. 1 : N (*One-To-Many*)

   Sebuah *entity* A diasosiasikan dengan sejumlah *entity* B, tetapi *entity* B dapat diasosiasikan paling banyak satu *entity* A.



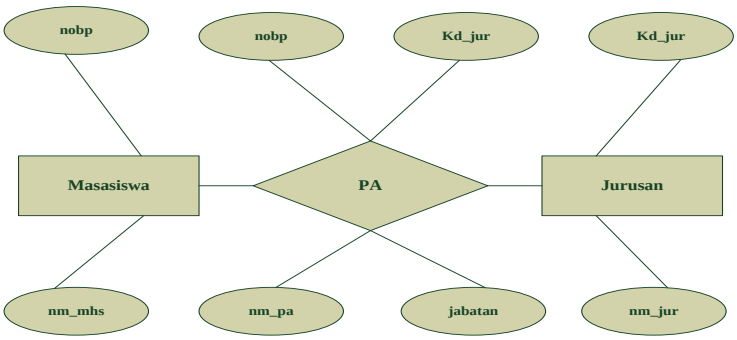
Sumber (Yuhefizard, 2008:19)

Gambar II.8.

Contoh relasi 1 : N (*One-To-Many*)

3. N : 1 (*Many-To-One*)

   Suatu *entity* A dapat diasosiasikan dengan paling banyak sebuah *entity* B, tetapi *entity* B dapat diasosiasikan dengan sejumlah *entity* di A.



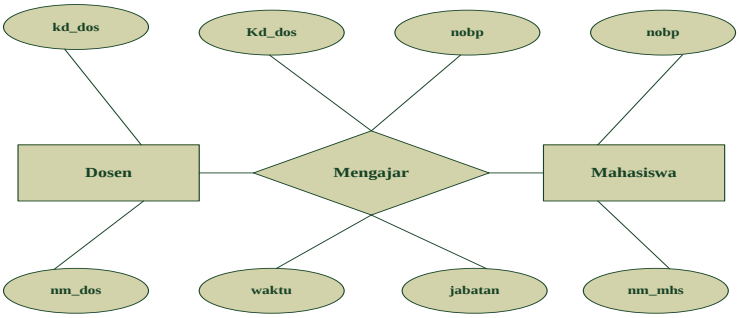
Sumber (Yuhefizard, 2008:20)

Gambar II.9.

Contoh relasi N : 1 (*Many-To-One*)

4. M : N (*Many-To-Many*)

   Suatu *entity* A dapat diasosiasikan dengan sejumlah *entity* B dan *entity* B dapat diasosiasikan dengan sejumlah *entity* di A.



Sumber (Yuhefizard, 2008:21)

Gambar II.10.

Contoh relasi M : N (*Many-To-Many*)

d. *Participation Constraint*

Terdapat dua macam *Participation Constraint*, yaitu:

1. *Total Participation*

   Keberadaan suatu *entity* tergantung pada hubungannya dengan *entity* lain.

2. *Partial Participation*

   Keberadaan suatu *entity* tidak tergantung pada hubungannya dengan *entity* lain.

e. Tahapan pembuatan ERD

Menurut Yuhefizard (2008:21) terdapat empat tahapan pembuatan ERD, yaitu:

1. Mengidentifikasi dan menetapkan seluruh *entity* yang terlibat dalam sistem *database* tersebut.
2. Menentukan *attribute-attribute* atau *field* dari masing-masing *entity*

beserta kunci (*key*)-nya.

3. Mengidentifikasi dan menetapkan seluruh himpunan relasi di antara himpunan-himpunan *entity* yang ada beserta kunci tamu (*foreign key*)-nya.
4. Menentukan derajat relasi untuk setiap himpunan relasi.

G. Mesin

Menurut Yulirianto (2014:1) memberi batasan bahwa “mesin adalah suatu media atau alat untuk mengubah energi seperti air, panas, angin, listrik, atom menjadi energi gerak *(mechainical energy)*”. Beberapa komponen mesin motor, antara lain:

1. *Cylinder Head*, merupakan salah satu bagian terpenting dalam mesin motor yang berarti kepala silinder. Beberapa komponen *Cylinder Head*, antara :
   a. *Noken as*/*Cam shaft* yang berfungsi mengatur buka tutup klep/*valve* dengan teratur.
   b. Klep/*Valve* berfungsi sebagi pintu jalur pemasukan dan pembuangan bahan bakar yang berupa gas atau kabut dari karburator/*injector* kedalam ruang bakar.
   c. *Rocker arm/*Pelatuk berfungsi sebagai perantara untuk membuka klep.
   d. Pen Platuk berfungsi sebagai poros tempat kedudukan pelatuk.
   e. *Valve Spring*/Per klep berfungsi untuk menekan klep kembali ke posisi semula setelah terbuka agar dapat menutup rapat.

f. *Stem Seal*/Penuntun Klep disebut juga bos klep/*valve guides* berfungsi sebagai penuntun gerakan klep.

2. *Cylender Block*/Blok Silinder salah satu fungsi *cylender block* sebagai tempat naik turunnya piston dan untuk menampung bahan bakar sementara sebelum di kompresikan. Beberapa bagian dari *cylender block*, antara lain:

   a. Piston/Torak salah satu fungsi piston adalah mengisap dan mengkompresikan gas baru yang masuk ke ruang bakar serta membuang gas hisap sisa hasil proses pembakaran ke knalpot.

   b. *Ring* Piston berfungsi mempertahankan kerapatan antara piston dinding dengan dinding silinder agar tidak terjadi kebocoran pada saat gas di kompresi oleh piston.

   3. Blok Magnet mempunyai beberapa bagian antara lain :

   a. Spul adalah inti besi atau inti magnet yang dililit atau di gulung oleh kumparan kawat *email* berfungsi untuk menyuplai arus listrik ke *CDI (Capasitor Dischanger ignitation).*

   b. Magnet *(Flywheel Generator)* berfungsi untuk menginduksi spul agar menghasilkan arus listrik.

2.2. Penelitian Terkait

Pengembangan sistem pakar untuk mendeteksi kerusakan dan penanganannya sudah banyak diusulkan. Seperti, Tamin (2015 : 44) melakukan penelitian perancangan dan pembuatan sistem pakar untuk menditeksi kerusakan printer menggunakan bahasa pemograman *visual basic* dengan metode *forward chaining*. Untuk membantu para pengguna printer mengetahui penyebab, akibat dan gejala-gejala yang ditimbulkan dari kerusakan printer, memudahkan para pengguna printer jenis canon untuk mencari solusi kerusakan printer , memudahkan para pengguna printer untuk mendapatkan informasi mengenai cara merawat printer dan memudahkan para teknisi untuk memperbaiki printer.

Selain itu, Gunawan (2013:125) juga melakukan penelitian terhadap sistem pakar kerusakan hardware laptop. Penelitian ini membahas pengembangan sistem kerusakan hardware laptop dengan metode *forward chaining* untuk memudahkan pengguna laptop mendapatkan solusi atas kerusakan hardware dengan dibuatnya perangkat lunak untuk mengatasi kerusakan hardware.

Berdasarkan jurnal diatas untuk kerusakan yang lebih spesifik pada mesin sepeda motor matic juga membutuhkan pengetahuan dan memperbaikinya. Baik itu pengetahuan tentang gejala-gejala kerusakannya sampai cara memperbaikinya. Untuk lebih memberi kemudahan dalam penanganannya, maka perlu dibuatnya sebuah program sistem pakar untuk mendiagnosis kerusakan pada mesin sepeda motor matic dengan metode *forward chainig.*