BAB II
KAJIAN PUSTAKA

2.1 Teori Dasar

Dalam kaitannya dengan penelitian, teori berfungsi untuk memperjelas dan mempertajam ruang lingkup atau konstruk variabel yang akan diteliti. Deskripsi teori paling tidak berisi tentang penjelasan terhadap variabel-variabel yang diteliti melalui pendefinisian, dan uraian yang lengkap dan mendalam dari berbagai referensi, sehingga ruang lingkup, kedudukan dan prediksi terhadap hubungan antara variabel yang akan diteliti menjadi lebih jelas dan terarah (Sugiyono, 2014: 57-58).

Untuk mendukung penelitian, maka pada bagian ini akan dijelaskan teori mengenai Kecerdasan Buatan *(Artificial Intelligence)* dan beberapa sub disiplin ilmunya seperti Logika *Fuzzy (Fuzzy Logic)*, Jaringan Saraf Tiruan (JST), Sistem Pakar *(Expert System)*. Selain itu akan dijelaskan juga teori mengenai pengujian sistem.

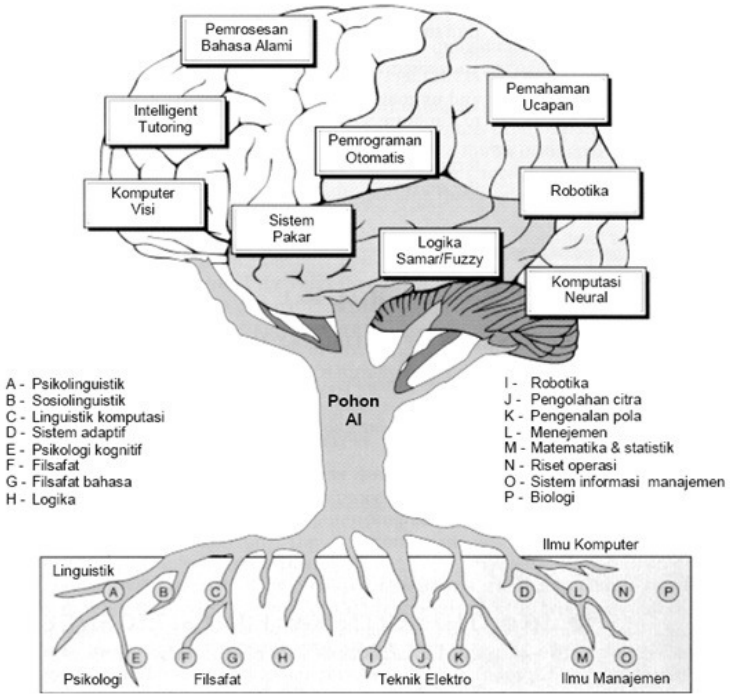
2.1.1 Kecerdasan Buatan *(Artificial Intelligence)*

Kecerdasan dapat diartikan kemampuan untuk membuat keputusan yang tepat dengan mempertimbangkan beberapa masukan dan kemungkinan tindakan yang dapat diambil. Kecerdasan dapat diterapkan pada sistem komputer, contohnya adalah pada permainan catur (Jones, 2008: 1-2).

Husda (2012: 192-193) menyatakan bahwa kecerdasan buatan atau *Artificial Intelligence (AI)* adalah suatu sistem informasi yang berhubungan dengan penangkapan, pemodelan dan penyimpanan kecerdasan manusia dalam sebuah sistem teknologi informasi sehingga sistem tersebut memiliki kecerdasan seperti yang dimiliki manusia. Sistem ini dikembangkan untuk mengembangkan metode dan sistem untuk menyelesaikan masalah, biasanya diselesaikan melalui aktifitas intelektual manusia.

Perkembangan kecerdasan buatan bermula pada tahun 1965, John McCarty melakukan penelitian bersama 3 rekannya di Darthmouth College. Selama kurang lebih 2 bulan, mereka melakukan penelitian dalam bidang *automata*, jaringan saraf dan pembelajaran intelejensia. Hasilnya adalah program yang dinamakan *Principia Mathematica* yang mampu berpikir non-numerik dan menyelesaikan masalah pemikiran. Hal ini menjadikan McCarty disebut sebagai *father of AI* atau bapak AI. (Suyanto, 2014: 5).

Pada awal tahun 1980-an, penelitian tentang AI dibangkitkan kembali dengan kesuksesan komersial dari *software* jenis *expert system* (salah satu bentuk AI yang mensimulasikan pengetahuan dan kemampuan analisis dari satu atau lebih pakar). Kemudian pada era 1990-an dan awal abad 21, AI mencapai kesuksesan terbesarnya karena telah diadopsi secara luas pada industri teknologi, memberikan sumbangan besar pada logistik, data *mining*, diagnosis medis dan berbagai bidang lainnya (Al Fatta, 2009: 4).



Gambar 2.1 Pohon AI
(Sumber: http://pbsabn.lecture.ub.ac.id/files/2012/05/Untitled8.jpg)

Induk keilmuan AI dapat dilihat pada akar dari pohon AI seperti pada gambar 2.1 yang antara lain meliputi: Bahasa/linguistik, Psikologi, Filsafat, Teknik Elektro, Ilmu Komputer dan Ilmu Manajemen. Sutojo, Mulyanto, dan Suhartono (2011: 12-26) menjelaskan bahwa kombinasi antara AI dengan bidang ilmu yang lainnya melahirkan sub disiplin ilmu dalam AI, beberapa diantaranya adalah Logika *Fuzzy* (*Fuzzy Logic*), Jaringan Saraf Tiruan (JST) dan Sistem Pakar (*Expert System*).

2.1.2 Logika Fuzzy *(Fuzzy Logic)*

Fuzzy logic diciptakan oleh Lotfi Zadeh di University of California, Berkeley pada tahun 1965 (Jones, 2008: 410). Menurut Naba (2009: 1), secara

umum f*uzzy logic* adalah sebuah metodologi berhitung dengan variabel berupa kata-kata *(linguistic variable)* sebagai pengganti berhitung dengan bilangan. Kata-kata yang dipakai dalam *fuzzy logic* sesuai dengan intuisi manusia dimana manusia bisa langsung merasakan nilai dari variabel kata-kata yang sudah dipakai sehari-hari. *Fuzzy logic* memberi ruang dan bahkan mengeksploitasi toleransi terhadap ketidakpresisian. Sistem *fuzzy* adalah sebuah sistem yang dibangun dengan definisi, cara kerja dan deskripsi yang jelas berdasar pada teori *fuzzy logic*.

Fuzzy logic merupakan suatu cara yang cocok untuk melakukan tugas pemetaan atau *mapping* hubungan *input* dan *output* dari suatu sistem berdasarkan data *input-output*. Sebagai ilustrasi, seseorang meminta orang lain untuk mengatur panas air yang akan dipakai untuk mandi, orang yang dimintai tolong akan mengatur putaran keran air panas dan keran air dingin agar menghasilkan air yang panasnya sesuai permintaan. Dalam hal ini orang yang mengatur keran air dapat dikatakan bertindak sebagai sistem *fuzzy* karena hanya berdasarkan variabel kata-kata yang tidak presisi bisa merespon dengan memberikan jawaban atau tindakan yang cocok sesuai dengan keinginan orang yang minta tolong (Naba, 2009: 2-3).

Ada beberapa alasan mengapa *fuzzy logic* digunakan untuk memecahkan persoalan, seperti yang diungkapkan oleh Naba (2009: 3-4) berikut ini:

1. Konsep *fuzzy logic* sangat sederhana sehingga mudah dipahami.
2. *Fuzzy logic* adalah fleksibel dalam arti dapat dibangun dan dikembangkan dengan mudah tanpa harus memulai dari nol.
3. *Fuzzy logic* memiliki toleransi terhadap ketidakpastian data.

4. Pemodelan/pemetaan untuk mencari hubungan data *input-output* dari sembarang sistem *black-box* bisa dilakukan dengan memakai sistem *fuzzy*.
5. Pengetahuan atau pengalaman dari para pakar dapat dengan mudah dipakai untuk membangun *fuzzy logic*.
6. *Fuzzy logic* dapat diterapkan dalam desain sistem kontrol tanpa harus menghilangkan teknik desain sistem kontrol konvensional yang sudah ada.
7. *Fuzzy logic* berdasarkan pada bahasa manusia.

Menurut Sutojo, dkk. (2011: 233-237) ada beberapa metode yang digunakan dalam sistem inferensi *fuzzy*, yaitu:

1. Metode Tsukamoto

Dalam inferensinya, metode Tsukamoto menggunakan tahapan sebagai berikut:

a. *Fuzzyfikasi*
b. Pembentukan basis pengetahuan *fuzzy* (*rule* dalam bentuk IF…THEN)
c. Mesin inferensi menggunakan fungsi implikasi MIN (*minimum*)
d. *Defuzzyfikasi* menggunakan metode rata-rata (*average*)

2. Metode Mamdani

Metode ini sering digunakan karena strukturnya yang sederhana. Pada metode ini, untuk mendapatkan *output* diperlukan 4 tahapan sebagai berikut:

a. *Fuzzyfikasi*
b. Pembentukan basis pengetahuan *fuzzy* (*rule* dalam bentuk IF…THEN)

c. Aplikasi fungsi implikasi menggunakan fungsi MIN (*minimum*) dan komposisi antar-*rule* menggunakan fungsi MAX (*maximum*) sehingga menghasilkan himpunan *fuzzy* baru

d. *Defuzzyfikasi* menggunakan metode titik tengah *(centroid)*

3. Metode Sugeno

Metode ini diperkenalkan oleh Takagi-Sugeno Kang pada tahun 1985. Dalam metode ini, *output* sistem berupa konstanta atau persamaan linier. Dalam inferensinya, metode Sugeno menggunakan tahapan sebagai berikut:

a. *Fuzzyfikasi*

b. Pembentukan basis pengetahuan *fuzzy* (*rule* dalam bentuk IF…THEN)

c. Mesin inferensi menggunakan fungsi implikasi MIN *(minimum)*

d. *Defuzzifikasi* menggunakan metode rata-rata *(average)*

2.1.3 Jaringan Saraf Tiruan (JST)

Jaringan Saraf Tiruan (JST) merupakan salah satu upaya untuk memodelkan cara kerja atau fungsi sistem saraf manusia dalam melaksanakan tugas tertentu. Pemodelan ini didasari oleh kemampuan otak manusia dalam mengorganisasikan sel-sel penyusunnya yang disebut *neuron*, sehingga mampu melaksanakan tugas-tugas tertentu, khususnya pengenalan pola dengan efektivitas yang sangat tinggi. Sel saraf *(neuron)* adalah unit pemrosesan informasi yang merupakan dasar dari operasi JST (Suyanto, 2014: 169-170).

Cara kerja JST sama seperti cara kerja manusia, yaitu belajar melalui contoh. Pola atau susunan *neuron-neuron* pada JST berhubungan erat dengan

proses belajar yang digunakan untuk melatih jaringan. Secara umum, arsitektur jaringan pada JST dibagi menjadi 4 macam, yaitu (Suyanto, 2014: 174-177):

1. *Single-Layer Feedforward Networks*

   Bentuk jaringan ini adalah bentuk jaringan berlapis yang paling sederhana, hanya terdapat *input layer* dengan *node* sumber yang terproyeksi ke dalam *output layer.*

2. *Multi-Layer Feedforward Networks*

   Pada bentuk jaringan ini, terdapat satu atau lebih lapisan tersembunyi *(hidden layer)* yang berada di antara *input layer* dan *output layer*.

3. *Recurrent Networks*

   JST dengan bentuk jaringan *recurrent networks* memiliki minimal 1 *feedback loop* yang berfungsi untuk mengembalikan hasil *output* suatu *neuron* sebagai *input* bagi *neuron* yang lain.

4. *Lattice Structure*

   Sebuah *lattice* (kisi-kisi) terdiri dari minimal 1 dimensi *array neuron* dengan himpunan *node* sumber yang bersesuaian yang memberikan sinyal *input* ke *array*.

Berdasarkan cara memodifikasi bobot dari *neuron*-nya, proses pelatihan jaringan saraf tiruan dibagi menjadi dua, yaitu (Sutojo, dkk., 2011: 301- 392):

1. Pelatihan dengan supervisi (pembimbing)

   Dalam pelatihan ini, jaringan dipandu oleh sejumlah pasangan data (masukan dan target) yang berfungsi sebagai pembimbing untuk melatih jaringan

hingga diperoleh bobot yang terbaik. Algoritma yang termasuk dalam pelatihan dengan supervisi antara lain:

a. *Hebb-Rule*

   Model ini diperkenalkan oleh D.O. Hebb yang menggunakan cara menghitung bobot dan bias secara iteratif dengan memanfaatkan model pembelajaran dengan supervisi sehingga bobot dan bias dapat dihitung secara otomatis tanpa harus melakukan cara coba-coba. Arsitektur jaringan ini terdiri dari beberapa unit *input* dihubungkan langsung dengan sebuah unit *output*, ditambah dengan sebuah bias.

b. *Perceptron*

   Model ini ditemukan oleh Rosenblatt (1962) dan Minsky – Papert (1969). Model jaringan ini merupakan model yang terbaik pada saat itu. Algoritma pelatihan *perceptron* digunakan baik untuk *input* biner maupun bipolar, dengan θ tertentu.

c. *Delta-Rule*

   Selama pelatihan pola, *Delta-Rule* akan mengubah bobot dengan cara meminimalkan *error* antara *output* jaringan dengan target.

d. *Backpropagation*

   Backpropagation adalah metode penurunan gradien untuk meminimalkan kuadrat *error* keluaran. Pelatihan jaringan ini terdiri dari 3 tahap, yaitu tahap perambatan maju (*forward propagation*), tahap perambatan balik, dan tahap perubahan bobot dan bias. Arsitektur jaringan ini terdiri dari *input layer*, *hidden layer*, dan *output layer*.

e. *Heteroassociative Memory*

Jaringan saraf *heteroassociative memory* adalah jaringan yang dapat menyimpan kumpulan pengelompokan pola dengan cara menentukan bobot-bobotnya sedemikian rupa. Algoritma pelatihan yang biasa digunakan adalah *Hebb-Rule*.

f. *Bidirectional Associative Memory* (*BAM*)

Bidirectional Associative Memory (*BAM*) adalah model jaringan saraf yang memiliki 2 lapisan, yaitu lapisan *input* dan lapisan *output* yang mempunyai hubungan timbal balik antara keduanya (bersifat *bidirectional*). Arsitektur jaringan ini terdiri dari 3 *neuron* pada lapisan *input* dan 2 *neuron* pada lapisan *output*. Model jaringan ini terbagi menjadi 2 jenis yaitu BAM diskrit dan BAM kontinu.

g. *Learning Vector Quantization* (*LVQ*)

Learning Vector Quantization (*LVQ*) adalah suatu model pelatihan pada lapisan kompetitif terawasi yang akan belajar secara otomatis untuk mengklasifikasikan vektor-vektor *input* ke dalam kelas-kelas tertentu.

2. Pelatihan tanpa supervisi (pembimbing)

Dalam pelatihan ini, tidak ada pembimbing yang digunakan untuk memandu proses pelatihan. Jaringan hanya diberi *input* tetapi tidak mendapatkan target yang diinginkan sehingga modifikasi bobot pada jaringan dilakukan menurut parameter tertentu. Model jaringan yang termasuk dalam pelatihan tanpa supervisi adalah jaringan kohonen yang diperkenalkan oleh Prof. Teuvo Kohonen pada tahun 1982.

Pada jaringan kohonen, *neuron-neuron* pada suatu lapisan data akan menyusun dirinya sendiri berdasarkan *input* nilai tertentu dalam suatu *cluster*. *Cluster* yang dipilih sebagai pemenang adalah *cluster* yang mempunyai vektor bobot paling cocok dengan pola *input*, yaitu *cluster* yang memiliki jarak yang paling dekat.

2.1.4 Sistem Pakar *(Expert System)*

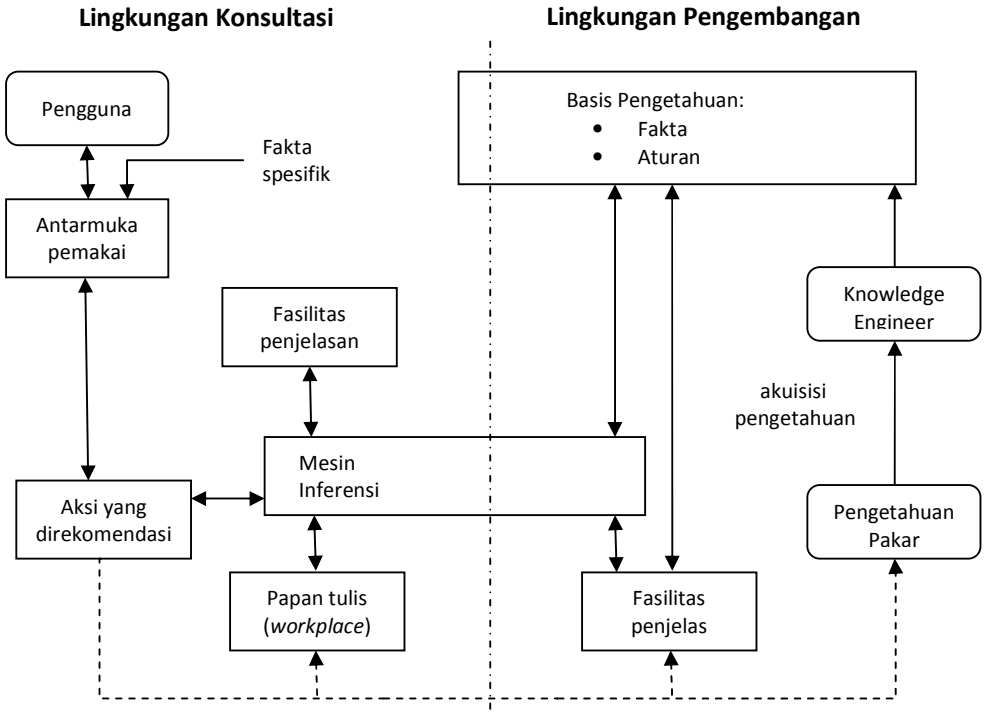
Menurut Wijaya (2007) dalam Hayadi (2016: 2), sistem pakar adalah sebuah program komputer yang dirancang untuk mengambil keputusan seperti keputusan yang diambil oleh seorang pakar, dimana sistem pakar menggunakan pengetahuan *(knowledge)*, fakta dan teknik berpikir dalam menyelesaikan masalah-masalah yang biasanya hanya dapat diselesaikan oleh seorang pakar dari bidang yang bersangkutan. Menurut Kusrini (2008: 3), pakar adalah orang yang mempunyai keahlian khusus yang dapat menyelesaikan masalah yang tidak dapat diselesaikan oleh orang awam.

Terdapat beberapa contoh sistem pakar yang berhasil dikembangan untuk membantu pekerjaan manusia, diantaranya adalah (Husda, 2012: 187-188):

1. MYCIN : sistem pakar ini dibuat oleh Edward Shortlife pada tahun 1970-an. MYCIN adalah sistem pakar yang bisa mendiagnosa penyakit infeksi dan memberikan rekomendasi pengobatan.
2. DENDRAL : mengidentifikasi sturktur molekular campuran bahan kimia yang tidak dikenal.

3. XCON : merupakan sistem pakar untuk membantu konfigurasi sistem komputer besar.
4. XSEL : dirancang untuk membantu karyawan bagian penjualan dalam memilih komponen sistem VAX.
5. PROSPECTOR : sistem pakar yang membantu ahli geologi dalam mencari dan menemukan deposit.
6. DELTA : dibuat oleh perusahaan General Electric untuk membantu karyawan bagian pemeliharaan mesin lokomotif diesel dalam memantau mesin-mesin yang tidak berfungsi dengan baik dan membimbing ke arah prosedur perbaikan.
7. FOLIO : sistem pakar yang menolong broker dan manajer dalam menangani investasi untuk kepentingan pelanggan.

Husda (2012: 182) menyatakan bahwa sistem pakar tediri dari 2 bagian utama yaitu lingkungan pengembangan *(development environment)* dan lingkungan konsultasi *(consultation environment)*. Lingkungan pengembangan digunakan untuk memasukkan pengetahuan pakar ke dalam lingkungan sistem pakar, sedangkan lingkungan konsultasi digunakan oleh pengguna yang bukan pakar untuk memperoleh pengetahuan pakar. Pada gambar 2.2 dapat dilihat arsitektur sistem pakar.



Gambar 2.2 Arsitektur Sistem Pakar
(Sumber: Husda, 2012: 183)

Menurut Husda (2012: 183-185), komponen-komponen sistem pakar yaitu:

1. Antarmuka pengguna *(user interface)*

   Antarmuka pengguna merupakan mekanisme yang digunakan oleh pengguna dari sistem pakar untuk berkomunikasi. Antarmuka menerima informasi dari pemakai dan mengubahnya ke dalam bentuk yang dapat diterima oleh sistem. Selain itu antarmuka menerima dari sistem dan menyajikannya ke dalam bentuk yang dapat dimengerti oleh pemakai.ma

2. Basis pengetahuan

   Basis pengetahuan mengandung pengetahuan untuk pemahaman, formulasi, dan penyelesaian masalah. Komponen sistem pakar ini disusun atas 2 elemen dasar, yaitu:

a. Fakta : informasi tentang objek dalam area permasalahan tertentu.

b. Aturan : informasi tentang cara bagaimana memperoleh fakta baru dari fakta yang telah diketahui.

3. Akuisisi pengetahuan *(knowledge acquisition)*

Akuisisi pengetahuan adalah akumulasi, transfer, dan transformasi keahlian dalam menyelesaikan masalah dari sumber pengetahuan ke dalam program komputer. Dalam tahap ini *knowledge engineer* berusaha menyerap pengetahuan untuk selanjutnya ditransfer ke dalam basis pengetahuan. Pengetahuan diperoleh dari pakar, dilengkapi dengan buku, basis data, laporan penelitian dan pengalaman pemakai dengan cara-cara berikut:

a. Wawancara

Metode yang paling digunakan, yang melibatkan pembicaraan dengan pakar secara langsung dalam suatu wawancara.

b. Analisis protokol

Dalam metode ini pakar diminta untuk melakukan suatu pekerjaan dan mengungkapkan proses pemikirannya dengan menggunakan kata-kata. Pekerjaan tersebut direkam, dituliskan dan dianalisis.

c. Observasi pada pekerjaan pakar

Pekerjaan dalam bidang tertentu yang dilakukan pakar direkam dan diobservasi.

d. Induksi aturan dari contoh

Induksi adalah suatu proses penalaran dari khusus ke umum. Suatu sistem induksi aturan diberi contoh-contoh dari suatu masalah yang

hasilnya telah diketahui. Setelah diberikan beberapa contoh, sistem induksi aturan tersebut dapat membuat aturan yang benar untuk kasus-kasus contoh. Selanjutnya aturan dapat digunakan untuk menilai kasus lain yang hasilnya tidak diketahui.

4. Mesin/motor inferensi *(inference engine)*

   Elemen ini mengandung mekanisme pola pikir dan penalaran yang digunakan oleh pakar dalam menyelesaikan suatu masalah. Mesin inferensi adalah program komputer yang memberikan metodologi untuk penalaran tentang informasi yang ada dalam basis pengetahuan dan dalam *workplace*, dan untuk memformulasikan kesimpulan.

   Menurut Kusrini (2008: 8-11), ada 2 metode inferensi yang penting dalam sistem pakar yaitu:

   a. 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.

   b. Runut Balik *(Backward Chaining)*

      Runut balik merupakan metode penalaran kebalikan dari runut maju. Dalam metode runut balik penalaran dimulai dengan tujuan kemudian merunut balik ke jalur yang akan mengarahkan ke tujuan tersebut. Runut

balik merupakan cara yang efisien untuk memecahkan masalah yang dimodelkan sebagai masalah pemilihan terstruktur.

Kedua metode inferensi tersebut dipengaruhi 3 mcam penelusuran, yaitu (Rosnelly, 2012: 17):

a. *Depth-first search*, penelusuran kaidah dari simpul akar menurun ke tingkat dalam yang berurutan.
b. *Breadth-first seacrh*, bergerak dari simpul akar, simpul yang ada pada setiap tingkat diuji sebelum pindah ke tingkat selanjutnya.
c. *Best-first search*, bekerja berdasarkan kombinasi dari 2 metode sebelumnya.

5. *Workplace/Blackboard*

   Workplace merupakan area dari sekumpulan memori kerja (*working memory*), digunakan untuk merekam kejadian yang sedang berlangsung termasuk keputusan sementara. Ada 3 keputusan yang dapat direkam yaitu:

   a. Rencana : bagaimana menghadapi masalah
   b. Agenda : aksi-aksi potensial yang sedang menunggu untuk dieksekusi
   c. Solusi : calon aksi yang akan dibangkitkan

6. Fasilitas penjelasan

   Elemen tambahan yang akan meningkatkan kemampuan sistem pakar, digunakan untuk melacak respon dan memberikan penjelasan tentang kelakuan sistem pakar secara interaktif .

7. Perbaikan pengetahuan

   Pakar memiliki kemampuan untuk menganalisis dan meningkatkan kinerjanya serta kemampuan untuk belajar dari kinerjanya. Kemampuan tersebut adalah penting dalam pembelajaran terkomputerisasi, sehingga program akan mampu menganalisis penyebab kesuksesan dan kegagalan yang dialaminya dan juga mengevaluasi apakah pengetahuan-pengetahuan yang ada masih cocok untuk digunakan di masa mendatang.

Sistem pakar tidak terlepas dari elemen manusia yang terlibat di dalamnya. Menurut Rosnelly, (2012: 9-12) personel yang terkait dengan sistem pakar ada 4, yaitu:

1. Pakar *(expert)*

   Pakar yaitu seorang individu yang memiliki pengetahuan khusus, pemahaman, pengalaman dan metode-metode yang digunakan untuk memecahkan persoalan dalam bidang tertentu.

2. Pembangun pengetahuan *(knowledge engineer)*

   Pembangun pengetahuan bertugas menterjemahkan dan merepresentasikan pengetahuan yang diperoleh dari pakar, baik pengalaman pakar dalam menyelesaikan masalah maupun sumber terdokumentasi lainnya ke dalam bentuk yang bisa diterima oleh sistem pakar.

3. Pembangun sistem *(system enginer)*

   Pembangun sistem adalah orang yang bertugas merancang antarmuka sistem pakar, merancang pengetahuan yang udah diterjemahkan oleh pembangun

pengetahuan ke dalam bentuk yang sesuai dan dapat diterima oleh sistem pakar dan mengimplementasikannya ke dalam mesin inferensi.

4. Pemakai *(user)*

   Sistem pakar memungkinkan mempunyai beberapa kelas pengguna diantaranya, klien bukan pakar, pembangun sistem dan pakar itu sendiri.

Terdapat 11 area permasalahan yang bisa diselesaikan dengan menggunakan sistem pakar, sebagaimana yang dijelaskan oleh Husda (2012: 191-192) yaitu:

1. Interpretasi

   Interpretasi yaitu pengambilan keputusan dari hasil observasi, diantaranya: pengawasan, pengenalan ucapan, analisis citra, interpretasi sinyal dan beberapa analisis kecerdasan.

2. Prediksi

   Memprediksi akibat-akibat yang dimungkinkan dari situasi-situasi tertentu, diantaranya: peramalan, prediksi demografis, peramalan ekonomi, prediksi lalu lintas, estimasi hasil, militer, pemasaran atau peramalan keuangan.

3. Diagnosis

   Menetukan sebab malfungsi dalam situasi kompleks yang didasarkan pada gejala-gejala yang teramati, diantaranya: medis, elektronis, mekanis, dan diagnosis perangkat lunak.

4. Desain

   Menentukan konfigurasi komponen-komponen sistem yang cocok dengan tujuan-tujuan kinerja tertentu dan kendala-kendala tertentu, diantaranya: *layout* sirkuit dan perancangan bangunan.

5. Perencanaan

   Merencanakan serangkaian tindakan yang akan dapat mencapai sejumlah tujuan dengan kondisi awal tertentu, diantaranya: perencanaan keuangan, komunikasi, militer, pengembangan politik, *routing* dan manajemen proyek.

6. *Monitoring*

   Membandingkan tingkah laku suatu sistem yang teramati dengan tingkah laku yang diharapkan darinya, diantaranya: *Computer Aided Monitoring System*.

7. *Debugging* dan *repair*

   Menentukan dan mengimplementasikan cara-cara untuk mengatasi malfungsi, diantaranya memberikan resep obat terhadap suatu kegagalan.

8. Instruksi

   Melakukan instruksi untuk diagnosis, *debugging* dan perbaikan kinerja.

9. Kontrol

   Mengatur tingkah laku suatu *environment* yang kompleks seperti kontrol terhadap interpretasi, prediksi, perbaikan dan monitoring kelakuan sistem.

10. Seleksi

    Mengidentifikasi pilihan terbaik dari sekumpulan kemungkinan.

11. Simulasi

    Pemodelan interaksi antara komponen-komponen sistem.

Menurut Sutojo dalam Hayadi (2016: 2-3), sistem pakar menjadi populer karena memberikan banyak manfaat, yaitu:

1. Meningkatkan produktivitas karena sistem pakar dapat bekerja lebih cepat dari pada manusia.

2. Membuat seseorang yang awam bekerja seperti layaknya seorang pakar.
3. Meningkatkan kualitas dengan memberikan nasihat yang konsisten dan mengurangi kesalahan.
4. Mampu menangkap pengetahuan dan kepakaran seseorang.
5. Memudahkan akses pengetahuan seorang pakar.
6. Bisa digunakan sebagai media pelengkap dalam pelatihan.
7. Meningkatkan kemampuan untuk menyelesaikan masalah karena sistem pakar mengambil sumber pengetahuan dari banyak pakar.

Selain banyak manfaat yang diperoleh, sistem pakar juga memiliki kelemahan, yaitu (Hayadi, 2016: 3):

1. Biaya yang sangat mahal untuk membuat dan memeliharanya.
2. Sulit dikembangkan karena keterbatasan keahlian dan ketersediaan pakar.
3. Sistem pakar tidak 100% bernilai benar.

2.1.4.1 Representasi Pengetahuan

Representasi pengetahuan merupakan metode yang digunakan untuk mengkodekan pengetahuan ke dalam sebuah sistem pakar. Representasi dimaksudkan untuk menangkap sifat-sifat penting masalah dan membuat informasi itu dapat diakses oleh prosedur pemecahan masalah. Terdapat beberapa metode representasi pengetahuan yaitu: jaringan semantik, *frame*, logika predikat dan kaidah produksi (Kusrini, 2008: 6).

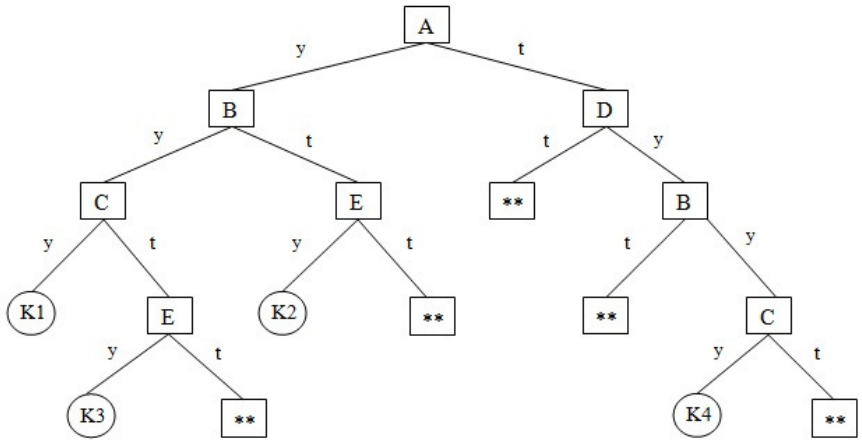
Sistem pakar pada penelitian ini menggunakan representasi pengetahuan model kaidah produksi. Sebelum sampai pada bentuk kaidah produksi,

pengetahuan yang berhasil didapatkan dari *domain* tertentu disajikan dalam bentuk tabel keputusan kemudian dibuat pohon keputusannya. Berikut ini adalah contoh penyajian dalam bentuk tabel keputusan dan pohon keputusan (Hartati dan Iswanti, 2008: 26-39).

Tabel 2.1 Tabel Keputusan

Konklusi / ***Premis***	***Konklusi 1***	***Konklusi 2***	***Konklusi 3***	***Konklusi 4***
Premis A	ya	ya	ya	tidak
Premis D	tidak	tidak	tidak	ya
Premis B	ya	tidak	ya	ya
Premis C	ya	tidak	tidak	ya
Premis E	tidak	ya	ya	tidak

Sumber: Hartati dan Iswanti (2008: 34)



Gambar 2.3 Pohon Keputusan
(Sumber: Hartati dan Iswanti, 2008: 35)

Keterangan:

A = *premis* A, K1 = *konklusi* 1, y = ya

B = *premis* B, K2 = *konklusi* 2, t = tidak

C = *premis* C, K3 = *konklusi* 3, ** = tidak menghasilkan *konklusi* tertentu

D = *premis* D, K4 = *konklusi* 4

Berdasarkan tabel 2.1 dan gambar 2.3 dapat diketahui bahwa *konklusi* K1 terpenuhi jika memenuhi *premis* A, B, dan C. *Konklusi* K2 terpenuhi jika memiliki *premis* A dan *premis* E. *Konklusi* K3 akan terpenuhi jika memiliki *premis* A, B, dan E. *Konklusi* K4 akan dihasilkan jika memenuhi *premis* B, C, dan D. Notasi "y" mengandung arti memenuhi *node* (*premis*) di atasnya, notasi "t" artinya tidak memenuhi.

Kusrini (2008: 6-7) menyatakan bahwa pengetahuan model kaidah produksi direpresentasikan dalam bentuk:

IF *[antecendent]* THEN *[konsekuen]*

IF *[kondisi]* THEN *[aksi]*

IF *[premis]* THEN *[konklusi]*

Contoh:

Aturan 1 : IF *terjadi luka* THEN *berikan Betadine*

Aturan 2 : IF *tidak punya uang cash* THEN *ambil tabungan*

Aturan 3 : IF *bersin-bersin* THEN *terserang influensa*

Aturan juga terkadang menggunakan operator logika AND atau OR, misalnya:

Aturan 4 : IF *dana mencukupi*
AND *pengiriman bisa dilakukan kurang dari 1 bulan*
THEN *beli laser printer*

Aturan 5 : IF *kontraktor tidak bisa menyelesaikan proyek tepat waktu*
OR *biaya melebihi anggaran*
THEN *kontrak batal*

Berdasarkan cara merepresentasikan pengetahuan model kaidah produksi seperti contoh di atas, yaitu bentuk IF *[premis]* THEN *[konklusi]*, maka dari pohon keputusan pada gambar 2.3 dapat dihasilkan aturan sebagai berikut:

Aturan 1 : IF *A* AND *B* AND *C* THEN *K1*

Aturan 2 : IF *A* AND *B* AND E THEN *K3*

Aturan 3 : IF *A* AND *E* THEN *K2*

Aturan 4 : IF *B* AND *C* AND *D* THEN *K4*

2.1.4.2 Metode Inferensi *Forward Chaining*

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

Sistem pakar pada penelitian ini menggunakan metode inferensi *forward chaining*. Dalam penelitiannya, Dahria (2011: 203) menyatakan bahwa *forward chaining* merupakan strategi pencarian yang memulai proses pencarian dari sekumpulan data atau fakta, kemudian dari data-data tersebut dicari suatu kesimpulan yang menjadi solusi dari permasalahan yang dihadapi. Mesin inferensi mencari kaidah-kaidah dalam basis pengetahuan yang premisnya sesuai dengan data-data tersebut, kemudian dari kaidah-kaidah tersebut diperoleh suatu kesimpulan. Metode inferensi *forward chaining* dipilih jika diinginkan sistem pakar menghasilkan konklusi dari data-data atau fakta-fakta yang sudah ada.

Untuk menjelaskan cara kerja dari sistem pakar metode *forward chaining*, diberikan ilustrasi sebagaimana yang dijelaskan oleh Kusrini (2008: 8-11), misalkan dikehendaki sebuah konklusi dari beberapa daftar konklusi yang ada berdasarkan premis-premis dalam aturan dan fakta yang diberikan oleh *user*. Berikut ini adalah aturan-aturannya:

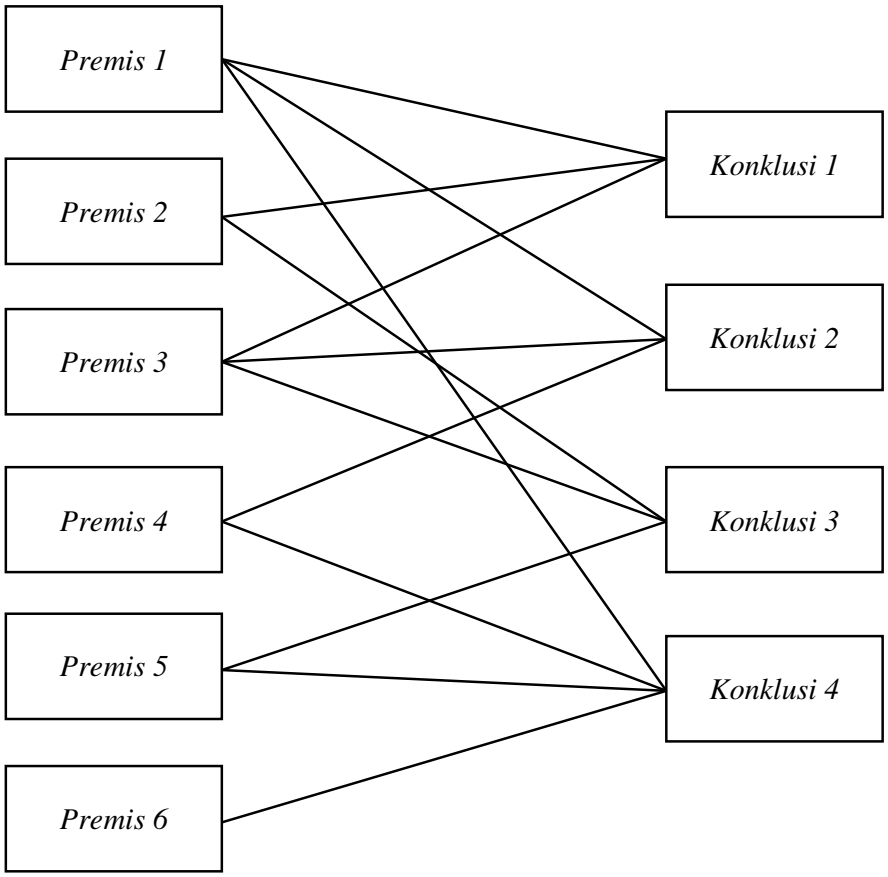
Aturan 1 : IF *Premis 1*
AND *Premis 2*
AND *Premis 3*
THEN *Konklusi 1*

Aturan 2 : IF *Premis 1*
AND *Premis 3*
AND *Premis 4*
THEN *Konklusi 2*

Aturan 3 : IF *Premis 2*
AND *Premis 3*
AND *Premis 5*
THEN *Konklusi 2*

Aturan 4 : IF *Premis 1*
AND *Premis 4*
AND *Premis 5*
AND *Premis 6*
THEN *Konklusi 4*

Jika aturan di atas digambarkan sebagai sebuah *graph* yang memetakan antara premis-premis dan konklusi-konklusi akan tampak seperti gambar 2.4. Penelusuran maju *(forward chaining)* pada kasus ini adalah untuk mengetahui apakah suatu fakta yang dialami oleh *user* itu termasuk konklusi 1, konklusi 2, konklusi 3 atau konklusi 4 atau bahkan bukan salah satu dari konklusi tersebut, yang artinya sistem belum mampu mengambil kesimpulan karena keterbasan aturan.



Gambar 2.4 *Graph* Pengetahuan
(Sumber: Kusrini, 2008: 10)

Dalam penalaran ini *user* diminta memasukkan premis-premis yang dialami. Untuk memudahkan *user*, sistem dapat memunculkan daftar premis yang mungkin dialami sehingga *user* dapat memberikan umpan balik premis mana yang dialami dengan memilih satu atau beberapa dari daftar permis yang tersedia. Dengan demikian daftar premisnya adalah:

Premis 1, Premis 2, Premis 3, Premis 4, Premis 5 dan *Premis 6*

Berdasarkan premis-premis yang dipilih, maka sistem akan mencari aturan yang sesuai sehingga akan diperoleh konklusinya.

Seandainya *user* memilih *Premis 1, Premis 2* dan *Premis 3* maka aturan yang sesuai adalah Aturan 1 dengan konklusinya adalah *Konklusi 1*. Seandainya *user* memilih *Premis 1* dan *Premis 6*, maka sistem akan mengarah pada Aturan 4 dengan konklusinya adalah *Konklusi 4*, tetapi karena aturan tersebut premisnya adalah *Premis 1*, *Premis 4, Premis 5* dan *Premis 6*, maka premis-premis yang dipilih oleh *user* tidak cukup untuk mengambil kesimpulan *Konklusi 4* sebagai konklusi yang terpilih.

2.1.5 Pengujian Sistem

Menurut A.S. dan Shalahuddin (2011: 210), pengujian adalah satu set aktivitas yang direncanakan dan sistematis untuk menguji atau mengevaluasi kebenaran yang diinginkan. Pengujian diperlukan tidak hanya untuk meminimalkan kesalahan secara teknis tetapi juga meminimalkan kesalahan non teknis (misalnya pengujian pesan kesalahan sehingga *user* tidak bingung dengan pesan kesalahan yang muncul).

Pengujian untuk validasi sistem memiliki beberapa pendekatan yaitu (A.S. dan Shalahuddin, 2011: 213-214):

1. *Black-Box Testing* (pengujian kotak hitam) yaitu menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Tujuannya untuk mengetahui apakah fungsi-fungsi, masukan, dan keluaran dari sistem atau perangkat lunak sesuai dengan spesifikasi yang dibutuhkan. Pengujian dilakukan dengan membuat kasus uji yang bersifat mencoba semua fungsi dengan menggunakan sistem atau perangkat lunak apakah sesuai dengan spesifikasi yang dibutuhkan.
2. *White-Box Testing* (pengujian kotak putih) yaitu menguji perangkat lunak dari segi desain dan kode program apakah mampu menghasilkan fungsi-fungsi, masukan, dan keluaran yang sesuai dengan spesifikasi yang dibutuhkan. Pengujian dilakukan dengan memeriksa logika dari kode program. Pembuatan kasus uji dapat mengikuti standar pengujian dari standar pemrograman yang ada.

2.2 Variabel Penelitian

Menurut Sugiyono (2014: 38) variabel penelitian merupakan segala sesuatu yang berbentuk apa saja yang dimiliki orang, objek atau kegiatan yang mempunyai variasi tertentu yang ditetapkan oleh peneliti untuk dipelajari sehingga diperoleh informasi tentang hal tersebut, kemudian ditarik kesimpulannya. Objek pada penelitian ini adalah *Integrated Circuit (IC)* dan

variabel penelitian yang ditetapkan yaitu kesalahan pada proses pengujian *Integrated Circuit (IC).*

2.2.1 *Integrated Circuit (IC)*

Integrated Circuit (IC) adalah komponen elektronika semikonduktor yang merupakan gabungan dari ratusan atau ribuan komponen-komponen lain. Bentuk IC berupa kepingan silikon padat, biasanya berwarna hitam yang mempunyai banyak kaki (pin). IC merupakan gabungan dari beberapa komponen seperti *resistor, kapasitor, dioda* dan *transistor* yang telah terintegrasi menjadi sebuah rangkaian berbentuk *chip* kecil. IC digunakan untuk beberapa keperluan pembuatan peralatan elektronik agar mudah dirangkai menjadi peralatan yang berukuran relatif kecil (Hakiem, 2015: 23).

2.2.1.1 Sejarah IC

Menurut Hakiem (2015: 22), IC ditemukan pada tahun 1958 oleh seorang insinyur bernama Jack Kilby yang bekerja pada Texas Instrument yang pada waktu itu sedang mencoba memecahkan masalah dengan memikirkan sebuah konsep menggabungkan seluruh komponen elektronika dalam satu blok yang dibuat dari bahan semikonduktor. Penemuan itu kemudian dinamakan *Integrated circuit (IC).* Beberapa saat setelah itu, Robert Noyce yang bekerja pada Fairchild Semiconductor Corporation menemukan hal serupa, meskipun mereka bekerja pada 2 tempat yang berbeda.

Semenjak saat itu banyak riset yang dilakukan untuk mengembangkan IC. Pada tahun 1965, seorang pendiri Intel bernaa Gorden Moore memperkirakan bahwa jumlah transistor yang terdapat dalam sebuah IC akan bertambah 2 kali lipat setiap 18 bulan sekali. Kecenderungan peningkatan jumlah tranistor ini telah terbukti setelah sekian lama dan diperkirakan akan terus berlanjut. Hal ini dapat dilihat pada perkembangan IC, sebuah 64-MBit DRAM yang pertama kali dipasarkan pada tahun 1994 terdiri dari 3 juta transistor, bahkan mikropresessor Intel Pentium 4 terdiri lebih dari 42 juta transistor (Hakiem, 2015: 22).

2.2.1.2 Jenis-jenis IC

Berdasarkan jumlah komponen atau transistor yang terkandung di dalamnya, IC dapat dikelompokkan menjadi beberapa jenis yaitu sebagai berikut (Hakiem, 2015: 22):

1. *Small Scale Integration (SSI)*, yaitu IC dengan maksimum 100 komponen elektronik.
2. *Medium Scale Integration (MSI)*, yaitu IC dengan jumlah komponen elektronik antara 100 sampai dengan 3.000 komponen.
3. *Large Scale Integration (LSI)*, yaitu IC dengan dengan jumlah komponen elektronik antara 3.000 sampai dengan 100.000 komponen.
4. *Very Large Scale Integration (VLSI)*, yaitu IC dengan dengan jumlah komponen elektronik antara 100.000 sampai dengan 1.000.000 komponen.
5. *Ultra Large Scale Integration (ULSI)*, yaitu IC dengan jumlah komponen elektronik lebih dari 1.000.000 komponen.

Berdasarkan teknologi yang digunakan, IC dapat dibagi menjadi 2 jenis yaitu (Hakiem, 2015: 24):

1. IC *Transistor-transistor Logic (TTL)*

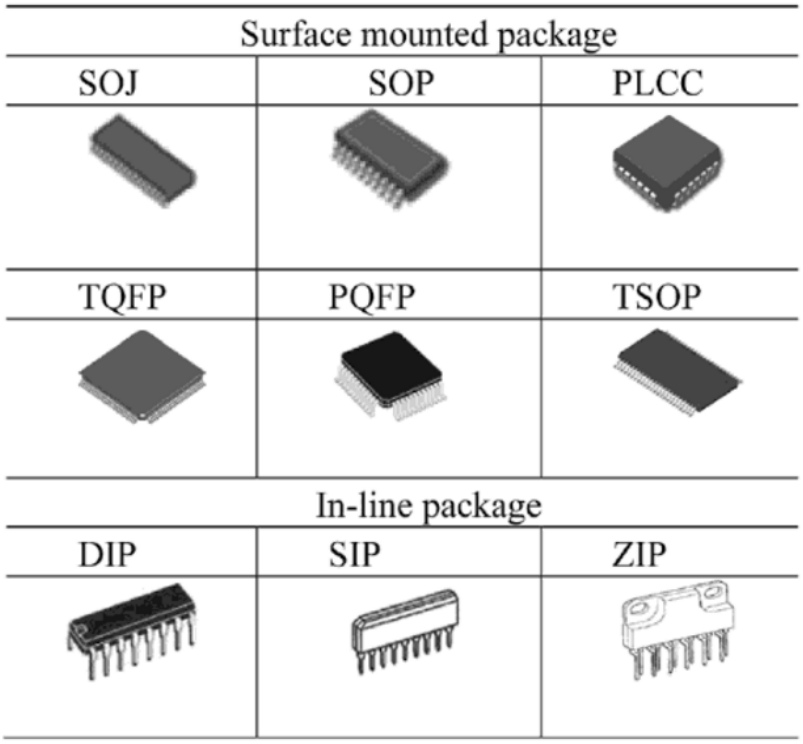
   IC TTL adalah IC yang banyak digunakan dalam rangkaian digital karena menggunakan sumber tegangan yang relatif rendah yaitu antara 4,75 volt sampai 5,25 volt. IC TTL dibangun dengan menggunakan transistor sebagai komponen utamanya dan dipergunakan untuk berbagai variasi logika sehingga dinamakan *Transistor-transistor Logic (TTL)*. Pada IC TTL, beberapa transistor digabungkan sehingga membentuk 2 keadaan yaitu ON dan OFF dan dengan mengendalikan kondisi ON/OFF transistor ini dapat dibuat berbagai fungsi logika seperti fungsi AND, OR dan NOT.

2. IC *Complementary Metal Oxide Semiconductor (CMOS)*

   Sebenarnya antara IC TTL dan IC CMOS memiliki pengertian yang sama, perbedaannya adalah IC CMOS memerlukan daya yang sangat rendah dan memungkinkan pemilihan tegangan sumber yang jauh lebih lebar yaitu antara 3 volt sampai 15 volt. Kelemahan IC CMOS diantaranya bisa rusak apabila terkena listrik elektrostatis dan harganya lebih mahal.

Kemasan IC berfungsi untuk melindungi komponen elektronik yang ada di dalamnya. Dalam penelitiannya, Wang, Jiang dan Liu (2013: 280-281) menjelaskan bahwa secara garis besar kemasan IC dibagi menjadi 2 jenis yaitu *surface mounted package* dan *in-line package*. IC yang termasuk ke dalam kelompok *surface mounted package* diantaranya adalah *Small Outline J-leaded (SOJ)*, *Small Outline Package (SOP)*, *Plastic Leaded Chip Carrier (PLCC)* dan

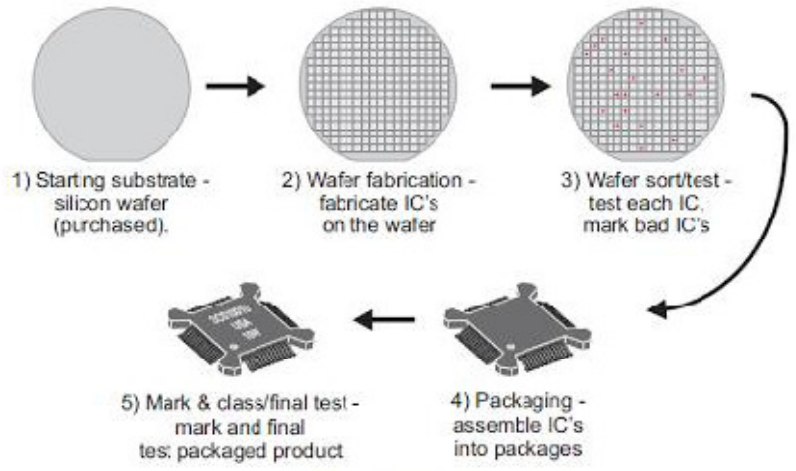
Thin Quad Flat Package (TQFP). Selanjutnya IC yang termasuk ke dalam kelompok *in-line package* diantaranya adalah *Single In-line Package (SIP)*, *Dual In-line Package (DIP)* dan *Zig-zag In-line Package (ZIP)*. Contoh dari kemasan IC dapat dilihat pada gambar 2.5 berikut.



Gambar 2.5 Kemasan IC
(Sumber: Wang, Jiang dan Liu, 2013: 280-281)

2.2.1.3 Proses Pembuatan IC

Menurut Hakiem (2015: 31-40), secara garis besar proses pembuatan IC terdiri dari 5 tahapan seperti pada gambar 2.6 di bawah ini.



Gambar 2.6 Proses Pembuatan IC
(Sumber: Hakiem, 2015: 31)

Berikut ini adalah penjelasan dari gambar di atas:

1. *Substrate silicon wafer*

   Pada tahap ini perusahaan pembuat IC membeli *wafer* silikon dari pihak ketiga *(supplier).*

2. *Wafer fabrication*

   Wafer fabrication adalah proses pembuatan rangkaian IC sesuai dengan desain yang sudah direncanakan, hasilnya adalah sejumlah *die* yang masih menyatu pada kepingan *wafer* silikon.

3. *Wafer sort*

   Wafer sort adalah proses pengetesan yang dilakukan pada *die-die* yang dihasilkan dari proses *wafer fabrication.*

4. *Packaging*

   Packaging adalah proses pengemasan untuk melindungi *die* sehingga memudahkan penanganan serta penyambungan dengan komponen elektronika lainnya. Beberapa proses pada tahapan *packaging* ini diantaranya:

   a. Pemotongan *die*.
   b. Pemasangan *die* pada *leadframe* yang akan berfungsi sebagai kaki-kaki IC.
   c. *Wire bonding*, yaitu proses penyambungan kaki-kaki *leadframe* ke *bond pad* yang ada pada *die*.
   d. *Molding*, yaitu menutup *leadframe* menggunakan *compound* pada suhu dan tekanan udara tertentu.
   e. *Solder plating*, yaitu proses penyepuhan kaki-kaki IC dengan timah sehingga kaki-kaki IC berwarna perak.
   f. *Marking*, yaitu proses pemberian label IC.

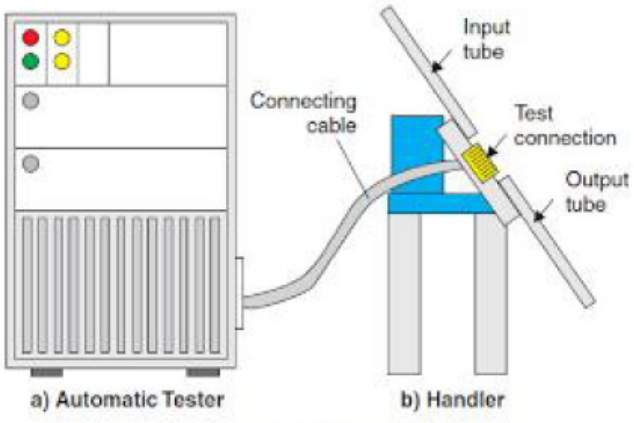
5. *Final Test*

   Final test adalah proses pengujian akhir terhadap semua IC yang telah dikemas dengan tujuan agar IC yang mengalami kerusakan selama proses *packaging* tidak ikut terkirim bersama-sama IC yang bagus.

2.2.1.4 Proses Pengujian IC

Selama proses pembuatan IC dari mulai berupa *die* sampai proses *packaging*, sangat memungkinkan terdapat unit yang mengalami kerusakan. Oleh karena itu proses pengujian akhir *(final test)* dilakukan terhadap semua IC agar IC

yang mengalami kerusakan selama proses produksi dapat dipisahkan dari IC yang bagus. Pengujian IC dilakukan menggunakan mesin *tester* dan *handler* seperti ilustrasi pada gambar 2.7. *Handler* sepenuhnya dikendalikan oleh *tester* untuk melakukan pengujian berbagai sifat kelistrikan pada setiap IC sekaligus memilah-milah kualitas dari masing-maing IC (Hakiem, 2015: 40).



Gambar 2.7 Ilustrasi Proses Pengujian IC
(Sumber: Hakiem, 2015: 40)

Menurut Utama (2015: 3), beberapa elemen yang diperlukan pada proses pengujian IC adalah:

1. Mesin *tester*

   Dalam penelitiannya, Satish, dkk. (2013: 942) menyatakan bahwa *Automatic Test Equipment (ATE)* atau mesin *tester* adalah sebuah sistem yang terdiri dari banyak instrumen pengujian, serta memiliki kemampuan secara otomatis untuk menguji dan mendiagnosis kesalahan pada IC.

Gambar 2.8 Mesin T*ester*
(Sumber: http://techon.nikkeibp.co.jp/article/NEWS/20051208/111460/J750.jpg)

2. *Test Handler*

Fungsi dari *test handler* adalah untuk memindahkan unit ke bagian *test site* dan menyortirnya sesuai dengan hasil pengetesan apakah *pass* atau *fail* (Golio, 2008: 10-6).

Gambar 2.9 *Test Handler*
(Sumber: http://www.cohu.com/assets/images/rasco-so1000-v2.jpg)

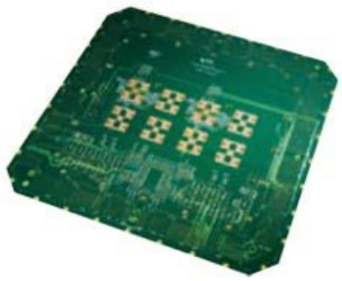
3. Set Unit Korelasi

Set unit korelasi adalah sekelompok *known good* dan *reject* unit (unit yang telah diketahui sebagai unit bagus dan unit *reject*) yang digunakan untuk

proses tes korelasi. Unit korelasi disediakan oleh *test engineer* (Yuliarta, 2015: 3). Standar 1 set unit korelasi terdiri dari 3 *good* unit dan 2 *reject* unit.

4. *Loadboard*

   Dalam penelitiannya, Ching (2015: 103) menyatakan bahwa *loadboard* adalah *Printed Circuit Board (PCB)* elektronik yang digunakan untuk menghubungkan antara *tester* dan *handler*.



Gambar 2.10 *Loadboard*
(Sumber: http://www.dynamic-test.com/images/p1.jpg)

5. *Test Program*

   Test program berfungsi untuk mengendalikan perangkat pengujian (mesin *tester* dan *loadboard*). *Test program* mengendalikan *tester* sehingga menampilkan hasil pengujian IC apakah *pass* atau *fail*.

6. Perlengkapan Anti *ESD (Electrostatic Discharge)*

   Perlengkapan anti ESD berfungsi untuk melindungi IC dari muatan listrik statis yang berasal dari tubuh manusia. Beberapa perlengkapan anti ESD diantaranya: *Wrist Strap* dan *Finger Cots.*

Gambar 2.11 *Wrist Strap*
(Sumber: http://www.zeph.com/esdwriststrapcoilcordplug.jpg)

Gambar 2.12 *Finger Cots*
(Sumber: https://www.mdsassociates.com/content/images/gloves_cleanroom/96-7C_A.jpg)

2.2.1.5 Proses Tes Korelasi

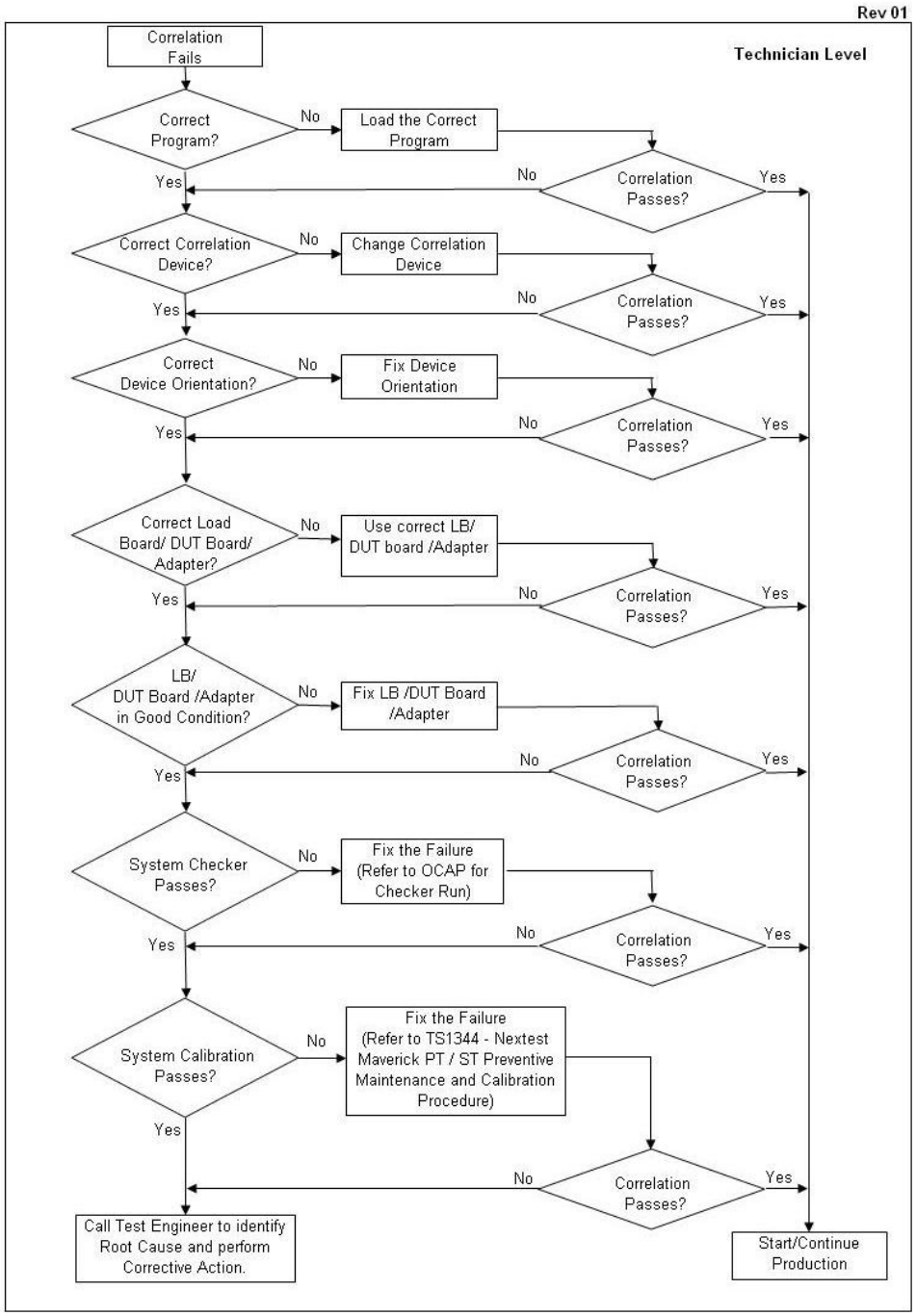
Tes korelasi adalah suatu proses yang digunakan untuk memastikan integritas (kelayakan) dari *set-up* antara mesin *tester* dengan *loadboard*. Aktivitas tes korelasi dilakukan pada situasi tertentu dan hasilnya harus sesuai dengan *binning (pass/fail)* yang telah ditentukan oleh *test engineer* (Yuliarta, 2015: 3). Selanjutnya Yuliarta (2015: 4) juga menyatakan bahwa tes korelasi harus dilakukan pada situasi-situasi berikut:

1. Setiap *set-up* yang memakai *test program* yang berbeda dengan *set-up* sebelumnya.
2. Sebelum melanjutkan produksi setelah ditemukan gangguan sistem *(system breakdown)* yang serius dan berpotensi mengakibatkan unit *reject* menjadi *good* atau sebaliknya.
3. Korelasi harus dijalankan sebelum produksi *final test* dan dilakukan per *lot*.

Apabila proses tes korelasi gagal memenuhi kondisi *binning pass/fail* yang ditentukan oleh *test engineer*, maka personel teknisi harus memperbaikinya (Yuliarta, 2015: 4). Selanjutnya menurut Titis (2016: 16), prosedur perbaikan tersebut adalah seperti ditunjukkan pada gambar 2.13.

Berikut adalah penjelasan langkah-langkah perbaikan oleh teknisi berdasarkan gambar 2.13:

1. Memastikan *test* program yang digunakan adalah benar.
2. Memastikan unit korelasi sesuai dengan *device name* yang akan diproses.
3. Memastikan orientasi unit benar (tidak terbalik).
4. Memastikan *loadboard* yang digunakan sudah tepat.
5. Memastikan *loadboard* dalam kondisi baik (tidak ada kerusakan fisik).
6. Memastikan *tester* dalam kondisi baik dengan melakukan *system checker*.
7. Melakukan kalibrasi pada *tester*.
8. Memberikan laporan kepada *test engineer* apabila semua langkah perbaikan di atas tidak dapat mengatasi masalah pada tes korelasi.



Gambar 2.13 Prosedur Untuk Mengatasi Masalah Tes Korelasi
(Sumber: Titis, 2016: 16)

2.2.1.6 IC Mikroprosesor dan Mikrokontroler Zilog

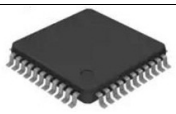
Mikroprosesor adalah suatu *chip* (rangkaian terintegrasi yang sangat kompleks) yang berfungsi sebagai pemroses data dari input yang diterima pada suatu sistem digital. Mikroprosesor membutuhkan komponen eksternal tambahan seperti memori, pengolah analog ke digital dan perangkat komunikasi serial. Oleh karena itu dikembangkanlah chip yang di dalam kemasan tersebut sudah terdapat mikroprosesor, I/O pendukung, memori bahkan ADC yang dikenal dengan istilah mikrokontroler (Budiharto, 2007: 19-20). Zilog adalah salah satu perusahaan semikonduktor yang mengembangkan IC mikroprosesor dan mikrokontroler. Proses perakitan *(assembly)* dan pengujian *(final test)* IC mikroprosesor dan mikrokontroler Zilog dilakukan oleh PT Unisem sebagai perusahaan sub kontraktor.

IC mikroprosesor dan mikrokontroler Zilog sangat beragam dan masing-masing IC memiliki *device name* yang unik, contohnya Z84C0006PEG-HB. Dari situs *web* Digi-Key (www.digikey.com), yaitu sebuah perusahaan distributor komponen elektronik yang berkedudukan di Amerika Serikat, didapatkan informasi bahwa Z84C0006PEG adalah IC mikroprosesor Z80 dengan kecepatan 6 MHz, kemasan 40 pin *Plastic Dual In-line Package (PDIP).*

Gambar 2.14 IC Mikroprosesor Zilog Z80
(Sumber: http://media.digikey.com/photos/Zilog%20Photos/269-40-DIP_sml.jpg)

Pada penelitian ini ditetapkan sebanyak 10 jenis IC mikroprsesor dan mikrokontroler Zilog sebagai objek penelitian yang disajikan pada tabel 2.2.

Tabel 2.2 Daftar IC Sebagai Objek Penelitian

No	Nama Produk / Deskripsi	Kecepatan	Kemasan	Gambar Kemasan
1	Z88C0020PSG-CC / Z8 *Microcontroller* IC	20 MHz	PDIP	
2	Z16C3220VSG00TR-CC /*Universal Serial Controller*	20 MHz	PLCC	
3	Z84C1506FEG-KA / Z80 *Microprocessor* IC	6 MHz	PQFP	
4	Z8523L10VEG-HB /*Communications Controller*	10 MHz	PLCC	
5	Z86C3010VEG-KC / Z8 *Microcontroller* IC	10 MHz	PLCC	
6	Z84C0006AEG-AN / Z80 *Microprocessor* IC	6 MHz	LQFP	
7	Z86E83VEG-HA / *Microcontroller* IC	10 MHz	PLCC	
8	Z84C0006VEG-HB / Z80 *Microprocessor* IC	6 MHz	PLCC	
9	Z86E3016SSG-DG / Z8 *Microcontroller* IC	16 MHz	SOIC	

Tabel 2.2 Lanjutan

No	Nama Produk / Deskripsi	Kecepatan	Kemasan	Gambar Kemasan
10	Z86E6316VSG-AN / Z8 *Microcontroller* IC	16 MHz	PLCC	

Sumber: Data Penelitian (2016)

2.2.2 Kesalahan *(Failure)* Pada Proses Pengujian IC

Berdasarkan dokumen spesifikasinya, masing-masing IC mikroprosesor dan mikrokontroler Zilog memiliki pengelompokan *(binning)* untuk memisahkan IC sesuai dengan hasil pengujiannya. Unit-unit yang *reject* dikelompokkan berdasarkan kategori *reject*-nya. Contohnya *Bin-6* adalah kategori untuk IC yang mengalami kesalahan fungsional *(functional fail)*, *Bin-7* adalah kategori untuk IC yang mengalami kesalahan pada tegangan atau arus DC *(DC fail)* dan sebagainya. Secara garis besar, kategori *reject* untuk IC mikroprosesor dan mikrokontroler Zilog disajikan pada tabel 2.3.

Tabel 2.3 Kategori *Reject* Pada IC Mikroprosesor/Mikrokontroler Zilog

Fail Category	***Fail Name***
Bin-2	*ROM fail*
Bin-3	*VBO fail*
Bin-4	*ICC fail*
Bin-5	*Gross fail*
Bin-6	*Functional fail*
Bin-7	*DC fail*
Bin-8	*Continuity fail*

Sumber: Data Penelitian (2016)

Menurut Titis (2016: 5), hasil pengujian/pengetesan IC dapat dilihat pada layar komputer mesin *tester*. Tampilan *Bin-1* dalam kotak berwarna hijau

menunjukkan hasil unit yang *good*, sedangkan tampilan *bin* yang lain dalam kotak berwarna merah menunjukkan hasil unit yang *reject* seperti ditunjukkan pada gambar 2.15.

(a) (b)

Gambar 2.15 Tampilan Hasil Pengujian IC yang *Good* (a) dan *Reject* (b)
(Sumber: *Standard Operating Procedure (SOP)*, 2016)

Berdasarkan hasil wawancara dengan *test engineer* pada PT Unisem, beberapa penyebab terjadinya kesalahan *(failure)* pada proses pengujian IC terutama pada saat tes korelasi diantaranya adalah kesalahan yang disebabkan oleh *set-up hardware* yang tidak berfungsi dengan baik, misalnya komponen elektronik pada *loadboard* yang rusak, kontaktor yang sudah rusak dan sebagainya. Selain itu bisa juga *failure* terjadi karena kesalahan pada *test program* sehingga *test program* perlu dimodifikasi.

Beberapa penelitian telah dilakukan untuk meminimalisir terjadinya kesalahan pada pengujian IC (mencegah unit *reject*). Dalam penelitian Khade dan Gourkar (2013: 73), unit *reject* dapat diminimalisir dengan cara menambahkan sebuah metode yang disebut *Signature Analysis Method* dalam proses pengujian IC CMOS. Selanjutnya berdasarkan penelitian Idris, dkk. (2014: 91-95), untuk

menguji IC yang memiliki ukuran yang kecil lebih cocok menggunakan kontaktor jenis *Micro Contact* sehingga meminimalisir unit yang *reject*, sedangkan dalam penelitian lainnya, Hashizume, dkk. (2015: 202-207) telah menciptakan sebuah rangkaian tambahan untuk mendeteksi adanya sirkuit yang tidak terhubung *(open defect)* pada proses pengujian IC.

2.3 Perangkat Lunak Pendukung

Perangkat lunak pendukung digunakan sebagai alat bantu dalam membuat sistem pakar dalam penelitian ini. Perangkat lunak tersebut terbagi menjadi 5 kategori yaitu: *Unified Model Language (UML)*, *flowchart*, bahasa pemrograman *web* (HTML, CSS, PHP dan JavaScript), perangkat lunak aplikasi (StarUML, XAMPP, phpMyAdmin, Sublime Text dan Mozilla Firefox) dan sistem manajemen basis data (MySQL).

2.3.1 *Unified Modeling Language (UML)*

Unified Model Language (UML) adalah bahasa pemodelan untuk sistem atau perangkat lunak yang berparadigma berorientasi objek. Pemodelan *(modeling)* digunakan untuk penyederhanaan masalah-masalah yang kompleks sedemikian rupa sehingga lebih mudah dipelajari dan dipahami. Tujuan dari pemodelan perangkat lunak adalah sebagai sarana analisis, pemahaman, visualisasi dan komunikasi antar anggota tim yang beranggotakan beberapa orang (Nugroho, 2010: 6-7).

Menurut A.S. dan Shalahuddin (2011: 120-121), terdapat 13 macam diagram dalam UML yang dibagi menjadi 3 kategori yaitu:

1. *Structure Diagrams*

   Kategori ini terdiri dari kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan. Diagram UML yang termasuk dalam kategori ini antara lain *class diagram, object diagram, component diagram, composite structure diagram, package diagram,* dan *deployment diagram*.

2. *Behaviour Diagrams*

   Kategori ini terdiri dari kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem. Diagram UML yang termasuk dalam kategori ini antara lain *use case diagram, acitivity diagram,* dan *state machine diagram*.

3. *Interaction Diagrams*

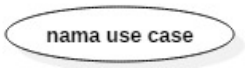
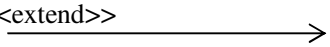
   Kategori ini terdiri dari kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antar subsistem pada suatu sistem. Diagram UML yang termasuk dalam kategori ini antara lain *sequence diagram, communication diagram, timing diagram,* dan *interaction overview diagram*.

Dalam penelitian ini, digunakan 4 jenis UML untuk memodelkan sistem pakar yaitu:

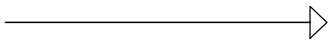
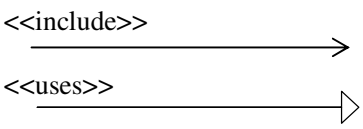
1. *Use Case Diagram*

   Use case diagram mendeskripsikan sebuah interaksi antara satu sistem atau lebih aktor dengan sistem informasi yang akan dibuat. *Use case diagram* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi itu (A.S. dan Shalahuddin, 2011: 130). Simbol-simbol yang digunakan dalam *use case diagram* disajikan pada tabel 2.4.

Tabel 2.4 Simbol Pada *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 menggunakan kata kerja 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. Aktor belum tentu merupakan orang, biasanya dinyatakan menggunakan kata benda di awal frase nama aktor
asosiasi / *association* 	Komunikasi antara aktor dan *use case* yang berpartisipasi 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 walaupun tanpa *use case* tambahan itu. Arah panah mengarah pada *use case* yang ditambahkan.

Tabel 2.4 Lanjutan

Simbol	Deskripsi
generalisasi / *generalization* 	Hubungan generalisasi dan spesialisasi (umum – khusus) antara 2 buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari fungsi lainnya. Arah panah mengarah pada *use case* yang menjadi generalisasinya (umum)
menggunakan / *include*/*uses* 	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankannya *use case* ini.

Sumber: A.S. dan Shalahuddin (2011: 131-133)

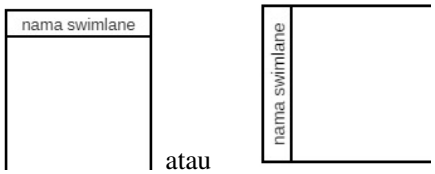
2. *Activity diagram*

Activity diagram merupakan diagram yang menggambarkan aliran kerja *(workflow)* atau aktifitas dari sebuah sistem atau proses. *Activity diagram* menggambarkan aktifitas sistem bukan apa yang dilakukan oleh aktor, tetapi aktivitas yang dapat dilakukan oleh sistem (A.S. dan Shalahuddin: 2011: 134). Simbol-simbol yang digunakan dalam *activity diagram* ditampilkan dalam tabel 2.5.

Tabel 2.5 Simbol Pada *Activity Diagram*

Simbol	Deskripsi
status awal 	Status awal aktivitas sistem, sebuah diagram aktifitas memiliki sebuah status awal
aktifitas 	Aktifitas yang dilakukan sistem, aktifitas biasanya diawali dengan kata kerja
percabangan / *decision* 	Asosiasi percabangan dimana jika ada pilihan aktifitas lebih dari satu

Tabel 2.5 Lanjutan

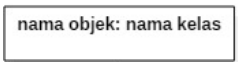
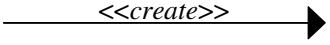
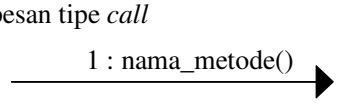
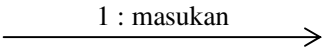
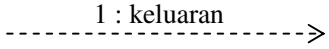
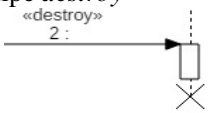
Simbol	Deskripsi
penggabungan / *join*	Asosiasi penggabungan dimana lebih dari satu aktifitas digabungkan menjadi satu
status akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktifitas memiliki sebuah status akhir
swimlane 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktifitas yang terjadi

Sumber: A.S. dan Shalahuddin (2011: 134-135)

3. *Sequence diagram*

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup *(life cycle)* objek dan pesan *(message)* yang dikirimkan dan diterima antar objek. Banyaknya *sequence diagram* yang harus digambar adalah sebanyak pendefinisian *use case* yang memiliki proses sendiri. Semakin banyak *use case* yang didefinisikan semakin banyak pula *sequence diagram* yang harus dibuat (A.S dan Shalahuddin, 2011: 137-138). Simbol-simbol yang digunakan pada *sequence diagram* ditampilkan dalam tabel 2.6.

Tabel 2.6 Simbol Pada *Sequence Diagram*

Simbol	Deskripsi
aktor/*actor*  	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri. Aktor belum tentu merupakan orang, biasanya dinyatakan menggunakan kata benda di awal frase nama aktor
garis hidup/*lifeline* 	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor
objek  	Menyatakan objek yang berinteraksi pesan
waktu aktif 	Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan di dalamnya. Aktor tidak memiliki waktu aktif
pesan tipe *create*  	Menyatakan suatu objek membuat objek yang lain. Arah panah mengarah pada objek yang dibuat
pesan tipe *call*  	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri. Arah panah mengarah pada objek yang memiliki operasi/metode.
pesan tipe *send*  	Menyatakan bahwa suatu objek mengirimkan data/masukan/informasi ke objek lainnya. Arah panah mengarah pada objek yang dituju
pesan tipe *return*  	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu. Arah panah mengarah pada objek penerima
pesan tipe *destroy*  	Menyatakan suatu objek mengakhiri hidup objek lain. Arah panah mengarah pada objek yang diakhiri

Sumber: A. S. dan Shalahuddin (2011: 138-139)

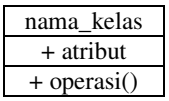
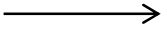
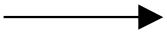
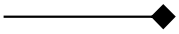
4. *Class diagram*

Class diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki atribut dan metode. Atribut yaitu variabel-variabel yang dimiliki oleh suatu kelas sedangkan 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. Susunan struktur kelas yang baik pada diagram kelas sebaiknya memiliki jenis-jenis kelas berikut (A.S dan Shalahuddin, 2011: 122):

a. Kelas *main*, yaitu kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.
b. Kelas yang menangani tampilan sistem, yaitu kelas yang mendefinisikan dan mengatur tampilan ke pemakai.
c. Kelas yang diambil dari pendefinisian *use case*, yaitu kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisian *use case*.
d. Kelas yang diambil dari pendefinisian data, yaitu kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke *database*.

Simbol-simbol yang digunakan pada *class diagram* ditampilkan dalam tabel 2.7 berikut (A.S dan Shalahuddin, 2011: 123-124).

Tabel 2.7 Simbol Pada *Class Diagram*

Simbol	Deskripsi
kelas 	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 ini biasanya juga disertai dengan *multiplicity*
generalisasi 	Relasi antar kelas dengan makna generalisasi-spesealisasi (umum-khusus)
kebergantungan / *dependency* 	Relasi antar kelas dengan makna kebergantungan antar kelas
agregasi / aggregation 	Relasi antar kelas dengan makna semua bagian *(whole part)*

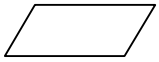
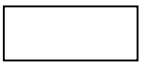
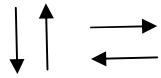
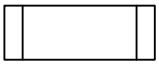
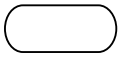
Sumber: A. S. dan Shalahuddin (2011: 123-124)

2.3.2 *Flowchart*

Flowchart adalah bagan *(chart)* yang menunjukkan aliran *(flow)* di dalam program atau prosedur sistem secara logika, digunakan terutama sebagai alat bantu komunikasi dan untuk dokumentasi (Kusrini dan Koniyo, 2007: 80). Salah satu jenis *flowchart* adalah program *flowchart*. Menurut Kusrini dan Koniyo (2007: 83), program *flowchart* yaitu bagan yang menjelaskan secara rinci

langkah-langkah proses program. Simbol-simbol yang terdapat pada program *flowchart* disajikan pada tabel 2.8 Berikut.

Tabel 2.8 Simbol Pada Program *Flowchart*

Simbol	Deskripsi
input / output 	Digunakan untuk mewakili data input / output
proses 	Digunakan untuk mewakili suatu proses
garis alir 	Menunjukan arus dari proses
keputusan 	Digunakan untuk seleksi suatu kondisi di dalam program
penghubung 	Menunjukan penghubung ke halaman yang sama atau halaman lain
proses terdefinisi 	Menunjukan suatu operasi yang rinciannya ditunjukkan di tempat lain
persiapan 	Digunakan untuk memberi nilai awal suatu besaran
terminal 	Menunjukan awal dan akhir dari suatu proses

Sumber: Kusrini dan Koniyo (2007: 84)

2.3.3 Bahasa Pemrograman *Web*

Menurut Simarmata (2010: 47-49), *web* adalah sebuah sistem dengan informasi yang disajikan dalam bentuk teks, gambar, suara dan lain-lain yang

tersimpan dalam sebuah *web server* yang disajikan dalam bentuk *hypertext*. *Web* dapat diakses oleh pengguna dengan menggunakan aplikasi yang disebut *web browser*. Fungsi *web browser* adalah membaca halaman-halaman *web* yang tersimpan dalam *web server* melalui protokol yang disebut *HTTP (Hypertext Transfer Protokol*.

Sebuah halaman *web* dibuat dengan menggunakan bahasa pemrograman *web*. Menurut Saputra (2012: 2-4), setidaknya terdapat 12 bahasa pemrograman yang dapat digunakan untuk membuat *web* yaitu: HTML, PHP, ASP, XML, WML, PERL, CFM, JavaScript, CSS, JSP, Ruby dan Phyton. Sistem pakar berbasis *web* dalam penelitian ini menggunakan bahasa pemrograman PHP untuk mengolah dan memproses data.

2.3.3.1 HTML *(Hyper Text Markup Language)*

Gambar 2.16 Logo HTML5
(Sumber: https://www.w3.org/html/logo/)

HTML adalah singkatan dari *Hyper Text Markup Language* yang merupakan suatu kode semi pemrograman yang menjadi dasar terwujudnya *web*. Kode-kode yang digunakan dalam HTML disebut dengan *tag*. HTML diciptakan pada tahun 1989 oleh Tim Barners Lee, seorang staf ahli dari CERN, yaitu

organisasi penelitian yang berlokasi di Jenewa, Swiss (Yuhefizar, Mooduto dan Hidayat, 2009: 1).

Menurut Prasetio (2012: 94-97), sebuah *file* HTML merupakan *file* teks yang berisi *tag-tag mark up*. *Tag mark up* memberitahukan *web browser* bagaimana harus menampilkan sebuah halaman. *File* HTML harus memiliki ekstensi *htm* atau *html*. *File* HTML dapat dibuat menggunakan teks editor biasa. Berikut ini adalah contoh dari isi sebuah *file* HTML:

```
<html>
        <head>
                <title>Judul halaman</title>
        </head>
        <body>
                Ini halaman pertama saya. <b>Teks ini ditebalkan</b>
        </body>
</html>
```

Penjelasan dari kode diatas:

1. Setiap *tag* diapit oleh tanda lebih kecil "<" dan lebih besar ">".
2. Tag <html> memberitahu *web browser* bahwa inilah awal dari dokumen HTML. *Tag* pasangannya yaitu </html> menyatakan bahwa inilah akhir dari dokumen HTML.
3. Teks diantara *tag* <head> dan </head> adalah teks informasi *header*. Informasi *header* ini tidak ditampilkan pada jendela *web browser*.
4. Teks diantara *tag* <title> dan </title> adalah judul dokumen. Judul ini akan ditampilkan di bagian paling atas pada *web browser*.
5. Teks diatara *tag* <body> dan </body> adalah teks yang akan ditampilkan pada *web browser*.
6. Teks diantara *tag* dan akan ditampilkan dalam huruf tebal.

2.3.3.2 CSS *(Cascading Style Sheet)* dan Bootstrap

CSS *(Cascading Style Sheet)* merupakan bahasa pemrograman *web* yang digunakan untuk mengendalikan dan membangun berbagai komponen dalam *web* sehingga tampilan *web* lebih rapi, terstruktur dan seragam. CSS berfungsi sebagai penopang atau pendukung dari *file* HTML yang berperan dalam penataan kerangka dan *layout* (Saputra, 2012: 27-28). Menurut Olsson (2008: 2-3), keuntungan utama menggunakan CSS adalah bahwa file CSS dapat dibuat terpisah isi dari *web* sehingga memungkinkan untuk mengatur banyak halaman *web* sekaligus hanya dengan menggunakan sebuah file CSS.

Gambar 2.17 Logo CSS 3
(Sumber: http://jaspreetchahal.org/images/css3.svg)

Keuntungan lain yang didapatkan dengan menggunakan CSS menurut Saputra (2012: 29) adalah:

1. Memisahkan pembuatan *file* antara HTML dan CSS.
2. Mempermudah dan mempersingkat pembuatan dan pemeliharaan dokumen *web*.
3. Akses *web* lebih cepat saat dimuat.

4. Fleksibel dan interaktif sehingga tampilan *web* lebih menarik dan nyaman dipandang.
5. Memiliki ukuran *file* yang kecil.
6. Dapat digunakan pada semua *web browser*.

Berikut ini adalah contoh penulisan kode CSS (Saputra, 2012: 33):

```
p {
color : blue;
}
```

Dengan menggunakan CSS di atas, maka *tag* <p> yang ada pada *file* HTML akan ditampilkan teks-nya dengan warna biru. Pada kode CSS di atas, *p* adalah *selector* yang berfungsi untuk memilih *tag* <p> pada *file* HTML, *color* adalah *property* yang akan mengubah atribut warna dari *tag* <p>, sedangkan *blue* adalah *value* yang diberikan untuk atribut warna tersebut.

Menurut Saputra (2012: 29-32), ada 3 cara untuk menerapkan CSS pada file HTML, yaitu:

1. *Embedded Style Sheet*

   Merupakan cara menerapkan CSS dimana kode CSS menyatu dengan *file* HTML, yaitu diantara *tag* <style> dan </style> sebelum *tag* <body>.

2. *Inline Style Sheet*

   Merupakan cara menerapkan CSS langsung pada *tag* HTML yang dibutuhkan saja. Hal ini dilakukan jika CSS digunakan hanya sekali pakai pada *tag* HTML tertentu.

3. *Linked Style Sheet*

   Merupakan cara menerapkan CSS dengan cara memisahkan *file* CSS dan *file* HTML. Untuk menggunakan CSS yang terpisah ini, maka pada *file* HTML dibuat kode untuk memanggil file CSS tersebut.

Bootstrap adalah *framework* HTML, CSS dan JavaScript yang paling populer untuk mengembangkan *web* yang *responsive* dan mengutamakan tampilan pada perangkat *mobile*. Bootstrap membuat proses pengembangan *web* menjadi lebih cepat dan lebih mudah (www.getbootstrap.com).

Menurut penelitian Listiyono dan Setiawan (2013), Bootstrap adalah sekumpulan perangkat gratis yang digunakan untuk membuat *website* dan aplikasi *web*. Bootstrap dapat membuat *layout* halaman *website*, tabel, tombol, *form*, navigasi dan komponen lainnya dalam sebuah *website* hanya dengan memanggil fungsi CSS *(class)* dalam berkas HTML yang telah didefinisikan.

2.3.3.3 PHP *(PHP: Hypertext Preprocessor)*

Gambar 2.18 Logo PHP
(Sumber: https://www.php.net/download-logos.php)

PHP adalah bahasa pemrograman *web* yang ditanam di sisi *server*. Ketika sebuah halaman *web* yang mengandung kode PHP diakses, prosesor PHP pada *server* akan mengeksekusi kode PHP tersebut kemudian menampilkan hasilnya ke

web browser sebagai halaman *web*. PHP dibuat oleh Rasmus Lerdorf pada tahun 1994 (Prasetio, 2012: 122-126).

Beberapa keunggulan PHP menurut Prasetio (2012: 125-126) yaitu:

1. Kesederhanaan, bahasa pemrograman PHP mudah dipelajari terutama untuk pemula.
2. PHP bersifat *open source*, artinya kode PHP tersedia secara gratis dan tidak bergantung pada suatu perusahaan tertentu.
3. Stabilitas dan kompatibilitas, PHP dapat berjalan dengan stabil pada berbagai macam sistem operasi sepertu UNIX, Windows dan MAC.
4. PHP dilengkapi dengan berbagai macam pendukung lain seperti mendukung berbagai macam jenis *database*, arsitektur yang dapat dikembangkan, menggunakan sumber daya yang kecil pada komputer dan mampu menampilkan halaman *web* dengan cepat.
5. Secara umum dapat dikatakan bahwa PHP adalah bahasa yang sangat mudah dipelajari.

Menurut Saputra (2012: 91), ada 4 macam format untuk menulis program PHP yaitu:

1. <?php ….. ?>
2. <? ….. ?>
3. <script language = “php”> ….. </script>
4. <% ….. %>

Dari 4 macam format penulisan bahasa pemrograman PHP di atas, format <?php ….. ?> dan <? ….. ?> adalah format penulisan yang paling banyak digunakan oleh *programmer*.

2.3.3.4 JavaScript dan jQuery

JavaScript adalah bahasa pemrograman yang digunakan untuk membuat *web* lebih dinamis dan interaktif. Kode JavaScript terintegrasi langsung dengan *file* HTML. Kode JavaScript biasanya dituliskan dalam bentuk fungsi yang diletakkan pada tag <head>, dibuka dengan tag <script type = "text/javascript"> (Prasetio, 2012: 300).

Sidik (2011) dalam Prasetio (2012: 301) menjelaskan bahwa JavaScript dijalankan oleh *interpreter* yang telah ditanamkan ke dalam *web browser* sehingga *web browser* dapat mengeksekusi program JavaScript. Berbeda dengan PHP yang merupakan bahasa pemrograman *server side*, JavaScript adalah bahasa pemrograman *client side*, artinya JavaScript dijalankan oleh *web browser* pada sisi klien (pengguna).

jQuery adalah pustaka JavaScript kecil yang menekankan pada interaksi antara JavaScript dan HTML. jQuery disimpan sebagai file JavaScript tunggal yang berisi semua metode jQuery. Fitur-fitur yang ditawarkan oleh jQuery adalah sebagai berikut (Prasetio, 2012: 344-345):

1. Mempermudah akses dan manipulasi ke bagian halaman tertentu.
2. Mempermudah perubahan tampilan dokumen, jQuery dapat mengubah tampilan CSS dengan mudah.

3. Merespon interaksi *user* dengan halaman *web*.
4. Menambah animasi pada halaman *web*.
5. Mempermudah AJAX.

2.3.4 Perangkat Lunak Aplikasi

Perangkat lunak aplikasi *(software application)* adalah suatu sub kelas perangkat lunak komputer yang memanfaatkan kemampuan komputer langsung untuk melakukan suatu tugas yang diinginkan pengguna (Husda, 2012: 35). Untuk mendukung pembuatan sistem pakar, pada penelitian ini digunakan beberapa perangkat lunak aplikasi yaitu: StarUML, XAMPP, phpMyAdmin, Sublime Text dan Mozilla Firefox.

2.3.4.1 StarUML

Gambar 2.19 Logo StarUML
(Sumber: http://staruml.io)

StarUML adalah salah satu aplikasi UML yang paling populer di dunia karena telah diunduh oleh lebih dari 4 juta orang yang tersebar di lebih dari 150 negara. StarUML dapat di pasang pada komputer dengan sistem operasi Windows, Linux dan MAC OS X. StarUML dapat digunakan untuk membuat 11 jenis diagram UML yaitu: *Class, Object, Use Case, Component, Deployment,*

Composite Structure, Sequence, Communication, Statechart, Activity dan *Profile Diagram* (http://staruml.io).

2.3.4.2 XAMPP

Gambar 2.20 Logo XAMPP
(Sumber: https://www.apachefriends.org/index.html)

XAMPP adalah sebuah *software* yang berfungsi untuk menjalankan *web* berbasis PHP yang menggunakan *database* MySQL di komputer lokal. XAMPP berperan sebagai *web server* pada komputer lokal. XAMPP juga dapat disebut sebuah *CPanel server virtual*, yang membantu melakukan *preview* sehingga dapat memodifikasi *web* tanpa harus *online* atau terakses internet (Wicaksono, 2008: 7).

2.3.4.3 phpMyAdmin

Gambar 2.21 Logo phpMyAdmin
(Sumber: https://www.phpmyadmin.net/static/images/logo-og.png)

phpMyAdmin adalah perangkat lunak gratis yang ditulis dalam bahasa pemrograman PHP yang ditujukan untuk menangani administrasi *database*

MySQL melalui tampilan *web*. phpMyAdmin mendukung berbagai operasi pada *database* MySQL dan MariaDB. Operasi-operasi yang sering digunakan seperti mengelola *database*, tabel, kolom, relasi, indeks, *users*, *permissions* dan lain-lain dapat dilakukan baik melalui antarmuka pengguna ataupun mengeksekusi pernyataan SQL secara langsung (www.phpmyadmin.net).

2.3.4.4 Sublime Text

Gambar 2.22 Logo Sublime Text
(Sumber: https://upload.wikimedia.org/wikipedia/en/4/4c/Sublime_Text_Logo.png)

Sublime Text adalah perangkat *text editor* yang canggih yang dapat digunakan untuk menulis bahasa pemrograman. Sublime Text mendukung sejumlah bahasa pemrograman diantaranya C, C++, C#, PHP, CSS, HTML, ASP dan lain-lain (www.sublimetext.com).

Ada beberapa kelebihan yang ada pada Sublime Text diantaranya (www.sublimetext.com):

1. *Goto anything,* digunakan untuk membuka *file* diawali dengan menarik satu *project file* yang sedang dikerjakan kemudian dengan menekan CTRL+P maka dapat dicari *file* apa yang akan dibuka dengan menuliskan nama *file* tersebut.

2. *Multiple Selection*, berfungsi untuk membuat perubahan pada kode program pada saat yang sama dalam beberapa baris yang berbeda.
3. *Command Pallete,* dengan menekan CTRL+SHIFT+P untuk menutup semua *file*, *convert case: lower case, remove tag* dan lain-lain.
4. *Split Editing,* memperbolehkan mengedit *file* berdampingan, atau mengedit dua lokasi pada satu *file* dengan beberapa baris dan kolom yang diinginkan.
5. *Cross Platform*, Sublime Text tersedia untuk platform Windows, Linux, OS X dan satu lisensi untuk semua sistem oprasi yang digunakan.

2.3.4.5 Mozilla Firefox

Gambar 2.23 Logo Mozilla Firefox
(Sumber: https://www.mozilla.org/media/img/firefox/template/header-logo-inverse-high-res.6a42e6a524f0.png)

Mozilla Firefox merupakan aplikasi *web browser* gratis yang dikembangkan oleh yayasan Mozilla dan beberapa *developer* pendukungnya. Mozilla Firefox pertama kali dirilis pada tanggal 9 November 2004 (versi 1.0) dengan nama Phoenix dan kemudian berganti nama menjadi Mozilla Firebird, hingga akhirnya berganti nama menjadi Mozilla Firefox. Dalam 12 hari setelah dirilis, versi 1.0 telah diunduh oleh lebih dari 5 juta pengguna karena sifatnya yang gratis dan *open source*. Versi 2.0 dirilis pada tanggal 24 Oktober 2006 dan versi 3.0 dirilis pada tanggal 17 Juni 2008 (Manzur, 2010: 2).

2.3.5 Sistem Manajemen Basis Data

Kusrini (2007: 2) menyatakan basis data adalah kelompok data yang saling berhubungan yang diorganisasi sedemikan rupa sehingga kelak dapat dimanfaatkan dengan cepat dan mudah. Basis data bertujuan untuk mengatur data sehingga diperoleh kemudahan, ketepatan dan kecepatan dalam pengambilan kembali. Beberapa operasi dasar basis data yaitu Kusrini (2007: 9):

1. Pembuatan basis data.
2. Penghapusan basis data.
3. Pembuatan *file* atau tabel.
4. Penghapusan *file* atau tabel.
5. Pengubahan tabel.
6. Penambahan atau pengisian.
7. Pengambilan data.
8. Penghapusan data.

Sistem Manajemen Basis Data atau *Database Management System (DBMS)* adalah *software* yang menangani semua akses ke basis data. Contoh dari DBMS antara lain: Microsoft SQL Server 2000, Oracle, MySQL, Interbase, Paradox, Microsoft Access dan lain-lain (Kusrini, 2007: 12).

Sistem pakar pada penelitian ini menggunakan *database* MySQL. MySQL termasuk ke dalam Sistem Manajemen Basis Data Relasional atau *Relational Database Management System (RDBMS)*. Menurut Nugroho (2010: 17-18), RDBMS adalah sistem penyimpanan data berbasis komputer yang elemen pokoknya adalah tabel-tabel yang saling berhubungan satu sama lain, dengan

masing-masing tabel memiliki kolom-kolom (atau sering juga disebut sebagai *field*) yang digunakan sebagai tempat untuk menyimpan data/informasi yang diperlukan oleh aplikasi yang akan dikembangkan. Relasi antar tabel dalam RDBMS dinyatakan dalam derajat kardinalitas. Derajat kardinalitas dibagi menjadi 3 jenis yaitu (Yanto, 2016: 40-41):

1. Derajat kardinalitas *One to One*

   Derajat kardinalitas *One to One* terjadi jika 1 entitas X hanya berelasi dengan 1 entitas Y atau sebaliknya.

2. Derajat kardinalitas *One to Many*

   Derajat kardinalitas *One to Many* terjadi jika 1 entitas X berelasi dengan banyak entitas Y atau sebaliknya.

3. Derajat kardinalitas *Many to Many*

   Derajat kardinalitas *Many to Many* terjadi jika banyak entitas X berelasi dengan banyak entitas Y atau sebaliknya.

Gambar 2.24 Logo MySQL
(Sumber: https://www.mysql.com/about/legal/logos.html)

Menurut Saputra (2012: 77), MySQL merupakan salah satu *database* yang cocok bila dipadukan dengan bahasa pemrograman PHP. MySQL bekerja menggunakan bahasa *Structure Query Language (SQL)* yang merupakan bahasa standar yang digunakan untuk memanipulasi *database*. Pada umumnya, perintah

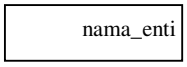
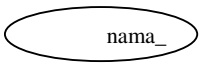
yang sering digunakan dalam MySQL adalah SELECT (mengambil), INSERT (menambah), UPDATE (mengubah) dan DELETE (menghapus).

Terdapat beberapa alasan yang menjadikan MySQL dipilih untuk menangani *database*, yaitu (Saputra, 2012: 78):

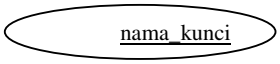
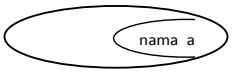
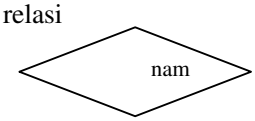
1. Bersifat *open source*.
2. Menggunakan bahasa SQL yang merupakan standar bahasa dalam pengolahan data.
3. Pemrosesan *database* sangat cepat dan stabil.
4. Sangat mudah dipelajari.
5. Memiliki dukungan pengguna MySQL.
6. Dapat digunakan pada berbagai sistem operasi.
7. *Multiuser*, MySQL dapat digunakan oleh banyak *user* dalam waktu yang bersamaan tanpa mengalami konflik.

Menurut A.S dan Shalahuddin (2011: 49), pemodelan awal *database* yang paling banyak digunakan adalah menggunakan *Entity Relationship Diagram (ERD)*. ERD digunakan untuk pemodelan *relational database*. Tabel 2.9 menyajikan simbol-simbol yang digunakan pada ERD.

Tabel 2.9 Simbol Pada *Entity Relationship Diagram (ERD)*

Simbol	Deskripsi
entitas / *entity* 	Entitas merupakan data inti yang akan disimpan; bakal tabel pada *database*
atribut 	*Field* atau kolom data yang butuh disimpan dalam suatu entitas

Tabel 2.9 Lanjutan

Simbol	Deskripsi
atribut kunci primer 	Field atau kolom data yang butuh disimpan dalam suatu entitas dan digunakan sebagai kunci akses *record* yang diinginkan; biasanya berupa *id*
atribut multi nilai / *multivalue* 	*Field* atau kolom data yang butuh disimpan dalam suatu entitas yang dapat memiliki nilai lebih dari satu
relasi 	Relasi yang menghubungkan antara entitas; biasanya diwakili dengan kata kerja
asisiasi / *association* 	Penghubung antara relasi dan entitas dimana di kedua ujungnya memiliki *multiplicity* kemungkinan jumlah pemakaian

Sumber: AS dan Shalahuddin (2011: 49-50)

2.4 Penelitian Terdahulu

Beberapa penelitian telah dilakukan baik itu penelitian tentang sistem pakar maupun tentang pengujian IC. Penelitian-penelitian tersebut didokumentasikan ke dalam jurnal dan menjadi referensi bagi penulis untuk mendukung penelitian ini. Berikut ini dijelaskan bebrapa penelitian tersebut.

Uky Yudatama (2008), Sistem Pakar untuk Diagnosis Kerusakan Mesin Mobil Panther Berbasis *Mobile*. Kerusakan mobil biasanya baru disadari oleh pemilikya setelah mobil tidak dapat beroperasi sebagaimana mestinya. Oleh karena itu diperlukan perawatan berkala dengan mendeteksi kerusakan yang terjadi pada mobil. Hasil dari penelitian ini adalah sebuah aplikasi sistem pakar yang mampu mendeteksi kerusakan mesin mobil Phanter.

Yin-Ho Yao dan Gilbert Y.P. Lin (2008), *Development of Intelligent Equipment Diagnosis and Maintenance System using JESS: Java Expert System Shell Technology*. Penelitian ini dilakukan pada sebuah pabrik TFT LCD di Taiwan. Sistem pakar yang dihasilkan dari penelitian ini digunakan untuk proses *maintenance* mesin produksi dan mampu meningkatkan efisiensi dari proses *maintenance* mesin produksi tersebut.

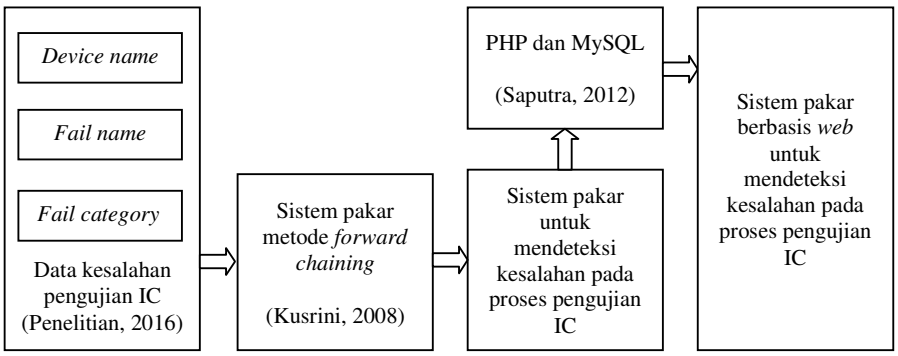
R.H. Khade dan Swapnil Gourkar (2013), *Fault Testing of CMOS Integrated Circuits Using Signature Analysis Method*. Penelitian ini dilakukan untuk mendukung perusahaan semikonduktor dalam rangka meminimalisir unit IC yang *reject* dengan cara menambahkan sebuah metode yang disebut *Signature Analysis Method* dalam proses pengujian IC CMOS. Kesimpulan dari penelitian ini adalah bahwa dengan menggunakan *Signature Analysis Method* mampu mengurangi jumlah unit *reject* pada proses produksi IC.

M. Idzdihar Idris, Nowshad Amin, Puvaneswaran Chelvanathan dan Faiz Arith (2014), *Thickness Dependence of The Surface Roughness of Micro Contact Deposited By Dc Magnetron Sputtering Technique*. Penelitian ini dilakukan untuk mencari jenis kontaktor baru untuk digunakan pada pengujian IC. Kontaktor lama dengan model *pogo pins* dianggap kurang cocok untuk digunakan terutama untuk menguji IC yang memiliki ukuran yang kecil karena akan menimbulkan bekas goresan pada kemasan IC. Kesimpulan dari penelitian ini adalah kontaktor jenis *Micro Contact* mampu menggantikan kontaktor jenis *pogo pins*.

Masaki Hashizumea, Shohei Suenaga dan Hiroyuki Yotsuyanagi (2015), *A Built-in Test Circuit for Detecting Open Defects by IDDT Appearance Time in*

CMOS ICs. Kesimpulan dari penelitian ini yaitu sebuah rangkaian tambahan telah diajukan untuk mendeteksi adanya sirkuit yang tidak terhubung *(open defect)* pada proses pengujian IC. Hasil simulasi mengunjukkan bahwa *open defect* dapat dideteksi dengan menggunakan rangkaian tambahan ini.

2.5 Kerangka Pemikiran



Gambar 2.25 Kerangka Pemikiran
(Sumber: Data Penelitian, 2016)

Sistem pakar pada penelitian ini dirancang sebagai alat bantu bagi personel teknisi pada *Test Departement* PT Unisem. Sistem pakar ini digunakan oleh teknisi untuk mencari solusi mengatasi kesalahan pada proses pengujian IC. Data tentang jenis-jenis kesalahan pada proses pengujian IC meliputi *device name*, *fail name* dan *fail category* beserta solusinya didapatkan dari personel *test engineer* PT Unisem. Selanjutnya data tersebut dianalisis dan diakuisisi ke dalam basis pengetahuan sistem pakar. Representasi pengetahuan yang digunakan adalah model kaidah produksi. Metode inferensi pada sistem pakar ini menggunakan metode *forward chaining* sehingga menghasilkan sebuah sistem pakar untuk mendeteksi kesalahan pada proses pengujian IC. Selanjutnya sistem pakar tersebut

diimplementaiskan ke dalam sebuah aplikasi berbasis *web* dengan menggunakan bahasa pemrograman PHP dan *database* MySQL. Hasil dari penelitian ini adalah sebuah sistem pakar berbasis *web* untuk mendeteksi kesalahan pada proses pengujian IC. Sistem pakar ini akan memberikan solusi untuk membantu personel teknisi pada *Test Departement* PT Unisem dalam mengatasi masalah pada proses pegujian IC.