BAB II

LANDASAN TEORI

2.1. Teori Dasar

Dalam penelitian ini terdapat beberapa teori dasar yang mendukung dalam menyelesaikan permasalahan antara lain yaitu kecerdasan buatan (*artificial intelligence*) dan beberapa sub disiplin ilmunya yaitu sistem pakar (*expert system*), logika *fuzzy* (*fuzzy logic*) dan jaringan saraf tiruan.

2.1.1. Kecerdasan Buatan (*Artificial Intelligence*)

Kecerdasan buatan adalah salah satu bidang ilmu komputer yang mendayagunakan komputer sehingga dapat berperilaku cerdas seperti manusia. Ilmu komputer tersebut mengembangkan perangkat lunak dan perangkat keras untuk menirukan tindakan manusia. Aktifitas manusia yang ditirukan seperti penalaran, penglihatan, pembelajaran, pemecahan masalah, pemahaman bahasa alami dan sebagainya (Hartati & Iswanti, 2008:1).

Kecerdasan buatan (*artificial intelligence*) didefinisikan sebagai kecerdasan entitas ilmiah. Sistem seperti ini umumnya dianggap komputer. Kecerdasan diciptakan dan dimasukan ke dalam suatu mesin (komputer) agar dapat melakukan

pekerjaan seperti yang dapat dilakukan manusia. Beberapa macam bidang yang menggunakan kecerdasan buatan antara lain sistem pakar, logika *fuzzy* dan jaringan saraf tiruan (Husda, 2012:192):

1. Jaringan Saraf Tiruan

Cabang ilmu kecerdasan buatan cukup luas dan erat kaitannya dengan disiplin ilmu yang lainnya. Jaringan saraf tiruan (*neural network*) merupakan kategori ilmu *soft computing*. *Neural network* sebenarnya mengadopsi dari kemampuan otak manusia yang mampu memberikan stimulasi/rangsangan, melakukan proses, dan memberikan *output* (Budiharto & Suhartono, 2014:168).

Beberapa metode yang ada pada jaringan saraf tiruan:

a. *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*.

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. 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.

2. Logika *Fuzzy* (*Fuzzy Logic*)

Logika *fuzzy* adalah metodologi sistem kontrol pemecahan masalah, yang sesuai untuk diimplementasikan pada sistem, mulai dari sistem yang sederhana, sistem kecil, *embedded system*, jaringan komputer, *multi-channel* atau *workstation* berbasis akuisisi data, dan sistem kontrol. Dalam logika *fuzzy* memungkinkan nilai keanggotaan berada di antara 0 dan 1, artinya suatu keadaan memungkinkan mempunyai dua nilai "Ya" dan "Tidak" secara bersamaan, namun besar nilainya tergantung pada bobot keanggotaan yang dimilikinya. Logika *fuzzy* dapat digunakan di berbagai bidang seperti pada sistem diagnosis penyakit (dalam bidang kedokteran); pemodelan sistem pemasaran, sistem operasi (dalam bidang ekonomi); kendali kualitas air, prediksi adanya gempa bumi, klasifikasi dan pencocokan pola (dalam bidang teknik) (Sutojo, dkk., 2011:211-212).

Beberapa metode yang digunakan dalam sistem inferensi *fuzzy* adalah (Sutojo, dkk., 2011:233-237):

1. Metode *Tsukamoto*

Dalam inferensinya, metode *tsukamoto* menggunakan tahapan sebagai berikut:

a. Fuzzifikasi.
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. Metode *Mamdani*

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

a. Fuzzifikasi.
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*) dengan menghasilkan himpunan *fuzzy* baru.
d. Defuzzifikasi menggunakan metode *centroid* (titik tengah).

3. Metode *Sugeno*

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

a. Fuzzifikasi.
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*).

3. Sistem Pakar (*Expert System*)

Sistem pakar (*expert system*) adalah suatu program komputer yang mengandung pengetahuan dari satu atau lebih pakar manusia mengenai suatu bidang spesifik (Husda, 2012:181).

Menurut Hartati & Iswanti (2008:45-111), metode yang digunakan dalam sistem pakar diantaranya:

1. *Forward*

Konsep ini dapat juga disebut sebagai pencarian yang dimotori data (*data driven search*). Runut maju melakukan proses perunutan (penalaran) dimulai dari premis-premis atau informasi masukan (*IF*) terlebih dahulu kemudian menuju konklusi atau *derived information* (*THEN*) (Hartati & Iswanti, 2008:45-46).

2. *Backward*

Secara umum, konsep ini diaplikasikan ketika tujuan ditentukan sebagai kondisi atau keadaan awal. Konsep ini disebut juga *goal-driven search*. Arah penalaran atau perunutan dalam konsep ini berlawanan dengan *forward chaining* (Hartati & Iswanti, 2008:46)

3. *Dempster Sheefer*

Berdasarkan penelitian Hartati dan Iswanti (2008:111), metode *dempster-shafer* pertama kali diperkenalkan oleh Dempster, yang melakukan percoban model ketidakpastian dengan *range* probabilitas sebagai probabilitas tunggal. Kemudian pada tahun 1976 Shafer mempublikasikan teori *dempster* tersebut pada sebuah buku yang berjudul *Mathematical Theory of Evident*.

2.1.2. Sistem Pakar (*Expert System*)

Menurut Kusrini (2008:3) sistem pakar adalah aplikasi berbasis komputer yang digunakan untuk menyelesaikan masalah sebagaimana yang dipikirkan oleh pakar, pakar yang dimaksud disini adalah orang yang mempunyai keahlian khusus yang dapat menyelesaikan masalah yang tidak dapat diselesaikan oleh orang awam.

Ada beberapa definisi tentang sistem pakar, antara lain sebagai berikut:

1. Menurut Durkin: sistem pakar adalah suatu program komputer yang dirancang untuk memodelkan kemampuan penyelesaian masalah yang dilakukan seorang pakar
2. Menurut Ignizio: sistem pakar adalah suatu model dan prosedur yang berkaitan, dalam suatu domain tertentu, yang mana tingkat keahliannya dapat dibandingkan dengan seorang pakar
3. Menurut Giarratano dan Riley: sistem pakar adalah suatu sistem komputer yang bisa menyamai atau meniru kemampuan seorang pakar.

Menurut Budiharto & Suhartono (2014:132-133) sistem pakar adalah program komputer yang menyimulasi penilaian dan perilaku manusia atau organisasi yang memiliki pengetahuan dan pengalaman ahli dalam bidang tertentu. Dengan sistem pakar, permasalahan yang seharusnya hanya dapat diselesaikan oleh para ahli, dapat diselesaikan oleh orang biasa/awam. Sedangkan, untuk para ahli, sistem pakar membantu aktivitas mereka sebagai asisten yang seolah-olah sudah mempunyai banyak pengalaman.

Kelebihan dan Kelemahan Sistem Pakar

Menurut Merlina & Hidayat (2012:4-5) manfaat dan kemampuan sistem pakar antara lain yaitu meningkatkan *output* dan produktivitas, menurunkan waktu pengambilan keputusan, meningkatkan kualitas proses dan produk, mengurangi *downtime*, menyerap keahlian langka, fleksibilitas, operasi peralatan yang lebih mudah, eliminasi kebutuhan peralatan yang mahal, operasi di lingkungan yang berbahaya, aksesibilitas ke pengetahuan dan *help desk*, kemampuan untuk bekerja dengan informasi yang tidak lengkap/tidak pasti, kelengkapan pelatihan, peningkatan pemecahan masalah dan pengambilan keputusan, meningkatkan proses pengambilan keputusan, meningkatkan kualitas keputusan, kemampuan untuk memecahkan persoalan kompleks, *transfer* pengetahuan ke lokasi terpencil

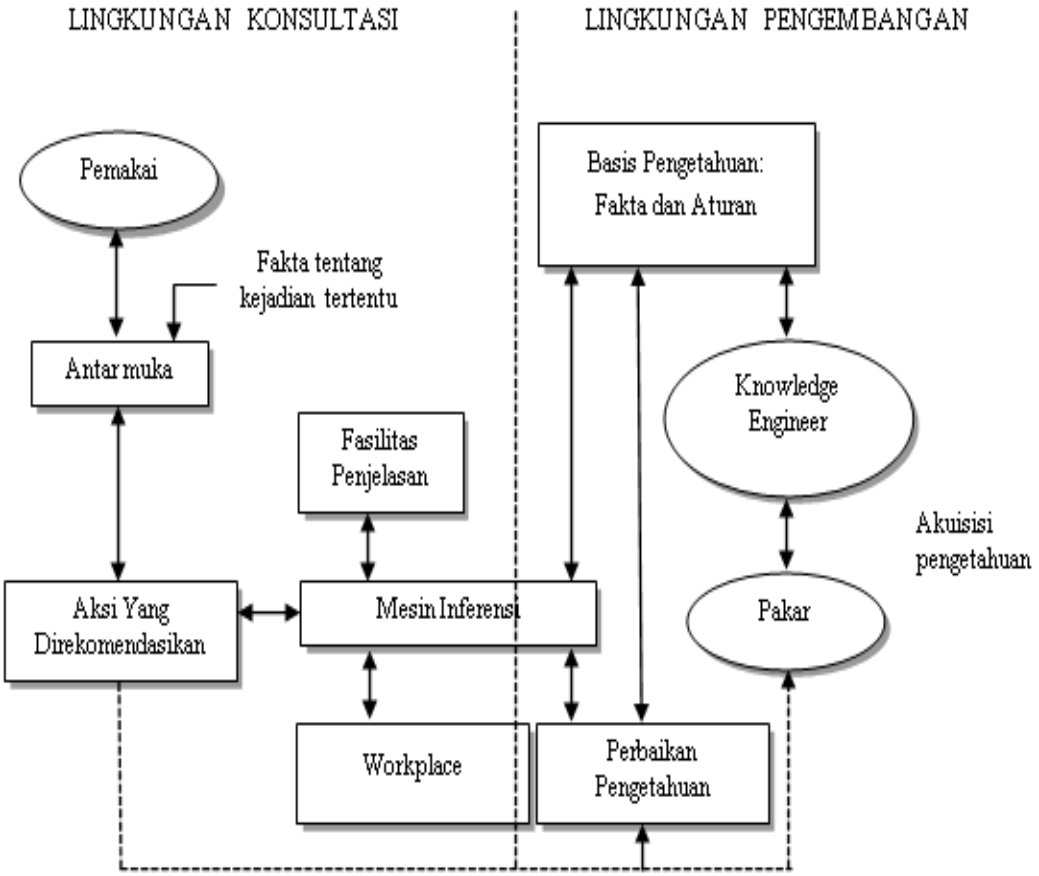
Disamping manfaat dan kemampuan sistem pakar, ada beberapa kelemahan sistem pakar antara lain yaitu pengetahuan tidak selalu siap tersedia, akan sulit mengekstrak keahlian dari manusia, pendekatan tiap pakar pada suatu penilaian situasi mungkin berbeda, tetapi benar, sulit, bahkan bagi pakar berkemampuan tinggi untuk mengikhtisarkan penilaian situasi yang baik pada saat berada dalam tekanan waktu, penggunaan sistem pakar memiliki batasan kognitif alami, sistem pakar bekerja dengan baik hanya dalam domain pengetahuan sempit, kebanyakan pakar tidak memiliki sarana mandiri untuk memeriksa apakah kesimpulannya masuk akal, kosa kata yang digunakan pakar untuk menyatakan fakta dan hubungan (Merlina & Hidayat, 2012:4).

Struktur Sistem Pakar

Sistem Pakar disusun oleh dua bagian utama, antara lain yaitu (Husda, 2012:182):

1. Lingkungan Pengembangan (*Development Environment*)
2. Lingkungan Konsultasi (*Consultation Environment*)

Menurut Husda (2012:182), lingkungan pengembangan sistem pakar digunakan untuk memasukkan pengetahuan pakar ke dalam lingkungan sistem pakar, sedangkan lingkungan konsultasi digunakan oleh pengguna yang bukan pakar guna memperoleh pengetahuan pakar.



Gambar 2. 1. Struktur Sistem Pakar
(Sumber: Husda, 2012:183)

Komponen-komponen yang terdapat dalam arsitektur/struktur sistem pakar (Husda, 2012:183):

1. Antarmuka Pengguna (*User Interface*)

   Merupakan mekanisme yang digunakan oleh pengguna dan 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.

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 obyek 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 di *transfer* ke dalam basis pengetahuan. Pengetahuan diperoleh dari pakar, dilengkapi dengan buku, basis data, laporan penelitian dan pengalaman pemakai. Metode akuisisi pengetahuan yaitu:

a. Wawancara

Metode yang paling banyak 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 di observasi.

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*)

Komponen 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.

5. *Workplace*/*Blackboard*

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

   a. Rencana: Bagaimana menghadapi masalah.
   b. Agenda: Aksi-aksi yang potensial yang sedang menunggu untuk di eksekusi.
   c. Solusi: Calon aksi yang akan dibangkitkan.

6. Fasilitas Penjelasan

   Adalah komponen tambahan yang akan meningkatkan kemampuan sistem pakar. Digunakan untuk melacak respon dan memberikan penjelasan tentang kelakuan sistem pakar secara interaktif melalui pertanyaan:

   a. Mengapa suatu pertanyaan ditanyakan oleh sistem pakar?
   b. Bagaimana konklusi dicapai?
   c. Mengapa ada alternatif yang dibatalkan?
   d. Rencana apa yang digunakan untuk mendapatkan solusi?

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.

Basis Pengetahuan

Basis pengetahuan berisi pengetahuan-pengetahuan dalam penyelesaian masalah, tentu saja di dalam domain tertentu. Ada 2 bentuk pendekatan basis pengetahuan yang sangat umum digunakan, yaitu sebagai berikut (Merlina & Hidayat, 2012:3-4):

1. Penalaran Berbasis Aturan (*Rule-Base Reasoning*)

   Pada penalaran berbasis aturan, pengetahuan direpresentasikan dengan menggunakan aturan berbentuk: *IF-THEN*. Bentuk ini digunakan apabila kita memiliki sejumlah pengetahuan pakar pada suatu permasalahan tertentu, dan si pakar dapat menyelesaikan masalah tersebut secara berurutan. Di samping itu, bentuk ini juga digunakan apabila dibutuhkan penjelasan tentang jejak (langkah-langkah) pencapaian solusi.

2. Penalaran Berbasis Kasus (*Case-Base Reasoning*)

   Pada penalaran berbasis kasus, basis pengetahuan akan berisi solusi-solusi yang telah dicapai sebelumnya, kemudian akan diturunkan suatu solusi untuk keadaan yang terjadi sekarang (fakta yang ada). Bentuk ini digunakan apabila *user* menginginkan untuk tahu lebih banyak lagi pada kasus-kasus yang hampir sama (mirip). Selain itu, bentuk ini juga digunakan apabila kita telah memiliki sejumlah situasi atau kasus tertentu dalam basis pengetahuan.

Representasi Pengetahuan

Untuk membuat sistem pakar yang efektif harus dipilih representasi pengetahuan yang tepat. Pemilihan representasi pengetahuan yang tepat akan

membuat sistem pakar dapat mengakses basis pengetahuan tersebut untuk keperluan pembuatan keputusan (Hartati & Iswanti, 2008:22).

Beberapa teknik dikembangkan untuk dapat digunakan sebagai representasi pengetahuan, diantaranya adalah (Budiharto & Suhartono, 2014:75-81):

1. Representasi Logika

   Logika didefinisikan sebagai ilmu untuk berfikir dan menalar dengan benar untuk mendapatkan kesimpulan yang abash. Tujuan dari logika adalah memberikan aturan-aturan penalaran sehingga orang dapat menentukan nilai benar atau salah suatu kalimat. Representasi logika terbagi menjadi 2, yaitu *propositional logic* dan *predicate logic*. *Propositional logic* adalah suatu pernyataan yang dapat bernilai benar (B) atau salah (S). *Predicate logic* digunakan untuk merepresentasikan hal-hal yang tidak dapat direpresentasikan menggunakan *propositional logic*. Pada *Predicate logic*, kita dapat merepresentasikan fakta-fakta sebagai suatu pernyataan yang disebut dengan WFF (*Well-Formed Formula*).

2. Kaidah Produksi (*Production Rule*)

   Kaidah menyediakan cara formal untuk merepresentasikan rekomendasi, arahan, atau strategi. Kaidah produksi dituliskan dalam bentuk jika-maka (*IF-THEN*). Kaidah *IF-THEN* menghubungkan antesenden (*antecedent*) dengan konsekuensi yang diakibatkannya (Hartati & Iswanti, 2008:25). Pada pengetahuan ini disajikan dalam aturan-aturan yang berbentuk pasangan keadaan-aksi (*condition-action*): "*IF* keadaan terpenuhi atau terjadi *THEN* suatu aksi akan terjadi". Sistem pakar yang basis pengetahuannya disajikan dalam bentuk aturan

produk disebut sistem berbasis-aturan (*rule-based system*). Kondisi dapat terdiri atas banyak bagian, demikian pula dengan aksi. Urutan keduanya juga dapat dipertukarkan letaknya (Merlina & Hidayat, 2012:18).

3. Jaringan Semantik (*Semantic Network*)

   Menurut Merlina & Hidayat (2012:17) jaringan semantik merupakan jaringan data dan informasi, yang menunjukan hubungan antar berbagai objek dimana informasi yang terhubung tersebut adalah informasi yang proporsional (suatu pernyataan yang dapat bernilai benar atau salah). Dalam matematika, istilah jaringan semantik merupakan suatu label atau graph berarah. Struktur jaringan semantik terdiri atas *node* atau simpul dan busur atau *arc* yang menghubungkannya. Simpul menyatakan objek sedangkan busur menyatakan *link*. *Link* dari jaringan semantik digunakan untuk menunjukan hubungan (*relationship*) antar simpul-simpul tersebut.

4. Bingkai (*Frame*)

   Bingkai adalah struktur data yang mengandung semua informasi/pengetahuan yang relevan dari suatu objek (Merlina & Hidayat, 2012:18). Bingkai berupa kumpulan slot-slot yang berisi atribut untuk mendeskripsikan pengetahuan. Pengetahuan yang termuat dalam slot dapat berupa kejadian, lokasi, situasi ataupun elemen-elemen lainnya (Hartati & Iswanti, 2008:24).

3.1.2.1. Kaidah Produksi

Kaidah menyediakan cara formal yang dituliskan dalam bentuk jika-maka (*IF-THEN*) untuk merepresentasikan rekomendasi, arahan, atau strategi. Kaidah *IF-THEN* menghubungkan antesenden (*antecedent*) dengan konsekuensi yang diakibatkannya. Berikut ini adalah contoh struktur kaidah *IF-THEN* yang menghubungkan obyek (Adedeji, 1992 *dalam* Hartati dan Iswanti, 2008: 25):

IF premis *THEN* konklusi

IF masukan *THEN* keluaran

IF kondisi *THEN* tindakan

IF antesenden *THEN* konsekuen

IF data *THEN* hasil

IF tindakan *THEN* tujuan

IF aksi *THEN* reaksi

IF sebeb *THEN* akibat

IF gejala *THEN* diagnosis

Premis mengacu pada fakta yang harus benar sebelum konklusi tertentu dapat diperoleh. Masukan mengacu pada data yang harus tersedia sebelum keluaran dapat diperoleh. Kondisi mengacu pada keadaan yang harus berlaku sebelum tindakan dapat diambil. Antesenden mengacu situasi yang terjadi sebelum konsekuensi dapat diamati. Data mengacu pada informasi yang harus tersedia sehingga sebuah hasil dapat diperoleh. Tindakan mengacu pada kegiatan yang harus dilakukan sebelum hasil dapat diharapkan. Aksi mengacu pada kegiatan yang menyebabkan munculnya efek dari tindakan tersebut. Sebab mengacu pada keadaan tertentu yang

menimbulkan akibat tertentu. Gejala mengacu pada keadaan yang menyebabkan adanya kerusakan atau keadaan tertentu yang mendorong adanya pemeriksaan (Hartati& Iswanti, 2008:25-26).

Sebelum sampai pada bentuk kaidah produksi, pengetahuan yang berhasil didapatkan dari domain tertentu disajikan dalam bentuk tabel keputusan kemudian dibuat pohon keputusannya (Hartati& Iswanti, 2008:26).

3.1.2.2. Tabel Keputusan dan Pohon Keputusan

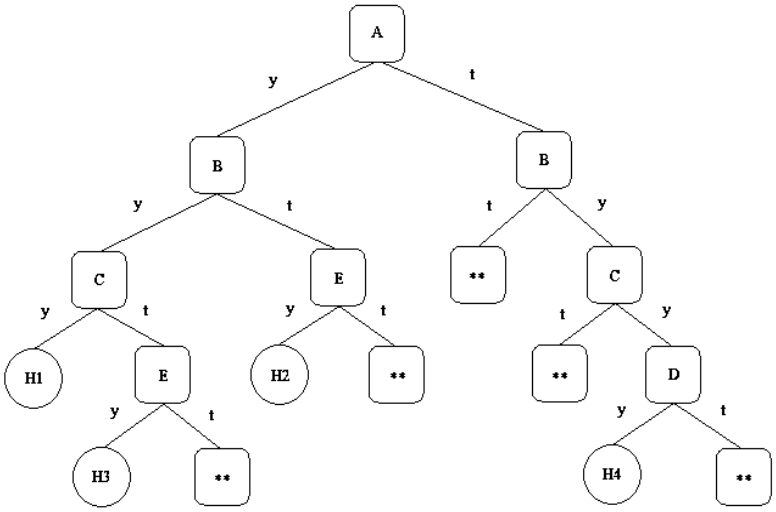
Tabel keputusan merupakan suatu cara untuk mendokumentasikan pengetahuan. Kaidah yang disajikan dalam bentuk kaidah produksi disusun dari tabel keputusan. Meskipun kaidah secara langsung dapat dihasilkan dari tabel keputusan tetapi untuk menghasilhan kaidah yang efisien terdapat suatu langkah yang harus ditempuh yaitu dengan membuat pohon keputusan (Hartati& Iswanti, 2008:26-27). Berikut ini adalah contoh penyajian dalam bentuk tabel keputusan dan pohon keputusan:

Tabel 2.1. Tabel Keputusan

Hipotesa / ***Evidence***	**Hipotesa 1**	**Hipotesa 2**	**Hipotesa 3**	**Hipotesa 4**
Evidence A	Ya	Ya	Ya	Tidak
Evidence B	Ya	Tidak	Ya	Ya
Evidence C	Ya	Tidak	Tidak	Ya
Evidence D	tidak	Tidak	Tidak	Ya
Evidence E	tidak	Ya	Ya	Tidak

Sumber: Hartati dan Iswanti (2008:32)

Berdasarkan pada tabel 2.1 diatas, maka dapat dibuat pohon keputusan sebagai berikut:



Keterangan:

A = *evidence* A,	H1 = hipotesa 1,	y	= ya
B = *evidence* B,	H2 = hipotesa 2,	t	= tidak
C = *evidence* C,	H3 = hipotesa 3,	**	= tidak menghasilkan hipotesa tertentu
D = *evidence* D,	H4 = hipotesa 4		

Gambar 2. 2. Pohon Keputusan
(Sumber: Hartati dan Iswanti, 2008:33)

Dari gambar 2.2 pohon keputusan diatas dapat diketahui bahwa hipotesa H1 terpenuhi jika memenuhi *evidence* A, B, dan C. Hipotesa H2 terpenuhi jika memiliki *evidence* A dan *evidence* E. Hipotesa H3 akan terpenuhi jika memiliki *evidence* A, B, dan E. Hipotesa H4 akan dihasilkan jika memenuhi *evidence* B, C, dan D. Notasi "y" mengandung arti memenuhi *node* (*evidence*) di atasnya, notasi "t" artinya tidak memenuhi.

Dalam sesi konsultasi pada sistem pakar, *node-node* yang mewakili *evidence* biasanya akan menjadi pertanyaan yang diajukan oleh sistem. Dengan melihat

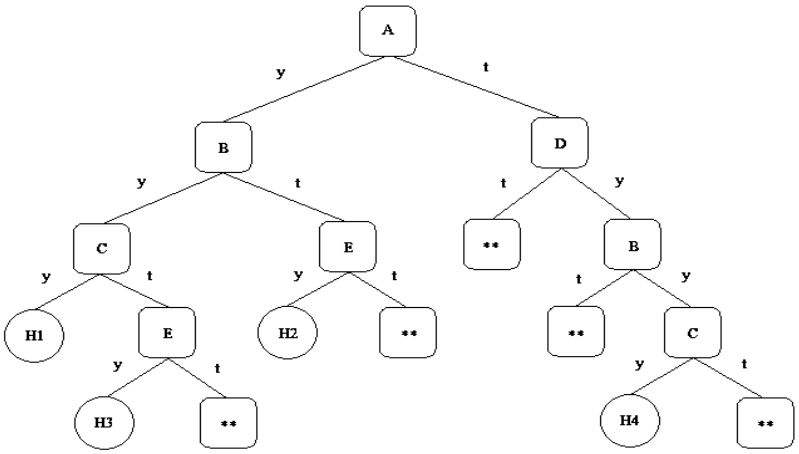
pohon keputusan pada gambar 2.2 permasalahan dapat saja terjadi pada awal sesi konsultasi yaitu pada saat sistem pakar menanyakan “apakah memiliki *evidence* A?”. Permasalahannya adalah apapun jawaban pengguna baik “ya” atau “tidak” maka sistem akan menanyakan *evidence* B. Ini berarti jawaban pengguna tidak akan mempengaruhi sistem. Salah satu cara untuk mengatasi hal ini adalah dengan mengubah urutan pada tabel keputusan seperti terlihat pada tabel 2.2.

Tabel 2.2. Alternatif Tabel Keputusan

Hipotesa / ***Evidence***	**Hipotesa 1**	**Hipotesa 2**	**Hipotesa 3**	**Hipotesa 4**
Evidence A	Ya	Ya	Ya	Tidak
Evidence D	Tidak	Tidak	Tidak	ya
Evidence B	Ya	Tidak	Ya	Ya
Evidence C	Ya	Tidak	Tidak	Ya
Evidence E	Tidak	Ya	Ya	tidak

Sumber: Hartati dan Iswanti (2008:34)

Berdasarkan tabel 2.2 diatas, dapat dihasilkan pohon keputusan seperti pada gambar dibawah ini:



Keterangan:

A = *evidence* A,	H1 = hipotesa 1,	y	= ya
B = *evidence* B,	H2 = hipotesa 2,	t	= tidak
C = *evidence* C,	H3 = hipotesa 3,	**	= tidak menghasilkan hipotesa tertentu
D = *evidence* D,	H4 = hipotesa 4		

Gambar 2. 3. Pohon Keputusan
(Sumber: Hartati dan Iswanti, 2008:35)

Dari gambar 2.3 pohon keputusan diatas, masing-masing *node* yang mewakili *evidence* tertentu untuk kondisi "y" dan "t" sudah tidak mengarah pada *evidence* yang sama. Hal ini berarti jawaban pengguna yang berbeda akan mengarah pada pertanyaan yang berbeda pula.

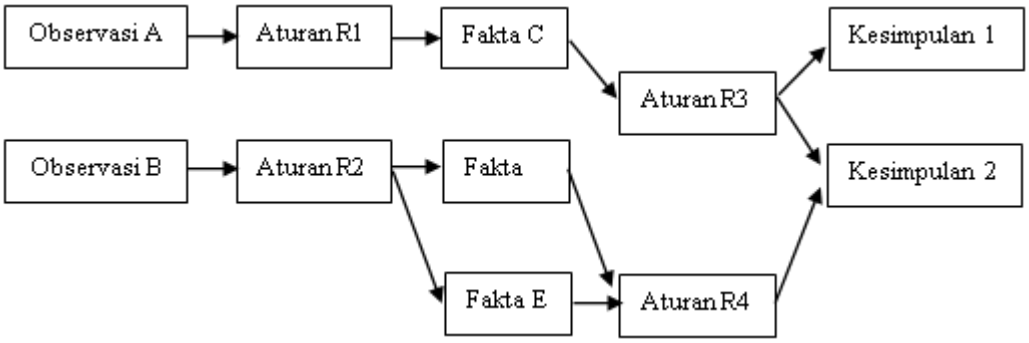
Kaidah yang dapat dihasilkan berdasarkan pohon keputusan pada gambar 2.3 adalah sebagai berikut:

1. Kaidah 1: *IF* A *AND* B *AND* C *THEN* H1
2. Kaidah 2: *IF* A *AND* B *AND* E *THEN* H3
3. Kaidah 3: *IF* A *AND* E *THEN* H2
4. Kaidah 4: *IF* D *AND* B *AND* C *THEN* H4

Model representasi pengetahuan kaidah produksi banyak digunakan pada aplikasi sistem pakar karena model representasi ini mudah dipahami dan bersifat deklaratif sesuai dengan jalan pikiran manusia dalam menyelesaikan suatu masalah, dan mudah diinterpretasikan (Hartati& Iswanti, 2008:39).

3.1.2.3. Forward Chaining

Forward chaining merupakan metode pencarian atau teknik pelacakan ke depan yang dimulai dengan informasi yang ada dan penggabungan *rule* untuk menghasilkan suatu kesimpulan atau tujuan. Pada gambar di bawah ini merupakan contoh dari pelacakan runut maju (*forward chaining*):



Gambar 2. 4. Proses *Forward Chaining*
(Sumber: Jurnal ISSN: 2086-9398 Vol. I Nomor 4, Nopember 2011)

2.1.3. *Database*

Menurut A.S. dan Shalahuddin (2013:44), *system database* atau basis data adalah sistem terkomputerisasi yang tujuan utamanya adalan memelihara data atau informasi yang sudah diolah dan membuat informasi tersedia saat dibutuhkan. *Database* adalah media untuk menyimpan data agar dapat diakses dengan mudah dan cepat. Kebutuhan basis data meliputi memasukkan, menyimpan, dan mengambil data serta membuat laporan berdasarkan data yang telah disimpan. *Database Management System* (*DBMS*) adalah suatu sistem aplikasi yang digunakan untuk menyimpan, mengelola, dan menampilkan data.

Syarat minimal dari *DBMS* antara lain (A.S. dan Shalahuddin, 2011:45):

1. Menyediakan fasilitas untuk mengelola akses data.
2. Mampu menangani integritas data.
3. Mampu menangani akses data yang dilakukan secara bersamaan.
4. Mampu menangani *backup* data.

Ada beberapa *DBMS* versi komersial yang paling banyak digunakan saat ini antara lain:

1. *DBMS* versi komersial, yaitu *Oracle, Microsoft SQL Server, IBM DB2,* dan *Microsoft Access.*
2. *DBMS* versi *open source*, yaitu *MySQL, PostgreSQL, Firebird,* dan *SQLite.*

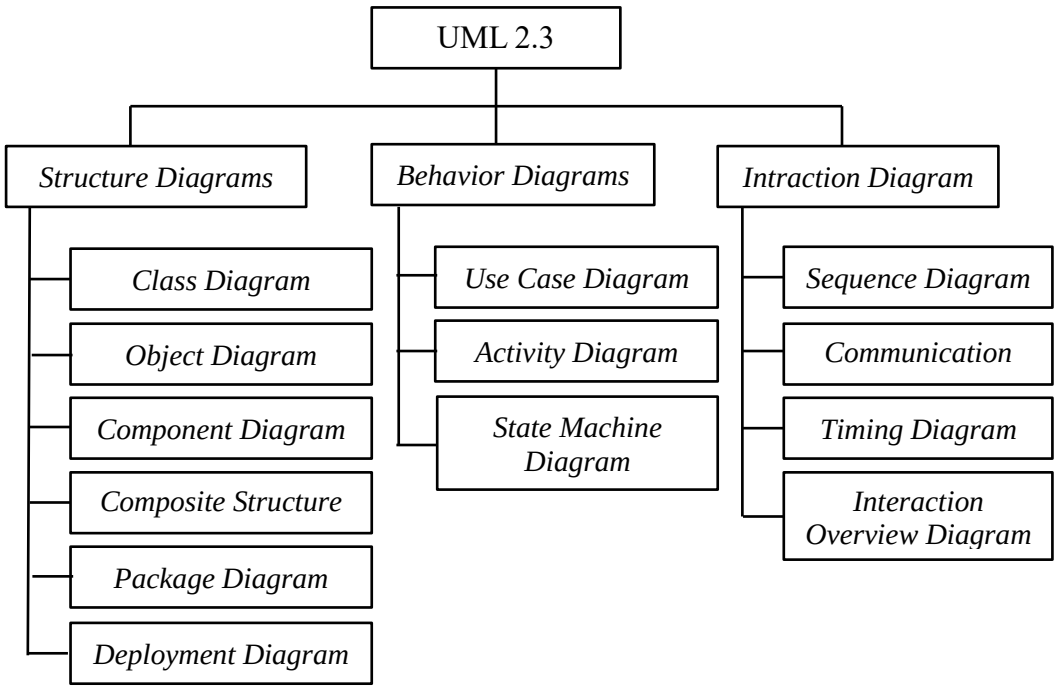
Sedangkan *DBMS* versi *open source* yang cukup berkembang dan paling banyak digunakan saat ini antara lain *MySQL, PostgreSQL, Firebird,* dan *SQLite.* Hampir semua *DBMS* mengadopsi *SQL* (*Structure Query Language*) sebagai bahasa untuk mengelola data pada *DBMS.*

2.1.4. *Unified Modelling Language* (*UML*)

UML (*Unified Modeling Language*) adalah standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement*, membuat analisis &*desain*, serta menggambarkan arsitektur dalam pemrograman berorientasi objek (Rosa dan Shalahuddin, 2011:113).

Menurut Rosa dan Shalahuddin (2011:120-121) secara fisik, *UML* adalah sekumpulan spesifikasi yang dikeluarkan oleh OMG (*Object Management Group*).

UML terbaru adalah *UML 2.3* yang terdiri dari 4 macam spesifikasi, yaitu Diagram *Interchange Spesification, UML Infrastructure, UML Superstrukture,* dan *Object Constraint Language* (*OCL*). Pada *UML 2.3* terdiri dari 13 macam diagram yang dikelompokan dalam 3 kategori, seperti yang terlihat pada gambar dibawah.



Gambar 2. 5. Diagram UML
(Sumber: Rosa dan Shalahuddin, 2011:121)

Berikut penjelasan singkat dari pembagian kategori tersebut:

1. *Structure Diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
2. *Behavior Diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.

3. *Interaction Diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antar sub sistem pada suatu sistem.

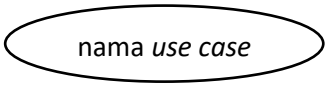
Berikut akan dijelaskan beberapa diagram diantaranya *use case diagram, activity diagram, sequence diagram* dan *class diagram*:

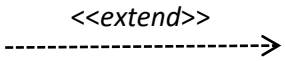
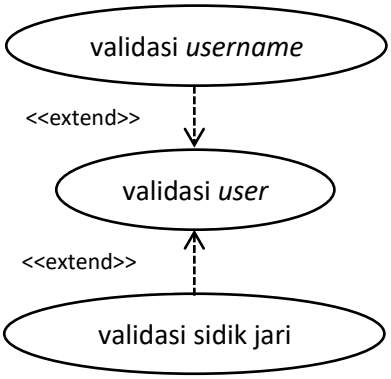
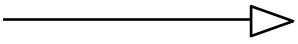
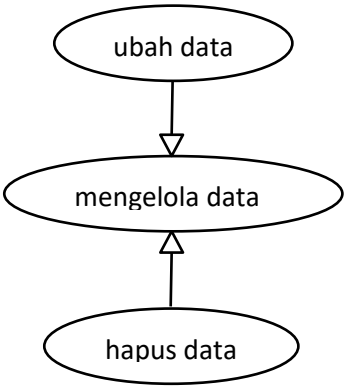
1. *Use Case Diagram*

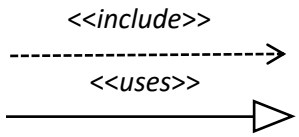
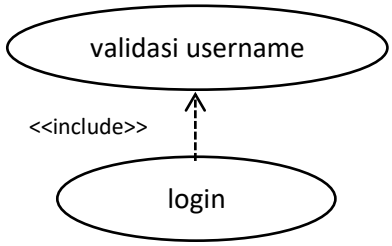
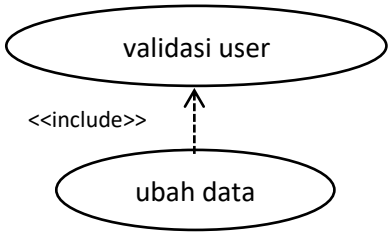
Menurut Rosa dan Shalahuddin (2011:130), *use case diagram* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, *Use case* digunakan untuk mengetahui fungsi apa saja yang ada didalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu.

Berikut adalah simbol-simbol yang ada pada *Use case*:

Tabel 2.3. Simbol-Simbol Pada *Use Case*

Simbol	Deskripsi
Use Case 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal di awal frase nama *use case*
Aktor/*Actor*  nama aktor	Orang, proses, atau sistem lain yang berintraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor

Asosiasi/*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 walau tanpa *use case* tambahan itu; mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek; biasanya *use case* tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan, misal  arah panah mengarah pada *use case* yang ditambahkan
Generalisasi/*Generalizatipon* 	Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya, misalnya: 

	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 dijalankan use case ini ada dua sudut pandang yang cukup besar mengenai *include* di *use case*: a. *Include* berarti *use case* yang ditambahkan akan selalu dipanggil saat *use case* tambahan dijalankan, misal pada kasus berikut:  b. *Include* berarti *use case* yang tambahan akan selalu melakukan pengecekan apakah *use case* yang ditambahkan telah dijalankan sebelum *use case* tambahan dijalankan, misal pada kasus berikut:  kedua interpretasi di atas dapat dianut salah satu atau keduanya tergantung pada pertimbangan dan interpretasi yang dibutuhkan

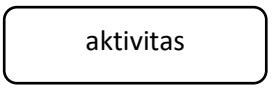
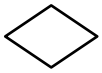
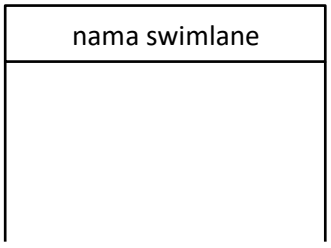
(Sumber: Rosa dan Shalahuddin, 2011:131-133)

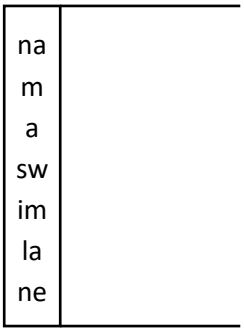
2. *Activity Diagram*

Activity diagram menggambarkan aktivitas dari sebuah sistem. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan sistem (Rosa dan Shalahuddin, 2011:134)

Berikut adalah simbol-simbol yang ada pada diagram aktivitas (*activity diagram*), seperti yang terlihat di bawah:

Tabel 2.4. Simbol-Simbol Pada *Activity Diagram*

Simbol	Deskripsi
Status Awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal
Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja
Percabangan/*Decision* 	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu
Penggabungan/*Join*	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu
Status Akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir
Swimlane 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi

atau  	

(Sumber: Rosa dan Shalahuddin, 2011:134-135).

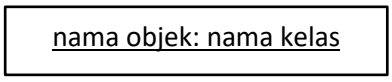
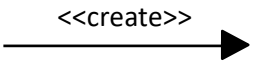
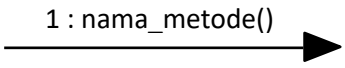
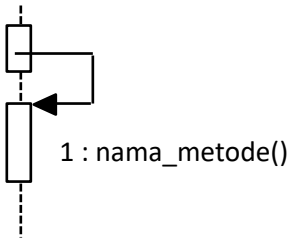
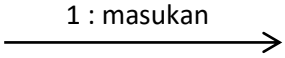
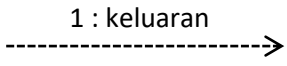
3. *Sequence Diagram*

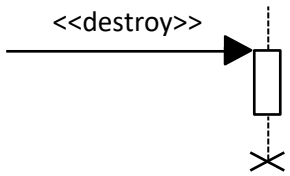
Menurut Rosa dan Shalahuddin (2011:137), *sequence diagram* menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek.

Berikut adalah simbol-simbol yang ada pada diagram sekuen (*sequence diagram*), seperti yang terlihat pada tabel dibawah:

Tabel 2.5. Simbol-Simbol Pada *Sequence Diagram*

Simbol	Deskripsi
Aktor  nama aktor atau nama aktor tanpa waktu aktif	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama actor
Garis Hidup/*Lifeline* 	Menyatakan kehidupan suatu objek

Objek 	Menyatakan objek yang berinteraksi pesan
Waktu Aktif 	Menyatakan objek dalam keadaan aktif dan berinteraksi pesan
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, karena ini memanggil operasi/metode maka operasi/metode yang dipanggil harus ada pada diagram kelas sesuai dengan kelas objek yang berinteraksi
Pesan Tipe *Send* 	Menyatakan bahwa suatu objek mengirimkan data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi
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 yang menerima kembalian

Pesan Tipe *Destroy*  	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada *create* maka ada *destroy*

(Sumber: Rosa dan Shalahuddin, 138-139).

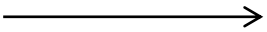
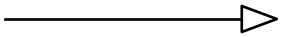
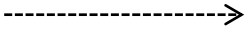
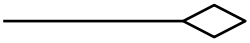
4. *Class Diagram*

Class diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem (Rosa dan Shalahuddin, 2011:122).

Berikut adalah simbol-simbol yang ada pada diagram kelas (*class diagram*) antara lain:

Tabel 2.6. Simbol-Simbol pada *Use Case*

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

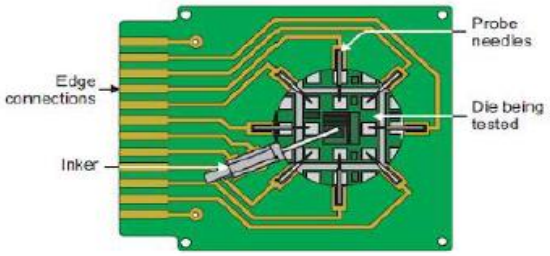
	lain, asosiasi biasanya juga disertai dengan *multiplicity*
Generalisasi 	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum-khusus)
Kebergantungan/*Dependency* 	Relasi antar kelas dengan makna kebergantungan antar kelas
Agregasi/*Aggregation* 	Relasi antar kelas dengan makna semua-bagian (*whole-part*)

(Sumber: Rosa dan Shalahuddin, 2011:123-124).

2.2. Variabel

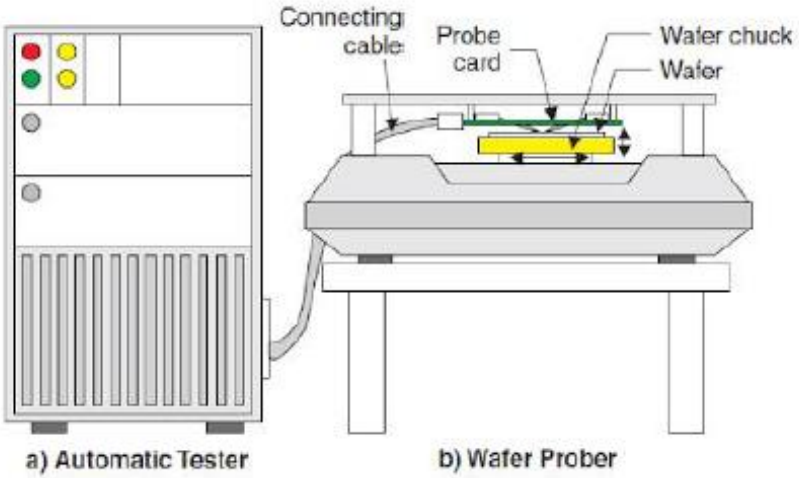
Variabel penelitian adalah segala sesuatu yang berbentuk apa saja yang ditetapkan oleh peneliti untuk dipelajari sehingga diperoleh informasi tentang hal tersebut, kemudian ditarik kesimpulannya (Sugiyono, 2014:38).

Objek yang digunakan dalam penelitian ini adalah *wafer*, dan *die* yang dihasilkan pada proses *wafer fabrication* tidak 100% berfungsi dengan baik. Untuk mengetahui *die* mana saja yang rusak, diperlukan pengetesan awal. Setiap jenis *die* yang diproduksi memiliki alat pengetesan tersendiri yang disebut *probe card.* Alat spesial ini dibuat khusus untuk setiap jenis *die* yang berbeda, tetapi semuanya telah dilengkapi dengan jarum kecil yang dirancang sedemikian rupa supaya pas dengan posisi *Bond Pad* pada *Die* yang akan dites.



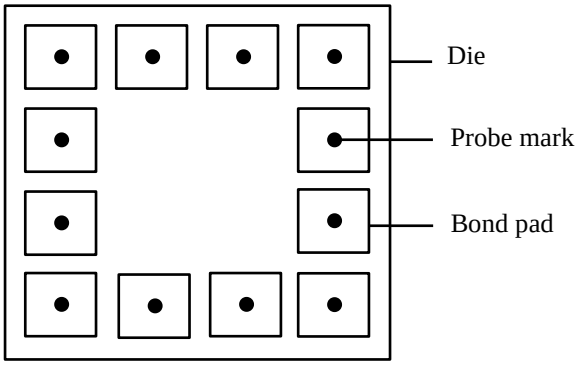
Gambar 2. 6. *Simple Probe Card*

Silikon *wafer* yang akan dites kemudian dijepit pada alat penjepit yang terdapat pada *wafer prober*. *Wafer prober* sepenuhnya dikendalikan oleh *Tester*, yaitu sistem komputerisasi yang bisa diprogram secara otomatis untuk menggerakan *probe card* guna melakukan pengetesan berbagai sifat kelistrikan pada setiap *die* yang terdapat pada permukaan silikon *wafer* kemudian menandai setiap *die* yang tidak berfungsi dengan baik (rusak).



Gambar 2. 7. *Wafer Test Setup*

Sebuah contoh *Probe Mark* yang terdapat di salah satu *Die* pada *Wafer* bisa dilihat pada gambar 2.8 *Simple Probe Mark*, di bawah ini:



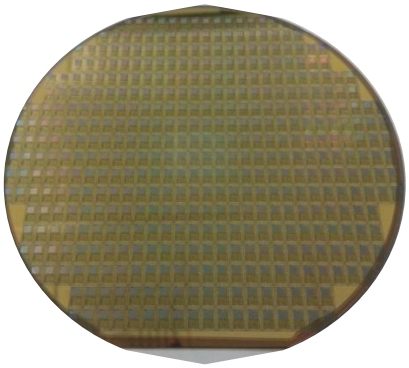
Gambar 2. 8. *Simple Probe Mark*
Sumber: Data penelitian 2016

Prober

Gambar 2. 9. Mesin/*Prober UF200*
Sumber: Data penelitian 2016

Pada gambar 2.9 diatas merupakan sebuah mesin *UF200* yang digunakan untuk melakukan proses pengujian *wafer* di PT Unisem Batam. *UF200* juga digunakan pada beberapa area yaitu *Zilog* dan juga *Mixed Signal.*

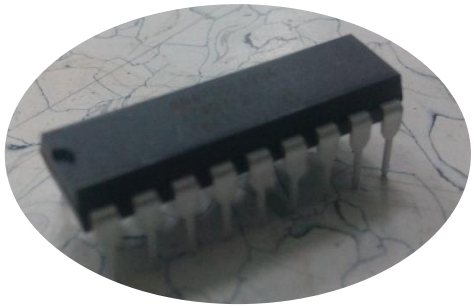
Wafer



Gambar 2. 10. Contoh Fisik *Wafer*
Sumber: Data penelitian 2016

Pada gambar 2.10 diatas merupakan penampakan secara fisik dari sebuah *wafer* yang akan dilakukan proses pengujian. *Wafer* ini terdiri dari banyak *Die* (*Circuit*) yang merupakan cikal bakal dari IC (*Integrated Circuit*)

***Integrated Circuit* (*IC*)**

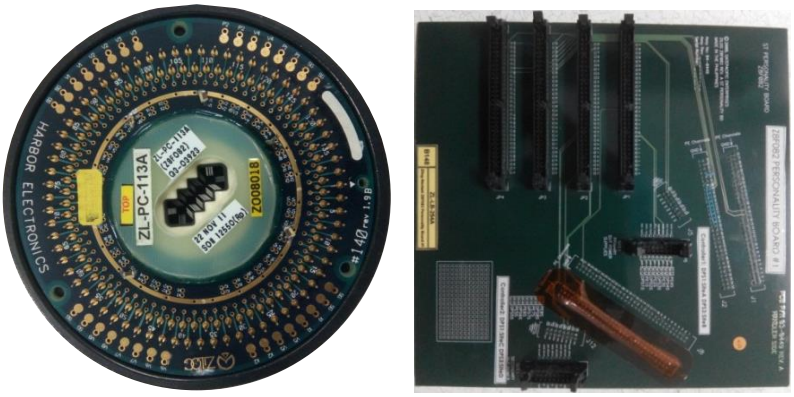


Gambar 2. 11. *Integrated Circuit* (*IC*)
Sumber: Data penelitian 2016

Pada gambar 2.11 diatas merupakan salah satu bentuk dari *integrated circuit*, dimana *wafer* yang telah dilakukan proses pengujian kemudian dikemas menjadi sebuah sirkuit terpadu (*Integrated Circuit*). IC adalah komponen elektronika semi konduktor yang merupakan gabungan dari ratusan atau ribuan komponen-komponen lain seperti *Resistor, Kapasitor, Dioda* dan *Transistor* yang terintegrasi menjadi sebuah rangkaian berbentuk chip kecil. IC digunakan untuk beberapa

keperluan pembuatan peralatan elektronik agar mudah dirangkai menjadi peralatan yang berukuran relatif kecil.

Probe Card* dan *Loard board



Gambar 2. 12. *Probe Card* dan *Loard Board*
(Sumber: Data penelitian 2016)

Pada gambar 2.12 diatas, disebelah kiri adalah salah satu contoh dari sebuah *probe card* dan disisi sebelah kanan adalah salah satu contoh dari *loard board*. Kedua komponen tersebut adalah komponen pendukung yang sangat penting dan tidak bisa dipisahkan pada saat sedang melakukan proses pengujian *wafer*.

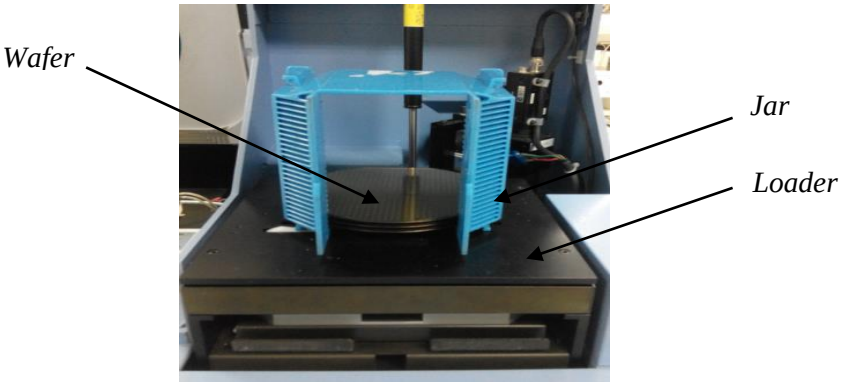
GPIB Cable

Gambar 2. 13. *GPIB Cable*
Sumber: Data penelitian 2016

Pada gambar 2.13 diatas, merupakan bentuk fisik dari kabel *GPIB* yaitu sebuah kabel *interface* yang berfungsi sebagai penghubung antara mesin/*Prober* dengan *Tester*.

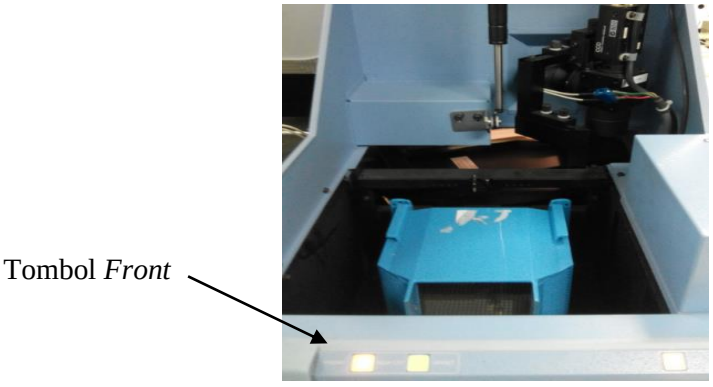
Proses Pengujian *Wafer*

Tempatkan *Wafer* yang akan diuji ke dalam sebuah wadah (*Jar*) dan letakan pada bagian *Loader* di mesin *UF200.*



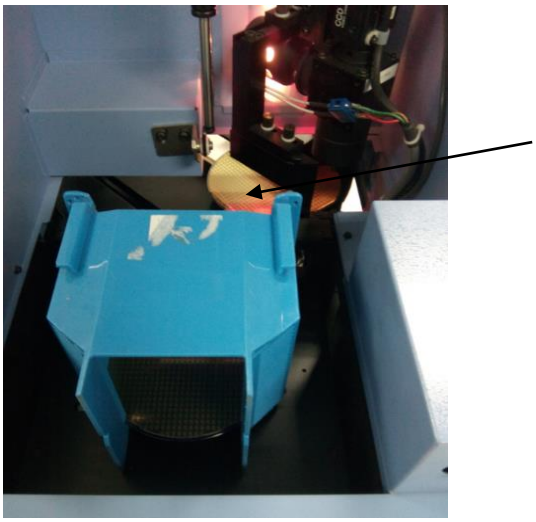
Gambar 2. 14. Bagian *Loader*
(Sumber: Data Penelitian 2016)

Setelah *Wafer* berada pada posisi bagian *Loader* mesin *UF200*, tekan tombol *Front* sehingga *Wafer* akan berada pada posisi seperti yang terlihat pada gambar 2.15 dibawah ini.



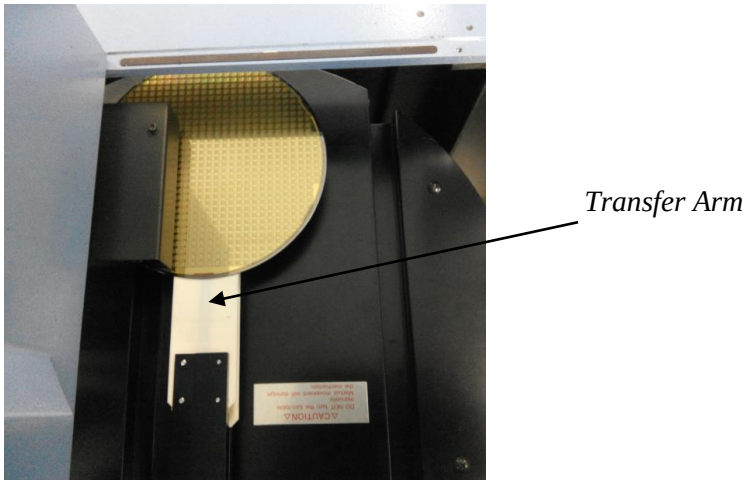
Gambar 2. 15. Tampilan Setelah Tombol *Front* di Tekan
(Sumber: Data Penelitian 2016)

Selanjutnya setelah tombol *Front* di tekan, maka *Wafer* akan diambil oleh *transfer arm,* untuk dilakukan pengecekan *ID Number.*



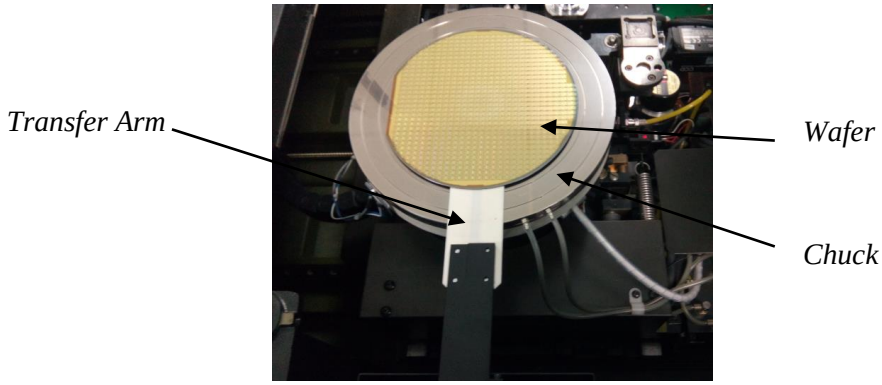
Gambar 2. 16. Tampilan Saat Pengecekan *ID Number*
(Sumber: Data penelitian 2016)

Setelah melakukan pengecekan *ID Number*, selanjutnya *wafer* akan diambil lagi oleh *transfer arm* untuk diletakan pada bagian *chuck.*



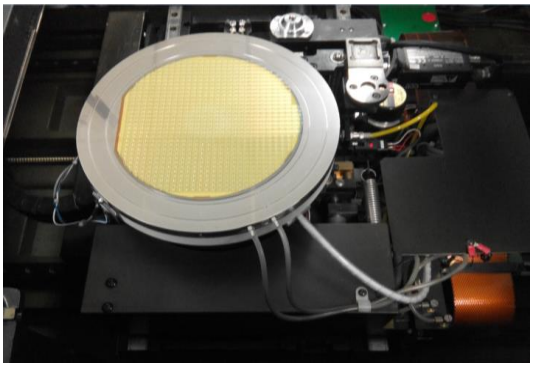
Gambar 2. 17. Tampilan Saat *Wafer* di Ambil *Transfer Arm*
(Sumber: Data Penelitian 2016)

Seperti yang terlihat pada gambar 2.18, *Transfer Arm* meletakan *Wafer* pada posisi tepat di atas permukaan *Chuck.*



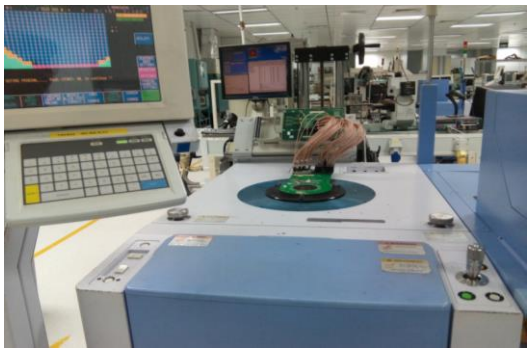
Gambar 2. 18. Tampilan Ketika *Wafer* di Letakan Pada Permukaan *Chuck*
(Sumber: Data penelitian 2016)

Selanjutnya akan dilakukan proses inisialisasi terlebih dahulu sebelum proses pengujian *wafer* dimulai.



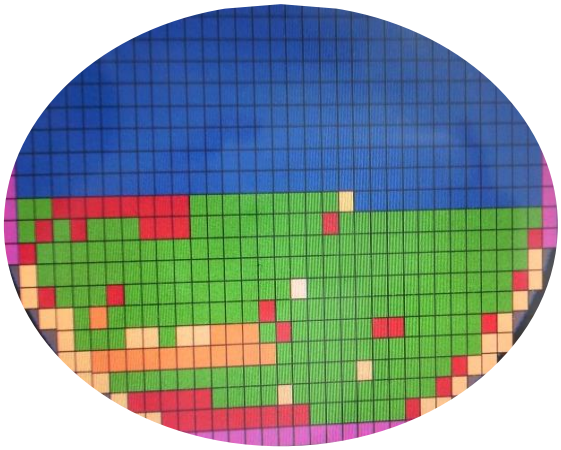
Gambar 2. 19. Proses Inisialisasi Sebelum Pengujian di Mulai
(Sumber: Data penelitian 2016)

Setelah proses inisialisasi selesai, maka proses pengujian pada *Wafer* siap untuk dijalankan.



Gambar 2. 20. Proses Pengujian Siap di Mulai
(Sumber: Data penelitian 2016)

Tampilan layar mesin *UF200* ketika proses pengujian *Wafer Probe* dilakukan



Gambar 2. 21. Tampilan Layar Saat Proses Pengujian *Wafer* Berlangsung
(Sumber: Data penelitian 2016)

Pada gambar 2.21 diatas adalah tampilan saat dilakukannya proses pengujian *wafer* yang tampak pada mesin *Prober UF200.* Seperti yang tampak pada gambar diatas, warna hijau menandakan bahwa hasil dari pengujian *Wafer* adalah *Pass* (*Good*) sedangkan warna merah menandakan bahwa hasil dari pengujian *Wafer* adalah *Fail* (*Reject*).

2.3. *Software* Pendukung

Software pendukung merupakan beberapa perangkat lunak yang digunakan untuk mendukung pembuatan sistem pakar didalam penelitian. Perangkat lunak tersebut antara lain: *Mozila, Xampp* dan *Notepade++*. Dengan bahasa pemrograman *PHP* dan *database MySQL PhpMyAdmin*.

2.3.1. PHP

PHP adalah singkatan dari *PHP: Hypertext Prepocessor*, yaitu bahasa pemrograman yang digunakan untuk membuat halaman web dinamis dan juga bisa digunakan bersamaan dengan HTML. PHP diciptakan oleh Rasmus Lerdorf pada tahun 1994. Pada awalnya pengembangannya PHP adalah singkatan dari *Personal Home Page* (Antonius Nugraha Widhi Pratama, 2010:9).

Gambar 2. 22. Logo PHP
(Sumber: http://php.net)

2.3.2. *Notepad++*

Notepad++ adalah sebuah *Editor* pemrograman yang mendukung banyak fitur yang memudahkan proses penulisan kode dari duplikasi baris (Ctrl+D), pemindahan baris ke atas (Ctrl+Shift+Up Arrow) atau ke bawah (Ctrl+Shift+Down Arrow), pelipatan blok kode, hingga pembandingan isi berkas program (Antonius Nugraha Widhi Pratama, 2010:5).

Gambar 2. 23. Logo Notepad++
(Sumber: Aplikasi)

2.3.3. *Mozilla Firefox*

Mozilla Firefox merupakan aplikasi *web browser* gratis yang dikembangkan oleh yayasan *Mozilla* dan beberapa *developer* pendukungnya. Pembuatan *Firefox* ditujukan untuk mengembangkan sebuah *web browser* yang kecil, cepat, simpel dan masih terbuka untuk dikembangkan karena sudah terpisah dari *Mozilla Suite* yang lebih besar (Aunurrofiq Mansur, 2010:2).

Gambar 2. 24. Logo *Mozilla Firefox*
(Sumber: Aplikasi)

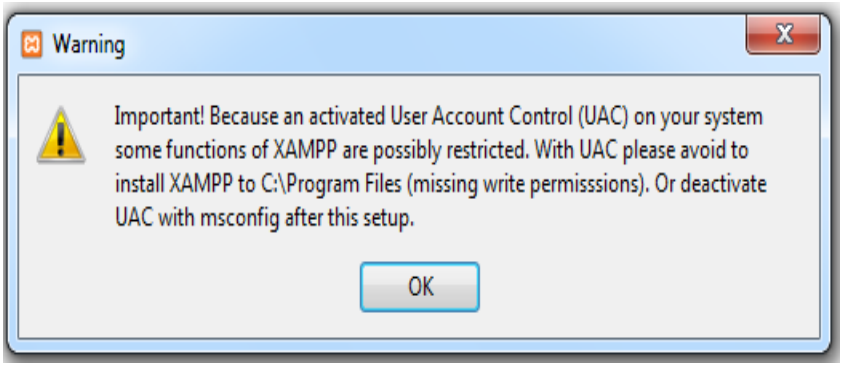
2.3.4. XAMPP

XAMPP adalah sebuah *Software* yang berfungsi untuk menjalankan *Website* berbasis PHP dan menggunakan pengolah data MySQL di komputer lokal, dan berperan sebagai *Server Web*. XAMPP disebut juga sebuah CPanel *server virtual*, yang dapat membantu melakukan preview sehingga dapat memodifikasi *W*ebsite tanpa harus *Online* (Wicaksono & SmitDev, 2008:7).

Gambar 2. 25. Logo XAMPP
(Sumber: Aplikasi)

Berikut ini akan dijelaskan bagaimana cara *Install XAMPP V5.5.3*:

1. *Download* aplikasi XAMPP di internet atau bisa *download* langsung pada situs resminya www.apachefriend.org
2. *Double* klik aplikasi XAMPP yang baru anda download tadi, secara otomatis akan muncul sebuah jendela *User Account Control*, klik *Yes.*
3. Akan muncul sebuah jendela *Warning* seperti yang terlihat pada gambar dibawah, langsung saja klik *Ok.*

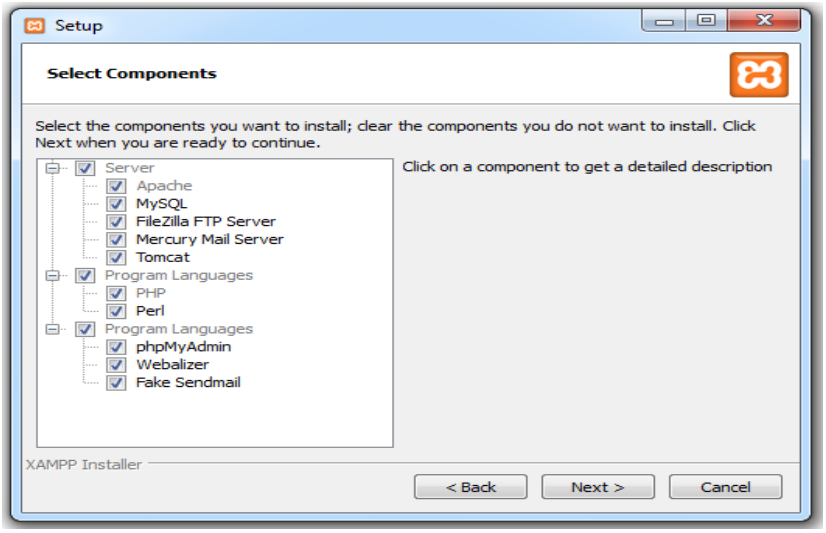


Gambar 2. 26. *Warning*
(Sumber: Data Penelitian 2016)

4. Maka akan muncul tampilan baru seperti yang terlihat dibawah, klik *Next* untuk melanjutkan.

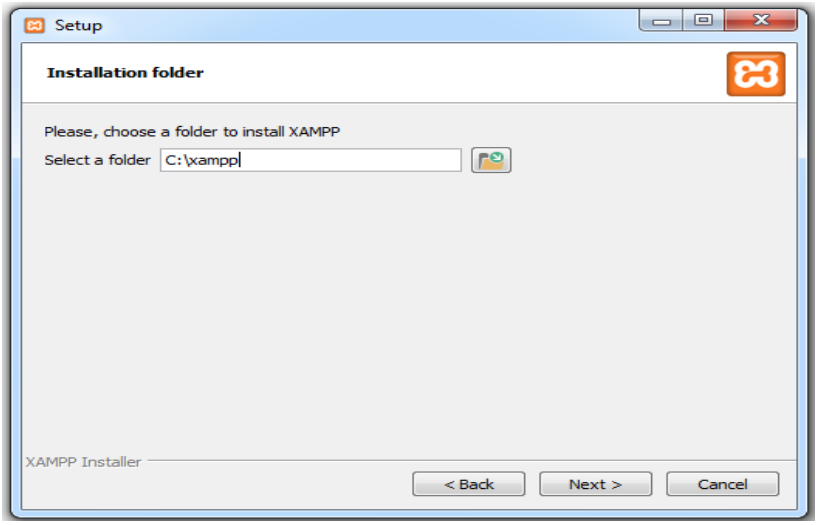
Gambar 2. 27. Jendela *Setup*
(Sumber: Data Penelitian 2016)

5. Setelah itu akan muncul tampilan *Select Components*, disini kita bisa memilih komponen apa saja yang akan di *Install*, klik *Next* untuk melanjutkan.



Gambar 2. 28. *Select Component*
(Sumber: Data Penelitian 2016)

6. Pada tampilan *Installation Folder*, disini kita bisa memilih *Folder* atau lokasi dimana *File* akan disimpan. Secara *default file* akan disimpan di *directory* c:\xampp. Klik *Next* untuk melanjutkan.

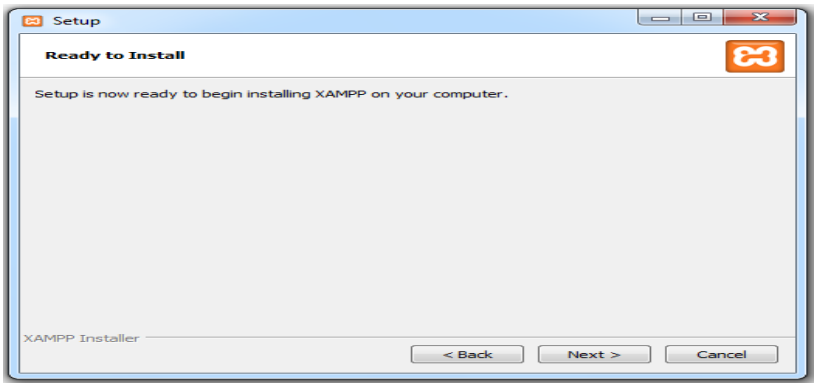


Gambar 2. 29. *Installation Folder*
(Sumber: Data Penelitian 2016)

7. Pada jendela baru yang muncul, hilangkan tanda cek pada tulisan *Learn more about Bitnami for XAMPP*. Klik *Next* untuk melanjutkan.

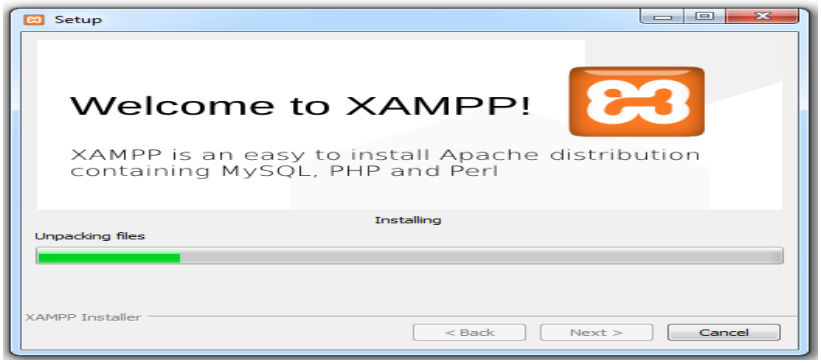
Gambar 2. 30. *Bitnami For XAMPP*
(Sumber: Data Penelitian 2016)

8. Muncul tampilan *Ready to Install*. Klik *Next* untuk memulai proses instalasi.



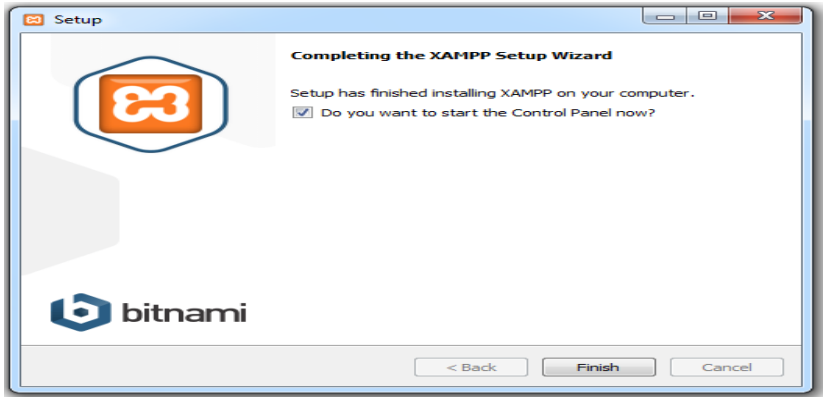
Gambar 2. 31. *Ready to Install*
(Sumber: Data Penelitian 2016)

9. Berikut adalah saat dimana proses instalasi sedang berjalan, bisa dilihat pada gambar 2.32 dibawah.



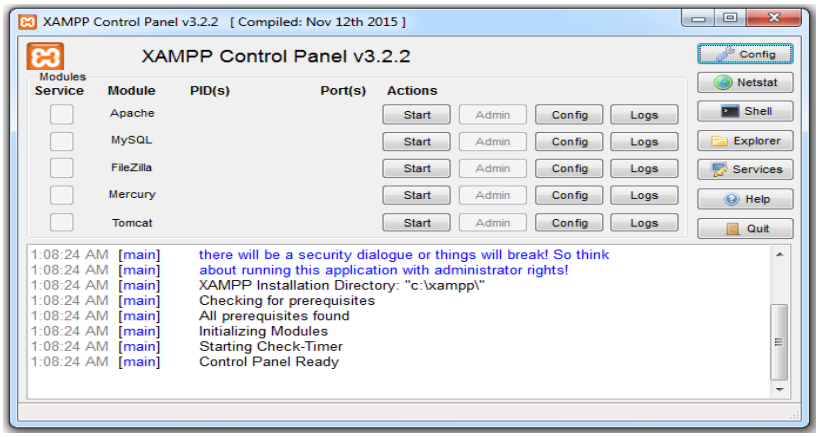
Gambar 2. 32. Proses Instalasi Sedang Berjalan
(Sumber: Data Penelitian 2016)

10. Setelah instalasi selesai, langsung saja klik *Finish.*



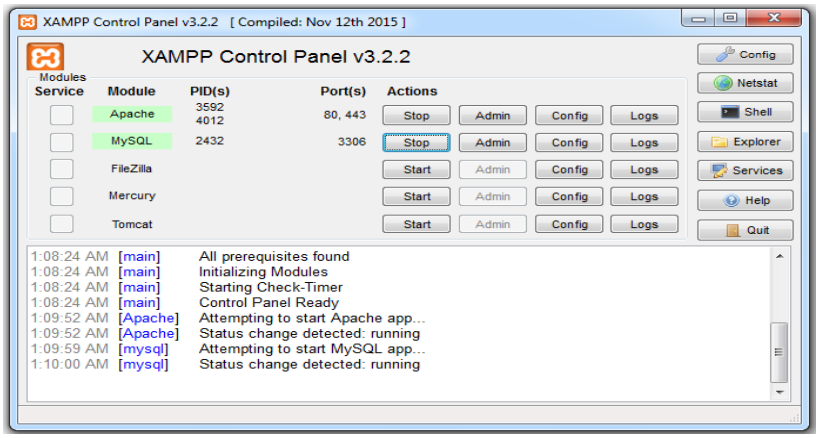
Gambar 2. 33. Proses Instalasi Selesai
(Sumber: Data Penelitian 2016)

11. Berikut adalah tampilan awal dari XAMPP.



Gambar 2. 34. Tampilan Awal XAMPP
(Sumber: Data Penelitian 2016)

12. Tampilan setelah *Apache* dan *MySQL* diaktifkan, dengan cara klik tombol *Start* didalam XAMPP *Control Panel*



Gambar 2. 35. *Apache* dan *MySQL* Aktif
(Sumber: Data Penelitian 2016)

2.3.5. *phpMyAdmin*

phpMyAdmin adalah sebuah *Software* berbasis pemrograman PHP yang dipergunakan sebagai *administrator* MySQL melalui *Browser* (*Web*) yang digunakan untuk manajemen *Database*, dan mendukung berbagai aktivitas MySQL

seperti pengelolaan data, tabel, relasi antar tabel dan sebagainya (Su Rahman, 2013:21).

Gambar 2. 36. Logo *phpMyAdmin*
(Sumber: Aplikasi)

2.3.6. *MySQL*

MySQL merupakan salah satu *Database* kelas dunia yang sangat cocok bila dipadukan dengan bahasa *Pemrograman PHP*. *MySQL* bekerja menggunakan bahasa *SQL* (*Structure Query Language*) yang merupakan bahasa standar yang digunakan untuk manipulasi *Database* (Saputra, 2012:77). Pada umumnya, perintah yang paling sering digunakan dalam *MySQL* adalah *SELECT* (mengambil), *INSERT* (menambah), *UPDATE* (mengubah), dan *DELETE* (menghapus). Selain itu, *SQL* juga menyediakan perintah untuk membuat *Database*, *Field*, ataupun *Index* untuk menambah atau menghapus data.

Gambar 2. 37. Logo *MySQL*
(Sumber: https://www.mysql.com)

2.3.7. CSS

CSS merupakan singkatan dari *Cascading Style Sheet* yaitu bahasa pemrograman *Web* yang didesain khusus untuk mengendalikan dan membangun berbagai komponen dalam *Web* sehingga tampilan *Web* lebih rapih, terstruktur, dan seragam (Saputra, 2012:27).

Gambar 2. 38. Logo CSS
(Sumber: http://www.digitalactivist.net)

2.3.8. *JavaScript*

JavaScript adalah *Script* program berbasis *Client* yang dieksekusi oleh *Browser* sehingga membuat halaman *Web* bisa melakukan tugas-tugas tambahan yang tidak bisa dilakukan oleh *script html* biasa. Kode *JavaScript* biasanya dituliskan dalam bentuk fungsi yang ditaruh di tag <head> yang dibuka dengan tag <script type=”text/javascript”> (Zaki & Community, 2008:26).

Gambar 2. 39. Logo JavaScript
(Sumber: https://www.brandsoftheworld.com)

2.3.9. *Bootstrap*

Bootstrap adalah aplikasi *Open Source* yang berupa *Framework* atau kerangka kerja untuk membangun *Website* dinamis dengan menggunakan bahasa *script* CSS. *Bootstrap* memudahkan *Developer* dan *Designer* untuk membuat sebuah aplikasi *Web* menjadi cepat dan mudah dibandingkan membuatnya dari awal. (Sumber: https://id.wikipedia.org/wiki/PHP_Bootstrap)

Gambar 2. 40. Logo Bootstrap
(Sumber: http://getbootstrap.com)

2.3.10. StarUML

StarUML adalah *Platform* pemodelan perangkat lunak yang mendukung UML (*Unified Modelling Language*). Berdasarkan pada UML version 1.4 dan dilengkapi 11 macam diagram yang berbeda, mendukung notasi UML 2.0 dan juga mendukung pendekatan MDA (*Model Driven Architecture*) dengan mendukung konsep UML *Profile* (Triandini & Suardika, 2012:1).

Gambar 2.1. Logo StarUML
(Sumber: https://en.wikipedia.org)

2.4. Penelitian Terdahulu

Untuk mendukung teori yang berkaitan dengan penelitian, peneliti mencantumkan beberapa penelitian terdahulu di bidang sistem pakar. Penelitian terdahulu merupakan penelitian-penelitian yang dilakukan oleh para peneliti terdahulu untuk kemudian dijadikan referensi penelitian di masa yang akan datang.

Budi S. Ginting (2014), Perancangan Sistem Pakar Diagnosa Kerusakan *Blackberry Smartphone* Berbasis *Web*, aplikasi sistem pakar dibuat untuk mendiagnosa kerusakan *Blackberry* serta solusi perbaikannya dengan menggunakan penalaran *forward chaining*. Sistem pakar dibangun menggunakan bahasa pemrograman *PHP* dan *database MySQL*. Setelah dilakukan implementasi

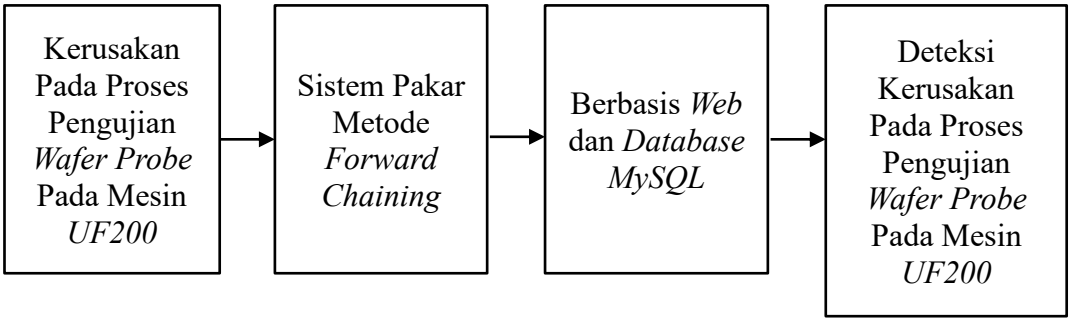
dan pengujian, sistem pakar yang dibuat bertujuan untuk menggantikan seorang pakar dalam mendiagnosa kerusakan *Blackberry*.

Dwi Agus Diartono (2009), Rancang Bangun Sistem Pakar Untuk Mendeteksi Kerusakan Pada Perangkat Keras Komputer, dengan semakin meluasnya penggunaan komputer pada masyarakat sehingga tidak sedikit yang mengalami permasalahan dengan perangkat keras ataupun perangkat lunak komputer. Penggunaan sistem pakar adalah suatu solusi untuk membantu dalam memberikan pengetahuan dari pakar kepada orang awam khususnya dalam menangani kerusakan komputer. Sistem pakar ini digunakan oleh pemakai komputer untuk membantu dalam pemahaman pendeteksian kerusakan komputer khususnya pada bagian hardisk. Hasil solusi dari sistem pakar ini bisa dimanfaatkan untuk membantu *user* dalam menangani kerusakan harddisk komputer atau jenis kerusakan lain yang dialami komputer.

Cholil Jamhari, **Agus Kiryanto** dan **Sri Huning Anwariningsih (2014),** Sistem Pakar Diagnosis Kerusakan Sepeda Motor Non Matic, membuat aplikasi sistem pakar dibidang otomotif yang akan menghasilkan jenis-jenis kerusakan yang terjadi pada sepeda motor serta penanganan dari kerusakan tersebut. Dengan menggunakan *Visual Studio* 2013, perencanaan sistem dalam membuat knowledge base memakai pohon keputusan dan kaidah produksi sebagai representasi pengetahuan dengan memakai metode *forward chaining*, menghasilkan jenis-jenis kerusakan yang terjadi pada sepeda motor non matic serta penanganan dari kerusakan tersebut.

Harison dan Alexyusanderia (2014), Sistem Pakar Perawatan Dan Perbaikan Ringan Mobil Bensin Menggunakan Video Tutorial Berbasis Web, merancang sistem pakar dengan metode *forward chaining* dalam memberikan peranan informasi yang sangat berarti dalam membantu masyarakat dalam menyelesaikan permasalahan yang mereka hadapi tanpa harus mengeluarkan dana yang besar. Dengan menggunakan metode pelacakan yang memulai proses pelacakan dari sekumpulan data atau fakta. Dari data-data tersebut dicari suatu kesimpulan yang menjadi solusi dari masalah yang dihadapi. Mesin inferensi mencari kaidah-kaidah dalam basis pengetahuan yang premisnya sesuai dengan data data tersebut. Kemudian dari kaidah-kaidah yang disusun tersebut diperoleh suatu kesimpulan dari permasalahan tersebut, kesimpulan ini merupakan data yang diambil dari pakar yang berupa tulisan, lisan dan video perbaikan ringan yang nantinya bisa diakses oleh pengguna.

2.5. Kerangka Pemikiran



Gambar 2.2. Kerangka Pemikiran
(Sumber: Data penelitian 2016)

Berdasarkan gambar diatas, kerusakan pada proses *Wafer Probe* pada mesin *UF200* di PT Unisem Batam ini akan dibuat sebuah program sistem pakar dengan metode *Forward Chaining*, untuk kemudian di implementasikan dengan bahasa pemrograman berbasis *web* dan *database MySQL*, yang menghasilkan sebuah *output* program berbasis *web* untuk mendeteksi kerusakan proses *wafer probe* pada mesin *UF200.*