SISTEM PAKAR DIAGNOSA KERUSAKAN *INTEGRATED CIRCUIT* (*IC*) DI MESIN *WIRE BOND* ICONN DENGAN METODE *FORWARD CHAINING* BERBASIS *WEB*
(Studi Kasus: PT UNISEM Batam)

SKRIPSI

Oleh
Sri Setiyo Purnomo
130210117

PROGRAM STUDI TEKNIK INFORMATIKA
UNIVERSITAS PUTERA BATAM
2017

SISTEM PAKAR DIAGNOSA KERUSAKAN *INTEGRATED CIRCUIT* (*IC*) DI MESIN *WIRE BOND* ICONN DENGAN METODE *FORWARD CHAINING* BERBASIS *WEB*
(Studi Kasus: PT UNISEM Batam)

SKRIPSI
Untuk memenuhi salah satu syarat
guna memperoleh gelar sarjana

Oleh
Sri Setiyo Purnomo
130210117

PROGRAM STUDI TEKNIK INFORMATIKA
UNIVERSITAS PUTERA BATAM
2017

PERNYATAAN

Dengan ini saya menyatakan bahwa:

1. Skripsi ini adalah asli dan belum pernah diajukan untuk mendapatkan gelar akademik (sarjana, dan/atau magister), baik di Universitas Putera Batam maupun di perguruan tinggi lain.
2. Skripsi ini adalah murni gagasan, rumusan, dan penelitian saya sendiri, tanpa bantuan pihak lain, kecuali arahan pembimbing.
3. Dalam skripsi ini tidak terdapat karya atau pendapat yang telah ditulis atau dipublikasikan orang lain, kecuali secara tertulis dengan jelas dicantumkan sebagai acuan dalam naskah dengan disebutkan nama pengarang dan dicantumkan dalam daftar pustaka.
4. Pernyataan ini saya buat dengan sesungguhnya dan apabila di kemudian hari terdapat penyimpangan dan ketidakbenaran dalam pernyataan ini, maka saya bersedia menerima sanksi akademik berupa pencabutan gelar yang telah diperoleh, serta sanksi lainnya sesuai dengan norma yang berlaku di perguruan tinggi.

Batam, 14 Februari 2017

Yang membuat pernyataan,

Materai

Sri Setiyo Purnomo
NPM: 130210117

SISTEM PAKAR KERUSAKAN *INTEGRATED CIRCUIT* (*IC*) DI MESIN *WIRE BOND* ICONN DENGAN METODE *FORWARD CHAINING* BERBASIS *WEB* (Studi Kasus: PT UNISEM Batam)

Oleh
Sri Setiyo Purnomo
130210117

SKRIPSI
Untuk memenuhi salah satu syarat
guna memperoleh gelar Sarjana

Telah disetujui oleh Pembimbing pada tanggal
seperti tertera di bawah ini

Batam, 14 Februari 2017

Anggia Dasa Putri, S.Kom., M.Kom.
Pembimbing

KATA PENGANTAR

Syukur Alhamdulillah penulis panjatkan kehadirat Allah SWT yang telah melimpahkan segala rahmat dan karuniaNya, sehingga penulis dapat menyelesaikan laporan tugas akhir yang merupakan salah satu persyaratan untuk menyelesaikan program studi strata satu (S1) pada Program Studi Teknik Informatika Universitas Putera Batam.

Penulis menyadari bahwa skripsi ini masih jauh dari sempurna. Karena itu, kritik dan saran akan senantiasa penulis terima dengan senang hati. Dengan segala keterbatasan, penulis menyadari pula bahwa skripsi ini takkan terwujud tanpa bantuan, bimbingan, dan dorongan dari berbagai pihak. Untuk itu, dengan segala kerendahan hati, penulis menyampaikan ucapan terima kasih kepada:

1. Rektor Universitas Putera Batam.
2. Ketua Program Studi Bapak Andi Maslan, S.T., M.SI.
3. Ibu Anggia Dasa Putri, S.Kom., M.Kom. selaku pembimbing Skripsi pada Program Studi Teknik Informatika Universitas Putera Batam.
4. Dosen dan Staff Universitas Putera Batam.
5. Bapak Moch. Ali Rochman S.T. Selaku Nara sumber penelitian.
6. Kedua orang tua penulis yang telah melahirkan dan membesarkan penulis sehingga bisa berguna bagi agama, bangsa dan negara.
7. Istri penulis yang tidak pernah lelah memberikan semangat pantang menyerah untuk terus maju.

8. Anakku M. Zaky Purnomo yang pengertian dan baik dengan orang tua, jadilah anak sholeh yang bisa membanggakan keluarga.
9. Mas Dedi yang bersedia mengajarkan dan membimbing pembuatan program dalam penyusunan skripsi.
10. Seluruh PE, teknisi dan operator di PT UNISEM Batam, sukses selalu dan menjadi lebih baik.
11. Seluruh teman-teman seperjuangan selama kuliah yang sangat luar biasa, selalu maju dan berjuang menggapai cita-cita.

Semoga Allah SWT membalas setiap butir kebaikan dengan tujuh kali kebaikan dan keberkahan di dunia dan akhirat, amin.

Batam, Februari 2017

Penulis

ABSTRAK

PT United Semiconductor Malaysia (UNISEM) Batam adalah perusahaan yang bergerak dibidang semikonduktor. Perusahaan memproduksi barang berupa IC (*Integrated Circuit*). PT. UNISEM Batam mempunyai beberapa bagian produksi salah satunya adalah departemen *Wire Bond*. Area *Wire Bond* adalah area produksi yang bertugas menghubungkan pin dari *micro chip* ke kaki pin *lead frame*/kaki IC. Semua pekerjaan dikerjakan dengan mesin *wire bond*, salah satunya adalah mesin KNS IConn. PT. UNISEM Batam harus dapat menghasilkan produk yang berkualitas tinggi agar mamapu bersaing dengan pesaingnya di seluruh dunia. Semenjak PT UNISEM Batam menerapkan sistem kontrak dengan pekerjanya, maka kekurangan tenaga kerja teknisi senior memberikan dapak pada *quality product* di area produksi *Wire Bond*. Penelitian ini dilakukan berdasarkan kebutuhan area produksi *Wire Bond* PT UNISEM Batam akan alat bantu teknisi dalam menganalisa permasalah di mesin *Wire Bond* KNS seri IConn, sehingga dapat meningkatkan kualitas produksi dan menyingkat *downtime* produksi. Alat bantu tersebut berupa sistem pakar berbasis *web*. Sistem pakar ini sebagai alat bantu untuk mendiagnosis kerusakan unit produksi berupa *Integrated Circuit* (*IC*) dan juga memberikan solusi permasalahan yang timbul pada mesin *Wire Bond* IConn. Untuk mengimplementasikan sistem pakar dibutuhkan data dari seorang pakar mesin *Wire Bond* sebagai *knowledge base* dan mesin inferensi yang berfungsi untuk mengolah aturan menggunakan metode *forward chaining* sehingga pengetahuan seorang pakar dapat diterapkan pada sistem pakar. Dari *knowledge base* dan mesin inferensi yang sudah disusun maka dibuat *user interface* menjadi sebuah program berbasis *web*. Konsultasi pengguna dengan sistem pakar dilakukan melalui menjawab setiap pertanyaan dengan jawaban Ya atau Tidak sesuai dengan gejala kerusakan yang terjadi. Keluaran sistem pakar berupa nama kerusakan, gejala kerusakan, penyebab kerusakan dan solusi permasalahan pada mesin *Wire Bond* IConn PT UNISEM Batam. Aplikasi sistem pakar yang dibuat mempunyai keakuratan yang baik yaitu dengan prersentase keakuratan sebesar 90%. Dengan tingkat keakurasian yang baik diharapkan sistem pakar ini dapat membantu teknisi dalam menganalisa kerusakan *IC* di mesin *Wire Bond* IConn PT UNISEM Batam sehingga dapat meningkatkan kualitas dan menyingkat *downtime* produksi.

Kata kunci: sistem pakar, diagnosa kerusakan, mesin *Wire Bond*, KNS IConn, *forward chaining*.

ABSTRACT

PT. United Semiconductor Malaysia (UNISEM) Batam is a company engaged in the semiconductor. Companies producing goods such as IC (Integrated Circuit). PT UNISEM Batam has several productions sections one of which is a department of Wire Bond. Wire Bond area is in charge of the production area of micro chip connecting pin to pin lead frame leg / pin IC. All work is done by wire bond machines, one of which is the machine KNS IConn. PT UNISEM Batam should be able to produce high quality products in order can competed with its competitors around the world. Since the PT UNISEM Batam implement the contracts system with its workers, the labor shortage of senior technicians provides impact on product quality in the Wire Bond production area. This study was conducted based on the needs of the production area Wire Bond PT Unisem Batam technicians will be tools in analyzing the problems in the machine Wire Bond KNS series IConn, so as to improve production quality and shorten production downtime. That tool as expert system with a web based. This expert system as atool for diagnosing damaged production unit like Integrated Circuit (IC) and also provide problem solutions on the Wire Bond IConn machine. To implemen the expert system takes data from Wire Bond engine experts as a knowledge base and the inference engine thet serves to cultivate of rule using forward chaining methode so that an expert knowledge can be applied to expert systems. From the knowledge base and the inference engine that has been prepared then made the user interface into a web-based program. Consultation users with expert system is done through aswering each question with the answer is Yes or No to the symptoms of the damaged. Expert system output in the form of damage name, symptoms damage, cause damaged and solutions to problem on the Wire Bond machine machine in PT UNISEM Batam. Application of expert system created has good accuracy prersentase with an accuracy of 90%. With a good level of accuracy expected this expert system can help the technician in analyzing IC damaged on Wire Bond IConn machine in PT Unisem Batam so as to quality improve and shorten production downtime.

Keywords*: expert systems, damage diagnose, Wire Bond machine, KNS IConn, forward chaining*

.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang Penelitian

PT United Semiconductor Malaysia (UNISEM) Batam adalah perusahaan yang bergerak dibidang semikonduktor. Perusahaan memproduksi barang berupa *IC* (*Integrated Circuit*). *IC* (*Integrated Circuit*) adalah komponen elektronik semi konduktor yang merupakan gabungan dari ratusan atau ribuan komponen-komponen lain (Hakiem, 2015:22). PT UNISEM Batam mempunyai beberapa bagian produksi yaitu departemen *FOL* (*Front of Line*) yang terdiri dari *Wafer Mount*, *Wafer Saw*, 2nd *Optical*, *Die Attach*, *Wire Bond*, dan 3rd *Optical*; sedangkan departemen *EOL* (*End of Line*) terdiri dari *Molding, PM Cure, Silver Platting, Post PLBlake, Bottom Mark, Top Mark,* 4th *Optical, Formsingulation, Final Visual Inspection, Packing, Shipping.*

Area *Wire Bond* pada PT UNISEM Batam merupakan area produksi yang bertugas menghubungkan pin dari *micro chip* ke kaki pin *lead frame*/kaki *IC*. Semua pekerjaan dikerjakan dengan mesin *Wire Bond* yang terdiri dari beberapa jenis, yaitu: mesin KNS, mesin ASM, dan mesin ESEC. Dari beberapa mesin tersebut yang paling sering digunakan adalah mesin KNS. Mesin KNS dengan seri IConn merupakan mesin keluaran terbaru dibanding dengan yang lainnya. Terdapat 3 *line* mesin *Wire Bond* pada PT UNISEM Batam yang menggunakan mesin KNS dengan jumlah 72 mesin.

Untuk menghasilkan unit yang bagus dibutuhkan *setup* mesin dan penanganan kesalahan yang tepat dan cepat. Tugas teknisi dituntut untuk lebih berhati-hati dan tanggap terhadap mesin *Wire Bond*. Teknisi yang senior dibutuhkan untuk melakukan perbaikan terhadap mesin dengan kerusakan yang berat, supaya kerusakan *IC* yang diproduksi tidak bertambah banyak. Jumlah teknisi *Wire Bond* KNS sebelumnya 5 teknisi tiap *shift* yang salah satunya adalah senior teknisi (*leader*). Senior teknisi bertugas membantu teknisi lainnya untuk memperbaiki mesin dengan tigkat kerusakan yang serius. Tapi semenjak adanya efisiensi tenaga kerja oleh perusahaan, maka jumlah teknisi *Wire Bond* menjadi 3 orang tiap *shift*. Senior teknisi yang biasanya membantu teknisi lain yang kesulitan, sekarang hanya menangnani 1 *line* mesin dengan jumlah mesin sebanyak 24 unit dan tidak sempat untuk membantu teknisi lainnya. PT UNISEM Batam menerapkan sistem kontrak kerja selama 1 atau 2 tahun pada teknisi. Sistem kontrak kerja pada teknisi menyebabkan sering bergantinya teknisi baru yang menangani mesin.

Laporan kerusakan *IC* yang diproduksi, dibuat tiap minggunya. Lima *top defect* pada area produksi *FOL* (*Front of Line*), tiga diantaranya kerusakan unit yang berasal dari area *Wire Bond* dan dua berasal dari area *Die Attach*. Kontribusi *reject* terbanyak ada pada area *Wire Bond* yaitu *damaged wire*, *ball non stick*, dan *lifted wedge*. Kerusakan yang terjadi bisa disebabkan dari beberapa faktor, yaitu kerusakan karena mesin, kesalahan *handling*, ataupun kerusakan karena meterial.

Sistem pakar adalah sistem komputer yang ditujukan untuk meniru semua aspek (*emulates*) kemampuan pengambilan keputusan (*decision making*) seorang pakar (Rosnelly, 2012: 2). Sistem pakar bekerja dengan mengadopsi pengetahuan

yang ada pada seorang pakar dan disimpan dalam komputer. Informasi yang sudah didapatkan dari pakar, akan disimpan dan nantinya akan digunakan sebagai acuan dalam proses pengambilan keputusan suatu permasalahan yang dihadapi pengguna.

Wilson (1998) *forward chaining* berarti menggunakan aturan kondisi-aksi. Dalam metode ini, data digunakan untuk menentukan aturan mana yang akan dijalankan, kemudian aturan tersebut dijalankan. Giarattanto dan Riley (1994) berpendapat bahwa metode *forward chaining* cocok digunakan untuk menangani masalah pengendalian (*controlling*) dan penalaran (*prognosis*). Karena itulah metode ini sangat cocok untuk digunakan sebagai penalaran dalam proses pengidentifikasian masalah sehingga ditemukan solusi terhadap permasalah tersebut (dalam Kusrini, 2008:8).

Aplikasi sistem pakar yang akan dibuat, diharapkan bisa mengadopsi pengetahuan pakar mesin *Wire Bond* IConn dan bisa digunakan sebagai acuan dalam tindakan perbaikan mesin. Dari latar belakang tersebut dapat diangkat judul penelitian yaitu: "**SISTEM PAKAR DIAGNOSA KERUSAKAN *INTEGRATED CIRCUIT* (*IC*) DI MESIN *WIRE BOND* ICONN DENGAN METODE *FORWARD CHAINING* BERBASIS *WEB***".

1.2 Identifikasi Masalah

Berdasarkan uraian latar belakang diatas, maka permasalahan dapat diidentifikasikan sebagai berikut ini:

1. Mesin *Wire Bond* IConn adalah mesin yang paling banyak menghasilkan *reject unit IC* pada area *Wire Bond* PT UNISEM Batam.

2. Sistem kontrak kerja pada teknisi baru PT UNISEM Batam menyebabkan sedikitnya jumlah teknisi yang berpengalaman dengan mesin *Wire Bond* IConn, sehingga teknisi baru kurang optimal dalam melakukan tindakan perbaikan terhadap kerusakan *IC* pada mesin *Wire Bond* IConn PT UNISEM Batam.
3. Berkurangnya jumlah teknisi mesin *Wire* Bond KNS IConn di PT UNISEM Batam yang sebelumnya 5 orang menjadi 3 orang menyebabkan teknisi senior tidak bisa membantu teknisi lainnya dalam bekerja dan menyebabkan tingginya *downtime* mesin di PT UNISEM Batam.
4. Belum tersedianya sistem pakar yang dapat digunakan oleh teknisi untuk membantu dalam menganalisa permasalahan yang berkaitan dengan mesin *Wire Bond* IConn di PT UNISEM Batam.

1.3 Pembatasan Masalah

Agar masalah tidak melebar, maka dalam penelitian ini terdapat batasan-batasan masalah sebagai berikut:

1. Penelitian dilakukan di bagian produksi area *Wire Bond* PT UNISEM Batam.
2. Penelitian dilakukan untuk mendeteksi kerusakan *IC* di mesin *Wire Bond* bertipe KNS seri IConn.
3. Penelitian ini menggunakan metode *forward chaining* (pelacakan maju).
4. Pakar yang akan diambil sebagai acuan sumber data adalah Moch. Ali Rochman, S.T. yaitu seorang *Process Engineering* yang berpengalaman selama lebih dari 6 tahun menangani dan menganalisa kerusakan *IC* di area *Wire Bond* PT UNISEM Batam.

5. Sistem pakar yang dibuat berbasis *web* yang ditulis menggunakan bahasa pemrograman *HTML*, *PHP*, *CSS*, *JavaScript*, *jQuery* dengan editor *Notepad++*.
6. Implementasi sistem pakar ini hanya sampai pada *server* lokal menggunakan aplikasi *XAMPP* yang bisa diakses dari beberapa komputer yang ada di area *Wire Bond* dan menggunakan *database phpMyAdmin*.

1.4 Perumusan Masalah

Berdasarkan pembatasan masalah yang telah diuraikan, dapat dirumuskan masalah, bagaimana mengimplementasikan aplikasi sistem pakar untuk mendeteksi kerusakan *Integrated Circuit* (*IC*) di mesin *Wire Bond* IConn dengan metode *forward chaining* berbasis *web*?

1.5 Tujuan Penelitian

Secara umum, tujuan penelitan ada tiga macam yaitu yang bersifat penemuan, pembuktian, dan pengembangan (Sugiyono, 2014: 3). Adapun penelitian ini dilakukan dengan tujuan untuk mengimplementasikan aplikasi sistem pakar untuk mendiagnosis gejala kerusakan *IC* di mesin *Wire Bond* IConn dengan menggunakan metode *forward chaining* berbasis *web* di PT UNISEM Batam.

1.6 Manfaat Penelitian

Penelitian ini dilakukan dengan harapan supaya memberikan beberapa manfaat antara lain:

1. Sebagai sumber informasi yang dibutuhkan teknisi untuk menyelesaikan masalah kerusakan *IC* di mesin *Wire Bond* IConn PT UNISEM Batam.
2. Dengan adanya sistem pakar sebagai sumber informasi untuk teknisi, maka dapat mempermudah kerja teknisi dan menyingkat *downtime* mesin produksi di PT UNISEM Batam.
3. Hasil penelitian dapat menjadi bahan kajian perbandingan dan referensi dalam pengembangan sistem pakar di mesin produksi yang lainnya.

BAB II
TINJAUAN PUSTAKA

2.1 Teori Dasar

Deskripsi teori dalam penelitian merupakan uraian sistematis tentang teori (bukan sekedar pendapat pakar atau penulis) dan hasil-hasil penelitian yang relevan dengan variabel yang diteliti (Sudaryono, 2015: 15).

Pada bab ini memuat beberapa teori dasar tentang cabang ilmu kecerdasan buatan (*Artificial Intelligence*), beberapa teori yang mendukung penelitian dan teori dasar yang mendukung pembuatan program sistem pakar.

2.1.1 Kecerdasan Buatan

Kecerdasan buatan (*Artificial Intelligence*) 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 (Husda, 2012:192).

Cerdas berarti memiliki pengetahuan, pengalaman, dan penalaran untuk membuat keputusan dan mengambil tindakan. Untuk membuat sebuah mesin menjadi cerdas (dapat bertindak seperti manusia) maka harus diberi bekal pengetahuan dan diberi kemampuan untuk menalar. Kecerdasan buatan memungkinkan komputer untuk berpikir atau menalar dan menirukan proses belajar manusia sehingga informasi baru dapat diserap sebagai pengetahuan, pengalaman,

dan proses pembelajaran serta dapat digunakan sebagai acuan di masa-masa yang akan datang (Sutojo, dkk., 2011: 3).

Menurut Husda (2012: 196-198), sistem kecerdasan buatan yang banyak dikembangkan saat ini adalah:

1. Sistem Pakar (*Expert System)*

   Sistem pakar yaitu program konsultasi (*advisory*) yang mencoba menirukan proses penalaran seorang pakar/ahli dalam memecahkan masalah yang rumit.

2. Pemrosesan Bahasa Alami (*Natural Language Processing*)

   Pemrosesan Bahasa Alami (*Natural Language Processing*) yang memberi kemampuan pengguna komputer untuk berkomunikasi dengan komputer dalam bahasa mereka sendiri (bahasa manusia).

3. Pemahaman Ucapan/Suara (*Speech/Voice Understanding*)

   Adalah teknik agar komputer dapat mengenali dan memahami bahasa ucapan.

4. Sistem Sensor dan Robotika

   Sistem sensor, seperti sistem visi dan pencitraan, serta sistem pengolahan sinyal, merupakan bagian dari robotika. Sebuah robot, yaitu perangkat elektromekanik yang diprogram untuk melakukan tugas manual, tidak semuanya merupakan bagian dari *AI*. Robot yang cerdas biasanya mempunyai perangkat sensor, seperti kamera, yang mengumpulkan informasi mengenai operasi dan lingkungannya. Kemudian bagian *AI* robot tersebut menerjemahkan informasi tadi dan merespon serta beradaptasi jika terjadi perubahan lingkungan.

5. Komputer Visi

   Merupakan kombinasi dari pencitraan, pengolahan citra, pengenalan pola serta

proses pengambilan keputusan. Tujuan utama dari komputer visi adalah untuk menerjemahkan suatu pemandangan.

6. *Intelligent Tutoring/Intelligent Computer-Aided Instruction*

   Merupakan komputer yang mengajari manusia. Dengan menambahkan aspek kecerdasan di dalam komputer, dapat tercipta komputer "guru" yang dapat mengatur teknik pengajarannya untuk menyesuaikan dengan kebutuhan "murid" secara individiual.

7. Mesin Belajar (*Machine Learning*)

   Yang berhubungan dengan sekumpulan metode untuk mencoba mengajari/melatih komputer untuk memecahkan masalah atau mendukung usaha pemecahan masalah dengan menganalisa kasus-kasus yang telah terjadi. Dua metode mesin belajar yang paling populer adalah Komputasi *Neural* dan Logika Samar.

8. Aplikasi lain dari *AI* misalnya untuk merangkum berita, pemrograman komputer secara otomatis, atau menerjemahkan dari suatu bahasa ke bahasa yang lain, serta aplikasi dalam permainan (Ingat pertandingan catur antara Grand Master Anatoly Karpov dengan komputer *Deep Thought* dari IBM).

Persoalan-persoalan yang ditangani oleh kecerdasan buatan makin lama makin berkembang sehingga memungkinkan bagi kecerdasan buatan untuk merambah ke bidang ilmu yang lainnya (Sutojo, dkk., 2011: 12). Beberapa sub disiplin ilmu dalam kecerdasan buatan, antara lain:

1. Logika *fuzzy* (*fuzzy logic*)

   Logika *fuzzy* (*fuzzy logic*) adalah metodologi sistem kontrol pemecahan masalah, yang cocok 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. Metode ini dapat diterapkan pada perangkat keras, perangkat lunak, atau kombinasi keduanya (Sutojo, dkk., 2011: 211).

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

a. Metode Tsukamoto

   Dalam inferensinya, metode Tsukamoto menggunakan tahapan yaitu: *fuzzifikasi*, pembentukan basis pengetahuan *fuzzy* (*rule* dalam bentuk *IF…THEN*), mesin inferensi menggunakan fungsi implikasi MIN (*Minimum*), dan *defuzzifikasi* menggunakan metode Rata-rata (*Average*).

b. Metode Mamdani

   Metode Mamdani menggunakan operasi MIN-MAX atau MAX-PRODUCT. Terdapat 4 tahapan untuk mendapatkan *output* pada metode Mamdani, yaitu: *fuzzifikasi*, pembentukan basis pengetahuan *fuzzy* (*rule* dalam bentuk *IF…THEN*), aplikasi fungsi implikasi menggunakan fungsi MIN (*Minimum*) dan komposisi antar-*rule* menggunakan fungsi MAX (*Maximum*) dengan menghasilkan himpunan *fuzzy* baru, dan *defuzzifikasi* menggunakan metode Centroid (Titik Tengah).

c. Metode Sugeno

   Pada Metode Sugeno, mempunyai output sistem berupa konstanta atau persamaan linear. Metode Sugeno menggunakan beberapa tahapan dalam inferensinya, yaitu: *fuzzifikasi*, pembentukan basis pengetahuan *fuzzy* (*rule* dalam bentuk *IF…THEN*), mesin inferensi menggunakan fungsi implikasi

MIN (*Minimum*), dan *defuzzifikasi* menggunakan metode Rata-rata (*Average*).

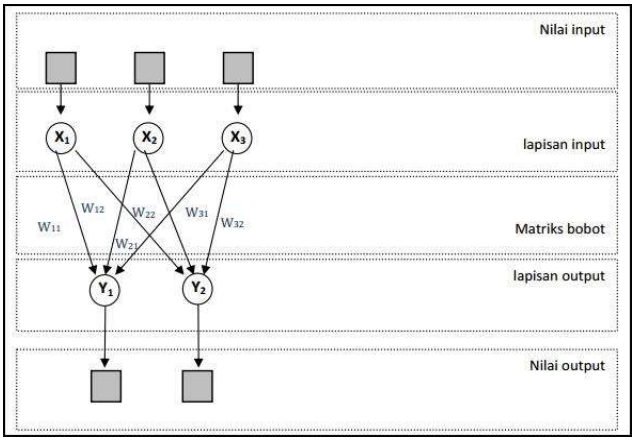
2. Jaringan syaraf tiruan (*artificial neural network*)

Jaringan saraf tiruan adalah paradigma pengolahan informasi yang terinspirasi oleh sistem saraf secara biologis, seperti proses informasi pada otak manusia. Elemen utamanya adalah struktur dari sistem pengolahan informasi yang terdiri dari sejumlah besar elemen pemrosesan yang saling berhubungan (*neuron*), bekerja serentak untuk menyelesaikan masalah tertentu. Cara kerja jaringan saraf tiruan sama seperti cara kerja manusia, yaitu belajar melalui contoh. Beberapa contoh aplikasi jaringan saraf tiruan adalah implementasi di bidang kedokteran, yaitu pemodelan dan diagnosis sistem kardiovaskular, hidung elektronik, dan dokter instan; dan implemetasi di bidang bisnis, yaitu jaringan saraf tiruan yang diintegrasikan dengan merek dagang *The Airline Marketing Tactician* (*AMT*) menggunakan *back-propagation* untuk membantu kontrol pemasaran dari alokasi kursi penerbangan (Sutojo, dkk., 2011: 283-288).

Menurut Sutojo, dkk.(2011: 292-295) tiga arsitektur jaringan yang sering digunakan dalam jaringan saraf tiruan adalah:

a. Jaringan lapisan tunggal

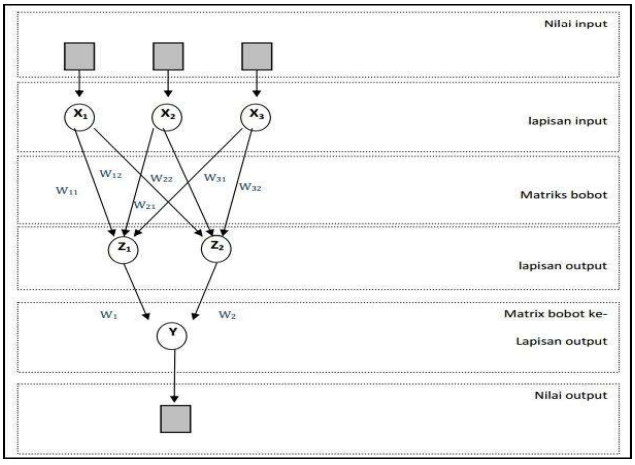
Jaringan ini terdiri dari 1 lapisan *input* dan 1 lapisan *output*, yang mana setiap unit dalam lapisan *input* selalu terhubung dengan setiap unit yang terdapat pada lapisan *output*. Berikut ini adalah gambar arsitektur jaringan lapis tunggal:



Gambar 2.1 Jaringan Saraf Dengan Lapis Tunggal
(Sumber: Sutojo, dkk., 2011: 293)

b. Jaringan lapisan banyak

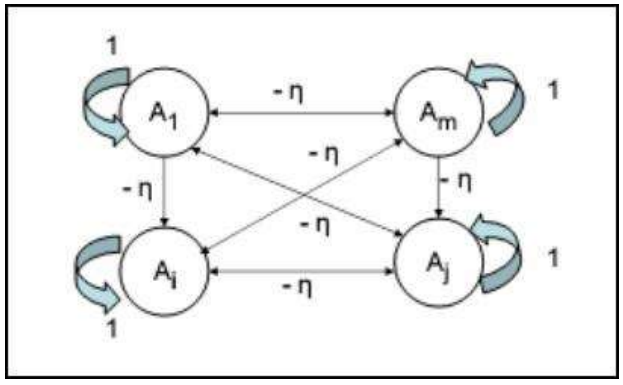
Jaringan ini mempunyai 3 jenis lapisan, yaitu lapisan *input*, lapisan tersembunyi, dan lapisan *output*. Jaringan ini dapat menyelesaikan permasalahan yang lebih kompleks dibandingkan dengan jaringan lapisan tunggal. Berikut ini adalah gambar arsitektur jaringan lapisan banyak:



Gambar 2.2 Jaringan Saraf Dengan Lapis Banyak
(Sumber: Sutojo, dkk., 2011: 294)

c. Jaringan dengan lapisan kompetitif

Jaringan ini memiliki bobot yang telah ditentukan dan tidak memiliki proses pelatihan. Jaringan ini digunakan untuk mengetahui *neuron* pemenang dari sejumlah *neuron* yang ada sehingga sekumpulan *neuron* bersaing untuk mendapatkan hak menjadi aktif.



Gambar 2.3 Jaringan Saraf Dengan Lapis Kompetitif
(Sumber: Sutojo, dkk., 2011: 295)

3. Sistem pakar (*expert system*)

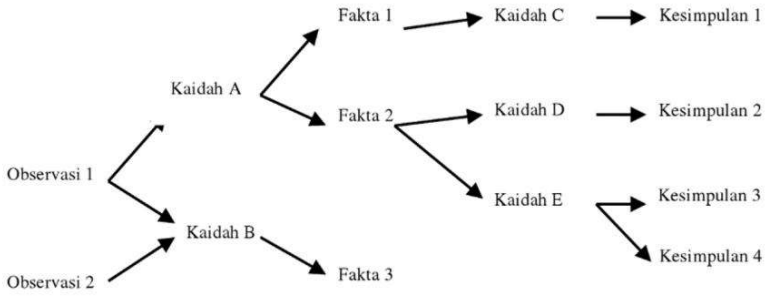
Sistem pakar adalah suatu program komputer yang mengandung pengetahuan dari satu atau lebih pakar manusia mengenai suatu bidang spesifik (Husda, 2012: 181).

Sedangkan menurut Hartati dan Iswanti (2008: 2) sistem pakar merupakan salah satu teknik kecerdasan buatan yang menirukan proses penalaran manusia.

Mesin *inferensi* dalam sistem pakar adalah bagian yang mengandung mekanisme fungsi berfikir dan pola penalaran sistem yang digunakan oleh seorang pakar. Ada dua teknik *inferensi* yang digunakan dalam sistem pakar yaitu pelacakan ke belakang (*backward chaining*) yang memulai penalaran dari kesimpulan hipotesa menuju fakta yang mengandung hipotesa tersebut. Dan

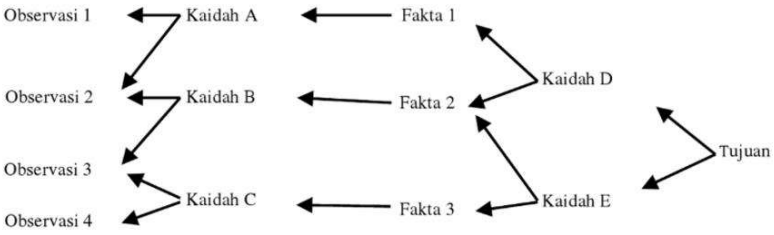
yang kedua yakni pelacakan ke depan (*forward chaining*) yang merupakan kebalikan dari pelacakan kebelakang yaitu memulai dari sekumpulan data menuju kesimpulan (Hayadi, 2016: 7).

Berikut ini adalah gambar teknik *inferensi* dengan metode *forward chaining*:



Gambar 2.4 Metode *Forward Chaining*
(Sumber: Hayadi, 2016: 7)

Berikut ini adalah gambar teknik *inferensi* dengan metode *backward chaining*:



Gambar 2.5 Metode *Backward Chaining*
(Sumber: Hayadi, 2016: 8)

Menurut Hayadi (2016:8), metode *inferensi* sistem pakar dipengaruhi oleh tiga macam teknik penelusuran yaitu:

a. *Depth-first search*: melakukan penelusuran kaidah secara mendalam dari simpul akar bergerak menurun ke tingkat dalam yang berurutan.
b. *Breadth-first search*: melakukan penelusuran dengan bergerak dari simpul akar, simpul yang ada pada setiap tingkat diuji sebelum pindah.

c. *Best-first search*: bekerja berdasarkan kombinasi kedua metode sebelumnya.

2.1.2 Sistem Pakar

Menurut Arhami, sistem pakar adalah salah satu cabang dari *Artificial Intelligence* (kecerdasan buatan) yang membuat penggunaan secara luas *knowledge* yang khusus untuk penyelesaian masalah tingkat manusia yang pakar. Menurut Marimin, sistem pakar adalah sistem perangkat lunak komputer yang menggunakan ilmu, fakta, dan teknik berfikir dalam pengambilan keputusan untuk menyelesaikan masalah-masalah yang biasanya hanya dapat diselesaikan oleh tenaga ahli dalam bidang yang bersangkutan (dalam Husda, 2012: 181-182).

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

Ada 4 tipe penjelasan yang digunakan dalam sistem pakar, yaitu (Schnupp, 1989 dalam jurnal penelitian Suwondo, 2014: 91):

1. Penjelasan mengenai jejak aturan yang menunjukkan status konsultasi.
2. Penjelasan mengenai bagaimana sebuah keputusan diperoleh.
3. Penjelasan mengapa sistem menanyakan suatu pertanyaan.
4. Penjelasan mengapa sistem tidak memberikan keputusan seperti yang dikehendaki pengguna.

Menurut Husda (2012: 182) ciri-ciri sistem pakar adalah:

1. Memiliki informasi yang handal.

2. Mudah dimodifikasi
3. Dapat digunakan dalam berbagai jenis komputer.
4. Memiliki kemampuan belajar beradaptasih.

Menurut Hayadi (2016: 5-6) sistem pakar terdiri dari beberapa konsep yang harus dimiliki. Konsep dasar dari suatu pakar sebagai berikut:

1. Keahlian

   Adalah suatu pengetahuan khusus yang diperoleh dari latihan, belajar, dan pengetahuan untuk memecahkan masalah.

2. Ahli (*Expert*)

   Melibatkan kegiatan mengenali dan memformulasikan permasalahan, memecahkan masalah secara cepat dan tepat, menerangkan pemecahannya, belajar dari pengalaman, merestrukturisasi pengetahuan, memecahkan aturan serta menentukan relevansi.

3. Mentransfer keahlian (*Transfering Expertise*)

   Adalah proses pentransferan keahlian dari seorang pakar kedalam komputer agar dapat digunakan oleh orang lain yang bukan pakar.

4. Menyimpulkan aturan (*Inferencing Rule*)

   Merupakan kemampuan komputer yang telah diprogram. Penyimpulan ini dilakukan oleh mesin inferensi yang meliputi prosedur tentang penyelesaian masalah.

5. Peraturan (*Rule*)

   Diperlukan karena mayoritas dari sistem pakar bersifat *rule based sistem*, yang berarti pengetahuan disimpan dalam bentuk peraturan.

6. Kemampuan menjelaskan (*Explanation Capability*)

   Adalah karakteristik dari sistem pakar yang memiliki kemampuan menjelaskan atau memberi saran mengapa tindakan tertentu dianjurkan atau tidak dianjurkan.

Pada dasarnya sistem pakar diterapkan untuk mendukung aktivitas pemecahan masalah. Beberapa aktivitas pemecahan yang dimaksud antara lain: pembuatan keputusan (*decision making*), pemaduan pengatahuan (*knowledge using*), pembuatan desain (*designing*), perencanaan (*planning*), prakiraan (*forecasting*), pengaturan (*regulating*), pengendalian (*controlling*), diagnosis (*diagnosing*), perumusan (*prescribing*), penjelasan (*explaining*), pemberian nasihat (*advising*) dan pelatihan (*tutoring*). Selain itu sistem pakar juga dapat berfungsi sebagai asisten yang pandai dari seorang pakar (menurut Martin dan Oxman, 1988 dalam jurnal penelitian Suwondo, 2014: 90).

Sistem pakar merupakan sebuah program komputer yang memiliki kemampuan seperti seorang pakar, ada banyak keuntungan bila menggunakan sistem pakar (Husda, 2012: 186), yaitu:

1. Menjadikan pengetahuan dan nasehat lebih mudah didapat.
2. Meningkatkan *output* dan produktifitas.
3. Menyimpan kemampuan dan keahlian seorang pakar.
4. Meningkatkan penyelesaian masalah yang khusus.
5. Meningkatkan reliabilitas.
6. Memberikan respon (jawaban) yang cepat.
7. Merupakan panduan yang cerdas.

8. Dapat bekerja dengan informasi yang kurang lengkap dan mengandung ketidakpastian.
9. Sebagai basis data cerdas.

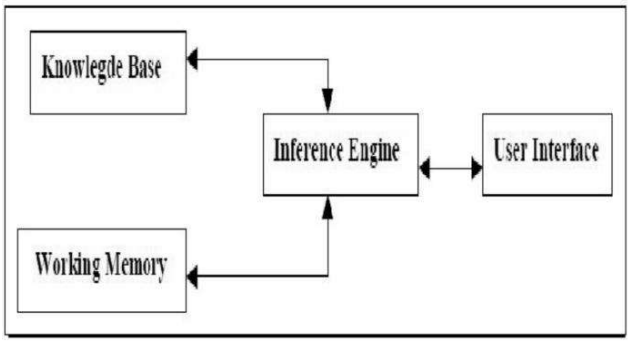
Selain mempunyai banyak keuntungan, sistem pakar juga memiliki beberapa kelemahan. Diantaranya adalah (Husda, 2012: 186-187):

1. Masalah dalam mendapatkan pengetahuan.
2. Untuk membuat suatu sistem pakar yang berkualitas tinggi sangat sulit dan memerlukan biaya yang sangat besar.
3. Boleh jadi sistem pakar tidak dapat membuat keputusan.
4. Sistem pakar tidaklah 100% menguntungkan, walaupun seseorang tetap tidak sempurna atau tidak selalu benar.

Kelemahan-kelemahan dari sistem pakar tersebut bukanlah sama sekali tidak bisa diatasi, tetapi dengan terus melakukan perbaikan dan pengolahan berdasarkan pengalaman yang telah ada maka hal itu diyakini adakan dapat diatasi walaupun dalam waktu yang panjang dan terus menerus.

2.1.2.1 Komponen Sistem Pakar

Komponen sistem pakar terbagi menjadi empat bagian, untuk lebih jelasnya dapat kita gambarkan sebagai berikut (dalam jurnal penelitian Harison dan Alexyusanderia, 2014: 9):



Gambar 2.6 Komponen Utama Sistem Pakar
(Sumber: Harison dan Alexyusanderia, 2014: 9)

1. *Knowledge Base* (Basis Pengetahuan)

   Knowledge Base merupakan inti dari program sistem pakar karena basis pengetahuan itu merupakan presentasi pengetahuan atau knowledge representation basis pengetahuan adalah sebuah basis data.

2. *Working Memory* (Basis Data atau Memori Kerja)

   Working memory adalah bagian yang mengandung semua fakta-fakta baik fakta awal pada saat sistem beroperasi maupun fakta-fakta pada saat pengambilan kesimpulan sedang dilaksanakan selama sistem pakar beroperasi basis data berada di adalam memori kerja.

3. *Inference Engine* (Mesin Inferensia)

   Inference Engine adalah bagian yang menyediakan mekanisme fungsi berfikir dan pola-pola penalaran sistem yang digunakan oleh seorang pakar. Mekanisme ini akan menganalisa masalah tertentu dan selanjutnya akan mencari jawaban atau kesimpulan yang terbaik. Mesin ini juga akan dimulai pelacakannya dengan mencocokan kaidah-kaidah dalam basispengetahuan dengan faktafakta yang ada dalam basis data.

4. *User Interface* (Antarmuka Pemakai)

   Antarmuka pemakai adalah bagian penghubung antara program sistem pakar dengan emakai. Pada bagian memungkinkan pengguna untuk memasukkan instruksi dan informasi ke dalam sistem pakar serta menerima penjelasan dan kesimpulan.

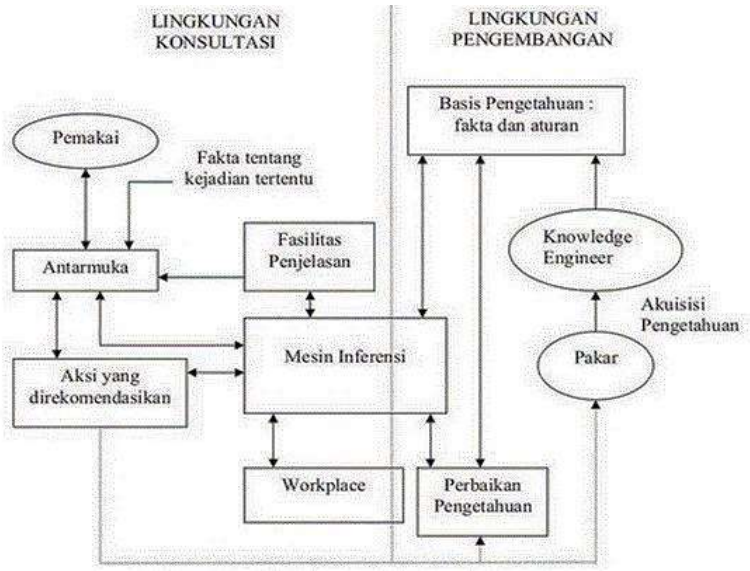
Menurut Husda (2012: 182) terdapat dua bagian utama yang menyusun sistem pakar, yaitu:

1. Lingkungan pengembangan (*Development Environtment*)

   Lingkungan pengembangan digunakan untuk memasukkan pengetahuan pakar ke dalam lingkungan sistem pakar.

2. Lingkungan konsultasi (*Consultation Environtment*)

   Lingkungan konsultasi digunakan oleh pengguna yang bukan pakar untuk memperoleh pengetahuan pakar.



Gambar 2.7 Arsitektur Sistem Pakar
(Sumber: Husda, 2012: 183)

Komponen-komponen yang terdapat dalam arsitektur/struktur sistem pakar (Husda, 2012: 183-186) adalah:

1. Antarmuka Pengguna (*User Interface*)

   Merupakan mekanisme yang digunakan oleh pengguna dan sistem pakar untuk berkomunikasi. Antar muka menerima informasi dari pemakai dan mengubahnya ke dalam bentuk yang dapat diterima oleh sistem.

2. Basis Pengetahuan

   Basis pengetahuan mengandung pengetahuan untuk pemahaman, formulasi, dan penyelesaian masalah. Terdapat 2 elemen dasar, yaitu: fakta (informasi tentang obyek dalam area permasalahan tertentu) dan aturan (informasi tentang cara bagaimana memperoleh fakta baru dari fakta yang telah diketahui).

3. Akuisisi Penegtahuan (*Knowledge Acquistion*)

   Akuisisi pengetahuan adalah akumulasi, transfer, dan transformasi keahlian dalam menyelesaikan masalah dari sumber pengetahuan ke dalam komputer.

4. Mesin/Motor Inferensi (*Inference Engine*)

   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 merekam kejadian yang sedang berlangsung termasuk dalam keputusan sementara.

6. Fasilitas Penjelasan

   Fasilitas penjelasan adalah 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.

2.1.2.2 *Rule* Sebagai Teknik Representasi Pengetahuan

Basis pengetahuan adalah komponen yang berisi sekumpulan kaidah yang berasal dari pengetahuan dalam domain tertentu dan secara umum disajikan dalam bentuk kaidah produksi (*IF*...*THEN*...). Pengetahuan pakar yang disajikan dalam format tertentu didapat dari sekumpulan pengetahuan pakar dan sumber-sumber pengetahuan lainnya seperti buku-buku, jurnal ilmiah, majalah, maupun dokumentasi tercetak lainnya (Hartati dan Iswanti, 2008: 5).

Setiap *rule* terdiri dari dua bagian, yaitu bagian *IF* disebut *evidence* (fakta-fakta) dan bagian *THEN* disebut hipotesis atau kesimpulan (Sutojo, dkk., 2010 dalam Hayadi, 2016: 9).

Hartati dan Ismawati (2008: 25) berikut ini adalah contoh struktur kaidah *IF-THEN* yang menghubungkan obyek:

1. *IF* premis *THEN* konklusi
2. *IF* masukan *THEN* keluaran
3. *IF* kondisi *THEN* tindakan
4. *IF* antesenden *THEN* konsekuen
5. *IF* data *THEN* hasil
6. *IF* tindakan *THEN* tujuan
7. *IF* aksi *THEN* reaksi
8. *IF* gejala *THEN* diagnosa

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 sehingg 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. Gejala mengacu pada keadaan yang menyebabkan adanya kerusakan atau keadaan tertentu yang mendorong adanya pemeriksaan/diagnosa (Hartati dan Iswanti, 2008: 25-26).

Menurut Hartati dan Iswanti, (2008: 26-39) sebelum sampai pada bentuk kaidah produksi, pengetahuan yang berhasil didapatkan dari *domain* tertentu

disajikan dalam bentuk tabel keputusan kemudian dibuat pohon keputusannya sehingga bisa menggambarkan kondisi pada sistem pakar.

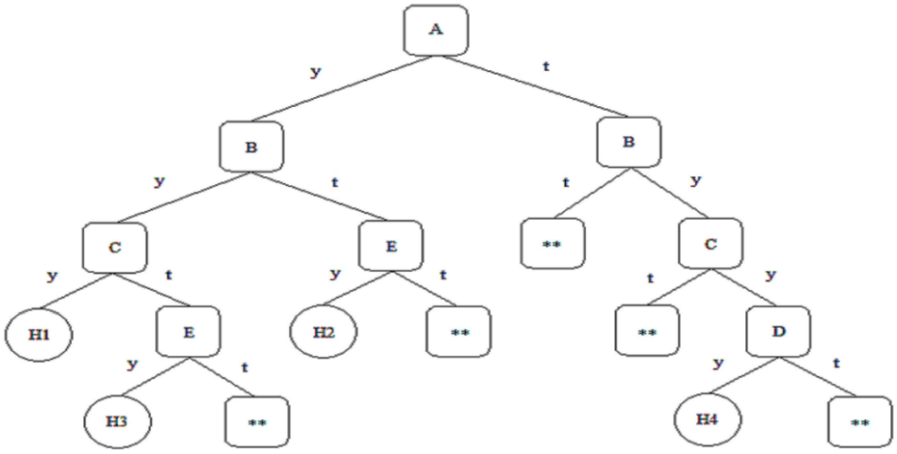
Berikut ini adalah contoh penyajian dalam bentuk tabel keputusan dan pohon keputusan:

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)

Berikut ini adalah contoh gambar pohon keputusan:



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
D = *evidence* D	H4 = hipotesa 4	
E = *evidence* E		

Gambar 2.8 Pohon Keputusan
(Sumber: Hartati dan Iswanti, 2008: 33)

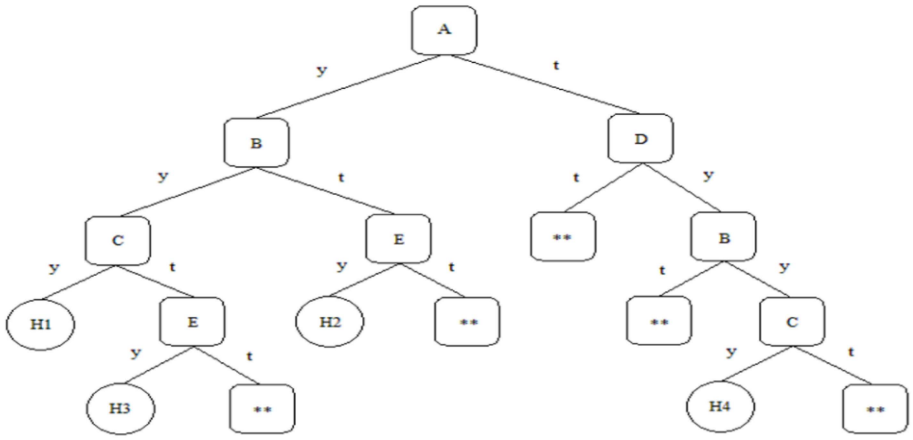
Dari Gambar 2.8 dapat diketahui bahwa hipotesa H1 terpenuhi jika memenuhi *evidence* A, B, dan C. Hipotesa H2 terpenuhi jika memiliki *evidence* A dan *evidence* E. Hipotesa H3 akan terpenuhi jika memiliki *evidence* A, B, dan E. Hipotesa H4 akan dihasilkan jika memenuhi *evidence* B, C, dan D. Notasi "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 2.8 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. Berikut ini adalah tabel keputusan untuk mengatasi hal tersebut:

Tabel 2.2 Tabel Alternatif 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 dapat dihasilkan pohon keputusan (Gambar 2.9) 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
D = *evidence* D	H4 = hipotesa 4	
E = *evidence* E		

Gambar 2.9 Alternatif Pohon Keputusan
(Sumber: Hartati dan Iswanti, 2008: 35)

Dilihat dari Gambar 2.9, 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.

Berdasarkan pohon keputusan pada Gambar 2.9 diatas, maka kaidah yang dapat dihasilkan 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

2.1.2.3 Mesin Inferensi

Mesin inferensi merupakan otak dari sistem pakar, berupa perangkat lunak yang melakukan tugas inferensi penalaran sistem pakar, biasa dikatakan sebagai mesin pemikir (*thinking machine*) (Hartati dan Iswanti, 2008: 5).

Menurut Hartati dan Ismawati (2008: 45), dalam melakukan proses inferensi, sistem pakar memerlukan adanya proses pengujian kaidah-kaidah dalam urutan tertentu untuk mencari suatu kondisi yang sesuai dengan kondisi awal atau untuk memastikan kondisi yang sedang berjalan sudah dimasukkan ke dalam *database*. Proses pengujian itu disebut dengan perunutan atau penalaran, yaitu proses pencocokan fakta atau kondisi tertentu yang tersimpan dalam basis pengetahuan maupun pada memori kerja dengan kondisi yang dinyatakan dalam *premis* atau bagian kondisi pada suatu kaidah atau aturan.

Terdapat beberapa konsep penalaran yang dapat digunakan dalam mesin inferensi (Hartati dan Iswanti, 2008: 45-47), yaitu:

1. Penalaran maju (*forward chaining*)

   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*). Konsep ini dapat dimodelkan sebagai berikut:

   IF (informasi masukan)

   THEN (konklusi)

Informasi masukan dapat berupa suatu pengamatan sedangkan konklusi dapat berupa diagnosa sehingga dapat dikatakan jalannya penalaran runut maju dimulai dari pengamatan menuju diagnosa. Pada metode ini, sistem tidak melakukan praduga apapun terhadap konklusi, namun sistem akan menerima semua gejala yang diberikan pengguna lalu sistem akan memeriksa gejala-gejala tersebut dan selanjutnya mencocokkan dengan konklusi yang sesuai.

2. Penalaran mundur (*backward chaining*)

   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*. Konsep ini dapat dimodelkan sebagai berikut:

   Tujuan

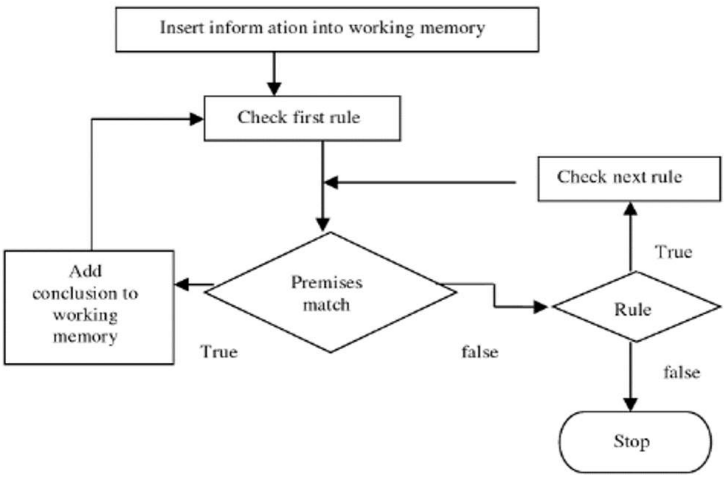
   IF (kondisi)

   Proses penalaran pada *backward chaining* dimulai dari tujuan kemudian merunut balik ke jalur yang mengarah ke tujuan tersebut, untuk membuktikan bahwa bagian kondisi pada kaidah atau aturan benar-benar terpenuhi. Proses *internal* selalu memeriksa konklusi (tujuan) terlebih dahulu sebagai praduga awal, kemudian memeriksa dan memastikan gejala-gejala (kondisi) telah terpenuhi dan selanjutnya mengeluarkan konklusi sebagai *out put*. Jika sistem menemukan ada bagian kondisi yang tidak terpenuhi maka sistem akan memeriksa konklusi (tujuan) pada aturan atau kaidah berikutnya.

2.1.3 Pelacak Maju (*Forward Chaining*)

Forward Chaining adalah teknik pencarian yang dimulai dengan fakta yang diketahui, kemudian mencocokkan fakta-fakta tersebut dengan bagian *IF* dari *rules IF-THEN*. Bila ada fakta yang cocok dengan bagian *IF,* maka *rule* tersebut dieksekusi. Bila sebuah *rule* dieksekusi, maka sebuah fakta baru (bagian *THEN*) ditambahkan ke dalam *database.* Setiap kali pencocokan, dimulai dari *rule* teratas. Setiap rule hanya boleh dieksekusi sekali saja. Proses pencocokan berhenti bila tidak ada lagi rule yang bisa dieksekusi (Sutojo, *dkk.,* 2011: 171).

Operasi dari sistem *forward chaining* dimulai dengan memasukkan sekumpulan fakta yang diketahui kedalam memori kerja (*working memory*), kemudian menurunkan fakta baru berdasarkan aturan yang premisnya cocok dengan fakta yang diketahui (Hayadi, 2016: 11).



Gambar 2.10 Operasi Sistem *Forward Chaining*
(Sumber: Hayadi, 2016: 11)

Menurut Hayadi (2016: 11-12) langkah-langkah yang harus dilakukan dalam membuat sistem *forward chaining* berbasis aturan, yaitu:

1. Pendefinisian masalah

   Tahapan ini meliputi pemilihan domain masalah dan akuisisi pengetahuan.

2. Pendefinisian data *input*

   Sistem *forward chaining* memerlukan data awal untuk memulai inferensi.

3. Pendefinisian struktur pengendali data

   Aplikasi yang komplek memerlukan premis tambahan untuk membantu mengendalikan pengaktifan suatu aturan.

4. Penulisan kode awal

   Tahap ini berguna untuk menentukan apakah sistem telah menangkap domain pengetahuan secara efektif dalam struktur aturan yang baik.

5. Pengujian sistem

   Pengujian sistem dilakukan dengan beberapa aturan untuk menguji sejauh mana sistem berjalan dengan benar.

6. Perencangan antarmuka

   Antarmuka adalah suatu komponen penting dari suatu sistem. Perancangan antarmuka dibuat bersama-sama dengan pembuatan basis pengetahuan.

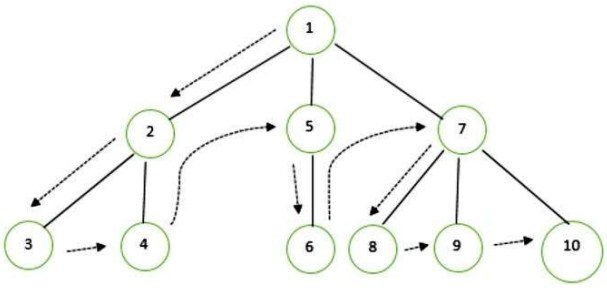
7. Penegembangan sistem

   Pengembangan sistem meliputi penambahan antarmuka dan pengetahuan sesuai dengan *prototype* sistem.

8. Evaluasi sistem

   Pada tahap ini dilakukan pengujian sistem dengan masalah yang sebenarnya. Jika sistem belum berjalan dengan baik maka akan dilakukan pengembangan kembali.

Metode *forward chaining* yang dipakai dalam penelitian dipengaruhi oleh penelusuran *depth first search*. *Depth first search* merupakan metode pencarian yang dilakukan pada suatu simpul dalam setiap level yang dimulai dari kiri. Jika pada level yang terdalam solusi belum ditemukan, maka pencarian dilanjutkan pada simpul sebelah kanan dan simpul yang kiri dapat dihapus dari memori (Suyanto, 2014 dalam jurnal penelitian Harsono, *dkk*., 2015). Gambar dibawah ini menunjukan aliran pencarian *depth first search*.



Gambar 2.11 Diagram Alir Teknik Penelusuran *Depth First Search*
(Sumber: Harsono, *dkk*., 2015: 75)

2.1.4 Basis Data (*Database*)

A.S. dan Shalahuddin (2011: 44) sistem basis data adalah sistem terkomputerisasi yang tujuan utamanya adalan memelihara data atau informasi yang sudah diolah dan membuat informasi tersedia saat dibutuhkan. Pada intinya basis data adalah media untuk menyimpan data agar dapat diakses dengan mudah dan cepat.

Database Management System (*DBMS*) atau sistem manajemen basis data adalah suatu sistem aplikasi yang digunakan untuk menyimpan, mengelola, dan menampilkan data (A.S. dan Shalahuddin, 2011: 45).

Persyaratan minimal sistem aplikasi disebut *DBMS* (A.S. dan Shalahuddin (2011: 445), yaitu:

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

Alur hidup atau *Database Life Cycle* (*DBLC*) terdapat beberapa fase (A.S. dan Shalahuddin, 2011: 48-49), yaitu:

1. Analisis kebutuhan (*requirement analysis*)

   Hal-hal yang harus dilakukan pada tahapan ini adalah:

   a. Didefinisikan dengan mewawancarai produsen dan pemakai data
   b. Membuat kontrak spesifikasi basis data
   c. *Entity Relationship Diagram* (*ERD*)

2. Desain logik basis data (*logical database design*)

   Pada tahap ini harus dibuat rencana logik basis data. Biasanya pada tahap ini dibuat *Conceptual Data Model* (*CDM*).

3. Desain fisik basis data (physical database design)

   Pada tahap ini harus dibuat rencana fisik basis data. Biasanya pada tahap ini dibuat *Physical Data Model* (*PDM*).

4. Implementasi

   a. Membuat *Query SQL*
   b. Aplikasi *DBMS* atau *file*

2.1.5 UML (*Unified Modeling Language*)

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

A.S. dan Shalahuddin (2011: 118) menyatakan bahwa *UML* muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak. *UML* merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung. *UML* hanya berfungsi untuk melakukan pemodelan. Jadi penggunaan *UML* tidak terbatas pada metodologi tertentu, meskipun pada kenyataannya *UML* paling banyak digunakan pada metodologi berorientasi objek.

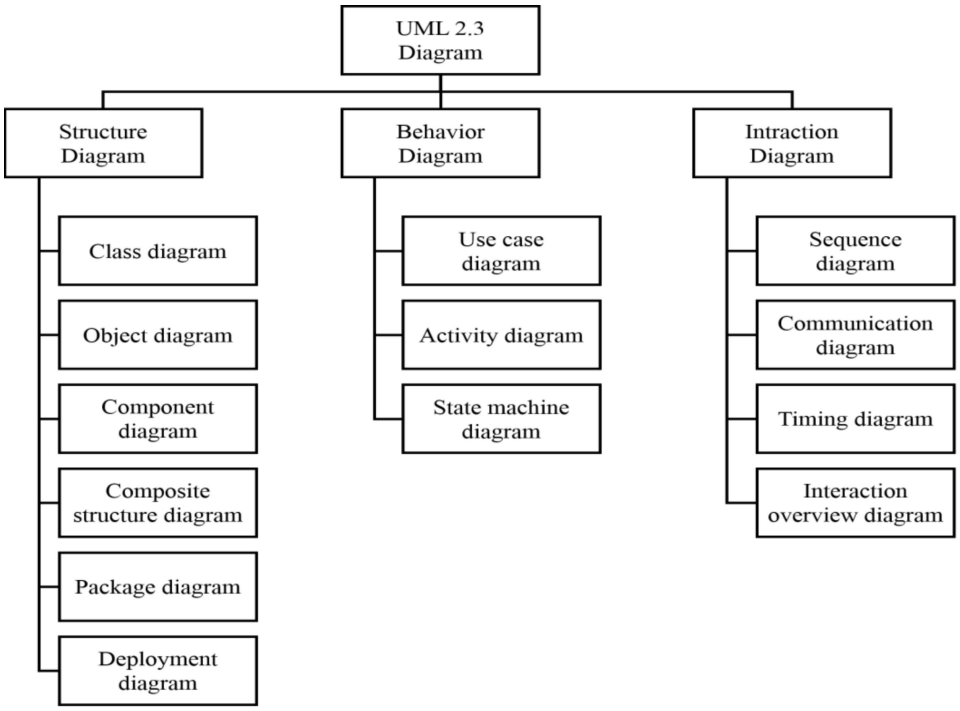
Pada *UML 2.3* terdiri dari 13 macam diagram yang dikelompokkan dalam 3 kategori, A.S. dan Shalahuddin (2011: 121), kategori tersebut adalah:

1. *Structure diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan. Terdiri dari 6 macam diagram antara lain: *class diagram, object diagram, component diagram, composite structure diagram, package diagram,* dan *deployment diagram*.
2. *Behaviour diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada

sebuah sistem. Terdiri dari 3 macam diagram antara lain: *use case diagram, acitivity diagram,* dan *state machine diagram*.

3. *Interaction diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antar subsistem pada suatu sistem. Terdiri dari 4 macam diagram antara lain: *sequence diagram, communication diagram, timing diagram,* dan *interaction overview diagram*.

Berikut ini adalah gambar pembagian kategori dan macam-macam diagram tersebut:



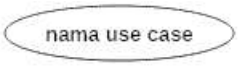
Gambar 2.12 Diagram UML
(Sumber: A.S. dan Shalahuddin, 2011: 113)

Dalam penelitian ini, 3 diagram yang akan digunakan sebagai desain untuk menggambarkan sistem pakar yang dibuat yaitu:

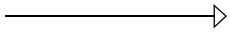
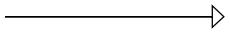
1. *Use case diagram*

Use case atau diagram *diagram use case* merupakan pemodelan untuk menggambarkan kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* 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. Ada 2 hal utama yang terdapat pada *use case* yaitu aktor dan *use case*. Berikut ini adalah simbol-simbol yang digunakan dalam *use case diagram* (A.S dan Shalahuddin, 2011: 130-131).

Tabel 2.3 Simbol *Use Case Diagram*

Simbol	Deskripsi
Use case 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; 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* <<extend>> .....>	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walaupun tanpa *use case* tambahan itu. Biasanya use casetambahan memiliki nama depan yang sama dengan use case yang ditambahkan. Arah panah mengarah pada *use case* yang ditambahkan.

Tabel 2.3 Lanjutan

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* <<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)

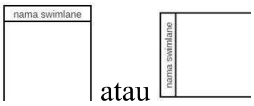
2. *Activity diagram*

Diagram aktiviti atau activity diagram menggambarkan *workflow* (aliran kerja) atau aktifitas dari sebuah sistem atau proses bisnis. Yang perlu diperhatiakan adalah diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor. Berikut ini adalah simbol-simbol yang ada pada *activity diagram* (A.S dan Shalahuddin: 2011: 134):

Tabel 2.4 Simbol *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 2.4 Lanjutan

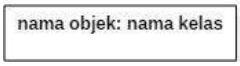
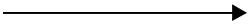
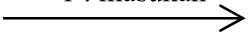
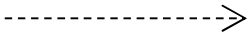
Penggabungan/*join*	Asosiasi penggabungan dimana lebih dari satu aktifitas digabungkan menjadi satu
Status akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktifitas memiliki sebuah status akhir
Swimlane 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktifitas yang terjadi

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

3. *Sequence diagram*

Diagram sekuan (s*equence diagram*) menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambar diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu. Banyaknya diagram sekuen adalah sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua use case yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehingga banyak *use case* yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak. Berikut ini adalah simbol-simbol yang ada pada *sequence diagram* (A.S dan Shalahuddin: 201: 137-138):

Tabel 2.5 Simbol *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: 137-139)

2.1.6 Bahasa Pemrograman

Bahasa pemrograman yang digunakan dalam pembuatan sistem pakar *forward chaining* kerusakan *Integrated Circuit* (*IC*) di mesin Wire Bond IConn adalah:

1. *PHP: Hypertext Preprocessor* (*PHP*)

Gambar 2.13 Logo *PHP*
(Sumber: https://upload.wikimedia.org/wikipedia/commons/thumb/2/27/ PHP-logo.svg/260px-PHP-logo.svg.png)

Menurut Saputra (2012: 91) ada 4 macam format yang bisa digunakan untuk memulai pemrograman PHP didalam kode Anda yaitu:

a. *<?php . . . . . ?>*
b. *<? . . . . . ?>*
c. *<script language=”php”> . . . . . </script>*
d. *<% . . . . . %>*

Aturan penulisan variabel dalam PHP yang perlu diketahui adalah sebagai berikut (Saputra, 2012: 92):

a. Penulisan variabel harus diawali dengan simbol *dollar* (*$*).
b. Karakter pertama setelah simbol dollar, tidak boleh menggunakan angka (harus huruf).

c. Setelah simbol *dollar* (*$*) dan huruf, maka karakter selanjutnya boleh menggunakan angka.

2. *HTML (Hyper Text Markup Language)*

Gambar 2.14 Logo *HTML*
(Sumber: https://www.w3.org/html/logo/downloads/HTML5_Logo_512.png)

HTML merupakan singkatan dari *Hyper Text Markup Language*. *HTML* bisa disebut bahasa paling dasar dan penting yang digunakan untuk menampilkan dan mengolah tampilan pada halaman *website*. Menurut wikipedia, *HTML* digunakan untuk menampilkan berbagi informasi didalam sebuah penjelajah *web* internet dengan *formatting hypertext* sederhana yang ditulis kedalam berkas format ASCII agar dapat menghasilkan tampilan wujud yang terintegrasi (dalam Saputra, 2012:1).

Menurut Saputra (2012:4-6) beberapa elemen-elemen wajib yang ada pada file *HTML* apabila kita ingin membangun suatu pondasi kerangka *website*:

a. Elemen *HTML*

Elemen ini merupakan tag dasar apabila kita ingin memulai suatu dokumen *HTML*. Tag ini merupakan perintah wajib bagi pemrogram *web* untuk menulis tag pertama dalam dokumen *html*. Contoh tag-nya adalah:

</html> dan diakhiri dengan *</html>*

b. Elemen *Head*

Head merupakan tag berikutnya setelah elemen *html*, yang berfungsi untuk menuliskan keterangan tentang dokumen *web* yang akan ditampilkan. Contoh format penulisannya:

<*html*>

<*head*>

</*head*>

</*html*>

c. Elemen *Title*

Element *Title* merupakan suatu elemen yang harus dituliskan didalam elemen *head* yang digunakan untuk memberikan judul/informasi pada *cation browser web* tentang topik/tema atau judul dari suatu dokumen *web* yang ditampilkan pada *browser*. Contoh struktur penggunaannya:

<*html*>

<*head*>

<*title*> Tuliskan Judul disini </*title*>

</*head*>

</*html*>

d. Elemen Body

Elemen *body* merupakan bagian utama dalam dokumen *web*. Jika kita ingin menampilkan suatu teks atau informasi (konten), maka kita harus meletakkan teks tersebut pada elemen *body*. Struktur elemnnya sebagai berikut:

<*html*>

<*head*>

<*title*> Tulis Judul Disini </*title*>

</*head*>

<*body*>

Tuliskan Konten Disini

</*body*>

</*html*>

3. *CSS (Cascading Style Sheet)*

Gambar 2.15 Logo *CSS*
(Sumber: http://w3widgets.com/responsive-slider/img/css3.png)

Menurut Saputra (2012: 27) *CSS* (*Cascading Style Sheet*) merupakan bahasa pemrograman *web* yang didesain khusus untuk mengendalikan dan membangun berbagi komponen didalam web sehingga tampilan web lebih rapi, terstruktur dan seragam. Tujuan untama dari *CSS* adalah memisahkan konten utama dengan tampilan dokumen lainnya (*html* dan sejenisnya). Tujuan lainnya adalah untuk mempercepat pembuatan halaman *web*.

Keuntungan yang diperoleh denganmenggunakan *CSS* adalah (Saputra, 2012:27):

a. Memisahkan pembuatan dokumen (*CSS* dan *HTML*).

b. Mempermudah dan mempersingkat pembuatan dan pemeliharaan dokumen *web*.

c. Akses *web* lebih cepat saat *loading* (mempercepat pembacaan *HTML*).

d. Fleksibel, interaktif, tampil lebih menarik dan nyaman dipandang.

e. Lebih kecil ukuran *file* sehingga *bandwith* yang digunakan juga otomatis menjadi lebih kecil.

f. Dapat digunakan pada semua *web browser*.

Saputra (2012: 29-32) untuk menggunakan *CSS*, setidaknya ada 3 cara yang bisa kita gunakan, yaitu:

a. *Embedded Style Sheet,*

Merupakan cara penulisan kode dimana penulis *CSS* dilakukan pada tag *html*, yaitu pada tag <style> ... </style> dan sebelum tag <body>.

b. *Inline Style Sheet,*

Metode *Inline style sheet*, yang merupakan salah satu cara penggunaan CSS langsung pada tag html yang dibutuhkan saja. Cara ini dilakukan karena hanya sedikit properti yang dibutuhkan.

c. *Linked Style Sheet,*

Metode ini merupakan cara pengerjaan dimana antara kode *css* dan *html* telah dipisahkan. Untuk menggunakan kode *css* yang telah dipisahkan ini, maka dalam kode *html* dibuat skrip yang isinya adalah memanggil *file css* tersebut untuk digunakan dalam kode *html*. Contoh *link* untuk memanggil kode *css* ini:

<link rel=”stylesheet” href=”NamaFile.css” type=”text/css”>

4. *JavaScript dan jQuery*

JavaScript adalah bahasa *scripting* handal yang berjalan pada sisi *client*. *JavaScript* merupakan sebuah bahasa *scripting* yang dikembangkan oleh *Netscape*. Untuk menjalankan *script* yang ditulis dengan *JavaScript* kita membutuhkan *JavaScript-enable browser* yaitu *browser* yang mampu menjalankan *JavaScript* (Saputra, 2012: 3).

Dalam jurnal Warma dan Zahni (2013: 33) j*Query* adalah sebuah *framework* berbasiskan *JavaScript*. j*Query* sama dengan *JavaScript Library* yaitu kumpulan kode atau fungsi *JavaScript* siap pakai, sehingga mempermudah dan mempercepat kita dalam membuat kode *JavaScript*. Hal yang menarik dari j*Query* adalah penekanan interaksi antara *JavaScript* dan *HTML*. Berikut ini adalah logo *JavaScript* dan *jQuery*:

Gambar 2.16 Logo *JavaScript*
(Sumber: http://www.w3devcampus.com/wp-content/uploads/logoAndOther/logo_JavaScript.png)

Gambar 2.17 Logo *jQuery*
(Sumber: http://precision-software.com/wp-content/uploads/2014/04/jQurery.gif)

2.1.7 Validitas Sistem

Validasi mengacu pada sekumpulan aktifitas yang berbeda yang menjamin bahwa perangkat lunak yang dibangun dapat ditelusuri sesuai dengan kebutuhan pelanggan (*customer*) (A.S. dan Shalahuddin, 2011: 211).

A.S. dan Shalahuddin (2011: 213-214) pengujian validasi memiliki beberapa pendekatan sebagai berikut:

1. *Black-box* t*esting* (pengujian kotak hitam)

   Black-box testing yaitu menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian dimaksudkan untuk mengetahui apakah fungisi-fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan.

2. *White-box testing*

   White-box testing yaitu menguji perangkat lunak dari segi desain dan kode program apakah mampu menghasilkan fungsi-fungsi, masukan, dan keluaran yang sesuai dengan spesifikasi kebutuhan. Pengujian ini dilakukan dengan memeriksa logik dari kode program.

2.2 Variabel Penelitian

Variabel penelitian pada dasarnya adalah segala sesuatu yang berbentuk apa saja yang ditetapkan oleh peneliti untuk dipelajari sehingga diperoleh informasi dan kesimpulannya (Sudaryono, 2015:16). Obyek yang digunakan dalam penelitian ini adalah mesin *Wire Bond* IConn dan variabel yang ditetapkan dalam penelitian ini

adalah mesin *Wire Bond* IConn yang berupa gejala kerusakannya dan kerusakan unit (*IC*) yang ditimbulkan oleh permasalahan mesin *Wire Bond* IConn.

2.2.1 Mesin *Wire Bond* IConn

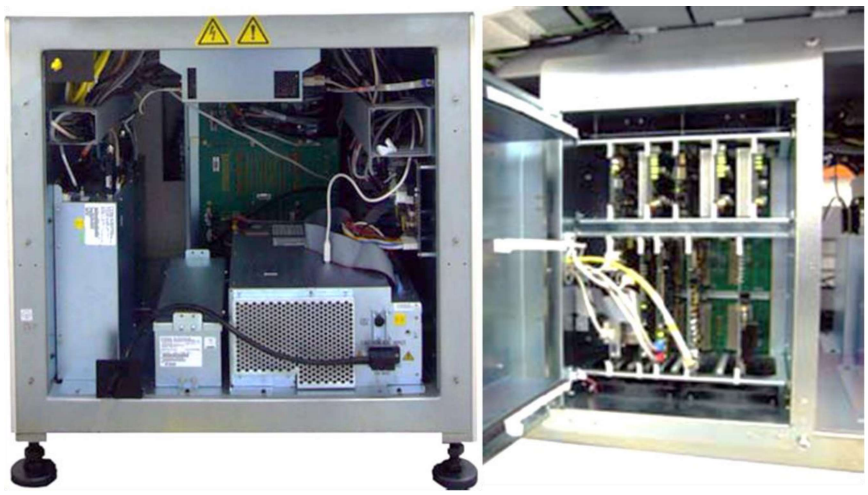
Proses *Wire Bond* adalah proses penyambungan *bond pad* yang terdapat pada *die* ke *lead finger* yang terdapat pada *leadframe* dengan menggunakan benang emas (*gold wire*), artinya koneksi benang emas yang dihasilkan oleh proses *wire bond* bisa dikatakan sempurna karena tidak terkandung unsur *defect* sesuai dengan spesifikasinya (dalam jurnal penelitian Supriyanto, dkk., 2013: 131).

Dalam jurnal penelitian Gan dan U. Hashim (2015) perkembangan teknologi *wire bond* menggunakan benang emas (*gold wire*) dan benang tembaga (*copper wire*) sudah berjalan lebih dari 25 tahun. Teknologi terus berkembang dan sekarang sudah menggunakan teknologi benang perak (*silver wire*). Gold wire bonding merupakan teknologi pertama yang diperkenalkan dalam teknologi semikonduktor, tetapi karena harga dalam pembuatan meningkat maka dibuatlah teknologi menggunakan *copper wire bonding* sehingga bisa menekan biaya pembuatan. Tetapi *copper wire bonding* mengalami masalah dalam *temperature cycle test* dan *pressure cooker test*. Sedangkan *silver wire bonding* memiliki sifat mekanikal seperti *gold wire bonding* serta mempunyai konduktivitas listrik dan panas lebih baik dari *copper wire bonding*, tetapi teknologi *silver wire bonding* membutuhkan lebih gas argon dalam pembuatannya.

Mesin Wire Bond KNS IConn terdiri dari beberapa bagian (KNS, 2008: 1-2 – 1-5), yaitu:

1. *Lower Consule*

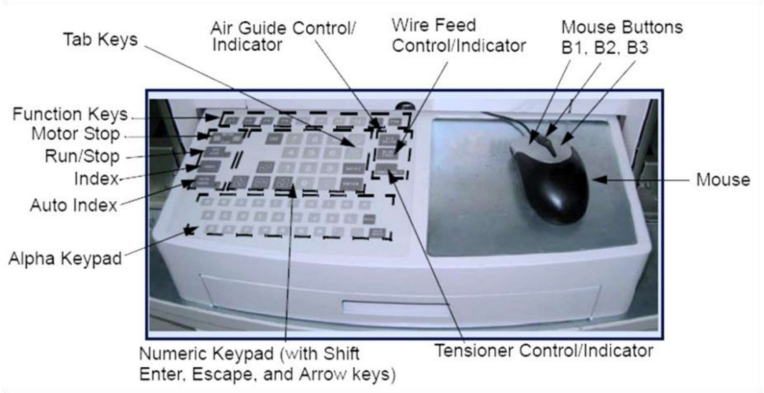
   Lower Console terdiri dari bagian *cardchange*, *power supply*, *distribution system*, dan *cooling system*. Bagian ini juga terdiri dari *circuit board*, komputer, dan *visio system*.



Gambar 2.18 *Lower Console*
(Sumber: Operations manual book K&S 2008: i-11)

2. *XY Table*

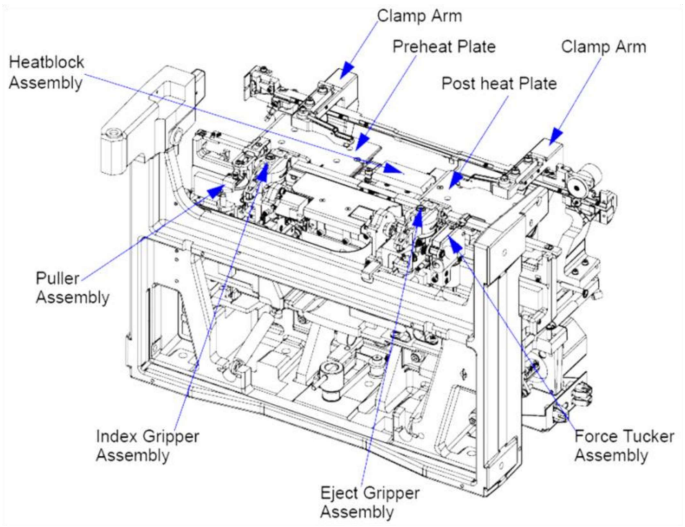
   Bagian *XY table* terdiri dari 2 *slider* dan sistem penggerak yang dipasang pada badan mesin yang besar. *Linear motor servo* bergerak kearah X (kiri – kanan) dan Y (depan – belakang). Pergerakan bisa diposisikan menggunakan *mouse* dan tombol panah pada *Machine Interface Man* (*MMI*).



Gambar 2.19 *Machine Interface Man (MMI)*
(Sumber: Operations manual book K&S 2008: 2-2)

3. *Material Handling System (MHS)*

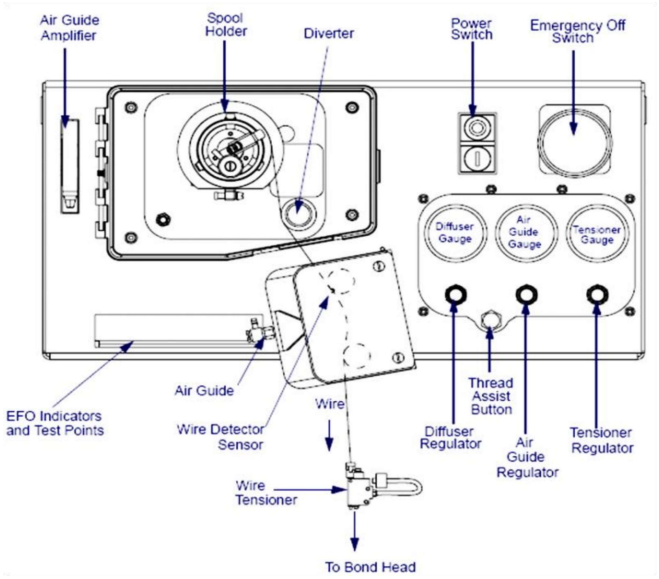
   MHS menyediakan *automated loading*, *indexing*, dan *removal of lead frame* selama proses *bonding*. *MHS* juga menggerakkan *magazines* dari awal sampai penyimpanan. *MHS* bertugas mengambil *lead frame*, memindahkan *lead frame* ke posisi *bonding*, kemudian mengeluarkan *lead frame* setelah proses *bonding* selesai.



Gambar 2.20 *IConn Workholder (Model 3129)*
(Sumber: Operations manual book K&S 2008: 2-2)

4. *Wire Feed System (Upper Console)*

   Wire spool holder menggunakan ukuran *spool wire* sebesar 2 inchi yang dipasang pada *panel* diatas *bond head*. *Wire spool holder* adalah konduktif dan elektrik yang menghubungkan *wire* pada *wire spool* untuk *Bond Integrated Test System* (*BITS*). Ini digunakan untuk mendeteksi masalah *Non Stick on Pad* (*NSOP*), *Non Stick on Lead* (*NSOL*), dan *Short Tail* (*SHTL*). Perputaran motor pada *wire spool holder* dimonitor oleh serat *optic/sensor optic*.



Gambar 2.21 *Wire Feed System*
(Sumber: Operations manual book K&S 2008: 3-14)

5. *Bond Head*

   Bond head dipasang sebgai bagian integral dari depan Y *slide* XY *table* terdiri dari:

   a. *Tranducer* dengan *capillary*

   b. *Flat coil Z motor with motor housing*

   c. *Beryllium Aluminum link*

d. *High resolution Z encoder*

e. *Bond head temperature stabilization via Z motor heating elements and constant cooling air knife*

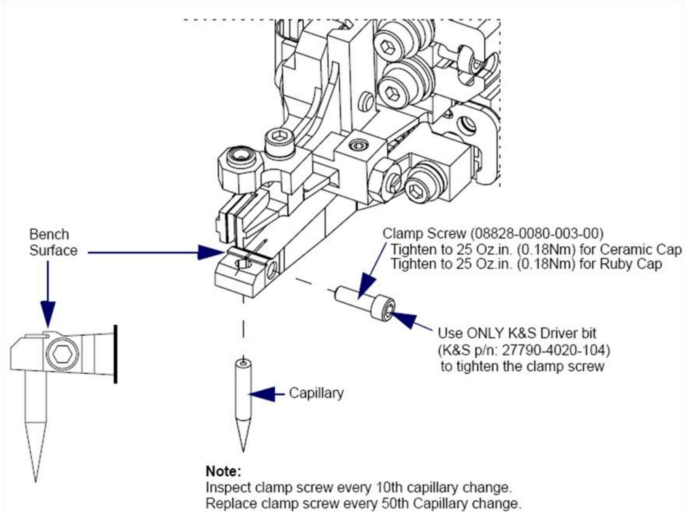
f. *Integrated parallel wire clamps*

g. *Modular flexure bearing assembly*

h. *Electronic Flame-Off* (*EFO*) *assembly*

i. *Contact force sensor* (*Piezo sensor*)

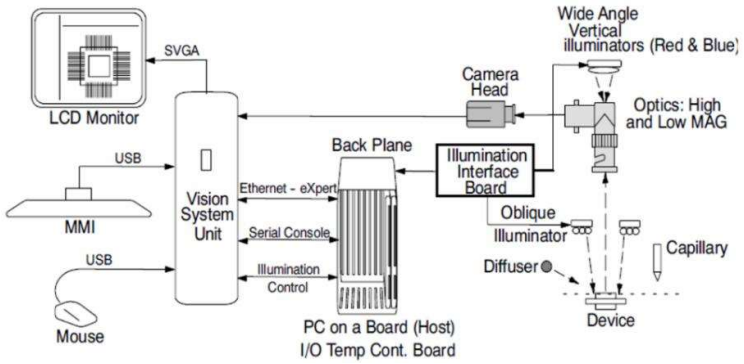
j. *Moveable Electronic Flame-Off* (*MEFO*) *Assembly* (*Optional*, *attached to the Y slide replacing the EFO wand*)



Gambar 2.22 *Bond Head With Capillary Change*
(Sumber: Operations manual book K&S 2008: 3-8)

6. *Vision System and Optics*

Pada bagian ini terdapat beberapa bagian yaitu: 2 perbesaran terdiri dari *vertical* dan *oblique illumination*, lensa, kaca, 2 *progressive scan cameras*, dan *board set* yang menampilkan tampilan *video grafis*.



Gambar 2.23 *Vision System And Optics*
(Sumber: Operations manual book K&S 2008: 1-4)

7. *Software*

   Software yang dijalankan pada platform IConn Ball Bonder adalah berbasis menu dan mempunyai tingkatan. Pesan dan petunjuk yang muncul di monitor berfungsi untuk menginformasikan penggunaan status mesin atau untuk memasukkan informasi yang dibutuhkan. *MMI* control digunakan oleh pengguna untuk merespon pada saat yang tepat.

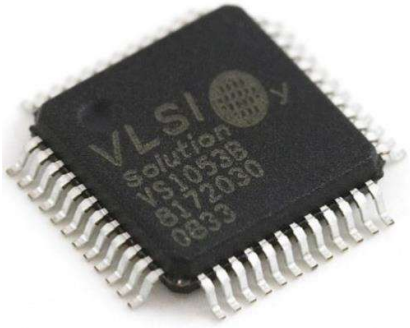
Dalam buku panduan operasional mesin *Wire Bond* IConn menerangkan bahwa mesin *Wire Bond* IConn memilikin beberapa perangkat tambahan utama, (K&S, 2008: 1-1) yaitu:

1. Peningkatan kemampuan *Ultra Fine Pitch*
2. Peningkatan produktivitas.
3. Peningkatan *bondable area*.
4. Dual frekuensi ultrasonik
5. *Enhanced loop mode*.
6. Dual *mag look ahead optics*.
7. Peningkatan *service*
8. Penambahan *extended holdup power supply* dengan pemulihan otomatis.

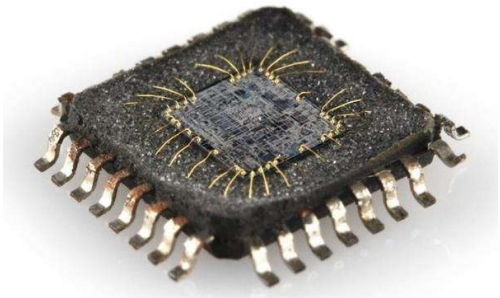
2.2.2 Kerusakan *IC* pada Area *Wire Bond*

IC (*Integrated Circuit*) adalah nama lain *chip*. *IC* adalah piranti elektronis yang dibuat dari material semikonduktor. *IC* merupakan gabungan dari beberapa komponen seperti Resistor, Kapasitor, Dioda dan Transistor yang telah terintegrasi menjadi sebuah rangkaian berbentuk *chip* kecil. *IC* ditemukan pada tahun 1985 oleh seorang insinyur bernama Jack Kilby yang bekerja pada Texas Instrumens. (Hakiem, 2015: 23).

Berikut ini adalah gambar *IC* bagian luar dan gambar *IC* yang terbuka dan terlihat bagian *wire bond* dengan *gold wire*:



Gambar 2.24 Bentuk Fisik *IC* Tampak Luar
(Sumber: http://static.sunrom.com/p/835/835_800.jpg)



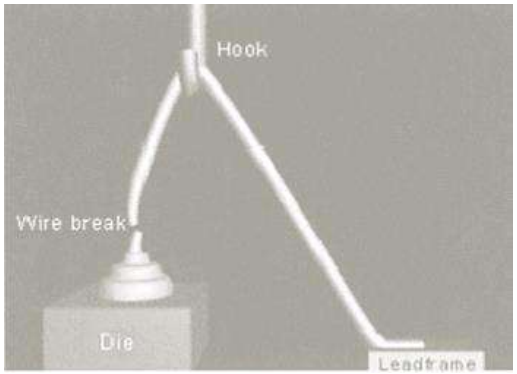
Gambar 2.25 Bentuk Fisik *IC* Tampak Dalam
(Sumber: https://cdn.sparkfun.com/r/600-600/assets/7/a/6/9/c/
51c0d009ce395feb33000000.jpg)

Kemampuan kinerja *IC* dalam aplikasi apapun baik itu di mobil, pesawat ruang angkasa atau komputer pribadi, sangat tergantung pada kualitas interkoneksi *wire bond*. Jika kualitas interkoneksi *wire bond* lemah dan tidak konsisten, memiliki dampak yang signifikan tidak hanya pada keandalan dan ketergantungan dari perangkat, tetapi juga memiliki dampak ekonomi yang besar. Oleh karena itu, jaminan kualitas dari *wire bond* sangat penting dalam alur kerja perakitan IC (dalam jurnal penelitian Wang dan Sun, 2009: 50).

Dalam jurnal penelitian Wang dan Sun (2009: 50-53) kualitas dari *wire bond* ditentukan oleh beberapa tes, yaitu:

1. *Bond pull test* (*wire pull test*)

   Pengujian *bond pull* dilakukan dengan cara meletakkan *hook* pada *wire* kemudian *hook* akan menerapkan gaya tarik keatas dan *wire* ditarik sampai putus. Selama dilakukan tes *bond pull*, jika terjadi kegagalan seperti *ball lift failure* atau *weld* (*wedge*) *lift failure* maka kondisi ini tidak diijinkan karean kualitas *bonding* yang rendah. Berikut ini adalah gambar *bond pull test*:



Gambar 2.26 *Bond Pull Test*
(Sumber: Wang dan Ronglu, 2009:54)

2. *Ball-bond shear test*

   Pengujian *ball shear* digunakan untuk menilai *integritas* antara *ball bond* dengan *bond pad*. Pengujian ini merupakan uji destruktif. *Shear tool* mendorong *ball bond* and menghasilkan angka pengukuran. Data *ball shear* mencerminkan *intermetallic* dan menempelnya *ball bond*. Hal ini diukur dengan besaran gram *force*.

3. Tampilan *visual ball bond*

   Kondisi *ball bond* yang bagus dan tidak *flat ball* serta kondisi *wire* yang lurus tidak *damaged wire*.

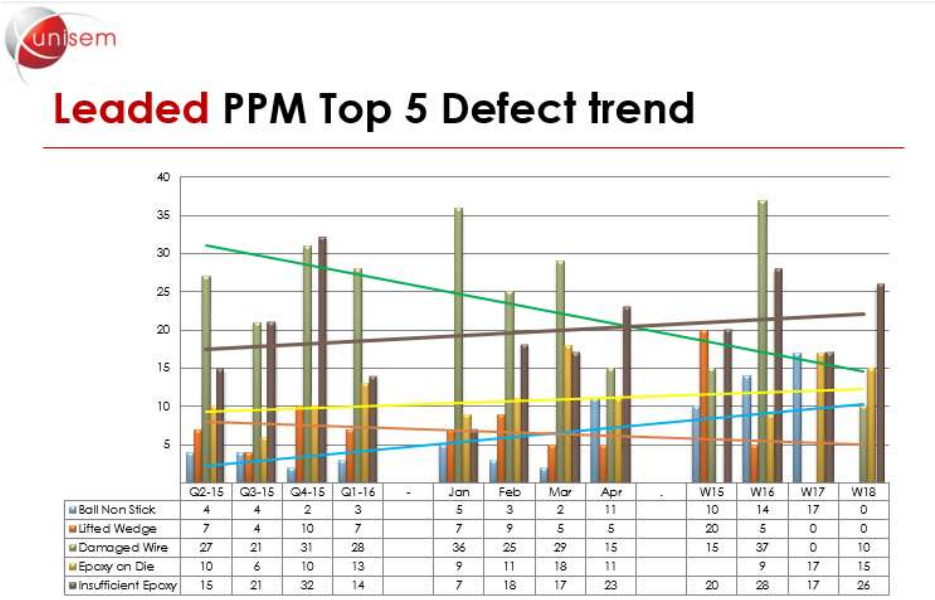
4. Spesial tes untuk kualitas

   Kualitas dari *wire bond* ditentukan oleh kekuatan tampilan *ball bond* pada *bond pad* sebagus *wedge* pada *lead finger*. Ada beberapa faktor kualitas lainnya untuk memeriksa kekuatan *wire bond* yaitu: *wire sweep*, *wire sway*, *wire sagging*, *heel cracking*, *ball deformation* dan banyak kerusakan lainnya.

iAS1006 (*Assembly Spesification* versi Bahasa Indonesia) mempunyai tujuan untuk mendefinisikan Acuan Operasi, Spesifikasi Process dan *Total Control Methodology* (*TCM*) untuk operasi *Wire Bond*. iAS1006 menerangkan bahwa area produksi *Wire Bond* merupakan salah satu area produksi pembuatan *IC* (*Integrated Circuit*) yang bertugas menghubungkan *die bond pads* dengan *lead finger* dan atau *lead frame pads* dengan *gold/copper wire*. *Ball Bond* adalah *wire* yang dibentuk menyerupai bola dengan terlebih dahulu diberikan lecutan elektronik untuk memproduksi ujung bola FAB (*Free Air Ball*) dan kemudian ditumbukkan ke *metallized bond* pad dengan *capillary*. *Wedge Bond* adalah *wire* yang dibentuk

menyerupai huruf "V" dengan menggunakan bagian *capillary*. *Capillary* adalah alat yang terbuat dari keramik yang berfungsi sebagai penyalur tekanan dan tenaga ultrasonic dalam proses menghubungkan *lead finger* dan atau *lead frame pads* (Yunus, 2016: 1-3).

Berdasarkan laporan mingguan pada minggu ke 18 tahun 2016, data menunjukkan 5 top defect pada area *FOL* (*Front of Line*) 3 diantaranya kerusakan unit yang berasal dari area *Wire Bond* dan 2 berasal dari area *Die Attach*. Berikut ini adalah gambar grafik laporan LRR pada WW18:



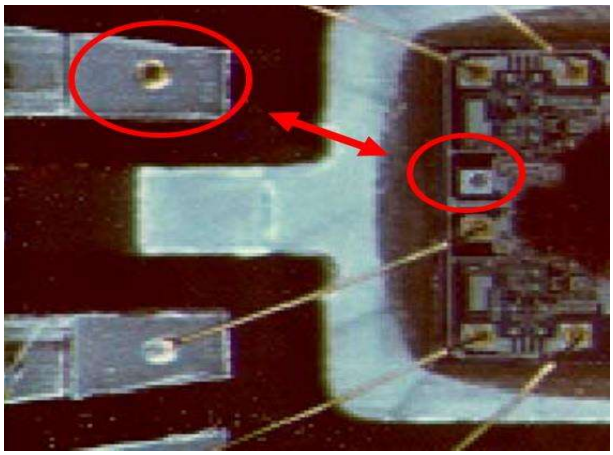
	Q2-15	Q3-15	Q4-15	Q1-16	-	Jan	Feb	Mar	Apr	.	W15	W16	W17	W18
Ball Non Stick	4	4	2	3		5	3	2	11		10	14	17	0
Lifted Wedge	7	4	10	7		7	9	5	5		20	5	0	0
Damaged Wire	27	21	31	28		36	25	29	15		15	37	0	10
Epoxy on Die	10	6	10	13		9	11	18	11			9	17	15
Insufficient Epoxy	15	21	32	14		7	18	17	23		20	28	17	26

Gambar 2.27 Lima *Top Defect* Pada Area *FOL*
(Sumber: LRR Weekly report WW18 PT UNISEM Batam, 2016)

Reject unit yang berasal dari area *Die Attach* adalah *Epoxy on Die* dan *Insufficient Epoxy*, sedangkan reject unit yang berasal dari area *Wire Bond* adalah *Ball Non Stick*, *Lifted Wedge*, dan *Damaged Wire*.

2.2.2.1 *Ball Non Stick*

Ball Non Stick adalah *ball* yang gagal terbonding meninggalkan bekas di posisi *wedge*-nya (Yunus, 2016:11). Berikut ini adalah gambar *IC* dengan kerusakan *Ball Non Stick*:



Gambar 2.28 *Ball Non Stick*
(Sumber: iAS1007 rev.AV, 2016: 11)

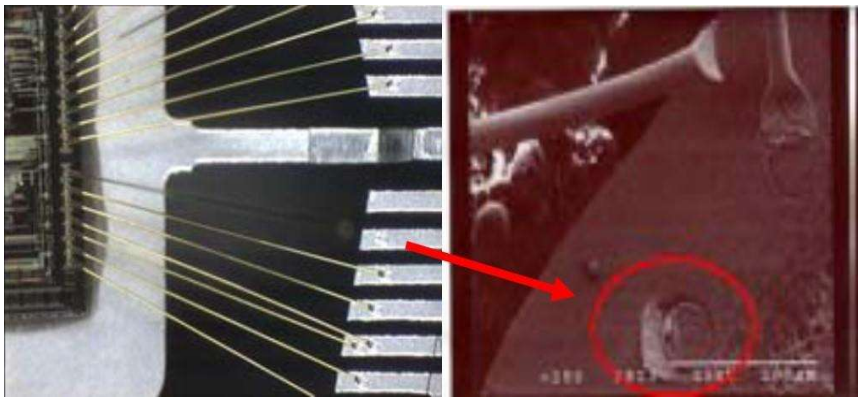
Langkah-langkah yang harus dilakukan untuk mengatasi masalah *Ball Non Stick* pada *IC* (Yunus, 2016:188), adalah:

1. Hilangkan *floating* pada *die pad*.
2. Mengoptimalkan *ball*/*serach parameter*.
3. Cek aktual suhu *heater block*.
4. Kurangi *EFO Current* dan *EFO Time parameter*.

2.2.2.2 *Lifted Wedge*

Lifted Wedge adalah *wedge* yang gagal terbonding di permukaan lead atau *wedge* yang tidak menempel pada permukaan *lead* (Yunus, 2016:11).

Berikut ini adalah gambar *IC* dengan kerusakan *Lifted Wedge*:



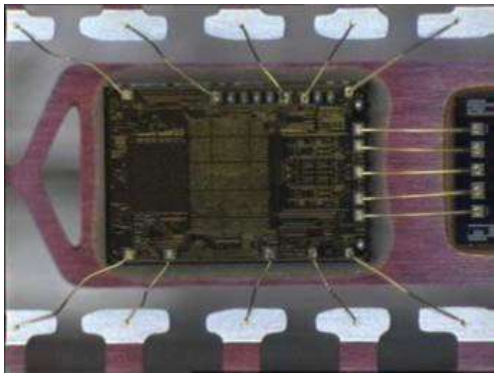
Gambar 2.29 *Lifted Wedge*
(Sumber: iAS1007 rev. AV, 2016: 12)

Langkah-langkah yang harus dilakukan untuk mengatasi masalah *lifted wedge* pada *IC* (Yunus, 2016:129), adalah:

1. Hilangkan *foreign* material pada *down holder*.
2. Menghilangkan *floating* pada *lead*.
3. Naikkan parameter *wedge*.
4. Mengganti *capillary* jika sudah *limit*.
5. Mengecek suhu *heater block*.
6. Kalibrasi mesin.

2.2.2.3 *Damaged Wire*

Damaged Wire adalah *wire* yang berubah bentuk, rusak disebabkan oleh mesin atau salah penanganan (Yunus, 2016:12). Berikut ini adalah gambar *IC* dengan kerusakan *Damaged Wire*:



Gambar 2.30 *Damaged Wire*
(Sumber: iAS1007 rev. AV, 2016: 13)

Langkah-langkah yang harus dilakukan untuk mengatasi masalah *damaged wire* pada *IC* (Yunus, 2016:128), adalah:

1. Optimalkan urutan *bonding wire bond*.
2. Ganti *capillary* jika salah tipe.
3. Optimalkan parameter *loop*.
4. *Adjust* posisi/letak *top clamp*.
5. Kurangi kecepatan *indexer*.
6. *Adjust indexer rails* jika terlalu sempit.

2.3 *Software* Pendukung

Software pendukung merupakan beberapa perangkat lunak yang digunakan untuk mendukung pembuatan sistem pakar dalam penelitian ini. Perangkat lunak tersebut antara lain: *StarUML*, *XAMPP*, *Notepad++*, *phpMyAdmin*, dan *Mozilla Firefox*.

2.3.1 StarUML

Gambar 2.31 Logo *StarUML*
(Sumber: http://staruml.sourceforge.net/image/staruml-logo.jpg)

StarUML merupakan proyek *open source* untuk mengembangkan platform *Unified Modeling Language* (*UML*) atau *Model Driven Architecture* (*MDA*) yang cepat, fleksibel, dapat diperluas, memiliki banyak fitur, dan tidak dipungut biaya. Tujuan dari proyek ini dalam untuk membangun sebuah perangkat lunak pemodelan dan sekaligus platform yang dapat menggantikan perangkat *UML* berbayar lain, seperti *Rational Rose*, *Together*, dan sebagainya. *StarUML* dikembangkan dalam Bahasa Pemrograman *Delphi*. Walaupun begitu, *StarUML* merupakan proyek yang multi-lingual dan tidak bergantung pada bahasa pemrograman yang spesifik, sehingga bahasa pemrograman apapun dapat digunakan untuk mengembangkan *StarUML*, seperti *C/C++*, *Java*, *Visual Basic*, *Delphi*, *Jscript*, *VBScript*, *C#, VB.NET*, dan sebagainya (dalam jurnal Iswari, 2015: 73).

S*tarUML* adalah *platform* pemodelan perangkat lunak yang mendukung *UML* (*Unified Modeling Language*). *StarUML* yang berbasiskan pada *UML* versi 1.4, menyediakan sebelas jenis diagram yang berbeda dan mendukung notasi *UML* 2.0 (Triandini dan Suardika, 2012: 1).

Triandini dan Suardika (2012: 2-3) fitur-fitur utama dalam *StarUML* versi 5.0 adalah sebagai berikut:

1. Dukungan terhadap Diagram *UML 2.0*:
    a. *Use Case Diagram*
    b. *Clas Diagram*
    c. *Sequence Diagram*
    d. *Collaboration Diagram*
    e. *Statechart Diagram*
    f. *Activity Diagram*
    g. *Componen Diagram*
    h. *Deployment Diagram*
    i. *Composite Structure Diagram* (*UML 2.0*)
2. Dukungan terhadap beberapa bahasa pemrograman:
    a. *Java Profile, Code Generator*, dan *Reverse Engineer*
    b. *C++ Profile, Code Generator*, dan *Reverse Engineer*
    c. *C# Profile, Code Generator*, dan *Reverse Engineer*
    d. *Microsoft Office Document Generator*
    e. *Microsoft Word document template and generation*
    f. *Microsoft Excel document template and generation*
    g. *Microsoft PowerPoint document template and generation*
3. *Customizable Code Generation*
4. Mendukung teknologi *MDA*
5. Diagram yang dapat diperluas

6. *Extensibility*
7. Kompabilitas yang tinggi
8. *Editing*
9. *User Interface*
10. *Model verification*
11. *Pattern Support*

2.3.2 *XAMPP*

Gambar 2.32 Logo *XAMPP*
(Sumber: https://a.fsdn.com/allura/p/xampp/icon)

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

2.3.3 *Notepad* ++

Gambar 2.33 Logo *Notepad*++
(http://freesoftwaretools.org/wp-content/uploads/2013/08/notepad-plus-plus.jpg)

Gilmore (2010: 36-37) *Notepad*++ adalah kode *editor open source* yang bagus dan sebagai pengganti *Notepad* yang tersedia untuk *platform Windows* dan dirilis di bawah *GNU GPL*. *Notepad*++ menawarkan beragam kenyamanan fitur yang diharapkan dari setiap kemampuan *IDE*, termasuk kemampuan untuk menunjukkan baris tertentu dari suatu dokumen sebagai referensi yang mudah, sintaks, tanda kurung, *indentation highlighting*, fasilitas pencarian yang tangguh, *macro recording* untuk tugas-tugas seperti memasukkan *template* komentar, dan sebagainya. Dengan dukungan kode spesifik *PHP* sehingga banyak kemudahan yang didapat melalui fitur umum tersebut. Dengan dukungan dasar untuk *auto-completion* fungsi nama yang ditawarkan akan mengurangi beberapa pengetik kode program, meskipun pengguna masih harus mengingat beberapa parameter pengkodena pada aplikasi yang dibuat.

2.3.4 *phpMyAdmin*

Gambar 2.34 Logo *phpMyAdmin*
(Sumber**:** https://www.phpmyadmin.net/static/images/logo.png)

phpMyadmin adalah perangkat lunak bebas yang ditulis dalam bahasa pemrograman *PHP* yang digunakan untuk menangani administrasi *MySQL* melalui *WWW* (*World Wide Web*). *phpMyAdmin* mendukung berbagai operasi *MySQL*. Operasi pengolahan dalam *phpMyAdmin* yaitu mengelola basis data (*database*), table-tabel, kolom (*fields*), relasi (*relations*), *indeks*, pengguna (*users*), perijinan (*permissions*), dan lain-lain. Pada dasarnya, mengelola basis data dengan *MySQL* harus dilakukan dengan cara mengetikkan baris-baris perintah yang sesuai (*command line*) untuk setiap maksud tertentu. Jika seseorang ingin membuat basis data (*database*), ketikkan baris perintah yang sesuai untuk membuat basis data (dalam jurnal penelitian Barri, dkk., 2015: 25).

2.3.5 *Mozilla Firefox*

Manzur (2010: 2) *Mozilla Firefox* merupakan aplikasi *web browser* gratis yang dikembangkan oleh Yayasan *Mozilla* dan beberapa *developer* pendukungnya. Pada saat pertama kali dirilis tanggal 9 November 2004 (versi 1.0) *browser* ini bernama *Phoenix*. Kemudian berganti nama *Mozilla Firebird*, hingga akhirnya berganti nama menjadi *Mozilla Firefox*. Sedangkan versi 2.0 untuk pemakaian

pertamakalinya dirilis pada tanggal 24 Oktober 2006 dan versi 3.0 dirilis pada tanggal 17 Juni 2008. Berikut ini adalah logo *Mozilla Firefox*:

Gambar 2.35 Logo *Mozilla Firefox*
(Sumber: https://www.mozilla.org/media/img/firefox/template/header-logo-inverse.510f97e92635.png)

Saat ini Firefox tercatat sebagai perangkat lunak yang banyak digunakan para pengguna rumahan karena sifatnya yang gratis dan open source (sumber terbuka). Proyek pembuatan Firefox ditujukan untuk mengembangkan sebuah web browser yang kecil, cepat, simple dan masih terbuka untuk dikembangkan karena sudah terpisah dari Mozilla Suite yang lebih besar (Manzur, 2010: 2).

2.4 Penelitian Terdahulu

Penelitian dilakukan oleh **Caiyuan Wang** dan **Ronglu Sun** (2009) dengan judul "*The Quality Test of Wire Bonding*". Berdasarkan penelitian yang dilakukan, dapat disimpulkan bahwa kualitas *wire bond* ditentukan oleh kekuatan *wire* yang menempel pada *bond pad* dan *lead finger*. Selain itu ada juga beberapa faktor yang mempengaruhinya sperti *wire sweep*, *wire sway*, *wire sagging*, *hell cracking*, *bond deformation* dan faktor lainnya. Kualitas *wire bond* dikontrol dengan metode *pull test*, *bond shear test*, *visual inspect* dan spesial tes kualitas lainnya.

Penelitian dilakukan oleh **Supruyanto**, **Refdilzon Yasra**, dan **Hery Irawan** (2013) dengan judul "Analisa Tingginya Broken Wire Di Mesin *Wire Bond* Untuk Peningkatan Produktivitas Dengan Pendekatan Metode Fishbone Diagram". Berdasarkan penelitian yang dilakukan, maka dapat disimpulkan bahwa untuk menurunkan *reject broken wire* pada mesin *Wire Bond* ESEC dengan cara melakukan proses perbaikan terhadap alat kerja *(tool)* yaitu memodifikasi *top clamp down holder* sehingga mengoptimalkan pandangan ataupun penglihatan oleh teknisi ataupun operator ketika menjalankan proses produksi terutama saat proses *hands-off inspection* terhadap material yang berada di daerah *bonding connection area*. Dari sisi produktivitas, pengurangan jumlah *broken wire* sebesar 45.950 unit pada bulan Mei 2013 diikuti penurunan persentase *downtime* mesin sebesar 2% dan penurunan jumlah *overall reject* sebesar 72.370 unit, lalu diikuti peningkatan rata-rata *output* produksi sebesar 255.703 unit dan peningkatan *yield* sebesar 0,10%.

Penelitian dilakukan oleh **Chong Leong Gan** & **U. Hashim** (2015) dengan judul "Evolutions of bonding wires used in semiconductor electronics: perspective over 25 years". Berdasarkan penelitian yang dilakukan, maka dapat disimpulkan bahwa perkembangan teknologi *wire bond* menggunakan benang emas (*gold wire*) dan benang tembaga (*copper wire*) sudah berjalan lebih dari 25 tahun. Teknologi terus berkembang dan sekarang sudah menggunakan teknologi benang perak (*silver wire*). Gold wire bonding merupakan teknologi pertama yang diperkenalkan dalam teknologi semikonduktor, tetapi karena harga dalam pembuatan meningkat maka dibuatlah teknologi menggunakan *copper wire bonding* sehingga bisa menekan biaya pembuatan. Tetapi *copper wire bonding* mengalami masalah dalam

temperature cycle test dan *pressure cooker test*. Sedangkan *silver wire bonding* memiliki sifat mekanikal seperti *gold wire bonding* serta mempunyai konduktivitas listrik dan panas lebih baik dari *copper wire bonding*, tetapi teknologi *silver wire bonding* membutuhkan lebih gas argon dalam pembuatannya..

Penelitian dilakukan oleh **RS Sethu** (2012) dengan judul "Reducing Non-Stick On Pad For Wire Bond: A Review". Berdasarkan penelitian yang dilakukan, problem *Non-Stick On Pad* (*NSOP*) disebabkan karena beberapa faktor yaitu: *wafer fabrication*, *assembly causes*, *wire bond failure mode*.

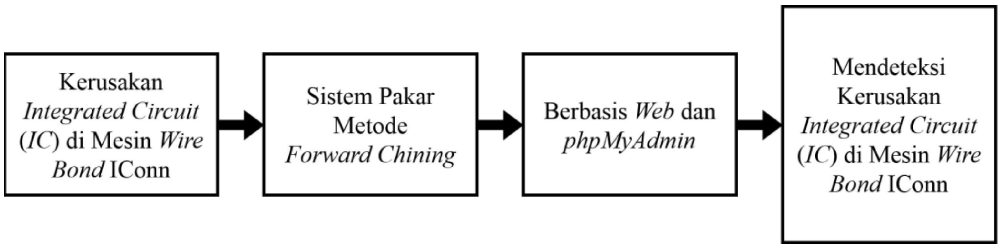
Penelitian dilakukan oleh **Rosmawati Tamin** (2015) dengan judul "Sistem Pakar untuk Diagnosa Kerusakan Pada Printer Menggunakan Metode *Forward Chaining*". Berdasarkan penelitian yang dilakukan, dapat disimpulkan bahwa aplikasi yang dibangun ini memudahkan para pengguna printer jenis canon untuk mengetahui penyebab, akibat dan gejala-gejala yang ditimbulkan dari kerusakan printer, memudahkan para pengguna printer jenis canon untuk mencari solusi kerusakan printer, memudahkan para pengguna printer untuk mendapatkan informasi mengenai cara merawat printer dengan baik melalui penyajian informasi berita yang terdapat dalam *website* aplikasi mendeteksi kerusakan pada printer dan memudahkan para teknisi untuk memperbaiki printer. Sistem ini tidak dapat dijadikan dasar utama dalam perbaikan sistem yang sedang berjalan, melainkan harus terus mengevaluasi sistem baru ini sehingga menghasilkan sistem yang lebih sempurna.

Penelitian dilakukan oleh **Adi Suwondo** (2014) dengan judul "Sistem Pakar Sebagai Alat Bantu Mengatasi Masalah (Studi Kasus Kerusakan Sepeda Motor)".

Berdasarkan penelitian yang dilakukan, dapat disimpulkan bahwa sistem pakar dalam penggunaan sehari-hari dirasa sangat perlu untuk menjawab kebutuhan pengguna dalam menyelesaikan permasalahan, tentunya dengan masalah yang spesifik dan memiliki tingkat keakuratan yang tinggi, sehingga kebutuhan seorang pakar dalam setiap permasalahan dapat diwakilkan oleh sistem pakar berbasis komputer. Sistem pakar ini juga diharapkan dapat membantu para mekanik dalam memecahkan permasalahan tentang kerusakan sepeda motor. Rule yang dibuat memiliki tingkat akurasi yang cukup, sehingga dalam sesi konsultasi, kemungkinan seorang pengguna dengan masalahnya dapat didiagnosis dan dapat diberikan saran untuk penyelesaiannya.

2.5 Kerangka Pemilkiran

Kerangka pemikiran memuat pemikiran terhadap alur yang dipahami sebagai acuan dalam pemecahan masalah yang diteliti secara logis dan sistematis. Kerangka berfikir yang baik akan menjelaskan secara teoritis pertautan antar variabel yang diteliti (Sugiyono, 2014: 60). Berikut ini adalah kerangka pemikiran yang menjadi dasar dalam penelitian ini:



Gambar 2.36 Kerangka Penelitian
(Sumber: Penelitian 2017)

Kerusakan yang terjadi pada *Integrated Circuit* (*IC*) di mesin *Wire Bond* IConn yaitu *Ball Non Stick*, *Damaged Wire*, dan *Lifted Wedge* menjadi sumber data penelitian. Kemudian data tersebut akan diolah dengan metode *forward chaining* sehingga menghasilkan sebuah sistem pakar yang mempunyai kemamapuan seperti seorang pakar mesin *Wire Bond* IConn. Sistem pakar tersebut diimplementasikan kedalam sebuah aplikasi *web* dengan *phpMyAdmin* sebagai basis datanya. Aplikasi tersebut memiliki kecerdasan layaknya seorang pakar yang bisa mendeteksi kerusakan *IC* di mesin *Wire Bond* IConn sehingga didapatkan solusi dari permasalahan mesin *Wire Bond* IConn.

BAB III
METODE PENELITIAN

Metode penelitian diperlukan untuk menjawab masalah dan menguji hipotesa. Untuk itu, dibagian ini perlu ditetapkan metode penelitian apa yang digunakan, apakah metode survei, metode eksperimen, metode kasus, metode penelitian dan pengembangan, atau kaji tindakan (*action research*) (Sudaryono, 2015: 157).

Penelitian ini dilakukan dengan cara diagnosis terhadap beberapa penyebab kerusakan *IC* di mesin *Wire Bond* IConn PT UNISEM Batam. Dari beberapa analisa yang dilakukan, maka data diolah dan dibuat sistem pakar untuk mengatasi beberapa penyebab kerusakan *IC* di mesin *Wire Bond* IConn PT UNISEM Batam. Data-data yang didapat digunakan untuk membuat *rule* sebuah aplikasi sistem pakar *forward chaining* untuk menduplikasikan kemampuan pakar, sehingga bisa dimanfaatkan teknisi di area produksi *Wire Bond* PT UNISEM Batam.

3.1 Desain Penelitian

Desain penelitian menggambarkan apa yang akan dilakukan oleh peneliti dalam terminologi teknis. Dalam hal ini, desain penelitian harus mencakup antara lain tahapan yang akan dilakukan, informasi mengenai cara penarikan *sampel* bila diperlukan survei primer, besarnya sampel, metode pengumpulan data, instrumen

penelitian, prosedur pengujian validitas, dan reliabilitas instrumen (Kuncoro, 2009 dalam Sudaryono, 2015: 157).

Berikut ini adalah gambar konsep dasar desain penelitian yang digunakan dalam penelitian ini (Gambar 3.1):

Gambar 3.1 Metode penelitian
(Sumber: Data Penelitian, 2017)

Berikut ini adalah langkah-langkah yang dilakukan dalam penelitian yaitu sebagai berikut:

1. Identifikasi Masalah

   Tahap pertama dalam melakukan penelitian adalah mengidentifikasi masalah. Pada tahapan identifikasi masalah, dilakukan diagnosa terhadap faktor-faktor penyebab kerusakan *IC* di area produksi *Wire Bond* PT UNISEM Batam. Setelah identifikasi masalah selesai, kemudian dilakukan perumusan masalah.

2. Perumusan Masalah

   Perumusan masalah bertujuan untuk menspesifikasikan masalah yang ada sehingga dapat dijawab dengan baik melalui penelitian. Setelah masalah dirumuskan, maka tahapan berikutnya adalah menentukan tujuan penelitian.

3. Menentukan Tujuan Penelitian

   Tujuan penelitian yaitu mengetahui bagaimana sistem pakar forward chaining berbasis web mendeteksi kerusakan *IC* di area produksi *Wire Bond* PT UNISEM Batam. Setelah tujuan penelitian ditentukan, maka langkah selanjutnya adalah mengumpulkan data.

4. Mengumpulkan Data

   Dalam pengunpulan data, peneliti mengambil data dengan melakukan wawancara dengan seorang pakar mesin *Wire Bond* (Moch. Ali Rohman, S.T.) dan studi literature yaitu mengumpulkan buku yang berkaitan dengan kecerdasan buatan, sistem pakar, pemrograman *web*, *UML*, *manual book* mesin *Wire Bond* IConn, dan *Asembly Spesification* (*AS*) PT UNISEM Batam tentang kerusakan *IC*. Setelah bahan-bahan yang mendukung dalam penelitian didapatkan maka tahapan selanjutnya adalah melakukan studi literatur.

5. Melakukan Studi Literatur

   Mempelajari dan mengkaji buku-buku yang berhubungan dengan penelitian sehingga menambah wawasan dan pengetahuan tentang objek yang akan diteliti. Dalam hal ini peneliti mempelajari tentang kecerdasan buatan, sistem pakar, *UML*, membangun pemrograman *web*, *manual book* mesin *Wire Bond* KNS IConn, dan *Asembly Spesification* (AS) PT UNISEM Batam tentang

kerusakan *IC*. Setelah studi literatur dilakukan, maka tahapan selajutnya adalah menganalisa data yang didapat.

6. Menganalisa Data

   Setelah data-data yang berhubungan dengan penelitian didapatkan, maka langkah selanjutnya dengan mengalisa data-data yang diperoleh. Pada tahapan mengalalisa data, ditentukan terlebih dahulu operasional variabel dan indikator yang akan digunakan supaya mempermudah dalam proses pengolahan data. Jika operasional variable dan indikator sudah ditentukan maka tahapan berikutnya adalah mengolah data.

7. Mengolah Data Menggunakan Sistem Pakar *Metode Forward Chaining*

   Data-data yang sudah dianalisa kemudian diolah menggunakan sistem pakar *forward chaining* untuk membuat *rule-rule* yang akan digunakan saat sistem pakar melakukan penelusuran berdasarkan gelaja yang ada. Setelah *rule-rule* sebagai *knowlege base* sistem pakar dibuat maka tahapan selanjutnya adalah mengimplementasikan sistem pakar kedalam pemrograman berbasis *web*.

8. Mengimplementasikan Sistem Pakar Dalam Bentuk Pemrograman Berbasis *Web*

   Setelah data yang diperoleh diolah menggunakan sistem pakar metode forward chaining, data-data tersebut disusun dan dibuat kode-kode untuk mempermudah pembuatan program. Pembuatan aplikasi sistem pakar dilakukan mulai dari perancangan sistem yaitu: desain *UML*, desain antarmuka, desain *database*, dan desain basis pengetahuan (aturan). Setelah itu proses pengodean menggunakan bahasa *HTML, PHP, CSS, JavaScript*, dan *jQuery* dengan *editor Notepad++*.

Langkah terakhir dalam pembuatan aplikasi sistem pakar adalah membuat *database* dengan menggunakan *phpMyAdmin* yang ada pada program *XAMPP*. Dalam pembuatan program, peneliti menggunakan referensi program dari buku karya Bunafit Nugroho yang berjudul Aplikasi Sistem Pakar dengan PHP & Editor Dreamweaver, juga arahan mengenai tampilan program oleh Ibu Anggia Dasa Putri, S.Kom., M.Kom. sebagai dosen pembimbing dan dosen pengajar mata kuliah Pemrograman web, serta masukan dan ajaran dari Mas Dedi Suhendra, S.Kom yang pernah melakukan penelitian dengan judul tesis Sistem Pakar Untuk Mendeteksi Kerusakan *Air Conditioner* (*AC*) Menggunakan Metode *Forward Chaining* Berbasis *Web*. Sistem pakar yang sudah dibuat kemudian diuji keakuratannya pada tahapan pengujian hasil penelitian. Setelah program dibuat, maka aplikasi sistem pakar tersebut diimplementasikan pada salah satu komputer yang sudah terinstal aplikasi *XAMPP* ada di area produksi *Wire Bond* PT UNISEM Batam.

9. Pegujian Hasil Penelitian

   Tahapan ini adalah pengujian aplikasi yang telah dibuat. Pengujian dilakukan oleh pakar yaitu sorang engineer proses produksi (Moch. Ali Rohman, S.T.). Pengujian dilakukan dengan 2 sistem yaitu *black box testing* dan pengujian keakuratan dengan pakar. Pengujian *black box testing* yaitu menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian dimaksudkan untuk mengetahui apakah fungisi-fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan. Setelah lolos pengujian aplikasi maka tahapan berikutnya adalah dengan

pengujian tingkat keakuratan sistem pakar. Pengujian dengan pakar dilakukan dengan merujuk pada data diagnosa gejala kerusakan unit yang telah disusun (data dari *Line Distribution Alert*) sebanyak 10 studi kasus. Kemudian hasil pengujian dihitung tingkat persentasi keakuratannya. Setelah pengujian dilakukan maka tahapan berikutnya adalah menarik kesimpulan.

10. Menarik Kesimpulan

    Tahapan terakhir dalam penelitian ini yaitu menyimpulkan hasil penelitian yang berisi jawaban singkat terhadap rumusan masalah berdasarkan data-data yang ada. Dalam tahap ini, peneliti juga memberikan saran yang penting untuk membantu dalam memecahkan permasalahan yang ada.

3.1.1 Teknik Pengumpulan Data

Teknik pengumpulan data merupakan langkah yang paling strategis dalam penelitian, karena tujuan utama dari penelitian adalah mendapatkan data (Sugiyono, 2012:224). Langkah-langkah yang dilakukan untuk pengumpulan data dalam penelitian ini adalah sebagai berikut:

1. Studi literatur

   Studi literatur yaitu mengumpulkan data dan menambah wawasan dengan belajar dari iAS1006 yang berisi cara penangana kerusakan pada mesin *Wire Bond*, iAS1007 yang berisi spesifikasi tentang kerusakan *IC* yang ada di PT UNISEM Batam, mempelajari tentang *manual book* mesin *Wire Bond* IConn, mempelajari buku tentang pembuatan aplikasi sistem pakar bebasis *web*, membaca jurnal penelitian yang berkaitan dengan topik bahasan penelitian.

2. Wawancara

   Melakukan wawancara dengan Moch. Ali Rochman S.T. yang bekerja sebagai *Process Engineer Wire Bond* dan berpengalaman lebih dari 6 tahun dalam menganalisa dan menangani kerusakan *IC* di PT UNISEM Batam. Wawancara dilakukan dengan memberikan beberapa pertanyaan yang sudah disusun sebelumnya, kemudian jawaban tersebut ditulis oleh peneliti pada *form* wawancara. Semua kegiatan wawancara dengan pakar direkam dan di foto sebagai bukti penelitian.

3.2 Operasional Variabel

Variabel penelitian adalah suatu atribut atau sifat atau nilai dari orang, objek atau kegiatan yang mempunyai variasi tertentu yang ditetapkan oleh peneliti untuk dipelajari dan kemudian ditarik kesimpulannya (Sugiyono, 2014:38).

Berikut ini adalah tabel hubungan antara variabel dan indikator dalam penelitian ini yaitu (Tabel 3.1):

Tabel 3.1 Tabel Hubungan Variabel dan Indikator

Variabel	Indikator
Kerusakan mesin *Wire Bond* Iconn	*Ball non stick*
	Lifted wedge
	Damaged wire

(Sumber: Data Penelitian, 2017)

Dalam Tabel 3.1 diatas menjelaskan hubungan anatar variabel dan indikator. Variabelnya adalah kerusakan mesin *Wire Bond* IConn, sedangkan indikatornya adalah 3 kerusakan *IC* yaitu: *Ball Non Stick, Lifted Wedge*, dan *Damaged Wire*.

Pada Tabel 3.2 menjelaskan indikator, penyebab, dan solusi kerusakan *IC* dimesin *Wire Bond* KNS IConn.

Tabel 3.2 Tabel Penyebab dan Solusi Kerusakan *IC*

Indikator	Penyebab	Solusi
Ball Non Stick	Tidak *optimize 1st bond* parameter	Gunakan maksimum *ball* parameter dan kurangi FAB parameter sesuai DC (*Data Collector*)
	Downholder kotor (terdapat *foreign material*)	Bersihkan *downholder*
	Floating die pad	Atur ketinggian *top clamp*
	Material yang *reject*	Hubungi *Process Engineer* untuk membuat *optimize parameter* dan membuat laporan ke *Customer*
	Temperatur pada *Heater Block* tidak tepat	Naikkan temperatur *offset* pada menu *heater block*
Lifted Wedge	*Lead frame* terkontaminasi	Dilakukan plasma *cleaning* pada material dengan program 3
	Floating pada permukaan *lead*	Cek kondisi *red tape* pada *top clamp*, cek kebersihan permukaan *downholder*, atur ketinggian *top clamp*
	Kurang *optimize wedge bond* parameter	Gunakan maksimum *wedge* parameter sesuai *DC* (*Data Collector*)
	Impedansi *capillary* terlalu tinggi	Kalibrasi ulang *USG impedance* pada mesin
	Temperatur pada *Heater Block* tidak tepat	Naikkan temperatur *offset* pada menu *heater block*
Damaged Wire	Tidak pas/sejajar *indexer table* dengan slot *magazine output*	*Setting magazine output*
	Die pad tidak tepat pada *cavity Down Holder*	Lakukan *teaching* ulang *OLC* pada mesin

Tabel 3.2 Lanjutan

Damaged Wire	Tidak lancar strip pada *indexer table* (*indexer tabel* terlalu sempit)	Atur lebar *indexer track* sesuai dengan lebar *lead frame*
	Tidak *optimize loop* parameter	*Optimize loop* parameter sesuai DC dan ukur ketinggian *loop*
	Tertabrak oleh *top clamp*	Atur ketinggian *top clamp* mencapai maksimal dan perlambat *step indexer*
	Tertabrak oleh capillary	Cek jarak *bondpad opening*, ganti aplikasi *capillary* lebih kecil atau ganti *bonding sequence*

(Sumber: Data Penelitian 2017)

Pada Tabel 3.2 diatas, masing-masing indikator memiliki beberapa penyebab kerusakan. Dari penyebab kerusakan tersebut, juga disertakan solusi dari setiap penyebab kerusakan.

3.3 Perancangan Sistem

A.S. dan Shalahuddin (2011: 23) perancangan sistem merupakan upaya untuk mengkonstruksi sebuah sistem yang memberikan kepuasan akan spesifikasi kebutuhan fungsional, memenuhi target, memenuhi kebutuhan secara implisit atau eksplisit dari segi performa maupun penggunaan sumber daya, kepuasan batasan pada proses desain dari segi biaya, waktu, dan perangkat.

3.3.1 Desain Basis Pengetahuan

Peneliti melakukan proses akuisisi pengetahuan dengan mengumpulkan pengetahuan dan fakta dari sumber-sumber yang tersedia. Sumber pengetahuan dan fakta diperoleh melalui wawancara dengan seorang *Process Engineer* yang berpengalaman, selain itu peneliti melakukan studi literatur tentang materi yang berkaitan mesin *Wire Bond* IConn dan penyebab-penyebab kerusakan *IC*.

Sumber pengetahuan yang telah didapatkan dari seorang pakar ditampilkan dalam table nama kerusakan *IC* (Tabel 3.3), table penyebab dan solusi kerusakan *IC* (Tabel 3.4), table gejala kerusakan *IC* (Tabel 3.5) dan table aturan (Tabel 3.6) berikut ini:

Tabel 3.3 Tabel Nama Kerusakan *IC*

Kode	Indikator
KU01	Ball non stick
KU02	Lifted wedge
KU03	Damaged wire

(Sumber: Data Penelitian 2017)

Pada Tabel 3.3 diatas, masing-masing indikator diberikan kode untuk membedakan satu dengan yang lainnya. Kode KU01 untuk kerusakan *IC* berupa *Ball Non Stick*, Kode KU02 untuk kerusakan *IC* berupa *Lifted Wedge*, dan Kode KU03 untuk kerusakan *IC* berupa *Damaged Wire*.

Tabel 3.4 Tabel Penyebab dan Solusi Kerusakan *IC*

Kode	Penyebab	Solusi
PP01	Tidak *optimize 1st bond* parameter	Gunakan maksimum *ball* parameter dan kurangi FAB parameter sesuai *DC* (*Data Collector*)

Tabel 3.4 Lanjutan

PP02	Temperatur pada *Heater Block* tidak tepat	Cek temperatur *HB*, naikkan temperatur *offset* pada menu *heater block*
PP03	Material yang *reject*	Hubungi *Process Engineer* untuk membuat o*ptimize* parameter dan membuat laporan ke *Customer*
PP04	*Downholder* kotor (terdapat *foreign material*)	Bersihkan *downholder*
PP05	*Floating die pad*	Atur ketinggian *top clamp*
PP06	*Lead frame* terkontaminasi	Dilakukan plasma *cleaning* pada material dengan program 3, *optimize wedge* parameter
PP07	*Floating* pada permukaan *lead*	Cek kondisi *red tape* pada *top clamp*, cek kebersihan permukaan *downholder*, atur ketinggian *top clamp*
PP08	Kurang *optimize wedge bond* parameter	Gunakan maksimum *wedge* parameter sesuai *DC* (*Data Collector*)
PP09	Impedansi *capillary* terlalu tinggi	Kalibrasi ulang *USG impedance* pada mesin
PP10	Tidak pas/sejajar *indexer table* dengan slot *magazine output*	*Setting magazine output*
PP11	*Die pad* tidak tepat pada *cavity Down Holder*	Lakukan *teaching* ulang *OLC* pada mesin
PP12	Tidak lancar strip pada *indexer table* (*indexer tabel* terlalu sempit).	Atur lebar *indexer track* sesuai dengan lebar *lead frame*
PP13	Tidak *optimize loop* parameter	*Optimize loop* parameter sesuai DC dan ukur ketinggian loop.
PP14	Tertabrak oleh *top clamp*	Atur ketinggian *top clamp* mencapai maksimal dan perlambat *step indexer*
PP15	Tertabrak oleh *capillary*	Cek jarak *bondpad opening*, ganti aplikasi capillary lebih kecil atau ganti *bonding sequence*

(Sumber: Data Penelitian 2017)

Pada Tabel 3.4, menunjukkan bahwa masing-masing diberikan kode untuk membedakan antara satu penyebab dengan yang lainnya. Dari masing-masing penyebab diberikan solusi untuk mengatasi penyebab kerusakan *IC*.

Tabel 3.5 Tabel Gejala Kerusakan

Kode Gejala	Nama Gejala
GG01	*Reading ball shear test* rendah *(lower control level)*
GG02	Bentuk ball bond normal dengan ukuran antara 2x - 5x diameter *wire*
GG03	ketebalan ball bond lebih dari 1x diameter *wire*
GG04	Sebagian ball bond pada unit terlihat tipis (*flat ball*)
GG05	Terdapat kelainan warna pada *bondpad*
GG06	Terdapat tanda gores pada bagian bawah *die pad*
GG07	Terdapat kelainan warna pada permukaan *lead*
GG08	Tidak terlihat imprint *capillary* di *wedge*
GG09	Terlihat *peeloff* pada *wedge flare*
GG10	Ukuran *wedge flare* kurang dari 1.3x diameter *wire*
GG11	Reading wedge pull rendah
GG12	Terlihat halfmoon imprint capillary
GG13	Ada bekas luka pada ujung *side rail* sebelah kanan atas atupun kiri atas
GG14	Ada bekas luka pada *tie bar*
GG15	Ada bekas luka pada *side rail* bagian bawah atau atas sebelah kanan
GG16	Tidak ada bekas luka pada permukaan *lead frame*
GG17	Terlihat goresan pada *wire*
GG18	Ada indikasi *vacum error*
GG19	Bentuk *wire* dan *loop* tertekan bengkok ke kiri
GG20	Bentuk *wire* bengkong di daerah necking (1st *kink*)
GG21	Terlihat *wire* melengkung kebawah (*sagging wire*)
GG22	*Strip* tidak lancar di *indexer table*
GG23	Tidak terdapat luka gores pada *strip*

(Sumber: Data Penelitian 2017)

Pada Tabel 3.5 diatas, menunjukkan pengkodean dari masing-masing gejala kerusakan *IC* supaya membedakan gejala kerusakan satu dengan yang lainnya.

Data aturan berisi relasi antara data-data nama kerusakan *IC*, penyebab kerusakan *IC* dan gejala kerusakan *IC* yang telah diberi kode sebelumnya. Data-data yang didapat kemudian dibuat relasi antar data sehingga menghasilkan *rule-rule* dalam sistem pakar yang memudahkan penyusunan basis pengetahuan. Berikut ini adalah tabel data aturan (Tabel 3.6):

Tabel 3.6 Tabel Data Aturan

Kode Indikator	Kode Penyebab	Kode Gejala
KU01	PP01	GG01, GG03
KU01	PP02	GG01,GG02
KU01	PP03	GG01, GG02, GG05
KU01	PP04	GG04, GG06
KU01	PP05	GG04
KU02	PP06	GG07, GG10, GG11
KU02	PP07	GG09, GG12
KU02	PP08	GG08, GG11
KU02	PP09	GG10, GG11
KU03	PP10	GG13, GG21
KU03	PP11	GG14, GG18
KU03	PP12	GG15, GG22
KU03	PP13	GG16, GG17
KU03	PP14	GG19, GG23
KU03	PP15	GG20, GG23

(Sumber: Data Penelitian 2017)

Pada Tabel 3.6 diatas, indikator, penyebab dan gejala dibuat kode yang berbeda-beda. Pengkoden ini dibuat untuk memudahkan dalam penyusunan kaidah produksi yang akan dibuat. Setiap penyebab mempunyai gejala yang berbeda, tetapi ada beberapa penyebab mempunyai salah satu ciri gejala sama dengan penyabab

lainnya. Urutan pengkodean penyebab disesuaikan/dikelompokkan sesuai dengan kode kerusakan *IC* (Kode Indikator).

3.3.2 Pembentukan aturan

Sutojo, dkk. (2016: 9) setiap rule terdiri dari dua bagian, yaitu bagian *IF* disebut *evidence* (fakta-fakta) dan bagian *THEN* disebut hipotesis atau kesimpulan. Representasi pengetahuan pada dasarnya berupa aturan *IF – THEN* dalam sebuah sistem pakar. Data-data yang sudah disusun dalam tabel aturan (Tabel 3.7), dirangkai menjadi suatu kaidah aturan dalam sistem pakar. Berikut ini adalah tabel aturan *inference* dalam sistem pakar ini:

Tabel 3.7 Aturan *Inference*

Aturan	Kaidah
R01	IF GG01 AND GG03 THEN PP01
R02	IF GG01 AND GG02 THEN PP02
R03	IF GG01 AND GG02 AND GG05 THEN PP03
R04	IF GG04 AND GG6 THEN PP04
R05	IF GG04 THEN PP05
R06	IF GG07 AND GG10 AND GG11 THEN PP06
R07	IF GG09 AND GG12 THEN PP07
R08	IF GG10 AND GG11 THEN PP08
R09	IF GG08 AND GG10 THEN PP09
R10	IF GG13 AND GG21 THEN PP10
R11	IF GG14 AND GG18 THEN PP11
R12	IF GG15 AND GG22 THEN PP12
R13	IF GG16 AND GG17 THEN PP13
R14	IF GG19 AND 23 THEN PP 14
R15	IF GG20 AND GG23 THEN PP15

(Sumber: Data Penelitian 2017)

Berdasarkan Tabel 3.7, dapat disimpulkan bahwa terdapat 15 aturan (*rule*). Berikut ini adalah pembahasannya:

1. *Rule* 1

   Jika gejala kerusakan adalah *reading ball shear test* rendah (*lower control level*) (GG1) dan ketebalan *ball bond* lebih dari 1x diameter *wire* (GG03) maka kerusakan disebabkan oleh tidak *optimize 1st bond parameter* (PP01).

2. *Rule* 2

   Jika gejala kerusakan adalah *reading ball shear test rendah* (*lower control level*) (GG01) dan bentuk *ball bond* normal dengan ukuran antar 2x - 5x diameter *wire* (GG02) maka kerusakan disebabkan oleh temperatur pada *heater block* tidak tepat (PP02).

3. *Rule* 3

   Jika gejala kerusakan adalah *reading ball shear test* rendah (*lower control level*) (GG01) dan bentuk *ball bond normal* dengan ukuran antara 2x - 5x (GG02) dan terdapat kelainan warna pada *bond pad* (GG05) maka kerusakan disebabkan oleh material yang *reject* (PP03).

4. *Rule* 4

   Jika gejala kerusakan adalah sebagian *ball bond* pada unit terlihat tipis (*flat ball*) (GG04) dan terdapat tanda gores pada bagian bawah *die pad* (GG06) maka kerusakan disebabkan oleh *downholder* kotor (terdapat *foreign material*) (PP04).

5. *Rule* 5

   Jika gejala kerusakan adalah sebagian *ball bond* pada unit terlihat tipis (*flat ball*)

(GG04) maka kerusakan disebabkan oleh *floating die pad* (PP05).

6. *Rule* 6

   Jika gejala kerusakan adalah terdapat kelainan warna pada permukaan *lead* (GG07) dan ukuran *wedge flare* kurang dari 1.3x diameter *wire* (GG10) dan *reading wedge pull* rendah (GG11) maka kerusakan disebabkan oleh *lead frame* terkontaminasi (PP06).

7. *Rule* 7

   Jika gejala kerusakan adalah terlihat *peeloff* pada *wedge flare* (GG09) dan terlihat *halfmoon imprint capillary* (GG12) maka kerusakan disebabkan oleh *floating* pada permukaan *lead* (PP07).

8. *Rule* 8

   Jika gejala kerusakan adalah ukuran *wedge flare* kurang dari 1.3x diameter *wire* (GG10) dan *reading wedge pull* rendah (GG11) maka kerusakan disebabkan oleh kurang *optimize wedge bond parameter* (PP08).

9. *Rule* 9

   Jika gejala kerusakan adalah tidak terlihat *imprint capillary* di *wedge* (GG08) dan ukuran *wedge flare* kurang dari 1.3x diameter *wire* (GG10) maka kerusakan disebabkan oleh impedansi *capillary* terlalu tinggi (PP09).

10. *Rule* 10

    Jika gejala kerusakan adalah ada bekas luka pada ujung *side rail* sebelah kanan atas atupun kiri atas (GG13) dan terlihat *wire* melengkung kebawah (*sagging wire*) (GG21) maka kerusakan disebabkan oleh tidak pas/sejajar *indexer table* dengan *slot magazine output* (PP10).

11. *Rule* 11

    Jika gejala kerusakan adalah ada bekas luka pada *tie bar* (GG14) dan ada indikasi *vacum error* (GG18) maka kerusakan disebabkan oleh *die pad* tidak tepat pada *cavity down holder* (PP11).

12. *Rule* 12

    Jika gejala kerusakan adalah ada bekas luka pada *side rail* bagian bawah atau atas sebelah kanan (GG15) dan strip tidak lancar di *indexer table* (GG22) maka kerusakan disebabkan oleh *indexer tabel* terlalu sempit (PP12).

13. *Rule* 13

    Jika gejala kerusakan adalah tidak ada bekas luka pada permukaan *lead frame* (GG16) dan terlihat goresan pada *wire* (GG17) maka kerusakan disebabkan oleh tidak *optimize loop parameter* (GG13).

14. *Rule* 14

    Jika gejala kerusakan adalah bentuk *wire* dan *loop* tertekan bengkok ke kiri (GG19) dan tidak terdapat luka gores pada *strip* (GG23) maka kerusakan disebabkan oleh tertabrak oleh *top clamp* (PP14).

15. *Rule* 15

    Jika gejala kerusakan adalah bentuk *wire* bengkong di daerah *necking* (*1st kink*) (GG20) dan tidak terdapat luka gores pada *strip* (GG23) maka kerusakan disebabkan oleh tertabrak oleh *capillary* (PP15).

Setelah tabel aturan *inference* (Table 3.7) disusun, maka langkah selanjutnya adalah membuat tabel keputusan. Berikut ini adalah tabel relasi gejala dan penyebab (Tabel 3.8) dari sistem pakar yang akan dibuat:

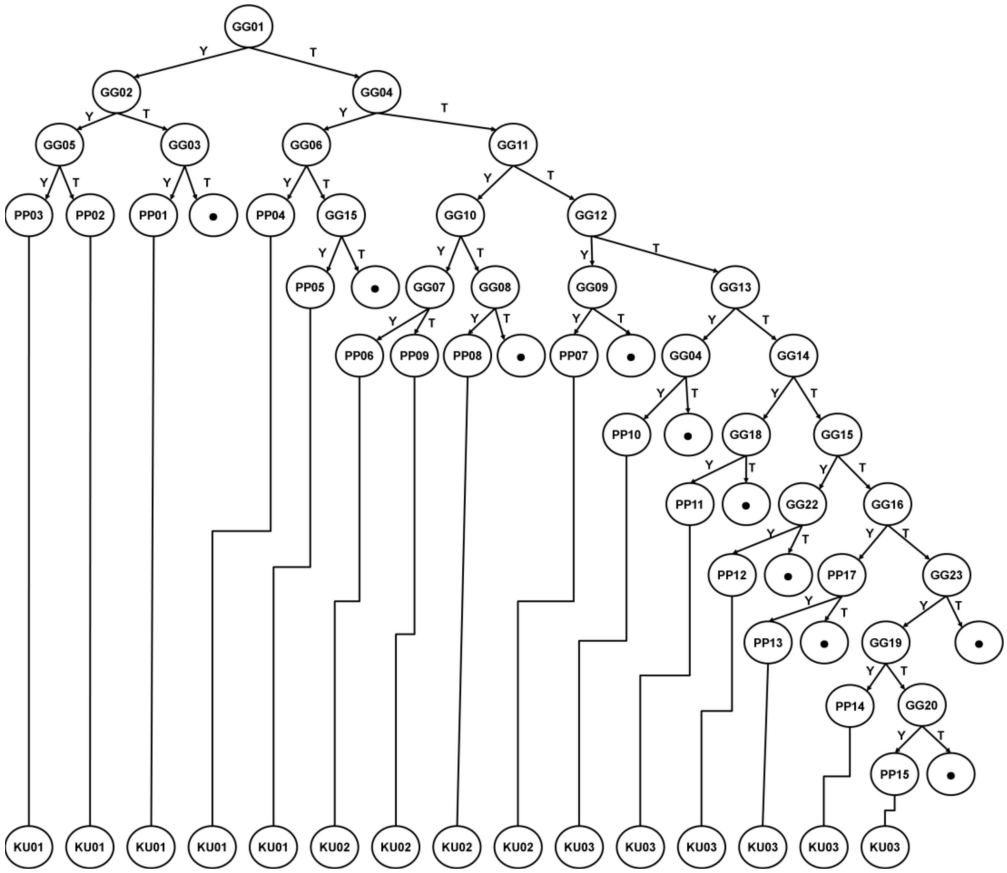
Tabel 3.8 Tabel Relasi Gejala dan Penyebab

Gejala	KU01					KU02				KU03					
	PP01	PP02	PP03	PP04	PP05	PP06	PP07	PP08	PP09	PP10	PP11	PP12	PP13	PP14	PP15
GG01	√	√	√												
GG02		√	√												
GG03	√														
GG04				√	√										
GG05			√												
GG06				√											
GG07						√									
GG08								√							
GG09							√								
GG10						√			√						
GG11						√		√	√						
GG12							√								
GG13										√					
GG14											√				
GG15												√			
GG16													√		
GG17													√		
GG18											√				
GG19														√	
GG20															√
GG21										√					
GG22												√			
GG23														√	√

(Sumber: Data Penelitian, 2017)

Pada Tabel 3.8 diatas, kolom penyebab (PP) dikelompokkan dan diurutkan berdasarkan kode kerusakan *IC* (KU), setelah itu diberi tanda centang untuk baris kode gejala (GG) yang memenuhi aturan dari masing-masing penyebab. Hal ini dibuat untuk memudahkan dalam menyusun aturan kaidah produksi sistem pakar yang akan dibuat.

Setelah disusun tabel relasi gejala dan penyebab (Tabel 3.8**)** diatas maka dapat dibuat pohon keputusan (Gambar 3.2) sebagai berikut:



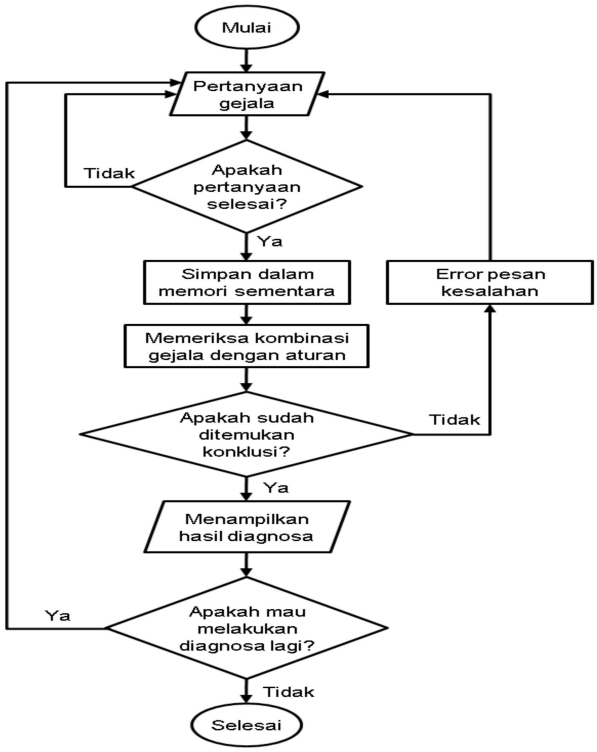
Gambar 3.2 Pohon Keputusan
(Sumber: Data Penelitian, 2017)

Pada Gambar 3.2 menunjukkan pohon keputusan yang memperlihatkan hubungan antara gejala dan penyebab. Penelusuran menggunakan sistem *dept first search*. *Depth first search* merupakan metode pencarian yang dilakukan pada suatu simpul dalam setiap level yang dimulai dari kiri. Jika pada level yang terdalam solusi belum ditemukan, maka pencarian dilanjutkan pada simpul sebelah kanan dan simpul yang kiri dapat dihapus dari memori (Suyanto, 2014 dalam jurnal penelitian Harsono, dkk., 2015). Penelusuran dimulai dari gejala awal yaitu *reading ball shear test* rendah (*lower control level*) (GG01) . Gejala ini dipilih pertama karena untuk kerusakan *IC* yaitu *Ball Non Stick*, yang mengukur daya tempel *ball*

bond terhadap *bond pad* adalah *ball shear test*. Semakin tinggi hasil pengukuran *ball shear* maka kondisinya baik, jika hasilnya rendah maka kondisinya kurang baik (berpotensi untuk menghasilkan *reject Ball Non Stick*). Proses penelusuran dari pohon keputusan ditentukan berdasarkan jawaban yang dipilih oleh *user*. Apabila *user* menjawab "ya" (Y), maka penelusuran menuju ke simpul sebelah kiri level berikutnya yaitu bentuk *ball bond* normal dengan ukuran antara 2x - 5x diameter *wire* (GG02). *User* akan diberikan pertanyaan (GG02), jika menjawab "ya" (Y) maka pohon akan mengarah kesimpul kiri yaitu akan mengajukan pertanyaan lagi tentang bentuk ball bond normal dengan ukuran antara 2x - 5x diameter wire (GG05). Jika *user* menjawab "ya" (Y) maka ditemukan penyebab yaitu material yang *reject* (PP03) tetapi apabila menjawab "tidak" (T) maka jawaban diagnosanya adalah temperatur pada *Heater Block* tidak tepat (PP02). Jika user menjawab "tidak" (T) pada saat pertanyaan *ball bond* normal dengan ukuran antara 2x - 5x diameter *wire* (GG02), maka penelusuran akan menuju ke simpul sebelah kanan level berikutnya yaitu gejala ke 3 (GG03). Apabila pertanyaan ketebalan *ball bond* lebih dari 1x diameter *wire* (GG03) dijawab "ya" (Y) maka ditemukan penyebab yaitu tidak *optimize 1st bond* parameter, tetapi apabila menjawab "tidak" (T) maka akan menemukan simpul mati (•) yang berati tidak menghasilkan kesimpulan atau tidak menemukan kunklusi. Jika penelusuran sampai pada simpul mati (•) maka penelusuran akan kembali lagi ke kondisi awal yaitu melakukan penelusuran dari simpul pertama (GG1).

3.3.3 Struktur Kontrol (Mesin Inferensi)

Mesin inferensi dalam sistem pakar ini menggunakan metode penelusuran *forward chaining*, yaitu analisa penelusuran maju mulai dari fakta-fakta yang berupa ciri-ciri/gejala yang ditimbulkan pada kerusakan *IC* sehingga dapat meguji kebenaran hipotesis yaitu penyebab kerusakan, sehingga menghasilkan solusi dari penyebab kerusakan. Berikut ini f*lowchart* mesin inferensi sistem pakar *forward chining* digambarkan pada Gambar 3.3 berikut ini:



Gambar 3.3 Flowchart Mesin Inferensi Sistem Pakar
(Sumber: Data Penelitian, 2017)

Langkah-langkah yang digunakan dalam proses penelusuran masalah adalah sebagai berikut:

1. Mulai mengakses sistem.

2. Mengajukan pertanyaan tentang gejala-gejala kerusakan *IC* di mesin *Wire Bond* KNS IConn.
3. Setelah jawaban pertanyaan dijawab oleh user, maka jawaban akan disimpan kedalam tabel gejala kerusakan sementara.
4. Sistem akan memeriksa jawaban tersebut dan membandingkannya denga rule yang sudah dibuat sehingga menghasilkan jawaban, jika semua pertanyaan yang diajukan kepada user tetapi belum memenuhi konklusi pada sistem, maka akan keluar pesan pengulangan diagnosa.
5. Jika jawaban pertanyaan dari user sesuai dengan rule yang ada pada database, maka aplikasi akan menampilkan diagnosa berupa gejala kerusakan, penyebab kerusakan, dan solusi untuk memperbaiki kerusakan mesin Wire Bond KNS IConn.
6. Ada pertanyaan untuk melakukan pengulangan diagnosa, jika menjawab "Ya" maka akan kembali ke menu diagnosa awal dan jika menjawab "Tidak" maka diagnosa berhenti. Selesai.

3.3.4 Desain *UML* (*Unified Modeling Language*)

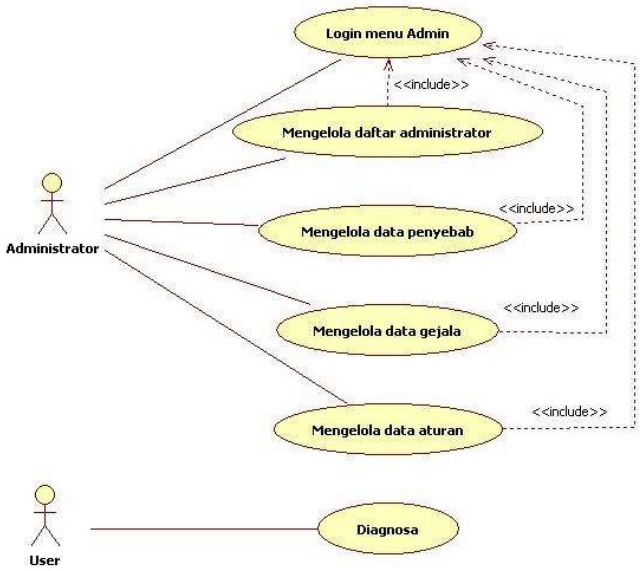
UML (*Unified Modeling Language*) adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement*, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek (A.S. dan Shalahuddin, 2011: 113).

Desain *UML* dibuat untuk memudahkan dalam pembuatan program. Pemodelan *UML* menggunakan alat bantu *software StarUML* versi 2.5.1. Berikut ini adalah diagram UML yang digunakan dalam perancangan program:

1. *Use Case Diagram*

Use case diagram merupakan pemodelan untuk menggambarkan kelakuan (*behavior*) sistem informasi yang akan dibuat. Diagram ini mendeskripsikan sebuah interaksi antara satu sistem atau lebih aktor dengan sistem informasi yang akan dibuat (A.S dan Shalahuddin, 2011: 130-131).

Pada Gambar 3.4 berikut ini menggambarkan *use case diagram* yang dirancang dalam penelitian ini:



Gambar 3.4 Use Case Diagram
(Sumber: Data Penelitian, 2017)

Terdapat 2 aktor yaitu *administrator* dan u*ser*. *Administrator* melakukan interaksi dengan sistem berupa mengelola daftar *administrator*, mengelola data penyebab, mengelola data gejala, dan mengelola data aturan. Semua interaksi yang dilakukan

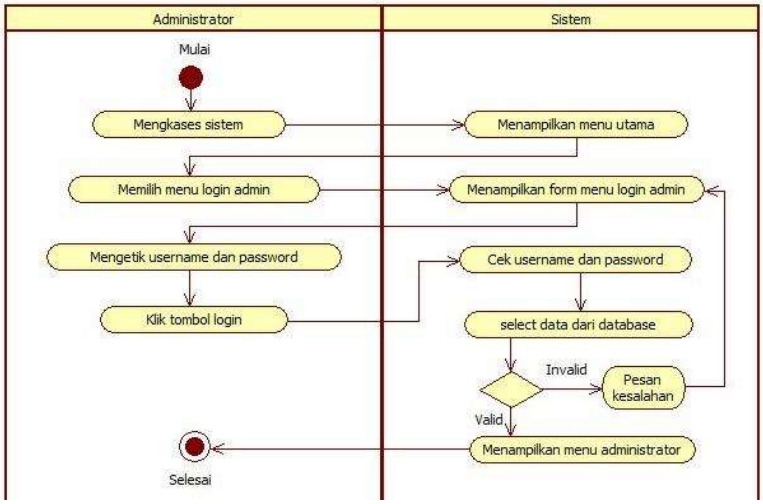
administrator dilakukan setelah *login* pada Menu Admin. Sedangkan *user* berinteraksi dengan sistem yaitu melakukan diagnosa. Sebelum diagnosa dilakukan, user dimunta untuk memasukkan nama pada form pendaftaran. Diagnosa dilakukan dengan cara menjawab pertanyaan yang diajukan oleh sistem, setelah semua jawaban sesuai *rule*, maka sistem akan menampilkan penyebab dan solusi permasalah. Kegiatan yang dilakukan *user* tanpa melalui proses *login* ke sistem.

2. *Activity Diagram*

Activity diagram menggambarkan *workflow* (aliran kerja) atau aktifitas dari sebuah sistem atau proses bisnis. *Activity diagram* menggambarkan aktivitas sistem bukan apa yang dilakukan aktor (A.S dan Shalahuddin: 2011: 134). Berikut ini adalah *activity diagram* yang dirancang dalam penelitian ini:

a. *Activity diagram login adminiatrator*

Activity diagram login administrator merupakan *UML* yang menggambarkan kegiatan pengguan pada halaman *login administrator*.

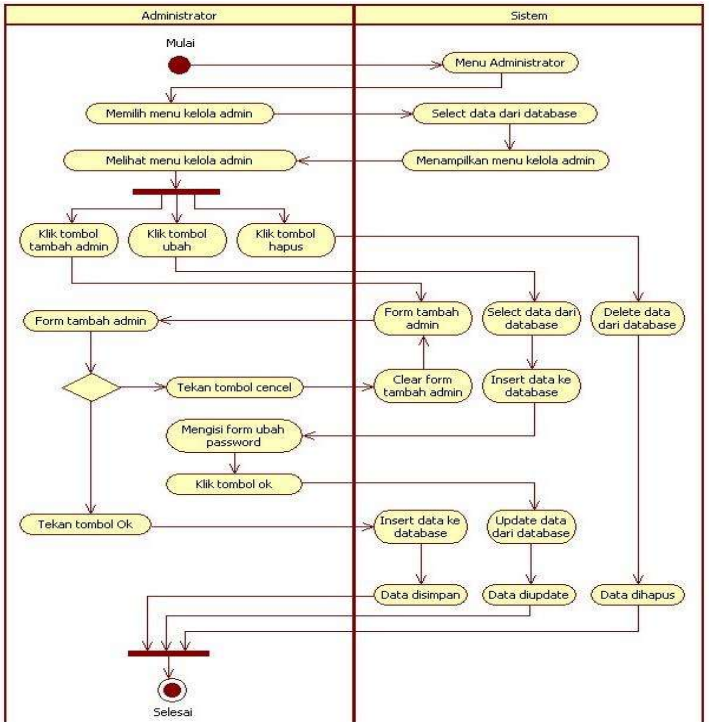


Gambar 3.5 *Activity Diagram* Login Administrator
(Sumber: Data Penelitian, 2017)

Pada Gambar 3.5 diatas, proses *login administrator* adalah *administrator* mengases sistem, kemudian sistem akan menampilkan halaman Menu Utama. Kemudian *administrator* memilih Menu Login Admin, maka sistem akan menampilkan *form* menu *login* admin. Administrator mengetik *username* dan *password* kemudian klik tombol *login*. Maka sistem akan mengecek *username* dan *password* kemudian dicocokkan dengan data yang ada di *database*. Jika *username* dan *password* tidak sesuai dengan yang ada di *database* maka sistem akan menampilkan pesan kesalahan dan sistem kembali menampilkan *form* menu *login* admin, apabila benar maka sistem akan menampilkan menu *administrator*.

b. *Activity diagram* kelola menu admin

Activity diagram kelola admin merupakan kegiatan *administrator* dalam mengelola daftar pengguna (*administrator*). Berikut ini gambar *activity diagram* kelola menu admin (Gambar 3.6):



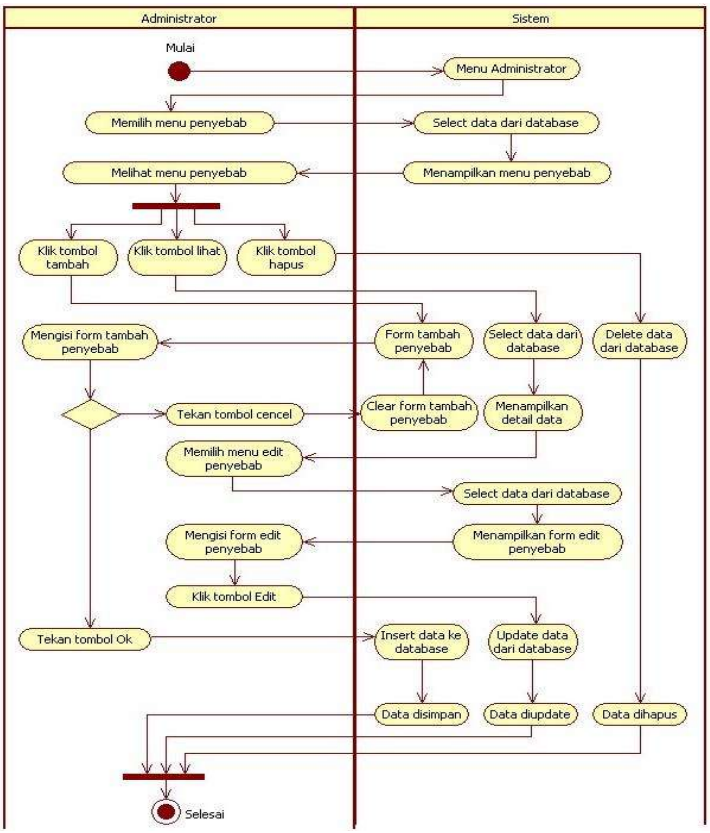
Gambar 3.6 *Activity Diagram* Kelola Menu Admin
(Sumber: Data Penelitian, 2017)

Pada Gambar 3.6 diatas, menu administrator yang terbuka setelah pengguna melakukan *login* admin. Sistem akan menampilkan menu admin, kemudian pengguna (administrator) memilih menu kelola admin. Sistem memanggil data dari database dan menampilkan menu kelola admin. Pengguna melihat 3 pilihan yaitu tambah admin, ubah, dan hapus. Jika pengguna mengklik tombol tambah admin maka sistem akan menampilkan *form* tambah admin, kemudian pengguna mengisi *form* tambah admin. Pengguna klik tombol Ok maka data akan dimasukkan ke *database* kemudian data disimpan di *database*, jika pengguna menekan tombol *cancel* maka sistem akan membersihkan *form* tambah admin. Jika pengguna mengklik tombol ubah, maka sistem akan mengambil data dari *database*. Pengguna mengisi *form* ubah *password* kemudian klik tombol Ok. Sistem akan melakukan

update database. Jika pengguna pengguna menekan tombol *delete* maka data yang ada di *database* akan terhapus.

c. *Activity diagram* mengelola penyebab

Activity diagram mengelola penyebab merupakan diagram *UML* yang menggambarkan kegiatan administrator dalam mengelola data penyebab. Berikut ini adalah gambar *activity diagram* kelola menu admin (Gambar 3.6):



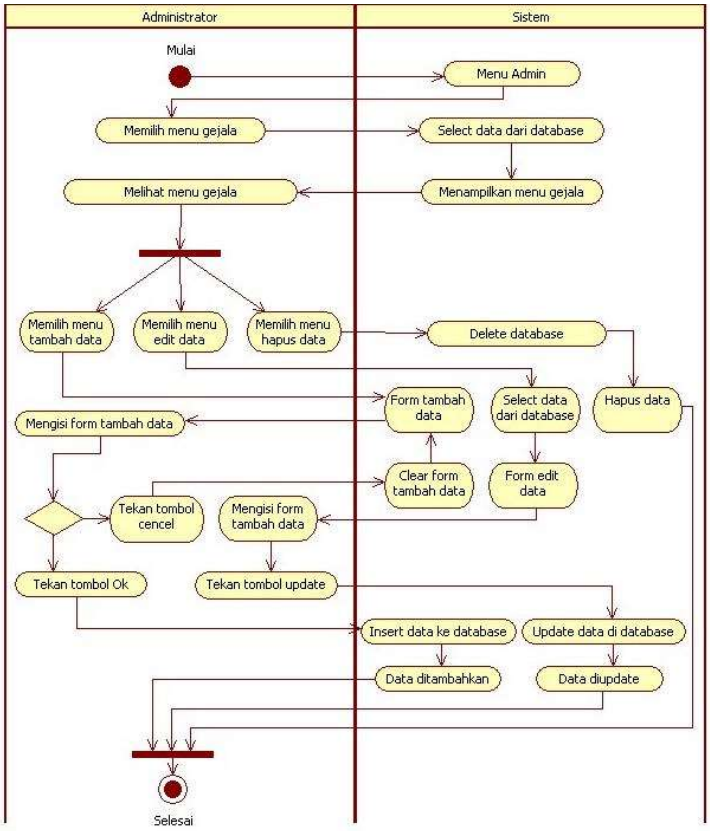
Gambar 3.7 *Activity Diagram* Mengelola Penyebab
(Sumber: Data Penelitian, 2017)

Pada Gambar 3.7 diatas, *administrator* mulai mengakses Menu Administrator kemudian sistem menampilkan Menu Administrator. *Administrator* memilih menu penyebab sehingga sistem mengambil data dari *database* dan menampilkan Menu

Penyabab. *Administrator* melihat ada 3 pilihan yaitu tambah, lihat, dan hapus penyebab. Pertama, jika *administrator* menekan tombol tambah maka sistem akan menampilkan *form* tambah penyebab. *Administrator* akan mengisi *form* tambah penyebab, kemudian tekan tombol Ok maka sistem akan memasukkan data baru tersebut ke *database* kemudian menampilkan pesan data disimpan. Kedua, jika *administrator* menekan tombol lihat maka sistem akan mengambil data dari *database* kemudian menampilkan detail data. Administrator menekan tombol *edit* penyebab maka sistem akan mengambil data dari *database* dan menampilkan *form edit* penyebab. Kemudian administrator mengisi *form edit* penyebab setelah itu menekan tombol *edit* maka sistem akan melakukan *update* data ke *database* dan menampilkan pesan data di-*update*. Ketiga, jika *administrator* menekan tombol hapus maka sistem akan mengapus data dari *database* dan menampilkan pesan data dihapus. Proses selesai.

d. *Activity diagram* mengelola gejala

Activity diagram mengelola gejala merupakan diagram *UML* yang menggambarkan kegiatan *administrator* dalam mengelola data gejala. Berikut ini gambar *activity diagram* mengelola gejala (Gambar 3.8):



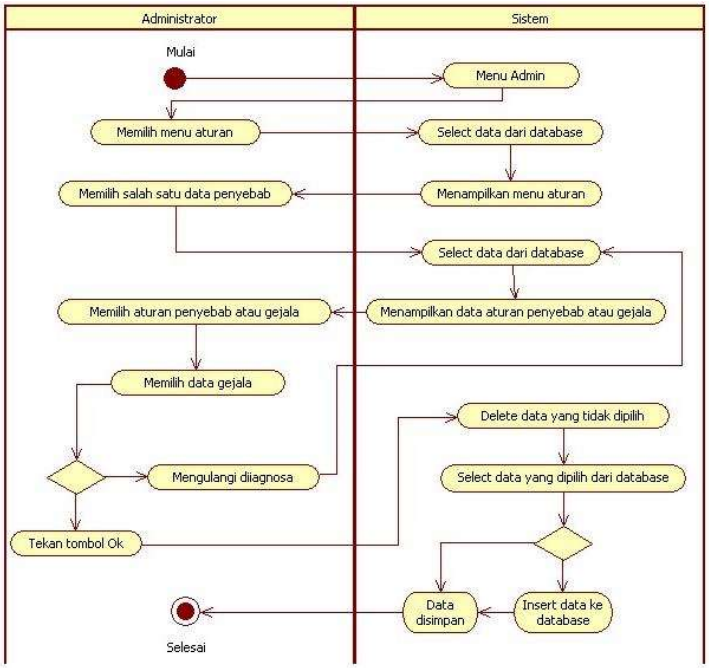
Gambar 3.8 *Activity Diagram* Mengelola Gejala
(Sumber: Data Penelitian, 2017)

Pada Gambar 3.8 diatas, *administrator* mulai mengakses Menu Admin dan sistem menampilkan Menu Admin. *Administrator* memilih menu gejala kemudian sistem mengambil data dari *database* dan menampilkan menu gejala. *Administrator* melihat ada 3 pilihan yaitu tambah, *edit*, dan hapus gejala. Pertama, jika *administrator* menekan tombol tambah maka sistem akan menampilkan *form* tambah gejala. *Administrator* akan mengisi *form* tambah gejala, kemudian tekan tombol tambah maka sistem akan memasukkan data baru tersebut ke *database* kemudian menampilkan pesan data disimpan. Kedua, jika *administrator* menekan tombol *edit* maka sistem akan mengambil data dari *database* kemudian

menampilkan *form edit* data gejala. *Administrator* menekan tombol *update* maka sistem akan melakukan *update* data ke *database* dan menampilkan pesan data di-*update*. Ketiga, jika *administrator* menekan tombol hapus maka sistem akan mengapus data gejala dari *database*. Proses selesai.

e. *Activity diagram* aturan

Activity diagram aturan merupakan diagram *UML* yang menggambarkan kegiatan *administrator* dalam mengelola data aturan. Berikut ini gambar *activity diagram* aturan (Gambar 3.9):



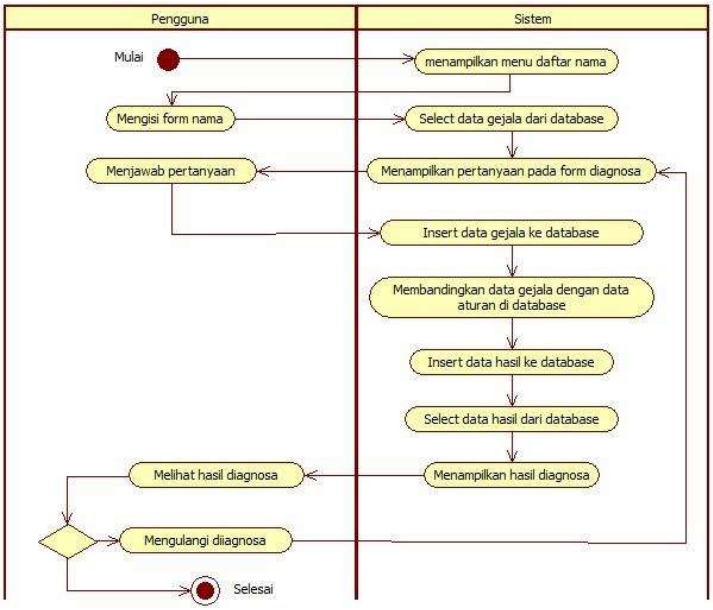
Gambar 3.9 *Activity Diagram* Aturan
(Sumber: Data Penelitian, 2017)

Pada Gambar 3.9 diatas, *administrator* mulai dengan mengakses Menu Admin dan sistem menampilkan Menu Admin. *Administrator* memilih Menu Aturan kemudian sistem akan mengambil data dari *database* dan menampilkan data aturan penyebab atau gejala. Administrator memilih aturan penyebab atau gejala

dan mamilih salah satua atau beberapa data gejala kemudian menekan tombol ok maka sistem akan menghapus data yang tidak dipilih. Data yang sudah dipilih dari *database* akan ditambahkan jika data sebelumnya belum ada dan jika sudah ada maka data langsung disimpan.

f. *Activity diagram* diagnosa

Activity diagram diagnosa merupakan diagram *UML* yang menggambarkan kegiatan pengguna (*user*) dalam menggunakan menu diagnosa. Berikut ini gambar *activity diagram* diagnosa (Gambar 3.10):



Gambar 3.10 *Activity Diagram* Diagnosa
(Sumber: Data Penelitian, 2017)

Pada Gambar 3.10 diatas, pengguna memulai diagnosa dengan menampilkan *form* daftar nama, maka sistem mengambil data dari *database* dan menampilkan pertanyaan pada *form* diagnosa. Pengguna menjawab pertanyaan yang berupa *multiple choice* ya atau tidak. Kemudian sistem akan mengambil data dari *database* dan membandingkan jawaban dengan data aturan di *database*. Kemudian sistem

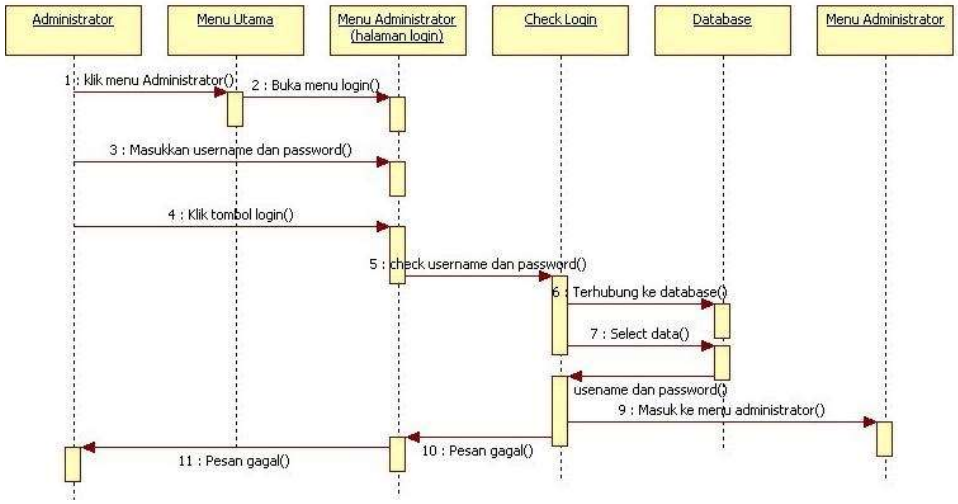
akan memasukkan data hasil ke *database* sementara dan mengambil data dari *database* untuk ditampilkan. Pengguna akan melihat hasil diagnosa. Jika pengguna menekan tombol diagnosa ulang maka sistem akan mengulangi diagnosa dan menampilkan kembali pertanyaan pada *form* diagnosa. Jika pengguna tidak mengulangi diagnosa maka proses selesai.

3. *Sequence Diagram*

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek (A.S dan Shalahuddin: 2011: 137). Berikut ini adalah *sequence diagram* yang dirancang dalam penelitian ini:

a. *Sequence diagram log in administrator*

Sequence diagram log in administrator merupakan urutan waktu kegiatan *administrator* saat melakukan *log in*. Berikut ini gambar *sequence* diagram *login administrator* (Gambar 3.11):

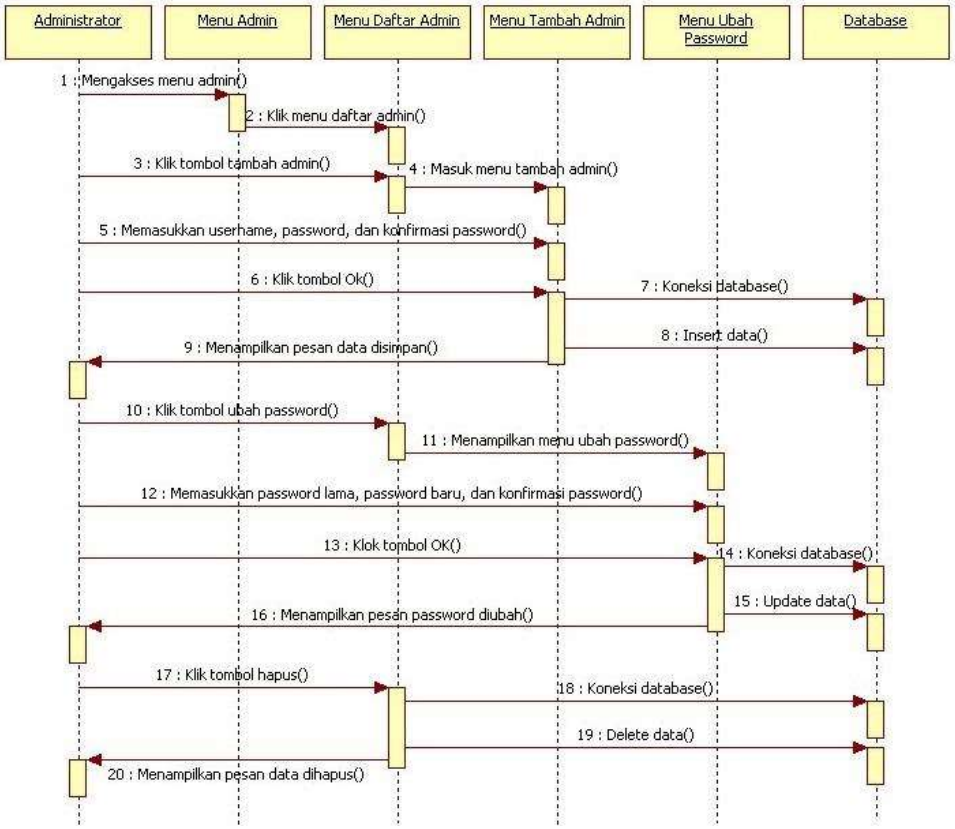


Gambar 3.11 *Sequence Diagram Login Administrator*
(Sumber: Data Penelitian, 2017)

Administrator melakukan klik Menu Administrator, kemudian Menu Utama Administrator menampilkan/membuka *form login*. *Administrator* memasukkan *username* dan *password* ke menu Halaman Login kemudian *administrator* melakukan klik tombol Login. Dari Halaman Login Administrator, sistem akan mengecek *username* dan *password* yang sudah dimasukkan kemudian sistem akan terhubung dengan *database*. Sistem Check Login akan mengambil data dari *database*, setelah itu *username* dan *password* akan dicocokan oleh sistem. Jika *username* dan *password* benar maka Menu Halaman Administrator akan ditampilkan. Jika *username* dan *password* salah (tidak sesuai dengan *database*) maka sistem akan menampilkan pesan gagal di Halaman Login, dan *adminitrstor* akan mengetahui pesan gagal tersebut.

b. *Sequence diagram* kelola *administrator*

Sequence diagram kelola *administartor* merupakan urutan waktu kegiatan *administrator* saat mengelola daftar pengguna (*administartor*). Berikut ini *sequence diagram* kelola *administartor* digambarkan pada Gambar 3.12.

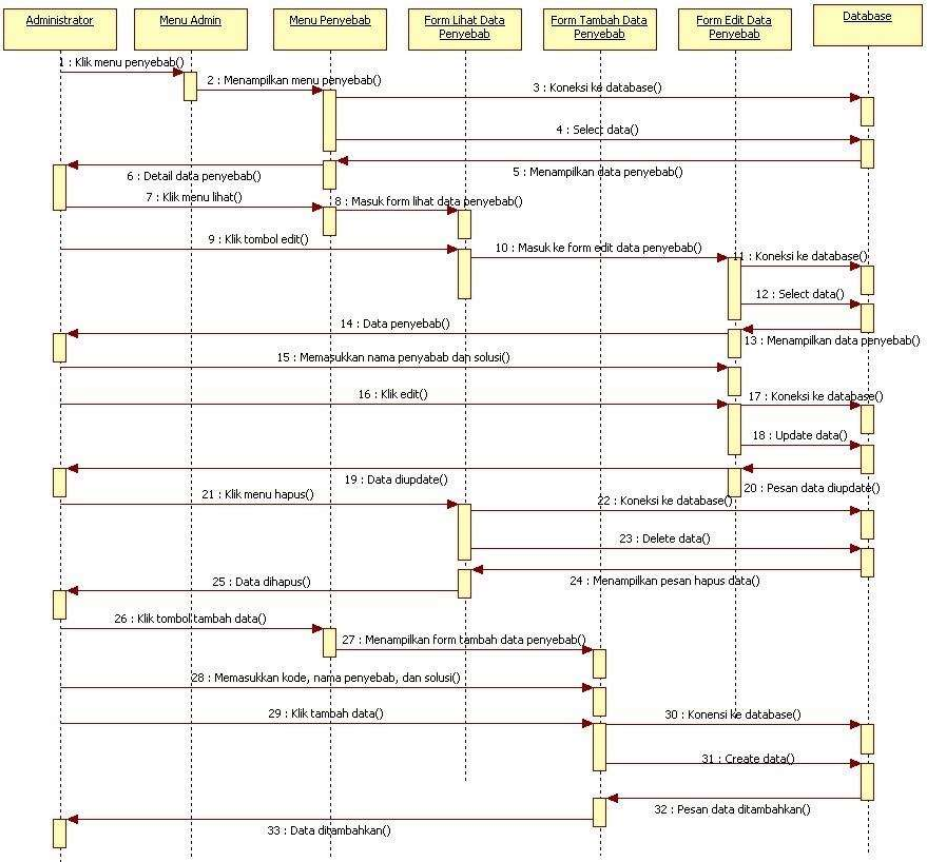


Gambar 3.12 *Sequence Diagram* Kelola Menu Admin
(Sumber: Data Penelitian, 2017)

Administrator mengakses Menu Admin, kemudian menekan Menu Daftar Admin. Setelah Menu Daftar Admin terbuka, *administrator* menekan tombol Tambah Admin. Dari menu Daftar Admin, sistem akan mengarahkan ke Menu Tambah Admin. Kemudian administrator akan memasukkan *username*, *password* dan konfirmasi *password* pada Menu Tambah Admin dan administrator menekan tombol Ok. Menu Tambah Admin melakukan koneksi ke *database* dan memasukkan data ke dalam *database*. Menu Tambah Admin akan menampilkan pesan data disimpan pada *administrator*.

c. *Sequence diagram* mengelola penyebab

Sequence diagram mengelola penyebab merupakan urutan waktu kegiatan *administrator* saat melakukan pengelolaan penyebab kerusakan. Berikut ini gambar s*equence diagram* mengelola penyebab (Gambar 3.13):



Gambar 3.13 *Sequence Diagram* Mengelola Penyebab
(Sumber: Data Penelitian, 2017)

Administrator mengakses Menu Admin, kemudian pilih Menu Penyebab maka terhubung ke *database* dan mengambil data. Pada Menu Admin akan menampilkan seluruh data penyebab. *Administrator* menekan tombol lihat pada Menu Penyebab, kemudian sistem akan menampilkan *Form* Lihat Data Penyebab. *Administrator* menekan tombol e*dit* pada *Form* Lihat Data Penyebab maka sistem akan menampilkan *Form Edit* Data Penyebab. Dari *Form Edit* Data Penyebab akan

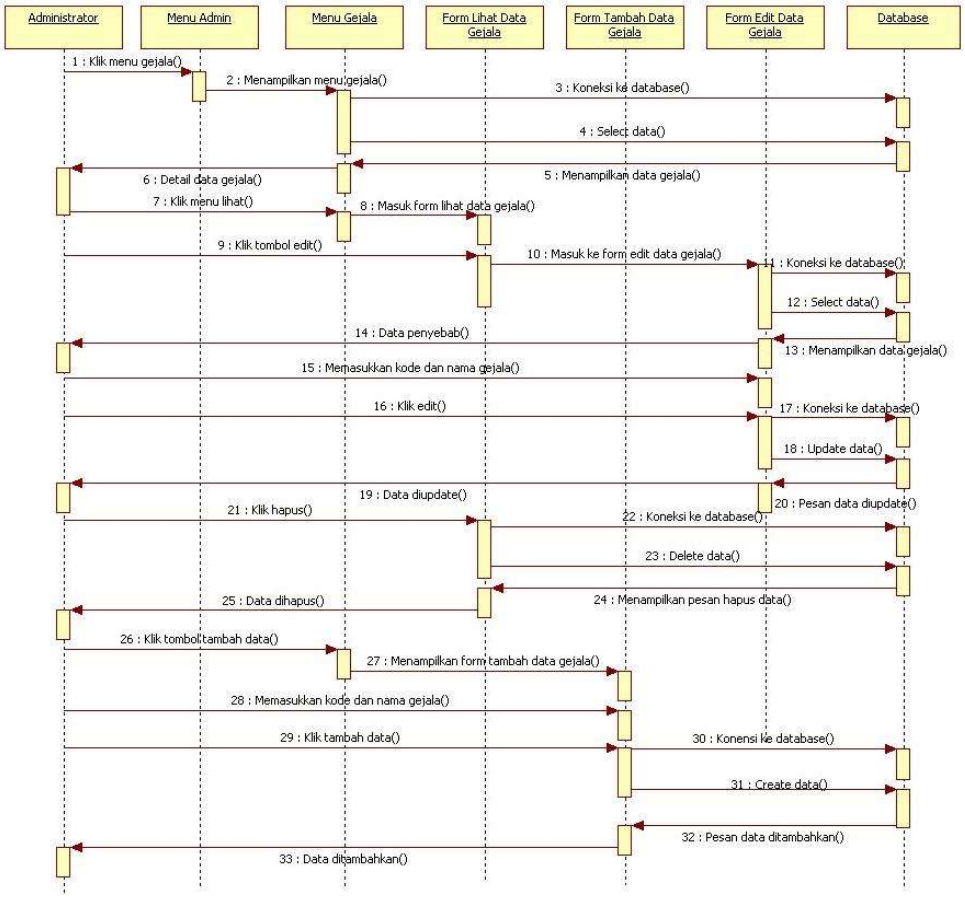
terkoneksi ke *database* dan mengambil data. *Form Edit* Data Penyebab menampilkan data penyebab yang dipilih *administrator* untuk diubah. *Administrator* memasukkan data penyebab dan solusi yang baru kemudian menekan tombol e*dit*, maka sistem akan melakukan *update* data ke *database*. Setelah *database* diperbarui maka akan keluar pesan bahwa data diperbarui.

Administrator menekan tombol Hapus pada Form Lihat Data Penyebab, maka akan terhubung ke *database* dan menghapus data. Jika data sudah terhapus dari *database* maka sistem akan menampilkan pesan data sudah dihapus.

Administrator menekan tombol Tambah Data pada Form Lihat Data Penyebab. Menu Penyebab menampilkan Form Tambah Data Penyebab. Kemudian *administrator* memasukkan nama penyebab dan solusi kedalam Form Tambah Data Penyebab, tetapi untuk kode sudah secara otomatis dari sistem. Setelah semua *form* terisi maka *administrator* menekan tombol Tambah Data, maka sistem akan terkoneksi ke *database* dan membuat data baru. Kemudian sistem akan menampilkan pesan data ditambahkan ke pada *administrator*.

d. *Sequence diagram* mengelola gejala

Sequence diagram mengelola gejala merupakan urutan waktu kegiatan *administrator* saat mengelola gejala kerusakan. Berikut ini *sequence diagram* mengelola gejala digambarkan pada Gambar 3.14.



Gambar 3.14 *Sequence Diagram* Mengelola Gejala
(Sumber: Data Penelitian, 2017)

Administrator menekan tombol Menu Gejala pada Menu Admin, kemudian sistem menampilkan Menu Gejala. Menu Gejala terkoneksi dengan *database* dan mengambil data, kemudian menampilkan data gejala ke Menu Gejala. Menu Gejala akan terkoneksi dengan *database* dan mengambil data kemudian menampilkan detail data gejala ke dalam Menu Gejala. *Administrator* menekan tombol Lihat, maka sistem akan menampilkan Form Lihat Data Gejala. Kemudian *administrator* menekan tombol Edit, maka akan masuk ke Form Edit Data Gejala. Sistem akan terhubungi dengan *database* dan mengambil data, kemudian menampilkan data

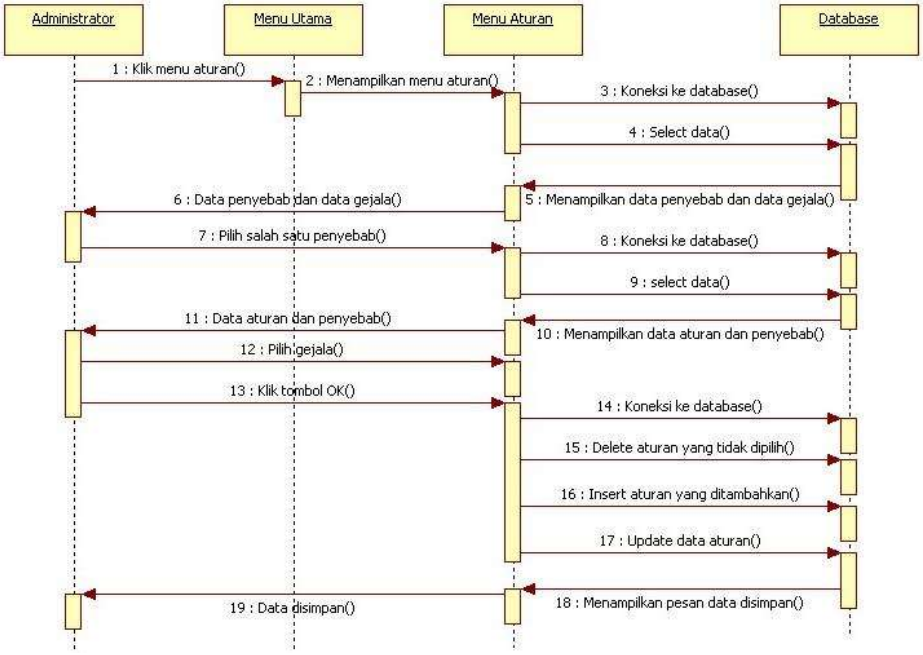
yang akan diubah. Administrator memasukkan data gejala yang baru dan menekan tombol Edit. Sistem akan terhubung dengan *database* dan memperbarui data, kemudian menampilkan pesan bahwa data sudah diperbarui kapada *administrator*.

Administrator menekan tombol Hapus pada Form Lihat Data Gejala, maka sistem akan terhubung dengan *database* dan menghapus data gejala dari *database*. Sistem akan menampilkan pesan bahwa data telah dihapus kepada *administrator*.

Administrator menekan tombol Tambah Data maka sistem menampilkan Form Tambah Data. *Administrator* memasukkan nama gejala pada Form Edit Data Gejala. Sistem akan terhubung dengan *database* dan akan membuat data baru, setelah data ditambahkan maka sistem akan menampilkan pesan data ditambahkan.

e. *Sequence diagram* mengelola aturan

Sequence diagram mengelola aturan merupakan urutan waktu kegiatan administrator saat melakukan pengelolaan aturan. Berikut ini *sequence diagram* mengelola aturan digambarkan pada Gambar 3.15.



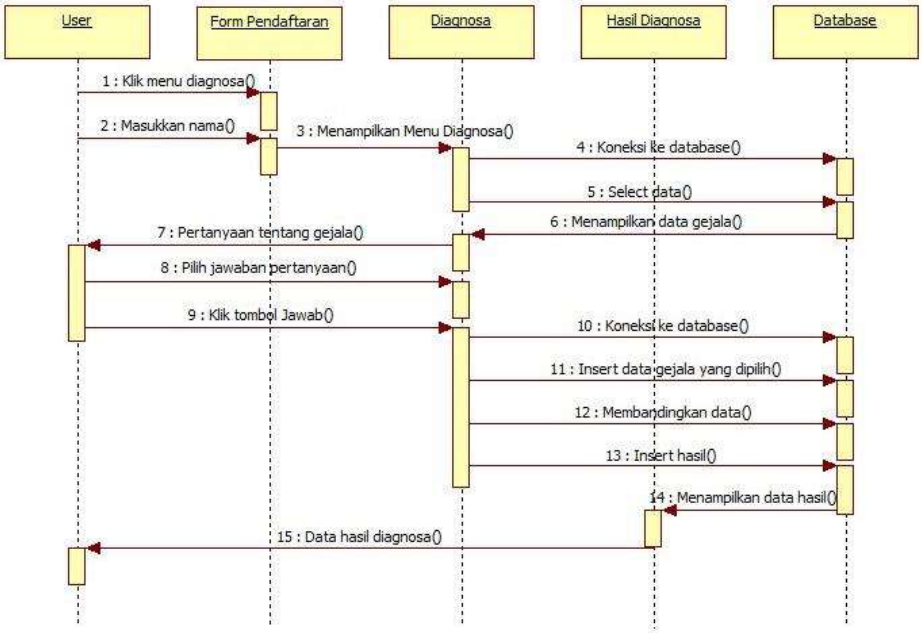
Gambar 3.15 *Sequence Diagram* Aturan
(Sumber: Data Penelitian, 2017)

Administrator menekan tombol Menu Aturan dari Menu Utama Administrator, maka sistem akan menampilkan Menu Aturan. Kemudian terhubung ke *database* dan mengambil data. Sistem menampilkan data penyebab dan data gejala. *Administrator* memilih salah satu kode kerusakan *IC* dan juga penyebab, kemudian mencentang gejala yang sesuai. Sistem akan terhubung dengan *database*. Jika gejala yang dicentang merupakan pembaruan data maka sistem menghapus gejala yang lama dan akan memasukkan data yang baru. Jika gejala yang dicentang merupakan gejala baru maka sistem akan menambahkan data baru ke *database*. Setelah itu administrator menekan tombol Ok untuk menyimpan data. Setelah data disimpan dalam *database* maka sistem akan menampilkan pesan data disimpan.

f. *Sequence diagram* diagnosa

Sequence diagram diagnosa merupakan urutan waktu kegiatan pengguna

(*user*) saat melakukan diagnosa kerusakan *IC*. Berikut ini gambar *sequence diagram* diagnosa (Gambar 3.16):



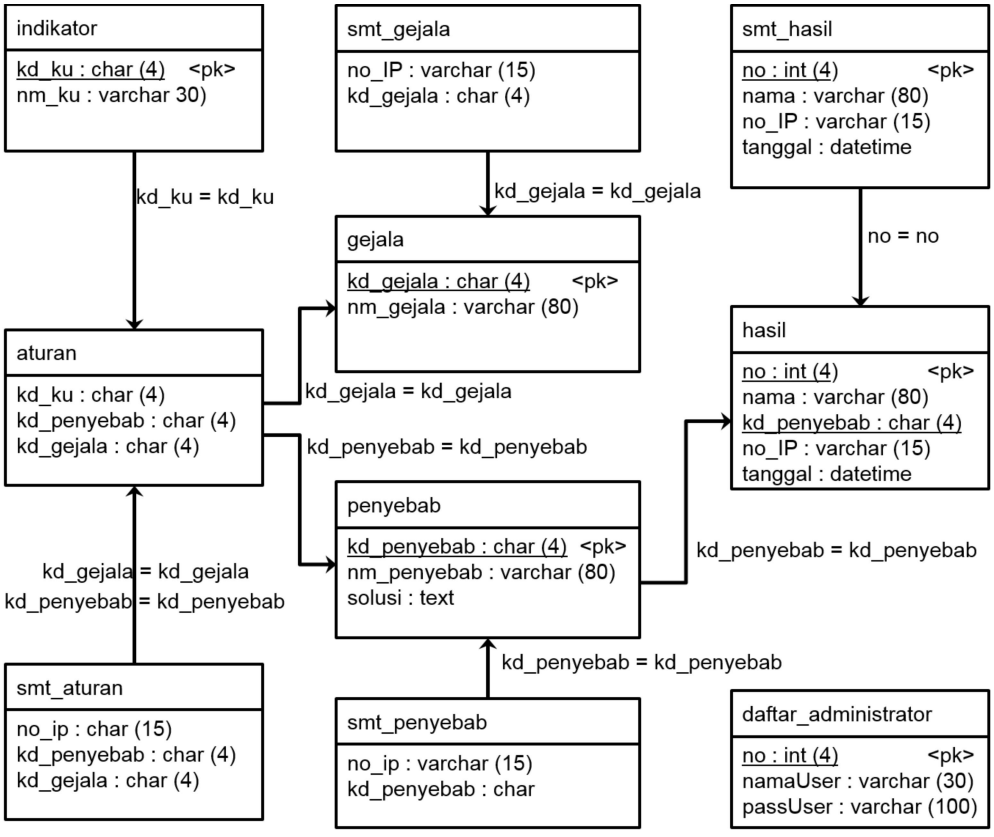
Gambar 3.16 *Sequence Diagram* Diagnosa
(Sumber: Data Penelitian, 2017)

User menekan tombol Menu Diagnosa kemudian tampil Form Pendaftaran. *User* memasukkan nama di Form Pendaftaran kemudian sistem akan menampilkan Menu Diagnosa. Selanjutnya terhubung dengan *database* dan mengambil data dari *database*. Sistem menampilkan data gejala, kemudian mengajukan pertanyaan tentang gejala kerusakan *IC*. *User* memilih salah satu jawaban kemudian tekan tombol Jawab. Maka akan terhubung dengan *database* dan memasukkan data gejala yang diperoleh ke *database* sementara, sistem akan membandingkan data sementara dengan data aturan. Jika sudah menemukan konklusi maka hasil diagnosa disimpan ke *database* dan memampilkan data hasil. Data hasil diagnosa

berupa nama kerusakan *IC*, gejala kerusakan, penyebab kerusakan, dan solusi kerusakan.

3.3.5 Desain *Database*

A.S. dan Shalahuddin (2011: 48) *PDM* (*Physical Data Model*) adalah model yang menggunakan sejumlah table untuk menggambarkan data serta hubungan antar data-data tersebut. PDM merupakan konsep yang menerangkan detail bagaimana data disimpan di dalam *database*. Berikut ini adalah konsep database yang dibuat dalam penelitian:



Gambar 3.17 Desain *Phisical Data Model*
(Sumber: Data Penelitian, 2017)

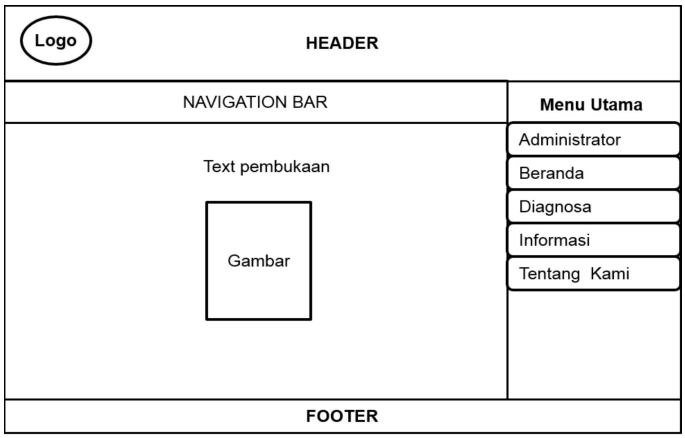
Pada Gambar 3.17 diatas, terdapat 10 tabel *database* yang terdiri dari: tabel indikator, tabel aturan, tabel smt_aturan, tabel gejala, tabel smt_gejala, tabel penyebab, tabel smt_penyebab, tabel hasil, tabel smt_hasil, dan tabel daftar_administrator. Dalam tabel indikator mempunyai kd_ku yang akan digunakan pada tabel aturan yaitu memiliki relasi yang sama pada kd_ku tabel aturan. Tabel tabel smt_aturan merupakan penyimpanan sementara untuk menyimpan data penyebab (kd_penyebab) dan gejala (kd_gejala) yang akan digunakan pada tabel aturan. Tabel smt_gejala berfungsi untuk menyimpan data sementara dari tabel gejala yang berupa gejala (kd_gejala) yang dijawab pengguna saat diagnosa. Begitu juga tabel smt_penyebab yang menyimpan data sementara penyebab yang akan terhubung dengan tabel penyebab dengan kesamaan relasi kode penyebab (kd_penyebab). Dari data gejala yang didapatkan saat diagnosa akan dicocokan denga tabel aturan yang sudah dibuat, maka akan diketahui penyebab kerusakan (kd_penyebab) yang akan dicocokkan dengan data pada tabel aturan. Sedangkan tabel smt_hasil digunakan menyimpan sementara data pengunjung. Tabel hasil akan memanggil isi dari tabel pengunjung (smt_hasil) dan tabel penyebab (kd_penyebab) berupa nama penyebab dan solusi. Tabel daftar_administrator berisi nama dan *password* pengguna.

3.3.6 Desain Antarmuka

Berikut ini adalah desain antarmuka yang akan dibuat pada aplikasi sistem pakar kerusakan *IC* di mesin *Wire Bond* IConn:

1. Rancangan Halaman Beranda

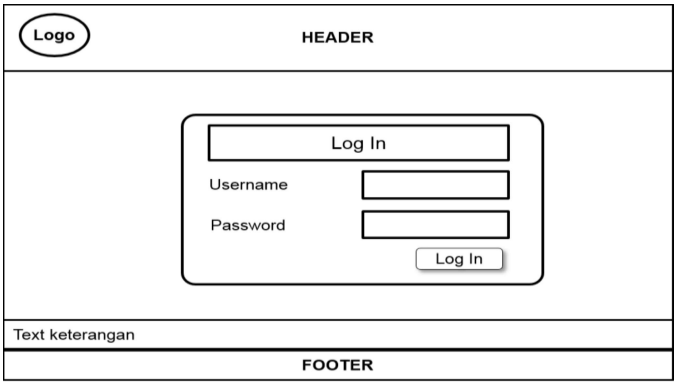
Halaman Beranda menampilkan informasi tentang aplikasi sistem pakar deteksi kerusakan *IC* di mesin *Wire Bond* IConn. Berisi tentang pengertian sistem pakar dan acara menggunakan aplikasi.



Gambar 3.18 Rancangan Halaman Beranda
(Sumber: Data Penelitian, 2017)

2. Rancangan Halaman *Log In Administrator*

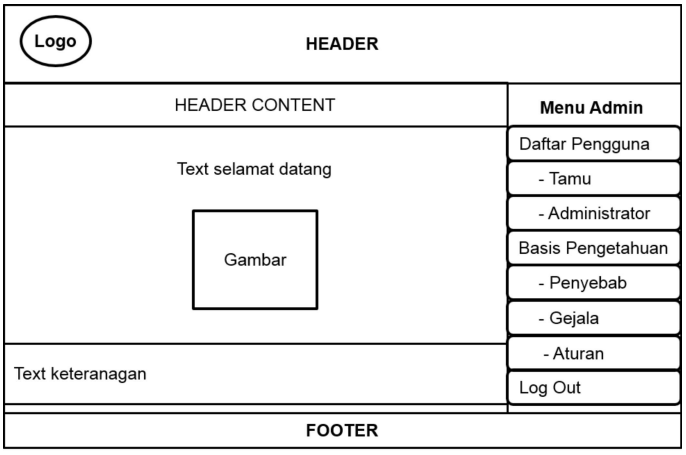
Halaman *Log In Administrator* akan tampil saat *user* menekan Menu Administrator. Pada halaman ini *user* dihadapkan *form* untuk mengisi *username* dan *password* sebagai *administrator*. Jika *username* dan *password* benar maka akan diarahkan ke halaman Administrator, jika salah maka akan keluar pesan kesalahan.



Gambar 3.19 Rancangan Halaman *Login Administrator*
(Sumber: Data Penelitian, 2017)

3. Rancangan Halaman Utama Administrator

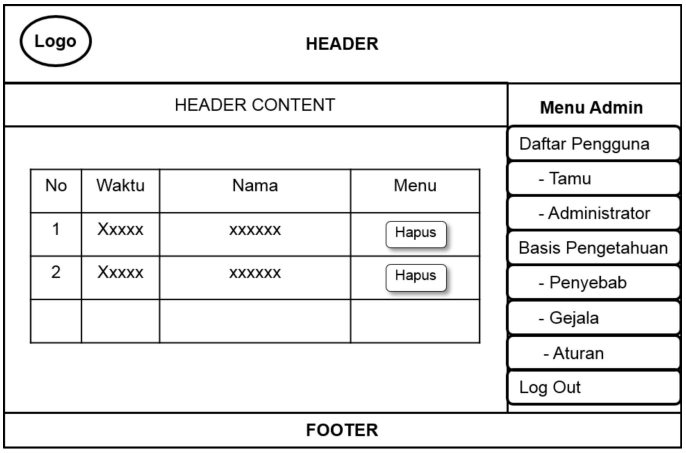
Halaman Utama Administrator berbeda dengan halaman pengguna, yaitu pengguna akan diijinkan masuk jika sudah benar memasukkan *username* dan *password* di menu *Log In Administrator*. Halaman ini terdapat tambahan beberapa menu yaitu menu Penyebab, menu Gejala, menu Aturan, menu Daftar Pengguna tamu dan administrator. Jika ingin keluar dari halaman *Administrator*, ada tombol *Logout* yang akan mengarahkan ke halaman *Log In Administrator*.



Gambar 3.20 Rancangan Halaman Utama Administrasi
(Sumber: Data Penelitian, 2017)

4. Rancangan Halaman Daftar Pengguna (Tamu)

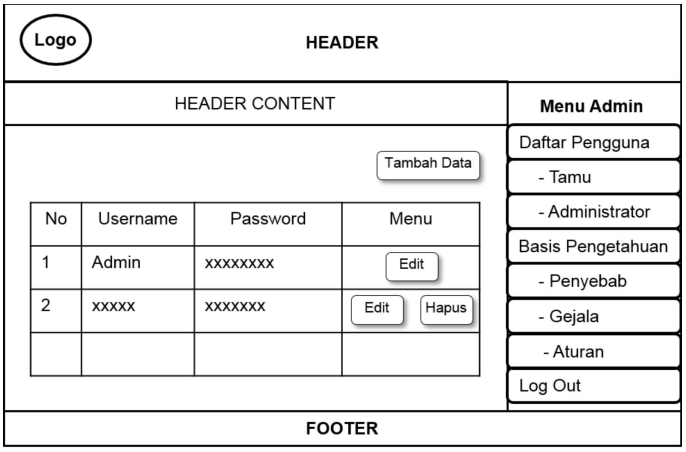
Halaman Daftar Pengguna (Tamu) menampilkan daftar tabel pengunjung yang sudah pernah melakukan diagnosa di sistem pakar ini. Pada halaman ini terdapat tabel dan tombol hapus yang befungsi untuk menghapus pengguna dari *database*.



Gambar 3.21 Rancangan Halaman Administrasi
(Sumber: Data Penelitian, 2017)

5. Rancangan Halaman Daftar Pengguna (*Administrator*)

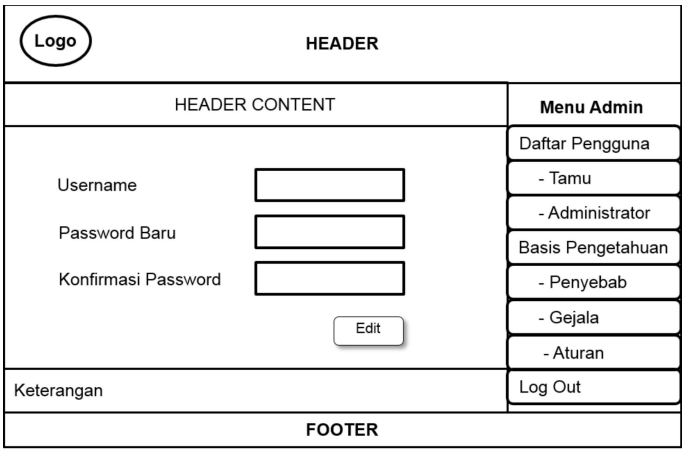
Halaman Daftar Pengguna (*Administrator*) memuat tampilan daftar nama-nama *administrator* yang ada pada program sistem pakar. Pada halaman ini terdapat 3 tombol, yaitu: tombol Tambah Admin berfungsi untuk mengarahkan *user* ke halaman Tambah Administrator, tombol Ubah berfungsi untuk mengubah *password administrator*, dan tombol Hapus berfungsi untuk menghapus daftar *administrator* yang akan dihapus.



Gambar 3.22 Rancangan Halaman Daftar Pengguna (*Administrator*)
(Sumber: Data Penelitian, 2017)

6. Rancangan Halaman *Edit Password Administrator*

Halaman *Edit Password* dapat diakses dengan cara mengklik tombol Ubah Password pada halaman *Administrator* ataupun dengan menekan tombol Ubah pada halaman Daftar Pengguna (*Administrator*). Halaman ini berfungsi untuk mengubah *pasword administrator*. Pengguna (*administrator*) akan dihadapkan dengan *form* yang berisi *username*, *password* lama, *password* baru, dan konformasi *password*. Setelah mengisinya maka tekan tombol Ok, jika *password* benar maka akan ada pesan *password* berhasil di-*update* dan jika salah maka akan tampil pesan kesalahan.

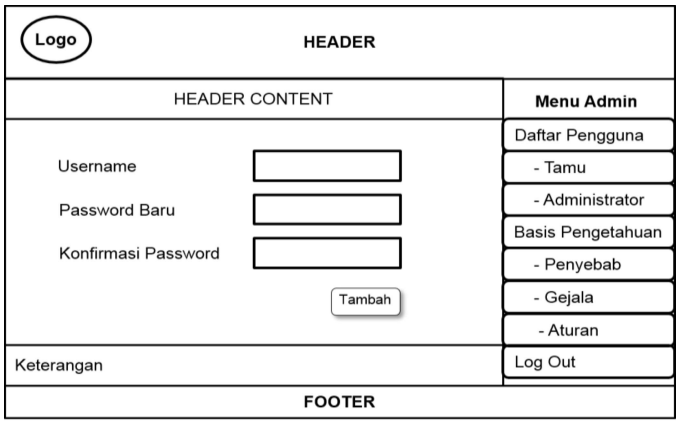


Gambar 3. Rancangan halaman *Edit Password Administrator*
(Sumber: Data Penelitian, 2017)

7. Rancangan Halaman Tambah *Administrator*

Halaman Tambah *Administrator* berfungsi untuk menambah daftar *administrator* pada *database*. Pada halaman ini pengguna (*administrator*) akan dihadapkan dengan *form* yang berisi u*sername*, *password*, dan konformasi

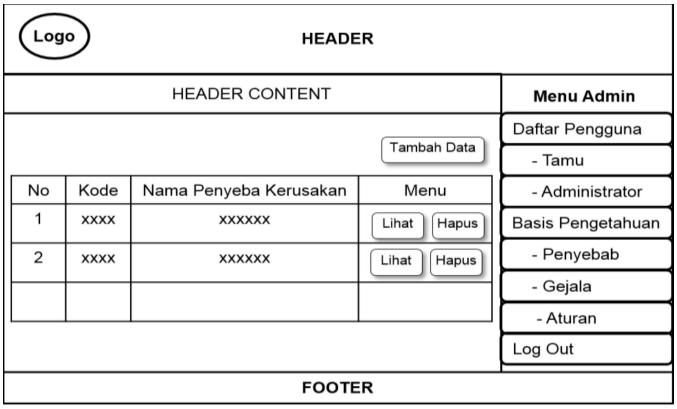
password. Setelah mengisi *form*, maka klik tombol Ok untuk menambahkan data ke *database*.



Gambar 3.23 Rancangan Halaman Tambah *Administrator*
(Sumber: Data Penelitian, 2017)

8. Rancangan Halaman Basis Pengetahuan (Penyebab)

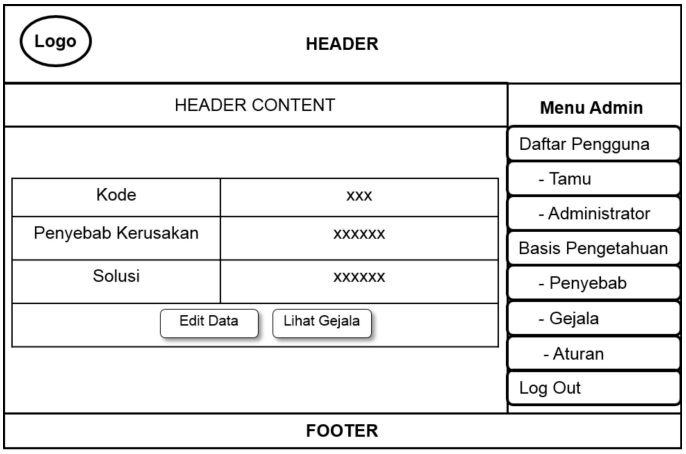
Halaman Basis Pengetahuan (Penyebab) menampilkan kode kerusakan, gejala kerusakan dan disediakan menu untuk melihat dan menghapusnya. Pada halaman ini juga disediakan tombol Tambah Data yang akan menghubungkan ke halaman Tambah Penyebab. Menu Lihat akan menghubungkan ke halaman Lihat Data Penyebab.



Gambar 3.24 Rancangan Halaman Utama Penyebab
(Sumber: Data Penelitian, 2017)

9. Rancangan Halaman Lihat Data Penyebab

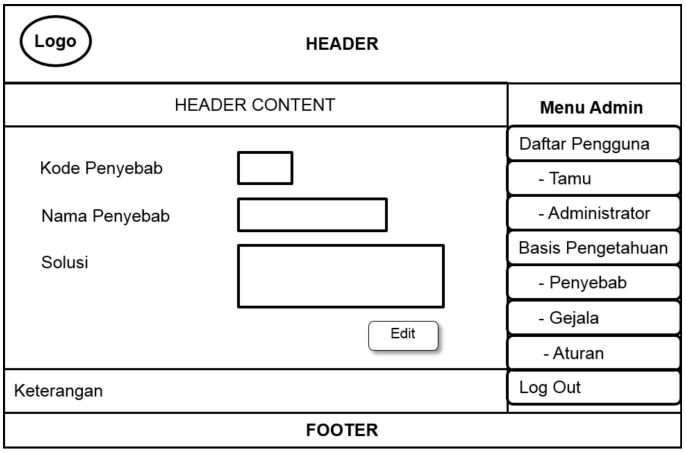
Halaman Lihat Data Penyebab akan menampilkan kode kerusakan, penyebab kerusakan, dan solusi. Jika pengguna (*administrator*) akan mengubah data penyebab yang sudah dipilih maka klik tombol Edit Data. Dengan menekan tombol Edit Data maka *administrator* akan ditampilkan halaman Edit Penyebab.



Gambar 3.25 Rancangan Halaman Lihat Data Penyebab
(Sumber: Data Penelitian, 2017)

10. Rancangan Halaman *Edit* Penyebab

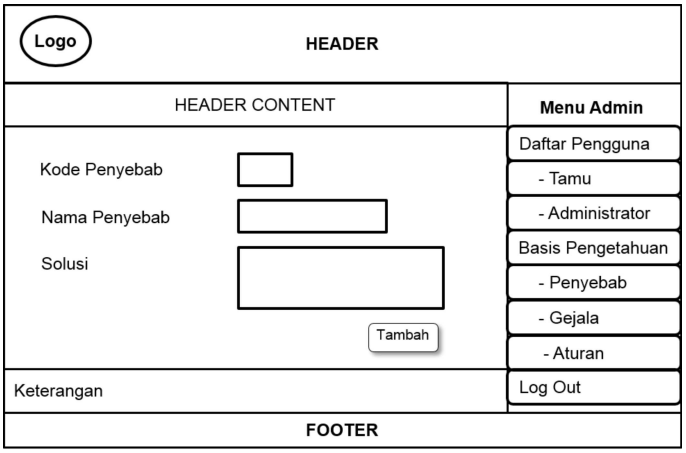
Halaman *Edit* Penyebab menampilkan kode kerusakan, penyebab kerusakan, dan solusi. *Administrator* bisa langsung mengubah data penyebab dengan mengisi *form* edit penyebab. Setelah mengisi/mengubahnya maka klik tombol *Edit* supaya data dapat diperbarui di *database*.



Gambar 3.26 Rancangan Halaman Edit Penyebab
(Sumber: Data Penelitian, 2017)

11. Rancangan Halaman Tambah Penyebab

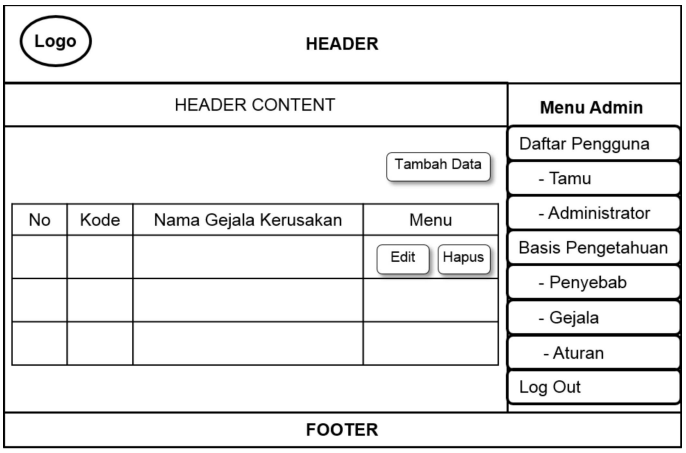
Halaman Tambah Penyebab menampilkan *form* yang berfungsi untuk menambah data penyebab baru di *database*. Halaman ini berisi *form* kode kerusakan, penyebab kerusakan, dan solusi. Tombol Tambah berfungsi untuk memperbarui data ke *database* jika semua *form* sudah diisi.



Gambar 3.27 Rancangan Halaman Tambah Penyebab
(Sumber: Data Penelitian, 2017)

12. Rancangan Halaman Basis Pengetahuan (Gejala)

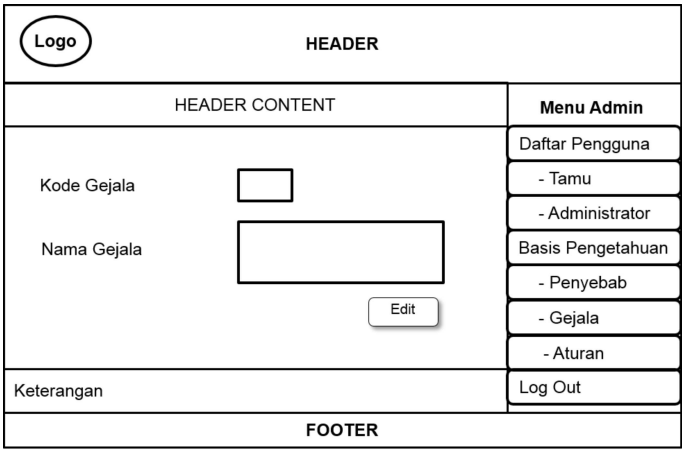
Halaman Basis Pengetahuan (Gejala) menampilkan tabel daftar gejala yang berupa kode gejala, nama gejala kerusakan dan juga menu untuk melihat dan menghapus. Pada halaman ini terdapat tombol Tambah Data yang berfungsi untuk menampilkan halaman Tambah Gejala. Menu Lihat berfungsi untuk menampilkan halaman Lihat Gejala dan Menu Hapus untuk menghapus data gejala yang dipilih untuk dihapus.



Gambar 3.28 Rancangan Halaman Basis Pengetahuan (Gejala)
(Sumber: Data Penelitian, 2017)

13. Rancangan Halaman *Edit* Gejala

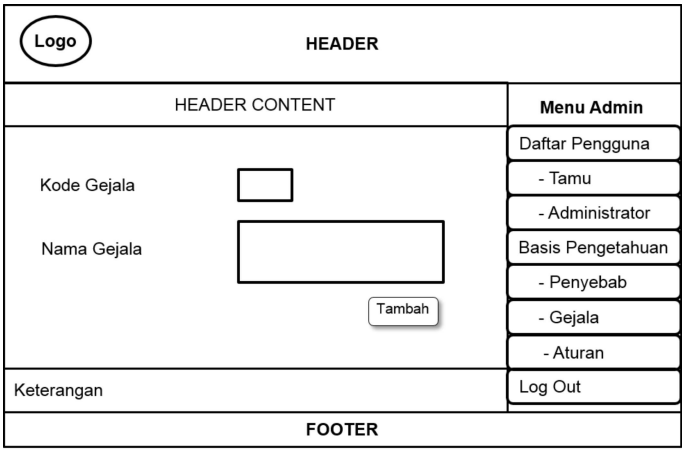
Halaman *Edit* Gejala menampilkan *form* untuk mengubah gejala kerusakan. Berisi tampilan form kode gejala, gejala kerusakan, dan tombol *Edit*. Jika tombol *Edit* ditekan maka data yang ada di *form* akan di-*update* di *database*.



Gambar 3.29 Rancangan Halaman *Edit* Gejala
(Sumber: Data Penelitian, 2017)

14. Rancangan Halaman Tambah Gejala

Halaman Tambah Gejala menampilkan *form* berisi kode gejala dan gejala kerusakan. Tombol Tambah berfungsi untuk menambahkan isi *form* Tambah Gejala kedalam *database*.

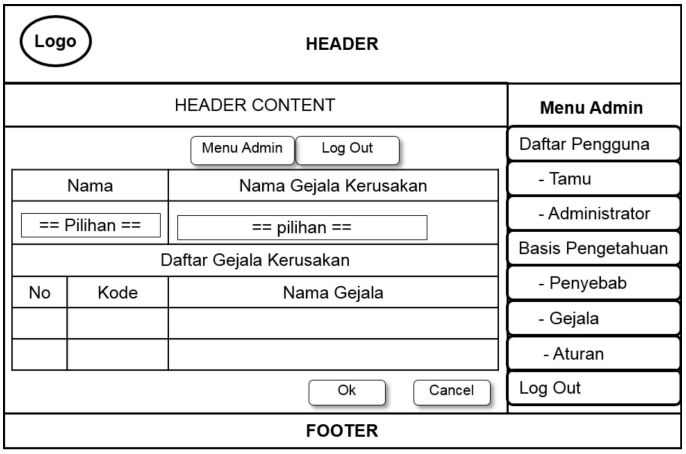


Gambar 3.30 Rancangan Halaman Tambah Gejala
(Sumber: Data Penelitian, 2017)

15. Rancangan Halaman Aturan

Halaman Aturan berisi daftar aturan penyebab kerusakan *IC* yaitu: kode dan nama gejala. Untuk memudahkan *administrator* dalam mencari daftar gejala, maka

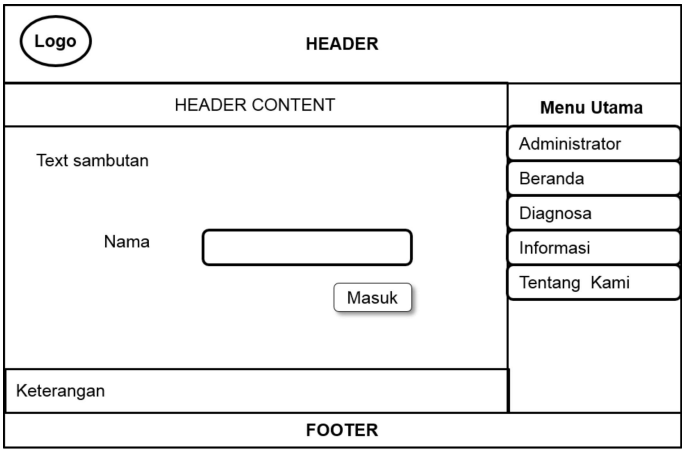
disediakan *filter list* yaitu tombol nama kerusakan *IC* dan penyebab kerusakan. Pada halaman ini juga disediakan tombol untuk mengubah data aturan. Radio button yang ada pada daftar digunakan untuk memilih data aturan yang akan diubah, setelah itu klik tombol Ok mengubah ataupun menambahkan data aturan. Sedangkan tombol *Cancel* untuk membatalkan perintah mengubah data aturan.



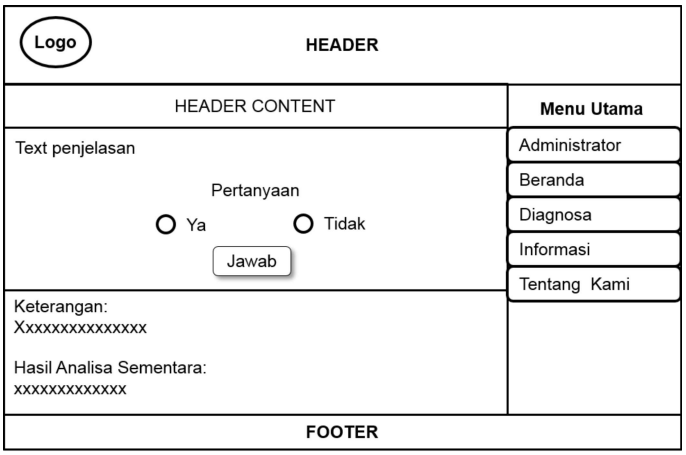
Gambar 3.31 Rancangan Halaman Aturan
(Sumber: Data Penelitian, 2017)

16. Rancangan Halaman Diagnosa

Halaman Diagnosa akan menampilkan beberapa pertanyaan kepada *user*. *User* akan memilih jawaban dengan mengklik *radiobuttom* pilihan "Ya" atau "Tidak" sesuai gejala yang dialami pada mesin, setelah itu baru klik tombol JAWAB. Pertanyaan akan berulang lagi sampai akhir gejala.



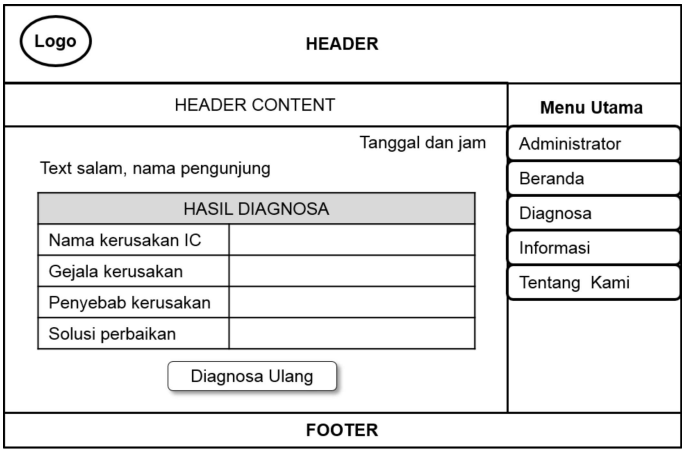
Gambar 3.32 Halaman Daftar Tamu
(Sumber: Data Penelitian, 2017)



Gambar 3.33 Rancangan Halaman Diagnosa
(Sumber: Data Penelitian, 2017)

17. Rancangan Halaman Hasil Diagnosa

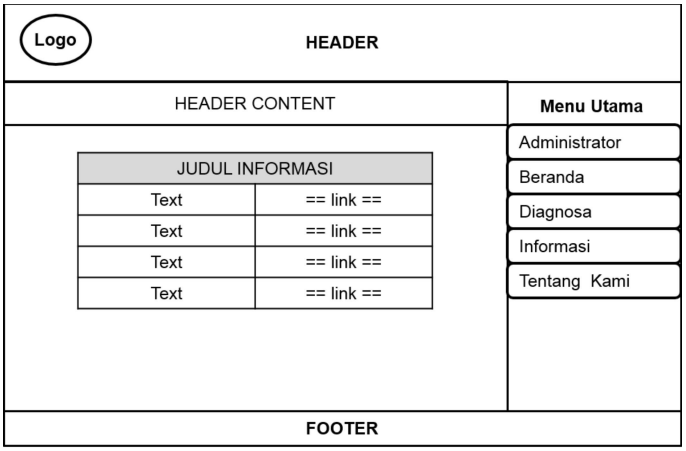
Setelah beberapa pertanyaan dijawab di halaman Diagnosa, maka pada saat jawaban terakhir dijawab akan menampilkan halaman Hasil Diagnosa. Halaman ini akan menampilkan hasil diagnosa berupa: nama *reject IC*, gejala kerusakan, penyebab kerusakan, dan solusi. Jika *user* akan mengulangi diagnosa, maka klik tombol Diagnosa Ulang. Tombol Diagnosa Ulang akan menampilkan halaman Diagnosa kembali.



Gambar 3.34 Rancangan Halaman Hasil Diagnosa
(Sumber: Data Penelitian, 2017)

18. Rancangan Halaman Informasi

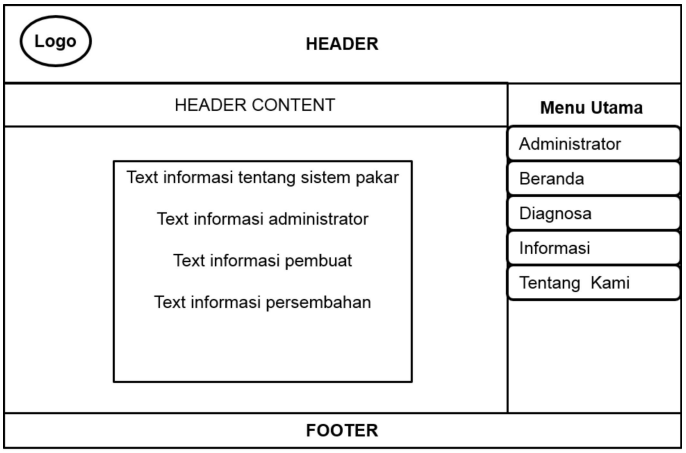
Halaman Informasi akan menampilkan informasi yang berhubungan tentang kerja teknisi *Wire Bond* PT UNISEM Batam yaitu iAS1006 (A*ssembly* S*pesification* tentang proses *Wire Bond*), iAS1007 (A*ssembly* S*pesification* tentang kriteria *reject Wire Bond* dan *Die Attach*), Handbook Machine IConn (penduan tentang mesin KNS IConn). Masing-masing item tersebut disediakan *link* yang tinggal klik untuk membukanya.



Gambar 3.35 Rancangan Halaman Informasi
(Sumber: Data Penelitian, 2017)

19. Rancangan Halaman Tentang Kami

Halaman Tentang Kami memuat informasi mengenai sistem pakar Wire Bond, informasi alamat email administrator, informasi tentang pembuat program, dan informasi tentang persembahan program ini.



Gambar 3.36 Rancangan Halaman Tentang Kami
(Sumber: Data Penelitian, 2017)

3.4 Lokasi dan Jadwal Penelitian

3.4.1 Lokasi Penelitian

Penelitian ini dilakukan di PT UNISEM Batam Jl. S. Parman Kav 201, Batamindo Industrial Park, Mukakuning, Batam (29433). Alasan peneliti memilih perusahaan ini sebagai lokasi penelitian adalah peneliti bekerja di PT UNISEM Batam sebagai *Process Engineer* area *Wire Bond* sehingga peneliti mudah mendapatkan data dan efisiensi waktu dalam melakukan penelitian. Selain itu juga,

peneliti sudah bekerja selama 10 tahun lebih perusahaan tersebut dengan bidang yang sama, jadi peneliti juga bisa sebagai narasumber penelitian.

3.4.2 Jadwal Penelitian

Jadwal penelitian perlu dibuat untuk menggambarkan kapan dan berapa lama waktu yang diperlukan untuk melakukan setiap langkah dalam penelitian. Selain itu, jadwal penelitian juga merupakan tengat (*deadline*) bagi peneliti yang bersangkutan untuk dapat melaksanakan dan menyelesaikan penelitian. Berikut ini adalah tabel jadwal kegiatan yang dilakukan selama penelitian berlangsung:

Tabel 3.9 Tabel Jadwal Penelitian

No	Kegiatan	Jadwal																					
		September 2016				Oktober 2016				November 2016				Desember 2016				Januari 2017				Fabruari 2017	
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2
1	Pengajuan Judul																						
2	Penyusunan Bab I																						
3	Penyusunan Bab II																						
4	Penyusunan Bab III																						
5	Penyusunan Bab IV																						
6	Penyusunan Bab V, Daftar Pustaka, Lampiran																						

(Sumber: Data Penelitian 2017)