BAB II

TINJAUAN PUSTAKA

II.1. Penelitian Berdasarkan Konsep Sistem Pakar

Selama melakukan penelitian pada PT. Super Andalas Steel mengenai untuk menentukan jenis kerusakan mesin katel uap, yaitu bagaimana peneliti memecahkan masalah yang ada pada perusahaan, sehingga peneliti merancang suatu sistem pakar untuk menentukan jenis kerusakan mesin katel uap yang menggunakan metode *Dempster Shafer,* dan peneliti membanding penelitiannya dengan beberapa jurnal yang ada yang menggunakan metode *Dempster Shafer.*

Muhammad Syahril (2016) melakukan penelitian dalam mendiagnosa penyakit *bell's palsy* yang menggunakan metode *Dempster Shafer*. Hasil dari penelitian ini adalah penyakit *Bell,s Palsy* adalah disfungsi *nervus facialis*, saat saraf berjalan dalam *canalis facialis,* kelainan ini biasanya *unilateral*. Sistem pakar bukanlah untuk menggantikan fungsi dokter, akan tetapi hanya digunakan sebagai pelengkap dan alat bantu. Metode yang digunakan pada sistem pakar ini menggunakan teori *dempster sahafer*.

Yasidah Nur Istiqomah (2013) melakukan penelitian sistem pakar untuk mendiagnosa penyakit saluran pencernaan menggunakan metode *Dempster Shafer.* Hasil dari penelitian ini adalah sistem pakar untuk mendiagnosis penyakit saluran pencernaan sebanyak 19 jenis penyakit dan menggunakan motode dempster shafer. Subjek dalam penelitian ini adalah sistem pakar untuk

mendiagnosa penyakit saluran pencernaan. Pada penelitian ini menggunakan metode ketidakpastiannya menggunakan metode *dempster shafer*.

Mikha Dayan Sinaga (2016) melakukan penelitian penerapan metode *Dempster Shafer* untuk mendiagnosa penyakit dari akibat bakteri *salmonella.* Hasil dari penelitian ini adalah untuk membuat aplikasi sistem pakar yang dapat mendiagnosa bakteri dari akibat bakteri salmonella dengan menggunakan metode *Dempster Shafer*. Untuk dapat mengetahui tingkat kepastian infeksi bakteri ni peneliti menggunakan metode *Dempster Shafer*. Metode ini dipilih karena motode ini dianggap mampu untuk memberikan tingkat kepastian yang tinggi.

Deby Putri Indraswari (2015) melakukan penelitian sistem pendukung keputusan deteksi dini penyakit stroke menggunakan metode *Dempster Shafer.* Hasil dari penelitian ini adalah membantu pengambilan keputusan dengan baik. Penelitian ini menggunakan metode *Dempster Shafer* dalam mendeteksi dini penyakit stroke. Metode *Dempster Shafer* digunakan karena mampu mengatasi permasalahan yang memiliki banyak unsur ketidakpastian kerap kali ditemukan dalam melakukan pendeteksian penyakit.

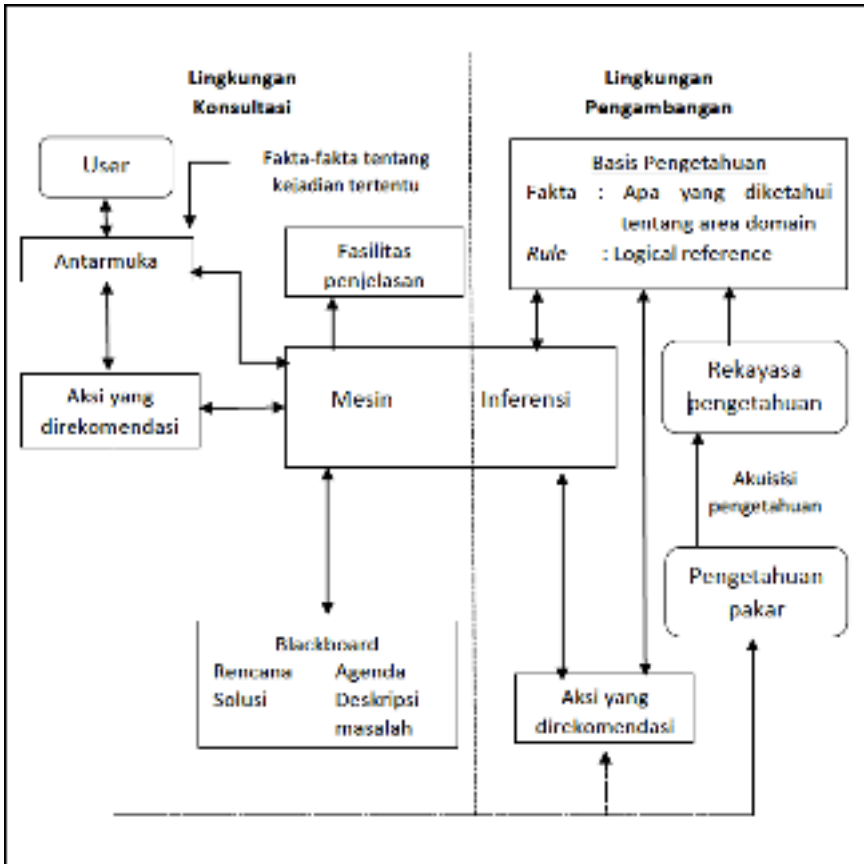
Eli Rosmita Ritonga (2017) melakukan penelitian sistem pakar diagnosa penyakit paru-paru pada anak dengan metode *Dempster Shafer.* Hasil dari penelitian ini adalah bertujuan untuk mengetahui bagaimana cara mendiagnosa penyakit paru pada anak sehingga dapat dilakukan penanganan yang tepat sesuai dengan ciri-ciri penyakit yang terdapat pada anak tersebut. Didalam penerapan sistem pakar ini dibantu dengan metode *Dempster Shafer*. *Dempster Shafer* adalah

suatu teori matematika untuk pembuktian berdasarkan *belief functions and plausible reasoning* (fungsi kepercayaan dan pemikiran yang masuk akal).

II.2. Sistem Pakar

Istilah sistem pakar dari istilah *knowledge- based expert system*. Sistem pakar menggunakan pengetahuan seorang pakar yang dimasukkan ke dalam komputer. Seorang yang bukan pakar/ahli menggunakan sistem pakar untuk meningkatkan kemampuan pemecahan masalah, sedangkan seorang pakar menggunakan sistem pakar untuk *knowledge assistant*.

Ada dua bagian penting dalam sistem pakar, yaitu lingkungan pengembangan dan lingkungan konsultasi. Lingkungan pengembangan digunakan oleh pembuatan sistem pakar untuk membangun komponenkomponennya dan memperkenalkan pengetahuan ke dalam *knowledge base* (basis pengetahuan). Lingkungan konsultasi digunakan oleh pengguna untuk berkonsultasi sehingga pengguna mendapat pengetahuan dari sistem pakar seperti berkonsultasi dengan seorang pakar, (Aryu Hanifa Aji; 2018: 2). Adapun gambar komponen arsitektur pada sistem pakar dapat dilihat pada gambar II.1 :



Gamabar II.1. Arsitektur Sistem Pakar
(*Sumber : Aryu Hanifa Aji; 2018 :2*)

Beberapa komponen arsitektur sistem pakar dapat dijelaskan sebagai berikut, (Aryu Hanifa Aji; 2018 :2) :

1. Antar Muka Pengguna (*User Interface*)

   User interface merupakan mekanisme yang digunakan oleh pengguna dan sistem pakar untuk berkomunikasi, informasi yang diterima dari pemakai diubah kedalam bentuk yang dapat diterima oleh sistem menghasilkan pengetahuan yang dimengerti oleh pemakai.

2. Basis Pengetahuan

Basis pengetahuan mengandung pengetahuan untuk pemahaman, formulasi, dan penyelesaian masalah, komponen ini disusun atas elemen dasar, yaitu fakta.

3. Mesin inferensi

   Mesin inferensi adalah bagian yang mengandung mekanisme fungsi berfikir dan pola-pola penalaran sistem yang digunakan oleh seorang pakar. Mekanisme ini akan menganalisa suatu masalah tertentu dan selanjutnya akan mencari jawaban atau kesimpulan yang terbaik. Mesin inferensi disebut juga otak dari sistem pakar.

4. Memori Kerja

   Merupakan bagian dari sistem pakar yang menyimpan fakta-fakta yang diperoleh saat dilakukan proses konsultasi. Fakta-fakta inilah yang nantinya akan diolah oleh mesin inferensi berdasarkan pengetahuan yang disimpan dalam basis pengetahuan untuk menentukan suatu keputusan pemecahan.

5. Fasilitas Penjelasan

   Karena pemakai kadangkala bukanlah ahli dalam bidanh tertentu, maka dibuatlah fasilitas penjelasan. Fasilitas penjelasan ini penting untuk memberikan gambaran mengenai bagaimana jalannya penalaran sehingga dihasilkan suatu keputusan.

6. Fasilitas Akuisisi Pengetahuan

   Fasilitas akuisisi pengetahuan ini penting agar pengetahuan yang ada dapat ditambahkan kapan saja ketika pengetahuan baru diperoleh ataui saat pengetahuan yang sudah ada tidak berlaku lagi.

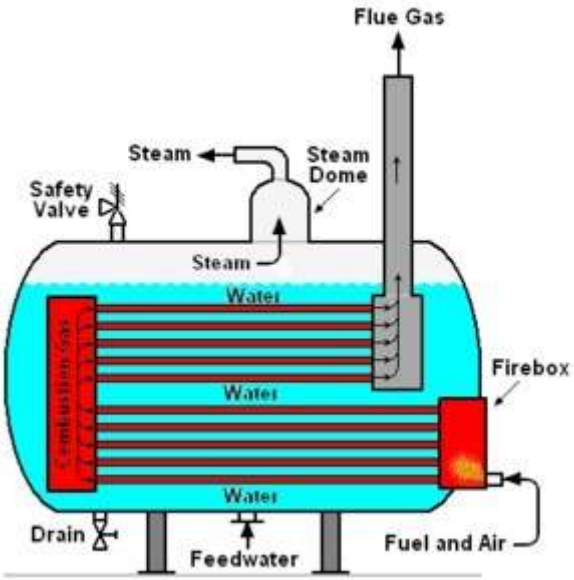
II.3. Mesin Boiler (Katel Uap)

Boiler atau ketel uap adalah suatu bejana tertutup yang di dalamnya berisi air untuk dipanaskan. Energi panas dari uap air keluaran boiler tersebut selanjutnya digunakan untuk berbagai macam keperluan, seperti untuk turbin uap, pemanas ruangan, mesin uap, dan lain sebagainya. Secara proses konversi energi, boiler memiliki fungsi untuk mengkonversi energi kimia yang tersimpan di dalam bahan bakar menjadi energi panas yang tertransfer ke fluida kerja. Bejana bertekanan pada boiler umumnya menggunakan bahan baja dengan spesifikasi tertentu yang telah ditentukan dalam standard ASME (*The ASME Code Boilers*), terutama untuk penggunaan boiler pada industri-industri besar. Dalam sejarah tercatat berbagai macam jenis material digunakan sebagai bahan pembuatan boiler seperti tembaga, kuningan, dan besi cor. Namun bahan-bahan tersebut sudah lama ditinggalkan karena alasan ekonomis dan juga ketahanan material yang sudah tidak sesuai dengan kebutuhan industri. Panas yang diberikan kepada fluida di dalam boiler berasal dari proses pembakaran dengan berbagai macam jenis bahan bakar yang dapat digunakan, seperti kayu, batubara, solar/minyak bumi, dan gas. Dengan adanya kemajuan teknologi, energi nuklir pun juga digunakan sebagai sumber panas pada boiler (artikel-teknologi.com).Berikut adalah beberapa contoh jenis boiler :

1. *Fire-Tube Boiler*

Pada perkembangan selanjutnya muncul desain bari boiler yakni boiler pipa-api. Boiler ini terdapat 2 bagian di dalamnya, yaitu sisi *tube*/pipa dan

sisi *barrel*/tong. Pada sisi *barrel* berisi fluida/air, sedangkan sisi pipa merupakan tempat terjadinya pembakaran.

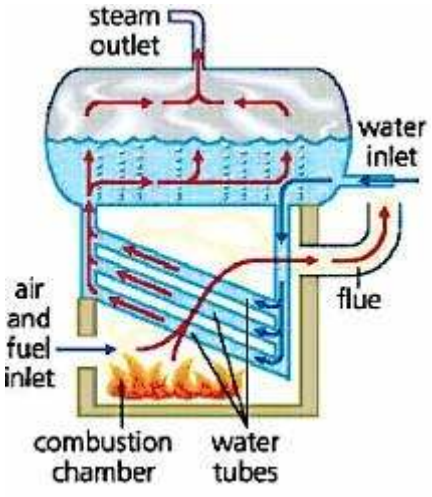


Gambar II.2. Fire-Tube Boiler
(Sumber : artikel-teknologi.com)

Boiler pipa-api biasanya memiliki kecepatan produksi uap air yang rendah, tetapi memiliki cadangan uap air yang lebih besar.

2. *Water-Tube Boiler*

Sama seperti boiler pipa-api, boiler pipa-air juga terdiri atas bagian pipa dan *barrel*. Tetapi sisi pipa diisi oleh air sedangkan sisi *barrel* menjadi tempat terjadinya proses pembakaran. Boiler jenis ini memiliki kecepatan yang tinggi dalam memproduksi uap air, tetapi tidak banyak memiliki cadangan uap air di dalamnya.



Gambar II.3. Wire-Tube Boiler
(Sumber : artikel-teknologi.com)

II.4. Metode *Dempster Shafer*

Metode *Dempster Shafer* pertama kali diperkenalkan oleh *Dempster* yang melakukan percobaan model ketidak pastian dengan *range probabilitas* dari pada sebagai probabilitas tunggal. Kemudian pada tahun1976 *Shafer* mempublikasikan teori Dempster itu pada sebuah buku yang berjudul Mathematical Theory Of Evident, Dempster-Shafer Theory Of Evidence, menunjukan suatu cara untuk memberikan bobot kenyakinan sesuai fakta yang dikumpulkan. Pada teori ini dapat membedakan ketidak pastian dan ketidaktahuan. Teori Dempster-Shafer adalah representasi, kombinasi dan propogasi ketidakpastian, dimana teori ini memiliki beberapa karakteristrik yang secara instutitif sesuai dengan cara berfikir seorang pakar, namun dasar matematika yang kuat. (Mikha Dayan Sinaga; 2016: 96).

Secara umum teori *Dempster Shafer* ditulins dalam suatu interval *Belief,Plausibility*. *Belief* (Bel) adalah ukuran kekuatan *evidence* dalam

mendukung suatu himpunan proposisi. Jika bernilai 0 maka mengindikasikan bahwa tidak ada *evindence*, dan jika bernilai 1 menunjukan adanya kepastian. *Plausibility* (Pls) akan mengurai tingkat kepastian dari *evidence*. *Plausibility* bernilai 0 sampai 1. Jika yakin akan X, maka dapat dikatakan bahwa Bel (X) = 1 sehingga rumus diatas nilai dari Pls(X) = 0.

Fungsi Belief dapat dirumuskan dan ditunjukan pada persamaan II.1 :

$$Bel\ (x) = \sum_{Y \subset X} M\ (y) \quad \text{(II.1)}$$

Dan Plausibility dinotasikan pada persamaan II.2 :

$$Pls\ (x) = 1 - Bel\ (x) = 1 - \sum_{Y \subset X} m(X) \quad \text{(II.2)}$$

Dimana :

Bel (x) = *Belief* (x)

Pls (x) = *Plausibility* (x)

M (x) = Mass *function* dari (x)

M (y) = Mass *function* dari (y)

Teori *Dempster Shafer* menyatakan adanya *frame of discrement* yang dinotasikan dengan simbol $(\emptyset)$. *Frame of discrement* merupakan semeta pembicaraan dari sekumpulan hipotesis sehingga sering disebut dengan *environment* yang ditunjukan pada persamaan II.3 :

$(\emptyset)$ = { 01,02, . . . 0N} . . . . . (II.3)

Dimana :

$(\emptyset)$ = *Frame of discrement* atau *environment*

01, . . ., 0N = element/unsur bagian dalam *environment*

Environment mengandung elemen-elemen yang menggambarkan kemungkinan sebagai jawaban, dan hanya ada satu yang akan sesuai dengan jawaban yang dibutuhkan. Kemungkinan ini dalam teori *Dempster Shafer* disebut dengan *power set* dan dinotasikan dengan P $^{(\emptyset)}$, setiap elemen dalam power set ini memiliki nilai interval antara 0 sampai 1.

M : P $^{(\emptyset)}$ [0,1]

Sehingga dapat dirumuskan pada persamaan II.4 :

$$\Sigma_{Xep\ (\emptyset)}\, m(x) = 1 \quad \text{(II.4)}$$

Dengan :

P $^{(\emptyset)}$ = *Power set*

M (x) = mass *function* (x)

Mass *Function* (m) dalam teori *Dempster Shafer* adalah tingkat kepercayaan dari suatu *evidence* (gejala), sering disebut dengan *evidence measure* sehingga dinotasikan dengan (m). Tujuanya adalah mengaitkan ukuran kepercayaan elemen-elemen 0. Tidak semua *evidence* secara langsung mendukung tiap-tiap elemen. Untuk itu perlu adanya probabilitas fungsi densitas (m), Nilai m tidak hanya mendefinisikan elemen-elemen 0 saja, namun juga semua subsetnya. Sehinga jika 0 berisi n elemen, maka subset 0 adalah 2n. Jumlah semua m dalam subset 0 sama dengan 1. Apabila tidak ada informasi apapun untuk memilih hipotesis, maka nilai :

M{0} = 1,0

Apabila diketahui X adalah subset dari 0, dengan m1 sebagai fungsi densitasnya, dan Y juga merupakan subset dari 0 dengan m2 sebagai fungsi

densitasnya, maka dapat dibentuk fungsi kombinasi m1 dan m2 sebagai m3, yaitu ditunjukan pada persamaan II.5 :

$$m3(Z) = \frac{\sum_{X \cap Y = Z} M_1(X).\, M_2(Y)}{1 - \sum_{X \cap Y = \emptyset} M_1(X).\, M_2(Y)} \quad \text{(II.5)}$$

Dimana :

M3(Z) = mass *function* dari *evidence* (Z)

M1(X) = mass function dari evidence (X), yang diperoleh dari nilai keyakinan suatu evidence dikalikan dengan nilai disbelief dari evidence tersebut.

M2(Y) = mass *function* dari *evidence* (Y), yang diperoleh dari nilai keyakinan suatu evidence dikalikan dengan nilai disbelief dari evidence tersebut.

$$\sum_{X \cap Y = Z} M1(X).\, M2(Y)$$

Merupakan nilai kekuatan dari *evidence* Z yang diperoleh dari kombinasi nilai keyakinan sekumpulan *evidence*.

II.5. Data

Data merupakan deskripsi tentang benda, kejadian, aktivitas, dan transaksi yang tidak mempunyai makna atau tidak berpengaruh secara langsung kepada makna pemakai. Data juga dapat diartikan suatu bahan mentah yang kelak dapat diolah lebih lanjut untuk menjadi sesuatu yang lebih bermakna. Dan deta inilah yang nantinya akan disimpan dalam database. Sedangkan informasi adalah data yang telah diolah menjadi sebuah bentuk yang berarti bagi penerimanya dan bermanfaat dalam pengambilan keputusan saat ini atau saat mendatang, (Muhammad Taufiq; 2013 : 50).

II.6. Database Management System (DBMS)

Database Management System atau disingkat DBMS adalah perangkat lunak atau (*software)* yang berfungsi untuk mengelola *database*. Mulai dari mengelola *database* sampai dengan proses yang berlaku dalam *database* tersebut, baik berupa *entry*, *edit,* hapus, *query* terhadap data, membuat laporan dan lain sebgainya secara efektif dan efesien, (Nurlaila Hasyim ; 2014: 2 - 3).

II.7. SQL Server 2008

SQL Server 2008 adalah sebuah terobosan baru dari Microsoft dalam bidang *database*. SQL Server adalah DBMS (Database Management *System*) yang dibuat oleh Microsoft untuk ikut berkecimpung dalam persaingan dunia pengolahan data menyusul pendahulunya seperti IBM dan Oracle. SQl Server 2008 dibuat pada saat kemajuan dalam bidang *hardware* sedemikian pesat. Oleh karena itu sudah dapat dipastikan bahwa SQL Server 2008 membawa beberapa terobosan dalam bidang pengolahan dan penyimpanan data. Microsoft merilis SQL Server 2008 dalam beberapa versi yang disesuaikan dengan segment-segment pasar yang dituju. Versi-versi tersebut adalah sebagai berikut. Menurut cara pemrosesan data pada prosesor maka Microsoft mengelompokkan produk ini berdasarkan 2 jenis yaitu :

a. Versi 32-bit(x86), yang biasanya digunakan untuk komputer dengan singleprosesor (Pentium 4) atau lebih tepatnyaprosesor 32 bit dan sistem operasi Windows XP.

b. Versi 64-bit(x64), yang biasanya digunakan untuk komputer dengan lebih dari satu prosesor (Misalnya Core 2 Duo) dan system operasi 64 bit seperti Windows XP 64, Vista, dan Windows 7.
c. Versi Compact, ini adalah versi "Tipis" dari semua versi yang ada. Versi ini seperti versi desktop pada SQL Server 2000. Versi ini juga digunakan pada handled drvice seperti Pocket PC, PDA, SmartPhone, Tablet PC.
d. Versi Express, ini adalah versi "Ringan" dari semua versi yang ada(tetapi versi ini berbeda dengan versi compact) dan paling cocok untuk latihan. Express Manager standar, integrasi dengan CLR dan XM, (Agus Tinus Setiawan;2016:54).

II.8. Microsoft Visual Studio 2010

Microsft Visual Studio merupakan sebuah perangkat lunak lengkap (*suite*) yang dapat digunakan untuk melakukan pengembangan aplikasi, baik itu aplikasi bisnis, aplikasi personal, ataupun komponen aplikasinya, dalam bentuk aplikasi console, aplikasi Windows, ataupun aplikasi Web. Visual Studio mencakup kompiler, *Software Development Kit* (SDK), *Integrated Development Environment* (IDE), dan dokumentasi (umumnya berupa MSDN Library). Kompiler yang dimasukkan ke dalam paket Visual Studio antara lain Visual C++, Visual C#, Visual Basic, Visual Basic .NET, Visual InterDev, Visual J++, Visual J#, Visual FoxPro, dan Visual SourceSafe. Microsoft Visual Studio dapat digunakan untuk mengembangkan aplikasi dalam *native code* (dalam bentuk bahasa mesin yang berjalan di atas Windows) ataupun *managed code* (dalam

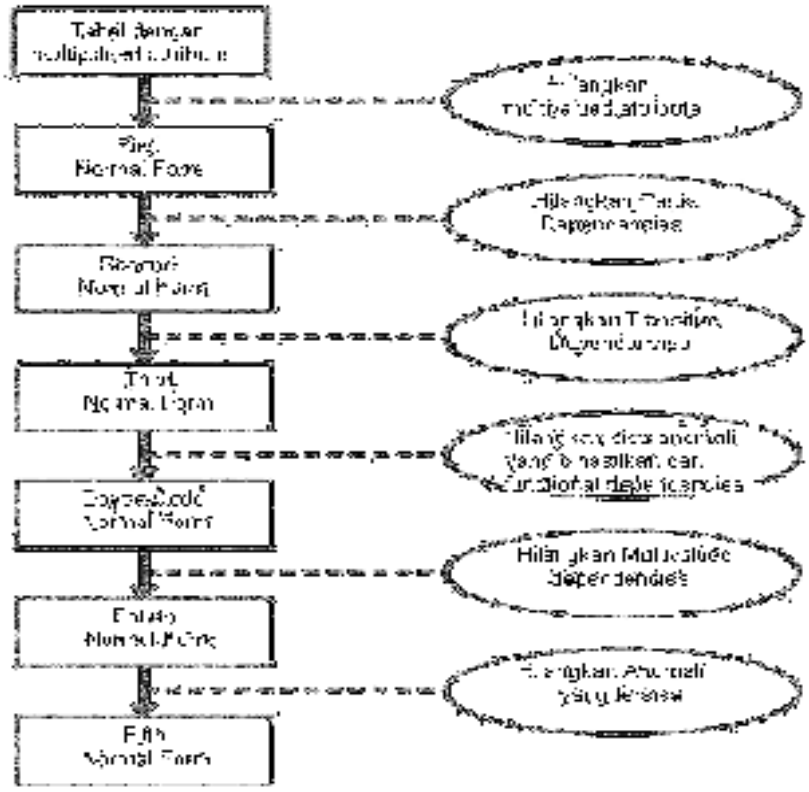
bentuk Microsoft Intermediate Language di atas .NET Framework). Selain itu, Visual Studio juga dapat digunakan untuk mengembangkan aplikasi Silverlight, aplikasi Windows Mobile (yang berjalan di atas .NET Compact Framework). Visual Studio sebelumnya versi Visual Studio 9.0.21022.08, atau dikenal dengan sebutan Microsoft Visual Studio 2008 yang diluncurkan pada 19 November 2007, yang ditujukan untuk platform Microsoft .NET Framework 3.5. Versi sebelumnya, Visual Studio 2005 ditujukan untuk platform .NET Framework 2.0 dan 3.0. Visual Studio 2003 ditujukan untuk .NET Framework 1.1, dan Visual Studio 2002 ditujukan untuk .NET Framework 1.0. Versi-versi tersebut di atas kini dikenal dengan sebutan Visual Studio .NET, karena memang membutuhkan Microsoft .NET Framework. Sementara itu, sebelum muncul Visual Studio .NET, terdapat Microsoft Visual Studio 6.0 (VS1998), (Herpendi; 2016: 1).

II.9. Normalisasi

Normalisasi merupakan parameter digunakan untuk menghindari duplikasi trhadap tabel dalam basis data dan juga merupakan proses mendekomposisikan sebuah tabel yang masih memiliki beberapa anomali atau ketidak wajaran sehingga menghasilkan tabel yang lebih sederhana dan struktur yang bagus, yaitu sebuah tabel yang tidak memiliki data r*edundancy* dan memungkinkan *user* untuk melakukan *insert, delete*, dan *update* pada baris (*recod*) tanpa menyebabkan inkonsistensi data. Tujuannya untuk menghindari beberapa anomali:

1. *Insertion Anomaly* adalah proses melakukan penambahan *recod* baru akan tetapi mempengaruhi *user* untuk terjadinya duplikasi data.

2. *Deletion Anomaly* adalah proses melakukan penghapusan *record* akan tetapi akan menyebabkan hilangnya data yang akan dibutuhkan pada *record* lain.
3. *Modification Anomaly* adalah proses merubah data pada sebuah *record* mempengaruhi perubahan pada *record* lain karena adanya duplikasi.



Gambar II.4. Langkah-Langkah Dalam Pernomalan Tabel
(Sumber : Gandung Triyono;2012:2)

Proses pernomalan tabel ada beberapa tahap yang harus dilakukan yaitu :

1. First Normal Form (1 NF)

   Sudah tidak ada repeating group yaitu pengulangan yang terjadi pada beberapa atribut atau kolom dalam sebuah tabel, dan juga setiap atribut harus bernilai tunggal. Atribut multivalued, composite, derive tidak tunggal. Setiap nilai dari atribut hanya mempunyai nilai tunggal.

2. Second Normal Form (2 NF)

   Untuk menjadikan tabel normal tingkat ke 2 maka sudah 1NF dan setiap atribut yang bukan primary key sepenuhnya secara fungsional tergantung pada semua atribut pembentuk primary key.

3. Third Normal Form (3 NF)

   Tabel sudah 2NF dan tidak memiliki transitive dependencies, Transitive Dependencies adalah ketika atribut yanng secara tidak langsung tergantung pada primary key dan atribut tersebut juga tergantung pada atribut lain yang bukan primary key.

4. Boyce-codd Normal Form (BCNF)

   Tabel dalam BCNF jika sudah 3NF dan semuah determinants adalah candidate keys. Perbedaan 3NF dan BCNF adalah untuk functional dependency A – B, 3NF memperbolehkan ketergantungan ada dalam relasi juka B adalah primary key dan A bukan merupakan candidate key. Sedangkan BCNF menuntuk untuk ketergantungan tetap ada dalam relasi, A harus menjadi candidate key.

5. Fourt Normal Form (4 NF)

   Relasi berada pada pentuk normal keempat apabila memenuhi syarat BCNF dan tidak mempunyai multivalue dependency.

6. Fifth Normal Form (5 NF)

   Tabel bentuk normal kelima sering disebut PJNF (Projection Join Normal Form), Penyebutan PJNF karena untuk suatu relasi akan berbentuk normal kelima jika tabel tersebut dapat dipecah atau diproyeksikan menjadi beberapa

tabel dan dari proyeksi-proyeksi itu dapat disusun kembali (join) menjadi tabel yang sama dengan keadaan semula. Jika penyusunan ini tidak mungkin dilakukan dikatakan pada relasi itu terdapat join dependencies dan dikatakan bersifat lossy join. (Gandung Triyono;2012:2).

II.10. UML (*Unified Modeling Language*)

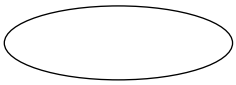
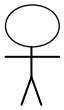
Unified Modeling Language (UML). UML adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak. UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodelan umum dalam industri perangkat lunak dan pengembangan sistem. (Gellysa; 2015 : 93-94).

Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut :

1. *Use case* Diagram

   Use case diagram merupakan pemodelan untuk kelakukan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dapat dikatakan *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Simbol-simbol yang digunakan dalam *use case* diagram, yaitu :

Tabel II.1. Simbol *Use Case*

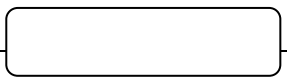
Gambar	Keterangan
	*Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukan pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama *use case*.
	Aktor adalah *abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan aktor, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa muncul dalam beberapa peran. Perlu dicatat bahwa aktor berinteraksi dengan *use case*, tetapi tidak memiliki control terhadap *use case*.
	Asosiasi antara aktor dan *use case*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengidikasikan aliran data.
	Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengidinkasikan bila aktor berinteraksi secara pasif dengan sistem.
	Include, merupakan di dalam *use case* lain (*required*) atau pemanggilan *use case* oleh *use case* lain, contohnya adalah pemanggilan sebuah fungsi program.
	Extend, merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.

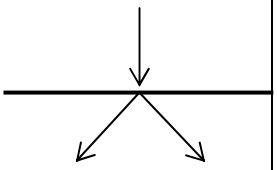
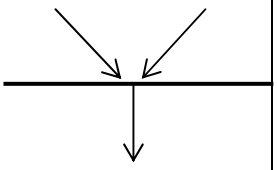
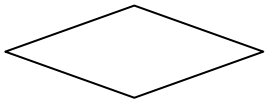
(Sumber : Gellysa; 2015 : 93-94)

2. Diagram Aktivitas (*Activity Diagram*)

Activity Diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Simbol-simbol yang digunakan dalam *activity diagram,* yaitu :

Tabel II.2. Simbol *Activity Diagram*

Gambar	Keterangan
	*Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	*End point*, akhir aktifitas.
	*Activites*, menggambarkan suatu proses/kegiatan bisnis.

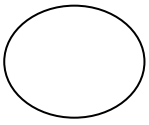
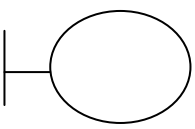
	*Fork* (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan pararel menjadi satu.
	*Join* (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi.
	*Decision Points,* menggambarkan pilihan untuk pengambilan keputusan, *true, false.*
	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa.

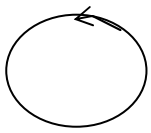
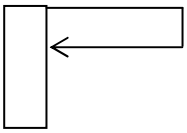
(Sumber : Gellysa; 2015 : 94)

3. Diagram Urutan (*Sequence Diagram*)

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Simbol-simbol yang digunakan dalam *sequence diagram*, yaitu :

Tabel II.3. Simbol *Sequence Diagram*

Gambar	Keterangan
	Entity Class, merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data.
	Boundary Class, berisi kumpulan kelas yang menjadi interface atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan formentry dan form cetak

	Control class, suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek.
	Message, simbol mengirim pesan antar class.
	Recursive, menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri.
	Activation, activation mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivitas sebuah operasi.
	Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang lifeline terdapat activation.

(Sumber : Gellysa; 2015 : 95)

4. *Class Diagram* (Diagram Kelas)

Merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggng jawab entitas yang menentukan perilaku sistem.

Class diagram juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan. *Class diagram* secara khas meliputi: Kelas (*Class*), Relasi, *Associations*, *Generalization* dan *Aggregation*, Atribut (*Attributes*), Operasi (*Operations/Method*), *Visibility,* tingkat akses objek eksternal kepada suatu

operasi atau atribut. Hubungan antar kelas mempunyai keterangan yang disebut dengan *multiplicity* atau kardinaliti.

Tabel II.4. ***Multiplicity Class Diagram***

Multiplicity	Penjelasan
1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1 atau lebih
1..*	1 atau lebih
0..1	Boleh tidak ada, maksimal 1
n..n	Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimum 4

(Sumber : Gellysa; 2015 : 95)