BAB II TINJAUAN PUSTAKA

2.1. Penelitian Terdahulu

Penelitian sistem pakar ini dikembangkan dengan tujuan untuk membantu dan mempermudah dalam mengidentifikasi serta dapat memberikan solusi kepada *user* secara cepat. Dalam penelitian sebelumnya menunjukan bahwa sistem pakar dapat membantu pekerjaan para ahli/pakar tanpa harus bertemu langsung. Referensi dan rujukan hasil penelitian sebelumnya yang telah dilakukan oleh peneliti lain merupakan tujuan pada tinjauan pustaka ini.

Salah satu penelitian terkait pada tahun 2016, yaitu berjudul "Perancangan Aplikasi Sistem Pakar Diagnosa Kerusakan *Hardware* Komputer Metode *Forward Chaining*" Yang ditulis oleh Ali Akbar Rismayadi[1]. Pada penelitian tersebut menggunakan metode *Forward Chaining* dalam kerusakan *hardware* komputer kemudian dalam pengembangan sistem pada jurnal tersebut menggunakan metode *waterfall* serta metode penelitian tersebut menggunakan metode pengamatan, wawancara serta studi pustaka. Hasil yang didapatkan pada penelitian tersebut yaitu dapat digunakan sebagai alat bantu untuk menemukan penyebab kerusakan pada perangkat keras komputer[1].

Penelitian lainnya pada tahun 2017, yaitu berjudul "Implementasi Metode *Naïve Bayes* Pada Aplikasi Sistem Pakar Berbasis Web Untuk Mengdiagnosa Kerusakan Pada *Hardware* Komputer" yang di tulis oleh Eric Agung Lestari Jiwono, Syaiful Rahman, Hasniati[7]. Metode yang digunakan dalam penelitian ini adalah metode *naïve bayes*, yang bertujuan untuk menguji kinerja dari metode *naïve bayes* pada sistem pakar yang dapat membantu pengguna sistem tersebut. Berdasarkan hasil penelitian ini metode *naïve bayes* berhasil diterapkan kedalam sistem pakar untuk mendiagnosis kerusakan perangkat keras komputer dengan tingkat akurasi 100% berdasarkan 18 pengujian[7].

Penelitian lainnya pada tahun 2018, yaitu berjudul "Implementasi Metode *Forward Chaining* Dalam Sistem Pendeteksi Kerusakan *Hardware* Pada Komputer Dan Laptop Berbasis Android" yang ditulis oleh Peti Savitri dan Trisna Hadi[8]. Pada penelitian ini disebutkan bahwa metode yang digunakan adalah metode *Forward Chaining* kemudian metode yang digunakan untuk membangun sistem tersebut adalah metode *prototyping*. Hasil yang didapatkan dalam penelitian ini bahwa aplikasi sistem pakar pendeteksi kerusakan *hardware* komputer ini dapat membantu *user* pemula mengetahui letak kerusakan pada komputer atau laptop, sehingga pada akhirnya membantu mereka untuk segera mengambil tindakan dalam penangan *error* yang mereka temukan[8].

Penelitian lainnya pada tahun 2017, yaitu berjudul "Penerapan Metode *Certainty Factor* Untuk Sistem Pakar Diagnosis Hama Dan Penyakit Pada Tanaman Tembakau" yang ditulis oleh Mohammad Arifin, Slamin, Windi Eka Yulia Retnani[9]. Pada penelitian tersebut menggunakan metode *certainty factor,* yang bertujuan untuk membantu para petani dari serangan penyakit dan hama pada tanaman tembakau. Hasil dari penelitian ini menghasilkan tingkat akurasi sebesar 99.9% dan keakuratan proses perhitungan metode *certainty factor* dipengaruhi oleh pemilihan data gejala yang ada, dan metode *certainty factor* untuk mendiagnosis hama dan penyakit pada tanaman tembakau masih sangat cocok[9].

Penelitian lainnya pada tahun 2020, yaitu berjudul "Sistem Pakar Diagnosa Kerusakan Pada Laptop Menggunakan Metode *Certainty Factor*" yang ditulis oleh Heri Mulyono, Regina Ade Darman, Gefli Ramadhan[10]. Pada penelitian ini menggunakan metode *certainty factor*, yang bertujuan untuk memberikan informasi tentang masalah, penyebab, serta solusi yang didapat untuk menangani kerusakan laptop. Dan hasil yang didapat dengan metode *certainty factor* memperoleh ke validan 91,66% menjadi bukti nyata bahwa diagnosa gejala setiap pakar mempengaruhi tingkat keakuratan sistem. Jika melibatkan lebih dari satu pakar, maka pakar-pakar tersebut harus mendiskusikan gejala yang tepat sehingga keakuratan sistem memiliki presentase yang lebih baik [10].

Tabel 2.1 Penelitian Terdahulu

No.	Judul Penelitian	Masalah	Metode	Hasil
1.	"Perancangan Aplikasi Sistem Pakar Diagnosa Kerusakan *Hardware* Komputer Metode *Forward Chaining*[1]"	Bagaimana cara membangun sistem pakar dengan metode *forward chaining* untuk mengidentifikasi kerusakan pada *hardware* komputer.	*Forward Chaining*	Hasil yang didapatkan pada penelitian tersebut yaitu dapat digunakan sebagai alat bantu untuk menemukan penyebab kerusakan pada perangkat keras komputer.
2.	"Implementasi Metode *Naïve Bayes* Pada Aplikasi Sistem Pakar Berbasis Web Untuk Mengiagnosa Kerusakan Pada *Hardware* Komputer[7]"	Bagaimana cara membangun sistem pakar kerusakan pada *hardware* komputer menggunakan metode *naïve bayes* lalu bagaimana hasil kinerjanya.	*Naïve Bayes*	Berdasarkan hasil penelitian ini metode *naïve bayes* berhasil diterapkan kedalam sistem pakar untuk mendiagnosis kerusakan perangkat keras komputer dengan tingkat akurasi 100% berdasarkan 18 pengujian.

3.	"Implementasi Metode *Forward Chaining* Dalam Sistem Pendeteksi Kerusakan *Hardware* Pada Komputer Dan Laptop Berbasis Android[8]"	Meningkatnya kasus kerusakan *hardware* sehingga diperlukan pecegahan untuk dapat mengatasi kerusakan *hardware* tersebut dengan mengidentifikasinya terlebih dahulu.	*Forward Chaining*	Hasil yang didapatkan dalam penelitian ini bahwa aplikasi sistem pakar pendeteksi kerusakan *hardware* komputer ini dapat membantu *user* pemula mengetahui letak kerusakan pada komputer atau laptop, sehingga pada akhirnya membantu mereka untuk segera mengambil tindakan dalam penangan *error* yang mereka temukan.
4.	"Penerapan Metode *Certainty Factor* Untuk Sistem Pakar Diagnosis Hama Dan Penyakit Pada Tanaman Tembakau[9]"	Meningkatnya penyakit dan hama pada tanaman tembakau sering sekali terjadi salah penanganan oleh petani dikarenakan kurangnya pengalaman dalam menangani penyakit atau hama pada tanaman tembakau, sehingga memberikan penanganan yang terkadang salah.	*Certainty Factor*	Hasil dari penelitian ini menghasilkan tingkat akurasi sebesar 99.9% dan keakuratan proses perhitungan metode *certainty factor* dipengaruhi oleh pemilihan data gejala yang ada, dan metode *certainty factor* untuk mendiagnosis hama dan penyakit pada

				tanaman tembakau masih sangat cocok.
5.	"Sistem Pakar Diagnosa Kerusakan Pada Laptop Menggunakan Metode *Certainty Factor*[10]"	Bagaimana membuat aplikasi yang dapat membantu para *user* dalam mendiagnosa kerusakan pada laptop	*Certainty Factor*	Hasil yang didapat dengan metode *certainty factor* memperoleh ke validan 91,66% menjadi bukti nyata bahwa diagnosa gejala setiap pakar mempengaruhi tingkat keakuratan. sistem. Jika melibatkan lebih dari satu pakar, maka pakar-pakar tersebut harus mendiskusikan gejala yang tepat sehingga keakuratan sistem memiliki presentase yang lebih baik

2.2. Dasar Teori

Dalam laporan tugas akhir ini digunakan beberapa teori yang diperlukan untuk mendukung kegiatan yang dilakukan. Beberapa landasan teori yang dikemukakan tersebut meliputi konsep dasar dan definisi yang berkaitan dengan perangkat yang digunakan sebagai faktor-faktor pendukung dalam melaksanakan tugas akhir ini.

2.2.1. Perangkat Keras / *Hardware*

Hardware adalah bagian fisik komputer yang sifatnya dapat dilihat langsung, perangkat keras tersebut digunakan sebagai perangkat bantuan untuk memaksimalkan kinerja komputer yang dimiliki, dan ada banyak sekali macam-macam perangkat keras yang dibutuhkan komputer dan dapat di bedakan dengan data yang berada di dalamnya[11].

2.2.2. Komputer

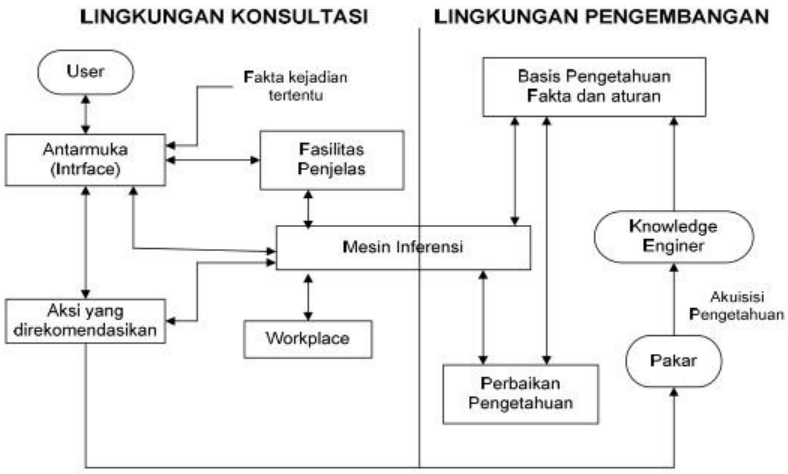
Di masa kehidupan manusia saat ini sepertinya sudah tidak asing serta tidak lepas dengan adanya teknologi, khususnya komputer. Bahkan, banyak peralatan komputer saat ini yang sudah dimiliki oleh masyarakat, seperti *mobile phone, laptop, dan* komputer *desktop.* Komputer semula dipergunakan untuk menggambarkan orang yang perkerjaannya melakukan perhitungan aritmatika[12]. Yang dimana komputer dapat melakukan perhitungan yang besar dan cepat, termasuk perhitungan aritmatika yang besar atau operasi logika, dan menghasilkan *output* berdasarkan instruksi-instuksi yang tersimpan pada memori [13].

Pada intinya, komputer adalah sebuah perangkat elektronik yang dapat menerima input sesuai dengan instruksi yang diberikan, mengolah instruksi tersebut, lalu memberikan output berupa informasi. komputer juga dapat menyimpan program dan hasil pengolahan data tersebut, serta dapat berkerja secara otomatis.

2.2.3. Sistem Pakar

Sistem pakar adalah aplikasi berbasis komputer yang digunakan untuk menyelesaikan masalah sebagaimana yang dipikirkan oleh pakar. Pakar yang dimaksud disini adalah orang yang mempunyai keahlian khusus yang dapat menyelesaikan masalah yang tidak dapat diselesaikan oleh orang awam [14]. Sistem pakar merupakan salah satu bagian dari Kecerdasan Buatan yang mengandung pengetahuan dan pengalaman yang dimasukkan oleh satu atau banyak pakar ke dalam suatu area pengetahuan tertentu, sehingga setiap orang dapat menggunakannya untuk memecahkan berbagai masalah yang bersifat spesifik[15]. Dengan sistem pakar ini, orang awampun dapat menyelesaikan masalah yang cukup rumit yang sebenarnya hanya dapat diselesaikan dengan bantuan para ahli. Bagi para ahli, sistem pakar juga akan membantu aktivitasnya sebagai asisten yang sangat berpengalaman[16].

Sistem pakar memiliki 2 bagian utama antara lain lingkungan pengembangan dan lingkungan konsultasi, lingkungan pengembangan yaitu bagian yang digunakan untuk memasukan pengetahuan pakar maupun pengembangan sistem tersebut dan sedangkan bagian lingkungan konsultasi yaitu lingkungan yang digunakan pengguna (*users*) atau yang bukan ahli untuk melakukan konsultasi, struktur sistem pakar dalam dua bagian tersebut dapat dilihat pada gambar 2.1 berikut ini[17].



Gambar 2.1 Struktur Sistem Pakar

2.2.4. Certainty Factor

Certanty Factor Teori *Certainty Factor* (CF) diusulkan oleh Shortlife dan Buchanan pada 1975 untuk mengakomodasi ketidakpastian pemikiran (*inexact reasoning*) seorang pakar. Seorang pakar, (misalnya dokter) sering kali menganalisis informasi yang ada dengan ungkapan seperti "mungkin", "kemungkinan besar", hampir pasti". Untuk mangakomodasi hal ini kita menggunakan *certainty factor* (CF) guna menggambarkan tingkat keyakinan pakar terhadap masalah yang sedang dihadapi[18]. Berikut merupakan perhitungan metode *net belief* yang diusulkan oleh E.H. Shortliffe dan B.G.Buchanan[19]:

$$\mathrm{CF(H)} = \mathrm{MB[h,e]} - \mathrm{MD[h,e]} \quad \text{.....(1)}$$

$$\mathrm{CF(H,E)} = \mathrm{CF[E]} * \mathrm{CF[H]} \quad \text{.....(2)}$$

Keterangan:

CF (H): Faktor Kepastian

MB (h, e): *Measure of Belief* (Ukuran kepercayaan) terhadap hipotesis H, jika diberikan *evidence* E (antara 0 dan 1).

MD (h.e): *Measure of Disbelief* (Ukuran ketidakpercayaan) terhadap e*vidence* H, jika diberikan *evidence* E (antara 0 dan 1).

P (H): Probabilitas kebenaran hipotesis H.

P (H|E): Probabilitas bahwa H benar karena faktor E.

Certainty factor untuk kaidah *evidence* tunggal:

$$\mathrm{CF[H,E]1} = \mathrm{CF[H]} * \mathrm{CF[E]} \quad \text{.....(3)}$$

Jika ditemukan dua *evidence*, maka kombinasi CF dari setiap *evidence* menggunakan persamaan seperti di bawah.

$$\mathrm{CF} = \mathrm{CF[H,E]1,2} = \mathrm{CF[H,E]1} + \mathrm{CF[H,E]2} * [1 - \mathrm{CF[H,E]1}] \quad (4)$$

Jika ditemukan lebih dari dua *evidence*, maka kombinasi dari nilai cf sebelumnya (CFold) dengan *evidence* baru, menggunakan persamaan seperti berikut:

$$\mathrm{CF} = \mathrm{CF[H,E]old,3} = \mathrm{CF[H,E]old} + \mathrm{CF[H,E]3} * [1 - \mathrm{CF[H,E]old}] \quad \text{.....(5)}$$

Metode *certainty factor* ini hanya bisa mengolah 2 bobot dalam sekali perhitungan. Untuk bobot yang lebih dari 2 banyaknya, untuk melakukan perhitungan tidak terjadi masalah apabila bobot yang dihitung teracak, artinya tidak ada aturan untuk mengkombinasikan bobotnya, karena untuk kombinasi seperti apapun hasilnya akan tetap sama[20].

2.2.5. *Website*

Website adalah suatu sumber informasi, yang berbentuk teks, gambar, video, suara serta berupa animasi. Web sendiri sebenarnya adalah sekumpulan data dan dokumen yang sangat banyak yang berada pada komputer *server*[21]. Adapun pengertian lainnya adalah sebuah aplikasi yang berjalan di internet, seperti halnya email, web sendiri memiliki sebuah sistem protokol dimana sistem dapat memformat, menampilkan, dan menyimpan informasi dan protokol yang digunakan adalah HTTP (*Hypert Text Transport Protocol*) dan bahasa

pemrograman yang sangat indentik dengan web adalah Bahasa pemrograman *HTML (Hyper Text Markup Language)*[22].

2.2.6. PHP

PHP (*Hypertext Preprocessor*) merupakan salah satu Bahasa pemrograman *opensource* yang cocok digunakan untuk membuat *website* dan dapat ditanamkan pada sebuah *script. PHP* sendiri juga bersifat *server- side script* yang mampu dijalankan berbagai sistem operasi seperti linux, windows, dan lainnya dan dalam database *PHP* memiliki kedinamisan yang dapat dihubungkan langsung dengan *MySQL*[23].

2.2.7. Basis Data

Basis data atau *Database* adalah sekumpulan informasi yang disusun dan merupakan suatu kesatuan yang utuh yang disimpan di dalam perangkat keras (komputer) secara sistematis sehingga dapat diolah menggunakan perangkat lunak[24]. Adapun pengertian lain adalah kumpulan data yang secara *logic* berkaitan dalam mempresentasikan fenomena/fakta secara terstruktur dalam *domain* tertentu untuk mendukung aplikasi dalam sistem tertentu[25]. Dari pendapat tersebut basis data terdiri atas semua fakta yang diperlukan, dimana fakta tersebut dapat digunakan untuk memenuhi kondisi yang dibutuhkan oleh sistem. hampir semua program aplikasi yang melibatkan pengolahan data dapat dipastikan menggunakan basis data sebagai tempat penyimpanan datanya[26].

2.2.8. MYSQL

MySQL merupakan basis data yang memiliki satu atau lebih jumlah tabel. MySQL adalah salah satu jenis *database server* yang sangat terkenal dan banyak digunakan untuk membangun aplikasi web yang mana *database* sebagai sumber dan pengelolaan datannya[27].

2.2.9. Laragon

Laragon ialah sebuah *software* atau perangkat lunak bebas, yang di dalamnya terdapat banyak sistem operasi yang digunakan sebagai

localhost atau *server* mandiri. Banyak sekali layanan yang diberikan dalam laragon tersebut, macam-macam fitur tersebut terdiri dari *Apache, PHP server, PHPMyAdmin, MySQL, Laravel*[23].

2.2.10. *Blackbox Testing*

Blackbox testing merupakan sebuah metode untuk menguji sebuah perangkat lunak tanpa harus memperhatikan perangkat lunak tersebut secara rinci. *Blackbox testing* merupakan penguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program[26]. Pada *black box testing* ini, perangkat lunak tersebut akan dieksekusi kemudian berusaha dites apakah sudah memenuhi kebutuhan pengguna yang didefinisikan pada saat awal tanpa harus membongkar *source* programnya[28]. Pada proses *Black Box Testing* dengan cara mencoba program yang telah dibuat, dengan mencoba memasukkan data pada setiap formnya. Pengujian ini diperlukan untuk mengetahui program tersebut berjalan sesuai dengan yang dibutuhkan[29].

2.2.11. *Whitebox Testing*

Whitebox testing adalah sebuah metode yang dikenal sebagai pengujian struktural, pengujian *transparent box*, pengujian berbasis logika atau pengujian berbasis kode. Kata *whitebox* (kotak putih/transparan) mengacu pada sebuah metode *test case*, perangkat lunak yang sedang diuji dianggap sebagai kotak(*box*), sedangkan kata *white/transparent* mengacu pada bahwa kotak itu terlihat jelas isinya[30]. Penguji dalam *whitebox testing* memiliki pengetahuan tentang kode dan penulisan kasus uji dengan parameter yang sesuai[31]. Hal ini terutama menyangkut dengan aliran kontrol dan aliran data suatu program[32].

a. Basic path testing

Basic path testing merupakan metode untuk merancang *test case* dalam membuat pengukuran kompleksitas logikal dalam perancangan prosedural dan menggunakan pengukuran ini sebagai panduan untuk mendefinisikan

sebuah himpunan basis dari jalur eksekusi[33]. Pada tahap pengujian menggunakan *basis path* terdapat beberapa tahapan yaitu dengan membuat *flowgraph* dari fungsi yang akan diuji, menghitung *cyclomatic complexity* (CC) dan melakukan *unit test*, rumus pada *cyclomatic complexity*[34]. yaitu:

$$V(G) = E - N + 2$$

Keterangan:

V(G) = *Cyclomatic Complexity*

E = Jumlah *Edge*

N = Jumlah *Node*

2.2.12. *Confusion Matrix*

Confusion matrix adalah suatu metode yang digunakan untuk melakukan perhitungan akurasi pada konsep data *mining*. Evaluasi dengan *confusion matrix* menghasilkan nilai akurasi, *presisi* dan *recall*. Akurasi dalam klasifikasi adalah persentase ketepatan *record* data yang diklasifikasikan secara benar setelah dilakukan pengujian pada hasil klasifikasi[35]. Berikut ini adalah model metode dari *confusion matrix*:

Tabel 2.2 *Confusion Matrix*

Nilai Prediksi	Nilai Aktual	
	TP	FP
	FN	TN

Dimana:

- TP adalah *True Positif*, yaitu jumlah data positif yang terklasifikasi dengan benar oleh sistem.
- TN adalah *True Negatif*, yaitu jumlah data negatif yang terklasifikasi dengan benar oleh sistem.
- FN adalah *False Negatif*, yaitu jumlah data negatif namun terklasifikasi salah oleh sistem.
- FP adalah *False Positif*, yaitu jumlah data positif namun terklasifikasi salah oleh sistem.

Dengan itu nilai akurasi merupakan perbandingan data yang terklasifikasi dengan keseluruhan data, nilai akurasi juga dapat di dapat kan dengan rumus sebagai berikut[36]:

a. Akurasi digunakan untuk menggukur kinerja sebuah metode.
   Dengan rumus:

$$\text{Akurasi} = \frac{TP+TN}{TP+TN+FP+FN} \; X \; 100\% \quad (4)$$

b. Nilai *presisi* didefinisikan sebagai rasio item relevan yang dipilih terhadap semua item yang terpilih atau dapat diartikan sebagai kecocokan antara permintaan informasi degan jawaban terhadap permintaan tersebut, dengan rumus sebagai:

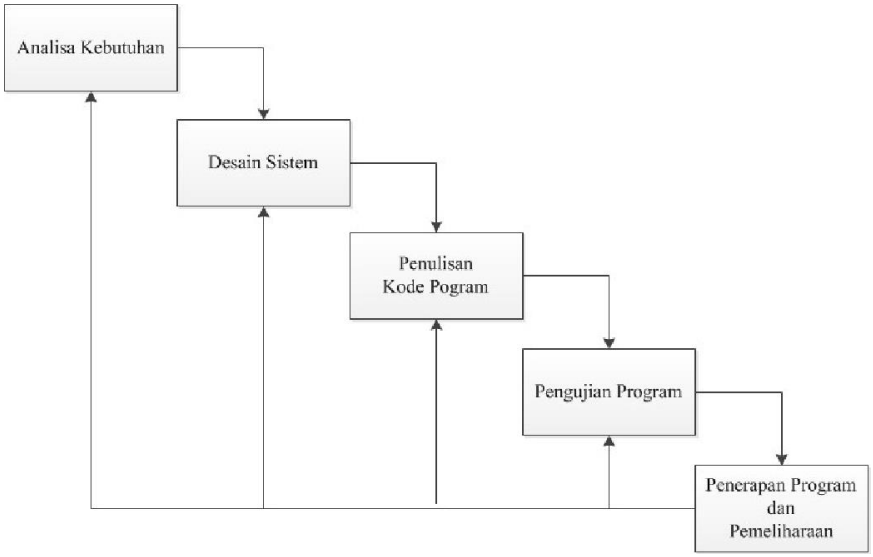
$$Presisi = \frac{TP}{TP+FP} \quad (5)$$

c. *Recall* didefinisikan sebagai rasio dari item relevan yang dipilih terhadap total jumlah item relevan yang tersedia. *Recall* dihitung dengan rumus:

$$Recall = \frac{TP}{TP+FN} \quad (6)$$

2.2.13. *Waterfall*

Model pengembangan *software* yang diperkenalkan oleh Winston Royce pada tahun 70-an ini merupakan model klasik yang sederhana dengan aliran sistem yang linier keluaran dari tahap sebelumnya[37]. Metode *Waterfall* adalah suatu proses pengembangan perangkat lunak berurutan, di mana kemajuan dipandang sebagai terus mengalir ke bawah (seperti air terjun) melewati fase-fase perencanaan, pemodelan, implementasi (konstruksi), dan pengujian[38].



Gambar 2.2 Model *Waterfall*

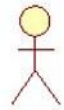
Model ini melakukan pendekatan secara sistematis dan berurutan. Disebut dengan *waterfall* karena tahap demi tahap yang dilalui harus menunggu selesainya tahap sebelumnya dan berjalan berurutan[39].

2.2.14. UML

Unified Modeling Language (UML) merupakan Teknik pemrograman berorientasi objek ini yang muncul akibat adanya pemodelan visual yang berfungsi untuk menggambarkan dan mendokumentasikan dari sistem perangkat lunak[40]. UML hanya berfungsi sebagai pemodelan jadi penggunaan UML tidak terbatas pada metodologi tertentu[41]. Terdapat beberapa diagram UML yang sering digunakan dalam pengembangan maupun pembuatan sebuah sistem, yaitu[42]:

a. *Use case*

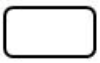
Use case diagram menjelaskan manfaat dari aplikasi jika dilihat dari sudut pandang orang yang berada diluar sistem (*aktor*). Diagram ini menunjukkan fungsionalitas suatu sistem atau kelas dan bagaimana sistem berinteraksi dengan dunia luar, berikut merupakan contoh simbol pada *use case*:

Simbol	Nama	Keterangan
	*Actor*	Menggambarkan manusia atau suatu hal yang menggunakan atau berinteraksi dengan sistem.
	*Use Case*	Menjelaskan bagian utama dari kegunaan sistem.
	Association Relationship	Sebagai penghubung antara *actor* dengan *use case* yang saling berinteraksi.
	Directed Association Relationship	Hubungan asosiasi yang diarahkan hanya kepada satu arah.

Gambar 2.3 Simbol *Use case* Diagram

b. *Activity* Diagram

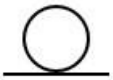
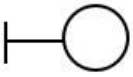
Merupakan gambaran alir dari aktivitas-aktivitas didalam sistem yang berjalan, berikut merupakan contoh simbol dari *activity* diagram:

Simbol	Nama	Keterangan
	Status awal	Sebuah diagram aktivitas memiliki sebuah status awal.
	Aktivitas	Aktivitas yang dilakukan sistem,aktivitas biasanya diawali dengan kata kerja.
	Percabangan / Decision	Percabangan dimana ada pilihan aktivitas yang lebih dari satu.
	Penggabungan / Join	Penggabungan dimana yang mana lebih dari satu aktivitas lalu digabungkan jadi satu.
	Status Akhir	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir
	Swimlane	Swimlane memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi

Gambar 2.4 Simbol *Activity* Diagram

c. *Seqeunce* Diagram

Sequence diagram menjelaskan interaksi antar objek di dalam dan di sekitar sistem berupa pesan (*message*) yang disusun dalam suatu urutan waktu yaitu urutan kejadian yang dilakukan oleh seorang aktor dalam menjalankan sistem berikut merupakan contoh simbol dari *sequence* diagram:

NO	GAMBAR	NAMA	KETERANGAN
1		*Actor*	Menggambar orang yang sedang berinteraksi dengan sisitem.
2		*Entity Class*	Menggambarkan hubungan yang akan dilakukan
3		*Boundary Class*	Menggambarkan sebuah gambbaran dari foem
4		*Control Class*	Menggambarkan penghubung antara boundary dengan tabel
5		*A focus of Control & A Life Line*	Menggambarkan tempat mulai dan berakhirnya massage
6		*A massage*	Menggambarkan Pengiriman Pesan

Gambar 2.5 Simbol *Sequence* Diagram

d. *Class* Diagram

Class diagram memberikan gambaran hubungan antara tabel-tabel yang ada dalam *database*, berikut adalah contoh simbol dari *class* diagram:

SIMBOL CLASS DIAGRAM

NO	GAMBAR	NAMA	KETERANGAN
1		*Generalization*	Hubungan dimana objek anak *(descendent)* berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
2		*Nary Association*	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
3		*Class*	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
4		*Collaboration*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu actor
5		*Realization*	Operasi yang benar-benar dilakukan oleh suatu objek.
6		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempegaruhi elemen yang bergantung padanya elemen yang tidak mandiri
7		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya

Gambar 2.6 Simbol *Class* Diagram