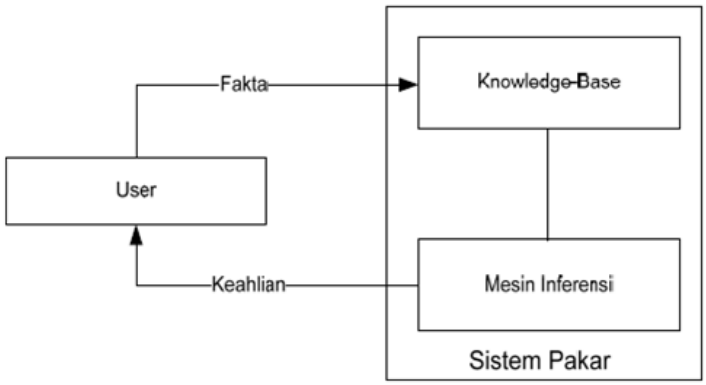
BAB II
TINJAUAN PUSTAKA

2.1 Teori Judul
2.1.1 Sistem Pakar

Andriani (2017:9) menjelaskan sistem pakar adalah sebuah sistem yang kinerjanya mengadopsi keahlian yang dimiliki seorang pakar dalam bidang tertentu ke dalam sistem atau program komputer yang disajikan dengan tampilan yang dapat digunakan oleh pengguna yang bukan seorang pakar sehingga dengan sistem tersebut pengguna dapat membuat sebuah keputusan atau menentukan kebijakan layaknya seorang pakar.



Gambar 2.1 Konsep Dasar Sistem Pakar

Dalam konsep sistem pakar tersebut, user atau pengguna menyampaikan fakta atau informasi ke dalam sistem pakar, yang selanjutnya fakta dan infromasi tersebut akan disimpan ke knowledge-base dan diolah oleh mesin inferensi, sehingga sistem dapat memberikan timbal balik kepada user berupa keahlian atau jawaban berdasarkan pengetahuan yang disampaikan sebelumnya.

Ciri-ciri sistem pakar menurut (Andriani, 2017) adalah sebagai berikut.

1. Memiliki dan memberikan informasi yang andal.
2. Mudah untuk dimodifikasi.
3. Terbatas pada domain keahlian tertentu.
4. Dapat memberikan penalaran untuk data-data yang sifatnya tidak pasti.
5. Sistem berdasarkan pada kaidah/*rule* tertentu.
6. Memiliki kemampuan untuk belajar beradaptasi.

7. Keluarannya bersifat anjuran.

Menurut Andriani (2017:14) sistem pakar mempunyai komponen utama pada strukturnya, antara lain sebagai berikut.

1. Basis pengetahuan (*Knowladge Base*)

   Inti dari suatu sistem pakar adalah basis pengetahuan yang merupakan representasi pengetahuan yang dimiliki oleh seorang pakar yang tersusun oleh atas fakta dan kaidah. Basis pengetahuan bisa kita dapatkan langsung dari seorang pakar maupun dari data histori yang berisi data-data pengetahuan dari seorang pakar.

2. Mesin Inferensi (*Inference Engine*)

   Otak dari sebuah sistem pakar adalah mesin inferensi yang berfungsi untuk memandu proses penalaran terhadap suatu kondisi berdasarkan pada basis pengetahuan yang tersedia. Terdapat dua penalaran yang dapat dilakukan dalam melakukan inferensi, yaitu:

   a. *Forward Chaining*

      Merupakan cara penalaran dengan memulai dari fakta terlebih dahulu untuk menguji kebenaran hipotesis atau mencocokan fakta atau pernyataan dimulai dari bagian sebelah kiri dulu (*IF* dulu). *Forward Chaining* merupakan grup dari multiple inferensi yang melakukan pencarian dari suatu masalah kepada solusinya. Jika klausa premis sesuai dengan situasi (bernilai *TRUE*), maka proses akan meng-*assert* konklusi.

   b. *Backward Chaining*

      Merupakan cara penalaran dengan memulai dari hipotesis (ekspektasi apa yang diinginkan terjadi) terlebih dahulu, dan untuk menguji kebenaran hipotesis tersebut harus dicari fakta-fakta yang ada dalam basis pengetahuan. *Backward Chaining* juga merupakan penalaran dengan mencocokan fakta atau pernyataan yang dimulai dari bagian sebelah kanan (*THEN* dulu).

3. Basis Data (*Database*)

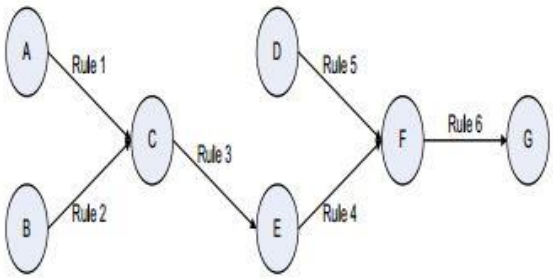
   Merupakan kumpulan data yang terdiri dari semua fakta yang diperlukan, dimana fakta-fakta tersebut digunakan untuk memenuhi kondisi dari kaidah-kaidah dalam sistem.

4. Antarmuka Pemakai (*User Interface*)

   Merupakan fasilitas yang dapat digunakan sebagai perantara komunikasi antara pemakai dengan komputer dalam menggunakan sistem pakar. Antarmukan ini memudahkan pengguna sistem pakar yang bukan merupakan seorang pakar dapat bekerja dan bertindak atau membuat keputusan layaknya seorang pakar.

2.1.2 *Forward Chaining*

Arifin dalam Extise P, (2016: 55) menjelaskan bahwa *forward chaining* adalah metode pelacakan yang diawali dengan informasi atau fakta dan proses mencocokkan dengan kaidah berlanjut terus hingga menemukan kesimpulan.



Gambar 2.2 *Forward Chaining*

Keterangan:

A, B … F = Kondisi atau gejala

G = Hasil diagnosis

Rule = Aturan

Contoh:

RULE 1:

IF CPU Mati AND Komputer sering restart AND Kipas power supply tidak berputar THEN Power suplay CPU bermasalah.

RULE 2:

IF CPU Mati AND Komputer sering Hang AND Komputer menjadi lambat AND Terdengar bunyi beep tidak terputus AND Terdengar bunyi beep panjang berkali-kali THEN Memory (RAM) bermasalah.

2.1.3 Diagnosa

Menurut Thorndike dan Hagen dalam Marbun (2018: 130), diagnosa dapat diartikan sebagai upaya atau juga proses dalam menemukan kelemahan atau penyakit (*weakness and disease*) apa yang dialami seseorang melalui pengujian mengenai gejala-gejalanya secara seksama.

Dari penjelasan di atas dapat disimpulkan bahwa diagnosa adalah suatu cara untuk menganalisis suatu jenis penyakit atau kerusakan berdasarkan pada gejala-gejalanya.

2.2 Teori Khusus

2.2.1 *Unified Modelling Language* (UML)

Menurut Rosa dan Shalahuddin (2018:137), "UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung".

Pada perkembangan teknologi perangkat lunak, diperlukan adanya bahasa yang digunakan untuk memodelkan perangkat lunak yang akan dibuat dan perlu adanya standarisasi agar orang di berbagai negara dapat mengerti pemodelan perangkat lunak. UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak.

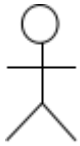
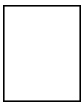
UML hanya berfungsi untuk melakukan pemodelan. Sehingga penggunaan UML tidak terbatas pada metodologi tertentu, meskipun pada kenyataannya UML paling banyak digunakan untuk metodologi berorientasi objek.

2.2.2 *Use Case Diagram*

Menurut Rosa dan Shalahuddin (2018:155), *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 itu.

Berikut adalah simbol-simbol yang ada pada diagram *use case*:

Tabel 2.1 Simbol *Use Case Diagram*

No	Gambar	Nama	Keterangan
1.		Aktor/ Actor	Aktor menggambarkan pengguna sistem, dapat berupa manusia atau sistem *terotomatisasi* lain yang berinteraksi dengan sistem lain untuk berbagi, mengirim, dan menerima informasi.
2.		Use Case	Simbol ini menggambarkan interaksi antara actor dengan software apliasi tersebut.
3.		System Boundary	Menggambarkan batasan anatara sistem dengan actor.
4.		Asosiansi/ Asosiation	Menggambarkan hubungan antar aktor dan *use case*.

(Sumber : Indrajani, 2015:46)

2.2.3 *Activity Diagram* (Diagram Aktivitas)

Menurut Rosa dan Shalahuddin (2018:161), *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Berikut adalah simbol-simbol yang ada pada diagram aktivitas:

Tabel 2.2 Simbol Diagram Aktivitas

No	Simbol	Nama	Deskripsi
1.		Status awal	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
2.		Aktivitas	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
3.		Percabangan / *decision*	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu
4.		Penggabungan / *join*	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu
5.		Status akhir	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.
6.	 Atau 	*Swimlane*	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

(Sumber : Rosa dan Shalahuddin, 2018:162)

2.2.4 *Class Diagram* (Diagram Kelas)

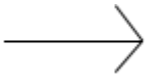
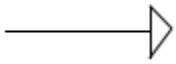
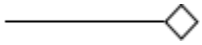
Menurut Rosa dan Shalahuddin (2018:141), diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan

dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan *method* atau operasi. Berikut penjelasan atribut dan operasi:

1. Atribut merupakan variable-variabel yang dimiliki oleh suatu kelas.
2. Operasi atau *method* adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

Berikut adalah simbol-simbol yang ada pada diagram kelas:

Tabel 2.3 Simbol Diagram Kelas

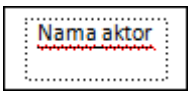
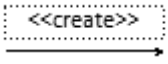
No	Simbol	Nama	Keterangan
1.	**Class** Attribute Operation	Kelas	Kelas pada struktur sistem
2.		Antarmuka/ *Interface*	Sama dengan konsep *interface* dalam pemrograman berorientasi obyek
3.		Asosiasi/ *Association*	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*
4.		Asosiasi berarah/*directed*	Asosiasi antarkelas dengan makna kelas yang satu digunakan oleh kelas lain, asosiasi biasanya juga disertain dengan *multiplicity*
5.		Generalisasi	Asosiasi antar kelas dengan makna generelisasi spesialisasi (umum – khusus)
6.		Kebergantungan/ *dependency*	Relasi antar kelas dengan makna kebergantungan antarkelas
7.		Agregasi / *aggregation*	Relasi antar kelas dengan makna semua bagian (*whole-part*)

(Sumber : Rosa dan Shalahuddin, 2018:146)

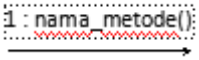
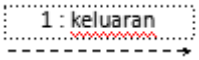
2.2.5 *Sequence Diagram* (Diagram Sekuen)

Menurut Rosa dan Shalahuddin (2018:165), *sequence diagram* menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek tersebut. Membuat diagram sekuen juga dibutuhkan untuk melihat skenario yang ada pada *use case*. Simbol-simbol yang digunakan dalam *sequence diagram*, yaitu :

Tabel 2.4 Simbol Diagram Sekuen

No	Simbol	Nama	Keterangan
1.		Garis hidup / *lifeline*	Menyatakan kehidupan suatu objek
2.	 Atau 	Aktor	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang, biasanya dinyatakan menggunakan kata benda di awal frase nama aktor
3.		Waktu aktif	Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan di dalamnya
4.		Pesan tipe *create*	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat

Lanjutan **Tabel 2.4** Simbol Diagram Sekuen

No	Simbol	Nama	Keterangan
5.		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
6.		Pesan tipe *send*	Menyatakan bahwa suatu objek mengirimkan data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi
7.		Pesan tipe keluaran	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian

(Sumber : Rosa dan Shalahuddin, 2018:165)