BAB II

TINJAUAN PUSTAKA

II.1 Penelitian Terkait

Tiara Eka Putri, Vol. 4 No. 1 Maret 2016, ISSN: 2303-0755. Implementasi Metode CBR (Case Based Reasoning) Dalam Pemilihan Pestisida Terhadap Hama Padi Sawah Menggunakan Algoritma K-Nearest Neighbor (KNN) (Studi Kasus Kabupaten Seluma). Hasil jurnal ialah penerapan metode Case Based Reasoning dalam mendiagnosa hama tanaman padi dengan menggunakan bahasa pemrograman Visual Basic .Net. Perbedaan pada penelitian ini terletak pada bahasa pemrograman berbasis web.

Rabiah Adawiyah, Vol. 1 No. 1 Juni 2017, ISSN : 2622-1659. Case Based Reasoning Untuk Diagnosis Penyakit Demam Berdarah. Hasil jurnal ialah penerapan metode Case Based Reasoning dalam mendiagnosa penyakit demam berdarah dengan menggunakan pemodelan *Unified Modelling Language* (UML). Perbedaan pada penelitian ini terletak pada kasus yang diteliti yaitu mengenai kecanduan game online.

Fhatiah Adiba, dkk, Vol. 6 No. 1 Februari 2020, ISSN : 2597-4963. Diagnosa Penyakit Kulit Menggunakan Case Based-Reasoning dan Self Organizing Maps. Hasil jurnal ialah penerapan metode Case Based Reasoning dalam mendiagnosa penyakit kulit dengan menggunakan bahasa pemrograman Visual Basic .Net. Perbedaan pada penelitian ini terletak pada bahasa pemrograman berbasis web.

Minarni, dkk, Vol. 8 No. 1 Februari 2020, ISSN : 2745-7265. Penerapan Metode Case Based-Reasoning Pada Sistem Pakar Diagnosa Penyakit Tanaman Pangan. Hasil jurnal ialah penerapan metode Case Based Reasoning dalam mendiagnosa penyakit pada tanaman pangan dengan menggunakan bahasa pemrograman berbasis web. Perbedaan dengan penelitian yang penulis lakukan yaitu pada jenis penyakit yang diteliti.

Irvan Muzakkir, Vol. 12 No. 1 April 2020, ISSN : 2548-7779. Case Based Reasoning Untuk Sistem Pakar Diagnosa Penyakit Sapi. Hasil jurnal ialah penerapan metode Case Based Reasoning dalam mendiagnosa penyakit pada sapi ternak. Perbedaan dengan penelitian yang penulis lakukan terletak pada jenis pemodelan sistem yang digunakan untuk merancang aplikasi.

II.2. Sistem

Kata Sistem berasal dari bahasa yunani (*systema*) dan bahasa latin (*systema*) adalah suatu kesatuan yang terdiri dari komponen atau elemen yang dihubungkan bersama untuk memudahkan aliran informasi, materi atau energi. Definisi sistem berkembang sesuai dengan konteks dimana pengertian sistem itu digunakan. Sistem adalah kumpulan komponen atau subsistem yang saling terkait dan bekerja sama untuk mencapai suatu tujuan (Rini Asmara ; 2016).

Sistem merupakan elemen-elemen yang saling berinteraksi untuk mencapai tujuan tertentu. Suatu sistem terdiri dari beberapa sub sistem yang saling berhubungan untuk membentuk suatu kesatuan sebagai sasaran dari sistem tersebut dapat tercapai.

Jadi pengertian sistem secara umum adalah jaringan kerja sama bagian-bagian atau unsur-unsur yang saling berhubungan guna mencapai tujuan yang diinginkan.

II.3. Sistem Pakar

Sistem pakar merupakan salah satu bidang kecerdasan buatan (*Artificial Intelligence*). Definisi sistem pakar itu sendiri adalah sebuah program komputer yang dirancang untuk mengambil keputusan seperti keputusan yang diambil oleh seorang pakar, dimana sistem pakar menggunakan pengetahuan (*knowledge)*. Fakta dan teknik berfikir dalam menyelesaikan masalah-masalah yang biasanya hanya dapat diselesaikan oleh seorang pakar atau ahli dari bidang yang bersangkutan (Ni Wayan ; 2018).

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.

Knowledge dalam sistem pakar mungkin saja seorang ahli, atau *knowledge* yang umumnya terdapat dalam buku, majalah dan orang yang mempunyai pengetahuan tentang suatu bidang. Istilah sistem pakar, sistem *knowledge-base*, atau sistem pakar *knowledge-base*, sering digunakan dengan arti yang sama. Kebanyakan orang menggunakan istilah sistem pakar karena lebih singkat, bahkan walau belum benar-benar pakar, hanya menggunakan *knowledge* secara umum (Harvei Desmon Hutahaean ; 2016).

Tabel II.1 Perbandingan Kemampuan Sistem Pakar dan Seorang Pakar

Faktor	Pakar	Sistem Pakar
Time availability	Hari Kerja	Setiap Saat
Geografis	Lokal/tertentu	Dimana saja
Keamanan	Tidak tergantikan	Dapat diganti
Perishable/dapat habis	Ya	Tidak
Performansi	*Variable*	Konsisten
Kecepatan	*Variable*	Konsisten
Biaya	Tinggi	Terjangkau

II.3.1 Ciri-Ciri Sistem Pakar

Ciri-ciri sistem pakar terdiri dari 8 bagian seperti dijelaskan berikut di bawah ini (Anggri Sartika Wiguna ; 2017):

1. Terbatas pada domain keahlian tertentu.
2. Dapat memberikan penalaran untuk data yang tidak pasti.
3. Dapat mengemukakan rangkaian alasan-alasan yang diberikannya dengan cara yang dapat dipahami.
4. Berdasarkan pada kaidah atau ketentuan atau *rule* tertentu.
5. Dirancang dapat untuk dapat dikembangkan secara bertahap.
6. pengetahuan dan mekansme penalaran (*interfance)* jelas terpisah.
7. keluarnya bersifat anjuran.
8. Sistem dapat mengaktifkan kaidah secara searah yang sesuai dituntun oleh dialog dengan *user*.

II.3.2 Bentuk Sistem Pakar

Ada 4 (empat) bentuk sistem pakar, yaitu seperti yang dijelaskan di bawah ini (Gusti Ayu ; 2017):

a. Berdiri sendiri.

   Sistem pakar jenis ini merupakan *software* yang berdiri sendiri tidak tergabung dengan *software* yang lainnya.

b. Tergabung.

   Sistem pakar jenis ini merupakan bagian program yang terkandung di dalam suatu algoritma atau merupakan program dimana di dalamnya memanggil algoritma sub rutin lain.

c. Menghubungkan ke *software* lain.

   Bentuk ini biasanya merupakan sistem pakar yang menghubungkan ke suatu paket program tertentu, misalnya dengan DBMS.

d. Sistem mengabdi.

   Sistem pakar ini merupakan bagian dari komputer khusus yang dihubungkan dengan suatu fungsi tertentu.

II.3.3 Komponen Sistem Pakar

Komponen utama pada struktur sistem pakar meliputi Basis Pengetahuan/*Knowledge Base,* Mesin Inferensi/*Inference Engin, Working Memory,* dan Antarmuka Pemakai/*User Interface.* Komponen-komponen dalam sistem pakar yaitu (Kresna Ramanda ; 2017) :

1. Antarmuka Pengguna

   Antarmuka pengguna (*user interface)* merupakan mekanisme yang digunakan oleh pengguna dan sistem pakar untuk berkomunikasi. Antar muka menerima informasi dari pemakai dan mengubahnya kedalam bentuk yang dapat diterima oleh sistem dan sebaliknya antarmuka juga menerima informasi dari sistem dan menyajikannya kedalam bentuk yang dimengerti oleh pemakai. Jadi pada bagian ini terjadi dialog antara program dan pemakai.

2. Basis Pengetahuan

   Basis pengetahuan mengandung pengetahuan untuk pemahaman, formulasi, dan penyelesaian masalah. Komponen sistem pakar ini disusun atas dua elemen dasar yaitu fakta dan aturan. Fakta merupakan informasi tentang obyek dalam area permasalahan tertentu, Aturan merupakan informasi tentang cara memperoleh fakta baru dari fakta yang telah diketahui.

3. Akuisisi Pengetahuan

   Akuisisi pengetahuan (*knowledge acquisition)* adalah akumulasi, transfer dan transformasi keahlian dalam menyelesaikan masalah dari sumber pengetahuan kedalam program komputer. Pengetahuannya diperoleh dari pakar yang dilengkapi dari buku, basis data, laporan penelitian dan pengalaman si pemakai.

4. Mesin Inferensi

   Mesin inferensi merupakan otak dari sebuah sistem pakar dan dikenal juga dengan sebutan struktur kontrol (*control structure*) atau *rule interpreter* (dalam sistem pakar berbasis kaidah).

II.3.4 Ruang Lingkup Sistem Pakar

Terdapat tiga orang yang terlibat dalam lingkungan sistem pakar, yaitu sebagai berikut:

1. Pakar

   Pakar adalah orang yang memiliki pengetahuan khusus, pendapat, pengalaman dan metode, serta kemampuan untuk mengaplikasikan keahliannya tersebut guna menyelesaikan masalah.

2. *Knowledge engineer* (perekayasa sistem)

   Knowledge engineer adalah orang yang membantu pakar dalam menyusun area permasalahan dengan menginterpretasikan dan mengintegrasikan jawaban-jawaban pakar atas pertanyaan yang diajukan, menggambarkan analogi, mengajukan *counter example* dan menerangkan kesulitan konseptual.

3. Pemakai

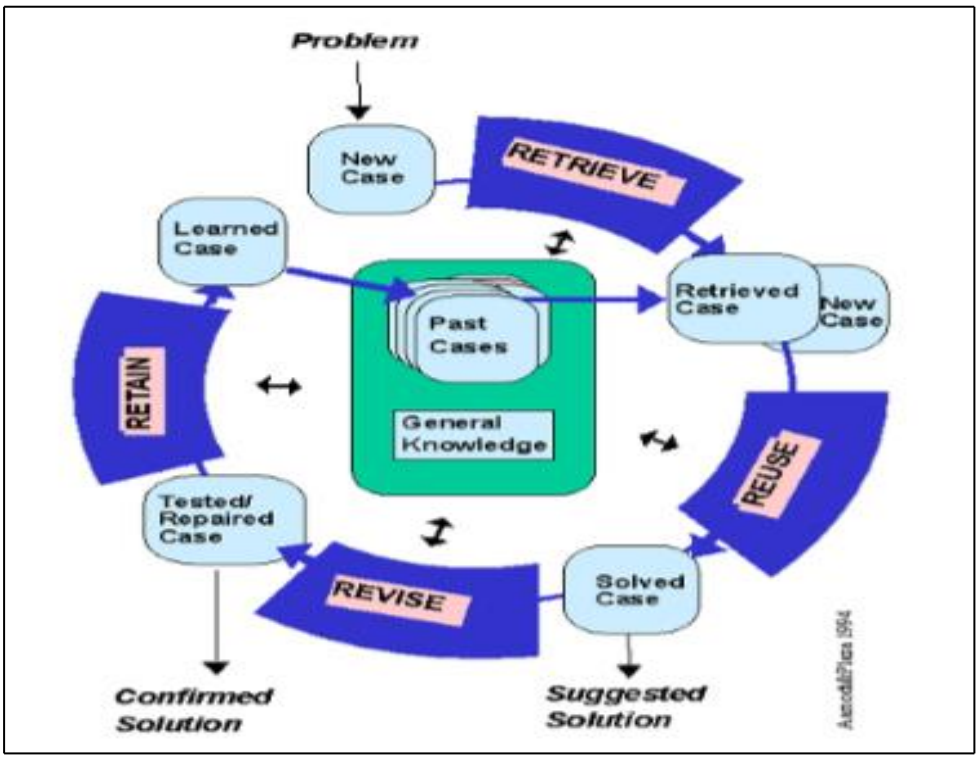
   Sistem Pakar memiliki beberapa pemakai, yaitu : Pemakai bukan pakar, pelajar, pembangun sistem pakar yang ingin meningkatkan dan menambah basis pengetahuan dan pakar.

II.4. Metode Case Based Reasoning

Case Based Reasoning (CBR) merupakan salah satu metode pemecahan masalah yang dalam mencari solusi dari suatu kasus yang baru, sistem akan melakukan pencarian terhadap solusi dari kasus lama yang memiliki permasalahan yang sama dan sudah pernah terjadi sebelumnya (Tiara Eka Putri ; 2018).

Metode CBR dikembangkan oleh Roger Schank dan rekannya di Universitas Yale pada awal tahun 1980. Terdapat dua prinsip dasar pada metode CBR, prinsip pertama adalah setiap permasalahan yang sama akan memiliki solusi yang sama pula. Oleh karena itu, solusi dari permasalahan yang sudah pernah terjadi dapat digunakan kembali untuk memecahkan masalah baru dengan permasalahan yang sama dengan masalah yang lama. Prinsip kedua adalah setiap permasalahan dapat terjadi berulang kali. Oleh karena itu, terdapat kemungkinan bahwa masalah yang akan muncul di masa yang akan datang memiliki kesamaan dengan masalah yang pernah terjadi sebelumnya. *Case Base Reasoning* (CBR) telah menjadi teknik yang sukses untuk sistem berbasis pengetahuan dalam banyak domain. *Case-Based Reasoning* (CBR) berarti menggunakan pengalaman sebelumnya dalam kasus yang mirip untuk memahami dan memecahkan permasalahan baru yang akan terjadi. Ide dasar dari *Case-Based Reasoning* adalah asumsi bahwa permasalahan yang serupa mempunyai solusi serupa. Meskipun asumsi ini tidaklah selalu benar, hal ini tergantung banyaknya domain praktis.

Terdapat empat proses yang terjadi pada metode CBR dalam menyelesaikan masalah, yaitu (lihat gambar II.1) :



Gambar II.1 *Case Based Reasoning*

Berikut keterangan mengenai gambar diatas :

1. *Retrieve*

   Pada proses ini sistem akan melakukan identifikasi parameter pencocokan yang dapat dijadikan sebagai acuan lalu melakukan pencarian kasus lama yang memiliki kesamaan dengan kasus baru. Selanjutnya sistem akan melakukan pencocokan parameter antara kasus baru dengan kasus lama dan memilih kasus yang memiliki tingkat kecocokan tertinggi.

2. *Reuse*

   Pada proses ini sistem akan menggunakan kembali informasi yang berasal dari kasus sebelumnya atau sistem akan melakukan adaptasi terlebih dahulu untuk memecahkan masalah pada kasus yang baru.

3. *Revise*

   Pada proses ini sistem akan meninjau kembali solusi yang telah didapatkan dari kasus yang lama apakah solusi tersebut akan diterapkan pada kasus yang baru atau solusi tersebut perlu diperbaiki terlebih dahulu.

4. *Retain*

   Pada proses ini apabila ternyata ditemukan solusi baru yang lebih baik dari solusi yang telah ada sebelumnya maka solusi baru tersebut akan diberi indeks dan disimpan untuk kemudian digunakan kembali pada kasus serupa pada masa yang akan datang.

Penerapan CBR banyak dilakukan untuk memodelkan domain permasalahan yang bisa merupakan kombinasi:

1. Sulit dalam melakukan elisitasi pengetahuan;
2. Sulit menerapkan aturan *if-then*, karena sangat bergantung pada tingkat kepakaran yang spesifik dan berkembang dalam waktu lama;
3. Jika penerapan aturan dapat dilakukan, maka aturan dapat bertahan lamadan besaran data akan dapat berkembang pesat;
4. Ada kecenderungan sulit dalam pengelolaan data karena perkembangan yang pesat;
5. Banyak memiliki pengetahuan historis.

Perhitungan untuk mendapatkan kesamaan antar kasus, sangat penting pada metode CBR terutama dalam proses *retrieval*. Efektifitas dari pengukuran kesamaan tergantung dari seberapa banyak kasus yang ditelah, membantu dalam memecahkan masalah pada kasus yang baru. Ada dua pendekatan yang

digunakan untuk melakukan perhitungan kesamaan pada kasus pendekatan jarak dan indeks.

Kedekatan antara dua kasus dalam metode *case based reasoning* dihitung dengan menggunakan rumus :

$$Sim\ (A, B) = \frac{1}{P}\sum_{i=1}^{P} Sim_i(a, b) \quad \text{..... (2.1)}$$

dimana :

Sim (A, B) = kemiripan kasus ke-a dan ke-b.

Semakin tinggi nilai yang didapat maka kedekatannya pun akan semakin tinggi. Sebaliknya semakin rendah nilai yang didapat maka, kedekatannya pun semakin rendah. Pada aplikasi ini ditentukan sebuah batas, yaitu jika nilai yang didapat >= nilai yang telah ditentukan maka kasus lama tersebut dapat langsung digunakan kembali untuk menyelesaikan kasus yang baru.

Proses mencari kedekatan antara masalah baru dengan kasus lama dapat menggunakan berbagai macam metode, dimana metode ini mempengaruhi keberhasilan dari kerja CBR dalam menentukan kasus lama yang paling mirip dengan masalah baru (*target case*). Salah satu metode yang dapat digunakan dalam penelitian ini untuk menghitung kemiripan (*similarity*) adalah *Nearest Neighbor* (Rabiah Adawiyah ; 2017).

II.5. *UML* (*Unified Modeling Language*)

UML (*Unified Modeling Language*) adalah metode pemodelan secara visual sebagai sarana untuk merancang dan membuat *software* berorientasi

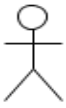
objek. Karena UML ini merupakan bahasa visual untuk pemodelan bahasa berorientasi objek. UML adalah salah satu *tool*/model untuk merancang pengembangan *software* yang berbasis *object oriented*.

UML adalah keluarga notasi grafis yang didukung meta-model tunggal, yang membantu pendeskripsian dan desain sistem perangkat lunak, khususnya sistem yang dibangun menggunakan pemrograman beroreintasi objek. UML tidak hanya merupakan sebuah bahasa pemograman visual saja, namun juga dapat secara langsung dihubungkan ke berbagai bahasa pemograman.

II.5.1. *Use Case Diagram*

Use case adalah rangkaian atau uraian sekelompok yang saling terkait dan membentuk sistem secara teratur yang dilakukan oleh sebuah aktor atau lebih. Diagram use case digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut (Andi Irmayana ; 2016).

Tabel II.2. Simbol *Use Case Diagram*

No	Gambar	Nama	Keterangan
1		*Actor*	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor

2		*Association*	Komunikasi antara aktor dan use case yang berpartisipasi pada use case atau use case memiliki interaksi dengan aktor
3		*Use case*	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama use case
4		*Generalization*	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).

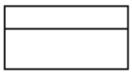
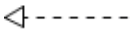
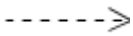
(Sumber : Andi Irmayana, 2016)

II.5.2. *Class Diagram* (Diagram Kelas)

Class diagram menggambarkan struktur sistem dari segi pendefenisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi (Winda Aprianti ; 2016).

Class Diagram memberikan pandangan secara luas dari suatu sistem dengan menunjukkan kelas-kelasnya dan hubungan mereka. Diagram *Class* bersifat statis, menggambarkan hubungan apa yang terjadi bukan apa yang terjadi jika mereka berhubungan. Simbol-simbol *Class Diagram* sebagai berikut ini.

Tabel II.3. 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

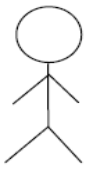
(Sumber : Winda Aprianti, 2016)

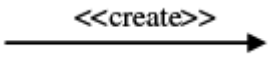
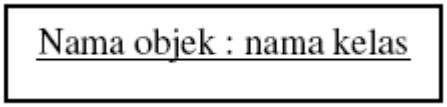
II.5.3. *Sequence Diagram*

Sequence diagram menggambarkan kelakuan objek pada *Use Case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek (Winda Aprianti ; 2016).

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 itu. Membuat diagram sekuen juga dibutuhkan untuk melihat skenario yang ada pada *use case*. Banyaknya diagram sekuen yang harus digambar adalah minimal sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup dalam diagram sekuen sehingga semakin banyak *use case* yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak.

Tabel II.4. *Sequence Diagram*

No	Simbol	Deskripsi
1	Actor 	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat
2	Garis Hidup/*Lifeline* ----------	Menyatakan kehidupan suatu objek

3	Pesan Tipe *Create* 	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat
4	Objek 	Menyatakan objek yang berinteraksi pesan
5	Waktu Aktif 	Menyatakan objek dalam keadaan aktif dan berinteraksi
6	Pesan Tipe *Send* 	Menyatakan bahwa suatu objek Mengirimkan data atau masukkan informasi ke objek lainnya, arah panah mengarah pada objek yang dikirim

(Sumber : Winda Aprianti, 2016)

II.5.4. *Activity Diagram*

Activity Diagram adalah teknik untuk mendeskripsikan logika procedural, proses bisnis dan aliran kerja dalam banyak kasus, diagram akivitas mempunyai peran seperti halnya diagram alur (*flowchart*), akan tetapi perbedaaannya dengan *flowchart*, adalah diagram aktivitas bisa mendukung perilaku parallel sedangkan *flowchart* tidak bisa. *Activity Diagram*

menggambarkan *workflow* (aliran kerja) dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak

Tabel II.5. Simbol *Activity Diagram*

No	Gambar	Nama	Keterangan
1		*Activity*	Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas
2		*Start*	Mendefinisikan suatu tindakan sebelum aktivitas dimasukkan
3		*Actifity Final Node*	Menandakan bahwa suatu tindakan atau aktivitas telah selesai
4		*Fork/Join*	Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal
5		*Decission*	Untuk menggambarkan *behaviour* pada kondisi tertentu.
6		*Control Flow*	Mendeskripsikan kemana aliran kegiatan berlangsung

(Sumber : Winda Aprianti, 2016)

II.6. Basis Data (*Database*)

Basis data adalah suatu kumpulan data terhubung yang disimpan secara bersama-sama pada satu media, tanpa adanya suatu kerangkapan data, sehingga mudah untuk digunakan kembali, dapat digunakan oleh satu atau lebih program aplikasi secara optimal, data disimpan tanpa mengalami ketergantungan pada program yang akan menggunakannya, data disimpan sedemikian rupa sehingga apabila ada penambahan, pengambilan dan modifikasi data dapat dilakukan dengan mudah dan terkontrol.

Dari pengertian tersebut dapat disimpulkan bahwa Sistem Basis Data adalah Sistem yang terdiri atas kumpulan tabel/*file* yang saling berhubungan dalam sebuah basis data dan sekumpulan program berupa DBMS yang memungkinkan beberapa pemakai atau program lain untuk mengakses dan memanipulasi tabel-tabel tersebut.

Sistem basis data merupakan sistem penyusunan berkas data yang saling terpadu. Mempunyai komponen-komponen sebagai berikut:

1. *Database* (Basis Data)

   Adalah kumpulan *file-file* yang saling berhubungan atau berelasi sehingga membentuk suatu basis data.

2. *Software* (Perangkat Lunak)

   Adalah perangkat lunak yang digunakan dalam suatu sistem basis data. Pengelolaan basis data secara fisik tidak dapat dilakukan pemakai secara langsung, tetapi ditangani oleh sebuah perangkat lunak yang khusus yang disebut DBMS (*Database Management System*) yang akan menentukan

bagaimana data diorganisasikan, disimpan, diubah dan diambil kembali. Perangkat lunak yang termasuk dalam DBMS.

3. *Hardware* (Perangkat Keras)

Adalah perangkat keras dalam suatu sistem basis data, dimana mempunyai komponen-komponen utama berupa, Unit Pusat Pengolah (*Central Processing Unit* atau CPU), unit penyimpanan (*Storage Unit)*, keyboard, monitor, printer dan lain-lain.

4. *Brainware* (manusia)

Manusia merupakan elemen penting pada sisitem basis data. Pemakai ini terbagi atas empat kategori.

5. *Administrator Basis Data*

Yaitu tenaga ahli yang mempunyai tugas untuk mengawasi sistem basis data, merencanakan dan mengaturnya.

6. *Programmer*

Yaitu bertugas membuat program aplikasi yang diperlukan oleh pemakai akhir dengan menggunakan data yang terdapat dalam sistem basis data.

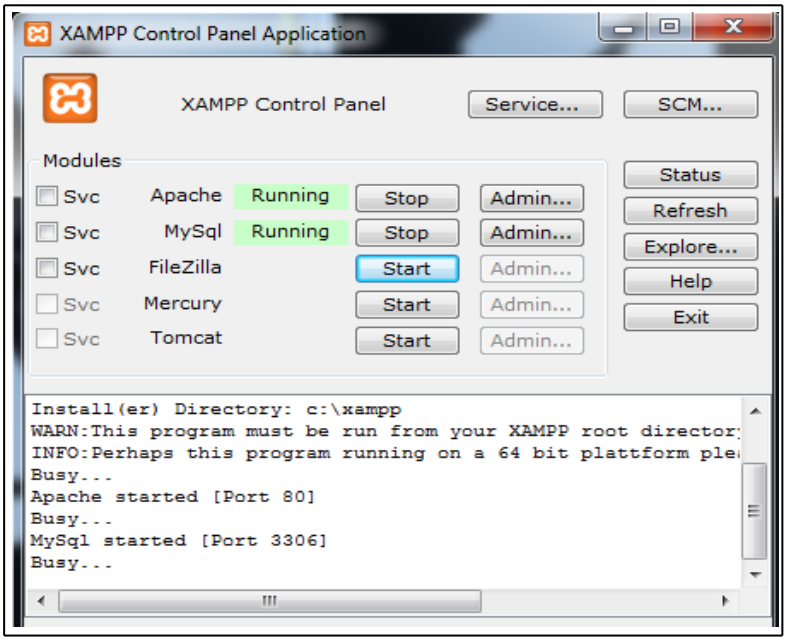
7. Pemakai Akhir

Yaitu tenaga ahli yang menggunakan data untuk mengambil suatu keputusan yang diperlukan dalam suatu instansi/perusahaan.

II.7 XAMPP

XAMPP merupakan salah satu paket *software web server* yang didalamnya telah terdapat *Apache*, MySQL, PHP, dan phpMyAdmin. Proses instalasi XAMPP sangat mudah, karena tidak perlu melakukan konfigurasi *Apache*, PHP dan MySQL secara manual, XAMPP melakukan instalasi dan konfigurasi secara otomatis.

XAMPP adalah perangkat lunak bebas, yang mendukung banyak sistem oprasi, merupakan kompilasi dari beberapa program. XAMPP merupakan tool yang menyediakan paket perangkat lunak ke dalam satu buah paket. Dengan menginstall XAMPP maka tidak perlu lagi melakukan instalasi dan konfigurasi *web server Apache*, PHP dan MySQL secara manual.



Gambar 2.2 Tampilan Control Panel XAMPP

II.8. MySQL

MySQL merupakan *database server* yang bersifat *multiuser* dan *multi-threaded*. SQL adalah bahasa *database* standar yang memudahkan penyimpanan, pengubahan dan akses informasi. Pada MySQL dikenal istilah *database* dan tabel. Tabel adalah sebuah struktur data dua dimensi yang terdiri dari baris-baris *record* dan kolom.

MySQL adalah salah satu jenis *database server* yang sangat terkenal, kepopulerannya disebabkan MySQL menggunakan *Structure Query Language* (SQL) sebagai dasar untuk mengakses basis datanya. Selain itu, MySQL bersifat *free* pada berbagai *platform* (kecuali pada Windows, yang bersifat *shareware* atau anda perlu membayar setelah melakukan evaluasi dan memutuskan untuk digunakan untuk keperluan produksi) atau tidak dicekal. MySQL termasuk jenis *Relational Database Management Sistem* (RDBMS). Itulah sebabnya istilah seperti tabel, baris dan kolom digunakan di dalam MySQL. Sebuah basis data mengandung satu atau sejumlah tabel, tabel terdiri atas sejumlah baris dan setiap baris mengandung satu atau beberapa kolom.

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (bahasa Inggris: *database management system*) atau DBMS yang *multithread, multi-user*, dengan sekitar 6 juta instalasi diseluruh dunia. *MySQL AB* membuat MySQL tersedia sebagai perangkat lunak gratis dibawah lisensi GNU *General Public License* (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL. MySQL merupakan bahasa pemrograman *open-source* yang

paling populer dan banyak digunakan di lingkungan Linux kepopuleran ini karena ditunjang oleh performansi *query* dari *database*-nya yang jarang bermasalah.

Kesimpulan : MySQL adalah singkatan dari *My Sequel* yang bisa didefinisikan sebuah *software* atau perangkat lunak dengan sistem manajemen basis data SQL atau DBMS yang *multithread* dan *multi-user* dengan jumlah instalasi sekitar 6 juta diseluruh dunia. Sederhananya MySQL merupakan sebuah perangkat lunak pada *Relational Database Mnagement Sytem* (RDMS) didalam sebuah manajemen *database* sebagai basis data.