BAB II

TINJAUAN PUSTAKA

II.1. Sistem

Secara sederhana, suatu sistem dapat diartikan sebagai suatu kumpulan atau himpunan dari unsur, komponen atau variabel yang terorganisir, saling berinteraksi, saling tergantung satu sama lain, dan terpadu. Teori sistem secara umum yang pertama kali diuraikan oleh Kannet Boulding, terutama menekan pentingnya perhatian terhadap setiap bagian yang membentuk sebuah sistem. Kecenderungan manusia yang dapat tugas memimpin suatu organisasi adalah terlalu memusatkan perhatian pada salah satu komponen saja dari sistem organisasi.(Siti Kholijah Ritonga, 2013 : 2)

II.2. Sistem Pakar

Sistem pakar adalah suatu sistem yang dirancang untuk dapat menirukan keahlian seorang pakar atau ahli dalam menjawab pertanyaan dan memecahkan suatu masalah. Tujuan utama sistem pakar adalah untuk memasyarakatkan atau memindahkan secara efektif pengetahuan dan pengalaman para pakar kepada mereka yang bukan pakar. Pemecahan masalah diberikan pada pemakai melalui dialog dengan mereka. Sistem pakar membantu seseorang yang bukan pakar/ahli dalam menjawab pertanyaan, menyelesaikan masalah dan mengambil keputusan yang biasanya dilakukan oleh seorang pakar. (Suwandy Asep Sandy, dkk, 2013)

II.2.1 Komponen Sistem Pakar

Sistem pakar terdiri dari tiga komponen pendukung, yaitu, Pangkalan Pengetahuan *(Knowledge Base)* yang berisi fakta-fakta, ide, interaksi suatu domain tertentu, Motor Inferensi *(Inference Engine)* yang bertugas menganalisa pengetahuan dan menarik kesimpulan berdasarkan pangkalan pengetahuan dan Anatarmuka Pemakai *(User Interface)* yang berfungsi sebagai media yang melakukan komunikasi dengan pemakai. (Suwandy Asep Sandy, dkk, 2013)

II.2.2 Representasi Pengetahuan

Pendekatan penyelesaian masalah dalam sistem pakar terdapat banyak metoda yang berbeda untuk merepresentasikan pengetahuan. Diantaranya terdapat beberapa metoda yang terbagi menjadi dua macam, pertama representasi pengetahuan yang bersifat deklaratif, seperti: logika, jaringan semantik, predikat kalkulus, *list*, *frame*, *script* dan kedua representasi pengetahuan yang bersifat prosedural, seperti: prosedur atau *subroutine* dan kaidah produksi. Sampai saat ini metoda paling banyak dipakai dalam pembuatan sistem pakar adalah jaringan semantik dan kaidah produksi. Representasi pengetahuan yang digunakan dalam skripsi ini adalah metoda pohon pelacakan yang merupakan salah satu bagian dari metode jaringan semantik dan metoda yang paling banyak dipakai adalah pohon pelacakan. Setiap pertanyaan yang diberikan dalam sistem pakar analisa kepribadian manusia bersumber dari Psikotes *The Instant Insight Inventory.* (Suwandy Asep Sandy, dkk, 2013)

II.2.3 Mekanisme Inferensi

Mekanisme inferensi adalah rangkaian prosedur yang digunakan untuk menguji pangkalan pengetahuan dengan cara yang sistematik pada saat menjawab pertanyaan, persoalan atau membuat ke putusan dalam satu domain yang telah ditentukan. Skema yang digunakan dalam mekanisme ini adalah skema deklaraif yaitu dengan menggunakan pohon

pelacakan yang merupakan bagian dari jaringan semantik dan skema prosedural dengan menggunakan kaidah produksi bentuk IF–THEN. Penggunaan skema ini karena sesuai dengan jenis pertanyaan dan cara penyelesaian masalah dalam skripsi ini. Dalam merepresentasikan pengetahuan akan digunakan metoda kaidah produksi karena metoda ini lebih praktis untuk penyelesaian masalah yang muncul dalam pembuatan sistem pakar. (Suwandy Asep Sandy, dkk, 2013)

II.2.4 Metoda Pelacakan

Operasi pelacakan dan pencocokan pola atau sering juga disebut sebagai penafsir kaidah atau kaidah interpreter memiliki dua macam pendekatan, yaitu *Backward Chaining* yaitu kaidah pelacakan dengan proses dari kesimpulan menuju sekumpulan data. dan *Forward Chaining* yaitu kaidah pelacakan dengan proses dari sekumpulan data menuju kesimpulan. Dalam skripsi ini digunakan pendekatan dengan menggunakan *Forward Chaining* yaitu karena persoalan berawal dari fakta-fakta dasar tipe kerpibadian. Metoda pelacakan dalam sistem pakar ada tiga macam yaitu *Bread-first Search* yaitu pelacakan dengan arah pelacakan pertama melebar menguji semua simpul yang setingkat, *Depth- first Search* yaitu pelacakan dengan arah pelacakan pertama ke bawah menguji satu simpul sampai tingkat terbawah dan *Best Search* yaitu gabungan dari kedua pelacakan di atas. Sistem pakar yang dirancang akan menggunakan *Depth-first Search* karena sesuai dengan jenis pertanyaan yang dipakai yaitu pertanyaan yang cukup dijawab ya atau tidak. (Suwandy Asep Sandy, dkk, 2013)

II.3. Burung *Lovebird*

Burung *lovebird* pada awalnya merupakan burung hiasan yang oleh sebagian orang dijadikan hewan peliharaan sebagai simbol kesetiaan. Namun saat ini burung *lovebird* tidak hanya dijadikan sebagai burung peliharaan yang dinikmati karena keindahan warna bulu,

bentuk tubuh dan perilaku yang lucu saja, tetapi juga dijadikan sebagai lahan bisnis yang memberikan keuntungan besar dan dimanfaatkan sebagai burung master beberapa burung berkicau serta dilombakan di dalam setiap kontes. (Faisal Nur Fauzi, 2013: 62)

II.4. Metode *Case Based Reasoning* (CBR)

Metode *Case Based Reasoning* (CBR) menggunakan pendekatan kecerdasan buatan (*Artificial Intelligent*) yang menitikberatkan pemecahan masalah dengan didasarkan pada *knowledege* dari kasus-kasus sebelumnya. Secara umum, metode ini terdiri dari 4 langkah, yaitu :

1. *Retrieve*

   Pada saat terjadi permasalahan baru, pertama tama sistem akan melakukan proses *Retrieve.* Proses ini akan melakukan dua langkah pemrosesan, yaitu pengenalan masalah dan pencarian persamaan masalah pada *database*.

2. *Reuse*

   Proses ini sistem akan menggunakan informasi permasalahan sebelumnya yang memiliki kesamaan untuk menyelesaikan permasalahan yang baru dan menggunakan kembali informasi dan pengetahuan dalam kasus tersebut untuk mengatasi masalah. Pada proses *Reuse* akan menyalin, menyeleksi, dan melengkapi informasi yang akan digunakan. Kriteria untuk pemilihan kasus adalah kasus yang memiliki kemiripan paling tinggi dengan kasus baru yang akan disarankan sebagai solusi. Walaupun demikian, setiap kasus baru belum tentu memiliki nilai kemiripan yang lumayan tinggi dengan basis kasus. Maka perlu diberikan kriteria kemiripan dengan menghitung nilai desimal dari setiap total atau nilai kemiripan.

3. *Revise* (meninjau ulang solusi yang diajukan).

Proses ini informasi tersebut akan dikalkulasi, dievaluasi, dan diperbaiki kembali untuk mengatasi kesalahan-kesalahan yang terjadi pada permasalahan baru.

4. *Retain*

Proses ini akan mengindeks, mengintegrasi, dan mengekstrak solusi yang baru. Selanjutnya, solusi baru itu akan disimpan ke dalam *knowledge base* untuk menyelesaikan permasalahan yang akan datang. Permasalahan yang akan diselesaikan adalah permasalahan yang memiliki kesamaan dengannya. Proses *Retrive* merupakan proses pencarian kemiripan kasus baru dengan kasus yang lama. Pencarian kemiripan antara kasus baru dengan kasus lama dilakukan dengan cara mencocokan gejala yang diinputkan oleh pengguna dengan gejala yang ada pada basis pengetahuan. Pada proses *retrieve* ini akan dilakukan pembobotan dengan menggunakan metode *Nearest Neighbour*. (Diki Andita Kusuma, dkk, 2014:2).

Adapun rumus untuk menghitung tingkat kemiripan (jarak) suatu kasus sebagai berikut :

$$Similarity\ (S, T) = \sum_{t=1}^{n} f\ (T_t, S_t) \times wt$$

Keterangan :

T = Kasus target/ baru.
S = Kasus/ lama/ pembanding.
n = Jumlah atribut dalam setiap kasus.
i = Atribut individu dari 1 sampai n.
f = Fungsi kemiripan untuk atribut I dalam kasus T dan S.
w = Bobot atribut i

II.5. Basis Data

Pada dasarnya basis data bukanlah sistem yang selalu terkait dengan komputer. Adapun beberapa penjelasan terkait dengan basis data adalah pengertian data, operasi dasar basis data, dan pengertian sistem informasi itu *sensistem management* basis data. Basis data terdiri dari 2 kata, yaitu basis dan data, basis dapat di artikan sebagai maskas atau

gudang tempat bersarang atau berkumpul. Sedangkan data adalah representasi fakta dunia nyata yang mewakili suatu objek seperti manusia (pegawai, siswa, pembeli dan lain-lain), barang hewan, peristiwa, konsep keadaan dan sebagainya yang direkam dalam bentuk angka, huruf, simbol, teks, gambar, bunyi, atau kombinasinya. (Syaifudin Ramadhani, 2014:2)

II.6. *MySQL*

MySQL adalah sebuah perangkat lunak sistem manajemen basis data *SQL* yang multithread, multi-user, dengan sekitar 6 juta instalasi di seluruh 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*. *Relational Database Management System (RDBMS*). *MySQL* adalah *Relational Database Management System (RDBMS)* yang didistribusikan secara gratis dibawah lisensi *GPL* (*General Public License).* Dimana setiap orang bebas untuk menggunakan *MySQL*, namun tidak boleh dijadikan produk turunan yang bersifat komersial. *MySQL* sebenarnya merupakan turunan salah satu konsep utama dalam database sejak lama, yaitu *SQL* (*Structured Query Language*). (Syaifudin Ramadhani, dkk, 2013:2)

II.7. PHP *(Hypertext* Preprocessor)

PHP *(Hypertext Preprocessor),* merupakan bahasa pemrograman pada sisi *server* yang memperbolehkan programmer menyisipkan perintah perangkat lunak *web server* (*Apache, IIS*, atau apapun) akan dieksekusi sebelum perintah itu dikirim oleh halaman ke *browser* yang me-*request*-nya, contohnya adalah bagaimana memungkinkannya memasukkan tanggal sekarang pada sebuah halaman web setiap kali tampilan tanggal dibutuhkan. Sesuai dengan fungsinya yang berjalan di sisi server maka PHP adalah bahasa pemrograman yang digunakan untuk membangun teknologi *web application.* PHP telah

menjadi bahasa *scripting* untuk keperluan umum yang pada awalnya hanya digunakan untuk pembangunan web yang menghasilkan halaman web dinamis. Untuk tujuan ini, kode PHP tertanam ke dalam dokumen sumber *HTML* dan diinterpretasikan oleh server web dengan modul PHP prosesor, yang menghasilkan dokumen halaman web. Sebagai bahasa pemrograman untuk tujuan umum, kode PHP diproses oleh aplikasi penerjemah dalam modus baris - baris perintah modus dan melakukan operasi yang diinginkan sesuai sistem operasi untuk menghasilkan keluaran program dichannel output standar. Hal ini juga dapat berfungsi sebagai aplikasi grafis. PHP tersedia sebagai prosesor untuk server web yang paling modern dan sebagai penerjemah mandiri pada sebagian besar *system* operasi dan komputer *platform.* **(**Herny Februariyanti,dkk, 2013:128)

II.8. *Unified Modeling Language* (UML)

Unified Modelling Language (UML) adalah sebuah bahasa yang telah menjadi standar dalam industri untuk visualisasi merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML dapat dibuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan class dan operation dalam konsep dasarnya, maka lebih cocok untuk penulisan piranti lunak dalam bahasa berorientasi objek seperti *C++*, *Java*, atau *VB. NET*. Diagram UML terdiri dari beberapa bagian sebagai berikut :

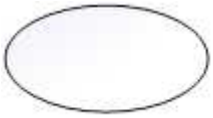
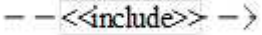
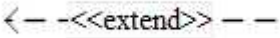
II.8.1. *Use Case* Diagram

Use Case Diagram adalah gambaran fungsionalitas dari suatu sistem, sehingga *customer* atau pengguna sistem paham dan mengerti mengenai kegunaan

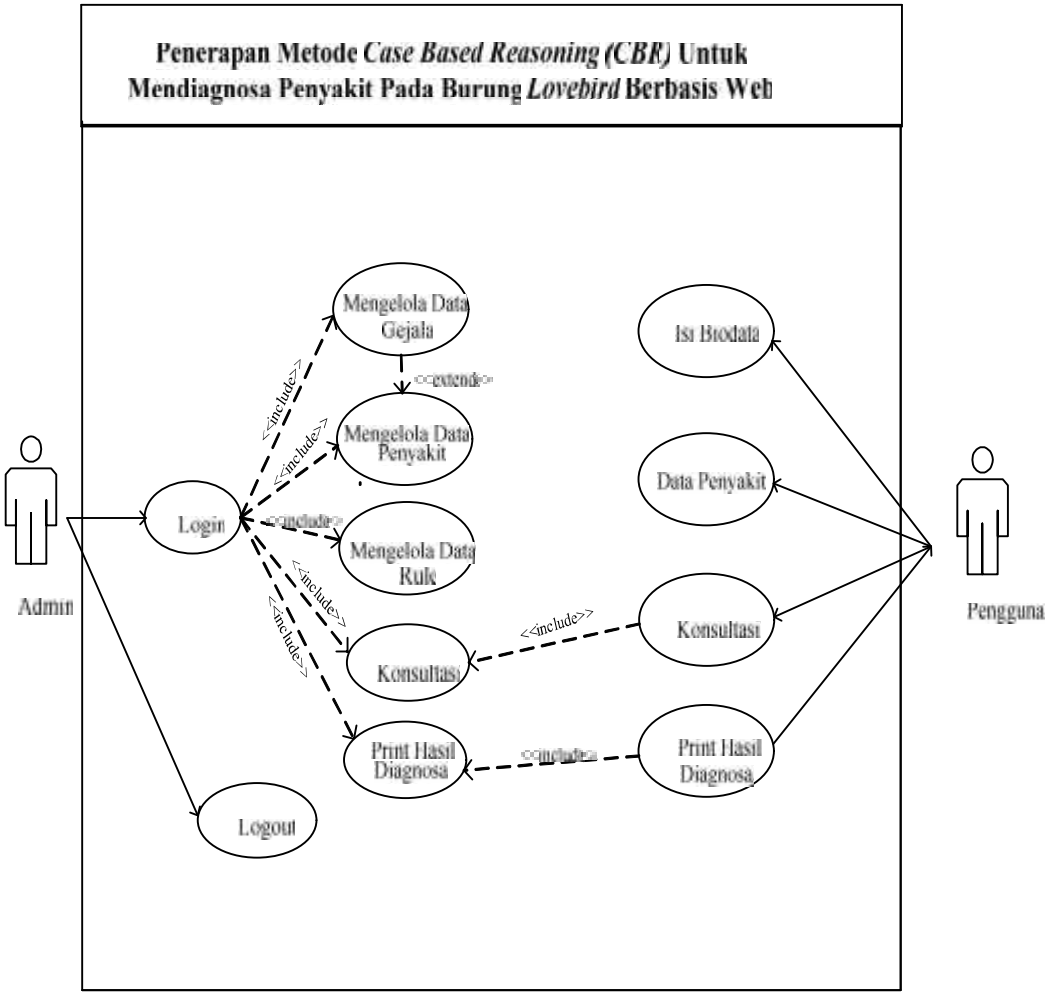
sistem yang akan dibangun (I Made Budi Adnyana, 2016 : 51). *Use case* diagram merupakan pemodelan untuk kelakuakn (*behavior*) sistem yang akan dibuat. *Use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut (Ade Hendini, 2016 : 108).

Simbol-simbol yang digunakan dalam *use case* diagram dapat dilihat pada tabel II.1.

Tabel II.1. Simbol *Use Case* Diagram

Gambar	Keterangan
	*Use Case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktor, yang dinyatakan dengan menggunakan kata kerja.
	*Actor* atau 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 kontrol 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 mengindikasikan data.
	Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengindikasikan 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 : Ade Hendini, 2016 : 108-109)



Gambar II.1. *Usecase Diagram* Penerapan Metode *Case Based Reasoning* (CBR) Untuk Mendiagnosa Penyakit Pada Burung *Lovebird* Berbasis Web

II.8.2. *Class* Diagram

Class diagram menggambarkan struktur dan deskripsi *class*, *package* dan *object* beserta hubungan satu sama lain seperti pewarisan, asosiasi, dan lain-lain (I Made Budi Adnyana, 2016 : 52).

Class diagram menggambarkan struktur sistem dari segi pendifinisian kelas-kelas yang akan di buat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi (Winda Aprianti dan Umi Maliha, 2016 : 22).

Simbol-simbol yang digunakan pada *class* diagram dapat dilihat pada tabel II.2 sebagai berikut.

Tabel II.2. Simbol *Class* Diagram

Gambar	Keterangan
	Kelas. Kelas pada struktur sistem
	Antarmuka/*Interface.* Sama dengan konsep *interface* dalam pemrograman berorientasi objek
	Asosiasi/*Association* Relasi antar kelas dengan makna umum, asosiasi biasanya disertai dengan *multiplicity*
	Asosiasi berarah/*Directed association.* Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya disertai dengan *multiplicity*
	Generalisasi. Relasi antar kelas dengan makna generalisasi-spesialisasi (umum-khusus)
	Kebergantungan/*Dependency.* Relasi antar kelas dengan makna kebergantungan antar kelas
	Agregasi/*Aggregation.* Relasi antar kelas dengan makna semua bagian.

(Sumber : Winda Aprianti dan Umi Maliha, 2016 : 22)

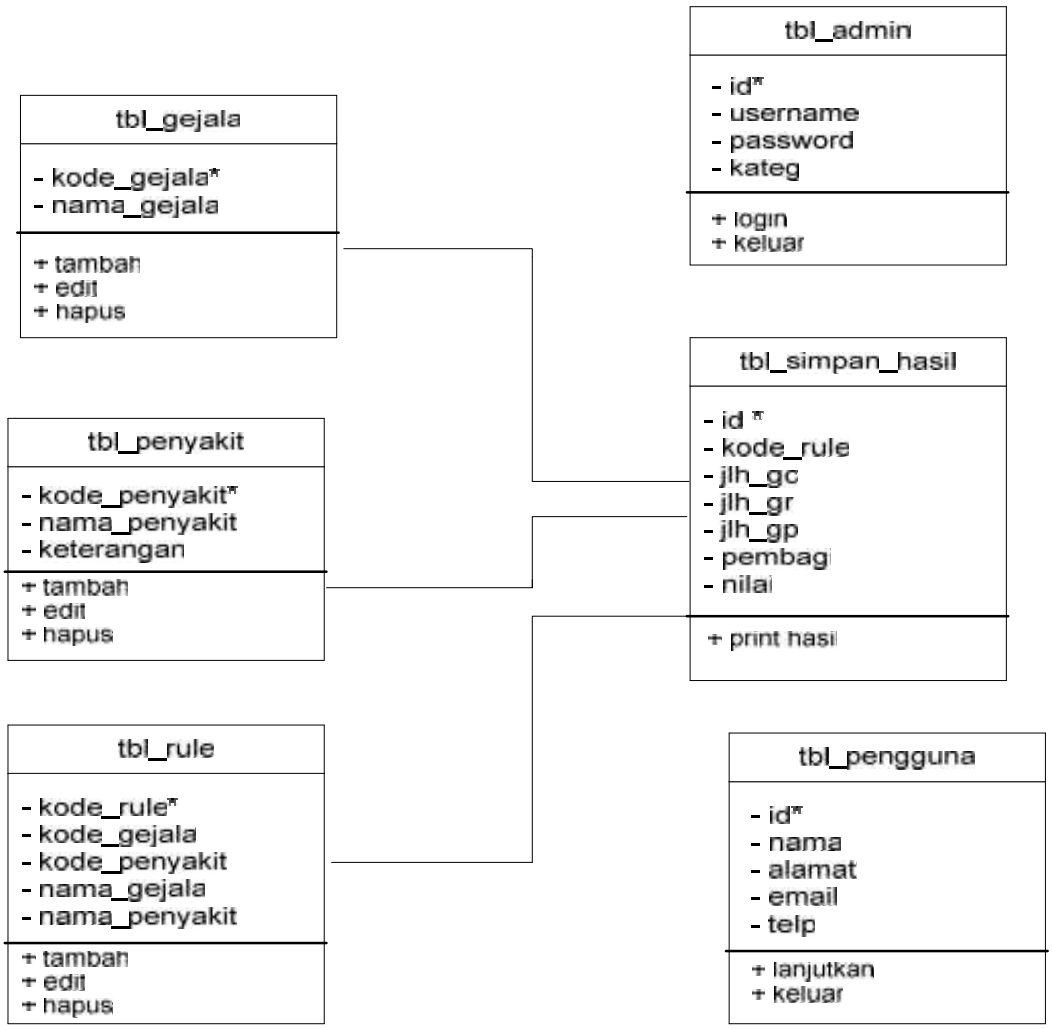
Hubungan antar kelas mempunyai keterangan yang disebut *multiplicity* atau *cardinality* yang dapat dilihat pada tabel II.3.

Tabel II.3. *Multiplicity Class* Diagram

Multiplicity	**Keterangan**
1	Satu dean 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 maksimal 4

(Sumber : Ade Hendini, 2016 : 111)



Gambar II.2. *Class Diagram* Penerapan Metode *Case Based Reasoning* (CBR) Untuk Mendiagnosa Penyakit Pada Burung *Lovebird* Berbasis Web.

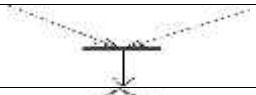
II.8.3. *Activity* Diagram

Activity diagram menggambarkan sebagai alur aktifitas dalam sistem yang sedang dirancang. Bagaimana masing-masing alur berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. Sebuah aktifitas dapat direalisasikan oleh satu *use case* atau lebih. Aktifitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan

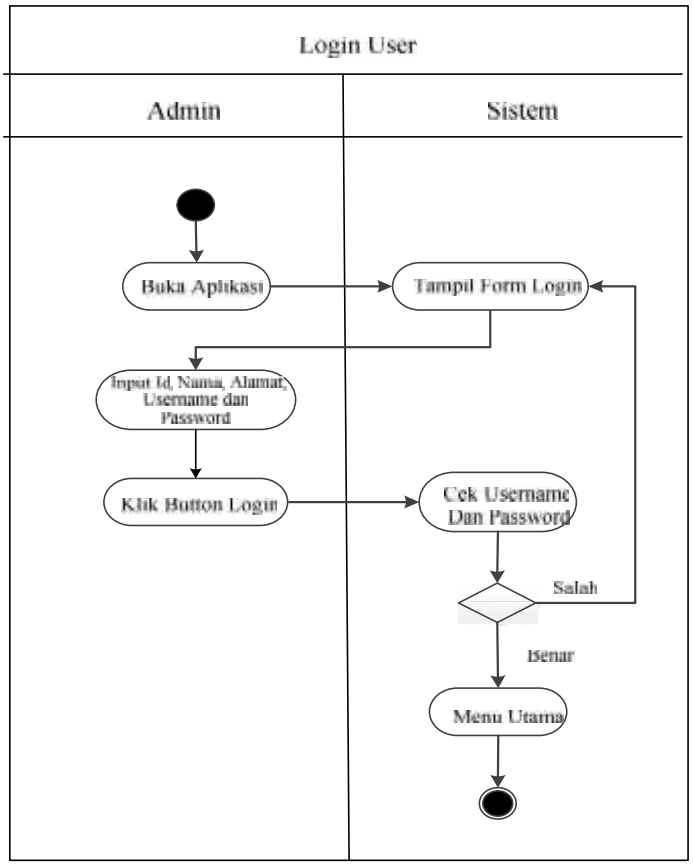
bagaimana *actor* menggunakan sistem untuk melakukan aktivitas. *Decision* digunakan untuk menggambarkan *behaviour* pada kondisi teretentu. Untuk mengilustrasikan proses-proses *parallel* (*fork* dan join) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertical. *Activity* diagram dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktifitas tertentu (I Made Budi Adnyana, 2016 : 52).

Simbol-simbol yang digunakan dalam *activity* diagram dapat dilihat pada tabel II.4.

Tabel II.4. Simbol *Activity* Diagram

Gambar	Keterangan
	*Start Point*, diletakkan pada pojok kiri atas dan merupakan awal aktivitas
	*End Point*, akhir aktivitas
	*Activities*, menggambarkan suatu proses/ kegiatan bisnis
	*Fork/*percabangan, digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel atau untuk menggabungkan dua kegiatan paralel menjadi satu
	*Join* (penggabungan) atau *rake*, digunakan untuk menunjukkan adanya dekomposisi
	*Decision Points,* menggambarkan pilihan untuk pengambilan keputusan, *true* atau *false*
	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa

(Sumber : Ade Hendini, 2016 : 109-110)



Gambar II.4. *Activity Diagram* Login User

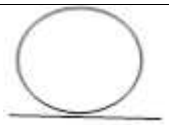
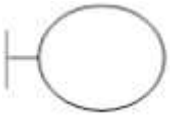
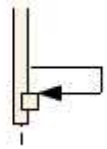
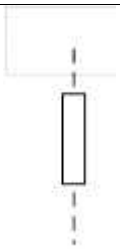
II.8.4. *Sequence* Diagram

Sequence Diagram menggambarkan interaksi atar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence* Diagram terdiri atas dimensi vertical (waktu) dan dimensi horizontal (obyek– obyek yang terkait). *Sequence* Diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu (I Made Budi Adnyana, 2016 : 52).

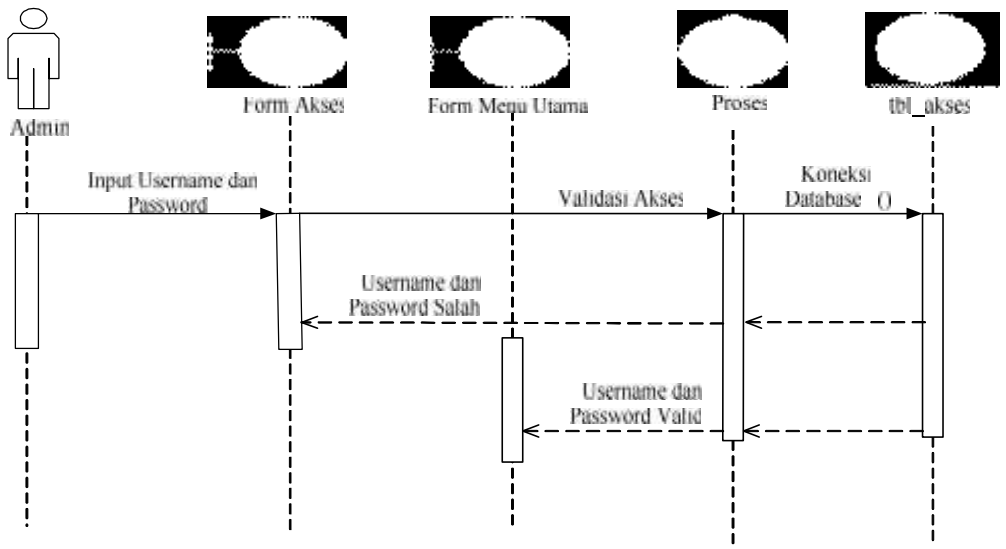
Simbol-simbol yang digunakan dalam *sequence* diagram dapat dilihat pada tabel II.5.

Tabel II.5. 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 *interfaces* atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan *form entry* 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*, mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivasi sebuah operasi
	*Lifeline*, garis titik-titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation*

(Sumber : Ade Hendini, 2016 : 110)



Gambar II.5. *Sequence Diagram* Login Akses

II.8.5. *Object* Diagram

Object diagram menggambarkan struktur sistem dari segi penamaan objek dan jalannya objek dalam sistem. *Object* diagram memastikan bahwa semua kelas yang sudah didefinisikan pada *Class* diagram harus dipakai objeknya, karena jika tidak, pendefinisian kelas itu tidak dapat dipertanggung jawabkan. Simbol-simbol *object* diagram ditunjukkan pada tabel II.6 (Winda Aprianti dan Umi Maliha, 2016 : 22).

Tabel II.6. Simbol *Object* Diagram

Gambar	Keterangan
Nama_objek: nam_kelas Atribut = nilai	Objek. Objek dari kelas yang berjalan saat sistem dijalankan
	Link. Relasi antar objek

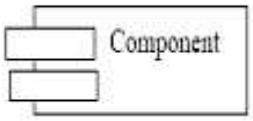
(Sumber : Winda Aprianti dan Umi Maliha, 2016 : 22)

II.8.6. *Deployment* Diagram

Deployment diagram menampilkan rancangan fisik jaringan dimana berbagai komponen akan terdapat di sana. *Deployment* diagram juga dapat menunjukkan perangkat-perangkat *nodes* diantara hubungan yang dimilikinya antar komponen dan menunjukkan tata letak sebuah sistem secara fisik, menampakkan bagian-bagian *software* yang berjalan pada bagian-bagian *hardware* (Marsha Sevin Aldilla, et al., 2015 : 6).

Simbol-simbol yang digunakan dalam *deployment* diagram dapat dilihat pada tabel II.7.

Tabel II.7. Simbol *Deployment* Diagram

Gambar	Keterangan
	Pada *deployment* diagram, komponen-komponen yang ada diletakkan didalam *node* untuk memastikan keberadaan posisi mereka
	*Node* menggambarkan bagian-bagian *hardware* dalam sebuah sistem. Notasi untuk *node* digambarkan sebagai sebuah kubus 3 dimensi
	Sebuah *association* digambarkan sebagai sebuah garis yang menghubungkan dua *node* yang mengindikasikan jalur komunikasi antara elemen-elemen *hardware*

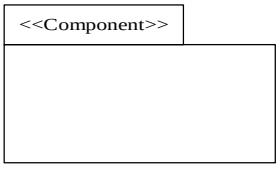
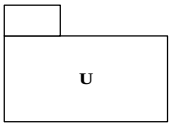
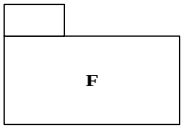
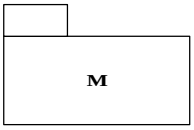
(Sumber : Ade Hendini, 2016 : 111)

II.8.7. *Component* Diagram

Component diagram menunjukkan model secara fisik komponen perangkat lunak pada sistem dan hubungannya antar mereka. Komponen pada piranti lunak adalah berupa modul-modul yang berisikan kode. Umumnya komponen yang terbentuk dari beberapa kelas atau juga terbentuk dari komponen komponen yang lebih kecil(Marsha Sevin Aldilla, et al., 2015 : 6).

Simbol-simbol yang digunakan dalam *component* diagram dapat dilihat pada tabel II.8.

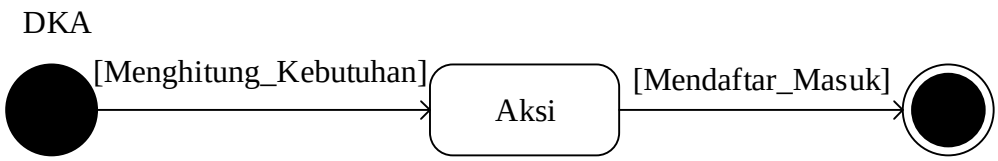
Tabel II.8. Simbol *Component* Diagram

Gambar	Keterangan
	*Peckage* merupakan tempat komponen-komponen di mana komponen tersebut untuk memodelkan sesuatu
	Komponen *user interface* untuk mengatur interaksi antara aktor dan fungsi
	Komponen *function* : memberikan fungsi untuk model
	Komponen model/*database* : digunakan untuk menyimpan objek-objek yang tergambar dalam problem domain
	Dependency merupakan penghubung antara komponen di antara *client-server*

(Sumber : Indrajani, 2015 : 48)

II.8.8. *Statechart* Diagram

Statechart diagram digunakan untuk membuat model bagaimana suatu objek mengalami perubahan *state,* menggambarkan *behaviour* dari sub-sistem, membuat model interaksi antara *class-class* dan model dari tampilan sistem. *Statechart* diagram banyak digunakan pada saat peralihan antara analisis dan fase desain. Umumnya digunakan untuk menggambarkan sistem interaktif yang *"real tima"*. Diagram ini bersifat optional dalam suatu perancangan sistem. Kumpulan sub-*state* yang dikelompokkan ke dalam sebuah *State* disebut sebagai *composite state* (Indrajani, 2015 : 46).



Gambar II.6. *Statechart* Diagram

(Sumber : Indrajani, 2015 : 47)