BAB 2
TINJAUAN PUSTAKA

Tinjauan pustaka memberikan gambaran dari teori yang terkait dengan pembangunan sistem. Teori pokok pada tinjauan pustaka yang terdapat di dalam skripsi ini adalah sebagai berikut :

2.1 Kecerdasan Buatan

Kecerdasan Buatan atau *Artificial Intelligence* merupakan salah satu bagian dari ilmu komputer yang membuat agar mesin komputer dapat melakukan pekerjaan seperti dan sebaik yang dilakukan oleh manusia. Makin pesatnya perkembangan teknologi menyebabkan adanya perkembangan dan perluasan lingkup yang membutuhkan kehadiran kecerdasan buatan. Adanya irisan penggunaan kecerdasan buatan di berbagai disiplin ilmu tersebut menyebabkan cukup rumitnya untuk mengklasifikasikan kecerdasan buatan menurut disiplin ilmu yang menggunakannya, untuk memudahkan hal tersebut maka untuk pengklasifikasian lingkup kecerdasan buatan didasarkan pada *output* yang diberikan yaitu [7]:

1. Sistem pakar (*expert system*) disini computer digunakan sebagai sarana untuk menyimpan pengetahuan para pakar. Dengan demikian komputer akan memiliki keahlian untuk menyelesaikan permasalahan dengan meniru keahlian yang dimiliki oleh pakar.
2. Pengolahan Bahasa Alami (*Natural Language Processing*) dengan pengolahan bahasa alami ini diharapkan user dapat berkomubikasi dengan computer dengan menggunakan bahasa sehari-hari.
3. Pengenalan ucapan (*Speech Recognition*) melalui pengenalan ucapan diharapkan manusia dapat berkomunikasi dengan computer dengan menggunakan suara.
4. Robotika dan sistem sensor (*Robotics & Sencory Systems*)

5. *Computer Vision*, mencoba untuk dapat mengintrepertasikan gambar atau objek-objek tampak melalui komputer.
6. *Intelligent Computer-aided Instruction Computer* dapat digunakan sebagai tutor yang dapat melatih dan mengajar.
7. *Game playing*.

2.1.1 *Fuzzy Logic*

Logika *fuzzy* merupakan salah satu komponen pembentuk *soft computing*. Logika fuzzy pertama kali diperkenalkan oleh Prof. Lotfi A. Zadeh pada tahun 1965. Dasar logika *fuzzy* adalah teori himpunan *fuzzy*. Pada teori himpunan *fuzzy*, peranan derajat keanggotaan sebagai penentu keberadaan elemen dalam suatu himpunan sangatlah penting. Nilai keanggotaan atau derajat keanggotaan atau *membership function* menjadi ciri utama dari penalaran dengan logika *fuzzy* tersebut. Menurut Cox (1994) ada beberapa alasan mengapa orang menggunakan logika *fuzzy* antara lain [7]:

1. Konsep logika *fuzzy* mudah dimengerti, karena logika *fuzzy* menggunakan dasar teori himpunan, maka konsep matematis yang mendasari penalaran fuzzy tersebut cukup mudah untuk dimengerti
2. Logika *fuzzy* sangat fleksibel, artinya mampu beradaptasi dengan perubahan-perubahan, dan ketidakpastian yang menyertai permasalahannya.
3. Logika *fuzzy* memiliki toleransi terhadap data yang tidak tepat. Jika diberikan sekelompok data yang cukup homogen, dan kemudian ada beberapa data yang "eksklusif", maka logika *fuzzy* memiliki kemampuan untuk menangani data eksklusif tersebut.
4. Logika *fuzzy* mampu memodelkan fungsi nonlinear yang sangat kompleks.
5. Logika *fuzzy* dapat membangun dan mengaplikasikan pengalaman pakar secara langsung tanpa harus melalui proses pelatihan. Dalam hal ini, sering dikenal dengan nama *fuzzy expert system* menjadi bagian terpenting

6. Logika *fuzzy* dapat bekerjasama dengan teknik-teknik kendali secara konvensional. Hal ini umumnya terjadi pada aplikasi-aplikasi di bidang teknik mesin maupun teknik elektro.
7. Logika *fuzzy* didasarkan pada bahasa alami. Logika *fuzzy* menggunakan bahasa sehari-hari sehingga mudah dimengerti.

Ada beberapa hal yang perlu diketahui dalam memahami sistem *fuzzy,* yaitu [8]:

a. Variabel *fuzzy* merupakan variabel yang hendak dibahas dalam suatu sistem fuzzy. Contoh yang diterapkan pada penelitian ini adalah kondisi nafsu makan, Kesehatan, dan Kebersihan, dan lain-lain.
b. Himpunan *fuzzy* merupakan suatu grup yang mewakili suatu kondisi atau keadaan tertentu dalam suatu variabel *fuzzy*. Pada himpunan tegas (*crisp*), nilai keanggotaan suatu item x dalam suatu himpunan A, yang sering ditulis dengan $\mu_A(x)$, memiliki dua kemungkinan, yaitu:
   1. Satu (1), yang berarti bahwa suatu item menjadi anggota dalam suatu himpunan, atau
   2. Nol (0), yang berarti bahwa suatu item tidak menjadi anggota dalam suatu himpunan.

   Contoh, Jika diketahui:

   S = {1, 2, 3, 4, 5, 6} adalah semesta pembicaraa

   A = {1, 2, 3} dan B = {3, 4, 5}

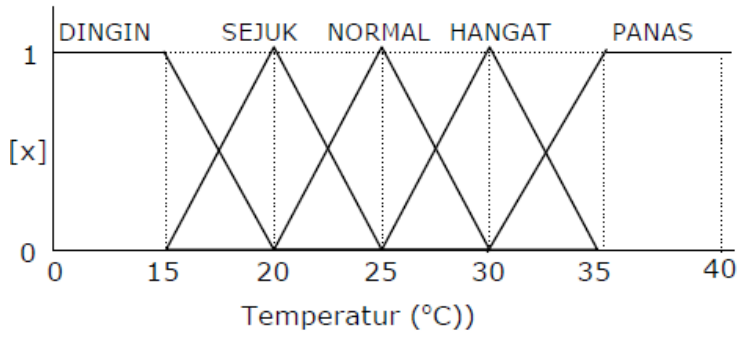
   Bisa dikatakan bahwa:

      a. Nilai keanggotaan 2 pada himpunan A, μA[2]= 1, karena 2 ∈ A.
      b. Nilai keanggotaan 3 pada himpunan A, μA[3]= 1, karena 3 ∈ A.
      c. Nilai keanggotaan 4 pada himpunan A, μA[4]= 0, karena 4 ∉ A.
      d. Nilai keanggotaan 2 pada himpunan B, μA[2]= 0, karena 2 ∉ B.
      e. Nilai keanggotaan 3 pada himpunan B, μA[3]= 1, karena 3 ∈ A.

      Contoh:

         a. Variabel Lapar, terbagi menjadi dua himpunan *fuzzy*, yaitu: Lapar (rendah), dan kenyang (tinggi).

b. Variabel temperatur, terbagi menjadi 5 himpunan *fuzzy*, yaitu: Dingin, sejuk, normal, hangat dan panas.



Gambar 2.1 Himpunan *Fuzzy* Pada Variabel Temperatur [7]

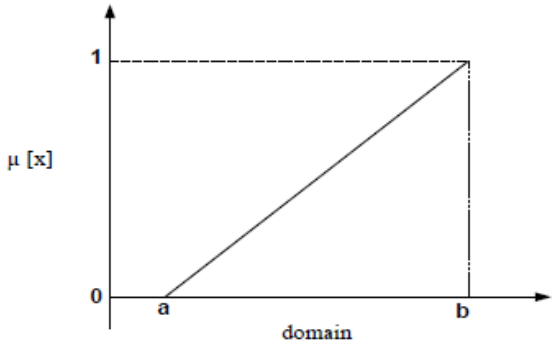
c. Semesta Pembicaraan adalah keseluruhan nilai yang diperbolehkan untuk dioperasikan dalam suatu variabel *fuzzy*. Semesta pembicaraan merupakan himpunan bilangan *real* yang senantiasa naik (bertambah) secara monoton dari kiri ke kanan. Nilai semesta pembicaraan dapat berupa bilangan positif maupun negatif. Adakalanya nilai semesta pembicaraan ini tidak dibatasi batas atasnya. Contoh semesta pembicara pada penerapan penelitian ini adalah parameter kalori dengan semesta pembicara 0-100.

d. Fungsi Keanggotaan adalah suatu kurva yang menunjukkan pemetaan titik-titik input data kedalam nilai keanggotaannya (sering juga disebut dengan derajat keanggotaan) yang memiliki interval antara 0 sampai 1. Salah satu cara yang dapat digunakan untuk mendapatkan nilai keanggotaan adalah dengan melalui pendekatan fungsi. Ada beberapa fungsi yang digunakan, diantaranya adalah:

1. Representasi Linear

Pada representasi linear, pemetaan *input* derajat keanggotaan yang digambarkan sebagai suatu garis lurus. Bentuk ini paling sederhana dan menjadi pilihan yang baik untuk mendekati suatu konsep yang kurang jelas. Terdapat dua keadaan himpunan *fuzzy* yang linear. Pertama, kenaikan himpunan dimulai pada nilai domain yang memiliki derajat nol

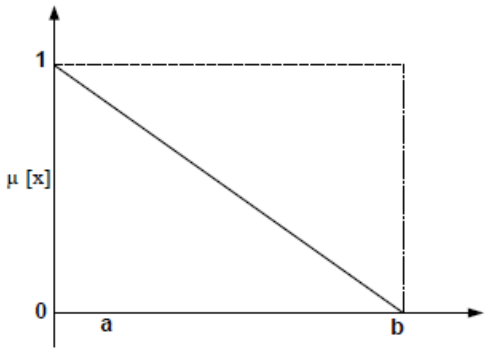
(0) bergerak ke kanan menuju ke nilai dimain yang memiliki derajat keanggotaan lebih tinggi.



Gambar 2.2 Kurva Linear Naik [7]

Fungsi keanggotaan:

$$\mu[x] = \begin{cases} 0; x \leq a \\ \frac{x-a}{b-a}; \quad a \leq x \leq b \\ 1; x \geq b \end{cases} \tag{2.1}$$



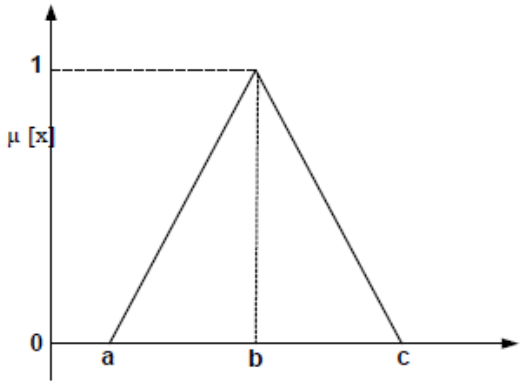
Gambar 2.3 Kurva Linear Turun [7]

Fungsi keanggotaan:

$$\mu[x] = \begin{cases} \frac{b-x}{b-a}; a \leq x \leq b \\ 0; x \geq b \end{cases} \tag{2.2}$$

2. Representasi Kurva Segitiga

Kurva segitiga pada dasarnya merupakan gabungan antara dua garis linear. Seperti dapat dilihat pada gambar



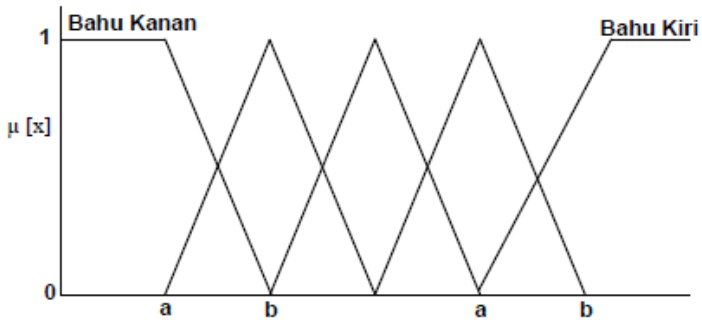
Gambar 2.4 Kurva Segitiga [7]

Fungsi keanggotaan:

$$\mu[x] = \begin{cases} 0; x \leq a \; atau \; x \geq c \\ \frac{x-a}{b-a}; & a \leq x \leq b \\ \frac{b-x}{c-b}; & b \leq x \leq c \end{cases} \quad (2.3)$$

3. Representasi Kurva Bahu

Representasi kurva bahu digunakan untuk mengakhiri variabel suatu daerah variabel fuzzy. untuk lebih jelasnya dapat dilihat pada Gambar 2.5.



Gambar 2.5 Kurva Bahu [7]

Fungsi keanggotaan Kurva Bahu Kiri:

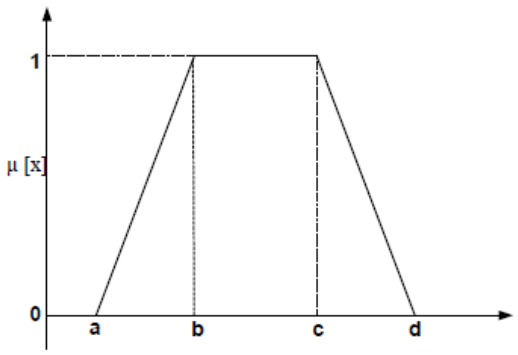
$$\mu[x] = \begin{cases} 1; x \leq a \\ \frac{(b-x)}{(b-a)}; \quad a \leq x \leq b \\ 0; x \geq b \end{cases} \tag{2.4}$$

Fungsi keanggotaan Kurva Bahu Kanan:

$$\mu[x] = \begin{cases} 0; x \leq a \\ \frac{(x-a)}{(b-a)}; \quad a \leq x \leq b \\ 1; x \geq b \end{cases} \tag{2.5}$$

4. Representasi Kurva Trapesium

Kurva trapesium pada dasarnya seperti bentuk segitiga, hanya saja ada beberapa titik yang memiliki nilai kenggotaan 1.



Gambar 2.6 Kurva Trapesium [7]

FungsiKeanggotaan

$$\mu[x] = \begin{cases} 0; x \leq a \; atau \; x \geq d \\ \frac{x-a}{b-a} \\ 1; b \leq x \leq c \qquad ; a \leq x \leq b \\ \frac{d-x}{d-c}; x \geq d \end{cases} \tag{2.6}$$

e. Sistem Inferensi *Fuzzy*

Inferensi *fuzzy* merupakan kerangka komputasi yang didasarkan pada teori himpunan *fuzzy*, aturan *fuzzy* berbentuk IF-THEN, dan penalaran *fuzzy*.

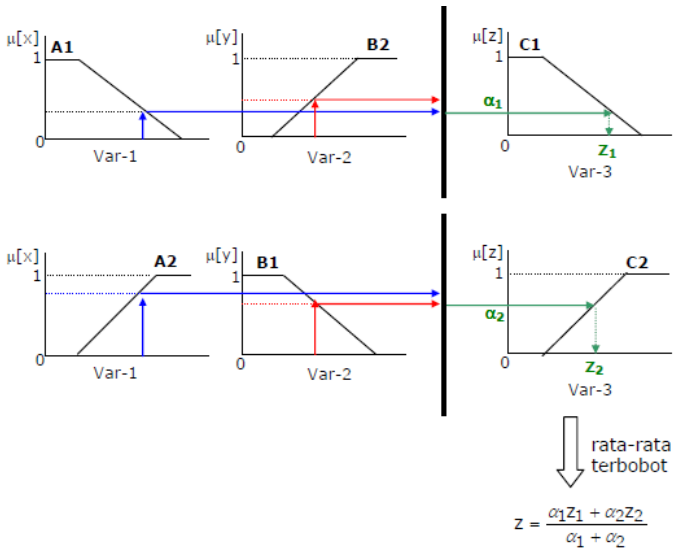
Inferensi *fuzzy* telah berhasil diterapkan di bidang-bidang seperti kontrol otomatis, klasifikasi data, analisis keputusan, dan sistem pakar. Sehingga dari penerapan yang ada dikenal beberapa istilah lain dalam inferensi *fuzzy* yaitu *fuzzy rule based.*

Dalam inferensi *fuzzy* ada beberapa komponen utama yang dibutuhkan. Komponen tersebut meliputi data variabel *input*, data variabel *output*, dan data aturan. Untuk mengolah data variabel *input* dibutuhkab beberapa fungsi meliputi fungsi *fuzzifikasi* yang terbagi dua, yaitu fungsi untuk menentukan nilai jenis keanggotaan suatu himpunan dan fungsi penggunaan operator.

Sistem inferensi *fuzzy* dari sumber buku yang ditulis oleh Kusumadewi, terdapat tiga metode, yaitu:

1. Metode Tsukamoto

   Pada Metode Tsukamoto, setiap konsekuen pada aturan yang berbentuk IF-Then harus direpresentasikan dengan suatu himpunan *fuzzy* dengan fungsi keanggotaan yang monoton. Sebagai hasilnya, output hasil inferensi dari tiap-tiap aturan diberikan secara tegas (*crisp*) berdasarkan α-predikat (*fire strength*). Hasil akhirnya diperoleh dengan menggunakan rata-rata terbobot.



Gambar 2.7 Inferensi Menggunakan Metode Tsukamoto [7]

Agregasi untuk menentukan keluaran semua *rules* dan dikombinasikan menjadi sebuah *fuzzy set* tunggal. *Defuzzifikasi* untuk menghasilkan nilai kesehatan menggunakan rata-rata terbobot.

$$z = \frac{\alpha 1 z 1 + \alpha 2 z 2 \ldots + \alpha n z n}{\alpha 1 + \alpha 2 \ldots + \alpha n} \quad (2.7)$$

1. z = nilai crisp
2. $\alpha 1$ = derajat keanggotaan masing-masing variabel
3. $z1$ = nilai domain pada variabel linguistik z.

2. Metode Mamdani

Metode Mamdani sering juga dikenal dengan nama Metode Max-Min. Metode ini diperkenalkan oleh Ebrahim Mamdani pada tahun 1975. Untuk mendapatkan *output*, diperlukan 4 tahapan, yaitu Pembentukan himpunan fuzzy, Aplikasi fungsi implikasi (aturan), Komposisi aturan, dan Penegasan (*deffuzy*).

3. Metode Sugeno

Penalaran dengan metode Sugeno hampir sama dengan penalaran MAMDANI, hanya saja output (konsekuen) sistem tidak berupa himpunan

fuzzy, melainkan berupa konstanta atau persamaan linear. Metode ini diperkenalkan oleh Takagi-Sugeno Kang pada tahun 1985.

a. Model *fuzzy* Sugeno Orde-Nol

   Secara umum bentuk Model *fuzzy* Sugeno Orde-Nol adalah:

   IF (X_1 is A_1) ● (X_2 is A_2) ● (X_3 is A_3) ●....● (X_N is A_N) THEN z=k

   Dengan A_1 adalah himpunan *fuzzy* ke-i sebagai antaseden, dan k adalah sebagai konstanta (tegas) sebagai konsekuen.

a. Model *fuzzy* Sugeno Orde-Satu

   Secara umum bentuk Model *fuzzy* Sugeno Orde-Satu adalah:

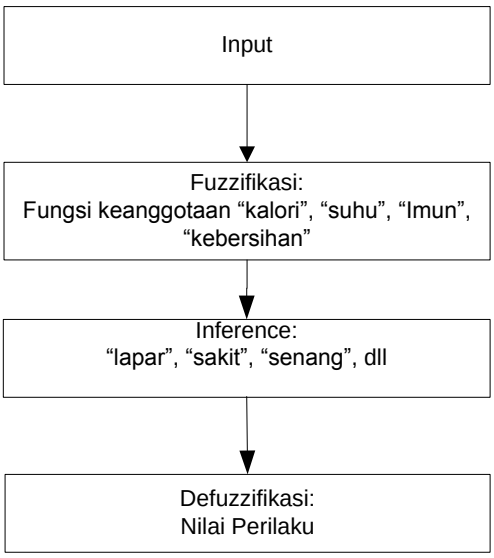
   IF (X_1 is A_1) ● ... ● (X_N is A_N) THEN z = p_1 * x_1 +...+ p_N*p_N + q

   Dengan A_1 adalah himpunan *fuzzy* ke-i sebagai antaseden, dan p_1 adalah sebagai konstanta (tegas) ke-i dan q juga merupakan konstanta sebagai konsekuen.

   Apabila komposisi aturan menggunakan metode Sugeno, maka *defuzzifikasi* dengan cara mencari nilai rata-ratanya.

2.1.2 Desain *fuzzy* Pada Permainan

Dalam penelitian ini dikembangkan sebuah model *fuzzy* yang mampu memodelkan perilaku Bonny dan akan menghasilkan hasil akhir *output* emosi. Hal tersebut merupakan elemen *game* dari kecerdasan buatan supaya dapat berperilaku sealami mungkin selayaknya *pet*. Sebuah sistem yang dibangun menggunakan logika *fuzzy* pada dasar pembangunan *game* ini alurnya adalah sebagai berikut:



Gambar 2.8 Logika *Fuzzy* untuk Meghasilkan Representasi Perilaku

Berikut ini adalah penjelasan dari tahap-tahap gambar di atas:

a. *Fuzzifikasi*, merupakan proses konversi nilai-nilai kebenaran yang bersifat (*crisp input*), yang digunakan sebagai inputan, menjadi bentuk *fuzzy input*, yang berupa nilai *linguistic* yang semantiknya ditentukan berdasarkan fungsi keanggotaan.
b. *Inference,* merupakan proses untuk mengubah input fuzzy dengan cara mengikuti aturan-aturan (*IF-THEN Rules*) yang telah ditetapkan pada basis pengetahian *fuzzy*.
c. *Defuzzifikasi,* merupakan proses untuk mengubah *output fuzzy* yang diperoleh dari proses sebelumnya (*inference)* menjadi nilai tegas menggunakan fungsi keanggotaan yang sesuai dengan saat dilakukan *fuzzifikasi*.

2.2 *Game*

2.2.1 Pengertian *Game*

Menurut Erick Zimmerman, dan Katie Salen [9] *Game* merupakan suatu sistem yang memiliki aturan-aturan tertentu, dimana pemain akan terlibat di dalam suatu permasalahan sehingga dapat menghasilkan suatu hasil yang dapat

diukur yaitu menang atau kalah. *Game* merupakan sesuatu hal yang dimainkan dengan suatu aturan tertentu yang biasa digunakan untuk tujuan kesenangan dan dapat juga digunakan sebagai sarana edukasi.

Game umumnya melibatkan stimulasi mental, fisik, atau keduanya. Banyak game yang dapat membantu mengembangkan keterampilan praktis yang berfungsi sebagai latihan, atau melakukan peran pendidikan, simulasional, atau psikologis.

2.2.2 *Game Virtual Pet*

Permainan *Bonny's tooth booth* merupakan jenis permainan yang mengusung konsep *virtual pet,* berikut ini adalah penjelasan mengenai permainan *virtual pet.*

Permainan *virtual pet* merupakan jenis permainan yang masuk ke dalam kategori genre simulasi, g*ame* simulasi merupakan jenis permainan yang bertujuan untuk memberikan pengalaman simulasi kepada pemain. Biasanya melibatkan sebagian besar tantangan fisik, dan taktik, tapi tidak ada eksplorasi, dan penilaian. [9]

Virtual pet adalah konsep memelihara hewan dalam konsep virtual. Menjaga *pet* tetap hidup dan tumbuh adalah yang menjadi *goal* atau tujuan dalam jenis permainan ini. Interaksi antara pengguna dan karakter *pet* biasanya sederhana dengan memberikan makan, dan membuat *pet* tetap senang. Salah satu permainan *virtual pet* yang populer adalah Tamagotchi. [10]

Permainan *Virtual pet* saat ini banyak dikembangkan sebagai bentuk permainan untuk media hiburan, salah satu perwujudan dari pengembangannya adalah bentuk aplikasi perangkat lunak dan juga robotika. [11]



Gambar 2.9 Contoh *Interface* Permainan *Virtual Pet* Kimimon[20]

Sebagian besar pengguna yang memainkan permainan ini dapat merasakan efek gembira dan bahkan dapat menimbulkan perasaan kasih sayang, yang merupakan tingkat interaksi lanjutan dari aspek psikologis dari *user* ke karakter *pet*. Beberapa lingkup di mana pengaruh *virtual pet* pada manusia telah diuji dan dipelajari yang meliputi pendidikan, kesehatan, reaksi sosial dan psikologis terhadap pengguna. Beberapa studi telah mengembangkan permainan ini dengan menggunakan aspek teknologi yang menggunakan kecerdasan buatan di balik menciptakan perilaku agar lebih alami terhadap karakter *pet*-nya. Dalam rangka untuk lebih meniru realitas dan membuat *user* lebih kaya pengalaman, kombinasi antara strategi hiburan, algoritma cerdas dan jenis interaksi harus digunakan ketika mengembangkan *virtual pet*. Penerapan umumnya terhadap target dalam tampilan dan perilaku tak terduga dari robot dibuat harus merangsang psikologi pengguna, bahkan dapat meniru aspek perilaku hewan sungguhan. Lahey dkk, menyebutkan dalam studi mereka dengan strategi *agent pet* yang menarik bagi pengguna diwakili dengan menambah beberapa fitur fisik dari hewan peliharaan *virtual* yang membantu meningkatkan "Kepribadian" pada robot *pet*. [22]

2.3 Fungsi Gigi dan Menjaga Kesehatan Gigi

Bonny's Tooth Booth adalah permainan yang mengusung tema mengenai menjaga dan merawat kesehatan gigi. Berikut ini adalah penjelasan mengenai gigi dan kesehatan gigi.

a. Definisi Gigi

Gigi adalah alat yang digunakan untuk mengunyah makanan. Gigi tumbuh pada rahang atas dan rahang bawah yang berpasangan atau simetris kiri dan kanan. Secara anatomi gigi manusia terdiri dari 2 bagian, yaitu mahkota gigi yang dilapisi oleh email yang sangat keras dengan warna yang tidak sama dengan setiap orang, serta akar gigi yaitu bagian gigi yang tertanam dalam tulang rahang berfungsi sebagai penguat akan berdirinya gigi [12].

b. Fungsi Gigi

Secara garis besar gigi berfungsi sebagai berikut:

1. Alat untuk mengunyah makanan. Penghancuran dan pelumatan dari makanan dalam mulut sangat perlu untuk membantu pencernaan. Makanan yang dikunyah dengan baik akan mempermudah penyerapan zat makanan oleh darah.
2. Alat untuk menjaga agar ucapan kata-kata kita tepat dan jelas.
3. Alat untuk menjaga estetis (kecantikan).

Menjaga kesehatan gigi dan mulut sangatlah penting. Karena jika tidak berbagai masalah kesehatan seputar gigi dan mulut dapat muncul kapan saja. Misalnya sakit gigi, masalah gigi berlubang, gusi bengkak dan sebagainya. Masalah mulut yang biasa muncul adalah bau mulut tak sedap, sariawan hingga sakit tenggorokan. Semua itu adalah akibat dari kita yang tidak memperhatikan kesehatan, terutama kesehatan gigi dan mulut [21].

Anjuran untuk memelihara gigi agar tetap sehat diantaranya yaitu rajin membersihkan gigi, konsumsi buah-buahan, konsumsi sayuran, rajin sikat gigi, dan periksakan gigi setiap maksimal enam bulan sekali ke dokter gigi.

2.4 *Tools* yang Digunakan

Tools yang digunakan meliputi perancangan berbasis objek, UML, Android, dan bahasa pemrograman Java.

2.4.1 Perancangan Berbasis Objek

Perancangan berorientasi objek adalah strategi perancangan di mana perancangan sistem memikirkan ‘benda’ dan bukan operasi atau fungsi. Proses perancangan berorientasi objek berarti bahwa kita mengorganisasi perangkat lunak sebagai kumpulan dari objek tertentu yang memiliki struktur data dan perilaku. Beberapa cara untuk menentukan karakteristik dalam pendekatan berorientasi objek secara umum mencakup empat hal, yaitu identifikasi, klasifikasi polimorfisme, dan pewarisan. Keuntungan menggunakan perancangan berorientasi objek diantaranya yaitu alami (Natural), dapat diandalkan (*Reliable*), dapat digunakan kembali (*Reusable*), mudah untuk dalam perawatan (*Maintainable*), dapat diperluas (*Extendable)*, dan efisiensi waktu. [13]

2.4.2 UML

UML (*Unified Modeling Language*) adalah sebuah bahasa yang menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. Pada UML terdapat sepuluh macam diagram untuk memodelkan aplikasi dengan berorientasi objek, yaitu [14]:

a. *Use Case Diagram* untuk memodelkan proses bisnis.
b. *Conceptual Diagram*untuk memodelkan konsep yang ada di dalam aplikasi.
c. *Sequence Diagram*untuk memodelkan pengiriman pesan antar objek.
d. *Collaboration Diagram* untuk memodelkan interaksi antara objek-objek.
e. *State Diagram* untuk memodelkan perilaku objek di dalam sistem.
f. *Activity Diagram* untuk memodelkan perilaku *use case*
g. *Class Diagram* untuk memodelkan struktur kelas.
h. *Object Diagram* untuk memodelkan struktur objek.
i. *Component Diagram* untuk memodelkan komponen objek, dan
j. *Deployment Diagram* untuk memodelkan distribusi aplikasi.

Pada perancangan aplikasi ini menggunakan empat model UML diagram, yaitu *use case diagram*, *activity diagram, class diagram*, dan *sequence diagram.* [15]

1. Use Case Diagram

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya *login* ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang atau sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu.

Use case diagram dapat sangat membantu menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem.

Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang umum. Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.

Setelah rangkaian *use case*, menghasilkan skenario *use case* yang merupakan bagian dari detail dan persyaratan fungsional yang dijelaskan dengan narasi ke dalam daftar bernomor. [16]

2. Class Diagram

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi).

Class diagram menggambarkan struktur dan deskripsi *class, package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain.

3. Activity Diagram

Activity diagram menggambarkan berbagai alur aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alur berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

Activity diagram merupakan *state diagram* khusus, dimana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan *behaviour* internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

4. Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan.

2.4.3 Android

Android adalah sebuah sistem operasi untuk perangkat *mobile* berbasis linux yang mencakup sistem operasi, *middleware*, dan aplikasi. Android menyediakan platform terbuka bagi para pengembang untuk menciptakan aplikasi

mereka. Awalnya, Google Inc. membeli Android Inc, yang merupakan pendatang baru yang membuat piranti lunak untuk ponsel/smartphone. Kemudian untuk mengembangkan Android, dibentuk Open Handset Alliance, kemudian Open Handset Alliance menyatakan mendukung pengembangan open source pada perangkat mobile. Di lain pihak, Google merilis kode-kode Android di bawah lisensi Apache, sebuah lisensi perangkat lunak dan *open platform* untuk perangkat seluler [17].

2.4.3.1 Arsitektur Pada Android

Secara garis besar arsitektur Android dapat dijelaskan dan digambarkan sebagai berikut [18]:



Gambar 2.10 Lapisan Arsitektur sistem operasi Android [18]

a. *Application* dan *Widgets*

Application dan *Widgets* ini adalah layer dimana kita berhubungan dengan aplikasi saja, dimana biasanya kita download aplikasi kemudian kita lakukan instalasi dan jalankan aplikasi tersebut. Di layer terdapat aplikasi inti termasuk klien email, program SMS, kalender, peta, browser, kontak dan lain-lain. Semua aplikasi ditulis menggunakan bahasa pemrograman java.

b. *Applications Frameworks*

Android adalah "*Open Development Platform*" yaitu Android menawarkan kepada pengembang atau memberi kemampuan kepada pengembang untuk membangun aplikasi yang bagus dan inovatif. Arsitektur aplikasi dirancang supaya pengembang dengan mudah dapat menggunakan kembali komponen yang sudah digunakan (*reuse*).

Komponen-komponen yang termasuk di dalam *Application Frameworks* adalah sebagai berikut:

1. *View* yang kaya dan *extensible* yang dapat digunakan untuk membangun aplikasi, termasuk list, grid, kotak teks, tombol, dan bahkan sebuah *embeddable web*.
2. *Content provider* yang memungkinkan aplikasi untuk mengakses data (seperti dari daftar kontak *help*) atau membagi data tersebut.
3. *Resource Manager* yang menyediakan akses ke kode non sumber lokal seperti string, gambar, dan tata letak file.
4. *Notification Manager* yang memungkinkan semua aplikasi menampilkan alert yang bisa dokostumisasi didalam status bar.
5. *Activity Manager* yang mengelola siklus hidup aplikasi dan menyediakan navigasi umum *backstack.*

c. *Libraries*

Libraries ini adalah layer dimana fitur-fitur Android berada, biasanya para pembuat aplikasi mengakses *libraries* untuk menjalankan aplikasinya. Berjalan di atas kernel. Layer ini meliputi berbagai Library C/C++ inti seperti Libe dan SSL, serta:

1. *Libraries* media untuk pemutaran media audio dan video.
2. *Libraries* untuk manajemen tampilan.
3. *Libraries* grafik mencakup OpenGL untuk grafis 2D dan 3D.
4. *Libraries* SQLite untuk dukungan database, dan lain-lain.

d. *Android Run Time.*

Layer yang membuat aplikasi android dapat dijalankan dimana dalam prosesnya menggunakan Implementasi Linux. *Dalvik Virtual Machine* (DVM) merupakan mesin yang membentuk dasar kerangka aplikasi Android. Di dalam Android *Run Time* dibagi menjadi dua bagian yaitu:

1. *Core Libraries*: Aplikasi Android dibangun dalama bahasa java, sementara Dalvik sebagai virtual mesinnya buka Virtual Machine Java, sehingga diperlukan sebuah *libraries* yang berfungsi untuk menterjemahkan bahasa java/C yang ditangani oleh *Core Libraries*.
2. *Dalvik Virtual Machine*: Mesin virtual berbasis register yang dioptimalkan untuk menjalankan fungsi-fungsi secara efisien, dimana merupakan pengembangan yang mapu membuat linux kernel untuk melakukan threading dan manajemen tingkat rendah.

e. Linux Kernel

Linux Kernel adalah *layer* dimana inti dari *operating system* Android itu berada. Berisi file-file sistem yang mengatur sistem *processing, memory, resource, drivers,* dan sistem-sistem operasi Android lainnya. Linux *Kernel* yang digunakan Android adalah *linux kernel* versi 2.6.

2.4.3.2 *Tools* Pendukung Pembuatan Aplikasi Android

Tools pendukung pembuatan aplikasi Android dalam pembuatan aplikasi meliputi *Java Development Kit, Software Development Kit,* dan Eclipse.

a. *Java Development Kit* (JDK)

JDK adalah *software development kit* merupakan seperangkat kode, library, form dan paket yang berguna untuk membuat aplikasi atau applet java. Di dalam JDK terdapat Java *Runtime Environment* (JRE) yang akan memproses *byte code* yang dihasilkan dari interpenter java.

Tugas dari interpenter java ini adalah untuk mengcompile source.java menjadi *source*.class yang selanjutnya source.class inilah yang akan dijalankan oleh JRE.

b. *Software Development Kit* (SDK)

Android SDK adalah *tools* API (*Application Programming Interface*) yang diperlukan untuk mulai mengembangkan aplikasi Android menggunakan bahasa pemrograman Java. *Source* SDK Android ini dapat diunduh langsung di situs resmi pengembang SDK Android.

c. Eclipse

Eclipse adalah sebuah IDE (*Integrated Development Environtment*) untuk mengembangkan perangkat lunak dan dapat dijalankan di semua *platform* (*platform independent*). Selain sebagai IDE untuk pengembangan aplikasi, Eclipse pun bisa digunakan untuk aktivitas dalam siklus pengembangan perangkat lunak, seperti dokumentasi, *test* perangkat lunak, pengembangan web, dan lain sebagainya.

2.6.2 Java

Menurut definisi yang diberikan oleh Sun MicroSystem, JAVA adalah bahasa pemrograman yang sederhana, *object-oriented, distributed, interpreted, robust, secure, architecture neutral, portable, high-performance, multithreaded, dan dinamyc*. Berikut ini adalah penjelasan dari definisi-definisi di atas [19]:

1. Java bahasa yang sederhana: Rancangan bahasa java dibuat sedemikian rupa sehingga dengan cepat dapat dikenali dan dipahami oleh pemrogram pemula sekalipun karena mirip dengan bahasa C/C++ dan yang kedua lebih sederhana dari C/C++ dalam beberapa hal yang sulit pada C/C++ sudah tidak ada pada Java, misalnya pemakaian pointer.
2. Java adalah bahasa yang *object oriented*: Konsep objek diterapkan pada pembuatan program Java, bahkan java menggunakan konsep objek secara murni. Java adalah bahasa yang distributed: Bahasa Java dirancang

sedemikian rupa sehingga mendukung pembuatan aplikasi dalam jaringan computer. Java adalah bahasa yang robust: rancangan java dibuat sedemikian rupa hingga mengurangi kemungkinan menjadi beku (*freeze*) ketika dijalankan. Java adalah bahasa yang *secure*: secara otomatis java menerapkan pengamanan terhadap aplikasi sehingga mengurangi kemungkinan terjadinya serangan dari pengguna jaringan. Java adalah bahasa yang *architecture neutral*: program java yang telah dikompilasi, dapat dijalankan. Java adalah bahasa yang *portable*: program java sangat mudah dipindahkan dari satu mesin ke mesin lain tanpa harus dikompilasi ulang.

3. Java adalah bahasa yang *high-performance*: pada hakikatnya program C yang telah dikompilasi berjalan jauh lebih cepat dari Java, namun hasil kompilasi ini belum tentu bisa jalan pada komputer yang berbeda arsitektur. Program java dikompilasi menjadi *bytecode* yang bisa jalan diberbagai mesin, namun ketika dijalankan *bytecode* ini harus diterjemahkan oleh JVM ke dalam bahasa mesin yang sesuai, sehingga lebih lambat dari C, namun karena java dapat digunakan pada jaringan komputer yang kenyataannya tidak terlalu cepat karena hambatan saluran komunikasi maka program Java masih terasa cukup cepat.
4. Java adalah bahasa yang *multithreaded:* Java dirancang untuk menangani berbagai aplikasi yang berjalan secara bersamaan, misalnya memainkan lagu, sambil melakukan *download*. Kejadian ini disebut dengan *multith*
5. *readed*. Java adalah bahasa yang *dynamic*: bahasa java dirancang untuk beradaptasi dengan lingkungan yang dapat berubah dengan cepat.