BAB II
LANDASAN TEORI

2.1 Game

Video game merupakan salah satu media hiburan yang paling popular untuk semua kalangan usia. Sejak pertama kali ditemukan sampai saat ini, teknologi game telah mengalami kemajuan yang pesat. Hal ini ditandai dengan berkembangnya jenis, produk, alat, dan jenis interaksi game dengan penggunaan yang semakin beragam bentuknya.

2.1.1 Pengertian *Game*

Game berasal dari kata bahasa inggris yang memiliki arti dasar Permainan. Permainan dalam hal ini merujuk pada pengertian "kelincahan intelektual" (*intellectual playability*). *Game* juga bisa diartikan sebagai arena keputusan dan aksi pemainnya. Ada target-target yang ingin dicapai pemainnya. Kelincahan intelektual, pada tingkat tertentu, merupakan ukuran sejauh mana *game* itu menarik untuk dimainkan secara maksimal. *Game* merupakan sebuah bentuk seni dimana penggunanya, disebut dengan pemain (*player*), diharuskan membuat keputusan-keputusan dengan tujuan untuk mengelola sumber daya yang diperoleh dari kesempatan-kesempatan bermain (token) miliknya, untuk mencapai suatu tujuan tertentu.

Video *game* adalah bentuk *game* yang interaksi utamanya melibatkan media video (dan biasanya melibatkan audio). Berdasarkan representasi visualnya, *game* dapat dibedakan menjadi 2 jenis, yaitu game 2 dimensi (2D) dan 3 dimensi (3D). *Game* 2D adalah game, yang secara matematis, hanya melibatkan 2 elemen koordinat kartesius, yaitu x dan y, sehingga konsep kamera pada game 2D hanya menentukan "gambar" mana pada game yang dapat dilihat oleh pemain. Sedangkan game 3D adalah game yang selain melibatkan elemen x dan y, juga melibatkan elemen z pada perhitungannya, sehingga konsep kamera pada game 3D benar-benar menyerupai konsep kamera pada kehidupan nyata, yaitu selain digeser (seperti pada game 2D), juga dapat diputar dengan sumbu tertentu.

Menariknya akan tantangan maupun visualisasi yang ada pada game sering mengakibatkan ketergantungan terhadap game tersebut (*addicted)*. Ini merupakan salah satu akibat buruk yang disebabkan oleh game (*game addicted)* [8].

2.1.2 Jenis-jenis *Game* Berdasarkan *Platform*

Berikut ini beberapa jenis *game* berdasarkan cara pembuatannya, cara pemasarannya dan mesin yang menjalankannya. Jenis-jenis *game* tersebut adalah [8] :

1. *Game* PC

   Game PC adalah *game* yang dimainkan pada PC (*Personal Computer*) yang memiliki kelebihan yaitu tampilan antarmuka yang baik untuk *input* maupun *output*. *Output* visual berkualitas tinggi karena layar komputer biasanya memiliki resolusi yang jauh lebih tinggi dibandingkan dengan layar televisi biasa.Kekurangannya adalah spesifikasi komputer yang sangat bervariasi antar satu komputer dengan komputer yang lainnya menyebabkan beberapa *game* dapat ditampilkan dengan baik pada satu komputer tetapi tidak berjalan dengan baik pada komputer yang lainnya.
2. *Game Console*

   Game console adalah *game* yang dijalankan pada suatu mesin spesifik yang biasanya tersedia di rumah seperti Xbox,Nintendo Wii dan lain-lain.
3. *Game Arcade*

   Game arcade adalah *game* yang dijalankan pada mesin dengan *input* dan *output* audio visual yang telah terintegrasi dan tersedia ditempat-tempat umum.
4. *Game Online*

   Game online adalah *game* yang hanya dapat dumainkan secara *online* melaui LAN atau internet.

2.1.3 Genre *Game*

Berdasarkan *genre* permainannya, *game* dapat dibagi menjadi beberapa *genre* yaitu:

1. *Action*

   Action game adalah *game* dimana kebanyakan dari tantangan yang disajikan merupakan dari tes *physical skill* dan koordinasi pemain. Salah satu *sub-genre action game* adalah *shooter game*, baik yang 2D ataupun 3D seperti *First Person Shooter (FPS).*

2. *Strategy*

   Strategy game menantang pemain untuk mencapai kemenangan dengan perencanaan, khususnya melalui perencanaan serangkaian tindakan yang dilakukan melawan satu lawan atau lebih. Kemenangan diraih dengan perencanaan matang dan pengambilan keputusan optimal.

3. *Role Playing Game* (RPG)

   RPG adalah *game* dimana pemain mengontrol satu atau lebih karakter yang biasanya di desain oleh pemain itu sendiri, dan memandu mereka melewati berbagai rintangan yang diatur oleh komputer. Perkembangan karakter dalam hal kekuatan dan kemampuannya adalah kunci dari *game* jenis ini.

4. *Sports*

   Sports game mensimulasikan berbagai aspek dari olahraga atletik nyata ataupun imajiner, apakah itu memainkan pertandingan, me-*manage* tim dan karir, atau keduanya. Salah satu contoh *game* jenis ini adalah *Pro Evolution Soccer 2012* (PES 2012), dimana pemain bisa memainkan pertandingan, menjadi manajer tim, maupun menjadi pemain dan mengembangkan karirnya sendiri.

5. *Vehicle Simulation*

   Vehicle simulation membuat *feeling* mengendarai kendaraan, baik kendaraan nyata maupun kendaraan imajiner. Performa dan karakteristik kasar mesin harus menyerupai kenyataan, kecuali jika yang didesain adalah kendaraan imajiner.

6. *Construction and Management Simulation*

   CMS game adalah *game* tentang proses. Tujuan pemain bukan untuk mengalahkan musuh, tetapi membangun sesuatu dengan konteks proses yang sedang berjalan. Semakin pemain mengerti dan mengontrol proses, semakin sukses sesuatu yang ia bangun. *Game* seperti ini biasanya menyediakan dua jenis permainan, yaitu mode bebas dimana pemain bebas membangun sesuatu, dan mode misi dimana terdapat skenario hal apa yang harus dibangun oleh pemain.

7. *Adventures*

   Adventure game adalah cerita interaktif tentang karakter protagonist yang dimainkan oleh pemain. Penyampaian cerita dan eksplorasi adalah elemen inti dari *game* ini. Penyelesaian teka-teki dan tantangan konseptual adalah bagian besar dari permainan.

8. *Artificial Life and Puzzle Game*

   Artificial Life game adalah *game* yang membuat tiruan dari kehidupan sebenarnya. Biasanya ada dua jenis *game* ini, tiruan kehidupan manusia, contohnya The SIMS, dan tiruan kehidupan binatang, contohnya Tamagochi.

9. *Online Game*

   Istilah *online game* disini mengacu kepada *multiplayer game* dimana mesin dari para pemain terhubung dengan jaringan.

2.1.4 Kategori *Game* Lainnya

Selain berdasarkan genre permainannya terdapat pula kategori-kategori *game* lainnya, yaitu:

1. *Multiplayer Online*

   Game yang dapat dimainkan secara bersamaan oleh lebih dari 2 orang (bahkan dapat mencapai puluhan ribu orang dalam satu waktu) membuat pemain dapat bermain bersama dalam satu dunia virtual dari sekedar *chatting* hingga membunuh naga bersama teman yang entah bermain di mana. Umumnya permainan tipe ini dimainkan di PC dan bertema RPG,

walau ada juga yang bertema musik atau *action*.

2. *Casual Games*

   Sesuai namanya, *game casual* itu tidak kompleks, mainnya rileks dan sangat mudah untuk dipelajari (bahkan cenderung langsung bisa dimainkan). Jenis ini biasanya memerlukan spesifikasi komputer yang standar. Genre permainannya biasanya *puzzle* atau *action* sederhana dan umumnya dapat dimainkan hanya menggunakan *mouse* (biasanya *game* lain menggunakan banyak tombol tergantung *game*-nya). Contohnya: *Dinner Dash, Zuma, Feeding Frenzy*.

   Fitur dasar pada *game casual*, yaitu:

   1) *Gameplay* sangat sederhana, bisa dimainkan menggunakan *mouse, keyboard* atau keypad ponsel.
   2) Dimainkan dalam waktu singkat. Biasanya dibawah 10 menit. Bisa dimainkan saat istirahat kerja atau pada transportasi umum.
   3) Kemampuan yang cepat dalam mencapai tahap akhir tanpa perlu *save game*.
   4) Kadang digunakan untuk model bisnis atau periklanan.
   5) Cocok dalam *genre* apapun.
   6) Dapat dimainkan oleh siapa saja, anak – anak sampai orang dewasa.
   7) Dapat dimainkan dengan atau tanpa pengalaman dalam bermain *game.*

3. *Edugames*

   Video game jenis ini dibuat dengan tujuan spesifik sebagai alat pendidikan, entah untuk belajar mengenal warna untuk balita, mengenal huruf dan angka, matematika, sampai belajar bahasa asing. *Developer* yang membuatnya, harus memperhitungkan berbagai hal agar game ini benar-benar dapat mendidik, menambah pengetahuan dan meningkatkan ketrampilan yang memainkannya. Target segmentasi pemain harus pula disesuaikan dengan tingkat kesulitan dan desain visual ataupun animasinya.

4. *Advergames*

   Jenis *game* yang biasanya mudah dimainkan ini mengusung dan menampilkan produk atau *brand* secara gamblang maupun tersembunyi. Di era tumbuhnya media-media baru berteknologi tinggi sekarang ini, dunia periklanan memang sudah tidak lagi terbatas pada TV, koran, majalah, *billboard* dan radio, *video game* sekarang telah menjadi sarana beriklan atau membangun *brand-awareness* yang efektif. Baik melalui internet maupun di mainkan di *event-event* mereka, *edugames* terasa semakin dibutuhkan untuk menjaring calon konsumen bagi produk yang menggunakan *advergames* ini.

2.1.5 Elemen Pada *Game*

Elemen – elemen yang ada dalam sebuah *game* biasanya terdiri dari:

1. Desain *game*

   Desain adalah langkah awal untuk membuat semua elemen *game*. Desain *game* dibuat semenarik mungkin agar pemain tidak cepat bosan yang dampaknya membuat *game* tersebut cepat ditinggalkan.

2. Pemrograman *game*

   Pemrograman *game* sebagian besar digunakan untuk mengontrol gerakan objek di layar. Selain itu, pemrograman *game* juga digunakan untuk pemrograman suara, *input device,* deteksi keadaan lain – lain.

3. Grafis *game*

   Grafis *game* memegang peranan penting dalam pembuatan tampilan. Tampilan haruslah dibuat semenarik mungkin, sehingga dengan melihatnya saja *end user* langsung tertarik untuk memainkannya.

4. Musik dan *sound*

   Musik dan *sound* dalam suatu *game* merupakan suatu hal yang wajib dikarenakan dalam suatu *game* akan lebih terasa efek emosi dari *game* tersebut, sehingga akan membuat kesan *game* tersebut lebih terasa nyata.

2.2 Kecerdasan Buatan Pada *Game*

Kecerdasan buatan atau AI merupakan kegiatan membuat komputer agar dapat berpikir dan mengerjakan kegiatan yang dapat dilakukan oleh manusia maupun binatang.

Saat ini dapat ditemukan program komputer yang memiliki kemampuan menangani masalah seperti aritmatik, sorting, searching. Bahkan komputer juga dapat bermain beberapa board *game* seperti catur dan reversi lebih baik daripada manusia.

Namun, masih banyak hal yang tidak dapat dilakukan dengan baik oleh komputer. Seperti, mengenali wajah, berbicara bahasa manusia, menentukan sendiri apa yang harus dilakukan, dan bertingkah kreatif. Hal itu semua merupakan domain dari AI untuk mencoba menentukan algoritma apa yang dibutuhkan untuk memenuhi kebutuhan diatas.

Dalam bidang akademik, beberapa peniliti AI termotivasi oleh filosofi, yaitu memahami alam pikiran dan alam kecerdasan dan membangun program untuk memodelkan bagaimana proses berpikir. Beberapa juga termotivasi oleh *psychology*, bertujuan untuk memahami mekanisme otak manusia dan proses mental. Dan lainya termotivasi oleh engineering, dengan tujuan membangun algoritma untuk melakukan kegiatan seperti manusia atau hewan.

Dalam pembangunan *game*, umumnya akan cenderung hanya pada sisi engineering yang bertujuan membangun algoritma yang dapat membuat *game* karakter mengerjakan kegiatan seperti yang dilakukan manusia atau binatang [5].

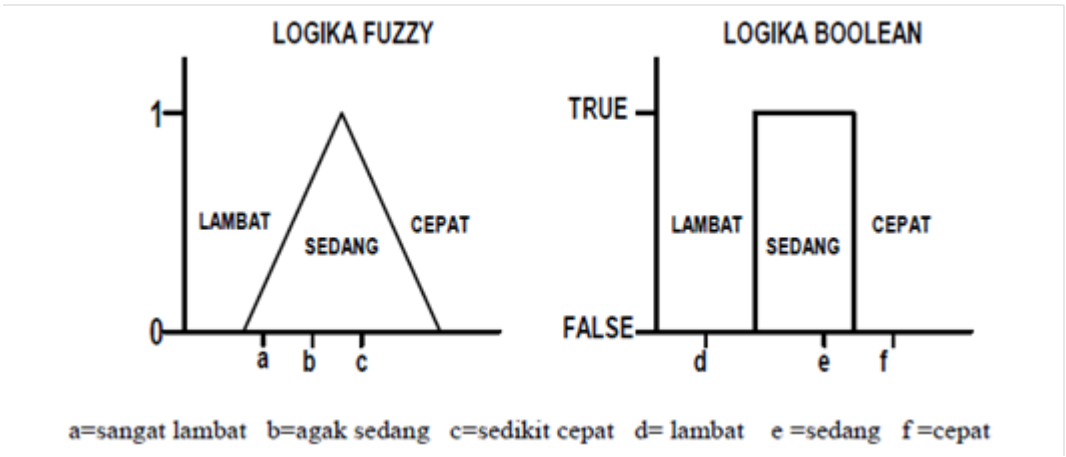
2.3 Algoritma Fuzzy Logic

Logika fuzzy adalah peningkatan dari logika Boolean yang berhadapan dengan konsep kebenaran sebagian. Dimana logika klasik (*crisp*) menyatakan bahwa segala hal dapat diekspresikan dalam istilah binary (0 atau 1, hitam atau putih, ya atau tidak). Logika fuzzy menggantikan kebenaran Boolean dengan tingkat kebenaran. Logika fuzzy memungkinkan nilai keanggotaan antara 0 dan 1, tingkat keabuan dan juga hitam dan putih, dan dalam bentuk linguistic, konsep tidak pasti seperti “sedikit”, “lumayan”, dan “sangat”. Logika ini diperkenalkan

oleh Dr.Lotfi Zadeh dari Universitas California, Barkeley pada tahun 1965. Logika fuzzy telah digunakan pada bidang-bidang seperti taksonomi, topologi, linguistik, teori automata, teori pengendalian, psikologi, pattern recogniti on, pengobatan, hukum, decision analysis, system theory and information retrieval.Pendekatan fuzzy memiliki kelebihan pada hasil yang terkait dengan sifat kognitif manusia, khususnya pada situasi yang melibatkan pembentukan konsep, pengenalan pola, dan pengambilan keputusan dalam lingkungan yang tidak pasti atau tidak jelas.[1]

2.3.1 Himpunan Fuzzy

Himpunan fuzzy merupakan suatu pengembangan lebih lanjut tentang konsephimpunan dalam matematika. Himpunan Fuzzy adalah rentang nilai-nilai. Masing-masing nilai mempunyai derajat keanggotaan (membership) antara 0 sampai dengan 1. Ungkapan logika Boolean menggambarkan nilai-nilai “benar” atau “salah”. Logika fuzzy menggunakan ungkapan misalnya : “sangat lambat”, ”agak sedang”, “sangat cepat”dan lain-lain untuk mengungkapkan derajat intensitasnya. Ilustrasi antara keanggotaan fuzzy dengan Boolean set dapat dilihat pada gambar dibawah ini.



Gambar 2.1 Pendefinisia kecepatan dalam bentuk fuzzy logic [3].

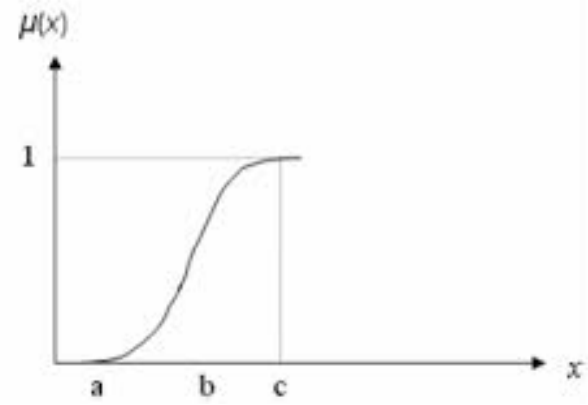
2.3.2 Fungsi – Fungsi Keanggotaan

Didalam fuzzy system, fungsi keanggotaan memainkan peranan yang sangat penting untuk merepresentasikan masalah dan menghasilkan keputusan yang akurat. Terdapat banyak sekali fungsi keanggotaan yang biasa digunakan. Disini hanya membahas empat fungsi keanggotaan yang sering digunakan di dunia nyata, yaitu :[3].

1. Fungsi sigmoid

Sesuai dengan namanya, fungsi ini berbentuk kurva sigmoidal seperti huruf S. Setiap nilai x (anggota crisp set) dipetakan ke dalam interval [0,1]. Grafik dan notasi matematika untuk fungsi sigmoid dapat dilihat pada Gambar 2.2.

$$\text{Sigmoid (a,b,c)} = \begin{cases} 0, x \le a \\ 2\left(\frac{x-a}{c-a}\right)^2, a < x \le b \\ 1 - 2\left(\frac{x-a}{c-a}\right)^2, b < x \le c \\ 1, c < x \end{cases} \quad (2.1)$$

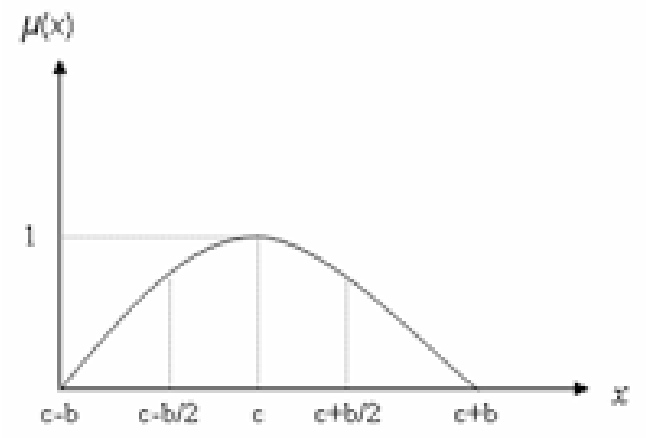


Gambar 2.2 Grafik dan notasi fungsi sigmoid [3].

2. Fungsi phi

Pada fungsi keanggotaan ini, hanya terdapat satu nilai x yang memiliki derajat keanggotaan yang sama dengan 1, yaitu ketika x=c. Nilai-nilai di sekitar c memiliki derajat keanggotaan yang masih mendekati 1. Grafik dan notasi matematika untuk fungsi phi dapat dilihat pada Gambar 2.3.

$$\text{Phi}(x,b,c) = \begin{cases} Phi\left(x, c-b, c-\frac{b}{2}, c\right), x \leq c \\ 1 - Phi\left(x, c, c+\frac{b}{2}, c+b\right), x > c \end{cases} \quad (2.2)$$

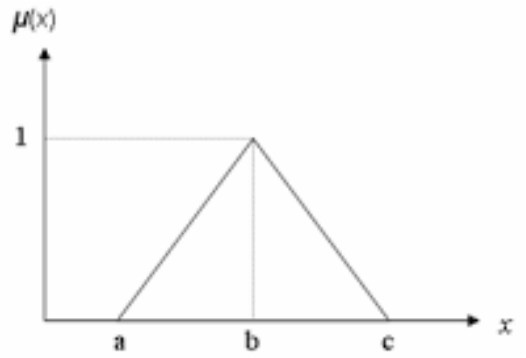


Gambar 2.3 Grafik dan notasi fungsi phi [3].

3. Fungsi segitiga

Sama seperti fungsi phi, pada fungsi ini juga terdapat hanya satu nilai x yang memiliki derajat keanggotaan sama dengan 1, yaitu ketika x=b. Tetapi, nilai-nilai di sekitar b memiliki derajat keanggotaan yang turun cukup tajam menjauhi 1. Grafik dan notasi matematika untuk fungsi segitiga dapat dilihat pada Gambar 2.4.

$$\text{Segitiga}(x,a,b,c) = \begin{cases} 0, x \leq a, x \geq c \\ \frac{x-a}{b-a}, a < x \leq b \\ -\frac{x-c}{c-b}, b < x \leq c \end{cases} \quad (2.3)$$

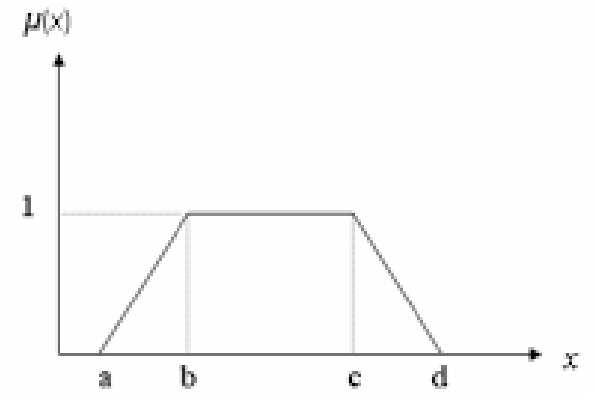


Gambar 2.4 Garfik dan notasi fungsi segitiga [3].

4. Fungsi trapesium

Berbeda dengan fungsi segitiga, pada fungsi ini terdapat beberapa nilai x yang memiliki derajat keanggotaan sama dengan 1, yaitu ketika b≤ x ≤ c. Tetapi derajat keanggotaan untuk a< x <b dan c< x ≤ d memiliki karakteristik yang sama dengan fungsi segitiga. Grafik dan notasi matematika untuk fungsi ini dapat dilihat pada Gambar 2.5.

Trapesium *(x,a,b,c,d)*=. $\begin{cases} 0, x \leq a, x \geq d \\ \frac{x-a}{b-a}, a < x < d \\ 1, b \leq x \leq c \\ -\frac{x-d}{d-c}, c < c \leq d \end{cases}$ (2.4)



Gambar 2.5 Garfik dan notasi fungsi trapesium [3].

Himpunan fuzzy memiliki 2 atribut, yaitu:

1. Linguistik, yaitu penamaan suatu grup yang mewakili suatu keadaan atau kondisi tertentu dengan menggunakan bahasa alami, seperti: MUDA, PAROBAYA, TUA
2. Numeris, yaitu suatu nilai (angka) yang menunjukkan ukuran dari suatu variable seperti: 40, 25, 50, dsb.

2.3.3 Variabel Linguistik

Variabel linguistik adalah suatu interval numerik dan mempunyai nilai-nilai linguistik , yang semantiknya di definisikan oleh fungsi keanggotaannya. Misalnya, *suhu* adalah suatu variabel linguistik yang bisa di definisikan pada interval [-10°C, 40°C]. Variabel tersebut bisa memiliki nilai – nilai linguistik seperti ‘Dingin’, ‘Hangat’, ‘Panas’ yang semantiknya di definisikan oleh fungsi – fungsi keanggotaan tertentu.

1. Fuzzyfication

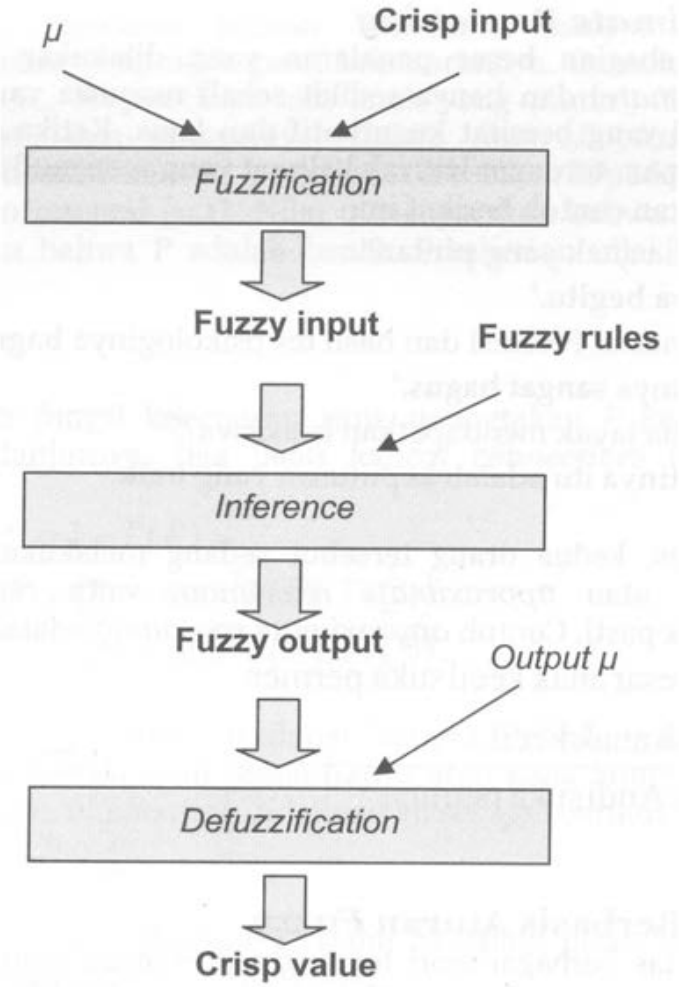
   Mengubah masukan – masukan yang nilai kebenarannya bersifat pasti (*Crips input*) kedalam bentuk fuzzy input, yang berupa nilai linguistik yang semantiknya ditentukan berdasarkan fungsi keanggotaan tertentu.

2. Inference

   Melakukan penalaran menggunakan fuzzy infut dan fuzzy rules yang telah di tentukan sehingga menghasilkan fuzzy output.

3. Defuzzyfication

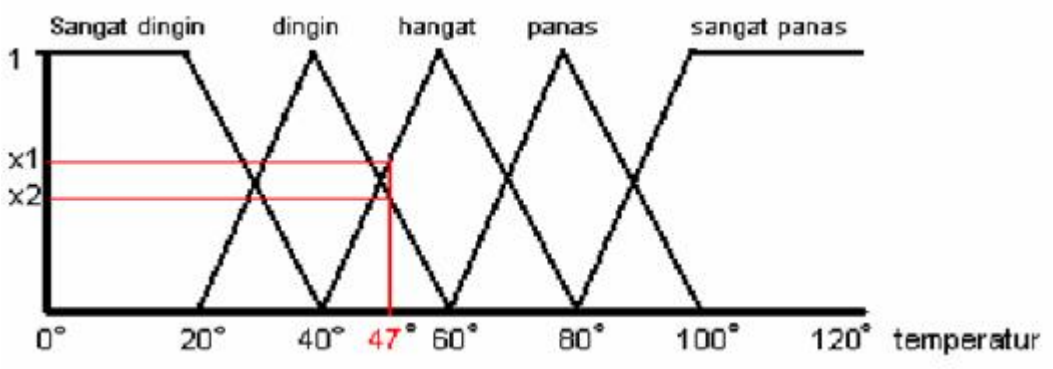
   Mengubah fuzzy output menjadi crisp value berdasarkan fungsi keanggotaan yang telah ditentukan.



Gambar 2.6 diagram blok untuk sistem berbasis aturan fuzzy [3].

2.3.4 Fuzzyfikasi

Fuzzifikasi yaitu suatu proses untuk mengubah suatu masukan dari bentuk tegas (crisp) menjadi fuzzy (variabel linguistik) yang biasanya disajikan dalam bentuk himpunan-himpunan fuzzy dengan suatu fungsi kenggotaannya masing-masing. Contoh dari proses Fuzzification adalah seperti yang ditunjukkan di gambar 2.2. Sebuah sistem fuzzy untuk mengukur suhu mempunyai 5 buah membership function yang mempunyai label sangat dingin, dingin, hangat, panas, sangat panas. Kemudian input yang diperoleh dari crisp input adalah 47° maka pengambilan fuzzy input-nya adalah seperti pada gambar.



Gambar 2.7 Proses perubahan dari crisp input menjadi fuzzy input [3].

2.3.5 Inference

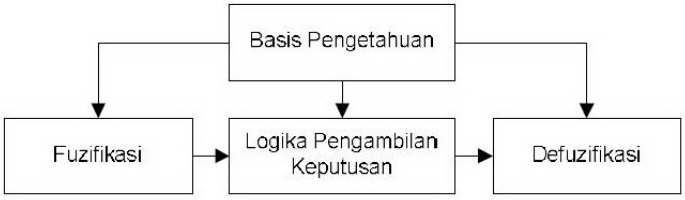
Dalam suatu sistem aturan fuzzy, proses inference memperhitunkan semua aturan yang ada dalam basis pengetahuan. Hasil dari proses inference direpresentasikan oleh suatu fuzzy set untuk setiap variabel bebas (pada consequent). Derajat keanggotaan untuk setiap nilai variabel tidak bebas menyatakan ukuran kompatibilitas terhadap variabel bebas (pada antecdent). Misalkan, terdapat suatu sistem dengan *n* variabel *x1, ..., xn* dan *m* variabel tidak bebas y1,...,ym. Misalkan *R* adalah suatu basis dari sejumlah *r* aturan fuzzy.

IF *P1*(x1,...,*xn*) THEN *Q1* (*y1*,...,*ym*),

IF *Pr*(x1,...,*xn*) THEN *Qr* (*y1*,...,*ym*),

Dimana *p1*,...*pr* menyatakan fuzzy predicate untuk variabel bebas, dan *Q1*,...*Qr* menyatakan fuzzy predicate untuk variabel tidak bebas.

Struktur sistem inferensi *fuzzy* dapat dilihat pada gambar 2.1



Gambar 2.8 Struktur sistem inferensi *fuzzy* [3].

Keterangan:

1) Basis Pengetahuan *Fuzzy* merupakan kumpulan *rule-rule fuzzy* dalam bentuk pernyataan IF…THEN.
2) *Fuzzyfikasi* adalah proses untuk mengubah input sistem yang mempunyai nilai tegas menjadi variabel linguistic menggunakan fungsi keanggotaan yang disimpan dalam basis pengetahuan *fuzzy*.
3) Logika pengambil keputusan merupakan proses untuk mengubah input *fuzzy* dengan cara mengikuti aturan-aturan (IF-THEN Rules) yang telah ditetapkan pada basis pengetahuan *fuzzy*.
4) *Defuzzyfikasi* merupakan proses mengubah output *fuzzy* yang diperoleh dari mesin inferensi menjadi nilai tegas menggunakan fungsi keanggotaan yang sesuai dengan saat dilakukan *fuzzyfikasi*.

Terdapat dua model aturan fuzzy yang digunakan secara luas dalam berbagai aplikasi.

1. Model Mamdani

Pada model ini, aturan fuzzy didefinisikan sebagai:

IF x_1 is A_1 AND …AND x_n is A_n THEN y is B

di mana A_1, …, A_n, dan B adalah nilai-nilai linguistik (atau fuzzy set) dan "x_1 is A_1" menyatakan bahwa nilai x_1 adalah anggota fuzzy set A_1.

2. **Model Sugeno**

Model ini dikenal juga sebagai Takagi-Sugeno-Kang (TSK) model, yaitu suatu varian dari Model Mamdani.

Model ini menggunakan aturan yang berbentuk:

IF x_1 is A_1 AND…AND x_n is A_n THEN $y=f(x_1,\ldots,x_n)$

di mana f bisa sembarang fungsi dari variabel-variabel input yang nilainya berada dalam interval variabel output. Biasanya, fungsi ini dibatasi dengan menyatakan f sebagai kombinasi linier dari variabel-variabel input:

$f(x_1,\ldots,x_n) = w_0 + w_1x_1 + \ldots + w_nx_n$

di mana w_0, w_1,…,w_n adalah konstanta yang berupa bilangan real yang merupakan bagian dari spesifikasi aturan fuzzy.

2.3.6 Defuzzyfikasi

Defuzzification: mengubah fuzzy output menjadi crisp value berdasarkan fungsi keanggotaan yang telah ditentukan.Terdapat berbagai metode defuzzification yang telah berhasil diaplikasikan untuk berbagai macam masalah, di sini dibahas 5 metode di antaranya, yaitu:[2]

1. Centroid method

Metode ini disebut juga sebagai Center of Area atau Center of Gravity. Metode ini menghitung nilai crisp menggunakan rumus:

$$y^* = \frac{\int y\mu_R(y)dy}{\int \mu_R(y)dy}$$

di mana y* suatu nilai crisp.

Fungsi integration dapat diganti dengan fungsi summation jika y bernilai diskrit, sehingga menjadi:

$$y^* = \frac{\sum y\mu_R(y)}{\sum \mu_R(y)}$$

di mana y* adalah nilai crisp dan $\mu_R(y)$ adalah derajat keanggotaan dari y.

2. Height method

Metode ini dikenal sebagai prinsip keanggotaan maksimum karena metode ini secara sederhana memilih nilai crisp yang memiliki derajat keanggotaan maksimum. Oleh karena itu, metode ini hanya bisa dipakai untuk fungsi keanggotaan yang memiliki derajat keanggotaan 1 pada suatu nilai crisp tunggal dan dan 0 pada semua nilai crisp yang lain. Fungsi seperti ini sering disebut sebagai singleton.

3. First (or Last) of Maxima

Metode ini juga merupakan generalisasi dari height method untuk kasus di mana fungsi keanggotaan output memiliki lebih dari satu nilai maksimum.Sehingga nilai crisp yang digunakan adalah salah satu dari nilai yang dihasilkan dari maksimum pertama atau maksimum terakhir (tergantung pada aplikasi yang akan dibangun).

4. Mean-Max Method

Metode ini disebut juga sebagai Middle of Maxima. Merupakan generalisasi dari height method untuk kasus di mana terdapat lebih dari satu nilai crisp yang memiliki derajat keanggotaan maksimum.Sehingga y* didefinisikan sebagai titik tengah antara nilai crisp terkecil dan nilai crisp terbesar

$$y^* = \frac{m + M}{2}$$

di mana m adalah nilai crisp yang paling kecil dan M adalah nilai crisp yang paling besar.

5. Weighted Average

Metode ini mengambil nilai rata-rata dengan menggunakan pembobotan berupa derajat keanggotaan. Sehingga y* didefinisikan sebagai:

$$y^* = \sum \frac{\mu(y)y}{\mu(y)}$$

di mana y adalah nilai crisp dan μ(y) adalah derajat keanggotan dari nilai crisp y.

2.4 Pemrograman Berorientasi Objek

Objek adalah kesatuan entitas yang memiliki sifat dan tingkah laku. Dalam kehidupan sehari-hari, objek adalah benda, baik benda berwujud nyata seperti manusia, hewan, mobil, komputer, handphone, pena, ataupun benda yang tidak nyata atau konsep, seperti halnya tabungan bank, sistem antrian, sistem internet banking, dan sebagainya. Jadi pengertian OOP adalah konsep yang membagi program menjadi objek-objek yang saling berinteraksi satu sama lain.

Objek adalah benda, baik benda yang berwujud nyata maupun benda yang tidak nyata (konsep). Jika menggunakan OOP maka akan ada enam keuntungan yang dapat diperoleh, yaitu [6]:

1. Alami (*Natural*).
2. Dapat diandalkan (*Reliable*).
3. Dapat digunakan kembali (*Reusable*).
4. Mudah untuk dalam perawatan (*Maintainable*).
5. Dapat diperluas (Extendable).
6. Efisiensi waktu.

Berikut ini beberapa bahasa pemrograman yang sudah menggunakan konsep OOP, adalah :

1. *C++*.
2. *Visual C++*.
3. *Visual Basic*.
4. *Java*.

2.4.1 Objek *(Object)*

Orientasi objek merupakan teknik dalam menyelesaikan masalah yang kerap muncul dalam pengembangan perangkat lunak. Teknik ini merupakan titik kulminasi dalam menemukan cara yang efektif dalam membangun sistem dan menjadi metode yang paling banyak dipakai oleh para pengembang perangkat lunak saat ini. Orientasi objek merupakan teknik pemodelan sistem riil yang berbasis objek. Inti dari konsep ini adalah objek yang merupakan model dari sistem nyata.

Objek adalah entitas yang memiliki atribut, karakter dan kadangkala disertai kondisi. Objek merepresentasikan sesuatu sistem nyata seperti siswa, sistem kontrol permukaan sayap pesawat, sensor atau mesin. Objek juga merepresentasikan sesuatu dalam bentuk konsep seperti nasabah bank, merek dagang, pernikahan atau sekedar *listing*. Bahkan bisa juga mengatakan visualisasi seperti, bentuk huruf, histogram, poligon, garis atau lingkaran. Semuanya memiliki fitur atribut (untuk data), *behavior* (*operation* atau *method*), keadaan

(memori), identitas dan tanggung jawab. Proses menjabarkan sistem nyata menjadi objek dinamakan abstraksi *(abstraction)*. Abstraksi mengeliminir aspek yang tidak perlu dalam suatu objek.

2.4.2 Kelas *(Class)*

Kelas adalah penggambaran satu set objek yang memiliki atribut dan *behaviour* yang sama. Kelas mirip tipe data pada pemrograman non objek, tapi lebih komprehensif karena terdapat struktur sekaligus karakteristiknya. Programmer dapat membentuk kelas baru yang lebih spesifik dari kelas *general*-nya. Kelas dan objek merupakan jantung dari pemrograman berorientasi objek. Untuk menghasilkan program jenis ini sangat penting untuk selalu berfikir dalam bentuk objek.

2.4.3 Pembungkusan *(Encapsulation)*

Pembungkusan sebagai penggabungan potongan-potongan informasi dan perilaku-perilaku spesifik yang bekerja pada informasi tersebut, kemudian mengemasnya menjadi apa yang disebut sebagai objek. Dalam perbankan dikenal objek rekening yang memiliki perilaku-perilaku misalnya buka, tutup, penarikan, penyimpanan, ubah nama, ubah alamat, dan sebagainya. Akibatnya, perubahan-perubahan pada sistem perbankan yang berkaitan dengan rekening-rekening dapat secara sederhana diimplementasikan satu kali saja pada objek rekening. Keuntungan lainnya adalah membatasi efek-efek perubahan pada sistem. Misalnya, saat manajemen bank menentukan jika seseorang memiliki rekening pinjaman di bank yang bersangkutan, rekening pinjaman itu harus dapat juga digunakan sebagai sarana bagi penarikan rekening.

2.4.4 Pewarisan *(Inheritance)* dan Generalisasi/Spesialisasi

Konsep dimana metode dan atau atribut yang ditentukan di dalam sebuah objek kelas dapat diwariskan atau digunakan lagi atau digunakan lagi oleh objek kelas lainnya. Sedangkan generalisasi/spesialisasi merupakan teknik dimana atribut dan perilaku yang umum pada beberapa tipe kelas objek, dikelompokkan

(atau diabstraksi) ke dalam kelasnya sendiri (dinamakan *supertype*). Atribut dan metode kelas objek *supertype* kemudian diwariskan oleh kelas objek tersebut (dinamakan *subtype*).

2.4.5 *Polimorfisme*

Polimorfisme berarti suatu fungsionalitas yang diimplementasikan dengan berbagai cara yang berbeda. Pada terminologi berorientasi objek, ini berarti dapat memiliki berbagai implementasi untuk sebagian fungsionalitas tertentu. Sebagai contoh, misalkan pengembang akan mengembangkan sistem berbasis grafis. Saat pengguna mau menggambar sesuatu, misalnya garis atau lingkaran, sistem akan memunculkan perintah gambar. Sistem akan mengenali berbagai bentuk gambar, masing-masing dengan perilakunya sendiri-sendiri. Manfaat dari *polimorfisme* adalah kemudahan pemeliharaannya. Jika perlu menambahkan gambar baru (misalnya segitiga) maka cukup menambahkan fungsi baru (fungsi menggambar segitiga) sedangkan fungsi umumnya (fungsi gambar) tidak mengalami perubahan.

2.5 *UML (Unified Modeling Language)*

UML adalah keluarga notasi grafis yang didukung oleh meta-model tunggal, yang membantu pendeskripsian dan desain sistem perangkat lunak, khususnya sistem yang dibangun menggunakan pemrograman berorientasi objek. Definisi ini merupakan definisi sederhana. Pada kenyataannya, pendapat orang-orang tentang UML berbeda satu sama lain. Hal ini dikarenakan oleh sejarahnya sendiri dan oleh perbedaan persepsi tentang apa yang membuat sebuah proses rancang bangun perangkat lunak efektif.

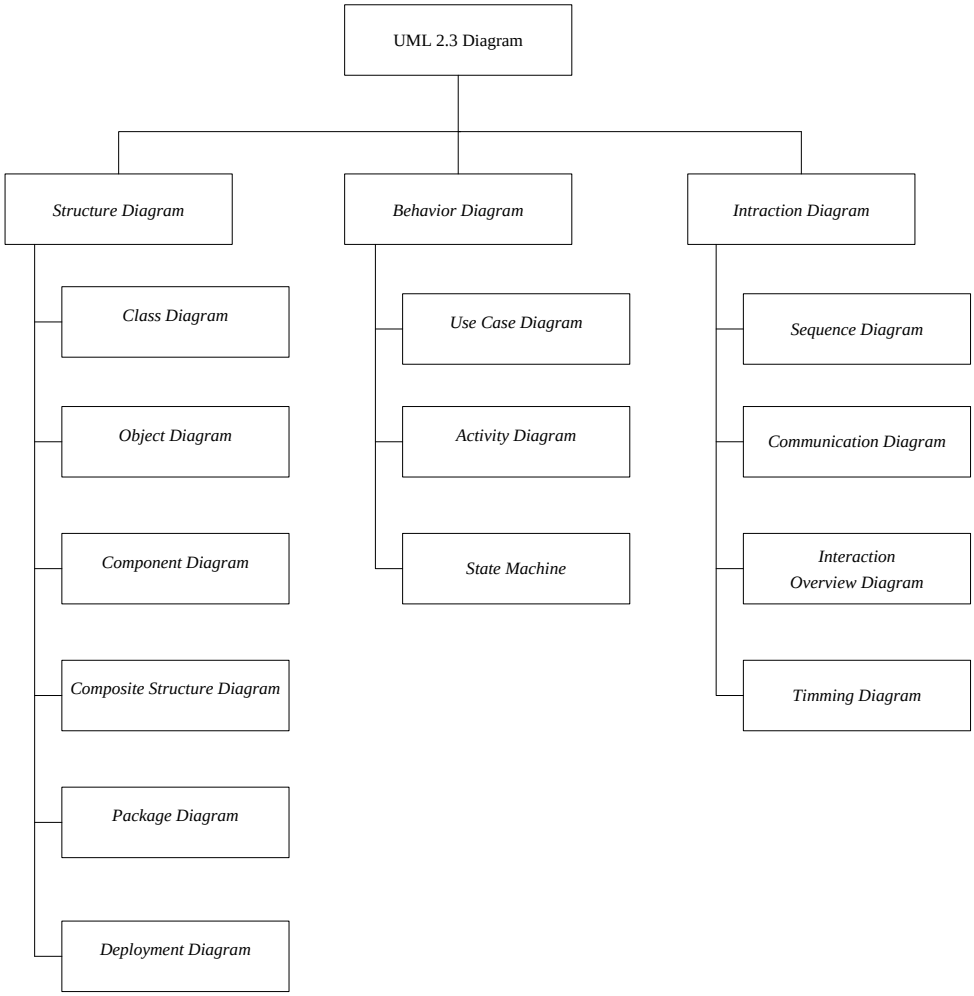
UML merupakan standar yang relatif terbuka yang dikontrol oleh O*bject Management Group (OMG),* sebuah konsorsium terbuka yang terdiri dari banyak perusahaan. OMG dibentuk untuk membuat standar-standar yang mendukung interoperabilitas, khususnya interoperabilitas sistem berorientasi objek. OMG mungkin lebih dikenal dengan standar-standar *COBRA (Common Object Request Broker Architecture).*

UML lahir dari penggabungan banyak bahasa pemodelan grafis berorientasi objek yang berkembang pesat pada akhir 1980-an dan awal 1990-an. Sejak kehadirannya pada tahun 1997, UML menggantikan menara Babel yang telah menjadi sejarah. UML merupakan dasar bagi perangkat (*tool*) desain berorientasi objek dari IBM.

Bagian-bagian utama dari UML adalah *view*, *diagram*, *model element*, dan *general mechanism* [7]. Diagram berbentuk grafik yang menunjukkan simbol elemen model yang disusun untuk mengilustrasikan bagian atau aspek tertentu dari sistem. Sebuah diagram merupakan bagian dari suatu view tertentu dan ketika digambarkan biasanya.

2.6 Diagram UML

Pada UML 2.3 terdiri dari 13 macam diagram yang dikelompokkan dalam 3 kategori. Pembagian kategori dan macam-macam diagram tersebut dapat dilihat pada gambar Gambar 2.2.



Gambar 2.9 Diagram UML [7]

Berikut ini penjelasan singkat dari pembagian kategori tersebut.

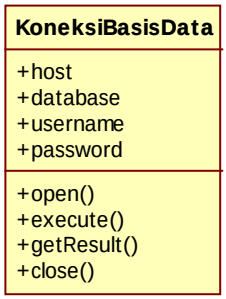
1. *Structure diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
2. *Behavior diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkain perubahan yang terjadi pada sebuah sistem.
3. *Interaction diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antarsubsistem pada suatu sistem.

2.6.1 Diagram Kelas *(Class Diagram)*

Diagram kelas menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi.

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

Diagram kelas dibuat agar pembuat program atau *programmer* membuat kelas-kelas sesuai rancangan di dalam diagram kelas agar antara dokumentasi perancangan dan perangkat lunak sinkron. Berikut adalah contoh dari diagram kelas.



Gambar 2.10 Contoh *Class Diagram* [7]

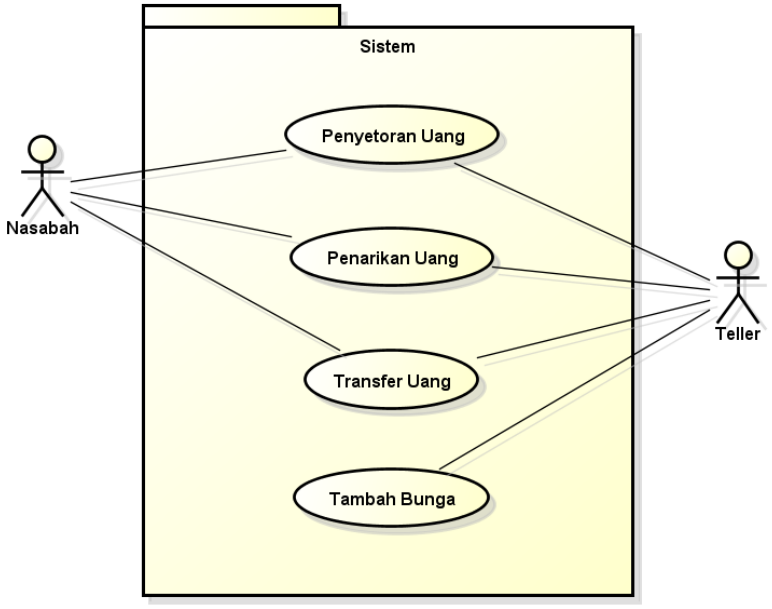
2.6.2 Diagram *Use Case*

Diagram *use case* merupakan pemodelan untuk kelakuan *(behaviour)* sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu. Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut aktor dan *use case*.

1. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan

dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.

2. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.



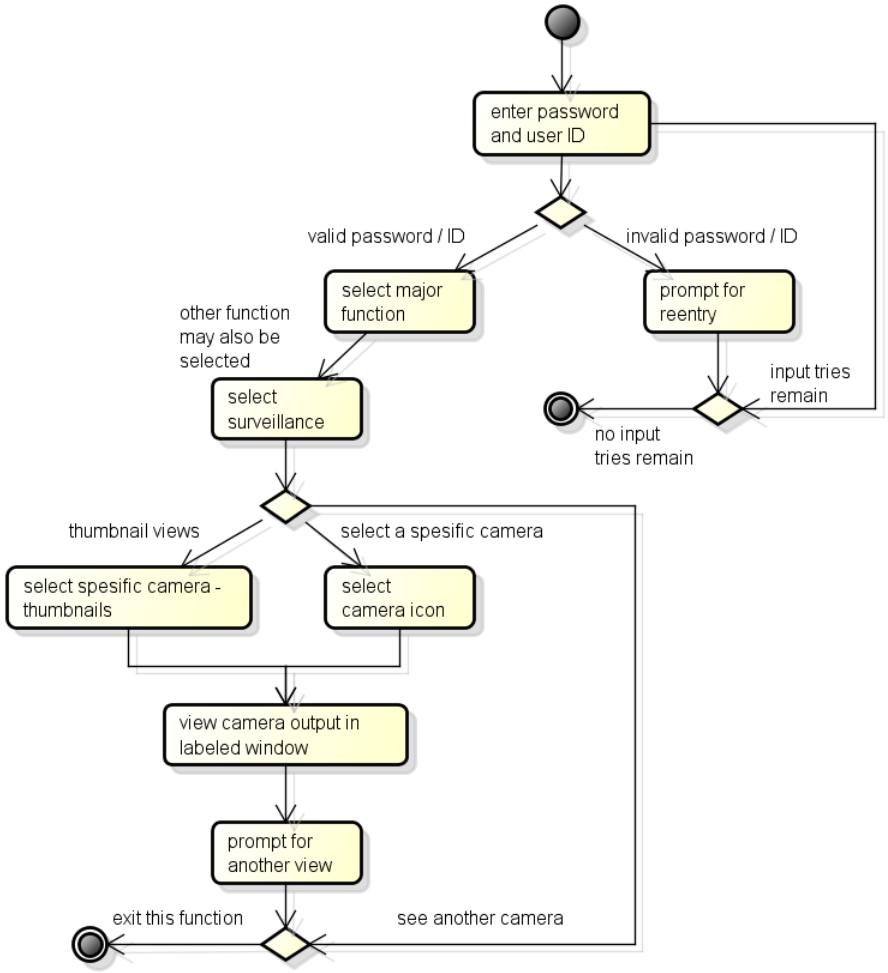
Gambar 2.11 Contoh *Use Case Diagram* [7]

2.6.3 Diagram Aktivitas *(Activity Diagram)*

Diagram aktivitas menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem. Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut:

1. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
2. Urutan atau pengelompokkan tampilan dari sistem/*user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
3. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.

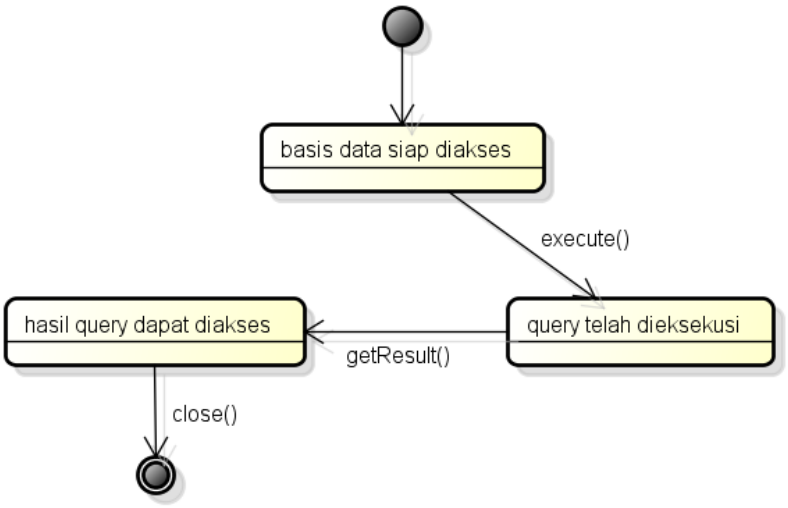
4. Rancangan menu yang ditampilkan pada perangkat lunak.



Gambar 2.12 Contoh *Activity Diagram* [7]

2.6.4 Diagram *State Machine*

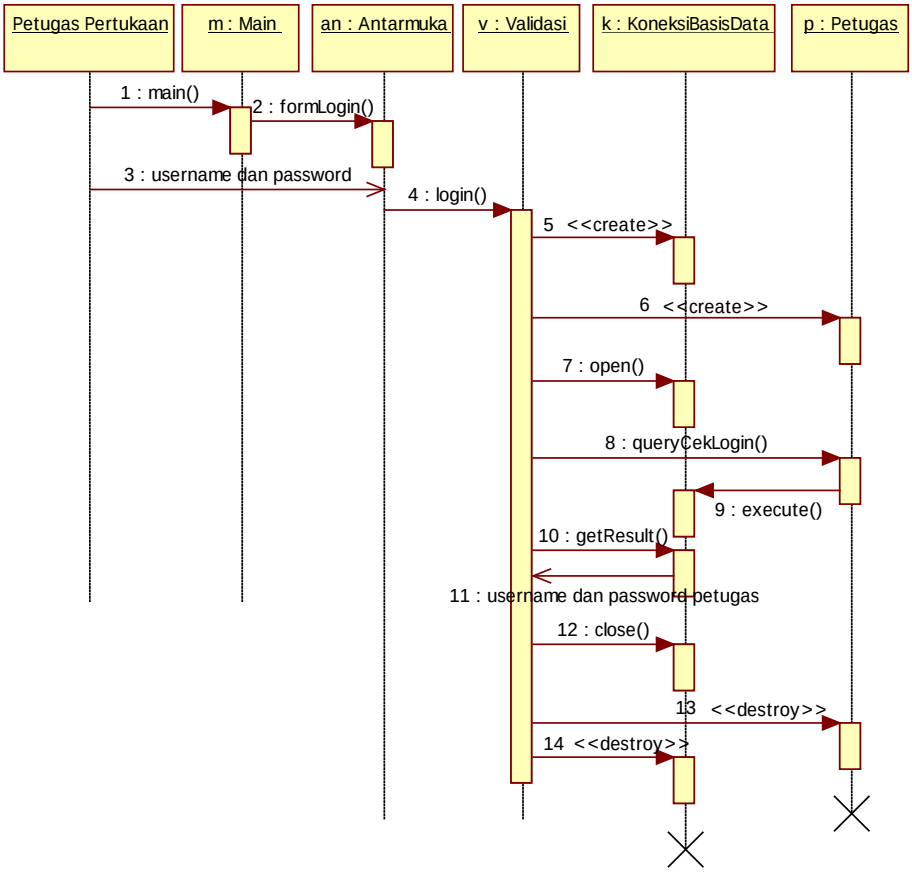
State machine diagram atau *statechart diagram* atau dalam bahasa Indonesia disebut diagram mesin status atau sering juga disebut diagram status digunakan untuk menggambarkan perubahan status atau transisi status dari sebuah mesin atau sistem atau objek. Jika diagram sekuen digunakan untuk interaksi antar objek maka diagram status digunakan untuk interaksi di dalam sebuah objek. Perubahan tersebut digambarkan dalam suatu graf berarah.



Gambar 2.13 Contoh *Statechart Diagram* [7]

2.6.5 Diagram Sekuen *(Sequence Diagram)*

Diagram sekuen menggambarkan kelakuan pada objek *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambar 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 pada diagram sekuen sehingga semakin banyak *use case* yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak.



Gambar 2.14 Contoh *Sequence Diagram* [7]