BAB 2
LANDASAN TEORI

2.1 Game

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

2.1.1 Pengertian *Game*

Game merupakan kata dalam bahasa inggris yang berarti permainan. Permainan adalah sesuatu yang dapat dimainkan dengan aturan tertentu sehingga ada yang menang dan ada yang kalah, biasanya dalam konteks tidak ada serius atau dengan tujuan hiburan. Suatu cara belajar yang digunakan dalam menganalisa interaksi antara sejumlah pemain maupun perorangan yang menunjukan strategi-strategi yang rasional.

Teori permainan pertama kali ditemukan oleh sekelompok ahli Matematika, yang berisi [6] :

"Permainan terdiri atas sekumpulan peraturan yang membangun situasi bersaing dari dua sampai beberapa orang atau kelompok dengan memilih strategi yang dibangun untuk memaksimalkan kemenangan sendiri atau pun untuk meminimalkan kemenangan lawan. Peraturan-peraturan memnentukan kemungkinan tindakan untuk setiap pemain, sejumlah keterangan diterima setiap pemain sebagai kemajuan bermain, dan sejumlah kemenangan atau kekalahan dalam berbagai situasi.

2.1.2 3D Grafis

Grafik komputer 3 dimensi biasa disebut 3D atau adalah bentuk dari benda yang memiliki panjang, lebar, dan tinggi. Grafik 3 Dimensi merupakan teknik penggambaran yg berpatokan pada titik koordinat sumbu x(datar), sumbu y(tegak), dan sumbu z(miring). Representasi dari data geometrik 3 dimensi

sebagai hasil dari pemrosesan dan pemberian efek cahaya terhadap grafika komputer 2D.

Tiga Dimensi, biasanya digunakan dalam penanganan grafis. 3D secara umum merujuk pada kemampuan dari sebuah video card (link). Saat ini video card menggunakan variasi dari instruksi-instruksi yang ditanamkan dalam video card itu sendiri (bukan berasal dari software) untuk mencapai hasil grafik yang lebih realistis dalam memainkan game komputer.

Grafik 3D merupakan perkembangan dari grafik 2D. Didalam grafika komputer, 3D merupakan bentuk grafik yang menggunakan representasi data geometri tiga dimensi.Suatu objek rangka 3D apabila disinari dari arah tertentu akan membentuk bayangan pada permukaan gambar. Proses pembuatan grafik komputer 3D dapat dibagi ke dalam tiga fase, yaitu 3D modeling yang mendeskripsikan bentuk dari sebuah objek, layout dan animation yang mendeskripsikan gerakan dan tata letak sebuah objek, dan 3D rendering yang memproduksi image dari objek tersebut. Istilah atau Pengertian Grafik 3D adalah sebuah gambar, garis, lengkungan dan sebagainya yang memiliki titik-titik yang menghubungkan menjadi sebuah bentuk 3D Di dalam dunia game, 3D secara umum merujuk pada kemampuan dari sebuah video card (link). Saat ini video card menggunakan variasi dari instruksi-instruksi yang ditanamkan dalam video card itu sendiri (bukan berasal dari *software*) untuk mencapai hasil grafik yang lebih realistis dalam memainkan game komputer.

2.1.3 Game 3D

Industri *game* selalu berusaha untuk mengikuti perkembangan teknologi yang ada di dunia ini. Ketika perangkat-perangkat komputer seperti *processor*, *graphic card* versi baru mulai muncul di pasaran, para *developer* game selalu berusaha mengikuti perkembangan tersebut.

Pada saat kemampuan proses pada komputer semakin cepat, para *developer* juga senantiasa menciptakan game yang semakin canggih sehingga muncul *engine* dengan grafik 3D (3 dimensi). Maka dari itu, game 3D dengan hitungan *polygon* yang sangat besar dan pencahayaan yang sudah canggih, juga tekstur *maping*

mulai diproduksi. Game 3D merepresentasikan objek dalam bentuk 3 dimensi sehingga objek akan terlihat lebih nyata seperti dalam kehidupan nyata.

Game bertipe 3 dimensi merupakan *game* dengan *grafis* yang baik dalam penggambaran secara realita, kebanyakan game-game ini memiliki perpindahan kamera *(angle)* hingga 360 derajat sehingga dapat melihat secara keseluruhan dunia *game* tersebut [3].

2.1.4 Jenis-jenis *Game*

Berikut ini beberapa jenis *game* berdasarkan cara pembuatannya,cara pemasarannya dan mesin yang menjalankannya. Jenis-jenis *game* tersebut adalah [7]:

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.

2.1.5 Genre *Game*

Berdasarkan 4 jenis *game*, dapat dibagi menjadi beberapa *genre game* yaitu [7] :

1. *Action Game*

Action game dikategorikan sebagai *game play* dengan model pertarungan. Berikut beberapa macam *game* yang termasuk dalam *genre action game* yaitu :

a. *Action Adventure Game*

Genre game yang berfokus pada eksplorasi dan biasanya mempunyai unsur *item gathering*, penyelesaiaan *puzzle simple* dan Pertarungan. Contoh *game* dari *genre* ini adalah Final Fantasy series dan Mario Bross series.

b. *Stealth Game*

Termasuk dalam *genre* terbaru, biasanya digolongkan dalam mata-mata yang biasa melakukan aksinya secara rahasia.Contoh *game* dari *genre* ini adalah Hitman Absolution.

c. *Survival Horror Game*

Genre game yang berusaha membuat pemain menjadi tegang dan takut dengan elemen-elemen horror.Contoh *game* dari *genre* ini adalah Resident Evil series dan Fatal Frame.

d. *Beat'em Up Game*

Genre game combat dimana satu orang melawan banyak musuh yang telah disediakan.Contoh *game* dari *genre* ini adalah Dynasty Warrior series dan The Warriors.

e. *Fighting Game*

Game pertarungan dua pemain dengan jurus-jurus yang biasa dikeluarkan dengan menekan beberapa tombol pada *keyboard* atau *Joystick* dengan urutan tertentu. Contoh *game* dari *genre* ini adalah Street Fighter dan Tekken series.

f. *Maze Game*

Genre game yang membutuhkan kecepatan berpikir dan bereaksi serta berunsur ketepatan menavigasi.Contoh *game* dari *genre* ini adalah Pac-Man.

g. *Platfrom Game*

Genre Game dengan *game play* berlari,melompat,mengayun dan sebagainya.Contoh *game* dari *genre* ini adalah Donkey Kong..

h. *Shooter Game*

1. *First Person Shooter Game*

Genre game yang mengutamakan *shooting* dan *combat* dari perspektif langsung mata karakter yan bertujuan untuk memberikan pemain perasaan berada ditempat itu dan bisa fokus menembak.

2. *Massively Multiplayer Online First Person Shooter Game*

Genre game yang mengkombinasikan *game play first person shooter* dengan dunia virtual dimana banyak *player* juga ikut bermain melaui internet.Contoh *game* dari *genre* ini adalah Counter Strike *Online*.

3. *Third Person Shooter Game*

Genre game yang sama seperti *first person shooter game* yaitu mengutamakan *shooting* dan *combat* dari perspektif karakter yang bertujuan untuk memberikan pemandangan yang lebih luas dan gerakan yang lebih banyak.

4. *Tactical Shooter Game*

Genre yang mengutamakan perencanaan dan kerja sama tim untuk memenangkan *game*.Contoh *game* dari *genre* ini adalah Tom Clancy‟s Ghost Recon series.

5. *Light Gun Game*

Genre dengan lebih banyak pada *arcade* dengan peralatan tertentu seperti senjata mainan yang mempunyai sensor khusus terhadap layar.Contoh *game* dari *genre* ini adalah Time Crisis dan

Duck Hunt.

6. *Shoot'em Up Game*

Genre dengan ciri khas gambar 2D dan *scrolling playing area*.Contoh *game* dari *genre* ini adalah Star Fox series.

2. Adventure Game

Adventure game dikategorikan sebagai *game play* yang mengharuskan pemain memecahkan bermacam-macam teka-teki melaui interaksi dengan orang lingkungan dalam *game* tersebut.

a. *Text Adventure*

Pemain akan menggunakan *keyboard* untuk mengetikkan berupa perintah dan komputer akan menganalisa perintah tersebut lalu menjalankan karakter sesuai perintah tersebut.

b. *Graphical Adventure Game*

Genre yang merupakan perkembangan dari *text adventure*.Pemain dapat menggunakan *mouse* untuk menggerakkan karakter.

c. *Visual Novel Game*

Genre yang memberikan keleluasaan untuk memilih jalan ceritanya sendiri.

d. *Interactive Movie Game*

Genre game dengan rangkaian *live action* dari karakter yang dimainkan pemain.Contoh *game* dari *genre* ini adalah Space Ace.

e. *Dialog Game*

Pada *genre* ini,pemain akan mengalami kemajuan tergantung pada apa yang mereka katakan.Contoh *game* dari *genre* ini adalah Walking Dead.

3. Role Playing Game

Role playing game adalah *game* yang memiliki *game play* dimana karakter *player* akan berpetualang dengan *skill combat* dalam cerita *game*.

a. Action Role Playing Game

Genre game yang memasukkan unsur *action game* dan *action*

adventure game.Contoh *game* dari *genre* ini adalah Diablo 1 & 2.

b. *Massively Multyplayer Online Role Playing Game*

Konsep dari *genre* ini terkombinasi dengan *genre-genre* lainnya yang berupa fantasi.Contoh *game* dari *genre* ini adalah Rising Force *Online*.

c. *Tactical Role Playing Game*

Dalam *genre* ini,pemain akan diberikan giliran masing-masing untuk menentukan langkah-langkah yang akan dilakukan oleh karakter. Contoh *game* dari *genre* ini adalah Final Fantasy Tactics.

4. Simulation Game

Genre ini bertujuan untuk memberikan pengalaman simulasi kepada pemain.

a. *Construction and Management Simulation Game*

Genre ini merupakan bagian dari *economi simulation game*. Contoh *game* dari *genre* ini adalah Sims City series.

b. *Economic Simulation Game*

Genre ini berupa simulasi keadaan ekonomi dimana pemain mengontrol keadaan ekonomi dari *game* tersebut.Contoh dari *genre* ini adalah Monopoly Tycoon.

c. *God Game*

Dalam *genre* ini tidak ada tujuan akhir yang membuat pemain memenangakan *game*.Contoh *game* dari *genre* ini adalah The Sims series.

d. *Government Simulation Game*

Genre game yang memasukkan unsur kepolisian,pemerintahan atau politik sebuah negara.

5. *Strategy Game*

Genre strategy game berfokus pada *game play* dimana dibutuhkan pemikiran yang tepat agar dapat meraih kemenangan.

a. *Real Time* Strategi

Dalam *real time* strategi,*action* dilakukan dalam waktu yang bersamaan oleh masing-masing pihak dimana *action* dimainkan per *ronde* atau bergiliran.Contoh *game* dari *genre* ini adalah Warcraft series.

b. *Tactical Game*

Dalam *genre* ini pemain harus menggunakan bermacam-macam taktik dan strategi untuk mencapai kemenangan.Contoh *game* dari *genre* ini adalah Dark Omen.

c. *4X Game*

Genre ini berarti penjelajahan,menjajah dan memusnahkan.Contoh *game* dari *genre* ini adalah Galactic Civizations.

d. *Artillery Game*

Genre game ini biasanya mengikutkan *combat* dengan tank atau tentara militer.Contoh *game* dari *genre* ini adalah Tanarus.

6. *Vehicle Simulation Game*

Genre ini merupakan simulasi yang memberikan pemain sebuah pengalaman realistik dalam mengendarai kendaraan tertentu.

a. *Flight Game*

Dalam *genre* ini,pemain tidak hanya bersimulasi mengontrol pesawat terbang tetapi juga bisa *combat* di udara.Contoh *game* dari *genre* ini adalah Falcon 4.0.

b. *Racing Game*

Genre yang menempatkan pemain sebagai *driver* dengan kendaraan seperti mobil.Contoh *game* dari *genre* ini adalah Need For Speed series.

c. *Space Game*

Genre ini bersifat pertarungan di angkasa luar. Contoh *game* dari genre ini adalah Star Wars dan Homeworld.

d. *Train Game*

Genre ini mensimulasikan yang berhubungan dengan trasnportasi kereta. Contoh dari *genre* ini adalah Rail Simulator.

2.1.6 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 Artificial Intelligence (AI)

Kecerdasan buatan berasal dari bahasa Inggris "*Artificial Intelligence*" atau disingkat AI, yaitu *intelligence* adalah kata sifat yang berarti cerdas, sedangkan *artificial* artinya buatan. Kecerdasan yang dimaksud disini merajuk pada mesin yang mampu berpikir, menimbang tindakan yang diambil, dan mampu mengambil keputusan seperti yang dilakukan oleh manusia.

Menurut Prof. Elder Dadios [3] kecerdasan buatan adalah bagian dari ilmu komputer yang mempelajari bagaimana membuat mesin (komputer) dapat melakukan pekerjaan seperti manusia bahkan bisa lebih baik dibandingkan manusia. Agar mesin (komputer) bisa cerdas maka harus diberi bekal pengetahuan dan mempunyai kemampuan untuk menalar. Ada dua bagian utama yang dibutuhkan aplikasi, kecerdasan buatan yaitu :

1. Basis Pengetahuan (Knowledge Base) adalah berisi fakta-fakta, teori, pemikiran dan hubungan satu dengan lainnya.
2. Motor Inferensi (Inference Engine) adalah kemampuan menarik kesimpulan berdasarkan pengetahuan.

Kecerdasan buatan mempunyai kelebihan dibanding dengan kecerdasan alami antara lain :

1. Lebih bersifat permanen. Sebab kecerdasan buatan tidak berubah selama sistem komputer dan program tidak mengubahnya.
2. Lebih mudah diduplikasi dan disebarkan.
3. Lebih murah, karena menyediakan layanan komputer yang lebih mudah dan murah dibandingkan mendatangkan seseorang untuk mengerjakan sejumlah pekerjaan dalam jangka waktu yang lama.
4. Bersifat konsisten dan teliti.
5. Dapat mengerjakan task lebih cepat.

2.2.1 Lingkup utama kecerdasan buatan

Lingkup utama kecerdasan buatan, yaitu [3] :

a. Sistem pakar. Komputer digunakan sebagai saran untuk menyimpan pengetahuan para pakar. Dengan demikian komputer akan memiliki keahlian untuk menyelesaikan masalah dengan meniru keahlian yang dimiliki para pakar
b. Pengolahan bahasa alami. Dengan pengolahan bahasa alami ini diharapkan user mampu berkomunikasi dengan komputer dengan menggunakan bahasa sehari-hari.
c. Pengenalan ucapan. Melalui pengenalan ucapan diharapkan manusia

mampu berkomunikasi dengan komputer dengan menggunakan suara.

d. Robotika dan Sistem sensor
e. *Computer vision*, mencoba untuk dapat mengintrepetasikan gambar atau objek-objek tampak melalui komputer

Intelligent Computer aid Instruction. Komputer dapat digunakan sebagai tutor yang dapat melatih dan mengajar.

2.2.2 Kelebihan Kecerdasan Buatan

Keuntungan kecerdasan buatan, yaitu [4] :

1. Kecerdasan buatan lebih bersifat permanen. Kecerdasan alami akan cepat mengalami perubahan. Hal ini dimungkinkan karena sifat manusia yang pelupa. Kecerdasan buatan tidak akan berubah sepanjang sistem komputer dan program tidak mengubahnya.
2. Kecerdasan buatan lebih mudah diduplikasi dan disebarkan. Mentransfer pengetahuan manusia dari satu orang ke orang lain butuh proses dan waktu lama. Disamping itu suatu keahlian tidak akan pernah bisa diduplikasi secara lengkap. Sedangkan jika pengetahuan terletak pada suatu sistem komputer, pengetahuan tersebuat dapat ditransfer atau disalin dengan mudah dan cepat dari satu komputer ke komputer lain
3. Kecerdasan buatan lebih murah dibanding dengan kecerdasan alami. Menyediakan layanan komputer akan lebih mudah dan lebih murah dibanding dengan harus mendatangkan seseorang untuk mengerjakan sejumlah pekerjaan dalam jangka waktu yang sangat lama.
4. Kecerdasan buatan bersifat konsisten. Hal ini disebabkan karena kecerdasan buatan adalah bagian dari teknologi komputer. Sedangkan kecerdasan alami senantiasa berubah-ubah.
5. Kecerdasan buatan dapat didokumentasikan. Keputusan yang dibuat komputer dapat didokumentasikan dengan mudah dengan melacak setiap aktivitas dari sistem tersebut. Kecerdasan alami sangat sulit untuk direproduksi.
6. Kecerdasan buatan dapat mengerjakan pekerjaan lebih cepat dibanding

dengan kecerdasan alami

7. Kecerdasan buatan dapat mengerjakan pekerjaan lebih baik dibanding dengan kecerdasan alami.

2.2.3 Implementasi kecerdasan buatan

Implementasi kecerdasan buatan dapat diterapkan pada beberapa bidang, sebagai berikut [4] :

a. Visualisasi komputer

   Kecerdasan buatan pada bidang visualisasi komputer ini memungkinkan sebuah sistem komputer mengenali gambar sebagai *input*. Contohnya mengenali sebuah pola pada suatu gambar.

b. Pengenalan Suara

   Kecerdasan buatan pada pengenalan suara ini dapat mengenali suara manusia. Cara mengenali suara ini dengan mencocokannya pada acuan yang telah diprogramkan terlebih dahulu. Contohnya perintah komputer dengan menggunakan suara *user*.

c. Sistem Pakar

   Kecerdasan buatan pada Sistem Pakar ini memungkinkan sebuah sistem komputer memiliki cara berpikir dan penalaran seorang ahli dalam mengambil keputusan, untuk memecahkan masalah yang ada pada saat itu. Contohnya program komputer yang dapat mendiagnosa penyakit dengan memasukan gejala-gejala yang dialami pasien.

d. Permainan

   Kecerdasan buatan pada permainan ini memungkinkan sebuah sistem komputer untuk memiliki cara berpikir manusia dalam bermain. Contohnya permainan yang memiliki fasilitas orang melawan komputer. Komputer sudah di program sedemikian rupa agar memiliki cara bermain seperti seorang manusia bahkan bisa melebihi seorang manusia.

2.2.4 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 [3].

2.3 Algoritma Fuzzy Logic

Logika Fuzzy adalah suatu cara yang tepat untuk memetakan suatu ruang input ke dalam ruang output. Untuk sistem yang sangat rumit, penggunaan logika fuzzy (fuzzy logic) adalah salah satu pemecahannya. Sistem tradisional dirancang untuk mengontrol keluaran tunggal yang berasal dari beberapa masukan yang tidak saling berhubungan. Karena ketidaktergantungan ini, penambahan masukan yang baru akan memperumit proses kontrol dan membutuhkan proses perhitungan kembali dari semua fungsi. Kebalikannya, penambahan masukan baru pada sistem fuzzy, yaitu sistem yang bekerja berdasarkan prinsip-prinsip logika fuzzy, hanya membutuhkan penambahan fungsi keanggotaan yang baru dan aturan-aturan yang berhubungan dengannya [5].

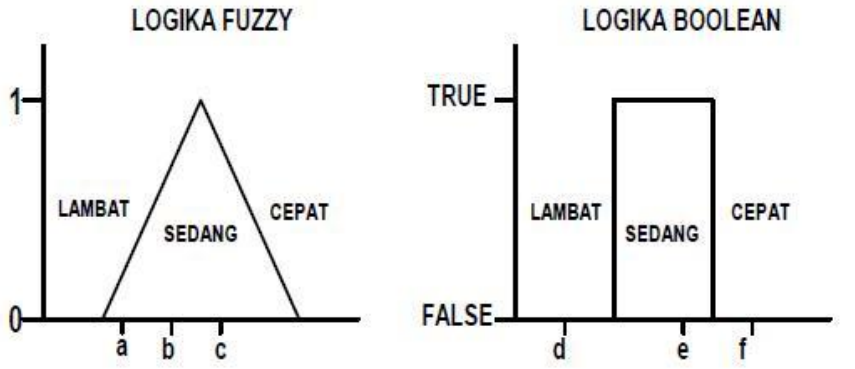
Secara umum, sistem fuzzy sangat cocok untuk penalaran pendekatan terutama untuk sistem yang menangani masalah-masalah yang sulit didefinisikan dengan menggunakan model matematis Misalkan, nilai masukan dan parameter sebuah sistem bersifat kurang akurat atau kurang jelas, sehingga sulit mendefinisikan model matematikanya.

Sistem fuzzy mempunyai beberapa keuntungan bila dibandingkan dengan sistem tradisional, misalkan pada jumlah aturan yang dipergunakan. Pemrosesan awal sejumlah besar nilai menjadi sebuah nilai derajat keanggotaan pada sistem fuzzy mengurangi jumlah nilai menjadi sebuah nilai derajat keanggotaan pada sistem fuzzy mengurangi jumlah nilai yang harus dipergunakan pengontrol untuk membuat suatu keputusan. Hal ini disebabkan karena sistem fuzzy mempunyai kemampuan untuk memberikan respon berdasarkan informasi yang bersifat kualitatif, tidak akurat, dan ambigu.

Sistem fuzzy pertama kali diperkenalkan oleh Prof. L. A. Zadeh [5]. Sistem ini mempunyai kemampuan untuk mengembangkan sistem intelijen dalam lingkungan yang tak pasti. Sistem ini menduga suatu fungsi dengan logika fuzzy. Dalam logika fuzzy terdapat beberapa proses yaitu penentuan himpunan fuzzy, penerapan aturan *IF-THEN* dan proses inferensi fuzzy

2.3.1 Himpunan Fuzzy

Himpunan fuzzy merupakan suatu pengembangan lebih lanjut tentang Konsep himpunan 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 2.1 dibawah ini [5]:



Gambar 2.1 Pendefinisian kecepatan dalam bentuk fuzzy logic [4]

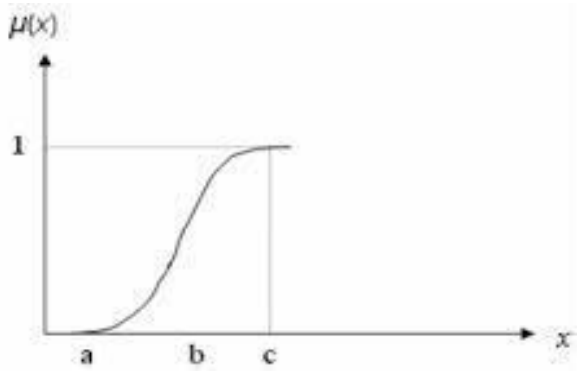
2.3.2 Fungsi – Fungsi Keanggotaan

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 [4] :

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.

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

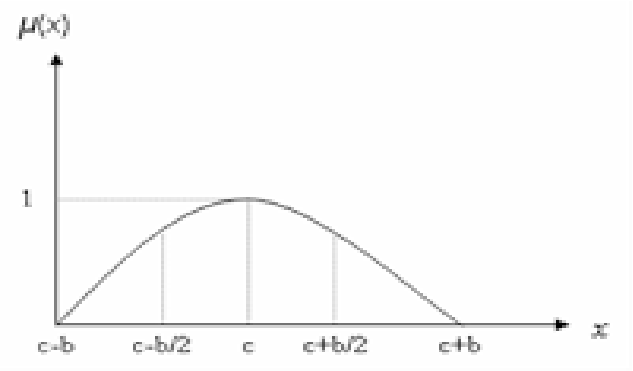


Gambar 2.2 Grafik dan notasi fungsi sigmoid [4]

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.

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

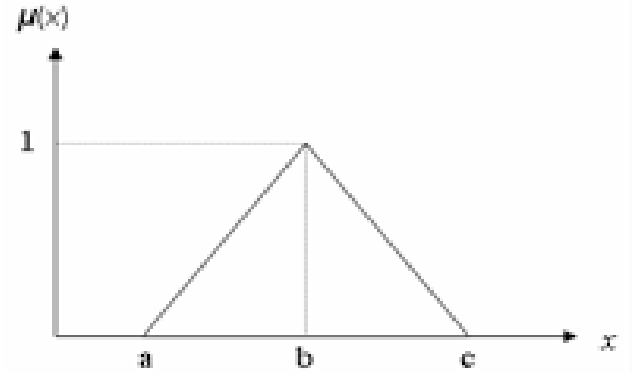


Gambar 2.3 Grafik dan notasi fungsi Phi [4]

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 Gambar2.4.

$$Segitiga\ (x,a,b,c) = \begin{cases} 0, x \le a, x \ge c \\ \frac{x-a}{b-a}, a < x \le b \\ -\left(\frac{x-a}{c-a}\right), b < x \le c \end{cases} \quad (2.3)$$

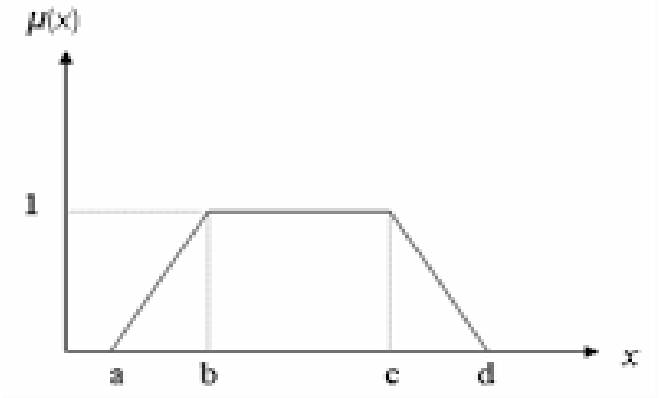


Gambar 2.4 Grafik dan notasi fungsi segitiga [4]

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.

$$Trapesium\ (x,a,b,c,d) = \begin{cases} 0, x \le a, x \ge d \\ \frac{x-a}{b-a}, a < x \le d \\ 1, b \le x \le c \\ -\left(\frac{x-d}{d-c}\right), c < c \le d \end{cases} \quad (2.4)$$



Gambar 2.5 Grafik dan notasi fungsi trapesium [4]

Himpunan fuzzy memiliki 2 atribut, yaitu:

1. Linguistik, yaitu penamaan suatu grup yang mewakili suatu keadaan atau kondisi tertentu dengan menggunakan bahasa alami, seperti: DINGIN, HANGAT, PANAS.
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 dan Panas yang semantiknya di definisikan oleh fungsi – fungsi keanggotaan tertentu.[4]

Suatu sistem berbasis aturan fuzzy yang lengkap terdiri dari 3 komponen utama: [4]

1. Fuzzyfication

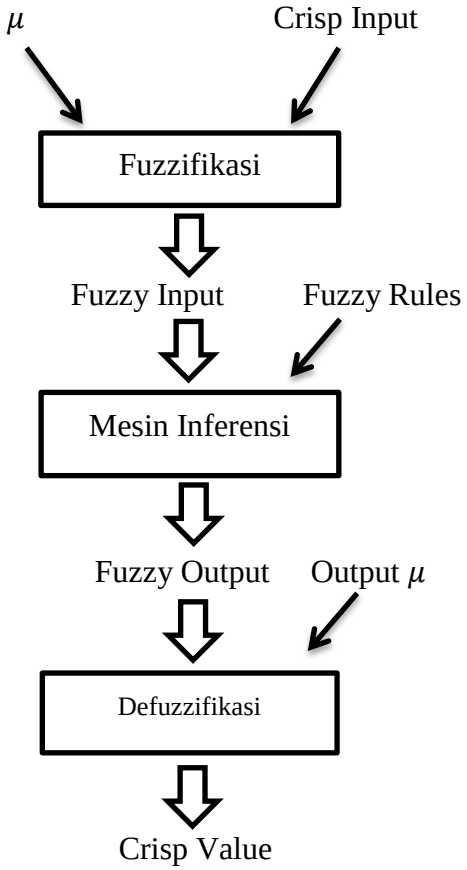
   Mengubah masukan – masukan yang nilai kebenarannya bersifat pasti (*Crips input*) kedalam bentuk fuzzy input, yang berupa nilai linguistik yang semantiknya ditentukan berdasarkan fungsi keanggotaan tertentu.

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.

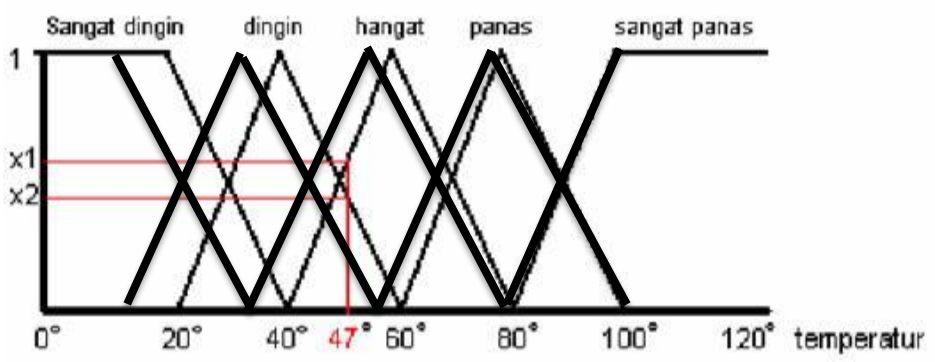


Gambar 2.5 Diagram blok untuk sistem berbasis aturan Fuzzy [4]

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 2.7. 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 2.6.



Gambar 2.6 Proses Perubahan Dari Crisp Input Menjadi Fuzzy Input [3]

2.3.5 Inference

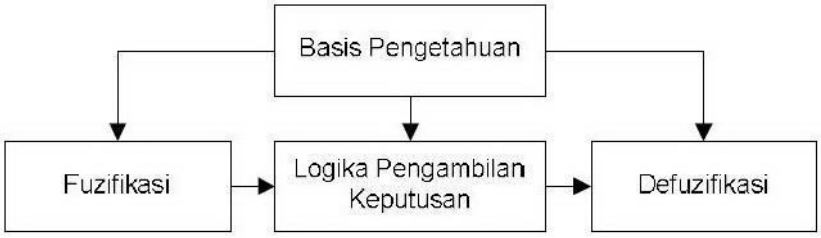
Dalam suatu sistem aturan fuzzy, proses inference memperhitunkan semua aturan yang ada dalam basis pengetahuan. Hasil dari proses inference direpresentasikan oleh suatu fuzzy set untuk setiap variabel bebas (pada consequent). Derajat keanggotaan untuk setiap nilai variabel tidak bebas menyatakan ukuran 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.[5]

Struktur sistem inferensi *fuzzy* dapat dilihat pada gambar 2.7.



Gambar 2.7 Struktur Sistem Inferensi Fuzzy [4]

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 3 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, $w_1,\ldots,w_n$ adalah konstanta yang berupa bilangan real yang merupakan bagian dari spesifikasi aturan fuzzy.

3. Model Tsukamoto

Pada metode penarikan kesimpulan samar Tsukamoto, setiap konsekuen pada aturan yang berbentuk IF-THEN harus direpresentasikan dengan suatu himpunan samar dengan fungsi keanggotaan yang monoton. Sebagai hasilnya, output hasil penarikan kesimpulan (inference) dari tiap-tiap aturan diberikan secara tegas (cnsp) berdasarkan α-predikat (fire strength). Hasil akhir diperoleh dengan menggunakan rata-rata berbobot (weight average).

2.3.6 Defuzzyfikasi

Defuzzyfikasi merupakan proses mengubah output *fuzzy* yang diperoleh dari mesin inferensi menjadi nilai tegas menggunakan fungsi keanggotaan yang sesuai dengan saat dilakukan *fuzzyfikasi*. Terdapat berbagai metode defuzzification yang telah berhasil diaplikasikan untuk berbagai macam masalah, di sini dibahas 5 metode di antaranya, yaitu:[4]

1. Centroid method

Metode ini menghitung nilai crisp menggunakan rumus:

$$y* = \frac{\int y\mu_R(y)dy}{\int \mu_R(y)dy} \quad (2.5)$$

di mana y* suatu nilai crisp.

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

$$y* = \frac{\sum y\mu_R(y)}{\sum \mu_R(y)} \quad (2.6)$$

di mana y* adalah nilai crisp dan μR(y) adalah derajat keanggotaan dari y.

2. Height method

Metode ini disebut prinsip keanggotaan maksimum karena metode ini secara sederhana memilih nilai crisp yang memiliki derajat keanggotaan maksimum. Sehingga 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

$$y* = \frac{m+M}{2} \tag{2.7}$$

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. Y* didefinisikan sebagai:

$$y* = \sum \frac{\mu(y)y}{\mu(y)} \tag{2.8}$$

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

2.4 Lampu

Lampu DC adalah lampu pijar yang menghasilkan cahaya dengan cara memanaskan kawat logam filamen sampai ke suhu tinggi sehingga menghasilkan sinar. Filamen panas dilindung dari udara oleh bola kaca yang diisi dengan gas lembam atau divakumkan.

Lampu pijar dibuat dalam berbagai macam bentuk dan tersedia untuk tegangan (voltase) kerja yang bervariasi dari mulai 1,25 volt hingga 300 volt. Energi listrik yang diperlukan lampu pijar untuk menghasilkan cahaya yang terang lebih besar dibandingkan dengan sumber cahaya buatan lainnya seperti

lampu pendar dan dioda cahaya, maka secara bertahap pada beberapa negara peredaran lampu pijar mulai dibatasi.

Pada penelitian yang dilakukan oleh Pilipus Tarigan [13], Menyatakan bahwa intensitas cahaya lampu dapat diatur dengan mengatur besaran tegangan dan arus yang masuk pada lampu.

2.5 Pintu

Pintu adalah sebuah bukaan pada dinding / bidang yang memudahkan sirkulasi antar ruang-ruang yang dilingkupi oleh dinding / bidang tersebut. Pintu biasanya ditemukan pada bangunan, misalnya rumah. Selain itu, pintu juga terdapat pada kendaraan, lemari, dan lain-lain.

Kebanyakan pintu terbuat dari kayu, dalam penggunaan yang terbatas terbuat dari aluminium, besi dan plastic PVC. Pada perkembangan nya pintu menjadi beberapa model salah satunya pintu geser atau *Rolling Door* yang di kombinasikan dengan motor penggerak, sehingga buka tutup pintu menggunakan kontrol dengan tenaga listrik, Kecepatan pergerakan pintu terbuka atau menutup dipengaruhi oleh tegangan dan arus listrik, Serta IC (Integrated Circuit) pada motor penggerak yang menyesuaikan pergerakan pintu dengan tegangan dan arus yang masuk [14].

2.6 Hukum Ohm

Hukum Ohm, yaitu Hukum dasar yang menyatakan hubungan antara Arus Listrik (I), Tegangan (V) dan Hambatan (R). Hukum Ohm dalam bahasa Inggris disebut dengan “Ohm’s Laws”. Hukum Ohm pertama kali diperkenalkan oleh seorang fisikawan Jerman yang bernama Georg Simon Ohm [12]. Pada Dasarnya bunyi dari Hukum Ohm adalah “Besar arus listrik (I) yang mengalir melalui sebuah penghantar atau Konduktor akan berbanding lurus dengan beda potensial / tegangan (V) yang diterapkan kepadanya dan berbanding terbalik dengan hambatannya (R)”. ecara matematis hukum Ohm diekspresikan dengan persamaan:

$$V = IR \tag{2.9}$$

Dimana :

- I adalah arus listrik yang mengalir pada suatu penghantar dalam satuan Ampere.
- V adalah tegangan listrik yang terdapat pada kedua ujung penghantar dalam satuan volt.
- R adalah nilai hambatan listrik (resistansi) yang terdapat pada suatu penghantar dalam satuan ohm.

2.6.1 Arus Listrik

Arus Listrik atau *Electrict Current* dapar didefinisikan sebagai jumlah muatan listrik yang mengalir tiap satuan waktu. Biasanya arus memiliki satuan A (Ampere) dan dalam rumus dinyatakan dengan I, Arus listrik merupakan gerakan kelompok partikel bermuatan listrik dalam arah tertentu.

2.6.2 Tegangan Listrik

Tegangan Listrik atau Voltase adalah perbedaan potensial listrik antara dua titik dalam rangkaian listrik, dan dinyatakan dalam satuan volt. Besaran ini mengukur energi potensial dari sebuah medan listrik yang mengakibatkan adanya aliran listrik dalam sebuah konduktor listrik. Tergantung pada perbedaan potensial listriknya, suatu tegangan listrik dapat dikatakan sebagai ekstra rendah, rendah, tinggi atau ekstra tinggi.

2.6.3 Hambatan Listrik

Hambatan listrik adalah perbandingan antara tegangan listrik dari suatu komponen elektronik seperti resistor dengan arus listrik yang melewatinya. Hambatan dinyatakan dalam satuan ohm. Gerakan berlawanan ini biasanya disebut dengan hambatan. Besarnya arus didalam rangkaian adalah jumlah dari energi yang ada untuk mendorong elektron dan juga jumlah dari hambatan sebuah rangkaian untuk menghambat lajunya arus.

2.7 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. *C#.*
3. *Visual Basic*.
4. *Java*.

2.7.1 Analisis dan Desain Berorientasi Objek (Object-Oriented Analysis and Design Process)

Pemrograman berorientasi objek bekerja dengan baik ketika dibarengi dengan *object-oriented analysis and design process (OOAD).* Jika membuat program berorientasi objek tanpa OOAD, ibarat membangun rumah tanpa terlebih dahulu menganalisa apa saja yang dibutuhkan oleh rumah itu, tanpa perencanaan tanpa *blueprint*, tanpa menganalisis ruangan apa saja yang diperlukan, berapa besar rumah yang akan dibangun dan sebagainya.

2.7.2 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.

2.7.3 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 [8].

2.7.4 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 [8].

2.7.5 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*) [8].

2.5.1 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.

2.8 UML (Unified Modeling Language)

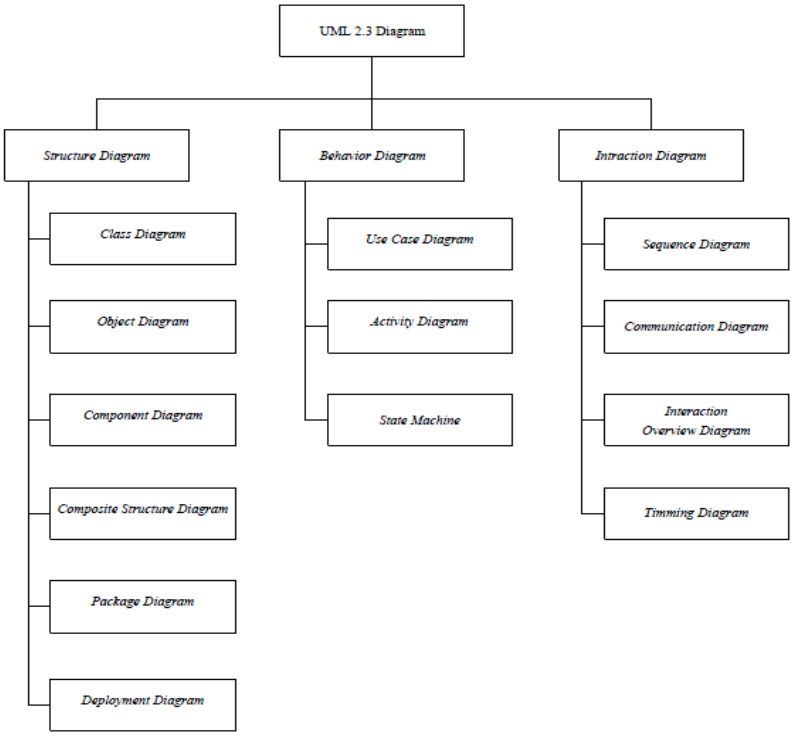
UML adalah keluarga notasi grafis yang didukung oleh meta-model tunggal, yang membantu pendeskripsian dan desain sistem perangkat lunak, khususnya sistem yang dibangun menggunakan pemrograman berorientasi objek. Definisi ini merupakan definisi sederhana. Pada kenyataannya, pendapat orang-orang tentang 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* [9]. 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.



Gambar 2.8 Diagram UML [9]

Berikut ini penjelasan singkat dari pembagian kategori tersebut.

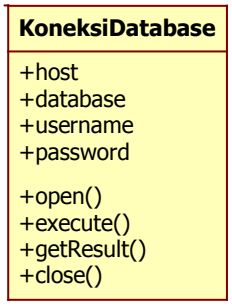
1. *Structure diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
2. *Behavior diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkain perubahan yang terjadi pada sebuah sistem.
3. *Interaction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

2.8.1 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.

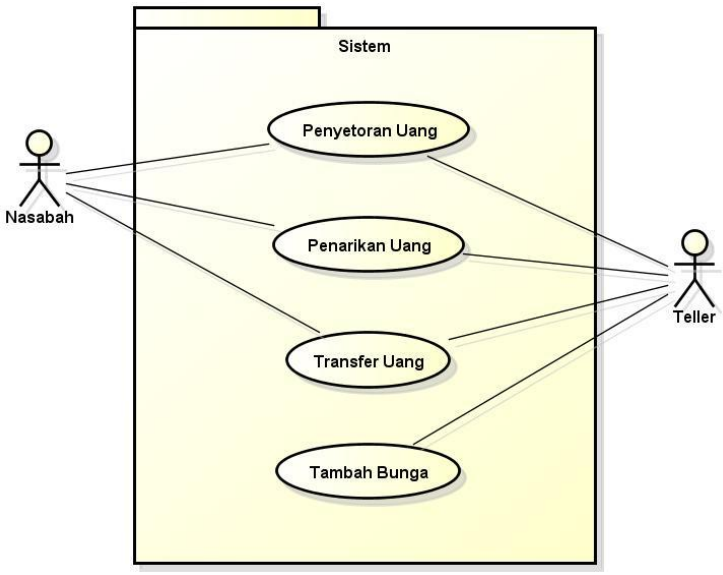
Diagram kelas dibuat agar pembuat program atau *programmer* membuat kelas-kelas sesuai rancangan di dalam diagram kelas agar antara dokumentasi perancangan dan perangkat lunak sinkron. Berikut adalah contoh dari diagram kelas.



Gambar 2.9 Contoh *Class Diagram* [9]

2.8.2 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*.

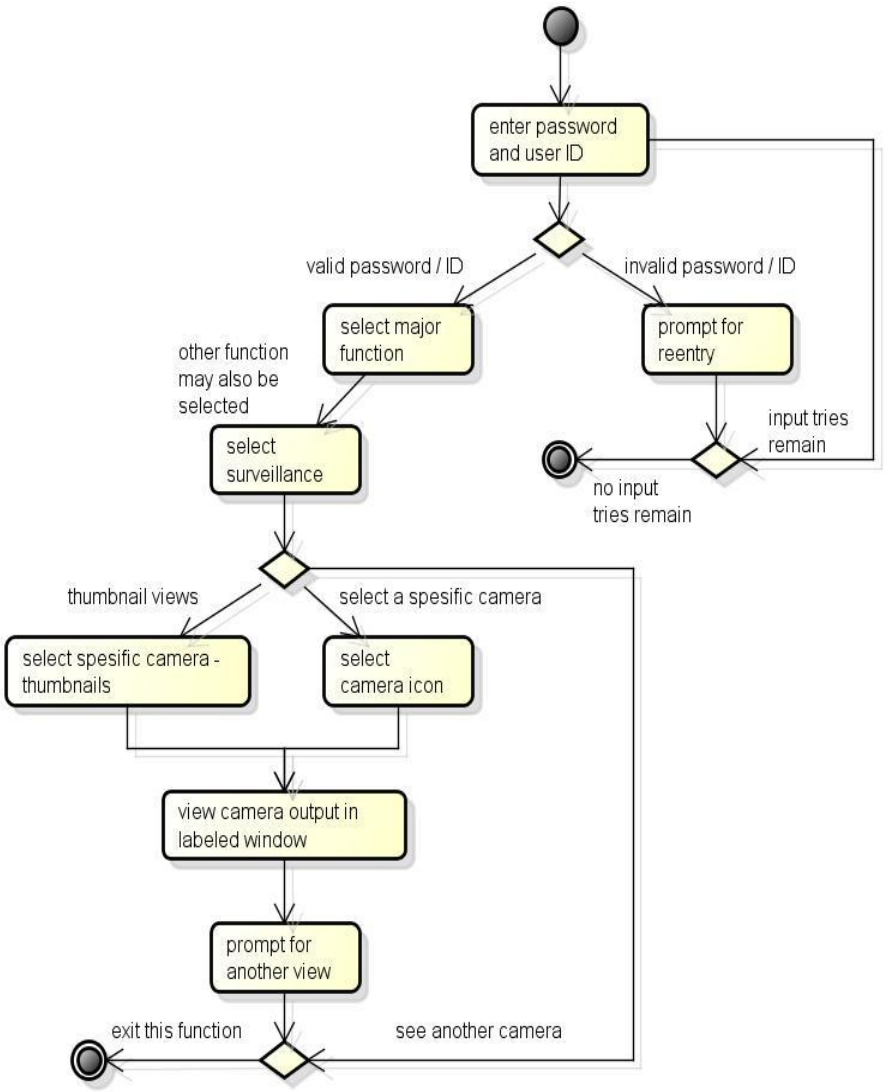


Gambar 2.10 Contoh *Use Case Diagram* [9]

2.8.3 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:

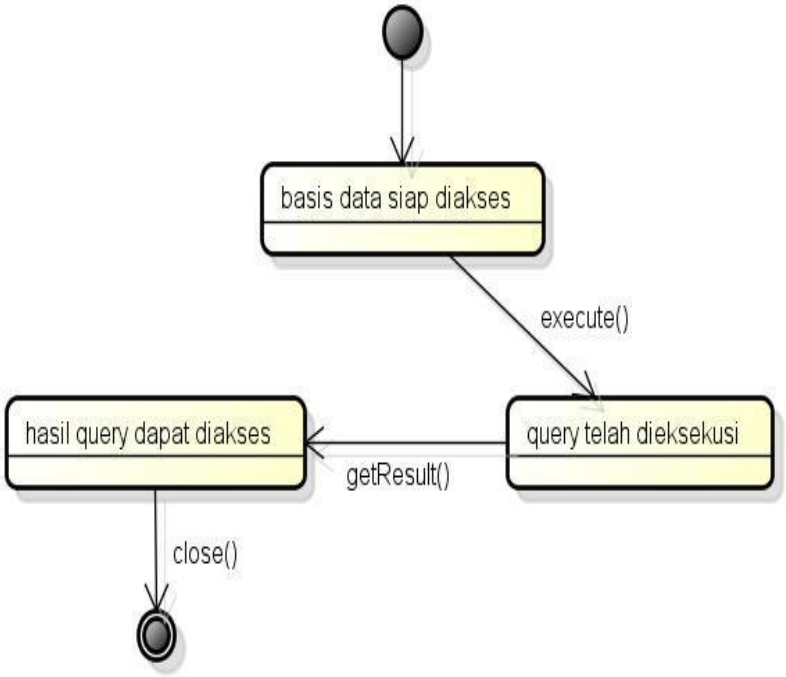
1. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
2. Urutan atau pengelompokkan tampilan dari sistem/*user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
3. Rancangan pengujian dimana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



Gambar 2.11 Contoh *Activity Diagram* [9]

2.8.4 Diagram State Machine

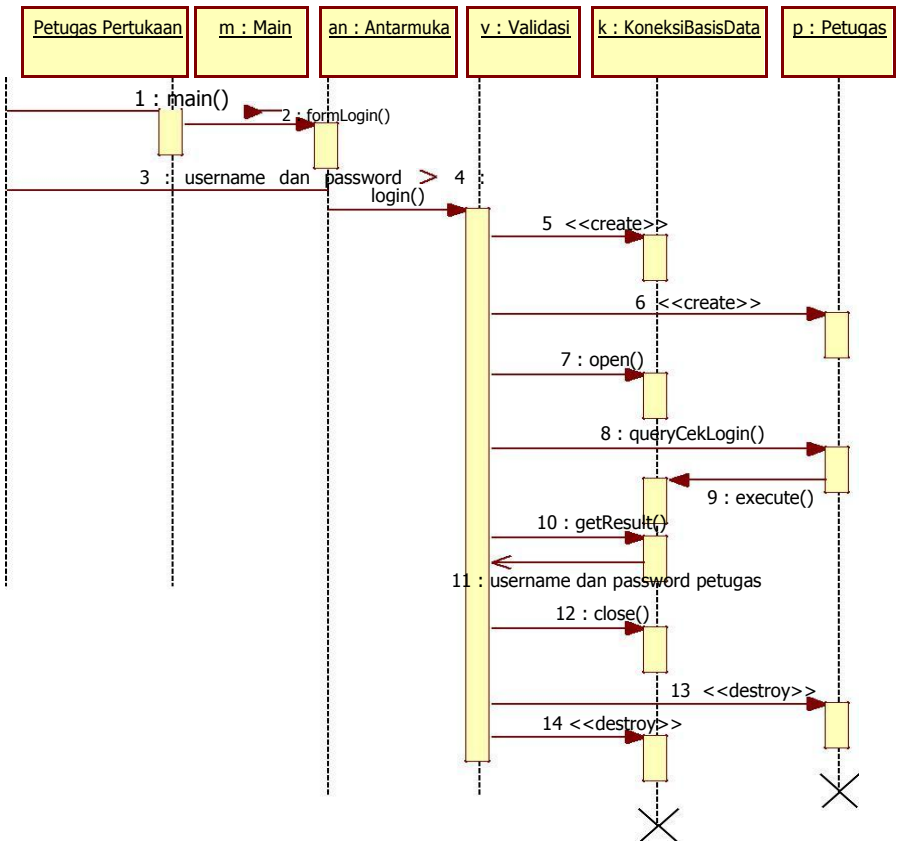
State machine diagram atau *statechart diagram* atau dalam bahasa Indonesia disebut diagram mesin status atau sering juga disebut diagram status digunakan untuk menggambarkan perubahan status atau transisi status dari sebuah mesin atau sistem atau objek. Jika diagram sekuen digunakan untuk interaksi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



Gambar 2.12 Contoh *Statechart Diagram* [9]

2.8.5 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 2.13 Contoh *Sequence Diagram* [8]

2.9 Unity

Unity Game Engine adalah *software* atau *game engine* yang digunakan untuk membuat *video game* berbasis dua atau tiga dimensi dan dapat digunakan secara gratis. Selain untuk membuat *game*, *unity* juga dapat digunakan untuk membuat konten yang interaktif lainnya seperti, visual arsitektur dan *real-time* 3D animasi.

Unity adalah sebuah sebuah *tool* yang terintegrasi untuk membuat *game*, arsitektur bangunan dan simulasi. *Unity* bisa digunakan untuk *game*s PC dan *games online*. Untuk *games online* diperlukan sebuah plugin, yaitu *Unity Web Player*, yang sama halnya dengan *flash player* pada *browser*. Bahasa pemrograman yang digunakan bermacam-macam, mulai dari *javascript*, C#, dan *boo* [11].

Unity tidak bisa melakukan desain atau *modelling*, dikarenakan *unity* bukan merupakan *tools* untuk mendesain. Banyak hal yang bisa di lakukan di unity, ada fitur *audio reverb zone , particle effect , sky box* untuk menambahkan langit, dan masih banyak lagi, dan juga bisa langsung *edit texture* dari *editor* seperti *photoshop* dll.

Features (Scripting) di dalam *unity* adalah sebagai berikut (Unity Technologies, 2013):

1. Mendukung 3 bahasa pemrograman, *JavaScript*, C#, dan *Boo*.
2. *Flexible and EasyMoving*, *rotating*, dan *scaling objects* hanya perlu sebaris kode. Begitu juga dengan *duplicating*, *removing*, dan *changing properties.*
3. *Multi Platform Game* bisa di deploy di PC, Mac, Wii, iPhone, iPad dan *browser, android*.
4. *Visual Properties Variables* yang di definisikan dengan *scripts* ditampilkan pada editor. Bisa digeser, di *drag* and *drop*, bisa memilih warna dengan *color picker.*
5. Berbasis .NET, penjalanan program dilakukan dengan *Open Source* .NET *platform, Mono.*

2.10 Bahasa Pemrograman C#

C# merupakan sebuah bahasa pemrograman yang berorientasi objek yang dikembangkan oleh Microsoft sebagai bagian dari inisiatif kerangka .NET Framework. C# adalah Java versi Microsoft, sebuah bahasa multi flatform yang didesain untuk bisa berjalan di berbagai mesin. C# adalah pemrograman berorientasi Object (OOP). C# memiliki kekuatan bahasa C++ dan portabilitas seperti Java. Fitur-fitur yang diambilnya dari bahasa C++ dan Java adalah desain berorientasi objek, seperti *garbage collection, reflection*, akar kelas *(root class)*, dan juga penyederhanaan terhadap pewarisan jamak *(multiple inheritance)*.

Bahasa pemrograman C# dibuat sebagai bahasa pemrograman yang bersifat *general-purpose* (untuk tujuan jamak), berorientasi objek, modern, dan sederhana. C# ditujukan agar cocok digunakan untuk menulis program aplikasi baik dalam sistem klien-server (*hosted system*) maupun sistem embedded

(embedded system), mulai dari program aplikasi yang sangat besar yang menggunakan sistem operasi yang canggih hingga kepada program aplikasi yang sangat kecil. Meskipun aplikasi C# ditujukan agar bersifat 'ekonomis' dalam hal kebutuhan pemrosesan dan memori komputer, bahasa C# tidak ditujukan untuk bersaing secara langsung dengan kinerja dan ukuran program aplikasi yang dibuat dengan menggunakan bahasa pemrograman C.

2.11 Pengujian Aplikasi

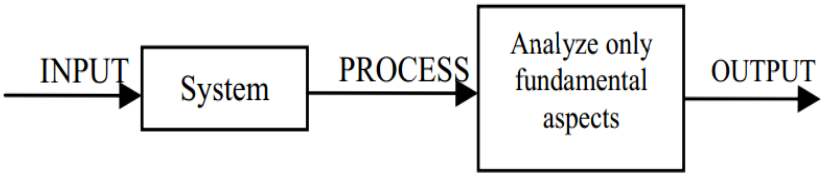
Pengujian adalah proses untuk menemukan error pada perangkat lunak sebelum dikirim kepada pengguna. Pengujian Software adalah kegiatan yang ditujukan untuk mengevaluasi atribut atau kemampuan program dan memastikan bahwa itu memenuhi hasil yang dicari, atau suatu investigasi yang dilakukan untuk mendapatkan informasi mengenai kualitas dari produk atau layanan yang sedang diuji (under test), yang bermanfaat dalam operasional bisnis untuk memahami tingkat risiko pada implementasinya [16].

2.11.1 Pengujian Blackbox

Metode Black Box memungkinkan perekayasa perangkat lunak mendapatkan serangkaian kondisi input yang sepenuhnya menggunakan semua persyaratan fungsional untuk suatu program.

Black Box dapat menemukan kesalahan dalam kategori berikut [16] :

1. Fungsi-fungsi yang tidak benar atau hilang
2. Kesalahan interface
3. Kesalahan dalam strutur data atau akses basisdata eksternal
4. Inisialisasi dan kesalahan terminasi
5. validitas fungsional
6. kesensitifan sistem terhadap nilai input tertentu
7. batasan dari suatu data



Gambar 2.14 Sistem kerja dari Teknik Pengujian Black Box

Terdapat beberapa tipe dari Black Box Testing, yaitu :

1. Equivalence class partitioning
   a) Bagi domain Input ke dalam beberapa kelas yang nantinya akan dijadikan sebagai kasus uji.
   b) Kelas yang telah terbentuk disajikan sebagai kondisi input dalam kasus.
   c) Kelas tersebut merupakan himpunan nilai-nilai yang valid dan tidak valid.
   d) Kondisi input bisa merupakan suatu range, harga khusus, suatu. himpunan, atau suatu boolean.
   e) Bila kondisi input berupa suatu range, maka input kasus ujinya satu valid dan dua yang invalid.
   f) Bila kondisi input berupa suatu harga khusus, maka input kasus ujinya satu valid dan dua yang invalid.
   g) Bila kondisi input berupa suatu anggota himpunan, maka input kasus ujinya satu valid dan dua yang invalid.

2. Sample testing
   a) Melibatkan sejumlah nilai yang dipilih dari data masukan kelas ekivalensi.
   b) Integrasikan nilai tersebut ke dalam kasus uji.
   c) Nilai yang dipilih dapat berupa konstanta atau variabel Limit Testing.
   d) Kasus uji yang memproses nilai batas (atau titik singular).
   e) Nilai batas disimpulkan dari kelas ekivalensi dengan mengambil nilai yang sama atau mendekati nilai yang membatasi kelas akivalensi tersebut.

f) Limit test also juga melibatkan data keluaran dari ekivalensi kelas
g) Pada kasus segi tiga, misalnya limit testing mencoba untuk mendeteksi apakah a+b >= c dan bukan a + b > c
h) Bila kondisi input menentukan suatu range, maka kasus ujinya harus mencakup pengujian nilai batas dari range dan nilai invalid yang dekat dengan nilai batas. Misal bila rangenya antara [-1.0, +1.0], maka input untuk kasus ujinya adalah -1.0, 1.0, -1.001,1.001
i) Bila kondisi inputnya berupa harga khusus kasaus ujinya harus mencakup nilai minimum dan maksimum. Misal suatu file dapat terdiri dari 1 to 255 record, maka kasus ujinya harus mencakup untuk.

3. Limit testing

Data dipilih dari luar range yang didefinisikan. Tujuan pengujian ini adalah untuk membuktikan tidak adanya kejadian yang katastropik yang dihasilkan akibat adanya keabnormalan.

4. Robustness testing

Suatu pengujian yang hasilnya hanya dapat dievaluasi per sub program, tidak bisa dilakukan per modul

5. Behavior testing

Suatu pengujian yang hasilnya hanya dapat dievaluasi per sub program, tidak bisa dilakukan per modul

6. Requirement testing

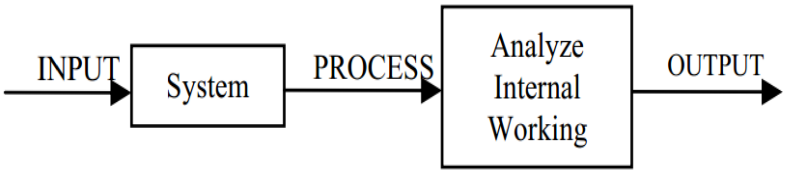
a) Menyusun kasus uji untuk tiap kebutuhan yang berkorelasi dengan modul / CSU
b) Tiap kasus uji harus dapat dirunut dengan kebutuhan perangkat lunaknya melalui matriks keterunutuan

2.11.2 Pengujian Whitebox

Metode pengujian dengan menggunakan struktur kontrol program untuk memperoleh kasus uji [16].

Dengan menggunakan white box akan didapatkan kasus uji yang :

a) Menjamin seluruh jalur independen di dalam modul yang dieksekusi sekurang-kurangnya sekali
b) Menguji semua keputusan logikal
c) Menguji seluruh Loop yang sesuai dengan batasannya
d) Menguji seluruh struktur data internal yang menjamin validitas.



Gambar 2.15 Sistem kerja dari Teknik Pengujian White Box

e) Basis Path adalah teknik uji coba white box
f) Basis Path : untuk mendapatkan kompleksitas lojik dari suatu prosedur dan menggunakan ukuran ini sebagai petunjuk untuk mendefinisikan himpunan jalur yang akan diuji
g) Basis Path menggunakan notasi graph untuk menggambarkan aliran kontrolnya.