BAB II

LANDASAN TEORI

2.1 Logika *Fuzzy*

Konsep tentang logika *fuzzy* diperkenalkan oleh Prof. Lotfi Astor Zadeh pada 1962. Logika *fuzzy* adalah metodologi sistem kontrol pemecahan masalah yang cocok diimplementasikan pada sistem, mulai dari sistem yang sederhana, sistem kecil, *embedded* sistem, jaringan PC, *multi-channel* atau *workstation* berbasis akurasi data, dan sistem kontrol. Metodologi ini dapat diterapkan pada perangkat keras, perangkat lunak, atau kombinasi keduanya. Dalam logika klasik dinyatakan bahwa segala sesuatu bersifat biner, yang artinya adalah hanya mempunyai dua kemungkinan, "Ya atau Tidak", "Benar atau Salah", "Baik atau Buruk" dan lain-lain. Oleh karena itu, semua ini dapat mempunyai nilai keanggotaan 0 dan 1. Akan tetapi, dalam logika *fuzzy* memungkinakan nilai keanggotaan berada diantara 0 dan 1. Artinya, bisa saja suatu keadaan mempunyai dua nilai "Ya dan Tidak", "Benar dan Salah", "Baik dan Buruk" secara bersamaan, namun besar nilainya tergantung pada bobot keanggotaan yang dimilikinya. Logika *fuzzy* dapat digunakan diberbagai bidang, seperti pada sistem diagnosis penyakit (dalam bidang kedokteran), pemodelan sistem pemasaran, riset operasi (dalam bidang ekonomi), kendali kualitas air, prediksi adanya gempa bumi, klasifikasi dan pencocokan pola (dalam bidang teknik)[20].

2.1.1 Alasan digunakannya Logika *Fuzzy*

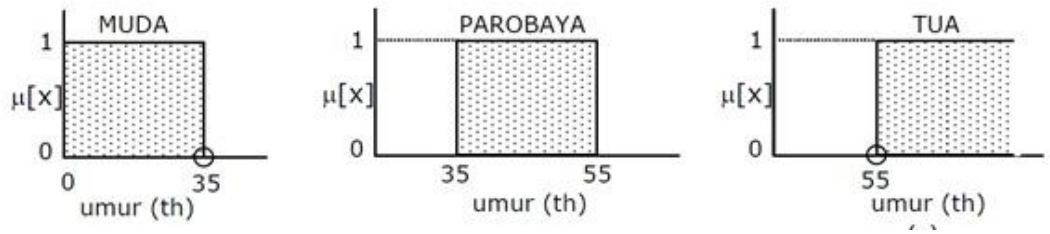
Menurut Cox (1994) , ada beberapa alasan mengapa orang menggunakan logika *fuzzy* adalah Konsep logika *fuzzy* mudah dimengerti , karena *fuzzy* menggunakan dasar teori himpunan , maka konsep matematis yang mendasari penaralan *fuzzy* tersebut cukup mudah untuk dimengerti dan logika *fuzzy* sangat fleksibel, artinya mampu beradaptasi dengan perubahan-perubahan dan ketidakpastian yang menyertai permasalahan.

2.1.2 Variabel dan Himpunan *Fuzzy*

Dalam ilmu logika *fuzzy* dikenal mengenal dua himpunan, yaitu himpunan *crisp* (tegas) dan himpunan *fuzzy* (samar).

Himpunan *crisp* adalah himpunan yang menyatakan suatu obyek merupakan anggota dari satu himpunan memiliki nilai keanggotaan (μ) = ya (1) atau tidak (0), oleh karena itu himpunan *crisp* disebut himpunan tegas.

Misalkan variabel umur dibagi 3 kategori, yaitu: MUDA < 35 tahun ; PAROBAYA 35 ≤ umur ≤ 55 tahun ; TUA > 55 tahun. yang dapat di lihat pada gambar 2.1:



Gambar 2.2 Himpunan *crisp*

Jika menggunakan himpunan *crisp*, dapat diambil kesimpulan bahwa:

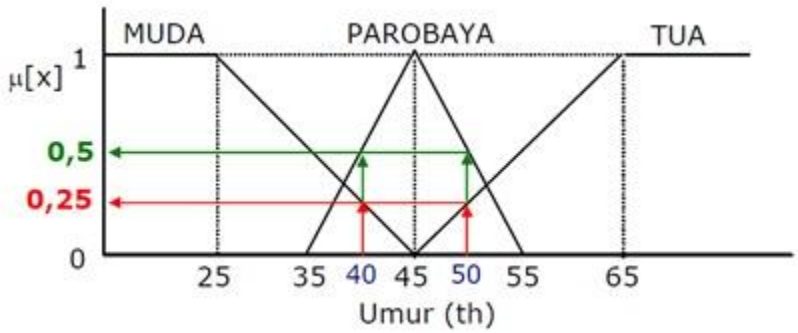
2. Usia 34 tahun, dikatakan MUDA → μMUDA[34]=1
3. Usia 35 tahun kurang 1 hari, dikatakan MUDA → μMUDA[35th-1hr]=1
4. Usia 35 tahun, dikatakan TIDAK MUDA → μMUDA[35]=0
5. Usia 55 tahun, dikatakan PAROBAYA → μPAROBAYA[55]=1
6. Usia 55 tahun lebih 1 hari, dikatakan TIDAK PAROBAYA → μPAROBAYA[55th+1hr]=0 atau
7. Usia 55 tahun lebih 1 hari, dikatakan TUA → μTUA[55th+1hr]=1

Dari kesimpulan diatas, himpunan *crisp* menyatakan umur seseorang kedalam suatu kategori secara tidak adil, karena orang yang berusia 35 tahun dikatakan parobaya, sedangkan orang yang berusia 35 tahun kurang 1 hari

dikatakan tidak parobaya (karena masuk kategori muda). selisih 1 hari saja menimbulkan berbedaan kategori yang signifikan.

Sedangkan untuk himpunan *fuzzy* adalah himpunan yang menyatakan suatu obyek dapat menjadi anggota dari beberapa himpunan dengan nilai keanggotaan (μ) yang berbeda. Untuk lebih jelasnya , perhatikan contoh dibawah:

Himpunan *Fuzzy* digunakan untuk mengatasi hal tersebut, sehingga dengan menggunakan himpunan *fuzzy*, seseorang dapat masuk ke dua kategori secara bersamaan, misalnya seseorang yang berusia 35 tahun kurang 1 hari dapat masuk kategori MUDA dan PAROBAYA sekaligus. Lihat gambar 2.3 :



Gambar 2.3 Himpunan *Fuzzy*

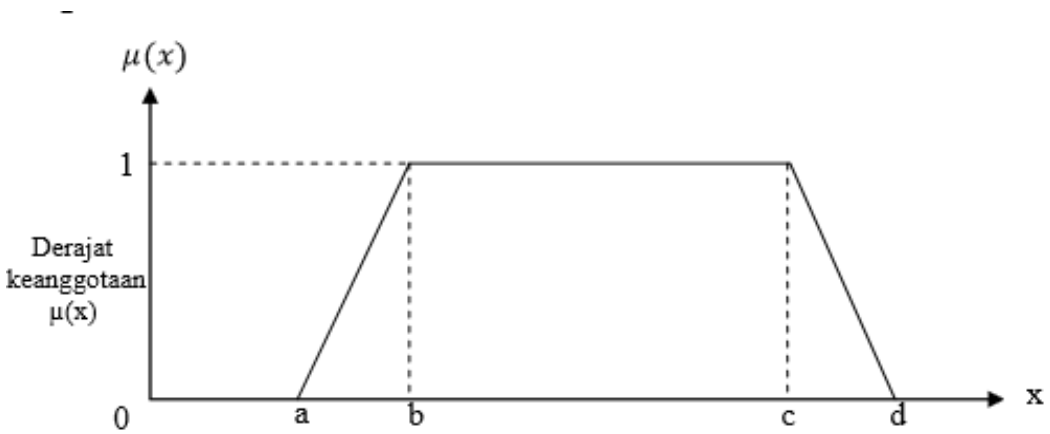
contoh:

1. Seseorang yang berumur 40 tahun termasuk dalam himpunan MUDA dengan μMUDA[40]=0,25; namun dia juga termasuk dalam himpunan PAROBAYA dengan μPAROBAYA[40]=0,5.
2. Seseorang yang berumur 50 tahun termasuk dalam himpunan MUDA dengan μMUDA[50]=0,25; namun dia juga termasuk dalam himpunan PAROBAYA dengan μPAROBAYA[50]=0,5 [20]

2.1.3 Fungsi Keanggotaan

Fungsi keanggotaan adalah suatu kurva yang menunjukan pemetaan titik-titik *Input* data ke dalam nilai keanggotaan (sering juga disebut dengan derajat keanggotaan) yang memiliki interval antara 0 sampai 1. Salah satu cara yang

dapat digunakan untuk mendapatkan nilai keanggotaan adalah dengan menggunakan pendekatan fungsi. Salah satunya representasi kurva trapesium. Representasi kurva trapesium pada dasarnya berbentuk segitiga trapesium. Pada kurva ini memiliki nilai keanggotaan 1 (gambar 2.3).



Gambar 2.4 Representasi Kurva Trapesium

Fungsi Keanggotaan :

$$\mu[X] = \begin{cases} 0; & x \leq a \; atau \; x \geq d \\ \frac{x-a}{b-a} & ; a \leq x \leq b \\ 1; & b \leq x \leq c \\ \frac{d-x}{d-c}; & c \leq x \leq d \end{cases} \qquad (2.1)$$

Keterangan:

a = nilai domain terkecil yang mempunyai derajat keanggotaan nol

b = nilai domain terkecil yang mempunyai derajat keanggotaan satu

c = nilai domain terbesar yang mempunyai derajat keanggotaan satu

d = nilai domain terbesar yang mempunyai derajat keanggotaan nol

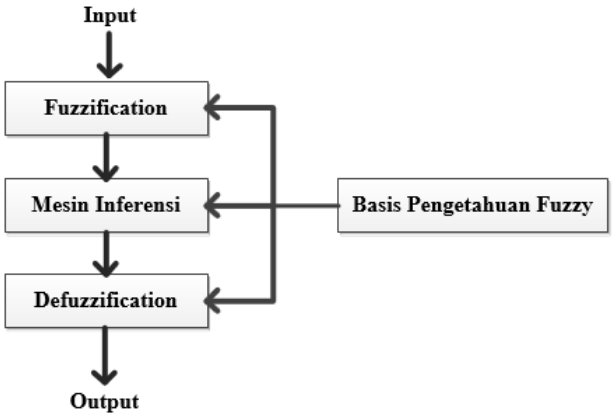
x = nilai *Input* yang akan di ubah ke dalam bilangan *fuzzy*

2.1.4 *Fuzzy Inference System*

Pada *Fuzzy Inference System*, terdapat 3 metode, yaitu metode tsukamoto, mamdani dan sugeno.

1. Pada metode Tsukamoto , setiap konsekuen pada aturan yang berbentuk IF-Then harus direpresentasikan dengan suatu himpunan *fuzzy* dengan fungsi keanggotaan yang monoton. Sebagai hasilnya, *output* hasil inferensi dari tiap-tiap aturan diberikan secara tegas (*crisp*) berdasarkan α-predikat (*fire strength*). Hasil akhirnya diperoleh dengan menggunakan rata-rata terbobot[20].
2. Metode Mamdani sering disebut metode Max-Min. Metode ini diperkenalkan oleh Ebrahim Mamdani pada tahun 1975. Untuk mendapatkan *output* diperlukan 4 tahapan yaitu tahapan pembentukan himpunan *fuzzy*, aplikasi fungsi implikasi, komposisi aturan, dan penegasan(*defuzzy*).
3. Penalaran dengan metode Sugeno hampir sama dengan penalaran metode Mamdani , hanya saja *output* (konsekuen) sistem tidak berupa himpunan *fuzzy*, melainkan berupa konstanta atau persamaan linear. Metode ini diperkenalkan oleh Takagi-Sugeno Kang pada tahun 1985, sehingga metode ini sering juga dinamakan metode TSK. Apabila komposisi aturan menggunakan metode Sugeno, maka de*fuzzy*fikasi dilakukan dengan cara mencari nilai rata-ratanya.

Cara kerja logika *fuzzy* dapat digambarkan sebagai struktur elemen dasar sistem inferensi *fuzzy* pada gambar 2.4:



Gambar 2.5 Struktur sistem inferensi *fuzzy*.

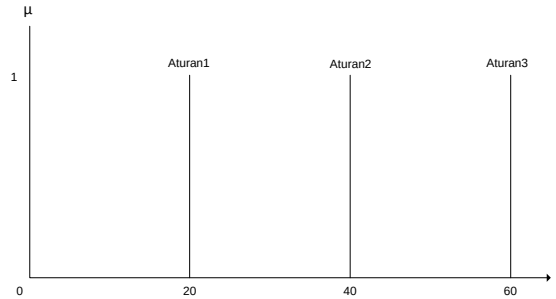
Keterangan :

1. Basis Pengetahuan *Fuzzy* merupakan kumpulan *rule-rule fuzzy* dalam bentuk pernyataan IF…THEN.
2. *Fuzzy*fikasi adalah proses untuk mengubah *Input* sistem yang mempunyai nilai tegas menjadi variabel linguistic menggunakan fungsi keanggotaan yang disimpan dalam basis pengetahuan *fuzzy*.
3. Mesin Inferensi merupakan proses untuk mengubah *Input fuzzy* dengan cara mengikuti aturan-aturan (IF-THEN *Rule*s) 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

2.1.5 Model Defuzzifikasi Metode SUGENO

Langkah-langkah dalam penentuan defuzzifikasi menggunakan metode Sugeno dilakukan dengan cara sebagai berikut:

1. Menggunakan aturan *Conjunction* (∧) dengan memilih derajat keanggotaan minimum pada perhitungan kasus yang ada.
2. Setelah aturan *Conjunction* selesai, lanjut pada Aturan *Disjunction* (∨) dimana memilih derajat keangotaan maksimum dari hasil aturan *Conjunction*.
3. Setelah aturan *Conjunction* dan *Disjunction* selesai, dibuatlah grafik derajat keanggotaan model sugeno.



Gambar 2.6 Grafik derajat keanggotaan model sugeno

Keterangan :

1. Aturan1, Aturan2, Aturan3 = merupakan hasil *output* yang dihasilkan dari sistem fuzzy
2. 0-20 = nilai (*range*) yang ada pada hasil *output* Aturan1
3. 20-40 = nilai (*range*) yang ada pada hasil *output* Aturan2
4. 40-60 = nilai (*range*) yang ada pada hasil *output* Aturan3

4. Kemudian mencari rata-rata perhitungan menggunakan rumus (*weight average Method*)[12]:
   a. Rumus *Weighted Average Method*

$$y^* = \Sigma \frac{\mu(y)y}{\mu(y)} \qquad (2.1)$$

Keterangan:

y^* dan y = suatu nilai *crisp*

μ (y) = derajat keanggotaan dari nilai crisp y

2.2 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 *refreshing*. 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 pada tahun 1944. Teori itu dikemukakan oleh John Von Neumann dan Oskar Morgenstern yang berisi: "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 menentukan kemungkinan tindakan untuk setiap pemain, sejumlah keterangan diterima setiap

pemain sebagai kemajuan bermain, dan sejumlah kemenangan atau kekalahan dalam berbagai situasi" [18]. Pengertian *game* menurut para ahli :

1. Menurut Agustinus Nilwan dalam bukunya Pemograman Animasi dan *Game* Profesional terbitan Media Komputindo, *game* merupakan permainan komputer yang dibuat dengan teknik dan metode animasi. Jika ingin mendalami penggunaan animasi haruslah memahami pembuatan *game*. Atau jika ingin membuat *game*, maka haruslah memahami teknik dan metode animasi, sebab keduanya saling berkaitan.
2. Menurut Clark C. Abt, *Game* adalah kegiatan yang melibatkan keputusan pemain, berupaya mencapai tujuan dengan dibatasi oleh konteks tertentu (misalnya, dibatasi oleh peraturan).
3. Menurut Chris Crawford, seorang computer *game designer* mengemukakan bahwa *game*, pada intinya adalah sebuah interaktif. Aktivitas yang berpusat pada sebuah pencapaian, ada pelaku aktif (*player*), ada pelaku pasif (NPC).
4. Menurut David Parlett, *game* adalah sesuatu yang memiliki "akhir dan cara mencapainya" : artinya ada tujuan, hasil dan serangkaian peraturan untuk mencapai keduanya.
5. Menurut Roger Caillois, seorang sesiolog Perancis dalam bukunya yang berjudul *Les Jeux Et Les Hommes* menyatakan *game* adalah aktivitas yang mencakup karakteristik berikut: *fun* (bebas bermain adalah pilihan bukan kewajiban), *separate* (terpisah), *uncertain*, *non-productive*, *governed by rules* (ada aturan), *fictitious* (pura-pura).
6. Menurut Greg Costikyan, *game* adalah "sebentuk karya seni dimana peserta, yang disebut pemain, membuat keputusan untuk mengelola sumber daya yang dimilikinya melalui benda di dalam *game* demi mencapai tujuan".

2.2.1 Klasifikasi *Game*

Game biasanya digunakan untuk mengkatagorikan sebuah *game* berdasarkan dari interaksi *gameplay* daripada perbedaan tampilan atau narasi.

Sebuah jenis *game* dapat didefinisikan dengan kumpulan dari sebuah tipe permainan dari *game* tersebut. *Game* dapat diklasifikasikan menjadi dua, yaitu[2]:

2.2.2 Berdasarkan Jenis Platform yang Digunakan

Tiga kategori utama *platform* pada *game* adalah *personal computer* (PC), console, dan mobile[2]:

1. *Personal Computer* (PC)

   PC *game* adalah *game* yang dibuat untuk computer baik berbasis Windows, Mac, ataupun Linux. PC menyediakan kekuatan grafis dan pemrosesan yang kuat yang memungkinkan pengembang untuk membuat *game* yang mutakhir. Tetapi kelemahan *game* PC adalah mahal karena pengguna harus menggunakan hardware yang up-to-date untuk memainkan *game* PC dengan baik. Selain itu, kelemahan *game* PC bagi pengembang adalah banyaknya varian dari konfigurasi PC membuat sulit bagi pengembang untuk memastikan *game* tersebut berjalan dengan benar pada semua *setup* PC.

2. *Console*

   Console adalah *hardware* yang dibuat oleh pihak ketiga seperti Sony, Microsoft, dan Nintendo. *Console* terhubung ke televisi dan tujuan utamanya adalah untuk bermain *game*. *Game Console* sangat menarik bagi *game* pengembang karena mereka hanya perlu memikirkan satu konfigurasi hardware ketika membuat software untuk *console*. Sangat kontras dengan PC yang memiliki opsi konfigurasi yang tak terbatas.

3. *Mobile*

   Mobile platform terdiri atas sesuatu yang portable dan bisa digenggam, termasuk ponsel, PDA, iPods, dan handheld *game* seperti Nintendo DSi atau Sony PSP. *Game mobile* memiliki kontrol yang sederhana (terutama jika dibandingkan dengan PC).

2.2.3 Berdasarkan *Genre* Permainan

Genre *game* bisa dibagi menjadi 9 genre, yang dimana tiap genre memiliki karakteristik tersendiri , yaitu[2]:

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 shooters *game*, baik yang 2D maupun 3D seperti First Person, Shooters (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 yang 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 maupun 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. *Vechicle Simulation*

Vechicle 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. *Adventure*

   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.

2.2.4 OOP (*Object Oriented Programming*)

Metodologi berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya. Metodologi berorientasi objek merupakan suatu cara bagaimana sistem perangkat lunak dibangun melalui pendekatan objek secara sistematis. Metode berorientasi objek didasarkan pada penerapan prinsip-prinsip pengelolaan kompleksitas. Metode

berorientasi onjek meliputi rangkaian aktivitas analisis berorientasi objek, perancangan berorientasi objek, pemrograman berorientasi objek, dan pengujian berorientasi objek.

Pada saat ini, metode berorientasi objek banyak dipilih karena metodologi lama banyak menimbulkan masalah seperti adanya kesulitan pada saat mentransformasi hasil dari satu tahap pengembangan ke tahap berikutnya, misalnya pada metode pendekatan terstruktur, jenis aplikasi yang dikembangkan saat ini berbeda dengan masa lalu. Aplikasi yang dikembangkan saat ini sangat beragam (aplikasi bisnis, *real-time*, utility, dan sebagainya) dengan *platform* yang berbeda-beda, sehingga menimbulkan tuntutan kebutuhan metodologi pengembangan yang dapat mengakomodasi ke semua jenis aplikasi tersebut[24].

Keuntungan menggunakan metodologi berorientasi objek adalah sebagai berikut[24]:

a. Meningkatkan produktivitas

   Karena kelas dan objek yang ditemukan dalam suatu masalahmasih dapat dipakai ulang untuk masalah lainnya yang melibatkan objek tersebut (*reusable).*

b. Kecepatan pengembangan

   Karena sistem yang dibangun dengan baik dan benar pada saat analisis dan perancangan akan menyebabkan berkurangnya kesalahan pada saat pengkodean.

c. Kemudahan pemeliharaan

   Karena dengan model objek, pola-pola yang cenderung tetap dan stabil dapat dipisahkan dan pola-pola yang mungkin sering diubah-ubah.

d. Adanya konsistensi

   Karena sifat pewarisan dan penggunaan notasi yang sama pada saat analisis, perancangan maupun pengkodean.

e. Meningkatkan kualitas perangkat lunak

   Karena adanya pendekatan pengembangan lebih dekat dengan dunia nyata dan adanya konsistensi pada saat pengambangannya, perangkat lunak yang

dihasilkan akan mampu memenuhi kebutuhan pemakai serta mempunyai sedikit kesalahan.

Berikut beberapa contoh bahasa pemrograman yang mendukung pemrograman berorientasi objek[24] :

a. *Smalltalk*

Smalltalk merupakan salah satu bahasa pemrogramanyang dikembangkan untuk mendukung pemrograman berorientasi objek mulai tahun 1978.

b. Bahasa Pemrograman Eiffel

Eiffel merupakan bahasa pemrograman yang dikembangkan untuk mendukung pemrograman berorientasi objek mulai tahun 1985 oleh Bertrand Meyer dan *compiler* Eiffel selesai pada tahun 1987.

c. Bahasa Pemrograman C++

C++ merupakan pengembangan lebih lanjut bahasa pemrograman C untuk mendukung pemrograman berorientasi objek.

d. Bahasa Pemrograman (web) PHP

PHP dibuat pertama kali oleh seorang perekayasa perangkat (*software engineering)* yang bernama Rasmus Lerdoff.

e. Bahasa Pemrograman Java

Java dikembangkan oleh perusahaan Sun Microsystem. Java menurut definisi dari Sun Microsystem adalah nama untuk sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada komputer *standalone* ataupun pada lingkungan jaringan[24].

2.2.5 Konsep Dasar Berorientasi Objek

Pendekatan berorientasi objek merupakan suatu teknik atau cara pendekatan dalam melihat permasalahan dan sistem (sistem perangkat lunak, sistem informasi, atau sistem lainnya). Pendekatan berorientasi obejek akan memandang sistem yang akan dikembangkan sebagai suatu kumpulan objek yang berkorespondensi dengan objek-objek dunia nyata[24].

Berikut ini adalah beberapa konsep dasar yang harus dipahami tentang metodelogi berorientasi objek:

1. Kelas (*class*)

Kelas adalah kumpulan objek-objek dengan karakteristik yang sama. Kelas merupakan definisi statis dan himpunan objek yang sama yang mungkin lahir atau tercipta dan kelas tersebut. Sebuah kelas akan mempunyai sifat (atribut), kelakuan (operasi/metode), hubungan (relationship) dan arti. Suatu kelas dapat diturunkan dan kelas yang lain, dimana atribut dan kelas semula dapat diwariskan ke kelas yang baru.

2. Objek (*object*)

Objek adalah abstraksi dan sesuatu yang mewakilkan dunia nyata seperti benda, manusia, satuan organisasi, tempat, kejadian, struktur, status, atau hal-hal lain yang bersifat abstrak. Objek merupakan entitas yang mampu menyimpan informasi (status) dan mempunyai operasi (kelakuan) yang dapat diterapkan atau dapat berpengaruh pada status objeknya. Objek mempunyai siklus hidup yaitu diciptakan, dimanipulasi, dan dihancurkan.

3. Metode (*method*)

Operasi atau metode pada sebuah kelas hampir sama dengan fungsi prosedur pada metodelogi structural. Sebuah kelas boleh memiliki lebih dari satu metode atau operasi yang berfungi untuk memanipulasi objek itu sendiri. Operasi atau metode merupakan fungsi atau transformasi yang dapat dilakukan terhadap objek atau dilakukan oleh objek.

4. Atribut (*attribute*)

Atribut dari sebuah kelas adalah variable global yang dimiliki sebuah kelas. Atribut dapat berupa nilai atau elemen-elemen data yang dimiliki oleh objek dalam kelas objek. Atribut dipunyai secara individual oleh sebuah objek, misalnya berat, jenis, nama dan sebagainya.

5. Abstraksi (*abstraction*)

Prinsip untuk merepresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan.

6. Enkapsulasi (*encapsulation*)

Pembungkusan atribut data dan layanan (operasi-operasi)yang dipunyai objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerjanya.

7. Pewarisan (*inheritance*)

Mekanisme yang memungkinkan suatu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dari dirinya.

8. Antarmuka (*Interface*)

Antarmuka (*Interface*) sangat mirip dengan kelas, tapi tanpa atribut kelas dan memiliki metode yang dideklarasikan tanpa isi. Deklarasi metode pada sebuah interface dapat diimplementasikan oleh kelas lain.

9. *Reusability*

Pemanfaatan kembali objek yang sudah didefinisikan untuk suatu permasalah pada permasalahan lainnya yang melibatkan objek tersebut

10. Generalisasi dan Spesialisasi

Menunjukkan hubungan antara kelas dan objek yang umum dengan kelas dan objek yang khusus. Misalnya kelas yang lebih umum (generalisasi) adalah kendaraan darat dan kelas khususnya (spesialisasi) adalah mobil, motor, dan kereta.

11. Komunikasi Antar Objek

Komunikasi Antar Objek dilakukan lewat pesan (message) yang dikirim dari satu objek ke objek yang lain.

12. Polimorfisme (*polymorphism*)

Kemampuan suatu objek untuk digunakan di banyak tujuan yang berbeda dengan nama yang sama sehingga menghemat baris program.

13. *Package*

Package adalah sebuah container atau kemasan yang dapat digunakan untuk mengelompokkan kelas-kelas sehingga memungkinkan beberapa kelas yang bernama sama disimpan dalam package yang berbeda

2.2.6 Analisis Berorientasi Objek

Analisis berorientasi objek atau *Object Oriented Analysis* (OOA) adalah tahap untuk menganalisis spesifikasi atau kebutuhan akan sistem yang akan dibangun dengan konsep berorientasi objek, apakah benar kebutuhan yang ada dapat diimplementasikan menjadi sebuah sistem berorientasi objek. Pendekatan analisa berarah objek terdiri dari lima pokok aktivitas, yaitu[24]:

1. Menentukan Kelas dan Objek

   Pengertian objek adalah suatu abstraksi dari suatu lingkup permasalahan / implementasi yang menggambarkan kemampuan dari sistem untuk menangkap informasi tentang objek tersebut, berinteraksi dengan atau keduanya; "pembung-kusan" pada nilai atribut dan *service*-nya. Kelas adalah suatu gambaran dari satu atau lebih objek, penggambaran dengan sekumpulan atribut dan service yang sama. Menggambarkan bagaimana untuk menciptakan objek-objek baru dalam suatu kelas.
2. Identifikasi Struktur

   Struktur adalah suatu ekspresi dari lingkup permasalahan yang komplek, berhubungan dengan tanggungjawab terhadap sistem. Istilah struktur digunakan sebagai sebuah istilah yang menggambarkan struktur generalisasi-spesialisasi (*gen-spec*) dan struktur keseluruhan dan bagiannya (*whole-part*).
3. Identifikasi Subjek

   Subjek adalah suatu mekanisme yang membantu pembaca untuk menganalisa model permasalahan dari yang kecil hingga ke yang besar dan komplek. Pada lapisan subjek, gambarkan setiap subjek sebagai kotak persegi yang sederhana, dengan nama subjek dan nomor didalamnya sebagai tambahan, daftarkan kelas-kelas yang termasuk didalam subjek tersebut. Pada lapisan

yang lain indikasikan subjek dengan label subjek pada sekat pemisah kotak subjek untuk memandu pembaca dari subjek ke subjek lainnya. Dalam model yang besar dan membutuhkan fasilitas komunikasi, pertimbangkan penggunaan "set" dari lapisan secara terpisah untuk setiap subjek. Sebuah kelas dari subjek berada didalam lebih dari satu subjek. Subjek boleh berisi subjek lainnya. Persiapkan pemetaan multi *level* untuk membantu pembaca dalam suatu model yang besar.

4. Pendefinisian Atribut

   Atribut adalah beberapa data (*state information*) dimana setiap objek di dalam suatu kelas mempunyai nilai tersendiri.

5. Pendefinisian Service

   Service adalah tingkah laku yang spesifik, yaitu tingkah laku yang memperlihatkan "tanggungjawabnya" terhadap sebuah objek.

OOA biasanya menggunakan kartu CRC (Component, Responsibility, Collaborator) untuk membangun kelas-kelas yang akan digunakan atau menggunakan UML (Unified Modeling Language) pada bagian diagram use case, diagram kelas, dan diagram objek.[24].

2.3 UML (Unified Modelling Language)

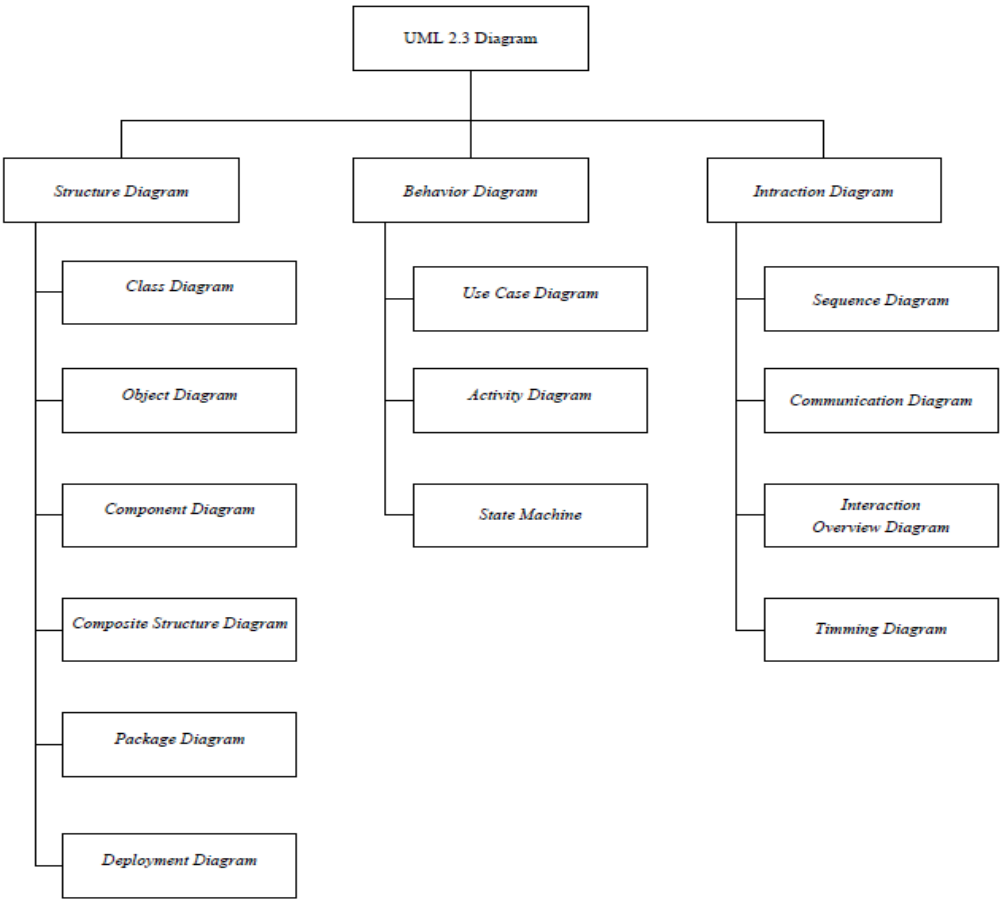
Unified Modelling Language (UML) adalah sebuah “bahasa” yang telah menjadi standar dalam industry visualisasi, merancang, dan mendokumentasikan sistem peranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

UML dapat digunakan untuk membuat model untuk semua jenis perangkat lunak, dimana perangkat lunak tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan class dan operation dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa berorientasi objek.

Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan syntax. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML syntax mendefinisikan begaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya yaitu Grady Booch OOD (Object Oriented Design), Jim Rumbaugh OMT (Object Modeling Technique) dan Ivar Jacobson OOSE (Object Oriented Software Engineering).

2.3.1 Diagram UML

Pada UML 2.6 terdiri dari 13 macam diagram yang dikelompokkan dalam 3 kategori. Pembagian kategori dan macam-macam diagram tersebut dapat dilihat pada gambar di bawah :



Gambar 2.7 Diagram UML[19]

Berikut ini penjelasan singkat dari pembagian katagori 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 antar subsistem pada suatu sistem.

2.3.2 *Use Case Diagram*

Use case atau diagram *use case* merupakan pemodelan untuk kelakuan (*behavior*) 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 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*.

Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang dapat dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupkan orang. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit aktor[19].

Berikut adalah simbol-simbol yang ada pada diagram use case dapat dilihat pada tabel 2.1 :

2.3.3 Activity Diagram

Diagram aktivitas atau *Activity Diagram* menggambarkan workflow (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. 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 di mana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
2. Urutan atau pengelompokan tampilan dari sistem /user interface di mana setiap aktivitas memiliki sebuah rancangan antarmuka tampilan.
3. Rancangan pengujian di mana setiap aktivitas memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.

Berikut adalah simbol-simbol yang ada pada diagram aktivitas tabel 2.2 :

2.3.4 Sequance Diagram

Diagram sekuen menggambarkan kelakuan objek pada use case dengan mendeskrifsikan waktu hidup objek dan message yang dikirimkan dan diterima antarobjek. 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.

Banyaknya diagram sekuen yang harus digambar adalah 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 sehinga semakin banyak use case yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak[19].

Berikut adalah simbol-simbol yang ada pada diagram sekuen pada tabel 2.3 :

2.3.5 Class Diagram

Diagram kelas atau class diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas mimiliki apa yang disebut atribut dan metode atau operasi[19].

a. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas.
b. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

Berikut adalah simbol-simbol yang ada pada diagram kelas pada tabel 2.4

2.4 *Tools* yang digunakan

Tools adalah unsur penting dalam pembangunan sebuah aplikasi. Ada beberapa *tools* yang akan digunakan disini, diantaranya :

1. Unity

   Unity merupakan sebuah *game engine* dan *editor*, yang memungkinkan pengembang membuat *object*, meng-*import external asset* dan menggabungkan semuanya dengan menggunakan kode secara efisien dan cepat dengan versi 4.3. Prinsip dari Unity sendiri adalah agar pengembang dapat melakukan aktivitasnya dengan menggabungkan script dan membuat beberapa variabel. Unity juga mendukung kemampuan *networking* dan dapat digunakan pada banyak *platform*. *Scripting* di dalam Unity dapat menggunakan bahasa pemrograman, yaitu C#. Pengembang *game* dapat menggunakan bahasa pemrograman tersebut dalam satu proyek miliknya. Beberapa konsep penting di dalam scripting yang harus pengembang mengerti adalah[20]:

   a. *Function Update*(){}

      Merupakan *container* dari kode yang dieksekusi berkali-kali setiap detik (per –*frame*).

   b. *Transform*

Merupakan *refence* untuk komponen *transform game object*, yang mendefinisikan posisi, rotasi, dan skala dari setiap *game object*. Setiap *object* memiliki *transform*.

c. *Rotate*()

Merupakan sebuah fungsi yang ada pada komponen *transform*. Angka-angka yang berada di dalam koma merepresentasikan derajat dari rotasi untuk setiap axis di dalam 3D space : X,Y,Z.

d. Time.deltaTime

Merupakan anggota dari kelas Time yang berjalan pada setiap detik, sehingga sebuah *game object* dapat berputar dengan kecepatan yang sama seberapa pun frame per detik yang dapat dihasilkan setiap detik. Misalnya : 5 * Time.deltaTime artinya 5 derajat perdetik

e. C#

C# (baca : *See-Sharp*) adalah bahasa pemrograman baru yang diciptakan Microsoft yang digunakan oleh banyak developer .NET untuk mengembangkan aplikasi dengan *platform* .NET.

2. Adobe photoshop

Adobe Photoshop, atau biasa disebut Photoshop, adalah perangkat lunak editor citra buatan Adobe Sistems yang dikhususkan untuk pengeditan foto/gambar dan pembuatan efek. Photoshop memiliki kemampuan untuk membaca dan menulis gambar berformat raster dan vektor seperti .png, .gif, .jpeg, dan lain-lain. Photoshop juga memiliki beberapa format file khas[22]:

a. PSD (Photoshop Document) format yang menyimpan gambar dalam bentuk layer, termasuk teks, mask, opacity, blend mode, channel warna, channel alpha, clipping paths, dan setting duotone. Kepopuleran photoshop membuat format file ini digunakan secara luas, sehingga memaksa programer program penyunting gambar lainnya menambahkan kemampuan untuk membaca format PSD dalam perangkat lunak mereka.

b. PSB' adalah versi terbaru dari PSD yang didesain untuk file yang berukuran lebih dari 2 GB.
c. PDD adalah versi lain dari PSD yang hanya dapat mendukung fitur perangkat lunak Photshop Deluxe