BAB 2
LANDASAN TEORI

2.1 *Game*

Game adalah permainan yang menggunakan media elektronik, merupakan sebuah hiburan berbentuk multimedia yang dibuat semenarik mungkin agar pemain bisa mendapatkan sesuatu sehingga adanya kepuasan batin.

G*ame* adalah aktivitas terstuktur atau semi terstuktur yang biasanya dilakukan untuk bersenang-senang dan kadang juga digunakan sebagai alat pembelajaran. Sebuah *game* bisa di karakteristik dari "apa yang pemain lakukan". Hal tersebut adalah sebagai berikut:[5]

a. Peralatan, seperti bola, kartu, atau sebuah komputer.
b. Peratuan bisa menentukan giliran permainan, hak dan kewajiban pemain dan tujuan permainan.
c. *Skill* strategi dan keburuntungan. *Game* dengan *skill* misalnya kekuatan fisik dan kekuatan mental.

2.1.1 Klasifikasi *Game*

2.1.1.1 Berdasarkan Jenis *Platform* yang Digunakan

Berdasarkan jenis platform yang digunakan *game* dibagi menjadi beberapa jenis, yaitu:[7]

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 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 konsol. 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, iPosd, dan *handheld game* seperti Nintendo Dsi atau Sony PSP. Game mobile memiliki kontrol yang sederhana jika dibandingkan dengan PC.

2.1.1.2 Berdasarkan *Genre* Permainannya

Menurut buku "*Fundamentals of Game Design*", *genre game* bisa dibagi menjadi 9 [7], yaitu:

1. *Action*

   Action game adalah *game* dimana kebanyakan dari tantangan yang disajikan merupakan dari tes *physical skill* dan koordinasi pemain. Salah satu sub-*genre action game* adalah *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/ Platform*

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

8. *Artificial Life and Puzzle Game*

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

9. *Online Game*

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

2.1.1.3 Elemen *Game*

Elemen-elemen yang ada dalam sebuah game biasanya terdiri dari : [18]

1. ***Game Rule***

   Game rule merupakan aturan perintah, cara menjalankan, fungsi objek dan karakter di dunia permainan. Dunia *game* bisa berupa pulau, dunia khayal, dan tempat-tempat lain yang sejenis yang dipakai sebagai setting tempat dalam permainan *game*.

2. **Plot**

   Plot biasanya berisi informasi tentang hal-hal yang akan dilakukan oleh *player* dalam *game* dan secara *detail*, perintah tentang hal yang harus dicapai dalam *game*.

3. **Tema**

   Di dalam biasanya ada pesan moral yang akan disampaikan.

4. ***Character***

   Pemain sebagai karakter utama maupun karakter yang lain yang memiliki ciri dan sifat tertentu

5. ***Object***

   Merupakan sebuah hal yang penting dan biasanya digunakan pemain untuk memecahkan masalah, adakalanya pemain harus punya keahlian dan pengetahuan untuk bisa memainkannya.

6. ***Text, Grafik* dan *Sound***

   Game biasanya merupakan kombinasi dari media teks, grafik maupun suara, walaupun tidak harus semuanya ada dalam permainan *game*.

7. **Animasi**

   Animasi ini selalu melekat pada dunia *game*, khususnya untuk gerakan karakter, karakter yang ada dalam *game*, properti dari objek.

8. ***User Interface***

Merupakan fitur-fitur yang mengkomunikasikan *user* dengan *game*.

2.2 Atletik

Atletik merupakan salah satu unsur dari Pendidikan Jasmani dan Kesehatan juga merupakan komponen-komponen pendidikan keseluruhan yang mengutamakan aktivitas jasmani serta pembinaan hidup sehat dan pengembangan jasmani, mental, sosial dan emosional yang serasi, selaras, dan seimbang [9]. Sedangkan menurut Eddy Purnomo dan Dapan [10] bahwa atletik merupakan aktivitas jasmani yang terdiri dari gerakan-gerakan dasar yang dinamis dan harmoni, yaitu jalan, lari, lompat, dan lempar. Jadi atletik merupakan salah satu aktivitas fisik yang diperlombakan atau dipertandingkan dalam bentuk kegiatan jalan, lari, lempar, dan lompat.

2.2.1 *Sprint*

Sprint adalah semua perlombaan lari dimana peserta lari dengan kecepatan penuh sepanjang jarak yang harus ditempuh. Sampai pada jarak 400 masih dapat dikategorikan *sprint* [11].

kelangsungan gerak pada *sprint* secara teknik sama, kalau ada perbedaan hanyalah terletak pada penghematan penggunaan tenaga karena perbedaan jarak yang harus ditempuh. Makin jauh jarak yang harus ditempuh makin membutuhkan daya tahan yang besar, sehingga ada yang dinamakan *edurance*. Teknik lari jarak pendek terbagi menjadi tiga, yaitu *start* jongkok, gerakan lari, dan teknik memasuki garis *finish*.

2.2.2 Lari Gawang

Lari gawang merupakan salah satu nomor lari, akan tetapi menggunakkan gawang sebagai rintangan yang harus dilalui oleh pelari tanpa harus kehilangan kecepatan lari saat melewati gawang atau rintangan itu [12].

Gawang-gawang untuk lari gawang minimal 80 buah ukuran gawang untuk :

a. 100 meter gawang tinggi 0,838 meter ukuran putri.
b. 110 meter gawang tinginya 1,067 meter ukuran putra.
c. 400 meter gawang tingginya 0,762 meter ukuran putri.
d. 400 meter gawang tingginya 0,914 meter ukuran putra.
e. Lebar gawang maksimum 1,20 meter.
f. Panjang maksimum bagian dasar 0,70 meter.
g. Berat total sebuah gawang tidak boleh lebih dari 10 kg.
h. Lebar palang kayunya 7 cm, tebalnya 1-2 cm, tepi dari palang tersebut harus dibulatkan agar tidak tajam.
i. Palang tersebut harus di cat belang putih dan hitam dengan ketentuan warna hitamnya selebar 22,50 cm, kedua tepinya harus berwarna putih.

2.2.3 Lompat Tinggi

Lompat tinggi adalah termasuk ke dalam lompatan vertikal, karena si pelompat berusaha memindahkan titik berat badan setinggi-tingginya dalam upaya melampaui suatu ketinggian (mistar lompat) [12].

Tinggi tiang mistar minimal 2.5 meter, panjang mistar 3.15 meter.

1. Tahapan pada lompat tinggi

   Semua gaya lompatan dapat dibedakan menjadi 4 tahap, yaitu :

   a. Awalan, gerakan berlari menuju mistar.
   b. Tolakan, gerakan kaki menumpu pada lantai untuk menaikkan badan.
   c. Melayang, gaya dan kedudukan badan ketika berada di udara dan di atas mistar.
   d. Mendarat, jatuhnya badan diatas matras.

2. Gaya dalam lompat tinggi

   1. Gaya Gunting (***Scissors***)

      Gaya gunting bisa dikatakan Gaya *Sweney*, sebab pada waktu sebelumnya masih digunakan gaya jongkok. Tepatnya tahun 1880,

selanjutnya tahun 1896 *sweny* mengubahnya dari gaya jongkok menjadi gaya gunting. Diganti karena kurang ekonomis.

Cara melakukan: Si pelompat mengambil awalan dari tengah. Bila pelompat pada saat akan melompat, tumpuan pakai kaki kiri (bila ayunan kaki kanan), maka ia mendarat (jatuh) dengan kaki lagi. Waktu di udara badan berputar ke kanan, mendarat dengan kaki kiri, badan menghadap kembali ke tempat awalan tadi.

2. Gaya Guling sisi (***Western Roll***)

Pada gaya ini sama dengan gaya gunting, yaitu tumpuan kaki kiri jatuh kaki kiri lagi dan bila kaki kanan jatuhnya pun kaki kanan hanya beda awalan, tidak dari tengah tapi dari samping.

3. Gaya Guling (***Straddle***)

Pelompat mengambil awalan dari samping antara 3, 5, 7, 9 langkah Tergantung ketinggian yang penting saat mengambil awalan langkahnya ganjil.

Menumpu pada kaki kiri atau kanan, maka ayunan kaki kiri atau kanan kedepan. Setelah kaki ayun itu melewati mistar cepat badan dibalikkan, hingga sikap badan diatas mistar telungkap. Pantat usahakan lebih tinggi dari kepala, jadi kepala nunduk. Pada waktu mendarat atau jatuh yang pertama kali kena adalah kaki kanan dan tangan kanan bila tumpuan menggunakan kaki kiri, lalu bergulingnya yaitu menyusur punggung tangan dan berakhir pada bahu.

2.2.4 Lompat Jauh

Lompat jauh adalah suatu bentuk gerakan melompat mengangkat kaki ke atas ke depan dalam upaya membawa titik berat badan selama mungkin di udara (melayang di udara) yang dilakukan dengan cepat dengan jalan melakukan tolakan pada satu kaki untuk mencapai jarak sejauh-jauhnya [13].

Gerak lompat jauh merupakan gerakan dari perpaduan antara kecepatan (*speed*), kekuatan (*stenght*), kelenturan (*flexibility*), daya tahan (*endurance*), dan ketepatan (*acuration*).

Para peneliti membuktikan bahwa suatu prestasi lompat jauh tergantung pada kecepatan daripada awalan atau ancang-ancang. Oleh karenanya di samping memiliki kemampuan *sprint* yang baik harus didukung juga dengan kemampuan dari tolakan kaki atau tumpuan.

Lompat jauh merupakan salah satu nomor lompat selain lompat jangkit, lompat tinggi, dan lompat tinggi galah. Tujuan lompat jauh adalah lompat melompat sejauh-jauhnya dengan memindahkan seluruh tubuh dari titik-titik tertentu ke titik lainnya, dengan cara berlari secepat-cepatnya kemudian menolak, melayang di udara dan mendarat.

Ada tiga cara sikap melayang di udara (gaya) dalam lompat jauh, diantaranya:

1. Gaya jongkok (waktu melayang bersikap jongkok)
2. Gaya lenting (waktu di udara badan dilentingkan) atau sering disebut gaya menggantung.
3 .Gaya berjalan diudara (waktu di udara kaki bergerak seolah-olah berjalan).

2.2.5 Lempar Lembing

Lempar lembing terdiri dari dua kata yaitu lempar dan lembing. Lempar yang berarti usaha untuk membuang jauh-jauh, dan lembing adalah tongkat yang berujung runcing yang dibuang jauh-jauh [14].

Ada beberapa teknik dasar yang harus dikuasai oleh atlet lempar lembing, yaitu :

a. Memegang lembing
b. Cara membawa lembing
c. Lempar lembing tanpa awalan
d. Lempar lembing dengan awalan

2.5.5.1 Gerakan Melempar

Saat kaki kiri mendarat, kaki kanan ditekuk hingga badan benar-benar jauh condong ke belakang dan badan sebagian besar pada kaki kanan. Pada saat ini lengan yang membawa lembing sudah dalam sikap lurus serong ke bawah,

mata lembing dan pandangan terarah kesudut lemparan dan tangan kiri tetap rileks. Saat inilah terjadi sikap melempar yang sebenarnya. Setelah lembing ditarik melaui pundak/bahu mendekat telinga, seluruh badan ditinggikan dan dengan secepat-cepatnya melecutkan lembing. Bersamaan dengan itu lepasnya lembing dengan hentakan pergelangan tangan sebagai sumber kekuatan terakhir.

2.3 AI yang digunakan

2.3.1 Algortima Fuzzy

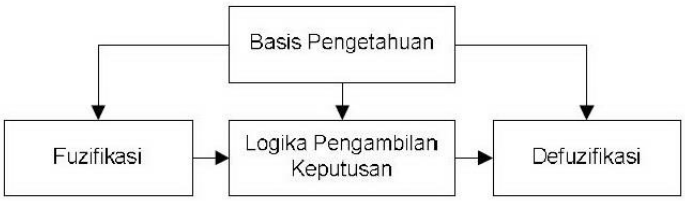
Konsep tentang logika fuzzy diperkenalkan oleh Prof. LotfiAstorZadeh pada 1962. Logika fuzzy adalah metodologi sistem kontrol pemecahan masalah, yang cocok diimplementasikan pada sistem, mulai dari sistem yang sederahana, sistem kecil, embeddedsystem, jaringan PC, multi-channel atau workstation berbasis 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 memungkinkan nilai kenaggotaan 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).

Konsep himpunan fuzzy memiliki 2 atribut, yaitu :[15]

1) Linguistik, yaitu nama suatu kelompok yang mewakili suatu keadaan tertentu dengan menggunakan bahasa alami, misalnya dingin, sejuk, panas mewakili variable temperatur.

2) Numeris, yaitu suatu nilai yang menunjukan ukuran dari suatu variabel, misalnya 10, 35, 40 dan sebagainya.

Struktur sistem inferensi *fuzzy* dapat dilihat pada gambar 2.1



Gambar 2.1 Struktur sistem inferensi *fuzzy* [15]

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*.

2.3.2 Himpunan Fuzzy

Himpunan adalah suatu kumpulan atau koleksi objek-objek yang mempunyai kesamaan sifat tertentu. Himpunan fuzzy merupakan suatu pengembangan lebih lanjut tentang konsep himpunan dalam matematika. Himpunan fuzzy adalah rentang nilai-nilai, masing-masing nilai mempunyai derajat keanggotaan antara 0 sampai dengan 1. Suatu himpunan fuzzy A dalam semesta pembicaraan U dinyatakan dengan fungsi keanggotaan μA, yang nilainya berada dalam interval [0,1], dapat dinyatakan dengan: $\mu A : U \rightarrow [0,1]$

Himpunan fuzzy A dalam semesta pembicaraan U biasa dinyatakan sebagai sekumpulan pasangan elemen u (u anggota U) dan derajat keanggotaannya dinyatakan sebagai berikut:

$A = \{(u, \mu A (u) \mid u \in U\}$.

Cara tradisional untuk menyatakan apakah sebuah obyek merupakan anggota sebuah himpunan atau tidak adalah dengan menggunakan fungsi karakteristik (kadang-kadang disebut juga dengan fungsi diskriminasi). Jika sebuah obyek merupakan anggota dari sebuah himpunan maka fungsi karakteristiknya 1. Jika sebuah obyek bukan merupakan anggota dari sebuah himpunan maka fungsi karakteristiknya 0. Fungsi karakteristik dapat didefinisikan menggunakan pemetaan fungsional sebagai berikut:

$\mu_A[x] : X \rightarrow \{0,1\}$

yang menyatakan bahwa fungsi karakteristik memetakan himpunan universal X ke himpunan yang terdiri dari 0 dan 1. Himpunan dimana fungsi ini berlaku disebut himpunan *crisp*. Pada himpunan fuzzy, sebuah obyek dapat berada pada sebuah himpunan secara parsial. Derajat keanggotaan dalam himpunan fuzzy diukur dengan fungsi yang merupakan generalisasi dari fungsi karakteristik yang disebut fungsi keanggotaan atau fungsi kompatibilitas, yang didefinisikan sebagai:

$\mu_A[x] : X \rightarrow \{0,1\}$

Fungsi keanggotaan memetakan X kedalam kodomain yang merupakan bilang riil yang terdefinisi pada interval dari 0 sampai 1 secara inklusif. Dimana 0 berarti tidak ada keanggotaan, dan 1 berarti keanggotaan penuh pada himpunan A. Sebuah nilai pada fungsi keanggotaan, misalnya 0.6, disebut sebagai nilai keanggotaan.

2.4 *Object Oriented Programing*

OOP adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya. Metode OOP didasarkan pada penerapan prinsip-prinsip pengelolaan kompleksitas. Metode ini meliputi rangkaian aktivitas analisis berorientasi objek, perancangan berorientasi objek, pemograman berorientasi objek, dan pengujian berorientasi objek [8]. Ketika sebuah program berjalan, objek individu biasanya tidak dapat berdiri sendiri.
Keuntungan menggunakan metodologi berorientasi objek adalah sebagai berikut :

1. Meningkatkan produktivitas
   Karena kelas dan objek yang ditemukan dalam suatu masalah masih dapat dipakai ulang untuk masalah lainnya yang melibatkan objek tersebut.
2. Kecepatan pengembangan
   Karen sistem yang dibangun dengan baik dan benar pada saat analisis dan perancangan akan menyebabkan berkurangnya kesalahan pada saat pengkodean.
3. Kemudahan pemeliharaan
   Karena dengan model objek, pola-pola yang cenderung tetap dan stabil dapat dipisahkan dan pola-pola yang mungkin sering berubah-ubah.
4. Adanya konsistensi
   Karena sifat pewarisan dan penggunaan notasi yang sama pasa saat analisis, perancangan maupun pengkodean.
5. Meningkatkan kualitas perangkat lunak
   Karena pendekatan pengembangan lebih dekat dengan dunia nyata dan adanya konsistenso pada saat pengembangannya, perangkat lunak dihasilkan akan mampu memenuhi kebutuhan pemakai serta mempunyai sedikit kesalahan [8].

Konsep dasar dari sebuah OOP adalah sebagai berikut :

1. Objek, yaitu benda di dunia nyata yang bisa dibedakan antara satu dengan yang lainnya. Objek mempunyai identitas, property, dan tingkah laku.
2. Kelas,yaitu sekumpulan objek yang memiliki kemiripan dalam hal properti, atribut, *behaviour*, dan semantik.
3. Abstraksi, yaitu mekanisme yang memungkinkan situasi dunia nyata dapat diwakili menggunakan model yang disederhanakan. Orientasi objek abstrak dunia nyata didasarkan pada objek dan interaksi antar objek lainnya.
4. Enkapsulasi dikenal sebagai informasi yang tersembunyi juga berperan pada pembungkusan properti dan *behaviour*. Enkapsulasi ini memastikan pengguna sebuah objek tidak dapat mengganti keadaan dari sebuah objek dengan cara yang tidak layak.
5. Polimorfisme, yaitu karakteristik sistem orientasi objek. Ketika warisan digunakan untuk memperpanjang *class* umum ke *class* yang lebih khusus, biasanya akan mencakup beberapa perilaku dari *class* umum. Perilaku dari *class* khusus ini akan berbeda dengan *class* umum, tetapi nama yang digunakan untuk mendefinikan perilaku tersebut akan menjadi sama.
6. Inheritas, yaitu mengatur polimorfisme dan enkapsulasi dengan mengijinkan objek didefinisikan dan diciptakan dengan jenis khusus dari objek yang sudah ada objek-objek ini dapat membagi dan memperluas perilaku mereka tanpa harus mengimplementasi ulang perilaku tersebut (bahasa berbasis-objek tidak selalu memiliki inheritas).

2.5 *Unified Modeling Language* (UML)

UML adalah salah satu bentuk *language* atau bahasa untuk menggambarkan *object oriented analysis and design.* Sebagian besar metode terdiri bahasa dan proses. UML disebut bahasa pemodelan bukan metode. Bahasa pemodelan

adalah notasi (terutama grafis) bahwa metode digunakan untuk mengekspresikan desain.



Gambar 2.2 Logo UML

2.5.1 Sejarah UML

Bahasa pemograman berorientasi objek yang pertama dikembangkan dikenal dengan nama Simula-67 yang dikembangkan pada tahun 1967. Bahasa pemograman ini kurang berkembang dan dikembangkan lebih lanjut, namun kemunculannya menjadi titik besar bagi developer pengembang bahasa pemograman berorientasi objek selanjutnya.

Karena banyaknya metodologi yang berkembang pesat pada saat itu, maka muncullah ide untuk membuat sebuah bahasa yang dapat dimengerti semua orang. Maka dibuatlah bahasa gabungan dari beberapa konsep seperti *Object Modelling Technique (OMT)* dari Rumbaugh dan Booch (1991), konsep *The Classes, Responsibilities, Collaborators (CRC)* dari Rebecca Wirfs-Brock (1990), konsep pemikiran Ivan Jacobson, dan beberapa konsep lainnya dimana James R. Rumbaigh, Grady Booch, dan Ivar Jacobson bergabung dalam sebuah perusahaan yang bernama *Rational Software Corporation* menghasilkan bahasa yang disebut dengan *Unified Modeling Language (UML)* [8].

2.5.2 Diagram UML

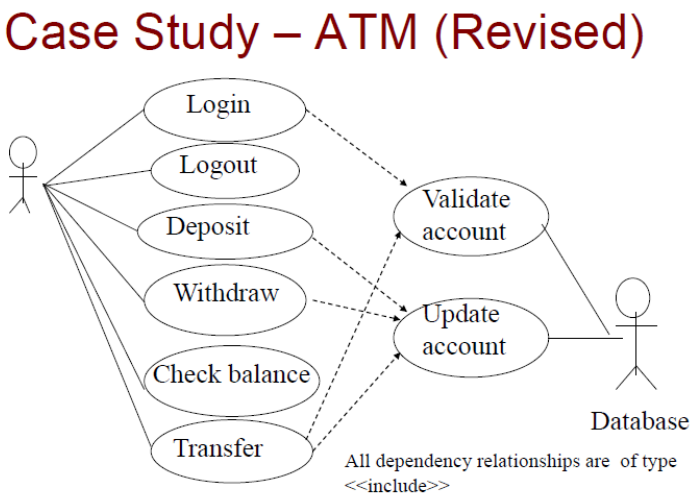
Menggunakan berbagai macam diagram dengan fungsi masing-masing untuk menggambarkan setiap proses dari sistem berorientasi objek. Berikut merupakan beberapa diagram UML diantaranya:

A. *Use Case Diagram*

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

1. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem yang akan dibuat diluar sistem yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
2. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antarunit atau aktor.

Contoh dari *use case* diagram dapat dilihat pada gambar 2.3 berikut :



Gambar 2.3 Contoh dari *Use Case Diagram*

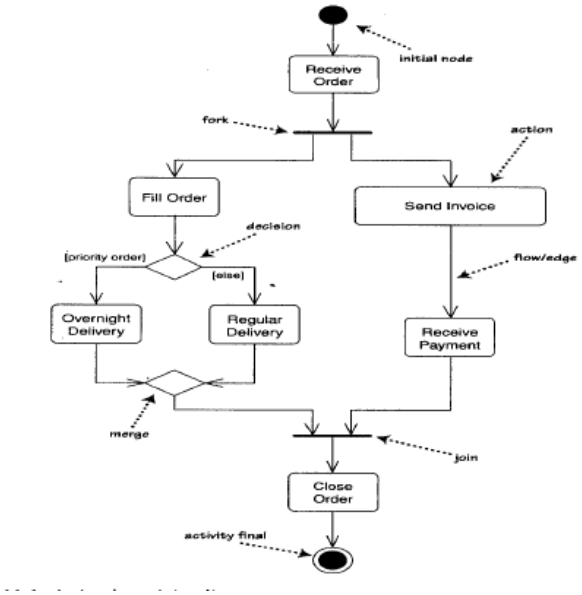
B. *Activity Diagram*

Diagram aktivitas atau *activity diagram* adalah sebuah diagram yang menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Dalam diagram aktivitas yang perlu diperhatikan 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 pengelompokan tampilan dari sistem *user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antaruka tampilan.
3. Rancangan pengunjian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.

Contoh dari *activity diagram* dapat dilihat pada gambar 2.4 berikut :



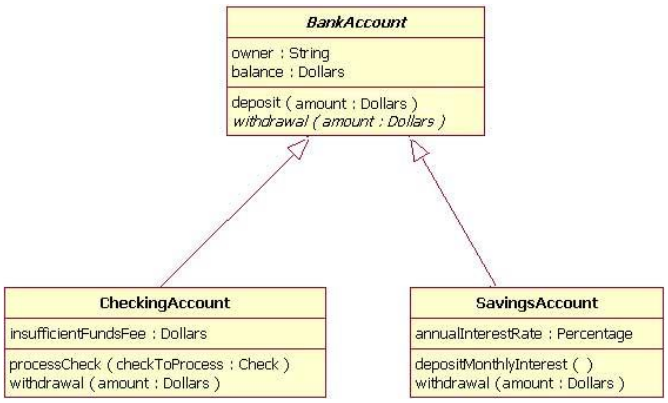
Gambar 2.4 Contoh dari *Activity Diagram*

C. *Class Diagram*

Diagram kelas atau *class* diagram 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

Contoh dari *class diagram* dapat dilihat pada gambar 2.5 berikut :



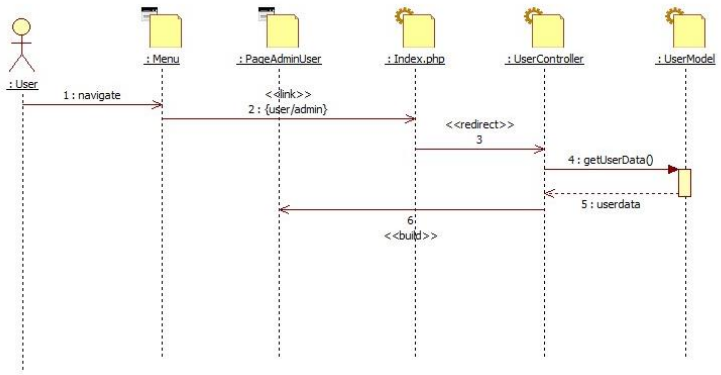
Gambar 2.5 Contoh dari *Class Diagram*

D. *Sequence Diagram*

Sequence diagram atau diagram sekuen adalah diagram yang menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antarobjek. Oleh karena itu untuk menggambarkan diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang dinstansiasi menjadi objek itu.

Banyaknya diagram sekuen yang harus digambarkan adalah sebanyak pendefinisian *use case* yang memiliki proses itu 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.

Contoh dari *sequence diagram* dapat dilihat pada gambar 2.6 berikut.



Gambar 2.6 Contoh dari *Sequence Diagram*

2.6 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 [17].

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*.

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.*

2.7 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.

2.8 Adobe PhotoShop

Adobe Photoshop adalah suatu program gambar atau manipulasi *image* yang diciptakan oleh Adobe system Incorporate. Aplikasi serupa yang dapat anda temui diantaranya Corel Draw, Macromedia Freehand, dan Microsoft photo editor.

Adobe Photoshop dapat dikatakan sebagai *sofware* manipulasi *image* yang paling lengkap fasilitasnya.

Aplikasi adobe photoshop bekerja dengan metode pemisah setiap komponen gambar menjadi layer yang berbeda, hal ini akan sangat memudahkan kreasi dalam melakukan proses penyuntingan gambar [16].

Meskipun pada awalnya Photoshop dirancang untuk menyunting gambar untuk cetakan berbasis-kertas, Photoshop yang ada saat ini juga dapat digunakan untuk memproduksi gambar untuk World Wide Web. Beberapa versi terakhir juga menyertakan aplikasi tambahan, Adobe ImageReady, untuk keperluan tersebut.

Photoshop juga memiliki hubungan erat dengan beberapa perangkat lunak penyunting media, animasi, dan *authoring* buatan-Adobe lainnya. File format asli Photoshop, PSD, dapat diekspor ke dan dari Adobe ImageReady, Adobe Illustrator, Adobe Premiere Pro, After Effects dan Adobe Encore DVD untuk membuat DVD profesional, menyediakan penyuntingan gambar non-*linear* dan layanan *special effect* seperti *background*, *tekstur*, dan lain-lain untuk keperluan televisi, film, dan situs web. Sebagai contoh, Photoshop CS dapat digunakan untuk membuat menu dan tombol (*button*) DVD.

2.9 Metode Pengujian Sistem

Metode pengujian sistem untuk mengetahui efektifitas dari *software* yang digunakan selain memberikan kesempatan kepada pengguna untuk mengoperasikan dan melakukan pengecekan terhadap laporan yang dihasilkan melalui *software*. Metode pengujian sistem terdiri dari Pengujian *Whitebox* dan Pengujian *Blackbox* [5].

2.9.1 Pengujian *Whitebox*

Pengujian *whitebox* dilakukan untuk menguji prosedur-prosedur yang ada. *Logic path* yang dilalui oleh setiap bagian prosedur diuji dengan memberikan kondisi/loop spesifik. Pengujian *whitebox* menjamin pengujian terhadap semua lintasan yang tidak bergantungan minimal satu kali, mencoba semua keputusan lojik dari sisi '*true*' dan '*false*', eksekusi semua *loop* dalam batasan kondisi dan

batasan operasionalnya dan pengujian validasi data internal. Konsep Pengujian Basis *Path* sebagai berikut*:*

1. Merupakan bagian dari pengujian *whitebox* dalam hal pengujian prosedur-prosedur.
2. Mempergunakan notasi aliran *graph* (*node, link* untuk merepresentasikan *sequence, if, while, until* dan sebagainya).
3. Konsep kompleksitas *cyclomatic* antara lain cara perhitungan daerah tertutup pada graph planar dimana dapat menghubungkan batas atas jumlah pengujian yang harus direncanakan dan dieksekusi untuk menjamin pengujian seluruh *statement* program.
4. Memunculkan kasus-kasus yang akan diuji dengan membuat daftar lintasan kasus pengujian berdasarkan kompleksitas dan *cyclomatic* yang didapat.
5. Membuat alat bantu graph matriks yang membantu pengawasan pengujian.

2.9.2 Pengujian *Blackbox*

Pengujian yang dilakukan untuk antarmuka perangkat lunak, pengujian ini dilakukan untuk memperlihatkan bahwa fungsi-fungsi bekerja dengan baik dalam arti masukan yang diterima dengan benar dan keluaran yang dihasilkan benar-benar tepat, pengintegrasian dari eksternal data berjalan dengan baik.

Metode pengujian *blackbox* memfokuskan pada *requirement* fungsi dari perangkat lunak, pengujian ini merupakan komplenetari dari pengujian *whitebox*. Pengujian *whitebox* dilakukan terlebih dahulu pada proses pengujian, sedangkan pengujian *blackbox* dilakukan pada tahap akhir dari pengujian perangkat lunak. Proses yang terdapat dalam proses pengujian *blackbox* antara lain sebagai berikut:

1. Pembagian kelas data untuk pengujian setiap kasus yang muncul pada pengujian *whitebox.*
2. Analisis batasan nilai yang berlaku untuk setiap data

2.10 Skala Likert

Menurut Sugiyono [6], Skala *Likert* digunakan untuk mengukur sikap, pendapat dan persepsi seseorang atau sekelompok orang tentang fenomena sosial. Dalam penelitian fenomena sosial ini telah ditetapkan secara spesifik oleh peneliti yang selanjutnya disebut sebagai variabel penelitian. Dengan skala *likert*, maka variabel yang akan diukur dijabarkan menjadi indikator variabel. Kemudian indikator tersebut dijadian sebagai titik tolak untuk menyusun item-item instrumen yang dapat berupa pernyataan atau pertanyaan. Jawaban setiap item instrumen yang menggunakan skala likert mempunyai gradasi dari sangat positif sampai sangat negatif.

Jawaban setiap item instrumen yang menggunakan skala *Likert* mempunyai gradasi dari sangat positif sampai negatif yang dapat berupa kata-kata antara :

1. Sangat setuju
2. Setuju
3. Biasa Saja
4. Kurang Setuju
5. Tidak Setuju

1. Setuju
2. Sering
3. Kadang-kadang
4. Hampir tidak pernah
5. Tidak pernah

Untuk keperluan analisis kuantitatif, maka jawaban itu dapat diberi skor, misalnya :

1	Sangat Setuju/Selalu/Sangat positif diberi skor	5
2	Setuju/Sering/Positif diberi skor	4
3	Ragu – Ragu/ kadang – kadang / Netral diberi skor	3
4	Tidak Setuju/ Hampir tidak pernah / Negatif diberi skor	2
5	Sangat Tidak Setuju / Tidak Pernah/Sangat Negatif diberi skor	1