BAB 2
LANDASAN TEORI

2.1 Game

Game atau permainan merupakan media hiburan yang sudah dikenal sejak dahulu. *Game* dapat dimainkan berbagai umur tua maupun muda. *Game* juga sudah berkembang dengan pesat sesuai dengan perkembangan teknologi, dari *game* sederhana sampai *game* modern saat ini. Hal ini terbukti dengan adanya perkembangan jenis, produk, serta alat yang digunakan.

2.1.1 Pengertian Game

Game merupakan sebuah bentuk seni dimana penggunanya disebut dengan pemain (*player*), diharuskan membuat keputusan-keputusan dengan tujuan mengelola sumber daya yang diperoleh dari kesempatan-kesempatan bermain (*token*) miliknya untuk mencapai tujuan tertentu. *Video game* adalah bentuk *game* yang interaksi umumnya melibatkan media *video* dan *audio*.

Menurut Andang Ismail terdapat dua pengertian *game* (permainan). Pertama, *game* (permainan) adalah sebuah aktifitas bermain yang murni mencari kesenangan tanpa mencari menang atau kalah. Kedua, permainan diartikan sebagai aktifitas bermain yang dilakukan dalam rangka mencari kesenangan dan kepuasan, namun ditandai pencarian menang-kalah.

Berdasarkan representasinya, *game* dapat dibedakan menjadi 2 jenis yaitu *game* 2 dimensi (2D) dan 3 dimensi (3D). *Game* 2D adalah *game* yang secara matematis hanya melibatkan 2 elemen koordinat kartesius yaitu x dan y, sehingga konsep kamera pada *game* 2D hanya menentukan gambar pada *game* yang dapat dilihat oleh pemain. Sedangkan *game* 3D adalah *game* yang selain melibatkan elemen x dan y juga melibatkan elemen z pada perhitungannya sehingga konsep kamera pada *game* 3D benar-benar menyerupai konsep kamera pada kehidupan nyata [9].

2.1.2 Game 2 Dimensi

Game 2D (2 dimensi) adalah sebuah game berbentuk dari benda yang hanya memiliki panjang dan lebar. Grafik 2 Dimensi merupakan teknik penggambaran yang berpatokan pada titik koordinat sumbu x (datar) dan sumbu y (tegak). Agar dapat tampil dengan sempurna, gambar yang akan ditampilkan dengan teknik ini harus memiliki nilai koordinat x dan y minimum 0 dan maksimum sebesar resolusi yang digunakan.

2.1.3 Simulasi Game

Simulasi *game* adalah *game* yang mengendalikan suatu objek yang bergerak/tidak bergerak baik secara langsung maupun tidak langsung dengan tujuan melakukan hal-hal tertentu. *Game* ini sangat menarik karena bisa membuat pemain berfantasi seakan objek yang berada dalam *game* tersebut adalah nyata dan milik pemain [4]. Beberapa jenis *game simulation* yaitu :

1. *Assets Simulation*

   Game simulasi yang menuntut kita untuk membuat / membangun / mengelola suatu objek / asset / bangunan / instalasi yang dikendalikan secara personal untuk mencapai tujuan tertentu. Contoh *game* : *game* SIM / Tycoon series

2. *Action Simulation*

   Game simulasi plus action yang menuntut kita untuk mengendalikan suatu objek tertentu secara langsung untuk melakukan suatu hal secara aktif, misalnya mengendalikan kendaraan perang untuk bertempur. Contoh *Game* : *Game* Pesawat Tempur, dan sebagainya.

3. *Vehicle Simulation*

   Game simulasi realitas yang tata cara permainan mengambil dari objek aslinya, dengan tujuan mengajari player bagaimana cara mengoprasikan suatu kendaraan yang tidak biasa dikendarai orang awam dan rumit untuk dioperasikan

   Contoh *Game* : Seri Flight Simulator, Rail Simulator & Ship Simulator

4. *Music simulation*

Game simulasi yang menuntut kita mengendalikan suatu objek secara tidak langsung untuk melakukan hal-hal yang berhubungan dengan musik. Contoh *game* : Seri Guitar Hero, seri Dance-Dance Revolution

5. *Lovesims/visual novel*

Game ini hanya berupa suatu percakapan antar tokoh disertai dengan gambar CG (*Computer Graphic*) Anime yang indah. Jalan ceritanya berjalan sendiri tanpa bisa kita kendalikan, kita hanya bisa mengatur jalan cerita melalui pilihan langkah sang tokoh di *game* tersebut untuk mencapai akhir yang buruk atau bahagia. *Game* ini hanya dibuat di negara asal anime yaitu jepang dan lumayan populer di sana namun tidak begitu di negara lainnya. *Game* ini kurang mendapat perhatian karena gameplaynya yang dinilai kurang berbobot, selain itu game tipe ini yang beredar di Indonesia kebanyakan *game* adult yang tidak baik untuk *gamer* usia dibawah umur.

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 [2] :

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 dimainkan secara *online* melalui LAN atau *internet*.

2.1.5 Unsur Game

Dalam sebuah *game* terdapat unsur-unsur yang melengkapinya. Berikut beberapa unsur dalam suatu *game* yaitu [3] :

1. Warna

   Warna mempunyai kemampuan untuk membuat orang tanggap terhadap semua yang dilihat karena tidak ada sesuatu hal bermakna tanpa warna. Mata manusia tertarik oleh warna pada suatu *level* karena warna dari objek diterima sebelum detail-detail dipisahkan oleh bentuk-bentuk dan garisnya. Warna merah memiliki panjang gelombang yang terpanjang, biru memiliki panjang gelombang yang pendek sedangkan hijau memiliki panjang gelombang menengah. Pada anak-anak cenderung tertarik pada warna-warna yang cerah dan mencolok. Warna-warna yang cerah terutama warna *primer* (merah, kuning, biru) dan warna *sekunder* (orange, ungu, hijau). Contoh warna *sekunder* dan *primer* dapat dilihat pada gambar 2.1.



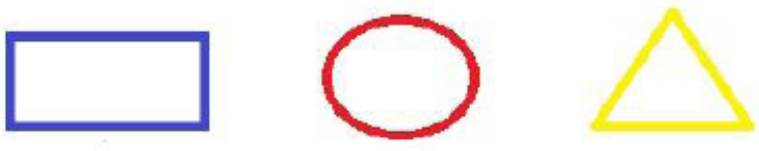
Gambar 2.1 Warna Primer Dan Sekunder

2. Komposisi

Komposisi adalah pengaturan segala elemen didalam sebuah karya desain yang sedemikian rupa dengan tujuan tertentu. Komposisi yang baik adalah komposisi yang mampu memenuhi kebutuhan dan tujuan desain, mudah dipahami dan membentuk kesatuan yang serasi dan harmonis. Kemudian *layout* yaitu perencanaan, penempatan semua unsur mulai dari tulisan, gambar, ilustrasi, teks, nama dan sebagainya dengan pengkuran secara seksama. Komposisi yang sesuai dengan anak-anak adalah komposisi yang sederhana dan tidak menggunakan petunjuk terlalu rumit. Kesederhanaan diwujudkan dengan penggunaan visual 2D dan penerapan warna-warna dalam seluruh aspek desain.

3. Bentuk dasar

Bentuk dasar adalah bentuk-bentuk yang mudah ditemui dalam kehidupan sehari-hari seperti kotak, lingkaran, segitiga dan lain sebagainya, seperti yang dapat dilihat pada gambar 2.2.



Gambar 2.2 Bentuk Dasar

4. Tipografi

   Tipografi merupakan representasi visual dari sebuah bentuk komunikasi *verbal* dan merupakan *property visual* yang pokok dan efektif. Huruf memainkan peranan penting dalam keberhasilan suatu bentuk komunikasi grafis. Dalam media pembelajaran untuk anak-anak, sebuah huruf harus *legible* yaitu jelas dan memiliki tingkat kemudahan untuk dibaca.

5. *Audio*

   Audio adalah sinyal elektrik yang digunakan untuk membawa suara dalam batas pendengaran manusia. *Audio* merupakan komponen sistem yang sudah termasuk didalamnya atau dapat ditambahkan pada komputer.

2.1.6 NPC

Penjabaran NPC atau *Non-Playable Character* adalah karakter *game* yang dikendalikan oleh sistem bukan oleh pemain. NPC melayani sejumlah tujuan di dalam *game,* termasuk:

1. Sebagai perangkat alur : NPC dapat digunakan untuk memajukan alur cerita.
2. Untuk bantuan atau musuh : NPC dapat bertindak sebagai mitra untuk pemain atau dapat menjadi musuh dalam *game*.
3. Fungsi *game* : NPC sering melayani sebagai menyimpan poin, toko item, poin kesehatan regenerasi dan sebagainya.

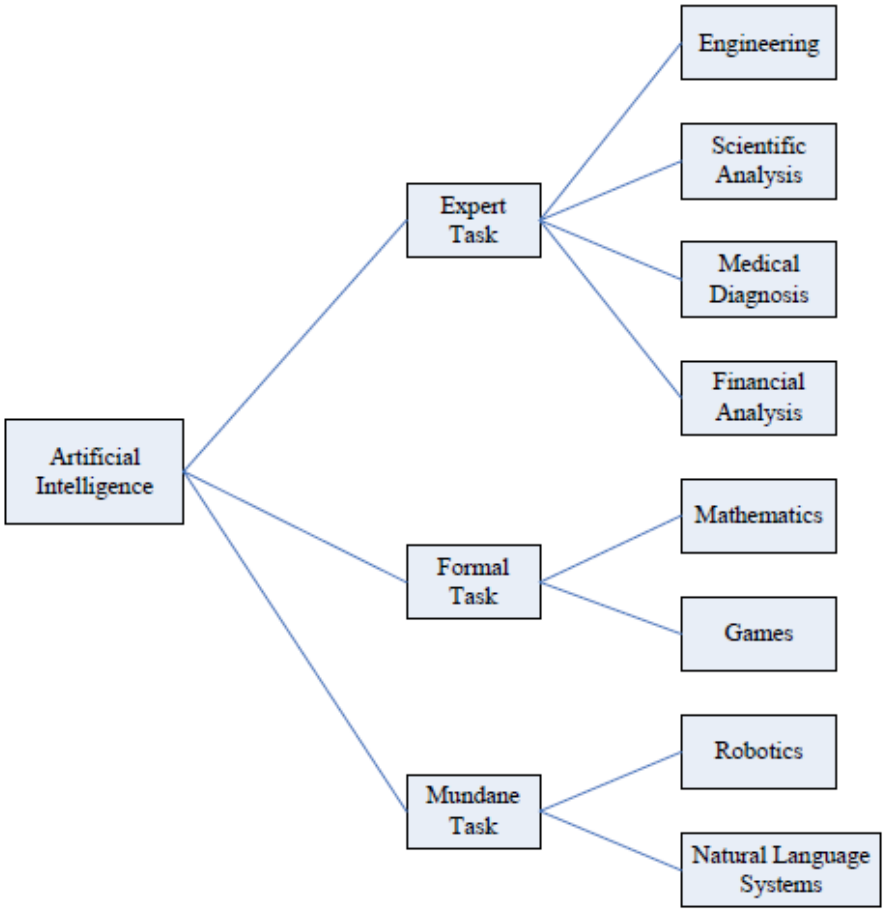
2.2 Kecerdasan Buatan

Kecerdasan buatan atau *Artificial Intelligence* (AI) merupakan cabang dari ilmu komputer yang berhubungan dengan pengautomatisan tingkah laku cerdas. Kecerdasan buatan didasarkan pada teori suara (*sound theoretical*) dan prinsip-prinsip aplikasi dari bidangnya. Prinsip-prinsip ini meliputi struktur data yang digunakan dalam representasi pengetahuan, algoritama yang diperlukan untuk mengaplikasikan pengetahuan tersebut serta bahasa dan teknik pemrograman yang

digunakan dalam mengimplementasikannya [9]. Berdasarkan sudut pandang, AI dapat dipandang sebagai berikut :

1. Sudut pandang kecerdasan, AI adalah bagaimana membuat mesin yang cerdas dan dapat melakukan hal-hal yang sebelumnya hanya dapat dilakukan manusia.
2. Sudut pandang bisnis, AI adalah sekelompok alat bantu yang berdayaguna dan metodologi yang menggunakan alat-alat bantu tersebut untuk menyelesaikan masalah-masalah bisnis.
3. Sudut pandang pemrograman, AI meliputi studi tentang pemrograman simbolik, pemecahan masalah dan proses pencarian.
4. Sudut pandang penelitian :
    a. Riset tentang AI dimulai pada awal tahun 1960-an, percobaan pertama adalah membuat program permainan catur, membuktikan teori dan *general problem solving*.
    b. AI adalah nama pada akar dari studi area.

Kecerdasan buatan memiliki sejumlah sub disiplin ilmu yang sering digunakan untuk pendekatan yang esensial bagi penyelesaian suatu masalah dan dengan aplikasi bidang AI yang berbeda. Pada gambar 2.3 dapat dilihat bidang-bidang tugas dari AI.



Gambar 2.3 Bidang-Bidang Tugas (*Taks Domain*) Dari AI

Aplikasi penggunaan AI dapat dibagi ke dalam tiga kelompok yaitu :

1) *Expert Task*

   AI dibentuk berdasarkan pengalaman dan pengetahuan yang dimiliki oleh para ahli. Penggunaan ini dapat membantu para ahli untuk menyampaikan ilmu-ilmu yang dimiliki. Contohnya adalah :

   a. Analisis finansial.
   b. Analisis medikal.
   c. Analisis ilmu pengetahuan.
   d. Rekayasa (desain, pencarian, kegagalan, perencanaan, manufaktur).

2) *Formal Task*

AI digunakan untuk melakukan tugas-tugas formal yang selama ini manusia biasa lakukan dengan lebih baik. Contohnya adalah :

a. *Game.*
b. Matematika (geometri, logika, kalkulus, integral).

3) *Mundane Task*

Secara harfiah *mundane* adalah keduniaan. AI digunakan untuk melalukan hal-hal yang sifatnya duniawi atau melakukan kegiatan yang dapat membantu manusia. Contohnya adalah :

1. Persepsi.
2. Bahasa alami.
3. Robot control.

Aplikasi kecerdasan buatan memiliki dua bagian utama yaitu :

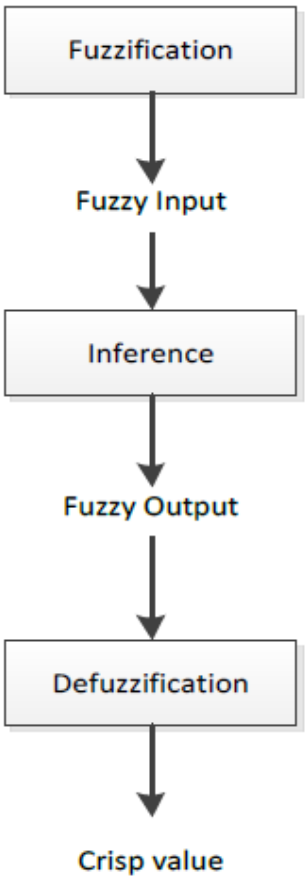
a. Basis Pengetahuan (*Knowledge Base*): Berisi fakta-fakta, teori, pemikiran dan hubungan antara satu dengan lainnya.
b. Motor Inferensi (*Inference Engine*): Kemampuan menarik kesimpulan berdasarikan pengalaman.

2.3 Logika Fuzzy

Konsep-konsep tentang logika *fuzzy* diperkenalkan oleh Prof. Lotfi Astor Zadeh pada 1962. Logika fuzzy adalah metodologi sistem kontrol pemecahan masalah, 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. Artinya, bisa saja suatu keadaan mempunyai nilai “Ya dan Tidak”, “Benar dan Salah”, “Baik dan Buruk” secara bersamaan, namun besar nilainya tergantung pada bobot keanggotaan yang dimilikinya. Dalam banyak hal, logika fuzzy digunakan sebagai suatu cara untuk memetakan permasalahan dari *input* menuju ke output yang diharapkan. Konsep himpunan fuzzy memiliki 2 atribut [7], yaitu:

1. Linguistik, yaitu nama suatu kelompok yang mewakili suatu keadaan tertentu dengan menggunakan bahasa alami, misalnya dingin, sejuk, panas mewakili variabel 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 2.4



Gambar 2.4 Struktur Sistem *Fuzzy*

Keterangan:

1. *Fuzzification*

   Masukan-masukan yang nilai kebenaranya bersifat pasti (crispt input) dikonversi ke bentuk *fuzzy input*, yang berupa nilai linguistik yang semantiknya ditentukan berdasarkan fungsi keanggotaan.

2. *Inference*

   Memperhitungkan semua aturan yang ada dalam basis pengetahuan. Hasil dari proses *inference* dipresentasikan oleh suatu fuzzy set untuk setiap variabel bebas. Derajat keanggotaan untuk setiap nilai variabel tidak bebas menyatakan ukuran kompatibilitas terhadap variabel bebas.

3. *DeFuzzyfication*

   merupakan proses mengubah output *fuzzy* yang diperoleh dari mesin inferensi menjadi nilai tegas menggunakan fungsi keanggotaan yang sesuai dengan saat dilakukan *fuzzyfikasi*.

2.3.1 Himpunan Fuzzy

Logika fuzzy dikembangkan dari teori himpunan fuzzy. Himpunan klasik yang sudah dipelajari selama ini disebut juga dengan himpunan tegas (*crisp set*). Di dalam himpunan tegas, keanggotaan suatu unsur di dalam himpunan dinyatakan secara tegas, apakah objek tersebut anggota himpunan atau bukan.

Pada himpunan tegas (*crisp*), nilai keanggotaan suatu *item* x dalam suatu himpunan A, yang sering ditulis dengan μA[x], memiliki dua kemungkinan, yaitu:

1. Satu (1), yang berarti bahwa suatu *item* menjadi anggota dalam suatu himpunan.
2. Nol (0), yang berarti bahwa suatu *item* tidak menjadi anggota dalam suatu himpunan.

Kalau pada himpunan tegas (*crisp*), nilai keanggotaan hanya ada dua kemungkinan yaitu 0 atau 1, maka pada himpunan *fuzzy* nilai keanggotaan terletak pada rentang 0 sampai 1. Apabila x memiliki nilai keanggotaan *fuzzy* μA[x]=0 berarti x tidak menjadi anggota himpunan A, demikian pula apabila x memiliki nilai keanggotaan *fuzzy* μA[x]=1 berarti x menjadi anggota penuh pada himpunan A. Hal yang perlu diketahui dalam memahami sistem *fuzzy* adalah:

a. Variabel *fuzzy*

   Variabel *fuzzy* merupakan variabel yang hendak dibahas dalam suatu sistem *fuzzy*, contoh : umur, temperatur dan lain-lain.

b. Himpunan *fuzzy*

Himpunan *fuzzy* merupakan suatu grup yang mewakili suatu kondisi atau keadaan tertentu dalam suatu variabel *fuzzy*.

c. Semesta pembicaraan

Semesta pembicaraan adalah keseluruhan nilai yang diperbolehkan untuk dioperasikan dalam suatu variabel *fuzzy*. Semesta pembicaraan merupakan himpunan bilangan *real* yang senantiasa naik (bertambah) secara monoton dari kiri ke kanan. Nilai semesta pembicaraan dapat berupa bilangan *positif* maupun *negatif*. Pada suatu kondisi tertentu nilai semesta pembicaraan ini tidak dibatasi batas atasnya.

d. Domain

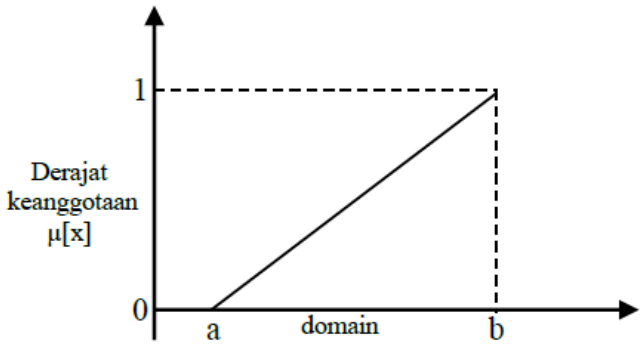
Domain himpunan *fuzzy* adalah keseluruhan nilai yang diijinkan dalam semesta pembicaraan dan boleh dioperasikan dalam suatu himpunan *fuzzy*. Seperti halnya semesta pembicaraan, domain merupakan himpunan bilangan *real* yang senantiasa naik (bertambah) secara monoton dari kiri ke kanan. Nilai domain dapat berupa bilangan *positif* maupun *negatif*.

2.3.2. Fungsi Keanggotaan

Basis data Fungsi keanggotaan *(membership function)* adalah suatu kurva yang menunjukkan pemetaan titik-titik input data ke dalam nilai keanggotaanya (sering juga disebut dengan derajat keanggotaan) yang memiliki interval antara 0 sampai 1. Salah satu cara yang dapat digunakan untuk mendapatkan nilai keanggotaan adalah dengan melalui pendekatan fungsi. Ada beberapa fungsi yang bisa digunakan.

1. Representasi Linear

Pada representasi linear, pemetaan input ke derajat keanggotaannya digambarkan sebagai suatu garis lurus. Bentuk kurva ini paling sederhana dan menjadi pilihan yang baik untuk mendekati suatu konsep yang kurang jelas. Ada 2 keadaan himpunan fuzzy yang linear. Pertama, kenaikan himpunan dimulai pada nilai domain yang memiliki derajat keanggotaan nol (0) bergerak ke kanan menuju ke nilai domain yang memiliki derajat keanggotaan lebih tinggi (Gambar 2.5).

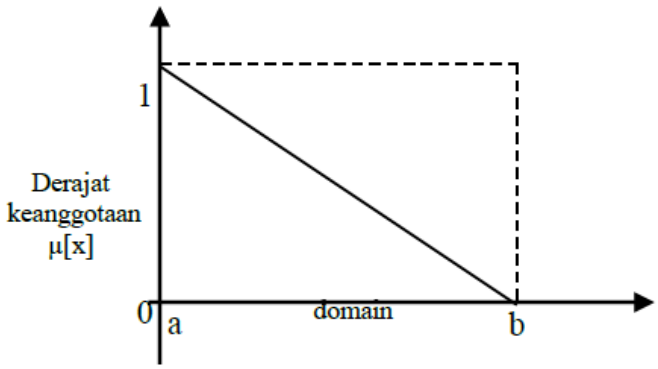


Gambar 2.5 Representasi Linear Naik

Fungsi keanggotaan :

$$\mu[x] = \begin{cases} 0; & x \leq a \\ (x-a)/(b-a); & a \leq x \leq b \\ (1; & x > b \end{cases} \quad (2.1)$$

Kedua, merupakan kebalikan dari yang pertama. Garis lurus dimulai dari nilai domain dengan derajat keanggotaan tertinggi pada sisi kiri, kemudian bergerak menurun ke nilai domain yang memiliki derajat keanggotaan lebih rendah (Gambar 2.6).



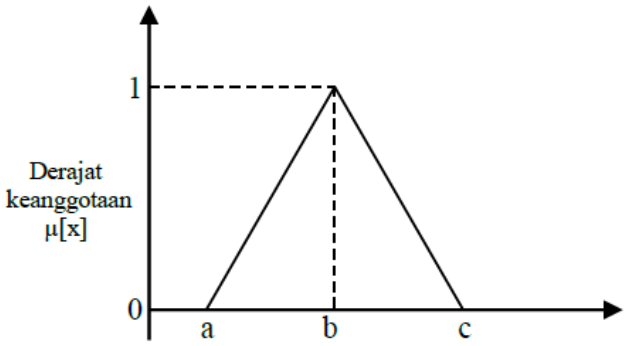
Gambar 2.6 Representasi Linear Turun

Fungsi Keanggotaan :

$$\mu[x] = \begin{cases} (b-x)/(b-a); & a \leq x < b \\ 0; & x \geq b \end{cases} \quad (2.2)$$

2. Representasi Kurva Segitiga

Kurva segitiga pada dasarnya merupakan gabungan antara 2 garis (*linear*) seperti terlihat pada Gambar 2.7.



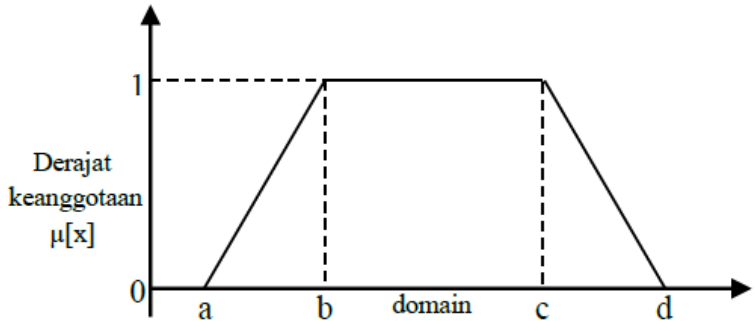
Gambar 2.7 Kurva Segitiga

Fungsi keanggotaan :

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

3. Representasi Kurva Trapesium

Kurva trapesium pada dasarnya seperti bentuk segitiga, hanya saja ada beberapa titik yang memiliki nilai keanggotaan 1 seperti terlihat pada gambar 2.8.



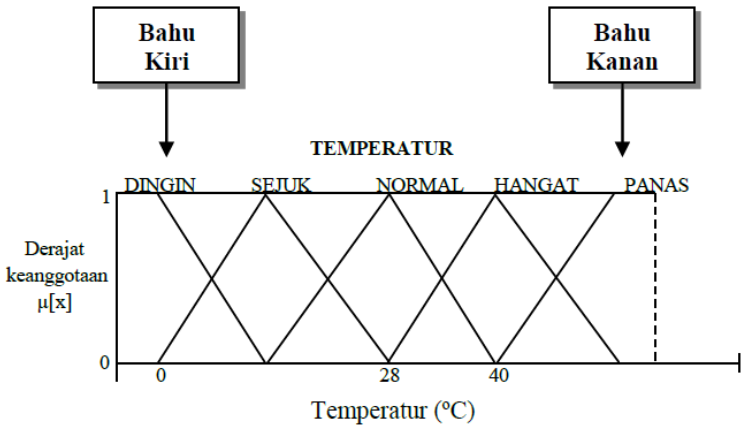
Gambar 2.8 Kurva Trapesium

Fungsi Keanggotaan :

$$\mu[x] = \begin{cases} 0; & x \leq a \; atau \; x \geq d \\ (x-a)/(b-a); & a \leq x \leq b \\ 1; & b \leq x \leq c \\ (d-x)/(d-c); & c \leq x \leq d \end{cases} \quad (2.4)$$

4. Representasi Kurva Bentuk Bahu

Daerah yang terletak di tengah-tengah suatu variabel yang direpresentasikan dalam bentuk segitiga, pada sisi kanan dan kirinya akan naik dan turun (misalkan: DINGIN bergerak ke SEJUK bergerak ke HANGAT dan bergerak ke PANAS). Tetapi terkadang salah satu sisi dari variabel tersebut tidak mengalami perubahan. Sebagai contoh, apabila telah mencapai kondisi PANAS, kenaikan temperatur akan tetap berada pada kondisi PANAS. Himpunan *fuzzy* „bahu", bukan segitiga, digunakan untuk mengakhiri variabel suatu daerah *fuzzy*. Bahu kiri bergerak dari benar ke salah, demikian juga bahu kanan bergerak dari salah ke benar. Gambar 2.9 dibawah menunjukkan variabel TEMPERATUR dengan daerah bahunya.



Gambar 2.9 Daerah Bahu Pada Daerah Temperatur

2.4 OOP (Object Oriented Programming)

Pemodelan OOP adalah teknik memodelkan suatu sistem dunia nyata dalam perangkat lunak berdasarkan objek. Objek tersebut adalah konsep inti. Suatu objek adalah sebuah perangkat lunak entitas atau konsep model dunia nyata.

Ketika sebuah program berjalan, objek individu biasanya tidak dapat berdiri sendiri. Mereka termasuk ke sebuah koleksi dari objek lain yang serupa yang merupakan anggota dari grup yang sama, atau *class*. Sebuah program akan terbentuk dari banyak *class* yang berbeda, setiap *class* terbentuk dari objek-objek yang serupa.

Beberapa sistem perangkat lunak orientasi objek akan memiliki sifat berikut:

1. Abtraksi dengan objek.

   Abtraksi adalah mekanisme yang memungkinkan kompleks, situasi dunia nyata dapat diwakili menggunakan model yang disederhanakan. Orientasi objek abstrak dunia nyata didasarkan pada objek dan interkasi antar objek lainnya.

2. Enkapsulasi *class*.

   Enkapsulasi adalah proses menyembunyikan semua bagian rinci sebuah objek dari dunia luar.

3. Interaksi lewat pesan.

   Untuk memenuhi suatu perintah, objek butuh berinteraksi dengan objek lain. Interaksi dapat antara objek di *class* yang sama, atau objek di *class* lain. Interaksi ini ditangani dengan cara mengirimkan pesan (di Java, ini dilakukan dengan memanggil *methods*) ke objek lain untuk melewati informasi atau meminta aksi.

4. Masa hidup objek.

   Semua objek memiliki masa hidup. Mereka dibuat dan diinisialisasi sebagaimana mereka dibutuhkan pada saat program dijalankan, hidup dan membawa keluar fungsi mereka, dan akhirnya dihancurkan. Sementara mereka ada, mereka mempertahankan identitas mereka sendiri dan kondisi. Banyak objek yaitu contoh dari *class* yang sama bisa hidup pada waktu tertentu. Setiap objek memiliki atribut yang berbeda dari yang lain yaitu contoh objek dari *class* yang sama.

5. Hirarki *class*.

   Dalam desain orientasi objek, *class* objek tersebut diatur ke dalam hirarki yang memodelkan dan menggambarkan hubungan antar *class*. Hubungan yang paling sederhana adalah sebuah asosiasi.

6. Polimorfisme

   Polimorfisme adalah karakteristik sistem orientasi objek. Ketika warisan digunakan untuk memperpanjang *class* umum ke *class* yang lebih khusus, biasanya akan mencakup memperluas beberapa perilaku dari *class* umum. *Class* khusus sering akan menerapkan perilaku yang agak berbeda dengan *class* umum, tetapi nama yang digunakan mendefinisikan perilaku akan menjadi sama. Adalah penting bahwa contoh yang diberikan dari sebuah objek menggunakan perilaku yang benar, dan *property* polimorfisme memungkinkan ini terjadi secara otomatis dan mulus. Polimorfisme sebenarnya lebih mudah digunakan daripada dijelaskan [5].

2.5 UML (Unified Modeling Language)

UML adalah penerus gelombang berorientasi objek alnalisis dan desain (OOA & D) metode yang muncul di akhir ‘80-an dan ’90-an awal. Hal yang paling langsung menyatukan metode Booch, Rumbaugh (OMT), dan Jacobson, tapi jangkauan lebih luas dari itu. UML pergi melalui proses standarisasi dengan OMG (*Obyek Manajement Group*) dan sekarang menjadi standar OMG.

UML disebut bahasa pemodelan, bukan metode. Sebagian besar metode terdiri, setidaknya pada prinsipnya, kedua model sebuah bahasa dan proses. Bahasa pemodelan adalah notasi (terutama grafis) bahwa metode digunakan untuk mengekspresikan desain. Proses ini menyarankan mereka apa langkah yang harus diambil dalam melakukan desain.

Bagian proses dalam buku banyak metode yang agak samar. Selain itu, kebanyakan orang, ketika mereka mengatakan bahwa mereka menggunakan metode, menggunakan bahasa pemodelan, tapi jarang mengikuti proses. Jadi dalam banyak hal pemodelan bahasa adalah bagian paling penting dari meode ini. Hal ini tentu bagian penting untuk komunikasi. Jika anda ingin mendiskusikan

desain anda dengan seseorang, itu adalah bahasa pemodelan yang anda berdua perlu pahami, bukan proses yang digunakan untuk sampai ke desain tersebut [5].

2.5.1 Diagram UML

Menggunakan berbagai macam diagram dengan fungsi masing-masing untuk menggambarkan setiap proses dari sistem berorientasi objek. Berikut merupakan beberapa diagram UML diantaranya [6]:

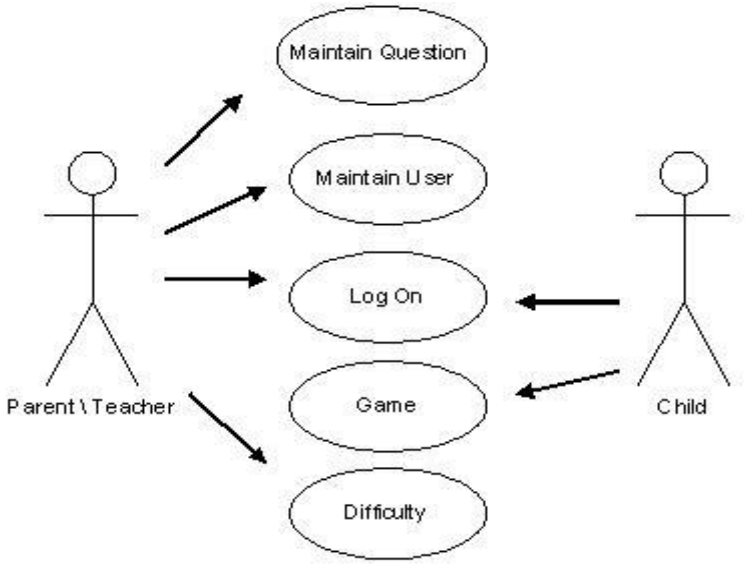
A. *Use Case Diagram*

Use Case atau diagram *use case* merupakan pemodelan yang digunakan untuk menggambarkan kelakuan (*behavior)* dari sistem yang akan dibuat [6]. *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* [6].

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 pesar antarunit atau aktor.

Contoh dari *use case diagram* dapat dilihat pada Gambar 2.10.



Gambar 2.10 Contoh Dari *Use Case Diagram*

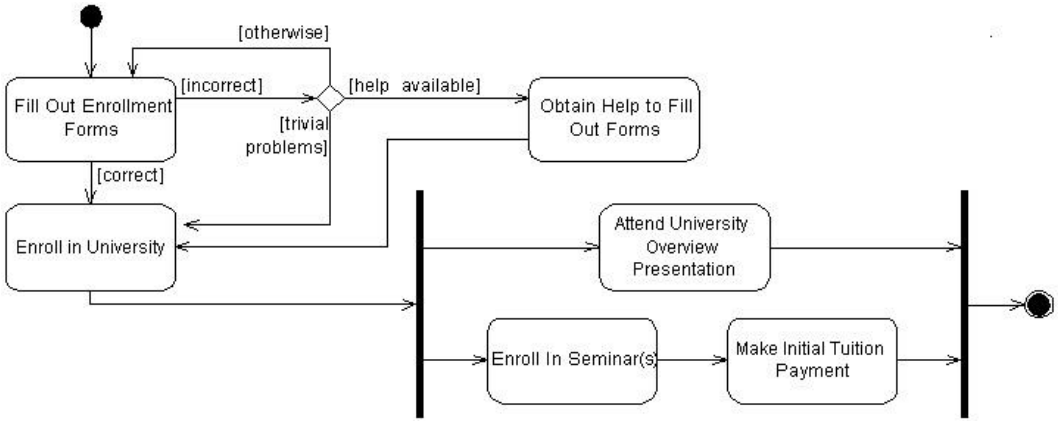
B. *Activity Diagram*

Diagram aktivitas atau *activity* diagram adalah sebuah diagram yang menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis [6]. 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 [6]:

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 dianggap memiliki sebuah rancangan antarmuka tampilan.
3. Rancangan pengunjian di mana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.

Contoh dari *activity diagram* dapat dilihat pada gambar 2.11.



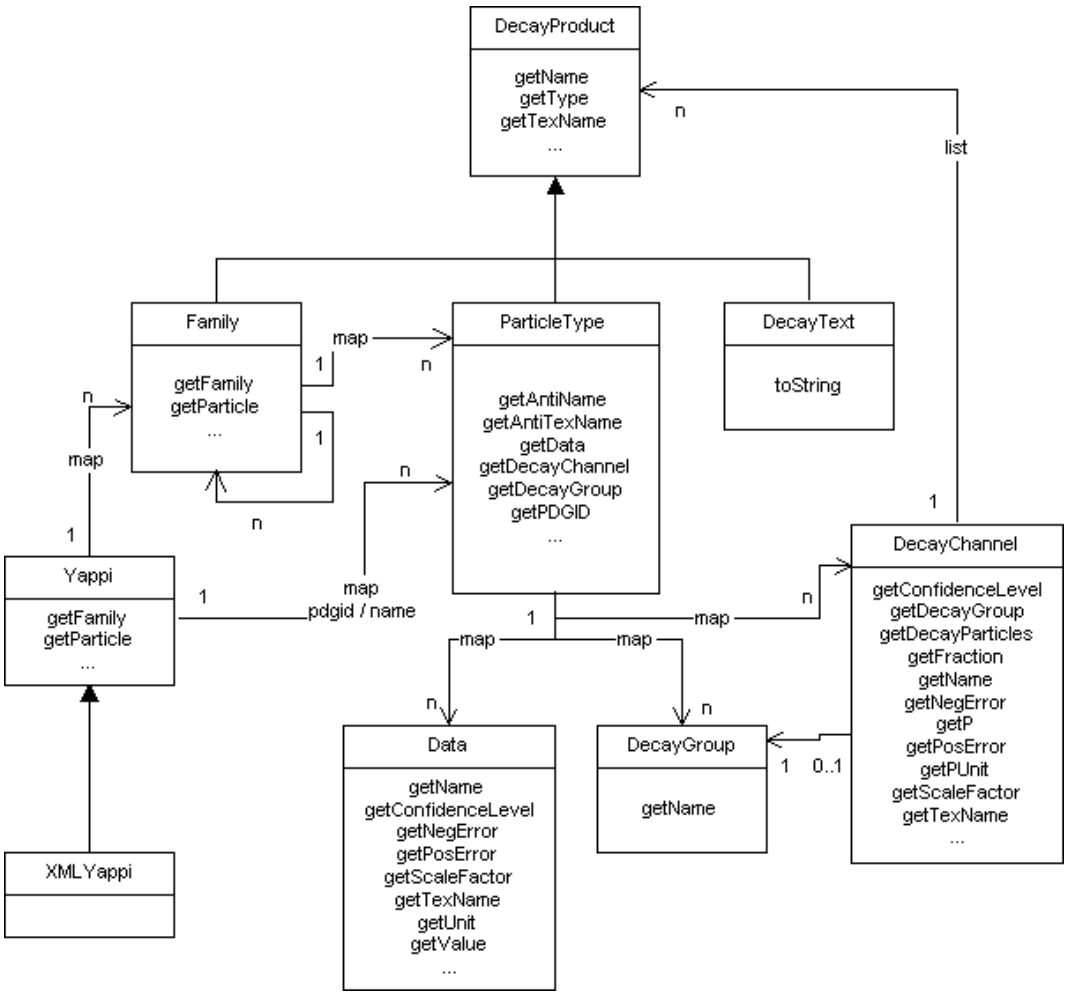
Gambar 2.11 Contoh Dari *Activity Diagram*

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 [6].

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 2.12.



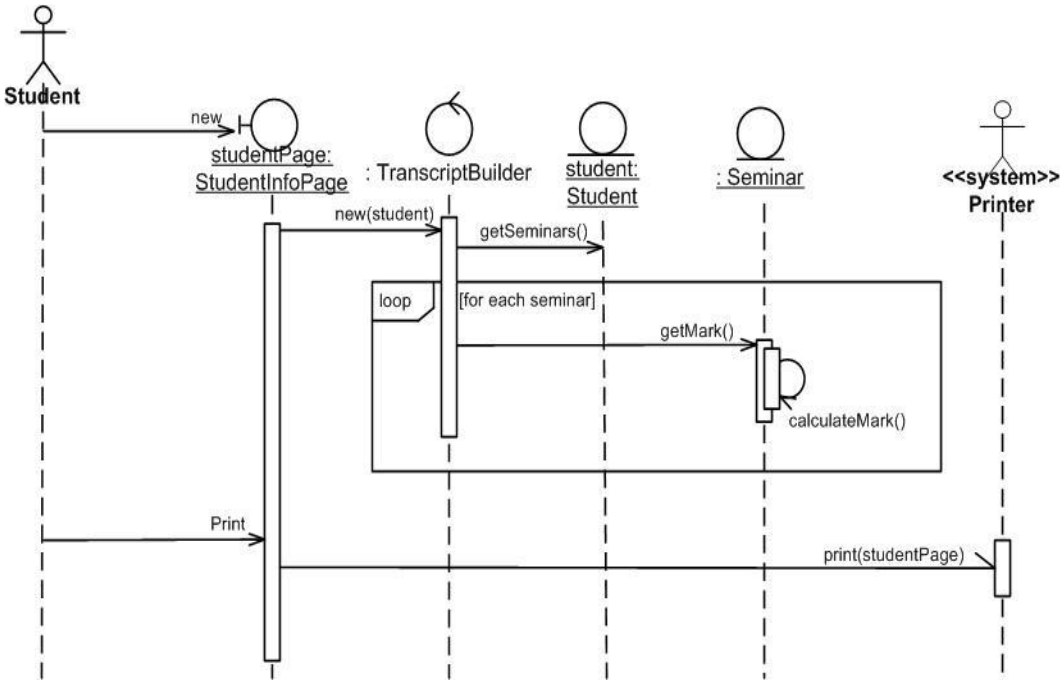
Gambar 2.12 Contoh Dari *Class Diagram*

D. *Sequence Diagram*

Diagram sekuen atau *sequence diagram* adalah diagram yang menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antarobjek [6]. 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 diinstansiasi menjadi objek itu.

Banyaknya diagram sekuen yang harus digambarkan adalah sebanyak pendefinisian *use case* yang memiliki prose situ 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 2.13.



Gambar 2.13 Contoh Dari *Sequence Diagram*

2.6 Pengujian Perangkat Lunak

Pengujian perangkat lunak adalah suatu proses yang digunakan untuk mengidentifikasi ketepatan, kelengkapan dan mutu dari perangkat lunak. Pada dasarnya, pengujian tidak pernah dapat menetapkan kebenaran mutlak dari perangkat lunak. Pengujian perangkat lunak adalah elemen kritis dari jaminan kualitas perangkat lunak dan merepresentasikan kajian pokok dari spesifikasi, desain dan pengkodean [1].

Pengembang perangkat lunak sesuai dengan sifatnya dasar, mereka adalah manusia pembangun. Pengujian mengharuskan pengembang membuang pemikiranpemikiran sebelumnya mengenai “kebenaran” perangkat lunak yang baru saja dikembangkan dan mengatasi konflik minat yang terjadi pada saat kesalahan ditemukan.

Berikut ini merupakan prinsip pengujian perangkat lunak :

1. Semua pengujian harus dapat ditelusuri hingga ke persyaratan pelanggan.
   Sebagai tujuan utama pengujian perangkat lunak yaitu untuk mengungkap kesalahan. Yang mana berarti kesalahan paling fatal apabila perangkat lunak tidak dapat memenuhi syarat yang ditentukan oleh pelanggan.
2. Pengujian harus direncanakan lama sebelum pengujian itu dimulai.
   Perencanaan pengujian dapat dimulai setelah model persyaratan telah dilengkapi. Definisi detail tentang *test case* dapat dimulai segera setelah model desain ditetapkan. Dengan demikian pengujian dapat direncanakn dan dirancang sebelum pengkodean dilakukan.
3. Prinsip Pareto berlaku untuk pengujian perangkat lunak.
   Prinsip Pareto mengimplikasikan bahwa 80% dari semua kesalahan yang ditemukan selama pengujian, hanya dapat ditelusuri 20% dari semua modul program. Hanya saja kita sulit untuk mengetahui modul yang mengalami kesalahan dan mengujinya dengan teliti
4. Pengujian harus mulai “dari yang terkecil” dan berkembang ke pengengujian “yang besar”.
   Pengujian biasanya dilakukan terhadap modul program individual. Selagi pengujian berlangsung, maka seluruh modul yang terintegrasi lebih mudah diuji.
5. Pengujian yang mendalam tidak mungkin.
   Jumlah jalur permutasi pada perangkat lunak sangat besar, oleh karena itu sulit untuk melakukan pengujian terhadap semua jalur skema pengujian. Akan tetapi setidaknya kita dapat mengetahui bahwa logika yang tertuang dalam perancangan perangkat lunak itu telah tepat dan memastikan semua kondisi telah teruji.
6. Untuk menjadi paling efektif, pengujian harus dilakukan oleh pihak ketiga yang independen.
   Arti dari “paling efektif” adalah pengujian yang memiliki peluang tertinggi untuk menemukan kesalahan. Perekayasa perangkat lunak yang

membuat system bukanlah orang yang paling tepat untuk melakukan semua pengujian bagi perangkat lunak.

2.7 Pengujian Performansi

Performance Testing atau pengujian performansi merupakan proses menentukan kecepatan atau efektivitas, jaringan program computer, perangkat lunak atau perangkat. Proses ini dapat melibatkan test kuantitatif yang dilakukan di laboratorium, seperti mengukur waktu respon atau jumlah MIPS (*Millions of Intructions per Second*) pada fungsi sistem. Atribut kualitatif seperti kehandalan, skalabilitas dan interoperabilitas juga dapat dievaluasi. Pengujian performansi sering dilakukan dalam hubungannya dengan stress *testing* [11].

Pengujian performansi dapat memverifikasi bahwa sistem memenuhi spesifikasi diklaim oleh produsen atau vendor. Proses ini dapat membandingkan dua atau lebih perangkat atau program dalam hal parameter seperti kecepatan, kecepatan transfer data, *bandwidth, throughput*, efisiensi atau keandalan.

Pengujian performansi merupakan evaluasi *users* dengan sistem yang ketat dibawah kondisi yang sangat realistis, untuk mengidentifikasi permasalahan *user* dan membandingkan langkah-langkah seperti tingkat keberhasilan, waktu penugasan dan kepuasan *user* dengan persyaratan.

2.8 Unity

Unity 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 *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* [8]. *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* dan lain-lain.
Features (Scripting) di dalam *unity* adalah sebagai berikut:

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

.

2.9 Adobe Photoshop

Adobe Photoshop, atau biasa disebut *Photoshop* adalah perangkat lunak editor citra buatan *Adobe System* yang dikhususkan untuk pengeditan foto/gambar dan pembuatan efek. Perangkat lunak ini banyak digunakan oleh fotografer digital dan perusahaan iklan sehingga dianggap sebagai pemimpin pasar (*marketleader*) untuk perangkat lunak pengolah gambar/foto, dan bersama *Adobe Acrobat* dianggap sebagai produk terbaik yang pernah diproduksi oleh *Adobe System*. Versi kedelapan aplikasi ini disebut dengan nama Photoshop CS (*Creative Suite*), versi kesembilan disebut *Adobe Photoshop* CS2, versi kesepuluh disebut *Adobe Photoshop* CS3, versi kesebelas disebut *Adobe Photoshop* CS4, versi keduabelas disebut *Adobe Photoshop* CS5, dan versi yang terakhir (ketigabelas) adalah *Adobe Photoshop* CS6 [10].

Photoshop tersedia untuk *Microsoft Windows*, Mac OS X, dan Mac OS, versi 9 ke atas juga dapat digunakan oleh sistem operasi lain seperti Linux dengan bantuan perangkat lunak tertentu seperti *CrossOver*.

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.