BAB 2

LANDASAN TEORI

2.1 Permainan (*Game*)

Game diambil dari bahasa Inggris yang diterjemahkan yang artinya permainan. Di dalam era perkembangan era perkembangan teknologi yang pesat ini, seperti halnya permainan kartu, catur dan lain-lainnya dapat ditemui melalui dunia *virtual* atau yang biasa kita mainkan di dalam komputer. Dengan perkembangan teknologi sekarang ini munculah berbagai sarana permainan, misalnya Playstation 3, Xbox 360, Nintendo Wii, PSP, Nintendo DS, maupun PC dari yang berbasis individu ataupun *multiplayer*.

Pengertian *Game* adalah *Game* di artikan sebagai suatu aktivitas tersetruktur atau juga digunakan sebagai alat pembelajaran. Sebuah *Game* bisa dikarakteristikan dari apa pemain lakukan misalnya : Peralatan Misal : bola, kartu, papan, atau sebuah Komputer. Peraturan digunakan untuk menentukan giliran pemain, hak dan keharusan masing-masing pemain, dan tujuan permainan.
Skill, Strategi dan Keberuntungan *Game* dengan dengan skill, contohnya dengan kekuatan fisik, misal gulat, menembak dan kekuatan mental seperti catur.

Single Player *Game* (pemain satu orang) dan Double Player (lebih dari satu pemain) Jika pemain tunggal, pemain harus bermain dengan keahlian, berpacu dengan waktu dan keberuntungan sedangkan pemain double, pemain diharuskan untuk menggunakan suatu strategi dan kekompakan sesama pemain, untuk mencapai tujuan tertentu atau sebaliknya pemain harus berlomba dengan pemain lainnnya untuk mencapai sesuatu tujuan [2].

Sejarah Perkembangan *Game*

Computer gaming telah ada sejak sekitar 1960an. Karena perkembangan industri microprocessor, mainframe, dan minicomputers. Salah satu *Game* komputer pertama dikembangkan pada tahun 1961. Ketika siswa MIT, Martin

Graetz dan Alan Kotok, dengan memperkerjakan Steve Russell, menghasilkan "Spacewar!" pada komputer mainframe PDP-1.

Generasi pertama PC *Game*s biasanya berupa 'text-adventures' atau 'interactive fiction', yang mana pemain berkomunikasi dengan komputer melalui input pada memakai keyboard. *Game* text-adventure pertama adalah 'Adventure' yang dibangun untuk PDP-11 oleh Will Crowther tahun 1976 dan dikembangkan lagi oleh Don Woods di tahun 1977. Ketika memasuki 1980an, PC telah mendukung penuh untuk menjalankan *Game* sejenis 'Adventure'.Di masa ini, grafik kemudian menjadin unsur penting dalam pembuatan *Game*.

Pada tahun 1952, A.S. Douglas menulis untuk gelar PhD di University of Cambridge jurusan Human-Computer Interraction. Douglas menciptakan permainan komputer grafis yang diberi nama Tic-Tac-Toe. Permainan ini deprogram pada komputer EDSAC vacuum-tube yang memiliki tampilan tabung sinar katode.William Higinborham menciptakan video *Game* pertama pada tahun 1958. Permainan ini diberi judul “Tennis for Two”, diciptakan dan dimainkan di osiloskop Brookhaven National Laboratory.

Video *Game*s pertama kali diperkenalkan pada tahun 1971 sebagai medium hiburan bersifat komersial. Pada akhir tahun 1970an dan awal tahun 1980an, video *Game*s menjadi sebuah dasar yang penting bagi industri hiburan di Amerika Serikat, Jepang, dan Eropa.

Pada 1967, Ralph Baer menciptakan video *Game* pertama yang dimaikan dengan televisi, diberi nama Chase. Ralph Baer akhirnya menjadi bagian dari Sanders Associates, sebuah perusahaan elektronik militer. Idenya Ralph ini dicetuskan untuk pertama kali pada 1951, saat sedang bekerja untuk perusahaan komputer Loral. Pada 1971, Nolan Bushnell bersama dengan Ted Dabney, menciptakan permainan arcade yang pertama. Permainan ini diberi nama 'Computer Space', yang diciptakan berdasarkan pada permainan 'SpaceWar!'. Lalu, tahun 1972, Permainan arcade Pong diciptakan oleh Nolan Bushnell (dengan bantuan Al Alcorn). Nolan Bushnell dan Ted Dabney memulai dengan Atari Computer pada tahun yang sama. Pada tahun 1975, Atari kembali meluncurkan Pong sebagai home video *Game*.Tahun 1972, hiburan pertama video

Game komersial dapat dimainkan di rumah. 'Odyssey', diluncurkan oleh Magnavox dan didesain oleh Ralph Baer. Permainan mesin ini aslinya didesain pada saat Ralph Baer masih berada di Sanders Associates pada tahun 1966, Baer mengurus untuk mendapatkan ijin legalnya setelah ditolak oleh Sanders Associates. Permainan Odyssey deprogram dengan dua belas permainan.Pada tahun 1976, Fairchild meluncurkan programmable home *Game* yang pertama dan diberi nama Fairchild Video Entertainment System atau Channel F. Channel F merupakan salah satu dari electronic system yang pertama menggunakanmicrochip yang diciptakan oelh Robert Noyce untuk Fairchild Semiconductor Corporation.

2.1.1 Klasifikasi *Game*

Berikut ini adalah klasifikasi *Game* berdasrkan pembatas usia, menurutEntertainment Software Rating Board (ESRB), yaitu adalah :



Gambar 2. 1 *Early Childhood* (3+) *Rating*

Early Childhood (3+) memiliki isi yang mungkin cocok untuk usia 3 dan lebih tua. Tidak mengandung materi yang tidak pantas orang tua akan menemukan.



Gambar 2. 2 *Everyone* (6+) *Rating*

Everyone (6+) memiliki konten yang mungkin cocok untuk usia 6 dan lebih tua. Judul-judul dalam kategori ini mungkin berisi sedikit kartun, fantasi atau kekerasan ringan dan / atau jarang menggunakan bahasa ringan.



Gambar 2. 3 *Everyone* (10+) *Rating*

Everyone (10+) memiliki konten yang mungkin cocok untuk usia 10 dan lebih tua. Judul-judul dalam kategori ini mungkin berisi lebih banyak kartun, fantasi atau kekerasan ringan, bahasa ringan dan / atau minimal tema sugestif.



Gambar 2. 4 *Teen* (13+) *Rating*

Teen (13+) memiliki konten yang mungkin cocok untuk usia 13 dan lebih tuaJudul-judul dalam kategori ini mungkin mengandung kekerasan, tema sugestif, humor kasar, sedikit darah, simulasi perjudian, dan / atau jarang menggunakan bahasa yang kuat.



Gambar 2. 5 *Mature* (17+) *Rating*

Mature (17+) memiliki konten yang mungkin cocok untuk orang usia 17 dan lebih tua. Judul-judul dalam kategori ini mungkin berisi intens kekerasan, darah dan darah kental, konten seksual dan / atau bahasa yang kuat.



Gambar 2. 6 *Adults Only* (18+) *Rating*

Adults Only (18+) memiliki konten yang hanya boleh dimainkan oleh orang-orang 18 tahun dan lebih tua. Judul-judul dalam kategori ini mungkin termasuk adegan yang intens lama kekerasan dan / atau grafik konten seksual dan ketelanjangan.



Gambar 2. 7 *RP (Rating Pending)*

RP (Rating Pending) telah disampaikan kepada ESRB dan sedang menunggu penilaian akhir. (Simbol ini hanya muncul dalam iklan sebelum permainan dibebaskan).

The Entertainment Software Rating Board (ESRB) adalah sebuah organisasi regulator mandiri yang menilai permainan video, panduan pengiklanan, prinsip privasi online permainan video dan software hiburan lainnya di Kanada dan Amerika Serikat. ESRB dibentuk pada tahun 1994 oleh Entertainment Software Association (sebelumnya Interactive Digital Software Association). Sejak tahun 2003, ESRB telah menilai 8,000 permainan yang di produksi oleh 350 penerbit permainan video.

2.1.2 Jenis-Jenis *Game*

Jenis – jenis *Game* terbagi menjadi beberapa kategori yaitu :

A. Simulasi

- Simulasi, Konstruksi dan manajemen. *Video Game* jenis ini seringkali menggambarkan dunia di dalamnya sedekat mungkin dengan dunia nyata dan memperhatikan dengan detil berbagai faktor. Dari mencari jodoh dan pekerjaan,

membangun rumah, gedung hingga kota, mengatur pajak dan dana kota hingga keputusan memecat atau menambah karyawan. Dunia kehidupan rumah tangga sampai bisnis membangun konglomerasi, dari jualan limun pinggir jalan hingga membangun laboratorium cloning. *Video Game* jenis ini membuat pemain harus berpikir untuk mendirikan, membangun dan mengatasi masalah dengan menggunakan dana yang terbatas. Contoh: Sim City, The Sims, Tamagotchi.

Konstruksi dan manajemen simulasi (CMS) [1] adalah jenis permainan simulasi di mana pemain membangun, memperluas atau mengelola komunitas fiksi atau proyek dengan sumber daya terbatas [2]. Video *Game* Strategi kadang-kadang menggabungkan aspek CMS ke dalam perekonomian permainan mereka , sebagai pemain harus mengelola sumber daya sementara memperluas proyek mereka . Tapi murni CMS permainan berbeda dari *Game* strategi bahwa "tujuan pemain tidak untuk mengalahkan musuh , tetapi untuk membangun sesuatu dalam konteks dari proses yang berkelanjutan" [1]. *Game* dalam kategori ini kadang-kadang juga disebut "permainan manajemen"[3] [4] [5].

SimCity merupakan contoh awal keberhasilan dalam genre . Permainan lainnya dalam kisaran bergenre dari kota - bangunan *Game* seperti Caesar , murni simulasi permainan bisnis Seperti halnya Hollywood Mogul , dan atau CMSS sejati seperti Theme Park . Kadang-kadang video *Game* strategi seperti Age of Empires menggabungkan aspek CMSS dengan war*Game*s [6].

CMSS sering disebut "*Game* simulasi" singkatnya . Meskipun *Game* dapat mensimulasikan banyak kegiatan dari kendaraan untuk olahraga , pemain biasanya menyimpulkan jenis simulasi dari judul permainan.

a. Tantangan Ekonomi

Ekonomi memainkan peran utama dalam konstruksi dan manajemen simulasi , karena mereka memungkinkan pemain untuk membangun sesuatu sementara operasi dalam kendala ekonomi [6]. Beberapa *Game* dapat menantang pemain untuk mengeksplorasi atau mengenali pola, namun sebagian besar tantangan permainan yang ekonomi dalam mereka fokus pada pertumbuhan [6]. Permainan ini berbasis di lingkungan di mana ekonomi dapat dibangun dan dikelola, biasanya beberapa jenis komunitas, institusi, atau kerajaan [2] peran

pemain jarang sesuai dengan kegiatan nyata, karena. Pemain biasanya lebih terlibat dalam keputusan rinci dari manajer nyata atau administrator [6] pemain biasanya memiliki dua jenis alat yang mereka miliki yaitu alat untuk membangun dan alat untuk mengelola [1]

Mekanisme konstruksi CMSS cenderung menjadi salah satu dari dua jenis, yaitu rencana dan membangun di mana pembangunan selesai secara bertahap, atau pembelian dan tempat dimana konstruksi segera muncul [6]. Bencana acak juga dapat membuat tantangan konstruksi baru [1] dan beberapa permainan memaksakan kendala pada bagaimana hal-hal harus dibangun [6]. Tapi biasanya tindakan konstruksi cukup sederhana [6], dan tantangan utama dari CMS adalah mendapatkan sumber daya yang diperlukan untuk menyelesaikan pembangunan [1]. Pemain harus mengelola sumber daya dalam pertumbuhan ekonomi, di mana sumber daya diproduksi, dikonsumsi, dan ditukar [1]. Sumber daya diambil dari sumber, seperti uang dari bank, atau emas dari tambang. Beberapa CMSS memungkinkan pemain untuk mengubah sumber daya dari satu jenis yang lain, seperti fermentasi gula menjadi rum [1] sumber daya umum termasuk uang, orang, dan bahan bangunan [6]. Sumber Daya digunakan dalam salah satu dari dua cara, yaitu baik konstruksi, di mana pemain membangun atau membeli sesuatu untuk melayani beberapa tujuan, atau pemeliharaan, di mana pemain harus melakukan pembayaran sedang berlangsung untuk mencegah kehilangan atau kerusakan [1]. Kadang-kadang menghancurkan struktur akan biaya sumber daya, tetapi ini sering dilakukan tanpa biaya [6].

b. Tujuan

CMSS biasanya permainan pemain tunggal , karena persaingan akan memaksa pemain untuk menjauhkan diri dari kreativitas dalam mendukung efisiensi, dan ras untuk mengumpulkan sumber daya [6]. Mereka biasanya memiliki modus konstruksi bebas-bentuk di mana pemain dapat membangun seperti yang mereka lihat cocok , yang menarik bagi rasa pemain kreativitas dan keinginan untuk kontrol [6] dengan demikian, banyak CMSS tidak memiliki kondisi kemenangan, meskipun pemain selalu bisa kalah dengan membangkrutkan sendiri sumber [1]. Permainan ini menekankan pertumbuhan,

dan pemain harus berhasil mengelola. ekonomi mereka dalam rangka untuk membangun kreasi yang lebih besar dan mendapatkan kekuatan yang lebih kreatif [6].

Tidak seperti genre lain, konstruksi dan manajemen simulasi jarang menawarkan kemajuan dalam alur cerita, dan desain tingkat adalah ruang sederhana di mana pemain dapat membangun. Beberapa *Game* menawarkan skenario pre-*built*, yang meliputi kondisi kemenangan seperti mencapai tingkat kekayaan, atau bertahan dalam kondisi memburuk. Tetapi keberhasilan dalam satu skenario jarang memiliki dampak pada skenario lain, dan pemain biasanya dapat mencobanya dalam urutan apapun [6].

c. Antarmuka

Karena pemain harus mengelola ekonomi internal yang kompleks, konstruksi dan manajemen simulasi sering menggunakan antarmuka berjendela. [6] Berbeda dengan genre seperti *Game* action, pemain CMS diberikan komputer seperti kontrol seperti menu dan tombol pull-down . Pemain juga dapat memahami ekonomi permainan melalui grafik dan alat analitik lainnya. Ini sering mencakup penasihat yang memperingatkan pemain masalah dan menggambarkan kebutuhan saat ini [6]. Dengan demikian, *Game* CMS memiliki beberapa antarmuka yang paling kompleks dari semua jenis permainan [1]. Permainan ini bisa sangat populer bahkan tanpa grafis terbaru [1].

Pemain dalam CMS biasanya di mana-mana, dan tidak memiliki avatar. Dengan demikian, pemain biasanya diberikan perspektif isometrik dunia, atau kamera bebas roaming dari sudut pandang udara untuk *Game* 3D modern [6]. Dunia *Game* sering mengandung unit dan orang-orang yang merespon tindakan pemain, tetapi. jarang diberikan perintah langsung [6].

d. *City Building Games*

City Building Games adalah sebuah subgenre dari CMS dimana pemain bertindak sebagai perencana kota atau pemimpin. Pemain biasanya melihat kota dari *point-of-view* tinggi di langit, tumbuh dan mengelola sebuah kota simulasi. Pemain hanya diperbolehkan untuk mengontrol penempatan bangunan dan fitur manajemen kota seperti gaji dan prioritas kerja, sedangkan bangunan yang

sebenarnya dilakukan oleh warga permainan yang merupakan NPC(*Non-Playable Character*). Hal ini banyak simulasi konstruksi dan manajemen, meskipun tidak harus menjadi sebuah kota. Ini bisa menjadi sebuah kerajaan (seperti di *Empire GoodGame* atau *Age of Empires*) atau mungkin kebun binatang atau jenis tanah yang melibatkan aspek pembangunan dan manajemen dan mampu membuat simulasi itu.

- *Simulasi kendaraan.* Video *Game* jenis ini memberikan pengalaman atau interaktifitas sedekat mungkin dengan kendaraan yang aslinya, muskipun terkadang kendaraan tersebut masih eksperimen atau bahkan fiktif, tapi ada penekanan khusus pada detil dan pengalaman realistik menggunakan kendaraan tersebut. Terbagi atas beberapa jenis :

a. *Perang*. Video *Game* simulasi kendaraan yang sempat tenar di tahun 90-an ini mengajak pemain untuk menaiki kendaraan dan berperang melawan kendaraan lainnya. Dan kebanyakan diantaranya memiliki judul sama dengan nama kendaraannya. Contoh : Apache 64, Comanche, Abrams, YF-23, F-16 fighting eagle.

b. *Balapan.* Dari namanya sudah jelas, siapa sampai duluan di garis finish dialah pemenangnya! Terkadang malah pemain dapat memilih kendaraan, mendandani, upgrade mesin bahkan mengecatnya. Contoh: Top Gear, Test Drive, Sega Rally Championship, Daytona, Grand Turismo, Need For Speed, Mario Cart, ManXTT.

c. *Luar Angkasa.* Walau masih dapat dikategorikan simulasi kendaraan perang, tetapi segala unsur fiksi ilmiah dan banyaknya judul yang beredar membuat subgenre ini pantas dikategorikan diluar simulasi kendaraan perang. Jenis ini memungkinkan pemain untuk menjelajah luar angkasa, berperang dengan mahluk alien, mendarat di planet antah berantah atau sekedar ingin merasakan bagaimana menjadi kapten di film fiksi ilmiah kesayangan kamu. Contoh: Wing Commander, Freelancer , Star Wars X-Wing, Star Wars Tie Fighter, dll.

d. *Mecha.* Pendapat bahwa hampir tidak ada orang yang terekspos oleh film robot jepang saat kecilnya tidak memimpikan ingin mengendalikan robot, memang sulit dibantah. Dipopulerkan oleh serial Mechwarrior oleh Activision,

subgenre Simulasi Mecha ini memungkinkan pemainnya untuk mengendalikan robot dan menggunakannya untuk menghancurkan gedung, helikopter dan tentu saja robot lainnya. Contoh: Mechwarrior, Gundam Last war Chronicles, dan Armored Core.

B. Edukasi

- *EduGames.* Video *Game* jenis ini dibuat dengan tujuan spesifik sebagai alat pendidikan, entah untuk belajr mengenal warna untuk balita, mengenal huruf dan angka, matematika, sampai belajar bahasa asing. Developer yang membuatnya, harus memperhitungkan berbagai hal agar *Game* ini benar-benar dapat mendidik, menambah pengetahuan dan meningkatkan ketrampilan yang memainkannya. Target segmentasi pemain harus pula disesuaikan dengan tingkat kesulitan dan design visual ataupun animasinya. Contoh edu*Game*s : Bobi Bola, Dora the explorer, Petualangan Billy dan Tracy.
- *Puzzle.* Video *Game* jenis ini sesuai namanya berintikan mengenai pemecahan teka-teki, baik itu menyusun balok, menyamakan warna bola, memecahkan perhitungan matematika, melewati labirin, sampai mendorong-dorong kota masuk ke tempat yang seharusnya, itu semua termasuk dalam jenis ini. Sering pula permainan jenis ini adalah juga unsur permainan dalam video *Game* petualangan maupun *Game* edukasi. Tetris, Minesweeper, Bejeweled, Sokoban dan Bomberman.

C. *Entertainment*

- Aksi – *Shooting*, (tembak-tembakan , atau hajar-hajaran bisa juga tusuk-tusukan, tergantung cerita dan tokoh di dalamnya), video *Game* jenis ini sangat memerlukan kecepatan refleks, koordinasi mata-tangan, juga timing, inti dari *Game* jenis ini adalah tembak, tembak dan tembak. Termasuk didalam-nya :

a. *First person shooting (FPS)* seperti Counter Strike dan Call of Duty.

b. *Drive n' shoot,* menggunakan unsur simulasi kendaraan tetapi tetap dengan tujuan utama menembak dan menghancurkan lawan, contoh : Spy Hunter, Rock and Roll. Racing, Road Rash.

c. *Shoot em' up*, seperti Raiden, 1942, dan gradius.

d. *Beat 'em up* (tonjok hajar) seperti Double Dragon dan Final Fight, lalu *hack and slash* (tusuk bebas) seperti Shinobi dan Legend of Kage.

e. *Light gun shooting*, yang menggunakan alay yang umumnya berbentuk seperti senjata, seperti Virtua Cop dan Time Crisis.

- Fighting (pertarungan) Ada yang mengelompokan video *Game* fighting di bagian Aksi, namun penulis berpendapat berbeda, jenis ini memang memerlukan kecepatan refleks dan koordinasi mata-tangan, tetapi inti dari *Game* ini adalah penguasaan jurus (hafal caranya dan lancar mengeksekusinya), pengenalan karakter dan timing sangatlah penting, o iya, *combo*-pun menjadi esensial untuk mengalahkan lawan secepat mungkin. Dan berbeda seperti *Game* Aksi pada umumnya yang umumnya hanya melawan Artificial Intellegence atau istilah umumnya *melawan komputer* saja, pemain jenis *fighting Game* ini baru teruji kemampuan sesungguhnya dengan melawan pemain lainnya. Seri Street Fighter, Tekken, Mortal Kombat, Soul Calibur dan King of Fighter adalah contohnya.

D. Aksi

- Petualangan. Memasuki gua bawah tanah, melompati bebatuan di antara lahar, bergelayutan dari pohon satu ke pohon lain, bergulat dengan ular sambil mencari kunci untuk membuka pintu kuil legendaris, atau sekedar mencari telepon umum untuk mendapatkan misi berikutnya, itulah beberapa dari banyak hal yang karakter pemain harus lakukan dan lalui dalam video *Game* jenis ini. Menurut penulis, *Game* jenis ini sudah berkembang jauh hingga menjadi genre campuran *action beat-em up* juga, dan sekarang, di tahun 2000 an, jenis ini cenderung untuk memiliki visual 3D dan sudut pandang orang ke-tiga. Tomb Rider, Grand Theft Auto dan Prince of Persia termasuk didalamnya.
- Petualangan. Bedanya dengan jenis video *Game* aksi-petualangan, refleks dan kelihaian pemain dalam bergerak, berlari, melompat hingga memecut atau menembak tidak diperlukan di sini. Video *Game* murni petualangan lebih menekankan pada jalan cerita dan kemampuan berpikir pemain dalam menganalisa tempat secara visual, memecahkan teka-teki maupun menyimpulkan

rangkaian peristiwa dan percakapan karakter hingga penggunaan benda-benda tepat pada tempat yang tepat. Termasuk didalamnya:

a. *Petualangan dengan teks* atau sistem tunjuk dan klik, contoh: Kings Quest, Space Quest, Heroes Quest, Monkey Island, Sam and Max,

b. *Novel atau film interaktif*, seperti *Game* "*dating*" yang banyak beredar di jepang, Dragons Lair dan Night Trap.

- Role Playing. Video *Game* jenis ini sesuai dengan terjemahannya, bermain peran, memiliki penekanan pada tokoh/peran perwakilan pemain di dalam permainan, yang biasanya adalah tokoh utamanya, dimana seiring kita memainkannya, karakter tersebut dapat berubah dan berkembang ke arah yang diinginkan pemain (biasanya menjadi semakin hebat, semakin kuat, semakin berpengaruh, dll) dalam berbagai parameter yang biasanya ditentukan dengan naiknya level, baik dari status kepintaran, kecepatan dan kekuatan karakter, senjata yang semakin sakti, ataupun jumlah teman maupun mahluk peliharaan.Secara kebudayaan, pengembang *Game* Jepang biasanya membuat Role Playing *Game* (RPG) ke arah cerita linear yang diarahkan seolah karakter kita adalah tokoh dalam cerita itu, seperti Final Fantasy, Dragon Quest dan Xenogears. Sedangkan pengembang *Game* RPG Eropa, cenderung membuat karakter kita bebas memilih jalan cerita sendiri secara non-linear, seperti Ultima, Never Winter Nights, baldurs gate, Elder Scroll, dan Fallout.
- *Olahraga*. Singkat padat jelas, bermain sport di PC atau konsol anda. Biasanya permainannya diusahakan serealistik mungkin walau kadang ada yang menambah unsur fiksi seperti NBA JAM. Contohnya pun jelas, Seri Winning Eleven, seri NBA, seri FIFA, John Madden NFL, Lakers vs Celtics, Tony hawk pro skater, dll.
- Multiplayer Online. *Game* yang lagi trend di Indonesia bahkan dunia,menjadi salah satu titik balik mengapa dunia *Game* dan internet di Indonesia dapat berkembang. Dan karena dimainkan online dan dengan sistem pembayaran menggunakan voucher, pembajakan sudah tidak menjadi masalah lagi. *Game* yang dapat dimainkan secara bersamaan oleh lebih dari 2 orang (bahkan dapat mencapai puluhan ribu orang dalam satu waktu) membuat pemain

dapat bermain bersama dalam satu dunia virtual dari sekedar chatting hingga membunuh naga bersama teman yang entah bermain di mana. Umumnya permainan tipe ini dimainkan di PC dan bertema RPG, walau ada juga yang bertema music atau action. Contoh: Ragnarok online, O2jam, World of Warcraft, Ayo Dance, Lineage, Rose online.

- Casual *Game*s. Sesuai namanya, *Game* yang casual itu tidak kompleks, mainnya rileks dan sangat mudah untuk dipelajari (bahkan cenderung langsung bisa dimainkan). Jenis ini biasanya memerlukan spesifikasi komputer yang standar pada jamannya dan ukurannya tidak lebih dari 100 MB karena biasanya dapat di download versi demo-nya di website resminya. Genre permainannya biasanya puzzle atau action sederhana dan umumnya dapat dimainkan hanya menggunakan mouse (biasanya *Game* lain menggunakan banyak tombol tergantung *Game*-nya). Contoh: Diner Dash, Sally Salon, Bejeweled, Zuma, Feeding Frenzy, Insaniquarium.

2.2 *Game* Design

Game Design merupakan proses menciptakan konten dan aturan permainan. *Design Game* yang baik adalah proses menciptakan tujuan dimana seorang pemain merasa termotivasi untuk mencapainya dan seorang pemain harus mengikuti aturan saat dia membuat keputusan yang berarti dalam mengejar tujuan-tujuan tersebut. Menurut Brenda Brathwaite dan Ian Schreiber dalam bukunya *Challenges For Game Designers*, merancang *Game* mencakup sebagai berikut [14]:

a. *World Design*, membuat desain keseluruhan latarbelakang cerita, pengaturan, dan tema dari permainan.
b. *System Design*, membuat desain aturan dan pola matematis yang mendasari dalam sebuah permainan. Ini adalah tugas desain yang umum untuk semua *Game*, karena semua permainan memiliki aturan.
c. *Content Design*, membuat desain karakter, item, teka-teki, dan misi.
d. *Game Writing*, membuat penulisan dialog, text, dan cerita dalam dunia permainan.

e. *Level Design*, menciptakan level dalam permainan berdasarkan tampilan permainan dan penempatan objek hingga tantangan dalam tampilan permainan.
f. *User Interface*, membuat desain yang terdiri dari dua hal yaitu, bagaimana pemain dapat berinteraksi dengan permainan, dan bagaimana pemain menerima informasi dan mengambil hasil dari permainan.

Para pengembang *Game* komersial biasanya mempersiapkan dokumen-dokumen yang berisi *design Game* yang sangat panjang sebelum memulai pembuatan *Game*. Setelah itu dari *design Game* yang telah dibuat kemudian dapat diketahui semua elemen-elemen berbeda yang dibutuhkan dalam pembuatan *Game*, misalnya karakter user, karakter musuh, animasi serangan dan sebagainya. Membuat *Game* akan membutuhkan gambar dari tiap elemen-elemen yang ada, *background image*, dan lagu. Semua hal tersebut dapat dikatakan sebagai *resource Game*. Hal-hal yang penting dalam *Game* adalah sebagai berikut [15]:

1. *Sprites*

Sprites adalah kumpulan gambar-gambar yang digunakan untuk icon, karakter ataupun benda bergerak dalam *Game*. *Sprites* dapat dibuat dari gambar-gambar yang sudah dibuat dalam sebuah *art packages*, download dari internet maupun dibuat secara manual dengan menggunakan perangkat lunak seperti *Adobe photoshop*.

2. *Object*

Sprites tidak dapat melakukan sesuatu dengan sendirinya. *Sprites* hanya menyimpan gambar-gambar dari elemen yang berbeda dalam sebuah fame. *Object* adalah bagian dari *Game* yang mengontrol bagaimana elemen-elemen tersebut bergerak dan berinteraksi satu sama lain. Contohnya adalah karakter dalam *Game*.

3. *Events* dan *Actions*

Events adalah hal-hal penting yang terjadi didalam sebuah *Game*, seperti ketika *objects* saling bertubrukan maupun ketika user menekan tombol pada *keyboard*. *Actions* adalah hal-hal yang terjadi sebagai respon dari sebuah *event*, seperti perubahan arah pergerakan *objects*, perubahan *score*, atau memainkan sebuah lagu.

4. *Backgrounds*

Backgrounds adalah gambar-gambar yang digunakan sebagai gambar latar belakang *Game* dan merupakan jenis lain dari *resource*, seperti *sprites* dan *objects*. Pada umumnya ukuran *backgrounds* sama dengan resolusi *Game*.

5. *Sounds*

Sounds juga termasuk *resource Game*, jenis *sounds* dalam *Game* ada dua, yaitu *music* dan *sound effect*. *Sounds* dapat mempengaruhi atmosfer dalam *Game*. *Masters* berupa lagu yang terus mengalun selama *Game* dimainkan, namun lagu yang dimainkan pada umumnya bergantung pada tempat atau kondisi *Game*. *Sound effect* adalah suara yang hanya terdengar ketika terjadi suatu hal tertentu, misalkan penekanan tombol, karakter terkena pukulan dan sebagainya. Kegunaan dari *sound effect* adalah untuk membantu user mengetahui *action* yang dilakukan maupun hal yang sedang terjadi.

6. Status

Status adalah sekumpulan informasi yang berguna untuk orang yang memainkan sebuah *Game* untuk mengetahui kondisi permainan saat itu. Bergantung dari jenis *Game*-nya, informasi yang disampaikan melalui status dapat bermacam-macam. Berikut beberapa macam status pada *Game* beserta penjelasannya:

- *Score* adalah nilai yang diperoleh oleh orang yang memainkan sebuah *Game*. Status ini sangat sering digunakan pada sebuah *Game* untuk mengetahui kemampuan yang memainkannya.
- *High Score* adalah nilai tertinggi yang pernah dicapai orang yang memainkan sebuah *Game*. Status ini biasanya terdapat pada *Game* yang dapat diselesaikan dalam waktu yang cukup singkat, sehingga membuat orang yang memainkannya tertarik untuk mengulangi permainan dari awal, dengan harapan mendapat *high score* yang lebih tinggi.
- HP (Hit Point) adalah status yang menunjukkan jumlah maksimal *damage* yang dapat diterima oleh karakter dalam *Game*. Apabila bernilai 0 maka karakter tersebut dinyatakan mati/kalah. Status ini biasanya terdapat pada *Game* ber-*genre fighting, action* atau RPG (*Role Playing Game*).

- SP (*Skill Point*) adalah sejumlah poin yang dapat digunakan untuk menggunakan *skill*. Jumlah yang dibutuhkan bergantung dari *skill* yang digunakan. Status ini biasanya terdapat pada *Game* ber-*genre* RPG (*Role Playing Game*).
- *Strength* adalah jumlah kekuatan karakter dalam *Game*. Nilai status biasanya menentukan kekuatan serangan karakter. Status ini biasanya terdapat pada *Game* ber-*genre fighting, action* atau RPG (*Role Playing Game*).
- *Vitality* adalah jumlah pertahanan karakter dalam *Game*. Nilai status ini biasanya akan mempengaruhi jumlah HP dan pertahanan yang dimiliki karakter. Status ini biasanya terdapat pada *Game* ber-*genre fighting, action* atau RPG (*Role Playing Game*).
- *Speed* adalah kecepatan karakter dalam *Game*. Nilai status ini biasanya akan mempengaruhi seberapa cepat karakter akan bergerak atau mendapatkan giliran serta menentukan mudah atau tidaknya terkena serangan musuh. Status ini biasanya terdapat pada *Game* ber-*genre fighting, action* atau RPG (*Role Playing Game*).
- *Intelligence* adalah jumlah kekuatan sihir dari karakter dalam *Game*. Nilai status ini biasanya akan berpengaruh pada jumlah SP, serangan sihir dan ketahanan terhadap serangan sihir. Status ini biasanya terdapat pada *Game* ber-*genre action* atau RPG (*Role Playing Game*).
- *Luck* adalah keberuntungan karakter dalam *Game*. Nilai status ini biasanya akan mempengaruhi karakter dalam mendapatkan hal-hal yang sulit didapatkan dalam *Game*, seperti terjadinya *critical hit*.
- Status *point* adalah sejumlah poin yang biasanya digunakan untuk menambah status karakter seperti *Strength, Vitality, Intelligence, Speed* dan *Luck*. Status ini biasanya terdapat pada *Game* ber-*genre action* atau RPG (*Role Playing Game*).
- *Time* adalah status yang digunakan untuk berbagai macam fungsi dalam *Game*, misal untuk menentukan waktu yang tersisa untuk mencapai tujuan tertentu, jumlah waktu yang telah dihabiskan seseorang untuk memainkan *Game*

tersebut, dan lain sebagainya. Status ini biasanya terdapat pada *Game* ber-*genre fighting, action* atau RPG (*Role Playing Game*).

- *Experience* dalam dunia *Game* adalah sejumlah poin yang biasanya digunakan untuk menentukan *level* karakter dalam *Game*. Status ini biasanya terdapat pada *Game* ber-*genre action* atau RPG (*Role Playing Game*).

2.3 Metode Animasi

Animasi merupakan suatu teknik menampilkan gambar berurut sedemikian rupa sehingga penonton merasakan adanya ilusi gerakan (motion) pada gambar yang ditampilkan [16]. Animasi dapat dibuat dengan tiga teknik berbeda, yaitu *Image*, *Xoring*, dan *Make*, dan dalam pergerakannya dapat bertipe *object sprite* atau *object frame*, dan juga bisa dibedakan atas metode animasi yang digunakan antara animasi *frame*, *bibliting* dan *realtime*. Walaupun terbagi atas berbagai definisi berbeda, tapi dalam prakteknya teori-teori tersebut dapat digabungkan atau saling berhubungan sehingga tidak murni dipakai sendiri. Macam-macam animasi yang digunakan dalam membuat sebuah *Game* akan dijelaskan sebagai berikut [29]:

a. Animasi dengan Teknik *Image*

Animasi dengan teknik ini adalah menyimpan *image* sebagai sebuah *sprite* dalam memori yang kemudian akan ditampilkan di *background*-nya. Dalam teknik ini animasi yang disimpan harus berlatar belakang sesuai *background*-nya. Animasi dengan teknik ini biasanya sulit dalam pembuatan gambarnya, sebab harus banyak dan melakukan penyamaan dan posisi. Akan tetapi teknik ini mudah dalam hal memainkan animasinya.

b. Animasi dengan Teknik *Xoring*

Teknik ini adalah teknik animasi yang mudah dan sederhana, sebab selain gambarnya satu *sprite*, cara menampilkannya juga jauh lebih mudah dibanding dengan teknik sebelumnya. Pembuatan gambarnya sangat mudah, sebab yang dibuat adalah *sprite*-nya saja dan tidak perlu menyamakan dengan *background*-nya. Kelemahan dari teknik ini adalah memiliki efek buruk yaitu tembus pandang

dan mengganti warna *sprite*, maka hal ini tidak baik digunakan dalam animasi yang *background*-nya bergambar.

c. Animasi dengan Teknik *Make*

Animasi dengan teknik ini biasanya digunakan untuk animasi umum, tapi biasanya digunakan untuk proses pembuatan animasi. Animasi dengan teknik ini memiliki *sprite* yang terus menerus di generate oleh program, kemudian ditampilkan dengan perhitungan tertentu. Animasi ini biasanya dilakukan oleh 3D *modelling* dan *shading* oleh *software* seperti AutoCad, 3D Studio, Presidio 3D Workshop dan lain-lain. Operasi diatas haruslah di generate secara langsung dengan perhitungan sehingga saat pembuatan hampir bersamaan dengan saat menampilkannya.

d. Animasi dengan Tipe *Object Sprite*

Animasi ini menggunakan *sprite* sebagai pemeran utama sedangkan *object* lainnya hanya *background* diam. Prosesnya adalah mebuat gambar *sprite* dengan latar belakang warna hitam, lalu dibuat juga *sprite* yang sama tetapi berwarna hitam dan latar belakangnya adalah warna tertinggi, kemudian ditempatkan dengan pertama-tama menyimpan *background* yang akan ditimpa oleh *sprite* dan ditempatkan *sprite* dengan Xor dan Ditimpa dengan warna tertinggi secara Xor.

e. Animasi dengan Tipe *Object Frame*

Animasi ini menitik beratkan animasi yang dimainkan hanya pada *sprite object*-nya saja, akan tetapi seluruh *background*-nya juga seolah-olah ikut digerakkan.

f. Metode Animasi *Frame*

Metode ini adalah metode animasi yang mendukung tipe *object frame*. Karena metode animasi dengan metode *fullscreen*, maka *frame* yang tampil haruslah disiapkan terlebih dahulu dalam bebera page sebelumnya. Karena hal tersebut maka pengambilan gambarnya haruslah sangat cepat, sehingga tidak menjadikan animasi lamban dan tersendat. Animasi *frame* ini haruslah menampilkan gambar *fullscreen* yang bergerak, agar efek tersendat dari pergantian *frame* tidak menyolok.

g. Metode Animasi *BitBlt*

Metode animasi ini biasanya disebut *sprite animation, array animation, blocked animation, partial screen animation, snapshot animation* atau *arcade animation*. Prinsip dari metode ini adalah menyimpan *image* dan memainkan animasinya dalam bentuk satu atau bebera *sprite* kecil. BitBlt ini digunakan untuk membuat program permainan dan program berbasis grafis terutama dalam membuat animasi sederhana.

h. Metode Animasi *Real-Time*

Dalam metode ini biasanya semua animasi yang sedang tampil atau yang akan dibuat dilakukan bersama sehingga tidak perlu disiapkan terlebih dahulu. Karena animasi ini lambat dan tersendat maka animasi dengan metode ini akan bagus jika pergerakan yang akan dilakukan adalah tidak diketahui sebelumnya dan tiba-tiba muncul. Metode ini tidak disarankan untuk animasi biasa-biasa saja, akan tetapi sebaiknya digunakan untuk keperluan khusus seperti *rotating* dan *tweening*.

2.3.1 Transformasi 2 Dimensi

Transformasi 2 dimensi merupakan adalah pemodelan dari objek 2 dimensi yang melakukan berbagai operasi transformasi geometri, dan pada dasarnya transformasi ini berfungsi memindahkan objek tanpa merusak bentuk. Macam-macam transformasi geometri adalah sebagai berikut [17] :

1. Translasi

Transformasi translasi merupakan suatu operasi yang menyebabkan perpindahan objek 2D dari satu tempat ke tempat yang lain. Perubahan ini berlaku dalam arah yang sejajar dengan sumbu X dan sumbu Y. Translasi dilakukan dengan penambahan translasi pada suatu titik koordinat dengan translation vector, yaitu (dx,dy), dimana dx adalah translasi menurut sumbu x dan dy adalah translasi menurut sumbu y. Koorinat baru titik yang ditranslasi dapat diperoleh dengan menggunakan rumus :

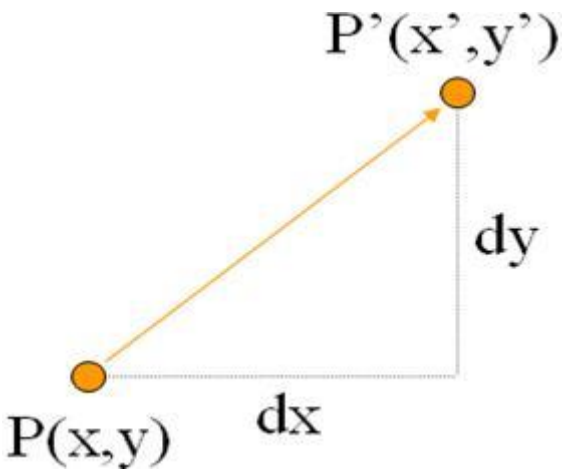
(x,y) = titik asal sebelum translasi

(x`,y`) = titik baru hasil translasi

x`= x + dx

y`= y + dy

Gambar 2.8 adalah translasi atau perpindahan objek dari titik P ke titik P` secara linier.



Gambar 2. 8 *Translasi* [17]

2. *Scalling*

Scalling atau disebut penskalaan adalah suatu operasi yang membuat suatu objek berubah ukurannya baik menjadi mengecil ataupun membesar secara seragam atau tidak seragam tergantung pada faktor penskalaan (scalling factor) yaitu (mx,my) yang diberikan. mx adalah faktor penskalaan menurut sumbu x dan my faktor penskalaan menurut sumbu y. Koordinat baru diperoleh dengan menggunakan rumus :

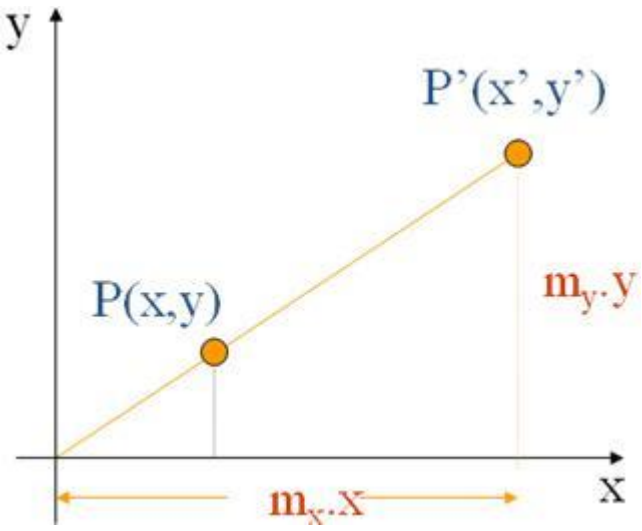
(x,y) = titik asal sebelum diskala

(x`,y`) = titik setelah diskala

x`= x * mx

y`= y * my

Gambar 2.9 adalah *scalling* atau perubahan ukuran objek dari titik P ke titik P`.



Gambar 2. 9 *Scalling* [17]

3. Rotasi

Rotasi adalah suatu operasi yang menyebabkan objek bergerak berputar pada titik pusat atau pada sumbu putar yang dipilih berdasarkan sudut putaran tertentu. Untuk melakukan rotasi diperlukan sudut rotasi θ dan pivot point (xp,yp) dimana objek akan dirotasi. Putaran biasa dilakukan pada satu titik terhadap sesuatu sumbu tertentu misalnya sumbu x, sumbu y atau garis tertentu yang sejajar dengan sembarang sumbu tersebut. Titik acuan putaran dapat sembarang baik di titik pusat atau pada titik yang lain. Aturan dalam geometri, jika putaran dilakukan searah jarum jam, maka nilai sudutnya adalah negatif. Sebaliknya, jika dilakukan berlawanan arah dengan arah jarum jam nilai sudutnya adalah positif.

Rumus transformasi untuk rotasi suatu titik(x,y) dengan sudut rotasi θ sebagai berikut :

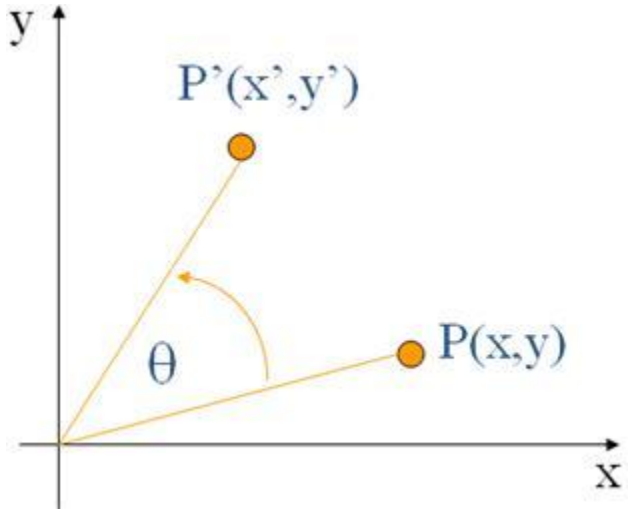
(x,y) = titik asal sebelum diskala

(x`,y`) = titik setelah diskala

$x` = x \cos(\theta) - y \sin(\theta)$

$y` = x \sin(\theta) + y \cos(\theta)$

Gambar 2.10 adalah rotasi atau perpindahan obyek dari titik P ke titik P', yang berupa pemindahan berputar sebesar sudut.



Gambar 2. 10 Rotasi [17]

2.4 *Artificial Intelligence* (AI)

Kecerdasan buatan sering disebut juga dengan AI (*Artificial Intelligence*). AI merupakan salah satu bagian ilmu komputer yang mempelajari tentang bagaimana caranya agar komputer dapat melakukan pekerjaan seperti yang dapat dilakukan oleh manusia. Pada awal diciptakannya, komputer hanya difungsikan sebagai alat hitung saja. Namun seiring dengan perkembangan jaman, maka peran komputer semakin mendominasi kenidupan umat manusia. Sekarang komputer tidak hanya digunakan sebagai alat hitung, tetapi lebih dari itu, komputer diharapkan untuk dapat dimanfaatkan untuk mengerjakan segala sesuatu yang bias dikerjakan oleh manusia.

Agar komputer bisa bertindak seperti apa yang dapat dilakukan manusia, maka komputer juga harus diberi bekal pengetahuan dan mempunyai kemampuan untuk menalar. Untuk itu pada *artificial intelligence* (AI), akan mencoba untuk memberikan beberapa metoda dengan membekali komputer oleh *artificial intelligence* (AI) agar komputer dapat menjadi pintar. Kecerdasan buatan dapat ditinjau dari berbagai sudut pandang, antara lain :

1) Sudut Pandang Kecerdasan.

Kecerdasan buatan akan membuat mesin menjadi ‘cerdas’ (mampu berbuat seperti apa yang dilakukan oleh manusia).

2) Sudut Pandang Penelitian.

Kecerdasan buatan adalah suatu studi bagaimana membuat agar komputer dapat melakukan sesuatu dengan baik yang dilakukan oleh manusia. Topik yang sering dibahas oleh para peneliti meliputi *mudane task* (*vision and speech, translation, robot control*), *formal task* (*Games*, matematika (*geometri*, kalkulus integral), *expert task* (analisis finansial, analisis medikal, analisis ilmu pengetahuan).

3) Sudut Pandang bisnis.

Kecerdasan buatan adalah kumpulan peralatan *tools* yang sangat *powerfull* dan metodologis dalam menyelesaikan masalah-masalah bisnis.

4) Sundut Pandang Pemrograman.

Kecerdasaan buatan meliputi studi tentang pemrograman simbolik, penyelesaian masalah (*problem solving*) dan pencarian (*searching*).

Secara umum, untuk membangun suatu sistem yang mampu menyelesaikan masalah, perlu dipertimbangkan 4 hal:

1. Mendefinisikan masalah dengan tepat. Pendefinisian ini mencakup spesifikasi yang tepat mengenai keadaan awal dan solusi yang diharapkan.
2. Menganalisis masalah tersebut serta mencari beberapa teknik penyelesaian masalah yang sesuai.
3. Merepresentasikan pengetahuan yang perlu untuk menyelesaikan masalah tersebut.
4. Memilih teknik penyelesaian masalah yang terbaik.

2.4.1 Teknik Penyelesaian Masalah AI (*Artificial Intelligence*)

Terdapat empat teknik dasar penyelesaian masalah yang terdapat pada bidang *artificial intelegence* diantaranya adalah [19]:

a. Searching

Pada teknik searching atau pencarian ini terdiri dari beberapa langkah untuk merealisasikannya. Langkah pertama adalah mendefinisikan ruang masalah untuk suatu masalah yang dihadapi. Langkah kedua adalah

mendefinisikan aturan produksi yang digunakan untuk mengubah suatu keadaan ke keadaan lainnya. Langkag terakhir adalah memilih metode pencarian yang tepat sehingga dapat menemukan solusi terbaik.

b. Reasoning

Teknik *reasoning* atau penalaran merupakan teknik penyelesaian masalah dengan cara merepresentasikan masalah ke dalam basis pengetahuan menggunakan *logic* atau bahasa formal (bahasa yang dipahami komputer). Teknik ini melakukan proses penalaran berdasarkan basis pengetahuannya untuk menemukan solusi.

c. Planning

Planning adalah suatu metode penyelesaian masalah dengan cara memecah masalah ke dalam sub-sub masalah yang lebih kecil, menyelesaikan sub-sub masalah satu demi satu, kemudian menggabungkan solusi-solusi dari sub-sub masalah tersebut menjadi sebuah solusi lengkap dengan tetap mengingat dan menangani interaksi yang terdapat pada sub-sub masalah tersebut.

d. Learning

Pada ketiga teknik sebelumnya, seseorang harus mengetahui aturan yang berlaku untuk system yang akan dibangunnya. Tetapi, pada masalah tertentu terkadang aturan tidak bisa didefinisikan secara benar ataupun lengkan. Hal tersebut mungkin dikarenakan data-data yang didapat tidak lengkap. Melalui teknik yang disebut *learning* ini, secara otomatis aturan yang diharapkan bisa berlaku umum untuk data-data yang belum pasti diketahui dapat ditemukan.

2.5 Logika *Fuzzy*

Fuzzy secara bahasa diartikan sebagai kabur atau samar-samar. Suatu nilai dapat bernilai besar atau salah secara bersamaan. Dalam fuzzy dikenal derajat keanggotaan yang memiliki rentang nilai 0 (nol) hingga 1(satu). Berbeda dengan himpunan tegas yang memiliki nilai 1 atau 0 (ya atau tidak).

Logika Fuzzy merupakan sesuatu *logika* yang memiliki nilai kekaburan atau kesamaran (*fuzzyness*) antara benar atau salah. Dalam teori *logika fuzzy* suatu nilai bisa bernilai benar atau salah secara bersama. Namun berapa besar

keberadaan dan kesalahan suatu tergantung pada bobot keanggotaan yang dimilikinya. *Logika fuzzy* memiliki derajat keanggotaan dalam rentang 0 hingga 1. Berbeda dengan logika digital yang hanya memiliki dua nilai 1 atau 0. *Logika fuzzy* digunakan untuk menterjemahkan suatu besaran yang diekspresikan menggunakan bahasa (*linguistic*), misalkan besaran kecepatan laju kendaraan yang diekspresikan dengan pelan, agak cepat, cepat, dan sangat cepat. Dan *logika fuzzy* menunjukan sejauh mana suatu nilai itu benar dan sejauh mana suatu nilai itu salah. Tidak seperti *logika klasik* (scrisp)/ tegas, suatu nilai hanya mempunyai 2 kemungkinan yaitu merupakan suatu anggota himpunan atau tidak. Derajat keanggotaan 0 (nol) artinya nilai bukan merupakan anggota himpunan dan 1 (satu) berarti nilai tersebut adalah anggota himpunan.

Logika fuzzy adalah suatu cara yang tepat untuk memetakan suatu ruang *input* kedalam suatu ruang *output*, mempunyai nilai kontinyu. *Fuzzy* dinyatakan dalam derajat dari suatu keanggotaan dan derajat dari kebenaran. Oleh sebab itu sesuatu dapat dikatakan sebagian benar dan sebagian salah pada waktu yang sama. Algoritma *fuzzy* memungkinkan nilai keanggotaan antara 0 dan 1, tingkat keabuan dan juga hitam dan putih, dan dalam bentuk *linguistik*, konsep tidak pasti seperti "sedikit", "lumayan" dan "sangat" [18].

Kelebihan dari teori logika *fuzzy* adalah kemampuan dalam proses penalaran secara bahasa (*linguistic reasoning*). Sehingga dalam perancangannya tidak memerlukan persamaan matematik dari objek yang akan dikendalikan.

2.5.1 Konsep Dasar Logika *Fuzzy*

Teori himpunan *fuzzy* diperkenalkan pertama kali oleh Lotfi A. Zadeh pada tahun 1965. Ada beberapa alasan mengapa orang menggunakan logika *fuzzy*, antara lain :

1. Konsep logika *fuzzy* mudah dimengerti. Konsep matematis yang mendasari penalaran *fuzzy* sangat sederhana dan mudah dimengerti.
2. Logika *fuzzy* sangat *fleksibel*.
3. Logika *fuzzy* memiliki toleransi terhadap data-data yang tidak tepat.
4. Logika *fuzzy* mampu memodelkan fungsi-fungsi *nonlinear* yang sangat kompleks.

5. Logika *fuzzy* dapat membangun dan mengaplikasikan pengalamanpengalaman para pakar secara langsung tanpa harus melalui proses pelatihan.
6. Logika *fuzzy* dapat bekerjasama dengan teknik-teknik kendali secara konvensional.
7. Logika *fuzzy* didasarkan pada bahasa alami.

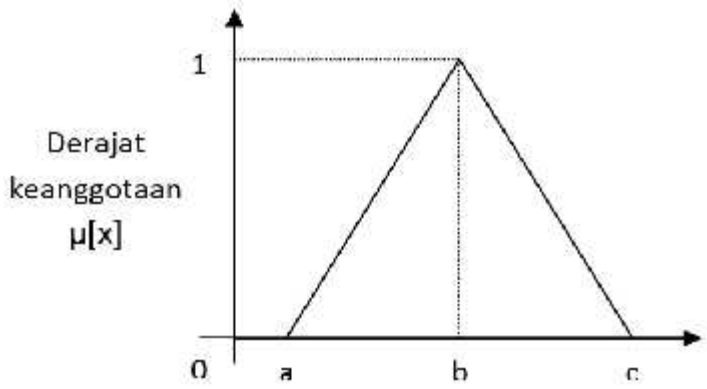
2.5.2 Fungsi Keanggotaan (*Membership Function*)

Fungsi keanggotaan adalah suatu kurva yang menunjukkan pemetaan titik-titik *input* data ke dalam nilai keanggotaannya (disebut juga 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. Derajat keanggotaan dalam himpunan (*degree of membership*) dilambangkan dengan μ. Dalam sistem *fuzzy* banyak dikenal bermacam-macam fungsi keanggotaan (*membership function*).

fungsi keanggotaan yang dipakai adalah Representasi Kurva Segitiga dan Representasi Kurva Bahu.

a. Representasi Kurva Segitiga

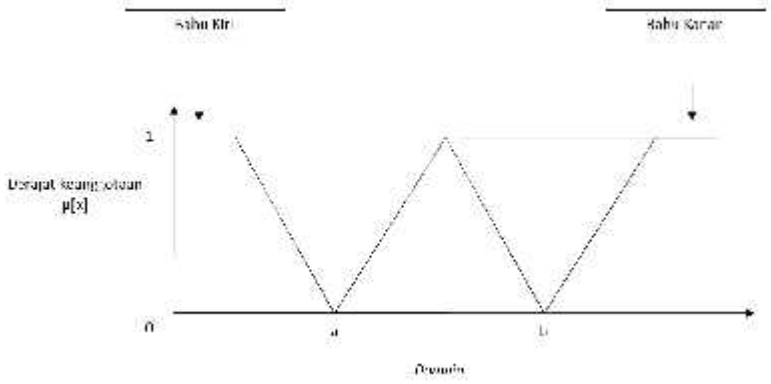
Kurva segitiga pada dasarnya merupakan gabungan antara dua garis.



Gambar 2. 11 Fungsi Keanggotaan Kurva Segitiga

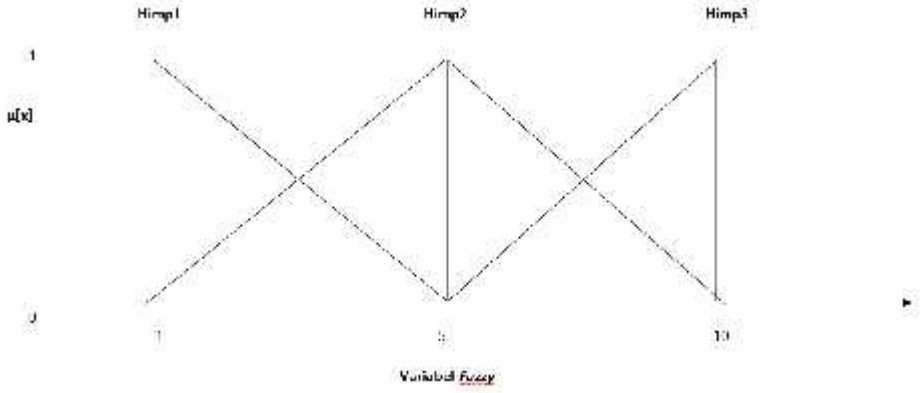
b. Representasi Kurva Bahu

Kurva bahu merupakan daerah yang terletak di tengah-tengah suatu variabel yang direpresentasikan dalam bentuk segitiga, pada sisi kanan dan sisi kirinya akan naik dan turun. Himpunan *fuzzy* bahu digunakan untuk mengakhiri variabel suatu daerah *fuzzy*.



Gambar 2. 12 Fungsi Keanggotaan Kurva Bahu

Fungsi keanggotaan pada kurva segitiga dan fungsi keanggotaan pada kurva bahu dapat dilihat sebagai berikut :



Gambar 2. 13 Himpunan *Fuzzy* untuk Suatu Variabel

2.5.3 *Inferensi Fuzzy* (Logika Pengambilan Keputusan)

Setelah fungsi keanggotaan untuk variabel masukan dan keluarannya ditentukan, basis aturan pengendalian dapat dikembangkan untuk menghubungkan aksi keluaran pengendali terhadap kondisi masukannya. Tahap ini disebut sebagai tahap *inferensi*, yakni bagian penentuan aturan dari sistem logika *fuzzy*. Sejumlah aturan dapat dibuat untuk menentukan aksi pengendali *fuzzy* [10].

Pada basis aturan, aturan *If-Then* tersebut dapat menghubungkan banyak variabel masukan dan keluaran. Masukan *x* dipetakan menjadi keluaran *y*. Aturan *if-then* diinterpretasikan sebagai implikasi *fuzzy*. Terdapat banyak sekali model

interpretasi implikasi yang telah dikembangkan. Pada aplikasi *Game* ini, metode yang digunakan adalah Metode MAMDANI.

Metode Mamdani sering juga dikenal dengan nama Metode Max-Min. Metode ini diperkenalkan oleh Ebrahim Mamdani pada tahun 1975. Untuk mendapatkan *output*, diperlukan empat tahapan :

1. Pembentukan Himpunan *Fuzzy*

   Pada Metode Mamdani, baik variabel *input* maupun variabel *output* dibagi menjadi satu atau lebih himpunan *fuzzy*.

2. Aplikasi Fungsi Implikasi

   Pada Metode Mamdani, fungsi implikasi yang digunakan adalah Min.

3. Komposisi Aturan

   Tidak seperti penalaran monoton, apabila sistem terdiri dari beberapa aturan, maka inferensi diperoleh dari kumpulan dan korelasi antar aturan. Pada Pembangunan Aplikasi *Building of House* ini metode yang digunakan dalam melakukan inferensi *system fuzzy*, yaitu metode *Maximum*.

   Pada metode ini, solusi himpunan *fuzzy* diperoleh dengan cara mengambil nilai maksimum aturan, kemudian menggunakannya untuk memodifikasi daerah *fuzzy*, dan mengaplikasikannya ke *output* dengan menggunakan operator OR (*union*). Secara umum dapat dituliskan [10]:

$$\mu_{sf}[x_i] \leftarrow \max(\mu_{sf}[x_i], \mu_{kf}[x_i])$$

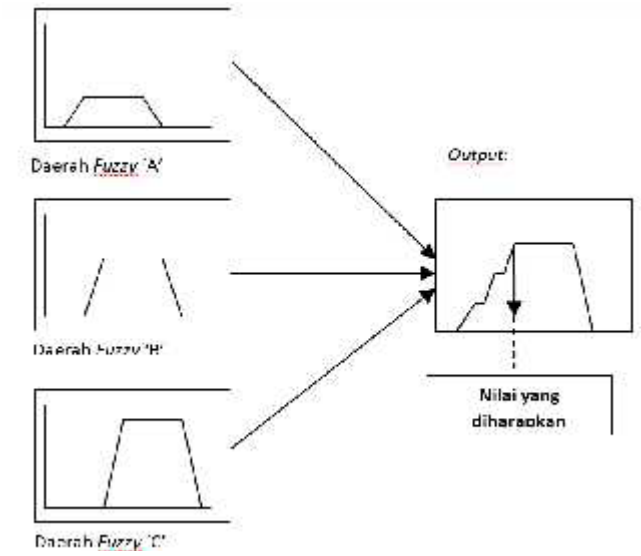
$\mu_{sf}[x_i]$ = nilai keanggotaan solusi *fuzzy* sampai aturan ke-i;

$\mu_{kf}[x_i]$ = nilai keanggotaan konsekuen *fuzzy* sampai aturan ke-i;

   Apabila digunakan fungsi implikasi MIN, maka metode komposisi ini sering disebut dengan nama MAX-MIN atau MIN-MAX atau MAMDANI.

4. Penegasan (*defuzzy*)

   Input dari proses *defuzzifikasi* adalah suatu himpunan *fuzzy* yang diperoleh dari komposisi aturan-aturan *fuzzy*, sedangkan *output* yang dihasilkan merupakan suatu bilangan pada *domain* himpunan *fuzzy* tersebut. Sehingga jika diberikan suatu himpunan *fuzzy* dalam *range* tertentu, maka harus dapat diambil suatu nilai crisp tertentu sebagai *output* terlihat pada gambar berikut :



Gambar 2. 14 Proses *Defuzzifikasi*

2.6 Interaksi Manusia dan Komputer

Menurut Ben Shneiderman dalam bukunya yang berjudul *Designing the User Interface-Strategies for Effective Human-Computer Interaction*, bahwa Interaksi Manusia dan Komputer (IMK) atau *Human-Computer Interaction* (HCI) adalah disiplin ilmu yang berhubungan dengan perancangan, evaluasi, dan implementasi sistem komputer interaktif untuk digunakan oleh manusia, serta studi fenomena-fenomena besar yang berhubungan dengannya.

Selain itu dalam bukunya Ben Shneiderman mengemukakan, terdapat 8 aturan (*Eight Golden Rules*) yang digunakan dalam perancangan antarmuka adalah sebagai berikut:

1. Berusaha untuk konsisten, urutan konsisten diperlukan pada situasi yang sama. Konsistensi harus digunakan pada *prompt*, menu, layar bantuan, warna tampilan, kapitalisasi, huruf, dan sebagainya.
2. Memungkinkan *frequent users* menggunakan *shortcuts*, ada kebutuhan dari pengguna yang sudah ahli untuk meningkatkan kecepatan interaksi, sehingga diperlukan singkatan, tombol fungsi, perintah tersembunyi, dan fasilitas makro.
3. Memberikan umpan balik yang informatif, untuk setiap tindakan operator, sebaiknya disertakan suatu sistem umpan balik (*feedback*). Untuk tindakan

yang sering dilakukan dan tidak terlalu penting, dapat diberikan umpan balik yang sederhana. Tetapi ketika tindakan merupakan hal yang penting, maka umpan balik sebaiknya lebih substansial. Misalnya muncul suatu suara ketika salah menekan tombol pada waktu input data atau muncul pesan kesalahannya.

4. Merancang dialog yang memberikan penutupan (keadaan akhir), urutan tindakan sebaiknya diorganisir dalam suatu kelompok dengan bagian awal, tengah, dan akhir. Umpan balik yang informatif akan memberikan indikasi bahwa cara yang dilakukan sudah benar dan dapat mempersiapkan kelompok tindakan berikutnya.
5. Memberikan pencegahan kesalahan dan penanganan yang sederhana, Sedapat mungkin sistem dirancang sehingga pengguna tidak dapat melakukan kesalahan fatal. Jika kesalahan terjadi, sistem dapat mendeteksi kesalahan dengan cepat dan memberikan mekanisme yang sederhana dan mudah dipahami untuk penanganan kesalahan.
6. Memungkinkan pembalikan aksi yang mudah, hal ini dapat mengurangi kekuatiran pengguna karena pengguna mengetahui kesalahan yang dilakukan dapat dibatalkan, sehingga pengguna tidak takut untuk mengekplorasi pilihan-pilihan lain yang belum biasa digunakan
7. Mendukung pusat kendali internal (*internal locus of control*), pengguna ingin menjadi pengontrol sistem dan sistem akan merespon tindakan yang dilakukan pengguna daripada pengguna merasa bahwa sistem mengontrol pengguna. Sebaiknya sistem dirancang sedemikan rupa sehingga pengguna menjadi inisiator daripada responden.

Mengurangi beban ingatan jangka pendek, keterbatasan ingatan manusia membutuhkan tampilan yang sederhana atau banyak tampilan halaman yang sebaiknya disatukan, serta diberikan cukup waktu pelatihan untuk kode, mnemonic, dan urutan tindakan.

2.7 *Unified Modeling Language* (UML)

UML adalah bahasa pemodelan standar yang terdiri dari serangkaian diagram, dikembangkan untuk membantu sistem dan pengembang perangkat lunak menyelesaikan tugas-tugas berikut: Spesifikasi, Visualisasi, Desain arsitektur, Konstruksi, Simulasi dan Pengujian serta Dokumentasi

UML pada awalnya dikembangkan dengan ide mempromosikan komunikasi dan produktivitas antara pengembang sistem berorientasi objek, tetapi kekuatan tampak jelas dari UML telah menyebabkannya untuk membuat terobosan ke dalam setiap jenis sistem dan pengembangan perangkat lunak.

2.7.1 Konsep Dasar UML

Dari berbagai penjelasan rumit yang terdapat di dokumen dan buku-buku UML. Sebenarnya konsepsi dasar UML bisa kita rangkumkan dalam gambar 2.15.

Major Area	*View*	*Diagrams*	*Main Concepts*
structural	static view	class diagram	class, association, generalization, dependency, realization, interface
	use case view	use case diagram	use case, actor, association, extend, include, use case generalization
	implementation view	component diagram	component, interface, dependency, realization
	deployment view	deployment diagram	node, component, dependency, location
dynamic	state machine view	statechart diagram	state, event, transition, action
	activity view	activity diagram	state, activity, completion transition, fork, join
	interaction view	sequence diagram	interaction, object, message, activation
		collaboration diagram	collaboration, interaction, collaboration role, message
model management	model management view	class diagram	package, subsystem, model
extensibility	all	all	constraint, stereotype, tagged values

Gambar 2. 15 Konsep Dasar UML

Seperti juga tercantum pada gambar di atas UML mendefinisikan diagram-diagram sebagai berikut:

1. *Use Case Diagram*

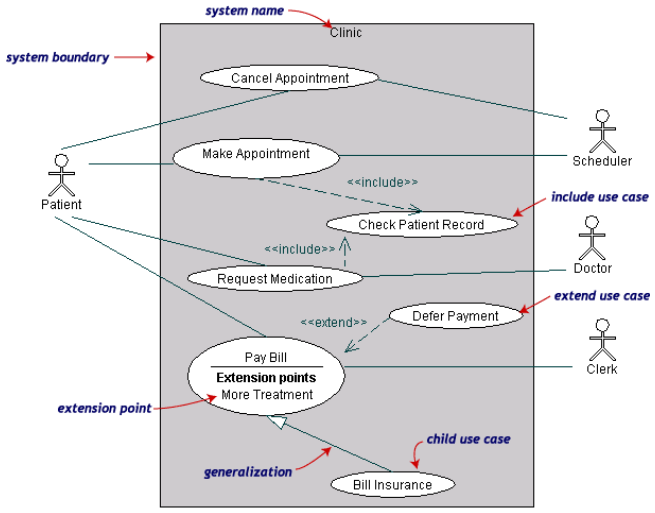
Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu.

Use case diagram dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem.

Sebuah *use case* dapat menggunakan relasi *include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal. Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common.*

Sebuah *use case* juga dapat menggunakan relasi *extend* terhadap *use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar u*se case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.

Contoh *use case diagram* pada gambar 2.16.



Gambar 2. 16 Contoh *Use Case* Diagram

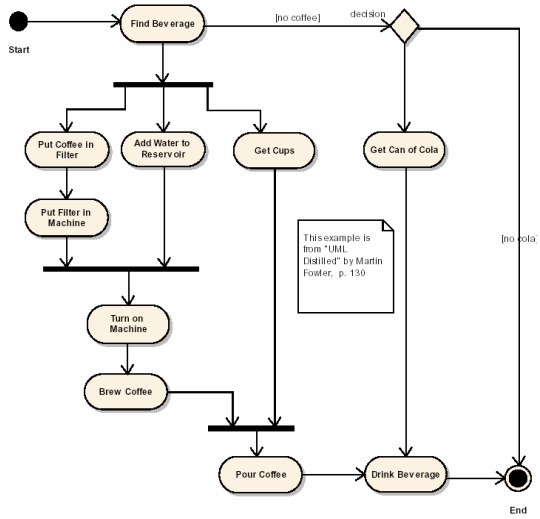
2. *Activity Diagram*

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

Activity diagram merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya *(internal processing).* Oleh karena itu *activity diagram* tidak menggambarkan *behaviour internal* sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum. Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas. Sama seperti *state*, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan *behaviour* pada kondisi tertentu. Untuk

mengilustrasikan proses-proses paralel *(fork dan join)* digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal.

Activity diagram dapat dibagi menjadi beberapa object *swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu. Contoh *activity diagram* tanpa *swimlane* pada gambar 2.17.



Gambar 2. 17 Contoh *Activity* Diagram Tanpa *Swimlane*

3. *Sequence Diagram*

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang men-trigger aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan.

Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek

lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi operasi/metoda dari *class. Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message*.

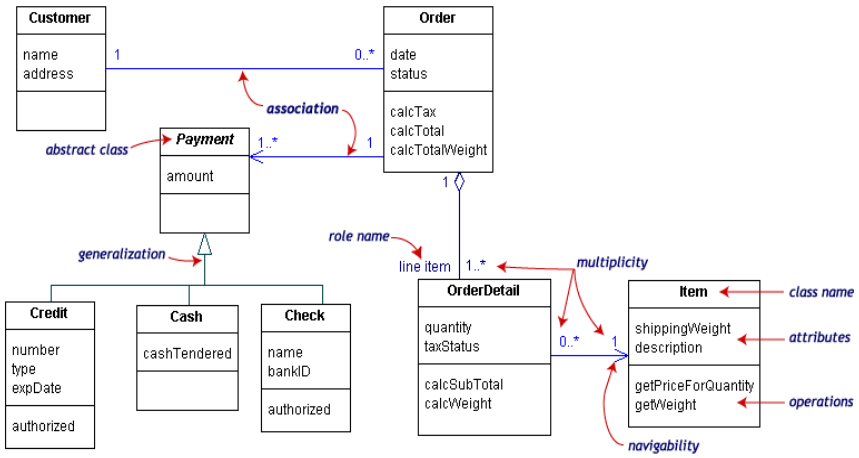
Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan *icon* khusus untuk objek *boundary, controller* dan *persistent entity*.

4. *Class Diagram*

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). *Class diagram* menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain.

Class memiliki tiga area pokok, yaitu: Nama (dan stereotype), Atribut dan Metoda. Atribut dan metoda dapat memiliki salah satu sifat berikut :

- *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan.
- *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya.
- *Public*, dapat dipanggil oleh siapa saja.

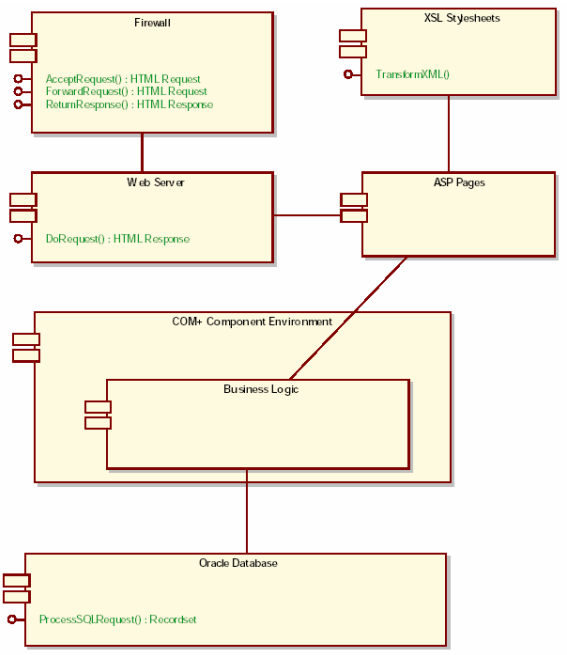


Gambar 2. 18 Contoh *Class* Diagram

5. *Component Diagram*

Component diagram menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan *(dependency)* di antaranya. Komponen piranti lunak adalah modul berisi *code*, baik berisi *source code* maupun *binary code*, baik *library* maupun *executable*, baik yang muncul pada *compile time*, *link time*, maupun *run time*. Umumnya komponen terbentuk dari beberapa *class* dan/atau *package*, tapi dapat juga dari komponen-komponen yang lebih kecil. Komponen dapat juga berupa *interface*, yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain.

Contoh *component diagram* pada gambar 2.19.



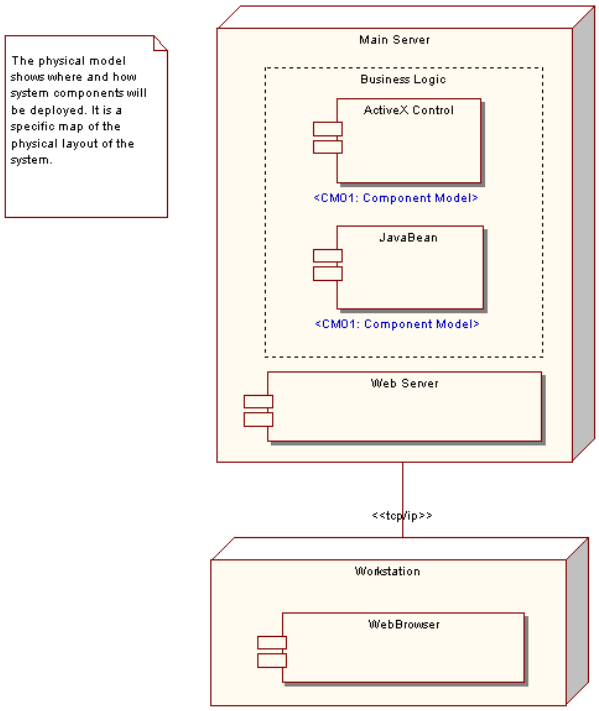
Gambar 2. 19 Contoh *Component* Diagram

6. *Deployment Diagram*

Deployment/physical diagram menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal.

Sebuah *node* adalah server, *workstation*, atau piranti keras lain yang digunakan untuk men-*deploy* komponen dalam lingkungan sebenarnya. Hubungan antar *node* (misalnya TCP/IP) dan *requirement* dapat juga didefinisikan dalam diagram ini.

Contoh *deployment diagram* pada gambar 2.20.



Gambar 2. 20 Contoh *Deployment* Diagram

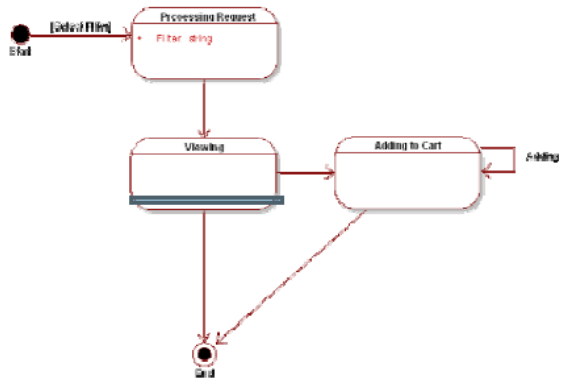
7. *State Diagram*

Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari *stimuli* yang diterima. Pada umumnya *statechart diagram* menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart diagram*).

Dalam UML, *state* digambarkan berbentuk segiempat dengan sudut membulat dan memiliki nama sesuai kondisinya saat itu. Transisi antar *state* umumnya memiliki kondisi *guard* yang merupakan syarat terjadinya transisi yang bersangkutan, dituliskan dalam kurung siku. *Action* yang dilakukan sebagai akibat dari *event* tertentu dituliskan dengan diawali garis miring.

Titik awal dan akhir digambarkan berbentuk lingkaran berwarna penuh dan berwarna setengah.

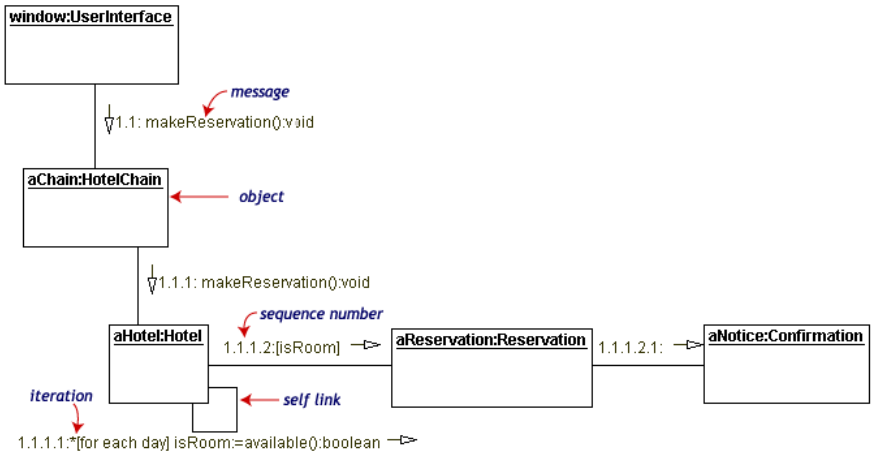
Contoh s*tatechart diagram* pada gambar 2.21.



Gambar 2. 21 Contoh *Statechart* Diagram

8. *Collaboration Diagram*

Collaboration diagram juga menggambarkan interaksi antar objek seperti *sequence diagram*, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian message. Setiap message memiliki *sequence number*, di mana *message* dari level tertinggi memiliki nomor 1. *Messages* dari level yang sama memiliki prefiks yang sama.



Gambar 2. 22 Contoh *Collaboration* Diagram

2.8 HTML5

HTML5 adalah sebuah bahasa markah untuk menstrukturkan dan menampilkan isi dari Waring Wera Wanua, sebuah teknologi inti dari Internet. HTML5 adalah revisi kelima dari HTML (yang pertama kali diciptakan pada tahun 1990 dan versi keempatnya, HTML4, pada tahun 1997) dan hingga bulan

Juni 2011 masih dalam pengembangan. Tujuan utama pengembangan HTML5 adalah untuk memperbaiki teknologi HTML agar mendukung teknologi multimedia terbaru, mudah dibaca oleh manusia dan juga mudah dimengerti oleh mesin.

HTML5 merupakan salah satu karya Konsortium Waring Wera Wanua (World Wide Web Consortium, W3C) untuk mendefinisikan sebuah bahasa markah tunggal yang dapat ditulis dengan cara HTML ataupun XHTML. HTML5 merupakan jawaban atas pengembangan HTML 4.01 dan XHTML 1.1 yang selama ini berjalan terpisah, dan diimplementasikan secara berbeda-beda oleh banyak perangkat lunak pembuat web.

2.9 Dreamweaver 8

Adobe Dreamweaver merupakan program penyunting halaman web keluaran Adobe Systems yang dulu dikenal sebagai Macromedia Dreamweaver keluaran Macromedia. Program ini banyak digunakan oleh pengembang web karena fitur-fiturnya yang menarik dan kemudahan penggunaannya. Versi terakhir Macromedia Dreamweaver sebelum Macromedia dibeli oleh Adobe Systems yaitu versi 8. Versi terakhir Dreamweaver keluaran Adobe Systems adalah Adobe Creative Suite 8 (sering disingkat Adobe CS8).