BAB 2
LANDASAN TEORI

2.1 Kecerdasan Buatan

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

Menurut Alan Turing, ahli matematika berkebangsaan Inggris yang dijuluki bapak komputer modern dan pembongkar sandi Nazi dalam era Perang Dunia II 1950, menetapkan definisi *Artificial Intelligence*: "Jika komputer tidak dapat dibedakan dengan manusia saat berbicara melalui terminal komputer, maka bisa dikatakan komputer itu cerdas, mempunyai kecerdasan".

Menurut John Carthy dari stanford mendefinisikan kecerdasan sebagai " kemampuan untuk mencapai sukses dalam menyelesaikan suatu permasalahan".

Menurut Herbert Alexander Simon (June 15, 1916-February 9, 2001): " Kecerdasan buatan (artificial intelligence) merupakan kawasan penelitian, aplikasi, dan instruksi yang terkait dengan pemrograman komputer untuk melakukan sesuatu hal yang dalam pandangan manusia adalah cerdas"

Menurut Rich and Knight (1991): " Kecerdasan buatan (AI) merupakan sebuah studi tentang bagaimana membuat komputer melakukan hal-hal yang pada saat ini dapat dilakukan lebih baik oleh manusia"

Dari sini dapat dikatakan bahwa :

Cerdas adalah memiliki pengetahuan, pengalaman, dan penalaran untuk membuat keputusan dan mengambil tindakan. Jadi, agar mesin bisa cerdas (bertindak seperti manusia) maka harus diberi bekal pengetahuan dan diberi kemampuan untuk menalar.[2].

2.1.1 Lingkup utama kecerdasan buatan

Lingkup utama kecerdasan buatan diantaranya : [2]

1. Sistem pakar. Komputer digunakan sebagai saran untuk menyimpan pengetahuan para pakar. Dengan demikian komputer akan memiliki keahlian untuk menyelesaikan masalah dengan meniru keahlian yang dimiliki para pakar
2. Pengolahan bahasa alami. Dengan pengolahan bahasa alami ini diharapkan user mampu berkomunikasi dengan komputer dengan menggunakan bahasa sehari-hari.
3. Pengenalan ucapan. Melalui pengenalan ucapan diharapkan manusia mampu berkomunikasi dengan komputer dengan menggunakan suara.
4. Robotika dan Sistem sensor
5. *Computer vision*, mencoba untuk dapat mengintrepetasikan gambar atau objek-objek tampak melalui komputer
6. *Intelligent Computer aid Instruction*. Komputer dapat digunakan sebagai tutor yang dapat melatih dan mengajar

2.1.2 Keuntungan Kecerdasan Buatan

Keuntungan kecerdasan buatan yaitu : [3]

1. Kecerdasan buatan lebih bersifat permanen. Kecerdasan alami akan cepat mengalami perubahan. Hal ini dimungkinkan karena sifat manusia yang pelupa. Kecerdasan buatan tidak akan berubah sepanjang sistem komputer dan program tidak mengubahnya.
2. Kecerdasan buatan lebih mudah diduplikasi dan disebarkan. Mentransfer pengetahuan manusia dari satu orang ke orang lain butuh proses dan waktu lama. Disamping itu suatu keahlian tidak akan pernah bisa diduplikasi secara lengkap. Sedangkan jika pengetahuan terletak pada suatu sistem komputer, pengetahuan tersebuat dapat ditransfer atau disalin dengan mudah dan cepat dari satu komputer ke komputer lain
3. Kecerdasan buatan lebih murah dibanding dengan kecerdasan alami. Menyediakan layanan komputer akan lebih mudah dan lebih murah

dibanding dengan harus mendatangkan seseorang untuk mengerjakan sejumlah pekerjaan dalam jangka waktu yang sangat lama.

4. Kecerdasan buatan bersifat konsisten. Hal ini disebabkan karena kecerdasan buatan adalah bagian dari teknologi komputer. Sedangkan kecerdasan alami senantiasa berubah-ubah.
5. Kecerdasan buatan dapat didokumentasikan. Keputusan yang dibuat komputer dapat didokumentasikan dengan mudah dengan melacak setiap aktivitas dari sistem tersebut. Kecerdasan alami sangat sulit untuk direproduksi.
6. Kecerdasan buatan dapat mengerjakan pekerjaan lebih cepat dibanding dengan kecerdasan alami
7. Kecerdasan buatan dapat mengerjakan pekerjaan lebih baik dibanding dengan kecerdasan alami.

2.1.3 Implementasi kecerdasan buatan dapat diterapkan pada bidang

Implementasi kecerdasan buatan dapat diterapkan pada beberapa bidang sebagai berikut :

a. Visualisasi komputer

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

b. Pengenalan Suara

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

c. Sistem Pakar

   Kecerdasan buatan pada Sistem Pakar ini memungkinkan sebuah sistem komputer memiliki cara berpikir dan penalaran seorang ahli dalam mengambil keputusan, untuk memecahkan masalah yang ada pada saat itu.

Contohnya program komputer yang dapat mendiagnosa penyakit dengan memasukan gejala-gejala yang dialami pasien.

d. Permainan

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

2.1.4 Kecerdasan Buatan Pada *Game*

Kecerdasan buatan atau AI merupakan kegiatan membuat komputer agar dapat berpikir dan mengerjakan kegiatan yang dapat dilakukan oleh manusia maupun binatang.

Saat ini dapat ditemukan program komputer yang memiliki kemampuan menangani masalah seperti aritmatik, sorting, searching. Bahkan komputer juga dapat bermain beberapa board *game* seperti catur dan reversi lebih baik daripada manusia.

Namun, masih banyak hal yang tidak dapat dilakukan dengan baik oleh komputer. Seperti, mengenali wajah, berbicara bahasa manusia, menentukan sendiri apa yang harus dilakukan, dan bertingkah kreatif. Hal itu semua merupakan domain dari AI untuk mencoba menentukan algoritma apa yang dibutuhkan untuk memenuhi kebutuhan diatas.

Dalam bidang akademik, beberapa peniliti AI termotivasi oleh filosofi, yaitu memahami alam pikiran dan alam kecerdasan dan membangun program untuk memodelkan bagaimana proses berpikir. Beberapa juga termotivasi oleh *psychology*, bertujuan untuk memahami mekanisme otak manusia dan proses mental. Dan lainya termotivasi oleh engineering, dengan tujuan membangun algoritma untuk melakukan kegiatan seperti manusia atau hewan.

Dalam pembangunan *game*, umumnya akan cenderung hanya pada sisi engineering yang bertujuan membangun algoritma yang dapat membuat *game* karakter mengerjakan kegiatan seperti yang dilakukan manusia atau binatang. [2].

2.2 Klasifikasi *Game*

Game biasanya digunakan untuk mengkatagorikan sebuah *game* berdasarkan dari interaksi *game*play daripada perbedaan tampilan atau narasi. Sebuah jenis *game* dapat didefinisikan dengan kumpulan dari sebuah tipe permainan dari *game* tersebut. *Game* dapat diklasifikasikan menjadi dua, yaitu : [4]

2.2.1 Berdasarkan Jenis Platform yang Digunakan

Tiga kategori utama platform pada *game* adalah personal computer (PC), console, dan mobile : [4]

1. Personal Computer (PC)

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

2. Console

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

3. Mobile

   Mobile platform terdiri atas sesuatu yang portable dan bisa digenggam, termasuk ponsel, PDA, ipod, dan handle *game* seperti Nintendo Dsi atau

Sony PSP. *Game* monile memiliki kontrol yang sederhana (terutama jika dibandingkan dengan PC)

2.2.2 Berdasarkan Genre Permainan

Genre *game* bisa dibagi menjadi 9, yaitu :

1. *Action*

   Action game adalah *game* dimana kebanyakan dari tantangan yang disajikan merupakan dari tes *physical skill* dan koordinasi pemain. Salah satu sub-genre action *game* adalah *shooters game*, baik yang 2D maupun 3D seperti *First Person, Shooters* (FPS)

2. *Strategy*

   Strategy game menantang pemain untuk mencapai kemenangan dengan perencanaan, khususnya melalui perencanaan serangkaian tindakan yang dilakukan melawan satu lawan atau lebih. Kemenangan diraih dengan perencanaan matang dan pengambilan keputusan yang optimal.

3. *Role Playing Game* (RPG)

   RPG adalah *game* dimana pemain mengontrol satu atau lebih karakter yang biasanya di desain oleh pemain itu sendiri, dan memandu mereka melewati berbagai rintangan yang diatur oleh komputer. Perkembangan karakter dalam hal kekuatan dan kemampuannya adalah kunci dari *game* jenis ini.

4. *Sports*

   Sports game mensimulasikan berbagai aspek dari olahraga atletik nyata maupun imajiner, apakah itu memainkan pertandingan, me-*manage* tim dan karir atau keduanya. Salah satu contoh *game* jenis ini adalah *Pro Evolution Soccer 2012* (PES 2012), dimana pemain bisa memainkan pertandingan, menjadi manajer tim, maupun menjadi pemain dan mengembangkan karirnya sendiri

5. *Vehicle Simulation*

   Vechicle Simulation membuat feeling mengedarai kendaraan, baik kendaraan nyata maupun kendaraan imajiner. Performa dan karakteristik kasar mesin

harus menyerupai kenyataan, kecuali jika yang didesain adalah kendaraan imajiner

6. Construction and Management Simulation

   CMS *game* adalah *game* tentang proses. Tujuan pemain bukan untuk mengalahkan musuh, tetapi membangun sesuatu dengan konteks proses yang sedang berjalan. Semakin pemain mengerti dan mengontrol proses, semakin sukses sesuatu yang ia bangun. *Game* seperti ini biasanya menyediakan dua jenis permainan, yaitu mode bebas dimana pemain bebas membangun sesuatu dan mode misi dimana terdapat skenario hal apa yang harus dibangun oleh pemain.

7. Adventure

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

8. Artificial Life and Puzzle *Game*

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

9. Online *Game*

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

2.2.3 Sudut Pandang Permainan (Game View Point)

Sebuah *game* biasanya mempunyai sudut pandang permainan tersendiri disesuaikan berdasarkan *genre game* yang diambil. Berikut beberapa macam sudut pandang permainan yang biasa digunakan : [3]

1. *Side Scrolling*

   Adalah sudut pandang permainan yang terlihat dari samping dan memungkinkan karakter utama untuk bergerak dari kiri ke kanan serta memungkinkan

background pada *game* seolah-olah bergeser mengikuti pergerakan karakter utama.

2. *Top Down*

   Adalah sudut pandang permainan yang memungkinkan karakter utama bermanuver ke empat arah namun cara permainannya sendiri bergeser dari bawah ke atas, dan biasanya *game* yang menggunakan sudut pandang permainan jenis ini adalah *shooter game*.

3. *Isometric*

   Adalah sudut pandang permainan yang memungkinkan permainan terlihat diantara sisi *Side Scrolling* dan juga *Top Down*, dan biasanya diterapkan pada *game* dengan *genre RTS* (*Real Time Strategy*).

2.2.4 Elemen Pada *Game*

Elemen – elemen yang ada dalam sebuah *game* biasanya terdiri dari:[3]

1. Desain *game*

   Desain adalah langkah awal untuk membuat semua elemen *game*. Desain *game* dibuat semenarik mungkin agar pemain tidak cepat bosan yang dampaknya membuat *game* tersebut cepat ditinggalkan.

2. Pemrograman *game*

   Pemrograman *game* sebagian besar digunakan untuk mengontrol gerakan objek di layar. Selain itu, pemrograman *game* juga digunakan untuk pemrograman suara, *input device,* deteksi keadaan lain – lain.

3. Grafis *game*

   Grafis *game* memegang peranan penting dalam pembuatan tampilan. Tampilan haruslah dibuat semenarik mungkin, sehingga dengan melihatnya saja *end user* langsung tertarik untuk memainkannya.

4. Musik dan *sound*

   Musik dan *sound* dalam suatu *game* merupakan suatu hal yang wajib dikarenakan dalam suatu *game* akan lebih terasa efek emosi dari *game* tersebut, sehingga akan membuat kesan *game* tersebut lebih terasa nyata.

2.3 *Linear Programming*

Linear programming adalah salah satu pendekatan matematika yang paling sering dipergunakan dan diterapkan dalam keputusan-keputusan manajerial. Tujuan dari penggunaan linear programming adalah untuk menyusun suatu model yang dapat dipergunakan untuk membantu pengambilan keputusan dalam menentukan alokasi yang optimal dari sumber daya ke berbagai alternatif. Penggunaan model *linear programming* dalam hal ini adalah mengalokasikan sumber daya tersebut sedemikian rupa sehingga akan maksimum atau alternatif minimum.[5]

Ada empat kondisi utama yang diperlukan dalam penerapannya :

1. Harus adanya sumber daya yang terbatas
2. Ada suatu fungsi tujuan seperti memaksimalkan atau meminimalkan
3. Harus ada linearitas
4. Harus ada keseragaman

2.3.1 Karakteristik *Linear Programming*

Sifat linearitas suatu kasus dapat ditentukan dengan menggunakan beberapa cara. Secara statistik, dapat memeriksa kelinearan menggunakan grafik (diagram pencar) ataupun menggunakan uji hipotesa. Secara teknis, linearitas ditunjukkan oleh adanya sifat proporsionalitas, additivitas, divisibilitas dan kepastian fungsi tujuan dan pembatas.

Sifat proporsional dipenuhi jika kontribusi setiap variabel pada fungsi tujuan atau penggunaan sumber daya yang membatasi proporsional terhadap level nilai variabel. Jika harga per unit produk misalnya adalah sama berapapun jumlah yang dibeli, maka sifat proporsional dipenuhi. Atau dengan kata lain, jika pembelian dalam jumlah besar mendapatkan diskon, maka sifat proporsional tidak dipenuhi. Jika penggunaan sumber daya per unitnya tergantung dari jumlah yang diproduksi, maka sifat proporsionalitas tidak dipenuhi.

Sifat additivitas mengasumsikan bahwa tidak ada bentuk perkalian silang diantara berbagai aktivitas, sehingga tidak akan ditemukan bentuk perkalian silang pada model. Sifat additivitas berlaku baik bagi fungsi tujuan maupun pembatas

(kendala). Sifat additivitas dipenuhi jika fungsi tujuan merupakan penambahan langsung kontribusi masing-masing variabel keputusan. Untuk fungsi kendala, sifat additivitas dipenuhi jika nilai kanan merupakan total penggunaaan masing-masing variabel keputusan. Jika dua variabel keputusan misalnya merepresentasikan dua produk substitusi, dimana peningkatan volume penjualan salah satu produk akan mengurangi volume penjualan produk lainnya dalam pasar yang sama, maka sifat additivitas tidak terpenuhi. Sifat divisibilitas berarti unit aktivitas dapat dibagi ke dalam sembarang level fraksional, sehingga nilai variabel keputusan non integer dimungkinkan.

Sifat kepastian menunjukkan bahwa semua parameter model berupa konstanta. Artinya koefisien fungsi tujuan maupun fungsi pembatas merupakan suatu nilai pasti, bukan merupakan nilai dengan peluang tertentu. Keempat asumsi (sifat) ini dalam dunia nyata tidak selalu dapat dipenuhi. Untuk meyakinkan dipenuhinya keempat asumsi ini, dalam pemrograman linier diperlukan analisis sensitivitas terhadap solusi optimal yang diperoleh.

2.3.2 Formulasi Permasalahan

Urutan pertama dalam penyelesaian adalah mempelajari sistem relevan dan mengembangkan pernyataan permasalahan yang dipertimbangkan dengan jelas. Penggambaran sistem dalam pernyataan ini termasuk pernyataan :

1. Tujuan sumber daya yang membatasi
2. Alternatif keputusan yang mungkin (kegiatan atau aktivitas)
3. batasan waktu pengambilan keputusan
4. Hubungan antara bagian yang dipelajari dan bagian lain dalam perusahaan, dan lain-lain.

Penetapan tujuan yang tepat merupakan aspek yang sangat penting dalam formulasi masalah. Untuk membentuk tujuan optimalisasi, diperlukan identifikasi anggota manajemen yang benar-benar akan melakukan pengambilan keputusan dan mendiskusikan pemikiran mereka tentang tujuan yang ingin dicapai.

2.3.3 Pembentukan Model Matematika

Tahap berikutnya yang harus dilakukan setelah memahami permasalahan optimasi adalah membuat model yang sesuai untuk analisis. Pendekatan konvensional riset operasional untuk pemodelan adalah membangun model matematik yang menggambarkan inti permasalahan. Kasus dari bentuk cerita diterjemahkan ke model matematik. Model matematik merupakan representasi kuantitatif tujuan dan sumber daya yang membatasi sebagai fungsi variabel keputusan. Model matematika permasalahan optimal terdiri dari dua bagian. Bagian pertama memodelkan tujuan optimasi. Model matematika tujuan selalu menggunakan bentuk persamaan. Bentuk persamaan digunakan karena ingin mendapatkan solusi optimum pada satu titik. Fungsi tujuan yang akan dioptimalkan hanya satu. Bukan berarti bahwa permasalahan optimasi hanya dihadapkan pada satu tujuan. Tujuan dari suatu usaha bisa lebih dari satu. Tetapi pada bagian ini kita hanya akan tertarik dengan permasalahan optimal dengan satu tujuan.

Bagian kedua merupakan model matematik yang merepresentasikan sumber daya yang membatasi. Fungsi pembatas bisa berbentuk persamaan (=) atau pertidaksamaan ($\leq$ atau $\geq$). Fungsi pembatas disebut juga sebagai konstrain. Konstanta (baik sebagai koefisien maupun nilai kanan) dalam fungsi pembatas maupun pada tujuan dikatakan sebagai parameter model. Model matematika mempunyai beberapa keuntungan dibandingakan pendeskripsian permasalahan secara verbal. Salah satu keuntungan yang paling jelas adala model matematik menggambarkan permasalahan secara lebih ringkas. Hal ini cenderung membuat struktur keseluruhan permasalahan lebih mudah dipahami, dan membantu mengungkapkan relasi sebab akibat penting. Model matematik juga memfasilitasi yang berhubungan dengan permasalahan dan keseluruhannya dan mempertimbangkan semua keterhubungannya secara simultan. Terakhir, model matematik membentuk jembatan ke penggunaan teknik matematik dan komputer kemampuan tinggi untuk menganalisis permasalahan.

2.3.4 Bentuk Umum *Linear Programming*

Fungsi tujuan :

Maksimumkan atau minimumkan

$$z = c_1 x_1 + c_2 x_1 + \cdots + c_n x_n$$

Sumber daya yang membatasi :

$$a_{11} x_1 + a_{12} x_2 + \cdots + a_{1n} x_n = /\leq/\geq b_1$$

$$a_{21} x_1 + a_{22} x_2 + \cdots + a_{2n} x_n = /\leq/\geq b_2$$

$$\ldots$$

$$a_{m1} x_1 + a_{m2} x_2 + \cdots + a_{mn} x_n = /\leq/\geq b_m$$

$$x_1, x_2, \ldots, x_n \geq 0$$

Simbol x_1, x_2, ..., x_n (x_i) menunjukkan variabel keputusan. Jumlah variabel keputusan (x_i) oleh karenanya tergantung dari jumlah kegiatan atau aktivitas yang dilakukan untuk mencapai tujuan. Simbol c_1, c_2,..., c_n merupakan kontribusi masing-masing variabel keputusan terhadap tujuan, disebut juga koefisien fungsi tujuan pada model matematiknya. Simbol a_{11}, ..., a_{1n},..., a_{mn} merupakan penggunaan per unit variabel keputusan akan sumber daya yang membatasi, atau disebut juga sebagai koefisien fungsi kendala pada model matematikanya.

Simbol b_1, b_2,...,bm menunjukkan jumlah masing-masing sumber daya yang ada. Jumlah fungsi kendala akan tergantung dari banyaknya sumber daya yang terbatas.

Pertidaksamaan terakhir ($x_1, x_2, \ldots, x_n \geq 0$) menunjukkan batasan non negatif. Membuat model matematik dari suatu permasalahan bukan hanya

menuntut kemampuan matematik tapi juga menuntut seni permodelan. Menggunakan seni akan membuat permodelan lebih mudah dan menarik.

Kasus pemrograman linier sangat beragam. Dalam setiap kasus, hal yang penting adalah memahami setiap kasus dan memahami konsep permodelannya. Meskipun fungsi tujuan misalnya hanya mempunyai kemungkinan bentuk maksimisasi atau minimisasi, keputusan untuk memilih salah satunya bukan pekerjaan mudah. Tujuan pada suatu kasus bisa menjadi batasan pada kasus yang lain. Harus hati-hati dal am menentukan tujuan, koefisien fungsi tujuan, batasan dan koefisien pada fungsi pembatas.

2.3.5 Penerapan Model *Linear Programming* Dalam Bidang-Bidang Fungsional

Penerapan Model Linear Programming terdiri dari :

1. Penerapan dalam transportasi dan masalah.
   Masalah transportasi timbul karena transportasi produk yang bersifat homogen dari berbagai sumber (supply) ke berbagai tempat penjualan (demand). Alokasi produk dari daerah supply ke daerah demand ini dilakukan dengan tujuan memaksimalkan laba dari penjualan produk atau alternatifnya adalah menimalkan biaya transportasi produk.
2. Penerapan dalam bidang keuangan
   Dalam analisis investasi, tujuan yang hendak dicapai biasanya ialah expected returns. Untuk memaksimalkannya sering dihadapkan dengan berbagai masalah, seperti dana yang tersedia.
3. Penerapan dalam bidang produksi.
   Dengan bantuan pemecahan melalui linear programming ini memungkinkan manajer perusahaan untuk merancang jadwal produksi yang efisien. Manajer dapat menentukan tingkat/level produksi yang memungkinkan perusahaan dapat memenuhi permintaan pasar, subyek kepada keterbatasan dari kapasitas produksi, kapasitas gudang, dan tenaga kerja.
4. Penerapan dalam bidang pemasaran

Diantaranya ialah masalah alokasi dengan anggaran biaya advertensi, pemilihan media promosi, serta manajemen retailer dan salesman.

5. Penerapan dalam bidang personalia.
   Diantaranya ialah untuk merencanakan jumlah jam kerja pegawai, menentukan jumlah petugas yang akan melaksanakan pekerjaan, dan masalah-masalah personalia lainnya.
6. Penerapannya dalam bidang perencanaan fasilitas.
   Berkaitan dengan masalah penentuan lokasi gudang, jumlah fasilitas, dan sistem logistik, serta masalah transportasi dan penentuan lokasi pabrik.

2.4 Elemen Pada Game RPG

Pada game Role Playing Game (RPG) memiliki beberapa elemen yang penting karena elemen tersebut merupakan yang membedakan game Role Playing Game (RPG) dengan game yang lainnya. Dan elemen-elemen tersebut adalah sebagai berikut :

2.4.1 Pengembangan Alur Cerita Dan Karakter Pada RPG

Pengembangan jalan cerita dan karakter merupakan unsur utama dalam Role Playing Game (RPG). Cerita yang ada dalam Role Playing Game (RPG) tidak harus selalu aneh, tetapi masing-masing Role Playing Game (RPG) harus memiliki cirri khas sendiri dan biasanya cerita dalam Role Playing Game (RPG) cukup panjang.Cerita dalam Role Playing Game (RPG) harus kreatif, inovatif dan megnalir seiring dengan perkembangan karakter utama. Biasanya cerita dalam Role Playing Game (RPG) mengandung unsure-unsur khusus yang membentuk sebuah game Role Playing Game (RPG) seperti berikut :

1. Karakter Utama
   Biasanya sang pahlawan adalah seorang pemuda, tetapi tidak jarang juga sang pahlawan itu seorang gadis. Misalnya Lighting pada Final Fantasy XIII, Putri Yggrda Union : We'll Never Fight Alone. Ada sesuatu yang membuatnya menjadi pahlawan, apakah pahlawan tersebut selamatdari suatu pembantaian massal, atau

pahlawan tersebut telah di pilih sejak lahir untuk menyelamatkan dunia atau dengan alasan yang lain.

2. Status *Effect* Pada RPG

Banyak sekali jenis-jenis status yang berada pada game Role Playing Game (RPG). Sekalipun namanya berbeda, mungkin saja efeknya sama, beberapa game Role Playing Game (RPG) biasanya menghapus status setelah pertarungan selesai. Ada yang membiarkan status hilang setelah beberapa saat, ada juga yang membiarkannya sampai di sembuhkan. Berikut status-status yang biasanya muncul pada game Role Playing Game (RPG) yaitu sebagai berikut :

a. *Poison* atau *Toxic*

Ini adalah pengurangan Human Power(HP) secara berkala yang terjadi pada karakter pemain. Ada yang terjadi saat pertarungan, ada yang terjadi saat pemain yang terkena berjalan.Biasanya status ini tidak bisa membunuh karakter pemain. Lawan dari status ini adalah Regen atau Healingyang justru menambah HP secara berkala.

b. Silence

Status ini menyebabkan karakter pemain yang bersangkutan tidak bisa mengeluarkan kemampuannya sampai efek dari status ini menghilang.

c. Death

Banyak kontroversi tentang status ini, mengingat pahlawan tidak benar-benar mati kalau HP nol. Karakter pemain yang terkena status ini tidak dapat melanjutkan pertarungan sampai dibangkitkan lagi.

3. Menu Sistem RPG

Menu sistem pada Role Playing Game (RPG) adalah suatu cirri khas dari
sebuah game Role Playing Game (RPG) Sistem ini merupakan tempat di mana pemain dapat melihat dan mengatur karakter, kemampuan, item, equipment karkater dan masih banyak lagi hal lainnya.

Bagian ini perlu di buat variatif dan tidak membosankan, sebab bagian ini merupakan bagian yang sering dilakukan oleh pemain di game Role Playing Game (RPG). Biasanya sistem menu berisikan petunjuk dan beberapa manual yang ada pada game Role Playing Game (RPG) tersebut. Sistem menu ini sangat membantu

pemain dalam memahami dan mengatur karakternya sehingga membuat sang karakter sesuai dengan keinginan sang pemain.

2.5 *Object Oriented Programing*

Secara spesifik, pengertian berorientasi objek berarti bahwa mengorganisasi perangkat lunak sebagai kumpulan dari objek tertentu yang memiliki struktur data dan perilakunya. Hal ini yang membedakan dengan pemograman konvensional dimana struktur data dan perilaku hanya berhubungan secara terpisah. Terdapat beberapa cara untuk menentukan karakteristik dalam pendekatan berorientasi objek, tetapi secara umum mencakup empat hal, yaitu identifikasi, klasifikasi, *polymorphism* (polimorfisme) dan *inheritance* (pewarisan).[6]

2.5.1 Konsep Dasar Berorientasi Objek

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

Berikut ini adalah beberapa konsep dasar yang harus dipahami tentang metodologi berorientasi objek :[6]

1. Kelas (*class*)

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

2. Objek (*object*)

   Objek adalah abstraksi dan sesuatu yang mewakilkan dunia nyata seperti benda, manusia, satuan organisasi, tempat, kejadian, struktur, status atau hal-hal lain yang bersifat abstrak. Objek merupakan entitas yang mampu menyimpan informasi

(status) dan mempunyai operasi (kelakukan) yang dapat diterapkan atau dapat berpengaruh pada status objeknya. Objek mempunyai siklus hidup yang diciptakan, dimanipulasi, dan dihancurkan.

3. Metode (*method*)

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

4. Atribut (*attribute*)

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

5. Abstraksi (*absraction*)

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

6. Enkapsulasi (*encapsulation*)

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

7. Pewarisan (*inheritance*)

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

8. Antarmuka (*interface*)

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

9. *Reusability*

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

10. Generalisasi dan Spesialisasi

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

11. Komunikasi Antar Objek

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

12. Polimorfisme (*polymorphism*)

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

13. *Package*

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

2.5.2 Analisis Berorientasi Objek

Analisis berorientasi objek atau *Object Oriented Analysis* (OOA) adalah tahap untuk menganalisis spesifikasi atau kebutuhan akan sistem yang akan dibangun dengan konsep berorientasi objek, apakah benar kebutuhan yang ada dapat diimplementasikan menjadi sebuah sistem berorientasi objek.

OOA biasanya menggunakan kartu CRC (*Component*, *Responsibility*, *Collaborator*) untuk membangun kelas-kelas yang akan digunakan atau menggunakan UML (*Unifed Modeling Language*) pada bagian diagram use case, diagram kelas, dan diagram objek.[6]

2.5.3 Desain Berorientasi Objek

Desain berorientasi objek atau Object Oriented Design (OOD) adalah tahapan perantara untuk spesifikasi atau kebutuhan sistem yang akan dibangun dengan konsep berorientasi objek ke desain pemodelan agar lebih mudah diimplementasikan dengan pemograman berorientasikan objek.

Pemodelan berorientasi objek biasanya dituangkan dalam dokumentasi perangkat lunak dengan menggunakan perangkat lunak dengan menggunakan perangkat pemodelan berorientasi objek, diantaranya adalah UML (Unified Modeling Language). Kendala dan permasalahan pembangunan sistem berorientasi objek biasanya dapat dikenali dalam tahap ini.6]

2.6 UML (*Unified Modelling Language*)

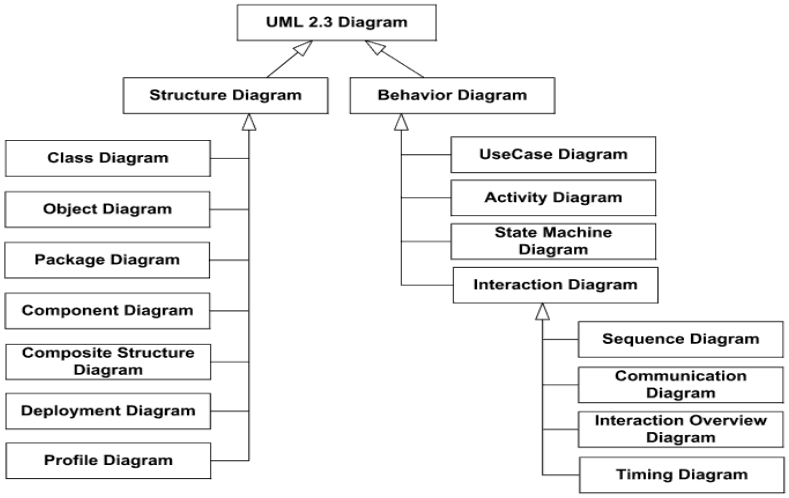
Unified Modelling Language (UML) adalah sebuah "bahasa" yang telah menjadi standar dalam industri visualisasi, merancang, dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

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

Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan syntax. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentum, dan UML syntax mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya yaitu : Grady Booch OOD (*Object Oriented Design*), Jim Rumbaugh OMT (*Object Modeling Technique*) dan Ivar Jacobson OOSE (*Object Oriented Software Engineering*).[6]

2.6.1 Diagram UML

Pada UML 2.3 terdiri dari 13 macam diagram yang dikelompokan dalam 3 kategori. Pembagian kategori dan macam-macam diagram tersebut dapat dilihat pada gambar 2.1



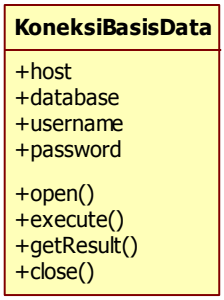
Gambar 2. 1 Diagram UML [6]

Berikut ini penjelasan singkat dari pembagian kategori tersebut.[6]

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

2.6.2 Class Diagram

Use case atau diagram *use case* merupakan pemodelan untuk kelakuan (*behavior*) sistem yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi itu [4]. Contoh *Class* Diagram dapat dilihat pada Gambar 2.2 [6]



Gambar 2.2 Contoh *Class* Diagram [6]

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.

a. 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.
b. Use case merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.[6]

2.6.3 Diagram *Use Case*

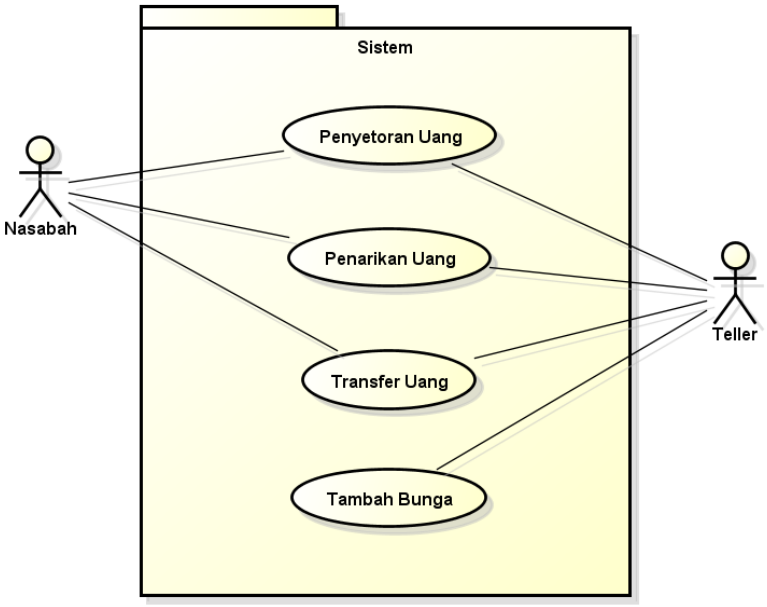
Diagram *use case* merupakan pemodelan untuk kelakuan *(behaviour)* sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu. Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut aktor dan *use case*.[6]

1. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat

itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.

2. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

Contoh Use Case Diagram dapat dilihat pada Gambar 2.3



Gambar 2.3 Contoh Use Case Diagram

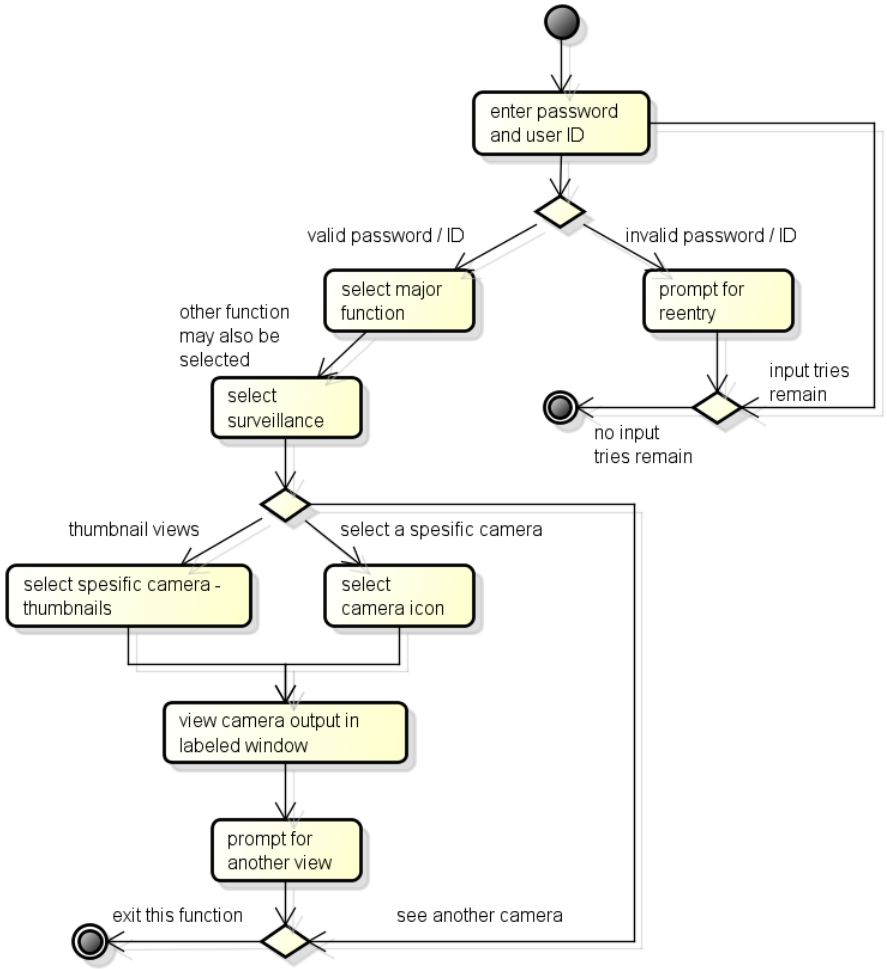
2.6.4 Diagram Aktivitas (*Activity* Diagram)

Diagram aktivitas menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem. Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut: [6]

1. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
2. Urutan atau pengelompokkan tampilan dari sistem/*user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.

3. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.
4. Rancangan menu yang ditampilkan pada perangkat lunak.

Contoh Activity Diagram dapat dilihat pada Gambar 2.4

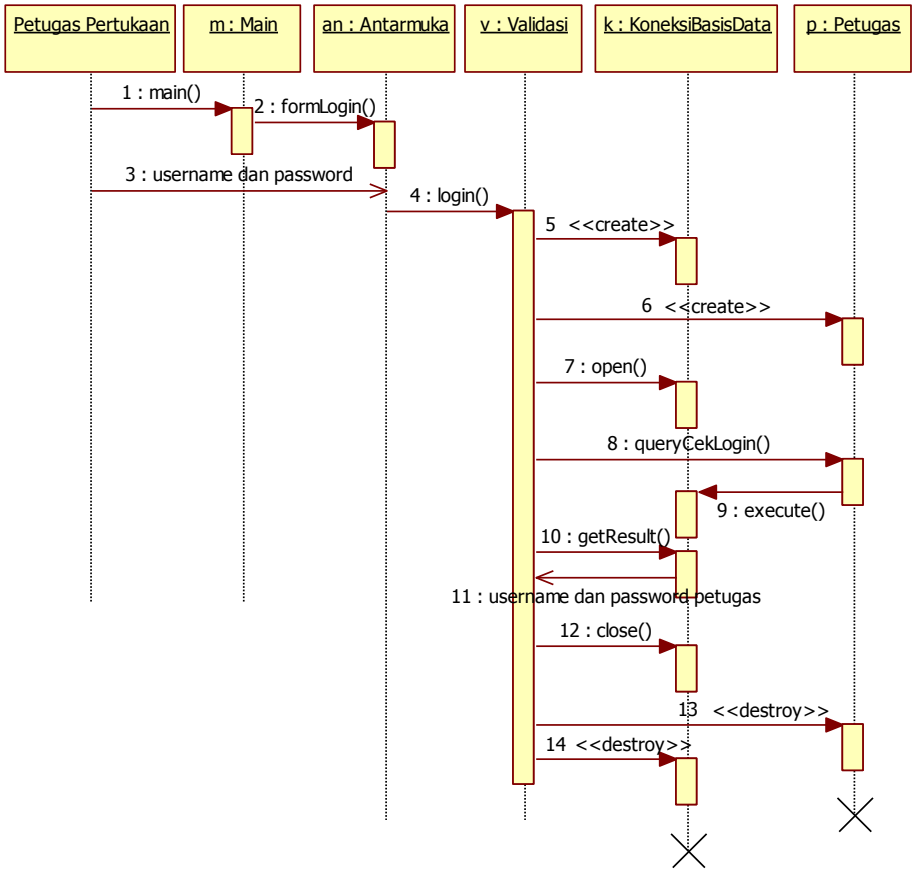


Gambar 2.4 Contoh *Activity* Diagram

2.6.5 Diagram Sekuen (*Sequence* Diagram)

Diagram sekuen menggambarkan kelakuan pada objek *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambar diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu. Membuat diagram sekuen juga dibutuhkan untuk melihat skenario yang ada pada *use case*.

Banyaknya diagram sekuen yang harus digambar adalah minimal sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak *use case* yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak. Contoh Sequence Diagram dapat dilihat pada Gambar 2.5 [6]



Gambar 2.5 Contoh *Sequence* Diagram

2.7 Unity

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

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

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

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

1. Mendukung 3 bahasa pemrograman, *JavaScript*, C#, dan *Boo*.
2. *Flexible and EasyMoving*, *rotating*, dan *scaling objects* hanya perlu sebaris kode. Begitu juga dengan *duplicating*, *removing*, dan *changing properties*.
3. *Multi Platform Game* bisa di deploy di PC, Mac, Wii, iPhone, iPad dan *browser, android*.
4. *Visual Properties Variables* yang di definisikan dengan *scripts* ditampilkan pada editor. Bisa digeser, di *drag* and *drop*, bisa memilih warna dengan *color picker*.
5. Berbasis .NET, penjalanan program dilakukan dengan *Open Source* .NET *platform, Mono*. [7]

2.8 Bahasa Pemrograman C#

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

Bahasa pemrograman C# dibuat sebagai bahasa pemrograman yang

bersifat *general-purpose* (untuk tujuan jamak), berorientasi objek, modern, dan sederhana. C# ditujukan agar cocok digunakan untuk menulis program aplikasi baik dalam sistem klien-server (*hosted system*) maupun sistem embedded *(embedded system),* mulai dari program aplikasi yang sangat besar yang menggunakan sistem operasi yang canggih hingga kepada program aplikasi yang sangat kecil.

Meskipun aplikasi C# ditujukan agar bersifat 'ekonomis' dalam hal kebutuhan pemrosesan dan memori komputer, bahasa C# tidak ditujukan untuk bersaing secara langsung dengan kinerja dan ukuran program aplikasi yang dibuat dengan menggunakan bahasa pemrograman C.

2.9 Adobe PhotoShop

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

Aplikasi adobe photoshop bekerja dengan metode pemisah setiap komponen gambar menjadi layer yang berbeda, hal ini akan sangat memudahkan kreasi dalam melakukan proses penyuntingan gambar [8].

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.