BAB 2

LANDASAN TEORI

2.1 *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.

Ada empat kondisi utama yang diperlukan dalam penerapannya[1] :

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

2.1.1 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[2].

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].

2.1.2 Formulasi Permasalahan

Urutan pertama dalam penyelesaian adalah mempelajari sistem relevan dan mengembangkan pernyataan permasalahan yang dipertimbangakan dengan jelas. Penggambaran sistem dalam pernyataan ini termasuk pernyataan tujuan, sumber daya yang membatasi, alternatif keputusan yang mungkin (kegiatan atau aktivitas), batasan waktu pengambilan keputusan, 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].

2.1.3 Pembentukan Model Matematik

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 matematik tujuan selalu menggunakan bentuk persamaan. Bentuk persamaan digunakan karena kita 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 (≤ atau ≥). 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 adalah 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].

Di sisi lain, model matematik mempunyai kelemahan. Tidak semua karakteristik sistem dapat dengan mudah dimodelkan menggunakan fungsi matematik. Meskipun dapat dimodelkan dengan fungsi matematik, kadang-kadang penyelesaiannya sulit diperoleh karena kompleksitas fungsi dan teknik yang dibutuhkan.

Bentuk Umum *Linear Programming* adalah sebagai berikut :

Fungsi tujuan :

Maksimumkan atau minimumkan

$$Z = c_1x_1 + c_2x_2 + \cdots + cnxn$$

Sumber daya yang membatasi :

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_1nxn = /\leq / \geq b_1$$

$$a_{21}x_1 + a_{22}x_2 + \cdots + a_2nxn = /\leq / \geq b_2$$

$$\cdots$$

$$am_1x_1 + am_2x_2 + \cdots + amnxn \ = / \leq / \geq b_m$$

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

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

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

Pertidaksamaan terakhir $(x_1, x_2, \ldots, xn \geq 0)$ 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 dalam menentukan tujuan, koefisien fungsi tujuan, batasan dan koefisien pada fungsi pembatas[2].

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

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 meminimalkan 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.

Pembahasan dalam penelitian ini dikaji secara deskriptif. Metode optimasi yang dipergunakan adalah metode simpleks. Metode simpleks ini merupakan salah satu dari model program linear.

2.1.5 Model Program Linear

Model Program Linear disebut juga dengan formulasi model. Model program linear digunakan untuk menunjukkan proses model yang semua masalah menyangkut usaha mencapai subjek tujuan dengan kumpulan batasan-batasan misalnya batasan-batasan sumber daya. Model program linear dari masalah-masalah ini memperlihatkan karakteristik-karakteristik umum seperti[4] :

a. Fungsi tujuan untuk dimaksimumkan dan diminimumkan
b. Kumpulan batasan-batasan
c. Variabel-variabel keputusan untuk mengukur tingkatan aktivitas
d. Semua hubungan batasan dan fungsitujuan adalah linear

Sebagian besar dari persoalan manajemen berkenaan dengan penggunaan sumber daya secara efisien atau alokasi sumber daya – sumber daya yang terbatas (tenaga kerja terampil, bahan mentah, modal) untuk mencapai tujuan yang diinginkan (desired objective). Dalam keadaan sumber daya yang terbatas harus dicapai suatu hasil yang optimum. Dengan perkataan lain bagaimana caranya agar dengan masukan (input) yang serba terbatas dapat dicapai hasil kerja yaitu keluaran (output) berupa produksi barang atau jasa yang optimum (Supranto, 2001)

Dalam model Linier Program dikenal 2 macam fungsi., yaitu fungsi tujuan *(objective function)* dan fungsi-fungsi batasan *(constraint function)*. Fungsi tujuan *(objective function)* adalah fungsi yang menggambarkan tujuan/sasaran di dalam permasalahan Linier Program yang berkaitan dengan pengaturan secara optimum sumber daya-sumber daya untuk memperoleh keuntungan maksimal atau biaya minimal. Fungsi tujuan selalu mempunyai salah satu target yaitu memaksimumkan atau meminimumkan suatu nilai. Pada umumnya nilai yang akan dioptimalkan dinyatakan sebagai Z. Fungsi kendala / batasan (constraint function) merupakan bentuk penyajian secara matematis batasan-batasan kapasitas yang tersedia yang akan dialokasikan secara optimal ke berbagai kegiatan.

Fungsi batasan juga merupakan hubungan linear dari variabel-variabel keputusan. Batasan-batasan dapat berupa keterbatasan sumber daya atau pedoman.

Adapun syarat-syarat agar suatu persoalan dapat dipecahkan dengan metode *Linier Programming* yaitu[4] :

a. Fungsi objektif harus didefinisikan secara jelas dan dinyatakan sebagai fungsi objektif yang linear. Misalnya jumlah hasil penjualan harus maksimum, jumlah biaya transport harus minimum.
b. Harus ada alternatif pemecahan untuk dipilih salah satu yang terbaik.
c. Sumber-sumber dan aktivitas mempunyai sifat dapat ditambahkan (additivity).
d. Fungsi objektif dan ketidaksamaan untuk menunjukkan adanya pembatasan harus linear.
e. Variabel keputusan harus positif, tidak boleh negatif.
f. Sumber-sumber dan aktivitas mempunyai sifat dapat dibagi (divisibility)
g. Sumber-sumber dan aktivitas mempunyai jumlah yang terbatas (finiteness)
h. Aktivitas harus proporsional terhadap sumber-sumber. Hal ini berarti ada hubungan yang linier antara aktivitas dengan sumber-sumber.
i. Model programming deterministik, artinya sumber dan aktivitas diketahui secara pasti (*single-valued expectations*).

Selain terdapat hubungan linier, model program linier juga mempunyai beberapa sifat lainnya. Terminologi linier tidak hanya berarti bahwa fungsi dalam model model digambarkan sebagai garis lurus, tetapi juga berarti bahwa hubungan memperlihatkan kemampuan yang sebanding. Dengan kata lain, tingkat perubahan atau kecondongan fungsi adalah konstan. Oleh karena itu, perubahan dari ukuran tertentu dalam nilai variabel keputusan akan menghasilkan perubahan yang relatif sama dalam nilai fungsi. Dalam program linier terminologi fungsi tujuan dan terminologi batasan adalah tambahan. Sifat lain dari model linier program adalah nilai pemecahan (dari variabel keputusan) tidak dapat dibatasi dalam nilai integer (bulat).

Untuk membuat formulasi batasan dan fungsi tujuan dilakukan lebih dari sekali, setelah pendefinisian variabel-variabel keputusan. Pendekatan yang lebih bijaksana pertama adalah menentukan fungsi tujuan (tanpa mempertimbangkan langsung batasan-batasan). Setelah itu memperhatikan setiap batasan masalah yang berhubungan dengan batasan-batasan model. Yang disarankan adalah pendekatan sistematis sehingga langkah-langkah perumusan sains manajemen dapat dilakukan satu persatu[4].

2.1.6 Metode Simpleks

Metode simpleks adalah suatu metode yang secara sistematis dimulai dari suatu pemecahan dasar yang fisibel ke pemecahan yang fisibel lainnya dan ini dilakukan berulang-ulang (dengan jumlah ulangan yang terbatas) sehingga akhirnya tercapai suatu pemecahan dasar yang optimum dan pada setiap langkah menghasilkan suatu nilai dari fungsi tujuan yang selalu lebih besar, lebih kecil, atau sama dari langkahlangkah sebelumnya. Apabila suatu masalah Linier Program hanya mengandung 2 kegiatan atau variabel- variabel keputusan saja, maka akan dapat diselesaikan dengan metode grafik. Tetapi bila melibatkan lebih dari dua kegiatan maka metode grafik tidak dapat digunakan lagi, sehingga diperlukan metode simpleks. Metode simpleks merupakan suatu cara yang lazim digunakan untuk menentukan kombinasi optimal dari tiga variabel atau lebih.

Dalam metode simpleks, model diubah kedalam bentuk suatu tabel, kemudian dilakukan beberapa langkah matematis pada tabel tersebut. Langkah-langkah matematis ini pada dasarnya merupakan replikasi proses pemindahan pemindahan dari suatu titik ekstrim ke titik ekstrim lainnya pada batas daerah solusi (solution boundary). Tidak seperti metode grafik, dimana kita dapat dengan mudah mencari titik terbaik di antara semua titik-titik solusi, metodesimpleks bergerak dari satu solusi ke solusi yang lebih baik sampai solusi yangterbaik didapat. Metode simpleks lebih efisien serta dilengkapi dengan suatu uji kriteria yang bisa memberitahukan kapan hitungan harus dihentikan dan kapan harus dilanjutkan sampai diperoleh suatu optimal solution (maksimum profit, maksimum revenue, minimum cost, dan lain sebagainya). Pada umumnya dipergunakan tabel-tabel dari tabel pertama yang memberikan pemecahan dasar permulaan yang fisibel (*initial basic feasible solution*) sampai pada pemecahan terakhir yang memberikan optimal solution. Semua informasi yang diperlukan (test kriteria, nilai variabel-variabel, nilai fungsi tujuan) akan terdapat pada setiap tabel, selain itu nilai fungsi tujuan dari suatu tabel akan lebih besar/kecil atau sama dengan tabel sebelumnya. Pada umumnya suatu persoalan linier programming bisa diklasifikasikan menjadi 3 kategori yaitu[4] :

a. Tidak ada pemecahan yang fisibel (*there is no feasible solution*)
b. Ada pemecahan optimum (*maksimum/minimum*).
c. Fungsi objektif tidak ada batasnya (*unbounded*)

Pada masa sekarang masalah-masalah Linier Programming yang melibatkan banyak variabel-variabel keputusan dapat dengan cepat dengan bantuan komputer, tetapi bila variabel keputusan yang dikandung tidak terlalu banyak, masalah tersebut dapat diselesaikan dengan suatu logaritma yang biasanya sering disebut metode simpleks tabel[4].

2.1.6.1 Langkah-langkah Metode Simpleks Tabel

Langkah 1 :

Mengubah fungsi tujuan dan batasanbatasan fungsi tujuan diubah menjadi fungsi implisit. Misalnya fungsi tujuan tersebut :

$$Z = c_1\ x_1 + c_2\ x_2 + \ldots\ c_n\ x_n \text{ diubah}$$

$$\text{menjadi } Z = c_x + c_x + \ldots\ldots\ldots c_x = 0$$

Pada bentuk standar semua batasan mempunyai tanda ≤.

Langkah 2 :

Menyusun persamaan-persamaan di dalam tabel. Setelah formulasi diubah kemudian disusun kedalam tabel dengan simbol seperti pada Tabel 2.1.

Tabel 2.1 Tabel Simpleks Dalam Bentuk Simbol

Variabel Dasar	Z	x_1	x_2	….	x_n	x_{n+1}	x_{n+2}	….	x_{n+m}	NK
Z	1	-C1	-C2	…	-C*n*	0	0	….	0	0
x_{n+1}	0	a_{11}	a_{12}	….	a_{1n}	1	0	…	0	b1
x_{n+1}	0	a_{21}	a_{12}	….	a_{2n}	0	1	…	0	b 2
.	.	.	.	….	.	.	.	…	.	.
.	.	.	.	….	.	.	.	…	.	.
.	.	.	.	….	.	.	.	…	.	.
.	.	.	.	….	.	.	.	…	.	.
x_{n+m}	0	a_{m1}	a_{m2}	….	a_{mn}	0	0	...	.	b *m*

NK adalah nilai kanan persamaan, yaitu nilai dibelakang tanda sama dengan (=).

Langkah 3 : Memilih kolom kunci

Langkah 4 : Memilih baris kunci

Baris kunci adalah baris yang merupakan dasar untuk mengubah tabel pada langkah ke 3 (tiga). Untuk itu terlebih dahulu carilah indeks tiap-tiap baris dengan cara membagi nilai-nilai pada kolom NK dengan nilai yang sebaris
pada kolom kunci.

$$\text{index} = \frac{nilai\ kolomNK}{nilai\ kolom\ kunci}$$

Pilihlah baris yang mempunyai indeks positif dengan angka terkecil. Nilai yang masuk dalam kolom kunci dan juga termasuk dalam baris kunci disebut angka kunci.

Langkah 5 : Mengubah nilai-nilai baris kunci

Langkah 6 : Mengubah nilai-nilai selain pada baris kunci

Langkah 7 : Melanjutkan perbaikan atau perubahan

2.2 Artificial Intelligence (AI)

Sebagian kalangan menerjemahkan *Artificial Intelligence* sebagai kecerdasan buatan, keecerdasan artifisial, intelejensia artifisial, atau intelejensia buatan. Artifisial sengaja tidak diterjemahkan ke dalam bahasa indonesia karena istilah tersebut sangat akrab bagi orang indonesia. Begitu juga dengan singkatan istilah tersebut yaitu AI, sudah sangat melekat di berbagai media ilmiah maupun non ilmiah.

Beberapa definisi tentang *Artificial Intelegence* (AI) yang disampaikan oleh beberapa ahli. Para ahli mendefinisikan AI secara berbeda-beda tergantung pada sudut pandang mereka masing-masing. Ada yang fokus pada logika berpikir manusia saja, tetapi ada juga yang mendefinisikan AI secara lebih luas pada tingkah laku manusia. Pada [*RUS95*], Stuart Russel dan Peter Norving mengelompokan definisi AI, yang diperoleh dari beberapa textbook berbeda kedalam empat kategori, yaitu[7]:

1. *Thinking humanly : the cognitive modeling approach*

   Pendekatan ini dilakukan dengan dua cara sebagai berikut:

   a) Melalui intropeksi : mencoba menangkap pemikiran-pemikiran kita sendiri pada saat kita berpikir. Tetapi, seseorang psikolog barat mengatakan "*how do you know that you understand?*" . Bagaimana anda tahu bahwa anda mengerti?. Karena pada saat anda menyadari pemikiran anda , ternyata pemikiran tersebut sudah lewat dan digantikan kesadaran anda. Sehingga definisi ini terkesan mengada-ada dan tidak mungkin dilakukan
   b) Melalui eksperimen-eksperimen psikologi.

2. *Acting humanly : the turing test approach*

   Pada tahun 1950, Alan Turing merancang suatu ujian bagi komputer berintelijensia untuk menguji apakah komputer tersebut mampu mengelabuhi seorang manusia yang mengintrogasinya melalui teletype (komunikasi berbasis teks jarak jauh). Jika interrogator tidak dapat membedakan yang diinterogasi adalah manusia atau komputer, maka komputer berintelijensia tersebut lolos dari Turing test. Komputer tersebut perlu memiliki kemampuan : *Natural Language Processing*, *Knowledge Representation, Automated Reasoning, Machine learning, Computer Vision, Robotics*. Turing test sengaja menghindari interaksi fisik antara interrogator dan komputer karena simulasi fisik manusia tidak memerlukan intelejensia.

3. *Thinking rationally : the laws of thought approach*

   Terdapat dua masalah dalam pendekatan ini, yaitu:

   a) Tidak mudah untuk membuat pengetahuan informal dan menyatakan pengetahuan tersebut ke dalam formal term yang diperlukan oleh notasi logika, khususnya ketika pengetahuan tersebut memiliki kepastian kurang dari 100%.
   b) Terdapat perbedaan besar antara dapat memecahkan masalah “dalam prinsip” dan memecahkannya “dalam dunia nyata”.

4. *Acting rationally : the rational agent approach*

   Membuat inferensi yang logis merupakan bagian dari rational agent. Hal ini disebabkan satu-satunya cara untuk melakukan aksi secara rasional adalah dengan menalar secara logis. Dengan menalar secara logis, maka bisa didapatkan kesimpulan bahwa aksi yang diberikan akan mencapai tujuan atau tidak. Jika mencapai tujuan, maka agent dapat melakukan aksi berdasarkan kesimpulan tersebut.

Thinking humanly dan *acting humanly* adalah dua definisi dalam arti yang sangat luas. Sampai saat ini, pemikiran manusia yang diluar rasio, yakni refleks dan intuitif (berhubungan dengan perasaan), belum dapat ditirukan sepenuhnya oleh komputer.

2.2.1 Implementasi kecerdasan buatan dapat diterapkan pada bidang

Implementasi kecerdasan buatan dapat diterapkan pada beberapa bidang sebagai berikut : [8]

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.

2.2.2 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[8].

2.2.3 Klasifikasi Game

Game biasanya digunakan untuk mengkatagorikan sebuah *game* berdasarkan dari interaksi *gameplay* daripada perbedaan tampilan atau narasi. Sebuah jenis *game* dapat didefinisikan dengan kumpulan dari sebuah tipe permainan dari *game* tersebut. *Game* dapat diklasifikasikan menjadi dua, yaitu[9]:

2.2.3.1 Berdasarkan Jenis Platform yang Digunakan

Tiga kategori utama *platform* pada *game* adalah *personal computer* (PC), console, dan mobile[9]:

1. *Personal Computer* (PC)

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

2. *Console*

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

3. *Mobile*

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

2.2.3.2 Berdasarkan Genre Permainan

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

1. *Action*

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

2. *Strategy*

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

3. *Role Playing Game* (RPG)

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

4. *Sports*

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

5. *Vechicle Simulation*

Vechicle simulation membuat feeling mengendarai kendaraan, baik kendaraan nyata maupun kendaraan imajiner. Performa dan karakteristik kasar mesin harus menyerupai kenyataan, kecuali jika yang didesain adalah kendaraan imajiner.

6. *Construction and Management Simulation*

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

7. *Adventure*

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

8. *Artificial Life and Puzzle Game*

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

9. *Online Game*

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

2.3 Game RPG

Prinsip dasar dari RPG adalah untuk mengontrol karakter yang mengikuti pengembaraan di alam semesta fiksi dimana bisa berbaur unsur fantasi. Fitur utama dari *genre* terletak pada pengembangan karakter. Ini keuntungan dan meningkatkan tingkat statistik, tetapi juga harus memperhatikan peralatan mereka karena juga mempengaruhi statistik karakter, yang dapat sangat berguna untuk menghadapi monster, kesulitan tumbuh, yang bar jalan. Quests yang diberikan oleh NPC (Non Player Character) tersebar luas dalam permainan seperti ini.

Dalam sebuah permainan RPG, jarang ada yang "kalah" atau "menang".Ini membuat permain RPG berbeda dari jenis permainan papan lainnya seperti Monopoli atau Ular Tangga. Seperti sebuah novel atau film, permainan RPG mempunyai daya tarik karena permainan-permainan ini mengajak para pemain untuk menggunakan imajinasi mereka. RPG biasa lebih mengarah ke kolaborasi sosial daripada kompetisi.Pada umumnya dalam RPG, para pemain tergabung dalam satu kelompok.

Permainan RPG rata-rata dimainkan seperti sebuah drama radio : ketika seorang pemain "berbicara", dia berbicara sebagai tokohnya dan ketika si pemain ingin tokohnya melakukan sesuatu yang fisik (seperti menyerang sebuah monster atau membuka sebuah gembok) dia harus menggambarkannya secara lisan.

2.3.1 Elemen Pada Game RPG

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

2.3.1.1 Pengembangan Alur Cerita Dan Karakter Pada RPG

Pengembangan jalan cerita dan karakter merupakan unsur utama dalam *RPG (Role Playing Game).* Cerita yang ada dalam *RPG (Role Playing Game)* tidak harus selalu aneh, tetapi masing-masing *RPG (Role Playing Game)* harus memiliki cirri khas sendiri dan biasanya cerita dalam *RPG (Role Playing Game)* cukup panjang. Cerita dalam *RPG (Role Playing Game)* harus kreatif, inovatif dan mengalir seiring dengan perkembangan karakter utama. Biasanya cerita dalam *RPG (Role Playing Game)* mengandung unsur-unsur khusus yang membentuk sebuah *game RPG (Role Playing Game)* 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 selamat dari 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 RPG (Role Playing Game).* Sekalipun namanya berbeda, mungkin saja efeknya sama, beberapa *game RPG (Role Playing Game)* 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 RPG (Role Playing Game)* 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 *Healing* yang 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 *RPG (Role Playing Game)* adalah suatu ciri khas dari sebuah *game RPG (Role Playing Game)*. 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 RPG (Role Playing Game)*. Biasanya sistem menu berisikan petunjuk dan beberapa manual yang ada *pada game RPG (Role Playing Game)* tersebut. Sistem menu ini sangat membantu pemain dalam memahami dan mengatur karakternya sehingga membuat sang karakter sesuai dengan keinginan sang pemain.

2.4 OOP *(Object Oriented Programming)*

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

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

Keuntungan menggunakan metodologi berorientasi objek adalah sebagai berikut[10]:

a. Meningkatkan produktivitas

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

b. Kecepatan pengembangan

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

c. Kemudahan pemeliharaan

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

d. Adanya konsistensi

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

e. Meningkatkan kualitas perangkat lunak

Karena adanya pendekatan pengembangan lebih dekat dengan dunia nyata dan adanya konsistensi pada saat pengambangannya, perangkat lunak yang dihasilkan akan mampu memenuhi kebutuhan pemakai serta mempunyai sedikit kesalahan.

Berikut beberapa contoh bahasa pemrograman yang mendukung pemrograman berorientasi objek[10] :

a. *Smalltalk*

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

b. Bahasa Pemrograman Eiffel

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

c. Bahasa Pemrograman C++

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

d. Bahasa Pemrograman (web) PHP

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

e. Bahasa Pemrograman Java

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

2.4.1 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 obejek akan memandang sistem yang akan dikembangkan sebagai suatu kumpulan objek yang berkorespondensi dengan objek-objek dunia nyata.

Berikut ini adalah beberapa konsep dasar yang harus dipahami tentang metodelogi berorientasi objek[10]:

1. Kelas (*class*)

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

2. Objek (*object*)

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

3. Metode (*method*)

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

4. Atribut (*attribute*)

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

5. Abstraksi (*abstraction*)

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

6. Enkapsulasi (*encapsulation*)

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

7. Pewarisan (*inheritance*)

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

8. Antarmuka (*Interface*)

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

9. *Reusability*

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

10. *Generalisasi dan Spesialisasi*

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

11. Komunikasi Antar Objek

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

12. Polimorfisme (*polymorphism*)

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

13. *Package*

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

2.4.2 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. Pendekatan analisa berarah objek terdiri dari lima pokok aktivitas, yaitu[10]:

1. Menentukan Kelas dan Objek

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

2. Identifikasi Struktur

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

3. Identifikasi Subjek

   Subjek adalah suatu mekanisme yang membantu pembaca untuk menganalisa model permasalahan dari yang kecil hingga ke yang besar dan komplek. Pada lapisan subjek, gambarkan setiap subjek sebagai kotak persegi yang sederhana, dengan nama subjek dan nomor didalamnya sebagai tambahan, daftarkan kelas-kelas yang termasuk didalam subjek tersebut. Pada lapisan yang lain indikasikan subjek dengan label subjek pada sekat pemisah kotak subjek untuk memandu pembaca dari subjek ke subjek lainnya. Dalam model yang besar dan membutuhkan fasilitas komunikasi, pertimbangkan penggunaan "set" dari lapisan secara terpisah untuk setiap subjek. Sebuah kelas dari subjek berada didalam lebih dari satu subjek. Subjek boleh berisi subjek lainnya. Persiapkan pemetaan multi *level* untuk membantu pembaca dalam suatu model yang besar.

4. Pendefinisian Atribut

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

5. Pendefinisian Service

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

OOA biasanya menggunakan kartu *CRC (Component, Responsibility, Collaborator)* untuk membangun kelas-kelas yang akan digunakan atau *menggunakan UML (Unified Modeling Language)* pada bagian diagram *use case*, diagram kelas, dan diagram objek[10].

2.5 UML (Unified Modelling Language)

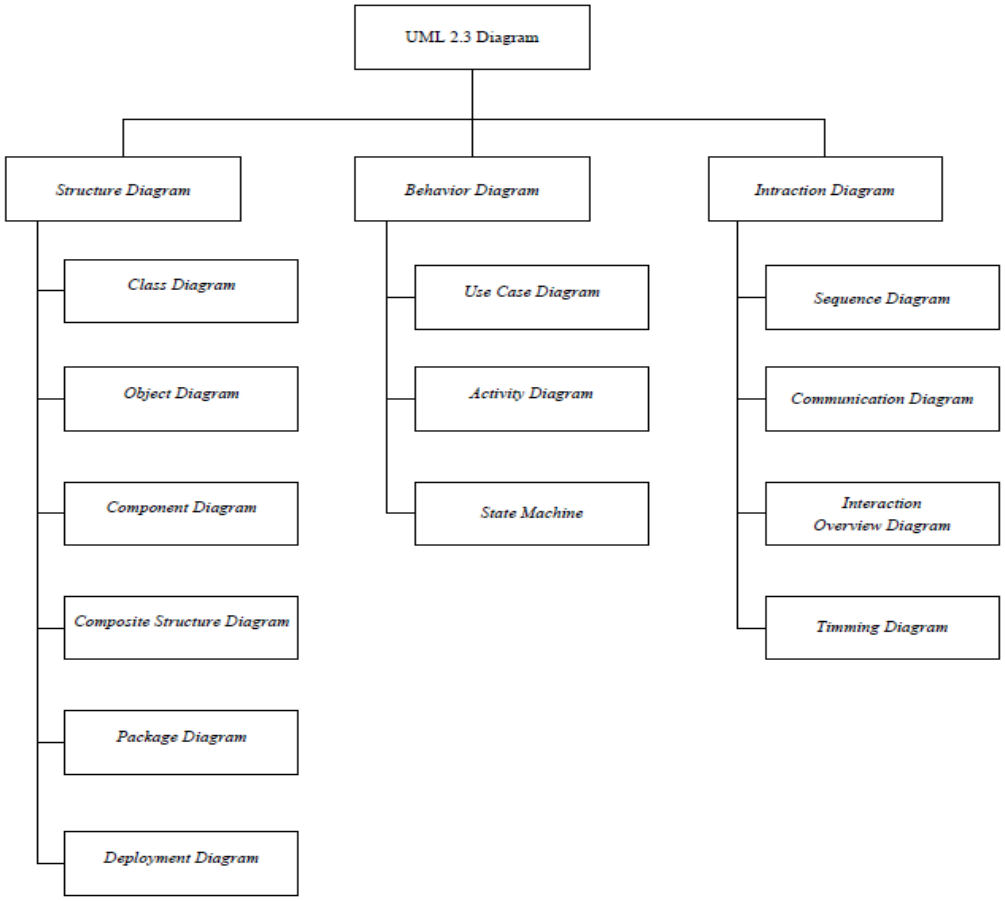
Unified Modelling Language (UML) adalah sebuah "bahasa" yang telah menjadi standar dalam industry visualisasi, merancang, dan mendokumentasikan sistem peranti lunak. *UML* menawarkan sebuah standar untuk merancang model sebuah sistem.

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

Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML syntax mendefinisikan begaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya yaitu Grady Booch OOD (*Object Oriented Design*), Jim Rumbaugh OMT (*Object Modeling Technique*) dan Ivar Jacobson OOSE (*Object Oriented Software Engineering*)[11].

2.5.1 Diagram UML

Pada UML 2.1 terdiri dari 13 macam diagram yang dikelompokkan dalam 3 kategori. Pembagian kategori dan macam-macam diagram tersebut dapat dilihat pada gambar di bawah :



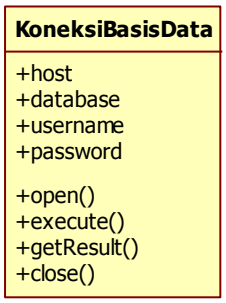
Gambar 2.1 Diagram UML

Berikut ini penjelasan singkat dari pembagian katagori tersebut.

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

2.5.2 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. Contoh Class Diagram dapat dilihat pada Gambar 2.2



Gambar 2. 2 Contoh Class Diagram

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[11].

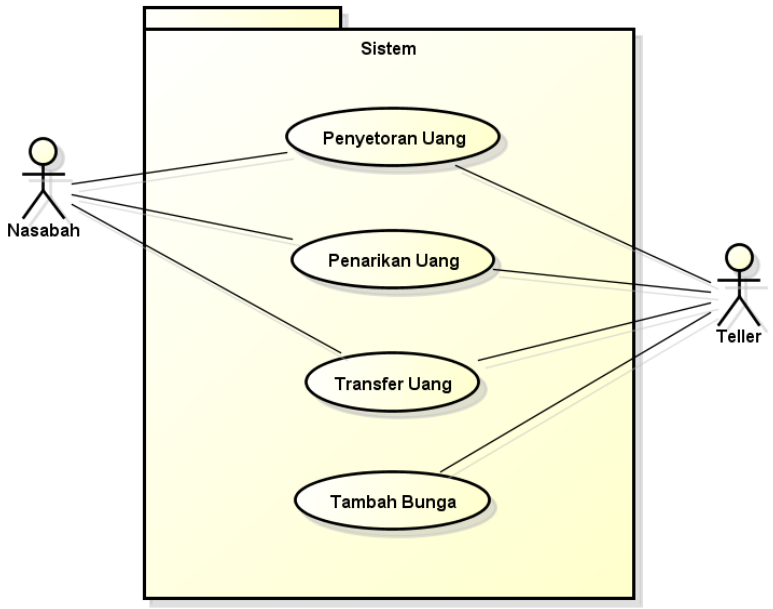
1. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem yang akan dibuat diluar sistem yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
2. Use case merupakan fungsionalitas yang disediakan sistem sebagai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

2.5.3 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[11].

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



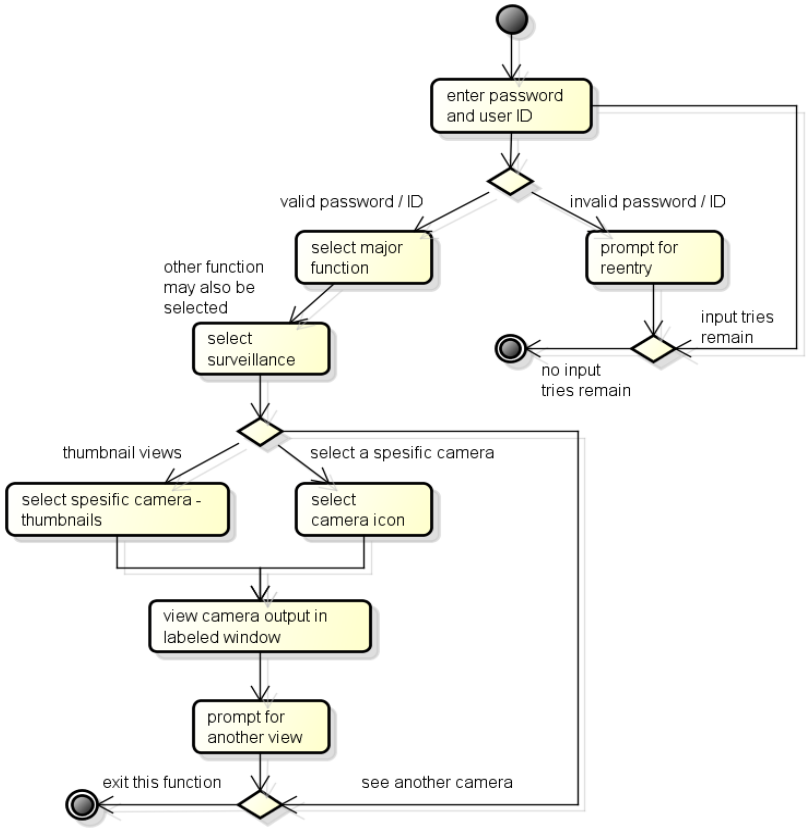
Gambar 2. 3 Contoh Use Case Diagram

2.5.4. 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[11]:

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.

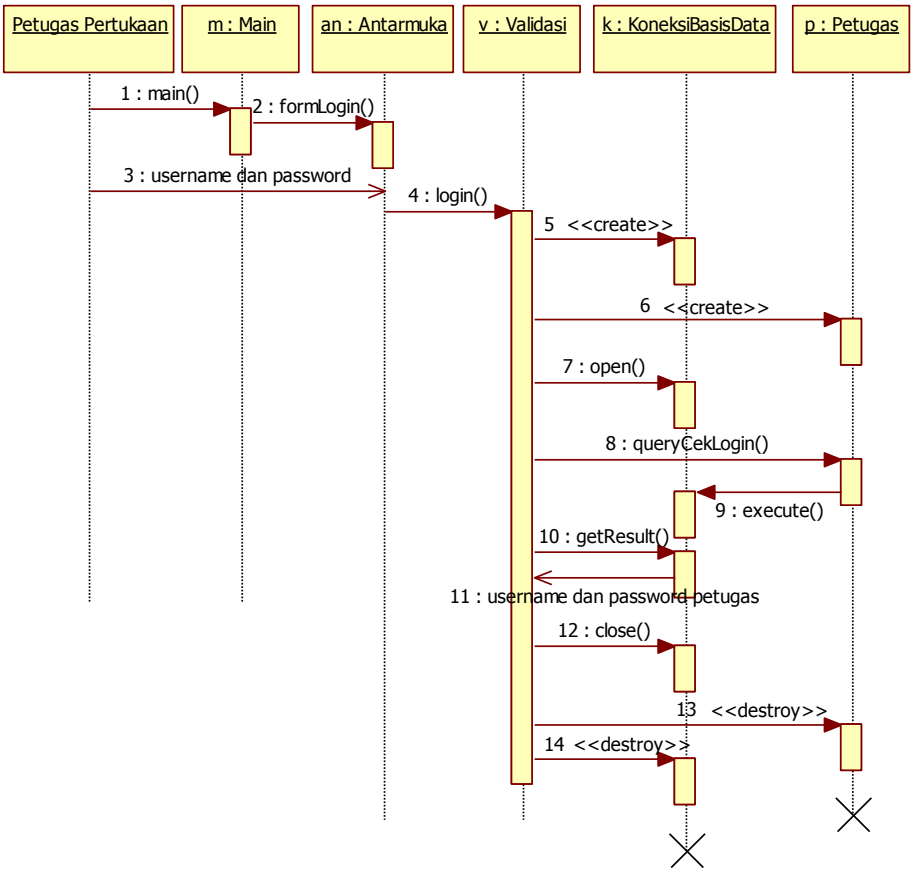


Gambar 2. 4 Contoh Activity Diagram

2.5.5. 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[11]. Contoh Sequence Diagram dapat dilihat pada Gambar 2.5



Gambar 2. 5 Contoh Sequence Diagram

2.6 *Tools* yang digunakan

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

1 *Unity*

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

a. *Function Update*(){}

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

b. *Transform*

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

c. *Rotate*()

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

d. *Time.deltaTime*

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

2 C#

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

3 Adobe photoshop

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

a. PSD (Photoshop Document) format yang menyimpan gambar dalam bentuk layer, termasuk teks, mask, opacity, blend mode, channel warna, channel alpha, clipping paths, dan setting duotone. Kepopuleran photoshop membuat format file ini digunakan secara luas, sehingga memaksa programer program penyunting gambar lainnya menambahkan kemampuan untuk membaca format PSD dalam perangkat lunak mereka.
b. PSB' adalah versi terbaru dari PSD yang didesain untuk file yang berukuran lebih dari 2 GB.
c. PDD adalah versi lain dari PSD yang hanya dapat mendukung fitur perangkat lunak Photshop Deluxe