BAB 2
LANDASAN TEORI

2.1. *Game*

Game merupakan kata dalam bahasa inggris yang berarti permainan. permainan adalah sesuatu yang dapat dimainkan dengan aturan tertentu sehingga ada yang menang dan ada yang kalah, biasanya dalam konteks tidak serius atau hanya untuk mengisi waktu luang. Dalam penggunaannya kata *game* sering digunakan untuk menyebutkan *video game* **[6]**.

2.1.1. Pengertian *Game*

Berikut ini adalah pengertian dan definisi *game* dari beberapa ahli **[6]** :

1. Menurut John C Beck dan Mitchell Wade, *game* adalah penarik perhatian yang telah terbukti. *Game* adalah lingkungan pelatihan yang baik bagi dunia nyata dalam organisasi yang menuntut pemecahan masalah secara kolaborasi.
2. Menurut Ivan C Sibero, *game* merupakan aplikasi yang paling banyak digunakan dan dinikmati para pengguna media elektronik saat ini.
3. Fauzi A, *game* merupakan suatu bentuk hiburan yang seringkali dijadikan sebagai penyegar pikiran dari rasa penat yang disebabkan oleh aktivitas dan rutinitas.
4. Samuel Henry, *game* merupakan bagian tak terpisahkan dari keseharian anak, sedangkan sebagian orang tua menuding *game* sebagai penyebab nilai anak turun, anak tak mampu bersosialisasi dan tindakan kekerasan yang dilakukan anak.
5. Andik Susilo, *game* adalah salah satu candu yang susah dihilangkan, bahkan ada yang mengatakan bahwa candu *game* online setara dengan narkoba.
6. John Naisbitt, *game* merupakan sistem partisipatoris dinamis karena *game* memiliki tingkat penceritaan yang tidak dimiliki film.
7. Albert Einstein, *game* adalah bentuk investigasi paling tinggi.
8. Wijaya Ariyana & Deni Arifianto, *game* merupakan salah satu kebutuhan yang menjadi masalah besar bagi pengguna komputer, karena untuk dapat

memainkan *game* dengan nyaman, semua komponen komputernya harus memiliki kualitas yang baik, terutama VGA card-nya.

2.1.2. Elemen Dasar *Game*

Menurut Teresa Dillon **[7]** elemen-elemen dasar sebuah *game* adalah :

1. *Game Rule*

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

2. Plot

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

3. Thema

   Di dalam biasanya ada pesan moral yang akan disampaikan.

4. *Character*

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

5. *Object*

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

6. *Text, Grafik dan Sound*

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

7. Animasi

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

8. *User Interface*

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

2.1.3. Klasifikasi *Game*

Dalam *game*, klasifikasi *game* terbagi menjadi dua **[6]**. yaitu :

2.1.4.1. Berdasarkan Jenis *Platform* yang digunakan

1. *Arcade games,* yaitu yang sering disebut ding-dong di Indonesia, biasanya berada di daerah/tempat khusus dan memiliki box atau mesin yang memang khusus di *design* untuk jenis *video games* tertentu dan tidak jarang bahkan memiliki fitur yang dapat membuat pemainnya lebih merasa masuk dan menikmati, seperti pistol, kursi khusus, sensor gerakan, sensor injakkan dan stir mobil (beserta transmisinya tentunya)
2. *PC Games,* yaitu *video game* yang dimainkan menggunakan *Personal Computers.*
3. *Console games,* yaitu *video games* yang dimainkan menggunakan *console* tertentu, seperti Playstation 2, Playstation 3, XBOX 360, dan Nintendo Wii.
4. *Handheld games,* yaitu yang dimainkan di *console* khusus *video game* yang dapat dibawa kemana-mana, contoh Nintendo DS dan Sony PSP.
5. *Mobile games*, yaitu yang dapat dimainkan atau khusus untuk *mobilephone* atau PDA.

2.1.4.2. Berdasarkan *Genre* permainannya

Genre dari *game* tersebut yaitu a*ction* s*hooting (aksi tembak-tembakan, fighting* (pertarungan), *action adeventure* (aksi petualangan), *adventure* (petualangan), *simulation* (simulasi), *construction and management* (konstruksi dan manajemen), *role playing* (permainan peran), *strategy* (strategi), *puzzle* (teka-teki), *vehicle simulation* (simulasi kendaraan), *sport* (olahraga).

1. *Action Shooting* (Aksi Tembak-tembakan)

Game jenis ini sangat memerlukan kecepatan refleks, koordinasi mata-tangan, juga timing, inti dari *game* jenis ini adalah tembak, tembak dan tembak. Termasuk didalam-nya :

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

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

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

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

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

2. *Fighting* (Pertarungan)

Ada yang mengelompokan *video game fighting* di bagian aksi, jenis ini memang memerlukan kecepatan refleks dan koordinasi mata-tangan, tetapi inti dari *game* ini adalah penguasaan jurus (hafal caranya dan lancar mengeksekusinya), pengenalan karakter dan *timing* sangatlah penting, berbeda seperti *game* aksi pada umumnya yang umumnya hanya melawan *Artificial Intellegence* atau istilah umumnya melawan komputer saja, pemain jenis *fighting game* ini baru teruji kemampuan sesungguhnya dengan melawan pemain lainnya. Seri Street Fighter, Tekken, Mortal Kombat, Soul Calibur dan King of Fighter adalah contohnya.

3. *Action Adventure* (Aksi Petualangan)

Memasuki gua bawah tanah, melompati bebatuan di antara lahar, bergelayutan dari pohon satu ke pohon lain, bergulat dengan ular sambil mencari kunci untuk membuka pintu kuil legendaris, atau sekedar mencari telepon umum untuk mendapatkan misi berikutnya, itulah beberapa dari banyak hal yang karakter pemain harus lakukan dan lalui dalam *video game* jenis ini. *game* jenis ini sudah berkembang jauh hingga menjadi *genre* campuran *action beat-em up* juga, dan sekarang di tahun 2000-an jenis ini cenderung untuk memiliki *visual* 3D dan sudut pandang orang ke-tiga. Tomb Rider, Grand Theft Auto dan Prince of Persia termasuk didalamnya.

4. *Adventure* (Petualangan)

Bedanya dengan jenis *video game* aksi-petualangan, refleks dan kelihaian pemain dalam bergerak, berlari, melompat hingga memecut atau menembak tidak diperlukan di sini. *Video Game* murni petualangan lebih menekankan pada jalan cerita dan kemampuan berpikir pemain dalam menganalisa tempat secara visual, memecahkan teka-teki maupun menyimpulkan rangkaian peristiwa dan percakapan karakter hingga penggunaan benda-benda tepat pada tempat yang tepat. Termasuk didalamnya:

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

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

5. *Simulation, Construction and Management* (Simulasi, Konstruksi dan Manajemen).

Video Game jenis ini seringkali menggambarkan dunia di dalamnya sedekat mungkin dengan dunia nyata dan memperhatikan dengan detil berbagai faktor. Dari mencari jodoh dan pekerjaan, membangun rumah, gedung hingga kota, mengatur pajak dan dana kota hingga keputusan memecat atau menambah karyawan. Dunia kehidupan rumah tangga sampai bisnis membangun konglomerasi, dari jualan limun pinggir jalan hingga membangun laboratorium *cloning*. *Video Game* jenis ini membuat pemain harus berpikir untuk mendirikan, membangun dan mengatasi masalah dengan menggunakan dana yang terbatas. Contoh: Sim City, The Sims, Tamagotchi.

6. *Role Playing* (Permainan Peran)

Video game jenis ini sesuai dengan terjemahannya, bermain peran, memiliki penekanan pada tokoh/peran perwakilan pemain di dalam permainan, yang biasanya adalah tokoh utamanya, dimana seiring dalam memainkannya, karakter tersebut dapat berubah dan berkembang ke arah yang diinginkan pemain (biasanya menjadi semakin hebat, semakin kuat, semakin berpengaruh, dll) dalam berbagai parameter yang biasanya ditentukan dengan naiknya *level*, baik dari status kepintaran, kecepatan dan kekuatan karakter,

senjata yang semakin sakti, ataupun jumlah teman maupun mahluk peliharaan. Secara kebudayaan, pengembang *game* Jepang biasanya membuat *Role Playing Game* (RPG) ke arah cerita linear yang diarahkan seolah karakter yang dimainkan adalah tokoh dalam cerita itu, seperti Final Fantasy, Dragon Quest dan Xenogears. Sedangkan pengembang *game* RPG Eropa, cenderung membuat karakter yang dipilih bebas memilih jalan cerita sendiri *secara non-linear*, seperti Ultima, Never Winter Nights, baldurs gate, Elder Scroll, dan Fallout.

7. *Strategy* (Strategi)

Kebalikan dari *video game* jenis *action* yang berjalan cepat dan perlu refleks secepat kilat, *video game* jenis strategi, layaknya bermain catur, justru lebih memerlukan keahlian berpikir dan memutuskan setiap gerakan secara hati-hati dan terencana. *Video game* strategi biasanya memberikan pemain atas kendali tidak hanya satu orang tapi minimal sekelompok orang dengan berbagai jenis tipe kemampuan, sampai kendaraan, bahkan hingga pembangunan berbagai bangunan, pabrik dan pusat pelatihan tempur, tergantung dari tema ceritanya. Pemain *game* strategi melihat dari sudut pandang lebih meluas dan lebih kedepan dengan waktu permainan yang biasanya lebih lama dan santai dibandingkan *game action*. Unsur-unsur permainannya biasanya berkisar sekitar, prioritas pembangunan, peletakan pasukan, mencari dan memanfaatkan sumberdaya (uang, besi, kayu, minyak, dll), hingga ke pembelian dan peng-*upgrade*-an pasukan atau teknologi. *Game* jenis ini terbagi atas:

a) *Real time Strategy, game* berjalan dalam waktu sebenarnya dan serentak antara semua pihak dan pemain harus memutuskan setiap langkah yang diambil saat itu juga berbarengan mungkin saat itu pihak lawan juga sedang mengeksekusi strateginya. Contoh: Starcraft, Warcraft dan Command and Conquer.

b) *Turn based Strategy , game* yang berjalan secara bergiliran, ketika pemain mengambil keputusan dan menggerakan pasukan, saat itu pihak lawan menunggu, begitu pula sebaliknya, layaknya catur. contoh: Front Mission,

Super robot wars, Final Fantasy tactics, Heroes of might and magic, Master of Orion.

8. *Puzzle* (Teka-Teki)

Video game jenis ini sesuai namanya berintikan mengenai pemecahan teka-teki, baik itu menyusun balok, menyamakan warna bola, memecahkan perhitungan matematika, melewati labirin, sampai mendorong-dorong kota masuk ke tempat yang seharusnya, itu semua termasuk dalam jenis ini. Sering pula permainan jenis ini adalah juga unsur permainan dalam *video game* petualangan maupun *game* edukasi. Tetris, Minesweeper, Bejeweled, Sokoban dan Bomberman.

9. *Vehicle Simulation* (Simulasi kendaraan)

Video Game jenis ini memberikan pengalaman atau interaktifitas sedekat mungkin dengan kendaraan yang aslinya, meskipun terkadang kendaraan tersebut masih eksperimen atau bahkan fiktif, tapi ada penekanan khusus pada detil dan pengalaman realistik menggunakan kendaraan tersebut.

10. *Sport* (Olahraga)

Singkat padat jelas, bermain *sport* di PC atau konsol anda. Biasanya permainannya diusahakan serealistik mungkin walau kadang ada yang menambah unsur fiksi seperti NBA JAM. Contohnya pun jelas, Seri Winning Eleven, seri NBA, seri FIFA, John Madden NFL, Lakers vs Celtics, Tony hawk pro skater, dll.

2.2. *Artificial Intelligence* (Kecerdasan Buatan)

Kecerdasan buatan atau *artificial intelligence* merupakan salah satu bagian ilmu komputer yang membuat agar mesin (komputer) dapat melakukan pekerjaan seperti dan sebaik yang dilakukan oleh manusia. Pada awal diciptakannya, komputer hanya difungsikan sebagai alat hitung saja. Namun seiring dengan perkembangan jaman, maka peran komputer semakin mendominasi kehidupan umat manusia. Komputer tidak lagi hanya digunakan sebagai alat hitung, lebih dari itu, komputer diharapkan untuk dapat diberdayakan untuk mengerjakan segala sesuatu yang bisa dikerjakan oleh manusia.

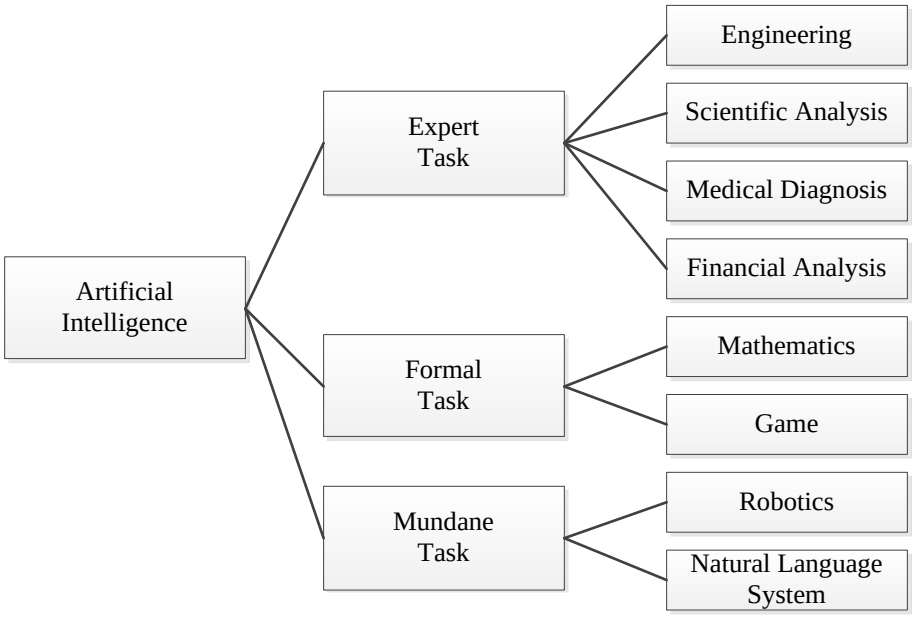
Pengertian kecerdasan buatan dapat dipandang dari berbagai sudut pandang, antara lain:

1. Sudut pandang kecerdasan.

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

2. Sudut pandang penelitian.

Kecerdasan Buatan adalah suatu studi bagaimana membuat agar komputer dapat melakukan sesuatu sebaik yang dikerjakan oleh manusia. Gambar 2.1 merupakan sejumlah bidang-bidang tugas (*task domains*) dari AI.



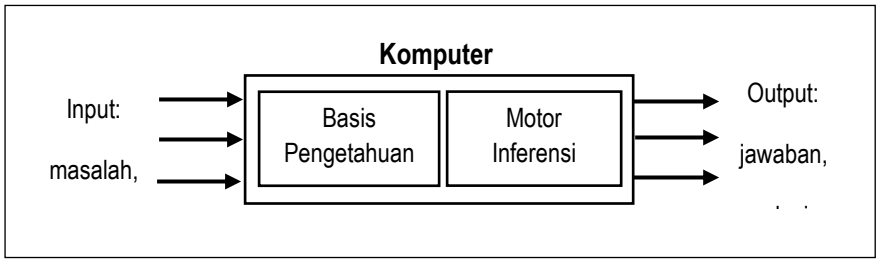
Gambar 2.1 Bidang-Bidang Tugas

1. *Mundane task*
   a. Persepsi (*vision* & *speech*).
   b. Bahasa alami (*understanding, generation* & *translation*).
   c. Pemikiran yang bersifat *commonsense*.
   d. *Robot control*.
2. *Formal task*
   a. Permainan/*games*.
   b. Matematika (geometri, logika, kalkulus integral, pembuktian).

3. *Expert task*
   a. Analisis finansial.
   b. Analisis medikal.
   c. Analisis ilmu pengetahuan.
   d. Rekayasa (desain, pencarian kegagalan, perencanaan manufaktur).
4. Sudut pandang bisnis.

   Kecerdasan buatan adalah kumpulan peralatan yang sangat *powerfull* dan metodologis dalam menyelesaikan masalah-masalah bisnis.
5. Sudut pandang pemrograman.

   Kecerdasan buatan meliputi studi tentang pemrograman simbolik, penyelesaian masalah (*problem solving*) dan pencarian (*searching*). Untuk melakukan aplikasi kecerdasan buatan ada 2 bagian utama yang sangat dibutuhkan (Gambar 2.2), yaitu:
   a. Basis Pengetahuan (*Knowledge Base*), berisi fakta-fakta, teori, pemikiran dan hubungan antara satu dengan lainnya.
   b. Motor Inferensi (*Inference Engine*), yaitu kemampuan menarik kesimpulan berdasarkan pengalaman.



Gambar 2.2 Penerapan Konsep Kecerdasan Buatan di Komputer

2.2.1. Faktor Pendorong Pekembangan Kecerdasan Buatan

Faktor pendorong bagi terlaksananya aplikasi kecerdasan buatan adalah :

1. Pesatnya perkembangan teknologi perangkat keras.

Hampir semua aplikasi kecerdasan buatan memerlukan perangkat keras yang memiliki kecepatan daya tampung yang lebih tinggi, walaupun hanya menjalankan perangkat lunak kecerdasan buatan yang paling sederhana sekalipun.

Disamping itu, harga perangkat keras yang dengan kemampuan lebih memiliki harga yang relatif semakin murah.

2. **Pengembangan perangkat lunak kecerdasan buatan**

Dewasa ini bahasa dan alat pemrograman baru yang lebih canggih sudah banyak dikembangkan dan dipasarkaan secara luas, termasuk bahasa khusus untuk Kecerdasan Buatan.

3. **Perkembangan khusus komputer pribadi (Personal Computer/PC).**

Sekarang sudah sangat banyak orang menggunakan komputer mikro (*microcomputer*) khususnya komputer pribadi baik di sekolah, perusahaan atau bahkan di rumah yang menyebabkan permintaan terhadap perangkat lunak yang lebih unggul untuk pekerjaan tersebut.

4. **Turut andilnya para investor dalam mendanai penelitian dan pengembangan teknologi kecerdasan buatan.**

Hal ini mengakibatkan terjadinya semacam tekanan di kalangan masyarakat, kecerdasan buatan untuk berlomba-lomba dalam mempercepat gerak dan langkah penelitiannya dan segera memproduksi kecerdasan buatan dalam waktu yang singkat.

Masalah utama Kecerdasan Buatan adalah sulitnya merumuskan dan memvisualisasi inteligensia itu sendiri, karena mempunyai arti yang banyak. Walaupun AI telah banyak membuat komputer menjadi lebih pintar dan lebih canggih, tapi tampaknya impian manusia agar bisa membuat komputer yang betul-betul bisa membuat duplikasi otak manusia atau bisa menjadi pengganti otak manusia yang sebenarnya masih jauh dari kenyataan. Mungkin belum bisa terlaksana pada zaman atau masa kini.

Walaupun masih banyak kritik, metode Kecerdasan Buatan tetap memperlihatkan nilai lebih dibanding dengan yang lain. Teknik Kecerdasan Buatan menunjukkan bagaimana dalam berfikir dan bagaimana menerapkan inteligensia dengan lebih baik. Teknik Kecerdasan Buatan akan membuat komputer lebih mudah digunakan dan pengetahuan akan semakin tersebar luas di kalangan masyarakat.

2.2.2. Lingkup Kecerdasan Buatan pada Aplikasi Komersial

Makin pesatnya perkembangan teknologi menyebabkan adanya perkembangan dan perluasan lingkup yang membutuhkan kehadiran kecerdasan buatan. Karakteristik ‘cerdas’ sudah mulai dibutuhkan di berbagai disiplin ilmu dan teknologi. Kecerdasan buatan tidak hanya dominan di bidang ilmu komputer (informatika), namun juga sudah merambah di berbagai disiplin ilmu yang lain. Interseksi antara psikologi dan kecerdasan buatan melahirkan sebuah area yang dikenal dengan nama *cognition & psycolinguistics*. Interseksi antara teknik elektro dengan kecerdasan buatan melahirkan berbagai ilmu seperti: pengolahan citra, teori kendali, pengenalan pola dan robotika.

Lingkup utama dalam kecerdasan buatan adalah:

1. Sistem Pakar (*Expert System*). Disini komputer digunakan sebagai sarana untuk menyimpan pengetahuan para pakar. Dengan demikian komputer akan memiliki keahlian untuk menyelesaikan permasalahan dengan meniru keahlian yang dimiliki oleh pakar.
2. Pengolahan Bahasa Alami (*Natural Language Processing*). Dengan pengolahan bahasa alami ini diharapkan user dapat berkomunikasi dengan komputer dengan menggunakan bahasa sehari-hari.
3. Pengenalan Ucapan (*Speech Recognition*). Melalui pengenalan ucapan diharapkan manusia dapat berkomunikasi dengan komputer dengan menggunakan suara.
4. Robotika & Sistem Sensor (*Robotics* & *Sensory Systems*).
5. *Computer Vision*, mencoba untuk dapat menginterpretasikan gambar atau obyek-obyek tampak melalui komputer.
6. *Intelligent Computer-aided Instruction*. Komputer dapat digunakan sebagai tutor yang dapat melatih dan mengajar.
7. *Game Playing*.

Beberapa karakteristik yang ada pada sistem yang menggunakan *artificial intelligence* adalah pemrogramannya yang cenderung bersifat simbolik ketimbang

algoritmik, bisa mengakomodasi input yang tidak lengkap, bisa melakukan inferensi dan adanya pemisahan antara kontrol dengan pengetahuan.

Namun, seiring dengan perkembangan teknologi, muncul beberapa teknologi yang juga bertujuan untuk membuat agar komputer menjadi cerdas sehingga dapat menirukan kerja manusia sehari-hari.

2.2.3. Teknik Dasar Pencarian

Pencarian atau pelacakan merupakan salah satu teknik untuk menyelesaikan permasalahan AI. Keberhasilan suatu sistem salah satunya ditentukan oleh kesuksesan dalam pencarian dan pencocokan. Teknik dasar pencarian memberikan suatu kunci bagi banyak sejarah penyelesaian yang penting dalam bidang AI. Ada beberapa aplikasi yang menggunakan teknik pencarian ini, yaitu :

1. Papan *game* dan puzzle (tic-tac-toe, catur, menara hanoi).
2. Penjadwalan dan masalah *routing* (*travelling salesman problem*).
3. Parsing bahasa dan inteprestasinya (pencarian struktur dan arti).
4. Logika pemrograman (pencarian fakta dan implikasinya).
5. Komputer *vision* dan pengenalan pola.
6. Sistem pakar bebasis kaidah (*rule based expert sistem*).

Pencarian adalah proses mencari solusi dari suatu permasalahan melalui sekumpulan kemungkinan ruang keadaan (*state space*). Ruang keadaan merupakan suatu ruang yang berisi semua keadaan yang mungkin. Kondisi suatu pencarian meliputi :

1. Keadaan sekarang atau awal.
2. Keadaan tujuan-solusi yang dijangkau dan perlu diperiksa apakah telah mencapai sasaran.
3. Biaya atau nilai yang diperoleh dari solusi.

Solusi merupakan suatu lintasan dari keadaan awal sampai keadaan tujuan. Secara umum, proses pencarian dapat dilakukan seperti berikut :

1. Memeriksa keadaan sekarang atau awal.
2. Mengeksekusi aksi yang dibolehkan untuk memindahkan ke keadaan berikutnya.

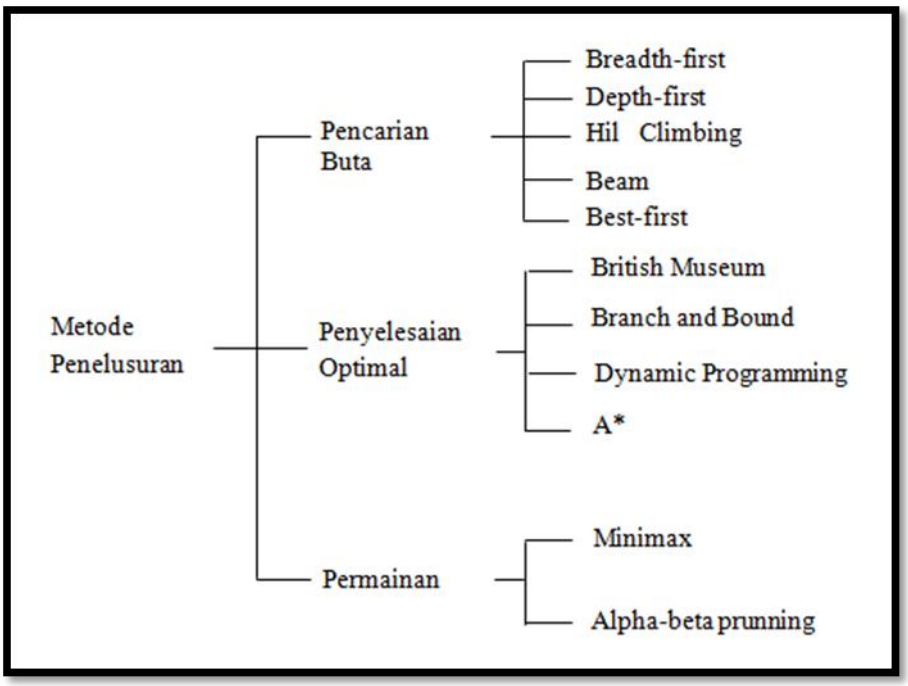
3. Memeriksa jika keadaan baru merupakan solusinya. Jika tidak, keadaan baru tersebut menjadi keadaan sekarang dan proses ini diulangi sampai solusi ditemukan atau ruang keadaan habis terpakai **[8]**.

2.2.4. Algoritma Pencarian

Permasalahan pencarian adalah merupakan yang sering dijumpai oleh peneliti di bidang kecerdasan buatan. Permasalahan ini merupakan hal penting dalam menentukan keberhasilan sistem kecerdasan buatan. Metode pencarian yang pertama adalah metode yang sederhana yang hanya berusaha mencari kemungkinan penyelesaian. Metode yang termasuk pada bagian ini adalah *dept-first search, hill climbing, breadth-first search, beam search* dan *best-first search*.

Yang kedua, adalah metode yang lebih kompleks yang akan mencari jarak terpendek. Metode ini adalah *British Museum Procedure*, *Branch and Bound*, *Dynamic Programming* dan A*. Metode-metode ini digunakan pada saat harga perjalanan untuk mencari kemungkinan menjadi perhitungan.

Yang ketiga, adalah beberapa prosedur/metode yang terapkan saat berhadapan dengan musuh. Prosedur ini adalah *minimax search*, *alpha-beta prunning*. Metode ini banyak digunakan pada program-program permainan seperti catur dsb. Dalam Gambar 2.3 terdapat bagan untuk Metode *Searching*.



Gambar 2.3 Metode Pencarian [8].

Menurut cara algoritma mengembangkan node dalam proses pencarian, gambar bagan metode penulusuran dibagi menjadi dua golongan, yakni pencarian buta (*blind search*) dan pencarian terbimbing (*heuristic search*).

1. Pencarian Buta (*Blind Search*)
2. Pencarian Terbimbing (*Heuristic Search*)

2.2.5. Algoritma A* (A *Star*)

Algoritma A*, dapat juga disebut sebagai Algoritma A *Star*, A *Star* merupakan salah satu contoh algoritma pencarian yang cukup popular di dunia. Beberapa terminologi dasar yang terdapat pada algoritma ini adalah simpul (*node*), simpul asal/mulai (*source node*), simpul akhir/tujuan (*destination node*), simpul sekarang (*current node*), *openlist*, *closedlist*, *bestpath*, biaya (*cost*), halangan (*unwalkable/obstacle*). Simpul (*node*) adalah petak-petak kecil sebagai representasi dari area pencarian (*pathfinding*). Bentuknya dapat berupa persegi, lingkaran, maupun segitiga. Simpul asal/mulai (*source node*) adalah sebuah terminologi untuk posisi awal sebuah benda. Simpul akhir/tujuan (*destination node*) adalah tempat tujuan yang ingin dicapai pada pencarian. Simpul sekarang

(*current node*) adalah simpul terbaik sebelumnya yang dipilih dan menjadi titik acuan untuk membangkitkan simpul tetangganya (*adjacent*). *Open list* adalah tempat menyimpan data simpul yang diakses dari simpul asal/mulai (*source node*) atau dari tetangga simpul sekarang (*current node*) yang belum pernah berada di *open list* maupun *closed list*. *Closed list* adalah tempat menyimpan simpul yang pernah menjadi simpul sekarang (*current node*). Biaya biasanya disimbolkan dengan huruf F (*total cost*) adalah nilai yang diperoleh dari penjumlahan nilai G (*actual cost*) dengan nilai H (*heuristic cost*). Halangan/penghalang adalah simpul yang tidak dapat dilalui. Awal dari algoritma ini bekerja adalah dengan menentukan terlebih dahulu simpul asal (*source node*) dan simpul tujuan (*destination node*).

Simpul asal (*source node*) dijadikan sebagai acuan atau simpul sekarang (*current node*) dan semua simpul tetangga (*adjacent*) dari simpul sekarang (*current node*) dibangkitkan. Ketika simpul tetangga tersebut belum pernah berada di *open list* maka hitung biaya G (*actual cost*) dan biaya H (*heuristic cost*).

Perhitungan nilai H yaitu nilai estimasi jalur terpendek dari *node* awal ke *node* tujuan dirumuskan dengan :

$$H = 2*(abs(n.x-tujuan.x) + abs(n.y-tujuan.y)) \quad \ldots(2.1)$$

Biaya G (*actual cost*) adalah biaya sebenarnya, biasanya berupa jarak yang pernah ditempuh. Tapi dalam kasus tertentu biaya G (*actual cost*) dapat berupa penjumlahan antara jarak dengan biaya medan (*terrain cost*). Kemudian menentukan parent dari simpul tersebut. Setelah biaya tersebut dihitung dan parent ditentukan kemudian simpul tersebut dimasukkan ke *open list*. Jika simpul tetangga pernah berada di *open list* maupun di *closed list* maka biaya G (*actual cost*) diperiksa jika biayanya sudah paling optimal maka tidak perlu untuk merubah parentnya. Setelah melakukan pemeriksaan terhadap simpul tetangga maka simpul sekarang (*current node*) dimasukkan ke c*losed list* dan setelah itu dipilih simpul sekarang (*current node*) yang baru. Pemilihan simpul sekarang (*current node*) yang baru adalah berdasarkan besarnya biaya F (*total cost*) yang terdapat pada simpul yang disimpan di *open list*. Proses tersebut terus dilakukan

sampai simpul sekarang (*current node*) adalah simpul tujuan. Kemudian setelah itu dilakukan proses penelusuran balik pada simpul parent dan disimpan di (*bestpath*) **[9]**.

2.3. Multimedia

Multimedia adalah penggunaan komputer untuk menyajikan dan menggabungkan teks, suara, gambar, animasi dan *video* dengan alat bantu (*tool*) dan koneksi (*link*) sehingga pengguna dapat bernavigasi, berinteraksi, berkarya dan berkomunikasi **[10]**. Multimedia sering digunakan dalam dunia hiburan. Selain dari dunia hiburan, multimedia juga diadopsi oleh dunia *game*.

Multimedia juga dapat diartikan sebagai penggunaan beberapa media yang berbeda dalam menyampaikan informasi berbentuk teks, suara, grafik, animasi dan *video*.

2.3.1. Definisi Multimedia menurut beberapa ahli [10] :

1. Rosch : Multimedia adalah kombinasi dari komputer dan video.
2. McComick : Multimedia adalah kombinasi dari tiga elemen: suara, gambar, dan teks.
3. Turban dan kawan-kawan : Multimedia adalah Kombinasi dari paling sedikit dua media *input* atau *output*. Media ini dapat berupa *audio* (suara, musik), animasi, *video*, teks, grafik dan gambar
4. Robin dan Linda : Multimedia adalah Alat yang dapat menciptakan presentasi yang dinamis dan interaktif yang mengkombinasikan teks, grafik, animasi, *audio* dan *video*
5. Steinmetz : Multimedia adalah gabungan dari seminimalnya sebuah media diskrit dan sebuah media kontinu. Media diskrit adalah sebuah media dimana validitas datanya tidak tergantung dari kondisi waktu, termasuk didalamnya teks dan grafik. Sedangkan yang dimaksud dengan media kontinu adalah sebuah media dimana validitas datanya tergantung dari kondisi waktu, termasuk di dalamnya suara dan *video*.

6. Vaughan : Multimedia adalah beberapa kombinasi dari teks, gambar, suara, animasi dan *video* dikirim ke anda melalui komputer atau alat elektronik lainnya atau dengan manipulasi digital.

2.3.2. Kategori Multimedia

Multimedia dapat di definisikan menjadi 2 kategori, yaitu *Multimedia Content Production* dan *Multimedia Communication*.

1. *Multimedia Content Production* adalah penggunaan beberapa media (*text, audio, graphics, animation, video dan interactivity*) yang berbeda dalam menyampaikan suatu informasi atau menghasilkan produk multimedia seperti *video, audio, musik, film, game, entertaintment,* dll. Bisa juga dikatakan sebagai penggunaan beberapa teknologi yang berbeda yang memungkinkan untuk menggabungkan media *(text, audio, graphics, animation, video, dan interactivity*) dengan cara yang baru untuk tujuan komunikasi. Dalam kategori ini media yang digunakan adalah :
   a. Media teks/tulisan
   b. Media *audio*/suara
   c. Media *video*
   d. Media animasi
   e. Media gambar
   f. Media Interaktif
   g. Media *special effect*
2. *Multimedia Communication* adalah penggunaan media (massa), seperti televisi, radio, media cetak dan internet untuk mempublikasikan/menyiarkan/mengkomunikasikan material periklanan, publikasi, entertaintment, berita, pendidikan, dll. Dalam kategori ini media yang digunakan adalah :
   a. TV
   b. Radio
   c. Film
   d. Media Cetak
   e. Musik

f. Game

g. Entertainment

h. Tutorial

i. Internet

2.3.3. Elemen Multimedia

Menurut Hofstetter **[11]** komponen multimedia terbagi atas lima jenis yaitu:

1. Teks

Teks merupakan elemen multimedia yang menjadi dasar untuk menyampaikan informasi, karena teks adalah jenis data yang paling sederhana dan membutuhkan tempat penyimpanan yang paling kecil. Teks merupakan cara yang paling efektif dalam mengemukakan ide-ide kepada pengguna, sehingga penyampaian informasi akan lebih mudah dimengerti oleh masyarakat. Jenis-jenis teks seperti *Printed Text*, yaitu teks yang dihasilkan oleh *word processor* atau *word editor* dengan cara diketik yang nantinya dapat dicetak. *Scanned Text* yaitu teks yang dihasilkan melalui proses *scanning* tanpa pengetikan dan *Hypertext* yaitu jenis teks yang memberikan *link* ke suatu tempat/meloncat ke topik tertentu.

2. Grafik (*image*)

Sangat bermanfaat untuk mengilustrasi informasi yang akan disampaikan terutama informasi yang tidak dapat dijelaskan dengan kata-kata. Jenis-jenis grafik seperti bitmap yaitu gambar yang disimpan dalam bentuk kumpulan *pixel*, yang berkaitan dengan titik-titik pada layar monitor. *Digitized picture* adalah gambar hasil rekaman *video* atau kamera yang dipindahkan ke komputer dan diubah ke dalam bentuk bitmaps. *Hyperpictures*, sama seperti *hypertext* hanya saja dalam bentuk gambar.

3. *Audio*

Multimedia tidak akan lengkap jika tanpa *audio* (suara). *Audio* bisa berupa percakapan, musik atau efek suara.

Format dasar *audio* terdiri dari beberapa jenis :

a. WAVE

Merupakan format file digital *audio* yang disimpan dalam bentuk digital dengan eksistensi WAV.

b. MIDI (*Musical Instrument Digital Interface*)

MIDI memberikan cara yang lebih efisien dalam merekam musik dibandingkan wave, kapasitas data yang dihasilkan juga jauh lebih kecil. MIDI disimpan dalam bentuk MID.

4. *Video*

Video menyediakan sumber yang kaya dan hidup untuk aplikasi multimedia. Dengan video dapat menerangkan hal-hal yang sulit digambarkan lewat kata-kata atau gambar diam dan dapat menggambarkan emosi dan psikologi manusia secara lebih jelas.

5. Animasi

Animasi adalah simulasi gerakan yang dihasilkan dengan menayangkan rentetan *frame* ke *layer*. *Frame* adalah satu gambar tunggal pada rentetan gambar yang membentuk animasi.

2.4. Budaya Indonesia

Kebudayaan Indonesia walau beraneka ragam, namun pada dasarnya terbentuk dan dipengaruhi oleh kebudayaan besar lainnya seperti kebudayaan Tionghoa, kebudayaan India dan kebudayaan Arab.

2.4.1. Kebudayaan Kalimantan

Suku dayak dan rumah panggung merupakan salah satu kebudayaan yang ada di Indonesia yang harus dilestarikan. Berikut merupakan deskripsi dari budaya yang ada di tersebut :

1. Suku Dayak

Suku Dayak adalah suku asli Kalimantan yang hidup berkelompok yang tinggal di pedalaman, di gunung dan sebagainya. Kata Dayak itu sendiri sebenarnya diberikan oleh orang-orang Melayu yang datang ke Kalimantan. Orang-orang Dayak sendiri sebenarnya keberatan memakai nama Dayak, sebab lebih diartikan agak negatif. Padahal, semboyan orang Dayak adalah “Menteng Ueh Mamut”, yang berarti seseorang yang memiliki kekuatan gagah berani, serta tidak kenal menyerah atau pantang mundur.



Gambar 2.4 Suku Dayak

2. Rumah Panggung

Rumah panggung merupakan salah satu bentuk kearifan tradisional yang bisa dibanggakan sebagai salah satu produk budaya masyarakat Indonesia. Bentuk rumah ini merupakan hasil adaptasi masyarakat terhadap lingkungan alam, misalnya pasang-surut air, menghindari banjir dan binatang buas. Di banyak tempat, terutama di daerah pedalaman, teknik rumah panggung ini masih dipertahankan karena keselarasaanya dengan alam sekitar. Lantai rumah panggung di daerah daratan biasanya berjarak 1-2 meter dari tanah, sedangkan di daerah rawa atau lahan basah bisa berjarak hingga 4-10 meter dari permukaan air terendah saat surut. Bagian bawah rumah panggung dapat tetap menyerap atau dilalui air. Artinya, ramah lingkungan dan selaras dengan fungsi hidrologi.



Gambar 2.5 Rumah Panggung

2.5. Tools yang Digunakan

Dalam membangun aplikasi *game Save Our Planet From Extinction* dibutuhkan tools yang dapat digunakan dalam pembangunan *game* tersebut.

2.5.1. OOP (Object Oriented Programming)

OOP (*Object Oriented Programming*) atau yang dikenal dengan Pemrograman Berorientasi Objek merupakan paradigma pemrograman yang berorientasikan kepada objek. Semua data dan fungsi di dalam paradigma ini dibungkus ke dalam kelas-kelas atau objek-objek.

Model data berorientasi objek dikatakan dapat memberi fleksibilitas yang lebih, kemudahan mengubah program dan digunakan luas dalam teknik piranti lunak skala besar. Lebih jauh lagi, pendukung OOP mengklaim bahwa OOP lebih mudah dipelajari bagi pemula dibanding dengan pendekatan sebelumnya, dan pendekatan OOP lebih mudah dikembangkan dan dirawat.

Dengan menggunakan OOP maka dalam melakukan pemecahan suatu masalah tidak harus melihat bagaimana cara menyelesaikan suatu masalah tersebut (terstruktur) tetapi objek-objek apa yang dapat melakukan pemecahan masalah tersebut. Sebagai contoh suatu departemen yang memiliki *manager*, sekretaris, petugas administrasi data dan lainnya. Misal manager tersebut ingin memperoleh data dari bagian administrasi maka *manager* tersebut tidak harus mengambilnya langsung tetapi dapat menyuruh petugas bagian administrasi untuk mengambilnya. Pada kasus tersebut seorang *manager* tidak harus mengetahui

bagaimana cara mengambil data tersebut tetapi manager bisa mendapatkan data tersebut melalui objek petugas administrasi. Jadi untuk menyelesaikan suatu masalah dengan kolaborasi antar objek-objek yang ada karena setiap objek memiliki deskripsi tugasnya sendiri **[12]**.

2.5.1.1. Istilah-Istilah OOP

Istilah OOP sudah sering didengar dalam ilmu pemrograman. Berikut ini istilah-istilah yang sering kali terdengar dalam teknik pemrograman OOP :

1. ***Object***

Untuk mempermudah pemahaman, maka disini akan dijelaskan melalui analogi. Pada dasarnya semua benda yang ada di dunia nyata dapat dianggap sebagai objek. Misalnya rumah, mobil, sepeda, motor, gelas, komputer, meja, sepatu, dll. Setiap objek memiliki atribut sebagai status (*state*) dan tingkah laku sebagai *behavior*. Contoh objek : Motor. Maka *atribute*- nya (*state*) adalah pedal, roda, jeruji, *speedometer*, warna, jumlah roda. Sedangkan tingkah laku (*behavior*) adalah kecepatan menaik, kecepatan menurun dan perpindahan gigi motor. Analogi pemrograman berorientasi objek sama dengan penggambaran pada dunia nyata seperti contoh di atas. Dalam OOP, *state* disimpan pada variabel dan tingkah laku disimpan pada *method*. Dalam bahasa teoretis OOP, Objek berfungsi untuk membungkus data dan fungsi bersama menjadi satu unit dalam sebuah program komputer. Objek merupakan dasar dari modularitas dan struktur dalam sebuah program komputer berorientasi objek.

2. ***Class***

Definisi *class* yaitu *template* untuk membuat objek. *Class* merupakan prototipe atau *blue prints* yang mendefinisikan variabel-variabel dan *method-method* secara umum. Objek merupakan hasil instansiasi dari suatu *class*. Proses pembentukan objek dari suatu kelas disebut sebagai *instantiation*. Objek disebut juga sebagai *instances*. Dalam bahasa teoritis OOP, *class* merupakan kumpulan atas definisi data dan fungsi-fungsi dalam suatu unit untuk suatu tujuan tertentu. Sebagai contoh *'class of dog'* adalah suatu unit yang terdiri atas definisi-definisi data dan fungsi-fungsi yang menunjuk pada berbagai macam perilaku/turunan dari

anjing. Sebuah *class* adalah dasar dari modularitas dan struktur dalam pemrograman berorientasi object. Sebuah *class* secara tipikal sebaiknya dapat dikenali oleh seorang *non-programmer* sekalipun terkait dengan domain permasalahan yang ada dan kode yang terdapat dalam sebuah *class* sebaiknya (relatif) bersifat mandiri dan independen (sebagaimana kode tersebut digunakan jika tidak menggunakan OOP). Dengan modularitas, struktur dari sebuah program akan terkait dengan aspek-aspek dalam masalah yang akan diselesaikan melalui program tersebut. Cara seperti ini akan menyederhanakan pemetaan dari masalah ke sebuah program ataupun sebaliknya.

3. *Attributes*

Atribut adalah data yang membedakan antara objek satu dengan yang lainnya. Contoh Objek : Volcano Robot (*a volcanic exploration vehicle*), mempunyai atribut sebagai berikut :

Status → exploring, moving, returning home

Speed → in miles per hour

Temperature → in Fahrenheit degrees

Dalam *class*, atribut sering disebut sebagai variabel. Atribut dibedakan menjadi dua jenis yaitu *Instance Variable* dan *Class Variable*. *Instance variable* adalah atribut untuk tiap objek dari kelas yang sama. Tiap objek mempunyai dan menyimpan nilai atributnya sendiri. Jadi, tiap objek dari *class* yang sama boleh mempunyai nilai yang sama atau berbeda. *Class Variable* adalah atribut untuk semua objek yang dibuat dari *class* yang sama. Semua objek mempunyai nilai atribut yang sama. Jadi semua objek dari *class* yang sama mempunyai hanya satu nilai yang *value* nya sama.

1. *Behavior*

   Behavior/tingkah laku adalah hal-hal yang bisa dilakukan oleh objek dari suatu *class*. Behavior dapat digunakan untuk mengubah nilai atribut suatu objek, menerima informasi dari objek lain, dan mengirim informasi ke objek lain untuk melakukan suatu tugas (*task*).

   Contoh behavior untuk objek Volcano Robot:

   a. *check current temperature*

b. *begin a survey*

c. *report its current position*

Dalam *class*, behavior disebut juga sebagai *methods. Methods* sendiri adalah serangkaian *statements* dalam suatu *class* yang menangani suatu *task* tertentu. Cara objek berkomunikasi dengan objek yang lain adalah dengan menggunakan *method*.

2. Abstraksi

Abstraksi adalah kemampuan sebuah program untuk melewati aspek informasi yang diproses olehnya, yaitu kemampuan untuk memfokus pada inti. Setiap objek dalam sistem melayani sebagai model dari "pelaku" abstrak yang dapat melakukan kerja, laporan dan perubahan keadaannya dan berkomunikasi dengan objek lainnya dalam sistem, tanpa mengungkapkan bagaimana kelebihan ini diterapkan. Proses, fungsi atau metode dapat juga dibuat abstrak dan beberapa teknik digunakan untuk mengembangkan sebuah pengabstrakan.

2.5.1.2. Konsep-Konsep OOP

Konsep dasar dari Pemrograman Berorientasi Objek Pemrograman orientasi-objek menekankan konsep berikut:

1. Enkapsulasi (*Encapsulation*)

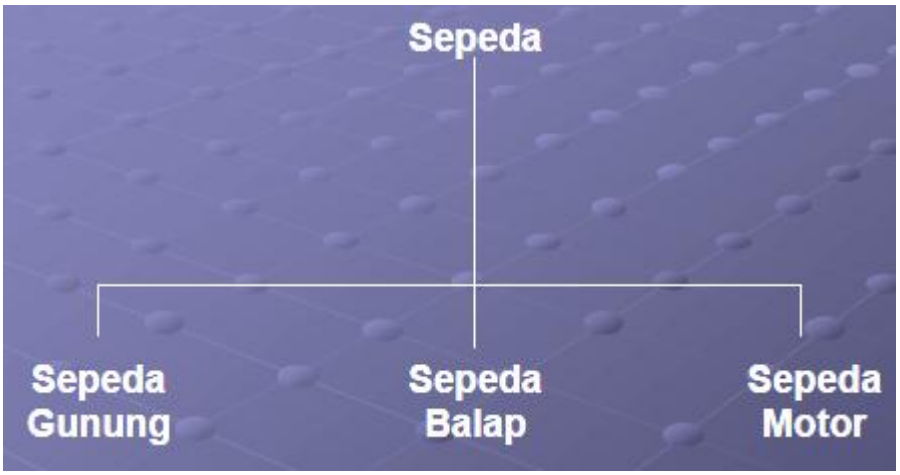
Definisi enkapsulasi adalah Pembungkusan variabel dan *method* dalam sebuah objek yang terlindungi serta menyediakan *interface* untuk mengakses variabel tersebut. Variabel dan method yang dimiliki oleh suatu objek, bisa ditentukan hak aksesnya. Dalam OOP, konsep enkapsulasi sebenarnya merupakan perluasan dari struktur dalam bahasa C. Contoh: jam tangan. Dalam hal ini, penting sekali untuk mengetahui waktu, sedangkan cara jam mencatat waktu dengan baik antara jam bertenaga baterai atau bertenaga gerak tidaklah penting untuk diketahui. Dengan kata lain enkapsulasi berfungsi untuk memastikan pengguna sebuah objek tidak dapat mengganti keadaan dalam/dari sebuah objek dengan cara yang tidak layak, hanya metode dalam objek tersebut yang diberi izin untuk mengakses keadaannya. Setiap objek mengakses *interface* yang menyebutkan bagaimana objek lainnya dapat berinteraksi dengannya. Objek

lainnya tidak akan mengetahui dan tergantung kepada representasi dalam objek tersebut.

2. Pewarisan (Inheritance)

Pewarisan merupakan pewarisan atribut dan method dari sebuah *class* ke *class* lainnya. *Class* yang mewarisi disebut *superclass* dan *Class* yang diwarisi disebut *subclass*. *Subclass* bisa berlaku sebagai *superclass* bagi *class* lainya, disebut sebagai *multilevel inheritance*. Contoh : terdapat *class* sepeda dan sepeda gunung. Sepeda termasuk superclass. Sepeda gunung termasuk *subclass*. Hal ini dikarenakan sepeda gunung memiliki variabel dan *method* yang dimiliki oleh sepeda. Prinsip dasar *inheritance* yaitu persamaan-persamaan yang dimiliki oleh beberapa kelas dapat digabungkan dalam sebuah *class* induk sehingga setiap kelas yang diturunkannya memuat hal-hal yang spesifik untuk kelas yang bersangkutan.

Contoh Pewarisan :



Gambar 2.6 Contoh pewarisan

Keuntungan pewarisan

a. *Subclass* menyediakan *state/behaviour* yang spesifik yang membedakan dengan *superclass*, sehingga memungkinkan *programmer* untuk menggunakan ulang *source code* dari *superclass* yang telah ada.

b. *Programmer* dapat mendefinisikan *superclass* khusus yang bersifat generik, yang disebut *abstract class* (abstraksi), untuk mendefinisikan *class* dengan tingkah laku dan *state* secara umum.

3. Polimorfisme

Polimorfisme adalah kemampuan suatu obyek untuk mempunyai lebih dari satu bentuk. Polimorfisme tidak bergantung kepada pemanggilan subrutin. Metode tertentu yang berhubungan dengan sebuah pengiriman pesan tergantung kepada objek tertentu di mana pesan tersebut dikirim. Contohnya, bila sebuah burung menerima pesan "gerak cepat", maka akan menggerakan sayapnya dan terbang. Bila seekor singa menerima pesan yang sama, maka akan menggerakkan kakinya dan berlari. Keduanya menjawab sebuah pesan yang sama, namun yang sesuai dengan kemampuan hewan tersebut. Ini disebut polimorfisme karena sebuah variabel tunggal dalam program dapat memegang berbagai jenis objek yang berbeda selagi program berjalan dan teks program yang sama dapat memanggil beberapa metode yang berbeda di saat yang berbeda dalam pemanggilan yang sama, hal ini berlawanan dengan bahasa fungsional yang mencapai polimorfisme melalui penggunaan fungsi kelas-pertama.

2.5.2. UML (*Unified Modeling Language*)

Unified Modelling Language (UML) adalah sebuah “bahasa” yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi 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 bahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C. Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan syntax /semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap

bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (*Object-Oriented Design*), Jim Rumbaugh OMT (*Object Modeling Technique*) dan Ivar Jacobson OOSE *(Object-Oriented Software Engineering*).

Sejarah UML sendiri cukup panjang. Sampai era tahun 1990 puluhan metodologi pemodelan berorientasi objek telah bermunculan di dunia. Diantaranya adalah: metodologi booch, metodologi coad, metodologi OOSE, metodologi OMT, metodologi shlaer-mellor, metodologi wirfs-brock, dsb. Masa itu terkenal dengan masa perang metodologi (*method war*) dalam pendesainan berorientasi objek. Masing-masing metodologi membawa notasi sendiri-sendiri, yang mengakibatkan timbul masalah baru apabila dalam bekerjasama dengan group/perusahaan lain yang menggunakan metodologi yang berlainan.

2.5.2.1. Konseps Dasar UML

Dari berbagai penjelasan rumit yang terdapat di dokumen dan buku-buku UML. Sebenarnya konsepsi dasar UML bisa dirangkumkan dalam tabel 2.1.

Tabel 2.1 Konsep Dasar UML

Major Area	View	Diagrams	Main Concepts
structural	static view	class diagram	class, association, generalization, dependency, realization, interface
	use case view	use case diagram	use case, actor, ssociation, extend, include, use casegeneralization
	implementation view	component diagram	component, interface, dependency, realization
dynamic	state machine view	statechart diagram	state, event, transition, action

	actifity view	activity diagram	state, activity, completion transition, fork, join
		sequence diagram	interaction, object, message, activation
	interaction view	colaborating diagram	collaborating, interaction, collaboration rule, message
model management	model management view	class diagram	package, subsystem, model
extensibility	all	all	constraint, stereotype, tagged values

Abstraksi konsep dasar UML yang terdiri dari *structural classification*, *dynamic behavior* dan *model management*, bisa dipahami dengan mudah apabila ketika melihat gambar diatas dari Diagram. *Main concepts* bisa dipandang sebagai *term* yang akan muncul pada saat dalam membuat diagram **[13]**. V*iew* adalah kategori dari diagaram tersebut. Dua hal yang harus di perhatikan:

1. Menguasai pembuatan diagram UML.
2. Menguasai langkah-langkah dalam analisa dan pengembangan dengan UML.

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

a. Use Case Diagram

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

adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. *Use case* diagram dapat sangat membantu dalam menyusun *requirement* sebuah sistem, mengkomunikasikan rancanga dan merancang *test case* untuk semua *feature* yang ada pada sistem. Sebuah *use case* dapat meng- *include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*includ*e akan dipanggil setiap kali *use case* yang meng- *include* dieksekusi secara normal. Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common* . Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviou*r-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.

b. Class Diagram

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

Class memiliki tiga area pokok :

1. Nama (dan *stereotype*)
2. Atribut
3. Metoda

Atribut dan metoda dapat memiliki salah satu sifat berikut :

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

Class dapat merupakan implementasi dari *sebuah interface*, yaitu *class* abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class*. Dengan demikian *interface* mendukung resolusi metoda pada saat *run-time*. Sesuai dengan perkembangan *class model*, *class* dapat dikelompokkan menjadi *package*. Bisa juga dalam membuat diagram yang terdiri atas *package*. Hubungan antar c*lass* terbagi menjadi :

1. Asosiasi, yaitu hubungan statis antar *class.* Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi class lain. Panah *navigability* menunjukkan arah *query* antar *class*.
2. Agregasi, yaitu hubungan yang menyatakan bagian ("terdiri atas..").
3. Pewarisan, yaitu hubungan hirarkis antar *class. Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.
4. Hubungan dinamis, yaitu rangkaian pesan (*message*) yang di- *passing* dari satu *class* kepada *class* lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence diagram.*

c. Statechart Diagram

*Statechar*t diagram menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari stimuli yang diterima. Pada umumnya *statechart* diagram menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart* diagram). Dalam UML, *state* digambarkan berbentuk segiempat dengan sudut membulat dan memiliki nama sesuai kondisinya saat itu. Transisi antar *state* umumnya memiliki kondisi *guard* yang merupakan syarat terjadinya transisi yang bersangkutan, dituliskan dalam kurung siku. *Action* yang dilakukan sebagai akibat dari *event* tertentu dituliskan dengan diawali garis miring. Titik awal dan akhir digambarkan berbentuk lingkaran berwarna penuh dan berwarna setengah.

d. Activity Diagram

Activity diagrams menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing aliran berawal, *decision* yang mungkin terjadi dan bagaimana itu bisa berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. *Activity diagram* merupakan *state* diagram khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity* diagram tidak menggambarkan *behaviour internal* sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalurjalur aktivitas dari *level* atas secara umum. Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas. Sama seperti *state* , standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan *behaviour* pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal. *Activity diagram* dapat dibagi menjadi beberapa object *swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.

e. Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display* dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence* diagram terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai *respons* dari sebuah *event* untuk menghasilkan output tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan. Masing-masing objek, termasuk aktor, memiliki *lifeline vertikal*. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada *fase*

desain berikutnya, *message* akan dipetakan menjadi operasi/metoda dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message*. Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan *icon* khusus untuk objek *boundary*, *controller* dan *persistent entity* .

f. Collaboration Diagram

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

g. Component Diagram

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

h. Deployment Diagram

Deployment/physical diagram menggambarkan detail bagaimana komponen di *deploy* dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, *server* atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi *server* dan hal-hal lain yang bersifat fisikal Sebuah *node* adalah *server, workstation*, atau piranti keras lain yang digunakan untuk men-*deploy* komponen dalam lingkungan sebenarnya. Hubungan antar *node* (misalnya TCP/IP) dan *requirement* dapat juga didefinisikan dalam diagram ini.

2.5.3. Java

Java adalah suatu teknologi di dunia *software* komputer, yang merupakan suatu bahasa pemrograman dan sekaligus suatu *platform*. Sebagai

bahasa pemrograman, Java dikenal sebagai bahasa pemrograman tingkat tinggi. Java mudah dipelajari, terutama bagi *programmer* yang telah mengenal C/C++. Java merupakan bahasa pemrograman berorientasi objek yang merupakan paradigma pemrograman masa depan. Sebagai bahasa pemrograman Java dirancang menjadi handal dan aman. Java juga dirancang agar dapat dijalankan di semua *platform* dan juga dirancang untuk menghasilkan aplikasi – aplikasi dengan performansi yang terbaik, seperti aplikasi *database* Oracle 8i/9i yang *core-nya* dibangun menggunakan bahasa pemrograman Java. Sedangkan Java bersifat *neutral architecture*, karena Java *Compile*r yang digunakan untuk mengkompilasi kode program Java dirancang untuk menghasilkan kode yang netral terhadap semua arsitektur perangkat keras yang disebut sebagai *Java Bytecode.*



Gambar 2.7 Logo Java

Sebagai sebuah platform, Java terdiri atas dua bagian utama, yaitu:

a. *Java Virtual Machine* (JVM).
b. *Java Application Programming Interface* (Java API).

Sun membagi arsitektur Java membagi tiga bagian, yaitu:

a. *Enterprise Java* (J2EE) untuk aplikasi berbasis web, aplikasi sistem tersebar dengan beraneka ragam klien dengan kompleksitas yang tinggi. Merupakan superset dari Standar Java.
b. *Standar Java* (J2SE), ini adalah yang biasa dikenal sebagai bahasa Java.

c. *Micro Java* (J2ME) merupakan subset dari J2SE dan salah satu aplikasinya yang banyak dipakai adalah untuk *wireless device/mobiledevice.*

2.5.4. Unity 4.0.

Unity 3D adalah sebuah *game engine* yang berbasis *cross-platform*. Unity dapat digunakan untuk membuat sebuah *game* yang bisa digunakan pada perangkat komputer, ponsel pintar android, iPhone, PS3, dan bahkan X-BOX. Unity adalah sebuah sebuah tool yang terintegrasi untuk membuat *game*, arsitektur bangunan dan simulasi. Unity bisa *untuk games* PC dan *games Online*. Untuk *games online* diperlukan sebuah *plugin*, yaitu Unity *Web Player*, sama halnya dengan Flash Player pada *Browser*. Unity tidak dirancang untuk proses desain atau modelling, dikarenakan unity bukan tool untuk mendesain. Jika ingin mendesain, pergunakan 3D editor lain seperti 3dsmax atau Blender. Banyak hal yang bisa dilakukan dengan unity, ada fitur *audio reverb zone, particle effect, dan Sky Box* untuk menambahkan langit.



Gambar 2.8 Logo Unity

Fitur *scripting* yang disediakan, mendukung 3 bahasa pemrograman, JavaScript, C# dan Boo. *Flexible and Easy Moving, rotating* dan *scaling objects* hanya perlu sebaris kode. Begitu juga dengan *Duplicating*, *removing*, dan *changing properties*. *Visual Properties Variables* yang di definisikan dengan *scripts* ditampilkan pada *Editor*. Bisa digeser, di *drag* and *drop*, bisa memilih warna dengan *color picker*. Berbasis .NET, artinya penjalanan program dilakukan dengan *Open Source* NET *platform*, Mono.

Unity menggunakan sebuah konsep yang disebut *Parenting*, ini digunakan untuk membuat sebuah *game object* menjadi anak dari g*ame object* yang lain. Tarik sebuah *game object* dan pindahkan tepat di atas tulisan *game*

*objec*t yang akan dijadikan parent dalam hierarchy. *Game object* yang terdapat dalam sebuah *game object* lainnya akan mengikuti perpindahan dan perputaran ketika *game object parent* mengalami perubahan posisi.

2.5.5. Adobe Photoshop CS5

Adobe Photoshop adalah sebuah program yang digunakan untuk melakukan perubahan-perubahan (manipulasi) pada gambar, yang dapat menghasilkan gambar digital dengan kualitas tinggi. Adobe Photoshop juga dapat melakukan perubahan-perubahan dari gambar atau foto hasil *scanner*. Misalnya dengan memperbesar atau memperkecil gambar yang sudah ada, melakukan rotasi (perputaran), merubah warna, memberi efek-efek khusus seperti yang terdapat pada menu *filter*. Hal yang utama dalam Adobe Photoshop adalah penggunaan istilah :

1. ***Image***

   Gambar yang tampak di monitor komputer sebagai salah satu *file* dokumen Adobe Photoshop setelah proses *scanning*. Komputer dengan *software* windows terbaru umumnya bekerja dengan jenis *image* .

   a. *Grafis pixel*

      Ideal untuk memproduksi foto dan *image* lain yang berisi sejumlah besar detail yang tidak terstruktur. *Pixel* akan menyimpan informasi suatu bentuk (lingkaran, kotak, gambar) dalam titik-titik kecil dengan kerapatan tertentu. Biasanya image *pixel* kalau diperbesar akan menimbulkan efek buruk pada gambar (gambar pecah). Karena titik-titik kecil pembentuknya akan terlihat membesar dan ruang kosong antara titik-titik tersebut akan terlihat jelas. Warna akan terurai (pecah).

   b. *Grafis Vektor*

      Merupakan deskripsi matematika dari bentuk yang dapat dibuat dari garis lurus dan kurva. *Grafis vektor* sangat kompak dan *portable* karena dapat diperbesar dan diperkecil tanpa menimbulkan efek buruk pada gambar (gambar pecah). Contohnya apabila menggunakan

software pengolah vektor atau ilustrasi seperti Ilustrator, Freehand dan Corel Draw. Program pengolah vektor ini sering disebut *resolution independent*.

2. ***Pixe*l (*picture element*)**

Titik terkecil yang menyusun image. Kerapatan dari *pixel* dikenal dengan istilah Resolusi Image. Satuannya ada 2 macam yaitu pixel/inch atau pixel/cm. Contoh gambar dengan 72 pixel/inch artinya dalam 1 inch terdapat 72 pixel.

2.5.6. Autodesk 3ds Max

3D Studio Max (kadangkala disebut 3ds Max atau hanya MAX) adalah sebuah perangkat lunak grafik vektor 3-dimensi dan animasi, ditulis oleh Autodesk Media & Entertainment (dulunya dikenal sebagai Discreet and Kinetix. Perangkat lunak ini dikembangkan dari pendahulunya 3D Studio fo DOS, tetapi untuk platform Win32. Kinetix kemudian bergabung dengan akuisisi terakhir Autodesk, Discreet Logic. Versi terbaru 3Ds Max pada Juli 2005 adalah 7. 3Ds Max Autodesk 8 diperkirakan akan tersedia pada akhir tahun. Hal ini telah diumumkan oleh Discreet di Siggraph 2005 **[14]**.



Gambar 2.9 Logo Autodesk 3ds Max

2.5.7. Blender

Blender adalah program 3D dan animasi yang bersifat *open source*, bebas untuk dikembangkan oleh penggunanya dan dapat didistribusikan kembali dan bersifat Legal. Blender memiliki *video compositor* dan *intergrated game engine*. Karya yang dihasilkan tidak ada sifat *royalt* kepada *developer* dan

dapat dipublikasikan baik free maupun untuk dikomersilkan. Blender merupakan salah satu program *Modeling* 3D dan *Animation*, tapi Blender mempunyai kelebihan sendiri dibandingkan program *modeling* 3D lainnya. Kelebihan yang dimiliki Blender adalah dapat membuat *game* tanpa menggunakan program tambahan lainnya, karena Blender sudah memiliki *Game Engine* sendiri dan menggunakan Python sebagai bahasa pemograman yang lebih mudah ketimbang menggunakan C++,C, dll.

Blender menggunakan OpenGL sebagai render grafiknya yang dapat digunakan pada berbagai macam sistem operasi seperti Windows, Linux dan Mac OS X. Saat ini Blender sudah mengeluarkan versi yang terbarunya, yaitu Versi 2.49 yang lebih ditujukan untuk pembuat *game*. Karena Versi ini memiliki fitur-fitur baru yang dirancang untuk membuat tampilan *game* yang lebih realistis dari pada versi sebelumnya.



Gambar 2.10 Logo Blender

2.6. Skala Likert

Skala Likert digunakan untuk mengukur sikap, pendapat dan persepsi seseorang atau sekelompok orang tentang fenomena sosial, dengan skala Likert maka variabel yang akan diukur dujabarkan menjadi indikator variabel. Kemudian indikator tersebut dijadikan sebagai titik tolak untuk menyusun item-item instrumen yang dapat berupa pernyataan atau pertanyaan.

Jawaban setiap item instrumen Skala Likert mempunyai gradasi dari sangat positif sampai negatif yang untuk keperluan analisis kuantitatif **[15]**, setiap jawaban dapat diberi skor seperti berikut :

a. Sangat setuju diberi skor 5
b. Setuju diberi skor 4

c. Ragu-ragu diberi skor 3
d. Tidak setuju diberi skor 2
e. Sangat tidak setuju diberi skor 1

Berdasarkan data tersebut, dapat dicari prosentase masing-masing jawaban dengan menggunakan rumus :

$$Y = P/Q*100\% \qquad \ldots(2.2)$$

Keterangan :
Y = Prosentase
P = Banyaknya Jawaban Responden Tiap Soal
Q = Jumlah Responden

2.7. Statistik *Non*-Parametrik

Statistik *non*-parametrik disebut juga statistik bebas sebaran. Statistik non-parametrik tidak mensyaratkan bentuk sebaran parameter populasi. Statistik non-parametrik dapat digunakan pada data yang memiliki sebaran normal atau tidak **[16]**. Contoh metode statistik *non*-parametrik diantaranya adalah :

a. Chi-square test
b. Binomial
c. Runs
d. One Sample Kolmogorov-Smirnov
e. Two Independent Samples Test
f. K Independent Samples Test
g. Two Related Samples Test
h. K Related Samples Test

Uji *non*-parametrik lebih mudah dihitung dan dimengerti terutama karena datanya berupa urutan (*order)* atau peringkat (*rank)*. Namun uji ini kurang akurat dan efisien bila dibanding dengan metode parametrik. Dalam penelitian ini pengolahan data menggunakan SPSS (*Statistical Package* for *Social Science*), aplikasi ini telah diakuisisi oleh IBM pada akhir 2009. SPSS merupakan *software*

aplikasi statistik yang sangat populer, baik bagi praktisi yang sedang melakukan riset ataupun bagi mahasiswa yang menyelesaikan tugas akhir **[16]**. Ciri-ciri statistik non-parametrik adalah :

a. Data tidak berdistribusi normal.
b. Umumnya data berskala nominal dan ordinal.
c. Umumnya dilakukan pada penelitian sosial.
d. Umumnya jumlah sampel kecil.

Pengujian Hipotesis terbagi menjadi dua yaitu 1 arah (*one tailed*) dan 2 arah (*two tailed*). Hipotesis 1 arah digunakan untuk menguji suatu hal yang sudah jelas akan lebih besar atau lebih kecil dari hipotesis awal. Sedangkan Hipotesis 2 arah digunakan untuk menguji suatu hal (hipotesis awal) pada suatu titik tertentu, dimana kemungkinan hipotesis tandingannya bisa lebih besar maupun lebih kecil dari titik tersebut.

Two Related Samples Test merupakan teknik yang akan digunakan dalam penelitian ini. Teknik ini melibatkan dua pengukuran pada subjek yang sama terhadap suatu pengaruh atau perlakuan tertentu. Pengukuran pertama dilakukan sebelum diberi perlakuan tertentu dan pengukuran kedua dilakukan sesudahnya.

Ada tiga uji populer *Two Related Samples Test*, yaitu wilcoxon, sign, dan McNemar. Uji Wilcoxon merupakan penyempurnaan uji sign. Uji Sign menguji perbedaan pasangan pengamatan hanya sebatas tanda positif dan negatif, tidak berdasar pada nilai perbedaan, sedangkan uji Wilcoxon memperhatikan besarnya perbedaan tersebut. Uji McNemar menguji dua variabel berhubungan dikotomi yang bertipe data nominal dan ordinal. *Wilcoxon Signed Rank test* ini digunakan hanya untuk data bertipe interval atau rasio, namun datanya tidak mengikuti distribusi normal.