BAB 2
TINJAUAN PUSTAKA

2.1 Multimedia

Multimedia adalah media yang menggabungkan dua unsur atau lebih media yang terdiri dari teks, grafik, gambar, *audio, video* dan animasi secara *terintegrasi*[4].

Menurut beberapa ahli multimedia memiliki beberapa definisi yaitu :

1. Rosch

   Multimedia adalah kombinasi dari komputer dan video.

2. McComick

   Multimedia adalah kombinasi dari komputer dari tiga elemen yaitu suara, gambar dan teks.

3. Turban dan kawan-kawan

   Multimedia adalah kombinasi dari paling sedikit dua media *input* dan *output*. Media ini dapat berupa *audio*, 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. Hofstetter

   Multimedia adalah pemanfaatan komputer untuk membuat dan menggabungkan teks, grafik, *audio*, *video* dengan menggunakan *tool* yang memungkinkan pemakai berinteraksi, berkreasi dan berkomunikasi. Dalam definisi ini terdapat empat komponen multimedia yaitu :

   1) Harus ada peralatan elektronik yang dapat mengkoordinasikan apa yang dapat dilihat dan didengar dan yang dapat berinteraksi dengan pengguna.
   2) Harus ada *link* yang dapat menghubungkan pengguna dengan informasi.

3) Harus ada alat navigasi yang dapat memandu pengguna untuk menjelajahi jaringan informasi yang saling terhubung.
4) Harus dapat menyediakan tempat kepada pengguna untuk mengumpulkan, memproses dan mengkomunikasikan informasi dan ide dari pengguna itu sendiri.

Dengan penggunaan multimedia, penyampaian informasi akan menjadi lebih menarik dan mempermudah pengguna dalam mendapatkan informasi tersebut. Seperti yang disebutkan dalam laporan hasil penelitian yang dikeluarkan oleh *Computer Technology Research* (*Hofstetter*) bahwa seseorang hanya akan mendapatkan 20% dari apa yang di lihat dan 30% dari yang di dengar. Sedangkan melalui multimedia akan mendapatkan 50% dari apa yang di lihat dan dengar, sampai 80% dari apa yang di lihat, dengar dan berinteraksi pada waktu yang bersamaan. Keuntungan dari penggunaan multimedia dalam suatu aplikasi adalah:

1) Meningkatkan aliran gagasan dan informasi.
2) Merupakan cara yang kaya untuk mengkomunikasikan sesuatu.
3) Mendorong partisipasi, keterlibatan, dan eksplorasi pengguna.
4) Menstimulasi panca indera.
5) Memberikan kemudahan pemakaian terutama bagi pengguna awam.

2.2 *Game*

Game merupakan salah satu media hiburan yang paling popular untuk semua kalangan usia. Sejak pertama kali ditemukan sampai saat sekarang, teknologi *game* telah mengalami kemajuan yang terbilang pesat. Hal ini ditandai dengan berkembangnya jenis *game*, produk, alat dan jenis interaksi *game* dengan penggunaan yang semakin beragam bentuknya.

2.2.1 Pengertian *Game*

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

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

Berdasarkan representasinya, *game* dapat dibedakan menjadi 2 jenis yaitu *game* 2 dimensi (2D) dan 3 dimensi (3D). *Game* 2D adalah *game* yang secara matematis hanya melibatkan 2 elemen koordinat kartesius yaitu x dan y, sehingga konsep kamera pada *game* 2D hanya menentukan gambar pada *game* yang dapat dilihat oleh pemain. Sedangkan *game* 3D adalah *game* yang selain melibatkan elemen x dan y juga melibatkan elemen z pada perhitungannya sehingga konsep kamera pada *game* 3D benar-benar menyerupai konsep kamera pada kehidupan nyata[5].

2.2.2 Game 3D

Pada saat kemampuan proses pada komputer semakin cepat, para developer juga senantiasa menciptakan *game* yang semakin canggih sehingga muncul engine dengan grafik 3D (3 dimensi). Dalam *game* 3D pengguna dapat melihat sudut pandang tiga ratus enam puluh derajat pemandangan yang ada di dalam *game* , di dalam *game* 3D merepresentasikan objek dalam bentuk 3 dimensi sehingga objek akan terlihat lebih nyata seperti dalam kehidupan nyata. Game bertipe 3 dimensi merupakan game dengan grafis yang baik dalam penggambaran secara realita, kebanyakan game-game ini memiliki perpindahan kamera (angle) hingga 360 derajat sehingga dapat melihat secara keseluruhan dunia game tersebut.

2.2.3 3D Grafis

Grafik komputer 3 dimensi biasa disebut 3D atau adalah bentuk dari benda yang memiliki panjang, lebar, dan tinggi. Grafik 3 Dimensi merupakan teknik penggambaran yang berpatokan pada titik koordinat sumbu x (datar), sumbu y (tegak), dan sumbu z (miring). Representasi dari data geometrik 3 dimensi sebagai hasil dari pemrosesan dan pemberian efek cahaya terhadap grafika komputer 2D.

Tiga Dimensi, biasanya digunakan dalam penanganan grafis. 3D secara umum merujuk pada kemampuan dari sebuah *video card* (*link*). Saat ini *video card* menggunakan variasi dari instruksi-instruksi yang ditanamkan dalam *video card* itu sendiri (bukan berasal dari *software*) untuk mencapai hasil grafik yang lebih realistis dalam memainkan *game* komputer.

Grafik 3D merupakan perkembangan dari grafik 2D. Didalam grafika komputer, 3D merupakan bentuk grafik yang menggunakan representasi data geometri tiga dimensi. Suatu objek rangka 3D apabila disinari dari arah tertentu akan membentuk bayangan pada permukaan gambar. Proses pembuatan grafik komputer 3D dapat dibagi ke dalam tiga fase, yaitu 3D modeling yang mendeskripsikan bentuk dari sebuah objek, *layout* dan *animation* yang mendeskripsikan gerakan dan tata letak sebuah objek, dan 3D rendering yang memproduksi *image* dari objek tersebut. Istilah atau Pengertian Grafik 3D adalah sebuah gambar, garis, lengkungan dan sebagainya yang memiliki titik-titik yang menghubungkan menjadi sebuah bentuk 3D Di dalam dunia *game*, 3D secara umum merujuk pada kemampuan dari sebuah *video card* (*link*). Saat ini *video card* menggunakan variasi dari instruksi-instruksi yang ditanamkan dalam *video card* itu sendiri (bukan berasal dari *software*) untuk mencapai hasil grafik yang lebih realistis dalam memainkan *game* komputer[6].

2.2.4 Jenis-jenis *Game*

Berikut ini beberapa jenis *game* berdasarkan cara pembuatannya, cara pemasarannya dan mesin yang menjalankannya. Jenis-jenis *game* tersebut adalah[1] :

1) *Game* PC

   Game PC adalah *game* yang dimainkan pada PC (*Personal Computer*) yang memiliki kelebihan yaitu tampilan antarmuka yang baik untuk *input* maupun *output*. *Output* visual berkualitas tinggi karena layar komputer biasanya memiliki resolusi yang jauh lebih tinggi dibandingkan dengan layar televisi biasa. Kekurangannya adalah spesifikasi komputer yang sangat bervariasi antar satu komputer dengan komputer yang lainnya

menyebabkan beberapa *game* dapat ditampilkan dengan baik pada satu komputer tetapi tidak berjalan dengan baik pada komputer yang lainnya.

2) *Game Console*

Game console adalah *game* yang dijalankan pada suatu mesin spesifik yang biasanya tersedia di rumah seperti Xbox, Nintendo, Wii dan lain-lain.

3) *Game Arcade*

Game arcade adalah *game* yang dijalankan pada mesin dengan *input* dan *output audio* visual yang telah terintegrasi dan tersedia ditempat-tempat umum.

4) *Game Online*

Game online adalah *game* yang hanya dapat dimainkan secara *online* melalui LAN atau *internet*.

2.2.5 *First person shooter*

Game ber*genre first person shooter* (FPS) adalah *genre game* perang dengan senjata api yang menggunakan sudut pandang orang pertama dengan tampilan layar yang mensimulasikan apa yang dilihat melalui mata karakter yang dimainkan, sehingga pengguna bisa merasakan langsung seperti berada di dalam sebuah *game*[1]. Ciri khas game ini adalah pengguna senjata jarak jauh, *game* jenis ini memiliki *gameplay* yang cukup mudah dimengerti dan dimainkan, karena *game* ini tidak membutuhkan keterampilan, pemikiran strategi, bahkan khayalan dan imajinasi dari pemain. Beberapa contoh *game* dengan *genre first Person Shooter* adalah *game Counter Strike*, *Point Blank*, *Crysis*, dan *Far Cry*.

2.2.6 Unsur *Game*

Dalam sebuah *game* terdapat unsur-unsur yang melengkapinya. Berikut beberapa unsur dalam suatu *game* yaitu[2] :

1. Warna

Warna mempunyai kemampuan untuk membuat orang tanggap terhadap semua yang dilihat karena tidak ada sesuatu hal bermakna tanpa warna. Mata manusia tertarik oleh warna pada suatu *level* karena warna dari objek diterima sebelum detail-detail dipisahkan oleh bentuk-bentuk

dan garisnya. Warna merah memiliki panjang gelombang yang terpanjang, biru memiliki panjang gelombang yang pendek sedangkan hijau memiliki panjang gelombang menengah. Pada anak-anak cenderung tertarik pada warna-warna yang cerah dan mencolok. Warna-warna yang cerah terutama warna *primer* (merah, kuning, biru) dan warna *sekunder* (orange, ungu, hijau). Contoh warna *sekunder* dan *primer* dapat dilihat pada gambar 2.1.



Gambar 2.1 Warna *Primer* dan *Sekunder*

2. Komposisi

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

3. Bentuk dasar

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



Gambar 2.2 Bentuk Dasar

4. *Tipografi*

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

5. *Audio*

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

2.2.7 Klasifikasi Permainan

Berikut ini adalah klasifikasi permainan berdasarkan pembatasan usia dari permainannya, menurut ESRB (*Entering software Ratting Board*) adalah sebagai berikut :

1. *Early Childhood (3+)* memiliki isi yang mungkin cocok untuk usia 3 keatas. Tidak mengandung materi yang tidak pantas seperti adegan kekerasan.
2. *Everyone (6+)* memiliki konten yang mungkin cocok untuk usia 6 keatas. Judul-judul dalam kategori ini mungkin berisi sedikit kartun, fantasi atau kekerasan ringan dan tidak jarang menggunakan bahasa yang ringan.
3. *Everyone (10+)* memiliki konten yang mungkin cocok untuk usia 10 keatas. Judul-judul dalam kategori ini mungkin berisi lebih banyak kartun, fantasi, atau kekerasan ringan, bahasa ringan dan juga minimal tema *sugestif*.
4. *Teen (13+)* memiliki konten yang mungkin cocok untuk usia 13 keatas. Judul-judul dalam kategori ini mungkin mengandung kekerasan, tema

sugestif, humor kasar, sedikit darah, simulasi perjudian, dan tidak jarang menggunakan bahasa yang kuat.

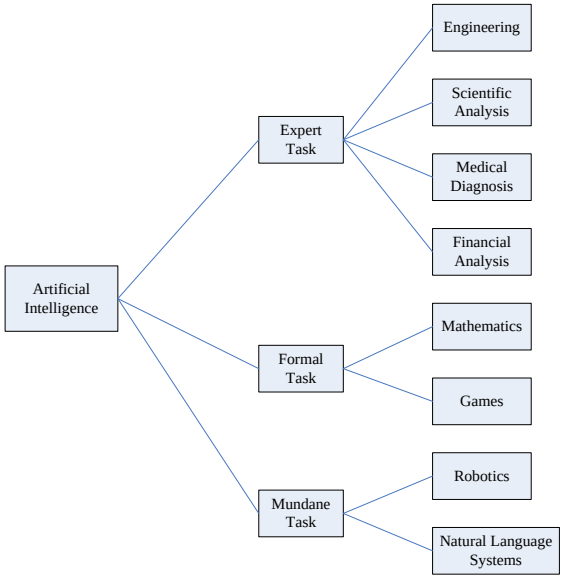
5. *Mature (17+)* memiliki konten yang mungkin cocok untuk orang usia 17 dan lebih tua. Judul-judul dalam kategori ini mungkin berisi *inten s*kekerasan, darah, konten seksual dan bahasa yang kuat.
6. *Adults Only (18+)* memiliki konten yang hanya boleh dimainkan oleh orang-orang 18 tahun keatas. Judul-judul dalam kategori ini mungkin termasuk adegan yang *intens,* kekerasan dan juga mengandung konten seksual.

2.3 Kecerdasan Buatan

Kecerdasan buatan atau *Artificial Intelligence* (AI) merupakan cabang dari ilmu komputer yang berhubungan dengan pengautomatisan tingkah laku cerdas. Kecerdasan buatan didasarkan pada teori suara (*sound theoretical*) dan prinsip-prinsip aplikasi dari bidangnya. Prinsip-prinsip ini meliputi struktur data yang digunakan dalam representasi pengetahuan, algoritama yang diperlukan untuk mengaplikasikan pengetahuan tersebut serta bahasa dan teknik pemrograman yang digunakan dalam mengimplementasikannya[7]. Berdasarkan sudut pandang, AI dapat dipandang sebagai berikut :

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

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



Gambar 2.3 Bidang-bidang tugas (*task domains*) dari AI

Aplikasi penggunaan AI dapat dibagi ke dalam tiga kelompok yaitu :

1) *Expert Task*

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

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

2) *Formal Task*

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

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

3) *Mundane Task*

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

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

Aplikasi kecerdasan buatan memiliki dua bagian utama yaitu :

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

2.4 Teknik Dasar Pencarian

Ada beberapa keuntungan dari kecerdasan buatan dibandingkan dengan kecerdasan alamiah yaitu :

1. Lebih permanen.
2. Memberikan kemudahan dalam duplikasi dan penyebaran.
3. Relatif lebih murah dari kecerdasan alamiah.
4. Konsisten dan teliti.
5. Dapat didokumemtasikan.
6. Dapat mengerjakan beberapa *task* dengan lebih cepat dan lebih baik dibanding manusia.

2.4.1 Algoritma Pencarian

Permasalahan pencarian dapat diselesaikan dengan beberapa metode yaitu[7] :

1. Metode pencarian yang pertama adalah metode sederhana yang hanya berusaha mencari kemungkinan penyelesaian yang disebut juga pencarian buta.
2. Metode yang lebih kompleks yang akan mencari jarak terpendek. Metode ini adalah *British Museum Procedure, Branch and Bound, Dynamic Programming, Best First Search, Greedy Search*, A* (A Star) dan *Hill*

Climbing Search. Metode-metode ini digunakan pada saat perjalanan untuk mencari kemungkinan menjadi perhitungan.

Metode pencarian sangat penting untuk menyelesaikan permasalahan karena setiap *state* atau keadaan menggambarkan langkah-langkah untuk menyelesaikan permasalahan. Dalam sebuah permainan, metode pencarian akan menentukan apa yang harus dilakukan dimana setiap state menggambarkan kemungkinan posisi pada suatu state. Metode pencarian adalah bagian dari kesimpulan dimana setiap state menggambarkan *hipotesis* dalam sebuah rangkaian *deduktif*. 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*).

Beberapa contoh algoritma pencarian yang menggunakan metode *heuristic* search adalah : *Best First Search, Greedy Search,* A* (A *Star*) *Search,* dan *Hill Climbing Search*.

2.4.2 Metode Pencarian Heuristik

Kata *heuristik* berasal dari kata kerja bahasa Yunani, *heuriskein,* yang berarti ‘mencari’ atau ‘menemukan. Dalam dunia pemograman, sebagaian orang menggunakan kata heuristik sebagai lawan kata dari algoritmik, dimana kata *heuristik* ini diartikan sebagai ‘suatu proses yang mungkin dapat menyelesaikan suatu masalah tetapi tidak ada jaminan bahwa solusi yang dicari selalu dapat ditemukan’. Di dalam mempelajari metode-metode pencarian ini, kata *heuristik* diartikan sebagai suatu fungsi yang memberikan suatu nilai berupa biaya perkiraan (estimasi) dari suatu solusi[7].

2.4.2.1 *Manhattan City Block Distance*

Dimana perhitungan jumlah *node* hanya yang bergerak secara *vertical* dan *horizontal* menuju tujuannya atau sama dengan total langkah yang diperlukan masing-masing kotak menuju posisi yang sebenarnya *goal satate* serta mengabaikan penghalang atau *collision*, yang kemudian nilainya dikalikan dengan 10. Atau di rumuskan dengan :

$$H = 10*(\text{abs}(current\text{X} - target\text{X}) + \text{abs}(current\text{Y} - target\text{Y})). \quad (2.1)$$

2.4.3 Pencarian Buta

Pencarian buta (*Blind Search*) adalah pencarian solusi tanpa adanya informasi yang dapat mengarahkan pencarian untuk mencapai *goal state* dari *current state*. Informasi yang ada hanyalah definisi *goal state* itu sendiri sehingga algoritma dapat mengenali *goal state*. Apabila tidak ada informasi maka pencarian buta dalam kerjanya akan memeriksa *node-node* secara tidak terarah dan kurang efisien untuk kebanyakan kasus karena banyanknya *node* yang dikembangkan. Beberapa contoh algoritma yang termasuk *blind search* adalah *Breadth First Search, Uniform Cost Search, Depth First Search, Depth Limited Search, Interative Deepening Search* dan *Bidirectional Search*[7].

2.4.4 Pencarian Terbimbing

Pencarian terbimbing (*Heuristic Search*) mempunyai informasi tentang biaya untuk mencapai *goal state* dari *current state*. Pencarian terbimbing dapat melakukan pertimbangan untuk mengembangkan atau memeriksa node-node yang mengarah ke *goal state*. Pencarian terbimbing untuk menghitung *cost* ke *goal state* digunakan fungsi *heuristic*. Fungsi *heuristic* berbeda dari pada algoritma dimana *heuristic* lebih merupakan perkiraan untuk membantu *algoritma* dan tidak harus *valid* setiap waktu. Beberapa contoh *algoritma* pencarian yang mengguanakan metode *heuristic search* adalah *Best First Search, Greedy Search*, A* (A Star) dan *Hill Climbing Search*[7].

2.5 AI yang Digunakan

2.5.1 Algoritma A* (*A star*)

Algoritma A* (A Star) merupakan perbaikan dari metode BFS dengan memodifikasi fungsi *heuristic*nya. A* (A Star) akan meminimumkan total biaya lintasan. Pada kondisi yang tepat, A* akan memberikan solusi yang terbaik dalam waktu yang optimal. Pencarian menggunakan algoritma A* mempunyai prinsip yang sama dengan algoritma BFS hanya saja dengan dua faktor tambahan yaitu :

1. Setiap sisi mempunyai *cost* yang berbeda-beda, seberapa *cost* untuk pergi dari satu simpul ke simpul lain.
2. *Cost* dari setiap simpul ke simpul tujuan bisa diperkirakan. Ini membantu pencarian sehingga lebih kecil kemungkinan mencari ke arah yang salah.

Cost untuk setiap simpul tidak harus berupa jarak. *Cost* bisa saja berupa waktu bila ingin mencari jalan dengan waktu tercepat untuk dilalui. Algoritma A* bekerja dengan prinsip yang hampir sama kecuali dengan dua perbedaan yaitu :

1. Simpul-simpul di *list* terbuka diurutkan oleh *cost* keseluruhan dari simpul awal ke simpul tujuan, dari *cost* terkecil sampai *cost* terbesar. *Cost* keseluruhan dihitung dari *cost* dari simpul awal ke simpul sekarang (*current node*) ditambah *cost* perkiraan menuju simpul tujuan.
2. Simpul di *list* tertutup bisa dimasukkan ke *list* terbuka bila jalan terpendek menuju simpul tersebut ditemukan.

Cost antara simpul adalah jaraknya dan perkiraan *cost* dari suatu simpul ke simpul tujuan adalah penjumlahan jarak dari simpul tersebut ke simpul tujuan. Untuk lebih mudah dimengerti, berikut rumusnya:

$$f(n) = g(n) + h(n) \quad (2.2)$$

keterangan :

1. f(n) = fungsi evaluasi
2. g(n) = biaya yang sudah dikeluarkan dari keadaan awal sampai keadaan n
3. h(n) = estimasi biaya untuk sampai pada suatu tujuan mulai dari n

Node dengan nilai terendah merupakan solusi terbaik untuk diperiksa pertama kali pada g(n) + h(n). Dengan fungsi *heuristic* yang memenuhi kondisi tersebut maka pencarian dengan algoritma A* dapat optimal. Keoptimalan dari A* cukup langsung dinilai optimal jika h(n) adalah *admissible heuristic* yaitu nilai h(n) tidak akan memberikan penilaian lebih pada *cost* untuk mencapai tujuan. Salah satu contoh dari *admissible heuristic* adalah jarak dengan menarik garis lurus karena jarak terdekat dari dua titik adalah dengan menarik garis lurus[7].

Beberapa terminologi dasar yang terdapat pada algoritma ini adalah *starting point, current node*, simpul, *neighbor node, open list*, *closed list*, harga (*cost*), *walkability*, target *point*.

1. Simpul awal adalah sebuah terminologi untuk posisi awal sebuah benda.
2. *Current node* adalah simpul yang sedang dijalankan dalam algoritma pencarian jalan terpendek.

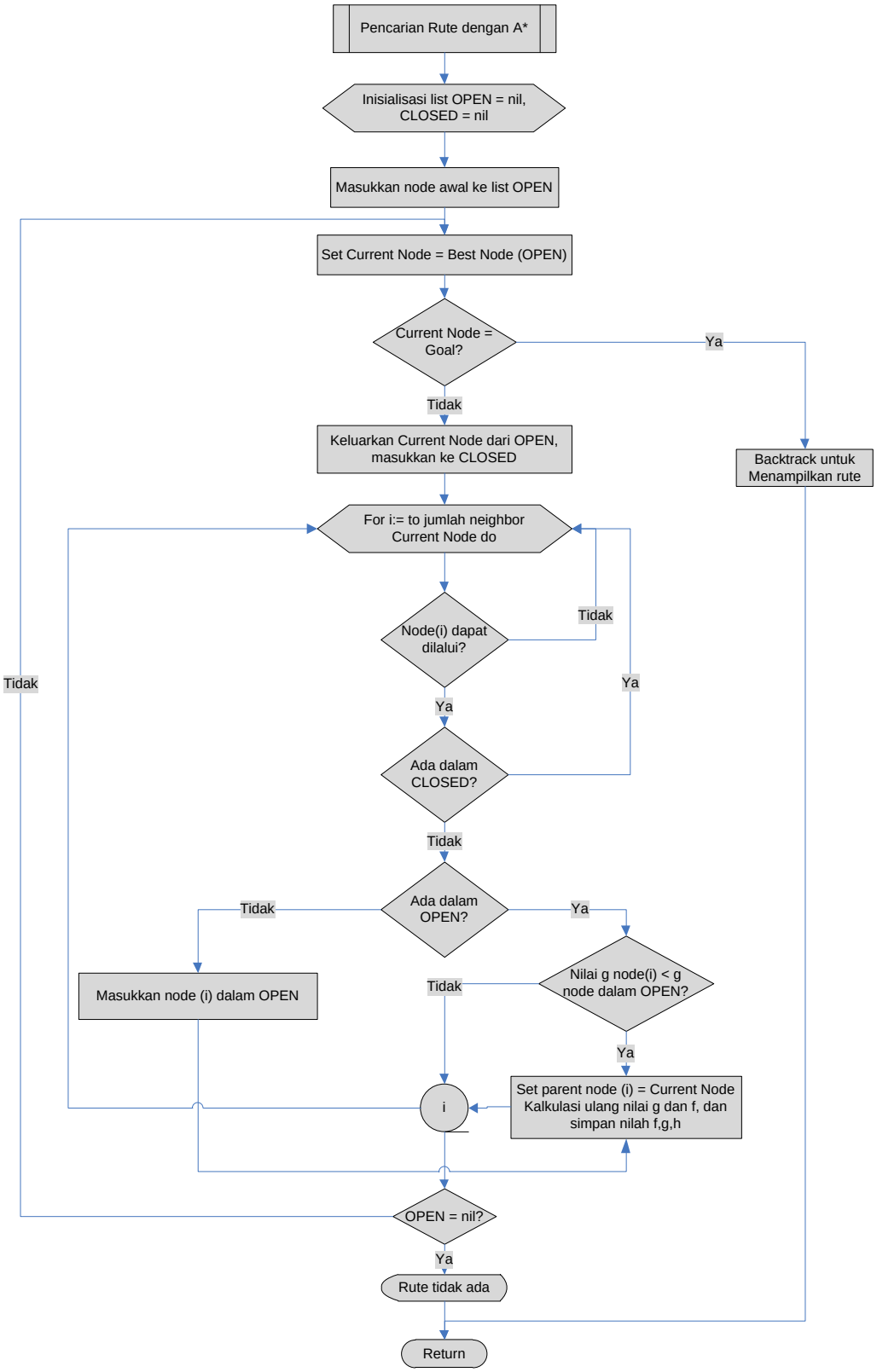
3. Simpul adalah petak-petak kecil sebagai representasi dari area *pathfinding*. Bentuknya dapat berupa persegi, lingkaran, maupun segitiga.
4. *Neighbor node* adalah simpul-simpul yang bertetangga dengan *current node*.
5. *Open List* adalah tempat menyimpan data simpul yang mungkin diakses dari *starting point* maupun simpul yang sedang dijalankan.
6. *Closed List* adalah tempat menyimpan data simpul sebelum *current node* yang juga merupakan bagian dari jalur terpendek yang telah berhasil didapatkan.
7. *Parent* adalah tempat menyimpan data ketetanggaan dari suatu simpul, misalnya y *parent* x artinya *neighbor node* y dari *current node* x.
8. Harga (F) adalah nilai yang diperoleh dari penjumlahan nilai G, jumlah nilai tiap simpul dalam jalur terpendek dari *starting point* ke *current node*, dan H, jumlah nilai perkiraan dari sebuah simpul ke *target point*.
9. *Target point* yaitu simpul yang dituju.
10. *Walkability* adalah sebuah atribut yang menyatakan apakah sebuah simpul dapat atau tidak dapat dilalui oleh *current node*.

Algoritma A* secara ringkas langkah demi langkahnya adalah sebagai berikut:

1. Tambahkan simpul awal ke dalam *open list*.
2. Ulangi langkah berikut sampai pencarian berakhir:
    1) Carilah simpul n dengan biaya F(n) paling rendah, dalam *open* list. Simpul dengan biaya F terendah kemudian disebut *current node*.
    2) Keluarkan *current node* dari *open list* dan masukkan ke dalam *closed list*.
    3) Untuk setiap 8 simpul (*neighbor node*) dari *current node* lakukan langkah berikut:
        a. Jika sudah terdapat dalam *closed list* atau tidak *walkable*, maka abaikan, jika tidak lanjutkan.

b. Jika belum ada pada *open list*, tambahkan ke *open list*. Simpan *current node* sebagai *parent* dari *neighbor node* ini. Simpan harga F masing-masing simpul.
c. Jika sudah ada dalam *open list*, periksa apakah ini jalan dari simpul ini ke *current node* yang lebih baik dengan menggunakan biaya G sebagai ukurannya. Simpul dengan biaya G yang lebih rendah berarti bahwa ini adalah jalan yang lebih baik. Jika demikian, buatlah simpul ini (*neighbor node*) sebagai *parent* dari *current node*, dan menghitung ulang nilai G dan F dari simpul ini.
d. Setelah mencapai *target point*, jika masih ada *neighbor node* yang memiliki nilai yang lebih kecil, maka simpul tersebut akan terus dipilih sampai bobotnya jauh lebih besar atau mencapai *target point* dengan bobot yang lebih kecil dibanding dengan simpul sebelumnya yang telah mencapai *target point*.
e. Pada saat pemilihan simpul berikutnya, nilai F(n) akan dievaluasi, dan jika terdapat nilai F(n) yang sama maka akan dipilih berdasarkan nilai G(n) terbesar.

3. Simpan jalan. Bekerja mundur dari *target point*, pergi dari masing-masing simpul ke simpul *parent* sampai mencapai *starting point*. Secara umum dapat digambarkan dengan *flowchart* pada gambar 2.4.



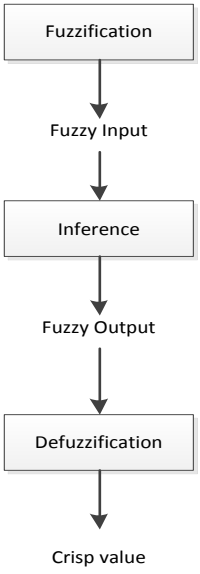
Gambar 2.4 *Flowchart* Algoritma A*[8]

2.5.2 *Fuzzy Logic*

Konsep tentang logika *fuzzy* diperkenalkan oleh Prof. Lotfi Astor Zadeh pada 1962. Logika fuzzy adalah metodologi sistem kontrol pemecahan masalah, dapat diterapkan pada perangkat keras, perangkat lunak, atau kombinasi keduanya. Dalam logika klasik dinyatakan bahwa segala sesuatu bersifat biner, yang artinya adalah hanya mempunyai dua kemungkinan, "Ya atau Tidak", "Benar atau Salah", "Baik atau Buruk" dan lain-lain. Oleh karena itu, semua ini dapat mempunyai nilai keanggotaan 0 dan 1. Artinya, bisa saja suatu keadaan mempunyai nilai "Ya dan Tidak", "Benar dan Salah", "Baik dan Buruk" secara bersamaan, namun besar nilainya tergantung pada bobot keanggotaan yang dimilikinya. Dalam banyak hal, logika fuzzy digunakan sebagai suatu cara untuk memetakan permasalahan dari *input* menuju ke output yang diharapkan. Konsep himpunan fuzzy memiliki 2 atribut, yaitu[9] :

1) Linguistik, yaitu nama suatu kelompok yang mewakili suatu keadaan tertentu dengan menggunakan bahasa alami, misalnya dingin, sejuk, panas mewakili variabel temperatur.
2) Numeris, yaitu suatu nilai yang menunjukan ukuran dari suatu variabel, misalnya 10, 35, 40 dan sebagainya.

Struktur sistem inferensi fuzzy dapat dilihat pada gambar 2.5.



Gambar 2.5 Struktur sistem fuzzy

Keterangan:

1) *Fuzzification*

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

2) *Inference*

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

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

2.5.2.1 Himpunan *Fuzzy*

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

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

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

Contoh :

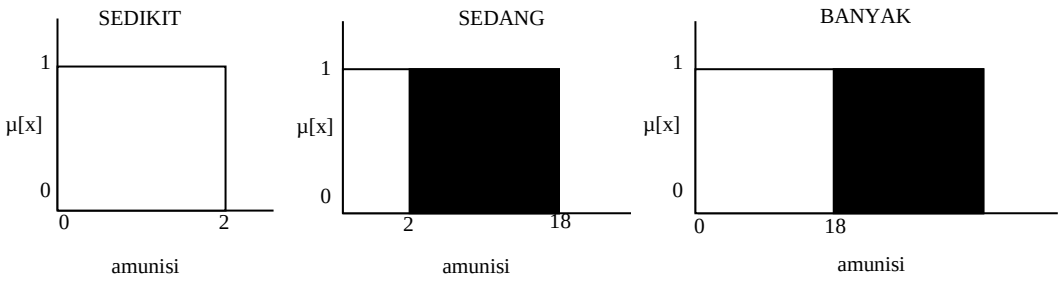
Misalkan variabel amunisi di dalam NPC (*Non Playable Character*) dibagi menjadi tiga kategori, yaitu :

SEDIKIT amunisi < 2

SEDANG $2 \leq$ amunisi ≤ 18

BANYAK amunisi > 18

Nilai keanggotaan secara grafis, himpunan SEDIKIT, SEDANG dan BANYAK ini dapat dilihat pada gambar 2.6 berikut.



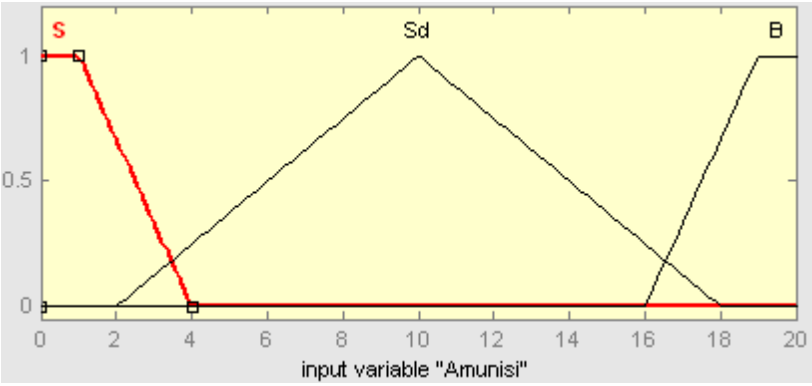
Gambar 2.6 Himpunan Sedikit, Sedang dan Banyak

Pada gambar di atas, dapat dijelaskan bahwa :

a. Apabila NPC memiliki jumlah amunisi 1, maka dikatakan SEDIKIT ($\mu_{SEDIKIT}[1]=1$).
b. Apabila NPC memiliki jumlah amunisi 2, maka dikatakan TIDAK SEDIKIT ($\mu_{SEDIKIT}[2]=0$).
c. Apabila NPC memiliki jumlah amunisi 2, maka dikatakan SEDANG ($\mu_{SEDANG}[2]=1$).
d. Apabila NPC memiliki jumlah amunisi 1, maka dikatakan TIDAK SEDANG ($\mu_{SEDANG}[1]=0$).
e. Apabila NPC memiliki jumlah amunisi 18, maka dikatakan SEDANG ($\mu_{SEDANG}[18]=1$)

Dari sini bisa dikatakan bahwa pemakaian himpunan tegas (*crisp*) untuk menyatakan amunisi sangat tidak adil. Adanya perubahan kecil saja pada suatu nilai mengakibatkan perbedaan kategori yang cukup signifikan.

Himpunan *fuzzy* digunakan untuk mengantisipasi hal tersebut. Amunisi pada NPC dapat masuk dalam dua himpunan yang berbeda, SEDIKIT dan SEDANG, SEDANG dan BANYAK dan sebagainya, seberapa besar eksistensinya dalam himpunan tersebut dapat dilihat pada nilai keanggotannya. Gambar 2.7 menunjukkan himpunan *fuzzy* untuk variabel amunisi.



Gambar 2.7 Himpunan *Fuzzy* Untuk Variabel Amunisi

Pada gambar di atas dapat dilihat bahwa :

a. Apabila amunisi pada NPC berjumlah 3, termasuk dalam himpunan SEDIKIT; namun juga termasuk dalam himpunan SEDANG.
b. Apabila amunisi pada NPC berjumlah 17, termasuk dalam himpunan SEDIKIT; namun juga termasuk dalam himpunan BANYAK.

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

a. Variabel *fuzzy*

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

b. Himpunan *fuzzy*

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

c. Semesta pembicaraan

   Semesta pembicaraan adalah keseluruhan nilai yang diperbolehkan untuk dioperasikan dalam suatu variabel *fuzzy*. Semesta pembicaraan merupakan himpunan bilangan *real* yang senantiasa naik (bertambah) secara monoton dari kiri ke kanan. Nilai semesta pembicaraan dapat

berupa bilangan *positif* maupun *negatif*. Pada suatu kondisi tertentu nilai semesta pembicaraan ini tidak dibatasi batas atasnya.

d. Domain

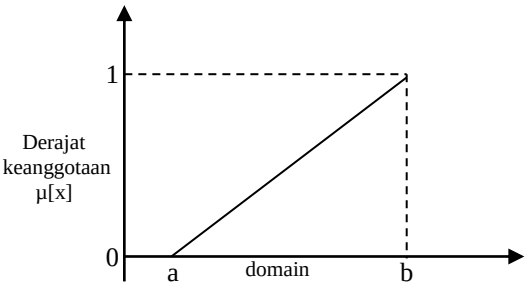
Domain himpunan *fuzzy* adalah keseluruhan nilai yang diijinkan dalam semesta pembicaraan dan boleh dioperasikan dalam suatu himpunan *fuzzy*. Seperti halnya semesta pembicaraan, domain merupakan himpunan bilangan *real* yang senantiasa naik (bertambah) secara monoton dari kiri ke kanan. Nilai domain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

2.5.2.2 Fungsi Keanggotaan

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

1. Representasi Linear

Pada representasi linear, pemetaan input ke derajat keanggotaannya digambarkan sebagai suatu garis lurus. Bentuk kurva ini paling sederhana dan menjadi pilihan yang baik untuk mendekati suatu konsep yang kurang jelas. Ada 2 keadaan himpunan fuzzy yang linear. Pertama, kenaikan himpunan dimulai pada nilai domain yang memiliki derajat keanggotaan nol [0] bergerak ke kanan menuju ke nilai domain yang memiliki derajat keanggotaan lebih tinggi (Gambar 2.8)**.**

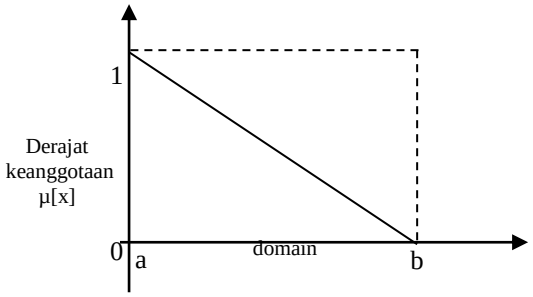


Gambar 2.8 Representasi Linear Naik

Fungsi keanggotaan :

$$\mu[x] = \begin{cases} 0; & x < a \\ (x-a)/(b-a); & a \le x \le b \\ 1; & x > b \end{cases} \quad (2.3)$$

Kedua, merupakan kebalikan dari yang pertama. Garis lurus dimulai dari nilai domain dengan derajat keanggotaan tertinggi pada sisi kiri, kemudian bergerak menurun ke nilai domain yang memiliki derajat keanggotaan lebih rendah (Gambar 2.9).



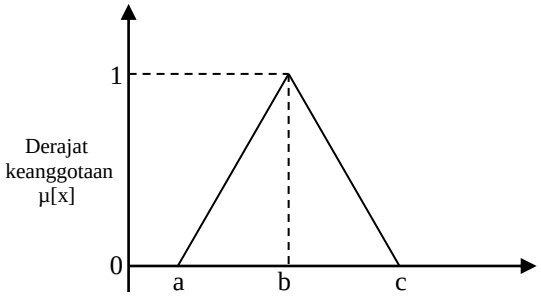
Gambar 2.9 Representasi Linear Turun

Fungsi Keanggotaan :

$$\mu[x] = \begin{cases} (b-x)/(b-a); & a \le x < b \\ 0; & x \ge b \end{cases} \quad (2.4)$$

2. Representasi Kurva Segitiga

Kurva segitiga pada dasarnya merupakan gabungan antara 2 garis (*linear*) seperti terlihat pada Gambar 2.10.



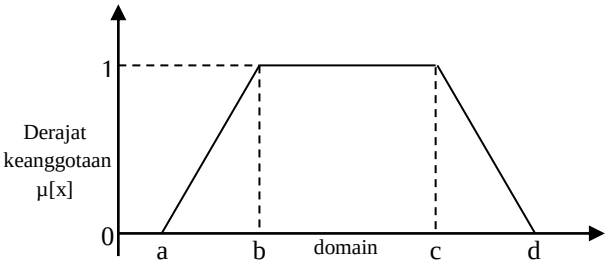
Gambar 2.10 Kurva Segitiga

Fungsi keanggotaan :

$$\mu[x] = \begin{cases} 0; & x \leq a \text{ atau } x \geq c \\ (x-a)/(b-a); & a < x \leq b \\ (c-x)/(c-b); & b < x \leq c \end{cases} \quad (2.5)$$

3. Representasi Kurva Trapesium

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



Gambar 2.11 Kurva Trapesium

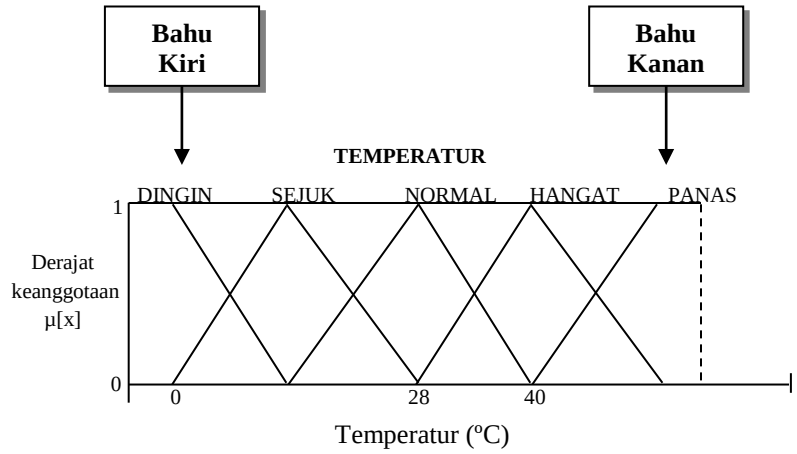
Fungsi Keanggotaan :

$$\mu[x] = \begin{cases} 0; & x \leq a \text{ atau } x \geq d \\ (x-a)/(b-a); & a \leq x \leq b \\ 1; & b \leq x \leq c \\ (d-x)/(d-c); & c \leq x \leq d \end{cases} \quad (2.6)$$

4. Representasi Kurva Bentuk Bahu

Daerah yang terletak di tengah-tengah suatu variabel yang direpresentasikan dalam bentuk segitiga, pada sisi kanan dan kirinya akan naik dan turun (misalkan: DINGIN bergerak ke SEJUK bergerak ke HANGAT dan bergerak ke PANAS). Tetapi terkadang salah satu sisi dari variabel tersebut tidak mengalami perubahan. Sebagai contoh, apabila telah mencapai kondisi PANAS, kenaikan temperatur akan tetap berada pada kondisi PANAS. Himpunan *fuzzy* ‘bahu’, bukan segitiga, digunakan

untuk mengakhiri variabel suatu daerah *fuzzy*. Bahu kiri bergerak dari benar ke salah, demikian juga bahu kanan bergerak dari salah ke benar. Gambar dibawah menunjukkan variabel TEMPERATUR dengan daerah bahunya.



Gambar 2.12 Daerah 'Bahu' Pada Daerah TEMPERATUR

2.6 OOP (Object Oriented Programming)

Objek adalah kesatuan entitas yang memiliki sifat dan tingkah laku. Dalam kehidupan sehari-hari, objek adalah benda, baik benda berwujud nyata seperti manusia, hewan, mobil, komputer, handphone, pena, ataupun benda yang tidak nyata atau konsep, seperti halnya tabungan bank, sistem antrian, sistem internet banking, dan sebagainya. Jadi pengertian OOP adalah konsep yang membagi program menjadi objek-objek yang saling berinteraksi satu sama lain. Objek adalah benda, baik benda yang berwujud nyata maupun benda yang tidak nyata (konsep). Jika menggunakan OOP maka akan ada enam keuntungan yang dapat diperoleh, yaitu[10]:

1. Alami (*Natural*).
2. Dapat diandalkan (*Reliable*).
3. Dapat digunakan kembali (*Reusable*).
4. Mudah untuk dalam perawatan (*Maintainable*).
5. Dapat diperluas (Extendable).
6. Efisiensi waktu.

Berikut ini beberapa bahasa pemrograman yang sudah menggunakan konsep OOP, adalah :

1. *C++*.
2. *Visual C++*.
3. *Visual Basic*.
4. *Java*.

2.7 *Tools* Rekayasa Perangkat Lunak (RPL)

1) *UML (Unified Modeling Language)*

UML adalah bahasa spesifikasi standar untuk mendokumentasikan, menspesifikasikan, dan membangun sebuah sistem. *UML* adalah himpunan struktur dan teknik untuk pemodelan desain program berorientasi objek (OOP) serta aplikasinya. *UML* adalah metodologi untuk mengembangkan sistem OOP dan sekelompok perangkat *tool* untuk mendukung pengembangan sistem tersebut. UML mulai diperkenalkan oleh *Object Management Group*, sebuah organisasi yang telah mengembangkan model, teknologi, dan standar OOP sejak tahun 1980-an. Sekarang UML sudah mulai banyak digunakan oleh para praktisi OOP. UML merupakan dasar bagi perangkat (*tool*) desain berorientasi objek dengan mendefinisikan diagram-diagram sebagai berikut[11] :

1). *use case diagram*
2). *class diagram*
3). *statechart diagram*
4). *activity diagram*
5). *sequence diagram*
6). *collaboration diagram*
7). *component diagram*
8). *deployment diagram*

1. ***Use Case Diagram***

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

sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang atau sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu.

Use case diagram dapat sangat membantu menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, 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-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal.

Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*.

Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain[11].

2. *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 (*atributte*/*propertis*) 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[11].

3. *Statechart Diagram*

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

Dalam UML, *state* digambarkan berbentuk segiempat dengan sudut membulat dan memiliki nama sesuai kondisinya saat itu. Transisi antar *state* umumnya memiliki kondisi *guard* yang merupakan syarat terjadinya transisi yang bersangkutan, dituliskan dalam kurung siku. *Action* yang dilakukan sebagai akibat dari *event* tertentu dituliskan dengan diawali garis miring[11].

4. *Activity Diagram*

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

Activity diagram merupakan *state diagram* khusus, dimana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan *behaviour* internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum[11].

5. *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[11].

6. *Collaboration Diagram*

Collaboration diagram juga menggambarkan interaksi antar objek seperti *sequence diagram*, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian *message*. Setiap *message* memiliki *sequence*

number, di mana *message* dari level tertinggi memiliki nomor 1. *Messages* dari level yang sama memiliki *prefiks* yang sama[11].

7. *Component Diagram*

Component diagram menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (*dependency*) di antaranya. Komponen piranti lunak adalah modul berisi *code*, baik berisi *source code* maupun *binary code*, baik *library* maupun *executable*, baik yang muncul pada *compile time*, *link time*, maupun *run time*.

Umumnya komponen terbentuk dari beberapa *class* dan/atau *package*, tapi dapat juga dari komponen-komponen yang lebih kecil. Komponen dapat juga berupa *interface*, yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain[11].

8. *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[11].

2.8 *Tools* Perangkat Lunak

1) C#

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

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

Meskipun aplikasi C# ditujukan agar bersifat 'ekonomis' dalam hal kebutuhan pemrosesan dan memori komputer, bahasa C# tidak ditujukan untuk bersaing secara langsung dengan kinerja dan ukuran program aplikasi yang dibuat dengan menggunakan bahasa pemrograman C[12].

2) Unity

Unity adalah sebuah sebuah *tool* yang terintegrasi untuk membuat *game*, arsitektur bangunan dan simulasi. Unity bisa untuk *game*s PC dan *games Online*. Untuk *games Online* memerlukan sebuah plugin, yaitu *Unity Web Player*, sama halnya dengan Flash Player pada Browser. Bahasa pemrograman yang digunakan bermacam-macam, mulai dari Javascript, C#, dan Boo[13].

Pada unity, tidak bisa melakukan desain / *modelling*, dikarenakan unity bukan *tool* untuk mendesain Jadi jika ingin mendesain, harus memerlukan 3D editor lain seperti *3dsmax* atau *Blender*. Banyak hal yang bisa di lakukan di unity , ada fitur *audio reverb zone , particle effect , Sky Box* Untuk menambahkan langit , dan masih banyak lagi tentunya, juga bisa langsung mengedit *texture* dari *editor* seperti *photoshop* dll , unity bagus untuk pemula maupun *expert* .

1. Mendukung 3 bahasa pemrograman, *JavaScript*, C#, dan Boo.
2. *Flexible and EasyMoving, rotating,* dan *scaling objects* hanya perlu sebaris kode . Begitu juga dengan *Duplicating, removing*, dan *changing properties.*
3. *Multi Platform Game* bisa di deploy di PC, Mac, Wii, iPhone, iPad dan *browser*, android.

4. *Visual Properties Variables* yang di definisikan dengan *scripts* ditampilkan pada Editor. Bisa digeser, di *drag* and *drop*, bisa memilih warna dengan *color picker*
5. Berbasis.NET –>Penjalanan program dilakukan dengan *Open Source*.NET *platform, Mono.*

3) 3D Studio *Max*

3ds Max atau *3D Studio Max* adalah salah *satu software* atau perangkat lunak yang sering digunakan oleh perancang produk untuk membuat animasi atau pemodelan dalam bentuk 3 dimensi. Aplikasi canggih ini dirilis oleh salah satu perusahaan *Autodesk Media & Entertainment* yang pada mulanya dikenal sebagai *Discreet and Kinetix*. *3D Max* merupakan salah satu dari sekian banyak aplikasi *modeling* untuk membuat model 3D dan paling banyak digunakan oleh perancang yang tersebar di seluruh dunia.

Sejalan dengan berkembangnya teknologi termasuk juga dalam bidang komputerisasi. 3D Max pun mengalami perubahan-perubahan untuk menyesuaikan dengan kemampuan komputer yang semakin tinggi dalam hal grafis. *3D Max* dikembangkan dari aplikasi sebelumnya yang bernama *3D Studio for Dos*, tetapi aplikasi ini hanya diperuntukan untuk *platform Win32*.

Dengan semakin canggihnya kemampuan *software* ini, maka tidak aneh *3D Max* menjadi program animasi komputer 3D dengan penjualan terbesar di dunia. *Software* ini memiliki kemampuan *modeling* yang kuat dan merupakan *plugin architecture* yang fleksibel dan bekerja dengan *platform Microsoft Windows*. *3D Studio Max* banyak digunakan oleh para pembuat dan perancang *video game, visual architecture, design product* dan juga studio TV untuk pembuatan animasi[14].

2.9 Pengolahan Data

Pengolahan data secara sederhana diartikan sebagai proses mengartikan data-data lapangan sesuai dengan tujuan, rancangan, dan sifat penelitian. Misalnya dalam rancangan penelitian kuantitatif, maka angka-angka yang diperoleh melalui alat pengumpul data tersebut harus diolah secara kuantitatif, baik melalui pengolahan statistik inferensial maupun statistik deskriptif. Lain halnya dalam

rancangan penelitian kualitatif, maka pengolahan data menggunakan teknik non statitistik, mengingat data-data lapangan diperoleh dalam bentuk narasi atau kata-kata, bukan angka-angka. Mengingat data lapangan disajikan dalam bentuk narasi kata-kata, maka pengolahan datanya tidak bisa dikuantifikasikan. Perbedaan ini harus dipahami oleh peneliti atau siapapun yang melakukan penelitian, sehingga penyajian data dan analisis kesimpulan penelitian relevan dengan sifat atau jenis data dan prosedur pengolahan data yang akan digunakan. Di atas dikatakan bahwa pengolahan data diartikan sebagai proses mengartikan data lapangan, yang berarti supaya data lapangan yang diperoleh melalui alat pengumpul data dapat dimaknai, baik secara kuantitatif maupun kualitatif, sehingga proses penarikan kesimpulan penelitian dapat dilaksanakan. Dengan demikian, pengolahan data tersebut dalam kaitannya dengan praktek pendidikan adalah sebagai upaya untuk memaknai data atau fakta menjadi makna.

2.9.1 Merancang *Kuesioner*

Kuesioner adalah pertanyaan terstruktur yang diisi sendiri oleh responden atau diisi oleh pewawancara yang membacakan pertanyaan dan kemudian mencatat jawaban yang diberikan. Pertanyaan yang diberikan pada kuesioner ini adalah pertanyaan menyangkut fakta dan pendapat responden, sedangkan kuesioner yang digunakan pada penelitian ini adalah kuesioner tertutup, dimana responden diminta menjawab pertanyaan dan menjawab dengan memilih dari sejumlah alternatif [15].

2.9.1.1 Jenis pertanyaan dalam *kuesioner*

Perbedaaan pertanyaan dalam wawancara dengan pertanyaan dalam kuesioner adalah dalam wawancara memungkinkan adanya interaksi antara pertanyaan dan artinya. Dalam wawancara analis memiliki peluang untuk menyaring suatu pertanyaan, menetapkan istilah-istilah yang belum jelas, mengubah arus pertanyaan, memberi respons terhadap pandanmgan yang rumit dan umumnya bisa mengontrol agar sesuai dengan konteksnya. Beberapa diantara peluang-peluang diatas juga dimungkinkan dalam kuesioner. Jadi bagi penganalisis pertanyaan-pertanyaan harus benar-benar jelas, arus pertanyaan

masuk akal, pertanyaan-pertanyaan dari responden diantisipasi dan susunan pertanyaan direncanakan secara mendetail.

1. Pertanyaan terbuka

   pertanyaan-pertanyaan yang memberi pilihan-pilihan respons terbuka kepada responden. Pada pertanyaan terbuka antisipasilah jenis respons yang muncul. Respons yang diterima harus tetap bisa diterjemahkan dengan benar.

2. Pertanyaan tertutup

   pertanyaan-pertanyaan yang membatasi atau menutup pilihan-pilihan respons yang tersedia bagi responden.

2.9.2 Skala dalam Kuesioner

Penskalaan adalah proses menetapkan nomor-nomor atau simbol-simbol terhadap suatu atribut atau karakteristik yang bertujuan untuk mengukur atribut atau karakteristik tersebut. Alasan penganalisis sistem mendesain *skala* adalah sebagai berikut.

1. Untuk mengukur sikap atau karakteristik orang-orang yang menjawab *kuesioner*.
2. Agar *respoden* memilih subjek *kuesioner*.

   Empat bentuk *skala* pengukuran yaitu :

1) *Nominal* : *Skala nominal* digunakan untuk mengklasifikasikan sesuatu. Skala nominal merupakan bentuk pengukuran yang paling lemah, umumnya semua analis bisa menggunakannya untuk memperoleh jumlah total untuk setiap klasifikasi. Contoh : Apa jenis perangkat lunak yang paling sering anda gunakan ? 1 = Pengolah kata, 2 = Spreadsheet, 3 = Basis Data, 4 = Program e-mail.
2) *Ordinal* : *Skala ordinal* sama dengan *skala nominal*, juga memungkinkan dilakukannya kalsifikasi. Perbedaannya adalah dalam ordinal juga menggunakan susunan posisi. *Skala ordinal* sangat berguna karena satu kelas lebih besar atau kurang dari kelas lainnya.
3) *Nterval* : *Skala interval* memiliki karakteristik dimana interval di antara masing-masing nomor adalah sama. Berkaitan dengan karakteristik ini,

operasi matematisnya bisa ditampilkan dalam data-data kuesioner, sehingga bisa dilakukan analisis yang lebih lengkap.

4) *Rasio* : *Skala rasio* hampis sama dengan *skala interval* dalam arti *interval-interval* di antara nomor diasumsikan sama. *Skala rasio* memiliki nilai *absolut nol*. *Skala rasio* paling jarang digunakan.

2.9.3 Skala *Likert*

Pemakaian *skala likert* digunakan untuk menganalisa *variabel perceived quality, brand loyalty*, dan keputusan pembelian konsumen. Nama skala ini diambil dari nama Rensis Likert, yang menerbitkan suatu laporan yang menjelaskan penggunaannya. Sewaktu menanggapi pertanyaan dalam skala *Likert*, responden menentukan tingkat persetujuan mereka terhadap suatu pernyataan dengan memilih salah satu dari pilihan yang tersedia. Pilihan jawaban pertanyaan kuesioner lebih dari 2 dan minimal 3. Tingkatan pada skala likert terdiri dari sangat setuju, setuju, biasa saja, tidak setuju, sangat tidak setuju, kelima penilaian tersebut diberikan bobot sebagai berikut :

Tabel 2.1 *Skala likert*

Skala penilaian	Bobot nilai
Sangat setuju	5
Setuju	4
Biasa saja	3
Tidak setuju	2
Sangat tidak setuju	1

Skala Likert dalam penelitian mempunyai gradasi nilai dari sangat positif sampai sangat negatif. Lalu hasil data kuesioner tersebut diolah dengan rumus sebagai berikut :

$$\mathrm{Y} = \frac{P}{Q} x 100\% \qquad (2.7)$$

Keterangan :

1. Y = Nilai presentase yang dicari
2. P = Bobot setiap pilihan jawaban
3. Q = Total bobot maksimal

2.10 Metode Pengujian

Pengujian perangkat lunak adalah elemen kritis dari jaminan kualitas suatu perangkat lunak dan merepresentasikan spesifikasi, desain dan pengkodeannya.

Pada dasarnya, Pengujian dilakukan terhadap aplikasi untuk memastikan bahwa aplikasi dapat berjalan dengan benar sesuai dengan kebutuhan dan tujuan yang diharapkan. Oleh karena itu, program harus diuji coba terlebih dahulu untuk menemukan kesalahan-kesalahan yang terjadi[16].

1. *Black box testing*

Sebuah Pengujian dilakukan terhadap aplikasi untuk memastikan bahwa aplikasi dapat berjalan dengan benar sesuai dengan kebutuhan dan tujuan yang diharapkan. Pengujian *black bok* berfokus kepada persyaratan fungsional perangkat lunak, pengujian ini berusaha menemukan kesalahan antara lain.

1) Fungsi-fungsi yang salah atau hilang
2) Kesalahan *interface*
3) Kesalahan dalam struktur data
4) Kesalahan kinerja
5) Kesalahan *inisialisasi*

2. *White box terting*

White Box merupakan metode desain uji kasus yang menggunakan struktur kontrol dari desain prosedural untuk menghasilkan kasus-kasus uji. Dengan menggunakan metode ujicoba *white box*, para pengembang *software* dapat menghasilkan kasus-kasus uji yang :

1) Menjamin bahwa seluruh independent paths dalam modul telah dilakukan sedikitnya satu kali.
2) Melakukan seluruh keputusan logikal baik dari sisi benar maupun salah
3) Melakukan seluruh perulangan sesuai batasannya dan dalam batasan operasionalnya
4) Menguji struktur data internal untuk memastikan validitasnya

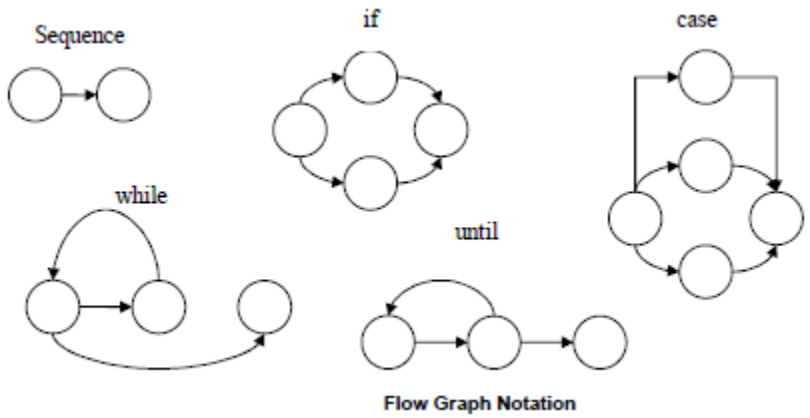
2.11 Uji coba berbasis alur (basis *path tasting*)

Ujicoba berbasis alur merupakan teknik ujicoba *whitebox* pertama yang diusulkan oleh Tom McCabe. Metode berbasis alur memungkinkan perancang kasus uji untuk menghasilkan ukuran kompleksitas logikal dari desain prosedural dan menggunakan ukuran ini untuk mendefinisikan himpunan basis dari alur

eksekusi. Kasus uji dihasilkan untuk melakukan sekumpulan basis yang dijamin untuk mengeksekusi setiap perintah dalam program, sedikitnya satu kali selama ujicoba.

2.11.1 *Notasi graft* alur (*Path Graph Notation*)

Notasi sederhana untuk merepresentasikan alur kontrol disebut graf alur (*flow graph*), seperti gambar dibawah ini :



Gambar 2.13 ***Notasi Graph***

Untuk mengilustrasikan kegunaan dari diagram alir dapat dilihat pada gambar dibawah ini. Urutan dari simbol proses dan simbol keputusan dapat digambarkan menjadi sebuah *node*, sedangkan anak panah disebut *edges*, menggambarkan aliran dari kontrol sesuai dengan diagram alir.

Sebuah *edge* harus berakhir pada sebuah node walaupun tidak semua node merepresentasikan perintah prosedural. *Area* yang dibatasi oleh *edge* dan *node* disebut *region*, *area* diluar *graph* juga dihitung sebagai *region*.