BAB 2
TINJAUAN PUSTAKA

2.1 Kecerdasan Buatan

Kecerdasan Buatan adalah bagian ilmu komputer yang membuat agar mesin komputer dapat melakukan pekerjaan seperti dan sebaik yang dilakukan manusia. Pada awal diciptakannya, komputer hanya difungsikan sebagai alat hitung saja. Namun seiring dengan perkembangan jaman, maka peran komputer semakin mendominasi kehidupan 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.

Manusia bisa menjadi pandai dalam menyelesaikan segala permasalahan di dunia ini karena manusia mempunyai pengetahuan dan pengalaman Pengetahuan diperoleh dari belajar. Semakin banyak bekal pengetahuan yang dimiliki oleh seseorang tentu saja diharapkan akan lebih mampu dalam menyelesaikan permasalahan. Namu bekal pengetahuan saja tidak cukup, manusia juga diberi akal untuk melakukan penalaran**,** mengambil kesimpulan berdasarkan pengetahuan dan pengalaman yang mereka miliki. Tanpa memiliki kemampuan untuk menalar dengan baik, manusia dengan segudang pengalaman dan pengetahuan tidak akan dapat menyelesaikan masalah dengan baik. Demikian pula dengan kemampuan menalar yang sangat baik, namun tanpa bekal pengetahuan dan pengalaman yang memadai, manusia juga tidak akan bisa menyelesaikan masalah dengan baik [4].

Agar komputer bisa bertindak seperti dan sebaik manusia, maka peneliti mencoba menerapkan kecerdasan buatan pada gerak NPC bertindak seperti manusia. Untuk itu AI akan mencoba untuk memberikan beberapa metoda untuk membekali komputer dengan kedua komponen tersebut agar NPC pada program bisa menjadi mesin pintar dan bertinkah laku layaknya manusia bergerak.

2.1.1 Lingkup utama kecerdasan buatan

Lingkup utama kecerdasan buatan ada 7 hal [4] antara lain:

1. Sistem pakar. Komputer digunakan sebagai saran untuk menyimpan pengetahuan para pakar. Dengan demikian komputer akan memiliki keahlian untuk menyelesaikan masalah dengan meniru keahlian yang dimiliki para pakar

2. Pengolahan bahasa alami. Dengan pengolahan bahasa alami ini diharapkan user mampu berkomunikasi dengan komputer dengan menggunakan bahasa sehari-hari.
3. Pengenalan ucapan. Melalui pengenalan ucapan diharapkan manusia mampu berkomunikasi dengan komputer dengan menggunakan suara.
4. Robotika dan Sistem sensor.
5. *Computer vision*, mencoba untuk dapat mengintrepetasikan gambar atau objek-objek tampak melalui komputer.
6. *Intelligent Computer aid Instruction*. Komputer dapat digunakan sebagai tutor yang dapat melatih dan mengajar.
7. *Game Playing*

2.1.2 Keuntungan Kecerdasan Buatan

Keuntungan kecerdasan buatan ada 7 hal [5]antara lain:

1. Kecerdasan buatan lebih bersifat permanen. Kecerdasan alami akan cepat mengalami perubahan. Hal ini dimungkinkan karena sifat manusia yang pelupa. Kecerdasan buatan tidak akan berubah sepanjang sistem komputer dan program tidak mengubahnya.

2. Kecerdasan buatan lebih mudah diduplikasi dan disebarkan. Mentransfer pengetahuan manusia dari satu orang ke orang lain butuh proses dan waktu lama. Disamping itu suatu keahlian tidak akan pernah bisa diduplikasi secara lengkap. Sedangkan jika pengetahuan terletak pada suatu sistem komputer, pengetahuan tersebuat dapat ditransfer atau disalin dengan mudah dan cepat dari satu komputer ke komputer lain

3. Kecerdasan buatan lebih murah dibanding dengan kecerdasan alami. Menyediakan layanan komputer akan lebih mudah dan lebih murah dibanding dengan harus mendatangkan seseorang untuk mengerjakan sejumlah pekerjaan dalam jangka waktu yang sangat lama.
4. Kecerdasan buatan bersifat konsisten. Hal ini disebabkan karena kecerdasan busatan adalah bagian dari teknologi komputer. Sedangkan kecerdasan alami senantiasa berubah-ubah.
5. Kecerdasan buatan dapat didokumentasikan. Keputusan yang dibuat komputer dapat didokumentasikan dengan mudah dengan melacak setiap aktivitas dari sistem tersebut. Kecerdasan alami sangat sulit untuk direproduksi.
6. Kecerdasan buatan dapat mengerjakan pekerjaan lebih cepat dibanding dengan kecerdasan alami
7. Kecerdasan buatan dapat mengerjakan pekerjaan lebih baik dibanding dengan kecerdasan alami.

2.2 Algoritma *particle swarm optimization* (PSO)

Algoritma *particle swarm optimization* (PSO) merupakan algoritma yang dikembangkan oleh Kennedy dan R. Eberhart pada tahun 1995 yang terinspirasi dari perilaku pergerakan kawanan hewan seperti ikan *(school of fish),* hewan herbivora *(herd),* burung *(flock),* dan kawanan serangga seperti semut rayap dan lebah. Tiap objek hewan disederhanakan menjadi partikel. Algoritma ini menerapkan sifat masing masing individu dalam satu kelompok besar kemudian menggabungkan sifat sifat tersebut untuk menyelesaikan masalah. Algoritma PSO adalah salah satu metode dari teknik kecerdasan buatan *swarm intelligence* yang berlandaskan perilaku kolektif (*collective beharviour*) dan dapat mengatur dirinya sendiri (*self-organizing*). Dalam *swarm intelligence* algoritma *particle swarm optimization* ini merupakan metoda yang digunakan untuk penentuan posisi partikel terbaik . Konsep optimasi partikel terdiri dari kerumunan. Pada persamaan dari algoritma PSO dasar yang di perkenalkan oleh oleh Kennedy dan R.Eberhart setiap partikel memiliki nilai fitness yang dievaluasi oleh fungsi yang

dioptimasi, selain itu parameter lain yang dimiliki oleh tiap partikel adalah posisi partikel, konstanta, kecepatan partikel, posisi terbaik partikel dan posisi global terbaik. dari beberapa parameter yang di gunakan diatas dapat membentuk update posisi dilakukan secara serempak di tiap tiap partikel. Sebelum update posisi setiap partikel terlebih dahulu melakukan update kecepatan. PSO dapat diterapkan pada berbagai kasus misalnya game, PSO digunakan untuk mengoptimasi gerak optimal NPC untuk mendapatkan tujuan yakni menentukan dari beberapa NPC yang memilik posisi terbaik sebagai pemimpin dari NPC lainnya.

2.2.1 Langkah–langkah penyelesaian Algoritma PSO

Dalam PSO untuk menyelesaikan masalah langkah langkahnya adalah sebagai berikut [1]:

1. Inisialisasi populasi partikel

Langkah ini merupakan tahap awal dimana jumlah dan koordinat partikel / posisi partikel ditentukan terlebih dahulu secara acak.

2. Melakukan perhitungan nilai fitness dari setiap partikel

Untuk memperoleh nilai fitness dibutuhkan nilai posisi setiap partikel tersebut untuk dievaluasi oleh fungsi fitness dengan menggunakan persamaan berikut:

$$F(X) = \sqrt{(player.x - NPC.x)^2 + (player.y - NPC.y)^2} \qquad (1)$$

F(X) adalah *fitness*

$player.x$ adalah koordinat X *player*

$player.y$ adalah koordinat Y *player*

$NPC.x$ = koordinat X NPC

$NPC.y$ = koordinat Y NPC

3. Menentukan Pbest (*personal best*)

Pbest adalah posisi terbaik setiap partikel. Posisi ini merupakan posisi yang ditempati oleh partikel saat ini.

4. Menentukan Gbest (*Global best*)

Nilai Gbest merupakan nilai posisi partikel yang memiliki posisi paling baik dari partikel lainnya. Nilai Gbest ini didapat dengan membandingkan nilai fitness dari setiap partikel untuk kemudian koordinat posisi dari partikel yang memiliki

fitness paling optimal tersebut akan dijadikan Gbest. Nilai fitness dikatakan optimal tergantung kasus yang dimiliki apakah maksimasi ataupun minimasi

5. Menghitung nilai *velocity* (V)

Untuk mendapatkan nilai V terbaru dapat digunakan persamaan untuk perbaharui kecepatan menggunakan persamaan berikut :

$V_i\ (t+1) = V_i(t)+C_1.r_1(Pbest_{\ i} - X_i(t)) + C_2.r_2(Gbest_{\ i} - X_i(t))$ (2)

Dimana :

V_i (t)	adalah kecepatan partikel ke (i) pada iterasi ke (t)
X_i (t)	adalah Posisi partikel ke (i) pada iterasi ke (t)
V	adalah *Velocity* (kecepatan partikel)
X	adalah Koordinat partikel (posisi partikel)
i	adalah Partikel ke (1,2,3 . . . . . N)
C_1 dan C_2	adalah Faktor *cognitive* dan sosial nilai ini adalah nilai konstanta bernilai diantara (0 dan 1)
r_1 dan r_2	adalah Bilangan acak antara 0 – 1
Pbest	adalah Posisi terbaik yang dimiliki setiap partikel
Gbest	adalah posisi partikel yang memiliki posisi paling baik dari partikel lainnya.

6. Menghitung nilai koordinat terbaru

Memasukan hasil perhitungan nilai V dari setiap partikel kedalam persamaan menggunakan persamaan berikut :

$X_i\ (t+1) = X_i\ (t) + V_i\ (t+1)$ (3)

Dimana :

V_i (t)	adalah kecepatan partikel ke (i) pada iterasi ke (t)
X_i (t)	adalah Posisi partikel ke (i) pada iterasi ke (t)

7. Mengecek posisi partikel apakah sudah mencapai titik tujuan atau belum

Nilai dikatakan konveregen jika posisi partikel mencapai posisi tujuan. Jika partikel belum mencapai titik tujuan maka langkah 2 diulangi dengan

mengevaluasi nilai partikel saat ini kedalam fungsi tujuan dibatasi hingga 30 kali iterasi.

2.2.2 Kelebihan dan Kekurangan PSO

A. Kelebihan dari algoritma PSO [6] :

1. Algoritma PSO didasari oleh *intelligence*. Hal ini dapat diterapkan baik kedalam penelitian ilmiah maupun teknik.
2. Algoritma PSO tidak memiliki perhitungan mutasi yang tumpang tindih. Pencarian dapat dilakukan dengan kecepatan setiap partikel. partikel yang optimal dapat mengirim informasi sehingga partikel lain dapat cepat melakukan pencarian.
3. Perhitungan dari algoritma PSO sangat sederhana dibandingkan dengan perhitungan algoritma yang berkembang lainnya.Algoritma PSO ini memiliki tingkat optimasi yang paling tinggi dan dapat diselesaikan dengan perhitungan yang sederhana.
4. Algoritma PSO mengandung bilangan real

B. Kekurangan dari algoritma PSO [6] :

1. Algoritma ini mudah mengalami optimasi yang parsial yang mengakibatkan kurang tepatnya pengaturan terhadap kecepatan dan tujuan.
2. Algoritma ini tidak dapat bekerja diluar masalah optimasi.
3. Algoritma ini tidak dapat bekerja diluar masalah sistem koordinat

2.3 Differential Evolution Algorithm (DEA)

Differential evolutional algorithm (DEA) atau lebih dikenal dengan algoritma DE (*Differential Evolution*) adalah sebuah metode yang dikembangkan oleh Kenneth Price dan dipublikasikan pada Oktober 1994 dalam majalah Dr. Dobb's Journal. Metode ini merupakan metode optimasi matematis yang fungsi multidimensional dan termasuk dalam kelompok evolutionary algorithm. DEA merupakan salah satu algoritma evolusi yang terkenal dengan kecepatan

konvergensinya dan merupakan salah satu algoritma evolusioner yang memiliki performansi lebih baik daripada algoritma evolusioner yang lain seperti *genetic algorithm* (GA). Kelebihan DEA adalah pada evolusi yang dialami oleh setiap individu dalam populasi dimana diferensiasi dan *crossover* terjadi secara berurutan pada setiap individu yang terpilih acak dari populasi setiap waktu. Parameter *crossover* diubah untuk mengetahui pengaruhnya pada proses komputasi. DEA dalam beberapa kasus disebutkan memiliki kecepatan komputasi yang lebih baik dibandingkan PSO dimana sistem akan lebih cepat menuju keadaan steady state [2].

2.3.1 Langkah-langkah penyelesaian dalam DEA

Penyelesaian Dalam DEA langkah- langkahnya sebagai berikut [7] :

1. Inisialisasi populasi partikel

Langkah ini merupakan tahap awal dimana jumlah dan koordinat partikel / posisi partikel ditentukan terlebih dahulu secara acak.

2. Mutation

Setelah diinisialisasi, DEA akan bermutasi dan me-rekombinasi populasi awal untuk menghasilkan populasi baru. Mutasi pada beberapa kamus bahasa menunjukan pengertian berubah dan dalam konteks genetika mutasi berarti perubahan dengan elemen acak.Proses mutasi menggunakan persamaan berikut :

$$V_{i,g} = X_{r1,g} + F \,.\, (X_{r2,g} - X_{r3,g}) \qquad (4)$$

Dimana :

$V_{i,g}$ adalah persamaan yang menunjukan bagaimana membentuk vektor mutan

$_{r1,r2,r3}$ adalah index acak

F adalah skala faktor konstan antara [0,2].

3. crossover

Untuk melengkapi strategi pencarian differential mutation, DE menggunakan crossover dengan tujuan meningkatkan diversitas parameter populasi. Crossover

membangun vektor uji dari nilai parameter yang telah dikopi dari dua vektor yang berbeda. Persamaan untuk vektor uji adalah sebagai berikut:.

$U_{j,i,g} = V_{j,i,g}$, if ($rand_{i,j}$ [0,1) ≤ Cr or $j=jrand$)

$X_{j,i,g}$, otherwise, (5)

Dimana:

U adalah persamaan crossover.

Cr adalah konstanta crossover yang ditentukan oleh pembuat.

j_{rand} adalah indeks yang dipilih secara acak.

4. Selection

Setelah diperoleh vektor hasil proses mutasi, dan crossover proses selanjutnya adalah seleksi antara vektor turunan dengan vektor target dimana vektor yang memiliki nilai fitness yang lebih baik akan digunakan untuk generasi berikutnya. Untuk proses seleksi menggunakan persamaan berikut :

$X_{I,G+1} = U_{i,G}$, if $f(U_{I,G}) \leq f(X_{I,G})$

$= X_{I,G}$, if $f(U_{I,G}) > f(X_{I,G})$ (6)

Dimana :

f adalah fungsi yang harus diminimalkan.

5. Kriteria penghentian iterasi

Nilai dikatakan konveregen jika posisi partikel mencapai posisi tujuan. Jika partikel belum mencapai titik tujuan maka langkah 2 diulangi dengan mengevaluasi nilai partikel saat ini kedalam fungsi tujuan dibatasi hingga 20 kali iterasi. [2]

2.3.2 Kelebihan dan kekurangan DEA

A. Kelebihan dari algoritma DEA [7]:

1. Algoritma DEA didasari oleh *intelligence.* Hal ini dapat diterapkan baik kedalam penelitian ilmiah maupun teknik.
2. Algoritma DEA tidak memiliki perhitungan mutasi yang tumpang tindih. Pencarian dapat dila

3. kukan dengan kecepatan setiap partikel. partikel yang optimal dapat mengirim informasi sehingga partikel lain dapat cepat melakukan pencarian.
4. Perhitungan dari algoritma DEA sangat sederhana dibandingkan dengan perhitungan algoritma yang berkembang lainnya.Algoritma DEA ini memiliki tingkat optimasi yang paling tinggi dan dapat diselesaikan dengan perhitungan yang sederhana.
5. Algoritma DEA mengandung bilangan real

B. Kekurangan dari algoritma DEA [7]:

1. Algoritma ini mudah mengalami optimasi yang parsial yang mengakibatkan kurang tepatnya pengaturan terhadap kecepatan dan tujuan.
2. Algoritma ini tidak dapat bekerja diluar masalah optimasi.
3. Algoritma ini tidak dapat bekerja diluar masalah sistem koordinat

Tabel 1.1 Tabel Penelitian PSO dan DEA

PENULIS	TEMA	HASIL	KESIMPULAN
Madehan anis, Widowati dan R.Heru Tjahjana *juusan matematika Universitas Dipenogoro, Semarang*	Perbandingan Algoritma Particle swarm optimization dan Differential Evoluitonal Algorithm untuk perancangan umpan balik keadaan : Studi Kasus Gerak Lateral Pesawat F-	performansi optimum sebesar 6.579 dengan membutuhkan waktu kinerja rata-rata CPU (CPU time) dalam dua puluh kali running sebesar 21.14403 detik dan DEA menghasilkan indeks	algoritma DEA lebih unggul dari segi performansi optimum dan membutuhkan waktu kinerja rata-rata CPU time lebih cepat dibandingkan algoritma PSO pada kasus gerak lateral pesawat

	16	performansi optimum sebesar 6.417 dengan membutuhkan waktu kinerja rata-rata CPU time dalam dua puluhkali running sebesar 10.60157 detik	
yoga baskara	implementasi algoritma particle swarm optimization untuk perilaku non player character menyerang secara berkelompok	hasil pengujian C1 dan C2 diperoleh nilai C1 dan C2 yang paling optimal yaitu C1 = 0,4 dan C2 = 0,8. dari pengujian 10 NPC dibutuhkan 15-18 kali proses iterasi untuk menentukan NPC yang menjadi pemimpin	setiap NPC dalam kelompok NPC dapat dengan tepat memilih NPC mana yang menjadi pemimpin dengan menyesuaikan nilai fitness yang dihasilkan setiap NPC dengan NPC mana yang terpilih menjadi pemimpin
Irmaduta Fahmiari dan Budi Santosa Jurusan Teknik Industri Institut Teknologi Sepuluh Nopember (ITS)	aplikasi algoritma differential evolution untuk permasalahan kompleks pemilihan portofolio	DE dengan parameter CR = 0.2 dan 0.9 memiliki akurasi yang baik yai menghasilkan solusi yang	Solusi yang dihasilkan DE lebih baik daripada solusi yang dihasilkan GRG2 (Solver).

Surabaya		konsisten yaitu masing-masing dengan standar deviasi 0.0016 dan 0.048.	
Rudi Nurdiansyah Jurusan Teknik Industri, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember Surabaya	pengembangan algoritma differential evolution untuk penjadwalan flow shop multi obyektif dengan banyak mesin	Hasil dari penelitian ini menunjukkan bahwa algoritma DE yang dikembangkan pada penelitian ini mempunyai performa yang lebih baik dibandingkan dengan algoritma CR(MC), HAMC1, HAMC 2, HAMC3 maupun MOACSA untuk obyektif makespan, total flowtime maupun multi obyektif keduanya.	DE dengan adaptive parameters dan local search pada riset ini dapat juga diaplikasikan pada permasalahan penjadwalan pada beberapa sistem manufaktur yang lain seperti job shop, cellular manufacturing dan flexible manufacturing.

2.4 Pemrograman Berorientasi Object (PBO)

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

Dalam rekayasa perangkat lunak, konsep pendekatan berorientasi objek dapat diterapkan pada tahap analisis perancangan,pemrograman dan pengujian perangkat lunak. Ada berbagai teknik yang dapat digunakan pada masing-masing tahap tersebut, dengan aturan dan alat bantu pemodelan tersebut

2.4.1 Konsep Dasar PBO

Berikut ini adalah beberapa konsep dasar yang harus dipahami tentangmetodologi berorientasi objek :

2.4.1.1 Object

Objek adalah abstraksi dan sesuatu yang mewakili dunia nyata,seperti benda, manusia, satuan, tempat, kejadian, struktur, statusatau hal-hal lain yang bersifat abstrak. Objek merupakan suatuentitas yang mampu menyimpan informasi (status) dan mempunyaioperasi (kelakuan) yang dapat diterapkan atau dapat berpengaruhpada status objek. Objek mempunyai siklus hidup yaitu diciptakan,dimanipulasi dan dihancurkan.

2.4.1.2 Kelas (Class)

Kelas adalah kumpulan objek-objek dengan karakteristik yang sama. Kelas merupakan definisi statis dan himpunan objek yang sama yang mungkin lahir atau diciptakan dan kelas tersebut.Sebuah kelas akan mempunyai sifat (atribut), kelakuan (operasi/metode), hubungan (relationship) dan arti. Suatu kelas dapat diturunkan dan kelas yang lain, di mana atribut dan kelas semula dapat diwariskan ke kelas yang baru.Secara teknis, kelas adalah sebuah struktur tertentu dalam

pembuatan perangkat lunak. Kelas merupakan bentuk struktur padakode program yang menggunakan metodologi berorientasi objek

2.4.1.3 Methode

Operasi atau metode atau method pada sebuah kelas hampir sama dengan fungsi atau prosedur pada metodologi struktural. Sebuah kelas boleh memiliki lebih dari satu metode atau operasi. Metode atau operasi yang berfungsi untuk memanipulasi objek itu sendiri.Operasi atau transformasi yang dapat dilakukan terhadap objek atau dilakukan oleh objek.Metode atau operasi dapat berasal dari event , aktivitas atau aksi keadaan, fungsi atau kelakuan dunia nyata. Contoh metode atau operasi misalnya :Read, Write, Move, Copy, dan sebagainya.

2.4.1.4 Encapsulation

Pembukusan atribut data dan layanan (operasi-operasi) yang dipunyai objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerja-nya. Misalkan seperti pada ilustrasi berikut.

2.4.1.5 Inheritance (Pewarisan)

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

2.4.1.6 Polimorfisme

Kemampuan suatu objek untuk digunakan di banyak tujuan yangberbeda dengan nama yang sama sehingga menghemat barisprogram.

2.4.1.7 Antarmuka

Antarmuka atau interface sangat mirip dengan kelas, tapi tanpa atribut kelas dan memiliki metode yang dideklarasikan tanpa isi.Deklarasi metode pada sebuah interface dapat diimplementasikan oleh kelas lain.Sebuah kelas dapat mengimplementasikan lebih dari satu antarmuka di mana kelas ini akan mendeklarasikan metode pada antarmuka yang dibutuhkan oleh kelas itu sekaligus mendefinisikanisinya pada kode program itu. Metode pada antarmuka yang diimplementasikan pada suatu kelas harus sama persis dengan yang ada pada antarmuka . antarmuka atau interface Biasanya digunakan agar kelas yang lain tidak mengakses langsung ke suatu kelas, mengakses antarmukanya.

2.4.1.8 Reusability

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

2.4.1.9 Generalisasi Spesialisai

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

2.4.1.10 Komunikasi Antar Object

Komunikasi antar objek dilakukan lewat pesan (message) yang dikirim dari satu objek ke objek lainnya.

2.4.1.11 Package

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

2.4.1.12 Atribute

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

2.4.1.13 Abstraksi

Prinsip untuk merepresentasikan dunia nyata yang kompleksmenjadi satu bentuk model yang sederhana dengan mengabaikanaspek-aspek lain yang tidak sesuai dengan permasalahan.

2.4.2 Unified Modelling Language (UML)

Unified Modelling Language (disingkat UML) adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan kebutuhan (requirement), membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek.

UML hanya berfungsi untuk melakukan pemodelan. Jadi penggunaan UML tidak terbatas pada metodologi tertentu, meskipun pada kenyataanya

UML paling banyak digunakan pada metodologi berorientasi objek. Berikut merupakan macam-macam diagram UML:

2.4.2.1 Use Case Diagram

a. Menggambarkan bagaimana sistem digunakan
b. Mendeskripsikan apa yang sistem lakukan tanpa mendeskripsikan bagaimana sistem menyelesaikannya.
c. Dibuat berdasarkan interaksi dan relasi dari individual use case.
d. Berisi aktor, event, dan use case.
e. Titik awal untuk pemodelan UML.

2.4.2.2 Activity Diagram

Menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir.

2.4.2.3 Sequence Diagram

Menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, display, dan sebagainnya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagrams* terdiri atas dimensi vertical (waktu) dan dimensi horizontal (objek-objek yang terkait).

2.4.2.4 Class Diagram

Menggambarkan keadaan suatu sistem dengan menjelaskan keterhubungan antara suatu *class* dengan *class* yang lain yang terdapat pada sistem.