BAB 2
LANDASAN TEORI

2.1 *Game* (Permainan)

Game merupakan salah satu media hiburan paling popular untuk semua kalangan usia. Sejak pertama kali ditemukan sampai saat ini, 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 [15].

2.1.1 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 matamatis 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 [15].

2.1.2 *Genre Game*

Menurut buku "*Fundamentals of Game Design*", *genre game* dapat dibagi menjadi 9 [4], antara lain:

1. *Action*

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

2. *Strategy*

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

3. *Role Playing Game (RPG)*

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

4. *Sport*

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

5. *Vechile Simulation*

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

6. *Construction and Management Simulation*

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

7. *Adventure/Platform*

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

8. *Artificial Life and Puzzle Game*

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

9. *Online Game*

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

2.2 AI (*Artificial Intelligence)*

Para ahli mendefinisikan AI secara berbeda-beda tergantung pada sudut pandang mereka masing-masing. Ada yang fokus pada logika berpikir manusia saja, ada juga yang mendefinisikan AI secara lebih luas pada tingkah laku manusia. Pengelompokan definisi AI ada empat kategori [5], yaitu:

1. *Acting humanly: The Turing test approach*

   Kecerdasan buatan dalam kategori ini dirancang oleh Alan Turing pada tahun 1950, beliau merancang sebuah komputer berintelejensia dengan tujuan apakah komputer tersebut mampu mengelabui seorang manusia

yang mengintrograsinya melalui *teletype* (komunikasi berbasis pesan jarak jauh). Jika integrator tidak dapat membedakan yang diintrograsi adalah manusia atau komputer maka komputer tersebut lulus dari *Turing test*. Untuk menerapkan kelompok AI ini komputer harus memiliki kemampuan mengolah bahasa secara natural, perangkat belajar berdasarkan kondisi apapun, representasi pengetahuan, pengambilan keputusan, *Computer Vision*, *robotics*. *Turing test* sengaja menghindari interaksi fisik antara integrator dan komputer karena simulasi fisik manusia tidak memerlukan intelejensia.

2. *Thinking humanly: The cognitive modelling approach*

   Pendekatan ini dilakukan dengan dua cara yaitu:

   a. Melalui introspeksi: mencoba menangkap pemikiran-pemikiran kita sendiri saat berpikir.
   b. Melalui eksperimen-eksperimen psikologi.

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

   Masalah pada pendekatan ini diantaranya:

   a. Tidak mudah untuk membuat pengetahuan informasi dan menyatakan pengetahuan tersebut ke dalam kondisi formal yang diperlukan oleh notasi logika, khususnya ketika pengetahuan tersebut memiliki kepastian kurang dari 100%.
   b. Terdapat perbedaan besar antara dapat memecahkan masalah "dalam prinsip" dan memecahkannya dalam dunia maya.

4. *Acting rationally: The rational agent approach*

   Membuat tindakan berdasarkan kesimpulan yang diambil secara logis merupakan bagian dari suatu *rational agent*. Aksi rasional harus dilakukan dengan cara menalar secara logis, dengan menalar secara logis maka bisa didapatkan kesimpulan aksi apa yang harus dilakukan untuk mencapai tujuan, sehingga *agent* memiliki aksi yang tepat.

Thinking humanly dan *acting humanly* adalah dua definisi dan arti yang sangat luas. Sampai saat ini, pemikiran manusia diluar rasio, yakni refleks dan intuitif (berhubungan dengan perasaan), belum dapat ditirukan sepenuhnya oleh komputer. Dengan demikian, kedua definisi ini dirasa kurang tepat untuk saat ini. Jika kita menggunakan definisi ini maka banyak produk komputasi cerdas saat ini tidak layak disebut sebagai produk AI.

Definisi *thinking rationally* terasa lebih sempit daripada *acting rationally* dengan pendekatan *rational agent*. Didasari dengan pemikiran komputer yang mampu melakukan penalaran secara logis dan juga bisa melakukan aksi secara rasional dengan pengambilan keputusan dengan menyimpulkan penalaran logis yang dilakukan komputer.

Definisi *thinking rationally* terasa lebih sempit daripada *acting rationally* dengan pendekatan *rational agent*. Didasari dengan pemikiran komputer yang mampu melakukan penalaran secara logis dan juga bisa melakukan aksi secara rasional dengan pengambilan keputusan dengan menyimpulkan penalaran logis yang dilakukan komputer.

2.3 MAS (*Multi Agent System*)

Konsep *agent* sudah dikenal lama dalam bidang kecerdasan buatan (AI), tepatnya dikenalkan oleh Carl Hewitt dengan model aktor konkurennya. Dalam modelnya, Hewitt mengemukakan teori tentang suatu objek yang disebut dengan aktor yang mempunyai karakterisitik menguasai dirinya sendiri, interaktif, dan bisa merespon pesan yang datang dari objek lain yang sejenis. Dari berbagai penelitian yang berhubungan dengan hal di atas, lahirlah cabang ilmu besar yang merupakan turunan dari kecerdasan buatan, yaitu kecerdasan buatan tersebar (DAI) yang antara lain membawahi bidang penelitian *Distributed Problem Solving* (DPS), *Parallel Artificial Intelligence* (PAI), dan *Multi-Agent System* (MAS).

Multi-agent system (MAS) adalah sistem yang terdiri dari sejumlah agent yang berinteraksi satu sama lain untuk mencapai suatu tujuan. *Agent* itu sendiri adalah suatu entitas perangkat lunak komputer yang memungkinkan pengguna untuk mendelegasikan tugas kepadanya secara mandiri (autonomi). *Agent* memiliki pemecahan masalah sendiri (yang diimplementasikan dalam perangkat keras, perangkat lunak, atau kombinasi keduanya).

Berdasarkan pada karakteristiknya, *agent* dapat diklasifikasikan menjadi tujuh jenis. Di antaranya [16]:

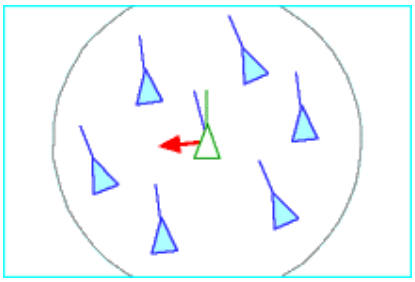
a. *Collaborative Agent*: *Agent* yang memiliki kemampuan melakukan kolaborasi dan koordinasi antar-*agent* dalam kerangka *Multi Agent System* (MAS).
b. *Interface Agent*: *Agent* yang memiliki kemampuan untuk berkolaborasi dengan user, melakukan fungsi *monitoring* dan *learning* untuk memenuhi kebutuhan user.
c. *Mobile Agent*: *Agent* yang memiliki kemampuan untuk bergerak dari suatu tempat ke tempat lain, dan secara mandiri melakukan tugas ditempat barunya tersebut, dalam lingkungan jaringan komputer.
d. *Information and Internet Agent*: *Agent* yang memiliki kemampuan untuk menjelajah internet untuk melakukan pencarian, pemfilteran, dan penyajian informasi untuk *user*, secara mandiri. Atau dengan kata lain, memanage informasi yang ada di dalam jaringan Internet.
e. *Reactive Agent*: *Agent* yang memiliki kemampuan untuk bisa cepat beradaptasi dengan lingkungan baru dimana dia berada.
f. *Hybrid Agent*: Kita sudah mempunyai lima klasifikasi *agent*. Kemudian *agent* yang memiliki katakteristik yang merupakan gabungan dari karakteristik yang sudah kita sebutkan sebelumnya adalah masuk ke dalam *hybrid agent*.
g. *Heterogeneous Agent System*: Dalam lingkungan *Multi Agent System* (MAS), apabila terdapat dua atau lebih *hybrid agent* yang memiliki perbedaan kemampuan dan karakteristik, maka sistem MAS tersebut kita sebut dengan *heterogeneous agent system*.

2.4 Algoritma *Boids*

Algoritma *boids* adalah algoritma yang dikembangkan oleh Craig Reynolds yang pada awalnya ditujukan untuk mensimulasikan perilaku berkelompok dari burung, tapi algoritma ini juga dapat diterapkan untuk setiap lingkungan yang melibatkan kerumunan atau perilaku berkelompok [6]. Algoritma ini menggabungkan tiga pengaturan perilaku sederhana yang menggambarkan bagaimana sebuah manuver boid individu berdasarkan pada posisi dan kecepatan kawanan terdekatnya. Berikut adalah tiga pengaturan perilaku sederhana dari algoritma *boids* akan dibahas dibawah ini:

1. *Alignment*

   Alignment merupakan kemampuan pengendalian perilaku untuk mensejajarkan dirinya dengan agen yang berada disekitarnya, tujuannya adalah untuk menuju kearah yang sama serta mencocokan kecepatan tiap agen yang berada di dalam kawanan tersebut. Berikut adalah ilustrasi dari *alignment* dapat dilihat pada gambar 2.1.



Gambar 2.1 *Alignment Rule* [7]

Alignment rule dapat dihitung dengan menggunakan rumus yang dapat dilihat pada persamaan 2.1 [8]:

$$d(Px, Pb) \leq d1 \cap d(Px, Pb) \geq d2 \Rightarrow Var = \frac{1}{n}\sum_{x}^{n} vx \quad [2.1]$$

dimana :

d = jarak

Px = posisi tetangga X

Pb = posisi boid B

$d1$ = jarak 1

$d2$ = jarak 2

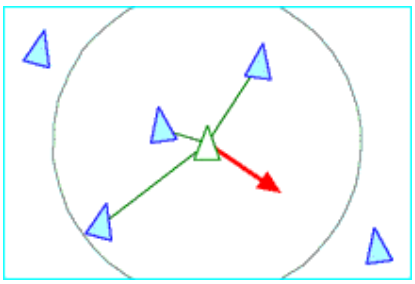
Var = kecepatan yang dipengaruhi oleh aturan *alignment*

n = jumlah total tetangga dari boid B

vx = kecepatan dari boid X

2. *Separation*

Separation merupakan kemampuan pengendalian perilaku dalam mempertahankan atau memelihara jarak antar agen yang berdekatan, agar tidak terjadi benturan atau kepadatan. Berikut adalah ilustrasi dari *separation* dapat dilihat pada gambar 2.2.



Gambar 2.2 *Separation Rule* [7]

Separation rule dapat dihitung dengan menggunakan rumus yang dapat dilihat pada persamaan 2.2 [8]:

$$d(Px,Pb) \leq d1 \cap d(Px,Pb) \geq d2 \Rightarrow Vsr = \sum_{x}^{n} \frac{Vx+Vb}{d(Px,Pb)} \quad [2.2]$$

dimana :

d = jarak

Px = posisi tetangga X

Pb = posisi boid B

$d1$ = jarak 1

$d2$ = jarak 2

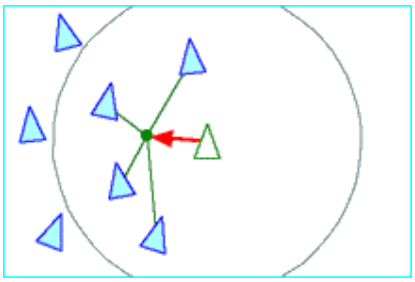
Vsr = kecepatan yang dipengaruhi oleh aturan *separation*

Vx = Kecepatan dari boid X

Vb = Kecepatan dari boid B

3. *Cohesion*

Cohesion merupakan kemampuan pengendalian perilaku untuk bergerak menuju posisi rata-rata kawanan terdekatnya. Berikut adalah ilustrasi dari *cohesion* dapat dilihat pada gambar 2.3



Gambar 2.3 *Cohesion Rule* [7]

Cohesion rule dapat dihitung dengan menggunakan rumus yang dapat dilihat pada persamaan 2.3 [8]:

$$d(Px,Pb) \leq d1 \cap d(Px,Pb) \geq d2 \Rightarrow Pavg = \sum_{x}^{n} \frac{Px}{n} \quad [2.3]$$

dimana :

d = jarak

Px = posisi tetangga X

Pb = posisi boid B

$d1$ = jarak 1

$d2$ = jarak 2

$Pavg$ = posisi rata-rata kawanan terdekat boid

Px = posisi tetangga X

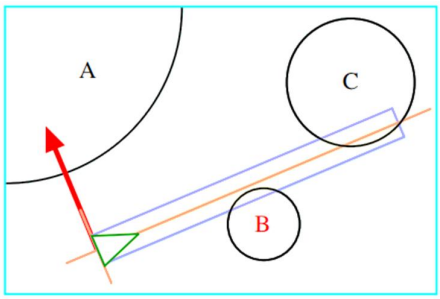
n = jumlah total tetangga dari boid B

2.5 *Steering Behavior*

Steering behavior bertujuan untuk membantu *autonomous characters* bergerak secara realistis, dengan menggunakan kekuatan sederhana yang dikombinasikan untuk menghasilkan hasil yang lebih hidup seperti navigasi improvisasi di sekitar lingkungan karakter. *Steering behavior* tidak didasarkan pada strategi yang kompleks yang melibatkan perencanaan jalur atau perhitungan global, namun menggunakan informasi lokal (*flockmates*) [9]. Terdapat beberapa macam *steering behavior*, namun yang akan dijelaskan disini hanya salah satu dari beberapa jenis perilaku yang dapat membantu bagi individu dan pasangan, yaitu *obstacle avoidance*.

2.5.1 *Obstacle Avoidance*

Perilaku menghindari rintangan (*obstacle avoidance behavior*) memberikan kemampuan kepada karakter untuk dapat bermanuver di dalam lingkungan yang berantakan dengan menghindari hambatan yang ada di sekitarnya. Berikut adalah ilustrasi dari perilaku menghindari rintangan dapat dilihat pada gambar 2.4.



Gambar 2.4 *Obsctacle Avoidance* [10]

Impelentasi dari perilaku menghindari rintangan yang dijelaskan disini akan membuat asumsi yang sederhana bahwa karakter dan hambatan dapat diperkirakan sebagai sphere, meskipun konsep dasar dapat dengan mudah meluas ke bentuk model yang lebih tepat. Perlu diingat bahwa ini hanya berkaitan dengan menghindari rintangan, bukan untuk mendeteksi tabrakan.

2.6 *Schooling* pada ikan

Tingkah laku hewan dibedakan menjadi 2 macam, yaitu tingkah laku individual dan tingkah laku sosial. Tingkah laku individual dilakukan oleh satu hewan, misalnya burung elang mencari makan di muara sungai. Tingkah laku sosial terjadi karena adanya kerjasama diantara anggota-anggotanya. Contoh dari tingkah laku sosial adalah *schooling* pada ikan [11].

Tingkah laku sosial diawali dengan daya tarik, kemudian dilanjutkan dengan pendekatan. Setelah terjadi pendekatan dilanjutkan dengan agregasi/ pengelompokan, dan akhirnya dilakukan kerjasama. Tujuan dari tingkah laku sosial adalah untuk pemeliharaan baik individu, kelompok maupun spesies [11].

Ikan berenang secara berkelompok, hal ini jelas merupakan suatu bentuk organisasi sosial. Biasanya individu dalam suatu kelompok ikan terdiri atas satu spesies memiliki ukuran yang hampir sama, tidak memiliki pemimpin, serta semua individu melakukan aktivitas yang sama dalam waktu yang sama pula [12].

2.7 *ActionScript*

ActionScript adalah bahasa pemrograman yang digunakan di Flash. *ActionScript* dapat digunakan untuk mengontrol objek di Flash, membuat navigasi dan elemen interaktif lain, serta membuat *movie* Flash dan aplikasi Web yang interaktif [13]. *ActionScript* mempunyai beberapa kesamaan dengan *JavaScript*, dengan begitu jika sebelumnya sudah akrab dengan *JavaScript* maka tidak akan kesulitan menguasasi *ActionScript*. Kesamaan antara kedua bahasa pemrograman ini misalnya, berbasis even, di JavaScript pemanggilan fungsi atau kode skrip di atribut dilakukan dengan menggunakan onClick, onLoad, unUnload, dan sebagainya, di *ActionScript* juga kode terpantik jiga terjadi sebuah even yang ditentukan, seperti tombol mouse ditekan atau dilepas, kursor masuk ke daerah tertentu, dan masih banyak lagi. Berikut adalah konsep-konsep dasar dari bahasa pemrograman *ActionScript*.

2.7.1 Objek dan Kelas

Objek adalah tipe data seperti suara, gambar, teks yang digunakan untuk mengontrol movie. Semua objek merupakan bagian dari suatu kelas. Sebelum objek dapat digunakan, kita perlu memberi nama pada objek tersebut, sama seperti tiap orang perlu memiliki nama supaya dapat dikenali. Misalnya Ballerina, Flintstone, dan Napi dapat dibedakan berdasar nama mereka. Orang, tanaman, dan gajah adalah contoh objek di dunia nyata. Sedangkan Ballerina, Flintstone, dan Napi adalah contoh objek dari kelas Manusia.

2.7.2 Metode dan Properti

Setiap objek selain berbeda dalam nama, juga berbeda dalam karakteristik. Setiap orang berbeda dalam berbagai karakteristik seperti jenis kelamin, tinggi badan, berat badan, dan warna rambut. Di dalam ActionScript karakteristik-karakteristik ini dinamakan *properti*. Misalnya kelas MovieClip memiliki berbagai properti seperti _height, _width, dan _rotation yang mengukur dimensi dan orientasi dari objek movie klip tersebut.

Objek juga melakukan suatu tugas. Seseorang dapat tidur, bekerja, makan. Pekerjaan ini di dalam ActionScript dinamakan *metode*. Misalnya kelas Sound memiliki metode setVolume yang dapat membuat suara lebih keras atau lebih lembut.

2.8 Pemrograman Berorientasi Objek

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

Pada saat ini, metode berorientasi objek banyak dipilih karena metodologi lama banyak menimbulkan masalah seperti adanya kesulitan pada saat mentransformasi hasil dari suatu tahap pengembangan ke tahap berikutnya, misalnya pada metode pendekatan terstruktur, jenis aplikasi yang dikembangkan saat ini berbeda dengan masa lalu. Aplikasi yang dikembangkan pada saat ini sangat beragam (aplikasi bisnis, *real-time*, *utility*, dan sebagainya) dengan *platform* yang berbeda-beda, sehingga menimbulkan tuntutan kebutuhan metodologi pengembangan yang dapat mengakomodasi ke semua jenis aplikasi tersebut [3].

Keuntungan menggunakan metodologi berorientasi objek adalah sebagai berikut [3]:

1. Meningkatkan produktivitas

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

2. Kecepatan pengembangan

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

3. Kemudahan pemeliharaan

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

4. Adanya konsistensi

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

5. Meningkatkan kualitas perangkat lunak

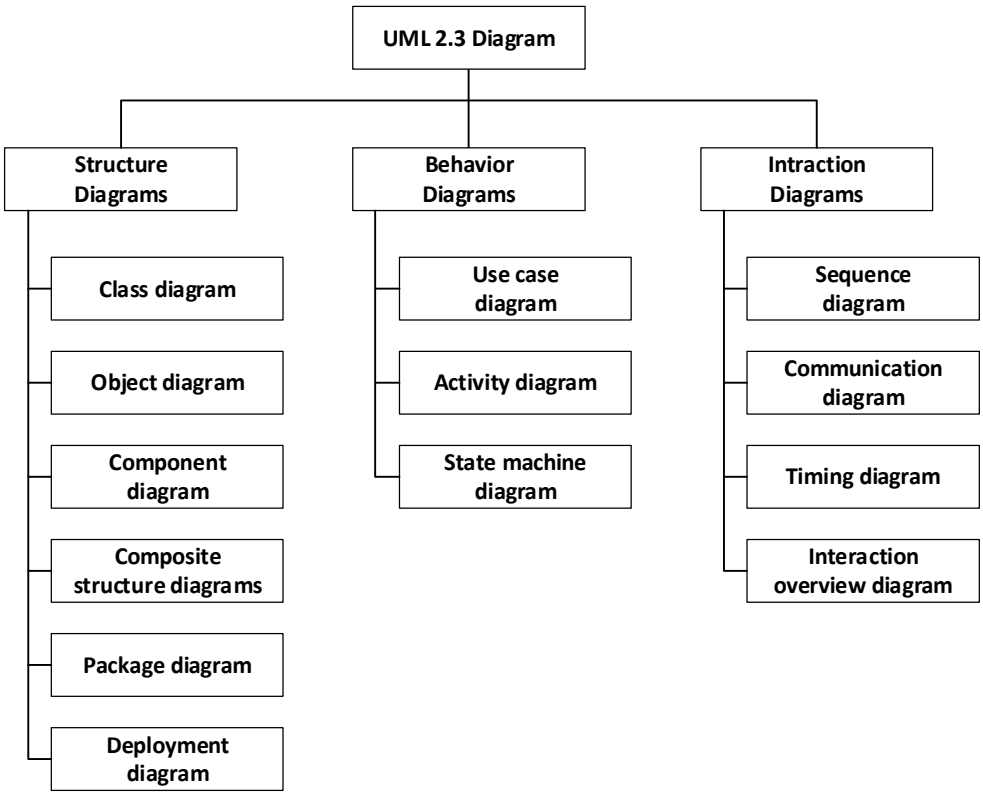
   Karena pendekatan pengembangan lebih dekat dengan dunia nyata dan adanya konsistensi pada saat pengembangannya, perangkat lunak yang dihasilkan akan mampu memenuhi kebutuhan pemakai serta mempunyai sedikit kesalahan.

2.9 UML (*Unified Modeling Language*)

UML adalah bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakan teknik pemrograman berorientasi objek. UML diperlukan karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak. UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung [3].

2.9.1 Diagram UML

Pada UML 2.3 terdiri dari 13 macam diagram yang dikelompokkan dalam 3 kategori [3]. Pembagian kategori dan macam-macam diagram tersebut dapat dilihat pada gambar 2.5



Gambar 2.5 Diagram UML [3]

Berikut ini adalah penjelasan singkat dari pembagian kategori tersebut:

a. *Structure diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
b. *Behavior diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.
c. *Interaction diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antar subsistem pada suatu sistem.

Penjelasan tentang masing-masing diagram adalah sebagai berikut:

1. *Class Diagram*

   Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi.
2. *Object Diagram*

   Diagram objek menggambarkan struktur sistem dari segi penamaan objek dan jalannya objek dalam sistem. Diagram objek juga berfungsi untuk mendefinisikan contoh nilai atau isi dari atribut tiap kelas.
3. *Component Diagram*

   Diagram komponen atau *component diagram* dibuat untuk menunjukan organisasi dan ketergantungan diantara kumpulan komponen dalam sebuah sistem. Diagram komponen fokus pada komponen sistem yang dibutuhkan dan ada di dalam sistem
4. *Composite Structure Diagram*

   Diagram ini dapat digunakan untuk menggambarkan struktur dari bagian-bagian yang saling terhubung maupun mendeskripsikan struktur pada saat berjalan (*runtime*) dari *instance* yang saling terhubung.
5. *Package Diagram*

   Package diagram menyediakan cara mengumpulkan elemen-elemen yang saling terkait dalam diagram UML. Hampir semua diagram dalam UML dapat dikelompokkan menggunakan *package diagram*.

6. *Deployment Diagram*

   Deployment diagram menunjukkan konfigurasi komponen dalam proses eksekusi aplikasi.

7. *Use Case Diagram*

   Use case diagram merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat.

8. *Activity Diagram*

   Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak.

9. *State Machine Diagram*

   State machine diagram atau *state chart diagram* digunakan untuk menggambarkan perubahan status atau transisi status dari sebuah mesin atau sistem atau objek.

10. *Sequence Diagram*

    Diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek.

11. *Communication Diagram*

    Diagram komunikasi menggambarkan interaksi antar objek/bagian dalam bentuk urutan pengiriman pesan. Diagram komunikasi merepresentasikan informasi yang diperoleh dari diagram kelas, diagram sekuen, dan diagram *use case* untuk mendeskripsikan gabungan antara struktur statis dan tingkah laku dinamis dari suatu sistem.

12. *Timing Diagram*

    Timing diagram merupakan diagram yang fokus pada penggambaran terkait batasan waktu. *Timing diagram* digunakan untuk menggambarkan tingkah laku sistem dalam periode waktu tertentu.

13. *Interaction Overview Diagram*

    Interaction overview diagram berfungsi untuk menggambarkan sekumpulan urutan aktivitas. *Interaction overview diagram* adalah bentuk aktivitas diagram yang setiap titik merepresentasikan diagram interaksi.