BAB 2
LANDASAN TEORI

2.1 Game

Game merupakan salah satu media hiburan yang paling populer untuk semua kalangan usia. Game dapat dimainkan dengan aturan tertentu sehigga ada yang menang dan ada yang kalah, biasanya dalam konteks tidak serius atau dengan tujuan refreshing. Suatu cara belajar yang digunakan dalam menganalisa interaksi antara sejumlah pemain maupun perorangan yang menunjukkan strategi-strategi yang rasional. Sejak pertama kali ditemukan sampai saat sekarang, teknologi game telah mengalami kemajuan yang terbilang pesat. Hal ini dapat dilihat dengan berkembangnya jenis, produk, alat, dan jenis interaksi game dengan penggunaan yang semakin beragam.

2.1.1 Pengertian *Game*

Game berasal dari kata bahasa inggris yang berarti dasar permainan. Permainan dalam hal ini merujuk pada pengertian kelincahan intelektual (*Intellectual Playability Game*) yang juga bisa diartikan sebagai arena keputusan dan aksi pemainnya. Dalam *game,* ada target-target yang ingin dicapai pemainnya. Permainan adalah kegiatan yang kompleks yang didalamnya terdapat peraturan, play dan budaya.Sebuah permainan adalah sebuah sistem dimana pemain terlibat dalam konflik buatan, disini pemain berinteraksi dengan sistem dan konflik dalam permainan merupakan rekayasa atau buatan, dalam permainan terdapat peraturan yang bertujuan untuk membatasi perilaku pemain dan menentukan permainan.

Game bertujuan untuk menghibur, *game* banyak disukai oleh anak-anak hingga orang dewasa. *Game* sebenarnya penting untuk perkembangan otak, untuk meningkatkan konsentrasi dan melatih untuk memecahkan masalah dengan tepat dan cepat karena dalam game terdapat berbagai konflik atau masalah yang menuntut kita untuk menyelesaikannya dengan cepat dan tepat. Tetapi game juga bisa merugikan karena apabila kita sudah kecanduan game kita akan lupa waktu

dan akan mengganggu kegiatan atau aktivitas yang sedang kita lakukan (Nilwan, 2010).

Berikut adalah Pengertian *Game* menurut para ahli (Nilwan, 2010):

1. Menurut David Parlett, *Game* adalah sesuatu yang memiliki "akhir dan cara mencapainya": artinya ada tujuan, hasil dan serangkaian peraturan untuk mencapai keduanya.
2. Menurut Clark C.Abt, *Game* adalah kegiatan yang melibatkan keputusan pemain, berupaya mencapai tujuan dengan;dibatasi oleh konteks tertentu (misalnya, dibatasi oleh peraturan).
3. Menurut Bernard Suits, *Game* adalah upaya sukarela untuk mengatasi rintangan yang tidak perlu.
4. Menurut Greg Costikyan, *Game* adalah sebentuk karya seni di mana peserta, yang disebut Pemain, membuat keputusan untuk mengelola sumberdaya yang dimilikinya melalui benda di dalam *game* demi mencapai tujuan.
5. Menurut Wijaya Ariyana & Deni Arifianto, game merupakan salah satu kebutuhan yang menjadi masalah besar bagi pengguna komputer, karena untuk dapat memainkan game dengan nyaman, semua komponen komputernya harus memiliki kualitas yang baik, terutama VGA card-nya.

2.1.2 Sejarah *Game*

Game pertama kali diperkenalkan pada tahun 1971 sebagai medium hiburan yang bersifat komersial. Pada akhir tahun 1970an dan awal tahun 1980an, *video games* menjadi sebuah dasar yang penting bagi industri hiburan di Amerika Serikat, Jepang, dan Eropa. Sejak masa kejatuhan dalam bidang industri pada tahun 1983 dan kebangkitannya dua tahun kemudian, industri *video game* telah mengalami perkembangan selama dua dekade dan menjadi industri yang berharga $10 miliar, yang menjadi saingan dengan industri ‘*motion picture’* sebagai industri hiburan yang paling berprofit besar. Pada tahun 1952, A.S. Douglas

menulis untuk gelar PhD di University of Cambridge jurusan Human-Computer Interraction. Douglas menciptakan permainan komputer grafis yang diberi nama Tic-Tac-Toe. Permainan ini diprogram pada komputer EDSAC vacuum-tube yang memiliki tampilan tabung sinar katode.William Higinborham menciptakan video game pertama pada tahun 1958. Permainan ini diberi judul "*Tennis for Two*", diciptakan dan dimainkan di osiloskop Brookhaven National Laboratory. Pada tahun 1962, Steve Russell menciptakan SpaceWar! Permainan ini merupakan permainan pertama yang dimaksudkan untuk pemakaian komputer. Russel menggunakan MIT PDP-1 komputer mainframe untuk meciptakan permainan ini. Pada 1967, Ralph Baer menciptakan video game pertama yang dimaikan dengan televisi, sebuah permainan yang diberi nama Chase. Ralph Baer akhirnya menjadi bagian dari Sanders Associates, sebuah perusahaan elektronik militer. Ralph mencetuskan idenya untuk pertama kali pada 1951, saat sedang bekerja untuk Loral, sebuah perusahaan komputer.

Pada 1971, Nolan Bushnell bersama dengan Ted Dabney, menciptakan permainan arcade yang pertama. Permainan ini diberi nama Computer Space, yang diciptakan berdasarkan pada permainan SpaceWar! Permainan arcade Pong diciptakan oleh Nolan Bushnell (dengan bantuan Al Alcorn) setahun kemudian pada tahun 1972. Nolan Bushnell dan Ted Dabney memulai dengan Atari Computer pada tahun yang sama. Pada tahun 1975, Atari kembali meluncurkan Pong sebagai home video game. Tahun 1972, hiburan pertama video game komersial dapat dimainkan di rumah, Odyssey, diluncurkan oleh Magnavox dan didesain oleh Ralph Baer. Permainan mesin ini aslinya didesain pada saat Ralph Baer masih berada di Sanders Associates pada tahun 1966, Baer mengurus untuk mendapatkan ijin legalnya setelah ditolak oleh Sanders Associates. Permainan Odyssey deprogram dengan dua belas permainan. Pada tahun 1976, Fairchild meluncurkan programmable home game yang pertama dan diberi nama Fairchild Video Entertainment System atau Channel F. Channel F merupakan salah satu dari electronic system yang pertama menggunakan microchip yang diciptakan oelh Robert Noyce untuk Fairchild Semiconductor Corporation (Rabin, 2010).

2.1.3 Jenis – jenis *Game*

Berdasarkan jenisnya game dibagi menjadi beberapa kategori diantaranya:

Berdasarkan platfrom atau alat yang digunakan adalah (Nilwan, 2010):

1. Arcade games, yaitu yang sering disebut ding-dong di Indonesia, biasanya berada di daerah / tempat khusus dan memiliki box atau mesin yang memang khusus di design untuk jenis video games tertentu dan tidak jarang bahkan memiliki fitur yang dapat membuat pemainnya lebih merasa "masuk" dan "menikmati", seperti pistol, kursi khusus, sensor gerakan, sensor injakkan dan stir mobil (beserta transmisinya tentunya).

2. PC Games , yaitu video game yang dimainkan menggunakan Personal Computers.

3. Console games, yaitu video games yang dimainkan menggunakan console tertentu, seperti Playstation 2, Playstation 3, XBOX 360, dan Nintendo Wii..

4. Handheld games, yaitu yang dimainkan di console khusus video game yang dapat dibawa kemana-mana, contoh Nintendo DS dan Sony PSP..

5. Mobile games, yaitu yang dapat dimainkan atau khusus untuk mobile phone atau PDA.

2.1.4 Game Poker

Poker adalah permainan kartu nasional Amerika, dan popularitasnya terus tumbuh. Dari Mississippi dan Michigan ke New Mexico dan North Dakota, Anda dapat menemukan permainan berlangsung di mana-mana. Jika Anda ingin bermain, Anda dapat menemukan poker dimainkan pada replika abad ke-19 kincir pengoperasian atau asli tanah suku Amerika. Anda dapat bermain poker di tweetable, tanpa embel-embel kartu dan elegan mega klub Los Angeles County di mana 150 pertandingan (dengan batas taruhan mulai dari $ 1- $ 2 menjadi $ 200- $ 400) sedang berlangsung sepanjang waktu (Horroch dan Krieger, 2003).

1. Tujuan dari permainan

Tujuan dari poker adalah untuk memenangkan uang dengan menangkap pot, yang berisi taruhan yang dibuat oleh berbagai pemain selama pertandingan. Seorang pemain bertaruh taruhan dengan harapan bahwa ia memiliki kartu terbaik, atau memberi kesan bahwa ia memegang kartu yang kuat dan dengan demikian meyakinkan lawan-lawannya untuk melipat (meninggalkan) kartu mereka. Karena uang yang disimpan adalah sama nilainya dengan uang menang, mengetahui kapan harus melepaskan kartu yang tampaknya dipukuli adalah sama pentingnya dengan mengetahui kapan harus bertaruh. Dalam kebanyakan game poker, kombinasi atas lima kartu adalah kartu terbaik.

2. Jumlah pemain

Sejumlah pemain, biasanya 2-10, bisa bermain, tergantung pada permainan. Kebanyakan permainan kasino ditetapkan dengan delapan pemain untuk pertandingan tujuh kartu seperti Stud poker atau Razz, dan sembilan atau sepuluh pemain untuk Texas Holdem.

3. Dasar permainan

Poker adalah permainan sederhana untuk belajar, meskipun salah satu dapat menghabiskan seumur hidup berusaha untuk menguasainya.

Dalam permainan seperti Seven-Card Stud dan Texas Hold'em tangan terbaik adalah tangan tinggi. Pada pertandingan lain, seperti Lowball dan Razz, tangan terbaik adalah tangan rendah. (Kemungkinan tangan rendah terbaik adalah 54-3-2-A, yang terbaik berikutnya adalah 64-3-2-A.)

Dalam permainan split-pot, dua pemenang membagi pot. Sebagai contoh, di Seven-Card Stud, Split Tinggi-Rendah, Delapan-atau-lebih baik (untungnya disingkat Tujuh-Stud / 8) dan Omaha High-Low Split,

Delapan-atau-lebih baik (atau hanya Omaha / 8) yang terbaik tangan tinggi dan terbaik tangan rendah membagi pot (asalkan seseorang membuat tangan rendah terdiri dari lima kartu berpasangan dengan pangkat 8 atau lebih rendah). Kemungkinan terburuk tangan rendah akan terdiri dari 87-6-54. Yang terbaik dari semua tangan rendah 54-3-2-A (dikenal sebagai roda atau sepeda). Sementara tangan yang tinggi selalu akan dibuat dalam permainan split-pot, ada tidak akan selalu menjadi tangan rendah. Dan ketika tidak ada tangan rendah, tangan tinggi memenangkan seluruh pot.

Kebanyakan game mengharuskan ante atau taruhan buta. Jika antes digunakan, tiap pemain harus memasukkan jumlah mata uang dalam rangka untuk menerima kartu. Adapun tirai, satu atau dua pemain diharuskan untuk membuat taruhan atau bagian dari taruhan sebelum tangan dibahas. Persyaratan ini berputar di sekeliling meja sehingga setiap pemain membayar bagiannya secara adil.

Setiap kali putaran kartu dibagikan, pemain memiliki kesempatan untuk memeriksa, bertaruh, lipat, menelepon, atau meningkatkan. Setiap kali seorang pemain memutuskan untuk kehilangan minatnya dalam panci, ia dapat melepaskan tangannya ketika giliran untuk bertindak (melakukan sesuatu yang berhubungan dengan taruhan: meningkatkan, lipat, cek, atau panggilan). Ketika seorang pemain lipatan tangan, ia tidak diperlukan untuk menempatkan lebih banyak uang dalam pot. Jika taruhan pemain atau kenaikan gaji dan tidak ada panggilan, pot milik pemain yang, kartu dikumpulkan dan dikocok, dan tangan berikutnya dibagikan. Jika ada dua atau lebih pemain masih aktif pada akhir tangan, tangan terbaik memenangkan pot.

Meskipun ada aturan yang berbeda untuk setiap versi tertentu, poker benar-benar sederhana ini. Namun dalam kesederhanaan terletak struktur permainan luar biasa bertekstur yang selalu menarik, sering

menyenangkan, dan, untuk beberapa, sumber seumur hidup kesenangan.

4. Peringkat kartu

Tujuh Kartu Stud dan Texas Hold'em adalah dua bentuk yang paling populer poker di mana ranking tertinggi menang. Permainan ini dimainkan dengan setumpuk 52- kartu tidak ada joker terdiri dari empat sesuai: sekop, hati, wajik, dan keriting. Setiap sesuai adalah sama nilainya, dan ada 13 peringkat di setiap setelan. Ace adalah kartu tertinggi peringkat dalam setelan jas, diikuti oleh raja, ratu, jack, dan 10 melalui 2 (atau deuce), dalam urutan. Ace juga dapat digunakan sebagai kartu peringkat terendah dalam lurus 5-tinggi (5-43-2-A), yang juga disebut roda atau sepeda. Meskipun Stud dan Hold'em dimainkan dengan tujuh kartu, tangan terbaik mengacu terbaik lima buah kartu. Peringkat tangan merupakan fungsi dari probabilitas. The jarang tangan, semakin berharga itu.

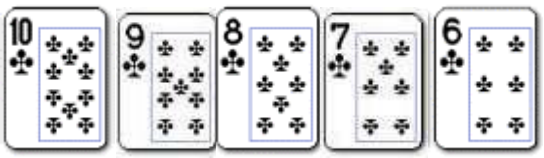
Kartu yang telah terbentuk dinamakan hand, berikut hands rangking yang diurutkan dari yang terkuat sampai terlemah:

a. **Royal flush**: Lima kartu tertinggi berurutan dalam satu suit



Gambar 2.1 Urutan Kartu Royal Flush

b. **Straight Flush**: Lima kartu berurutan dalam satu suit/sejenis



Gambar 2.2 Urutan Kartu Straight Flush

c. **Four of a Kind**: Empat kartu dengan angka atau nilai yang sama.



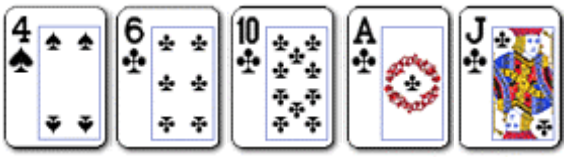
Gambar 2.3 Urutan Kartu Four Of a Kind

d. **Full House**: Tiga kartu dengan nilai yang sama di tambah dengan Dua kartu dengan nilai yang sama.



Gambar 2.4 Urutan Kartu Full House

e. **Flush**: Lima kartu dalam satu suit/sejenis tetapi tidak berurutan.



Gambar 2.5 Urutan Kartu Flush

f. **Straight**: Lima kartu berurutan.



Gambar 2.6 Urutan Kartu Straight

g. **Three of a kind**: Tiga kartu dengan nilai yang sama.



Gambar 2.7 Urutan Kartu Three Of a Kind

h. **Two pair**: Dua kartu dengan nilai yang sama/sepasang ditambah Dua kartu dengan nilai yang sama.



Gambar 2.8 Urutan Kartu Two Pair

i. **One pair**: 2 kartu dengan nilai yang sama/sepasang



Gambar 2.9 Urutan Kartu One Pair

Dalam permainan poker terdapat beberapa aksi yang dapat memperlihatkan tingkat ke agresifan dari lawan, berikut adalah tabel aksi dan tingkat keagresifannya (Horroch dan Krieger, 2003):

Tabel 2.1 Tingkat Keagresifan

Aksi	Tingkat Keagresifan
All In	100
Bet	25
Bet Bet (Berurutan 2 kali)	60
Call	0
reRaise	50
Fold	-25
Fold Fold (Berurutan 2 kali)	-60

2.2 Kecerdasan Buatan

Para ahli mendefinisikan AI secara berbeda-beda tergantung pada sudut pandang mereka masing-masing. Ada yang fokus pada logika berpikir manusia saja, tetapi ada juga yang mendefinisikan AI secara lebih luas pada tingkah laku manusia. Stuart Russel dan Peter Norvig mengelompokkan definisi AI, yang diperoleh dari beberapa *textbook* berbeda, ke dalam empat kategori (Russel dan Norvig, 1995), yaitu:

A. *Thinking humanly* : *the cognitive modeling approac* Pendekatan ini dilakukan dengan dua cara sebagai berikut :

1. Melalui introspeksi : mencoba menangkap pemikiran-pemikiran kita sendiri pada saat kita berpikir. Tetapi, seorang psikolog Barat mengatakan *"how do you know that you understand?"* Bagaimana anda tahu bahwa anda mengerti? Karena pada saat anda menyadari pemikiran anda. Sehingga definisi ini terkesan mengada-ngada dan tidak mungkin dilakukan.

2. Melalui eksperimen-eksperimen psikologi.

B. *Acting humanly* : *the Turing test approach* Pada tahun 1950, Alan Turing merancang suatu ujian bagi komputer berintelijensia untuk menguji

apakah komputer tersebut mampu mengelabui seorang manusia yang menginterogasinya melalui *teletype* (komunikasi berbasis teks jarak jauh). Jika *interrogator* tidak dapat membedakan yang dinterogasi adalah manusia atau komputer, maka komputer berintelijensia tersebut lolos dari *Turning test*. Komputer tersebut perlu memiliki kemampuan : *Natural Language Processing, Knowledge Representation, Automated Reasoning, Machine Learning, Computer Vision, Robotics*. *Turing test* sengaja menghindari interaksi fisik antara *interrogator* dan komputer karena simulasi fisik manusia tidak memerlukan intelijensia.

C. *Thinking rationally* : *the laws of thought approach* Terdapat dua masalah dalam pendekatan ini, yaitu :

1. Tidak mudah untuk membuat pengetahuan informal dan menyatakan pengetahuan tersebut ke dalam *formal term* yang diperlukan oleh notasi logika, khususnya ketika pengetahuan tersebut memiliki kepastian kurang dari 100%.

2. Terdapat perbedaan besar antara dapat memecahkan masalah "dalam prinsip" dan memecahkannya "dalam dunia nyata".

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

Thinking humanly dan *acting humanly* adalah dua definisi dalam arti yang sangat luas. Sampai saat ini, pemikiran manusia yang diluar rasio, yakni reflex 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 yang tidak layak disebut sebagai produk AI.

Definisi *thinking rationally* terasa lebih sempit daripada *acting rationally*. Oleh karena itu, definisi AI yang paling tepat untuk saat ini adalah *acting rationally* dengan pendekatan *rational agent*. Hal ini berdasarkan pemikiran bahwa komputer bisa melakukan penalaran secara logis dan juga bisa melakukan aksi secara rasional berdasarkan hasil penalaran tersebut.

2.3 Jaringan Syaraf Tiruan

Jaringan saraf tiruan (JST) atau umumnya hanya disebut neural network (NN)), adalah jaringan dari sekelompok unit pemroses kecil yang dimodelkan berdasarkan jaringan saraf manusia. JST merupakan sistem adaptif yang dapat mengubah strukturnya untuk memecahkan masalah berdasarkan informasi eksternal maupun internal yang mengalir melalui jaringan tersebut.

Secara sederhana, JST adalah sebuah alat pemodelan data statistik non-linier. JST dapat digunakan untuk memodelkan hubungan yang kompleks antara input dan output untuk menemukan pola-pola pada data (kusumadewi, 2003).

2.3.1 Jaringan Syaraf Tiruan *Feedforward*

Jaringan Syaraf Tiruan *feedforward* adalah sebuah jaringan yang sederhana mempunyai struktur *feedforward* dimana signal bergerak dari input kemudian melewati lapisan tersembunyi dan akhirnya mencapai unit output (mempunyai struktur perilaku yang stabil).

Tipe jaringan *feedforward* mempunyai sel syaraf yang tersusun dari beberapa lapisan. Lapisan input bukan merupakan sel syaraf. Lapisan ini hanya memberi pelayanan dengan mengenalkan suatu nilai dari suatu variabel. Lapisan tersembunyi dan lapisan output sel syaraf terhubung satu sama lain dengan lapisan sebelumnya. Kemungkinan yang timbul adalah adanya hubungan dengan beberapa unit dari lapisan sebelumnya atau terhubung semuanya (lebih baik).

1. Unit input (X_i, i=1,2,…..,n)

    1. Menerima input X_i

    2. Mengirimkannya ke semua unit layer diatasnya (Hidden layer).

2. Unit Hidden (Z_j, j=1,2,….,n)

1. Menghitung semua sinyal input dengan bobotnya :

$$\mathbf{z_in_j = v_{oj} + \Sigma x_i v_{ij}} \quad (2.1)$$

2. Menghitung nilai aktivasi setiap unit hidden sebagai output unit hidden

$$\mathbf{z_j = f(z_in_j)} \quad (2.2)$$

$$f(z_inj) = \frac{1}{1+e^{-z_inj}} \quad (2.3)$$

3. Mengirim nilai aktivasi sebagai input untuk unit output.

3. Unit Output (Y_k, k=1,2,…..,n)

1. Menghitung semua sinyal inputnya dengan bobotnya :

$$\mathbf{y_in_k = w_{ok} + \Sigma z_j w_{jk}} \quad (2.4)$$

2. Menghitung nilai aktivasi setiap unit output sebagai output jaringan.

$$\mathbf{y_k = f(y_in_k)} \quad (2.5)$$

$$f(y_ink) = \frac{1}{1+e^{-y_ink}} \quad (2.6)$$

4. Menghitung faktor δ

$$\delta k = (0 - y = f(y_ink))\ (y = f(y_ink))(1- y = f(y_ink)) \quad (2.7)$$

5. Menghitung error rate

$$\delta net = \delta k * W \quad (2.8)$$

6. Menghitung weight

$$W = Wsebelum + \delta \quad (2.9)$$

Yang termasuk dalam struktur feedforward :

a. *Single-layer perceptron*
b. *Multilayer perceptron*
c. *Radial-basis function networks*
d. *Higher-order networks*
e. *Polynomial learning networks*

2.4 OOP (Object Oriented Programming)

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

Pada saat ini, metode berorientasi objek banyak dipilih karena metodologi lama banyak menimbulkan masalah seperti adanya kesulitan pada saat mentransformasi hasil dari satu tahap pengembangan ke tahap berikutnya, misalnya pada metode pendekatan terstruktur, jenis aplikasi yang dikembangkan saat ini berbeda dengan masa lalu. Aplikasi yang dikembangkan saat ini sangat beragam (aplikasi bisnis, real-time, utility, dan sebagainya) dengan *platform* yang berbeda-beda, sehingga menimbulkan tuntutan kebutuhan metodologi pengembangan yang dapat mengakomodasi ke semua jenis aplikasi tersebut (Sutojo, Mulyanto dan Suhartono, 2010).

Keuntungan menggunakan metodologi berorientasi objek adalah sebagai berikut (Sutojo, Mulyanto dan Suhartono, 2010):

a. Meningkatkan produktivitas

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

b. Kecepatan pengembangan

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

c. Kemudahan pemeliharaan

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

d. Adanya konsistensi

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

e. Meningkatkan kualitas perangkat lunak

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

Berikut beberapa contoh bahasa pemrograman yang mendukung pemrograman berorientasi objek (Sutojo, Mulyanto dan Suhartono, 2010) :

a. Smalltalk

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

b. Bahasa Pemrograman Eiffel

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

c. Bahasa Pemrograman C++

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

d. Bahasa Pemrograman (web) PHP

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

e. Bahasa Pemrograman Java

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

2.5 UML (Unified Modelling 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 1980an. Sekarang UML sudah mulai banyak digunakan oleh para praktisi OOP. UML merupakan dasar bagi perangkat (tool) desain berorientasi objek dari IBM.

UML mendefinisikan diagram-diagram sebagai berikut (Prabowo, 2011):

1). *Use case diagram*
2). *Class diagram*
3). *Activity diagram*
4). *Sequence diagram*

2.5.1 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.

2.5.2 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 (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi).

Class Diagram menggambarkan struktur dan deskripsi class, package dan objek beserta hubungan satu sama lain seperti containment, pewarisan, asosiasi, dan lain-lain.

2.5.3 Activity Diagram

Activity diagram menggambarkan berbagai alur aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alur berawal, *decision* yang mungkin terjadi, dan bagaimana mereka 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.

2.5.4 Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pemain, *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.

2.6 NetBeans

NetBeans merupakan sebuah proyek kode terbuka yang sukses dengan pengguna yang sangat luas, komunitas yang terus tumbuh, dan memiliki hampir 100 mitra (dan terus bertambah!). Sun Microsystems mendirikan proyek kode terbuka NetBeans pada bulan Juni 2000 dan terus menjadi sponsor utama.

2.6.1 Pengertian NetBeans

NetBeans adalah Integrated Development Environment (IDE) berbasiskan Java dari Sun Microsystems yang berjalan di atas Swing. Swing sebuah teknologi Java untuk pengembangan aplikasi Desktop yang dapat bejalan di berbagai macam platforms seperti Windows, Linux, Mac OS X and Solaris.

Suatu IDE adalah lingkup pemrograman yang diintegrasikan kedalam suatu aplikasi perangkat lunak yang menyediakan pembangun Graphic User

Interface (GUI), suatu text atau kode editor, suatu compiler atau interpreter dan suatu debugger.

Netbeans merupakan software development yang Open Source, dengan kata lain software ini di bawah pengembangan bersama, bebas biaya.

2.6.2 Sejarah NetBeans

Menurut situs yang diunduh pada tanggal 5 Oktober 2014 netbeans.org Pengembangan NetBeans diawali dari Xelfi, sebuah proyek mahasiswa tahun 1997 di bawah bimbingan Fakultas Matematika dan Fisika Universitas Charles, Praha. Sebuah perusahaan kemudian dibentuk untuk proyek tersebut dan menghasilkan versi komersial NetBeans IDE hingga kemudian dibeli oleh Sun Microsystem pada tahun 1999. Sun kemudian menjadikan NetBeans open source pada bulan Juni tahun 2000. Sejak itu komunitas NetBeans terus berkembang.

NetBeans mengacu pada dua hal, yakni platform untuk pengembangan aplikasi desktop java, dan sebuah Integrated Development Environment (IDE) yang dibangun menggunakan platform NetBeans.

Platform NetBeans memungkinkan aplikasi dibangun dari sekumpulan komponen perangkat lunak moduler yang disebut ‘modul’. Sebuah modul adalah suatu arsip Java (Java archive) yang memuat kelas-kelas Java untuk berinetraksi dengan NetBeans Open API dan file manifestasi yang mengidentifikasinya sebagai modul. Aplikasi yang dibangun dengan modul-modul dapat dikembangkan dengan menambahkan modul-modul baru. Karena modul dapat dikembangkan secara independen, aplikasi berbasis platform NetBeans dapat dengan mudah dikembangkan oleh pihak ketiga secara mudah dan powerful.