BAB II
LANDASAN TEORI

II.1 Game

Game berasal dari bahasa Inggris yang berarti permainan. Permainan adalah suatu aktivitas rekreasi yang dapat dimainkan untuk menciptakan kesenangan, tujuan dari permainan itu sendiri adalah untuk *refreshing.*Biasanya permainan dilakukan sendiri ataupun bersama-sama.Teori permainan mula-mula dikembangkan oleh seorang ahli matematika perancis bernamaEmile Borel pada tahun 1921. Kemudian, Jhon Von Neumann dan Oscar morgensten mengembangkan lebih lanjut sebagai alat untuk merumuskan perilaku ekonomi yang bersaing.Aplikasi-aplikasi nyata yang palingsukses dari teori permainan banyak ditemukan dalam militer. Tetapi denganberkembangnya dunia usaha (bisnis) yang semakin bersaing dan terbatasnyasumber daya serta saling ketergantunga social, ekonomi, dan ekologi yangsemakin besar, akan meningkatkan pentingnya aplikasi-aplikasi teori permainan. Kontrak dan program tawar menawar serta keputusan-keputusan penetapan harga adalah contoh penggunaan teori permainan yang semakin meluas [***6***].

Model-model teori permainan dapat diklasifikasikan dengan sejumlah cara,seperti jumlah pemain, jumlah keuntungan dan kerugian dan jumlah strategi yangdigunakan dalam permainan. Sebagai contoh, bila jumlah pemain adalah dua,permainan disebut sebagai permainan dua-pemain. Begitu juga, bila jumlahpemain adalah N (dengan $N \geq 3$), permainan disebut permainan N-pemain.

II.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 umunya 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 **[*7*]**.

II.1.2 Jenis-jenis Game

Berikut klasifikasi *game* berdasarkan genre permainannya **[*8*]** :

1. Aksi *Shooting*

   Video game jenis ini sangat memerlukan kecepatan refleks, koordinasi mata dan tangan, juga timing, inti dari *game* jenis ini adalah tembak, tembak dan tembak. Termasuk didalamnya tedapat :

   a. *First person shooter (FPS)* seperti *Counter Strike* dan *Call of Duty.*
      Drive n shoot, menggunakan unsur simulasi kendaraan tetapi tetap dengan tujuan utama menembak dan menghancurkan lawan, contoh : *Spy Hunter, Rock and Roll Racing, Road Rash.*
   b. *Shoot em up,* seperti Raiden, 1942, dan gradius.
   c. Beat em up (tonjok hajar) seperti *Double Dragon* dan *Final Fight*, lalu *hack and slash* (tusuk tebas) seperti *Shinobi* dan *Legend of Kage.*
   d. *Light gun shooting*, yang menggunakan alat yang umumnya berbentuk seperti senjata, seperti *Virtua Cop* dan *Time Crisis.*
   e. *Fighting* (pertarungan) Ada yang mengelompokan *video game fighting* di bagian Aksi, jenis ini memang memerlukan kecepatan refleks dan koordinasi mata-tangan, tetapi inti dari *game* ini adalah penguasaan jurus (hafal caranya dan lancar mengeksekusinya).

2. Petualangan.

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

3. Simulasi, Konstruksi dan manajemen.

   Video Game jenis ini seringkali menggambarkan dunia di dalamnya sedekat mungkin dengan dunia nyata dan memperhatikan dengan detil berbagai faktor. *Video Game* jenis ini membuat pemain harus berpikir untuk mendirikan, membangun dan mengatasi masalah dengan menggunakan dana yang terbatas. Contoh: *Sim City, The Sims, Tamagotchi.*

4. *Role Playing.*

   Video game jenis ini sesuai dengan terjemahannya, bermain peran, memiliki penekanan pada tokoh/peran perwakilan pemain di dalam permainan, yang biasanya adalah tokoh utamanya, dimana siring kita memainkannya, karakter tersebut dapat berubah dan berkembang ke arah yang diinginkan pemain (biasanya menjadi semakin hebat, semakin kuat, semakin berpengaruh, dll) dalam berbagai parameter yang biasanya ditentukan dengan naiknya *level*, baik dari status kepintaran, kecepatan dan kekuatan karakter. seperti *Final Fantasy, Dragon Quest* dan *Xenogears.Never Winter Nights, baldurs gate, Elder Scroll,* dan *Fallout.*

5. *Puzzle.*

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

petualangan maupun *game* edukasi. *Tetris, Minesweeper, Bejeweled, Sokoban* dan *Bomberman.*

6. Simulasi kendaraan.

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

II.2 AI (Artificial Intelligence)

Salah satu unsur yang berperan penting dalam sebuah *game* adalah kecerdasan buatan. Dengan kecerdasan buatan, elemen-elemen dalam *game*seperti musuh biasa berinteraksi dengan *player* dalam berbagai hal, mulai dari bertarung hinggaberjalan. khusus proses berjalan, algoritma pathfinding adalah algoritma yangdapat dimanfaatkan membantu musuh menemukan alur jalannya.

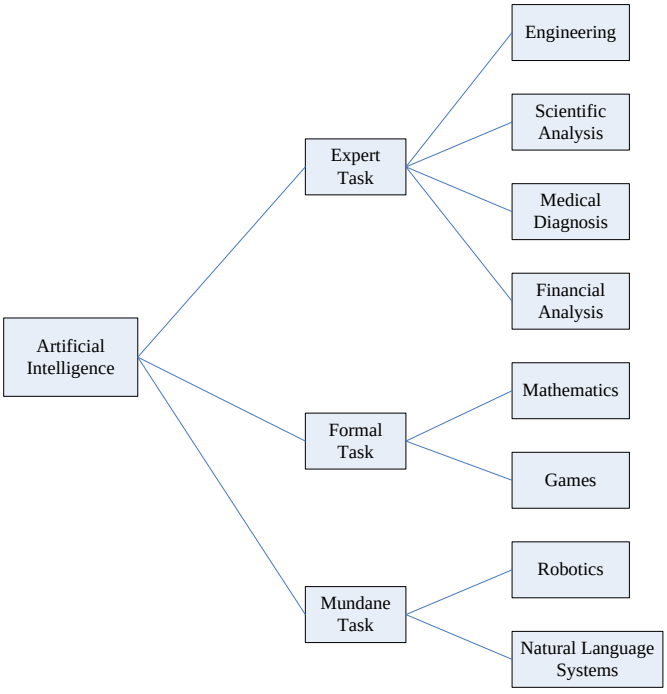
Kecerdasan buatan atau *Artificial Intelligence* (AI) merupakan cabang dari ilmu komputer yang berhubungan dengan pengotomatisan tingkah laku cerdas.Kecerdasan buatan didasarkan pada teori suara (*sound theoretical*) dan prinsip-prinsip aplikasi dari bidangnya.Prinsip-prinsip ini meliputi struktur data yang digunakan dalam representasi pengetahuanalgoritama yang diperlukan untuk mengaplikasikan pengetahuan tersebut serta bahasa dan teknik pemrograman yang digunakan dalam mengimplementasikannya.

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 II.1 Bidang-bidang tugas (*task domains*) dari AI **[*9*]**

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:
   a. Persepsi.
   b. Bahasa alami.
   c. Robot *control*.

Aplikasi kecerdasan buatan memiliki dua bagian utama yaitu :

1. Basis Pengetahuan (*Knowledge Base*): berisi fakta-fakta,teori,pemikiran dan hubungan antara satu dengan lainnya.
2. Motor Inferensi (*Inference Engine*): kemampuan menarik kesimpulan berdasarikan pengalaman.

II.3 **Algoritma A***

Algoritma ini merupakan algoritma *Best First Search* yang menggabungkan *Uniform Cost Search dan Greedy Best-First Search*. Biaya yang diperhitungkan didapat dari biaya sebenarnya ditambah dengan biaya perkiraan. Dalam notasi matematika dituliskan sebagai $f(n) = g(n) + h(n)$.Dengan perhitungan biaya seperti ini, Algoritma A* adalah compelete dan optimal.

Algorima A*, merupakan salah satu contoh algoritma pencarian yang cukuppopular di dunia. Jika mengetikkan Algoritma A* pada sebuah mesin pencari, seperti google.com, maka akan ditemukan lebih dari sepuluh ribu literatur mengenai algoritma A*.

Beberapa terminologi dasar yang terdapat pada algoritma ini adalah *starting point*, simpul (*nodes*), A, *open list*, *closed list*, harga (*cost*), halangan (*unwalkable*). *Starting point* adalah sebuah terminologi untuk posisi awal sebuah benda.A adalah simpul yang sedang dijalankan dalam algortima pencarian jalan terpendek. Simpul adalah petak-petak kecil sebagai representasi dari area *pathfinding*.Bentuknya dapat berupa persegi, lingkaran, maupun segitiga.*open list* adalah tempat menyimpan data simpul yang mungkin diakses dari *starting point* maupun simpul yang sedang dijalankan. *Closed list* adalah tempat menyimpan data simpul sebelum A yang juga merupakan bagian dari jalur terpendek yang telah berhasil didapatkan. Harga (F) adalah nilai yang diperoleh dari penjumlahan nilai G, jumlah nilai tiap simpul dalam jalur terpendek dari *starting point* ke A, dan H, jumlah nilai perkiraan dari sebuah simpul ke simpul tujuan. Simpul tujuan yaitu simpul yang dituju.Rintangan adalah sebuah atribut yang menyatakan bahwa sebuah simpul tidak dapat dilalui oleh A.

Prinsip algoritma ini adalah mencari jalur terpendek dari sebuah simpul awal (*starting point)* menuju simpul tujuan dengan memperhatikan harga (F) terkecil. Diawali dengan menempatkan A pada *starting point*, kemudian memasukkan seluruh simpul yang bertetangga dan tidak memilik atribut rintangan dengan A ke dalam *open list*. Kemudian mencari nilai H terkecil dari simpul-simpul dalam *open list* tersebut. Kemudian memindahkan A ke simpul yang memiliki nilai H terkecil. Simpul sebelum A disimpan sebagai *parent* dari A dan dimasukkan ke dalam *closed list*. Jika terdapat simpul lain yangbertetangga dengan A (yang sudah berpindah) namun belum termasuk kedalam anggota *open list*, maka masukkan simpul-simpul tersebut ke dalam *open list.* Setelah itu, bandingkan nilai G yang ada dengan nilai G sebelumnya (pada langkah awal, tidak perlu dilakukan perbandingan nilai G). Jika nilai G sebelumnya lebih kecil maka A kembali ke posisi awal. Simpul yang pernah dicoba dimasukkan ke dalam *closed list.* Hal terebut dilakukan berulangulang hingga terdapat solusi atau tidaka ada lagi simpul lain yang berada pada *open list.*

Terdapat beberapa hal yang perlu didefinisikan terlebih dahulu dalam kasus *game pathfinding* dengan penerapan algoritma A* *(A Star)*. Adapun istilah-istilah

yang akan dibahas yaitu *path, open list, closed list,* nilai f, g dan n.

Algoritma A* menggunakan dua senarai yaitu *OPEN* dan *CLOSED.OPEN* adalah senarai (*list*) yang digunakan untuk menyimpan simpul-simpul yangpernah dibangkitkan dan nilai heuristiknya telah dihitung tetapi belum terpilih sebagai simpul terbaik (*best node*) dengan kata lain, *OPEN* berisi simpul-simpul masih memiliki peluang untuk terpilih sebagai simpul terbaik, sedangkan *CLOSED* adalah senarai untuk menyimpan simpul-simpul yang sudah pernahdibangkitkan dan sudah pernah terpilih sebagai simpul terbaik. Artinya, *CLOSED* berisi simpul-simpul yang tidak mungkin terpilih sebagai simpul terbaik (peluang untuk terpilih sudah tertutup).

1. *OPEN LIST* adalah *list* yang menyimpan kemungkinan *path* yang akan diperiksa. *OPEN LIST* dibuat terurut berdasarkan nilai f. *OPEN LIST* digunakan untuk menentukan secara selektif (berdasarkan nilai f) jalan yang dikira lebih dekat menuju pada *path* tujuan. *OPEN* berisi simpul-simpul yang masih memiliki peluang untuk terpilih sebagai simpul terbaik (*best node*).
2. *CLOSED* adalah senarai (*list*) untuk menyimpan simpul-simpul yang sudahpernah dibangkitkan dan sudah pernah terpilih sebagai simpul terbaik (*bestnode*) atau senarai yang menyimpan jalan yang sudah diperiksa dari *open list*.Artinya, *CLOSED* berisi simpul-simpul yang tidak mungkin terpilih sebagai simpul terbaik (peluang untuk terpilih sudah tertutup). Kedua list (*OPENLIST* dan *CLOSED LIST*) ini bertujuan juga untuk menghindari penelusuranberkali-kali jalan (*rute*) yang memang sudah diidentifikasi agar tidak masuk kembali ke dalam *OPEN LIST*.
3. Nilai F adalah *cost* perkiraan suatu *path* yang teridentifikasi. Nilai F merupakan hasil dari f(n).
4. Nilai G hasil dari fungsi g(n), adalah banyaknya langkah yang diperlukan untuk menuju ke *path* sekarang.
5. Setiap simpul *(node)* harus memiliki informasi nilai h(n), yaitu estimasi harga simpul tersebut dihitung dari simpul tujuan yang hasilnya menjadi nilai H.

Fungsi f sebagai estimasi fungsi evaluasi terhadap *node* n, dapat dituliskan :

$$f(n) = g(n) + h(n)$$

dengan :

$f(n)$ = fungsi evaluasi (jumlah $g(n)$ dengan $h(n)$)

$g(n)$ = biaya *(cost)* yang dikeluarkan dari keadaan awal sampai keadaan n

$h(n)$ = estimasi biaya untuk sampai pada suatu tujuan mulai dari n

Perhitungan nilai H digunakan fungsi *heuristic*, metode yang digunakan di dalam contoh ini adalah metode *Manhattan* dimana perhitungan jumlah *node* hanya yang bergerak secara *vertical* dan *horizontal* menuju tujuannya serta mengabaikan penghalang atau *collision*, yang kemudian nilainya dikalikan dengan 10. Atau dirumuskan dengan **[*10*]**:

H = 10*(abs(*current*X-*target*X) + abs(*current*Y-*target*Y))

Di bawah ini merupakan *pseudocode* dari algoritma *A**.

Tabel II.1 Pseudocode Algoritma A*

```
1. fungsi A*(awal,tujuan)
2. var himpunan_tertutup <- himpunan_kosong
3. var q <- buat_antrian(titk_awal)
4. while (q tidak kosong)
5. var p <- hapus_elemen_pertama(q)
6. var x <- titik_akhir_dari_p
7. if (x = tujuan)
8. return p
9. for (tiap y pada suksesor(p))
10.   if (y bukan anggota himpunan_tertutup)
11.   masuk_antrian(q, y)
12.   tambahkan y ke himpunan_tertutup
13.   endif
14.   endfor
15.   endwhile
16.   return tidak_ada_jalur
```

II.4 Analisis Sistem Berbasis Objek

Analisis dan disain berorientasi objek adalah cara baru dalam memikirkan suatu masalah dengan mengguna kan model yang dibuat menurut konsep sekitar dunia nyata. Dasar pembuatan adalah objek,yang merupakan kombinasi antara struktur data dan perilaku dalam satu entitas

II.4.1 *Object Oriented Programming*

OOP (*Object Oriented Programming*) adalah pemodelan atau alat untuk mendesain sebuah *game.*Desainnya bisa dari objek karakter ataupun dari programnya.

Objek adalah kesatuan entitas yang memiliki sifat dan tingkah laku. Dalamkehidupan 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 **[*11*]**:

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*.

II.4.2 UML (Unified Modeling Language)

UML (Unified Modeling Language) adalah sebuah bahasa yang berdasarkan grafik/gambar untuk memvisualisasi, menspesifikasikan, membangun, dan pendokumentasian dari sebuah sistem pengembangan *software* berbasis *OO* (*Object-Oriented*). *UML* sendiri juga memberikan standar penulisan sebuah sistem *blue print,*yang meliputi konsep bisnis proses, penulisan kelas-kelas dalam bahasa program yang spesifik, skema *database*, dan komponen-komponen yang diperlukan dalam sistem *software*.

Metode *objek-oriented* banyak digunakan dalam pengembangan perangkat lunak.*Unified Modeling Language* (UML) adalah bahasa yang dapat digunakan untuk spesifikasi, visualisasi dan dokumentasi sistem *software* pada fase pengembangan.*UML* merupakan unifikasi metode-metode *Booch*, *OMT* dan *Objectory* serta beberapa metode lainnya, *de facto UML* merupakan standar bidang analisis dan desain sistem *objek-oriented*. *View* adalah jendela (*window*) yang merupakan aspek pandang *UML* terhadap sistem. *UML* memperkenalkan lima buah *view* untuk memandang sistem yaitu: *Use-Case View*, *Logical View*, *Componentview*, *Deployment Viewdan Concurrency View* (Eriksson dan Penker, 1998). Booch (1998) menyebut *view* ini sebagai arsitektur 4+1´ dan menyebut *Concurrency View* sebagai *Process View*. Tahap pengembangan sistem perangkat lunak di dalam UML meliputi:

Analisis Kebutuhan (*Requirement Analysis*), Analisis Sistem (*Analysis*), Desain (*Design*), Implementasi (Implementation) dan *Testing* **[*12*]**

II.4.1.1 *Use Case Diagram*

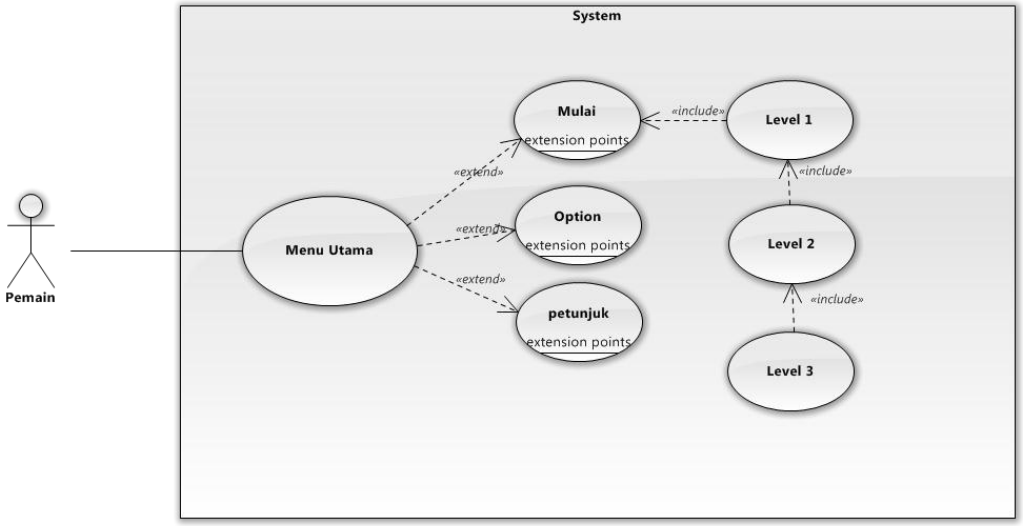
Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yangditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah *use case*merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuahpekerjaan tertentu, misalnya *login* ke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

Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalamdirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *usecase* yang meng-*include* dieksekusi secara normal.

Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitasdapat 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 merupakanspesialisasi dari yang lain **[*13*]**. contoh *use case* dapat dilihat pada gambar II.1

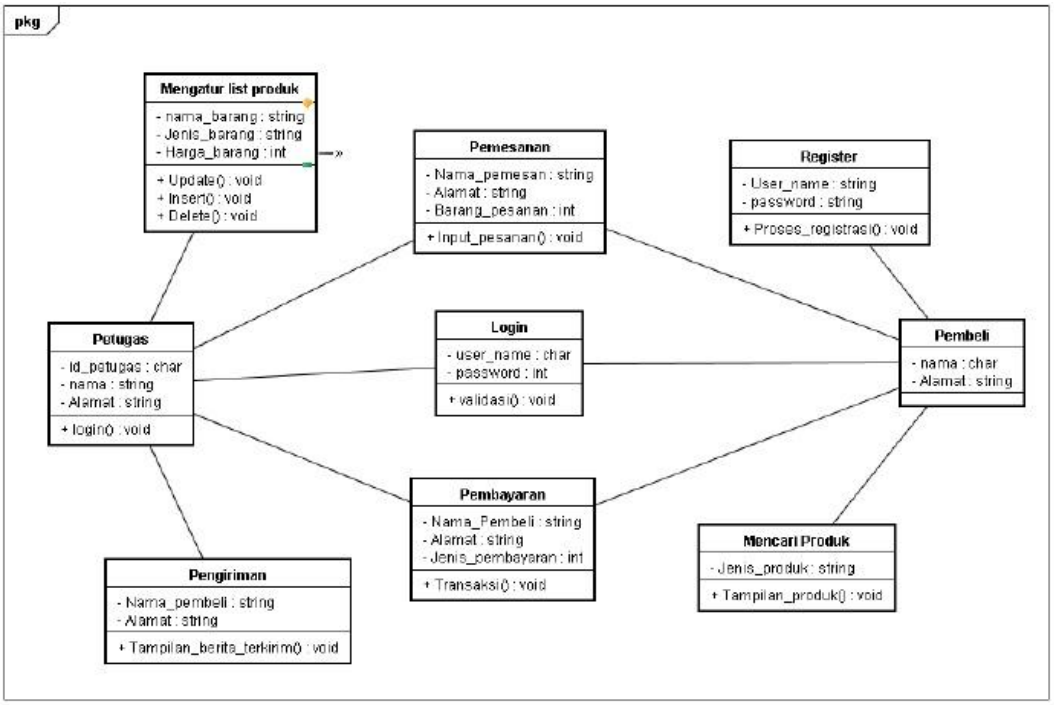


Gambar II.2 Diagram Use Case

II.4.1.2 *Class Diagram*

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek danmerupakan 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 hubungansatu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain [13]. Gambar II.3 menunjukkan contoh *class diagram*

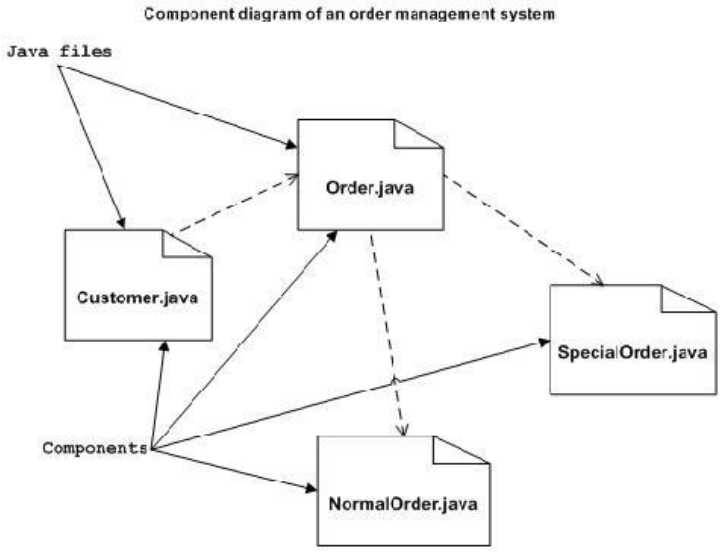


Gambar II.3 *Class Diagram*

II.4.1.3 *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 darikomponen-komponen yang lebih kecil.Komponen dapat juga berupa *interface*, yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain [13]. Contoh *component diagram* ditunjukkan pada gambar II.4.

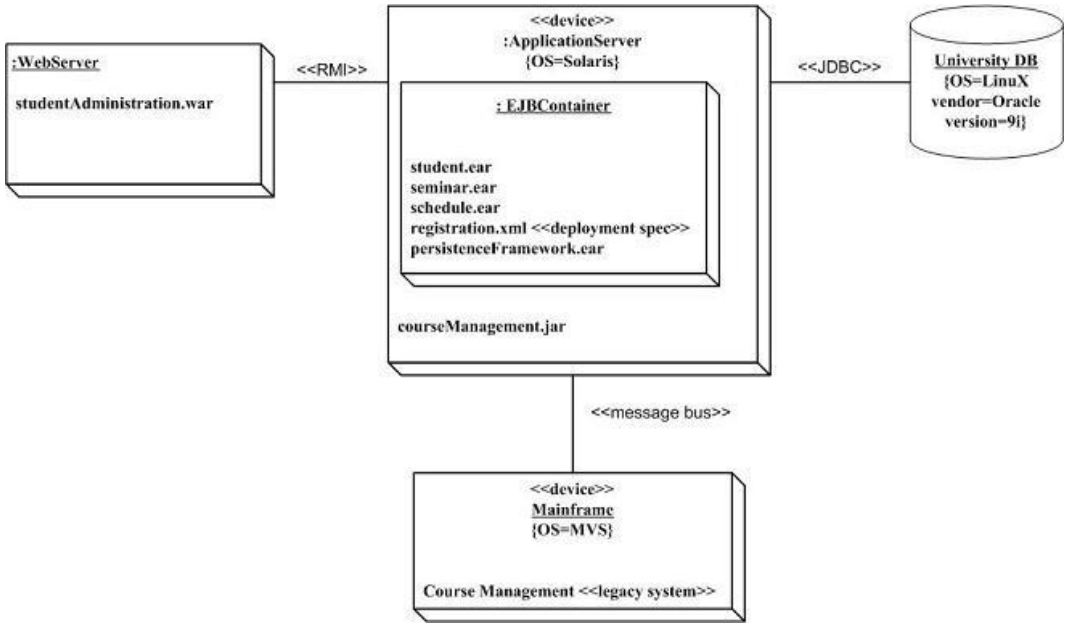


Gambar II.4 Component Diagram

II.4.1.4 *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 [13]. Gambar II.5 menunjukkan contoh *deployment diagram*.

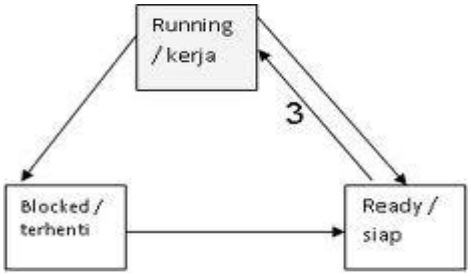


Gambar II.5 Deployment Diagram

II.4.1.5 State 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 namasesuai kondisinya saat itu. Transisi antar *state* umumnya memiliki kondisi *guard* yang merupakan syarat terjadinya transisi yang bersangkutan, dituliskan dalam kurung siku. *Action* yang dilakukansebagai akibat dari *event* tertentu dituliskan dengan diawali garis miring [13]. Contoh *state diagram* ada pada gambar II.6

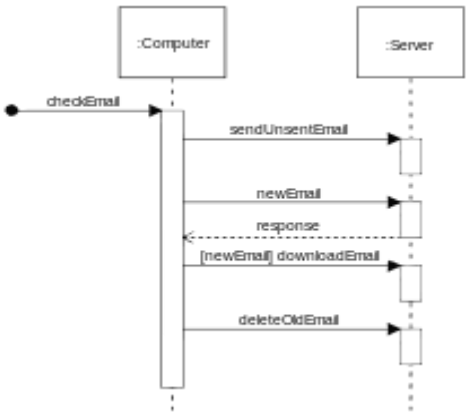


Gambar II.6 State Diagram

II.4.1.6 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 dariapa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan [13]. Salah satu contoh *sequence diagram* digambarkan pada gambar II.6.

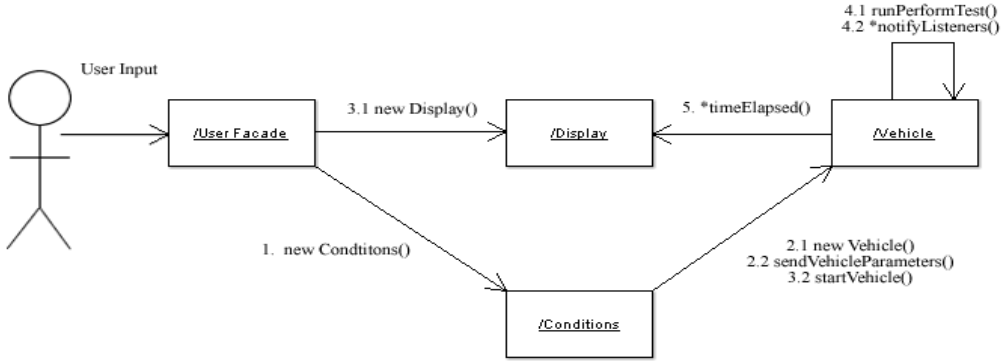


Gambar II.7 Sequence Diagram

II.4.1.7 *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 [13]. Gambar II.8 menunjukkan contoh *collaboration diagram*.



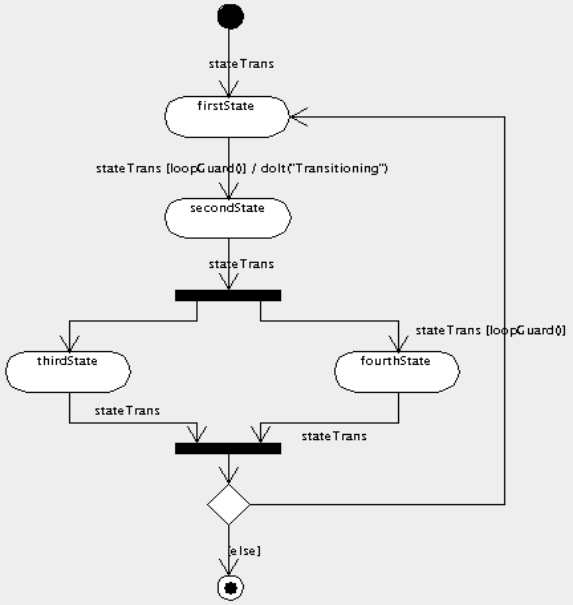
Gambar II.8 Collaboration Diagram

II.4.1.8 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 [13].

Contoh *activity diagram* bisa ditunjukkan pada gambar II.9



Gambar II.9 Activity Diagram

II.5 Bahasa Pemrograman C#

C# merupakan sebuah bahasa pemrograman yang berorientasi objek yangdikembangkan oleh Microsoft sebagai bagian dari inisiatif kerangka .NET *Framework* **[*13*].**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 *clien-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 **[14]**.

II.6 Java

Java adalah sebuah bahasa pemrograman pada komputer sama seperti pendahulunya *C++*. Bahasa pemrograman ini dikembangkan oleh *Sun microsystems* yang dikembangkan pada tahun 1995 . Awalnya *java*diciptakan pada tahun 1991 oleh Patrick Naughton, Mike Sheridan , James Gosling dan Bill Joy beserta *programer* dari *Sun Microsystems* . Uniknya nama *java*diambil dari nama pulau jawa (bahasa inggris dari jawa) karena James Gosling " Bapak *Java*" menyukai kopi tubruk yang berasal dari pulau Jawa . Jadi kita perlu berbangga menjadi warga negara Indonesia **[*15*]**.

Java sendiri merupakan bahasa pemrograman yang berbasis objek , maksudnya adalah semua aspek yang terdapat pada *java*adalah objek sehingga sangat memudahkan untuk mendesain , membuat dan mengembangkan program *java*dengan cepat . Kelebihan ini membuat program *java*menjadi mudah untuk digunakan oleh banyak orang . Bahasa pemrograman ini juga dapat digunakan oleh banyak sistem operasi , seperti : *Microsoft Windows*, *Linux*, *Mac OS*, dan *Sun Solaris* (*Multiplatform*) .

Kelebihan *Java*:

- Berorientasi objek, Memudahkan untuk mendesign dan mengembangkan program dengan cepat dan teliti, sehinnga mudah digunakan. Salah satu bahasa pemrograman yang berorientasi objek secara murni. Mirip *C++*, Mempunyai sintaks yang mirip dengan bahasa pemrograman *C++*. Sehingga para pengguna *C++* banyak yang hijrah menggunakan *Java*.
- Multiplatform, Dapat digunakan dibanyak sistem operasi. Perkembangan yang luas, Dari game sampai sistem operasi handphone menggunakan program *java*.

Kekurangan *Java*:

- Proses *Compile*, Mengharuskan pengguna mengcompile programnya sebelum dijalankan, berbeda dengan bahasa pemrograman *python* yang tidak perlu mengcompile terlebih dahulu .
- Penggunaan Memori yang besar, Berbeda dengan bahasa pemrograman lain yang hanya membutuhkan memori sedikit

SunMicrosystem mendefinisikan 3 (tiga) edisi *java*, yaitu :

1. J2ME (*Java*Micro Edition)
   Edisi ini ditunjukan bagi lingkungan dengan sumber daya terbatas, seperti *smartcard*, ponsel, dan *PDA*.
2. J2SE (*Java*2 Standard Edition)
   Edisi ini ditunjukan bagi lingkungan workstation, seperti pembuatan aplikasi-aplikasi *desktop*.
3. J2EE (*Java*2 Enterprise Edition)
   Edisi ini ditunjukan bagi lingkungan Internet atau aplikasi terdistribusi dalam skala besar.

Perbedaan setiap edisi meliputi fitur-fitur bahasa yang didukung dan *API* yang tersedia. Berdasarkan tingkatannya, edisi yang lebih tinggi mampu mengemulasi edisi yang lebih rendah. Adapun urutan edisi yang tertinggi ke rendah adalah J2EE, J2SE, J2ME.

Sedangkan pada Versi *java*kita cermati dari versi-versi *Java*yang telah dirilis *Sun Microsystems* yang menggunakan dua jenis versi, yaitu versi produk dan versi developer. Seperti kita ketahui, versi terbaru saat ini adalah versi 6 (Versi produk) atau versi 1.6.5 (Versi developer), dengan nama kode *Mustang*. Sejak tahun 2006, Sun juga menyederhanakan penamaan platform dengan tujuan untuk mencerminkan tingkat kematangan, stabilitas, skalabilitas, dan sekuriti yang lebih baik.Jika penamaan versi sebelumnya adalah *Java*2 *Platform, Standard Edition* 5.0 (J2SE 5.0), maka sekarang disederhanakan menjadi *Java Platform, Standard Edition* 6 (*Java*SE 6, atau lebih sering disebut *Java* 6).

II.5 Unity

Unity Game Engine adalah *software* atau *game engine* yang digunakan untuk membuat *video game* berbasis dua atau tiga dimensi dan dapat digunakan secara gratis. Selain untuk membuat *game*, *unity* juga dapat digunakan untuk membuat konten yang interaktif lainnya seperti, visual arsitektur dan *real-time* 3D animasi.

Unity adalah sebuah sebuah *tool* yang terintegrasi untuk membuat *game*, arsitektur bangunan dan simulasi.*Unity* bisa digunakan untuk *game*s PC dan *games online*. Untuk *games online* diperlukan sebuah plugin, yaitu *Unity Web Player*, yang sama halnya dengan *flashplayer* pada *browser*. Bahasa pemrograman yang digunakan bermacam-macam, mulai dari *javascript*, C#, dan *boo* **[*16*]**.

II.6 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 **[*17*].**