BAB 2
LANDASAN TEORI

2.1 Kecerdasan Buatan (*Artificial Intelligence*)

Menurut Prof. Elder Dadios [13] kecerdasan buatan adalah bagian dari ilmu komputer yang mempelajari bagaimana membuat mesin (komputer) dapat melakukan pekerjaan seperti manusia bahkan bisa lebih baik dibandingkan manusia. Agar mesin (komputer) bisa cerdas maka harus diberi bekal pengetahuan dan mempunyai kemampuan untuk menalar. Ada dua bagian utama yang dibutuhkan aplikasi kecerdasan buatan yaitu :

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

Kecerdasan buatan mempunyai kelebihan dibanding dengan kecerdasan alami antara lain :

1. Lebih bersifat permanen. Sebab kecerdasan buatan tidak berubah selama sistem komputer dan program tidak mengubahnya.
2. Lebih mudah diduplikasi dan disebarkan.
3. Lebih murah, karena menyediakan layanan komputer yang lebih mudah dan murah dibandingkan mendatangkan seseorang untuk mengerjakan sejumlah pekerjaan dalam jangka waktu yang lama.
4. Bersifat konsisten dan teliti.
5. Dapat mengerjakan *task* lebih cepat.

2.1.1 Algoritma Logika Fuzzy

Logika fuzzy adalah salah satu komponen pembentuk *soft computing*. Logika fuzzy pertama kali diperkenalkan oleh Prof Lotfi A. Zadeh pada tahun 1965 [14]. Dasar logika fuzzy adalah teori himpunan fuzzy, peranan derajat keanggotaan sebagai penentu elemen dalam suatu himpunan. Nilai keanggotaan atau derajat keanggotaan atau *membership function* menjadi ciri utama dalam penalaran dengan logika *fuzzy* tersebut [2]. Dalam prosesnya logika fuzzy memiliki beberapa tahapan diantaranya :

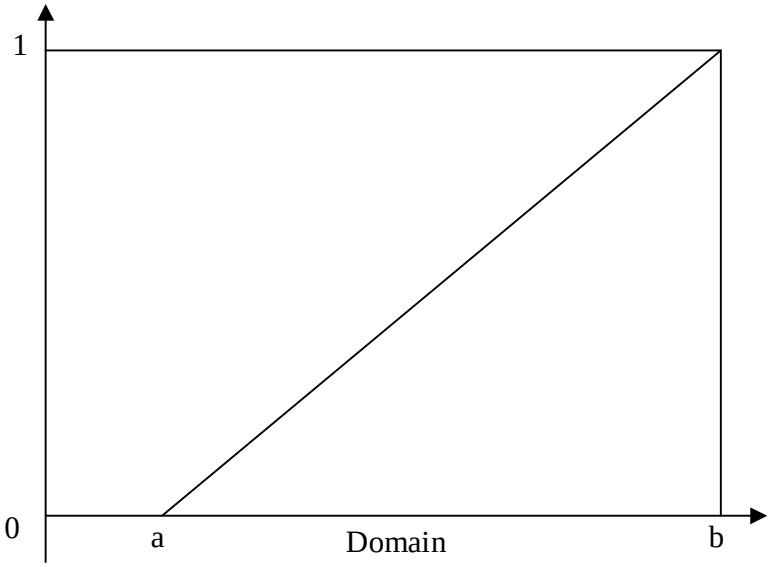
1. Proses Fuzzyfikasi
   Artinya himpunan fuzzy dan penentuan derajat keanggotaan dari *crisp input* pada sebuah himpunan fuzzy.
2. Proses Inferensi
   Merupakan evaluasi kaidah/aturan/*rule* fuzzy untuk menghasilkan *output* dari setiap *rule*.
3. Komposisi
   Merupakan kombinasi dari keluaran semua *rule*.
4. Proses Defuzzikasi
   Merupakan perhitungan hasil kombinasi dari *output* setiap *rule* yang menghasil *crisp output*.

2.1.1.1 Fungsi Keanggotaan Fuzzy

Fungsi keanggotaan atau *Membership Function* menurut Kusumadewi [2] adalah suatu kurva yang menunjukan pemetaan titik-titik input ke dalam nilai atau derajat keanggotaanya yang memiliki interval antara 0 dan 1. Salah satu cara untuk mendapatkan keanggotaan adalah melalui pendekatan fungsi. Ada beberapa fungsi yang bisa digunakan antara lain :

a. Representasi Linear

Ada dua kemungkinan himpunan fuzzy linear yaitu kenaikan himpunan dimulai pada domain yang memiliki derajat keanggotaan nol [0] bergerak ke kanan menuju nilai domain yang memiliki derajat keanggotaan lebih tinggi. Berikut adalah gambar contoh representasi linear naik pada gambar 2.1

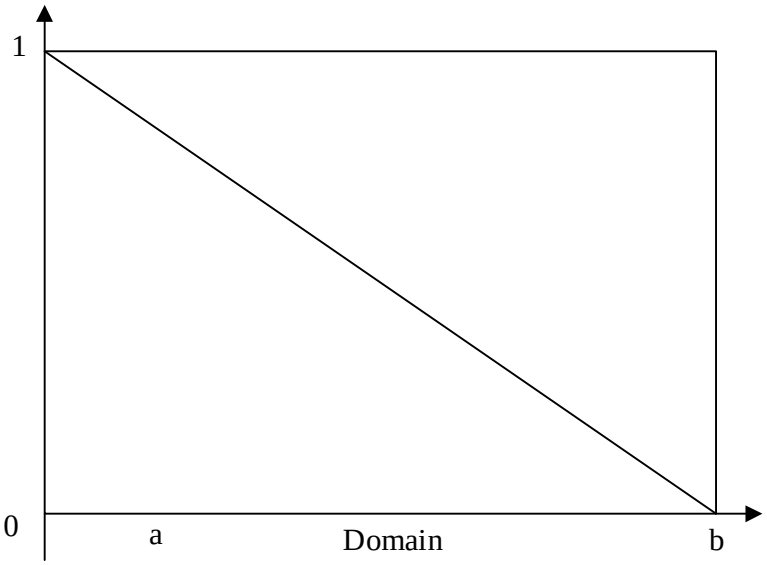


Gambar 2.1 Contoh Representasi Linear Naik

Fungsi keanggotaan pada contoh representasi linear naik adalah :

$$\mu[x] = \begin{cases} 0; & x \leq a \\ \dfrac{x-a}{b-a}; & a \leq x \leq b \\ 1; & x \geq b \end{cases}$$

Sedangkan untuk representasi linear turun garis lurus dimulai dari nilai domain dengan derajat keanggotaan tertinggi dari sisi kiri, kemudian bergerak menurun ke nilai domain yang memiliki derajat keanggotaan rendah. Berikut adalah gambar contoh representasi linear turun pada gambar 2.2.



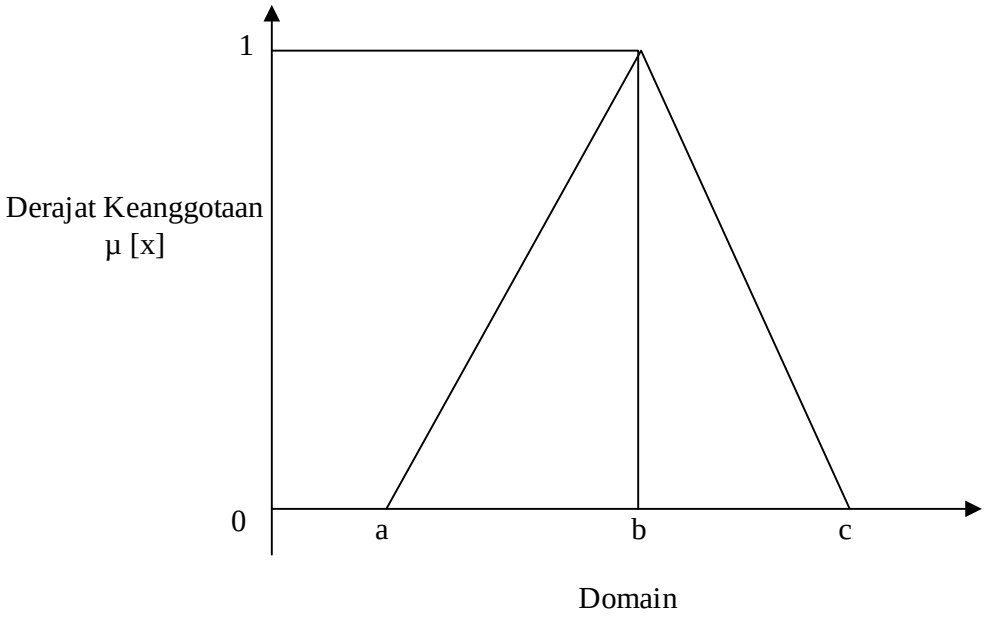
Gambar 2.2 Contoh Representasi Linear Turun

Fungsi keanggotaan pada contoh representasi linear turun adalah :

$$\mu[x] \begin{cases} \frac{x-a}{x-b}; & a \leq x \leq b \\ 0; & x \geq b \end{cases}$$

b. Representasi Kurva Segitiga

Kurva segitiga pada dasarnya merupakan gabungan antara dua garis linear. Berikut adalah gambar contoh representasi kurva segitiga pada gambar 2.3.



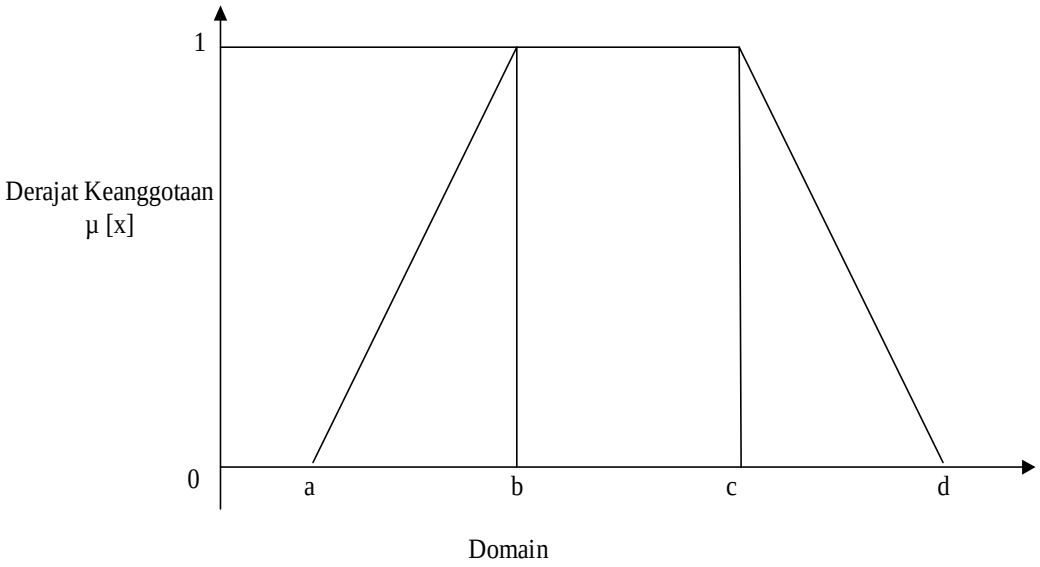
Gambar 2.3 Contoh Representasi Kurva Segitiga

Fungsi keanggotaan dari contoh representasi kurva segitiga adalah :

$$\mu[x] = \begin{cases} 0; & x \leq a \; atau \; x \leq c \\ \dfrac{(x-a)}{(x-b)}; & a \leq x \leq b \\ \dfrac{(c-x)}{(c-b)}; & b \leq x \leq c \end{cases}$$

c. Representasi Kurva Trapesium

Kurva trapesium pada dasarnya bentuknya seperti trapesium hanya saja ada beberapa titik yang memiliki nilai keanggotaan 1. Berikut adalah gambar contoh representasi trapesium pada gambar 2.4.



Gambar 2.4 Contoh Representasi Kurva Trapesium

Fungsi keanggotaan dari contoh representasi kurva trapesium adalah :

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

2.1.1.2 Algoritma Logika Fuzzy Tsukamoto

Algoritma logika fuzzy Tsukamoto merupakan algoritma perluasan dari penalaran monoton [1]. Pada dasarnya metode Tsukamoto mengaplikasikan penalaran monoton pada setiap aturannya. Karena menggunakan konsep dasar penalaran monoton setiap konsekuen pada aturan yang berbentuk IF-THEN harus direpresentasikan dengan suatu hinpunan fuzzy dengan fungsi keanggotaan monoton. *Output* hasil tiap-tiap aturan diberikan secara tegas (*crisp*) berdasarkan α-predikat.

2.1.1.3 Algoritma Logika Fuzzy Sugeno

Algoritma logika fuzzy Sugeno merupakan metode yang penalarannya menghasilkan *output* bukan berupa himpunan fuzzy tapi berupa konstanta atau persamaan linier [1]. Pada algoritma Sugeno terdapat dua jenis model yaitu model fuzzy Sugeno Orde-Nol dan Model Fuzzy Sugeno Orde-Satu.

2.1.1.4 Algoritma Logika Fuzzy Mamdani

Logika fuzzy Mamdani sering juga dikenal dengan nama metode Max-Min. Metode ini pertama kali Ebrahim Mamdani pada tahun 1975 [14]. Berikut adalah tahapan-tahapan dari metode Mamdani :

1. Pembentukan Himpunan Fuzzy

   Merupakan proses menentukan variabel fuzzy dan himpunan fuzzynya, kemudian menentukan derajat keanggotaan antara data *input* dengan himpunan fuzzy yang telah didefinisikan untuk setiap variabel *input*.
2. Aplikasi Fungsi Implikasi

   Fungsi implikasi yang digunakan adalah *min*. melakukan implikasi fuzzy berdasar pada kuat penyulutan dan himpunan fuzzy terdefinisi untuk setiap variabel keluaran di dalam bagian konsekuensi dari setiap aturan [2]. Hasil implikasi fuzzy dari setiap aturan ini kemudian digabungkan untuk menghasilkan keluaran infrensi fuzzy.
3. Komposisi Aturan

   Merupakan inferensi yang diperoleh dari korelasi beberapa aturan.
4. Defuzzikasi

   Merupakan himpunan fuzzy dari komposisi aturan-aturan fuzzy, sedangkan *output* yang dihasilkan merupakan suatu bilangan pada domain himpunan *fuzzy* tersebut.

Pada proses defuzzikasi ada beberapa metode yang digunakan salah satunya menggunakan metode centroid. Dimana pada metode centroid, solusi *crisp output* diperoleh dengan cara mengambil titik pusat daerah *fuzzy* [13]. Metode centroid bertujuan mencari nilai pada *output* fuzzy yang berada diantara interval A dan interval B sehingga diketahui nilainya, perhitungan pada metode centroid menghitung jumlah tiap variabel dan perhitungan pada variabel z, variabel z sendiri merupakan hasil perhitungan dari derajat keanggotaan dikalikan dengan jumlah *rule* yang ada berikut adalah rumus untuk perhitungan menentukan variabel z pada rumus 2.1.

$$\mu - (\min[i] * n) \qquad (2.1)$$

Keterangan :

μ : nilai domain dari derajat keanggotaan

min [i] : nilai minimum dari *rule* ke-i

n : jumlah *rule*

Setelah diperoleh nilai dari perhitungan menentukan variabel z maka tahap selanjutnya mencari nilai rata-rata dari seluruh nilai yang diperoleh dari perhitungan penentuan variabel z. Berikut adalah rumus perhitungan dari metode centroid rumus persamaan 2.2.

$$Z = \frac{\sum_{j}^{n} = 1\, Z\, j\, \mu(Z j)}{\sum_{j}^{n} = 1\, \mu\, (Z j)} \qquad (2.2)$$

Keterangan :

Z : Nilai yang dicari untuk memperoleh nilai defuzzikasi.

n : Banyaknya nilai tiap variabel.

j : Banyaknya perhitungan pada variabel z.

Zj : Hasil yang diperoleh dari perhitungan variabel z.

μ : Nilai yang diperoleh dari fungsi keanggotaan.

Ada dua keuntungan menggunakan metode centroid, yaitu [2] :

a. Nilai defuzzifikasi akan bergerak secara halus sehingga perubahan dari suatu himpunan *fuzzy* juga akan berjalan dengan halus.
b. Lebih mudah dalam perhitungan.

2.2 Pengertian *Game*

Dalam bahasa Indonesia "*Game*" berarti "permainan". Permainan yang dimaksud dalam *game* juga merujuk pada pengertian sebagai "kelincahan intelektual" (*intellectual playability*). Sementara kata "*game*" bisa diartikan sebagai arena keputusan dan aksi pemainnya [10]. Ada target-target yang ingin dicapai pemainnya. Kelincahan intelektual, pada tingkat tertentu, merupakan ukuran sejauh mana *game* itu menarik untuk dimainkan secara maksimal. *Game* adalah bentuk seni dimana penggunanya disbut dengan pemain (*player*), diharuskan membuat keputusan-keputusan dengan tujuan mengelola sumber daya yang diperoleh dari kesempatan-kesempatan bermain miliknya untuk mencapai tujuan tertentu.

2.2.1 Klasifikasi *Game* Berdasarkan *Genre*

Game dapat dibagi menjadi beberapa macam sesuai dengan karakter atau alur cerita dari *game* tersebut berikut adalah macam-macam yang ada :

1. *Shooting* yaitu *game* yang memerlukan kecepatan refleks, koordinasi mata, tangan, juga *timing*, inti dari *game* jenis ini adalah menembak target yang dituju. Contohnya yaitu *game Call Of Duty* dan *Battlefield*.
2. *Fighting* yaitu *game* yang permainannya memerlukan refleks dan koordinasi mata dan tangan dengan cepat, tetapi inti dari *game* ini adalah penguasaan hafalan jurus. Contohnya yaitu *game Mortal Kombat* dan *Tekken*.

3. Petualangan yaitu *game* yang lebih menekankan pada jalan cerita dan kemampuan berfikir pemain dalam menganalisia tempat secara visual, memecahkan teka-teki maupun menyimpulkan berbagai peristiwa. Contohnya yaitu *game Rayman* dan *Tomb Rider*.
4. Simulasi, Konstruksi, Manajemen yaiu *game* yang seringkali menggambarkan dunia di dalamnya sedekat mungkin dengan dunia nyata dan memperhatikan dengan detail berbagai faktor. Contohnya yaitu *game The Sims*.
5. Strategi (RTS) yaitu *game* jenis ini memerlukan koordinasi dan strategi dalam memainkan permainan ini bisa dengan mengumpulkan *resource* yang banyak, *upgrade* level dari karakter sehingga menjadi kuat. Kebanyakan *game* strategi adalah *game* perang. Contohnya yaitu *game Warcraft* dan *Empire Earth*.
6. *Role Playing Game* (RPG) yaitu game yang biasanya dimainkan dengan satu karakter atau dengan berkelompok untuk melawan musuh dengan tujuan tertntu yang sudah diatur dari alur cerita yang dibuat. Contohnya yaitu *game Final Fantasy* dan *Suikoden*.
7. *Sport* yaitu game yang merupakan adaptasi dari kenyataan olahraga yang dimainkan seperti sepak bola, basket dan tenis serta membutuhkan kelincahan dan juga strategi dalam memainkannya. Contohnya yaitu *game Pro Evolution Soccer* dan NBA.
8. *Puzzle* yaitu *game* yang bisa disebut *game* teka-teki, pemain diharuskan memecahkan teka-teki dalam *game* tersebut. Contohnya yaitu *game Tetris*, dan *Bejeweled*.
9. *Edugames* yaitu *game* yang dibuat dengan tujuan spesifik sebagai alat pendidikan, bisa untuk belajar mengenal warna untuk balita, mengenal huruf dan angka, matematika, sampai belajar bahasa asing. Contohnya yaitu *game Dora The Explorer*.

2.3 *Unified Modelling Language* (UML)

Unified Modelling Language (UML) adalah sebuah bahasa yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak [12]. UML menawarkan sebuah standar untuk merancangan model sebuah sistem. Dengan menggunakan UML dapat membuat model untuk semua jenis aplikasi perangkat lunak, aplikasi tersebut dapat berjalan pada perangkat keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka UML lebih cocok untuk penulisan dalm bahasa berorientasi objek seperti C++, Java, C#, atau VB.NET. Walaupun demikian UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C .

UML mendefinisikan notasi dan *syntax*, notasi UML merupakan sekumpulan bentuk khusus untuk meggambarkan berbagai diagram piranti lunak [12]. Setiap bentuk memiliki makna tertentu dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya yaitu Grady Booch OOD (*Object-Oriented Design*), Jim Rumbaugh OMT (*Object Modelling Technique*), Ivan Jacobsen OOSE (*Object-Oriented Software Engineering*).

2.3.1 *Use Case Diagram*

Use Case Diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah apa yang diperbuat sistem. Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem [8]. *Use case* dapat sangat membantu apabila sedang menyusun *requirement* untuk sebuah sistem, mengkomunikasikan rancangan untuk semua fitur yang ada pada sistem.

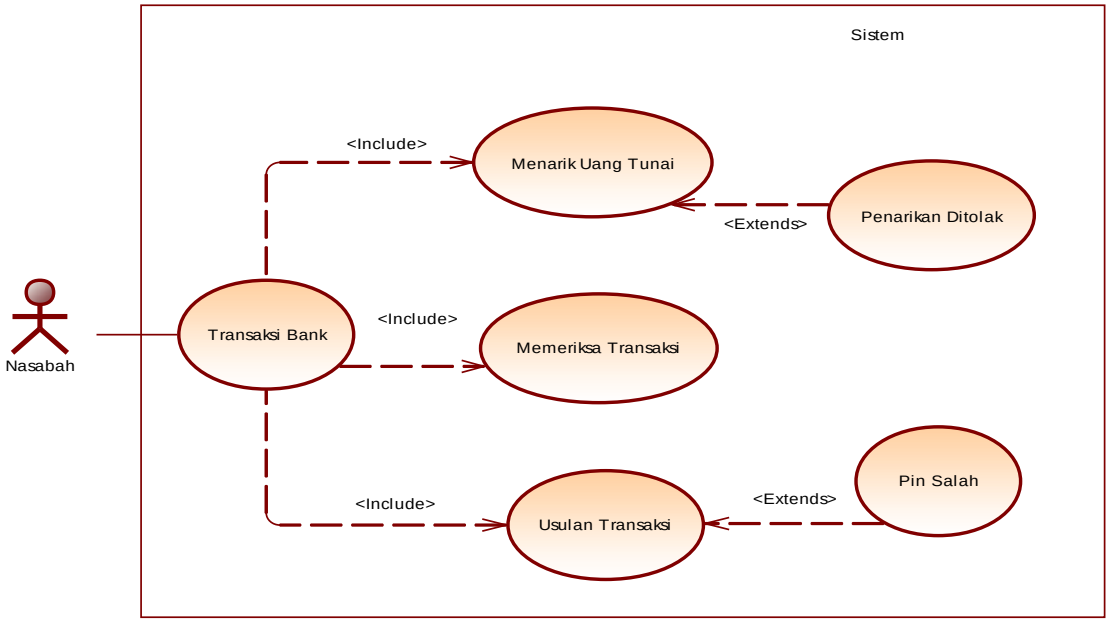
Sebuah *use case* juga dapat memasukan fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara

normal [8]. 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*.

Contoh pada *use case diagram* transaksi bank mempunyai enam fungsionalitas yang terdiri dari :

a. Transaksi Bank artinya nasabah berinteraksi dengan bank setelah proses penerimaannya oleh mesin ATM.
b. Menarik Uang Tunai artinya nasabah memasukkan kode PIN, jika Pin sah, rekening nasabah dapat diakses.
c. Memeriksa Transaksi artinya jika kode pin yang dimasukan pengguna tidak sah, maka mesin ATM akan menampilkan pesan yang sesuai.
d. Usulan Transaksi artinya nasabah berinteraksi dengan sistem bank setelah nasabah memasukan kode PIN yang sah, dengan meminta sejumlah uang tertentu. Jika jumlah uang/saldo pada rekeningnya memang cukup maka transaksi dilaksanakan.
e. Penarikan Ditolak artinya jika jumlah yang diminta oleh nasabah lebih besar dari jumlah yang ada pada rekeningnya (saldo) maka transaksi penarikan itu ditolak dan sistem/perangkat lunak memunculkan pesan tertentu.
f. Pin Salah artinya pada saat tertentu nasabah kadang mengetahui transaksi-transaksi yang telah nasabah lakukan pada periode waktu tertentu.

Berikut adalah contoh dari *use diagram* transaksi bank pada gambar 2.5.



Gambar 2.5 Contoh *Use Case Diagram* Transaksi Bank

2.3.2 *Class Diagram*

Class adalah spesifikasi yang jika diinisasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan atribut suatu sistem, sekaligus menawarkan layanan untuk manipulasi keadaan tersebut atau metode yang digunakan pada sistem tersebut [8]. *Class diagram* menggambarkan struktur dan deskripsi, *class*, *package*, dan objek berserta hubungan satu sama lain seperti asosiasi, *inheritance* dan lain-lain.

Class memiliki tiga area pokok yaitu Nama Kelas, Atribut dan Metode. Atribut dan metode dapat memiliki salah satu sifat berikut :

1. *Private* artinya data yang dikendalikan oleh suatu kelas tertentu dan tidak dapat diakses oleh objek-objek yang berbeda kelas. Satu ciri metode *sharing* yang bersifat *private* adalah hanya fungsi-fungsi atau metode – metode dalam kelas/objek yang bersangkutan yang dapat mengakses metode atau variabel serta

atribut dalam kelas tanpa batasan. Biasanya tanda (-) digunakan untuk atribut atau metode yang bersifat *private*.

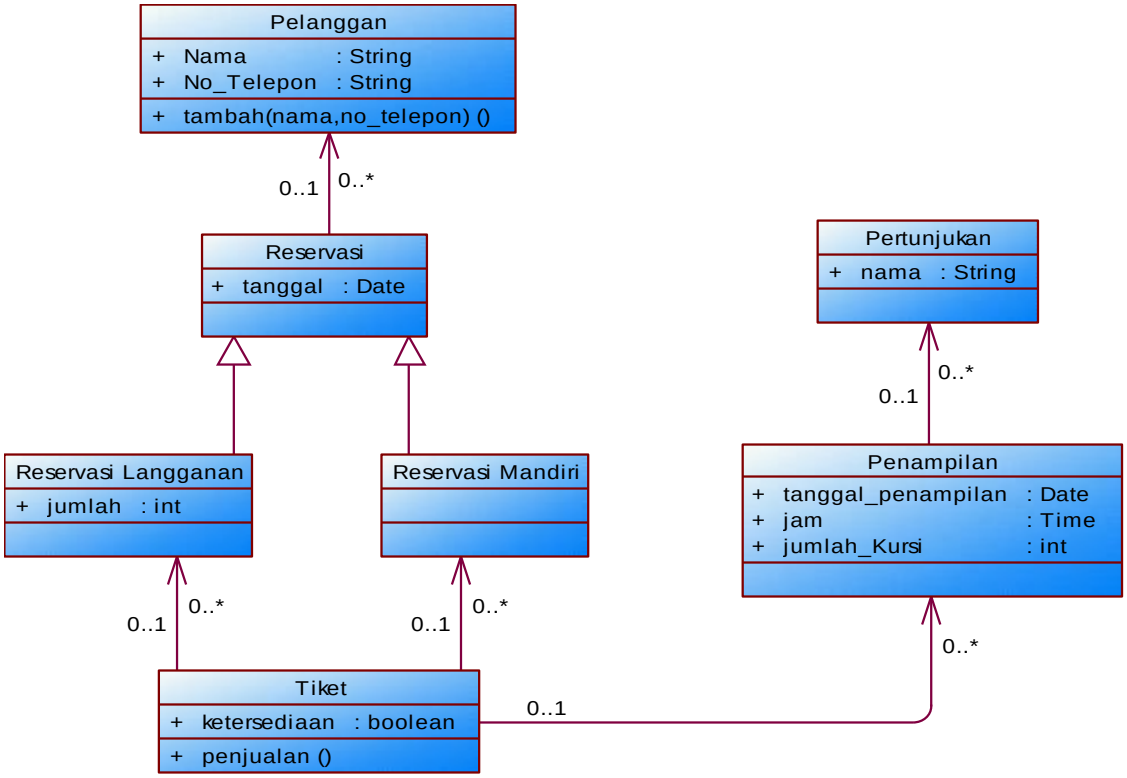
2. *Protected* artinya atribut atau metode yang bersifat *protected* hanya bisa diakses oleh kelas yang bersangkutan serta kelas-kelas yang merupakan turunannya mengikuti hierarki dari generalisasi. Biasanya tanda (#) digunakan untuk atribut atau metode yang bersifat *protected*.
3. *Public* artinya metode atau atribut yang bersifat *public* dapat diakses oleh semua objek dalam aplikasi tertentu tanpa batasan apa pun. Biasanya tanda (+) digunakan untuk atribut atau metode yang bersifat *public*.

Pewarisan (*Inheritance*) adalah hubungan antar *class* yang dapat diturunkan ke *class* yang lain dan mewarisi semua atribut dan metode *class* asalnya dan menambahkan fungsionalitas baru, sehingga anak dari *class* yang mewarisinya. Berikut adalah contoh *class diagram* pemesanan tiket pertunjukan mempunyai tujuh *class* antara lain :

a. *Class* Pelanggan, mempunyai atribut Nama dan No_Telepon dengan tipe data *String* dan mempunyai sifat *public* dan untuk metode tambah (nama, no_telepon) () dengan sifat *public*.
b. *Class* Reservasi, mempunyai atribut tanggal dengan tipe data *Date* dan mempunyai sifat *public*.
c. *Class* Reservasi Langganan, mempunyai atribut jumlah dengan tipe data *integer* dan mempunyai sifat *public*.
d. *Class* Reservasi Mandiri.
e. *Class* Tiket, mempunyai atribut ketersediaan dengan tipe data *boolean* dan mempunyai sifat *public*, dan untuk metodenya penjualan () dengan sifat *public*
f. *Class* Pertunjukan, mempunyai atribut nama dengan tipe data *String* dan mempunyai sifat *public*.

g. *Class* Penampilan, mempunyai atribut tanggal_penampilan dengan tipe data *Date*, atribut jam dengan tipe data *Time*, dan atribut jumlah_kursi dengan tipe data *integer* yang masing-masing atribut mempunyai sifat *public*.

Berikut adalah contoh class diagram pemesanan tiket pertunjukan pada gambar 2.6.



Gambar 2.6 Contoh *Class Diagram* Pemesanan Tiket Pertunjukan

2.3.3 *Activity Diagram*

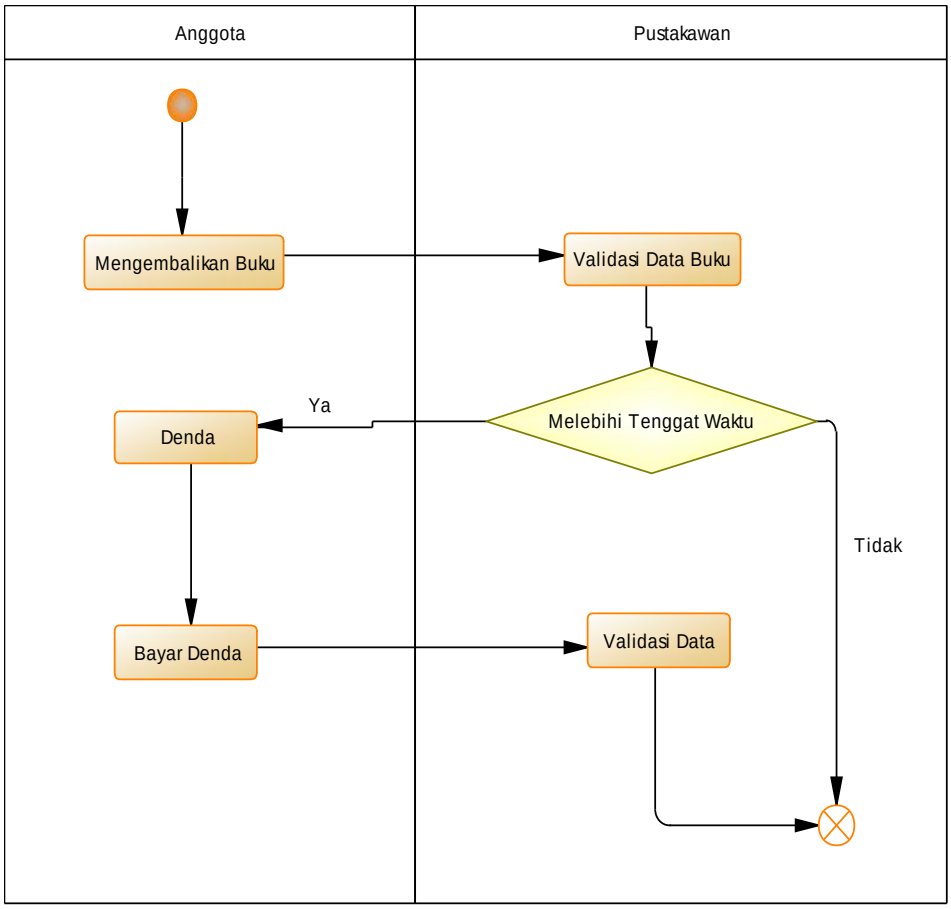
Activity diagram menggambarkan berbagai alir aktifitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, keputusan yang mungkin terjadi dan bagaimana berakhirnya. *Activity diagram* juga dapat menggambarkan proses pararel yang mungkin terjadi pada beberapa eksekusi [8]. *Activity diagram* merupakan state diagram khusus, di mana sebagian besar *state* adalah *action* dan

sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan perilaku internal sebuah sistem secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktifitas dari level atas secara umum.

Sebuah aktifitas dapat direalisasikan oleh suatu *use case* atau lebih. Aktifitas menggambarkan proses yang berjalan sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktifitas. Contoh pada *activity diagram* pengembalian buku perpustakaan, pada *activity diagram* pengembalian buku perpustakaan mempunyai alir aktifitas sebagai berikut :

a. Anggota perpustakaan mengembalikan buku perpustakaan kepada pustakawan.
b. Pustakawan memvalidasi data buku yang dipinjam oleh anggota perpustakaan.
c. Apabila anggota perpustakaan mengembalikan buku melebihi batas waktu maka anggota perpustakaan akan mendapat denda dengan membayar sejumlah uang.
d. Kemudian pustakawan memvalidasi data anggota perpustakaan.
e. Apabila anggota perpustakaan mengembalikan buku tidak melebihi batas waktu pengembalian maka anggota tidak mendapat denda.

Berikut adalah contoh *activity diagram* pengembalian buku perpustakaan pada gambar 2.7.



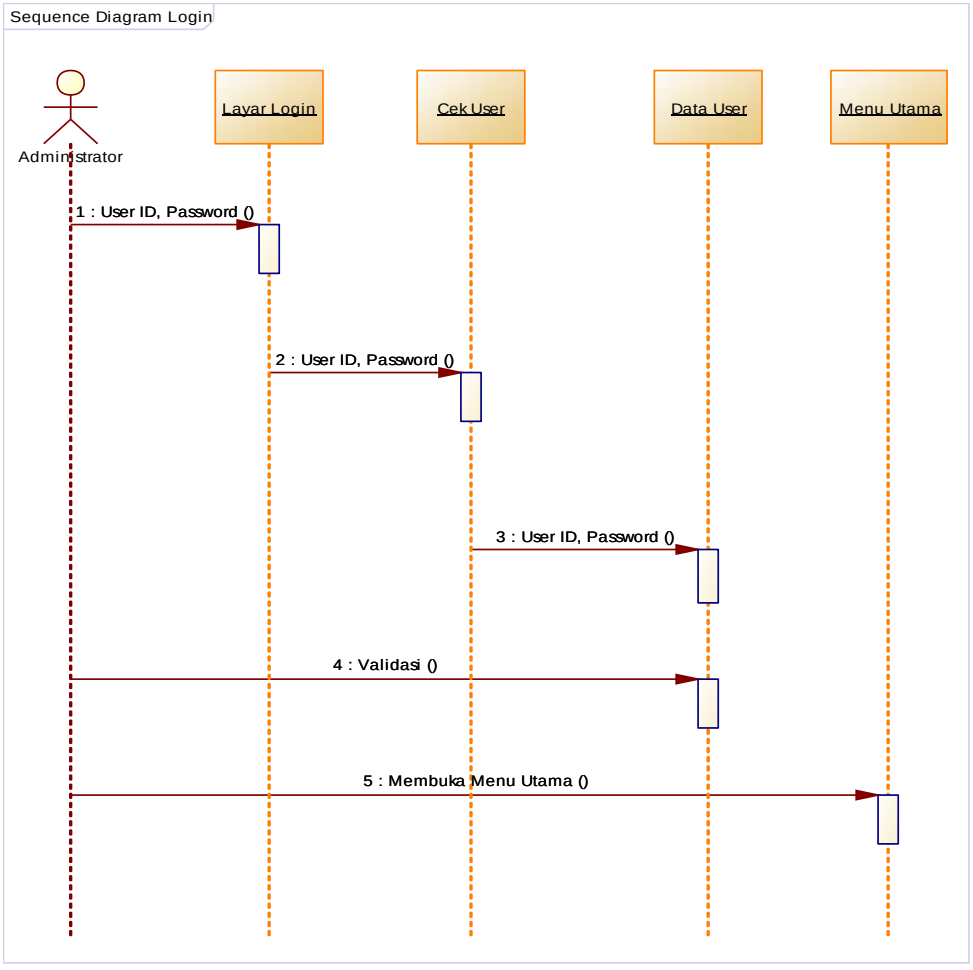
Gambar 2.7 Contoh *Activity Diagram* Pengembalian Buku Perpustakaan

2.3.4 *Sequence Diagram*

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atas dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait) [8]. *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* aktifitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan. Contoh pada *sequence diagram login* mempunyai interaksi sebagai berikut :

a. Administrator (Aktor) melakukan *login*, maka *method User ID*, *Password* () memanggil kelas Layar *Login*.
b. Kelas Layar *Login* memanggil kelas Cek *User* untuk mengecek *User ID* dan *Password*.
c. Kelas Cek *User* memanggil kelas Data *User* untuk memvalidasi *User ID* dan *Password* di *database*.
d. Apabila *User ID* dan *Password* valid maka Administrator beerhasil melakukan *Login*.

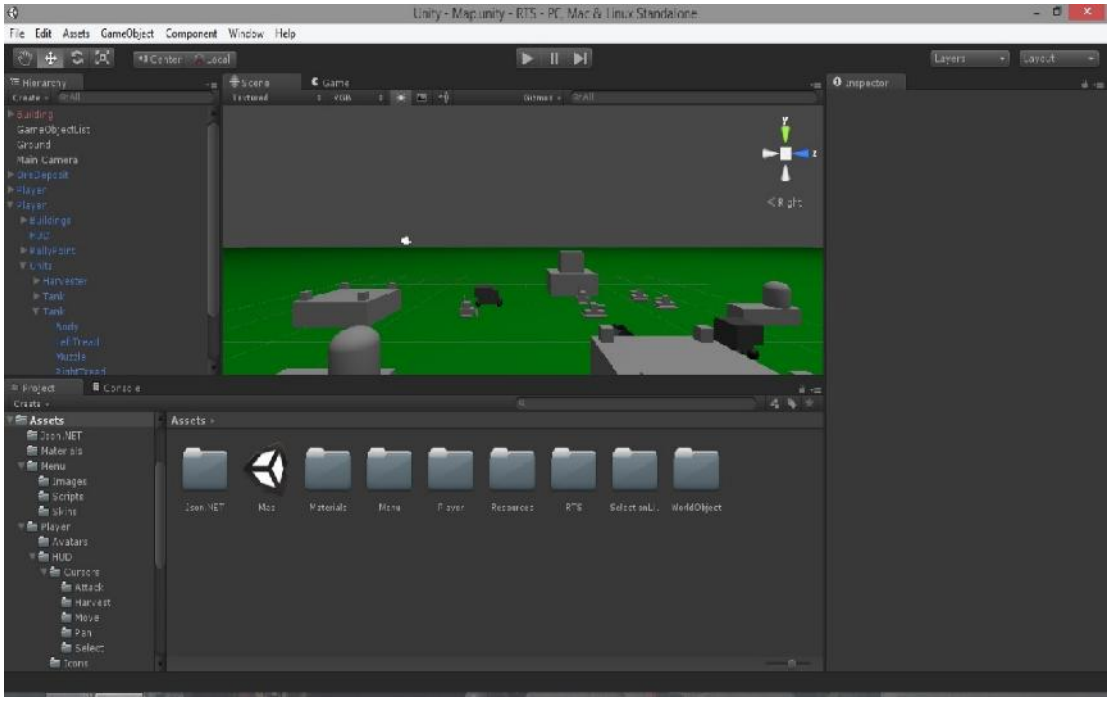
Berikut adalah contoh dari *sequence diagram login* pada gambar 2.8.



Gambar 2.8 Contoh *Sequence Diagram Login*

2.4 *Unity*

Unity adalah sebuah *game developing software*. Dengan menggunakan *software* ini karena dapat membuat *game* 3D. Karena *software* ini sangat mudah menggunakannya, dengan GUI yang memudahkan kita untuk membuat, memanpulasi dan membuat *script* untuk menciptakan sebuah *game* berkonsep 3D. Selain itu bisa untuk *build game PC*, Unity juga dapat digunakan untuk membangun *game console* seperti Nintendo Wii, PS3, Xbox 360, juga Ipad. Iphone dan Android. Namun masing-masih membutuhkan biaya lisensinya sendiri. Selain itu juga bisa juga membuat web, hanya membutuhkan install unity web player. Berikut adalah tampilan dari *Unity* pada gambar 2.9.



Gambar 2.9 Tampilan *Unity*

2.5 Pemrograman C

Bahasa pemrograman C diciptakan oleh Dennis Ritchie pada tahun 1972 di Bell Laboratories. Bahasa pemrograman C memiliki kelebihan yaitu :

1. Bahasa C tersedia hampir di semua jenis komputer.
2. Kode bahasa C sifatnya adalah *portable* dan flesksibel untuk semua jenis komputer.
3. Bahasa C hanya menyediakan sedikit kata-kata kunci, hanya terdapat 32 kata kunci.
4. Proses *executable* program bahasa C lebih cepat.
5. Bahasa C mempunyai dukungan pustaka yang banyak.
6. Bahasa C merupakan bahasa yang terstruktur.
7. Bahasa C termasuk bahasa tingkat menengah.

Penempatan ini hanya menegaskan bahwa C bukan bahasa pemrogaman yang berorintasi pada mesin yang merupakan ciri bahasa tingkat rendah, melainkan berorientasi objek tetapi dapat diintrepertasikan oleh mesin dengan cepat secepat bahasa mesin. Inilah salah satu kelebihan C yaitu memiliki kemudahan dalam menyusun programnya semudah bahasa tingkat tinggi namun dalam mengeksekusi program secepat bahasa tingkat rendah.

Selain kelebihan bahasa pemrograman C juga memiliki kekurangan yaitu :

1. Banyaknya operator serta fleksibilitas penulisan program kadang-kadang membingungkan pengguna.
2. Bagi pemula pada umumnya akan kesulitan dalam menggunakan pointer.

Dalam mengkompilasi program baru dapat dijalankan setelah melalui tahap kompilasi dan penggabungan. Tahap kompilasi dimaksudkan untuk memeriksa *source* program sesuai dengan kaidah-kaidah yang berlaku di dalam bahasa pemrograman C. Tahap kompilasi akan menghasilkan *relocatable object file*. *File-file* objek tersebut kemudian digabung dengan perpustakaan fungsi yang sesuai untuk menghasilkan suatu *executable program*.