BAB 2
TINJAUAN PUSTAKA

2.1 Pengertian *Video Game*

Video game merupakan salah satu sarana hiburan yang dapat dilakukan manusia secara aktif. Menonton *film*, mendengar lagu merupakan kegiatan hiburan yang pasif. Sedangkan bermain *video game* merupakan kegiatan dimana seorang pengguna memainkan sesuatu yang terjadi di dalam *game* tersebut. Terdapat dunia di dalam *game* tersebut yang memiliki aturan yang harus dipahami pengguna. Aturan tersebut adalah sesuatu yang harus dikerjakan agar tujuan dari *game* tersebut tercapai.

2.1.1 Jenis-jenis *Video Game* Berdasarkan *Genre*

Video *game* terbagi ke dalam beberapa jenis permainan *game* pertama yang diciptakan adalah *pong game* sederhana dimana pemainnya menahan bola agar tidak masuk ke area mereka. Ada beberapa jenis *game* yang popular saat ini dianataranya [4].

a. *Game Role Playing Game* (RPG)

Game RPG adalah *game* dimana pemain berperan sebagai seorang tokoh dimana tokoh tersebut memiliki peran yang sangat penting di dalam cerita *game* tersebut. Tokoh tersebut biasanya dimulai dengan sangat lemah dan akan berakhir sangat kuat di akhir cerita. *Game* RPG adalah satu jenis *game* yang aspek ceritanya sangat haruslah kuat. Karena pemain harus merasa seolah karakter yang ada di dalam *game* tersebut adalah dirinya sendiri. Sehingga pemain bisa merasakan apa yang dirasakan oleh karakter tersebut.

Game RPG lahir di Jepang. *Game* ini memiliki *game*play kebanyakan adalah turn-based battle. Dimana karakter dalam *game* akan bergantian menyerang musuh yang ada. Karakter dalam *game* akan mendapatkan kekuatan yang baru ketika mengalahkan lawan setiap pertarungan. Setiap gilirannya pemain bisa memilih untuk menyerang musuh yang mana saja. Sistem pertarungan seperti ini tidak laris di pasar barat, sehingga munculah jenis *game* yang baru yaitu *action-rpg*.

b. *Game* Action

Game action atau aksi merupakan jenis *game* dimana pemain mengendalikan satu karakter dan berusaha melawan semua yang menghalangi karakter tersebut dengan cara bertarung. *Game* ini berfokus pada *game*play bukan cerita. Pemain yang bermain *game* ini ingin melepas penat dengan melawan semua musuh yang ada dengan aksi yang memukau. *Game* berjenis ini biasanya mengandalkan kekerasa dalam setiap permainannya, sehingga cenderung buruk bagi anak yang masih berada di bawah umur.

Game aksi biasanya terdiri dari beberapa *stage* yang akan diakhiri dengan melawan *boss* dalam setiap *stage*. Dalam *game* aksi kadang terdapat unsur-unsur RPG seperti karkater dalam *game* bisa bertambah kuat jiga mengalahkan beberapa lawan tertentu, mendapatkan senjata baru, mendapatkan *power-up* yang baru.

c. *Game First Person Shooter* (FPS)

Game first person shooter adalah salah satu jenis permainan dengan kamera sudut pandang orang pertama dan menggunakan senjata api untuk menembak musuh yang ada. Permainan ini didesain untuk membutuhkan keakurasian dan refleks tingkat tinggi dari pemain. Permainan ini sama seperti tipe permainan aksi, mementingkan *game*play dibanding cerita. Permainan ini biasanya lebih baik dimainkan dengan orang lain. Tipe permainan ini adalah salah satu permainan yang memulai adanya *e-sport* ajang kompetisi bermain *game* untuk menentukan siapa yang terbaik.

d. *Game Real Time Strategy*(RTS)

Game real time strategy adalah *game* dimana kita bertugas untuk menentukan bagaimana suatu unit dibangun, kombinasi dari unit itu sendiri, formasi menyerang dari unit tersebut dan bagaimana unit tersebut harus menyerang musuh. Permainan ini membutuhkan waktu yang lebih lama dibanding tipe *game* lain untuk diselesaikan. Selain menyusun strategi kita juga harus mengurus *resources* yang dibutuhkan untuk membangun *unit* tersebut.

e. *Game Multiplayer Online Battle Arena* (MOBA)

MOBA merupakan modifikasi dari *game* RTS. Jika di dalam *game* RTS pemain harus membangun banyak unit dan mengontrolnya, dalam *game* MOBA kita hanya mengontrol satu karakter dan unit pasukan dikendalikan oleh kecerdasan buatan. MOBA juga memiliki unsur RPG dimana karakter yang pemain mainkan bisa bertambah kuat dengan *level up* dan memiliki lebih banyak kemampuan setiap *levelnya*.

2.2 Kecerdasan Buatan

Kecerdasan buatan merupakan salah satu bagian ilmu komputer yang membuat agar mesin komputer agar data melakukan kegiatan atau pekerjaan seperti dan sebaik yang dilakukan manusia. Awal diciptakannya, komputer merupakan hanyalah sebuah alat hitung saja. Semakin berkembangnya manusia, komputer saat ini dipakai dibanyak bidang. Komputer bahkan dituntut semakin bisa untuk melakukan pekerjaan manusia [5].

Manusia mampu menyelesaikan pekerjaannya karena memiliki kecerdasan dalam otak mereka. Kecerdasan ini didapat manusia melalui proses yang dinamakan belajar. Manusia semakin mampu dalam menyelesaikan pekerjaanya karena melalui proses belajar yang tidak sebentar, bahkan dengan percobaan berkali-kali, sehingga manusia memiliki kecerdasan dalam menyelesaikan pekerjaan dan masalahnya.

Agar komputer dapat melakukan pekerjaan sebaik ataupun lebih baik dari manusia, diperlukan suatu kecerdasan yang harus ditanam di dalam komputer. Kecerdasan buatan akan memberikan metode-metode bagi komputer agar komputer dapat memiliki nalar seperti manusia dan membuat komputer menjadi semakin pintar.

2.2.1 Lingkup Kecerdasan Buatan

Lingkup utama kecerdasan buatan antara lain [5]:

a. Sistem Pakar

Komputer dapat digunakan sebagai saran dalam menyimpan pengetahuan para ahli. Dengan demikian diharapkan komputer dapat memiliki keahlian

seperti para ahli yang pengetahuannya disimpan di dalam komputer tersebut.

b. Pengolahan Bahasa Alami

Pengolahan bahasa alami ini komputer diharapkan dapat berkomunikasi dengan pengguna melalui bahasa sehari-hari.

c. Pengenalan Ucapan

Pengolahan suara ini diharpakan komputer dapat mengenali suara penggunanya dan menjalankan perintah yang diberikan melalui perintah suara.

d. Robotika dan Sistem sensor

Kecerdasan yang dapat dibuat untuk suatu komputer dalam bentuk robot agar robot bisa bertindak selayaknya manusia. Kemudian dalam sistem sensor agar sensor dapat membaca setiap masukan yang ada dan menyampaikannya pada pengguna.

e. *Computer vision*

Kecerdasan buatan yang dapat digunakan untuk mengenali citra dan memanipulasi citra tersebut melalui objek-objek di dalam komputer.

f. *Intelligent Computer aid Instruction*

Kecerdasan buatan yang dapat digunakan agar komputer bisa digunakan sebagai guru dan pelatih dalam pelatihan dan pengajaran.

g. *Game Playing*

Kecerdasan buatan yang digunakan pada *video game* untuk menentukan suatu perilaku NPC *(non playable character)* ketika melawan pemain.

2.2.2 Kentungan Kecerdasan Buatan

Keuntungan dari kecerdasan buatan dalam penggunaannya diantara lain :

a. Kecerdasan buatan bersifat permanen, komputer tidak akan melupakan kecerdasan buatan yang sudah ditanam dalam dirinya. Suatu kecerdasan buatan akan hilang jika pengguna menghendakinya.

b. Kecerdasan buatan jauh lebih murah dibandingkan kecerdasan alami. Karena tidak perlu lagi memikirkan biaya untuk melatih lagi kecerdasan tersebut.

c. Kecerdasan buata mudah untuk dibagi ke komputer yang lain. Kecerdasan buatan dapat dipindah hanya dengan menyalin perintah dasar yang sudah diterapkan ke komputer yang lain.

d. Kecerdasan buatan bersifat konsisten. Sebagai bagian dari metode komputer kecerdasan buatan akan melakukan hasil dengan cara yang konsisten dibanding kecerdasan alami.

e. Kecerdasan buatan bisa mengerjakan pekerjaan dengan cepat dibandingkan kecerdasan alami.

f. Kecerdasan buatan bisa mengerjakan pekerjaan dengan baik dibandingkan kecerdasan alami.

g. Kecerdasan buatan dapat dengan mudah didokumentasikan. Kecerdasan buatan dapat didokumentasikan setiap tahapnya dengan melacak setiap kegiatan dari aktivitas sistem tersebut. Kecerdasan alami sangat sulit untuk dibuat ulang.

2.3 Algoritma Particle Swarm Optimization (PSO)

Algoritma PSO merupakan suatu algoritma yang dikembangkan oleh Kennedy dan R. Eberhart di tahun 1995 yang mendapatkan inspirasi dari perilaku pergerakan kelompok hewan seperti ikan, hewan herbivora, burung dan serangga seperti semut dan lebah. Setiap objek dari hewan tersebut akan disederhanakan menjadi partikel-partikel yang mudah digerakan. Algoritma ini menerapkan sifat masing-masing individu dalam satu kelompok besar dan lalu menggabungkan sifat-sifat tersebut agar dapat menyelasaikan masalah yang ada [2]. Algoritma PSO merupakan metode kecerdasan buatan *swarm intelligence* yang berasaskan perilaku kolektif dalam suat kelompok dan setiap individu dalam kelompok tersebut dapat mengatur dirinya sendiri. Dalam *swarm intelligence* algoritma PSO merupakan metode yang dapat digunakan untuk mengatur posisi partikel terbaik. Konsep pengotimalan partikel individu ini diperkenalkan oleh Kennedy dan R. Ebhert

setiap pertikel memiliki nilai kebugaran yang dievaluasi oleh fungsi dioptimalkan, selain itu parameter setiap partikel memiliki beberapa hal berikut ini konstanta kecapatan, koordinat posisi, koordinat posisi terbaik, koordinat posisi terbaik secara global. Dari parameter tersebut yang digunakan di atas dapat membentuk perubahan posisi yang dilakukan secara bersamaan di tiap-tiap partikel. Sebelum mengubah posisi, partikel tersebut terlebih dahulu mengubah kecepatannya. Perubahan kecepatan ini menggunakan persamaan dasar PSO [6]

Update partikel dapat digunakan dengan persamaan dasar PSO : [6]

$$V_i(t+1) = V_i(t) + C_1 r_1 (Pbest_i - X_i(t)) + C_2 r_2 (Gbest_i - X_i(t)) \quad (2.1)$$

Update posisi partikel dapat dilakukan dengan menggunakan persamaan dasar PSO : [6]

$$X_i(t+1) = X_i(t) + V_i(t+1) \quad (2.2)$$

Mencari nilai *fitness*

$$F(X) = \sqrt{((player.x - NPC.x)^2 + (player.y - NPC.y)^2)} \quad (2.3)$$

Di mana :

Vi (t) : kecepatan partikel ke (i) pada iterasi ke (t)

Xi (t) : Posisi partikel ke (i) pada iterasi ke (t)

V : *Velocity* (kecepatan partikel)

X : Koordinat partikel (posisi partikel)

i : Partikel ke (1,2,3..N)

C1 dan C2 : Faktor *cognitive* dan sosial nilai ini adalah nilai konstanta bernilai diantara (0 dan 1)

r1 dan r2 : Bilangan acak antara 0 - 1

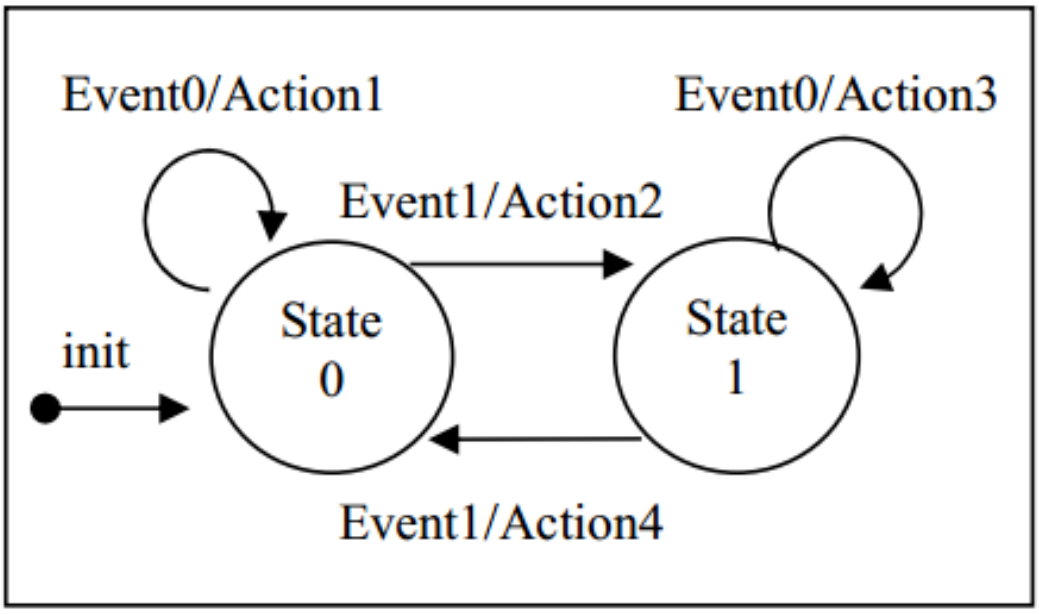
Pbest : Posisi terbaik yang dimiliki setiap partikel

Gbest : posisi partikel yang memiliki posisi paling baik dari partikel lainnya.

2.4 Algoritma *Finite State Machine*

Finite State Machine adalah sebuah metodelogi perancangan sistem control yang menggambarkan tingkah laku dan prinsip kerja sistem dengan menggunakan tiga hal berikut : *State* (keadaan), *event* (kejadian) dan *action* (aksi).

Dalam periode waktu tertentu sistem berada pada satu sistem *State* yang sedang dijalankan. Kemudian sistem akan beralih atau berubah menuju *State* lain jika mendapat *input* dalam kejadian tertentu, baik yang berasal dari luar ataupun komponen sistem tersebut sendiri. Perubahan keadaan ini juga biasanya disertai oleh aksi yang akan dilakukan oleh sistem ketika merespon *input* yang terjadi. Aksi yang dilakukan berikut dapat berupa aksi yang sederhana maupun rangkaian aksi proses yang bersifat kompleks [7].



Gambar 2.1 Diagram *Finite State Machine* [7]

Gambar 2.2 memperlihatkan FSM dengan dua buah *State* dan dua buah *input* serta empah buah *outpu* yang berbeda terlihat di dalam gambar, ketika mulai dinyalakan, sistem akan bertransisi menuju *State()*, pada keadaan ini sistem akan memunculkan hasil *action1* jika terjadi *input*an *event*(), sedangkan jika terjadi *event1* maka *action2* akan dieksekusi kemudian sistem akan transisi menuju *State1* dan seterusnya.

Implementasi FSM dapat diterapkan pada pemograman berorientasi objek atau yang sering disebut OOP. Kelebihan OOP dibanding yang lain dalam implementasi FSM adalah kefleksibilitasannya yang tinggi dan pemeliharaannya yang sudah mudah baik pada sistem dalam level manapun. Selain itu juga FSM

diuntungkan dengan kelebihan OOP dalam penggunaan kembali kode yang sudah diketik. [7]

2.5 Pemograman 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. Metologi berorientasi objek merupakan suatu cara bagaimana sistem perangkat lunak dibangun melalui pendekatan objek secara sistematis. Metode berorientasi objek didasarkan pada penerapan prinsip-prinsip pengelolaan kompleksitas. Metode berorientasi objek meliputi rangkaian aktifitas analisis berorientasi objek, perancangan berorientasi objek, pemrograman berorientasi objek dan pengujian berorientasi objek. [8]

Dalam rekayasa perangkat lunak, konsep pendekatan berorientasi objek dapat diterapkan pada tahap analisis perancangan,pemrograman dan pengujian perangkat lunak. Ada berbagai teknik yang dapat digunakan pada masing-masing tahap tersebut, dengan aturan dan alat bantu pemodelan tersebut. [8]

2.5.1 Konsep Dasar Pemograman Berorientasi Objek

Berikut ini adalah beberapa konsep dasar yang harus dipahami tentang metodelogi berorientasi objek [8] :

a. Objek

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

b. Kelas

Kelas adalah kumpulan objek-objek dengan karakteristik yang sama. Kelas merupakan definisi statis dan himpunan objek yang sama yang mungkin lahir atau diciptakan dan kelas tersebut.Sebuah kelas akan

mempunyai sifat (atribut), kelakuan (operasi/metode), hubungan (relationship) dan arti. Suatu kelas dapat diturunkan dan kelas yang lain, di mana atribut dan kelas semula dapat diwariskan ke kelas yang baru.Secara teknis, kelas adalah sebuah struktur tertentu dalam pembuatan perangkat lunak. Kelas merupakan bentuk struktur padakode program yang menggunakan metodologi berorientasi objek.

c. Metode

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

d. Enkapsulasi

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



Gambar 2.2 Proses Enkapsulasi

e. Inheritance

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

f. Polimorfisme

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

g. Antarmuka

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

h. Reuseability

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

i. Generalisasi Spesialisasi

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

j. Komunikasi Antar Objek

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

k. Paket

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

l. Atribut

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

m. Abstraksi

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

2.6 Unified Modeling Language (UML)

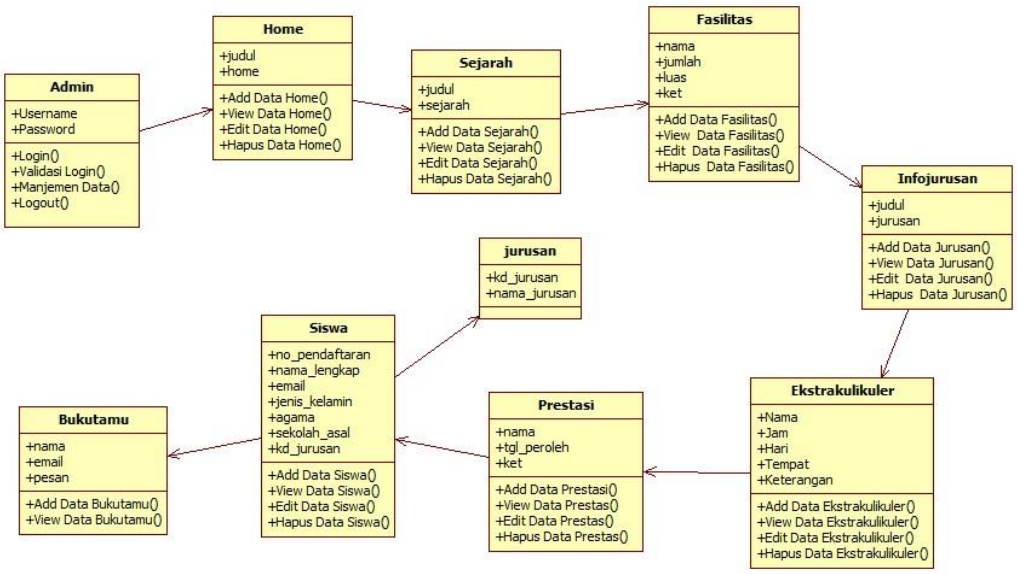
Unified Modelling Language (disingkat UML) adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan kebutuhan (requirement), membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek.

UML hanya dapat digunakan sebagai pemodelan. Penggunaanya tidak terbatas pada pemodelan tertentu saja. Tetapi pada kenyataannya UML sering digunakan pada pemograman berorientasi objek.

UML muncul 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 dengna menggunakan diagram dan teks-teks pendukung. UML hanya berfungsi untuk melakukan pemodelan. Penggunaan UML tidak terbatas pada metodologi tertentu, meskipun pada kenyataannya UML lebih banyak digunakan pada metodologi berorientasi objek.

2.6.1 Class Diagram

Class Diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibat untuk membangun sistem. Class diagram dibuat agar pembuat program membuat kelas-kelas sesuai rancangan didalam diagram kelas agar antara dokumentasi perancangan dan perangkat lunak sinkron. Contoh kelas diagram dapat dilihat pada gambar di bawah ini. [8]



Gambar 2.3 Contoh Class Diagram

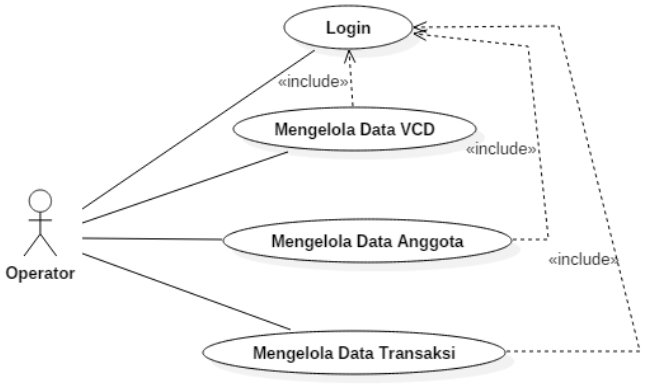
2.6.2 Use Case Diagram

Use case diagram merupakan pemodelan untuk kelakuan sistem informasi yang akan dibuat. Usecase mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, use case digunakan untuk mengetahui fungsi apa saja yang ada didalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Syarat penamaan pada use case adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada use case yaitu pendefinisian apa yang disebut aktor dan use case.

Aktor merupakan orang, proses, atau sisitem lain yang berinteraksi dengan sistem informasi yang akan dibuat. Namun walaupun aktor disimbolkan dengan gambar orang belum tentu aktor merupakan orang.

Use case merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor. [8]

Bentuk use case dapat dilihat pada gambar di bawah ini :

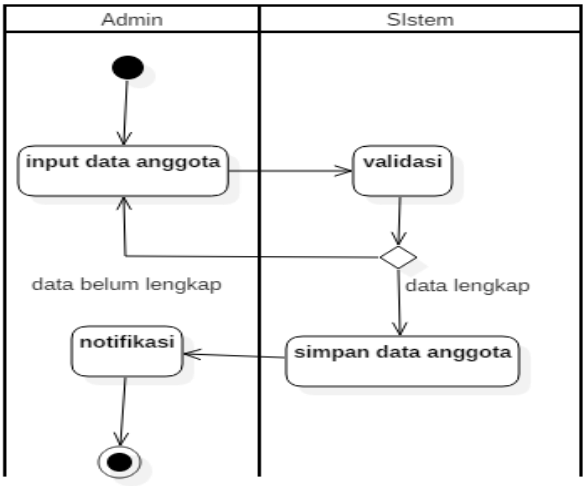


Gambar 2.4 Contoh Use Case Diagram

2.6.3 Activity Diagram

Diagram aktifitas atau *activity diagram* menggambarkan aliran kerja atau aktifitas dari sebuah sistem atau proses bisnis atau menu yan gada pada perangkat lunak. Perlu diperhatikan bahwa diagram aktifitas menggmabarkan aktifitas sistem bukan apa yang dilakukan aktor. [8]

Contoh aktifitas diagram dapat dilihat pada gambar di bawah ini

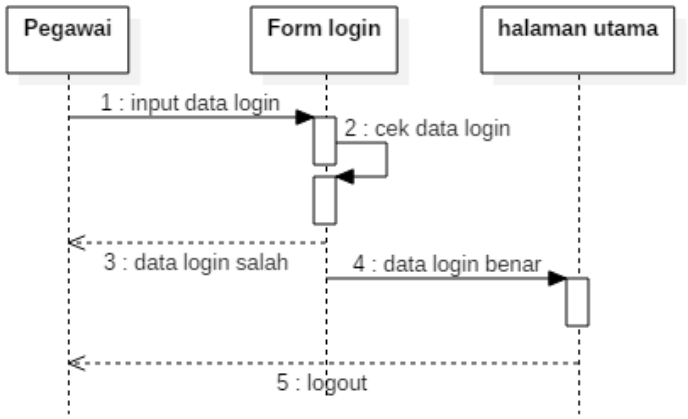


Gambar 2.5 Contoh Activity Diagram

2.6.4 Sequence Diagram

Diagram sekuen menggambarkan kelakuan objek pada use case dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Banyaknya diagram sekuen yang harus digambar adalah minimal sebanyak pendefinisian use case yang memiliki proses sendiri atau yang penting semua use case yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak use case yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak. [8]

Contoh *sequence* diagram dapat dilihat pada gambar di bawah ini



Gambar 2.6 Contoh Sequence Diagram

2.6.5 Sejarah UML

Bahasa pemrograman berorientasi objek yang pertama dikembangkan dikenal dengan nama Simula-67 yang dikembangkan pada tahun 1967. Bahasa pemrograman ini kurang berkembang dan dikembangkan lebih lanjut, namun dengan kemunculannya telah memberikan sumbangan yang besar pada developer pengembang bahasa pemrograman berorientasi objek selanjutnya.

Perkembangan aktif dari pemrograman berorientasi objek mulai menaik ketika ketika berkembangnya bahasa pemrograman Smaltalk pada awal 1980-an yang kemudian diikuti dengan perkembangan bahasa pemrograman berorientasi objek lainnya seprti C objek, C++, Eiffel dan CLOS.

Secara aktual, penggunaan pemrograman berorientasi objek pada saat itu masih terbatas, namun telah banyak mernarik perhatian disaat itu. Sekitar 5 tahun setelah Smaltalk berkembang, maka berkembang pula metode pengembangan berorientasi objek. Metode yang pertama diperkenalkan oleh Sally Shlaer dan Stephen Mellor (Shlaer-Mellor, 1988) dan Peter Coad dan Edward Yourdon (Coad-Yourdon, 1991), diikuti oleh Grady Booch (Booch, 1991), James R. Rumbaugh, Michael R. Blaha, William Lorensen, Frederick Eddy, Willian Premerlani (Rumbaugh-Blaha-Premerlani-Eddy-Lorensen, 1991) dan banyak lainnya.

Buku yang terkenal yang juga berkembang selanjutnya adalah karangan Ivar Jacobson (Jacobson, 1992) yang menerangkan perbedaan pendekatan yang

fokus pada *use case* dan proses pengembangan. Sekitar 5 tahun kemudian muncul buku yang membahasa mengenai metodologi berorientasi objek yang diikuti dengna buku-buku yang lainnya,. Didalamnya membahas mengenai konsep, definisi, notasi, terminologi dan proses mengenai metodologi berorientasi objek.

Banyaknya metodologi-metodologi yang berkembang pesat saat itu, maka munsullah ide untuk membuat bahasa yang dapat dimengerti semua orang. Usaha penyatuan ini banyak mengambil dari metodologi-metodologi yang berkembang saat itu. Maka dibuat bahasa yang merupakan gabungan dari beberapa konsep seperti konsep *Object Modelling Technique* (OMT) dari Raumbaugh dan Booch (1991), konsep *The Classes, Responsibilities,* pemikiran Ivar Jacobson, dan beberpa konsep lainnya dimana dalam sebuah perusahaan yang bernama *Rational Software Corporation* menghasilkan bahasa yang disebut dengan *Unified Modeling Object* (UML).

Pada tahun 1996, *Object Management Group* (OMG) mengajukan proposal agar adanya standarisasi pemodelan berorientasi objek dan pada bulan September 1997 UML diakomodasi oleh OMG sehingga sampai saat ini UML telah memberikan kontribusinya yang cukup besar didalam metodologi berorientasi objek dan hal-hal yang terkait didalamnya.

2.7 Unity

Unity 3D adalah sebuah *game engine* yang berbasis *cross-platform.* Unity dapat digunakan untuk membuat sebuah *game* yang bisa digunakan pada perangkat komputer, ponsel pintar, PS3 dan XBOX.

Unity adalah sebuah alat yang terintegrasi untuk membuat *game*, arsitektur bangunan dan simulasi. Unity bisa untuk *games* komputer baik *offline* maupun *online*. Unity tidak dirancang untuk proses modeling. Fitur scripting yang disediakan mendukung bahasa pemrograman JavaScript, C# dan Boo. Unity menggunakan konsep Parenting. Konsep ini digunakan untuk membuat sebuah *game* objek menjadi anak dari *game* objek lainnya. *Game* objek yang terdapat dalam sebuah gam objek yang lain akan mengikuti perpindahan ketika *game* objek parent mengalami perubahan posisi. [9]