BAB 2
LANDASAN TEORI

2.1 Game

Video game adalah adalah permainan yang menggunakan interaksi dengan antarmuka pengguna melalui gambar yang dihasilkan oleh piranti *video*. Permainan *video* umumnya menyediakan sistem penghargaan misalnya skor yang dihitung berdasarkan tingkat keberhasilan yang dicapai dalam menyelesaikan tugas-tugas yang ada di dalam permainan[4].

2.1.1 Sejarah Game

Game atau permainan dapat diklasifikasikan menjadi dua bagian besar game fisik dan game elektronik. Game fisik yaitu permainan yang sering kita lakukan sdalam kehidupan sehari hari sewaktu masa anak-anak. Seperti petak umpet, lompat tali, dan sebagainya. Permainan disini yaitu berhubungan dengan gerak fisik. Sedangkan game elektronik yaitu dimana sebuah permainan dimainkan melalui video atau konsol.

Electronic game atau selanjutnya disebut sebagai video game pertama kali ditemukan oleh Thomas T.Goldsmith Jr. dan Estle Ray Mann. Penemuannya dipatenkan pada januari 1947. Yang mendasari perkembangannya saat ini adalah ketika mereka menemukan Cathode-Ray Tube sebuah tabung vacuum yang digunakan sebagai media untuk membuat simulasi kecepatan tembakan dan arah tembakan sebuah roket. [5].

2.1.2 Jenis-jenis Game

Jenis *game* atau *genre game* adalah klasifikasi *game* yang didasari interaksi pemainnya. Visualisasi juga menjadi ukuran klasifikasi *genre* ini. Yaitu setiap pengembang *game* membuat kompilasi antar berbagai jenis *game*. Sehingga variasi format *game* lebih banyak[6]. Ada banyak jenis *game* yang beredar saat ini, diantaranya adalah:

a. *Side Scrolling Game*

Pada *game* jenis ini, kita dapat menggerakan karakter ke kanan, kiri, atas, atau bawah sesuai dengan gerakan background. *Game* jenis ini yang sangat terkenal diantaranya adalah: Contra, Super Mario, Metal Slug, dan lain sebagainya.

b. *Racing*

Game racing ini merupakan *game* balapan. Contoh *game racing* yang paling terkenal adalah Need For Speed, Grand Tourismo, Nascar Rumble, Top Gear, dan lain sebagainya.

c. *Fighting*

Yang termasuk kategori *game* jenis ini adalah *game-game fighting* atau tarung. Contoh *game* yang terkenal diantaranya Street Fighter, Mortal Kombat, Teken, dan lain sebagainya.

d. RPG (*Role Playing Game*)

RPG adalah salah satu game yang mengandung unsure experience atau leveling dalam *gameplay*-nya. Biasanya dalam *game* ini kita memiliki kebebasan untuk menjelajahi dunia *game* tersebut[6]. Contoh *game* RPG action, dan Turn Based RPG.

e. Olahraga

Permainan ini diusahakan serealistik mungkin walau kadang ada yang menambahkan usur fiksi seperti NBA JAM. Contohnya pun jelas, seri *winning eleven*, tony hawk pro skater, Pro Evolution Soccer, dan lain sebagainya[6].

f. Adventure

Adventure game adalah tipe game yang umumnya membuat pemain harus berjalan mengelilingi suatu tempat yang terkondisi, seperti sebuah istana, gua yang berkelok, dan planet yang jauh. Pemain melakukan navigasi suatu area mencari pesan-pesan rahasia, memperoleh obyek

yang memiliki kemampuan yang bervariasi, bertempur dengan musuh, dan lain-lain. Untuk membuat game ini, diperlukan perencanaan yang akurat sehingga memiliki alur cerita yang menarik bagi pemain, contohnya: Tomb Raider, Prince of Persia, Hitman, dll

2.2 Sejarah Gatotkaca

Gatotkaca adalah seorang tokoh dalam wiracarita Mahabharata, putra Bimasena (Bima) atau Wrekodara dari keluarga Pandawa. Ibunya bernama Hidimbi (Harimbi), berasal dari bangsa rakshasa. Gatotkaca dikisahkan memiliki kekuatan luar biasa. Dalam perang besar di Kurukshetra, ia menewaskan banyak sekutu Korawa sebelum akhirnya gugur di tangan Karna. Tokoh pewayangan Gatotkaca bisa dilihat pada gambar 2.1 :



Gambar 2. 1 Pewayangan Gatotkaca

Di Indonesia, Gatotkaca menjadi tokoh pewayangan yang sangat populer. Misalnya dalam pewayangan Jawa, ia dikenal dengan sebutan Gatutkaca (bahasa Jawa: Gathutkaca). Kesaktiannya dikisahkan luar biasa, antara lain mampu terbang di angkasa tanpa menggunakan sayap, serta terkenal dengan julukan "otot kawat tulang besi"[2].

Gatotkaca adalah putra Arya Bima/Wijasena/Werkodara, penenggak Pandawa di kesatrian Jodipati, dengan Dewi Arimbi/Hidimbi putrid kerajaan Pringgandani. Diberi nama gatotkaca yang mempunyai arti: gata berarti kelonting utkaca berarti kepala.

Arti kata : orang yang berkepala seperti kelonting besar, yang mempunyai arti bahwa ia adalah seorang yang sangat pandai.

Gatotkaca mempunyai tiga orang istri, masing-masing dianugerahi seseorang putra, yaitu :

1. Dewi Sumpani berputra Arya Jayasumpena.
2. Dewi Pregiwa mempunyai putra bernama Sasikirana/Sasikirna.
3. Dewi Suryati memperoleh seorang putra bernama Suryakeca.

Gatotkaca mempunyai sifat perwatakan: berani tak mengenal takut, teguh, tangguh, cerdik, pandai, waspada, gesit, tangkas, dan terampil, tabah dan mempunyai rasa tanggungjawab yang besar. Ia sangaty sakti, sehingga digambarkan sebagai ksatria yang mempunyai “otot kawat balung besi”, sumsum gagala, kulit tembaga, driji gunting, dengkul paron dan sebagainya. Pandai terbang tanpa sayap.

Gatotkaca tak pernah menggunakan senjata dalam peperangan. Keistimewaannya kalau memilit leher lawan sampai putus, berperang menyambar seperti burung rajawali. Ia memiliki aji Narantaka atas pemberian resi Seta, gurunya. Aji tersebut diturunkan untuk menghadapi Dursala, putra Dursasana, yang mempunyai aji Ginong yang luar biasa[7].

2.3 Sejarah AI (*Artificial Intelligence*)

Istilah AI pertama kali dikemukakan pada tahun 1956 di konferensi Darthmouth. Sejak saat itu, AI terus dikembangkan sebab berbagai penelitian mengenai teori dan prinsip-prinsipnya juga terus berkembang. Meskipun istilah AI baru muncul tahun 1956, tetapi teori-teori yang mengarah ke AI sudah mucul sejak tahun 1941[8].

2.3.1 Definisi AI

Definisi AI dapat dibagi menjadi empat kategori, yaitu :

A. *Thinking Humanly* : Pendekatan kognitif

B. *Acting Humanly* : pendekatan uji Turing

C. Thingking Rationally : The laws of Thought Approach

D. *Acting Rationally* : *The Rational Agent Approach*

2.3.2 Ruang lingkup dan Elemen-elemen AI

A. Sistem pakar

Komputer memiliki keahlian untuk menyelesaikan masalah dengan meniru keahlian yang dimiliki oleh pakar.

B. Natural language Processing

Diharapkan user dapat berkomunikasi dengan komputer menggunakan bahasa sehari-hari.

C. Speech recognition

Melalui pengenalan ucapan, diharapkan manusia dapat berkomunikasi dengan komputer menggunakan suara.

D. Robotics & Sensory Systems

E. Computer Vision

Menginterpretasikan gambar atau objek-objek tampak melalui computer

F. Inteliigence Computer

Komputer digunakan sebagai tutor yang dapat melatih dan mengajar Game Playing[8].

2.4 Target Pengguna

Anak-anak saat ini sudah semakin "melek" teknologi. Bagaimana tidak, mereka bisa mengoperasikan *smartphone* atau dibekali orangtua dengan tablet.

Hal ini mengakibatkan anak semakin cerdas dalam memanfaatkan teknologi, tetapi dapat juga menyebabkan anak kecanduan *game* yang mereka mainkan di gadget tersebut. Jika kondisi seperti ini, orangtua lain perlu memerhatikan umur berapa anak boleh dibekali gadget.

Penelitian terbaru dari Cashinyourgadgets.co.uk, sebuah situs yang menyediakan layanan purna jual gadget, menemukan bahwa 13 persen orangtua membelikan gadget pertama untuk anak yang berumur di bawah 10 tahun, dan 45 persen orangtua membelikan anak gadget ketika mereka berumur 13 tahun ke atas.

Lalu usia berapa, orangtua memberikan anak mereka gadget pertama?

Jawaban yang paling populer adalah 35 persen mengatakan di atas 13 tahun, seperempatnya memberikan pada usia antara 10 -13 tahun, dan lebih dari 13 persen memberi gadget ke anak saat usia enam tahun. Katy Hayden, dari lembaga pengasuh Childcare Expert, mengatakan waktu terbaik untuk membeli ponsel atau gadget pertama untuk anak tergantung pada individu.

Namun, diketahui bahwa anak-anak pada usia 11 akan selalu menginginkan gadget teknologi terbaru, karena kemungkinan besar mereka mulai sekolah menengah, dan melihat anak-anak yang lebih dewasa dengan iPhone dan iPad, dan ingin menjadi seperti mereka [9].

2.5 Algoritma A* (A Star)

Algoritma ini merupakan algoritma *Best First Search* yang meng gabungkan *Uniform Cost Search* dan *Greedy Best First Search*. Biaya yang diperhitungkan didapat dari biaya sebenarnya ditambah dengan biaya perkiraan. Dalam notasi matematika dituliskan sebagai berikut:

$$f(n) = g(n) + h(n) \qquad \ldots.(2.1)$$

dengan perhitungan biaya seperti ini, algoritma A* adalah komplit dan optimal.

Algorima A*, dapat juga disebut sebagai Algoritma A Star, merupakan salah satu contoh algoritma pencarian yang cukup popular di dunia. Jika kita

mengetikkan Algoritma A* pada sebuah mesin pencari, seperti google.com, maka akan kita temukan 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)[10].

Starting point adalah sebuah terminologi untuk posisi awal sebuah benda. A adalah simpul yang sedang dijalankan dalam algoritma 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[10].

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 yang bertetangga 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

tersebut dilakukan berulang-ulang hingga terdapat solusi atau tidaka ada lagi simpul lain yang berada pada open list[10].

2.5.1 Fungsi Heuristik

Algoritma A* adalah algoritma pencarian yang menggunakan fungsi heuristic untuk ‘menuntun’ pencarian rute, khususnya dalam hal pengembangan dan pemeriksaan node-node pada peta. Fungsi-fungsi heuristik yang biasa digunakan dalam algoritma A*, antara lain [20]:

1. Manhattan

   Fungsi ini merupaka fungsi heuristik yang paling umum digunakan. Fungsi ini hanya akan menjumlahkan selisih nilai x dan nilai y dari dua buah titik. Fungsi heuristik ini dinamakan Manhattan karena berasal dari Manhattan di Amerika, jarak dari dua lokasi umumnya dihitung dari blok-blok yang harus dilalui saja dan tentunya tidak bisa dilintasi secara diagonal. Perhitungannya dapat ditulis sebagai berikut :

   $$h(n)= abs(tujuan.x – n.x) + abs(tujuan.y – n.y) \quad \ldots.(2.2)$$

   Dimana h(n) merupakan perkiraan cost node n ke node tujuan yang dihitung dengan fungsi heuristik. Variabel n.x merupakan koordinat x dari node n, sedangkan n.y merupakan koordinat y dari node n. Variabel tujuan.x merupakan koordinat x dari node tujuan dan tujuan.y merupakan koordinat y dari node tujuan. Nilai h(n) akan selalu bernilai positif.

2.6 Android

Android adalah sistem operasi untuk telepon seluler berbasis Linux sebagai kernelnya. Saat ini Android bisa disebut raja dari smartphone. Mengapa Android begitu pesat perkembangan di era saat ini? Karena Android menyediakan platform terbuka (*Open Source*) bagi para pengembang untuk menciptakan aplikasi mereka sendiri. Awalnya, perusahaan search engine terbesar saat ini, yaitu Google Inc. membeli Android Inc. , pendatang baru yang membuat peranti lunak untuk ponsel. Android, Inc. didirikan oleh Andy Rubin, Rich Milner, Nick Sears dan Chris White pada tahun 2003. Pada Agustus 2005 Google

membeli Android Inc. Kemudian untuk mengembangkan Android dibentuklah *Open Handset Alliance* konsorsium dari 34 perusahaan *hardware*, *software* dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualqomm, T-Mobile dan Nividia [11].

1. Versi Android

   Berikut adalah perkembangan versi Android :

   a. Android versi 2.3 (Gingerbread)

      Pada 6 Desember 2010, Android versi 2.3 (Gingerbread) diluncurkan. Perubahan-perubahan umum yang didapat dari Android versi ini antara lain peningkatan kemampuan permainan (gaming), peningkatan fungsi copy paste, layar antar muka (User Interface) didesain ulang, dukungan format video VP8 dan WebM, efek audio baru (reverb, equalization, headphone virtualization, dan bass boost), dukungan kemampuan Near Field Communication (NFC), dan dukungan jumlah kamera yang lebih dari satu.

   b. Android versi 4.0 (Ice Cream)

      Android versi 4.0 akan dirilis akhir tahun 2011. Setelah kita ketahui versi Android ini perlu diketahui bahwa nama lain dari versi-versi tersebut diambil oleh Google dari nama makanan penutup [11].

2.7 Multimedia

Multimedia adalah penggunaan komputer untuk menyajikan dan menggabungkan teks, suara, gambar, animasi dan video dengan alat bantu (tool) dan koneksi (link) sehingga pengguna dapat bernavigasi, berinteraksi, berkarya dan berkomunikasi (Hofstetter 2001). Multimedia sering digunakan dalam dunia hiburan. Selain dari dunia hiburan, Multimedia juga diadopsi oleh dunia Game.Multimedia juga dapat diartikan sebagai penggunaan beberapa media yang berbeda dalam menyampaikan informasi berbentuk text, audio, grafik, animasi, dan video[12].

2.7.1 Elemen Multimedia

Menurut Hofstetter (2001, p16) komponen multimedia terbagi atas lima jenis yaitu:

1. Teks

   Teks merupakan elemen multimedia yang menjadi dasar untuk menyampaikan informasi, karena teks adalah jenis data yang paling sederhana dan membutuhkan tempat penyimpanan yang paling kecil. Teks merupakan cara yang paling efektif dalam mengemukakan ide-ide kepada pengguna, sehingga penyampaian informasi akan lebih mudah dimengerti oleh masyarakat. Jenis-jenis teks seperti Printed Text, yaitu teks yang dihasilkan oleh word processor atau word editor dengan cara diketik yang nantinya dapat dicetak. Scanned Text yaitu teks yang dihasilkan melalui proses scanning tanpa pengetikan. Dan Hypertext yaitu jenis teks yang memberikan link ke suatu tempat / meloncat ke topik tertentu.

2. Grafik (image)

   Sangat bermanfaat untuk mengilustrasi informasi yang akan disampaikan terutama informasi yang tidak dapat dijelaskan dengan kata-kata. Jenis-jenis grafik seperti bitmap yaitu gambar yang disimpan dalam bentuk kumpulan pixel, yang berkaitan dengan titik-titik pada layar monitor. Digitized picture adalah gambar hasil rekaman video atau kamera yang dipindahkan ke komputer dan diubah ke dalam bentuk bitmaps. Hyperpictures, sama seperti hypertext hanya saja dalam bentuk gambar.

3. Audio

   Multimedia tidak akan lengkap jika tanpa audio (suara). Audio bisa berupa percakapan, musik atau efek suara.

   Format dasar audio terdiri dari beberapa jenis :

   a. WAVE

      Merupakan format file digital audio yang disimpan dalam bentuk digital dengan eksistensi WAV

b. MIDI

MIDI (Musical Instrument Digital Interface) memberikan cara yang lebih efisien dalam merekam music dibandingkan wave, kapasitas data yang dihasilkan juga jauh lebih kecil. MIDI disimpan dalam bentuk MID.

4. Video

Video menyediakan sumber yang kaya dan hidup untuk aplikasi multimedia. Dengan video dapat menerangkan hal-hal yang sulit digambarkan lewat kata-kata atau gambar diam dan dapat menggambarkan emosi dan psikologi manusia secara lebih jelas.

5. Animasi

Animasi adalah simulasi gerakan yang dihasilkan dengan menayangkan rentetan frame ke layer. Frame adalah satu gambar tunggal pada rentetan gambar yang membentuk animasi.

Menurut Foley, Van Dam, Feiner dan Hughes (1997, p1057) Animate adalah untuk membuat sesuatu hidup, sebagian orang mengira bahwa animasi itu sama dengan motion (gerakan), tetapi animasi mencakup semua yang mengandung efek visual sehingga animasi mencakup perubahan posisi terhadap waktu, bentuk, warna, struktur, tekstur dari sebuah objek, posisi kamera, pencahayaan, orientasi dan focus dan perubahan dalam teknik rendering[12].

2.8 Adobe Flash

Flash merupakan software yang memiliki kemampuan menggambar sekaligus menganimasikannya, serta mudah dipelajari. Flash tidak hanya digunakan dalam pembuatan animasi, tetapi pada zaman sekarang ini flash juga banyak digunakan untuk keperluan lainnya seperti dalam pembuatan game, presentasi, membangun web, animasi pembelajaran, bahkan juga dalam pembuatan film.

Animasi yang dihasilkan flash adalah animasi berupa file movie. Movie yang dihasilkan dapat berupa grafik atau teks. Grafik yang dimaksud disini adalah grafik yang berbasis vektor, sehingga saat diakses melalui internet, animasi akan ditampilkan lebih cepat dan terlihat halus. Selain itu flash juga memiliki kemampuan untuk mengimpor file suara, video maupun file gambar dari aplikasi lain. Flash adalah program grafis yang diproduksi pertama kali oleh Macromedia corp, yaitu sebuah vendor software yang bergerak dibidang animasi web. Macromedia Flash pertama kali diproduksi pada tahun 1996. Macromedia flash telah diproduksi dalam beberapa versi. Versi terakhir dari Macromedia Flash adalah Macromedia flash 8. Sekarang Flash telah berpindah vendor ke Adobe. Semua tools pada dasarnya sama, hanya yang membedakan disini adalah adanya jenis Actionsript 3.0. Actionscript ini merupakan versi terbaru dari penulisan actionscript di flash. Namun anda jangan khawatir, actionscript 2.0 masih berlaku di Adobe Flash [13].

2.9 OOP (*Object Oriented Programming*)

Dalam dunia pemrograman,terdapat istilah yang dikenal dengan Object-Oriented Programming (disingkat OOP). Tipe pemrograman ini adalah berbasis object. Maksudnya adalah penggunaan sudut pandang object sebagai sebuah komponen dalam sebuah program yang dimodelkan dengan benda - benda dalam dunia nyata. Contoh object yang ada di sekitar adalah hewan, mobil, televisi, sepeda, dan masih banyak lainnya. Objek yang dilihat tentunya memiliki hal atau karakteristik yang melekat pada dirinya sendiri seperti ukuran, warna, tinggi, dan bentuk yang disebut attribute. Selain itu, object yang dikenal memiliki tingkah laku dan kemampuan untuk melakukan suatu tugas, berasal dari dalam dirinya sendiri, yang disebut behavior atau method seperti mobil berjalan ke depan, hewan tidur, televisi dinyalakan, dan lampu dimatikan. Pembuatan object dalam OOP ini sangat erat hubungannya dengan pembuatan class [14].

1. Objek

   Objek adalah orang, tempat, benda, kejadian, atau konsep-konsep yang ada di dunia nyata yang penting bagi suatu aplikasi (perangkat lunak dan/atau sistem

informasi). Objek orang misalnya adalah saya, anda, dosen dsb. Objek tempat misalnya adalah kampus, negara, jalan, kota dan sebagainya. Objek benda misalnya adalah mesin, gedung, komputer, keyboard gitar dsb. Objek kejadian misalnya adalah pembayaran, membaca buku dsb. Objek konsep misalnya liberalisme, kapitalisme dsb.

2. Kelas

   Kumpulan/himpunan objek dengan atribut/properti yang mirip, perilaku (operasi) yang mirip, serta hubungan dengan objek lain dengan cara yang mirip. Kebanyakan bahasa pemrograman berorientasi objek dapat menentukan kelas dari suatu objek saat aplikasi dijalankan. Dengan penggolongan objek-objek dalam suatu kelas, dapat dilakukan dengan abstraksi masalah. Fungsi-fungsi serta metoda-metoda yang sama cukup dituliskan satu kali saja dan dapat digunakan ulang oleh setiap objek yang termasuk dalam kelas yang sama. Hal ini akan sangat meningkatkan efisiensi dan efektivitas program.

3. Atribut

   Merupakan data yang dimiliki suatu objek dalam kelas. Setiap atribut akan memiliki nilai berbeda untuk setiap objek.

4. Metoda dan operasi

   merupakan fungsi atau transformasi yang mungkin dapat diaplikasikan ke/oleh suatu objek dalam kelas. Misalnya, suatu objek dalam kelas manusia mungkin memiliki fungsi-fungsi: tersenyum, marah, makan, minum dsb.

5. Konsep berorientasi objek

   a. *encapsulation* (pengapsulan)

      Proses pemaketan data bersama metode-metodenya. Manfaat utama pengapsulan adalah menyembunyikan rincian-rincian implementasi dari pemakai.Bagian internal objek mempunyai ketampakan terbatas dibanding bagian eksternal objek. Penyusunan seperti ini menjaga bagian intenal dari pengaksesan eksternal yang tidak diinginkan.

Pengapsulan menyediakan dua manfaat utama bagi pemrogram, yaitu: Penyembunyian informasi Penyembunyian implementasi (implementation hiding) mengacu perlindungan implementasi internal objek. Objek disusun dari antarmuka public dan bagian private yang merupakan kombinasi data dan metode internal. Manfaat utama adalah bagian internal dapat berubah tanpa mempengaruhi bagian-bagian program yang lain.

b. *inheritance* (pewarisan)

Merupakan proses penciptaan kelas baru dengan mewarisi karakteristik kelas yang telah ada, ditambah karakteristik kelas unik kelas baru itu. Pewarisan adalah mekanisme ampuh yang memungkinkan kelas mewarisi fungsionalitas kelas yang telah ada. Untuk menciptakan kelas baru, hanya perlu menspesifikasikan cara kelas baru itu berbeda dari kelas yang telah ada.

c. Method

Behavior atau tingkah laku adalah hal-hal yang bisa dilakukan oleh objek dari suatu *class.* Behavior dapat digunakan untuk mengubah nilai atribut suatu objek, menerima informasi dari objek lain, dan mengirim informasi ke objek lain untuk melakukan suatu tugas (*task*). Dalam *class*, behavior disebut juga sebagai *method. Method* sendiri adalah serangkaian pernyataan dalam suatu *class* yang menghandle suatu *task* tertentu. Cara objek berkomunikasi dengan objek yang lain dengan menggunakan *method.*

d. Polimorfisem

Polimorfisem adalah kemampuan suatu objek untuk mempunyai lebih dari satu bentuk. *Polimorfisme* tidak bergantung kepada pemanggilan subrutin. Metode tertentu yang berhubungan dengan sebuah pengiriman pesan tergantung kepada objek tertentu dimana pesan tersebut dikirim.

Polimorfisme bisa juga diartikan sebagai aksi yang sama yang dapat dilakukan terhadap beberapa objek. *Polimorfisme* berarti bahwa operasi yang sama mungkin mempunyai perbedaan dalam *class* yang berbeda.

2.10 UML (*Unified Modeling Language*)

Pemodelan (*modeling*) adalah proses merancang piranti lunak sebelum melakukan pengkodean (*coding*). Model piranti lunak dapat dianalogikan seperti pembuatan blueprint pada pembangunan gedung. Membuat model dari sebuah sistem yang kompleks sangatlah penting karena kita tidak dapat memahami sistem semacam itu secara menyeluruh. Semakin komplek sebuah sistem, semakin penting pula penggunaan teknik pemodelan yang baik.

Dengan menggunakan model, diharapkan pengembangan piranti lunak dapat memenuhi semua kebutuhan pengguna dengan lengkap dan tepat, termasuk faktor-faktor seperti *scalability, robustness, security*, dan sebagainya.

Kesuksesan suatu pemodelan piranti lunak ditentukan oleh tiga unsur, yang kemudian terkenal dengan sebuan segitiga sukses (*the triangle for success*). Ketiga unsur tersebut adalah metode pemodelan (*notation*), proses (*process*) dan tool yang digunakan.

Memahami notasi pemodelan tanpa mengetahui cara pemakaian yang sebenarnya (proses) akan membuat proyek gagal. Dan pemahaman terhadap metode pemodelan dan proses disempurnakan dengan penggunaan tool yang tepat [15].

Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan syntax/ semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. 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: Grady Booch OOD (*Object-Oriented Design*), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*) [15].

1. *Use Case*

   Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan

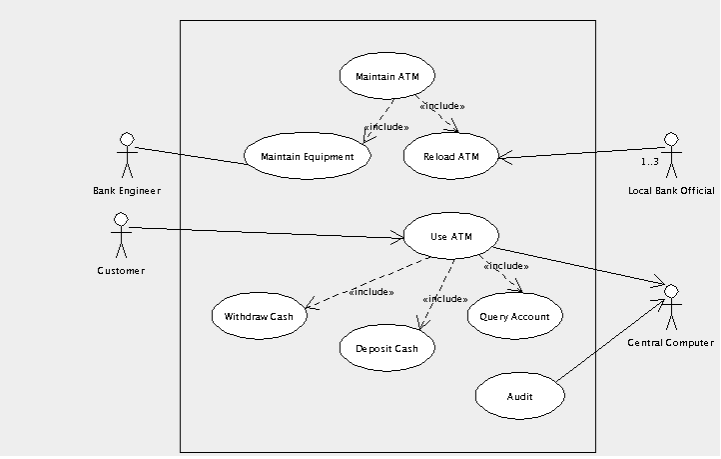
bukan “bagaimana”. Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-create sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu.

Use case diagram dapat sangat membantu bila kita sedang menyusun requirement sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang test *case* untuk semua feature yang ada pada sistem.

Sebuah *use case* dapat melakukan 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 melakukan 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 [15].

Contoh *use case* diagram bisa dilihat pada gambar 2.2 :



Gambar 2. 2 Contoh Use Case Diagram

2. *Activity Diagram*

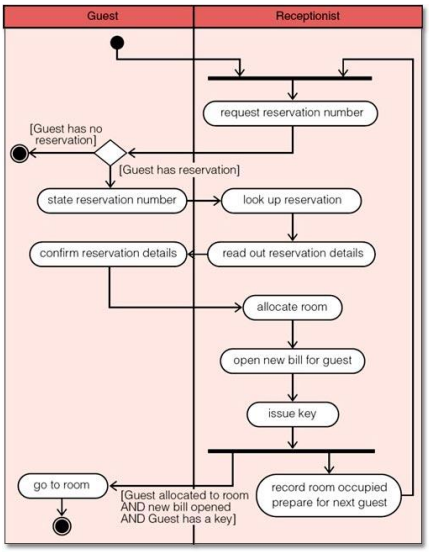
Activity diagrams menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, decision yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity* diagram juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

Activity diagram merupakan state diagram khusus, 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 behaviour internal sebuah system (dan interaksi antar sub sistem) secara eksak, tetapi lebih menggambarkan proses - proses dan jalur-jalur aktivitas dari level atas secara umum.

Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas.

Sama seperti state, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan *behaviour* pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal. Activity diagram dapat dibagi menjadi beberapa object *swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu [15].

Contoh *activity* diagram tanpa *swimlane* bisa dilihat pada gambar 2.3 :



Gambar 2. 3 Contoh Activity Diagram

3. Sequence Diagram

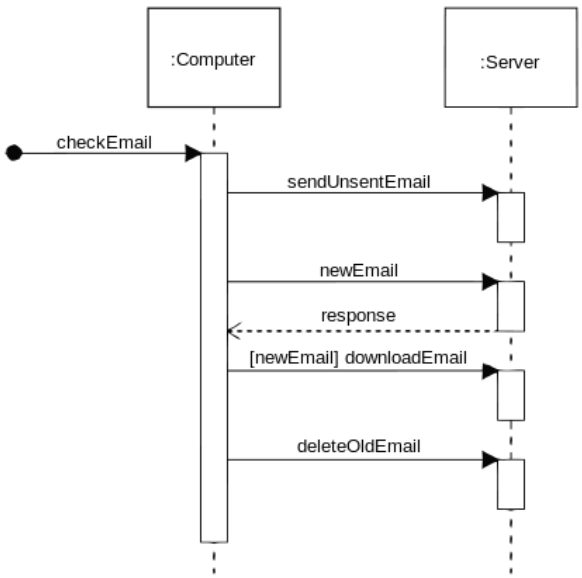
Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar system (termasuk pengguna, display, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. Sequence diagram terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah event untuk menghasilkan output tertentu. Diawali dari apa yang men-trigger aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan.

Masing-masing objek, termasuk aktor, memiliki lifeline vertikal. Message digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, message akan dipetakan menjadi operasi/metoda dari class. Activation bar menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah message.

Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan icon khusus untuk objek boundary, controller dan persistent entity [15].

Contoh *sequence diagram* bisa dilihat pada gambar 2.4 :



Gambar 2. 4 Contoh Sequence Diagram

4. *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 [15].

   Class memiliki tiga area pokok :

   1. Nama (dan stereotype)
   2. Atribut
   3. Metoda

   Atribut dan metoda dapat memiliki salah satu sifat berikut :

   1. *Private*, tidak dapat dipanggil dari luar class yang bersangkutan
   2. *Protected*, hanya dapat dipanggil oleh class yang bersangkutan dan anak-anak yang mewarisinya

3. *Public*, dapat dipanggil oleh siapa saja

Class dapat merupakan implementasi dari sebuah interface, yaitu *class* abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class*. Dengan demikian *interface* mendukung resolusi metoda pada saat *run-time*.

Hubungan Antar *Class*

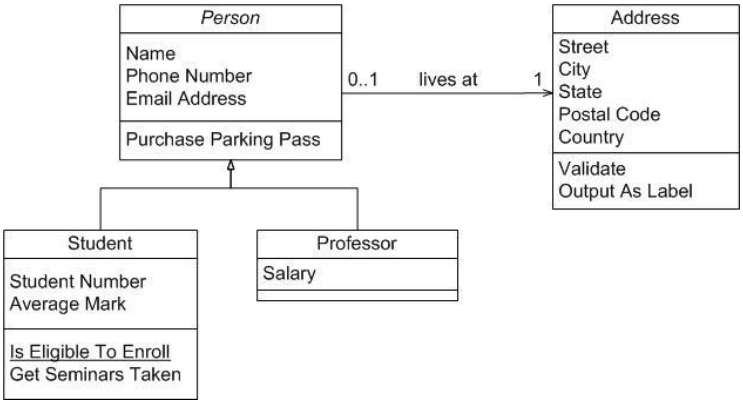
1. Asosiasi, yaitu hubungan statis antar class. Umumnya menggambarkan class yang memiliki atribut berupa class lain, atau class yang harus mengetahui eksistensi class lain. Panah navigability menunjukkan arah query antar class.

2. Agregasi, yaitu hubungan yang menyatakan bagian ("terdiri atas..").

3. Pewarisan, yaitu hubungan hirarkis antar class. Class dapat diturunkan dari class lain dan mewarisi semua atribut dan metoda class asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari class yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.

4. Hubungan dinamis, yaitu rangkaian pesan (message) yang di-passing dari satu class kepada class lain. Hubungan dinamis dapat digambarkan dengan menggunakan sequence diagram yang akan dijelaskan kemudian.

Contoh class diagram bisa dilihat pada gambar 2.5 :



Gambar 2. 5 Contoh class Diagram

2.10.1 Pengujian White Box

Pengertian *White Box Testing* merupakan cara pengujian dengan melihat ke dalam modul untuk meneliti kode-kode program yang ada, dan menganalisis apakah ada kesalahan atau tidak. Jika ada modul yang menghasilkan output yang tidak sesuai dengan proses bisnis yang dilakukan, maka baris-baris program, variabel, dan parameter yang terlibat pada unit tersebut akan dicek satu persatu dan diperbaiki, kemudian di-*compile* ulang [16].

Dengan menggunakan *white box* akan didapatkan kasus uji yang :

a. Menguji semua keputusan logikal
b. Menguji seluruh *Loop* yang sesuai dengan batasannya
c. Menguji seluruh struktur data internal yang menjamin validitas

1. Kelebihan *White Box Testing*

- Kesalahan Logika

Digunakan pada sintaks ‘if’ dan pengulangan. Dimana *White Box Testing* akan mendeteksi kondisi-kondisi yang tidak sesuai dan mendeteksi kapan proses pengulangan akan berhenti

- Ketidaksesuaian asumsi

Menampilkan asumsi yang tidak sesuai dengan kenyataan, untuk di analisa dan diperbaiki.

- Kesalahan ketik

Mendeteksi bahasa pemrograman yang bersifat *case sensitive*.

2.10.2 Pengujian Black Box

Metode ujicoba *blackbox* memfokuskan pada keperluan fungsional dari software. Karna itu ujicoba *blackbox* memungkinkan pengembang software untuk membuat himpunan kondisi input yang akan melatih seluruh syarat-syarat fungsional suatu program. Ujicoba *blackbox* bukan merupakan alternatif dari ujicoba *whitebox*, tetapi merupakan pendekatan yang melengkapi untuk

menemukan kesalahan lainnya, selain menggunakan metode *whitebox* [17]. Ujicoba *blackbox* berusaha untuk menemukan kesalahan dalam beberapa kategori, diantaranya :

1. Fungsi-fungsi yang salah atau hilang
2. Kesalahan *interface*
3. Kesalahan dalam struktur data atau akses database eksternal
4. Kesalahan performa
5. kesalahan inisialisasi dan terminasi

2.11 Skala Likert

Skala Likert adalah skala yang dapat dipergunakan untuk mengukur sikap, pendapat, dan persepsi seseorang atau sekelompok orang mengenai suatu gejala atau fenomena pendidikan. Dalam skala Likert terdapat dua bentuk pernyataan yaitu pernyataan positif yang berfungsi untuk mengukur sikap positif dan pernyataan negatif yang berfungsi untuk mengukur sikap negatif.

Skala Likert dalam penelitian mempunyai gradasi nilai dari sangat positif sampai sangat negatif. Lalu hasil data kuesioner tersebut diolah dengan rumus sebagai berikut:

$$Y = P/Q*100\% \quad \text{....(2.1)}$$

Keterangan :

Y = Persentase

P = Bobot setiap pilihan * jawaban

Q = Total bobot maksimal