BAB II

TINJAUAN PUSTAKA

II.1. Pengertian Implementasi

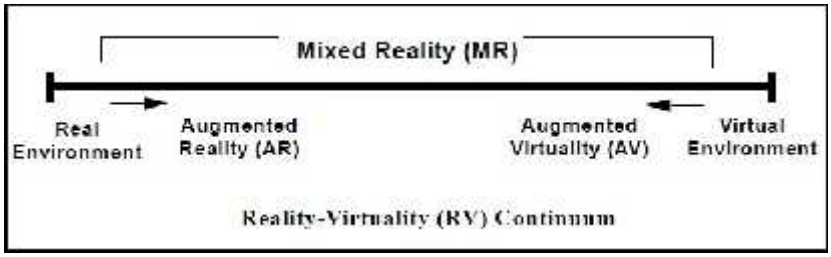
Menurut Nurdin Usman (2002: 70) dalam Jurnal Yoseph Pahlefi, mengemukakan "Implementasi adalah bermuara pada aktivitas, aksi, tindakan, atau adanya mekanisme suatu sistem. Implementasi bukan sekedar aktivitas, tetapi suatu kegiatan yang terencana dan untuk mencapai tujuan kegiatan". Sedangkan menurut Guntur Setiawan (2004: 39) dalam Jurnal Yoseph Pahlefi, mengatakan "Implementasi adalah perluasan aktivitas yang saling menyesuaikan proses interaksi antara tujuan dan tindakan untuk mencapainya serta memerlukan jaringan pelaksana, birokrasi yang efektif".

II.2. *Augmented Reality*

Ronald T. Azuma (2008) dalam Jurnal Atmoko Nugroho dan Basworo Adi, mendefinisikan *augmented reality* sebagai penggabungan benda-benda nyata dan maya di lingkungan nyata, berjalan secara interaktif dalam waktu nyata, dan terdapat integrasi antarbenda dalam tiga dimensi, yaitu benda maya terintegrasi dalam dunia nyata. Penggabungan benda nyata dan maya dimungkinkan dengan teknologi tampilan yang sesuai, interaktivitas dimungkinkan melalui perangkat-perangkat *input* tertentu, dan integrasi yang baik memerlukan penjejakan yang efektif. Sedangkan menurut Stephen Cawood & Mark Fiala dalam bukunya yang berjudul *Augmented reality: a practical guide*, mendefinisikan bahwa *Augmented*

Reality merupakan cara alami untuk mengeksplorasi objek 3D dan data, AR merupakan suatu konsep perpaduan antara *virtual reality* dengan *world reality*. Sehingga objek objek virtual 2 Dimensi (2D) atau 3 Dimensi (3D) seolah-olah terlihat nyata dan menyatu dengan dunia nyata. Pada teknologi AR, pengguna dapat melihat dunia nyata yang ada di sekelilingnya dengan penambahan obyek virtual yang dihasilkan oleh komputer.

Menurut Rujianto Eko Saputro, dkk (2014) Augmented reality memungkinkan perspektif diperkaya dengan menampilkan obyek virtual pada dunia nyata dengan cara mengajak penonton bahwa obyek virtual adalah bagian dari lingkungan nyata. Augmented reality merupakan crossover antara dunia nyata dan virtual (Milgram, 1994). konsep ini diilustrasikan pada Gambar II.1 sebagai berikut.

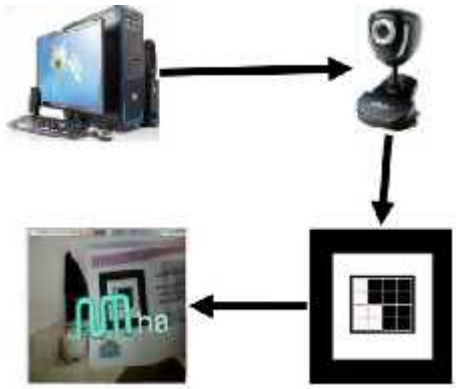


Gambar II.1. Arsitektur *Augmented Reality*
(*Sumber : Rujianto Eko Saputro, dkk. 2014)*

Menurut Riana Indriani, dkk (2016) Arsitektur teknologi perangkat *Augmented Reality* yaitu :

1. *Input* Pada AR, proses input ini sistem mendeteksi sensor dari benda nyata. Seperti gambar, sensor getaran, lokasi, hingga sensor gerakan.

2. Kamera/Alat penangkap sensor lainnya, Disini sensor-sensor dari dunia nyata diterima dan dijadikan informasi yang nantinya akan di proses oleh sistem.
3. Prosessor, Disinilah proses inti dari kera teknologi AR. Pada bagian ini, sistem akan memproses informasi yang masuk dan menemukan informasi apa yang akan di keluarkan.
4. *Output*, menampilkan informasi-informasi yang sudah ada di proses. Output dapat berupa Monitor, Layar Ponsel, dst. Illustrasinya dapat dilihat pada Gambar II.2 sebagai berikut :



Gambar II.2. Ilustrasi Arsitektur Perangkat Augmented Reality

(*Sumber: Setia Wardani. 2015)*

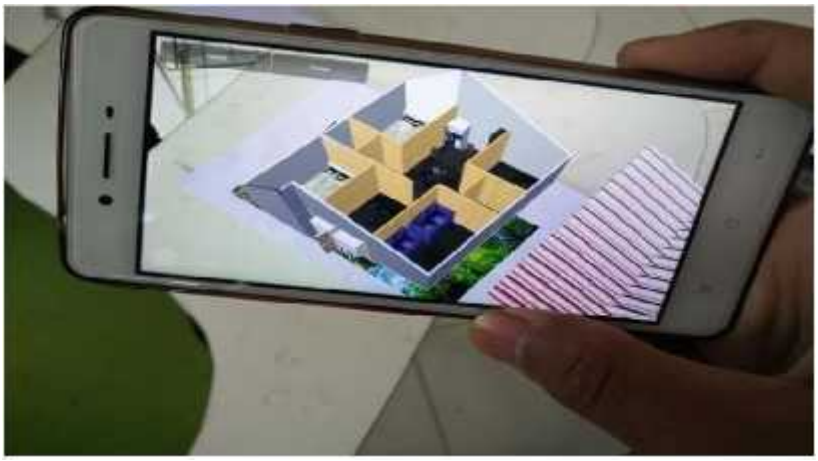
Menurut Gun Gun Maulana (2017) Sistem *augmented reality* pada umumnya bekerja berdasarkan pendeteksian citra, yang sering disebut dengan istilah *marker*. Prinsip kerjanya sebenarnya cukup sederhana. Kamera atau *webcam* akan mendeteksi *marker* yang diberikan, kemudian sistem akan mengenali dan menandai pola *marker*, citra yang tertangkap kamera atau *webcam* akan diolah oleh sistem dengan melakukan perhitungan apakah

marker sesuai dengan *database* yang dimiliki. Informasi citra yang diterima tidak akan diolah bila *marker* tidak sesuai dengan *database* yang dimiliki sistem, tetapi bila sesuai maka informasi tersebut akan digunakan untuk menampilkan teks, video, objek 3 dimensi atau animasi yang telah dibuat sebelumnya.

Secara umum ada dua metode yang digunakan dalam teknologi AR ini yaitu *Marker based AR* dan *Markerless AR*.

1. Marker based Tracking AR

Menurut Abdur Rahman , dkk (2014) *Marker based tracking* adalah AR yang menggunakan *marker* atau penanda objek dua dimensi yang memiliki suatu pola yang akan dibaca komputer melalui media *webcam* atau kamera yang tersambung dengan komputer, biasanya merupakan ilustrasi hitam dan putih persegi dengan batas hitam tebal dan latar belakang putih.



Gambar II.3. *Marker Based AR*

(Sumber: Gun Gun Maulana. 2017)

2. Markerless AR

Menurut Abdur Rahman , dkk (2014) Dengan metode *markerless* pengguna tidak perlu lagi mencetak sebuah *marker* untuk menampilkan elemen-elemen digital. Dalam hal ini, *marker* yang dikenali berbentuk posisi perangkat, arah, maupun lokasi.

Menurut Abdur Rahman , dkk (2014) *Augmented Reality* dengan berbagai macam teknik *Markerless* Tracking diantaranya adalah *Face Tracking, 3D Objects Tracking, Motion Tracking* dan *GPS Based Tracking*.

a. Face Tracking

Dengan Menggunakan algoritma yang mereka kembangkan komputer dapat mengenali wajah manusia secara umum dengan cara mengenali posisi mata, hidung, dan mulut manusia, kemudian akan mengabaikan objek-objek lain di sekitarnya seperti pohon, rumah, dan benda-benda lainnya.

b. 3D Object Tracking

Berbeda dengan Face Tracking yang hanya mengenali wajah manusia secara umum, teknik *3D Object Tracking* dapat mengenali semua bentuk benda yang ada disekitar, seperti mobil, meja, televisi, dan lain-lain.

c. Motion Tracking

Pada teknik ini komputer dapat menangkap gerakan, *Motion Tracking* telah mulai digunakan secara ekstensif untuk memproduksi film-film yang mencoba mensimulasikan gerakan. Contohnya pada film Avatar, di mana James

Cameron menggunakan teknik ini untuk membuat film tersebut dan menggunakannya secara real time.

d. GPS Based Tracking

Pengembangan teknik ini lebih diarahkan pada smartphone, karena teknologi GPS dan kompas yang tertanam pada *smartphone* tersebut. Dengan memanfaatkan fitur GPS yang berfungsi sebagai penentu lokasi pengguna pada saat itu berada sehingga lokasi terdekat yang ingin dituju dapat dilihat melalui implementasi *augmented reality*.

II.3. Vuforia SDK

Menurut I Made Adi Yoga Dewantara , dkk (2014) Vuforia merupakan software *library* untuk *augmented reality*, yang menggunakan sumber yang konsisten mengenai *computer vision* yang fokus pada *image recognition*. Vuforia mempunyai banyak fitur-fitur dan kemampuan, yang dapat membantu pengembang untuk mewujudkan pemikiran mereka tanpa adanya batas secara teknikal. Dengan support untuk *iOS, Android, dan Unity3D, platform* Vuforia mendukung para pengembang untuk membuat aplikasi yang dapat digunakan di hampir seluruh jenis *smartphone* dan *tablet*. Pengembang juga diberikan kebebasan untuk mendesain dan membuat aplikasi yang mempunyai kemampuan antara lain :

1. Teknologi *computer vision* tingkat tinggi.
2. Terus-menerus mengenali *multiple image*.
3. *Tracking* dan *Detection* tingkat lanjut.

4. Dan solusi pengaturan *database* gambar yang fleksibel. Target pada vuforia merupakan obyek pada dunia nyata yang dapat dideteksi oleh kamera, untuk menampilkan obyek *virtual*.

II.4. Android SDK (*Software Development Kit)*

Menurut Wandy Damarullah (2013) Android SDK merupakan tools bagi para programmer yang ingin mengembangkan aplikasi berbasis google android. Android SDK mencakup seperangkat alat pengembangan yang komprehensif. Android SDK terdiri dari debugger, libraries, handset emulator, dokumentasi, contoh kode, dan tutorial.

II. 5. *Java Development Kit (*JDK)

Menurut Andi Juansyah (2015) Java Development Kit (JDK) adalah sekumpulan perangkat lunak yang dapat kamu gunakan untuk mengembangkan perangkat lunak yang berbasis Java, sedangkan JRE adalah sebuah implementasi dari Java Virtual Machine yang benarbenar digunakan untuk menjalankan program java. Baisanya, setiap JDK berisi satu atau lebih JRE dan berbagai alat pengembangan lain seperti sumber compiler java, bundling, debuggers, development libraries dan lain sebagainya.

II.6. Unity 3D Engine

Menurut Berta Sihite, dkk (2013) Banyak sekali peminat yang menginginkan hasil-hasil kreatif dalam pembuatan *software* berbasis *game*. Sehingga *software house* yang bersedia untuk mengembangkan *game engine*.Terdapat *game engine* yang berbayar dan tidak berbayar.

Unity Engine suatu *game engine* yang terus berkembang. *Engine* ini merupakan salah satu *game engine* dengan lisensi source proprietary, namun untuk lisensi pengembangan dibagi menjadi 2, yaitu *free* (gratis) dan berbayar sesuai perangkat target pengembangan aplikasi. *Unity* tidak membatasi publikasi aplikasi, pengguna *unity* dengan lisensi gratis dapat mempublikasikan aplikasi yang dibuat tanpa harus membayar biaya lisensi atau royalti kepada *unity*. Tetapi penggunaan versi *free* dibatasi dengan beberapa fitur yang dikurangi atau bonus modul / prefab tertentu yang ditiadakan dan hanya tersedia untuk pengguna berbayar.

Seperti kebanyakan *game engine* lainnya, *Unity* Engine dapat mengolah beberapa data seperti objek tiga dimensi, suara, teksture, dan lain sebagainya. Keunggulan dari *unity engine* ini dapat menangani grafik dua dimensi dan tiga dimensi. Namun *engine* ini lebih konsentrasi pada pembuatan grafik tiga dimensi. Dari beberapa *game engine* yang samasama menangani grafik tiga dimensi, *Unity Engine* dapat menangani lebih banyak. Beberapa diantaranya yaitu Windows, MacOS X, iOS, PS3, wii, Xbox 360, dan Android yang lebih banyak daripada game engine lain seperti *Source Engine, GameMaker, Unigine, id Tech 3 Engine, id Tech 4 Engine, Blender Game Engine, NeoEngine, Unity, Quake Engine, C4 Engine* atau *game engine* lain.

Unity Engine memiliki kerangka kerja (*framework*) lengkap untuk pengembangan profesional. Sistem inti engine ini menggunakan beberapa pilihan bahasa pemrogramam, diantaranya C#, *javascript* maupun *boo*. *Unity 3D editor* menyediakan beberapa alat untuk mempermudah pengembangan yaitu *Unity Tree*

dan *terrain creator* untuk mempermudah pembuatan vegetasi dan terrain serta *Mono Develop* untuk proses pemrograman.

II.7. 3D Studio Max

Menurut Sitaresmi Wahyu Handani, dkk (2016) Untuk dapat membuat animasi tiga dimensi diperlukan sebuah perangkat lunak. *Autodesk 3dsMax* atau sebelumnya *3D Studio Max* merupakan salah satu perangkat lunak yang banyak digunakan dalam proses pembuatan film animasi 3 dimensi. Dengan perangkat lunak ini, pengguna dapat membuat animasi 3 dimensi, model, permainan, dan gambar. Secara umum, untuk dapat membuat sebuah animasi 3 dimensi ada beberapa tahap yang dapat dilakukan, yaitu: *modelling, texturing, rigging, animation, dan rendering.*

II.8. Sistem Operasi Android

Menurut Heru Supriyono, dkk (2014) Android adalah salah satu *platform* sistem operasi yang digemari masyakat karena sifatnya yang *open source* sehingga memungkinkan pengguna untuk melakukan pengembangan. Android merupakan generasi baru *platform mobile* berbasis *linux* yang mencakup sistem operasi, *middleware,* dan aplikasi. Arsitektur Android terdiri dari bagian-bagian seperti berikut :

a. *Applications* dan *Widgets*: layer (lapisan) dimana pengguna hanya berhubungan dengan aplikasi saja.
b. *Applications Framework*: lapisan dimana para pengembang melakukan pembuatan aplikasi yang akan dijalankan di sistem operasi Android dengan

komponen-komponennya meliputi *views, contents provider, resource manager, notification manager, activity manager*.

c. *Libraries*: lapisan dimana fitur-fitur android berada yang berada diatas kernel meliputi library C/C++ inti seperti Libc dan SSL.

d. *Android Run Time*: lapisan yang membuat aplikasi Android dapat dijalankan dimana dalam prosesnya menggunakan implementasi *Linux* yang terbagi menjadi dua bagian yaitu *Core Libraries* dan *Dalvik virtual Machine*.

e. *Linux Kernel*: Layer yang berisi file-file system untuk mengatur *processing, memory, resource, driver*, dan sistem operasi android lainnya.

Sistem operasi yang mendasari Android dilisensikan dibawah GNU, GPLv2 (*General Public License verse 2*) yang sering dikenal dengan istilah *copyleft*. Pendistribusian Android dibawah lisensi dari *Apache Software* yang memungkinkan untuk distribusi kedua dan seterusnya.

II.9. UML (*Unified Modeling Language*)

Dalam Buku Ratna Wardani (2012) menuliskan bahwa *Unified Modelling Language* (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem perangkat lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem (Wahono, R.S, 2003).

Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi perangkat lunak, di mana aplikasi tersebut dapat berjalan pada perangkat keras, sistem operasi dan jaringan apapun, serta di tulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan *operation*

dalam konsep dasarnya, maka ia lebih cocok untuk penulisan perangkat lunak dalam bahasa-bahasa berorientasi obyek seperti *C++*, *Java*, *C#* atau *VB.NET*. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam *VB* atau *C* (2012 : 32).

Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut:

1. *Use case* Diagram

Dalam Buku Ratna Wardani (2012) menuliskan bahwa *Use case* merupakan *design* yang berfokus pada *user* dan tugas-tugas *user* untuk memenuhi keinginan *user* (Thimoty C, 2002).

Dalam Buku Ratna Wardani (2012) menuliskan bahwa *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, menmbuat sebuah daftar belanja dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu (Wahono, R.S,2003).

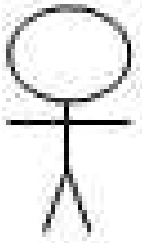
Use case diagram dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien dan merancang *test case* untuk semua fitur yang ada pada sistem. Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan

dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal. Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang umum.

Sebuah *use case* juga dapat memperpanjang *use case* lain dengan tingkah lakunya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.

Use case ini terdiri dari :

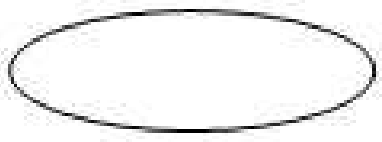
- *Actor* : Pemakai sistem/ sesuatu yang berinteraksi dengan sistem merepresentasikan pesan, bukan pemakai individual.



Gambar II.4. Tampilan Actor

(Sumber : Dr. Ratna Wardani, S.Si., M.T. 2012)

- *Use case* : Cara spesifik penggunaan sistem oleh aktor.



Gambar II.5. Tampilan Use Case

(Sumber : Dr. Ratna Wardani, S.Si., M.T. 2012)

Diagram *use case* :

- Merupakan salah satu diagram untuk memodelkan aspek perilaku sistem.
- Masing-masing menunjukkan sekumpulan *use case*, aktor, dan hubungannya.
- Untuk memvisualisasikan dan mendokumentasikan kebutuhan perilaku sistem.
- Melibatkan : sistem, aktor, *use case* dan relasi.

Tujuan utama memodelkan *use case* :

- Memutuskan dan mendeskripsikan kebutuhan fungsional sistem.
- Memberikan deskripsi jelas dan konsisten dari apa yang harus dilakukan.
- Menyediakan basis untuk melakukan pengujian sistem yang memverifikasi sistem.
- Menyediakan kemampuan melacak kebutuhan fungsi analistis menjadi *class-class*, operasi-operasi dan aktual sistem.

Ciri-ciri *use case* :

- Pola perilaku yang harus dipenuhi oleh sistem.
- Sekuen transaksi terhubung yang dilakukan aktor dan sistem.
- Memberikan sesuatu yang berharga (informasi) bagi *user*.

Kegunaan *use case* :

- Menangkap kebutuhan sistem.
- Berkomunikasi dengan pemakai akhir dan pakar domain masalah.
- Pengkajian sistem.

2. *Activity* Diagram

Dalam Buku Ratna Wardani (2012) menuliskan bahwa Diagram *activity* seperti diagram *state*, merupakan diagram yang dapat digunakan untuk memahami alur kerja dari objek/komponen yang dilakukan. Diagram *activity* dapat digunakan untuk memvisualisasikan interelasi dan interaksi antara *use case* yang berbeda, serta sering dipakai untuk mengasosiasikan dengan *class* yang berbeda. Kekuatan diagram *activity* adalah mempresentasikan *concurrent activity* (Thimoty C, 2002).

Diagram *Activity* menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi dan bagaimana mereka berakhir. Diagram *Activity* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi (Wahono, R.S, 2003).

Diagram *Activity* merupakan *state* diagram khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi dipicu oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu Diagram *Activity* 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.

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.

Diagram *Activity* dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.

3. *Class* Diagram

Dalam Buku Ratna Wardani (2012) menuliskan bahwa Diagram *class* merupakan diagram yang terdiri dari sekumpulan objek yang memiliki atribut-atribut dan *method*. Objek adalah sebuah benda/unit/sifat kerja yang memiliki atribut-atribut. Operasi adalah prosedural *abstraction* yang menspesifikasi tipe dari perilaku yang terdiri dari fungsi. *Super class* adalah *class* induk yang nantinya mempunyai *class-class* yang terdiri dari *class* dan *subclass* (Thimoty C, 2002).

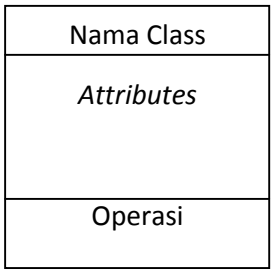
Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). Jadi, *Class* diagram menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi dan lain-lain (Wahono, R.S, 2003).

Diagram *class* terdiri dari :

a. Nama *class*

b. *Attributes*

c. Operasi



Gambar II.6. Tampilan *Class* Diagram

(Sumber : Dr. Ratna Wardani, S.Si., M.T. 2012)

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.

Elemen penting dalam diagram *class* adalah:

1. *Class*
2. Antar muka
3. Kolaborasi
4. Hubungan (*relationship*)

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 melewati dari satu *class* kepada *class* lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence* diagram yang akan dijelaskan kemudian.

Kegunaan diagram *class*:

1. Memodelkan kosakata sistem.
2. Memodelkan distribusi tanggung jawab dari sistem.
3. Memodelkan tipe.
4. Memodelkan entitas bukan perangkat lunak.
5. Memodelkan kolaborasi.
6. Memodelkan sistem basis data *logic*.

Kriteria diagram *class* yang baik:

1. Fokus
2. Essensial
3. Konsisten

4. Tidak hilang

Pentingnya UML untuk diagram *class*:

1. *Class*
2. *Association* (relasi antar class)
3. Atribut
4. *Operation*
5. *Generalization*

4. *Sequence* Diagram

Dalam Buku Ratna Wardani (2012) menuliskan bahwa Diagram *sequence* adalah diagram yang menunjukkan urutan dari pertukaran pesan antar objek dan tugas yang dilakukan dan merupakan diagram yang menjelaskan urutan kejadian dari sistem (urutan lacak kejadian) (Thimoty C, 2002).

Diagram *sequence* menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display* dan sebagainya) berupa *message* yang digambarkan terhadap waktu. Diagram *sequence* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait) (Wahono, R.S, 2003).

Diagram *sequence* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai *respons* dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang memicu 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

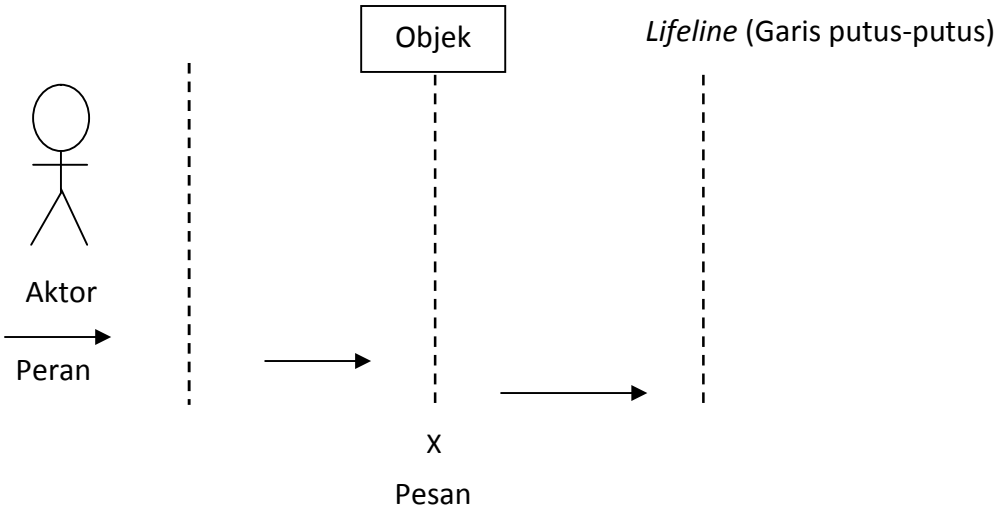
desain berikutnya, *message* akan dipetakan menjadi operasi/metoda dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message*.

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

Diagram *sequence* digunakan untuk:

1. *Overview* perilaku sistem.
2. Menunjukkan objek-objek yang diperlukan.
3. Mendokumentasikan skenario dari diagram *use case*.
4. Memberikan jalur-jalur pengaksesan.

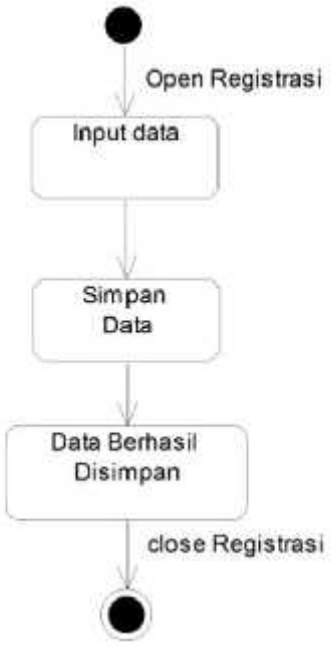
Notasi diagram *sequence*:



Gambar II.7. Tampilan *Sequence* Diagram
(Sumber : Dr. Ratna Wardani, S.Si., M.T. 2012)

5. Statechart Diagram

Menurut Gushelmi dan Deded (34:2012) Jika pada pemodelan interaksi menyiapkan detil spesifikasi dari *use case*, pada *statechart* akan diberikan detil deskripsi dari class yaitu perubahan *state* dari *class* menjadi lebih tepat. Perubahan dinamis inilah yang akan menjadi prilaku dari suatu objek. Biasanya *statechart* ini memodelkan aturan main suatu proses bisnis. *State* diagram khususnya digunakan untuk memodelkan taraf-taraf diskrit suatu siklus objek. *State* memodelkan objek dari semenjak dibuat sampai selesai.



Gambar II.8. Tampilan Statechart Diagram

(Sumber : Gushelmi dan Dede, 2012)

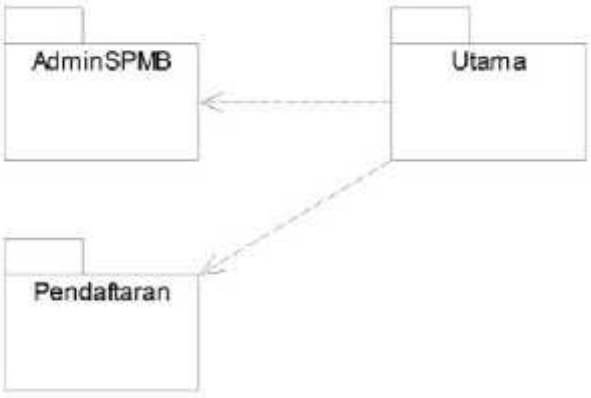
6. Package Diagram

Menurut Gushelmi dan Deded (37:2012) *Package* diagram bisa digunakan untuk mengelompokkan *use case* dan *class* diagram. Dari *use case* gambar 4.2

dapat dikelompokan ke dalam 3 (tiga) *package*. Dalam hal ini *use case* dikelompokan atas beberapa *package* diantaranya :

1. *Package* Utama *Package* ini terdiri dari *use case Login*, Melihat Persyaratan Pendaftar, Melihat Program Beasiswa, Melihat Program Studi, Melihat Prosedur Registrasi.
2. *Package* AdminSPMB *Package* ini terdiri dari *use case* Merekapitulasi Data Calon.
3. *Package* Pendaftaran *Package* ini terdiri dari *use case* Pengisian Formulir, Melihat Materi Test.

Berdasarkan pengelompokan *package* diatas maka dapat dilakukan pembuatan *package* diagram untuk *use case*. Hubungan antara *package* dari masing-masing *use case* bisa dilihat pada gambar berikut ini.



Gambar II.9. Tampilan Package Diagram dari Use Case

(Sumber : Gushelmi dan Dede, 2012)