BAB II

TINJAUAN PUSTAKA

II.1. Sistem Informasi

Menurut (Rachman,2012; 93), Sistem informasi adalah sekumpulan komponen-komponen yang saling berhubungan dan bekerja sama untuk mengumpulkan, memproses, menyimpan dan mendistribusikan informasi terkait untuk mendukung proses pengambilan keputusan, koordinasi, dan pengendalian.

II.2. Sistem Informasi Geografis

Menurut (Rachman,2012; 93), Sistem informasi geografis adalah sistem berbasis komputer yang digunakan untuk menyimpan dan memanipulasi informasi-informasi geografis. Sistem informasi geografis diciptakan untuk mengumpulkan, menyimpan dan menganalisis objek atau fenomena dimana lokasi geografis menjadi karakteristik atau kritik penting untuk analisi. Sistem informasi geografi adalah sistem berbasis komputer yang memiliki kemampuan dalam menangani data berefrensi dalam:

1. Masukkan data
2. Manajemen data
3. Manipulasi
4. Analisis

Pada awalnya data geografis hanya disajikan di atas peta dengan menggunakan simbol, garis, dan warna. Elemen-elemen geometri ini dideskripsikan di dalam legendanya misalnya, garis hitam tebal untuk jalan utama, garis hitam tipis untuk jalan

sekunder dan jalan-jalan yang berikutnya. Selain itu, berbagai data juga di dapat di-*overlay*-kan berdasarkan sistem koordinat yang sama. Akibatnya, sebuah peta menjadi media yang efektif baik sebagai alat presentasi maupun sebagai bank tempat penyimpanan data geografis. Tetapi, media peta masih mengandung kelemahan atau keterbatasan. Informasi-informasi yang tersimpan, diproses dan dipresentasikan dengan suatu cara tertentu, dan biasanya untuk tujuan tertentu pula. Tidak mudah untuk mengubah bentuk presentasi ini, sebuah peta selalu menyediakan gambar atau simbol unsur geografi dengan bentuk yang tetap atau statis meskipun diperlukan untuk kebutuhan yang berbeda.

II.3. Android

Menurut (Murtiwiyati dan Lauren**,** 2013; 2)**,** Android adalah sebuah sistem operasi untuk perangkat *mobile* berbasis linux yang mencakup sistem operasi, *middleware* dan aplikasi. Android menyediakan platform terbuka bagi para pengembang untuk menciptakan aplikasi mereka. Awalnya, *Google Inc.* membeli *Android Inc.* yang merupakan pendatang baru yang membuat piranti lunak untuk ponsel/*smartphone*. Kemudian untuk mengembangkan Android, dibentuklah *Open Handset Alliance,* konsorsium dari 34 perusahaan piranti keras, piranti lunak, dan telekomunikasi,termasuk *Google, HTC, Intel, Motorola, Qualcomm, T-Mobile,* dan *Nvidia*. Secara garis besar, arsitektur Android dapat dijelaskan dan digambarkan sebagai berikut:

a. *Applications* dan *Widgets Applications* adalah *layer* dimana berhubungan dengan aplikasi saja, dimana biasanya *download* aplikasi dijalankan kemudian dilakukan instalasi dan jalankan aplikasi tersebut.

b. Applications Frameworks Applications

frameworks ini adalah *layer* di mana para pembuat aplikasi melakukan pengembangan/pembuatan aplikasi yang akan dijalankan di sistem operasi Android, karena pada *layer* inilah aplikasi dapat dirancang dan dibuat, seperti *contect providers* yang berupa sms dan panggilan telepon.

c. Libraries

Libraries ini adalah *layer* di mana fitur-fitur Android berada, biasanya para pembuat aplikasi mengakses *libraries* untuk menjalankan aplikasinya. Berjalan diatas kernel, *Layer* ini meliputi berbagai *library* C/C++ inti seperti Libc dan SSL.

d. Android Run Time Layer

Android *Run Time Layer* yang membuat aplikasi Android dapat dijalankan dimana dalam prosesnya menggunakan Implementasi Linux.

e. Linux Kernel

Linux Kernel adalah layer dimana inti dari operating system dari Android itu berada. Berisi file - file systemyang mengatur sistem processing, memory, resource, drivers, dan sistem-sistem operasi android lainnya. Linux kernel yang digunakan android adalah linux kernel release 2.6.

II.3.1. Fitur dan Arsitektur Android

Menurut (Zulfariana dan Ernastuti, 2012; 231), Fitur yang tersedia pada Android adalah:

1. *Framework* Aplikasi : memungkinkan penggunaan dan pemindahan dari komponen yang tersedia.
2. *Dalvik Virtual Machine* : *virtual machine* yang dioptimalkan untuk perangkat *mobile*.
3. Grafik : grafik 2D dan grafik 3D yang didasarkan pada *Library OpenGL*.
4. *SQLite* : untuk menyimpan data.
5. Mendukung Media : *audio, video*, dan berbagai format gambar (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF).
6. *GSM, Bluetooth, Edge, 3G, WiFi, Camera, Global Positioning System (GPS), compass*, dan *accelerometer* (berdasarkan *hardware*).
7. Lingkungan pengembangan yang kaya, termasuk *emolator*, peralatan *debugging*, dan *plugin* untuk *Eclipse IDE*.

Sistem operasi Android dibangun berdasarkan kernel Linux, dan memiliki arsitekur sebagai berikut:

1. *Applications*

Lapisan ini adalah lapisan aplikasi, serangkaian aplikasi akan terdapat pada perangkat *mobile*. Aplikasi inti yang telah terdapat pada Android termasuk kalender, kontak, SMS (*Short Message Service*), dan lain sebagainya. Aplikasi-aplikasi ini ditulis dengan bahasa pemrograman *Java*.

2. *Application Framework*

Pengembang aplikasi memiliki akses penuh ke Android sama dengan aplikasi inti yang telah tersedia. Pengembang dapat mudah mengakses informasi lokasi, mengatur alarm, menambah pemberitahuan ke status bar dan lainnya sebagainya. Arsitektur aplikasi ini dirancang untuk menyederhanakan penggunaan kembali komponen, aplikasi apapun yang dapat mempublikasikan kemampuan dan aplikasi lainnya dapat menggunakan kemampuan mereka sesuai batasan keamanan. Dasar dari aplikasi adalah seperangkat layanan sistem, yaitu berbagai *view* yang digunakan untuk membangun *user interface*, *content provider* yang memungkinkan aplikasi berbagi data, *Resource Manager* menyediakan akses bukan kode seperti grafik, *string*, dan *layout, Notification Manager* yang akan membuat aplikasi dapat menampilkan tanda pada status bar dan *Activity Manager* yang berguna mengatur daur hidup dari aplikasi.

3. *Libraries*

Satu set libraries dalam bahasa C/C++ yang digunakan oleh berbagai komponen pada sistem multimedia.

4. Android *Runtime*

Satu set *libraries* inti yang menyediakan sebagian besar fungsi yang tersedia di *libraries* inti dari bahasa pemrograman Java. Setiap aplikasi akan berjalan sebagai proses sendiri pada *Dalvik Virtual Machine*.

5. Linux Kernel

Android bergantung pada Linux versi 2.6 untuk layanan sistem initi seperti keamanan, manajemen memori, manajemen proses, *network stack*, dan model *driver*. Kernel juga bertindak sebagai lapisan antara *hardware* dan seluruh *software*.

II.3.2 Jenis-Jenis Versi Android

Menurut (Masruri, 2014; 4), ada beberapa jenis-jenis android yaitu :

1. Android versi 1.1

Andoid versi 1.1 di rilis pada 9 Maret 2009 oleh *Google*. Android versi ini dilengkapi disupport oleh *Google Mail Service* dengan pembaruan estetis pada aplikasi, jam alarm, *voice search* (pencarian suara), pengiriman pesan dengan *Gmail*, dan pemberitahuan *Email*.

2. Android versi 1.5 *Cup Cake*

Android *Cup Cake* di rilis pada pertengahan Mei 2009, masih oleh *Google* Inc. Android ini dilengkapi *software development kit* dengan berbagai pembaharuan termasuk penambahan beberapa fitur antara lain yakni kemampuan merekam dan menonton video dengan modus kamera, mengunggah video ke *Youtube*, *upload* gambar ke Picasa langsung dari telepon, serta mendapat dukungan *Bluetooth A2DP*.

3. Android versi 1.6 *Donut*

Android *Donut* di rilis pada September 2009 menampilkan proses pencarian yang lebih baik dibandingkan versi-versi sebelumnya. Selain itu Android *Donut* memiliki fitur-fitur tambahan seperti galeri yang memungkinkan pengguna untuk memilih foto yang akan dihapus; kamera, *camcorder* dan galeri yang dintegrasikan;

Text-to-speech engine; kemampuan dial kontak; teknologi *text to change speech*. Android *Donut* juga dilengkapi baterai indikator, dan kontrol *applet VPN*.

4. Android versi 2.0/2.1 *Eclair*

Android *Eclair* dirilis pada 3 Desember 2009. Perubahan yang ada antara lain adalah pengoptimalan *hardware*, peningkatan *Google Maps* 3.1.2, perubahan UI dengan *browser* baru dan dukungan HTML5, daftar kontak yang baru, dukungan *flash* untuk kamera 3,2 MP, *digital Zoom*, dan *Bluetooth* 2.1. Android *Eclair* merupakan Adroid pertama yang mulai dipakai oleh banyak *smartphone*, fitur utama *Eclair* yaitu perubahan total struktur dan tampilan *user interface.*

5. Android versi 2.2 *Froyo* (*Frozen Yogurt*)

Android *Froyo* dirilis pada 20 mei 2012. Adroid versi ini memiliki kecepatan kinerja dan aplikasi 2 sampai 5 kali dari versi-versi sebelumnya. Selain itu ada penambahan fitur-fitur baru seperti dukungan *Adobe Flash* 10.1, intergrasi V8 *JavaScript engine* yang dipakai *Google Chrome* yang mempercepat kemampuan *rendering* pada *browser*, pemasangan aplikasi dalam *SD Card*, kemampuan *WiFi Hotspot portabel*, dan kemampuan *auto update* dalam aplikasi Android *Market*.

6. Android versi 2.3 *Gingerbread*

Android *Gingerbread* di rilis pada 6 Desember 2010. 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.

7. Android versi 3.0/3.1 *Honeycomb*

Android *Honeycomb* di rilis pada awal 2012. Merupakan versi Android yang dirancang khusus untuk *device* dengan layar besar seperti *Tablet PC*. Fitur baru yang ada pada Android *Honeycomb* antara lain yaitu dukungan terhadap *prosessor multicore* dan grafis dengan *hardware acceleration*. *User Interface* pada *Honeycomb* juga berbeda karena sudah didesain untuk tablet. Tablet pertama yang memakai *Honeycomb* adalah tablet Motorola *Xoom* yang dirilis bulan Februari 2011. Selain itu sebuah perangkat keras produksi Asus bernama *Eee Pad Transformer* juga menggunakan OS Android *honeycomb* dan diharapkan akan masuk ke pasaran Indonesia pada Mei 2011.

8. Android versi 4.0 *ICS (Ice Cream Sandwich*)

Android *Ice Cream Sandwich* diumumkan secara resmi pada 10 Mei 2011 di ajang *Google I/O Developer Conference* (San Francisco), pihak *Google* mengklaim Android *Ice Cream Sandwich* akan dapat digunakan baik di *smartphone* ataupun tablet. Android *Ice Cream Sandwich* membawa fitur *Honeycomb* untuk *smartphone* serta ada penambahan fitur baru seperti membuka kunci dengan pengenalan wajah, jaringan data pemantauan penggunaan dan kontrol, terpadu kontak jaringan sosial, perangkat tambahan fotografi, mencari *email* secara *offline*, dan berbagi informasi dengan menggunakan NFC. Ponsel pertama yang menggunakan sistem operasi ini adalah *Samsung Galaxy Nexus*.

9. Android versi 4.1 *Jelly Bean*

Android *Jelly Bean* juga diluncurkan pada acara *Google* I/O 10 Mei 2011 yang lalu. Android versi ini membawa sejumlah keunggulan dan fitur baru, diantaranya

peningkatkan *input keyboard*, desain baru fitur pencarian, UI yang baru dan pencarian melalui *Voice Search* yang lebih cepat. Versi ini juga dilengkapi *Google Now* yang dapat memberikan informasi yang tepat pada waktu yang tepat pula. Salah satu kemampuannya adalah dapat mengetahui informasi cuaca, lalu-lintas, ataupun hasil pertandingan olahraga. Sistem operasi Android *Jelly Bean* 4.1 pertama kali digunakan dalam produk tablet Asus, yakni *Google Nexus 7.*

10.Android versi 4.2 *Jelly Bean*

Fitur *photo sphere* untuk panaroma, *daydream* sebagai *screensaver, power control, lock screen widget*, menjalankan banyak *user* (dalam tablet saja), *widget* terbaru. Android 4.2 Pertama kali dikenalkan melalui *LG Google Nexus 4.*

II.4. *Google Map API*

Menurut (Tim EMS, 2014; 15), Aplikasi Android yang dibuat dapat diintegrasikan dengan *Google API*, misalnya seperti pembuatan aplikasi yang digunakan untuk mendeteksi lokasi seorang pengguna. Anda akan memerlukan waktu berjam-jam atau beberapa hari denganratusan/ribuan kode untuk membuat sebuah *mapping systems.*

Menurut (Elian, dkk ; 2012 : 1), *Google Maps* adalah layanan pemetaan berbasis *web service* yang disediakan oleh *Google* dan bersifat gratis, yang memiliki kemampuan terhadap banyak layanan pemetaan berbasis *web*. *Google Maps* juga memiliki sifat *server side*, yaitu peta yang tersimpan pada server *Google* dapat dimanfaatkan oleh pengguna. *Google Maps API* adalah suatu *library* yang berbentuk *JavaScript* yang berguna untuk memodifikasi peta yang ada di *Google Maps* sesuai

kebutuhan. Untuk membangun aplikasi yang memanfaatkan *Google Maps* di desktop dan *mobile device* maka akan digunakan *Google Maps JavaScript API* v3 yang memiliki keunggulan lebih cepat dari versi sebelumnya.

II.5. Dasar Pemrogrman Android

Menurut (Tim EMS, 2014; 9), Dasar pemrograman android adalah java, karena aplikasi Android ditulis dalam bahasa java. Android menyediakan lingkungan atau *run time environment* yang dikenal sebagai *Dalvik Virtual Machine.* Sehingga *Dalvik Virtual Machi*ne ini merupakan java *run time environment* yang telah dioptimasi untuk *device* dengan sistem memori yang kecil. Walaupun begiru, tidak semua murni menggunakan java, namun masih menggunkan bahasa XML dan dasar *Apache Ant* untuk pengembangan aplikasi.

Bahasa XML *(Extensible Markup Language)* merupakan bahsa web turunan dari SGML *(Standard Generalized Markup Language)* yang ada sebelumnya. XML hampir sama dengan HTML, dimanakedua-duanya diturunkan dari SGML.

Secara sederhan XML adalah suatu bahasa yang digunakan untuk mendeskripsikan dan memanipulasi dokumen secara terstruktur. Secara teknis, XML didefenisikan sebagai suatu bahasa *meta-markup* yang menyediakan format tertentu untuk dokumen-dokumen yang mempunyai data terstruktrur. Bahasa *markup* adalah mekanisme untuk mengenal suatu struktur di dokumen.

Sedangkan *Apache Ant* merupakan software berbasis java yang digunakan untuk keperluan buil tool. Sebagai build tool, *Apache Ant* akan menyediakan sumber daya dan

melaksanakan proses yang memungkinkan membangun suatu software dari bentuk *source code* menjadi aplikasi yang siap didistribusikan.

II.5.2. Pemrograman Java

Menurut (Wardhani dan Yaqin, 2013; 14), Java adalah sebuah bahasa pemrograman yang populer dikalangan para akademisi dan praktisi komputer. Java pertama kali dikembangkan untuk memenuhi kebutuhan akan sebuah bahasa komputer yang ditulis satu kali dan dapat dijalankan dibanyak system komputer berbeda tanpa perubahan kode berarti. Pada umumnya, para pakar pemrograman berpendapat bahwa bahasa Java memiliki konsep yang konsisten dengan teori pemrograman objek dan aman untuk

digunakan. Java sampai saat ini masih merupakan bahasa pemrograman yang masih sangat di minati dan banyak digunakan oleh para progremer dan software developer untuk mengembangkan berbagai tipe aplikasi, mulai dari aplikasi console, aplikasi desktop, game, dan applet (aplikasi yang berjalan di lingkungan *web browser*), sampai ke aplikasi-aplikasi yang berskala *enterprise*. Untuk memenuhi kebutuhan tipe aplikasi yang beragam tersebut, Java dikategorikan menjadi tiga edisi, yaitu: J2SE (Java 2 Platform Standart Edition) untuk membuat aplikasi-aplikasi desktop dan apllet, J2EE (Java 2 *Platform Enterprise Edition*) untuk membuat aplikasi-aplikasi multitier berskala enterprise, dan J2ME (Java 2 *Platform Micro Edition*) untuk membuat aplikasi-aplikasi yang dapat dijalankan dilingkungan perangkat-perangkat mikro seperti *handphone*, PDA dan *Smartphone*.

II.5.2. Karakteristik-karakjteristik Java

Menurut (Nyura, 2010; 19), Ada beberapa karakteritik java yaitu :

a. Sederhana

Bahasa pemrograman Java menggunakan sintaks yang mirip dengan bahasa C++ namun sintaks pada Java telah banyak diperbaiki, terutama dengan menghilangkan *pointer* yang rumit dan *multiple inheritance*. Java juga menggunakan automatic memory allocation dan garbage collection.

b. Berorientasi Objek

Java merupakan bahasa pemrograman berorientasi objek yang memungkinkan program untuk dibuat secara modular dan digunakan kembali.

c. Terdistribusi

Java dibuat untuk memudahkan distribusi aplikasi dengan adanya networking libraries yang terintegrasi dalam Java.

d. *Interpreted*

Program Java dijalankan menggunakan program Interpreter, yaitu Java Virtual Machine (JVM). Hal ini menyebabkan source code Java yang telah dikompilasi menjadi bytecodes dapat dijalankan pada berbagai platform.

e. *Robust*

Java mempunyai reliabilitas yang tinggi. Kompiler pada Java mempunyai kemampuan mendeteksi error yang lebih baik dibandingkan bahasa pemrograman yang lain. Java mempunyai Runtime Exception Handling untuk membantu mengatasi error pada pemrograman.

f. *Secure*

Sebagai bahasa pemrograman aplikasi internet dan terdistribusi, Java memiliki beberapa mekanisme keamanan untuk menjaga agar aplikasi tidak digunakan untuk merusak sistem komputer yang menjalankan aplikasi tersebut.

g. *Architecture Neutral*

Program Java tidak bergantung pada platform dimana program akan dijalankan. Cukup dibuat satu program yang dapat dijalankan pada berbagai platform dengan Java Virtual Machine.

h. *Portable*

Source code maupun program Java dapat dengan mudah dibawa ke berbagai platform berbeda tanpa harus dikompilasi ulang.

i. *Performance*

Kinerja Java sering kali dikatakan kurang, namun kinerja Java dapat ditingkatkan menggunakan compiler Java lain seperti buatan Inprise, Microsoft maupun Symantec yang menggunakan Just In Time Compilers (JIT).

j. *Multithreaded*

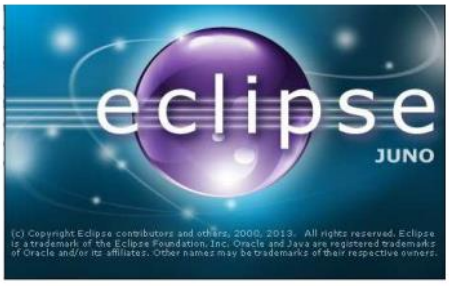
Java dapat membuat suatu program yang mampu melakukan beberapa pekerjaan secara sekaligus dan simultan.

k. *Dynamic*

Java dapat didesain untuk dapat dijalankan pada lingkungan yang dinamis. Perubahan suatu class dengan menambahkan properties ataupun metode dapat dilakukan tanpa menggangu program yang menggunakan class tersebut.

II.6. *Eclipse* IDE

Menurut (Lengkong, dkk, 2015; 21**),** *Eclipse* IDE adalah sebuah IDE *(Integrated Development Environment)* untuk mengembangkan perangkat lunak dan dapat dijalankan semua *platform (platform independent).Eclipse* pada saat ini merupakan salah satu IDE favorit dikarenakan gratis dan *open source* , yang berarti setiap orang boleh melihat kode pemrograman perangkat lunak ini. Selain itu, kelebihan dari *Eclipse* yang membuat popular adalah kemampuannya untuk dapat dikembangkan oleh pengguna dengan komponen yang dinamakan *plug-in.*



Gambar II.1. Logo Eclipse Juno

II.7. Konsep UML (*Unified Modelling Language*)

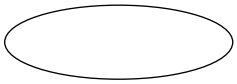
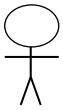
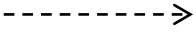
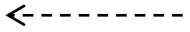
Menurut (Sulistyorini, 2009; 1), *Unified Modelling Language* (UML) adalah sebuah "bahasa" yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML dapat dibuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan class dan operation dalam konsep dasarnya, maka lebih cocok untuk penulisan piranti lunak dalam bahasa berorientasi objek seperti C++, Java, atau VB. NET.

II.7.1. Diagram – diagram UML

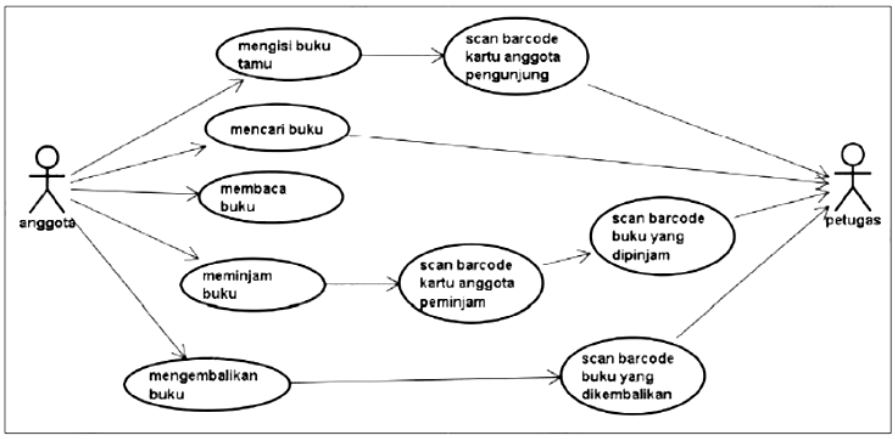
1. Use Case Diagram

Menurut (Sulistyorini, 2009; 2), Diagram ini bersifat statis. Diagram ini memperlihatkan himpunan use case dan aktor-aktor (suatu jenis khusus dari kelas).Diagram ini terutama sangat penting untuk mengorganisasi dan memodelkan perilaku dari suatu sistem yang dibutuhkan serta diharapkan pengguna.

Tabel II.1. Simbol *Use Case*

Gambar	Keterangan
	*Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukan pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama *use case*.
	Aktor adalah *abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan aktor, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa muncul dalam beberapa peran. Perlu dicatat bahwa aktor berinteraksi dengan *use case*, tetapi tidak memiliki control terhadap *use case*.
	Asosiasi antara aktor dan *use case*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengidikasikan aliran data.
	Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengidinkasikan bila aktor berinteraksi secara pasif dengan sistem.
	*Include,* merupakan di dalam *use case* lain (*required*) atau pemanggilan *use case* oleh *use case* lain, contohnya adalah pemanggilan sebuah fungsi program.
	*Extend*, merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.

Sumber : (Andry Setiawan, dkk, 2013 ; 4).



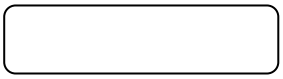
Gambar. II.2. *Use Case Diagaram*

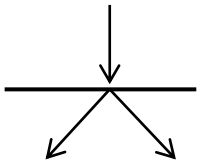
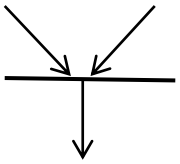
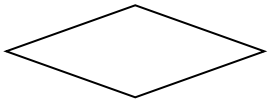
Sumber : (Andry Setiawan, dkk, 2013 ; 4).

2. Activity Diagram

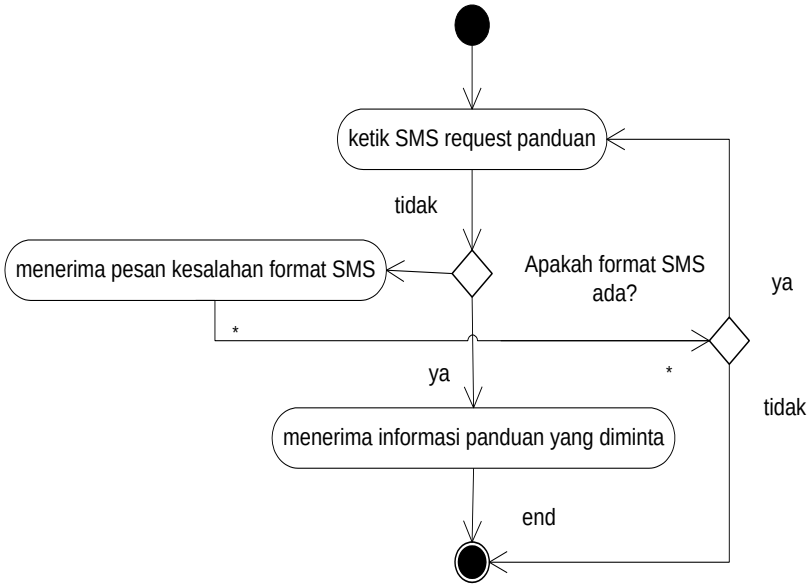
Menurut (Sulistyorini, 2009; 2), Diagram ini bersifat dinamis. Diagram ini adalah tipe khusus dari diagram *state* yang memperlihatkan aliran dari suatu aktifitas ke aktifitas lainnya dari suatu sistem. Diagram ini terutama penting dalam pemodelan fungsi – fungsi dalam suatu sistem dan memberi tekanan pada aliran kendali antar objek.

Tabel II.2. Simbol *Activity Diagram*

Gambar	Keterangan
	*Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	*End point*, akhir aktifitas.
	*Activites*, menggambarkan suatu proses/kegiatan bisnis.

	*Fork* (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan pararel menjadi satu.
	*Join* (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi.
	*Decision Points*, menggambarkan pilihan untuk pengambilan keputusan, *true, false.*
New Swimline	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa.

Sumber : (Andry Setiawan, dkk, 2013 ; 5)



Gambar. II.3. *Activity Diagaram*

Sumber : (Yani Rahardja, dkk, 2008; 5)

3. Class Diagram

Menurut (Sulistyorini, 2009; 4), Diagram kelas bersifat statis. Diagram ini memperlihatkan himpunan kelas-kelas, antarmuka-antarmuka, kolaborasi-kolaborasi serta relasi.

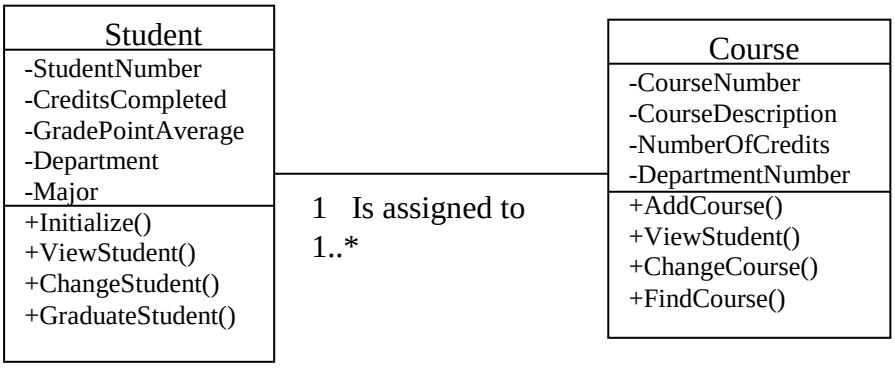
Berbagai simbol yang hadir didalam class diagram antara lain adalah:

a. *Class*, yang berfungsi untuk merepresentasikan tipe dari data yang dimilikinya. *Class* diagram dapat ditampilkan dengan menunjukkan atribut dan operasi yang dimilikinya atau hanya menunjukkan nama *class*-nya saja. Dapat juga kita tuliskan nama *class* dengan atributnya saja atau nama *class* dengan operasinya.
b. *Attribute*, merupakan data yang terdapat didalam *class* dan *instance*-nya dengan operator.
c. *Operation*, berfungsi untuk merepresentasikan fungsi-fungsi yang ditampilkan oleh class dan *instance*-nya dengan operator.
d. *Association*, digunakan untuk menunjukkan bagaimana dua *class* berhubungan satu sama lainnya. *Association* ditunjukkan dengan sebuah garis yang terletak diantara dua *class*. Didalam setiap *association* terdapat *multiplicity*, yaitu simbol yang mengindikasikan berapa banyak *instance* dari *class* pada ujung *association* yang satu dengan *instance class* di ujung *association* lainnya.
e. *Generalizations*, berfungsi untuk mengelompokkan *class* ke dalam hirarki *inheritance*.
f. *Aggregation*, merupakan bentuk khusus dari *association* yang merepresentasikan hubungan "*part-whole*". Bagian "*whole*" dari hubungan ini

sering disebut dengan *assembly* atau *aggregate*. yang satu dapat dikatakan merupakan bagian dari *class* yang lain yang ikut membentuk *class* tersebut.

g. *Composition*, merupakan jenis *aggregation* yang lebih kuat diantara dua *class* yang memiliki *association* dimana jika *whole* ditiadakan, maka partnya juga ikut ditiadakan. Berbeda dengan *aggregation part* akan tetap bisa berdiri sendiri meskipun bagian *whole*-nya ditiadakan.

h. Penggunaan operator (+) dalam *class* diagram diartikan dengan *public,* operator (-) diartikan *private*, dan operator (#) diartikan *protected*.



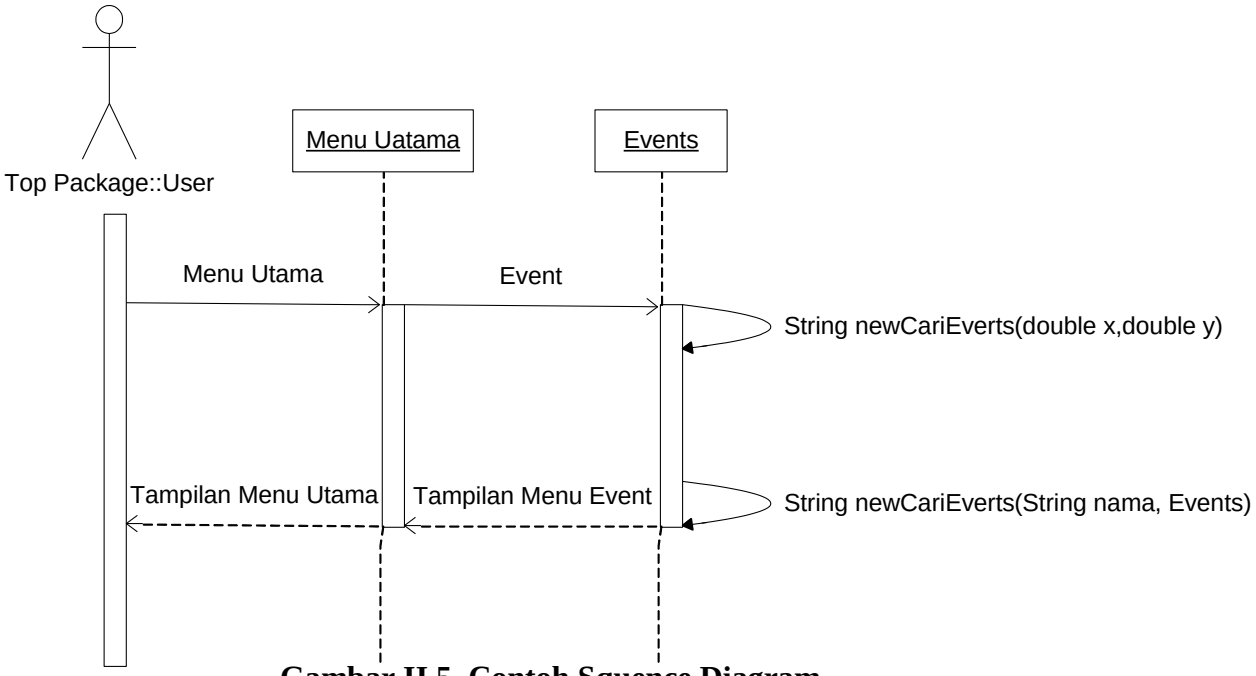
Gambar II.4.*Class Diagram*

Sumber :(Edgar Winata dan Johan Setiawan, 2013; 5)

4. Squence diagram

Menurut (Haviluddin, 2011; 5).Diagram sequence menjelaskan interaksi objek yang disusun dalam suatu urutan waktu. Diagram ini secara khusus berasosiasi dengan *use case*. Sequence diagram memperlihatkan tahap demi tahap apa yang

seharusnya terjadi untuk menghasilkan sesuatu didalam *use case*. Diagram sequence sebaiknya digunakan diawal tahap desain atau analisis karena kesederhanaannya dan mudah untuk dimengerti. Sequence diagram menjelaskan interaksi objek yang disusun berdasarkan urutan waktu. Secara mudahnya sequence diagram adalah gambaran tahap demi tahap, termasuk kronologi (urutan) perubahan secara logis yang seharusnya dilakukan untuk menghasilkan sesuatu sesuai dengan usecase diagram



Gambar II.5. Contoh Squence Diagram

Sumber : (S. Nofan Maulana Rachman, 2012; 9)