BAB II

TINJAUAN PUSTAKA

II.1. Mengenal Android

Android adalah sebuah sistem operasi mobile yang berbasiskan pada versi modifikasi dari Linux. Pertama kali system operasi ini dikembangkan oleh perusahaan Android.Inc. Nama perusahaan inilah yang pada akhirnya digunakan sebagai nama proyek system operasi mobile tersebut, yaitu system operasi Android.

Pada tahun 2005, sebagian dari strategi unutk memasuki pasar mobile, Google membeli Android dan mengambil alih proses pengembangannya sekaligus team developer Android. Google menginginkan Android untuk menjadi sistem operasi *Open Source* dan gratis, kebanyakan code Android dirilis dibawah lisensi *Open Source Apache* yang berarti setiap orang bebas untuk menggunakan dan mengunduh *source code* Android secara penuh.

Android telah dikembangkan dan diupdate beberapa kali sejak rilis pertamanya. Table dibawah ini memperlihatkan versi Android semenjak pertama kali dirilis (*Wahana Komputer : 2013: 2 -3*)

Tabel II.1. Versi Android
(sumber : Wahana Komputer: 2013: 3)

Versi Android	Tanggal Rilis	Nama Kode
1.1	9 Februari 2009	-
1.5	30 April 2009	Cupcake
1.6	15 September 2009	Donut

2.0 / 2.1	26 Oktober 2009	Éclair
2.2	20 Mei 2010	Froyo
2.3	6 Desember 2010	Gingerbread
3.0	Tidak Diketahui	Honeycomb
4.0	19 Oktober 2011	Ice Cream Sandwich

II.1.1. Aplikasi *Native*

Aplikasi *native* adalah aplikasi yang secara khusus ditujukan untuk platform mobile tertentu dan menggunakan bahasa pemrograman serta perangkat lunak pengembangan sesuai dengan *platform* tersebut. Sebagai contoh , aplikasi native Android ditulis menggunakan bahasa pemrograman *Java* dan *Tool Eclipse*, sementara aplikasi iOS/iPhone dibuat dengan menggunakan bahasa *Objective-C* dan *tool Xcode.*

Kelebihan:

1. Perfoma yang sangat baik karena ditulis secara native untuk platform spesifik.
2. Mampu mengakses semua fitur perangkat keras smartphone, seperti *info device, accelerator,* kamera, kompas, file dan lain sebagainya.
3. Menghasilkan antarmuka *look and feel* yang alami dengan sangat baik.

Kekurangan:

1. Pengembangan yang tidak mudah karena menggunakan lingkungan, bahasa dan API (*Application Programming Interface*) spesifik.
2. Aplikasi hanya berjalan pada platform yang sudah dispesifikasikan diawal pengembangan. Apabila ingin dikembangkan diplatform lain maka harus

ditulis ulang dengan tool pengembangan yang sesuai. .(*sumber : Didik Dwi Prasetsya : 2013 : 2*)

II.1.2. Tool Pengembangan Android

Untuk mengembangkan sistem operasi Android dapat menggunkan Mac, Windows atau PC. Linux, semua tool yang dibutuhkan adalah gratis dan dapat di download dari web.

a. Java JDK : Andorid berjalan dengan menggunakan resource dari Java SE Development Kit (JDK).
b. Android SDK : Android SDK berikan Debugger, library, dokumentasi, kode contoh dan tutorial. Andorid SDK dapat didownload dari alamat : http://developer.andorid/sdk/index.html .
c. Andorid Development Tools (ADT) : Plug-in Android Development Tools (ADT) untuk mendukung pembuatan dan ;proses debugging dari aplikasi Android yang sedang buat.

II. 2. Eclipse IDE

Integrated Development Environment (IDE) adalah program komputer yang memiliki beberapa fasilitas yang diperlukan dalam pembangunan perangkat lunak. Tujuan dari IDE adalah untuk menyediakan semua utilitas yang diperlukan dalam membangun perangkat lunak.

Eclipse adalah sebuah IDE untuk mengembangkan perangkat lunak dan dapat dijalankan di semua platform (*platform-independent*). Berikut ini adalah sifat dari Eclipse:

a. *Multi-platform*: Target sistem operasi Eclipse adalah Microsoft Windows, Linux, Solaris, AIX, HP-UX dan Mac OS X.
b. *Multi-language*: Eclipse dikembangkan dengan bahasa pemrograman Java, akan tetapi Eclipse mendukung pengembangan aplikasi berbasis bahasa pemrograman lainnya, seperti C/C++, JavaScript, Cobol, Python, Perl, PHP, dan lain sebagainya.
c. *Multi-role*: Selain sebagai IDE untuk pengembangan aplikasi, Eclipse pun bisa digunakan untuk aktivitas dalam siklus pengembangan perangkat lunak, seperti dokumentasi, test perangkat lunak, pengembangan web, dan lain sebagainya.

Eclipse saat ini merupakan salah satu IDE favorit dikarenakan gratis dan *open source*, yang berarti setiap orang dapat melihat dan memodifikasi *sourcecode software* ini. Selain itu, kelebihan Eclipse yang membuatnya populer adalah kemampuannya untuk dapat dikembangkan oleh pengguna dengan komponen yang dinamakan *plug-in*. (Nazruddin Safaat, 2012)

II. 3. Media Pembelajaran

Media berasal dari kata latin, merupakan bentuk jamak dari kata medium, secara harfiah kata-kata tersebut mempunyai arti perantara atau pengantar. Akan tetapi sekarang kata tersebut digunakan baik untuk bentuk jamak maupun mufrad. Kemudian telah banyak pakar dan juga organisasi yang memberikan batasan mengenai pengertian media, beberapa diantarannya mengemukakan bahwa media adalah sebagai berikut :

1. Teknolgi pembanwa pesan yang dapat dimanfaatkan untuk keperluan pembelajaran, jadi media adalah perluasan dari guru.
2. Sarana dalam bentuk cetak maupun audio visual, termasuk teknologi perangkat kerasnya.
3. Alat untuk memberikan perangsang bagi siswa supaya terjadi proses belajar.
4. Segala bentuk saluran yang dipergunakan untuk proses penyaluran pesan.
5. Berbagai jenis komponen dalam lingkungan siswa yang dapat merangsang siswa untuk belajar.
6. Segala sesuatu yang dapat digunakan untuk menyalurkan pesan yang dapat merangsang pikiran, perasaan, perhatian dan kemauan siswa untuk belajar.

Dari berbagai pendapat diatas dapat ditarik kesimpulan bahwa (a) media pembelajaran adalah merupakan wadah dari pesan. (b) Materi yang ingin disampaikan adalah proses pembelajaran. (c) tujuan yang ingin dicapai ialah proses pembelajaran. Selanjutnya penggunaan media secara kreatif akan memperbesar kemungkinan bagi siswa untuk belajar lebih banyak, mencamkan apa yang dipelajarinya lebih baik dan meningkatkan penampilan dalam melakukan keterampilan sesuai demham yang menjadi tujuan pembelajaran. (*Tim Pengembang Ilmu Pendidikan FIP-UPI, 2007; 205 -206*)

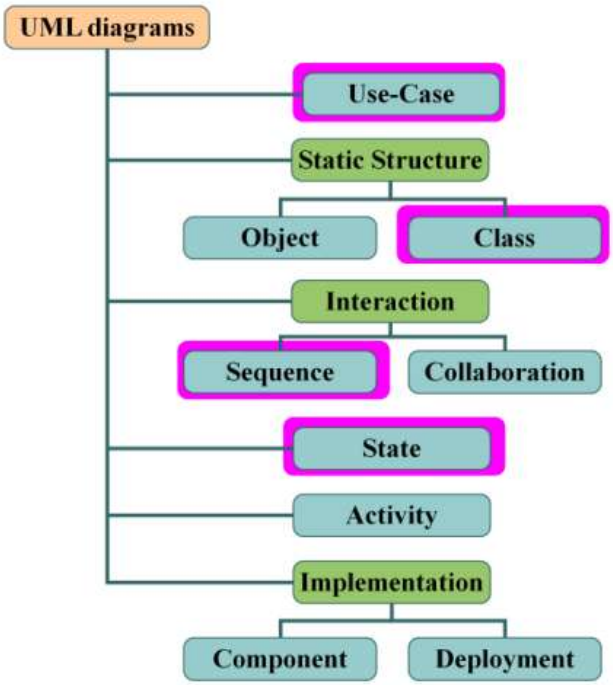
II.4. UML *(Unified Modelling Language)*

Unified Modelling Language merupakan alat perancangan sistem yang berorientasi pada objek. Secara filosofi kemunculan UML diilhami oleh konsep yang telah ada yaitu konsep permodelan *Object Oriented* (OO), karena konsep ini menganalogikan sistem seperti kehidupan nyata yang didominasi oleh obyek dan digambarkan atau dinotasikan dalam simbol-simbol yang cukup spesifik maka OO memiliki proses standard dan bersifat independen.

UML diagram memiliki tujuan utama untuk membantu tim pengembangan proyek berkomunikasi, mengeksplorasi potensi desain, dan memvalidasi desain arsitektur perangkat lunak atau pembuat program. Komponen atau notasi UML diturunkan dari 3 (tiga) notasi yang telah ada sebelumnya yaitu Grady Booch, OOD (Object-Oriented Design), Jim Rumbaugh, OMT (Object Modelling Technique), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*).

UML mempunyai tiga kategori utama yaitu *struktur diagram, behaviour diagram dan interaction diagram*. Dimana masing-masing kategori tersebut memiliki diagram yang menjelaskan arsitektur sistem dan saling terintegrasi. (*Haviluddin , 2011 ; 1*)

Menurut Haviluddin (*2011*) Secara filosofi UML diilhami oleh konsep yang telah ada yaitu konsep permodelan *Object Oriented* karena konsep ini menganalogikan sistem seperti kehidupan nyata yang didominasi oleh obyek dan digambarkan atau dinotasikan dalam simbol-simbol yang cukup spesifik. Berikut gambar dari diagram UML



Gambar II. 1. Diagram UML
(*Haviluddin , 2011 ; 2*)

Komponen-komponen UML

Sejauh ini para pakar merasa lebih mudah dalam menganalisa dan mendesain atau memodelkan suatu sistem karena UML memiliki seperangkat aturan dan notasi dalam bentuk grafis yang cukup spesifik (Sugrue J. 2009).

Komponen atau notasi UML diturunkan dari 3 (tiga) notasi yang telah ada sebelumnya yaitu Grady Booch, OOD (*Object-Oriented Design*), Jim Rumbaugh, OMT (*Object Modelling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*). (*Haviluddin, 2011 ; 3*)

Pada UML versi 2 terdiri atas tiga kategori dan memiliki 13 jenis diagram yaitu :

A. Struktur Diagram

Menggambarkan elemen dari spesifikasi dimulai dengan kelas, obyek, dan hubungan mereka, dan beralih ke dokumen arsitektur logis dari suatu sistem. Struktur diagram dalamUML terdiri atas :

1. *Class Diagram*

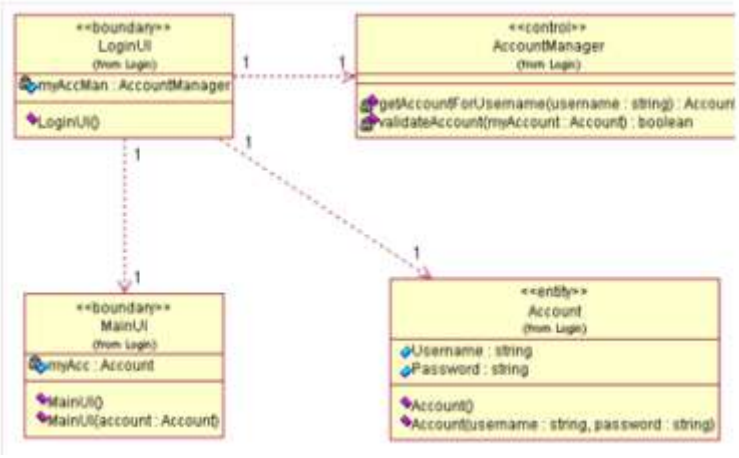
   Class diagram menggambarkan struktur statis dari kelas dalam sistem anda dan menggambarkan atribut, operasi dan hubungan antara kelas. *Class diagram* membantu dalam memvisualisasikan struktur kelas-kelas dari suatu sistem dan merupakan tipe diagram yang paling banyak dipakai. Selama tahap desain, class diagram berperan dalam menangkap struktur dari semua kelas yang membentuk arsitektur sistem yang dibuat.

   Class memiliki tiga area pokok :

   a. Nama (dan stereotype)

   b. Atribut

   c. Metoda

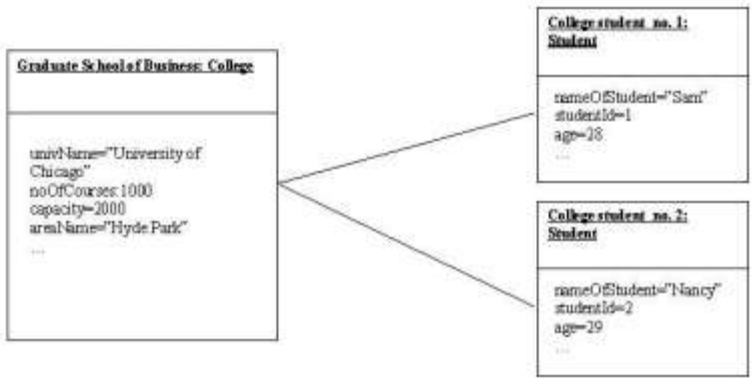


Gambar.II.2. Contoh Notasi Class Diagram
(*Sumber : Haviluddin , 2011 ; 3*)

2. *Object diagram*

   Object diagram menggambarkan kejelasan kelas dan warisan dan kadang-kadang diambil ketika merencanakan kelas, atau untuk membantu pemangku kepentingan non-program yang mungkin menemukan diagram kelas terlalu abstrak.

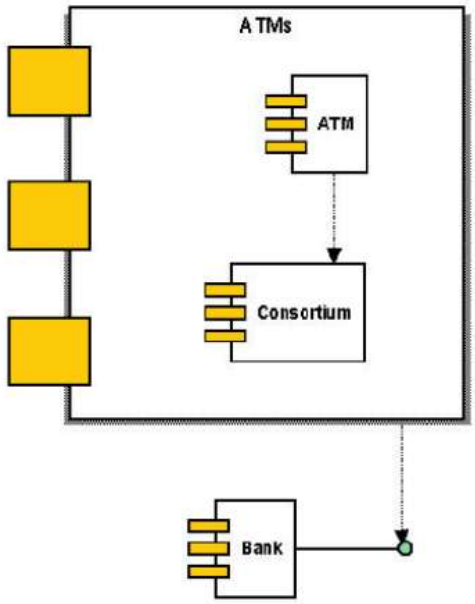
   Berikut notasi *object diagram :*



Gambar.II.3. Contoh Notasi *Object Diagram*
(*Sumber : Haviluddin , 2011 ; 3*)

3. *Component diagram*

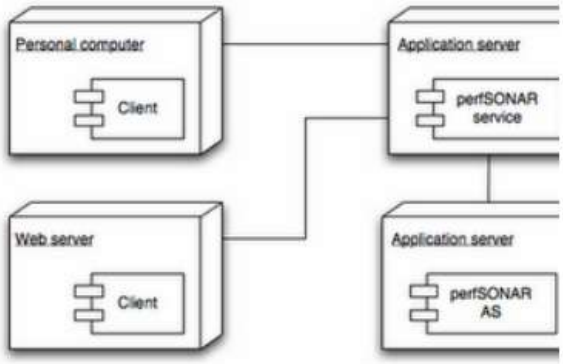
   Component diagram menggambarkan struktur fisik dari kode, pemetaan pandangan logis dari kelas proyek untuk kode aktual di mana logika ini dilaksanakan.



Gambar.II.4. Contoh Notasi *Component Diagram*
(*Sumber : Haviluddin , 2011 ; 3*)

4. *Deployment diagram*

   Deployment diagram memberikan gambaran dari arsitektur fisik perangkat lunak, perangkat keras, dan artefak dari sistem. *Deployment diagram* dapat dianggap sebagai ujung spektrum dari kasus penggunaan, menggambarkan bentuk fisik dari sistem yang bertentangan dengan gambar konseptual dari pengguna dan perangkat berinteraksi dengan sistem.



Gambar.II.5. Contoh Notasi *Deployment Diagram*
(*Sumber : Haviluddin , 2011 ; 4*)

5. *Composite structure diagram*

   Sebuah diagram struktur komposit mirip dengan diagram kelas, tetapi menggambarkan bagian individu, bukan seluruh kelas. Kita dapat menambahkan konektor untuk menghubungkan dua atau lebih bagian dalam atau ketergantungan hubungan asosiasi.

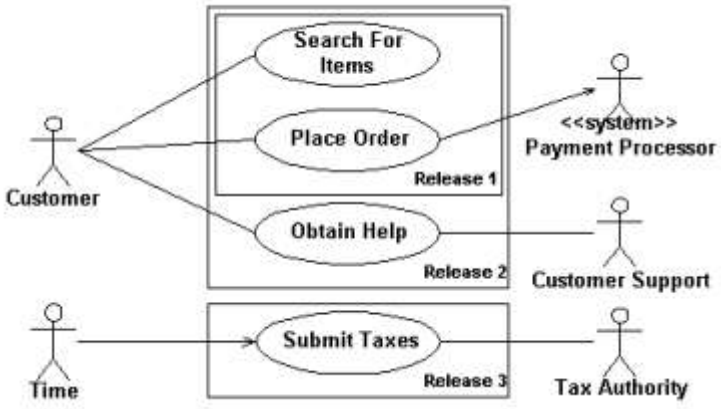
6. *Package diagram*\

   Paket diagram biasanya digunakan untuk menggambarkan tingkat organisasi yang tinggi dari suatu proyek software. Atau dengan kata lain untuk menghasilkan diagram ketergantungan paket untuk setiap paket dalam Pohon Model.

B. Behavior Diagram

1. *Usecase Diagram*

   Diagram yang menggambarkan actor, use case dan relasinya sebagai suatu urutan tindakan yang memberikan nilai terukur untuk aktor. Sebuah use case digambarkan sebagai elips horizontal dalam suatu diagram UML use case.



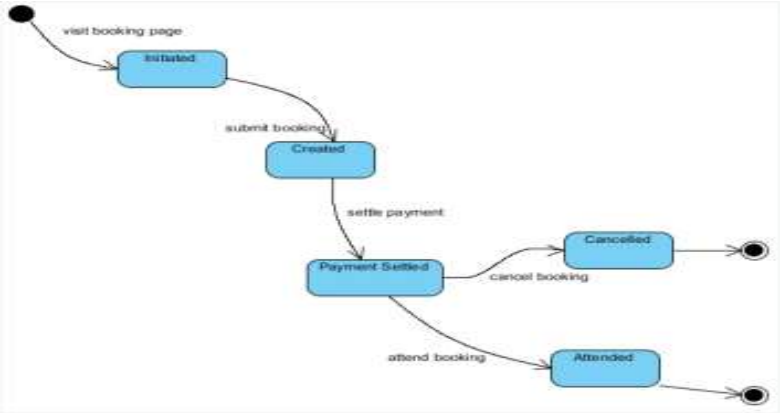
Gambar.II.6. Contoh Notasi *Usecase Diagram*
(*Sumber : Haviluddin , 2011 ; 4*)

2. *Activity diagram*

   Menggambarkan aktifitas-aktifitas, objek, state, transisi state dan event. Dengan kata lain kegiatan diagram alur kerja menggambarkan perilaku sistem untuk aktivitas

3. *State Machine diagram*

   Menggambarkan state, transisi state dan event.



Gambar.II.7. Contoh Notasi *State Machine Diagram*
(*Sumber : Haviluddin , 2011 ; 4*)

B. *Interaction diagram*

1. *Communication diagram*

   Serupa dengan sequence diagram, tetapi diagram komunikasi juga digunakan untuk memodelkan perilaku dinamis dari use case. Bila dibandingkan dengan Sequence diagram, diagram komunikasi lebih terfokus pada menampilkan kolaborasi benda daripada urutan waktu.

2. *Interaction Overview diagram*

   Interaksi *overview* diagram berfokus pada gambaran aliran kendali interaksi dimana node adalah interaksi atau kejadian interaksi.

3. *Sequence diagram*

   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 *use case diagram*

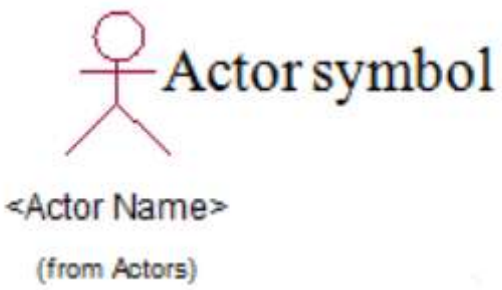
4. *Timing diagram*

   Timing diagram di UML didasarkan pada diagram waktu hardware awalnya dikembangkan oleh para insinyur listrik.

(Haviluddin , 2011 ; 3-5)

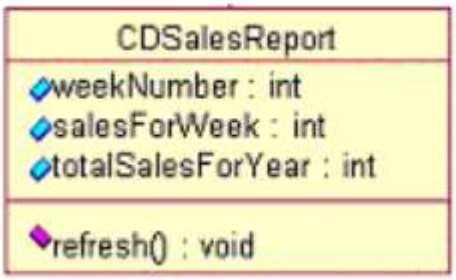
Untuk menggambarkan analisa dan desain diagram, UML memiliki seperangkat notasi yang akan digunakan ke dalam tiga kategori diatas yaitu struktur diagram, behaviour diagram dan interaction diagram. Berikut beberapa notasi dalam UML diantaranya :

1. *Actor*, menentukan peran yang dimainkan oleh user atau sistem lain yang berinteraksi dengan subjek. *Actor* adalah segala sesuatu yang berinteraksi langsung dengan sistem aplikasi komputer, seperti orang, benda atau lainnya. Tugas actor adalah memberikan informasi kepada sistem dan dapat memerintahkan sistem untuk melakukan sesuatu tugas.



Gambar.II.8. Notasi *Actor*
(*Sumber : Haviluddin , 2011 ; 6*)

2. *Class diagram* Notasi utama dan yang paling mendasar pada diagram UML adalah notasi untuk mempresentasikan suatu *class* beserta dengan atribut dan operasinya. *Class* adalah pembentuk utama dari sistem berorientasi objek.



Gambar.II.9. Notasi *Class*
(*Sumber : Haviluddin , 2011 ; 6*)

3. *Use Case* dan *use case specification*, *Use case* adalah deskripsi fungsi dari sebuah sistem perspektif pengguna. *Use case* bekerja dengan cara mendeskripsikan tipikal interaksi antara *user* (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Urutan langkah-langkah yang menerangkan antara pengguna dan sistem disebut skenario.
4. Realization,Realization menunjukkan hubungan bahwa elemen yang ada di bagian tanpa panah akan merealisasikan apa yang dinyatakan oleh elemen yang ada di bagian dengan panah.
5. Interaction, Interaction digunakan untuk menunjukkan baik aliran pesan atau informasi antar obyek maupun hubungan antar obyek.
6. Dependency, Dependency merupakan relasi yang menunjukan bahwa perubahan pada salah satu elemen memberi pengaruh pada elemen lain. Terdapat 2 stereotype dari dependency, yaitu include dan extend. Include menunjukkan bahwa suatu bagian dari elemen (yang ada digaris tanpa panah) memicu eksekusi bagian dari elemen lain (yang ada di garis dengan panah). Extend menunjukkan bahwa suatu bagian dari elemen di garis tanpa panah bisa disisipkan ke dalam elemen yang ada di garis dengan panah.

(Haviluddin , 2011 ; 6-7)