BAB II

TINJAUAN PUSTAKA

II.1. Pengertian Perancangan

Menurut Ahmad Afandi dikutip dari KBBI (Kamus Besar Bahasa Indonesia) perancangan adalah menata atau mengatur sesuatu yang diinginkan. Sementara perancangan sistem menentukan bagaimana suatu sistem akan menyelesaikan apa yang harus diselesaikan, tahap ini menyangkut mengkofigurasikan dari komponen-komponen perangkat lunak dan perangkat keras dari suatu sistem sehingga setelah instalasi dari suatu sistem akan benar-benar memuaskan rancang bangunan yang telah ditetapkan pada akhir analisis sistem.

Sehingga dari pengertian diatas perancangan adalah mengatur, menata menetukan sesuatu itu menyelesaikan apa yang harus diselesaikannya. **(*Sumber : Ahmad Afandi ; 2015*)**.

II.2. Aplikasi

Aplikasi adalah suatu subkelas perangkat lunak komputer yang memanfaatkan kemampuan komputer langsung untuk melakukan suatu tugas yang diinginkan pengguna. Contoh utama aplikasi adalah pengolah kata, lembar kerja, memanipulasi foto, merancang rumah dan pemutar media. Beberapa aplikasi yang digabung bersama menjadi suatu paket disebut sebagai suatu paket atau deretan aplikasi (*application suite*). Contohnya adalah Microsoft Office dan

OpenOffice.org, yang menggabungkan suatu aplikasi pengolah kata, lembar kerja dan beberapa aplikasi lainnya.

Aplikasi-aplikasi dalam suatu paket biasanya memiliki antarmuka pengguna yang memiliki kesamaan sehingga memudahkan pengguna untuk mempelajari dan menggunakan tiap aplikasi. Sering kali, mereka memiliki kemampuan untuk saling berinteraksi satu sama lain sehingga menguntungkan pengguna. Contohnya, suatu lembar kerja dapat dibenamkan dalam suatu dokumen pengolah kata walaupun dibuat pada aplikasi lembar kerja yang terpisah.

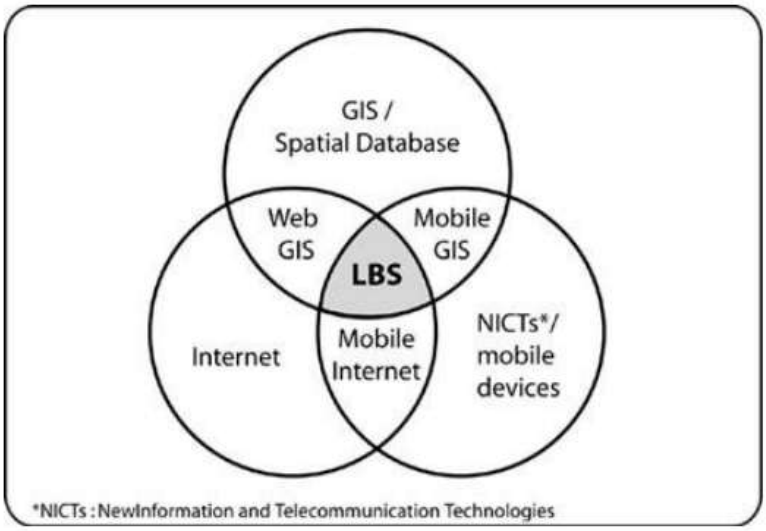
Aplikasi yang akan dirancang akan memanfaatkan fungsi dari LBS (Location Base Services) yang akan dipadankan dengan Google Map, dan pemanfaatan GPS. GPS disini berfungsi untuk mendapatkan koordinat dari pengguna aplikasi yang kemudian akan diterjemahkan dalam bentuk titik yang akan ditampilkan pada Google Map.

Akurasi dari GPS pada perangkat yang digunakan harus dibantu dengan kemampuan dari jaringan internet yang baik, hal ini bertujuan untuk mendapatkan akurasi yang tinggi. Aplikasi ini nantinya akan berperan sebagai alat yang berfungsi untuk mendapatkan rute alternatif yang dapat ditempuh oleh pengguna aplikasi. Rute dapat artikan sebagai perpindahan benda atau orang dari titik awal menuju titik tujuan melalui jalur-jalur yang memungkinkan untuk dilalui baik dengan menggunakan berjalan kaki atau dengan berkendara. **(*Sumber : Dahlan Abdullah dan Cut Ita Erliana ; 2012*)**.

II.3. *Location Based Services* (LBS)

Layanan Berbasis lokasi adalah layanan informasi yang dapat diakses melalui *mobile device* dengan mengunakan *mobile network*, yang dilengkapi kemampuan untuk memanfaatkan lokasi dari *mobile device* tersebut. *Location Based Services* (LBS) memberikan kemungkinan

komunikasi dan interaksi dua arah. Oleh karena itu pengguna memberitahu penyedia layanan untuk mendapatkan informasi yang dia butuhkan, dengan referensi posisi pengguna tersebut. Layanan berbasis lokasi dapat digambarkan sebagai suatu layanan yang berada pada pertemuan tiga teknologi yaitu : *Geographic Information System, Internet Service,* dan *Mobile Device*, hal ini dapat dilihat pada gambar *Location Based Services* (LBS) adalah pertemuan dari tiga teknologi.



Gambar II.1. LBS sebagai simpang tiga teknologi
(*Sumber : Juwita Imaniar ; 2013*)

Secara Garis besar jenis Layanan Berbasis Lokasi juga dapat dibagi menjadi dua,

yaitu :

1. *Pull Service*: Layanan diberikan berdasarkan permintaan dari pelanggan akan kebutuhan suatu informasi. Jenis layanan ini dapat dianalogikan seperti menggakses suatu *web* pada jaringan internet
2. *Push Service*: Layanan ini diberikan langsung oleh *sevice provider* tanpa menunggu permintaan dari pelanggan, tentu saja informasi yang diberikan tetap berkaitan dengan kebutuhan pelanggan. **(*Sumber : Juwita Imaniar ; 2013*)**

II.4. *Google Maps*

Google Maps merupakan layanan dari google yang mempermudah pengunanya untuk melakukan kemampuan pemetaan untuk aplikasi yang dibuat. Sedangkan *Google Maps* API memungkinkan pengembangan untuk mengintegrasikan *Google Maps* ke dalam situs *web*. Dengan menggunakan *Google Maps* API memungkinkan untuk menanamkan situs *Google Maps* ke dalam situs eksternal, di mana situs data tertentu dapat dilakukan *overlay*.

Meskipun pada awalnya hanya *JavaScript* API, API *Maps* sejak diperluas untuk menyertakan sebuah API untuk Adobe Flash aplikasi, layanan untuk mengambil gambar peta statis, dan layanan *web* untuk melakukan geocoding, menghasilkan petunjuk arah mengemudi, dan mendapatkan profil elevasi.

Kelas kunci dalam perpustakaan *Maps* adalah *MapView* , sebuah *subclass* dari *ViewGroup* dalam standar perpustakaan Android. Sebuah *MapView* menampilkan peta dengan data yang diperoleh dari layanan *Google Maps*. Bila *MapView* memiliki fokus, dapat menangkap tombol yang ditekan dan gerakan sentuh untuk pan dan zoom peta secara otomatis, termasuk penanganan permintaan jaringan untuk ubin peta tambahan. Ini juga menyediakan semua elemen UI yang diperlukan bagi pengguna untuk mengendalikan peta. Aplikasi tersebut juga dapat menggunakan metode *MapView* kelas untuk mengontrol *MapView* secara terprogram dan menarik sejumlah jenis Tampilan di atas peta.

Secara umum, kelas *MapView* menyediakan pembungkus di *Google Maps* API yang memungkinkan aplikasi tersebut memanipulasi data *Google Maps* melalui metode kelas, dan itu memungkinkan dikerjakan dengan data *Maps* seperti jenis lain *Views*. Perpustakaan *Maps* eksternal bukan bagian dari perpustakaan Android standar, sehingga tidak mungkin ada pada

beberapa perangkat Android biasa. Demikian pula, perpustakaan *Maps eksternal* tidak termasuk dalam perpustakaan Android standar yang disediakan dalam SDK. *Google* API pengaya menyediakan perpustakaan *Maps* untuk sehingga dapat mengembangkan, membangun, dan menjalankan aplikasi berbasis peta di SDK Android, dengan akses penuh ke data *Google Maps*. **(*Sumber : Juwita Imaniar ; 2013*)**

II.5. Android

Android adalah sebuah *platform* pertama yang betul-betul terbuka dalam pengembangannya dan komperehensif untuk perangkat *mobile*, semua perangkat lunak yang ada difungsikan menjalankan sebuah *device mobile* tanpa memikirkan kendala kepemilikan yang menghambat inovasi pada teknologi *mobile*. Aplikasi Android ditulis dalam bahasa pemrograman *java*, yaitu kode *java* yang terkompilasi bersama-sama dengan data dan *file-file* sumber yang dibutuhkan oleh aplikasi yang digabungkan oleh *app tools* menjadi paket aplikasi Android, sebuah *file* yang ditandai dengan akhiran .apk. *File* inilah yang di distribusikan sebagai aplikasi dan di instal pada handset Android. *File* ini di unduh oleh pengguna ke perangkat mobile mereka. Semua kode dijadikan satu *file .apk*, dan kemudian kita sebut sebagai sebuah aplikasi. **(*Sumber : Ahmad ; 2015*)**

Sebagian besar aplikasi yang terdapat pada *Play Store* Android bersifat gratis, dan ada juga aplikasi berbayar sebagai cara untuk me-monitize aplikasi Android. Uniknya penamaan versi Android selalu menggunakan nama makanan dan di awali abjad yang berurutan sebagai berikut :

1. Android version 1.5 (Cupcake).
2. Android version 1.6 (Donut).

3. Android version 2.0/2.1 (Eclair).
4. Android version 2.2 (Frozen Yogurt/Froyo).
5. Android version 2.3 (Gingerbread).
6. Android version 3.0/3.1/3.2 (Honeycomb).
7. Android version 4.0 (Ice Cream Sandwich).
8. Android version 4.1/4.2 (Jelly Bean).
9. Android version 4.4 (Kitkat). **(*Sumber : Arif Akbarul Huda ; 2013*)**

II.5.1. Sejarah Sistem Operasi Android

Telepon seluler menggunakan berbagai macam sistem operasi seperti *Symbian OS*®, *Microsoft's Windows Mobile*®, *Mobile Linux*®, *iPhone OS*® (berdasarkan *Mac OS X*), *Moblin* ®(dari *Intel*), dan berbagai macam sistem operasi lainnya. API yang tersedia untuk mengembangkan aplikasi *mobile* terbatas dan oleh karena itulah *Google* mulai mengembangkan dirinya. *Platform* Android menjanjikan keterbukaan, kemudahan untuk menjangkau, *source code* yang terbuka, dan pengembangan *framework* yang *high end.*

Google membeli perusahaan Android Inc., yang merupakan sebuah perusahaan kecil berbasis pengembangan perangkat lunak untuk ponsel, pada tahun 2005 untuk memulai pengembangan pada *platform* Android. Tokoh utama pada Android Inc. meliputi Andy Rubin, Rich Miner, Nick Sears, dan Chris White.

Pada tanggal 5 November 2007, kelompok pemimpin industri bersama-sama membentuk *Open Handset Alliance* (OHA) yang diciptakan untuk mengembangkan standar terbuka bagi

perangkat mobile. OHA terdiri dari 34 anggota besar dan beberapa anggota yang terkemuka diantaranya sebagai berikut: *Sprint Nextel®, T-Mobile®, Motorola®, Samsung®, Sony Ericsson®, Toshiba®, Vodafone®, Google, Intel®* dan *Texas Instruments*.

Android SDK dirilis pertama kali pada 12 November 2007 dan para pengembang memiliki kesempatan untuk memberikan umpan balik dari pengembangan SDK tersebut. Pada bulan September 2008, *T-Mobile* memperkenalkan ketersediaan *T-Mobile* G1 yang merupakan *smartphone* pertama berbasiskan *platform* Android. Beberapa hari kemudian, *Google* merilis Android SDK 1.0. Google membuat *source code* dari *platform* Android menjadi tersedia di bawah lisensi *Apache's open source*.

Google merilis perangkat genggam (disebut Android Dev Phone 1) yang dapat menjalankan aplikasi Android tanpa terikat oleh berbagai jaringan *provider* telepon seluler pada akhir 2008. Tujuan dari perangkat ini adalah memungkinkan pengembang untuk melakukan percobaan dengan perangkat sebenarnya yang dapat menjalankan Android OS tanpa berbagai kontrak. *Google* juga merilis versi 1.1 dari sistem operasi Android pada waktu yang tidak lama. Versi 1.1 dari Android tidak mendukung adanya *soft keyboards* dan membutuhkan perangkat yang memiliki *keyboard* secara fisik. Android menyelesaikan masalah ini dengan merilis versi 1.5 pada bulan April 2009 dengan sejumlah tambahan fitur seperti kemampuan perekaman media, *widgets*, dan *live folders*.

Versi 1.6 dari Android OS dirilis pada bulan September 2009 dan hanya dalam waktu satu bulan versi Android 2.0 dirilis dan membanjiri seluruh perangkat Android. Versi ini memiliki kemampuan *advanced search, text to speech, gestures, dan multi touch.* Android 2.0 memperkenalkan kemampuan untuk menggunakan HTML karena didukung oleh HTML 5.

Semakin banyak aplikasi berbasiskan Android setiap harinya yang terdapat pada *application store* secara *online* atau dikenal sebagai Android Market. **(*Sumber : Safaat Nazruddin H ; 2014*)**

II.5.2. Arsitektur Android

Secara garis besar Arsitektur Android dapat dijelaskan sebagai berikut:

1. *Applications dan Widgets*

Application dan *Widgets* ini adalah layer dimana kita berhubungan dengan aplikasi saja, dimana biasanya kita *download* aplikasi kemudian kita lakukan instalasi dan jalankan aplikasi tersebut. Di *layer* terdapat terdapat aplikasi inti termasuk *email client*, program *Short Message Service*(SMS), kalender, peta, *browser*, daftar kontak, dan lain-lain. Semua aplikasi ditulis dengan menggunakan bahasa pemgrograman *Java*.

2. *Applications Frameworks*

Android adalah "*Open Development Platform*" yaitu Android menawarkan kepada pengembang untuk membangun aplikasi yang bagus dan inovatif. Sehingga bisa kita simpulkan *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 *content prviders* yang berupa sms dan panggilan telepon. Komponen-komponen yang termasuk di dalam *Applications Frameworks* adalah sebagai berikut:

a. *Views*

b. *Content Provider*

c. *Resource Manager*

d. *Notification Manager*

e. *Activiti Manager*

3. *Libraries*

Libraries ini adalah layer dimana 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*. Beberapa dari library utama dijelaskan sebagai berikut:

a. *System C Library*

Merupakan implementasi turunan dari standar *system library C* (*libc*) yang diatur untuk peralatan berbasis *embedded Linux*.

b. *Media Libraries*

Disediakan oleh *Packet Video* (salah satu anggota dari OHA) yang memberikan *library* untuk memutar ulang dan menyimpan format suara dan *video*, serta *static image file* seperti *MPEG4*, *MP3, AAC, AMR, JPG, and PNG*.

c. *Surface Manager*

Mengatur akses ke dalam subsistem tampilan dan susunan grafis *layer* 2D dan 3D secara mulus dari beberapa aplikasi dan menyusun permukaan gambar yang berbeda pada layar ponsel.

d. *LibWebCore*

Merupakan *web browser* modern yang menjadi kekuatan bagi *browser* Android dan sebuah *embeddable web view*.

e. *Scalable Graphics Library (SGL)*

SGL mendasari mesin grafis 2D dan bekerja bersama-sama dengan lapisan pada level yang lebih tinggi dari kerangka kerja (seperti *Windows Manager* dan *Surface Manager*) untuk mengimplementasikan keseluruhan *graphics pipeline* dari Android.

f. *3DLibraries*

Implementasi yang didasarkan pada *OpenGL ES* 1.0 *APIs* dimana *library* menggunakan baik akselerasi perangkat keras 3D (jika tersedia) ataupun yang disertakan, dengan rasterisasi perangkat lunak 3D yang sangat optimal.

g. *FreeType Library*

Digunakan untuk menghaluskan semua tulisan *bitmap* dan vektor.

h. *SQLite*

Merupakan relational *database* yang kuat dan ringan serta tersedia untuk semua aplikasi.

4. *Android Runtime*

Merupakan lokasi dimana komponen utama dari DVM ditempatkan. DVM dirancang secara khusus untuk Android pada saat dijalankan pada lingkungan yang terbatas, dimana baterai yang terbatas, CPU, memori, dan penyimpanan data menjadi fokus utama. Android memiliki sebuah *tool* yang terintegrasi yaitu "*dx*" yang mengkonversi *generated byte code* dari (*.JAR*) ke dalam *file* (*.DEX*) sehingga *byte code* menjadi lebih efisien untuk dijalankan pada prosesor yang kecil. Hal ini memungkinkan untuk memiliki beberapa jenis dari DVM berjalan pada suatu peralatan tunggal pada waktu yang sama. *Core libraries* ditulis dalam bahasa *Java* dan berisi kumpulan *class*, *I/O* dan peralatan lain.

5. *Linux Kernel*

Linux kernel adalah *layer* dimana inti dari operating sistem dari android itu berada. Berisi *file-file system* yang mengatur sistem *processing*, *memory*, *resource*, *drivers*, dan sistem operasi android lainnya. **(*Sumber : Safaat Nazruddin H ; 2014*)**

Gambar II.2. Arsitektur Android
(*Sumber : Safaat Nazruddin H ; 2014*)

II.6. *Java*

Bahasa pemrograman *java* dikembangkan oleh *Sun Microsystem* yang dimulai oleh james Gosling dan dirilis pada tahun 1995. Saat ini *Sun Microsystem* telah diakui oleh *Oracle Corporation*. *Java* bersifat *Write Once, Run Anywhere* (program yang ditulis satu kali dan dapat berjalan pada banyak *platform*. (***Sumber : Jubilee Enterprise ; 2015***)

II.6.1. *JavaScript*

JavaScript merupakan salah satu bahasa script *website* yang paling banyak digunakan untuk menambah manipulasi *script* HTML dan CSS pada sisi *client/browser*. *JavaScript* mampu memberikan fungsionalitas lebih pada *website*, seperti validasi *form*, berkomunikasi dengan *sever*, serta membuat*website* lebih interaktif dan animatif.

JavaScript digunakan pada banyak , seperti *Internet Explorer*, *Firefox, Chrome, Opera, Safari,* dan lain sebagainya. Hampir seluruh *browser* mendukung *JavaScript* sehigga tidak perlu khawatir kode *JavaScript* yang digunakan pada *website* tidak berfungsi. **(*Sumber : Wahana Komputer ; 2012*)**

II.7. 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*)**

II.7.1. 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. Android Development Tools (ADT) : Plug-in Android Development Tools (ADT) untuk mendukung pembuatan dan ;proses debugging dari aplikasi Android yang sedang buat.

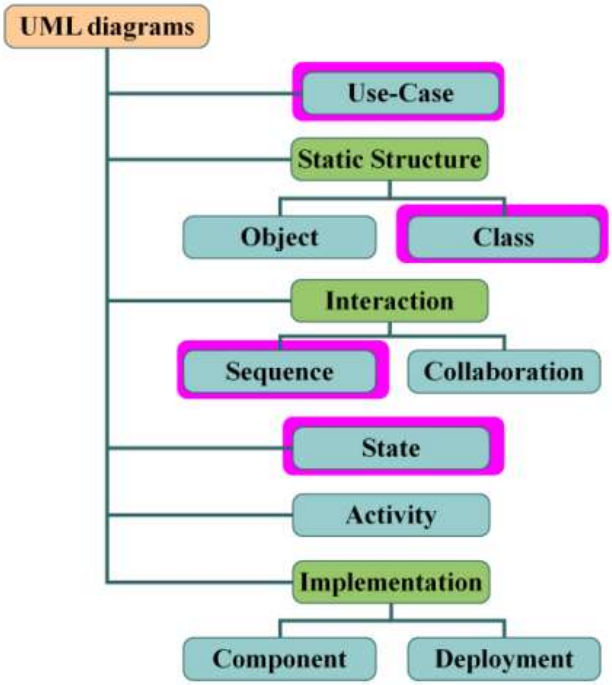
II.8. UML *(Unified Modelling Language)*

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. **(Sumber : Haviluddin ; 2011)**

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. 3. Diagram UML
(*Sumber : Haviluddin ; 2011*)

1. 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*). **(*Sumber : Haviluddin ; 2011*)**

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 dalam UML terdiri atas :

1. *Class Diagram*

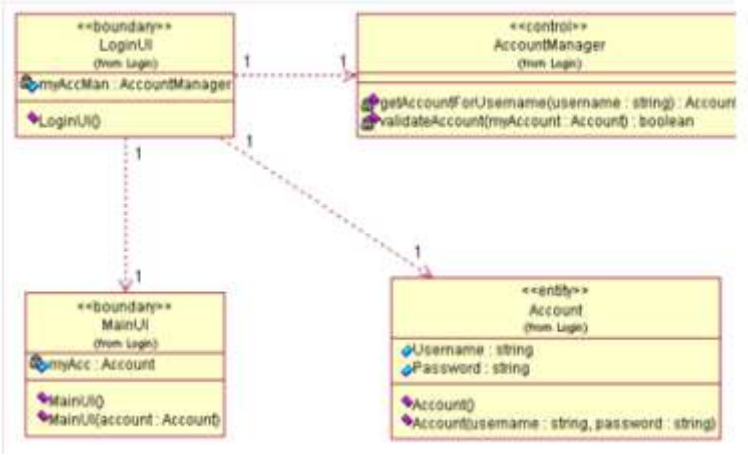
   Class diagram menggambarkan struktur statis dari kelas dalam sistem anda dan menggambarkan atribut, operasi dan hubungan antara kelas. *Class diagram* membantu dalam memvisualisasikan struktur kelas-kelas dari suatu sistem dan merupakan tipe *diagram* yang paling banyak dipakai. Selama tahap desain, *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. Metode

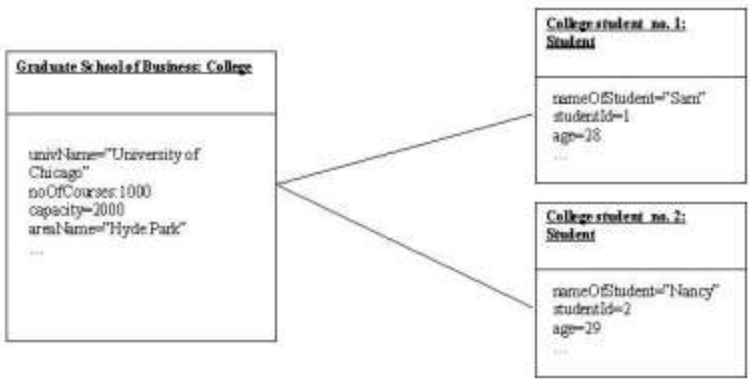


Gambar II.4. Contoh Notasi Class Diagram
(*Sumber : Haviluddin ; 2011*)

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.

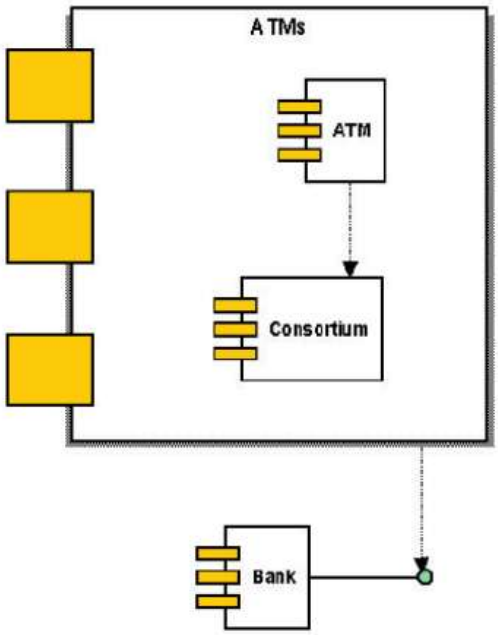
Berikut notasi *object diagram :*



Gambar II.5. Contoh Notasi Object Diagram
(*Sumber : Haviluddin ; 2011*)

3. *Component diagram*

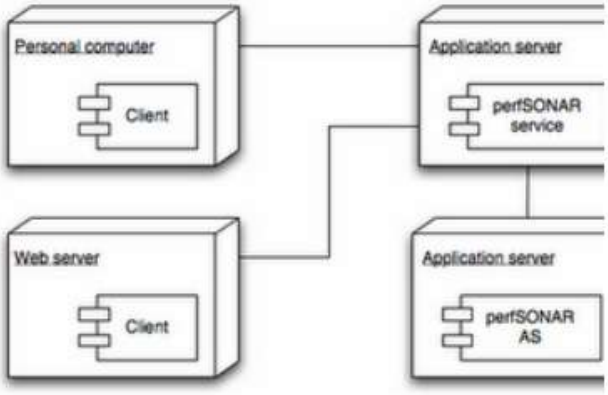
   Component diagram menggambarkan struktur fisik dari kode, pemetaan pandangan logis dari kelas proyek untuk kode aktual di mana logika ini dilaksanakan.



Gambar II.6. Contoh Notasi Object Diagram
(*Sumber : Haviluddin ; 2011*)

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.7. Contoh Notasi Deployment Diagram
(*Sumber : Haviluddin ; 2011*)

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.

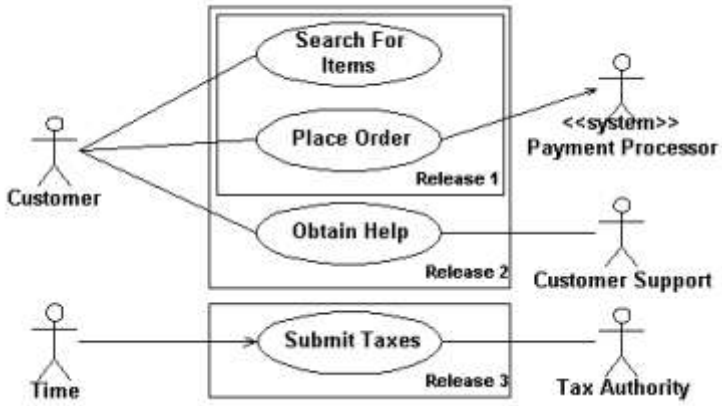
6. *Package diagram*

   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.*



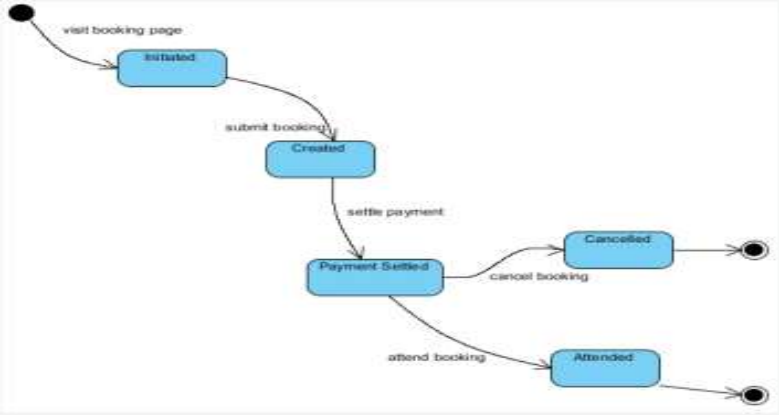
Gambar II.8. Contoh Notasi Usecase Diagram
(*Sumber : Haviluddin ; 2011*)

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.9. Contoh Notasi State Machine Diagram
(*Sumber : Haviluddin ; 2011*)

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*

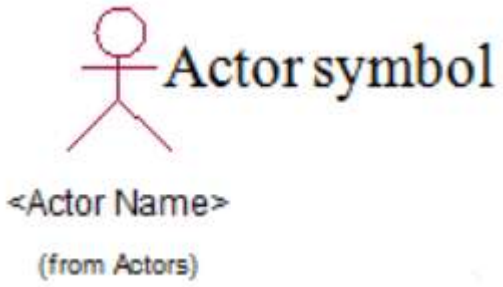
   Sequence diagram menjelaskan interaksi objek yang disusun berdasarkan urutan waktu. Secara mudahnya *sequence diagram* adalah gambaran tahap demi tahap, termasuk kronologi (urutan) perubahan secara logis yang seharusnya dilakukan untuk menghasilkan sesuatu sesuai dengan *use case diagram*

4. *Timing diagram*

Timing diagram di UML didasarkan pada diagram waktu hardware awalnya dikembangkan oleh para insinyur listrik. **(*Sumber : Haviluddin ; 2011*)**

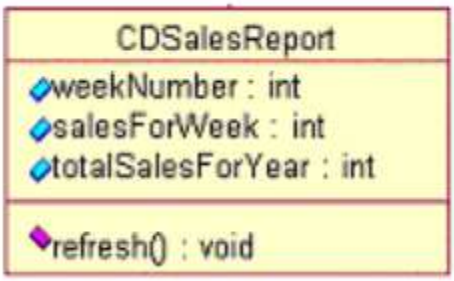
Untuk menggambarkan analisa dan desain diagram, UML memiliki seperangkat notasi yang akan digunakan ke dalam tiga kategori diatas yaitu struktur diagram, behaviour diagram dan interaction diagram. Berikut beberapa notasi dalam UML diantaranya :

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.10. Notasi Actor
(*Sumber : Haviluddin ; 2011*)

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.11. Notasi Class
(*Sumber : Haviluddin ; 2011*)

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. **(*Sumber : Haviluddin ; 2011*)**