BAB II

LANDASAN TEORI

2.1 Indekos

Indekos = kos yaitu tinggal di rumah orang lain dengan atau tanpa makan (dengan membayar setiap bulan), memondok. (Departemen Pendidikan Nasional, 2008).

2.2 *Unified Modelling Language* (UML)

2.2.1 Pengertian UML

UML (*Unified Modelling Language*) adalah sebuah bahasa untuk menentukan, visualisasi, konstruksi, dan mendokumentasikan *artifacts* dari sistem *software*, untuk memodelkan bisnis, dan sistem non-*software* lainnya. UML merupakan suatu kumpulan teknik terbaik yang telah terbukti sukses dalam memodelkan sistem yang besar dan kompleks (Suhender & Gunadi, 2002) .

2.2.2 *Artifact* UML

Artifact adalah sepotong informasi yang digunakan atau dihasilkan dalam suatu proses rekayasa *software* (Suhender & Gunadi, 2002). *Artifact* dapat berupa model, deskripsi, atau *software*. Untuk membuat suatu model, UML memiliki diagram grafis sebagai berikut:

1. *Use-case diagram*

   Use-case diagram menjelaskan manfaat sistem jika dilihat menurut pandangan orang yang berada di luar sistem (*actor*). Diagram ini menunjukkan fungsionalitas suatu sistem atau kelas dan bagaimana sistem berinteraksi dengan dunia luar.

2. *Class diagram*

Class diagram membantu untuk visualisasi struktur kelas-kelas dari suatu sistem dan merupakan tipe diagram yang paling banyak dipakai. *Class diagram* memperlihatkan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain (dalam *logical view*) dari suatu sistem.

3. *Behavior diagram:*

a) *Statechart diagram*

Statechart diagram memperlihatkan urutan keadaan sesaat (*state*) yang dilalui sebuah objek, kejadian yang menyebabkan sebuah transisi dari satu *state* atau aktivitas kepada yang lainnya dan aksi yang menyebabkan perubahan satu *state* atau aktivitas. *Statechart diagram* khususnya digunakan untuk memodelkan taraf-taraf diskrit dari sebuah siklus hidup objek sedangkan *activity diagram* paling cocok digunakan untuk memodelkan urutan aktivitas dalam suatu proses.

b) *Activity diagram*

Activity diagram memodelkan alur kerja (*workflow*) sebuah proses bisnis dan urutan aktivitas dalam suatu proses. Diagram ini sangat mirip dengan sebuah *flowchart* karena dapat memodelkan sebuah alur kerja dari satu aktivitas ke aktivitas lainnya atau dari satu aktivitas ke dalam keadaan sesaat (*state*). *Activity diagram* juga sangat berguna ketika ingin menggambarkan perilaku paralel atau menjelaskan bagaimana perilaku dalam berbagai *use-case* berinteraksi.

c) *Interaction diagram:*

i. *Sequence diagram*

Sequence diagram 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 di dalam *use-case*. Tipe diagram ini sebaiknya digunakan di awal tahap desain atau analisis karena kesederhanaannya dan mudah untuk dimengerti.

ii. *Collaboration diagram*

Collaboration diagram melihat pada interaksi dan hubungan terstruktur antar objek. Tipe diagram ini menekankan pada hubungan (*relationship*) antar objek, sedangkan *sequence diagram* menekankan pada urutan kejadian. Dalam satu *collaboration diagram* terdapat beberapa *object*, *link*, dan *message*. *Collaboration diagram* digunakan sebagai alat untuk menggambarkan interaksi yang mengungkapkan keputusan mengenai perilaku sistem.

4. Implementation diagram:

a) *Component diagram*

Componen diagram menggambarkan alokasi semua kelas dan objek ke dalam komponen-komponen dalam desain fisik sistem *software*. Diagram ini memperlihatkan pengaturan dan kebergantungan antara komponen-komponen *software*, seperti *source code*, *binary code* dan komponen tereksekusi (*executable components*).

b) *Deployment diagram*

Setiap model hanya memiliki satu *deployment diagram*. Diagram ini memperlihatkan pemetaan *software* kepada *hardware*.

Diagram-diagram tersebut diberi nama berdasarkan sudut pandang yang berbeda-beda terhadap sistem dalam proses analisis atau rekayasa. Dibuatnya berbagai jenis diagram di atas karena:

1. Setiap sistem yang kompleks selalu paling baik jika didekati melalui himpunan berbagai sudut pandang yang kecil yang satu sama lain hampir saling bebas (*independent*). Sudut pandang tunggal senantiasa tidak mencakupi untuk melihat sistem yang besar dan kompleks.
2. Diagram yang berbeda-beda tersebut dapat menyatakan tingkatan yang berbeda-beda dalam proses rekayasa.

Diagram-diagram tersebut dibuat agar model yang dibuat semakin mendekati realitas.

2.2.3 Faktor yang Mendorong Dibuatnya UML

Menurut (Suhender & Gunadi, 2002) ada 3 faktor yang mendorong dibuatnya suatu pemodelan UML, di antaranya sebagai berikut:

1. Pentingnya Model

a) Membangun model untuk suatu sistem *software* (terlebih *software* untuk suatu organisasi bisnis) sangat bergantung pada konstruksinya atau kemudahan dalam memperbaikinya. Oleh karena itu, membuat model sangatlah penting sebagaimana pentingnya memiliki cetak biru untuk suatu bangunan yang besar.

b) Model yang bagus sangat penting untuk menghasilkan komunikasi yang baik antar anggota tim dan untuk meyakinkan sempurnanya arsitektur sistem yang dibangun.

c) Jika membangun suatu model dari suatu sistem yang kompleks, tidak mungkin dapat memahaminya secara keseluruhan. Dengan meningkatnya kekompleksan sistem, visualisasi dan pemodelan menjadi sangat penting. UML dibuat untuk merespon kebutuhan tersebut.

2. Kecenderungan Dunia Industri terhadap *Software*

a) Sebagai suatu nilai yang strategis bagi pasar *software* adalah dengan meningkatnya kebutuhan dunia industri untuk memiliki teknik otomatisasi dengan *software*. Oleh karena itu, penting sekali adanya teknik rekayasa *software* yang dapat meningkatkan kualitas dan mengurangi biaya dan waktu.

b) Komplektisitas dunia industri semakin meningkat karena bertambah luasnya ruang lingkup dan tahapan proses. Satu motivasi kunci bagi para pembangun UML adalah untuk membuat suatu himpunan semantik dan notasi yang mampu menangani kerumitan arsitektural dalam semua ruang lingkup.

3. Terjadinya Konvergensi Metode dan *Tool* Pemodelan

a) Sebelum adanya UML, terdapat ketidakjelasan bahasa pemodelan apa yang paling unggul. Para *user* harus memilih di antara bahasa pemodelan dan *tool* (*software*) pemodelan yang banyak dan mirip UML dibuat untuk membuat integrasi baru dalam bahasa pemodelan antar *tool* dan “proses”.

2.2.4 Tujuan UML

UML (*Unified Modelling Language*) memiliki beberapa tujuan utama dalam pembentukannya yaitu (Suhender & Gunadi, 2002):

1. Memberikan model yang siap pakai, bahasa pemodelan visual yang ekspresif untuk mengembangkan dan saling menukar model dengan mudah dan dimengerti secara umum.

2. Memberikan bahasa pemodelan yang bebas dari berbagai bahasa pemrograman dan proses rekayasa.
3. Menyatukan praktek-praktek terbaik yang terdapat dalam pemodelan.

2.3 *Platform Android*

2.3.1 Pengertian *Android*

Android adalah sebuah sistem operasi untuk perangkat *mobile* berbasis *Linux* yang mencakup sistem operasi, *middleware* dan aplikasi (Harahap, 2011). *Android* menyediakan platform terbuka bagi para pengembang untuk menciptakan aplikasi mereka. *Android* merupakan generasi baru platform *mobile*, platform yang memberikan pengembang untuk melakukan pengembangan sesuai dengan yang diharapkannya.

Pengembang memiliki beberapa pilihan ketika membuat aplikasi yang berbasis *Android*. Kebanyakan pengembang menggunakan *Eclipse* yang tersedia secara bebas untuk merancang dan mengembangkan aplikasi *Android*. *Eclipse* adalah IDE yang paling populer untuk pengembangan *Android*, karena memiliki *Android plug-in* yang tersedia untuk memfasilitasi pengembangan *Android*. Selain itu *Eclipse* juga mendapat dukungan langsung dari *Google* untuk menjadi IDE pengembangan aplikasi *Android*, ini terbukti dengan adanya penambahan *plugins* dari *Eclipse* untuk bisa membuat *project Android* dimana *source software* langsung dari situs resminya *Google*. Tetapi hal di atas tidak menutup kemungkinan untuk menggunakan IDE yang lain seperti *Netbeans* untuk melakukan pengembangan *Android*.

2.3.2 Perkembangan *Platform Android*

Hasil survei terbaru yang dilakukan *Canalys* menunjukkan bahwa *Android* berhasil melampaui *Symbian* sebagai platform *smartphone* terbesar di dunia. Distribusi produk *smartphone* berbasis *Android* selama kuartal IV-2010 mencapai 33,3 juta unit dan menempatkannya sebagai pemegang pangsa pasar terbesar 32,9 persen. (Wahono, 2011)

Symbian kini harus puas menempati peringkat kedua dengan pangsa pasar 30,6 persen. Jumlah *smartphone Symbian* yang terdistribusi masih di bawah *Android,* yakni hanya 31 juta unit.

Menyusul *Android* dan *Symbian* berturut-turut iOS dari Apple, *BlackBerry* dari RIM, dan *Windows Phone* 7 dari *Microsoft*. iOS meraih pangsa pasar 16,2 persen dengan 16,2 juta unit *smartphone*. *BlackBerry* meraih pangsa pasar 14,4 persen dengan 14,6 juta unit *smartphone*. Microsoft meraih pangsa pasar 3,1 persen dengan 3,1 juta unit *smartphone*. Sisanya, 2,9 persen pangsa pasar dibagi 3 juta unit *smartphone* dengan berbagai platform yang lain.

"Distribusi *smartphone* berbasis *Android* mengalami lonjakan tujuh kali lipat ke 33 juta unit di seluruh dunia karena didorong penjualan yang tinggi di berbagai vendor *handset*, seperti Samsung dari Korea dan HTC Corp Taiwan," kata analis *Canalys* dalam laporannya, Senin (31/1/2011). Pada kuartal IV-2009, *Android* memang baru mencatat pangsa pasar 8,7 persen dengan 4,7 juta unit *smartphone*.

Dari survei tersebut diketahui bahwa jumlah distribusi *smartphone* selama kuartal IV-2010 mengalami kenaikan hampir dua kali lipat dibanding

tahun lalu. Jumlah *smartphone* yang didistribusikan mencapai 101,1 juta unit dibandingkan 53,7 juta unit pada kuartal IV-2009.

Meski semua platform mengalami kenaikan jumlah distribusi *handset*, hanya *Android* yang mencatat kenaikan pangsa pasar. *Symbian* sebelumnya di urutan pertama dengan pangsa pasar 44,4 persen, disusul *BlackBerry* 20 persen, iOS 16,3 persen, *Android* 8,7 persen, *Windows* 7,2 persen, dan lainnya 3,4 persen.

2.3.3 *The Dalvik Virtual Machine* (DVM)

Salah satu elemen kunci dari *Android* adalah *Dalvik Virtual Machine* (Harahap, 2011). *Android* berjalan di dalam *Dalvik Virtual Machine* (DVM) bukan di *Java Virtual Machine* (JVM), sebenarnya banyak persamaannya dengan *Java Virtual Machine* (JVM) seperti Java ME (*Java mobile Edition*), tetapi *Android* menggunakan *Virtual Machine* sendiri yang dikostumisasi dan dirancang untuk memastikan bahwa beberapa *feature-feature* berjalan lebih efisien pada perangkat *mobile*.

Dalvik Virtual Machine (DVM) adalah ”*register bases*” sementara *Java Virtual Machine* (JVM) adalah ”*stack based*”, DVM didesain oleh Dan Bornsten dan beberapa *engineers Google* lainnya. DVM menggunakan *kernel Linux* untuk menangani fungsionalitas tingkat rendah termasuk keamanan, *threading*, dan proses serta manajemen memori. Ini memungkinkan untuk menulis aplikasi C / C + sama halnya seperti pada OS *Linux* kebanyakan. Meskipun dalam kenyataannya harus banyak memahami Arsitektur dan proses sistem dari *kernel Linux* yang digunakan dalam *Android* tersebut.

Semua *hardware* yang berbasis *Android* dijalankan dengan menggunakan *Virtual machine* untuk eksekusi aplikasi, pengembang tidak perlu khawatir tentang implementasi perangkat keras tertentu. *Dalvik Virtual Machine* mengeksekusi *executable file*, sebuah format yang dioptimalkan untuk memastikan memori yang digunakan sangat kecil. *The executable file* diciptakan dengan mengubah kelas bahasa *java* dan dikompilasi menggunakan *tools* yang disediakan dalam SDK *Android*.

2.3.4 *Android Software Development Kit* (*Android* SDK)

Android SDK adalah *tools* API *(Application Programming Interface*) yang diperlukan untuk mulai mengembangkan aplikasi pada platform *Android* menggunakan bahasa pemrograman Java (Harahap, 2011). *Android* merupakan subset perangkat lunak untuk smartphone yang meliputi sistem operasi, middleware dan aplikasi kunci yang dirilis oleh *Google*.

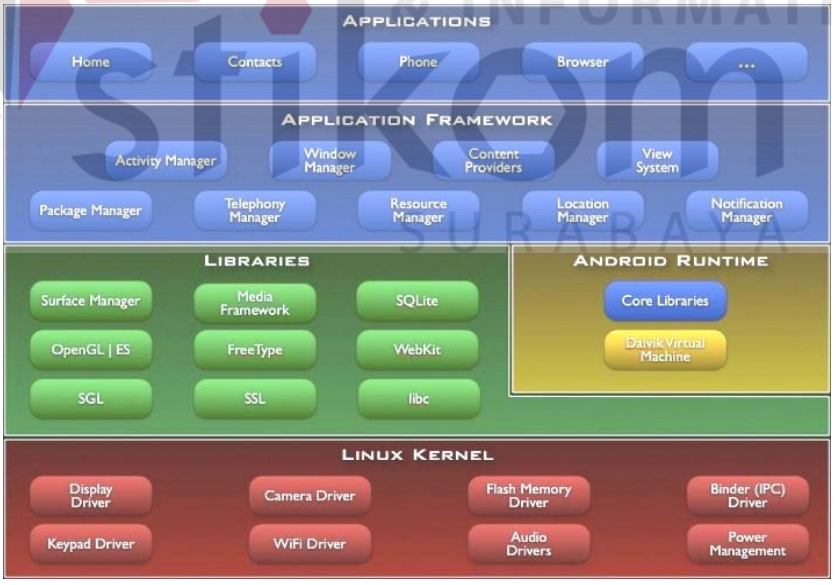
Saat ini disediakan *Android* SDK sebagai alat bantu dan API untuk mulai mengembangkan aplikasi pada platform *Android* menggunakan bahasa pemrograman Java. Sebagai platform aplikasi netral, *Android* memberi Anda kesempatan untuk membuat aplikasi yang bukan merupakan aplikasi bawaan *Smartphone*. Beberapa fitur-fitur *Android* yang paling penting adalah:

1. *Framework* Aplikasi yang mendukung penggantian komponen dan *reusable*.
2. Mesin Virtual *Dalvik* dioptimalkan untuk perangkat *mobile*.
3. *Integrated browser* berdasarkan *engine open source WebKit*.
4. Grafis yang dioptimalkan dan didukung oleh *libraries* grafis 2D, grafis 3D berdasarkan spesifikasi *OpenGL* ES 1.0 (Opsional akselerasi *hardware*).

5. *SQLite* untuk menyimpan data.
6. Media *Support* yang mendukung audio, video, dan gambar.
7. *Bluetooth*, EDGE, dan *WiFi* (tergantung *hardware*).
8. Kamera, GPS, kompas, dan *accelerometer* (tergantung *hardware*)
9. Lingkungan *Development* yang lengkap dan kaya termasuk perangkat *emulator*, *tool* untuk *debugging*, profil dan kinerja memori, dan *plugin* untuk IDE *Eclipse*.

2.3.5 Arsitektur *Android*

Arsitektur sistem terdiri atas 5 *layer*, pemisahan *layer* bertujuan untuk memberikan abstraksi sehingga memudahkan pengembangan aplikasi. *Layer-layer* tersebut adalah *layer* aplikasi, *layer framework* aplikasi, *layer libraries*, *layer runtime*, dan *layer kernel* (Harahap, 2011). Gambaran umum komponen-komponen dalam arsitektur *Android* dapat dilihat pada Gambar 2.4.



Gambar 2.1 Arsitektur *Android*

Secara garis besar Arsitektur *Android* dapat dijelaskan sebagai berikut:

1. *Applications* dan *Widgets*

Applications dan *Widgets* ini adalah layar di mana berhubungan dengan aplikasi saja, di mana biasanya setelah aplikasi diunduh, kemudian melakukan instalasi dan jalankan aplikasi tersebut, di *layer* inilah terdapat seperti aplikasi inti termasuk klien email, program SMS, kalender, peta, *browser*, kontak, dan lain-lain. Semua aplikasi ditulis menggunakan bahasa pemrograman *java*.

2. Aplikasi *Frameworks*

   Android adalah "*Open Development Platform*" yaitu *Android* menawarkan kepada pengembang atau memberi kemampuan kepada penembang untuk membangun aplikasi yang bagus dan inovatif. Pengembang bebas untuk mengakses perangkat keras, akses informasi *resource*, menjalankan *service background*, mengatur alarm, dan menambahkan tambahan seperti status *notifications*, dan masih banyak lagi. Pengembang memiliki akses penuh menuju API *framework* seperti yang dilakukan oleh aplikasi yang berkategori inti. Arsitektur aplikasi dirancang supaya dengan mudah dapat menggunakan komponen yang sudah digunakan (*reuse*).

3. *Libraries*

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

4. *Android Runtime*

   Layer yang membuat aplikasi *Android* dapat dijalankan di mana dalam prosesnya menggunakan implementasi *Linux*. *Dalvik Virtual Machine* (DVM) merupakan mesin yang membentuk dasar kerangka aplikasi *Android*.

5. *Linux Kernel*

   Linux kernel adalah layer di mana inti dari *operating sistem* dari *Android* itu sendiri, berisi *file-file system* yang mengatur *system processing*, *memory*, *resource*, *drivers*, dan sistem-sistem *operating Android* lainnya. *Linux Kernel* yang digunakan *Android* adalah *Linux Kernel release* 2.6.

2.4 *Google Maps*

Google Maps adalah sebuah layanan gratis peta digital dari *Google* berbasis *web* yang dapat digunakan dan ditempatkan pada *website* tertentu dengan menggunakan *Google Maps* API (*Google Inc*, 2011).

Google Maps sendiri mempunyai fitur-fitur antara lain navigasi peta dengan *dragging mouse*, *zoom-in* dan *zoom-out* untuk menunjukkan informasi peta secara detil memberi penanda pada peta dan memberi informasi tambahan. Mode *viewing* pada *Google Maps* berupa "*Map*" (peta topografi dan jalan), "*satelite*" (peta berupa foto satelit dan foto resolusi tinggi dari udara), "*Hybrid*" (peta berupa foto satelit dan peta jalan berada di atasnya) dan "*Street View*", fasilitas ini secara resmi diperkenalkan oleh *Google* pada Mei 2007.

Seperti layanan aplikasi *Google* lainnya. *Google Maps* dibangun dengan menggunakan *javascript*, seperti ketika *user* menggeser pada peta, sepetak peta akan diunduh dari *server* dan ditampilkan pada *user* tanpa harus *refresh* seluruh halaman *web*. Sebuah lokasi yang ditunjukkan dengan sebuah pin sebenarnya

adalah berupa *file* PNG transparan yang diletakkan di atas peta. Teknik yang digunakan untuk memberikan interaktifitas yang tinggi dengan cara melakukan *request* secara *asynchronous* dengan *Javascript* dan XML yang juga dikenal dengan AJAX.

2.4.1 *Google Maps* API

Google telah membuat *Google Maps* API untuk memfasilitasi para *developer* untuk mengintegrasikan *Google Maps* pada *website*-nya. Ini merupakan layanan gratis yang sementara tidak mengandung iklan, tetapi *Google* menyatakan pada perjanjian menggunakan layanan bahwa mereka berhak untuk menampilkan iklan dimasa yang akan datang (*Google Inc*, 2011). Dengan menggunakan *Google Maps* API, pengguna dapat menampilkan seluruh fasilitas *Google Maps* pada *website* pengguna. Dimulai dengan membuat API *key* (API *Key* ini berfungsi sebagai kunci akses untuk website pengguna) dan pengguna sudah dapat menggunakan fungsi-fungsi yang ada pada *Google Maps* API untuk *website*.

2.4.2 *Google Maps* API pada *Android*

Location Based Service (LBS) atau layanan berbasis lokasi adalah istilah umum yang digunakan untuk menggambarkan untuk menemukan lokasi perangkat yang digunakan. Dua unsur utama LBS adalah:

1. *Location Manager (*API *Maps)*

   Menyediakan *tool/ source* untuk LBS, *Application Programming Interface* (API) *Maps* menyediakan fasilitas untuk menampilkan, memanipulasi *maps*/ peta beserta *feature-feature* lainnya seperti tampilan satelit, *street* (jalan), maupun gabungannya. Paket ini berada pada *com.google.android.maps*.

2. *Location Providers (*API *Locations)*

   Menyediakan teknologi pencarian lokasi yang digunakan oleh *device*/ perangkat. API *Location* berhubungan dengan data GPS dan data lokasi *real-time*. Dengan *Location Providers*, penggunadapat menentukan lokasi pengguna saat ini, *Track* gerakan/ perpindahan, serta kedekatan dengan lokasi tertentu dengan mendeteksi perpindahan.

Android memberikan akses aplikasi ke layanan lokasi yang didukung oleh perangkat melalui kelas-kelas dalam *package android.location*. komponen utama dari kerangka lokasi adalah sistem layanan *Location Manager*, yang menangani akses ke layanan lokasi. *Location Manager* tidak secara langsung dibuat objeknya, akan tetapi digunakan permintaan ke sistem dengan memanggil *getSystemService (Context.LOCATION_SERRVICE)* sehingga akan didapatkan *handler* untuk objek *Location Manager*. Ketika telah didapatkan objek *Location Manager* maka dapat melakukan *query* ke semua daftar *Location Providers* yang dikenal oleh *Location Manager* sebagai lokasi terakhir.

Ketika bekerja dengan *emulator* maka fungsi untuk lokasi tidak dapat digunakan karena tidak tersedia GPS nyata pada *emulator* sehingga diperlukan data lokasi tiruan menggunakan *ddms* pada *eclipse* maupun melalui *adb*. Melalui *ddms* dapat dilakukan secara manual pengiriman koordinat bujur/ lintang ke perangkat, menggunakan *file* GPX menjelaskan rute untuk pemutaran ke perangkat, menggunakan *file* KML menjelaskan letak individu untuk pemutaran *sequencing* ke perangkat.

Geocoding adalah proses merubah alamat-alamat ke koordinat geografi (latitude dan longitude) sehingga dapat menandai posisi pada peta (Developers,

2014). *Geocoding* memungkinkan menerjemahkan antara alamat jalan dengan bujur/ lintang koordinat peta. Hal ini dapat memberikan konteks dikenali untuk lokasi dan koordinat yang digunakan dalam layanan berbasis lokasi dan peta berbasis *Activity*. Pencarian *geocoding* dilakukan di *server*, sehingga aplikasi akan meminta untuk memasukkan sebuah izin penggunaan internet. Kelas *geocoder* menyediakan akses untuk dua fungsi *geocoding*:

1. *Forward Geocoding*: mencari lintang dan bujur alamat.
2. *Reverse Geocoding*: Mencari alamat dan jalan untuk sesuai lintang dan bujur.