BAB II

LANDASAN TEORI

2.1 Media Pembelajaran

Cangara (2006) dalam (*Pengantar Ilmu Komunikasi*) mengatakan bahwa media adalah alat atau sarana yang digunakan untuk menyampaikan pesan dari komunikator kepada khalayak. Menurut Gagne (1970) dalam Arif S Sadiman (2009) menyatakan bahwa media adalah berbagai jenis komponen dalam lingkungan siswa yang merangsangnya untuk belajar. Sementara itu Briggs (1970) berpendapat bahwa media adalah segala alat fisik yang dapat menyajikan pesan serta merangsang siswa untuk belajar buku, film, dan kaset. Asosiasi Pendidikan Nasional (*National Education Association*) memiliki pengertian yang berbeda. Media adalah bentuk komunikasi baik tercetak maupun *audiovisual* serta peralatannya (Sadiman, 2009).

Apapun definisi yang diberikan, ada persamaan di antara definisi tersebut yaitu media adalah segala sesuatu yang dapat digunakan untuk menyalurkan pesan dari pengirim ke penerima sehingga dapat merangsang pikiran, perasaan, minat serta perhatian siswa sehingga tercipta proses belajar.

2.2 Sabak

Sabak adalah media untuk menulis yang terbuat dari semacam batu tipis yang dibingkai dengan kayu dan`grip sebagai alat tulisnya sedangkan untuk menghapus tulisan di sabak menggunakan air dan arang. Berikut ini bentuk dari sabak dapat dilihat pada Gambar 2.1

Gambar 2.1 Sabak dan Grip

2.3 Android

Android adalah sebuah sistem operasi untuk perangkat *mobile* berbasis linux yang mencakup sistem operasi, *middleware* dan aplikasi (Saffat, 2012). Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka. Pada 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* yang merupakan konsorsium dari 34 perusahaan piranti keras, piranti lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile dan Nvidia.

Pada perilisan perdana android, 5 November 2007, Android bersama *Open Handset Alliance* menyatakan mendukung pengembangan *open source* pada perangkat *mobile.* Dilain pihak, Google merilis kode-kode Android dibawah lisensi Apache, sebuah lisensi perangkat lunak dan *open platform* seluler.

Di dunia terdapat dua jenis distributor sistem operasi Android. Pertama yang mendapat dukungan penuh dari Google atau Google Mail Service (GMS) dan kedua adalah yang benar-benar bebas distribusinya tanpa dukungan langsung Google atau dikenal sebagai *Open Handset Distribution* (Safaat,2012).

Sekitar september 2007 Google mengenalkan Nexus One, Salah satu jenis *smartphone* yang menggunakan Android sebagai sistem operasinya. Telepon selular ini diproduksi oleh HTC Corporation dan tersedia di pasaran pada 5 Januari 2008. Pada 9 Desember 2008, diumumkan anggota baru yang bergabung dalam program kerja Android ARM Holdings, Atheros Communication, diproduksi oleh Asustek Computer Inc, Garmin Ltd, Softbank, Sony Ericson, Toshiba Corp, dan Vodafone Group Plc. Seiring pembentukan *Open Handset Alliance*, OHA mengumumkan produk perdana mereka Android, perangkat *mobile* yang merupakan modifikasi kernel Linux 2.6. Sejak Android dirilis telah dilakukan berbagai pembaharuan berupa perbaikan *bug* dan penambahan fitur baru.

Pada masa saat ini kebanyak vendor-vendor *smartphone* sudah memproduksi *smartphone* berbasis android, vendor-vendor itu antara lain HTC, Motorola, Samsung, LG, Dell, Sony Ericson, Acer, Asus dan masih banyak lagi vendor *smartphone* di dunia yang memproduksi Android. Hal ini karena Android itu adalah sistem operasi yang *open source* sehingga bebas didistribusikan dan dipakai oleh vendor manapun (Safaat,2012).

Tidak hanya menjadi sistem operasi di *smartphone*, saat ini Android menjadi pesaing utama dari Apple pada sistem operasi *Tablet PC.* Pesatnya pertumbuhan Android selain faktor yang disebutkan diatas adalah karena Android itu sendiri adalah *platform* yang lengkap, baik itu sistem operasinyaa, aplikasi dan *tools* pengembangannya, *market* aplikasi android serta dukungan yang sangat tinggi dari komunitas *open source* dunia (Safaat,2012). Sehingga Android terus

berkembang pesat baik dari segi teknologi maupun dari segi jumlah *device* yang ada di dunia.

2.3.1 The Dalvik Virtual Machine (DVM)

Salah satu elemen kunci dari Android (Safaat, 2012) adalah *Dalvik Virtual Machine* (DVM). Android berjalan didalam *Dalvik Virtual Machine* (DVM) bukan di *Java Virtual Machine* (JVM), sebenarnya banyak persamaanya dengan *Java Virtual Machine* (JVM) seperti Java ME (*Java Mobile Edition*), tetapi Android menggunakan *Virtual Machine* sendiri yang dikustomisasi dan dirancang untuk memastikan bahwa beberapa fitur-fitur berjalan lebih efisien pada perangkat *mobile.*

Dalvik Virtual Machine (DVM) adalah "*register bases*" sementara *Java Virtual Machine* (JVM) adalah "*stack based*", DVM didesain dan ditulis oleh Dan Bornsten dan beberapa *engineers* Google lainnya. Jadi bisa dikatakan "*Dalvik equals(Java) == False*". *Dalvik Virtual Machine* (DVM) menggunakan kernel Linux untuk menangani fungsionalitas tingkat rendah termasuk keamanan, *threading*, dan proses serta manajemen memori. Ini memungkinkan kita untuk menulis aplikasi C / C+ sama halnya seperti pada OS Linux kebanyakan (Saffat, 2012). Meskipun dalam kenyataannya kita 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.

2.3.2 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 (Safaat, 2012). Android merupakan subset perangkat lunak untuk *smartphone* yang meliputi sistem operasi, *middleware* dan aplikasi kunci yang *direlease* 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*.

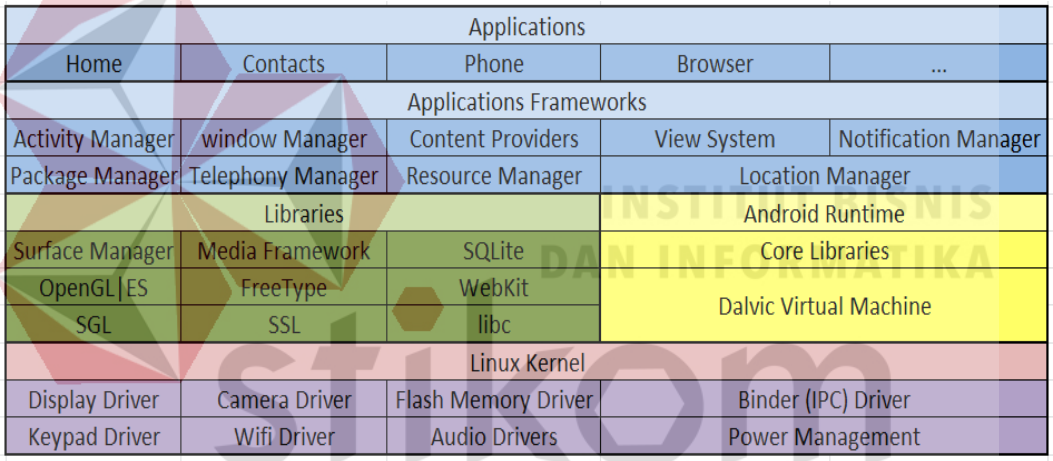
Beberapa fitur-fitur Android yang paling penting adalah:

1. Framework Aplikasi yang mendukung penggantian komponen dan *reusable*.
2. Mesin Virtual Dalvik dioptimalkan untuk perangkat *mobile*.
3. *Integrated browser* berdasarkan *engine open sourceWebKit*.
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 (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF).
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.

2.3.3 Arsitektur Android

Arsistektur 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*. Gambar 2.2 memberikan gambaran umum komponen-komponen dalam arsitektur sistem operasi Android.



Gambar 2.2 Arsitektur Android

Secara garis besar Arsitektur Android dapat dijelaskan dan digambarkan sebagai berikut:

1. *Applications* dan *Widgets*

   Applications dan *Widgets* ini adalah layar dimana berhubungan dengan aplikasi saja, dimana biasanya kita mengunduh aplikasi kemudian kita lakukan 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 pengembang 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 *notification*, dan masih banyak lagi. Pengembang memiliki akses penuh menuju API *framework* seperti yang dilakukan oleh aplikasi yang berkategori inti. Arsitektur aplikasi dirancang supaya kita dengan mudah dapat menggunakan komponen yang sudah digunakan (*reuse*).

3. *Libraries*

   Libraries ini adalah layer dimana fitur - fitur 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 dimana dalam prosesnya menggunakan implementasi Linux. *Dalvik Virtual Machine* (DVM) merupakan mesin yang membentuk dasar kerangka aplikasi Android.

5. Linux Kernel

   Linux kernel adalah layer dimana inti dari *operating* sistem dari Android itu sendiri, berisi file-file *system* yang mengatur sistem *processing*, *memory*,

resource, *drivers*, dan sistem-sistem *operating* Android lainnya. Linux Kernel yang digunakan Android adalah Linux Kernel *release* 2.6.

2.3.4 Fundamental Aplikasi Android

Aplikasi Android ditulis dalam baha pemrograman java, kode java dikompilasi bersama denga data *file resource* yang dibutuhkan oleh aplikasi dimana prosesnya di*package* oleh *tools* yang dinamakan "apk tools" ke dalam paket android sehingga menghasilkan file dengan ekstensi apk. File apk itulah yang sebenarnya kita sebut dengan aplikasi yang dapat diinstal di perangkat *mobile* nantinya.

Ada empat jenis komponen pada aplikasi Android yaitu:

1. Activities

   Suatu *activity* akan menyajikan *user interface* (UI) kepada pengguna, sehingga pengguna dapat melakukan interaksi. Sebuah aplikasi android bisa jadi hanya memiliki satu *activity*, tetapi umumnya aplikasi memiliki banyak *activity* tergantung pada tujuan aplikasi dan desain dari aplikasi tersebut. Satu *activity* biasanya akan dipakai untuk menampilkan aplikasi atau yang bertindak sebagai *user interface* (UI) saat aplikasi diperlihatkan kepada *user*. Untuk pindah dari satu *activity* ke *activity* lain kita dapat melakukan dengan satu *event* misalnya *click* tombol, memilih opsi atau menggunakan *triggers* tertentu.

2. Service

   Service tidak memiliki *visual user interface* (UI), tetapi *service* berjalan secara *background*, sebagai contoh dalam memainkan musik atau mengambil data dari jaringan, tetapi setiap *service* haruslah berada dalam kelas induknya.

Misalnya media *player* sedang memutar lagi dari *list* yang ada, aplikasi ini akan memiliki dua atau lebih *activity* yang memungkinkan *user* untuk memilih lagu misalnya, menulis sms sambil media *player* sedang jalan. Untuk menjaga musik tetap dijalankan, *activity player* dapat menjalankan *service* untuk membuat aplikasi tetap berjalan. *Service* dijalankan pada *thread* utama dari proses aplikasi.

3. Broadcast Receiver

   Broadcast receiver berfungsi menerima dan bereaksi untuk menyampaikan notifikasi. Contoh *broadcast* seperti notifikasi zona waktu berubah, baterai *low*, gambar telah selesai diambil oleh kamera, atau pengubahan referensi bahasa yang digunakan. Aplikasi juga dapat menginisiasi *broadcast* misalnya memberikan informasi pada aplikasi lain bahwa ada data yang telah diunduh ke perangkat dan siap untuk digunakan.

4. *Content Provider*

   Content provider membuat kumpulan aplikasi data secara spesifik sehingga bisa digunakan oleh aplikasi lain. Data tersimpan dalam file sistem seperti database SQLite. *Content provider* menyediakan cara untuk mengakses data yang dibutuhkan oleh suatu *activity*, misalnya ketika kita menggunakan aplikasi yang membutuhkan peta (*Map*), atau aplikasi yang membutuhkan untuk mengakses data kontak dan navigasi, maka disinilah fungsi *content provider.*

2.3.5 Versi Android

Adapun versi-versi Android yang pernah dirilis adalah sebagai berikut (Safaat,2012):

1. Android Versi 1.1

   Android versi ini dilengkapi dengan pembaruan estetis pada aplikasi, jam, alarm, *voice search* (pencarian suara), pengiriman pesan dengan Gmail, dan pemberitahuan *email*.

2. Android Versi 1.5 (*Cupcake*)

   Android pada versi ini terdapat beberapa pembaruan termasuk juga penambahan beberapa fitur dalam seluler versi ini yakni kemampuan merekam dan menonton video dengan modus kamera, meng-*upload* video ke Youtube dan gambar ke Picasa langsung dari telepon, dukungan *Bluetooth* A2DP, kemampuan terhubung secara otomatis ke *headset Bluetooth*, animasi layar, dan *keyboard* pada layar yang dapat disesuaikan dengan sistem.

3. Android Versi 1.6 (*Donut*)

   Android versi 1.6 menampilkan proses pencarian yang lebih baik dibanding sebelumnya, penggunaan baterai indikator dan kontrol *applet* VPN. Fitur lainnya adalah galeri yang memungkinkan pengguna untuk memilih foto yang akan dihapus; kamera, *camcorder* dan galeri yang diintegrasikan. CDMA/EVDO, 802.1x, VPN, Gestures, dan *text-to-speech engine*, kemampuan *dial* kontak, teknologi *text to change speech* serta pengadaan resolusi VWGA.

4. Android Versi 2.0/2.1 (*Eclair*)

   Perubahan yang dilakukan pada versi 2.0/2.1 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.

5. Android Versi 2.2 (*Froyo: Frozen Yoghurt*)

   Perubahan yang dilakukan pada versi 2.2 adalah optimalisasi kecepatan dan performa Android OS, integrasi Chrome v8 *javaScript* kedalam aplikasi *browser*, peningkatan dukungan Microsoft Exchange, peningkatan aplikasi *luncher* dengan *shotcuts* menuju aplikasi *phone* dan *browser*, pemasangan aplikasi dalam SD Card, kemampuan *WiFi Hotspot portable*, aplikasi Android Market yang telah diperbaharui dengan fitur *update* otomatis.

6. Android Versi 2.3 (*Gingerbread*)

   Perubahan pada versi 2.3 ini antara lain peningkatan kemampuan permainan (*gaming*), peningkatan layar antar muka (*User Interface*) didesain ulang, dukungan kemampuan *Near Field Communication* (NFC), dukungan jumlah kamera yang lebih dari satu, peningkatan fungsi *copy paste*.

7. Android Versi 3.0/3.1 (*Honeycomb*)

   Android *Honeycomb* dirancang khusus untuk *tablet PC*. Android ini mendukung ukuran layar yang lebih besar. *User Interface* juga berbeda karena sudah didesain untuk *tablet PC*. *Honeycomb* juga mendukung *multi procesor* dan juga akselerasi perangkat keras (*hardware*) untuk grafis.

8. Android Versi 4.0 (*ICS: Ice Cream Sandwich*)

Membawa fitur *honeycomb* untuk *smartphone* dan menambahkan fitur baru yang memaksimalkan fotografi, grafis dan resolusi gambar, dan sistem pengenalan wajah.

9. Android Versi 4.1 (*Jelly Bean*)

   Fitur terbaru pada Android versi 4.1 ini adalah peningkatan kemampuan *on-screen keyboard* yang lebih cepat serta lebih responsif, pencarian data kontak dengan fitur *voice search*.

2.4 Web Service

Web Service merupakan sistem yang dirancang untuk dapat mendukung interaksi komunikasi antar mesin-mesin pada suatu jaringan dengan menggabungkan SOAP, XML, and HTTP (Lucky,2008). Dengan teknologi *Web Service*, maka memungkinkan untuk dapat menghubungkan berbagai jenis *software* yang memiliki *platform* dan sistem operasi yang berbeda. Jadi, kita tetap mendapatkan sebuah potongan informasi dari suatu *website* tanpa harus mengunjungi *website* tersebut.

Pada *Web Services* hanya tersedia fungsi-fungsi yang nantinya dapat digunakan oleh aplikasi lainnya, jadi kita dapat mengakses potongan informasi itu dengan meletakkan fungsi/*method* itu pada aplikasi kita. Aplikasi *web service* tidak mempunyai sebuah "*User Interface*" atau tampilan *web* pada umumnya. *Web service* hanya berupa class dan method dari sebuah fungsi dan mempunyai *output* dalam format XML. *Web Service* tersimpan di *Web Server* sehingga dapat diakses oleh berbagai bahasa pemrograman dengan lebih mudah baik dalam lingkungan LAN maupun Internet. Sistem *Web Service* ini diharapkan meningkatkan kolaborasi antar pemrogram dan perusahaan, yang memungkinkan

sebuah fungsi di dalam *Web Service* dapat dipinjam oleh aplikasi lain tanpa perlu mengetahui detail pemrograman yang terdapat di dalamnya.

Web Services pada dasarnya bekerja menggunakan HTTP and SOAP untuk membuat data tersedia di dalam *Web*. SOAP dan HTTP mengizinkan *user* eksternal untuk masuk dan melakukan pemanggilan *function* secara *remote* tanpa proses registrasi dalam lingkungan internal terlebih dahulu seperti halnya aplikasi tersebut mengakses method lokal dan menggunakan format XML yang berbasis teks dalam melakukan pertukaran datanya.

2.5 UML (*Unfied Modeling language*)

2.5.1 Pengertian UML

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

2.5.2 *Artifact* UML

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

A. *Use-case diagram*

B. *Class diagram*

C. *Behavior diagram*

   i. *Statechart diagram*

ii. *Activity diagram*

iii. *Interaction diagram:*

a) *Sequence diagram*

b) *Collaboration diagram*

D. *Implementation diagram:*

i. *Component diagram*

ii. *Deployment diagram*

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

a) 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.

b) Diagram yang berbeda-beda tersebut dapat menyatakan tingkatan yang berbeda-beda dalam proses rekayasa.

c) Diagram-diagram tersebut dibuat agar model yang dibuat semakin mendekati realistas.

2.5.3 Faktor yang Mendorong Dibuatnya UML

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 kita 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 kita membangun suatu model dari suatu sistem yang kompleks, tidak mungkin kita dapat memahaminya secara keseluruhan. Dengan meningkatnya komplektisitas 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. Termasuk dalam teknik rekayasa *software* adalah teknik *visual programming*, juga pembuatan *framework* dan pola.

b) Komplektisitas dunia industri semakin meningkat karena bertambah luasnya runag 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

Sebelum adanya UML, terdapat ketidakjelasan bahasa pemodelan apa yang paling unggul. Para user harus memilih diantara bahasa pemodelan dan *tool*

(*software*) pemodelan yang banyak dan mirip UML dibuat untuk membuat integrasi baru dalam bahasa pemodelan antar *tool* dan "proses".

2.5.4 Tujuan UML

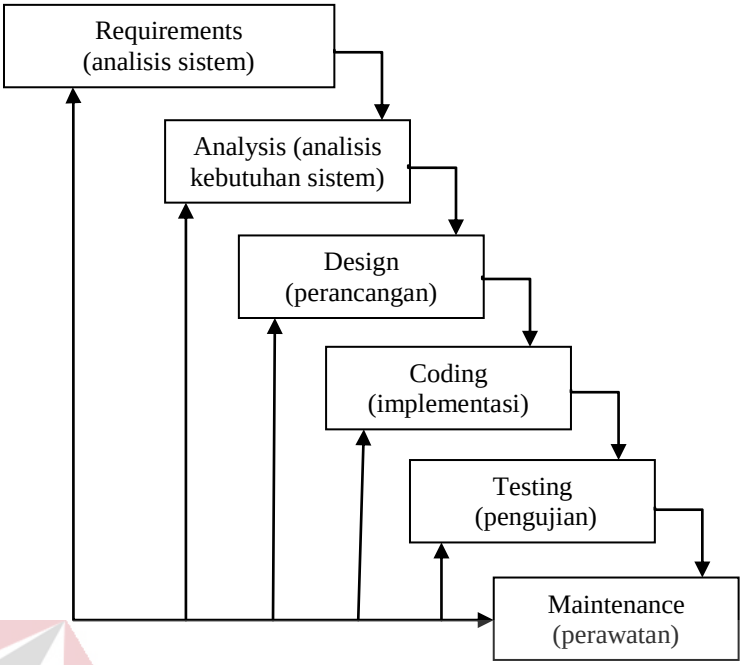
Tujuan utama UML diantaranya adalah:

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.

2.6 System Development Life Cycle (SDLC)

Model ini biasa disebut juga dengan model *waterfall* atau disebut juga *classic life cycle.* Adapun pengertian dari SDLC ini adalah suatu pendekatan yang sistematis dan berurutan. Tahapan-tahapannya adalah *Requirements* (analisis sistem), *Analysis* (analisis kebutuhan sistem), *Design* (perancangan), *Coding* (implementasi), *Testing* (pengujian) dan *Maintenance* (perawatan) (Pressman, 1997).

Model eksplisit pertama dari proses pengembangan perangkat lunak, berasal dari proses-proses rekayasa yang lain. Model ini memungkinkan proses pengembangan lebih terlihat. Hal ini dikarenakan bentuknya yang bertingkat ke bawah dari satu fase ke fase lainnya, model ini dikenal dengan model *waterfall*, seperti terlihat pada Gambar 2.3.



. Gambar 2.3 *System Development Life Cycle* (SDLC) Model Waterfall (Pressman, 1997)

Penjelasan SDLC Model*Waterfall*, adalah sebagai berikut:

a. *Requirement* (Analisis Sistem)

Dalam merancang sebuah perangkat lunak, yang pertama harus dilakukan adalah membangun semua elemen sistem yang diperlukan. Sistem merupakan hal yang penting dalam membuat sebuah perangkat lunak, karena perangkat lunak harus berhubungan langsung dengan elemen lainnya seperti perangkat keras, basis data, dan manusia. Tahap ini didefinisikan sebagai sebuah tahap yang menghasilkan sebuah kondisi yang diperlukan oleh pengguna untuk menyelesaikan permasalahan ataupunmencapai sebuah tujuan. Tahap ini bertujuan untuk mengumpulkan kebutuhan-kebutuhan pengguna dan kemudian mentransformasikan ke dalam sebuah deskripsi yang jelas dan lengkap.

b. *Analysis* (Analisis Kebutuhan Sistem)

Pada tahap ini dalam perancangan perangkat lunak, perlu mengetahui karakteristik dasar dari perangkat lunak yang akan dirancang, seperti fungsi, bentuk, dan tampilan dari perangkat lunak tersebut. Tahap analisis sistem ini bertujuan untuk menjabarkan segala sesuatu yang nantinya akan ditangani oleh perangkat lunak. Tahapan ini adalah tahapan pemodelan yang merupakan sebuah representasi *object* di dunia nyata.

c. *Design* (Perancangan)

Untuk membuat suatu perangkat lunak perlu dirancang struktur datanya, arsitektur perangkat lunak, detil prosedur dan karakteristik tampilan yang akan disajikan. Tahap perancangan perangkat lunak yang merupakan proses multi langkah dan berfokus pada beberapa atribut perangkat lunak yang berbeda, yaitu: struktur data, arsitektur perangkat lunak dan detil algoritma. Proses ini menterjemahkan kebutuhan ke dalam sebuah model perangkat lunak yang dapat diperkirakan kualitasnya sebelum memulai tahap implementasi.

d. *Coding* (Implementasi)

Rancangan yang telah dibuat dalam tahap sebelumnya akan diterjemahkan ke dalam suatu bentuk atau bahasa yang dapat dibaca dan diterjemahkan oleh komputer untuk diolah. Tahap ini juga dapat disebut dengan tahap implementasi, yaitu tahap yang mengkonversi apa yang telah dirancang sebelumnya ke dalam sebuah bahasa yang dimengerti oleh komputer. Kemudian komputer akan menjalankan fungsi-fungsi yang telah didefinisikan sehingga mampu memberikan layanan-layanan kepada penggunanya.

e. *Testing* (Pengujian)

Pengujian program dilakukan untuk mengetahui apabila terjadi kesalahan pada program yang telah dibuat. Dapat juga digunakan untuk memastikan apakah *input* proses dengan benar, sehingga dapat menghasilkan *output* yang sesuai. Tahap ini terdapat 2 metode pengujian perangkat yang dapat digunakan, yaitu: metode *black-box* dan *white-box.* Pengujian dengan metode *black-box* merupakan pengujian yang menekankan pada fungsionalitas dari sebuah perangkat lunak tanpa harus mengetahui bagaimana struktur di dalam perangkat lunak tersebut. Sebuah perangkat lunak yang diuji menggunakan metode *black-box* dikatakan berhasil jika fungsi-fingsi yang ada telah memenuhi spesifikasi kebutuhan yang telah dibuat sebelumnya. Pengujian dengan menggunakan metode *white-box* yaitu menguji struktur internal perangkat lunak dengan melakukan pengujian pada algoritma yang digunakan oleh perangkat lunak.

f. *Maintenance* (Perawatan)

Jika aplikasi tersebut telah sesuai, akan diberikan kepada pengguna dan terdapat penyesuaian atau perubahan sesuai dengan keadaan yang diinginkan, sehingga membutuhkan perubahan terhadap aplikasi tersebut. Tahap ini dapat pula diartikan sebagai tahap penggunaan perangkat lunak yang disertai dengan perawatan dan perbaikan. Perawatan dan perbaikan suatu perangkat lunak diperlukan, termasuk didalamnya adalah pengembangan, karena dalam prakteknya ketika perangkat lunak digunakan terkadang masih terdapat kekurangan ataupun penambahan fitur-fitur baru yang dirasa perlu**.**