BAB II
LANDASAN TEORI

2.1 Letak dan Luas Kota Pekanbaru

Kota Pekanbaru terletak antara 101°14' - 101°34' Bujur Timur dan 0°25' - 0°45' Lintang Utara. Dengan ketinggian dari permukaan laut berkisar 5 - 50 meter. Permukaan wilayah bagian utara landai dan bergelombang dengan ketinggian berkisar antara 5 - 11 meter.

Berdasarkan Peraturan Pemerintah No. 19 Tahun 1987 Tanggal 7 September 1987 Daerah Kota Pekanbaru diperluas dari ± 62,96 Km² menjadi ± 446,50 Km², terdiri dari 8 Kecamatan dan 45 Kelurahan/Desa. Dari hasil pengukuran/pematokan di lapangan oleh BPN Tk. I Riau maka ditetapkan luas wilayah Kota Pekanbaru adalah 632,26 Km².

Dengan meningkatnya kegiatan pembangunan menyebabkan meningkatnya kegiatan penduduk disegala bidang yang pada akhirnya meningkatkan pula tuntutan dan kebutuhan masyarakat terhadap penyediaan fasilitas dan utilitas perkotaan serta kebutuhan Lainnya. Untuk lebih terciptanya tertib pemerintahan dan pembinaan wilayah yang cukup luas, maka dibentuklan Kecamatan Baru dengan Perda Kota Pekanbaru No. 4 Tahun 2003 menjadi 12 Kecamatan dan Kelurahan/Desa baru dengan Perda tahun 2003 menjadi 58 Kelurahan/Desa.

2.2 Batas Kota Pekanbaru

Kota Pekanbaru berbatasan dengan daerah Kabupaten/Kota :

a. Sebelah Utara : Kabupaten Siak dan Kabupaten Kampar
b. Sebelah Selatan : Kabupaten Kampar dan Kabupaten Pelalawan
c. Sebelah Timur : Kabupaten Siak dan Kabupaten Pelalawan
d. Sebelah Barat : Kabupaten Kampar.

2.3 Sungai Kota Pekanbaru

Kota Pekanbaru dibelah oleh Sungai Siak yang mengalir dari barat ke timur. Memiliki beberapa anak sungai antara lain : Sungai Umban Sari, Air Hitam, Siban, Setukul, Pengambang, Ukui, Sago, Senapelan, Limau, Tampan dan Sungai Sail. Sungai Siak juga merupakan jalur perhubungan lalu lintas perekonomian rakyat pedalaman ke kota serta dari daerah lainnya.

2.4 Iklim Kota Pekanbaru

Kota Pekanbaru pada umumnya beriklim tropis dengan suhu udara maksimum berkisar antara 34,1º C - 35,6º C dan suhu minimum antara 20,2º C - 23,0º C Curah hujan antara 38,6 - 435,0 mm/tahun dengan keadaan musim berkisar :

a. Musim hujan jatuh pada bulan Januari s/d April dan September s/d Desember.
b. Musim Kemarau jatuh pada bulan Mei s/d Agustus.

2.5 Daftar Satuan Kerja Perangkat Daerah (SKPD) di Lingkungan Pemerintah Kota Pekanbaru.

Tabel 2.1 Daftar Satuan Kerja Perangkat Daerah (SKPD)

No	Nama Satuan Kerja Perangkat Daerah	Alamat	Telepon
1.	Sekretariat Daerah	Jl. Jend. Sudirman No. 464	(0761) 31543
2.	Sekretariat DPRD	Jl. Jend. Sudirman	(0761) 21661
3.	Inspektorat	Jl. Mustafa Sari	(0761) 35066
4.	Badan Kepegawaian Daerah	Jl. Jend. Sudirman No. 464	(0761) 31543
5.	Badan Lingkungan Hidup	Jl. Pepaya	(0761) 855850
6.	Badan Kesatuan Bangsa,	Jl. Cut Nyak Dien	(0761) 35071

	Politik dan Perlindungan Masyarakat		
7.	Badan Penanaman Modal dan Promosi	Jl. Arifin Ahmad	(0761) 39399, 39499
8.	Badan Perencanaan Pembangunan Daerah	Jl. Jend. Sudirman No. 464	(0761) 20098
9.	Badan Pelayanan Terpadu	Jl. Jend. Sudirman No. 464	(0761) 28262
10.	Badan Pemberdayaan Perempuan, Masyarakat dan Keluarga Berencana	Jl. Puyuh	(0761)
11.	Kantor Perpustakaan Arsip	Jl. Jend. Sudirman No. 464	(0761) 859318
12.	Kantor Satuan Polisi Pamong Praja	Jl. Jend. Sudirman No. 464	(0761) 31543
13.	Dinas Pendidikan	Jl. Patimura	(0761) 42788
14.	Dinas Kesehatan	Jl. Melur	(0761) 23213
15.	Dinas Sosial dan Pemakaman	Jl. Jend. Sudirman	(0761) 22602
16.	Dinas Tenaga Kerja	Jl. Kapling I	(0761) 21264
17.	Dinas Perhubungan Komunikasi dan Informatika	Jl. Sutomo	(0761) 21819
18.	Dinas Kependudukan dan Pencatatan Sipil	Jl. Mustafa Sari	(0761) 35463
19.	Dinas Kebudayaan dan Pariwisata	Jl. Arifin Ahmad	(0761) 39184
20.	Dinas Pekerjaan Umum	Jl. Parit Indah	(0761) 571524

21.	Dinas Tata Ruang dan Bangunan	Jl. Cut Nyak Dien	(0761) 40516
22.	Dinas Koperasi Usaha Mikro, Kecil dan Menengah	Jl. Teratai	(0761) 21462
23.	Dinas Perindustrian dan Perdagangan	Jl. Teratai	(0761) 21669
24.	Dinas Pertanian	Jl Ibrahim Sattah No. 30	(0761) 26095
25.	Dinas Kebersihan dan Pertamanan	Jl. Datuk Setia Maharaja	(0761) 31516
26.	Dinas Pasar	Jl. Dagang	(0761) 26360, 26361
27.	Dinas Pemadam Kebakaran	Jl. Teratai	(0761) 22382
28.	Dinas Pendapatan Daerah	Jl. Teratai	
29.	Dinas Pemuda dan Olahraga	Jl. kuantan II	(0761) 849661
30.	Kecamatan Tampan	Jl. Soebrantas No. 52	(0761) 63317
31.	Kecamatan Payung Sekaki	Jl. Arengka No. 56	
32.	Kecamatan Bukit Raya	Jl. Kaharudin Nasution No. 37	(0761) 674683
33.	Kecamatan Marpoyan Damai	Jl. Arifin Ahmad	
34.	Kecamatan Tenayan Raya	Jl. Hang Tuah	
35.	Kecamatan Lima Puluh	Jl. S. Syarif Qasim No. 132	(0761) 21720
36.	Kecamatan Sail	Jl. Mulyarejo No. 6	(0761) 35840
37.	Kecamatan Pekanbaru Kota	Jl. Tuanku Umar No. 20	(0761) 20550

38.	Kecamatan Sukajadi	Jl. Jend. A. Yani No. 198	(0761) 21086
39.	Kecamatan Senapelan	Jl. Panglima Undan No. 47	(0761) 22046
40.	Kecamatan Rumbai	Jl. Sri Indra Pekanbaru	(0761) 52795
41.	Kecamatan Rumbai Pesisir	Jl. Sekolah No. 3	(0761) 51080
42.	Kelurahan Simpang Baru	Jl. Hr Soebrantas No. 52	
43.	Kelurahan Sidomulyo Barat	Jl. Purawodadi	
44.	Kelurahan Tuah Karya	Jl. Budi Daya Gg. Budi Lestari	
45.	Kelurahan Delima	Jl. Delima	
46.	Kelurahan Labuh Baru Timur	Jl. Adi Sucipto Pekanbaru	
47.	Kelurahan Tampan	Jl. H. Subrantas	
48.	Kelurahan Air Hitam	Jl. Riau Ujung Pekanbaru	
49.	Kelurahan Labuh Baru Barat	Jl. musyawarah Pekanbaru	
50.	Kelurahan Simpang Tiga	Jl. Unggas Pekanbaru	
51.	Kelurahan Tangkerang Selatan	Jl. Harapan raya	
52.	Kelurahan Tangkerang Utara	Jl. Sakuntala	
53.	Kelurahan Tangkerang Labuai	Jl. Kelapa Sawit Pekanbaru	
54.	Kelurahan Tangkerang Tengah	Jl. Garuda Pekanbaru	
55.	Kelurahan Tangkerang Barat	Jl. Todak Pekanbaru	
56.	Kelurahan Maharatu	Jl. Kaharuddin Nasution	

57.	Kelurahan Sidomulyo Timur	Jl. Adi Sucipto Pekanbaru	
58.	Kelurahan Wonorejo	Jl. Paus Pekanbaru	
59.	Kelurahan Kulim	Jl. Lintas Timur / Hangtuah	
60.	Kelurahan Tangkerang Timur	Jl. Mangga Besar Pekanbaru	
61.	Kelurahan Rejosari	Jl. Pendopo Pekanbaru	
62.	Kelurahan Sail	Jl. Raya Hangtuah No.275	
63.	Kelurahan Rintis	Jl. Syarif Kasim No. 128	
64.	Kelurahan Sekip	Jl. Kuantan I no 2	
65.	Kelurahan Tanjung Rhu	Jl. Hijrah Sumber Sari No. 44 Pekanbaru	
66.	Kelurahan Pesisir	Jl. Syari Kasim gg Selamat No 44 Pekanbaru	
67.	Kelurahan Cinta Raja	Jl. Pattimura No. 42	
68.	Kelurahan Sukamaju	Jl. Sukamaju II	
69.	Kelurahan Sukamulia	Jl. Hang Tuah No. 32	(0761) 41878
70.	Kelurahan Simpang Empat	Jl. Rupat Pekanabru	(0761) 20069
71.	Kelurahan Sumahilang	Jl. Hang Tuah No 68	(0761) 22536
72.	Kelurahan Tanah Datar	Jl. Muslimin Pekanbaru	(0761) 22762
73.	Kelurahan Kota Baru	Jl. A Yani	(0761) 23708
74.	Kelurahan Sukaramai	Jl. Nilam Pekanbaru	(0761) 23564
75.	Kelurahan Kota Tinggi	Jl. Gatot Subroto	

76.	Kelurahan Jadirejo	Jl. Kusuma No 13	(0761) 32982
77.	Kelurahan Kampung Tengah	Jl. Ketitiran Pekanbaru	
78.	Kelurahan Kampung Melayu	Jl. Nuri Pekanbaru	
79.	Kelurahan Kedung Sari	Jl. Melur Pekanbaru	
80.	Kelurahan Harjo Sari	Jl. Pelanduk No 25	
81.	Kelurahan Sukajadi	Jl. Kamboja gg Surya No. 99	(0761) 848986
82.	Kelurahan Pulau Karam	Jl. Teratai Bawah Pekanbaru	(0761) 37499
83.	Kelurahan Padang Bulan		
84.	Kelurahan Padang Terubuk	Jl. Melur Ujung Pekanbaru	
85.	Kelurahan Sago	Jl. Samratulangi	
86.	Kelurahan Kampung Dalam	Jl. Kampung Dalam No.54	(0761) 22365
87.	Kelurahan Kampung Bandar	Jl. Merbau Pekanbaru	
88.	Kelurahan Kampung Baru	Jl. Jati Pekanbaru	
89.	Kelurahan Umban Sari	Jl. Umban Sari Atas	
90.	Kelurahan Rumbai Bukit	Jl. Sri Indra Pekanbaru	
91.	Kelurahan Muara Fajar		
92.	Kelurahan Palas	Jl. Siak II Pekanbaru	
93.	Kelurahan Sri Meranti		
94.	Kelurahan Meranti Pandak	Jl. yos Sudarso No. 2	(0761) 52975
95.	Kelurahan Limbungan	Jl. Sembilang Pekanbaru	

96.	Kelurahan Lembah Sari	Jl. Pramungka Pekanbaru	
97.	Kelurahan Lembah Damai	Jl. Lembah Damai Pekanbaru	
98.	Kelurahan Limbungan Baru	Jl. Muara Fajar Perumnas Rmbai Pekanbaru	
99.	Kelurahan Tebing Tinggi Okura	Jl. Utama Pekanbaru	

2.6 Pengertian Aplikasi

Suatu subkelas perangkat lunak komputer yang memanfaatkan kemampuan komputer langsung untuk melakukan suatu tugas yang diinginkan pengguna. Biasanya dibandingkan dengan perangkat lunak sistem yang mengintegrasikan berbagai kemampuan komputer, tapi tidak secara langsung menerapkan kemampuan tersebut untuk mengerjakan suatu tugas yang menguntungkan pengguna. Contoh utama perangkat lunak aplikasi adalah pengolah kata, lembar kerja, dan pemutar media. Aplikasi dapat digolongkan menjadi beberapa kelas, antara lain:

1. Perangkat lunak perusahaan (*enterprise*)
2. Perangkat lunak infrastruktur perusahaan
3. Perangkat lunak informasi kerja
4. Perangkat lunak media dan hiburan
5. Perangkat lunak pendidikan
6. Perangkat lunak pengembangan media
7. Perangkat lunak rekayasa produk

Pada pengertian umumnya, aplikasi adalah alat terapan yang difungsikan secara khusus dan terpadu sesuai kemampuan yang dimilikinya.. (http://id.wikipedia.org/wiki/Aplikasi).

Sedangkan menurut Dhanta (2009:32), aplikasi (*application*) adalah software yang dibuat oleh suatu perusahaan komputer untuk mengerjakan tugas-tgas tertentu,

misalkan *Microsoft Word, Microsoft Excel*. Sedangkan menurut Anisyah (2000:30), aplikasi adalah penerapan, penggunaan, atau penambahan.
(http://blog.binadarma.ac.idnayelwp-contentuploads2010...BAB-II.pdf).

2.7 Pengertian Android

Android merupakan salah satu *Operating System* (OS) *mobile* terpopuler saat ini. Maka dari itu saya akan memberikan sedikit penjelasan tentang sejarah, fitur-fitur dan jenis-jenis android yang saat ini beredar di pasaran. Yang pasti tulisan dibawah ini berisi informasi tentang android.

Android adalah sistem operasi yang berbasis Linux untuk telepon seluler seperti *Smartphone* dan komputer tablet. Android menyediakan platform terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri untuk digunakan oleh bermacam piranti bergerak. Pada awalnya, *Google Inc*. mengakusisi Android Inc., yaitu pendatang baru yang membuat peranti lunak untuk ponsel. Kemudian untuk mengembangkan android tersebut, dibentuklah *Open Handset Alliance*, yaitu konsorsium dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi, termasuk diantaranya ialah *Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia.*

Pada saat perilisan perdana Android, yaitu pada tanggal 5 November 2007, Android bersama *Open Handset Alliance* menyatakan mendukung pengembangan standar terbuka pada perangkat seluler. Di lain pihak, *Google* merilis kode–kode Android di bawah lisensi *Apache*, sebuah lisensi perangkat lunak dan standar terbuka perangkat seluler.

Di dunia saat ini terdapat dua jenis distributor sistem operasi Android:

1. Distributor yang mendapat dukungan penuh dari *Google* atau *Google Mail Services* (GMS),
2. Distributor yang benar–benar bebas distribusinya tanpa dukungan langsung *Google* atau dikenal sebagai *Open Handset Distribution* (OHD).

Pada Juli 2005, Google bekerjasama dengan Android Inc., perusahaan yang membuat peranti lunak untuk ponsel yang berada di Palo Alto, California Amerika Serikat. Para pendiri Android Inc. yaitu Andy Rubin, Rich Miner, Nick Sears, dan Chris White bekerja pada Google. Saat itu banyak yang menganggap fungsi Android Inc. hanyalah sebagai perangkat lunak pada telepon seluler. Sejak saat itu muncul rumor bahwa Google akan memasuki pasar telepon seluler. Di Google, tim yang dipimpin Andy Rubin bertugas mengembangkan program perangkat seluler yang didukung oleh kernel Linux. Hal ini menunjukkan indikasi bahwa Google sedang bersiap menghadapi persaingan dalam pasar telepon seluler.

Sekitar September 2007 sebuah studi melaporkan bahwa Google mengajukan hak paten aplikasi telepon seluler, dan akhirnya Google mengenalkan Nexus One, salah satu jenis Smartphone GSM yang menggunakan Android pada sistem operasinya. Smartphone ini diproduksi oleh *HTC Corporation* dan tersedia di pasaran pada 5 Januari 2010.

Pada tanggal 9 Desember 2008, diumumkan anggota baru yang bergabung dalam program kerja Android *ARM Holdings, Atheros Communications, diproduksi oleh Asustek Computer Inc, Garmin Ltd, Softbank, Sony Ericsson, Toshiba Corp, dan Vodafone Group Plc.* Seiring pembentukan *Open Handset Alliance*, OHA mengumumkan produk perdana mereka, Android, perangkat bergerak (*mobile*) yang merupakan modifikasi kernel Linux 2.6. Sejak Android dirilis telah dilakukan berbagai pembaruan berupa perbaikan bug dan penambahan fitur baru. Telepon pertama yang memakai sistem operasi Android adalah *HTC Dream*, yang dirilis pada 22 Oktober 2008.

Fitur-fitur yang tersedia di android:

a. Kerangka aplikasi: itu memungkinkan pengguna untuk melakukan penggunaan dan penghapusan komponen yang tersedia.
b. Dalvik mesin virtual: mesin virtual dioptimalkan untuk perangkat telepon seluler.
c. Grafik: grafik di 2D dan grafis 3D berdasarkan pustaka OpenGL.

d. SQLite: untuk penyimpanan data.
e. Mendukung media: audio, video, dan berbagai format gambar (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
f. GSM, Bluetooth, EDGE, 3G, 4G dan WiFi
g. Kamera, Global Positioning System (GPS), kompas, NFC dan accelerometer.

Versi Android yang Pertama hingga yang terbaru ;

a. Android Versi 1.1 (Rilis 9 Maret 2009).
b. Android Versi 1.5 (Cupcake – Rilis Pertengahan Mei 2009)
c. Android versi 1.6 (Donut – Rilis September 2009)
d. Android Versi 2.0/2.1 (Éclair – Rilis 3 Desember 2009)
e. Android Versi 2.2 (Froyo: Frozen Yoghurt – Rilis 20 Mei 2010)
f. Androis Versi 2.3 (Gingerbread – Rilis 6 Desember 2010)
g. Android Versi 3.0/3.1 (Honeycomb – Rilis Mei 2011)
h. Android Versi 4.0 (ICS : Ice Cream Sandwich – Rilis 19 Oktober 2011)
i. Android versi 4.1 (Jelly Bean – Rilis 27 Juni 2012)
j. Android versi 4.2 (A New Flavor of Jelly Bean – Rilis 13 November 2012).

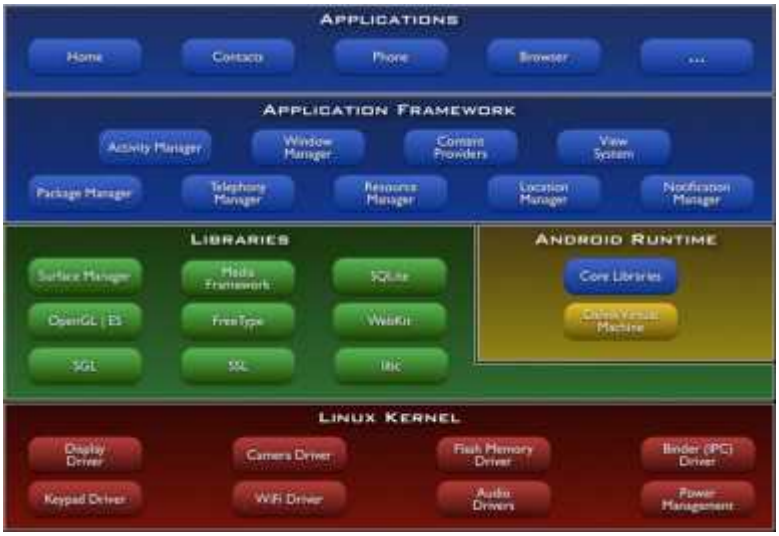
2.8 Fitur dan Arsitektur Android

Beberapa fitur-fitur Android yang paling penting adalah (Safaat 2012) :

1. *Fremework* Aplikasi yang mendukung penggatian 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 opengI ES 1,0 (Opsional akselerasi *hardware*).
5. SQLite untuk penyimpanan data.
6. *Media Support* yang mendukung audio, video, dan gambar (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF), GSM *Telephony* (tergantung *hardware*).
7. *Bluetooth*, EDGE, 3G, dan WiFi (tergantung *hardware*).

8. Kamera, GPS, kompas, dan *accelerometer* (tergantung *hardware*).
9. Lingkungan *Development* yang lengkap dan kaya termasuk perangkat *emulator*, *tools* untuk *debugging*, profil dan kinerja memori, dan *plugin* untuk IDE *Eclipse*.

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



Gambar 2.1 Arsitektur Android

1. *Applications* dan *Widgets*.

   Applications dan *Widgets* ini adalah *layer* di mana kita berhubungan dengan aplikasi saja, di mana biasanya kita *download* aplikasi kemudian kita lakukan instalasi dan jalankan aplikasi tersebut. Di *layer* terdapat aplikasi inti termasuk klien email, program SMS, kalender, peta, browser, kontak, dan lain-lain. Semua aplikasi ditulis menggunakan bahasa pemograman Java.

2. *Applicartions Frameworks*

   Android adalah “Open Development Platform” yaitu Android menawarkan kepada pengembang atau memberi kemampuan kepada pengembang untuk membangun aplikasi yang bagus dan inovatif. Pengembang memiliki akses

penuh menuju API *framework* seperti yang dilakukan oleh aplikasi yang kategori inti. Arsitektur aplikasi dirancang supaya mudah menggunakan kembali komponen yang sudah digunakan (*reuse*).

Komponen-komponen yang termasuk di dalam *Applications Frameworks* adalah sebagai berikut :

a. *Views.*
b. *Content Provider.*
c. *Resource Manager.*
d. *Notification Manager.*
e. *Activity Manager.*

3. *Libraries*

Adalalah *layer* dimana fitur-fitur Android berada, biasanya para pembuat aplikasi mengakses libraries untuk menjalankan aplikasinya. Berjalan di atas kernel, *Layer* ini meliputi berbagai *library* C/C++ inti seperti Libc dan SSL, serta :

a. *Libraries* media untuk pemutaran media audio dan video.
b. *Libraries* untuk manajemen tampilan.
c. *Libraries Graphics* mencangkup SGL dan OpenGL untuk grafis 2D dan 3D.
d. *Libraries* SQLite untuk dukungan *database*.
e. *Libraries* SSL dan *WebKit* terintegrasi dengan *web browser* dan *security*.
f. *Libraries* LiveWebcore mencangkup modern *web browser* dengan *engine embeded web view*.
g. *Libraries* 3D yang mencangkup implementasi OpenGL ES 1.0 API’s.

4. *Android Run Time*

   Di dalam Android *Run Time* dibagi menjadi dua bagian yaitu :

   a. *Core Libraries* : Aplikasi Android dibangun dalam bahasa java, sementara Dalvik sebagai virtual mesinnya bukan Virtual Machine Java, sehingga diperlukan sebuah *libraries* yang berfungsi untuk menterjemahkan bahasa java/c yang ditangani oleh *Core Libraries.*

   b. *Dalvik Virtual Machine* : Virtual mesin berbasis register yang dioptimalkan untuk menjalankan fungsi-fungsi secara efisien, di mana merupakan pengembangan yang mampu membuat linux kernel untuk melakukan *threading* dan manajemen tingkat rendah.

   c. Linux Kernel

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

2.9 Java

Beberapa definisi Java adalah sebagai berikut :

1. Java sebagai bahasa pemograman

   Sebagai sebuah bahasa pemograman, Java dapat membuat seluruh aplikasi, *desktop, web* dan lainnya. Sebagaimana dibuat dengan menggunakan baasa pemograman konvensional yang lain. Java adalah bahasa pemograman berorientasi objek (OOP) dan dapat dijalankan pada berbagai *platform* sistem operasi. Perkembangan java tidak hanya terfokus pada satu sistem operasi, tetapi dikembangkan untuk berbagai sistem operasi dan bersifat *open source.*

2. Java sebagai *Development Environment*

   Sebagai sebuah peralatan pembangun, teknologi Java menyediakan banyak *tools: compiler, interpreter,* penyusun dokumentasi, paket kelas dan sebagainya.

3. Java sebagai sebuah aplikasi

   Aplikasi dengan teknologi Java secara umum adalah aplikasi serba guna yang dapat dijalankan pada seluruh mesin yang memiliki *Java Runtime Environment* (JRE).

4. Java sebagai *Deployment Environment*

   Terdapat dua komponen utama dari *Deployment Environment*. Yang pertama adalah JRE, yang terdapat pada paket J2SDK, mngandung kelas-kelas untuk semua paket teknologi Java yang meliputi kelas dasar dari Java, komponen GUI dan sebagainya. Komponen yang lain terdapat pada Web Browser. Hampir seluruh web browser komersial menyediakan *interpreter* dan runtime *environment* dari teknologi Java.

2.10 XML *(Extensible Markup Language)*

XML adalah sebuah *meta-language* untuk mendeskripsikan data. XML merupakan sebuah cara mempresentasikan data tanpa tergantung kepada sistem. XML juga dapat digunakan sebagai *extension markup languages*. XML berbasis *text*, sehingga ia dapat dengan mudah dipindahkan dari satu sistem komputer ke sistem yang lain.

Secara sederhana XML merupakan bahasa berbasis penandaan (*tag*) untuk mendeskripsikan data atau informasi tanpa memperdulikan aplikasi yang kelak akan digunakannya. Hal ini cukup kontras dibanding HTML yang mendefenisikan bagaimana data atau informasi ditampilkan dan sangat bergantung pada aplikasi apa (misal *browser*) yang akan menggunakannya. Meski demikian aplikasi yang akan menggunakan XML harus tahu aturan yang berlaku dalam pembuatan berkas XML agar aplikasi tersebut mampu memanfaatkannya. XML memungkinkan kita untuk

membuat struktur kita sendiri, tidak seperti HTML yang menggunakan struktur yang bersifat baku (Nugroho, 2008).

Android menyediakan model pembuatan *User Interface* (UI) dengan file *layout* berbasis XML. Struktur umum dari sebuah file layout Android sangat sederhana. Dimana terdiri dari elemen XML disusun mirip hierarki pohon, dan setiap node adalah nama dari class *View*. Kita bisa menggunakan apa saja nama class tampilan yang kita atur sendiri dalam kode. Dengan metode ini, kita bisa membuat *layout UI* dengan cepat, karena menggunakan struktur dan sintaks yang lebih sederhana dari pada layout dengan metode *programmatic*. Model ini terinspirasi dari pengembangan web, dimana kita bisa memisahkan pengeturan tampilan (UI) dari logika aplikasi yang digunakan untuk mengisi, mengolah dan menampilkan data.

2.11 *Eclipse*

Eclipse adalah sebuah IDE (*Integrated Development Environment*) untuk mengembangkan perangkat lunak dan dapat dijalankan di semua platform (*platform-independent*). Berikut ini adalah sifat dari Eclipse (http://id.wikipedia.org, 2013) :

1. *Multi-platform* : Taget sistem operasi Eclipse adalah Microsoft Windows, Linux,, Solaris, AIX, HP-UX, dan Mac OS X.
2. *Multi*-language : Eclipse dikembangkan dengan bahasa pemograman Java, akan tetapi Eclipse mendukung pengembangan aplikasi berbasis bahasa pemograman lain, seperti, C/C++, Cobol, Phython, Perl, PHP, dan lain sebagainya.
3. *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 pada saat ini merupakan salah satu IDE favorit dikarenakan gratis dan *open source*, yang berarti setiap orang boleh melihat kode pemrograman perangkat lunak ini. Selain itu, kelebihan dari Eclipse yang membuatnya populer adalah

kemampuannya untuk dapat dikembangkan oleh pengguna dengan komponen yang dinamakan *plug-in*.

Eclipse awalnya dikembangkan oleh IBM untuk menggantikan perangkat lunak IBM Visual Age for Java 4.0. Produk ini diluncurkan oleh IBM pada tanggal 5 November 2001, yang menginvestasikan sebanyak US$ 40 juta untuk pengembangannya. Semenjak itu konsursium Eclipse Foundation mengambil alih untuk pengembangan Eclipse lebih lanjut dan pengaturan organisasinya.

Sejak versi 3.0, Eclipse pada dasarnya merupakan sebuah *kernel*, yang mengangkat *plug-in*. Apa yang dapat digunakan di dalam Eclipse sebenarnya adalah fungsi dari *plug-in* yang sudah diinstal. Ini merupakan basis dari Eclipse yang dinamakan *Rich Client Platform* (RCP). Berikut ini adalah komponen yang membentuk RCP (www.id.wikipedia.org, 2013) :

a. *Core platform*
b. OSGi
c. SWT (*Standard Widget Toolkit*)
d. Jface
e. *Eclipse Workbench*

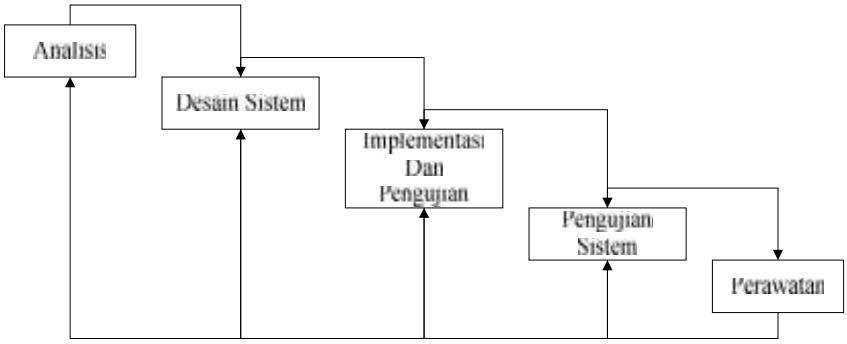
Secara standar Eclipse selalu dilengkapi dengan JDT (*Java Development Tools*), *plug-in* yang membuat Eclipse kompatibel untuk mengembangkan program Java, dan PDE (*Plug-in Development Environment*) untuk mengembangkan *plug-in* baru. Eclipse beserta *plug-in*-nya diimplementasikan dalam bahasa pemrograman Java.

Konsep Eclipse adalah IDE yang terbuka (*open*), mudah diperluas (*extensible*) untuk apa saja, dan tidak untuk sesuatu yang spesifik. Jadi, Eclipse tidak saja untuk mengembangkan program Java, akan tetapi dapat digunakan untuk berbagai macam keperluan, cukup dengan menginstal *plug-in* yang dibutuhkan. Apabila ingin mengembangkan program C/C++ terdapat *plug-in* CDT (*C/C++ Development Tools*). Selain itu, pengembangan secara visual bukan hal yang tidak mungkin oleh Eclipse,

plug-in UML2 tersedia untuk membuat diagram UML. Dengan menggunakan PDE setiap orang bisa membuat *plug-in* sesuai dengan keinginannya. Salah satu situs yang menawarkan *plug-in* secara gratis seperti *Eclipse downloads by project*.

2.12 Model SDLC *Waterfall*

Model SDLC air terjun (*waterfall*) atau bisa dikatakan sequensial linear (*sequential linear*) atau alur hidup klasik (*classic life cycle*). Model air terjun menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau terurut dimulai dari analisis, desain, pengodean, pengujian, dan tahap pendukung (*support*). Berikut adalah gambar model air terjun :



Gambar 2.2. Ilustrasi model waterfall

a. Analisis kebutuhan perangkat lunak

   Proses pengumpulan kebutuhan dilakukan secara intensif untuk menspesifikasikan kebutuhan perangkat lunak seperti apa yang dibutuhkan oleh *user*.

b. Desain

   Desain perangkat lunak adalah proses multilangkah yang fokus pada desain pembuatan program perangkat lunak termasuk struktur data, arsitektur perangkat lunak, representasi antar muka, dan prosedur pengodean. Tahap ini mentranslasi kebutuhan perangkat lunak dari tahap analisis kebutuhan ke representasi desain agar dapat di implementasikan menjadi program pada

tahap selanjutnya. Desain perangkat lunak pada tahap ini perlu di dokumentasikan.

c. Pembuatan kode program

   Desain harus di translasikan ke dalam program perangkat lunak. Hasil dari tahap ini adalah program komputer sesuai dengan desain yang telah di buat pada tahap desain.

d. Pengujian

   Pengujian fokus pada perangkat lunak secara dari segi lojik dan fungsional dan memastikan bahwa semua bagian sudah diuji. Hal ini dilakukan untuk meminimalisir kesalahan dan memastikan keluaran yang dihasilkan sesuai dengan yang di inginkan.

e. Pendukung atau pemeliharaan

   Tidak menutup kemungkinan sebuah perangkat lunak mengalami perubahan ketika sudah dikirimkan ke *user*. Perubahan bisa terjadi karena adanya kesalahan yang muncul dan tidak terdeteksi saat pengujian atau perangkat lunak harus beradaptasi denga lingkungan baru. Tahap pendukung atau pemeliharaan dapat mengulangi proses pengembangan mulai dari analisis spesifikasi untuk perubahan peerangkat lunak yang sudah ada, tapi tidak untuk membuat perangkat lunak baru.

2.13 UML *(Unified Modeling Language)*

2.13.1 Sejarah perkembangan UML

Uml pertama kali diperkenalkan oleh Ivar Jacobson (yang sebelumnya terkenal dengan konsep OOSE-*Object Oriented Software Engineering*) serta Grady Booch (yang sebelumnya terkenal dengan notasi *Booch* yang populer digunakan sebagai salah satu metodologi analisis dan perancangan berorientasi object). Uml pertama kali diperkenalkan pada tahun 1990-an ketika Grady Booch dan Ivar Jacobson mulai mengadopsi ide-ide serta kemampuan-kemampuan tambahan dari

masing-masing metodanya dan berusaha membuat metodologi terpadu yang kemudian dinamakan UML (Nugroho, 2005).

2.13.2 Fokus UML

Secara umum UML merupakan bahasa untuk visualisasi, specifikasi, kontruksi, serta dokumentasi. Dalam kerangka visualisasi, para pengembang menggunaka UML sebagai suatu cara untuk mengkomunikasikan idenya kepada pemrogram serta calon pengguna sistem/perangkat lunak. Dengan adanya bahasa yang bersifat standar, komunikasi perancang dengan pemrogram (lebih tepat lagi: komunikasi antar anggota kelompok pengembang) serta calon pengguna diharapkan menjadi mulus.

UML menyediakan beberapa diagram visual yang menunjukkan berbagai aspek dalam sistem, ada beberapa diagram yang disediakan dalam UML, antara lain : (Nogroho, 2005).

a. *Use Case Diagram*
b. *Activity Diagram*
c. *Sequential Diagram*
d. *Collaboration Diagram*
e. *Class Diagram*
f. *Statechart Diagram*
g. *Component Diagram*
h. *Deployment Diagram*

Tabel 2.2 Tipe Diagram *UML*

Diagram	**Tujuan**
Use Case	Menunjukkan sekumpulan kasus fungsional dan aktor dan hubungannya.
Activity	Pandangan operasi, bagaimana objek-objek bekerja, aksi-aksi yang

	mempengaruhi obyek, pandangan *use case workflow*.
Sequence	Berfungsi untuk *overview* prilaku sistem, menunjukkan objek-objek yang diperlukan, mendokumentasikan skenario dari suatu diagram *Use Case*, memeriksa jalur-jalur pengaksesan.
Class	Memodelkan kosakata di sistem, distribusi dan tanggung jawab, tipe primitif, kolaborasi, skema *database* logik.
Collaboration	Memodelkan pandangan prilaku sistem pada *link-link* di antara objek-objek. Ilustrasi dari *use case*, memeriksa jalur-jalur pengaksesan
Statechart	Pandangan objek secara waktu, pandangan dalam berkaitan dengan ransangan eksternal.
Component	Memodelkan *file* yang dapat dieksekusi dan pustaka, memodelkan tabel, *file* dan dokumen, memodelkan API (*Application Programming Interupt*)
Deployment	Konfigurasi pemprosesan saat jalan dan komponen-komponen yang terdapat didalamnya.

Sumber: Nugroho (2005)

2.13.3 Tool Yang Mendukung UML

Saat ini banyak sekali *tool* pendesainan yang mendukung UML, baik itu *tool* komersial maupun *opensource*. Beberapa diantaranya adalah *Rational Rose***,** *Together, Object Domain, Jvision, Objecteering, MagicDraw* **,** *Visual Object Modeller*

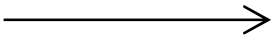
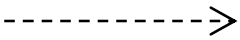
2.13.4 Diagram-Diagram *UML* Yang Digunakan

Adapun diagram yang sering digunakan dalam UML adalah :

1. *Use case Diagram (UC)*

   Diagram Use case merupakan salah satu diagram untuk memodelkan aspek perilaku sistem. Masing-masing diagram use case menunjukkan sekumpulan use case, aktor, dan hubungannya. Diagram use case adalah penting untuk memvisualisasikan, memspesifikasikan, dan mendokumentasikan kebutuhan perilaku sistem. Diagram use case merupakan pusat pemodelan perilaku sistem, subsistem, kelas. Berikut adalah elemen dalam use case :

Tabel 2.3 Notasi *Use Case* Diagram

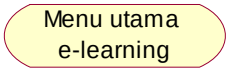
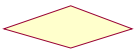
Penjelasan	***Notasi UML***
Orang, *prose,* sistem lain yang berinteraksi dengan sistem informasi yang akan di buat di luar sistem informasi itu sendiri, jadi walupun simbol aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakn menggunakan kata benda diawal *frase* nama aktor	 *Nama aktor*
Use Case : Abstraksi dari interaksi antara sistem dan aktor	
Association : adalah *abstraksi* dari penghubung antara actor dan *use case*	
Generalisasi : menunjukkan spesialisasi aktor untuk dapat berpartisipasi dalam use *case*	

Sumber: Rosa, dkk (2005)

2. *Activity Diagram*

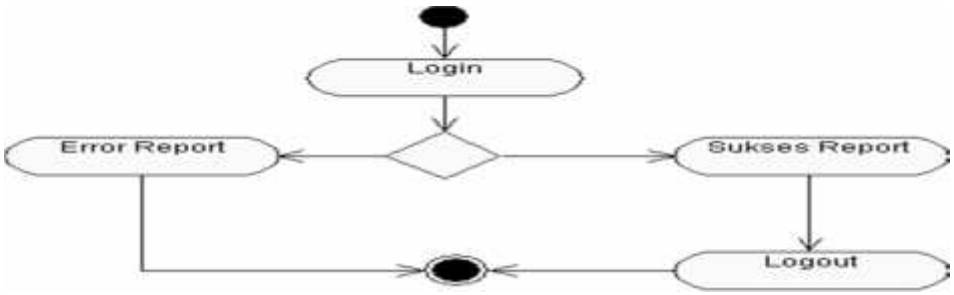
Pada dasarnya. Diagram aktivitas adalah Diagram *flowchart* yang diperluas yang menunjukkan aliran kendali satu aktivitas ke aktivitas lain. Kegunaan diagram ini adalah untuk memodelkan *workflow* atau jalur kerja, memodelkan operasi, bagaimana objek-objek bekerja, aksi-aksi dan pengaruh terhadap objek. Simbol-simbol yang terdapat dalam *Activity Diagram*, sebagai berikut :

Tabel 2.4 Simbol *Activity Diagram*

Keterangan	**Simbol**
Titik Awal atau permulaan.	
Titik Akhir atau akhir dari aktivitas.	
Aktivity, atau aktivitas yang dilakukan oleh aktor.	
Decision, atau pilihan untuk mengambil keputusan.	
Arah tanda panah alur proses.	

Sumber: Nugroho (2005)

Activity diagram merupakan salah satu diagram yang umum digunakan dalam *UML* untuk menjabarkan proses atau aktivitas dari aktor. Sebagai contoh, pelanggan melakukan *login* (masuk) pada halaman *website* untuk bergabung, jika pelanggan belum terdaftar, maka akan ditolak oleh sistem dan dikembalikan. Proses penjabarannya adalah sebagai berikut :



Gambar 2.3 *Activity Diagram* (Sumber: Nogroho, 2005)

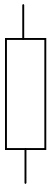
Di dalam *Activity* diagram tersebut dijelaskan bahwa *user* melakukan proses *login* untuk dapat memasuki area sistem, jika proses *login* dan/atau *user* belum teregistrasi, maka *user* akan ditolak oleh sistem tersebut dan diberi pesan *error*. Selain itu, bila *user* telah teregistrasi dan memasukkan kode *login* dengan benar maka pakan diberi akses untuk masuk ke sistem, dan diberikan pesan sukses. *User* dapat *logout* (keluar) untuk mengakhiri sesi.

3. *Sequence diagram*

   Sequence diagram mendokumentasikan komunikasi/interaksi antar kelas-kelas. Diagram ini menunjukkan sejumlah obyek dan *message* (pesan) – yang diletakkan diantara obyek-obyek didalam *use case*. Perlu diingat bahwa di dalam diagram ini, kelas-kelas dan aktoraktor diletakkan dibagian atas diagram dengan urutan dari kiri ke kanan dengan garis *lifeline* yang diletakkan secara vertikal terhadap kelas dan aktor. Berikut adalah notasi-notasinya.

Tabel 2.5. *Notasi Sequence Diagram*

Object	*Object* merupakan instance dari sebuah class dan dituliskan tersusun secara horizontal.Digambarkan sebagai sebuah class (kotak) dengan nama obyek didalamnya yang diawali dengan sebuah titik koma	: Object1

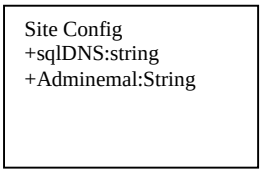
Actor	*Actor* juga dapat berkomunikasi dengan object, maka actor juga dapat diurutkan sebagai kolom. Simbol Actor sama dengan simbol pada Actor Use Case Diagram.	
Lifeline	*Lifeline* mengindikasikan keberadaan sebuah object dalam basis waktu. Notasi untuk Lifeline adalah garis putus-putus vertikal yang ditarik dari sebuah obyek.	
Activation	*Activation* dinotasikan sebagai sebuah kotak segi empat yang digambar pada sebuahlifeline. Activation mengindikasikan sebuah obyek yang akan melakukan sebuah aksi.	
Massage	*Message*, digambarkan dengan anak panah horizontal antara Activation.Message mengindikasikan komunikasi antara object-object.	

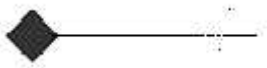
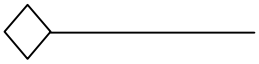
Sumber: Nugroho (2005)

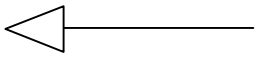
4. *Class Diagram*

   Class Diagram adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah obyek dan merupakan inti dari pengembangan dan desain berorientasi obyek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). Berikut adalah notasi – notasi yang ada pada *class diagram* :

Tabel 2.6 Notasi pada Class Diagram

Class	*Class* adalah blok - blok pembangun pada pemrograman berorientasi obyek. Sebuah class digambarkan sebagai sebuah kotak yang terbagi atas 3 bagian. Bagian atas adalah bagian nama dari *class*. Bagian tengah mendefinisikan property/atribut *class*. Bagian akhir mendefinisikan method-method dari sebuah *class*.	
Assosiation	Sebuah asosiasi merupakan sebuah *relationship* paling umum antara 2 class, dan dilambangkan oleh sebuah garis yang menghubungkan antara 2 *class*. Garis ini bisa melambangkan tipe-tipe *relationship* dan juga dapat menampilkan hukum-hukum multiplisitas pada sebuah *relationship* (Contoh: One-to-one, one-to-many, many-to-many).	1..n Owned by 1
Composition	Jika sebuah *class* tidak bisa berdiri sendiri dan harus merupakan bagian dari *class*	

	yang lain, maka *class* tersebut memiliki relasi *Composition* terhadap *class* tempat dia bergantung tersebut. Sebuah *relationship composition* digambarkan sebagai garis dengan ujungberbentuk jajaran genjang berisi/solid.	
Dependency	Kadangkala sebuah *class* menggunakan *class* yang lain. Hal ini disebut *dependency*. Umumnya penggunaan *dependency* digunakan untuk menunjukkan operasi pada suatu *class* yang menggunakan *class* yang lain. Sebuah *dependency* dilambangkan sebagai sebuah panah bertitik-titik.	
Aggregation	*Aggregation m*engindikasikan keseluruhan bagian *relationship* dan biasanya disebut sebagai relasi "mempunyai sebuah" atau "bagian dari". Sebuah *aggregation* digambarkan sebagai sebuah garis dengan sebuah jajaran genjang yang	

	tidak berisi/tidaksolid.	
Generalization	Sebuah relasi *generalization* sepadandengan sebuah relasi *inheritance* pada konsep berorientasi obyek. Sebuah *generalization* dilambangkan dengan sebuah panah dengan kepala panah yang tidak solid yang mengarah ke kelas "*paren*t"-nya/induknya.	

5. *Collaboration Diagram*

   Collaboration diagram menggunakan prinsip yang sama dengan *sequence diagram*, sama-sama memodelkan interaksi antar obyek-obyek, yang membedakannya hanya cara penggambarannya saja. Pada collaboration diagram ini, obyek-obyek dan *message* (pesan) yang ada digambarkan mirip seperti flowchart, hanya saja, untuk menjaga urutan pesan yang diterima oleh masing-masing obyek, pesan-pesan tersebut diberi nomor urutan pesan. Berikut adalah notasi untuk *collaboration diagram* :

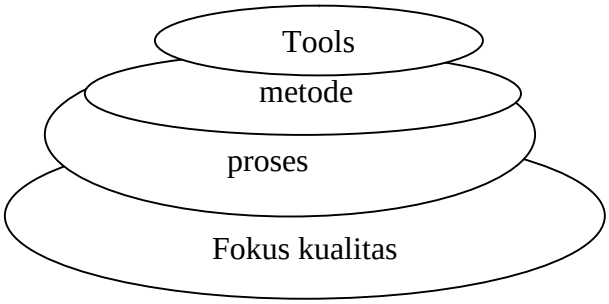
Tabel 2.7 Notasi collaboration diagram

Object	*Object* merupakan instance dari sebuah class. Digambarkan sebagai sebuah class (kotak) dengan nama obyek didalamnya yang diawali dengan sebuah titik koma.	: Object1
Actor	*Actor* juga dapat berkomunikasi dengan object, maka actor juga dapat disertakan ke dalam collaboration diagram. Simbol Actor sama dengan simbol pada Actor Use Case Diagram.	
Message	*Message*, digambarkan dengan anak panah yang mengarah antar obyek dan diberi label urutan nomor yang mengindikasikan urutan komunikasi yang terjadi antar obyek.	**Message** →

Sumber: Nogroho (2005)

2.14 Metodologi Pengembangan Rekayasa Perangkat Lunak

Pengembangan perangkat lunak dapat diartikan sebagai proses membuat suatu perangkat lunak baru untuk menggantikan perangkat lunak lama secara keseluruhan atau memperbaiki perangkat lunak yang telah ada. Agar lebih cepat dan tepat dalam mendeskripsikan solusi dan mengembangkan perangkat lunak, juga hasilnya mudah dikembangkan dan dipelihara, maka pengembangan perangkat lunak memerlukan suatu metodologi khusus. Metodologi pengembangan perangkat lunak adalah suatu proses pengorganisasian kumpulan metode dan konvensi notasi yang telah didefinisikan untuk mengembangkan perangkat lunak. Secara prinsip bertujuan untuk membantu menghasilkan perangkat lunak yang berkualitas. Berikut batu landasan yang menopang rekayasa perangkat lunak *(Rogers S. Pressman, 2002:28):*



Gambar 2.4 Metodologi Pengembangan

Metodologi pengembangan perangkat lunak (atau disebut juga model proses atau paradigma rekayasa perangkat lunak) adalah suatu strategi pengembangan yang memadukan proses, metode, dan perangkat (*tools*). Metode-metode rekayasa perangkat lunak, memberikan teknik untuk membangun perangkat lunak. Berkaitan dengan serangkaian tugas yang luas yang menyangkut analisis kebutuhan, konstruksi program, desain, pengujian, dan pemeliharaan *(Rogers S. Pressman:2002).*

Untuk menyelesaikan masalah di dalam pengembangan perangkat lunak, tim perekayasa harus menggabungkan strategi pengembangan yang melingkupi lapisan proses, metode, dan alat bantu. Model proses rekayasa perangkat lunak dipilih berdasarkan sifat aplikasi dan proyeknya, metode dan alat-alat bantu yang akan dipakai, dan control serta penyampaian yang dibutuhkan.

2.15 *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 (Imaniar,2011)