BAB II
LANDASAN TEORI

2.1 Teknologi *Smartphone*

Telepon cerdas *(smartphone)* adalah telepon genggam yang mempunyai kemampuan tingkat tinggi, kadang-kadang dengan fungsi yang menyerupai komputer. Belum ada standar pabrik yang menentukan arti telepon cerdas. Bagi beberapa orang, telepon pintar merupakan telepon yang bekerja menggunakan seluruh perangkat lunak sistem operasi yang menyediakan hubungan standar dan mendasar bagi pengembang aplikasi.

Bagi yang lainnya, *smartphone* hanyalah merupakan sebuah telepon yang menyajikan fitur canggih seperti *email*, internet dan kemampuan membaca buku elektronik (*e-book).* Dengan kata lain, *smartphone* merupakan komputer kecil yang mempunyai kemampuan sebuah telepon. Pertumbuhan permintaan akan alat canggih yang mudah dibawa kemana-mana membuat kemajuan besar dalam pemroses, ngingatan, layar dan sistem operasi yang di luar dari jalur telepon genggam sejak beberapa tahun ini. (*Yuniar Supardi,2011)*

Seperti yang dikutip dari website *http://www.AnneAhira.com*, berkembangnya ilmu pengetahuan dan teknologi secara tidak langsung telah mendorong berbagai perusahaan yang bergerak di dunia bisnis teknologi untuk menciptakan berbagai produk unggulan. Tidak terkecuali untuk perusahaan besar pencipta ponsel seperti Nokia dan perusahaan lainnya. Lahirnya handphone dirasa masih memiliki sejumlah kekurangan, meskipun fitur-fitur canggih telah dimasukkan ke dalamnya. Dengan begitu perlu adanya alat yang jauh lebih canggih dibanding dengan *handphone* agar perkembangan teknologi dapat dirasakan dengan lebih maksimal.

Alat yang lebih canggih dibanding dengan *handphone* tersebut dinamakan *smartphone*. *Smartphone* adalah sebuah ponsel yang menawarkan kemampuan komputasi yang lebih maju dan konektivitas dari dasar kontemporer fitur telepon. *Smartphone* dan fitur ponsel dapat dianggap sebagai komputer genggam yang

terintegrasi dalam telepon seluler, tetapi ketika fitur ponsel hanya mampu untuk menjalankan aplikasi berbasis pada platform seperti Java ME atau BREW, smartphone memungkinkan pengguna untuk menginstal dan menjalankan aplikasi lebih maju berdasarkan platform tertentu.
(*http://www.anneahira.com/teknologi-smartphone.htm*)

Seperti nama nya *Smartphone* berarti *handphone* pintar. Yang menjadikan nya beda terhadap *handphone* biasa dan PDA adalah :

1. Kemudahan dalam hal pengoperasian.
2. Sistem Operasi (OS).
3. Desain Fisik.
4. Kualitas Audio.
5. Masa Pakai Baterai.
6. Dan Fitur kegunaan (utility).

2.2 Java

Java adalah bahasa pemrograman yang dapat dijalankan diberbagai komputer termasuk telepon genggam. Aplikasi-aplikasi berbasis java umumnya dikompilasi ke dalam p-code (*bytecode)* dan dapat dijalankan pada berbagai mesin virtual java (JVM). Java merupakan bahasa pemrograman yang bersifat umum, dan secara khusus didesain untuk memanfaatkan dependensi implementasi seminimal mungkin.

Karena fungsionalnya yang memungkinkan aplikasi java mampu berjalan di beberapa platform sistem operasi yang berbeda, java dikenal pula dengan slogannya “Tulis sekali, jalankan dimanapun”. Saat ini java merupakan bahasa pemrograman yang paling populer digunakan, dan secara luas dimanfaatkan dalam pengembangan berbagai jenis perangkat lunak aplikasi ataupun aplikasi berbasis web.

JSON (*JavaScript Object Notation*)

Menurut Seto El Kahfi, JSON adalah struktur data yang universal, dalam artian bisa digunakan dalam berbagai bahasa pemograman. Hampir semua bahasa

pemrograman mendukung penuh JSON dalam berbagai format. Hal ini memungkinkan format data yang dapat dipertukarkan menggunakan bahasa pemograman juga menggunakan dasar dari struktur JSON.
Format data JSON mempunyai aturan sebagai berikut:

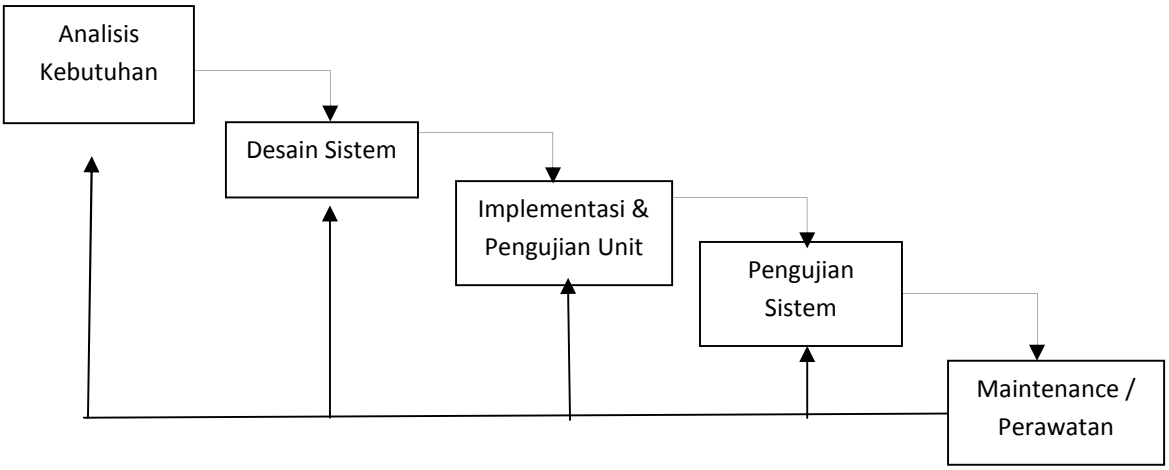
1. ***Object*** adalah satu set nama/nilai yang tidak terurut. Penulisan object dimulai dengan tanda { (*left brace*) dan diakhiri dengan tanda } (*right brace*). Setiap nama diikuti oleh tanda : (colon) dan pasangan nama/nilai dipisahkan dengan tanda , (koma).
2. ***Array*** adalah sekumpulan nilai yang teratur. Penulisan sebuah *array* dimulai dengan tanda [(*left bracket*) dan diakhiri dengan tanda] (*right bracket*). Nilai dipisahkan menggunakan tanda , (*koma*).
3. **Nilai** bisa berupa *string* dalam tanda kutip, atau *number* (angka), *TRUE* atau *FALSE* atau *NULL*, sebuah *object* atau sebuah *array*. Struktur ini dapat ditulis menggunakan metode bersarang.
4. ***String*** adalah rangkaian atau urutan karakter *unicode* yang berada dalam tanda kutip, bisa juga hanya berisi karakter kosong, menggunakan tanda \ (*backslash*) untuk *escape*. Karakter di representasikan sebagai *string* tunggal. Tipe data *string* pada JSON sangat mirip dengan definisi pada bahasa C atau Java.
5. ***Number*** sangat mirip dengan definisi pada bahasa C atau Java, hanya saja tipe bilangan oktal dan heksadesimal tidak digunakan.

2.3 Pengembangan Sistem Informasi

2.3.1 *System Development Life Cycle* (SDLC)

Pengembangan sistem informasi merupakan proses atau prosedur yang harus diikuti untuk melaksanakan seluruh langkah dalam menganalisis, merancang, mengimplementasikan dan memelihara sistem informasi. Proses-proses pengembangan ini dikenal dengan daur hidup pengembangan sistem atau

SDLC (*system Development Life Cycle*). SDLC yang terkenal adalah SDLC model klasik yang biasa disebut dengan model *waterfall.*



Gambar 2.1 SDLC *Waterfall* (Sumber : Mulyanto, 2009).

Pada fase analisis kebutuhan ini seorang analis sistem menentukan kebutuhan secara lengkap, kemudian dianalisis dan didefinisikan kebutuhan-kebutuhan yang harus dipenuhi oleh sistem yang akan dikembangkan. Pada fase ini harus dikerjakan secara lengkap sehingga akan menghasilkan desain yang lengkap. Biasanya kualitas informasi yang didapat dari fase analisis kebutuhan atau analisis sistem sangat mempengaruhi kualitas sistem yang dikembangkan.

Setelah kebutuhan dikumpulkan secara lengkap, informasi mengenai kebutuhan-kebutuhan tersebut diubah kedalam data berorientasi object dengan menggunakan beberapa alat (*tools*) seperti UML. Kemudian pada fase implementasi, desain sistem diterjemahkan kedalam kode-kode dengan menggunakan bahasa pemrograman yang sudah ditentukan. Kemudian dilakukan pengujian terhadap unit-unit yang dihasilkan. Pada fase pengujian sistem, unit-unit tersebut disatukan dan dilakukan pengujian secara keseluruhan. Kemudian dilakukan pengoperasian sistem pada lingkungan yang sebenarnya dan dilakukan perawatan atau pemeliharaan terhadap sistem tersebut.

2.3.2 Analisis Sistem

Analisis sistem adalah penguraian dari suatu sistem informasi yang utuh kedalam bagian-bagian komponennya dengan maksud untuk mengidentifikasikan dan mengevaluasi permasalahan, kesempatan, hambatan yang terjadi dan kebutuhan yang diharapkan sehingga dapat diusulkan perbaikannya.

Untuk membangun atau mengembangkan sistem informasi harus dilakukan penyelidikan dan analisis mengenai alasan timbunya ide atau gagasan untuk membangun atau mengembangkan sistem informasi. Dalam mengembangkan sistem informasi, beberapa hal yang harus diperhatikan adalah kebutuhan terhadap suatu sistem yang akan dikembangkan. Kebutuhan terhadap sistem yang akan dikembangkan ini meliputi keepatan dan ketepatan pengolahan informasi pada sistem lama dan sistem baru. Selain kecepatan dan ketepatan sistem dalam mengolah informasi, perlu juga diperhatikan mengenai keamanan yang dimiliki oleh sistem yang akan dikembangkan.

2.3.3 Desain Sistem

Menurut Sommerville, desain sistem adalah penggambaran, perencanaan dan pembuatan sketsa atau pengaturan beberapa elemen yang terpisah kedalam satu kesatuan yang utuh dan berfungsi (Mulyanto, 2009).

Dalam melakukan pengembangan sistem, perlu dipahami kondisi-kondisi yang ada sekarang dan masa datang. Dengan demikian, dalam melakukan pengembangan, akan memenhi kebutuhan-kebutuhan organisasi pada masa yang akan datang. Pemahaman tentang kondisi tersebut akan menghasilkan ide-ide yang dapat digunakan sebagai pemecah masalah yang mungkin akan muncul dimasa yang akan datang. Ide-ide tersebut kemudian diubah menjadi desain-desain yang akan memenuhi tujuan dari pengembangan sistem tersebut. Tujuan desain sistem adalah untuk menghasilkan suatu model atau representasi dari entitas yang kemudian akan dibangun.

2.3.4 Implementasi Sistem

Implementasi merupakan pengembangan dari tahap desain sistem. Tahap implementasi mencakup :

1. Pengkodean atau yang lebih dikenal dengan pemrograman merupakan kegiatan analisis kebutuhan yang dimengerti oleh computer dengan menggunakan bahasa pemrograman. Pemrograman merupakan metodologi pemecahan masalah, kemudian menuangkannya dalam suatu notasi tertentu yang mudah dibaca dan dipahami.
2. Pengujian

   Pengujian atau testing merupakan proses pengeksekusian program untuk menemukan kesaahan-kesalahan yang terdapat didalam sistem, kemudian dilakukan pembenahan. Tahap ini merupakan tahap yang penting dalam pengembangan sistem karena pada tahap ini merupakan tahapan untuk memastikan bahwa suatu sistem terbebas dari kesalahan.
3. Dokumentasi

   Dokumentasi penting untuk dilaksanakan, karena dapat digunakan untuk penelusuran jika terjadi kesalahan. Dokumentasi merupakan kegiatan melakukan pencatatan terhadap setiap langkah pekerjaan dari awal pembuatan program sampai akhir pembuatan program.

2.4 Object Oriented Analisys and Design (OOAD)

OOAD adalah metode analisis yang memeriksa *requirements* dari sudut pandang kelas dan objek yang ditemui dalam ruang lingkup permasalahan yang mengarahkan arsitektur software yang didasarkan pada manipulasi objek-objek sistem atau subsistem. OOAD merupakan cara baru dalam memikirkan suatu masalah dengan menggunakan model yang dibuat menurut konsep sekitar dunia nyata. Dasar pembuatan adalah objek, yang merupakan kombinasi antara struktur data dan perilaku dalam satu entitas. (Mulyanto, 2009).

2.4.1 Metodologi dalam OOAD

Metodologi adalah cara sistematis untuk mengerjakan analisys and design. Dengan metodologi, pihak yang membangun system software dapat merencanakan dan mengulangi pekerjaan dilain waktu. Metodologi juga menghilangkan perbedaan notasi untuk suatu hal yang sama karena setiap oarng akan berbicara dalam bahasa yang sama.

a. Teknik Pemodelan dalam OOAD

Teknik pemodelan objek menggunakan tiga macam model untuk menggambarkan sistem, diantaranya adalah sebagai berikut :

1. Model Objek :

   Model objek Menggambarkan struktur statis dari suatu objek dalam sistem dan relasinya, model objek berisi diagram objek. Diagram objek adalah graph dimana nodenya adalah kelas yang mempunyai relasi antar kelas.

2. Model Dinamik :

   Model dinamik menggambarkan aspek dari sistem yang berubah setiap saat. Model dinamik dipergunakan untuk menyatakan aspek kontrol dari sistem. Model dinamik berisi state diagram. State diagram adalah graph dimana nodenya adalah state dan arc adalah transisi antara state yang disebabkan oleh event.

3. Model Fungsional :

   Model fungsional menggambarkan transformasi nilai data di dalam sistem. Model fungsional berisi data flow diagram.

b. Karateristik dari Objek

Adapun kareakteristik dan objek yaitu :

a. Objek

Objek adalah benda secara fisik dan konseptual yang ada di sekitar kita. Beberapa contoh objek, misalnya hardware, software, dokumen, manusia, konsep, dan lainnya. Untuk kepentingan pemodelan, misalnya seorang eksekutif akan melihat karyawan, gedung, divisi,

dokumen, keuntungan perusahaan sebagai sebuah objek. Sedangkan seorang teknisi mobil, akan melihat ban, pintu, mesin, kecepatan tertentu dan banyaknya bahan baker sebagai sebuah objek. Contoh lainnya adalah seorang software engineer akan memandang tumpukan, antrian intruksi, window, check box sebagai sebuah objek. Sebuah objek mempunyai keadaan sesaat yang disebut state.

b. Kelas Objek

Kelas adalah definisi umum (pola, template, atau cetak biru) dari himpunan objek yang sejenis. Kelas menetapkan spesifikasi perilaku (*behaviour*) dan atribut-atribut dari objek tersebut. Class adalah abstraksi dari entitas dalam dunia nyata. Sedangkan objek adalah contoh ("*instances*") dari sebuah kelas. Misalnya, atribut dari kelas binatang adalah berkaki empat dan mempunyai ekor. Perilakunya adalah makan dan tidur. Sedangkan contoh (*instance*) untuk kelas binatang ini adalah kucing, gajah, dan kuda.

Adapun Istilah-istilah dari Objek yaitu :

a) Atribut : Atribut menggambarkan data yang dapat memberikan informasi mengenai kelas atau objek dimana atribut tersebut berada.
b) Operasi : Fungsi di dalam kelas yang dikombinasikan ke bentuk tingkah laku kelas
c) State dari sebuah objek : adalah kondisi dari objek itu atau himpunan keadaan yang menggambarkan objek tersebut. Sebagai contoh, state dari rekening tabungan, dapat memuat saldo yang berjalan, state dari sebuah jam adalah catatan saat itu; sedangkan state dari sebuah bohlam lampu adalah suatu keadaan "nyala" atau "mati".
d) Metode : Pelaksanaan prosedur (badan dari kode yang mengeksekusi respon terhadap permintaan objek lain di dalam sistem).

c. Karakteristik Metodologi Berorientasi Objek

Metodologi pengembangan sistem berorientasi objek mempunyai tiga karakteristik utama :

1) ***Encapsulation* (Pengkapsulan)**

Encapsulation merupakan dasar untuk pembatasan ruang lingkup program terhadap data yang diproses. Data dan prosedur atau fungsi dikemas bersama-sama dalam suatu objek, sehingga prosedur atau fungsi lain dari luar tidak dapat mengaksesnya. Data terlindung dari prosedur atau objek lain, kecuali prosedur yang berada dalam objek itu sendiri.

2) ***Inheritance* (Pewarisan)**

Pewarisan adalah teknik yang menyatakan bahwa anak dari objek akan mewarisi data/atribut dan metode dari induknya langsung. Atribut dan metode dari objek induk diturunkan kepada anak objek, demikian seterusnya. Pewarisan mempunyai arti bahwa atribut dan operasi yang dimiliki bersama di antara kelas yang mempunyai hubungan secara hirarki. Suatu kelas dapat ditentukan secara umum, kemudian ditentukan spesifik menjadi subkelas. Setiap subkelas mempunyai hubungan atau mewarisi semua sifat yang dimiliki oleh kelas induknya, dan ditambah dengan sifat unik yang dimilikinya. Kelas Objek dapat didefinisikan atribut dan service dari kelas Objek lainnya. Inheritance menggambarkan generalisasi sebuah kelas.

3) ***Polymorphism (Polimorfisme* – perbedaan bentuk*)***

Polimorfisme yaitu konsep yang menyatakan bahwa seuatu yang sama dapat mempunyai bentuk dan perilaku berbeda. *Polimorfisme* mempunyai arti bahwa operasi yang sama mungkin mempunyai perbedaan dalam kelas yang berbeda. Kemampuan objek-objek yang berbeda untuk melakukan metode yang pantas dalam merespon message yang sama. Seleksi dari metode yang sesuai bergantung pada kelas yang seharusnya menciptakan Objek.

2.4.5 OOA (*Object Oriented Analysis*)

OOA mempelajari permasalahan dengan menspesifikasikannya atau mengobservasi permasalahn tersebut dengan menggunakan metode berorientasi objek. Biasanya analisa sistem dimulai dengan adanya dokumen permintaan (*requirement*) yang diperoleh dari semua pihak yang berkepentingan. (Mis: klien,developer, pakar, dll). Dokumen permintaan memiliki 2 fungsi yaitu : memformulasikan kebutuhan klien dan membuat suatu daftar tugas

Analisis berorientasi obyek (OOA) melihat pada domain masalah, dengan tujuan untuk memproduksi sebuah model konseptual informasi yang ada di daerah yang sedang dianalisis. Model analisis tidak mempertimbangkan kendala-kendala pelaksanaan apapun yang mungkin ada, seperti konkurensi, distribusi, ketekunan, atau bagaimana sistem harus dibangun. Kendala pelaksanaan ditangani selama desain berorientasi objek (OOD). Analisis dilakukan sebelum Desain.

Sumber-sumber untuk analisis dapat persyaratan tertulis pernyataan, dokumen visi yang formal, wawancara dengan stakeholder atau pihak yang berkepentingan lainnya. Sebuah sistem dapat dibagi menjadi beberapa domain, yang mewakili bisnis yang berbeda, teknologi, atau bidang yang diminati, masing-masing dianalisis secara terpisah.

Hasil analisis berorientasi objek adalah deskripsi dari *apa* sistem secara fungsional diperlukan untuk melakukan, dalam bentuk sebuah model konseptual. Itu biasanya akan disajikan sebagai seperangkat menggunakan kasus, satu atau lebih UML diagram kelas, dan sejumlah diagram interaksi.

Tujuan dari analisis berorientasi objek adalah untuk mengembangkan model yang menggambarkan perangkat lunak komputer karena bekerja untuk memenuhi seperangkat persyaratan yang ditentukan pelanggan. (Mulyanto, 2009).

2.4.6 OOD (*Object Oriented Design*)

OOD mengubah model konseptual yang dihasilkan dalam analisis berorientasi objek memperhitungkan kendala yang dipaksakan oleh arsitektur yang dipilih dan setiap non-fungsional – teknologi atau lingkungan – kendala,

seperti transaksi *throughput, response time, run* – waktu *platform,* lingkungan pengembangan, atau bahasa pemrograman.

2.5 UML *(Unified Modeling Language)*

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. 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.

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

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 2.1 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)

3. Perangkat 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*

4. Diagram-Diagram *UML* Yang Digunakan

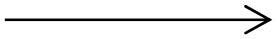
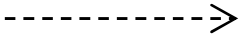
Adapun diagram yang sering digunakan dalam UML adalah :

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 2.2. Notasi *Use Case* Diagram

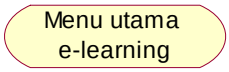
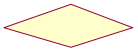
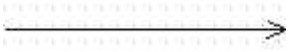
Penjelasan	***Notasi UML***
Orang, prose, sistem lain yang berinteraksi dengan sistem informasi yang akan di buat di luar sistem informasi itu sendiri, jadi walupun simbol aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakn menggunakan kata benda diawal frase nama actor	 *Nama aktor*
Use Case : Abstraksi dari interaksi antara sistem dan actor	 Membaca

Association : adalah abstraksi dari penghubung antara actor dan use case	
Generalisasi : menunjukkan spesialisasi actor untuk dapat berpartisipasi dalam use case	

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 :

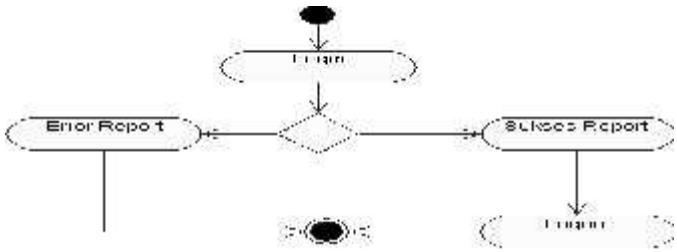
Tabel 2.3. Simbol *Activity Diagram*

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 2.2 *Activity Diagram* (Sumber: Nogroho, 2005).

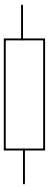
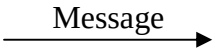
Di dalam *Activity* diagram tersebut dijelaskan bahwa *user* melakukan proses *login* untuk dapat memasuki area sistem, jika proses *login* dan/atau *user* belum teregistrasi, maka *user* akan ditolak oleh sistem tersebut dan diberi pesan *error*. Selain itu, bila *user* telah teregistrasi dan memasukkan 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 2.4. *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

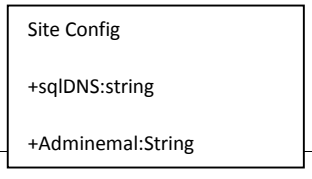
Actor	*Actor* juga dapat berkomunikasi dengan object, maka actor juga dapat diurutkan sebagai kolom. Simbol Actor sama dengan simbol pada Actor Use Case Diagram.	
Lifeline	*Lifeline* mengindikasikan keberadaan sebuah object dalam basis waktu. Notasi untuk Lifeline adalah garis putus-putus vertikal yang 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.	

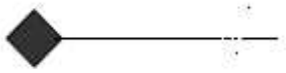
Sumber: Nugroho (2005)

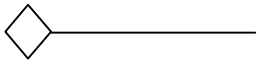
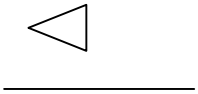
4. *Class Diagram*

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 2.5. 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* mengindikasikan 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 2.6 Notasi collaboration diagram

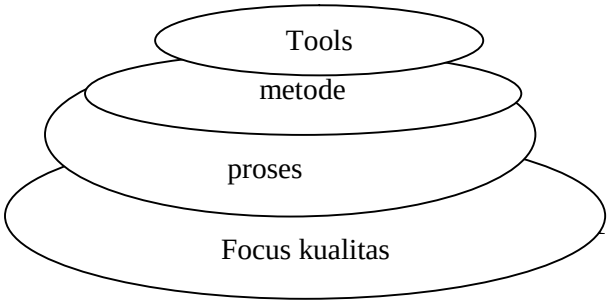
Object	*Object* merupakan instance dari sebuah class. Digambarkan sebagai sebuah class (kotak) dengan nama obyek didalamnya yang diawali dengan sebuah titik koma.	: Object1
Actor	*Actor* juga dapat berkomunikasi dengan object, maka actor juga dapat disertakan ke dalam collaboration diagram. Simbol Actor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

Sumber: Nogroho (2005).

2.6 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 2.3 : Metodologi Pengembangan

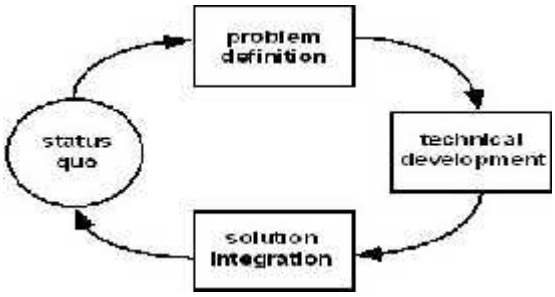
Metodologi pengembangan perangkat lunak (atau disebut juga model proses atau paradigma rekayasa perangkat lunak) adalah suatu strategi pengembangan yang memadukan proses, metode, dan perangkat (*tools*). Metode-metode rekayasa perangkat lunak, memberikan teknik untuk membangun perangkat lunak. Berkaitan 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.

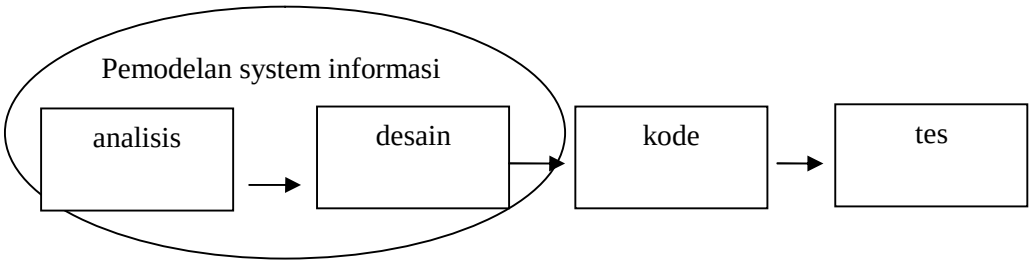
Berikut Metode-Metode Pengembangan Rekayasa Perangkat Lunak *(Rogers S. Pressman:2002).*

1. Model Sekuensial Linier

Sering juga disebut dengan “siklus kehidupan klasik” atau “model air terjun.” Penggambaran model ini :



Gambar 2.4 : Fase lingkaran pemecahan masalah



Gambar 2.5 : Model sekuensial linier

Metode pengembangan perangkat lunak dengan pendekatan pada perkembangan perangkat lunak yang sistematik dan sekuensial yang mulai pada tingkat dan kemajuan sistem pada seluruh analisis, desain, kode, pengujian, dan pemeliharaan. Model ini melingkupi aktivitas-aktivitas :

a. Rekayasa dan pemodelan sistem/informasi

   Perangkat lunak adalah bagian dari sistem yang lebih besar, pekerjaan dimulai dari pembentukan kebutuhan-kebutuhan untuk seluruh elemen sistem dan kemudian memilah mana yang untuk pengembangan perangkat lunak. Hal ini penting, ketika perangkat lunak harus berkomunikasi dengan *hardware*, orang dan basis data.

b. Analisis kebutuhan perangkat lunak

   Pengumpulan kebutuhan dengan fokus pada perangkat lunak, yang meliputi: Domain informasi, fungsi yang dibutuhkan, unjuk kerja/performansi dan antarmuka. Hasilnya harus didokumentasi dan di*review* ke pelanggan.

c. Desain

Ada 4 atribut untuk program yaitu : Struktur Data, Arsitektur perangkat lunak, Prosedur detil dan Karakteristik Antarmuka. Proses desain mengubah kebutuhan-kebutuhan menjadi bentuk karakteristik yang dimengerti perangkat lunak sebelum dimulai penulisan program. Desain ini harus terdokumentasi dengan baik dan menjadi bagian konfigurasi perangkat lunak.

d. Generasi kode

Penterjemahan perancangan ke bentuk yang dapat dimengerti oleh mesin, dengan menggunakan bahasa pemrograman

e. Pengujian

Setelah kode program selesai testing dapat dilakukan. Testing memfokuskan pada logika internal dari perangkat lunak, fungsi eksternal dan mencari segala kemungkinan kesalahan dan memriksa apakah sesuai dengan hasil yang diinginkan.

f. Pemeliharaan

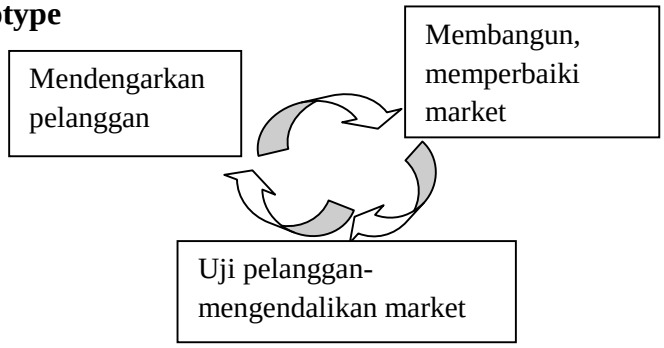
Merupakan bagian paling akhir dari siklus pengembangan dan dilakukan setelah perangkat lunak dipergunakan. Pemeliharaan perangkat lunak mengaplikasikan lagi Setiap fase program sebelumnya dan tidak membuat yang baru. Kegiatan :

1. *Corrective Maintenance* : Mengoreksi kesalahan pada perangkat lunak, yang baru terdeteksi pada saat perangkat lunak dipergunakan.
2. *Adaptive Maintenance* : Penyesuaian dengan lingkungan baru, misalnya sistem operasi atau sebagai tuntutan atas perkembangan sistem komputer, misalnya penambahan *printer driver*
3. *Perfektive Maintenance* : Bila perangkat lunak sukses dipergunakan oleh pemakai. Pemeliharaan ditujukan untuk menambah kemampuannya seperti memberikan fungsi-fungsi tambahan, peningkatan kinerja dan sebagainya

Model sekuensial adalah paradigm rekayasa perangkat lunak yang paling luas dipakai dan paling tua. Kelemahan model ini antara lain :

1. Proyek yang sebenarnya jarang mengikuti alur sekuensial seperti diusulkan, sehingga perubahan yang terjadi dapat menyebabkan hasil yang sudah didapat tim harus diubah kembali/iterasi sering menyebabkan masalah baru.
2. Linear sequential model mengharuskan semua kebutuhan pemakai sudah dinyatakan secara eksplisit di awal proses, tetapi kadang-kadang ini tidak dapat terlaksana karena kesulitan yang dialami pemakai saat akan mengungkapkan semua kebutuhannya tersebut.
3. Pemakai harus bersabar karena versi dari program tidak akan didapat sampai akhir rentang waktu proyek.
4. Adanya waktu menganggur bagi pengembang, karena harus menunggu anggota tim proyek lainnya menuntaskan pekerjaannya.

2. Model Prototype



Gambar 2.6 :Prototype paradigma

Model ini dimulai dengan pengumpulan kebutuhan. Pendekatan *prototyping* model digunakan jika pemakai hanya mendefenisikan objektif umum dari perangkat lunak tanpa merinci kebutuhan *input*, pemrosesan dan *output*nya, sementara pengembang tidak begitu yakin akan efisiensi algoritma, adaptasi sistem operasi, atau bentuk antarmuka manusia-mesin yang harus diambil. Cakupan aktivitas dari *prototyping* model terdiri dari :

a. Mendefinisikan objektif secara keseluruhan dan mengidentifikasi kebutuhan yang sudah diketahui.
b. Melakukan perancangan secara cepat sebagai dasar untuk membuat prototype.
c. Menguji coba dan mengevaluasi prototype dan kemudian melakukan penambahan dan perbaikan-perbaikan terhadap prototype yang sudah dibuat.

Secara ideal prototype berfungsi sebagai sebuah mekanisme untuk mengidentifikasi kebutuhan perangkat lunak. Bila prototype yang sedang bekerja dibangun, pengembang harus menggunakan fragmen-fragmen program yang ada atau mengaplikasikan alat-alat bantu (contoh: window manager, dsb) yang memungkinkan program yang bekerja agar dimunculkan secara cepat.

Kelemahan *prototyping* model :

a. Pelanggan yang melihat working version dari model yang dimintanya tidak menyadari, bahwa mungkin saja prototype dibuat terburu-buru dan rancangan tidak tersusun dengan baik
b. Pengembang kadang-kadang membuat implementasi sembarang, karena ingin working version bekerja dengan cepat.

3. Model RAD (*Rapid Application Development*)

Merupakan model proses pengembangan perangkat lunak secara linear sequential yang menekankan pada siklus pengembangan yang sangat singkat/pendek. Jika kebutuhan dipahami dengan baik, proses RAD memungkinkan tim pengembangan menciptakan “sistem fungsional yang utuh” dalam periode waktu yang sangat pendek (kira-kira 60-90 hari). Pendekatan RAD model menekankan cakupan :

a. Pemodelan data (*Data Modelling*)

   Aliran informasi yang didefinisikan sebagai bagian dari fase pemodelan bisnis disaring ke dalam serangkaian objek data yang dibutuhkan untuk menopang bisnis tersebut. Karakteristik/atribut dari masing-masing objek diidentifikasi dan hubungan antara objek-objek tersebut didefinisikan.

b. Pemodelan proses (*Process Modelling*)

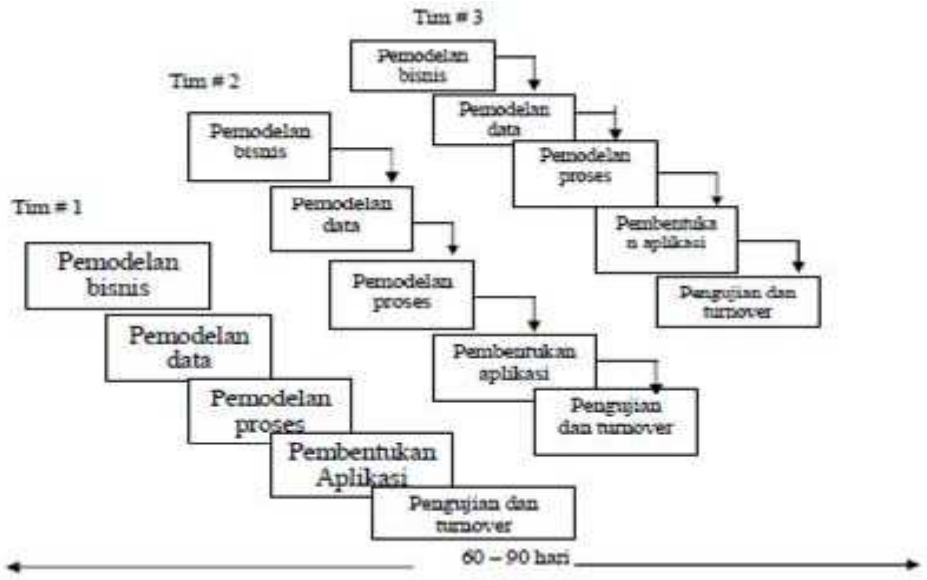
Aliran informasi yang didefinisikan dalam fase pemodelan data ditransformasikan untuk mencapai aliran informasi yang perlu bagi implementasi sebuah fungsi bisnis. Gambaran pemrosesan diciptakan untuk menambah, memodifikasi, menghapus atau mendapatkan kembali sebuah objek data.

c. Pembuatan aplikasi (*Application generation*)

Selain menciftakan perangkat lunak dengan menggunakan bahasa pemrograman generasi ketiga yang konvensional, RAD lebih banyak memproses kerja untuk memakai lagi komponen program yang telah ada atau menciftakan komponen yang bias dipakai lagi. Pada semua kasus, alat-alat Bantu otomatis dipakai untuk memfasilitasi kontruksi perangkat lunak.

d. Pengujian dan pergantian (*Testing and turnover*)

Karena proses RAD menekankan pada pemakaian kembali, banyak komponen yang telah diuji. Hal ini mengurangi keseluruhan waktu pengujian. Tapi komponen baru harus diuji.



Gambar 2.7 :Model RAD

Kelemahan model RAD :

1. Untuk proyek dengan skala besar, RAD membutuhkan sumber daya manusia yang cukup untuk membentuk sejumlah tim RAD yang baik.

2. RAD membutuhkan pengembang dan pemakai yang mempunyai komitmen dalam aktivitas *rapid-fire* untuk melaksanakan aktivitas melengkapi sistem dalam kerangka waktu yang singkat. Jika komitmen tersebut tidak ada, proyek RAD gagal.

Tidak semua aplikasi sesuai untuk RAD.bila system tidak dapat dimodulkan dengan teratur, pembangunan komponen penting pada RAD akan menjadi sangat problematic. RAD menjadi tidak sesuai jika risiko teknisnya tinggi.

4. Model Proses Perangkat Lunak Evolusioner

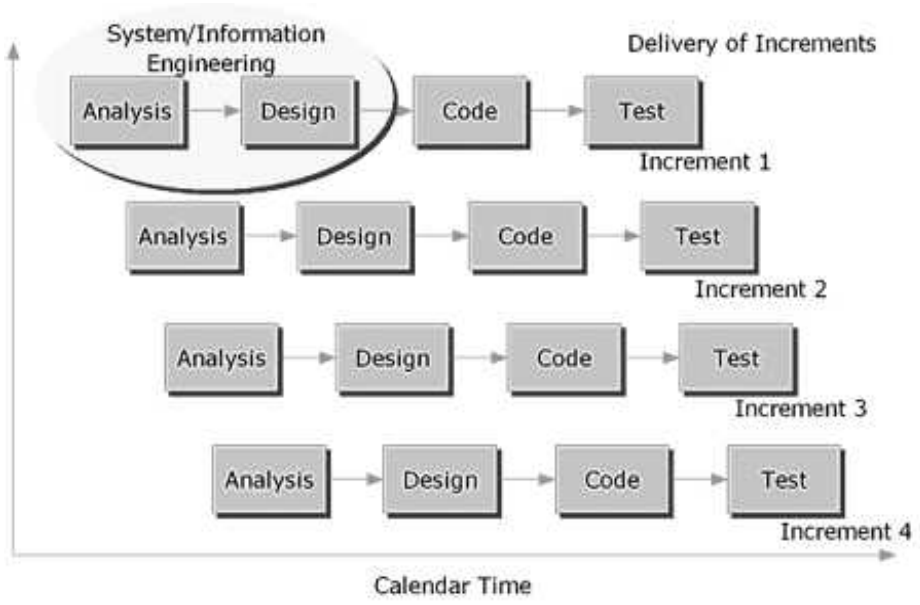
Model evolusioner adalah model iterative, ditandai dengan tingkah laku yang memungkinkan perekayasa perangkat lunak mengembangkan versi perangkat lunak yang lebih lengkap sedikit demi sedikit. Kebutuhan produk dan bisnis kadang-kadang berubah seiring dengan laju perkembanganya. Dalam situasi tersebut maupun lainya, perekayasa perangkat lunak membutuhkan sebuah model proses yang sudah dirancang secara eksplisit untuk mengakomodasi produk perkembangan sepanjang waktu. Model ini bukan termasuk rekayasa perangkat lunak klasik. Model evolusioner meliputi :

a. Model pertambahan

   Model incremental menggabungkan elemen-elemen model sekuensial linier (diaplikasikan secara berulang) dengan filosofi prototype iterative. Model ini memakai urutan-urutan linier di dalam model yang membingungkan, seiring dengan laju waktu kalender. Setiap urutan linier menghasilkan pertambahan, perangkat lunak yang bisa disampaikan. Contoh, perangkat lunak pengolah kata yang dikembangkan dengan menggunakan paradigm pertambahan akan menyampaikan manajemen

file, editing, serta fungsi penghasilan dokumen pada pertambahan pertama, dan selanjutnya. Pertambahan pertama dapat disebut sebagai produk inti (*core product*).

Model ini berfokus pada penyampaian produk operasional dalam Setiap pertambahanya. Pertambahan awal ada di versi *stripped down* dari produk akhir, tetapi memberikan kemampuan untuk melayani pemakai dan juga menyediakan platform untuk evaluasi oleh pemakai.



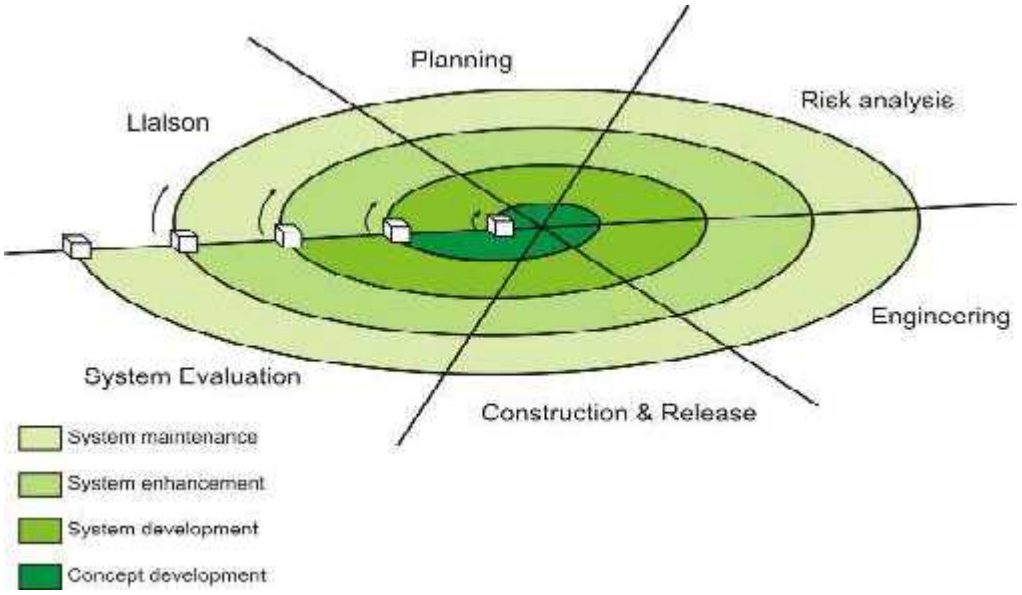
Gambar 2.8: Model Incremental

Perkembangan pertambahan, khususnya berguna pada staffing, tidak bisa dilakukan menggunakan implementasi lengkap oleh batas waktu bisnisyang sudah disepakati untuk proyek tersebut. jika produk inti diterima dengan baik, maka staf tambahan bisa ditambahkan untuk mengimplementasi pertambahan selanjutnya.

b. Model spiral

Awalnya diusulkan oleh Boehm (BOE88), adalah model proses perangkat lunak yang evolusioner, merangkai sifat iterative dari prototype dengan cara control dan aspek sistematis dari model sekuensial linier. Model yang berpotensi untuk pengembangan versi pertambahan perangkat lunak secara cepat. Model spiral dibagi menjadi sejumlah aktifitas kerangka kerja atau wilayah tugas, antara lain :

1. Komunikasi pelanggan, tugas-tugas yang dibutuhkan untuk membangun komunikasi yang efektif diantara pengembang dan pelanggan.
2. Perencanaan, tugas-tugas yang dibutuhkan untuk mendefinisikan sumber-sumber daya, ketepatan waktu, dan proyek informasi lain yang berhubungan.
3. Analisis resiko, tugas-tugas yang dibutuhkan untuk menaksir resiko-resiko, baik manajemen maupun teknis.
4. Perekayasaan, tugas-tugas yang dibutuhkan untuk membangun satu atau lebih representasi dari aplikasi tersebut.
5. Konstruksi dan peluncuran, tugas-tugas yang dibutuhkan untuk mengkonstruksi, menguji, memasang (instal), dan memberikan pelayanan kepada pemakai (contohnya pelatihan dan dokumentasi)
6. Evaluasi pelanggan, tugas-tugas untuk memperoleh umpan balik dari pelanggan dengan didasarkan pada evaluasi representasi perangkat lunak, yang dibuat selama masa perekayasaan, dan dimplementasikan selama masa pemasangan.



Gambar 2.9: Model Spiral

Model spiral menjadi pendekatan yang realistis bagi perkembangan system dan perangkat lunak skala besar. Karena perangkat lunak terus bekerja selama proses bergerak, pengembang dan pemakai memahami, dan bereaksi lebih baik

terhadap resiko dari Setiap tingkat evolusi. Model spiral menggunakan prototype sebagai mekanisme pengurangan resiko.

Model spiral membutuhkan keahlian penafsiran resiko yang masuk akal, dan sangat bertumpu pada keahlian ini untuk mencapai keberhasilan. Jika sebuah resiko tidak dapat ditemukan dan diatur, pasti akan terjadi masalah. Model ini membutuhkan waktu bertahun-tahun sampai kehandalannya bisa dipertimbangkan dengan kepastian absolute.

c. Model rakitan komponen

Model ini menggabungkan beberapa karakteristik model spiral. Bersifat evolusioner, sehingga membutuhkan pendekatan iterative untuk menciptakan perangkat lunak. Tetapi model ini merangkai aplikasi dari komponen perangkat lunak sebelum dipaketkan (kadang disebut kelas).

Aktivitas perangkat lunak dimulai dengan identifikasi calon kelas. Dipenuhi dengan mengamati data yang akan dimanipulasi oleh aplikasi dan algoritma-algoritma yang akan diaplikasikan. Data dan algoritma yang berhubungan dikemas ke dalam kelas. Kelas-kelas tersebut disimpan dalam *class library* (tempat penyimpanan).

Model ini membawa pada penggunaan kembali perangkat lunak, dan kegunaan kembali itu memberi sejumlah keuntungan yang bisa diukur pada rekayasa perangkat lunak.

d. Model perkembangan konkruen

Representasi aktivitas dalam model ini, meliputi aktivitas analisis, bisa berada dalam salah satu dari keadaan-keadaan yang dicatat pada saat tertentu. Dengan cara yang sama, aktivitas yang lain (desain atau komunikasi pelanggan) dapat direpresentasikan dalam sebuah sikap yang analog. Semua aktifitas ada secara konkruen tetapi dia tinggal didalam keadaan yang berbeda. Model ini sering digunakan sebagai paradigm bagi pengembangan aplikasi klien/server.

Kenyataanya model proses konkruen bisa diaplikasikan ke dalam semua tipe perkembangan perangkat lunak, dan memberikan gambaran akurat mengenai keadaan tertentu dari sebuah proyek. Selain membatasi ajtivitas

perekayasa ke dalam sederetan kejadian, model proses juga mendefinisikan jaringan aktivitas.

2.7 Perkembangan Sistem Operasi

Sistem Operasi adalah perangkat lunak sistem yang bertugas untuk melakukan kontrol dan manajemen perangkat keras serta operasi-operasi dasar sistem, termasuk menjalankan software aplikasi seperti program-program pengolah kata dan brower web. Terdapat dua Sistem Operasi yang digunakan yaitu Sistem Operasi Microsoft Windows dan Sistem Operasi Open Source.

a. Sistem Operasi Symbian (Symbian OS)

Sistem Operasi Symbian merupakan sebuah sistem operasi tak bebas *(closed source)* yang dikembangkan oleh Symbian Ltd. Sistem operasi ini dirancang khusus untuk digunakan peralatan bergerak *(mobile)* dan rata-rata yang menggunakan sistem operasi Symbian ini adalah varian / type Nokia Nseries. Pada saat ini Symbian OS telah banyak digunakan oleh berbagai vendor produk sebagai peralatan komunikasi *mobile* untuk berbagai jenis produk mereka yang bervariasi.

Variasi dari sisi hardware ini dimana Symbian OS diimplementasi dapat dimungkinkan karena sistem operasi ini memiliki antarmuka pemprograman aplikasi (Application Programming Interface; API). API mendukung terhadap komunikasi dan tingkah laku yang umum pada hardware yang dapat digunakan oleh objek aplikasi lain. Hal ini dimungkinkan karena API merupakan objek antarmuka yang didefenisikan pada level aplikasi, yang berisikan prosedur dan fungsi (dan juga variabel serta struktur data) yang mengelola/memanggil kernel dimana sebagai penghubung antara software dan hardware. Dengan adanya standar API ini membantu pihak pengembang untuk melakukan penyesuaian atas aplikasi yang dibuatnya agar dapat diinstal pada produk telepon bergerak yang bermacam-macam.

Keunggulan :

1. Mudah dalam dimasukkan game atau aplikasi apa saja (Format jar. dan sis.)
2. Cara penggunaannya yang mudah

Kelemahan :

1. Karena memiliki sistem operasi terbuka, handphone jenis ini sangat rentan terhadap serangan virus seperti Cabir, Commwarrior, SymbOS.skulls. dan masih banyak lagi.
2. Handphone dengan sistem ini gampang Hang, atau lambat dalam membuka gallery, lagu, serta pesan singkat(SMS)
3. Diperlukan memori card yang besar jika ingin menginstal banyak aplikasi

b. Sistem Operasi Apple (iOS)

Sistem Operasi iOS adalah sebuah sistem operasi yang ditanamkan pada produk-produk yang diciptakan perusahaan terkenal, Apple. Apple adalah sebuah nama orang yang berpengaruh pada perusahaan Apple yaitu Steve Jobs. Awalnya iOS hanya dikembangkan untuk iPphone. Namun seiring dengan berkembangnya teknologi maka iOS sendiri sekarang sudah diperluas untuk bisa diperguakan pada berbagai jenis perangkat lainnya seperti iPod Touch, iPad dan Apple TV.

Versi terbaru dari sistem operasi iOS adalah iOS versi 6. Menurut beberapa sumber Apple akan meninggalkan *Google Maps* dan menggantinya dengan perangkat lunak lain di sistem operasi *mobile* iOS 6 miliknya. Kelebihan dari sistem operasi iOS sendiri menurut desainer antarmuka iOS, Danilo Campos dari perusahaan Hipmunk adalah karena desain di mana tampilan iOS lebih elegan, dan mudah untuk dipergunakan dibandingkan dengan sistem operasi pada *smartphones* lainnya.

Kelebihan :

1. Sulit terserang Virus
2. *User Friendly.*
3. Memiliki tampilan yang bagus.
4. Tersedia aplikasi yang bagus.

5. Ukuran memory internal yang cukup besar.
6. *UpGrade* FW yang relatif mudah.

Kekurangan :

1. Harga relatif mahal.
2. Memory yang tersedia tidak bisa ditambah sesuai keinginan kita.
3. Tampilannya tidak bisa diubah-ubah sesukai hati.
4. Kualitas suara yang dihasilkan speaker outputnya sangat kecil.

2.8 Sistem Operasi Android

Sistem Operasi Android di dirikan oleh Andy Rubin, Rich Miner, Nick Sears dan Chris White pada tahun 2003. Android merupakan sebuah operasi perangkat mobile berbasis linux yang mencakup sistem operasi, middleware, dan aplikasi. Beberapa pengertian lain dari Android, yaitu :

1. Merupakan platform terbuka (Open Source) bagi para pengembang (Programmer) untuk membuat aplikasi.
2. Merupakan sistem operasi yang dibeli Google Inc. dari Android Inc.

Pada saat ini terdapat dua jenis distributor sistem operasi Android. Pertama yang mendapat dukungan penuh dari Google atau *Google Mail Services* (GMS). Dan kedua adalah yang benar-benar sebagai *Open Handset Distribution* (OHD).

Android merupakan sebuah Sistem Operasi perangkat lunak mobile berbasis linux yang mencangkup system operasi, middleware, dan aplikasi. Beberapa pengetian lain dari Android, yaitu :

a. Merupakan platform terbuka (Open Source) bagi para pengembang (programer) untuk membuat aplikasi.

b. Merupakan Sistem Operasi yang dibeli Google Inc. dari Android Inc.

c. Bukan bahasa pemograman, akan tetapi hanya menyediakan lingkungan hidup atau run time environment yang disebut DVM

(*Dalvik Virtual Machine*) yang telah di optimasikan untuk device/alat dengan system yang kecil.

Saat ini terdapat dua jenis distributor sistem operasi Android. Pertama yang mendapat dukungan penuh dari *Google* atau *Google Mail Services* (GMS) dan kedua adalah yang benar–benar bebas distribusinya tanpa dukungan langsung Google atau dikenal sebagai Open Handset Distribution (OHD). *(http://id.wikipedia.org/wiki/Android_(sistem_operasi))*.

Android memiliki beberapa model versi yang telah dikeluarkan, adalah sebagai berikut :

1. **Android versi 1.1.**
   Pertama kali dirilis oleh Google pada tanggal 9 Maret 2009. Versi ini memiliki fitur-fitur: jam, alarm, *voice search* (pencarian suara), dll.
2. **Android versi 1.5 *(Cupcake).***
   Merupakan versi yang dirilis Google pada pertengahan Mei 2009. Fitur-fitur yang dimiliki versi ini adalah merekam dan menonton video dengan modus kamera, meng-upload video ke *youtube*.
3. **Android versi 1.6 *(Donut).***
   Dirilis pada September 2009 dengan fitur-fitur: proses pencarian yang lebih baik dibandingkan versi sebelumnya, penggunaan baterai indikator dan control applet VPN, Galeri memilih photo yang dihapus, kamera, camcorder, dll
4. **Android versi 2.0/2.1 *(Eclair).***
   Merupakan versi yang dirilis Google pada tanggal 3 Desember 2009, memiliki fiturr-fitur: mengoptimalkan hardware versi sebelumnya, *Google Maps*, *browser* baru, *HTML5*, daftar kontak, dll.
5. **Android versi 2.2 *(Froyo : Frozen Yoghurt).***
   Versi yang di rilis Google pada Mei 2010. Versi ini memiliki fitur-fitur: dapat menghapus komponen, DVM dioptimalkan, Graphic 2D dan 3D, *SQLite*, *Media Audio&Video*, *Wifi*, *GSM*, *Bluetooth*, dan *GPS*.

6. **Android versi 2.3 *(Gingerbread)*.**

   Versi yang dirilis Google pada Desember 2010, memiliki fitur-fitur: SIP-*based VoIP*, *NFC* (*Near Field Communications*), *Gyroscope* dan Sensor, *Mixable Audio Effects*, dan Download Manager

7. **Android versi 3.0 dan 3.1*(Honeycomb)*.**

   Versi ini khusus dirilis Google untuk pengguna PC Tablet. Beberapa *Smartphone* tidak dapat menggunakan karena versi ini memang banyak dipakai bukan untuk *Voice Call*. Memiliki fitur-fitur: aksesoris API terbuka, USB host API, Keyboard External dan perangkat penunjuk, joystick dan gamepads, Wifi.

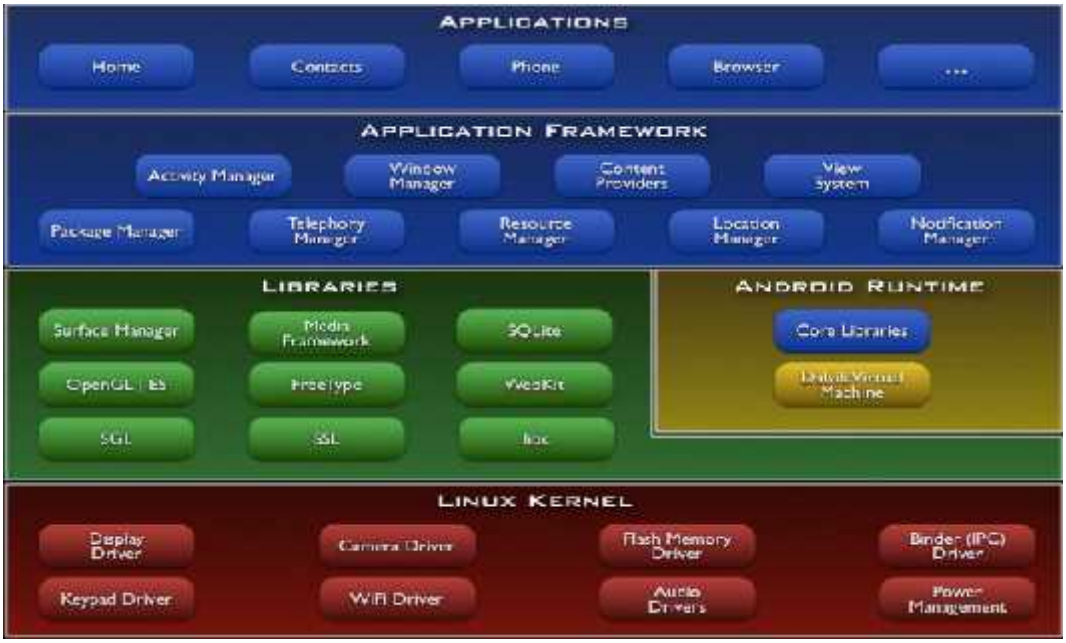
8. **Android 4.0 ➔ dirilis pada 19 Oktober 2011**

   Dirilis pada tanggal 19 Oktober 2011, membawa fitur Honeycomb untuk smartphone dan menambahkan fitur baru termasuk membuka kunci dengan pengenalan wajahm jaringan data pemantauan penggunaan dan kontrol, terpadu kontak jaringan sosial, perangkat tambahana fotografi, mencari email secara offline, dan berbagi informasi dengan menggunakan NFC. Ponsel pertama yang menggunakan sistem operasi ini adalah Samsung Galaxy Nexus.

Android versi 3.0 ke atas adalah generasi *platform* yang digunakan untuk tablet PC. Sementara versi 4.0 sudah merupakan *platform* yang bisa dipakai di *smartphone* dan *tablet PC*.

Arsitektur Android

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



Gambar 2.10 : Arsitektur Android

Keterangan:

1. *Applications*

Applications 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 aplikasi inti termasuk klien email, program SMS, kalender, peta, browser, kontak, dan lain-lain.

2. *Applications Frameworks*

Android adalah "Open Development Platform" yaitu Android menawarkan kepada pengembang atau memberi kemapuan kepada pengembang untuk membangun aplikasi yang bagus dan inovatif. Pengembang memiliki akses penuh menuju API *framework* seperti yang dilakukan oleh aplikasi yang dikategori inti. Arsitektur aplikasi dirancang supaya kita dengan mudah dapat menggunakan kembali komponen yang sudah digunakan. (*reuse*).

Komponen-komponen yang termasuk didalam *Applications Framework* adalah sebagai berikut:

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

3. *Libraries*

Libraries 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++ ini seperti Libc dan SSL, serta:

1. *Libraries* media untuk pemutaran media audio dan video
2. *Libraries* untuk manajemen tampilan
3. *Libraries Graphics* mencakup SGL dan OpenGL untuk grafis 2D dan 3D
4. *Libraries* SQLite untuk dukungan database
5. *Libraries* SSL dan WebKit terintegrasi dengan web browser dan *security*
6. *Libraries* LiveWebcore mencakup modern web browser dengan *engine embeded web view*
7. *Libraries* 3D yang mencakup implementasi OpenGL ES 1.0 API's

4. *Android Run Time*

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. 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 menerjemahkan bahasa jaca/c yang ditangani oleh *Core Libraries.*

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

5. *Application Layer*

Puncak dari diagram arsitektur android adalah lapisan aplikasi dan *widget*. Lapisan aplikasi merupakan lapisan yang paling tampak pada pengguna ketika menjalankan program. Pengguna hanya akan melihat program ketika digunakan tanpa mengetahui proses yang terjadi dibalik lapisan aplikasi. Lapisan ini berjalan dalam *Android runtime* dengan menggunakan kelas dan service yang tersedia pada *framework* aplikasi.

Lapisan aplikasi android sangat berbeda dibandingkan dengan sistem operasi lainnya. Pada android semua aplikasi, baik aplikasi inti (*native*) maupun aplikasi pihak ketiga berjalan diatas lapisan aplikasi dengan menggunakan pustaka API (*Application Programming Interface*) yang sama.

6. *Linux Kernel*

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

2.9 Contoh Aplikasi Android Kampus

2.9.1 Contoh Aplikasi Android Politeknik Telkom Bandung

Politeknik Telkom merupakan salah satu lembaga pendidikan yang dibangun oleh Yayasan Pendidikan Telkom untuk menghasilkan sumber daya manusia yang memiliki pandangan luas mengenai teknologi dan informasi yang sedang berkembang saat ini. Berikut adalah contoh tampilan Sistem Informasi Dosen pada Politeknik Telkom Bandung:

1. Halaman Login

   Halaman ini berguna bagi admin untuk dapat mengakses pengolahan *database*.

2. Menu Info Gaji

   Menu ini berisi informasi tentang gaji dosen, diantaranya informasi gaji pokok, tunjangan jabatan dan potongan.

3. Menu Jadwal Mengajar

   Menu ini berisi informasi tentang jadwal mengajar dosen, diantaranya tanggal mengajar, nama mata kuliah, kelas yang diajar, ruangan dan *shift* waktu mengajar.

4. Menu Absen Dosen

   Menu ini berisi informasi tentang absen dosen, diantaranya mata kuliah yang akan diajar, kelas, dan persentase kehadiran dosen selama ini.

Gambar 2.11: Aplikasi Portal Dosen

Gambar 2.12: Aplikasi Android Politeknik Telkom Bandung

2.9.2 Contoh Aplikasi Android STIKOM PGRI Banyuwangi

STIKOM PGRI merupakan salah satu perguruan tinggi yang ada di Banyuwangi. STIKOM PGRI memiliki sumber daya manusia yang memiliki pandangan luas terhadap teknologi saat ini. Berikut adalah contoh tampilan Sistem Akademik menggunakan Android pada STIKOM PGRI Banyuwangi:

1. Menu Login

   Menu ini berguna bagi admin untuk dapat mengakses pengolahan database
2. Jadwal Kuliah

   Menu ini berguna bagi mahasiswa untuk melihat jadwal kuliah, diantaranya tanggal masuk kuliah, nama mata kuliah, dan kelas.
3. Kartu Hasil Studi

   Menu ini berguna bagi mahasiswa untuk melihat hasil nilai yang diperoleh pada tiap semester
4. Info Kampus

Menu ini berguna bagi mahasiswa untuk melihat informasi-informasi apa saja yang ada dikampus.

Gambar 2.13 : Tampilan aplikasi sistem akademik pada STIKOM PGRI Banyuwangi

Gambar 2.14: Tampilan halaman yang berisi link kemasing-masing menu.

2.9.3 Contoh Aplikasi Android Universitas Komputer Indonesia

Universitas Komputer Indonesia (disingkat ***UNIKOM***) adalah sebuah perguruan tinggi swasta yang berada di kota Bandung,Jawa Barat, tepatnya berlokasi di Jl.Dipatiukur No 112-114. Rektornya saat ini dijabat oleh Dr. Ir. Eddy Soeryanto Soegoto. (*Sumber : http://www.unikomcenter.com*).

Berikut adalah contoh aplikasi android yang digunakan pada Universitas Komputer Indonesia. (*KampusOnlineUnikom.apk*)

Gambar 2.15 : Tampilan Login Universitas Komputer Indonesia Berbasis Android

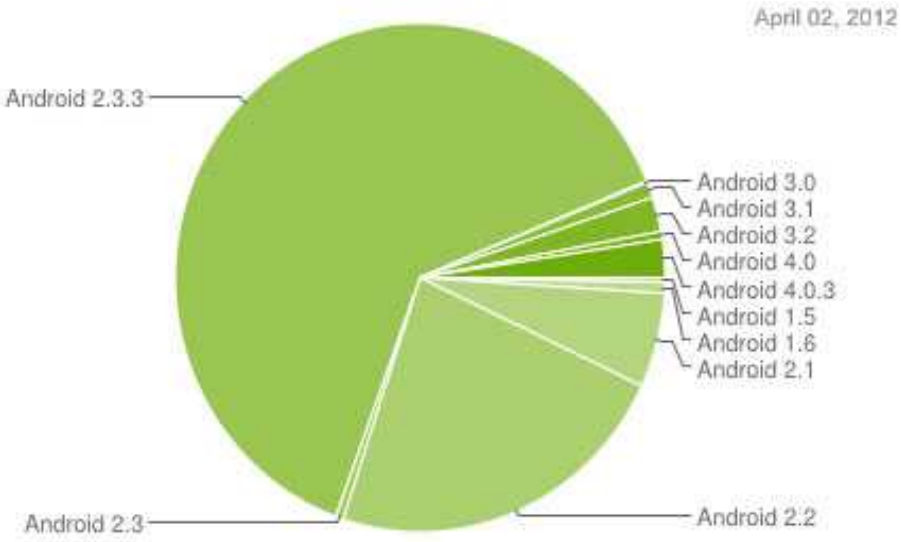
2.9.4 Contoh Aplikasi Android Universitas Pendidikan Indonesia

Contoh sistem aplikasi android yang digunakan Universitas Pendidikan Indonesia adalah sistem silabus online yang bisa diakses di website dan juga di android. (*Sumber : Silabus_UPI.apk*)

Gambar 2.16 : Tampilan Silabus Online Universitas Pendidikan Indonesia

2.10 Perkembangan android di Dunia

Berikut adalah Grafik Pengguna Android di Dunia pada tanggal 02 April 2012 :



Gambar 2.17 : Grafik Pengguna Android di Dunia

(Sumber : http://developer.android.com/about/dashboards/index.html)

2.11 Perkembangan Android di Indonesia

Pada saat ini, perkembangan android di Indonesia dipengaruhi oleh banyak *smartphone* yang telah beredar di Indonesia dan keinginan berbagai produsen *smartphone* tersebut untuk memangkas biaya produksi sehingga menghasilkan produk smartphone yang berkualitas dan mempunyai harga jual yang lebih terjangkau daripada menggunakan Sistem Operasi yang lain. Pada Tahun 2009 pasar android di Indonesia vendor *handphone* tersebut merilis *handphone* android pertama di Indonesia. Pertengahan akhir tahun 2010, *handphone* android mulai berkembang di Indonesia.

Beberapa faktor penyebab Android berkembang cepat di Indonesia :

a. Software Update

   Android selalu melakukan update secara terus menerus, melakukan perbaikan berbagai aplikasi dan penambahan fitur yang menjadikan Sistem Operasi ini lebih baik dari versi sebelumnya.

b. Didukung oleh Vendor Kelas Atas

   Dukungan penuh dari vendor-vendor kelas atas seperti Samsung, HTC, Motorola dalam menghasilkan *smartphone* yang berkelas akan membantu menaikkan pamor android.

c. User Friendly

   Teknologi layar sentuh, membuat mudah dalam penggunaannya serta didukung oleh tampilan yang menarik.