# BAB II

# TINJAUAN PUSTAKA

## 2.1 Tinjauan Tempat Penelitian

Tinjauan tempat penelitian adalah untuk mengetahui keadaan di tempat penelitian diantaranya adalah sejarah berdirinya, logo perusahaan, struktur organisasi, deskripsi tugas, serta visi dan misi.

### 2.1.1 Sejarah Perusahaan

Berawal dari seorang Mahasiswa Universitas Maranatha di Bandung yaitu Reza Nurhilman atau biasa dipanggil dengan sebutan AXL29. Ia mencoba untuk membangun sebuah usaha dibidang pangan yakni usaha keripik singkong pedas yang diberi nama keripik Maicih. Nama Maicih ini diambil dari memori sewaktu Ia masih kecil. Reza menyebutkan bahwa kalau Ia dibawa mama ke pasar, biasanya ada ibu-ibu tua memakai ciput dengan baju alakadarnya. Setiap belanja ibu tersebut mengeluarkan dompet, bonus dari toko emas yang bersleting untuk memasukkan uang receh dan ibu Reza menyebutkan bahwa itu dompet Maicih. Beberapa tahun kemudian, Reza bertemu dengan seorang ibu yang sosoknya menyerupai Maicih dalam memorinya. Ibu paruh baya yang pakaiannya tradisional. Ternyata ibu tersebut dapat membuat bumbu keripik pedas. Lalu Reza membuatlah *brand* dengan nama Maicih. Awalnya Reza memasarkan Maicih melalui *door-to-door* dan *mouth-to-mouth* untuk memperkenalkan pada saudara dan teman-teman terdekat, hingga berjalan 2 bulan Ia menemukan cara yang unik untuk memasarkan Maicih yaitu melalui jejaring sosial seperti *Facebook* dan

*Twitter*. Ternyata kekuatan *Twitter* sangat kuat dan orang-orang mulai memperbincangkan keripik Maicih.

Berangkat dengan modal sekitar Rp 15 Juta, ia membuat permainan yang memancing penasaran *Facebookers* dan *Tweeps*. Reza membuat suatu organisasi yang unik Ia namakan dengan Republik Icih (RI), sehingga di dalamnya terdapat Presiden, Menteri-Menteri, dan para Jendral (sebutan untuk agen penjual Maicih) yang membantu membangun produk Maicih hingga berkembang sampai saat ini. Lalu Ia merancang lokasi penjualan berpindah-pindah setiap hari, sehingga keripik Maicih tidak dijual di satu tempat, outlet, ataupun toko, namun penjualan keripik hanya menggunakan mobil pribadi yang berpindah-pindah dari satu tempat ke tempat lain. Untuk mendapatkan informasi keberadaan para Jendral konsumen dapat memantaunya dari jejaring sosial *Twitter*. Strategi tersebut berhasil, keripiknya menjadi barang buruan. Konsumen harus mengantri berjam-jam demi mendapatkan keripik super pedas itu. Bahkan, antrian pernah memanjang hingga satu Kilometer. Kini omset penjualan keripik Maicih mencapai Rp 4 Milyar dengan rata-rata Rp 750 Juta per minggu.

Dalam kurun waktu satu tahun kini Maicih telah menghasilkan 3 jenis produk yaitu keripik, gurilem, dan seblak. Tenaga kerja para Jendral kini mencapai ratusan orang yang tersebar di Indonesia bahkan hingga luar negeri. Dengan kegigihan Reza sang Presiden Maicih, pada bulan Agustus 2011 Maicih menjadi sebuah PT yang diberi nama dengan PT Maicih Inti Synergi.

### 2.1.2 Logo Perusahaan

PT Maicih Inti Synergi dalam menjalankan bisnis penjualan produk-produk selama ini selalu menggunakan logo yang berciri khas emak-emak berkonde yang mengahadap samping, ini membuktikan bahwa Produk Maicih masih mempunyai ciri khas tradisional dari pengolahan produk yang mereka buat dan penamaan Maicih pun diambil dari sebuah dompet emak-emak yang sering disebut dengan nama Maicih, Berikut ini (Gambar 2.1) adalah Logo dari PT Maicih Inti Synergi:

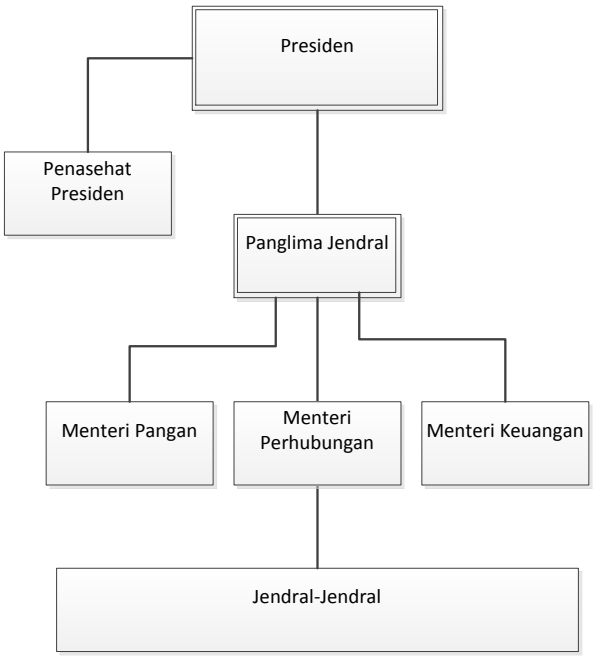



**Gambar 2.1 Logo Perusahaan**

### 2.1.3 Struktur Organisasi

Struktur organisasi (Gambar 2.2) perusahaan merupakan suatu gambaran yang menunjukan suatu jabatan atau pekerjaan yang harus dilaksanakan oleh masing-masing bagian atau anggota dari organisasi perusahaan yang tergambar dengan struktur.

Struktur organisasi Maicih dipimpin oleh seorang Panglima Jendral yang dibantu oleh para Menteri dan Jendral, yang bertanggung jawab kepada Presiden selaku pemilik perusahaan. Berikut adalah struktur organisasi dari PT Maicih Inti Synergi:




**Gambar 2.2 Struktur Organisasi PT. MAICIH INTI SYNERGI**

### 2.1.4 Dekripsi Tugas

Definisi Tugas digunakan untuk mengetahui tugas dan wewenang, tanggung jawab dari masing-masing bagian. Definisi tugas yang ada di PT. MAICIH INTI SYNERGI adalah sebagai berikut:

1. Presiden
    a. Sebagai pengambilan keputusan.
    b. Sebagai koordinator semua kegiatan yang telah dilaksanakan.
2. Penasehat Presiden
    a. Memberikan arahan nasehat kepada Presiden untuk melakukan jalan yang terbaik.
    b. Melakukan tugas sementara yang tidak bisa dilakukan oleh Presiden.

3. Panglima Jendral
    a. Mengatur jadwal-jadwal tugas yang dilakukan oleh Presiden.
    b. Melakukan tugas sebagai *public relation.*
    c. Mengawasi dan memberi perintah kepada para Menteri.
4. Menteri Pangan
    a. Melaporkan kepada Presiden tentang keadaan produk.
    b. Mengawasi dan melakukan pemeliharaan pada produk.
    c. Mengawasi penjualan produk yang dilakukan oleh Jendral.
5. Menteri Perhubungan
    a. Melakukan koordinasi kepada para Jendral di setiap daerah.
    b. Mengawasi dan mengatur pekerjaan yang dilakukan oleh Jendral.
6. Menteri keuangan
    a. Melaporkan kepada Presiden tentang penjualan produk yang diperoleh.
    b. Bertanggung jawab penuh atas keuangan.
7. Jendral
    a. Bagian penjualan yang bertatap langsung oleh pembeli.
    b. Bagian penjualan yang mengatur pemesanan produk oleh pembeli.

### 2.1.5 Visi dan Misi

#### 2.1.5.1 Visi

Menjadi produk dan *icon* kota Bandung yang Nasional serta *go International* dalam bidang pangan, *entertainment* dan edukasi.

#### 2.1.5.2 Misi

Memperkenalkan Maicih ke seluruh dunia, meningkatkan lagi jumlah produksi, serta mengikuti berbagai event untuk memberikan pengalaman dan *sharing* agar penduduk Indonesia bisa mengembangkan lagi cara-cara strategi berwirausaha.

## 2.2 Landasan Teori

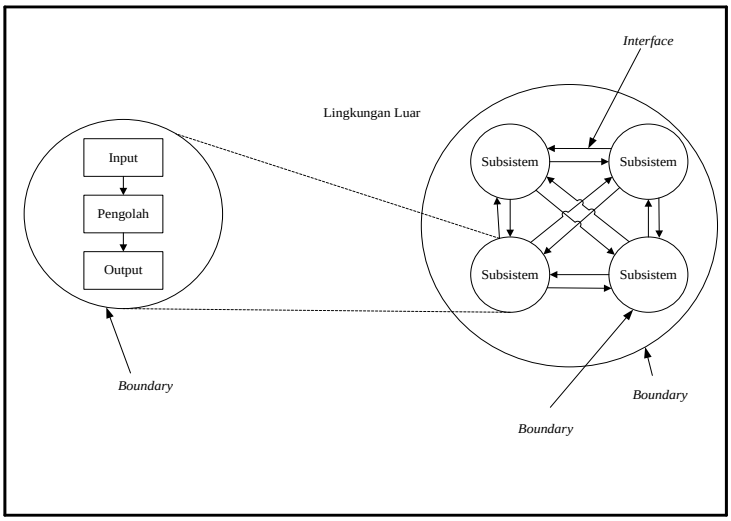
### 2.2.1 Konsep Dasar Sistem

Sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran tertentu **[3]**.

Dari pengertian di atas dapat diambil suatu kesimpulan bahwa suatu sistem merupakan elemen yang saling berkaitan dan saling mempengaruhi dalam melakukan kegiatan bersama untuk mencapai suatu tujuan tertentu.

### 2.2.2 Karakteristik Sistem

Suatu sistem memiliki karakteristik atau sifat-sifat tertentu, yaitu memiliki komponen-komponen *(components)*, batas sistem *(boundary)*, lingkungan luar sistem *(environment)*, penghubung *(interprest)*, masukan *(input)*, keluaran *(output)*, pengolah *(process)*, sasaran *(objective)* dan tujuan *(goal)*.




**Gambar 2.3 Karakteristik Sistem**

1. Komponen Sistem *(Components)*

Komponen sistem atau elemen-elemen sistem dapat berupa suatu subsistem atau bagian-bagian dari sistem. Setiap sistem baik besar maupun kecil, selalu mengandung komponen-komponen atau subsistem-subsistem. Setiap subsistem mempunyai sifat-sifat dari sistem untuk menjalankan suatu fungsi tertentu dan mempengaruhi proses sistem secara keseluruhan. Suatu sistem dapat mempunyai suatu yang lebih besar yang disebut *supra system.*

2. Batas Sistem (*System Boundary*)

Batas sistem merupakan daerah-daerah yang membatasi antara satu sistem dengan sistem lainnya dengan lingkungan luarnya. Batas sistem ini memungkinkan suatu sistem dipandang sebagai satu kesatuan. Batas suatu sistem menujukkan ruang lingkup (*scope*) dari sistem tersebut.

3. Lingkungan Luar Sistem (*System Environment*)

Lingkungan luar sistem dari suatu sistem adalah apapun di luar batas dari sistem yang memengaruhi operasi sistem yang dapat bersifat menguntungkan dan

dapat pula merugikan sistem tersebut. Lingkungan luar yang menguntungkan merupakan energi dari sistem yang harus dijaga dan dipelihara. Sedangkan yang merugikan harus ditahan dan dikendalikan, karena akan mengganggu kelangsungan hidup sistem

4. Penghubung Sistem (*System Interface*)

Penghubung sistem merupakan media penghubung antara satu subsistem dengan subsistem yang lainnya. Melalui penghubung ini memungkinkan sumber-sumber daya mengalir dari satu subsistem ke subsistem yang lainnya. Dengan penghubung akan terjadi interaksi antar subsistem, sehingga membentuk satu kesatuan.

5. Masukan Sistem (*System Input*)

Masukan adalah suatu energi yang dimasukkan ke dalam sistem. Masukan dapat berupa masukan perawatan (*maintenance input*) dan masukan sinyal (*signal input*). *Maintenance input* adalah energi yang dimasukkan supaya sistem tersebut dapat beroperasi. Contoh *maintenance input* di dalam sistem komputer adalah program, yang digunakan untuk mengoperasikan komputer. Sedangkan *signal input* adalah energi yang diproses untuk mendapatkan keluaran. Contoh *signal input* di dalam sistem komputer adalah data, yang dapat diolah menjadi informasi.

6. Keluaran Sistem (*System Output*)

Keluaran (*Output*) merupakan hasil dari energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna.

7. Pengolah Sistem (*System Process*)

Suatu sistem dapat mempunyai suatu bagian pengolah atau sistem itu sendiri sebagai pengolahnya, yang bertugas untuk merubah masukan menjadi keluaran.

8. Sasaran Sistem (*System Objective*)

Suatu sistem pasti memiliki tujuan (*goal*) atau sasaran (*objective*). Suatu operasi sistem akan berguna dan berhasil apabila mencapai sasaran atau tujuannya. Sasaran sistem sangat menentukan masukan yang dibutuhkan sistem dan keluaran yang akan dihasilkan sistem.

#### 2.2.2.1 Klasifikasi Sistem

Sistem dapat diklasifikasikan dari beberapa sudut pandangan, diantaranya adalah sebagai berikut ini:

1. Sistem diklasifikasikan sebagai sistem abstrak (*abstract system*) dan sistem fisik (*physical system*). Sistem abstrak adalah sistem yang berupa pemikiran atau ide-ide yang tidak tampak secara fisik. Sistem fisik merupakan sistem yang ada secara fisik.
2. Sistem diklasifikasikan sebagai sistem alamiah (*natural system*) dan sistem buatan manusia (*human made system*). Sistem alamiah adalah sistem yang terjadi melalui proses alam, tidak dibuat manusia. Sistem buatan manusia yang melibatkan interaksi antara manusia dengan mesin disebut dengan *human-machine system* atau ada yang menyebut dengan *man-machine system*.
3. Sistem diklasifikasikan sebagai sistem tertentu (*deterministic system*) dan sistem tak tentu (*probabilistic system*). Sistem tertentu beroperasi dengan tingkah laku yang sudah diprediksi. Interaksi di antara bagian-bagiannya dapat

dideteksi dengan pasti, sehingga keluaran dari sistem dapat diramalkan. Sistem tak tentu adalah sistem yang kondisi masa depannya tidak dapat diprediksi karena mengandung unsur probabilitas.

4. Sistem diklasifikasikan sebagai sistem tertutup (*closed system*) dan sistem terbuka (*open system*). Sistem tertutup merupakan sistem yang tidak berhubungan dan tidak terpengaruh dengan lingkungan luarnya. Sistem ini bekerja secara otomatis tanpa adanya campur tangan dari pihak diluarnya. Secara teoritis sistem tertutup ini ada, tetapi kenyataannya tidak ada sistem yang benar-benar tertutup, yang ada hanyalah *relatively closed system* (secara relatif tertutup, tidak benar-benar tertutup). Sistem terbuka adalah sistem yang berhubungan dan terpengaruh dengan lingkungan luarnya. Sistem ini menerima masukan dan menghasilkan keluaran untuk lingkungan luar atau subsistem yang lainnya.

#### 2.2.2.2 Pengertian Informasi

Informasi adalah data yang telah diolah menjadi suatu bentuk yang berarti bagi penerima dan bermanfaat dalam mengambil keputusan saat ini atau yang akan datang **[3]**.

Dalam hal ini, data bisa dianggap sebagai obyek dan informasi adalah suatu subyek yang bermanfaat bagi penerimanya. Informasi juga bisa disebut sebagai hasil pengolahan atau pemrosesan data.

Sumber dari informasi adalah data. Data adalah kenyataan yang menggambarkan suatu kejadian-kejadian dan kesatuan nyata, tetapi data merupakan bentuk yang masih mentah dan belum dapat memberikan arti banyak

bagi pemakai, sehingga perlu diolah lebih lanjut untuk menghasilkan informasi yang dibutuhkan. Dengan kata lain informasi adalah hasil dari pengolahan data.

#### 2.2.2.3 Kualitas Informasi

Kualitas dari suatu informasi tergantung pada tiga hal pokok yaitu:

a. Akurat *(Accurate)*

Informasi harus bebas dari kesalahan-kesalahan dan tidak menyesatkan, dalam hal ini informasi harus jelas mencerminkan maksudnya.

b. Tepat Waktu *(Time Lines)*

Informasi yang datang pada penerima tidak boleh terlambat. Informasi yang sudah usang tidak akan mempunyai nilai lagi karena informasi merupakan suatu landasan dalam mengambil sebuah keputusan dimana bila pengambilan keputusan terlambat maka akan berakibat fatal untuk organisasi.

c. Relevan *(Relevance)*

Informasi harus mempunyai manfaat untuk pemakainya, dimana relevansi informasi untuk tiap-tiap individu berbeda tergantung pada yang menerima dan yang membutuhkan.

#### 2.2.2.4 Nilai Informasi

Nilai Informasi *(value of information)* ditentukan oleh dua hal yaitu manfaat dan biaya. Suatu informasi dikatakan bernilai bila manfaatnya lebih efekif dibandingkan dengan biaya mendapatkannya.

### 2.2.3 Definisi Rekayasa Perangkat Lunak

Rekayasa perangkat lunak diperlukan untuk mengurutkan perubahan dari masalah menjadi solusi perangkat lunak yang dapat bekerja dengan baik dan

benar. Pengertian perangkat lunak itu sendiri adalah *source code* pada suatu program atau sistem dimana perangkat keras komputer baru akan bisa beroperasi setelah adanya instruksi- innstruksi pada perangkat lunak.

Menurut Fritz Bauer, Rekayasa Perangkat Lunak *(Software Engineering)* adalah penggunaan prinsip- prinsip rekayasa untuk mendapatkan perangkat lunak yang ekonomis dan dapat bekerja secara efisien pada mesin nyata. Rekayasa perangkat lunak dari sudut pandang lain adalah pendekatan sistematis untuk merekayasa perangkat lunak yang handal atau bermutu, tepat waktu dan dengan biaya yang optimal.

#### 2.2.3.1 Pengembangan Perangkat Lunak

Rekayasa perangkat lunak atau tim perekayasa harus menggabungkan strategi pengembangan yang melingkupi lapisan proses, metode, dan alat-alat bantu serta fase-fase *generic*. Strategi ini sering diacukan sebagai model proses atau paradigma rekayasa perangkat lunak.

Perangkat lunak yang disimpan atau digunakan dalam jangka waktu yang cukup lama akan menimbulkan masalah apabila sering diabaikan atau tidak diperhatikan perkembangannya. Permasalahan yang akan timbul dari penggunaan perangkat lunak yang cukup lama adalah:

1. Adanya kebutuhan baru. Sebagai contoh pada saaat ini berkembangnya perangkat lunak berbasis *web*.
2. Untuk mengurangi kompleksitas, biaya, dan waktu pemasaran. Sebagai contoh pemanfaatan dari perkembangan bahasa pemrograman tingkat tinggi.

3. Mengurangi cacat. Sebagai contoh dengan cara membuat standarisasi dari pengkodean yang tidak sesuai.

Evolusi perangkat lunak diperlukan untuk dapat menyelesaikan permasalahan yang timbul dari penggunaan perangkat lunak yang lama, yang terdiri dari:

1. Pembangunan Perangkat Lunak

   Yang termasuk dalam pembangunan perangkat lunak itu sendiri meliputi daur hidup perangkat lunak, yaitu permintaan *(requirement)*, spesifikasi, perancangan *(design)*, *testing* dan sebagainya.

2. Perawatan Perangkat Lunak

   Merupakan proses-proses untuk memperpanjang waktu pengguna sistem perangkat lunak yang ada, sehingga tetap dapat dipergunakan sebagaimana mestinya sesuai dengan baik.

3. Migrasi Perangkat Lunak

   Proses-proses yang memindahkan sistem yang ada ke dalam sistem yang baru, dikarenakan perkembangan dari kebutuhan perangkat lunak tersebut.

#### 2.2.3.2 Model Proses Perangkat Lunak

Model proses perangkat lunak merupakan representasi abstrak dari proses perangkat lunak. Setiap model proses merepresentasikan suatu proses dari sudut pandang tertentu sehingga hanya memberikan informasi parsial mengenai proses tersebut.

Racoon mengusulkan sebuah "Model Chaos" yang menggambarkan "*perkembangan perangkat lunak sebagai sebuah kesatuan dari pemakai ke perngembangan dan teknologi"*. Masing-masing model sudah ditandai dengan

cara tertentu sehingga (diharapkan) bisa membantu dalam kontrol dan koordinasi dari proyek perangkat lunak yang nyata. Dengan demikian, pada intinya semua model menunujukan karakteristik dari Model Chaos. Model-model proses perangkat lunak terdiri dari:

1. **Model *Waterfall* (Model Air Terjun/Model Sekuensial Linear)**

   Model *waterfall* adalah paradigma rekayasa perangkat lunak yang paling luas dipakai dan paling tua. Model *waterfall* mengusulkan sebuah pendekatan kepada perkembangan perangkat lunak yang sistematik dan sekuensial yang mulai pada tingkat dan kemajuan sistem pada seluruh analisis, desain, kode, pengujian, dan pemeliharaan. Setelah setiap tahap didefinisikan, tahap tersebut ‘diakhiri’ *(signed off)* dan pengembangan berlanjut ke tahap berikutnya.

2. **Model Prototipe**

   *Prototyping paradigm* dimulai dengan pengumpulan kebutuhan. Secara ideal prototipe berfungsi sebaga sebuah mekanisme untuk mengidentifikasi kebutuhan perangkat lunak. Bila prototipe yang sedang bekerja dibangun, maka pengembangan harus mempergunakan fragmen- fragmen program yang ada atau mengaplikasikan alat-alat bantu (contohnya *report generator, window manager*, dan lain- lain) yang memungkinkan program yang bekerja untuk dimunculkan secara tepat.

3. **Model RAD**

   *Rapid Aplication Development* (RAD) adalah sebuah model proses perkembangan perangkat lunak sekuensial linear yang menekankan siklus

perkembangan yang sangat pendek. Pendekatan RAD melingkupi fase-fase, diantaranya fase *business modeling*, *data modeling*, *process modeling*, *application modeling*, dan *testing and turnover*.

**4. Model Evolusioner**

Model Evolusioner adalah model iteratif. Model itu ditandai dengan tingkah laku yang memungkinkan perekayasa perangkat lunak mengembangkan versi perangkat lunak yang lebih lengkap sedikit demi sedikit. Berikut ini merupakan jenis-jenis model evolusioner:

a. Model Pertambahan

Model pertambahan menggabungkan elemen-elemen model sekuensial linear (diaplikasikan secara berulang) dengan filosofi prototipe iterarif.

b. Model Spiral

Model spiral (spiral model) yang pada awalnya diusulkan oleh Boehm, adalah model proses perangkat lunak yang evolusioner yang merangkai sifat iteratif dari prototipe dengan cara kontrol dan aspek sistematis dari model sekuensial linear.

c. Model Rakitan Komponen

Model rakitan komponen menggabungkan beberapa karakteristik model spiral. Model ini bersifat evolusioner, sehingga membutuhkan pendekatan iteratif untuk menciptakan perangkat lunak.

d. Model Perkembangan Konkuren

Model perkembangan konkuren atau disebut juga rekayasa konkuren yang digambarkan oleh A. Davis dan P. Sitaram. Model ini sering

digunakan sebagai paradigma bagi pengembangan aplikasi *client/server*. Sistem *client/server* dirancang dari serangkaian komponen fungsional.

e. Model Formal

Model metode formal mencakup sekumpulan aktivitas yang membawa kepada spesifikasi matematis perangkat lunak komputer. Metode formal memungkinkan perekayasa perangkat lunak mengkhususkan, mengembangkan, dan memverifikasi sistem berbasis komputer dengan menggunakan notasi matematis yang tepat. Variasi dalam pendekatan ini, disebut juga *clean room* rekayasa perangkat lunak, yang sedang diaplikasikan oleh banyak organisasi pengembang perangkat lunak.

f. Model Teknik Generasi Keempat

Bentuk teknik generasi keempat (4GT) mencakup serangkaian alat bantu perangkat lunak yang luas secara umum memiliki satu hal, masing-masing memungkinkan perekayasa perangkat lunak untuk mengkhususkan beberapa karakteristik perangkat lunak pada suatu tingkat yang tinggi. Paradigma 4GT untuk rekayasa perangkat lunak berfokus pada kemampuan spesifikasi perangkat lunak dengan menggunakan bentuk bahasa yang dikhususkan atau sebuah notasi grafik yang menggambarkan masalah yang akan dipecahkan ke dalam bentuk yang dapat dipahami oleh pelanggan.

### 2.2.4 Pengujian Perangkat Lunak

Pengujian perangkat lunak adalah elemen kritis dari jaminan kualitas perangkat lunak dan merepresentasikan spesifikasi, desain dan pengkodean.

Meningkatnya visibilitas perangkat lunak sebagai suatu elemen sistem dan "biaya" yang muncul akibat kegagalan perangkat lunak, memotivasi dilakukan perencanaan yang baik melalui pengujian yang teliti. Dalam melakukan uji coba ada 2 masalah penting yang akan dibahas, yaitu:

1. *Black box testing*
2. *White box testing*

#### 2.2.4.1 *Black Box Testing*

*Test case* ini bertujuan untuk menunjukkan fungsi perangkat lunak tentang cara beroperasinya, apakah pemasukan data keluaran telah berjalan sebagaimana yang diharapkan dan apakah informasi yang disimpan secara eksternal selalu dijaga kemutakhirannya.

#### 2.2.4.2 *White Box Testing*

*White box testing* adalah salah satu bentuk pengujian untuk meramalkan cara kerja perangkat lunak secara rinci, karenanya *logical path* (jalur logika) perangkat lunak akan ditest dengan menyediakan *test case* yang akan mengerjakan kumpulan kondisi secara spesifik. Secara sekilas dapat diambil kesimpulan *white box testing* merupakan petunjuk untuk mendapatkan program yang benar secara 100%.

### 2.2.5 Konsep Basis Data

Basis data adalah himpunan kelompok data (arsip) yang saling berhubungan yang diorganisasi sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah **[1]**.

### 2.2.6 Diagram Konteks

Diagram konteks merupakan alat pemodelan atau suatu diagram yang menggambarkan sistem berbasis komputer yang dirancang secara global dan merupakan suatu diagram alir data tingkat atas, dimana didalam diagram konteks ini menggambarkan seluruh jaringan, baik masukan maupun sebuah keluaran sebuah sistem.

Diagram konteks terdiri dari sebuah simbol proses tunggal yang menggambarkan sebuah sistem dan menunjukan data aliran utama untuk dan dari terminator. Diagram ini merupakan dasar yang digunakan untuk menentukan aliran data yang mengalir menuju sistem (*input* sistem) dan keluar dari sistem (*output* sistem), yang meliputi objek berupa kesatuan luar (*ekseternal entity*). Diagram konteks dapat mendefinisikan jangkauan proses penurunan sistem informasi yaitu menentukan apa yang menjadi bagian dari sistem informasi dan apa yang tidak menjadi bagian sistem informasi.

### 2.2.7 *Data Flow Diagram* (DFD)

Diagram Alir Data atau *Data Flow Diagram* (DFD) adalah suatu model yang menjelaskan arus data mulai dari pemasukan sampai dengan keluaran data. Tingkatan DFD dimulai dari diagram konteks yang menjelaskan secara umum suatu sistem atau batasan sistem aplikasi yang akan dikembangkan. Kemudian DFD dikembangkan menjadi DFD tingkat 0 atau 50 level 0 dan kemudian DFD level 0 dikembangkan lagi menjadi level 1 dan selanjutnya sampai sistem tersebut tergambarkan secara rinci menjadi tingkatan-tingkatan lebih rendah lagi.

DFD merupakan penurunan atau penjabaran dari diagram konteks. Dalam pembuatan DFD harus mengacu pada ketentuan sebagai berikut:

1. Setiap penurunan level yang lebih rendah harus mempresentasikan proses tersebut dalam spesifikasi proses yang jelas.
2. Penurunan dilakukan apabila memang diperlukan.
3. Tidak semua bagian dari sistem harus ditunjukkan dengan jumlah level yang sama.

### 2.2.8 UML *(Unified Modeling Language)*

UML *(Unified Modeling Language)* adalah sebuah bahasa untuk menetukan, visualisasi, kontruksi, dan mendokumentasikan artifact (bagian dari informasi yang digunakan atau dihasilkan dalam suatu proses pembuatan perangkat lunak). Artifact dapat berupa model, deskripsi atau perangkat lunak dari sistem perangkat lunak, seperti pada pemodelan bisnis dan sistem non perangkat lunak lainnya [**3**].

UML merupakan suatu kumpulan teknik terbaik yang telah terbukti sukses dalam memodelkan sistem yang besar dan kompleks. UML tidak hanya digunakan dalam proses pemodelan perangkat lunak, namun hampir dalam semua bidang yang membutuhkan pemodelan.

Bagian-bagian utama dari UML adalah view, diagram, model element, dan general mechanism.

A. *View*

*View* digunakan untuk melihat sistem yang dimodelkan dari beberapa aspek yang berbeda. View bukan melihat grafik, tapi merupakan suatu abstraksi yang

berisi sejumlah diagram. Beberapa jenis view dalam UML antara lain: *Use case view, logical view, component view, concurrency view,* dan *deployment view*.

*a) Use Case View*

Mendeskripsikan fungsionalitas sistem yang seharusnya dilakukan sesuai yang diinginkan *external actor*s. Aktor yang berinteraksi dengan sistem dapat berupa *user* atau sistem lainnya. *View* ini digambarkan dalam *Use case* diagrams dan kadang-kadang dengan *activity diagrams*. View ini digunakan terutama untuk pelanggan, perancang *(designer),* pengembang (*developer*), dan penguji sistem (*tester*).

*b) Logical View*

Mendeskripsikan bagaimana fungsionalitas dari sistem, struktur statis (*class, object*, dan *relationship*) dan kolaborasi dinamis yang terjadi ketika objek mengirim pesan ke objek lain dalam suatu fungsi tertentu. *View* ini digambarkan dalam *class diagrams* untuk struktur statis dan dalam *state, sequence, collaboration*, dan *activity diagram* untuk model dinamisnya. *View* ini digunakan untuk perancang *(designer)* dan pengembang (*developer*).

*c) Component View*

Mendeskripsikan implementasi dan ketergantungan modul. Komponen yang merupakan tipe lainnya dari kode modul diperlihatkan dengan struktur dan ketergantungannya juga alokasi sumber daya komponen dan informasi administratif lainnya. *View* ini digambarkan dalam componen *view* dan digunakan untuk pengembang (*developer*).

*d) Concurrency View*

Membagi sistem ke dalam proses dan prosesor. *View* ini digambarkan dalam diagram dinamis (*state, sequence, collaboration*, dan *activity diagrams*) dan diagram implementasi (*component* dan *deployment diagrams*) serta digunakan untuk pengembang *(developer)*, pengintegrasi *(integrator)*, dan penguji *(tester)*.

*e) Deployment View*

Mendeskripsikan fisik dari sistem seperti komputer dan perangkat *(nodes)* dan bagaimana hubungannya dengan lainnya. *View* ini digambarkan dalam deployment diagrams dan digunakan untuk pengembang (*developer*), pengintegrasi (*integrator*), dan penguji (*tester*).

B. Diagram

Diagram berbentuk grafik yang menunjukkan simbol elemen model yang disusun untuk mengilustrasikan bagian atau aspek tertentu dari sistem. Sebuah diagram merupakan bagian dari suatu *view* tertentu dan ketika digambarkan biasanya dialokasikan untuk *view* tertentu. Adapun jenis diagram antara lain:

*a) Use Case Diagram*

*Use case* adalah abstraksi dari interaksi antara sistem dan aktor. *Use case* bekerja dengan cara mendeskripsikan tipe interaksi antara *user* sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. *Use case* merupakan konstruksi untuk mendeskripsikan bagaimana sistem akan terlihat di mata *user*. Sedangkan *Use case* diagram memfasilitasi komunikasi diantara analis dan pengguna serta antara analis dan *client*.

*b) Class Diagram*

*Class* adalah dekripsi kelompok obyek-obyek dengan *property*, perilaku (operasi) dan relasi yang sama. Sehingga dengan adanya *class diagram* dapat memberikan pandangan global atas sebuah sistem. Hal tersebut tercermin dari *class-class* yang ada dan relasinya satu dengan yang lainnya. Sebuah sistem biasanya mempunyai beberapa *class diagram*. *Class diagram* sangat membantu dalam visualisasi struktur kelas dari suatu sistem.

*c) Component Diagram*

*Component software* merupakan bagian fisik dari sebuah sistem, karena menetap di komputer tidak berada di benak para analis. Komponen merupakan implementasi *software* dari sebuah atau lebih *class*. Komponen dapat berupa *source code*, komponent biner, atau *executable component*. Sebuah komponent berisi informasi tentang *logic class* atau *class* yang diimplementasikan sehingga membuat pemetaan dari *logical view* ke *component view*. Sehingga *component diagram* merepresentasikan dunia riil yaitu *component software* yang mengandung *component*, *interface* dan *relationship*.

*d) Deployment Diagram*

Menggambarkan tata letak sebuah sistem secara fisik, menampakkan bagian-bagian *software* yang berjalan pada bagian-bagian *hardware*, menunjukkan hubungan komputer dengan perangkat (*nodes*) satu sama lain dan jenis hubungannya. Di dalam *nodes, executeable component* dan objek yang dialokasikan untuk memperlihatkan unit perangkat lunak yang dieksekusi oleh *node* tertentu dan ketergantungan komponen.

*e) State Diagram*

Menggambarkan semua *state* (kondisi) yang dimiliki oleh suatu objek dari suatu *class* dan keadaan yang menyebabkan *state* berubah. Kejadian dapat berupa objek lain yang mengirim pesan. *State class* tidak digambarkan untuk semua *class*, hanya yang mempunyai sejumlah *state* yang terdefinisi dengan baik dan kondisi *class* berubah oleh *state* yang berbeda.

*f) Sequence Diagram*

*Sequence Diagram* digunakan untuk menggambarkan perilaku pada sebuah *scenario*. Kegunaannya untuk menunjukkan rangkaian pesan yang dikirim antara objek juga interaksi antara objek, sesuatu yang terjadi pada titik tertentu dalam eksekusi sistem.

*g) Collaboration Diagram*

Menggambarkan kolaborasi dinamis seperti *sequence diagrams*. Dalam menunjukkan pertukaran pesan, *collaboration diagrams* menggambarkan objek dan hubungannya (mengacu ke konteks). Jika penekannya pada waktu atau urutan gunakan *sequence diagrams*, tapi jika penekanannya pada konteks gunakan *collaboration diagram*.

*h) Activity Diagram*

Menggambarkan rangkaian aliran dari aktivitas, digunakan untuk mendeskripsikan aktifitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktifitas lainnya seperti *Use case* atau interaksi **[9]**.

### 2.2.9 Kamus Data

Kamus Data (KD) atau *Data Dictionary* (DD) atau disebut juga dengan istilah *systems data dictionary* adalah katalog fakta tentang data dan kebutuhan-kebutuhan informasi dari suatu sistem informasi. Dengan menggunakan KD, analis sistem dapat mendefinisikan data yang mengalir di sistem dengan lengkap. KD dibuat pada tahap analisis sistem dan digunakan baik pada tahap analisis maupun pada tahap perancangan sistem. Pada tahap analisis, KD dapat digunakan sebagai alat komunikasi antara analisis sistem dengan pemakai sistem tentang data yang mengalir di sistem, yaitu tentang data yang masuk ke sistem dan tentang informasi yang dibutuhkan oleh pemakai sistem. Pada saat perancangan sistem, KD digunakan untuk merancang input, merancang laporan-laporan dan database. KD dibuat berdasarkan arus data yang ada di Diagram Arus Data (DAD). Arus data di DAD sifatnya adalah global, hanya ditunjukkan nama dan arus datanya saja.

### 2.2.10 ERD (*Entity Relationship Diagram*)

*Entity Relationship Diagram* atau biasa dikenal dengan diagram E-R secara grafis menggambarkan isi sebuah *database*. Diagram ini memiliki dua komponen utama yaitu entitas (*entity*) dan relasi (*relation*). Kedua komponen ini dideskripsikan lebih jauh melalui sejumlah atribut/properti.

1. Entity (*Entitas*)

Pada E-R diagram, *entity* digambarkan dengan sebuah bentuk persegi panjang. Entitas merupakan individu yang mewakili sesuatu yang nyata (eksistensinya) dan dapat dibedakan dari sesuatu yang lain. Sederhananya entitas

menunjuk pada individu suatu objek, sedang himpunan entitas menunjuk pada rumpun (*family*) dari individu tersebut.

2. Relasi (*Relationship*)

Pada E-R diagram, *relationship* dapat digambarkan dengan sebuah bentuk belah ketupat. Relasi menunjukan adanya hubungan diantara sejumlah entitas yang berasal dari himpunan entitas yang berbeda.

3. Atribut

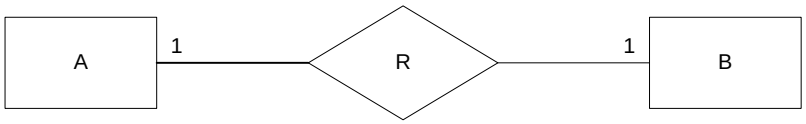
Setiap entitas pasti memiliki atribut yang mendeskripsikan karakteristik (*property*) dari entitas tersebut. Penentuan/pemilihan atribut-atribut yang relevan bagi sebuah entitas merupakan hal enting lainnya dalam pembentukan model data.

4. Kardinalitas

Kardinalitas relasi menunjukan jumlah maksimum tupel yang dapat berelasi dengan entitas yang lainnya. Dari sejumlah kemungkinan banyaknya hubungan yang terjadi dari entitas, kardinalitas relasi merujuk kepada hubungan maksimum yang terjadi dari entitas yang satu ke entitas yang lainnya dan begitu juga sebaliknya. Macam-macam kardinalitas relasi yaitu :

*a. One to One*

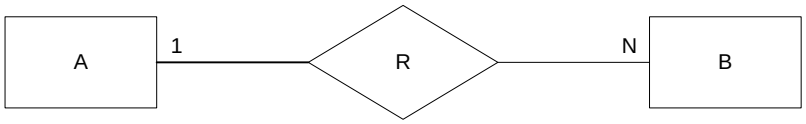
Yang berarti setiap entitas pada himpunan entitas A berhubungan dengan paling banyak dengan satu entitas pada himpunan entitas B, dan begitu juga sebaliknya.




**Gambar 2.4** ***One to One***

*b. One to Many*

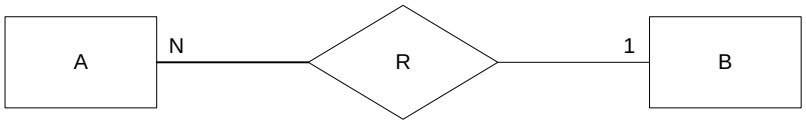
Yang berarti setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B, tetapi tidak sebaliknya, di mana setiap entitas pada himpunan entitas B berhubungan dengan paling banyak dengan satu entitas pada himpunan entitas A.




**Gambar 2.5** ***One to Many***

c. *Many To One*

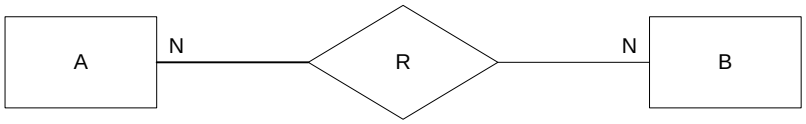
Yang berarti setiap entitas pada himpunan A berhubungan dengan paling banyak dengan satu entitas pada himpunan entitas B, tetapi tidak sebaliknya, di mana setiap entitas pada himpunan entitas A berhubungan dengan paling banyak satu entitas pada himpunan entitas B.




**Gambar 2.6** ***Many to One***

d. *Many to many*

Yang berarti setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B, dan demikian juga sebaliknya.




**Gambar 2.7 *Many to Many***

### 2.2.11 HTML

HTML merupakan kependekan dari *Hyper Text Markup Language*. Dokumen HTML adalah file teks murni yang dapat dibuat dengan editor teks sembarang yang dikenal sebagai *web page* dan merupakan dokumen yang disajikan dalam *browser web* yang umumnya berisi informasi atau pun *interface* aplikasi di dalam internet **[6]**.

### 2.2.12 PHP

PHP diciptakan oleh Rasmus Lerdorf pada tahun 1994 dan bersifat *open source*. Sampai bulan januari 2007, PHP sudah digunakan oleh kurang lebih 20 Juta *domain* dan terus berkembang sampai saat ini. PHP merupakan singkatan dari *Hypertext Preprocessor*, adalah sebuah bahasa *scripting* yang terpasang pada HTML. Sebagian besar *sintaks* mirip dengan bahasa *C, Java* dan *Perl*, ditambah beberapa fungsi PHP yang spesifik. Tujuan utama bahasa ini adalah untuk memungkinkan perancang *web* menulis halaman *web* dinamis dengan cepat. PHP merupakan bahasa pemograman *web* yang bersifat *server-side* HTML atau *embedded scripting*, di mana *script*-nya menyatu dengan HTML dan berada di

*server*. Artinya adalah *sintaks* dan perintah-perintah yang kita berikan akan sepenuhnya dijalankan di *server* tetapi disertakan HTML biasa. PHP dikenal sebagai bahasa *scripting* yang menyatu dengan *tag* HTML, dieksekusi di *server* dan digunakan untuk membuat halaman *web* yang dinamis seperti ASP (*Active Server Pages*) dan JSP (*Java Server Pages*).

Seluruh aplikasi berbasis *web* dapat dibuat dengan PHP. Namun kekuatan yang paling utama PHP adalah pada konektivitasnya dengan *system database* di dalam *web*. Sistem *database* yang dapat didukung oleh PHP adalah *Oracle, MySQL, Sybase, PostgreSQL.*

PHP dapat berjalan di berbagai *system* operasi seperti windows 98/NT, UNIX/LINUX, solaris maupun macintosh. PHP merupakan *software* yang *open source* yang dapat anda *download* secara gratis dari situs resminya yaitu *http://www.php.net*, ataupun dari situs-situs yang menyediakan software tersebut.

Software ini juga dapat berjalan pada *web* server seperti PWS (*Personal Web Server*), Apache, IIS, *AOLServer*, fhttpd, phttpd dan sebagainya. PHP juga merupakan bahasa pemograman yang dapat kita kembangkan sendiri seperti menambah fungsi-fungsi baru. Keunggulan lainnya dari PHP adalah PHP juga mendukung komunikasi dengan layanan seperti protocol IMAP, SNMP, NNTP, POP3 bahkan HTTP. PHP dapat diinstal sebagai bagian atau modul dari *apache web server* atau sebagai CGI *script* yang mandiri.

Banyak keuntungan yang dapat diperoleh jika menggunakan PHP sebagai modul dari *apache* di antaranya adalah :

1. Tingkat keamanan yang cukup tinggi

2. Waktu eksekusi yang lebih cepat dibandingkan dengan bahasa pemograman *web* lainnya yang berorientasi pada *server-side scripting.*
3. Akses ke sistem *database* yang lebih fleksibel seperti MySQL.

### 2.2.13 GPS (*Global Positioning System*)

GPS (*Global Positioning System*) adalah sistem satelit navigasi dan penentuan posisi dengan menggunakan satelit. Nama formalnya adalah NAVSTAR GPS, kependekan dari "*NAVigation Satellite Timing Adn Ranging Global Positioning System"* . Sistem ini digunakan oleh banyak orang sekaligus dalam segala cuaca in, didesain untuk memberikan posisi dan kecepatan tiga dimensi yang teliti, dan juga informasi mengenai waktu, secara kontinyu di seluruh dunia.

Pada dasarnya GPS terdiri dari tiga segmen utama, yaitu segmen ruang angkasa (*space segment*) yang terutama terdiri dari satelit-satelit GPS, segmen sistem kontrol (*control system segment*) yang terdiri dari stasiun-stasiun pemonitor dan pengontrol satelit, dan segmen pemakai (*user segment*) yang terdiri dari pemakai GPS termasuk alat-alat penerima dan pengolah sinyal dan data GPS.

GPS dapat memberikan informasi mengenai posisi, kecepatan, dan waktu secara tepat dan teliti, dan murah dimana saja di Bumi pada setiap waktu, siang maupun malam tanpa tergantung pada kondisi cuaca. Sampai saat ini GPS adalah satu-satunya sistem navigasi yang mempunyai karakteristik prima seperti itu [**5**].

### 2.2.14 Metode Absolut

Penentuan posisi secara absolut (*absolut positioning*) adalah metode penentuan posisi yang paling mendasar dari GPS. Bahkan dapat dikatakan bahwa

metode ini adalah metode penentuan posisi dengan GPS yang direncanakan pada awalnya oleh pihak militer Amerika untuk memberikan pelayanan navigasi terutama bagi personil dan wahana militer mereka. Catatan yang perlu diperhatikan dalam penentuan posisi secara absolut yaitu :

1. Disebut juga metode *point positioning*, karena penentuan posisi dapat dilakukan per titik tanpa tergantung titik lainnya.
2. Posisi ditentukan dalam sistem WGS-84 terhadap pusat massa bumi.
3. Prinsip penentuan posisi adalah reseksi dengan jarak ke beberapa satelit secara simultan.
4. Untuk penentuan posisi hanya memerlukan satu *receiver* GPS, dan tipe navigasi *receiver* yang umum dapat digunakan untuk keperluan ini adalah tipe navigasi atau dinamakan tipe genggam (*hand held*).
5. Titik yang ditentukan posisinya bisa dalam keadaan diam (moda statik) ataupun dalam keadaan bergerak (moda kinematik).
6. Ketelitian posisi yang diperoleh sangat tergantung pada tingkat ketelitian data serta geometri satelit.
7. Metode ini tidak dimaksudkan untuk penentuan posisi yang teliti.

### *2.2.15 Android*

#### 2.2.15.1 Pengertian *Android*

Android adalah kumpulan perangkat lunak yang ditujukan bagi perangkat bergerak mencakup sistem operasi, middleware, dan aplikasi kunci. *Android Standart Development Kid* (SDK) menyediakan perlengkapan dan *Application*

*Programming Interface* (API) yang diperlukan untuk mengembangkan aplikasi pada platform Android menggunakan bahasa pemrograman Java.

Android dikembangkan oleh Google bersama *Open Handset Allience* (OHA) yaitu aliansi perangkat selular terbuka yang terdiri dari 47 perusahaan *Hardware*, *Software* dan perusahaan telekomunikasi ditujukan untuk mengembangkan standar terbuka bagi perangkat selular.

#### 2.2.15.2 Sejarah dan Perkembangan *Android*

Pada mulanya terdapat berbagai macam sistem operasi pada perangkat selular, diantaranya sistem operasi *Symbian, Microsoft Windows Mobile*, *Mobile Linux, iPhone*, dan sistem operasi lainnya. Namun diantara sistem operasi yang ada belum mendukung standar dan penerbitan API yang dapat dimanfaatkan secara keseluruhan dan dengan biaya yang murah. Kemudian *Google* ikut berkecimpung didalamnya dengan platform *Android*, yang menjanjikan keterbukaan, keterjangkauan, *open source*, dan *framework* berkualitas. Pada tahun 2005, *Google* mengakuisisi perusahaan *Android* Inc. untuk memulai pengembangan platform *Android*. Dimana terlibat dalam pengembangan ini Andy Rubin, Rich Miner, Nick Sears, dan Chris White. Pada pertengahan 2007 sekelompok pemimpin industri bersama-sama membentuk aliansi perangkat selular terbuka, *Open Handset Alliance* (OHA). Bagian dari tujuan aliansi ini adalah berinovasi dengan cepat dan menanggapi kebutuhan konsumen dengan lebih baik, dengan produk awalnya adalah platform *Android*. Dimana *Android* dirancang untuk melayani kebutuhan operator telekomunikasi, manufaktur handset, dan pengembang aplikasi. OHA berkomitmen untuk membuat *Android open source* dengan lisensi *Apache* versi 2.0.

Android pertama kali diluncurkan pada 5 November 2007, dan *smartphone* pertama yang menggunakan sistem operasi Android dikeluarkan oleh T-Mobile dengan sebutan G1 pada bulan September 2008. Hingga saat ini *Android* telah merilis beberapa versi *Android* untuk menyempurnakan versi sebelumnya. Selain berdasarkan penomoran, pada setiap versi *Android* terdapat kode nama berdasarkan nama-nama kue. Hingga saat ini sudah terdapat beberapa versi yang telah diluncurkan, diantaranya: versi 1.5 dirilis pada 30 April 2009 diberi nama *Cupcake*, versi 1.6 dirilis pada 15 September 2009 diberi nama *Donut*, dan versi terakhir 2.0 dirilis pada 26 Oktober 2009 diberi nama *Éclair*.

**2.2.15.3 *Libraries***

Android menggunakan beberapa paket pustaka yang terdapat pada C/C++ dengan standar *Berkeley Software Distribution* (BSD) hanya setengah dari yang aslinya untuk tertanam pada kernel Linux. Beberapa pustaka diantaranya:

a. *Media library* untuk memutar dan merekam berbagai macam format audio dan video.
b. *Surface Manager* untuk mengatur hak akses layer dari berbagai aplikasi
c. *Graphic Library* termasuk didalamnya *SGL* dan *OpenGL*, untuk tampilan 2D dan 3D.
d. *SQLite* untuk mengatur relasi database yang digunakan pada aplikasi
e. *SSl* dan *WebKit* untuk *browser* dan keamanan internet.

Pustaka-pustaka tersebut bukanlah aplikasi yang berjalan sendiri, namun hanya dapat digunakan oleh program yang berada di level atasnya. Sejak versi Android 1.5, pengembang dapat membuat dan menggunakan pustaka sendiri menggunakan *Native Development Toolkit* (NDK).

#### 2.2.15.4 Android Runtime

Pada Android tertanam paket pustaka inti yang menyediakan sebagian besar fungsi Android. Inilah yang membedakan Android dibandingkan dengan *system* operasi lain yang juga mengimplementasikan Linux. *Android Runtime* merupakan mesin virtual yang membuat aplikasi *Android* menjadi lebih tangguh dengan paket pustaka yang telah ada. Dalam *Android Runtime* terdapat 2 bagian utama, diantaranya:

a. Pustaka Inti, Android dikembangkan melalui bahasa pemrograman Java, tapi *Android Runtime* bukanlah mesin virtual Java. Pustaka inti Android menyediakan hampir semua fungsi yang terdapat pada pustaka Java serta beberapa pustaka khusus Android.
b. Mesin Virtual Dalvik, Dalvik merupakan sebuah mesin virtual yang dikembangkan oleh Dan Bornstein yang terinspirasi dari nama sebuah perkampungan yang berada di Iceland. Dalvik hanyalah interpreter mesin virtual yang mengeksekusi file dalam format Dalvik Executable (*.dex). Dengan format ini Dalvik akan mengoptimalkan efisiensi penyimpanan dan pengalamatan memori pada file yang dieksekusi. Dalvik berjalan di atas kernel Linux 2.6, dengan fungsi dasar seperti threading dan manajemen memori yang terbatas.

#### 2.2.15.5 Application *Framework*

Kerangka aplikasi menyediakan kelas-kelas yang dapat digunakan untuk mengembangkan aplikasi *Android*. Selain itu, juga menyediakan abstraksi generic untuk mengakses perangkat, serta mengatur tampilan user interface dan sumber

daya aplikasi. Bagian terpenting dalam kerangka aplikasi *Android* adalah sebagai berikut :

1. *Activity Manager*, berfungsi untuk mengontrol siklus hidup aplikasi dan menjaga keadaan ”*Backstack*“ untuk navigasi penggunaan.
2. *Content Providers*, berfungsi untuk merangkum data yang memungkinkan digunakan oleh aplikasi lainnya, seperti daftar nama.
3. *Resuource Manager*, untuk mengatur sumber daya yang ada dalam program. Serta menyediakan akses sumber daya diluar kode program, seperti karakter, grafik, dan file layout.
4. *Location Manager*, berfungsi untuk memberikan informasi detail mengenai lokasi perangkat Android berada.
5. *Notification Manager*, mencakup berbagai macam peringatan seperti, pesan masuk, janji, dan lain sebagainya yang akan ditampilkan pada status bar.

### 2.2.16 *Google Maps*

*Google Maps* merupakan sebuah layanan peta dunia virtual berbasis web yang disediakan *Google*. Layanan ini gratis dan bisa ditemukan serta diakses di http://maps.google.com. Google Maps menawarkan peta yang dapat digeser (*panned*), diperbesar (*zoom in*), diperkecil (*zoom out*) dan dapat diganti dalam beberapa mode (*map, satellite, hybryd,* dan lain-lain), fitur pencarian rute (*routing*), penunjuk arah dari satu objek peta ke objek yang lain (*direction*), dan juga pencari tempat (*place*). *Google Maps* masih berada dalam tahap beta, dan amsih terus dikembangkan dengan data yang selalu diperbaharui secara berkala **[8]**.

### 2.2.17 *JavaScript Object Notation* (*Json*)

*JSON (JavaScript Object Notation)* adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (generate) oleh komputer. Format ini dibuat berdasarkan bagian dari Bahasa Pemprograman *JavaScript*, Standar ECMA-262 Edisi ke-3 Desember 1999. JSON merupakan format teks yang tidak bergantung pada bahasa pemprograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran-data.

### 2.2.18 *Adobe Dreamweaver*

*Adobe Dreamweaver* (dulunya *Macromedia Dreamweaver*) adalah pengembangan web aplikasi yang awalnya dibuat oleh *Macromedia* , dan sekarang dikembangkan oleh *Adobe Systems* , yang mengakuisisi *Macromedia* tahun 2005. *Dreamweaver* tersedia untuk *Mac* dan *Windows* sistem operasi . versi terbaru telah dimasukkan dukungan untuk web teknologi seperti CSS , *JavaScript*, dan berbagai *server*-side scripting bahasa dan kerangka termasuk ASP, *ColdFusion* , dan PHP.

*Dreamweaver* yang WYSIWYG mode dapat menyembunyikan HTML code rincian halaman dari pengguna, sehingga memungkinkan untuk non-coders untuk membuat halaman web dan situs. *Dreamweaver* memungkinkan pengguna untuk melihat situs di web browser secara lokal-instal. Ia juga memiliki alat manajemen situs seperti FTP / SFTP dan WebDAV mentransfer *file* dan fitur sinkronisasi,

kemampuan untuk mencari dan mengganti baris teks atau kode oleh istilah pencarian dan ekspresi reguler di seluruh situs, dan fitur template yang memungkinkan satu sumber *update* kode bersama dan tata letak di seluruh situs tanpa *server-side* termasuk atau *scripting*. Panel perilaku juga memungkinkan penggunaan *JavaScript* dasar tanpa pengetahuan *coding*, dan integrasi dengan *Adobe* kerangka AJAX sigap menawarkan akses mudah ke konten dinamis yang dihasilkan dan antarmuka. *Dreamweaver* dapat menggunakan pihak ketiga "*Extensions*" untuk memperluas fungsi inti dari aplikasi, yang setiap pengembang web bisa menulis (terutama dalam HTML dan *JavaScript* ) **[4]**.

### 2.2.19 *Mysql*

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (*database management system*) atau DBMS yang *multithread*, *multi-user*, dengan sekitar 6 juta instalasi di seluruh dunia. MySQL AB membuat MySQL tersedia sebagai perangkat lunak gratis dibawah lisensi GNU *General Public License* (GPL) ditunjukan pada Gambar 2.8, tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL.




**Gambar 2.8 *MySQL***

Tidak sama dengan proyek-proyek seperti *Apache*, dimana perangkat lunak dikembangkan oleh komunitas umum, dan hak cipta untuk kode sumber dimiliki oleh penulisnya masing-masing, MySQL dimiliki dan disponsori oleh sebuah perusahaan komersial *Swedia MySQL AB*, dimana memegang hak cipta hampir atas semua kode sumbernya. Kedua orang Swedia dan satu orang *Finlandia* yang mendirikan MySQL AB adalah: *David Axmark*, *Allan Larsson*, dan *Michael "Monty" Widenius*.

a. Sistem Manajemen Basis Data Relasional

MySQL adalah sebuah implementasi dari sistem manajemen basisdata relasional (RDBMS) yang didistribusikan secara gratis dibawah lisensi GPL (*General Public License*). Setiap pengguna dapat secara bebas menggunakan MySQL, namun dengan batasan perangkat lunak tersebut tidak boleh dijadikan produk turunan yang bersifat komersial. MySQL sebenarnya merupakan turunan salah satu konsep utama dalam basisdata yang telah ada

sebelumnya; SQL (*Structured Query Language*). SQL adalah sebuah konsep pengoperasian basisdata, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis.

Kehandalan suatu sistem basisdata (DBMS) dapat diketahui dari cara kerja pengoptimasi-nya dalam melakukan proses perintah-perintah SQL yang dibuat oleh pengguna maupun program-program aplikasi yang memanfaatkannya. Sebagai peladen basis data, MySQL mendukung operasi basis data transaksional maupun operasi basisdata non-transaksional. Pada modus operasi non-transaksional, MySQL dapat dikatakan unggul dalam hal unjuk kerja dibandingkan perangkat lunak peladen basisdata kompetitor lainnya. Namun demikian pada modus non-transaksional tidak ada jaminan atas reliabilitas terhadap data yang tersimpan, karenanya modus non-transaksional hanya cocok untuk jenis aplikasi yang tidak membutuhkan reliabilitas data seperti aplikasi *blogging* berbasis *web* (*wordpress*), CMS, dan sejenisnya. Untuk kebutuhan sistem yang ditujukan untuk bisnis sangat disarankan untuk menggunakan modus basisdata transaksional, hanya saja sebagai konsekuensinya unjuk kerja MySQL pada modus transaksional tidak secepat unjuk kerja pada modus non-transaksional.

b. Keistimewaan MySql

MySQL memiliki beberapa keistimewaan, antara lain :

1. **Portabilitas**. MySQL dapat berjalan stabil pada berbagai sistem operasi seperti *Windows*, *Linux*, *FreeBSD*, *Mac Os X Server*, *Solaris*, *Amiga*, dan masih banyak lagi.
2. **Perangkat lunak sumber terbuka**. MySQL didistribusikan sebagai perangkat lunak sumber terbuka, dibawah lisensi GPL sehingga dapat digunakan secara gratis.
3. **Multi-*user***. MySQL dapat digunakan oleh beberapa pengguna dalam waktu yang bersamaan tanpa mengalami masalah atau konflik.
4. ***Performance tuning***, MySQL memiliki kecepatan yang menakjubkan dalam menangani *query* sederhana, dengan kata lain dapat memproses lebih banyak SQL per satuan waktu.
5. **Ragam tipe data**. MySQL memiliki ragam tipe data yang sangat kaya, seperti *signed / unsigned integer, float, double, char, text, date, timestamp,* dan lain-lain.
6. **Perintah dan Fungsi**. MySQL memiliki operator dan fungsi secara penuh yang mendukung perintah *Select* dan *Where* dalam perintah (*query*).
7. **Keamanan**. MySQL memiliki beberapa lapisan keamanan seperti *level subnetmask*, nama *host*, dan izin akses *user* dengan sistem perizinan yang mendetail serta sandi terenkripsi.
8. **Skalabilitas dan Pembatasan**. MySQL mampu menangani basis data dalam skala besar, dengan jumlah rekaman (*records*) lebih dari 50 juta dan 60 ribu tabel serta 5 milyar baris. Selain itu batas indeks yang dapat ditampung mencapai 32 indeks pada tiap tabelnya.

9. **Konektivitas**. MySQL dapat melakukan koneksi dengan klien menggunakan protokol TCP/IP, *Unix soket (UNIX)*, atau *Named Pipes* (*NT*).
10. **Lokalisasi**. MySQL dapat mendeteksi pesan kesalahan pada klien dengan menggunakan lebih dari dua puluh bahasa. Meski pun demikian, bahasa Indonesia belum termasuk di dalamnya.
11. **Antar Muka**. MySQL memiliki antar muka (*interface*) terhadap berbagai aplikasi dan bahasa pemrograman dengan menggunakan fungsi API (*Application Programming Interface*).
12. **Klien dan Peralatan**. MySQL dilengkapi dengan berbagai peralatan (tool) yang dapat digunakan untuk administrasi basis data, dan pada setiap peralatan yang ada disertakan petunjuk *online*.
13. **Struktur tabel**. MySQL memiliki struktur tabel yang lebih fleksibel dalam menangani ALTER TABLE, dibandingkan basis data lainnya semacam *PostgreSQL* ataupun *Oracle*.

c. Bahasa Pemrograman

Terdapat beberapa API (*Application Programming Interface*) tersedia yang memungkinkan aplikasi-aplikasi komputer yang ditulis dalam berbagai bahasa pemrograman untuk dapat mengakses basis data MySQL antara lain: bahasa pemrograman C, C++, C#, bahasa pemrograman *Eiffel*, bahasa pemrograman *Smalltalk*, bahasa pemrograman Java, bahasa pemrograman *Lisp*, *Perl*, PHP, bahasa pemrograman *Python*, *Ruby*, *REALbasic* dan Tcl. Sebuah antarmuka ODBC memanggil MyODBC yang memungkinkan setiap bahasa

pemrograman yang mendukung ODBC untuk berkomunikasi dengan basis data MySQL. Kebanyakan kode sumber MySQL dalam ANSI C.

d. Penggunaan

MySQL sangat populer dalam aplikasi web seperti MediaWiki (perangkat lunak yang dipakai *Wikipedia* dan proyek-proyek sejenis) dan PHP-Nuke dan berfungsi sebagai komponen basis data dalam LAMP. Popularitas sebagai aplikasi web dikarenakan kedekatannya dengan popularitas PHP, sehingga seringkali disebut sebagai *Dynamic Duo* **[4]**.

### 2.2.20 WAMP *Server*

Wamp *Server* merupakan lingkungan pengembangan web Windows. Hal ini memungkinkan Anda untuk membuat aplikasi web dengan Apache, PHP dan database MySQL. Hal ini juga dilengkapi dengan PHP*MyAdmin* untuk dengan mudah mengelola database.

Wamp *Server* menginstal secara otomatis (*installer*), dan penggunaannya sangat intuitif. Anda akan dapat menyetel *server* Anda tanpa menyentuh pengaturan *file*.

Wamp *Server* merupakan satu-satunya solusi paket yang akan memungkinkan untuk mereproduksi *server* produksi. Wamp *Server* juga memiliki trayicon untuk mengelola *server* dan pengaturannya ditunjukan pada Gambar 2.9.




**Gambar 2.9 Wamp *server***

### 2.2.21 *Android Debug Bridge* (*ADB*)

ADB adalah ***Android Debug Bridge*** dari *Google*. Ini adalah program command-line yang dijalankan dari PC Anda, dengan ponsel yang terhubung melalui USB, yang bertujuan untuk menjalankan perintah yang harus dilakukan dengan telepon. Misalnya, seseorang dapat menginstal APK pada telepon dengan menggunakan perintah '*adb install apkname.apk*'.

### 2.2.22 *Java Development Kit* (*JDK*)

JDK singkatan dari *Java Development Kit* , merupakan seperangkat kode, librari, form dan paket yang berguna untuk membuat aplikasi atau applet java. JDK memiliki 3 kelompok pengembangan , yaitu :

a. J2SE (*Java 2 Standard Edition*, untuk kebutuhan standard)

b. J2EE (*Java 2 Enterprise Edition*, untuk kebutuhan skala besar)

c. J2ME (*Java 2 Mobile Edition*, untuk kebutuhan pemrograman mobile)

### 2.2.23 *Android Software Development Kit*

*Software Development Kit* (SDK) adalah suatu tools bantu untuk mendevelop suatu software, dalam topik ini tentunya untuk aplikasi berbasis *Android*, untuk menguji aplikasi yang sudah dibuat tanpa harus ada handset android.

### 2.2.24 *Eclipse*

*Eclipse* adalah sebuah project Java open source yang diprakarsai oleh IBM, di mana sebelumnya merupakan hasil kerja OTI, sebuah perusahaan yang diakusisi oleh IBM karena keahliannya membuat aplikasi yang dikenal dengan *VisualAge*. Eclipse dapat dikatakan adalah *VisualAge* versi baru. Perbedaannya *VisualAge* bersifat komersial, sedangkan Eclipse bersifat *open source* dengan lisensinya CPL (*Common Public License*). Selain itu, *Eclipse* juga memposisikan sebagai IDE multilanguage, jadi selain bahasa Java, juga mendukung bahasa C++, *Cobol*, *Fortran*. Dapat dikatakan Eclipse ini menjadi sebuah *single environment* untuk pengembangan aplikasi yang berjalan di banyak platform **[7]**.