BAB II

LANDASAN TEORI

2.1. Aplikasi Mobile

Aplikasi *mobile* saat ini sangat dibutuhkan karena alat-alat telekomunikasi yang tersebar di seluruh dunia membutuhkan aplikasi-aplikasi yang dapat memepermudah pekerjaan penggunanya dimanapun dan kapanpun terutama dalam hal informasi. Informasi sendiri menurut Sugiono (2001) merupakan hasil dari proses analisis, manipulasi dan presentasi data untuk mendukung proses pengambilan keputusan.

2.1.1. Pengertian Aplikasi

Menurut Buyens (2001) aplikasi adalah satu unit perangkat lunak yang dibuat untuk melayani kebutuhan akan beberapa aktivitas.

Jika ingin mengembangkan program aplikasi sendiri, maka untuk menulis program aplikasi tersebut, dibutuhkan suatu bahasa pemrograman, yaitu *language software*, yang dapat berbentuk *assembler*, *compiler* ataupun *interpreter*. Jadi *language software* merupakan bahasanya dan program yang ditulis merupakan program aplikasinya. *Language software* berfungsi agar dapat menulis program dengan bahasa yang lebih mudah, dan akan menterjemahkannya ke dalam bahasa mesin supaya bisa dimengerti oleh komputer. Bila hendak mengembangkan suatu program aplikasi untuk memecahkan permasalahan yang besar dan rumit, maka

supaya program aplikasi tersebut dapat berhasil dengan baik, maka dibutuhkan prosedur dan perencanaan yang baik dalam mengembangkannya.

Sekarang, banyak sekali program-program aplikasi yang tersedia dalam bentuk paket-paket program. Ini adalah program-program aplikasi yang sudah ditulis oleh orang lain atau perusahaan-perusahaan perangkat lunak. Beberapa perusahaan perangkat lunak telah memproduksi paket-paket perangkat lunak yang mempunyai reputasi internasional. Program-program paket tersebut dapat diandalkan, dapat memenhi kebutuhan pemakai, dirancang dengan baik, relatif bebas dari kesalahan-kesalahan, *user friendly* (mudah digunakan), mempunyai dokumentasi manual yang memadai, mampu dikembangkan untuk kebutuhan mendatang, dan didukung perkembangannya. Akan tetapi, bila permasalahannya bersifat khusus dan unik, sehingga tidak ada paket-paket program yang sesuai untuk digunakan, maka dengan terpaksa harus mengembangkan program aplikasi itu sendiri.

2.1.2. Pengertian Mobile

Mobile diartikan sebagai perpindahan yang mudah dari satu tempat ke tempat yang lain, misalnya telepon *mobile* berarti bahwa terminal telepon yang dapat berpindah dengan mudah dari satu tempat ke tempat lain tanpa terjadi pemutusan atau terputusnya komunikasi.

2.1.3. Pengertian Aplikasi Mobile

Menurut Buyens (2001) aplikasi *mobile* berasal dari kata *application* dan *mobile*. *Application* yang artinya penerapan, lamaran, penggunaan. Secara istilah aplikasi adalah program siap pakai yang direka untuk melaksanakan suatu fungsi bagi pengguna atau aplikasi yang lain dan dapat digunakan oleh sasaran yang dituju sedangkan *mobile* dapat di artikan sebagai perpindahan dari suatu tempat ke tempat yang lain.

Maka aplikasi *mobile* dapat di artikan sebuah program aplikasi yang dapat dijalankan atau digunakan walaupun pengguna berpindah – pindah dari satu tempat ke tempat yang lain serta mempunyai ukuran yang kecil. Aplikasi *mobile* ini dapat di akses melalui perangkat nirkabel, pager, PDA, telepon seluler, *smartphone*, dan perangkat sejenisnya.

2.2. Karakteristik Perangkat Mobile

Perangkat *mobile* memiliki banyak jenis dalam hal ukuran, desain dan *layout*, tetapi mereka memiliki karakteristik yang sangat berbeda dari sistem *desktop*.

Berikut karakteristik perangkat *mobile*, diantaranya :

a. Ukuran yang kecil

Perangkat *mobile* memiliki ukuran yang kecil. Konsumen menginginkan perangkat yang terkecil untuk kenyamanan dan mobilitas mereka.

b. *Memory* yang terbatas

Perangkat *mobile* juga memiliki *memory* yang kecil, yaitu *primary* (RAM) dan *secondary* (*disk*). Pembatasan ini adalah salah satu faktor yang mempengaruhi penulisan program untuk berbagai jenis dari perangkat ini. Dengan pembatasan jumlah dari *memory*, pertimbangan-pertimbangan khusus harus diambil untuk memelihara pemakaian dari sumber daya yang mahal ini.

c. Daya proses yang terbatas

Sistem *mobile* tidaklah setangguh rekan mereka yaitu *desktop*. Ukuran, teknologi dan biaya adalah beberapa faktor yang mempengaruhi status dari sumber daya ini. Seperti *harddisk* dan RAM, Anda dapat menemukan mereka dalam ukuran yang pas dengan sebuah kemasan kecil.

d. Mengkonsumsi daya yang rendah

Perangkat mobile menghabiskan sedikit daya dibandingkan dengan mesin desktop. Perangkat ini harus menghemat daya karena mereka berjalan pada keadaan dimana daya yang disediakan dibatasi oleh baterai-baterai.

e. Kuat dan dapat diandalkan

Karena perangkat *mobile* selalu dibawa kemana saja, mereka harus cukup kuat untuk menghadapi benturan-benturan, gerakan, dan sesekali tetesan-tetesan air.

f. Konektivitas yang terbatas

Perangkat *mobile* memiliki bandwith rendah, beberapa dari mereka bahkan tidak tersambung. Kebanyakan dari mereka menggunakan koneksi *wireless*.

g. Masa hidup yang pendek

Perangkat-perangkat konsumen ini menyala dalam hitungan detik kebanyakan dari mereka selalu menyala. Coba ambil kasus sebuah *handphone*, mereka *booting* dalam hitungan detik dan kebanyakan orang tidak mematikan *handphone* mereka bahkan ketika malam hari. PDA akan menyala jika anda menekan tombol power mereka.

Aplikasi *mobile* terdapat beberapa tipe, berikut beberapa tipe aplikasi *mobile* :

A. Berdasarkan Teknologi yang digunakan :

1. Server-Request Based

Mobile Equipment (ME) merequest data dari *server*, WAP. menggunakan WML, sebuah *markup language* khusus berbasis XML.

Keuntungan :

a. Platform Independent

b. Mudah diterapkan

c. Jenis aplikasi Beragam

Kerugian :

a. Masih banyak ME yang tidak support WAP.

b. Pembayaran untuk data.

c. *Interface* terbatas

Implementasi :

a. Menggunakan *Web Server* dengan *setting* tambahan di konfigurasi

b. *File* WML dibuat selayaknya membuat *file* HTML

2. *SMS based*

   SMS : Short Message Service. Pengiriman pesan singkat antar ME / dengan sebuah *server* Terbatas ± 160 *characters.*

   Keuntungan :

   a. *Platform Independent*
   b. ME sebagian besar sudah dapat menerapkan *SMS*

   Kerugian :

   a. Pembayaran untuk setiap aksi.
   b. *Interface* hanya *text*, *user* harus ingat perintah
   c. Panjang *text* sangat terbatas

   Implementasi :

   a. Menggunakan *interface* antara ME dengan *server*
   b. Aplikasi menunggu data dari *interface* dan mengirim data melalui *interface* tersebut

3. *SIM-card based*

   Aplikasi diletakkan di atas *SIM-card*, aplikasi dapat diakses melalui menu baru yang akan muncul di ponsel. *Javacard*, khusus untuk *javacard* aplikasi bersifat *downloadable*, pengiriman data lewat *SMS*

   Keuntungan :

   a. *Platform Independent*
   b. Data disimpan secara lokal dan dapat digunakan bila berganti ME

Kerugian :

a. *Size* terbatas.

b. Aplikasi harus sangat kecil

c. Sulit diterapkan

Implementasi :

a. *Download Javacard*.

b. Gunakan *Simulator*

4. Mobile Equipment Based

Aplikasi diletakkan di *Mobile Equipment*, contoh Pemrograman di atas *Symbian OS*, pengiriman data *SMS*.

Keuntungan :

a. *Interface* sangat mudah

b. Jenis aplikasi sangat beragam, bahkan dapat digunakan untuk aplikasi berat

Kerugian :

a. Platform Dependent

b. Bila mengubah ME kemungkinan aplikasi dan data tidak dapat digunakan

B. Berdasarkan fungsi :

1. Information

Pengiriman informasi ke ME user

2. *Communications*

   Komunikasi antara *user* dengan satu atau lebih *user* lain.

3. *Entertainment*

   Aplikasi menghibur seperti *Games*.

4. *Productivity Tools*

   Aplikasi yang dapat meningkatkan produktivitas walaupun sedang berpindah

5. *Commerce and Banking*

   Jual beli melalui ME.

6. *Location based*

   Aplikasi yang mengubah data sesuai dengan lokasi *user*. Lokasi dapat ditentukan oleh GPS dan sel yang ditempati oleh user.

2.3. Transportasi

Transportasi menurut Stopher dan Meyburg (1978) dalam Firgani Arif (2009) mendefinisikan sebagai pergerakan barang atau manusia dalam dimensi ruang, waktu, dan nilai (dalam bukunya, nilai disebut State, yang maksudnya adalah nilai pasar, sebagai contah: nilai suatu barang dalam waktu yang berlainan atau dalam tempat yang berbeda akan mempunyai nilai yang berbeda pula). Pergerakan barang atau manusia tersebut belum bisa berlangsung tanpa adanya sarana dan prasarana yang mendukungnya, maka pendekatan sistem lebih tepat digunakan dalam memahami transportasi.

Transportasi digunakan untuk memudahkan manusia dalam melakukan aktivitas sehari-hari. Di negara maju, mereka biasanya menggunakan kereta bawah

tanah (*subway*) dan taksi. Penduduk disana jarang yang mempunyai kendaraan pribadi karena mereka sebagian besar menggunakan angkutan umum sebagai transportasi mereka. Transportasi sendiri dibagi 3 yaitu, transportasi darat, laut, dan udara. Transportasi udara merupakan transportasi yang membutuhkan banyak uang untuk memakainya. Selain karena memiliki teknologi yang lebih canggih, transportasi udara merupakan alat transportasi tercepat dibandingkan dengan alat transportasi lainnya, dan Transportasi darat adalah segala bentuk transportasi menggunakan jalan untuk mengangkut penumpang atau barang.

Bentuk awal dari transportasi darat adalah menggunakan kuda, keledai atau bahkan manusia untuk membawa barang melewati jalan setapak. Seiring dengan berkembangnya perdagangan, jalan diratakan atau dilebarkan untuk mengakomodir aktivitas. Roda kemudian ditemukan.

2.4. Angkutan Kota (Angkot)

Untuk berpindah dari satu tempat ke tempat lain penduduk di suatu kota membutuhkan sarana untuk mempercepat dan mempermudah proses perpindahan tersebut. Setiap kota di Indonesia pasti memiliki Angkutan Kota (Angkot) sebagai sarana untuk melakukan proses perpindahan tersebut.

2.4.1. Definisi Angkutan Kota

Angkutan kota, menurut Setijowarno dan Frazila (2001) dalam Firgani Arif(2009), adalah angkutan dari suatu tempat ke tempat lain dalam wilayah suatu kota dengan menggunakan mobil bus umum dan/atau mobil penumpang umum yang terikat pada trayek tetap dan teratur. Dapat juga angkutan kota berupa angkutan

missal atau *mass rapid transit* yang dapat mengangkut penumpang dalam jumlah banyak dalam satu kali perjalanan.

2.4.2. Tujuan dan Peran Angkutan Kota

Menurut Warpani (1990) dalam Firgani Arif (2009) anggota masyarakat pemakai jasa angkutan dikelompokkan dalam dua golongan besar yaitu *paksawan* yaitu mereka yang tidak mampu memiliki kendaraan atau menyewa sendiri, dan *pilihwan* yaitu mereka yang mampu.

Tujuan utama keberadaan angkutan kota menurut Warpani (1990) dalam Firgani Arif (2009) adalah menyelenggarakan pelayanan angkutan yang aman, cepat, murah, dan nyaman bagi masyarakat. Karena sifatnya yang massal, maka diperlukan adanya kesamaan diantara para penumpang berkenaan dengan asal dan tujuan.

2.5. Java

Java menurut Shalahuddin dan Rosa (2010) adalah nama untuk sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada komputer *standalone* ataupun pada lingkungan jaringan. Bahasa ini awalnya dibuat oleh James Gosling saat masih bergabung di Sun Microsystems saat ini merupakan bagian dari Oracle dan dirilis tahun 1995. Bahasa ini banyak mengadopsi sintaksis yang terdapat pada C dan C++ namun dengan sintaksis model objek yang lebih sederhana serta dukungan rutin-rutin aras bawah yang minimal. 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/non-spesifik (*general purpose*), dan secara khusus didisain untuk

memanfaatkan dependensi implementasi seminimal mungkin. Karena fungsionalitasnya yang memungkinkan aplikasi java mampu berjalan di beberapa platform sistem operasi yang berbeda, java dikenal pula dengan slogannya, "*Tulis sekali, jalankan di mana pun*". 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.

2.5.1. Sejarah Perkembangan Java

Bahasa pemrograman Java menurut Benny Hemawan (2004) terlahir dari *The Green Project*, yang berjalan selama 18 bulan, dari awal tahun 1991 hingga musim panas 1992. Proyek tersebut belum menggunakan versi yang dinamakan Oak. Proyek ini dimotori oleh Patrick Naughton, Mike Sheridan, James Gosling dan Bill Joy, beserta sembilan pemrogram lainnya dari Sun Microsystems. Salah satu hasil proyek ini adalah maskot *Duke* yang dibuat oleh Joe Palrang.

Pertemuan proyek berlangsung di sebuah gedung perkantoran *Sand Hill Road* di Menlo Park. Sekitar musim panas 1992 proyek ini ditutup dengan menghasilkan sebuah program *Java Oak* pertama, yang ditujukan sebagai pengendali sebuah peralatan dengan teknologi layar sentuh (*touch screen*), seperti pada PDA sekarang ini. Teknologi baru ini dinamai "*7" (*Star Seven*).

Setelah era *Star Seven* selesai, sebuah anak perusahaan Tv kabel tertarik ditambah beberapa orang dari proyek The Green Project. Mereka memusatkan kegiatannya pada sebuah ruangan kantor di 100 Hamilton Avenue, Palo Alto.

Perusahaan baru ini bertambah maju: jumlah karyawan meningkat dalam waktu singkat dari 13 menjadi 70 orang. Pada rentang waktu ini juga ditetapkan pemakaian Internet sebagai medium yang menjembatani kerja dan ide di antara mereka. Pada awal tahun 1990-an, Internet masih merupakan rintisan, yang dipakai hanya di kalangan akademisi dan militer.

Mereka menjadikan perambah (*browser*) Mosaic sebagai landasan awal untuk membuat perambah Java pertama yang dinamai Web Runner, terinsipirasi dari film 1980-an, Blade Runner. Pada perkembangan rilis pertama, Web Runner berganti nama menjadi Hot Java.

Pada sekitar bulan Maret 1995, untuk pertama kali kode sumber Java versi 1.0a2 dibuka. Kesuksesan mereka diikuti dengan untuk pemberitaan pertama kali pada surat kabar *San Jose Mercury News* pada tanggal 23 Mei 1995.

Sayang terjadi perpecahan di antara mereka suatu hari pada pukul 04.00 di sebuah ruangan hotel Sheraton Palace. Tiga dari pimpinan utama proyek, Eric Schmidt dan George Paolini dari Sun Microsystems bersama Marc Andreessen, membentuk Netscape.

Nama Oak, diambil dari pohon oak yang tumbuh di depan jendela ruangan kerja "Bapak Java", James Gosling. Nama Oak ini tidak dipakai untuk versi release Java karena sebuah perangkat lunak lain sudah terdaftar dengan merek dagang tersebut, sehingga diambil nama penggantinya menjadi "Java". Nama ini diambil dari kopi murni yang digiling langsung dari biji (kopi tubruk) kesukaan Gosling. Konon

kopi ini berasal dari Pulau Jawa. Jadi nama bahasa pemrograman Java tidak lain berasal dari kata Jawa (bahasa Inggris untuk Jawa adalah Java).

2.5.2. Kelebihan Java

1. *Multiplatform*. Kelebihan utama dari Java ialah dapat dijalankan di beberapa *platform* / sistem operasi komputer, sesuai dengan prinsip *tulis sekali, jalankan di mana saja*. Dengan kelebihan ini pemrogram cukup menulis sebuah program Java dan dikompilasi (diubah, dari bahasa yang dimengerti manusia menjadi bahasa mesin / *bytecode*) sekali lalu hasilnya dapat dijalankan di atas beberapa platform tanpa perubahan. Kelebihan ini memungkinkan sebuah program berbasis java dikerjakan diatas operating system Linux tetapi dijalankan dengan baik di atas Microsoft Windows. Platform yang didukung sampai saat ini adalah Microsoft Windows, Linux, Mac OS dan Sun Solaris. Penyebanya adalah setiap sistem operasi menggunakan programnya sendiri-sendiri (yang dapat diunduh dari situs Java) untuk meninterpretasikan *bytecode* tersebut.
2. OOP (*Object Oriented Programming* - Pemrogram Berorientasi Objek) Perpustakaan Kelas Yang Lengkap, Java terkenal dengan kelengkapan *library*/perpustakaan (kumpulan program program yang disertakan dalam pemrograman java) yang sangat memudahkan dalam penggunaan oleh para pemrogram untuk membangun aplikasinya. Kelengkapan perpustakaan ini ditambah dengan keberadaan komunitas Java yang besar yang terus menerus

membuat perpustakaan-perpustakaan baru untuk melingkupi seluruh kebutuhan pembangunan aplikasi.

3. Bergaya C++, memiliki sintaks seperti bahasa pemrograman C++ sehingga menarik banyak pemrogram C++ untuk pindah ke Java. Saat ini pengguna Java sangat banyak, sebagian besar adalah pemrogram C++ yang pindah ke Java. Universitas-universitas di Amerika Serikat juga mulai berpindah dengan mengajarkan Java kepada murid-murid yang baru karena lebih mudah dipahami oleh murid dan dapat berguna juga bagi mereka yang bukan mengambil jurusan komputer.
4. Pengumpulan sampah otomatis, memiliki fasilitas pengaturan penggunaan memori sehingga para pemrogram tidak perlu melakukan pengaturan memori secara langsung (seperti halnya dalam bahasa C++ yang dipakai secara luas).

2.5.3. Kekurangan Java

1. Tulis sekali, jalankan di mana saja - Masih ada beberapa hal yang tidak kompatibel antara *platform* satu dengan *platform* lain. Untuk J2SE, misalnya *SWT-AWT bridge* yang sampai sekarang tidak berfungsi pada Mac OS X.
2. Mudah didekompilasi. Dekompilasi adalah proses membalikkan dari kode jadi menjadi kode sumber. Ini dimungkinkan karena kode jadi Java merupakan *bytecode* yang menyimpan banyak atribut bahasa tingkat tinggi, seperti nama-nama kelas, metode, dan tipe data. Hal yang sama juga terjadi pada Microsoft .NET Platform. Dengan demikian, algoritma yang digunakan

program akan lebih sulit disembunyikan dan mudah dibajak/di*reverse-engineer*.

3. Penggunaan memori yang banyak. Penggunaan memori untuk program berbasis Java jauh lebih besar daripada bahasa tingkat tinggi generasi sebelumnya seperti C/C++ dan Pascal (lebih spesifik lagi, Delphi dan Object Pascal). Biasanya ini bukan merupakan masalah bagi pihak yang menggunakan teknologi terbaru (karena trend memori terpasang makin murah), tetapi menjadi masalah bagi mereka yang masih harus berkutat dengan mesin komputer berumur lebih dari 4 tahun.

2.5.4. Contoh Program Java

Contoh program Hello World yang ditulis menggunakan bahasa pemrograman Java adalah sebagai berikut:

```
// Outputs "Hello, world!" and then exits
    public class HelloWorld {
      public static void main(String args[]) {
        System.out.println("Hello, world!");
      }
    }
```

2.5.5. Tahap Kompilasi Java

1. Tulis / Ubah. Pemrogram menulis program dan menyimpannya di media dalam bentuk berkas '.java'.
2. Kompilasi. Pengkompilasi membentuk *bytecodes* dari program menjadi bentuk berkas '.class'.
3. Muat. Pemuat kelas memuat *bytecodes* ke memori.
4. Verifikasi. Peng-verifikasi memastikan *bytecodes* tidak mengganggu sistem keamanan Java.
5. Jalankan. Penerjemah menerjemahkan *bytecodes* ke bahasa mesin.tidak bisa di pakai‘’

2.5.6. Integrated Development Environtment

Banyak pihak telah membuat IDE (*Integrated Development Environment* - Lingkungan Pengembangan Terintegrasi) untuk Java. Yang populer saat ini (Juli 2006) antara lain:

1. Dr. Java, program gratis yang dikembangkan oleh Universitas Rice, Amerika Serikat
2. BlueJ, program gratis yang dikembangkan oleh Universitas Monash, Australia
3. NetBeans (open *source- Common Development and Distribution License* (CDDL)). NetBeans disponsori Sun Microsystems, dan versi terkininya memilki Matisse, sebuah GUI Editor yang menurut pendapat umum merupakan yang terbaik.

4. Eclipse JDT *(open source- Eclipse Public License*). Eclipse dibuat dari kerja sama antara perusahaan-perusahaan anggota 'Eclipse Foundation' (beserta individu-individu lain). Banyak nama besar yang ikut dalam 'Eclipse Foundation', termasuk IBM, BEA, Intel, Nokia, Borland. Eclipse bersaing langsung dengan Netbeans IDE. Plugin tambahan pada Eclipse jauh lebih banyak dan bervariasi dibandingkan IDE lainnya.
5. IntelliJ IDEA (*commercial, free 30-day trial*)
6. Oracle JDeveloper (free)
7. Xinox JCreator (ada versi berbayar maupun *free*). JCreator ditulis dalam C/C++ sehingga lebih cepat (dan menggunakan memori lebih sedikit) dari kebanyakan IDE.

2.5.7. Edisi Java

Menurut M. Shalahuddin (2008) Java adalah bahasa yang dapat di jalankan dimanapun dan disembarang platform apapun, diberagam lingkungan : internet, intranet, consumer Electronic products dan computer Applications. The Java 2 platform tersedia dalam 3 edisi untuk keperluan berbeda. Untuk beragam aplikasi yang dibuat dengan bahasa java, java dipaketkan dalam edisi – edisi berikut :

1. Java 2 Standard Edition (J2SE)

Menurut M. Shalahuddin (2008) The Java 2 Standard Edition menyediakan lingkungan pengembangan yang kaya akan fitur, stabil, aman dan cross – platform. Edisi ini mendukung konektivitas basis data, rancangan antar muka pemakai,

masukan / keluaran dan pemograman jaringan dan termasuk sebagai paket – paket dasar bahasa Java.

2. Java 2 Enterprise Edition (J2EE)

Menurut M. Shalahuddin (2008)The Java Enterprise Edition menyediakan *tool* untuk membangun dan menjalankan *multitier enterprise applications*. J2EE bersisi paket – paket di J2SE ditambah paket – paket untuk mendukung pengembangan Enterprise JavaBeeans, Java Servlets, Java Server Pages, XML dan kendali transaksi yang fleksibel

3. Java 2 Micro Edition (J2ME)

Menurut M. Shalahuddin (2008) The Java 2 Micro Edition selain menyedikan bahasa Java yang sama, unggul dalam portabilitas (kemampuan dapat dijalankan dimanapun), *safe network delivery*, seperti J2SE dan J2EE. Aplikasi-aplikasi dapat diskalakan (dimampukan) agar dapat bekerja dengan J2SE dan J2EE.

J2ME untuk beragam consumer electronic product seperti pager, smart card, cell phone, handled PDA, dan set top box.

2.5.8. Gambaran J2ME

Menurut M. Shalahuddin (2008) J2ME adalah satu set spesifikasi dan teknologi yang fokus kepada perangkat konsumen. Perangkat ini memiliki jumlah memori yang terbatas, menghabiskan sedikit daya dari baterai, layar yang kecil dan *bandwith* jaringan yang rendah.

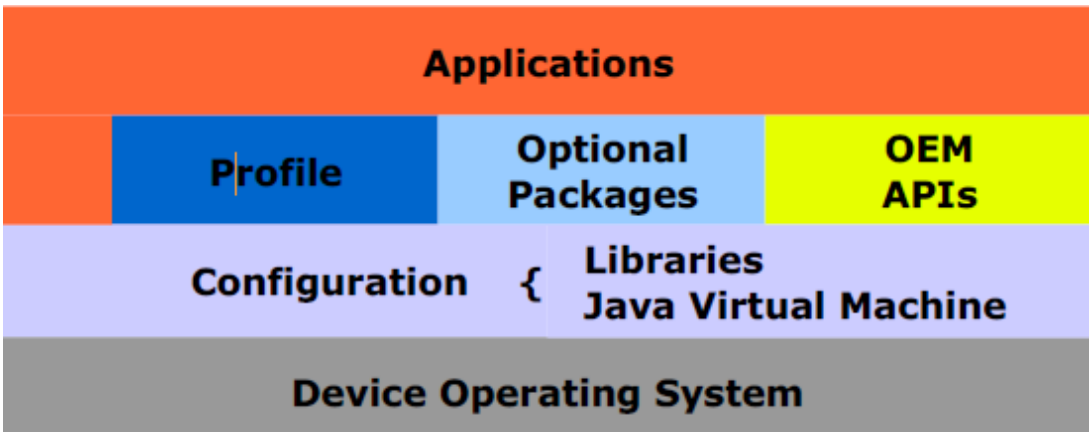
Dengan perkembangbiakan perangkat *mobile* konsumer dari telepon, PDA, kotak permainan ke peralatan-peralatan rumah, Java menyediakan suatu lingkungan

yang portable untuk mengembangkan dan menjalankan aplikasi pada perangkat ini. Program J2ME, seperti semua program JAVA adalah diterjemahkan oleh VM. Program-program tersebut dikompile ke dalam *bytecode* dan diterjemahkan dengan Java Virtual Machine(JVM).Ini berarti bahwa program-program tersebut tidak berhubungan langsung dengan perangkat.

J2ME menyediakan suatu *interface* yang sesuai dengan perangkat. Aplikasi-aplikasi tersebut tidak harus dikompile ulang supaya mampu dijalankan pada mesin yang berbeda.

Inti dari J2ME terletak pada *configuration* dan *profile-profile*. Suatu *configuration* menggambarkan lingkungan *runtime* dasar dari suatu sistem J2ME. Ia menggambarkan *core library*, *virtual machine*, fitur keamanan dan jaringan.

Gambar 2.1 Arsitektur J2ME



Diadopsi dari : Shalahuddin, M dan A.S, Rosa. (2010:)

Sebuah *profile* memberikan *library* tambahan untuk suatu kelas tertentu pada sebuah perangkat. *profile-profile* menyediakan user *interface*(UI) API *persistence*, *messaging library*, dan sebagainya.

Satu set library tambahan atau package tambahan menyediakan kemampuan program tambahan. Pemasukan package ini ke dalam perangkat J2ME dapat berubah-ubah karena tergantung pada kemampuan sebuah perangkat. Sebagai contoh, beberapa perangkat MIDP tidak memiliki *Bluetooth built-in*, sehingga *Bluetooth* API tidak disediakan dalam perangkat ini.

2.6. UML (*Unified Modeling Language*)

UML (*Unified Modeling Language*) merupakan pengganti dari metode analisis berorientasi *object* dan *design* berorientasi *object* (OOA & OOD) yang dimunculkan sekitar akhir tahun 80-an dan awal tahun 90-an.

UML merupakan gabungan dari metode Grady Booch (Booch *Method*), James Rumbaugh (OMT) dan Ivar Jacobson (OOSE). Tetapi UML ini akan mencakup lebih luas daripada OOA&D. Pada pertengahan pengembangan UML dilakukan standarisasi proses dengan OMG (*Object Management Group*) dengan harapan UML akan menjadi bahasa standar pemodelan pada masa yang akan datang.

UML disebut sebagai bahasa pemodelan bukan metode. Kebanyakan metode terdiri paling sedikit prinsip, bahasa pemodelan dan proses. Bahasa pemodelan (sebagian besar grafik) merupakan notasi dari metode yang digunakan untuk mendesain secara cepat.

Bahasa pemodelan merupakan bagian terpenting dari metode. Ini merupakan bagian kunci tertentu untuk komunikasi. Jika anda ingin berdiskusi tentang desain dengan seseorang, maka Anda hanya membutuhkan bahasa pemodelan bukan proses yang digunakan untuk mendapatkan desain.

UML merupakan bahasa standar untuk penulisan Blueprint Software yang digunakan untuk Visualisasi (*Visualize*), Spesifikasi (*Specify*), Pembentukan (*Construct*) dan Pendokumentasian (*Documentation*) alat-alat dari sistem perangkat lunak.

2.6.1. Pengertian UML

UML adalah bahasa untuk menspesifikasi, memvisualisasi, membangun dan mendokumentasikan artifacts (bagian dari informasi yang digunakan atau dihasilkan oleh proses pembuatan perangkat lunak, artifact tersebut dapat berupa model, deskripsi atau perangkat lunak) dari sistem perangkat lunak, seperti pada pemodelan bisnis dan sistem non perangkat lunak lainnya [HAN98]. Selain itu UML adalah bahasa pemodelan yang menggunakan konsep orientasi object. UML dibuat oleh Grady Booch, James Rumbaugh, dan Ivar Jacobson di bawah bendera Rational Software Corp [HAN98]. UML menyediakan notasi-notasi yang membantu memodelkan sistem dari berbagai perspektif. UML tidak hanya digunakan dalam pemodelan perangkat lunak, namun hampir dalam semua bidang yang membutuhkan pemodelan.

2.6.2. Sejarah UML

UML dimulai secara resmi pada oktober 1994, ketika Rumbaugh bergabung dengan Booch pada Relational Software Corporation. Proyek ini memfokuskan pada penyatuan metode Booch dan OMT. UML versi 0.8 merupakan metode penyatuan yang dirilis pada bulan Oktober 1995. Dalam waktu yang sama, Jacobson bergabung dengan Relational dan cakupan dari UML semakin luas sampai diluar perusahaan OOSE. Dokumentasi UML versi 0.9 akhirnya dirilis pada bulan Juni 1996. Meskipun pada tahun 1996 ini melihat dan menerima feedback dari komunitas Software Engineering . Dalam waktu tersebut, menjadi lebih jelas bahwa beberapa organisasi perangkat lunak melihat UML sebagai strategi dari bisnisnya. Kemudian dibangunlah UML Consortium dengan beberapa organisasi yang akan menyumbangkan sumber dayanya untuk bekerja, mengembangkan, dan melengkapi UML.

Di sini beberapa partner yang berkontribusi pada UML 1.0, diantaranya Digital Equipment Corporation, Hewlett-Packard, I-Logix, Intellicorp, IBM, ICON Computing, MCI Systemhouse, Microsoft, Oracle, Relational, Texas Instruments dan Unisys. Dari kolaborasi ini dihasilkan UML 1.0 yang merupakan bahasa pemodelan yang ditetapkan secara baik, expressive, kuat, dan cocok untuk lingkungan masalah yang luas. UML 1.0 ditawarkan menjadi standarisasi dari Object Management Group (OMG). Dan pada Januari 1997 dijadikan sebagai standar bahasa pemodelan.

Antara Januari–Juli 1997 gabungan group tersebut memperluas kontribusinya sebagai hasil respon dari OMG dengan memasukkan Adersen Consulting, Ericsson, ObjectTimeLimeted, Platinum Technology, Ptech, Reich Technologies, Softeam,

Sterling Software dan Taskon. Revisi dari versi UML (versi 1.1) ditawarkan kepada OMG sebagai standarisasi pada bulan Juli 1997. Dan pada bulan September 1997, versi ini dierima oleh OMG Analysis dan Design Task Force (ADTF) dan OMG ArchitectureBoard. Dan Akhirnya pada Juli 1997 UML versi 1.1 menjadi standarisasi.

Pemeliharaan UML terus dipegang oleh OMG Revision Task Force (RTF) yang dipimpin oleh Cris Kobryn. RTP merilis editorial dari UML 1.2 pada Juni 1998. Dan pada tahun 1998 RTF juga merilis UML 1.3 disertai dengan *user guide* dan memberikan *technical cleanup*.

2.6.3. Bangunan Dasar UML

Untuk memahami UML diperlukan pemahaman konseptual. Metodologi UML menggunakan 3 bangunan dasar untuk mendeskripsikan sistem.perangkat lunak yang akan dikembangkan, yaitu:

1. *Things* (sesuatu)
2. *Relationship* (hubungan)
3. *Diagrams* (diagram)

Setiap bangunan dasar tersebut dapat diterapkan sepanjang tahap pengembangan sistem, dan masing-masingnya dapat digunakan saling melengkapi satu sama lainnya. Penjelasan dari ketiga bangunan dasar UML yang disebutkan diatas adalah sebagai berikut:

1. Things (sesuatu)

Ada 4 macam *Things* dalam UML yaitu:

a. Structural Things

Merupakan bagian yang relatif statis dalam model UML. Bagian yang relatf statis dapat berupa elemen-elemen yang besifat fisik maupun konseptual. Secara keseluruhan ada 7 macam Structural Things:

1) *Classes* / Kelas adalah himpunan dari objek-objek yang berbagi atribut serta operasi yang sama. Kelas mengimplementasikan satu atau lebih antarmuka. Secara grafis, kelas digambarkan dengan empat persegi panjang yang memuat nama, atribut serta operasi yang dimilikinya.
2) *Interfaces* / Antarmuka adalah kumpulan dari operasi-operasi yang menspesifikasikan layanan/servis suatu kelas atau komponen/objek. Antarmuka ini mendeskripsikan perilaku yang tampak dari luar suatu elemen. Secara grafis, antarmuka digambarkan dengan lingkaran kecil dan nama yang didahului dengan garis tegak.
3) *Collaborations* / Kolaborasi adalah sesuatu yang mendefinisikan interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dan sinergis. Secara grafis, kolaborasi digambarkan elips bergaris putus-putus yang memuat nama kolaborasi itu.
4) *Use-Cases* adalah deskripsi dari urutan aksi-aksi yang ditampilkan sistem dengan hasil yang terukur bagi suatu *Actor*. *Use-Case*

digunakan untuk menstrukturkan perilaku pada suatu model. Secara grafis *use case* digambarkan dengan elips tegas yang berisi namanya.

5) *Active Class* / Kelas Aktif adalah kelas (biasa) dimana objek-objek yang dimilikinya memiliki 1 atau lebih proses dan lebih jauh menginisialisasi suatuu aktifitas kendali. Secara grafis, kelas aktif digambarkan sebagai kelas biasa tapi memiliki batas yang lebih tebal, yang memuat nama, atribut serta operasi yang dimilikinya.
6) *Components* / Komponen adalah bagian fisik dan bagian yang dapat digantikan dalam suatu sistem, misalnya berkas ActiveX, COM, DLL yang keberadaanya dibutuhkan oleh sistem yang akan dikembangkan. Komponen ini merepresentasikan konsep-konsep *reusable component*. Secara grafis, komponen digambarkan dengan empat persegi panjang seperti kelas tetapi ditambahkan *Tab*.
7) *Nodes* / Simpul adalah elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi; secara umum menggunakan kapasitas memori dan kemampuan pemrosesan. Secara grafis, simpul digambarkan sebagai kubus yang berisi namanya.

b. Behavioral_Things

Merupakan bagian yang dinamis pada model UML, biasanya merupakan kata kerja dari model UML yang mencerminkan perilaku sepanjang ruang dan waktu. Ada 2 macam *Behavior Things*, yaitu:

1) *Interactions* / interaksi adalah suatu perilaku yang mencakup himpunan pesan-pesan (*message*) atau kumpulan objek & operasinya yang diperlukan untuk menyelesaikan suatu fungsi tertentu. Sebuah interaksi terdiri dari beberapa unsur, yaitu: Pesan, Perilaku dan Link, Secara grafis, pesan digambarkan dengan tanda panah tegas dan memuat nama operasinya.
2) *State Machine*, adalah perilaku yang menspesifikasikan urutan kedudukan suatu objek atau interaksi-interaksi sepanjang waktu untuk menanggapi *event-event* yang terjadi. Penggambaran state memiliki beberapa unsur, yaitu: State itu sendiri, Transisi (perubahan dari suatu *state* ke *state* lain), *Event* (suatu keadaan yang memicu sebuah transisi), Aktifitas (tanggapan terhadap transisi). Secara grafis *State* digambarkan sebagai empat persegi panjang dengan sudut tumpul, yang memuat namanya serta subsistem didalamnya jika ada.

c. Grouping Things

Merupakan bagian pengorganisasian (*Packages*) dalam UML. Digunakan untuk menggambarkan paket-paket untuk menyederhanakan model-model UML yang rumit. Paket-paket ini kemudian dapat di dekomposisi. Paket berguna untuk pengelompokkan sesuatu misalnya model-model serta subsistem-subsistem.

d. Annotational Things

Merupakan bagian yang memperjelas model UML (*Notes*), seperti komentar-komentar yang menjelaskan fungsi secara rinci serta ciri-ciri tiap elemen dalam model UML.

2. *Relationship* (hubungan)

Relationship adalah hubungan-hubungan yang terjadi antara elemen dalam UML. Hubungan-hubungan ini Penting sekali dalam UML. Model-model UML harus dibuat menggunakan *relationship*. Ada 4 macam relationship dalam UML yang dapat digunakan untuk menggambarkan model-model UML yang *representative*:

a. *Dependency* (kebergantungan)

Dependency adalah hubungan dimana perubahan yang terjadi dalam suatu elemen *independent* (mandiri) akan mempengaruhi elemen yang bergantung padanya. Secara grafis, *dependency* digambarkan dengan tanda panah terputus-putus.

b. *Association* (asosiasi)

Asosiasi adalah sesuatu yang menghubungkan antara suatu objek dengan objek lainnya yang menggambarkan bagaimana hubungan antar objek tersebut dalam bentuk agregasi. Secara grafis, asosiasi digambarkan dengan garis tegas tanpa tanda panah.

c. *Generalizations* (generalisasi)

Generalisasi adalah hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari obyek yang ada diatasnya yaitu objek induk (*ancestor*). Arah dari atas ke bawah atau dari *ancestor* ke *descendent* dinamakan *Spesialisasi*. Arah sebaliknya yaitu bawah ke atas dinamakan *Generalisasi*. Secara grafis Generalisasi digambarkan dengan garis dan panah yang berkepala sigitiga kosong dan mengarah ke objek induk.

d. *Realizations* (realisasi)

Realisasi adalah operasi yang benar-benar dilakukan oleh suatu objek. Secara grafis, realisasi digambarkan dengan tanda panah bergaris putus-putus dengan kepala panah kosong.

3. *Diagrams* (diagram)

Suatu sistem yang kompleks harus dapat dipandang dari sudut yang berbeda-beda sehingga didapat pemahaman secara menyeluruh. Untuk keperluan tersebut UML menyediakan 9 jenis diagrams yang dapat dikelompokkan berdasarkan sifatnya; Statis atau Dinamis. Kesembilan diagram ini tidak mutlak digunakan atau dibuat sesuai kebutuhan. Pada pemodelan UML dimungkinkan menggunakan diagram lain sejauh diagram tersebut dapat membantu pemahaman mendalam tentang suatu sistem atau perangkat lunak. Ke-9 diagram tersebut adalah sebagai berikut:

a. *Class Diagrams* (Statis)

Diagram ini memperlihatkan himpunan kelas-kelas, antarmuka-antarmuka, kolaborasi-kolaborasi, dan relasi-relasi. Diagram ini umum ditemui pada pemodelan sistem berorientasi objek. Meski sifatnya statis, sering pula memuat kelas-kelas aktif.

1. *Class* Diagram menggambarkan interaksi antar kelas dalam sistem tersebut.
2. Pembuatan *Class* sama dengan pembuatan Objek-objek

b. *Object Diagrams* (Statis)

Diagram ini memperlihatkan objek-objek serta relasi-relasi antar objek. Diagram objek memperlihatkan instantiasi statis dari segala sesuatu yang dijumpai dari pada diagram kelas.

c. *Use-Case Diagrams* (Statis)

Dagram ini memperlihatkan himpunan *Use-Case* dan *actor-actor* (jenis khusus dari kelas). Diagram ini penting untuk mengorganisasi dan memodelkan perilaku dari suatu sistem yang dibutuhkan serta diharapkan pengguna.

1. *Use-Case* diagram menggambarkan hubungan *use-case* dengan *actor*.
2. *Use-case* merepresentasikan fungsi, kebutuhan dari perpektif *user*
3. *Actor* adalah orang atau sistem yang menerima atau memberikan informasi dari sistem ini

d. *Sequence Diagrams* (Dinamis)

Diagram *sequence* (urutan) adalah diagram interaksi yang menekankan pada pengiriman pesan (message) dalam suatu waktu tertentu.

1. *Sequence* Diagram mengambarkan alur kerja dari fungsi-fungsi dalam sistem dengan use-case dimana didalamnya terdapat actor
2. *Actor* adalah orang atau sistem yang menerima atau memberikan informasi dari sistem ini.
3. Diagram ini sangat memperhatikan waktu/ terurut berdasarkan kejadian (*sequence*)

e. *Collaboration Diagrams* (Dinamis)

Diagram kolaborasi adalah diagram interaksi yang menekankan organisasi *structural* dari objek-objek yang menerima serta mengirim pesan (*message*).

1. Informasi yang disampaikan sama dengan sequencial diagram namun beda dalam pengambaran dan kegunaan saja.
2. Dalam diagram ini digambarkan hubungan antar objek dan *actor* dengan tidak memperhatikan waktu/urutan

f. *State Chart Diagrams* (Dinamis)

Diagram ini memperlihatkan *state-state* pada sistem; memuat *state*, transisi, *event*, serta aktifitas. Diagram ini terutama penting untuk memperlihatkan sifat dinamis dari antarmuka, kelas, kolaborasi, dan terutama penting pada pemodelan sistem-sistem reaktif.

1. *State Chart* Diagram memberikan berbagai cara/jalan kepada model untuk berbagai kejadian yang mungkin terjadi dalam sebuah objek
2. Diagram ini digunakan untuk menggambarkan berbagai prilaku objek yang sifatnya dinamis dalam sebuah sistem

g. *Activity Diagrams* (Dinamis)

Diagram ini adalah tipe khusus dari diagram state yang memperlihatkan aliran dari suatu aktifitas ke aktifitas lainnya dalam suatu sistem. Diagram ini terutama penting dalam pemodelan fungsi-fungsi dalam suatu sistem dan memberi tekanan pada aliran kendali antar objek.

1. Memberikan gambaran ilustrasi alur dari setiap fungsi yang ada dalam sistem
2. Aliran dimulai dari suatu titik hingga ke titik akhir yang disepakati.

h. *Component Diagrams* (Statis)

Diagram ini memperlihatkan organisasi serta kebergantunngan pada komponen-komponen yang telah ada sebelumnya. Diagram ini berhubungan dengan diagram kelas dimana komponen secara tipikal dipetakan kedalam satu atau lebih kelas-kelas, antarmuka-antarmuka, serta kolaborasi-kolaborasi.

1. Menggambarkan model secara fisik sebagai sebuah software komponen yang ada dalam sebuah system
2. Komponen-komponen tersebut nantinya diarahkan pada suatu bahasa pemrograman tertentu

i. *Deployment Diagrams* (Statis)

Diagram ini memperlihatkan konfigurasi saat aplikasi dijalankan (*run-time*), memuat simpul-simpul atau node beserta komponen-komponen yang ada didalamnya. *Deployment Diagrams* berhubungan erat dengan diagram komponen dimana *Deployment Diagrams* memuat satu atau lebih komponen-komponen. Diagram ini sangat berguna saat aplikasi kita berlaku sebagai aplikasi yang dijalankan pada banyak mesin (*distributed computing*).

1. Diagram *Deployment* menggambarkan bentuk *layout* secara fisik bentuk jaringan dan posisi komponen-komponen dari system
2. Pendekatan yang digunakan dalah pendekatan terhadap hasil implementasi/ program

2.6.4. Tools Pendukung UML

Saat ini banyak sekali tool pendesainan yang mendukung UML, baik itu *tool* komersial maupun *opensource*. Beberapa diantaranya adalah:

1. Rational Rose (www.rational.com)
2. Together (www.togethersoft.com)
3. Object Domain (www.objectdomain.com)
4. Jvision (www.object-insight.com)
5. Objecteering (www.objecteering.com)
6. MagicDraw (www.nomagic.com/magicdrawuml)
7. Visual Object Modeller (www.visualobject.com)

Disamping itu, daftar *tool* UML berikut fungsi dan perbandingan kemampuannya juga dapat dilihat di http://www.jeckle.de/umltools.htm. *Pointer* Penting UML sebagai referensi dalam mempelajari dan menggunakan UML, situs-situs yang merupakan pointer penting adalah:

4. http://www.cetus-links.org/oo_uml.html
5. http://www.omg.org
6. http://www.omg.org/technology/uml/
7. http://www.rational.com/uml