BAB 2

LANDASAN TEORI

2.1 Teori Pendidikan

Teori pendidikan adalah suatu teori yang mengemukakan mengenai hal-hal yang berkaitan dengan pendidikan, misalnya mengenai kurikulum , kegiatan belajar, proses pengajaran, system belajar dan lain-lain. Dalam hal ini kurikulum memiliki keterkaitan yang erat dengan teori pendidikan. Ada tiga macam teori pendidikan, yaitu:

(a) Teori Tabularasa (John Locke dan Francis Bacon)

Teori ini mengatakan bahwa anak yang baru dilahirkan itu dapat diumpamakan sebagai kertas putih yang belum ditulisi (*a sheet ot white paper avoid of all characters*). Jadi, sejak lahir anak itu tidak mempunyai bakat dan pembawaan apa-apa. Anak dapat dibentuk sekehendak pendidiknya. Di sini kekuatan ada pada pendidik. Pendidikan dan lingkungan berkuasa atas pembentukan anak.

Pendapat John Locke seperti di atas dapat disebut juga empirisme, yaitu suatu aliran atau paham yang berpendapat bahwa segala kecakapan dan pengetahuan manusia itu timbul dari pengalaman (empiri) yang masuk melalui alat indera.

Kaum behavioris juga berpendapat senada dengan teori tabularasa itu. Behaviorisme tidak mengakui adanya pembawaan dan keturunan, atau sifat-sifat

yang turun-temurun. Semua Pendidikan, menurut behaviorisme, adalah pembentukan kebiasaan, yaitu menurut kebiasaan-kebiasaan yang berlaku di dalam lingkungan seorang anak.

(b) Teori Navitisme (Schopenhauer)

Lawan dari empirisme ialah nativisme. Nativus (latin) berarti karena kelahiran. Aliran nativisme berpendapat bahwa tiap-tiap anak sejak dilahirkan sudah mempunyai berbagai pembawaan yang akan berkembang sendiri menurut arahnya masing-masing. Pembawaan anak-anak itu ada baik dan ada yang buruk. Pendidikan tidak perlu dan tidak berkuasa apa-apa.

Aliran Pendidikan yang menganut paham nativisme ini disebut aliran pesimisme. Sedangkan yang menganut empirisme dan teori tabularasa disebut aliran optimism.

Kedua teori tersebut ternyata berat sebelah. Kedua teori tersebut ada benarnya dan ada pula yang tidak benarnya. Maka dari itu, untuk mengambil kebenaran dari keduanya, William Stern, ahli ilmu jiwa bangsa Jerman, telah memadukan kedua teori itu menjadi satu teori yang disebut teori konvergensi.

(c) Teori Konvergensi (William Stern)

Menurut teori konvergensi hasil pendidikan anak dipengaruhi oleh dua faktor, yaitu pembawaan dan lingkungan. Diakui bahwa anak lahir telah memiliki potensi yang berupa pembawaan. Namun pembawaan yang sifatnya potensial itu harus dikembangkan melalui pengaruh lingkungan, termasuk lingkungan

pendidikan, oleh sebab itu tugas pendidik adalah menghantarkan perkembangan semaksimal mungkin potensi anak sehingga kelak menjadi orang yang berguna bagi diri, keluarga, masyarakat, nusa, dan bangsanya.

Hak negara terhadap pengajaran dan pendidikan juga diterimanya dari Tuhan (bukan negara polisi atau totaliter), seperti hak orang tua terhadap anaknya. Tetapi, hak itu bukan karena kedudukannya sebagai orang tua, melainkan karena gezag atau kekuasaan yang menjadi milik negara untuk memajukan kesejahteraan umum dan mencerdaskan bangsanya, yang sudah menjadi tujuan negara itu sendiri.

Negara mempunyai hak dan kewajiban untuk menyelenggarakan pendidikan dan pengajaran bagi warga negaranya, sesuai dengan dasar-dasar dan tujuan negara itu sendiri, yaitu mengatur kehidupan umum menurut ukuran-ukuran yang sehat sehingga menjadi bantuan bagi pendidikan keluarga dan dapat mencegah apa-apa yang merugikan perkembangan anak untuk mencapai kedewasaannya.

Apabila keluarga tidak mungkin lagi melaksanakan pendidikan seluruhnya (misalnya pendidikan kecerdasan, pengajaran, dan sebagian dari pendidikan sosial;perkumpulan anak-anak), disitulah negara, sesuai dengan tujuannya, harus membantu orang tua dengan jalan mendirikan sekolah-sekolah dan badan-badan sosial lainnya. Demikian juga, negara berhak dan berkewajiban melindungi anak-anak, bila kekuatan orang tua – baik material maupun moral – tidak dapat mencukupi, misalnya karena kurang mampu, tidak sanggup, atau lalai.

2.2 Cerita Rakyat

Cerita rakyat di Indonesia merupakan bagian dari kebudayaan Indonesia. Cerita rakyat di Indonesia mempunyai peranan besar dalam kehidupan sosial budaya Indonesia, yakni pengungkap alam pikiran dan sikap sebagai pendukung nilai kebudayaan masyarakat serta penunjang perkembangan bahasa dan sastra Indonesia dan daerah.

Salah satu sumber kebudayaan nasional adalah kebudayaan daerah. Sastra daerah merupakan penyempurna dan berguna bagi kebutuhan dan kebudayaan nasional bangsa Indonesia. Kebudayaan daerah dan nasional mempunyai hubungan timbal-balik sehingga pembinaan dan pemeliharaan tidak dapat dipisahkan.

Fungsi Cerita rakyat selain sebagai hiburan juga bisa dijadikan suri tauladan terutama cerita rakyat yang mengandung pesan-pesan pendidikan moral. Banyak yang tidak menyadari bahwa negeri kita tercinta ini mempunyai banyak Cerita Rakyat Indonesia yang belum kita dengar, bisa dimaklumi karena cerita rakyat menyebar dari mulut ke mulut yang diwariskan secara turun–temurun. Namun sekarang banyak Cerita rakyat yang ditulis dan dipublikasikan sehingga cerita rakyat Indonesia bisa dijaga dan tidak sampai hilang dan punah.

Cerita rakyat (baca sastra lisan) juga mengandung informasi kebudayaan sebagaimana dikemukan oleh Heddy Shri Ahimsya-Putra(1966) bahwa sebagai suatu bentuk ekspresi budaya masyarakat pemiliknya, sastra lisan tidak hanya

mengandung unsur keindahan (estetik) tetapi juga mengandung berbagai informasi nilai-nilai kebudayaan tradisi yang bersangkutan.

Oleh karena itu, sebagai salah satu data budaya, sastra lisan dapat dianggap sebagai pintu untuk memahami salah satu atau mungkin keseluruhan unsur kebudayaan yang bersangkutan.

Sebagai warisan budaya yang kita miliki, maka sudah sepantasnyalah kita sebagai bagian dari masyarakat untuk melestarikannya agar tidak punah. Sayangnya perkembangan teknologi membuat cerita rakyat tersingkir. Tidak ada lagi orang tua yang menanamkan nilai moral kepada anaknya melalui tradisi tutur tersebut. Anak-anak hanya disuguhi hiburan yang serba instan, semisal tayangan televisi.

2.3 Edukasi dalam Cerita Rakyat

Kebudayaan merupakan konsep yang sangat luas dan kompleks yang dapat diinterpretasikan secara beragam. Selain kebudayaan universal dikenal pula kebuayaan lokal yang menyimpan kearifan lokal. Sementara kearifan lokal yang kesemuanya merupakan sebuah kompleksitas kebudayaan. Salah satu budaya tradisi lisan seperti cerita rakyat juga mengandung kearifan lokal dalam isi ceritanya. Cerita rakyat sebagai bagian dari foklore dapat dikatakan menyimpan sejumlah informasi sistem budaya seperti filosofi, nilai, norma, perilaku masyarakat.

Dalam Cerita Malin Kundang dan cerita Batu yang Menangis bertemakan tentang anak yang durhaka karena tidak mengakui pada orang tuanya. Hal ini

mengajarkan bahwa seorang anak tidak boleh berani bahkan tidak mengakui ibunya meskipun sudah kaya atau berparas cantik.

Selain itu, dalam cerita Legenda Asal Mula Kalimas mengajarkan kepatuhan seorang patih kepada rajanya, Asal Mula Upacara Kasada, dan Lembusura mengajarkan rela berkorban, ajaran tentang kejujuran tersirat dalam cerita Joko Dolog. Apabila digali lebih jauh sebenarnya cerita rakyat mempunyai kedudukan dan fungsi yang sangat penting dalam masyarakat pendukungnya.

Dalam cerita rakyat mengandung nilai luhur bangsa terutama nilai-nilai budi pekerti maupun ajaran moral. Apabila cerita rakyat itu dikaji dari sisi nilai moral, maka dapat dipilah adanya nilai moral individual, nilai moral sosial, dan nilai moral religi.

Adapun nilai-nilai moral individual, meliputi: (1) kepatuhan, (2) pemberani, (3) rela berkorban, (4) jujur, (5) adil dan bijaksana, (6) menghormati dan menghargai, (7) bekerja keras, (8) menepati janji, (9) tahu Balas Budi, (10) baik budi pekerti, (11) rendah hati, dan (12) hati-hati dalam bertindak.

Sedangkan nilai-nilai moral sosial, meliputi: (1) bekerjasama, (2) suka menolong, (3) kasih sayang, (4) kerukunan, (5) suka memberi nasihat, (6) peduli nasib orang lain, dan (7) suka mendoakan orang lain. Nilai-nilai moral religi, meliputi: (1) Percaya Kekuasaan Tuhan, (2) Percaya Adanya Tuhan, (3) Berserah Diri kepada Tuhan/Bertawakal, dan (4) Memohon Ampun kepada Tuhan.

2.4 Aplikasi

Perangkat lunak aplikasi adalah suatu sub kelas perangkat lunak komputer yang memanfaatkan kemampuan komputer langsung untuk melakukan suatu tugas yang diinginkan pengguna. Biasanya dibandingkan dengan perangkat lunak sistem yang mengintegrasikan berbagai kemampuan komputer, tapi tidak secara langsung menerapkan kemampuan tersebut untuk mengerjakan suatu tugas yang menguntungkan pengguna. Contoh utama perangkat lunak aplikasi adalah pengolah kata, lembar kerja, dan pemutar media.

Beberapa aplikasi yang digabungkan bersama menjadi suatu paket kadang disebut sebagai suatu paket atau suite aplikasi (*suite application*). Contohnya adalah Microsoft Office dan OpenOffice.org, yang menggabungkan suatu aplikasi pengolah kata, lembar kerja serta beberapa aplikasi lainnya. Aplikasi-aplikasi dalam suatu paket biasanya memiliki antarmuka pengguna yang memiliki kesamaan sehingga memudahkan pengguna untuk berinteraksi satu sama lain sehingga menguntungkan pengguna. Contohnya, suatu lembar kerja dapat dibenamkan dalam suatu dokumen pengolah kata walaupun dibuat pada aplikasi lembar kerja yang terpisah.

2.5 Data dan Informasi

Hubungan antara data dan informasi sangatlah erat sebagaimana hubungan antara sebab dan akibat. Bahwa data merupakan bentuk dasar dari sebuah informasi,

sedangkan informasi merupakan elemen yang dihasilkan dari suatu bentuk peng-olahan data.

2.5.1 Data

Secara konseptual, data adalah deskripsi tentang benda, kejadian, aktivitas dan transaksi, yang tidak mempunyai makna atau tidak berpengaruh secara langsung kepada pemakai. Data sering kali disebut sebagai bahan mentah informasi.

Berikut adalah kutipan pengertian data dari sudut pandang yang berbeda :

1. Menurut kamus bahasa Inggris-Indonesia, data diterjemahkan sebagai istilah yang berasal dari kata *"datum"* yang berarti fakta atau bahan-bahan keterangan.
2. Dari sudut pandang bisnis, terdapat pengertian data bisnis sebagai berikut: *"Bussines data is an organization's description of things (resources) and events (transactions) that it faces"*. Jadi data, dalam hal ini disebut sebagai data bisnis, merupakan deskripsi organisasi tentang sesuatu (*resource*) dan kejadian (*transaction*) yang terjadi.
3. Menurut (Jogiyanto, HM, 1989 : 8) data merupakan bentuk jamak dari datum (kenyataan) yang berupa fakta-fakta, angka-angka, gambar-gambar yang dapat ditarik kesimpulannya.
4. Gordon B. Davis dalam bukunya *Management Informations System : Conceptual Foundations, Structure, and Development* menyebut data sebagai

bahan mentah dari informasi, yang dirumuskan sebagai sekelompok lambang-lambang tidak acak yang menunjukkan jumlah atau tindakan atau hal-hal lain.

Dari keempat pengertian diatas, dapat diambil kesimpulan bahwa data adalah bahan baku informasi, didefinisikan sebagai kelompok teratur simbol-simbol yang mewakili kuantitas, tindakan, benda, dan sebagainya. Data terbentuk dari karakter, dapat berupa alphabet, angka, maupun simbol khusus seperti *, $, dan /. Data disusun untuk diolah dalam bentuk struktur data, struktur file, dan basis data.

2.5.2 Informasi

Informasi adalah data yang telah diproses menjadi bentuk yang memiliki arti bagi penerima dan dapat berupa fakta, suatu nilai yang bermanfaat. Jadi ada suatu proses transformasi data menjadi suatu informasi (input – proses – output).



Gambar 2.1 Pemrosesan data menjadi informasi

Definisi umum untuk informasi dalam sistem informasi menurut Jogiyanto H.M (1990;11) : “Informasi adalah bentuk data yang dapat diolah yang lebih berguna dan berarti bagi yang menerimanya”. Menurut Robert G. Munik (1973;12): “Informasi adalah data yang telah diolah menjadi suatu bentuk yang berarti bagi penerimanya dan bermanfaat dalam pengambilan keputusan saat ini atau mendatang.

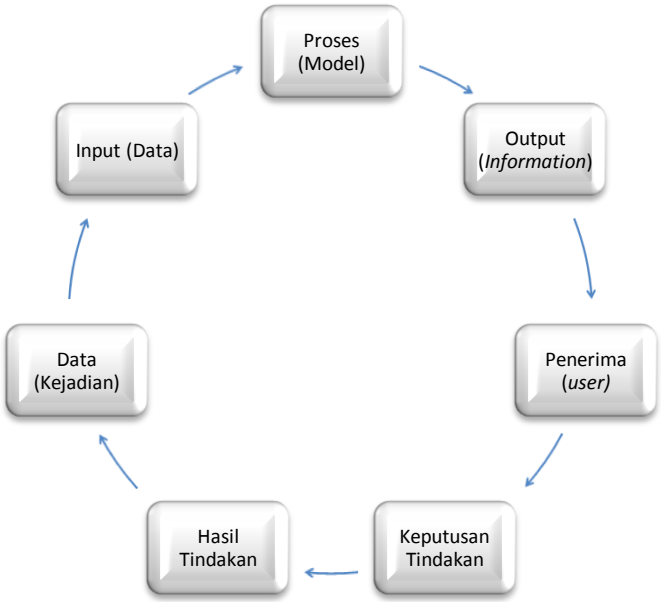
Berikut pengertian informasi dari berbagai sumber :

1. Menurut Gordon B. Davis dalam bukunya "*Management Informations System: Conceptual Foundations, Structures, and Development* menyebut informasi sebagai data yang telah diolah menjadi bentuk yang berguna bagi penerimanya dan nyata, berupa nilai yang dapat dipahami di dalam keputusan sekarang maupun masa depan.
2. Menurut Berry E. Cushing dalam bukunya*"Accounting Information System and Business Organization"* dikatakan bahwa informasi merupakan sesuatu yang menunjukkan hasil pengolahan data yang diorganisasi dan berguna kepada orang yang menerimanya.
3. Menurut Robert N. Anthony dan John Dearden dalam bukunya *"Managemet Control Systems"*, menyebut informasi sebagai suatu kenyataan, data, item, yang menambah pengetahuan bagi penggunanya.
4. Menurut Stephan A. Moscove dan Mark G. Simkin dalam bukunya *"Accounting Information Systems: Concepts and Practise"* mengatakan informasi sebagai kenyataan atau bentuk-bentuk yang bergunay yang dapat digunakan untuk pengambilan keputusan bisnis.

Dari keempat pengertian seperti tersebut di atas dapat disimpulkan bahwa informasi hasil dari pengolahan data menjadi bentuk yang lebih berguna bagi yang menerimanya yang menggambarkan suatu kejadian-kejadian nyata dan dapat digunakan sebagai alat bantu untuk pengambilan suatu keputusan.

2.5.3 Siklus Informasi

Untuk memperoleh informasi yang bermanfaat bagi penerimanya, perlu untuk dijelaskan bagaimana siklus yang terjadi atau dibutuhkan dalam menghasilkan informasi. Pertama-tama data dimasukkan ke dalam model yang umumnya memiliki urutan proses tertentu dan pasti, setelah diproses akan dihasilkan informasi tertentu yang membuat suatu keputusan atau melakukakn tindakan tertentu. Dari keputusan atau tindakan tersebut akan menghasilkan atau diperoleh kejadian-kejadian tertentu yang akan digunakan kembali sebagai data yang nantinya akan dimasukkan ke dalam model (proses), begitu seterusnya. Dengan demikian akan membentuk suatu siklus informasi (*information cycle*) atau siklus pengolahan data (*data processing cycles*), seperti gambar berikut :



Gambar 2.2 Siklus Informasi

2.6 Web Service

Web service adalah suatu sistem perangkat lunak yang dirancang untuk mendukung interoperabilitas dan interaksi antar sistem pada suatu jaringan. Web service digunakan sebagai suatu fasilitas yang disediakan oleh suatu web site untuk menyediakan layanan (dalam bentuk informasi) kepada sistem lain, sehingga sistem lain dapat berinteraksi dengan sistem tersebut melalui layanan-layanan (service) yang disediakan oleh suatu sistem yang menyediakan web service. Web service menyimpan data informasi dalam format XML, sehingga data ini dapat diakses oleh sistem lain walaupun berbeda platform, sistem operasi, maupun bahasa compiler.

Web service bertujuan untuk meningkatkan kolaborasi antar pemrogram dan perusahaan, yang memungkinkan sebuah fungsi di dalam Web Service dapat dipinjam oleh aplikasi lain tanpa perlu mengetahui detil pemrograman yang terdapat di dalamnya.

Beberapa alasan mengapa digunakannya web service adalah sebagai berikut:

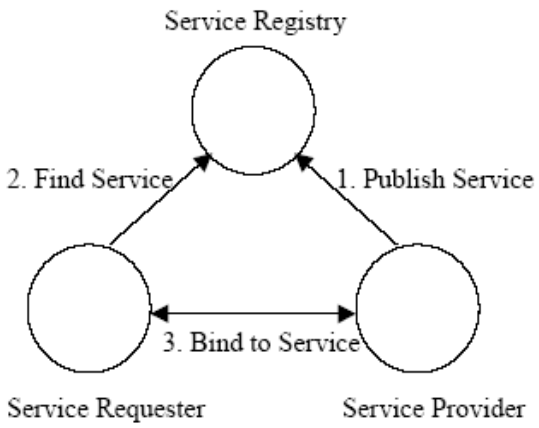
1. Web service dapat digunakan untuk mentransformasikan satu atau beberapa bisnis logic atau class dan objek yang terpisah dalam satu ruang lingkup yang menjadi satu, sehingga tingkat keamanan dapat ditangani dengan baik.
2. Web service memiliki kemudahan dalam proses deployment-nya, karena tidak memerlukan registrasi khusus ke dalam suatu sistem operasi. Web service cukup di-upload ke web server dan siap diakses oleh pihak-pihak yang telah diberikan otorisasi.

3. Web service berjalan di port 80 yang merupakan protokol standar HTTP, dengan demikian web service tidak memerlukan konfigurasi khusus di sisi firewall.

2.6.1 Arsitektur Web Service

Web service memiliki tiga entitas dalam arsitekturnya, yaitu:

1. Service Requester (peminta layanan)
2. Service Provider (penyedia layanan)
4. Service Registry (daftar layanan)



Gambar 2.3 Arsitektur Web Service

a. Service Provider: Berfungsi untuk menyediakan layanan/service dan mengolah sebuah registry agar layanan-layanan tersebut dapat tersedia.
b. Service Registry: Berfungsi sebagai lokasi central yang mendeskripsikan semua layanan/service yang telah di-register.

c. Service Requestor: Peminta layanan yang mencari dan menemukan layanan yang dibutuhkan serta menggunakan layanan tersebut.

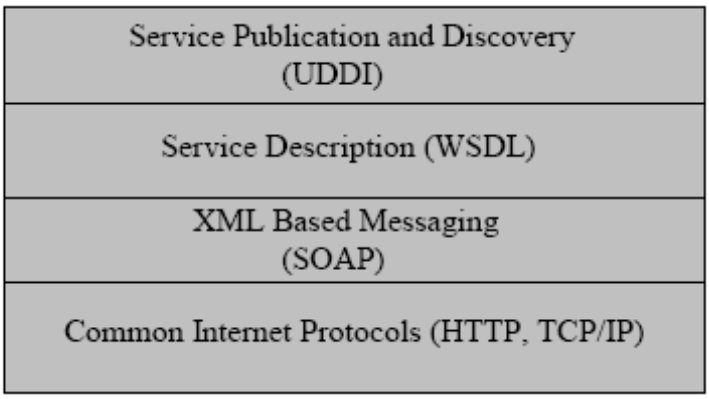
2.6.2 Operasi-Operasi Web Service

Secara umum, web service memiliki tiga operasi yang terlibat di dalamnya, yaitu:

1. Publish/Unpublish: Menerbitkan/menghapus layanan ke dalam atau dari registry.
2. Find: Service requestor mencari dan menemukan layanan yang dibutuhkan.
3. Bind: Service requestor setelah menemukan layanan yang dicarinya, kemudian melakukan binding ke service provider untuk melakukan interaksi dan mengakses layanan/service yang disediakan oleh service provider.

2.6.3 Komponen-Komponen Web Service

Web service secara keseluruhan memiliki empat layer komponen seperti pada gambar 2.4:



Gambar 2.4 Komponen-komponen Web Service

1. Layer 1: Protokol internet standar seperti HTTP, TCP/IP
2. Layer 2: Simple Object Access Protocol (SOAP), merupakan protokol akses objek berbasis XML yang digunakan untuk proses pertukaran data/informasi antar layanan.
3. Layer 3: Web Service Definition Language (WSDL), merupakan suatu standar bahasa dalam format XML yang berfungsi untuk mendeskripsikan seluruh layanan yang tersedia.

2.7 Multimedia

Multimedia adalah gabungan video, audio, grafik dan teks dalam suatu produksi bertingkat berbasis komputer yang dapat dialami secara interaktif atau menurut McCormick multimedia secara umum merupakan kombinasi tiga elemen yaitu suara, gambar dan teks atau menurut Robin dan Linda multimedia merupakan alat yang dapat menciptakan presentasi yang dinamis dan interaktif yang mengkombinasikan teks, grafik, animasi, audio dan gambar video.

Sedangkan menurut wikipedia.org : ”*Multimedia is the use of several different media to convey information (text, audio, graphics, animation, video, and interactivity). Multimedia also refers to computer data storage devices, especially those used to store multimedia content.“*. Orang yang merangkai atau mengkombinasikan elemen multimedia (teks, suara, animasi, grafik serta video)

menjadi suatu produk multimedia yang dapat memenuhi tujuan tertentu dikenal dengan istilah pengembang multimedia (*multimedia developers).*

Ketika pemakai (*user*) suatu produk multimedia dapat mengontrol, apa yang harus ditampilkan serta kapan, sesuai dengan kebutuhan *user*, maka multimedia tersebut dikenal dengan istilah multimedia interaktif (*interactive multimedia).* Meskipun definisi dari multimedia tersebut sederhana, tetapi untuk membuat multimedia sampai berfungsi bukanlah suatu pekerjaan yang mudah.

Tidak cukup hanya mengerti bagaimana membuat tiap-tiap elemen multimedia, tetapi kita harus mengetahui cara penggunaan setiap jenis *tools* multimedia serta teknologinya yang kita pilih, sehingga dapat menghasilkan produk multimedia yang baik mutunya. Dengan pertolongan sebuah tools yang disebut dengan authoring tools, elemen-elemen multimedia dapat diintegrasikan menjadi produk multimedia. *Tools* software ini didesain untuk memanajemen (*to manage)* tiap-tiap elemen multimedia serta dilengkapi dengan fasilitas untuk menghasilkan dan mengedit teks dan citra, dan dapat dihubungkan ke driver dari *videodisc player*, *videotape player* serta hardware pheriperi (*hardware pheriperals)* lainnya.

2.7.1 Sejarah Multimedia

Istilah multimedia berawal dari teater, bukan komputer. Pertunjukan yang memanfaatkan lebih dari satu medium sering kali disebut pertunjukan multimedia. Pertunjukan multimedia mencakup monitor video, *synthesized band*, dan karya seni

manusia sebagai bagian dari pertunjukan. Sistem Multimedia dimulai pada akhir 1980-an dengan diperkenal¬kannya Hypercard oleh Apple pada tahun 1987, dan pengumuman oleh IBM pada tahun 1989 mengenai perangkat lunak Audio Visual Connec¬tion (AVC) dan video adapter card bagi PS/2. Sejak permulaan tersebut, hampir setiap pemasok perangkat keras dan lunak melom¬pat ke multimedia. Pada 1994, diperkirakan ada lebih dari 700 produk dan sistem multimedia di pasaran.

Citra visual dapat dimasukkan ke dalam sistem dari paket perangkat lunak yang menyatukan digital, dan dari kamera video, pita dan piringan video, dan scanner optik. Input audio dapat dimasukkan melalui mikrofon, pita kaset, dan compact disk. Output visual dapat ditampilkan di layar komputer dan di monitor televisi yang tersambung. Output audio dapat disediakan oleh alat output suara, speaker stereo, dan headset. Pada 1990, harga sistem multimedia yang lengkap berkisar $10.000, tapi harganya sejak itu menurun, membuat teknologi itu dalam jangkauan perusahaan kecil yang benar-benar membutuhkan. Namun, harga perangkat keras tidak mencerminkan total biaya untuk Multimedia dalam makalah buku ini digunakan untuk menjelaskan suatu sistem yang terdiri dari perangkat keras, perangkat lunak dan alat-alat lain seperti televisi, monitor video dan sistem piringan optik atau sistem stereo-semua dimaksudkan untuk menghasilkan penyajian audiovisual penuh. Multimedia memungkinkan pemakai komputer untuk mendapatkan output dalam bentuk yang jauh lebih kaya daripada media tabel dan grafik konvensional. Pemakai

dapat melihat gambar tiga dimensi, foto, video bergerak, atau animasi, dan mendengar suara stereo, perekaman suara,atau musik.

2.7.2 Definisi Multimedia

Dalam industri elektronika Multimedia adalah kombinasi dari komputer dan video atau multimedia secara umum merupakan kombinasi tiga elemen, yaitu suara, gambar dan teks atau multimedia adalah kombinasi dari paling sedikit dua media input atau output dari data, media ini dapat audio (suara, musik), animasi, video, teks, grafik dan gambar atau multimedia merupakan alat yang dapat menciptakan presentasi yang dinamis dan interaktif yang mengkombinasikan teks, grafik, animasi, audio dan gambar (Binanto;2010). Definisi yang lain dari multimedia, yaitu dengan menempatkan dalam konteks (Binanto;2010), multimedia adalah pemanfaatan komputer untuk membuat dan menggabungkan teks, grafik, audio, gambar bergerak (video dan animasi) dengan menggabungkan link dan tool yang memungkinkan pemakai melakukan navigasi, berinteraksi, berkreasi dan berkomunikasi.

2.7.3 *Multimedia Personal Computer* (MPC)

Dengan kemajuan teknologi informasi dalam beberapa tahun terakhir ini, semua komputer yang ada dipasaran mempunyai kemampuan untuk memproses (*playback, recording, editing*) citra (*image*), suara dan video selain teks, menjadi suatu elemen multimedia. Komputer yang mempunyai kemampuan tersebut dikenal

dengan istilah Komputer multimedia *(Multimedia computer, MPC)*. Kemampuan ini tercipta disebabkan oleh hardware dan software multimedia yang terpasang pada PC multimedia.

Hardware multimedia antara lain meliputi sound card, video card, cd-player, mix serta box suara. Sound card, video card dan cd player terpasang dan terintegrasi pada motherboards yang merupakan bagian terpenting dari computer. Mix beserta loud speaker terpasang pada sound card sehingga dapat merekam serta dapat mengeluarkan suara dari sutau produk multimedia. PC Multimedia terbagi menjadi dua jenis MPC level 1(*MPC-1)* dan MPC level 2 (*MPC-2).*

Software multimedia terdiri dari software driver dan software aplikasi. Dengan pertolongan software driver, komputer (CPU) dapat mengontrol komponen hardware multimedia, jadi driver itu merupakan interface antara komponen hardware multimedia dengan komputer itu sendiri. Tiap pembelian komponen hardware multimedia, selalu dilengkapi software install untuk memasang driver tersebut pada komputer. Software aplikasi multimedia yang digunakan untuk menghasilkan elmen-elemen multimedia, tersedia dipasaran baik yang dibuat oleh perusahaan software terkenal seperti Microsoft atau perusahaan software lainnya.

2.7.4 Penggunaan Multimedia

Setelah diperkenalkan PC multimedia serta harganya yang dapat terjangkau oleh masyarakat, multimedia dapat diterapkan pada bidang-bidang kehidupan, seperti bisnis, sekolah, rumah, pariwisata, hotel dan lain-lain.

Aplikasi multimedia dibidang bisnis meliputi presentasi, training, pemasaran, dan demonstrasi suatu produk.

Aplikasi dibidang pemasaran mengakibatkan, salesman dapat memberi informasi secara baik dan lengkap tentang produk yang dipasarkan kepada konsumen, karena semua informasi dan pesan yang ingin disampaikan, tersajikan secara singkat dan jelas melalui tayangan gambar, animasi gerak, suara dan sebagainya. Melalui multimedia, konsumen dengan santai dapat mempelajari lebih detil mengenai suatu produk sebelum membeli.

Berkat multimedia untuk training, seorang operator dapat belajar dengan mudah bagaimana mengoperasikan mesin yang baru dibeli, seorang mekanik pada memperbaiki mesin yang rusak sesuai dengan instruksi yang terdapat pada software multimedia.

Pemakaian multimedia disekolah dapat membantu seorang guru dalam menerapkan suatu mata pelajaran yang sulit untuk dicerna oleh anak-anak didiknya, seperti dalam mata pelajaran IPA. Melalui animasi gerak dan visualisasi video, misalnya hukum-hukum newton dapat diterangkan dengan baik, serta dipahami

dengan mudah oleh murid-muridnya. Dengan demikian multimedia dapat meningkatkan qualitas proses belajar-mengajar disekolah.

Dengan pertolongan multimedia seorang tamu yang hendak menginap dihotel, dapat melihat terlebih dulu keadaan kamar, wc beserta perlengkapannya, sebelum memutuskan memilih suatu kamar tertentu. Dengan demikian qualitas service hotel terhadap tamunya dapat diperbaiki. Multimedia dapat membantu dinas kepariwisataan untuk mengiklankan potensi pariwisata daerahnya kepada turis local ataupun mancanegara, seperti kesenian tradisional daerah tersebut, tempat-tempat pariwisata, dan lain-lain.

2.7.5 Elemen-Elemen Multimedia

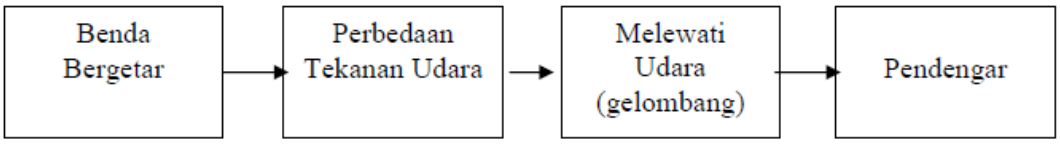
Dalam multimedia terdapat beberapa elemen pendukung, diantaranya sebagai berikut:

1. Suara (*Sound*) Suara adalah sesuatu yang dihasilkan oleh getaran yang berasal dari benda yang mengalami getaran sehingga menghasilkan gelombang yang berada diudara.

   Bentuk gelombang yang berulang-ulang pada waktu tertentu disebut suatu periode. Suatu bentuk gelombang yang tidak menghasilkan suara yang periodik sama seperti sebuah *noise* (gangguan) contohnya seperti suara gaduh. Frekuensi dari suara adalah banyaknya periode gelombang dalam waktu satu detik ukuran satuannya dalam Herz (Hz). 1000 Hz = 1 kiloHerz (kHz). Suara yang dapat

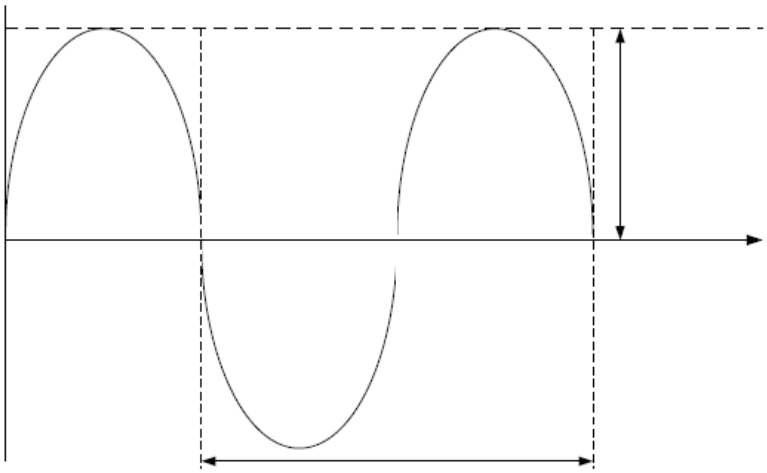
ditangkap/didengar oleh manusia mempunyai lebar frekuensi dari 20 Hz sampai dengan 20 KHz, jenis-jenis frekuensi yang ada sekarang ini adalah sebagai berikut.

- Infra-sound 0 - 20 Hz
- Ultra-sound 20 KHz - 1 GHz
- Hyper-sound 1 GHz - 10 THz



Gambar 2.5 Proses merambatnya gelombang sampai telinga

Amplitudo adalah tinggi suatu gelombang yang mengisyaratkan besar kecilnya suara yang dihasilkan.

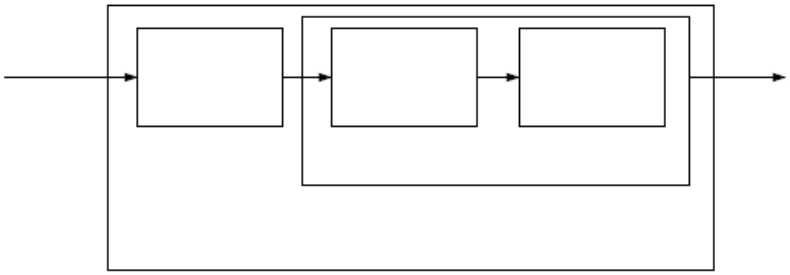


Gambar 2.6 amplitude

2. Audio Digital merupakan bunyi yang terjadi secara berkelanjutan ini dikarenakan adanya gelombang analog. Untuk merubah gelombang analog

kedalam komputer dapat dilakukan dengan cara manipulasi, yaitu melakukan digitalisasi gelombang analog tersebut.

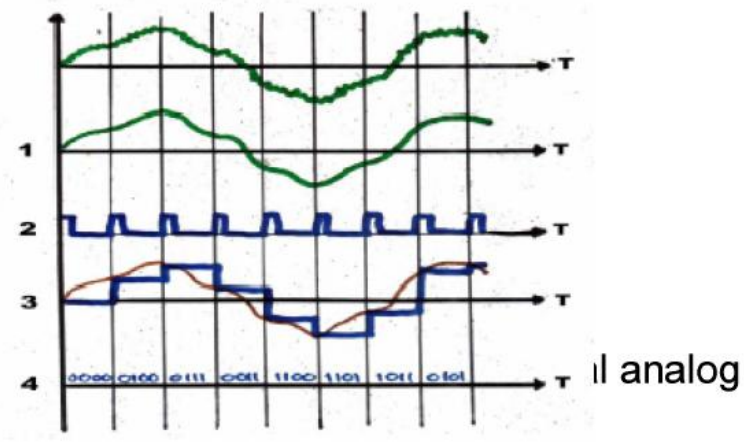
Analog to digital converter (ADC) mengubah amplitudo sebuah gelombang kedalam waktu interval (biasanya dengan *samples*) sehingga menghasilkan representasi digital dari suara. Sebaliknya untuk menampilkan suara digital kedalam alat suara analog (dalam hal ini speaker) digunakan *digital to analog* (DAC) untuk menkonversinya. Beberapa sample gelombang yang diambil dalam waktu 1 detik disebut dengan *sample rate.* Kualitas suara dari CD mempunyai kualitas frekuensi 44.100 Hz, ini berarti sample sebesar 44.100 kali per detik. Ini masih diatas jangkauan frekuensi dengar manusia. Menurut teorema Nyquist mengatakan bahwa "untuk melakukan digitalisasi secara *lossless* (tanpa kerugian) *sample rate* sedikitnya lebih besar 2 kali lebih besar dari frekuensi maksimum yang akan didengar".



Gambar 2.7 *Analog to Digital* (DAC)

Pendengaran manusia dapat mendengar pada kisaran antara 20 Hz sampai dengan 20 KHz, dan *bandwith*nya (lebar pita) adalah (19980 Hz) ini masih dibawah separuh dari kualitas *sampling rate* standar CD suara. Mengikuti

teorema Nyquist, mengartikan kualitas cd suara dapat menggunakan frekuensi sebesar 22050 Hz, ini hampir mendekati frekuensi dengar manusia.



Gambar 2.8 Proses digitalisasi

Tabel 2.1 Perbandingan kualitas suara dengan data rate

Kualitas	Sample Rate	Bits per sample	Mono/stereo	Data rate (tanpa kompresi	Lebar frekuensi
Telepon 3400 Hz	8	8	Mono	8 Kbytes/sec	200 - 3,4 KHz
AM Radio	11,025	8	Mono	11,0 Kbytes/sec	
FM Radio	22,050	16	Stereo	88,2 Kbytes/sec	
CD	44,1	16	Stereo	176,4 Kbytes/sec	20-20 KHz
DAT	48	16	Stereo	176,4 Kbytes/sec	20-20 KHz

Sama halnya dengan gelombang yang tidak berdasarkan waktu. Resolusi atau kuantisasi dari isi *sample* adalah bit yang mewakili amplitudo. Jumlah kapasitas bit yang dipakai menentukan kualitas dari resolusi suara, tapi makin besar

tingkat resolusi mengakibatkan penggunaan kapasitas tempat penyimpanan akan semakin besar.

3. Video Digital dimanfaatkan secara luas untuk berbagai aplikasi penggunaan video tidak terbatas untuk keperluan komunikasi video digital yang dimanfaatkan untuk bidang kesehatan, pendidikan, bisnis, hubungan, dan lain-lain.

   Dalam bidang pendidikan, video digital digunakan untuk *information, access, interaction teaching, distance learning, labolatorium virtual*, hingga perpustakaan elektronik dengan penggunaan video pada *distance learning*. Dosen dan mahasiswa tidak perlu bertatap muka secara langsung, melainkan cukup berinteraksi via *teleconference* dengan memanfaatkan video digital, perpustakaan elektronik menyajikan informasi yang lebih hidup dari sekedar teks dan *image.*

   Ciri teknologi video telah menjadi salah satu teknologi terpenting dalam komunikasi multimedia, *videophone* dimungkinkan dengan resolusi pada teknologi kompresi video sehingga data video dapat disalurkan pada bit rendah.

   Beberapa hal yang terkait dengan video digital:

   1. Karakteristik Video Digital

Video digital pada dasarnya tersusun atas serangkaian *frame*. Rangkaian frame tersebut ditampilkan pada layer dengan kecepatan tertentu, bergantung pada laju *frame* yamg diberikan (dalam *frame*/detik).

Laju *frame* sangat cukup tinggi, mata manusia tidak dapat menangkap gambar per *frame*, melainkan menangkapnya sebagai rangkaian yang kontinyu. Masing-masing *frame* merupakan gambar/citra (*image*) digital suatu *image* dipresentasikan dengan sebuah matriks yang masing-masimg elemenya mepresentasikan nilai intensitas.

Karakterristik suatu video digital ditentukan oleh resolusi (*resolution*) atau dimensi frame (*frame dimention*), kedalaman piksel (*pixel depth*), dan laju frame (*frame rate*). Karakteristik ini akan menentukan kualitas video dan jumlah bit yang dibutuhkan untuk menyimpan/mentransmisikannya.

2. Resolusi / Dimensi Frame

   Resolusi atau dimensi frame adalah ukuran sebuah frame. Resolusi dinyatakan dalam piksel x piksel. Semakin tinggi resolusi, semakin baik kualitas video tersebut dalem artian bahwa dalam ukuran fisik yang sama video dengan resulusi tinggi akan lebih detil. Namun, resolusi yang tinggi akan mengakibatkan jumlah bit yang diperlukan untuk menyimpan/mentransmisikannya meningkat.

3. Kedalaman Bit

Kedalaman bit (*bit depth*) menentukan jumlah bit yang digunakan untuk mempresentasikan tiap piksel pada sebuah frame. Kedalaman bit dinyatakan dalam bit/piksel.

Semakin banyak jumlah bit yang digunakan untuk mepresentasikan sebuah piksel, yang berarti semakin tinggi kedalaman pikselnya, maka semakin tinggi pula kualitas videonya, dengan bayaran jumlah bit yang diperlukan menjadi lebih tinggi.

Dengan 1 byte (8 bit) untuk tiap piksel diperoleh 2^8 atau 256 level intensitas. Dengan level intensitas sebanyak itu umumnya mata manusia dapat memuaskan. Kedalaman piksel paling rendah terdapat pada *binari- velue image* yang hanya menggunakan 1 bit untuk tiap piksel sehingga hanya dua kemungkinan bagi tiap piksel, yaitu 0 (hitam) atau 1 (putih).

4. Laju Frame

   Laju frame (*frame rate*) menunjukan jumlah frame yang digambar tiap detik, dan dinyatakan dengan frame/detik. Sehubungan dengan laju frame ini ada dua hal yang perlu diperhatikan, yaitu kehalusan gerakan (*smoth motion*) dan kilatan (*flash*). Kehalusan gerakan ditentukan oleh jumlah *frame* yang berbeda perdetik. Untuk mendapatkan gerakan halus, video digital setidaknya harus menampilkan 25 *frame* perdetik. Kilatan ditentukan oleh jumlah beberapa kali layer digambar perdetik. Dengan 20 *frame* perdetik, kilatan sudah dapat dilenyapkan.

Video yang berkualitas baik akan memiliki laju frame yang tinggi setidaknya sesuai dengan mata manusia, yang berarti membutuhkan jumlah bit yang lebih tinggi.

4. Animasi

Animasi adalah proses penciptaan efek gerak atau efek perubahan bentuk yang terjadi selama beberapa waktu, animasi bisanya berupa gerak sebuah objek dari tempat yang satu ke tempat yang lain, perubahan warna, atau perubahan bentuk.

2.8 Pemograman Berorientasi Objek

Konsep berorientasi objek saat ini mulai banyak digunakan oleh para pengembang perangkat lunak. Namun konsep tersebut tidak dapat menjangkau formalitas yang dapat dicapai oleh bahasa spesifikasi formal. Salah satu bentuk formalisasi yang dilakukan adalah pada metode berorientasi objek *Unified Modelling Language (UML)*.

Dalam penelitian ini dilakukan proses spesifikasi *requirement* dengan pendekatan metode formal untuk *model use case diagram* yang diarahkan pada proses verifikasi semantik formal untuk membentuk *class diagram*.

Objek dalam *'software analysis & design'* adalah sesuatu berupa konsep (*concept*), benda (*thing*), dan sesuatu yang membedakannya dengan lingkungannya. Secara sederhana objek adalah mobil, manusia, *alarm* dan lain - lainnya. Tapi objek

dapat pula merupakan sesuatu yang abstrak yang hidup di dalam sistem seperti tabel, *database, event, system messages.* Alasan mengapa saat ini pendekatan dalam pengembangan *software* dengan *object-oriented*, pertama adalah *scalability* dimana objek lebih mudah dipakai untuk menggambarkan sistem yang besar dan komplek. Kedua *dynamic modeling*, dapat dipakai untuk permodelan sistem dinamis dan *real time.*

Object oriented system memiliki karakteristik potensial menjadi lebih handal dan mampu mengurangi penyebaran dari program yang berubah menjadi kesalahan, disamping menunjang pemodelan dari dunia nyata. Suatu *object oriented system* dapat menjadi pemikiran sebagai kelompok objek independen yang dapat diminta untuk mengerjakan operasi-operasi tertentu atau menampilkan kelakuan tertentu. Objek–objek ini bekerja sama menyediakan fungsionalitas yang dibutuhkan sistem. Objek-objek ini memiliki identitas dan dapat dibuat sebagai program eksekusi. Untuk menyediakan karakteristik yang sesuai dari *object oriented system*, objek-objek dienkapsulasi yakni mereka hanya dapat diakses melalui *message* yang dikirimkan kepada mereka untuk meminta performa dari operasi yang mereka definisikan.

Objek dapat dilihat sebagai ‘kotak hitam’ yang detil internalnya tersembunyi dari amatan luar dan tidak dapat dimodifikasi secara biasa. Enkapsulasi didefinisikan *Grady Booch* sebagai proses penggolongan elemen–elemen dari suatu abstraksi yang merupakan struktur dan kelakuannya. Ia melayani untuk memisahkan antar muka yang sudah disepakati dari suatu abstraksi dan implementasinya.

Objek-objek juga menampilkan property pengganti, yang berarti bahwa objek menyediakan operasi-operasi yang kompatibel yang dapat disubstitusikan antara satu dengan yang lain. Objek memiliki status, kelakuan, dan identitas.

Dalam dunia pemodelan, metodologi implementasi objek walaupun terikat kaidah-kaidah standar, namun teknik pemilihan objek tidak terlepas pada subyektifitas *software analyst & designer*. Beberapa objek akan diabaikan dan beberapa objek menjadi perhatian untuk diimplementasikan di dalam sistem. Hal ini sah-sah saja karena kenyataan bahwa suatu permasalahan sudah tentu memiliki lebih dari satu solusi. Ada 3 (tiga) teknik/konsep dasar dalam *OOA/D*, yaitu:

1. Pembungkusan *(encapsulation)*, mengumpulkan data dengan pemanipulasian menjadi satu entitas. Terkait pada konsep penyembunyian informasi *(information hiding)*:
    a. Pemanipulasian rinci informasi dilakukan secara internal.
    b. Dunia luar hanya mengetahui informasi secukupnya dengan cara yang sudah ditentukan sebelumnya.
        - Objek, suatu entitas yang memiliki identitas, status, perilaku.
        - Kelas, abstraksi dari sejumlah objek dengan karakteristik yang serupa.
2. Pewarisan *(inherintance)*, pembentukan suatu hirarki atau jejaring akibat peng-umuman atau peng-khususan (bergantung pada fokus perhatian).
3. Kebanyakrupaan *(polymorphisme)* :
    a. Pensubstitusian dinamis *supertype* oleh *subtype*

b. Pemanipulasian objek secara seragam dengan tipe yang berbeda-beda.

2.8.1 *Object Management Group*

OMG (Object Management Group) adalah konsorsium vendor, pengembang, dan pengguna *software* yang dibentuk pada tahun 1989 yang ditujukan untuk mempromosikan penggunaan teknologi berarah objek pada aplikasi *software.*

Metode berorientasi objek merupakan metode yang relatif baru dan sekarang menjadi cukup populer di kalangan pengembang sistem informasi (e.g. Iivari dan Maansaari, 1998). Metode ini berfokus pada objek yang konsisten mulai tahap analisis, perancangan, dan implementasi sistem informasi. Salah satu varian metode ini yang paling kontemporer adalah *Unified Modelling Language (UML)* yang diperkenalkan oleh *Rumbaugh, Jacobson, dan Booch, 1999*.

Dimulai pada bulan Oktober 1994 *Booch, Rumbaugh dan Jacobson*, yang merupakan tiga tokoh yang bisa dikatakan metodologinya banyak digunakan mempelopori usaha untuk penyatuan metodologi pendesainan berorientasi objek. Pada tahun 1995 dirilis draft pertama dari *UML (versi 0.8)*. Sejak tahun 1996 pengembangan tersebut dikoordinasikan oleh *Object Management Group* (OMG – http://www.omg.org). Tahun 1997 *UML versi 1.1* muncul, dan saat ini versi terbaru adalah versi 1.5 yang dirilis bulan Maret 2003. *Booch*, *Rumbaugh* dan *Jacobson* menyusun tiga buku serial tentang *UML* pada tahun 1999, antara lain:

1. *The Unified Modeling Language User Guide*
2. *The Unified Software Development Process*

3. The Unified Modeling Language Reference Manual

Sejak saat itulah *UML* telah menjelma menjadi standar bahasa pemodelan untuk aplikasi berorientasi objek.

2.8.2 *Artifact UML*

Diagram–diagram yang digunakan untuk mendefinisikan *UML* adalah sebagai berikut:

a. Use case diagram

Menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang atau sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. *Use case diagram* dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem.

Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal. Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga

duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*. Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.

b. Class Diagram

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi).

Class diagram menggambarkan struktur dan deskripsi *class, package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain.

Class memiliki tiga area pokok :

1. Nama (dan *stereotype)*

2. Atribut

3. Metoda

Atribut dan metoda dapat memiliki salah satu sifat berikut :

1. *Private,* tidak dapat dipanggil dari luar *class* yang bersangkutan.
2. *Protected,* hanya dapat dipanggil oleh *class* yang bersangkutan dan anak anak yang mewarisinya.
3. *Public,* dapat dipanggil oleh siapa saja.

Class dapat merupakan implementasi dari sebuah *interface*, yaitu *class* abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class.* Diagram demikian *interface* mendukung resolusi metoda pada saat *run-time*.

Sesuai dengan perkembangan *class* model, *class* dapat dikelompokan menjadi *package*. *Class* dapat membuat diagram yang terdiri atas *package.*

Hubungan antar *Class* :

1. Asosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain. Panah *navigability* menunjukan arah *query* antar *class.*

2. Agregasi, yaitu hubungan yang menyatakan bagian ("terdiri atas").

3. Pewarisan, yaitu hubungan hirarkis antar *class. Class* dapat diturunkan dari class lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.

4. Hubungan dinamis, yaitu rangkaian pesan (*message*) yang di *passing* dari satu *class* kepada *class* lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence diagram* yang akan dijelaskan kemudian.

c. Statechart Diagram

Statechart Diagram merupakan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari *stimuli* yang

diterima. Pada umumnya *Statechart Diagram* menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *Statechart Diagram*).

Dalam *UML*, *state* digambarkan berbentuk segi empat dengan sudut membulat dan memiliki nama sesuai dengan kondisinya saat itu. Transisi antar *state* umumnya memiliki kondisi *guard* yang merupakan syarat terjadinya transisi yang bersangkutan dituliskan dalam kurung siku. *Action* yang dilakukan sebagai akibat dari *event* tertentu dituliskan dengan diawali garis miring. Titik awal dan titik akhir digambarkan berbentuk lingkaran berwarna penuh dan berwarna setengah.

d. Activity Diagram

Activity Diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana aktivitas itu berakhir. *Activity diagram* juga dapat menggambarkan proses pararel yang mungkin terjadi pada beberapa eksekusi. *Activity diagram* merupakan *state diagram* khusus, dimana sebagian besar *state* adalah *action* dan sebagian besar transisi di *triger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan *behaviour* internal sebuah sistem (dan interaksi antar subsistem) secara eksak , tetapi lebih menggambarkan proses-proses dan jalur aktivitas dari level atas secara umum.

Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan

bagaimana aktor menggunakan sistem untuk melakukan aktivitas. Sama dengan *state*, standar *UML* menggunakan segi empat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan *behaviour* pada kondisi tertentu. Untuk mengilustrasikan proses-proses pararel (*fork on join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal. *Activity diagram* dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.

e. Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan teradap waktu. *Sequence diagram* terdiri atas dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respon dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan.

Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi operasi/metoda dari *class*. *Actiation bar* menunjukan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message*. Untuk objek-objek yang memiliki sifat khusus, standar

UML mendefinisikan *icon* khusus untuk objek *boundary*, *controller* dan *persistent entity.*

f. Collaboration Diagram

Collaboration diagram juga menggambarkan interaksi antar objek seperti *sequence diagram*, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian *message*. Setiap *message* memiliki *sequence number,* dimana *message* dari level tertinggi memiliki nomor satu. *Message* dari level yang sama memiliki prefiks yang sama.

g. Component Diagram

Component diagram menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (*dependency)* diantaranya. Komponen piranti lunak adalah modul berisi *code*, baik berisi *source code* maupun *binary code*, baik *library* maupun *executable*, baik yang muncul pada *compile time, link time,* maupun *run time.* Umumnya komponen terbentuk dari beberapa *class* atau *package* tapi juga dari komponen-komponen yang lebih kecil. Komponen dapat juga berupa *interface*, yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain.

h. Deployment Diagram

Deployment/phisycal diagram menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, dimana komponen akan terletak (pada mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal. Sebuah *node* adalah server,

workstation atau piranti keras lain yang digunakan untuk men-*deploy* komponen dalam lingkungan sebenarnya. Hubungan antar *node* (misalnya *TCP/IP*) dan *requirement* dapat juga didefinisikan dalam diagram ini.

2.8.3 Semantik Dalam *UML*

Seperti bahasa-bahasa lainnya, *UML* mendefinisikan notasi dan *syntax* atau semantik. Notasi *UML* merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan *UML syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi *UML* terutama diturunkan dari 3 notasi yang telah ada sebelumnya yaitu *Grady Booch OOD (Object-Oriented Design)*, *Jim Rumbaugh OMT (Object Modeling Technique)*, dan *Ivar Jacobson OOSE (Object-Oriented Software Engineering)*.

OMG telah menetapkan semantik (makna istilah) semua notasi *UML* dalam *Model Struktural* dan *Model Behavioral.*

1. *Model Struktural* atau model statis, menekankan pada struktur objek dalam sebuah sistem, menyangkut kelas-kelas, *interface*, *atribute* dan hubungan antar komponen.
2. *Model Behavioral* atau model dinamis, menekankan pada perilaku objek dalam sebuah sistem, menyangkut metode, interaksi, kolaborasi dan *state history.*

2.8.4 Faktor Yang Mendorong Dibuatnya *UML*

Saat ini piranti lunak semakin luas dan besar lingkupnya, sehingga tidak bisa lagi dibuat asal-asalan. Piranti lunak saat ini seharusnya dirancang dengan memperhatikan hal-hal seperti *scalability*, *security*, dan eksekusi yang *robust* walaupun dalam kondisi yang sulit. Selain itu arsitekturnya harus didefinisikan dengan jelas, agar *bug* mudah ditemukan dan diperbaiki, bahkan oleh orang lain selain *programmer* aslinya. Keuntungan lain dari perencanaan arsitektur yang matang adalah dimungkinkannya penggunaan kembali modul atau komponen untuk aplikasi piranti lunak lain yang membutuhkan fungsionalitas yang sama. Pemodelan (*modeling*) adalah proses merancang piranti lunak sebelum melakukan pengkodean (*coding*). Model piranti lunak dapat dianalogikan seperti pembuatan *blueprint* pada pembangunan gedung. Membuat model dari sebuah sistem yang kompleks sangatlah penting karena kita tidak dapat memahami sistem semacam itu secara menyeluruh. Semakin komplek sebuah sistem, semakin penting pula penggunaan teknik pemodelan yang baik.

Dengan menggunakan model, diharapkan pengembangan piranti lunak dapat memenuhi semua kebutuhan pengguna dengan lengkap dan tepat, termasuk faktor-faktor seperti *scalability, robustness, security*, dan sebagainya. Kesuksesan suatu pemodelan piranti lunak ditentukan oleh tiga unsur, yang kemudian terkenal dengan sebutan segitiga sukses (*the triangle for success*). Ketiga unsur tersebut adalah metode pemodelan (*notation*), proses (*process*) dan *tool* yang digunakan.

2.8.5 Tujuan *UML*

Pada dunia *software engineering*, *UML* merupakan singkatan dari *Unified Modelling Language*, yaitu suatu metode *modeling* generasi ketiga dan bahasa spesifikasi yang sifatnya *non-proprietary*. Sebenarnya penggunaan dari *UML* itu sendiri tidak terbatas hanya pada dunia *software modeling*, tetapi bisa pula digunakan untuk *modeling hardware (engineering systems)* dan sering digunakan sebagai *modeling* untuk proses bisnis dan juga *modeling* untuk struktur organisasi. *UML* adalah suatu metode terbuka yang digunakan untuk menspesifikasi, memvisualisasi, membangun dan mendokumentasikan artifak-artifak dari suatu pengembangan sistem *software* yang berbasis pada objek. *UML* merupakan hasil kompilasi *best engineering practice* yang sudah terbukti sukses dalam pemodelan sistem yang besar, sistem yang kompleks, khususnya pada level arsitektural.

UML mengintegrasikan konsep dari *Booch, OMT, OOSE* dan juga *Class-Relation* dengan menggabungkan mereka menjadi suatu kesatuan bahasa *modeling* yang bisa berguna bagi siapa saja. *UML* bertujuan untuk menjadi standar bahasa *modeling* yang mampu untuk memodelkan sistem yang konkuren dan juga terdistribusi. Tujuan yang ingin dicapai dari penggunaan *UML* adalah:

1. Memberikan model yang siap pakai, bahasa permodelan visual yang ekspresif untuk mengembangkan dan saling menukar model yang mudah dan dimengerti secara umum.

2. Memberikan bahasa permodelan yang bebas dari berbagai bahasa pemrograman dan proses rekayasa.
3. Menyatukan praktek-praktek terbaik yang terdapat dalam permodelan.

Dan bahasa pemodelan *Unified Modeling Language (UML)* ini juga biasa digunakan untuk :

1. Menggambarkan batasan sistem dan fungsi-fungsi sistem secara umum, dibuat dengan *use case* dan *actor.*
2. Menggambarkan kegiatan atau proses bisnis yang dilaksanakan secara umum, dibuat dengan *interaction diagrams.*
3. Menggambarkan representasi struktur statik sebuah sistem dalam bentuk *class diagrams.*
4. Membuat *model behavior* yang menggambarkan kebiasaan atau sifat sebuah sistem dengan *state transition diagrams.*
5. Menyatakan arsitektur implementasi fisik menggunakan *component and development diagrams.*
6. Menyampaikan atau memperluas fungsionalitas dengan *stereotypes.*

2.8.6 Cakupan *UML*

Model *UML* dapat mencakup banyak perbendaharaan. Diantaranya adalah sebagai berikut :

1. Things :

a. *Structural*

1. *Use case*, deskripsi interaksi dengan *external actor*.
2. *Class*, deskripsi untuk objek-objek.
3. *Interface*, kumpulan operasi yang memberikan *service* tertentu untuk *class* atau *component.*
4. *Component*, bagian sistem yang dapat diganti *(replaceable)* dan realisasinya sesuai dengan *interface.*

b. *Behaviour*

1. Interaksi *(message sequence chart)*, pertukaran *messages* antar objek.
2. *State machine*, urutan *state* dari objek dalam berinteraksi dengan objek lain.

c. *Grouping*

Package, mekanisme untuk mengumpulkan elemen ke dalam satu set (group).

d. *Anotasi*

Catatan atau keterangan (teks) sebagai dokumentasi .

2. *Relationship :*

a. *Dependency*

Hubungan antar elemen dimana perubahan pada elemen yang satu dapat mempengaruhi elemen yang lain *(dependent)*.

b. *Association*

Hubungan struktur antara elemen dan bertindak sebagai *link.*

c. Generalization

Hubungan dimana elemen yang special *(child)* mewarisi elemen yang umum *(parent).*

d. Realization

1. Hubungan (semantik) antara 2 elemen, dimana elemen yang satu memberikan kontrak dan elemen yang lain menjamin realisasi kontrak tersebut.
2. Dimana elemen yang special *(child)* mewarisi elemen yang umum *(parent).*

3. *Diagram* : *use case diagram, class diagram, msc diagram,* dan lain–lain.
4. *UML Extensible*: dapat didefinisikan perbendaharaan baru berdasarkan yang telah ada.

Selain cakupan–cakupan di atas, ada tiga aspek utama dalam pemodelan sistem yang mampu didukung oleh *UML*:

1. *Functional Model*, untuk menunjukkan fungsionalitas dari suatu sistem dari sudut pandang *user* atau pengguna. Ini dicapai dengan menggunakan *Use Case Diagram.*
2. *Object Model,* untuk menunjukkan struktur dan substruktur dari suatu sistem dengan menggunakan *object*, atribut, operasi dan juga asosiasi. Ini dicapai dengan menggunakan *Class Diagram*.

3. *Dynamic Model*, menunjukkan *internal behavior* dan suatu sistem. Ini dicapai dengan menggunakan *Sequence Diagram*, *Activity Diagram* dan juga *Statechart Diagram*.

2.8.7 Perbedaan *UML* Dengan Bahasa Pemodelan Lain

Perbedaan yang paling mendasar jika kita menggunakan pemograman terstruktur seperti *DFD* dengan *UML* adalah, *Data Flow Diagram* sebagai *tools design system* berdasarkan *procedural* dan *Unified Modeling Language* sebagai *tools design OOP*. Salah satu kelebihan dari digunakannya *Unified Modelling Language* adalah kedekatan antara artifak yang sesungguhnya dengan model yang kita buat. Mental model dan cara berpikir kita apabila dibuat sedekat mungkin dengan artifak yang sesungguhnya memudahkan kita untuk berpikir 'objek-objek' apa saja dan interelasi antar objek yang terdapat dalam sebuah model. Sedangkan kekurangannya adalah tidak semua model bisa dilakukan dengan mudah dengan pendekatan ini. Misalnya *scientific model*, dan semacamnya.

Dengan menggunakan *UML* kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena *UML* juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa pemrograman

berorientasi objek seperti *C++*, *Java*, *C#* atau *VB.NET*. Walaupun demikian, *UML* tetap dapat digunakan untuk *modeling* aplikasi prosedural dalam *VB* atau *C*.

2.8.8 Alat Bantu *UML*

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

1. *Rational Rose*

Dasar-dasar pemodelan dengan *Rational Rose*:

a. Visual Modelling

Sistem terdiri dari komponen-komponen, *interface* dan relasi-relasi yang ada dalam sistem tersebut. Semua semantik dan notasi dalam *UML* dibuat untuk digunakan dalam *visual modelling.*

b. Software Engineering

Model dalam Rekayasa *Software*:

1. Sistem besar, didekati dengan sudut pandang–sudut pandang yang kecil yang satu sama lain hampir saling bebas.
2. Cara pandang membuat *software* dalam *Rational Rose*:

   a) Use Case View

   Use case view melihat pada bagaimana *actor* dan *use-case* berinteraksi. Pada *use case view* ada beberapa diagram yang digunakan:

1. *Use-Case Diagram*
2. *Sequence Diagram*
3. *Collaboration Diagram*
4. *Activity Diagram*

b) Logical View

Logical view mengarah pada persyaratan *(requirements)* fungsional sistem. View ini melihat pada kelas-kelas dan hubungan antar kelas-kelas tersebut. Diagram yang digunakan dalam *view* ini:

1. Class Diagram
2. Sequence Diagram
3. Collaboration Diagram
4. Statechart Diagram

c) Component View

Component view mengarah pada pengaturan *software*. *View* ini mengandung informasi mengenai komponen *software*, komponen *executable*, dan *library* untuk sistem yang kita modelkan. Diagram yang digunakan pada view ini adalah *component diagram.*

d) Deployment View

Deployment view memperlihatkan pemetaan setiap proses kedalam *hardware*. Diagram yang digunakan pada *view* ini adalah *deployment diagram*.

c. Business Model

Dalam rekayasa *software*, model bisnis sangat membantu kita untuk memahami masalah yang harus diselesaikan oleh *software* yang akan kita buat. *Business use case model* menggambarkan proses-proses dalam suatu bisnis dan interaksi–interaksi dengan pihak luar seperti para *customer* dan *partner*. Maksud dari proses bisnis disini adalah:

1. Sekumpulan aktivitas yang dirancang untuk menghasilkan keluaran tertentu bagi aktor tertentu. Proses bisnis menekankan pada bagaimana sebuah pekerjaan dikerjakan dalam sebuah organisasi, dengan berfokus pada produk yang dihasilkan oleh proses tersebut.
2. Urutan aktivitas tertentu terhadap waktu dan tempat, dengan sebuah titik awal, sebuah titik akhir dan berbagai input dan output yang didefinisikan dengan baik.

2. Together

3. Object Domain

4. Jvision

5. Objecteering

6. MagicDraw

7. Visual Object Modeller.

2.9 Java

Java adalah bahasa pemrograman yang dapat dijalankan di berbagai komputer termasuk telepon genggam. 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.

2.9.1 Sejarah Perkembangan Java

Bahasa pemrograman Java 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).

Versi awal Java ditahun 1996 sudah merupakan versi release sehingga dinamakan Java Versi 1.0. Java versi ini menyertakan banyak paket standar awal yang terus dikembangkan pada versi selanjutnya:

1. java.lang: Peruntukan kelas elemen-elemen dasar.
2. java.io: Peruntukan kelas input dan output, termasuk penggunaan berkas.
3. java.util: Peruntukan kelas pelengkap seperti kelas struktur data dan kelas kelas penanggalan.

4. java.net: Peruntukan kelas TCP/IP, yang memungkinkan berkomunikasi dengan komputer lain menggunakan jaringan TCP/IP.
5. java.awt: Kelas dasar untuk aplikasi antarmuka dengan pengguna (GUI)
6. java.applet: Kelas dasar aplikasi antar muka untuk diterapkan pada penjelajah web.

2.9.2 Kelebihan Java

Ada beberapa keunggulan Java yang membuatnya menjadi salah satu bahasa pemrograman utama, yaitu:

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).

3. 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.
4. 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.
5. 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).

2.9.3 Kekurangan Java

Dibalik kelebihan Java, ada beberapa kekurangan dari bahasa Pemprograman Java ini, yaitu diantaranya:

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/direverse-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.

2.9.4 Java Platform

Java mencakup banyak aspek sehingga dapat membingungkan. Istilah-istilah seperti J2SE, J2EE, J2ME, Struts, Spring, Seam, Servlet, Applet, Hibernate, Ibatis, Pojo, JSF, JPA dapat terlihat mengerikan. Kuncinya adalah tidak semua hal mengenai

Java harus dipelajari sekaligus. Yang terpenting adalah dasarnya dulu, kemudian yang lain dapat dipelajari sesuai dengan kebutuhan.

2.10 Eclipse

Eclipse merupakan sebuah IDE (*Integrated Development Environment*) untuk mengembangkan perangkat lunak dan dapat dijalankan di semua platform (platform-independent). Berikut adalah sifat dari Eclipse :

1. *Multi-platform*: Dapat berjalan pada berbagai platform seperti, Microsoft Windows, Linux, Solaris, AIX, HP-UX, dan Mac OS X.
2. *Multi-language*: Eclipse dikembangkan dengan bahasa pemrograman Java, akan tetapi Eclipse mendukung pengembangan aplikasi berbasis bahasa pemrograman lainnya, seperti C/C++, Cobol, Phyton, Perl, PHP, dan lain sebagainya.
3. *Multi-role*: Selain sebagai IDE untuk pengembangan aplikasi, Eclipse pun bisa digunakan untuk aktivitas dalam siklus pengembangan perangkat lunak, seperti dokumentasi, test perangkat lunak, pengembangan web, dan lain sebagainya

Eclipse pada saat ini merupakan salah satu IDE yang difavoritkan karena gratis dan *open source*, yang berarti setiap orang boleh melihat kode program perangkat lunak. Selain itu, kelebihan dari Eclipse yang membuatnya popular adalah kemampuannya untuk dapat dikembangkan oleh pengguna dengan komponan yang dinamakan *plug-in*.

2.10.1 Sejarah Eclipse

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

2.10.2 Arsitektur Eclipse

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

1. *Core Platform*
2. OSGi
3. SWT (*Standard Widget Toolkit*)
4. JFace
5. *Eclipse Workbench*

Standard pengoperasian Eclipse selalu dilengkapi dengan JDT (*Java Development Tools*), plug-in yang membuat Eclipse kompatibel untuk mengembangkan program Java, dan PDE (*Plug-in Development Environment*) untuk

mengembangkan *plug-in* baru. Eclipse beserta *plug-in* nya diimplementasikan dalam bahasa pemrograman Java.

Konsep Eclipse adalah IDE yang terbuka (*open*), mudah diperluas (*extensible*) untuk apa saja, dan tidak untuk sesuatu yang spesifik. Jadi, Eclipse tidak saja mengembangkan program Java, akan tetapi dapat digunakan untuk berbagai macam keperluan, cukup dengan menginstal *plug-in* yang dibutuhkan. Apabila ingin mengembangkan program C/C++ terdapat *plug-in* CDT (C/C++ *Development Tools*). Selain itu, pengembangan secara visual bukan hal yang tidak mungkin oleh Eclipse, plug-in UML2 tersedia untuk membuat diagram UML. Dengan menggunakan PDE setiap orang bisa membuat *plug-in* sesuai dengan keinginannya.

2.10.3 Perkembangan Eclipse

Sejak tahun 2006, Eclipse Foundation mengkoordinasikan peluncuran Eclipse secara rutin dan simultan yang dikenal dengan nama *Simultaneous Release*. Setiap versi peluncuran terdiri dari Eclipse Platform dan juga sejumlah proyek yang terlibat dalam proyek Eclipse. Tujuan dari sistem ini adalah untuk menyediakan distribusi Eclipse dengan fitur-fitur dan versi yang terstandarisasi. Hal ini juga dimaksudkan untuk mempermudah pengembangan (*deployment*) dan perawatan (*maintenance*) untuk sistem enterprise, serta untuk kenyamanan. Peluncuran simultan dijadwalkan pada bulan Juni setiap tahunnya.

Tabel 2.2 Perkembangan Versi Eclipse

Kode Peluncuran	**Tanggal Peluncuran**	**Platform**	**Nama Proyek**
Eclipse 3.0	21 Juni 2004	3.0	-
Eclipse 3.1	28 Juni 2005	3.1	-
Callisto	30 Juni 2006	3.2	Callisto projects
Europa	29 Juni 2007	3.3	Europa projects
Ganymede	25 Juni 2008	3.4	Ganymede projects
Galileo	24 Juni 2009	3.5	Galileo projects
Helios	23 Juni 2010	3.6	Indigo projects
Juno	27 Juni 2012	4.2	Juno projects

2.11 Software Development Kit (SDK)

Software Development Kit (SDK atau *devkit*) adalah sekumpulan alat pengembangan yang memungkinkan untuk menciptakan sebuah aplikasi untuk paket perangkat lunak tertentu (*software package*), framework perangkat lunak, platform perangkat keras, komputer sistem, konsol video game, sistem operasi, atau platform yang sama.

2.12 Java Development Kit (JDK)

Java Development Kit atau disingkat JDK merupakan produk dari Oracle Corporation yang ditujukan untuk pengembangan perangkat lunak berbasis Java. Sejak Java diperkenalkan, sampai saat ini Java SDK telah banyak digunakan. Pada tanggal, 17 November 2006, Sun memperkenalkan JDK dibawah lisensi GNU *General Public License* (GPL), sehingga penggunaannya bebas (gratis).

JDK juga dilengkapi dengan JRE (*Java Runtime Environment*), biasanya disebut *private*. JRE terdiri dari JVM (*Java Virtual Machine*) dan semua *library class* yang terdapat lingkungan produksi, sama baiknya dengan penambahan hanya library-library yang berguna bagi *developers*, seperti *library internationalization* dan *library* IDL.

2.13 Android Development Tools (ADT)

Android Development Tools (*ADT*) adalah *plugin* untuk Eclipse *Intergrated Development Environment* (*IDE*) yang dirancang untuk memberikan lingkungan yang terpadu di mana untuk membangun aplikasi Android.

ADT memperluas kemampuan Eclipse untuk membiarkan para *developer* lebih cepat dalam membuat proyek baru Android, membuat aplikasi UI, menambahkan komponen berdasarkan Android *Framework* API, *debug* aplikasi dalam pengunaan Android SDK, dan membuat file APK untuk mendistribusikan aplikasi.

Mengembangkan aplikasi di Eclipse dengan ADT sangat dianjurkan dan merupakan cara tercepat untuk memulai membuat aplikasi android, karena banyak kemudahan-kemudahan sebagai tools yang terintegrasi seperti, custom XML editor, dan *debug panel ouput*. Selain itu ADT memberikan dorongan luar biasa dalam mengembangkan aplikasi Android.

2.14 Android

Android adalah software platform yang open source untuk mobile device. Android berisi sistem operasi, middleware dan aplikasi-aplikasi dasar. Basis OS Android adalah kernel linux 2.6 yang telah dimodifikasi untuk mobile device. Android sendiri memiliki banyak fitur diantaranya adalah:

1. Menulis file di SD card.
2. Database dengan SQLite.
3. Application framework berbasis komponen yang memudahkan reuse.
4. Dalvik virtual machine dioptimisasi untuk menjalankan program Java di mobile devices
5. Integrated browser berbasis WebKit engine
6. Optimized graphics tersedia 2D graphics library; dan OpenGL ES 1.0 untuk 3D graphics
7. Media support untuk MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF
8. GSM Telephony fasilitas telepon.

9. Bluetooth, EDGE, 3G, dan WiFi
10. Kamera, GPS, compass dan accelerometer
11. Multitouch.

Android versi 1.0 dikeluarkan tanggal 23 Sep 08, versi 1.1 pertama kali digunakan pada mobile phone (T1). Versi 1.5 (Cupcake) muncul April 09 yang mendukung soft keyboard, 1.6 (Donut) Sept 09 dan versi-versi terakhir yaitu:

1. Versi 2.0/2.1 (Eclair) memperbaiki user interface, HTML5 dan Exchange ActiveSync 2.5.
2. Versi 2.2 (Froyo), mempercepat kinerja dengan Just In Time compiler dan Chrome V8 JavaScript engine, Wi-Fi hotspot tethering dan suppport Adobe Flash.
3. Versi 2.3 (Gingerbread), memperbaiki user interface, soft keyboard, copy/paste features dan support ke Near Field Communication (NFC).
4. Versi 3.0 (Honeycomb), dirancang khusus untuk tablet. Android versi ini mendukung ukuran layar yang lebih besar. Honeycomb juga mendukung multi prosesor dan juga akselerasi perangkat keras (hardware) untuk grafis.
5. Versi 4.0 (Ice-cream sandwich), Diumumkan pada tanggal 19 Oktober 2011, membawa fitur Honeycomb untuk smartphone dan menambahkan fitur baru termasuk membuka kunci dengan pengenalan wajah, jaringan data pemantauan penggunaan dan kontrol, terpadu kontak jaringan sosial, perangkat tambahan

fotografi, mencari email secara offline, dan berbagi informasi dengan menggunakan NFC.

2.14.1 Arsitektur Android

Android terdiri dari beberapa *stack software* yang terdiri dari: *Applications, Application Framework, Libraries, Android Runtime* dan *Kernel Linux.* Arsitektur lengkap platform ini dapat dilihat pada Gambar 2.1 berikut:



Gambar 2.9 Arsitektur platform android

2.14.1.1 Aplication

Application merupakan program yang langsung berhubungan dengan *user*. Baik program yang merupakan bawaan dari Android sendiri maupun program yang dibuat oleh *developer* menggunakan bahasa pemrograman java. Contoh program bawaan dari platform Android sendiri adalah *email client*, program SMS, *calendar, maps, web browser, contact* dan sebagainya.

2.14.1.2 Aplication Framework

Lapisan ini berisi sekumpulan API yang dapat digunakan oleh *programmer* maupun *core application* dari Android.Lapisan ini dirancang untuk memudahkan penggunaan komponen dari Android sendiri. Aplikasi manapun dalam Android dapat berbagi fungsi sehingga aplikasi lain dapat memanfaatkannya. Aplikasi pada Android disusun atas beberapa komponen:

1. Sekumpulan *Views*. Digunakan untuk mengatur tampilan pada aplikasi. Contohnya adalah *lists, grids, text box, button*, bahkan *embeddable webbrowser.*
2. *Content providers.* Komponen yang mengatur agar aplikasi dapat mengakses *resources* dari aplikasi lain (seperti *Contacts*), atau berbagi data dengan aplikasi lain.
3. *Resource Manager*. Menyediakan akses ke pada *resource non-code* seperti *localized string*, grafik dan *file layout*.

4. *Notification Manager*. Memungkinkan agar suatu aplikasi dapat menampilkanperingatan yang dapat di kostumasi pada *status bar*.
5. *Activity Manager*. Mengatur siklus aplikasi dan navigasi antar aplikasi yang sedang berjalan.

2.14.1.3 Libraries

Android mendukung beberapa *library* C/C++ yang digunakan pada berbagai komponen Android. Kemampuan ini dapat diakses oleh *developer* melalui Android *application framework*. Beberapa *library* diantaranya adalah:

1. *System* C *library*. Implementasi *library* C standar (libc).
2. *Media Libraries*. Mendukung berbagai format multimedia (termasuk MPEG4,H.264, MP3, AAC, AMR, JPG, PNG).
3. *Surface Manager*. Mengatur akses ke subsistem *display*.
4. LibWebCore. *Engine web browser* modern.
5. SGL. *Engine* grafis 2D.
6. 3D *Library*. Implementasi OpenGL ES 1.0 yang mendukung akselerasi hardware.
7. FreeType. *Rendering* untuk *bitmap* dan *vector font.*
8. SQLite. Basis data relasional yang kecil namun sangat ampuh.

2.14.1.4 Android Runtime

Tiap aplikasi pada Android memiliki proses-nya masing-masing.Tiap aplikasi tersebut memiliki *instans* dari Dalvik *virtual machine* (VM). Dalvik *virtual machine* dirancang agar suatu *device*dapat menjalankan beberapa VM secara efisien. Dalvik VM *mengeksekusi* file dengan format Dalvik *Executable format* (.dex) yang dirancang untuk meminimalkan *memory footprint* . Dalvik VM berbasis *register*, dan dapat menjalankan kelas-kelas yang dikompilasi dengan bahasa pemrograman java dan ditransformasikan menjadi format .dex. Dalvik VM sendiri bergantung pada Kernel Linux untuk fungsi dasarnya, seperti *threading* dan manajemen memori secara *low-level*.

2.14.1.5 *Linux* Kernel

Android menggunakan Kernel Linux versi 2.6 sebagai sistem utama. Fungsi kernel yang digunakan antara lain untuk keamanan, manajemen memori, manajemen proses, manajemen jaringan dan *driver model*. Kernel juga berfungsi sebagai *layer* abstrak antara *hardware* dan lapisan lainnya pada *software stack*.

2.14.2 Anatomi Aplikasi Android

Terdapat empat *building blocks* pada aplikasi Android yaitu:

1. *Activity*

2. *Intent Receiver*
3. *Service*
4. *Content Provider*

Tidak semua aplikasi membutuhkan keempat blok ini, tetapi suatu aplikasi dibuat menggunakan kombinasi beberapa blok ini. Setelah memutuskan blok komponen yang akan digunakan, blok tersebut didaftarkan pada suatu *file* yang disebut dengan AndroidManifest.xml. *File* XML ini digunakan untuk menyatakan komponen apa saja yang dibutuhkan oleh sebuah aplikasi dan kemampuan serta kebutuhan aplikasi tersebut.

2.14.2.1 Activity

Activity paling umum diantara keempat *building block*.*Activity* biasanya suatu tampilan (*screen*) dalam suatu aplikasi. Tiap *activity* diimplementasikan sebagai satu kelas yang meng-*extend* base class *Activity*. Kelas yang dibuat akan menampilkan *user interface* yang terdiri dari beberapa *views* dan akan merespon suatu *event*. Kebanyakan, aplikasi terdiri dari beberapa *screen*. Contohnya aplikasi pengiriman pesan akan memiliki *screen* untuk mendaftarkan pesan yang masuk, menulis pesan dan *screen* lainnya untuk melakukan pengaturan. Tiap *screen* pada contoh ini akan diimplementasi sebagai suatu *activity*. Aplikasi dapat berpindah ke *screen* lain

dengan cara memulai suatu *activity* baru. Pada suatu kasus tertentu, *activity* akan mengembalikan nilai kepada *activity* sebelumnya.

Contohnya *activity* untuk memilih foto akan mengembalikan foto yang dipilih pada *activity* yang memanggilnya. Pada saat *screen* baru terbuka, screen sebelumnya akan dihentikan dan *state* nya akan disimpan pada *history stack*. *User* dapat melakukan navigasi ke *activity* sebelumnya melalui *history* tersebut.*Screen* akan di hapus dari *history stack* apabila sistem memutuskan bahwa *screen* tersebut tidak layak untuk berada pada memori. Android menyimpan *history stack* untuk setiap aplikasi yang dijalankan dari tampilan awal (*home screen*).

2.14.2.2 Intent dan *Intent Filter*

Android menggunakan kelas khusus bernama *intent* untuk berpindah dari *screen* ke *screen* lainnya. Sebuah *intent* mendeskripsikan apa yang ingin aplikasi penuhi. Bagian struktur data dari *intent* yang paling penting adalah *action* dan *data* dari aksi yang dilakukan.Nilai *action* yang biasanya adalah MAIN, VIEW, PICK, EDIT dan sebagainya.Data diekspresikan sebagai URI.Sebagai contoh, untuk melihat informasi suatu *contact* seseorang, kita membuat *intent* dengan action VIEW dan data sebagai URI yang merepresentasikan orang tersebut. Terdapat suatu kelas yang dinamakan IntentFilter.Apabila *intent* adalah suatu permintaan untuk melakukan sesuatu,

IntentFilter adalah deskripsi yang dapat dilakukan oleh *intent activity*.*Activity* yang dapat menampilkan informasi *contact* dari seseorang akan mempublikasikan IntentFilter yang menyatakan bahwa ia dapat menangani *action* VIEW apabila dipakai pada data yang merepresentasikan seseorang. Suatu aplikasi mempublikasikan IntentFilter pada *file* AndroidManifest.xml.

Navigasi dari suatu *screen* ke *screen* lainnya dipenuhi dengan cara me-*resolve intent*. Untuk melakukan navigasi ke depan, *activity* memanggil startActivity(myIntent). Sistem akan mencari IntentFilter untuk semua aplikasi dan memilih *activity* yang IntentFilter-nya yang paling mendekati dengan myIntent. *activity*baru akan mendapat informasi tentang *intent* yang menyebabkan *activity* tersebut dijalankan. Proses pencarian *intent* terjadi sewaktu runtime pada saat startActivity dipanggil. Hal tersebut membawa keuntungan sebagai berikut:

Activity dapat melakukan *reuse functionality* dari komponen lain dengan cara melakukan permintaan dalam bentuk *intent*. *Activity* dapat di *replace*kapan pun dengan *activity* yang ekuivalen.

2.14.2.3 Intent Recivier

IntentReceiver dapat digunakan pada saat kita menginginkan kode kita dijalankan akibat reaksi dari *external event*tertentu.Contohnya, pada saat

telepon berbunyi, atau pada saat ada data pada jaringan, atau pada suatu waktu tertentu.

IntentReceiver tidak menampilkan *user interface*, meskipun ia dapat menggunakan NotificationManager untuk memperingatkan *user* apabila sesuatu terjadi. *IntentReceiver* didaftarkan pada *file* AndroidManifest.xml, tetapi secara manual juga bisa ditambahkan dengan fungsi Context.registerReceiver(). Suatu aplikasi tidak harus berjalan pada *IntentReceiver*nya agar bisa dipanggil, tetapi sistem yang akan menjalankan aplikasi tersebut. Apabila diperlukan, pada saat IntentReceiver mendapat *trigger*, aplikasi juga dapat mengirimkan *intent broadcast* kepada aplikasi lainnya dengan fungsi Context.broadcastIntent().

2.14.2.4 Service

Service adalah sebuah kode yang hidup terus-menerus dan berjalan tanpa memiliki tampilan. Contoh yang sederhana adalah suatu aplikasi *media player* yang sedang memainkan lagu dari *play list.* Pada aplikasi *media player*, kemungkinan terdapat satu atau lebih activity yang memperbolehkan *user* memilih lagu dan memainkannya. Tetapi musik yang sedang diputar tidak boleh ditangani oleh activity karena *user* akan menginginkan musik tetap diputar meskipun ia berpindah ke *screen* baru.

Pada kasus ini activity *media player* dapat menjalankan *service* menggunakan fungsi Context.startService() untuk menjalankan *service* di *background* yang memutar musik. Sistem akan tetap memutar musik sampai selesai. Selain itu *activity* dapat terhubung dengan *service* menggunakan fungsi Context.bindService(). Pada saat terhubung, aplikasi dapat berkomunikasi melalui *interface* yang diberikan oleh *service* yang bersangkutan. Pada contoh *media player*, *interface service* akan memberikan fungsi *pause, rewind* dan sebagainya.

2.14.2.5 Content Provider

Content provider digunakan untuk pertukaran data antara aplikasi. Data yang dipertukarkan bersifat *shared*. *Content provider* adalah sebuah kelas yang berisi *method* standar yang memungkinkan aplikasi lain dapat menyimpan dan mengambil suatu data yang ditangani oleh *content provider*.

2.14.3 Siklus Hidup Aplikasi Android

Pada banyak kasus, tiap aplikasi pada Android masing-masing memiliki Linux proses. Proses ini diciptakan untuk aplikasi tersebut pada saat kode program tersebut akan dieksekusi, akan terus berjalan sampai tidak dibutuhkan lagi dan sistem memerlukan memori untuk aplikasi lain.

Konsep dasar program Android adalah suatu proses pada aplikasi tidak langsung di atur oleh aplikasi tersebut. Proses tersebut ditentukan oleh sistem melalui kombinasi: informasi aplikasi bagi sistem, seberapa pentingnya aplikasi tersebut bagi *user* dan ketersediaan memori. Contoh kasus *life-cycle bug* adalah *IntentReceiver* yang menciptakan *thread* pada saat menerima *intent* pada *method* onReceiveIntent() dan selesai dari fungsi tersebut. Setelah fungsi tersebut selesai, sistem menganggap bahwa *IntentReceiver* tersebut menjadi tidak aktif sehingga ia memiliki proses yang tidak dibutuhkan (kecuali terdapat komponen aplikasi lain di dalamnya).

Oleh karena itu, sistem dapat menghapus proses tersebut kapanpun apabila dibutuhkan. Hal tersebut akan menghapus semua *thread* yang berjalan pada proses tersebut. Solusi masalah ini adalah memakai *Service* dari IntentReceiver, sehingga sistem mengetahui bahwa ada suatu yang aktif dalam proses tersebut.

Sebagai *programmer*, kita harus mengetahui bagaimana cara yang tepat untuk merancang aplikasi.

Kesalahan pada perancangan akan membuat suatu aplikasi akan dihentikan oleh sistem pada saat yang tidak diinginkan. Untuk menentukan proses mana yang akan dihapus pada situasi *low memory*, Android memiliki "*importancehierarchy*" berdasarkan komponen dan *state* yang sedang berjalan. Urutan berdasarkan kepentingan tersebut adalah:

1. *Foreground process*. Adalah suatu proses yang dibutuhkan untuk berinteraksi dengan *user* saat ini. Berbagai komponen suatu aplikasi dapat mengakibatkan

proses di dalamnya menjadi *foreground proses*. Suatu proses dikatakan *foreground* apabila memenuhi kondisi sebagai berikut:

- Menjalankan Activity pada *top screen* yang sedang berinteraksi dengan *user* (setelah *method* onResume() dipanggil).
- Mempunyai IntentReceiver yang sedang berjalan (method IntentReceiver.onReceiveIntent() sedang dieksekusi).
- Memilik service yang fungsi *callback*-nya (Service.onCreate(), Service.onStart() atau Service.onDestroy()) sedang dieksekusi.

Hanya akan terdapat beberapa proses seperti itu dalam sistem, proses ini hanya akan dihapus apabila sistem tidak memiliki pilihan lain. Contohnya apabila memori tinggal sedikit sehingga proses *foreground* tidak bisa lagi berjalan. Halini diperlukan untuk menjaga *user interface* tetap interaktif dengan *user*.

2. *Visible process*. Adalah proses yang memegang Activity yang dapat dilihat*user* pada layar tetapi tidak *foreground* (pada saat *method* onPause()dipanggil). Hal ini dapat terjadi, sebagai contoh *activity foreground* muncul dengan dialog sehingga *activity* sebelumnya terlihat di belakangnya. Proses ini sangat penting dan tidak akan di hapus kecuali untuk menjaga aplikasi *foreground* tetap berjalan.
3. *Service process*. Adalah proses yang memegang Service yang dimulai dengan method StartService().Meskipun proses ini secara langsung tidak dapat dilihat oleh *user*, biasanya proses ini menjalankan perintah dari *user* (seperti memutar lagu, men-*download* data dari jaringan dan lain-lain). Sistem akan tetap

membiarkan proses ini berjalan selama proses *foreground* dan *visible* masih bisa berjalan.

4. *Background process*. Adalah proses yang memegang Activity yang saat ini tidak dapat dilihat oleh *user* (method onStop() sedang dipanggil). Proses ini tidak secara langsung mempengaruhi *user*. Apabila *activity life cycle* telah dipenuhi dengan benar, sistem dapat menghapus proses ini sewaktu-waktu untuk memberikan memori tambahan. Biasanya banyak ditemukan proses semacam ini, sehingga proses tersebut disimpan dalam LRU (*Least RecentlyUsed*) untuk menjaga agar proses yang baru saja dilihat oleh *user* akan dihapus paling akhir pada kondisi *low memory*.
5. *Empty Process*. Adalah proses yang tidak memegang suatu komponen yangsedang aktif. Alasan mengapa membiarkan proses semacam ini adalah untukmelakukan *cache* sehingga proses *startup* menjadi lebih cepat apabila aplikasitersebut akan dijalankan kembali. Sistem seringkali menghapus prosessemacam ini untuk mencapai keseimbangan dari *system resource* antaracached processes dan kernel caches.

2.15 JavaScript

JavaScript adalah bahasa skrip yang populer di internet dan dapat bekerja di sebagian besar penjelajah web populer seperti Internet Explorer (IE), Mozilla

Firefox, Netscape dan Opera. Kode JavaScript dapat disisipkan dalam halaman web Menggunaka tag SCRIPT.

JavaScript pertama kali dikembangkan oleh Brendan Eich dari Netscape dibawah nama Mocha, yang nantinya namanya diganti menjadi LiveScript, dan akhirnya menjadi JavaScript.

Navigator sebelumnya telah mendukung Java untuk lebih bisa dimanfaatkan para programmer yang non-Java. Maka dikembangkanlah bahasa pemrograman bernama LiveScript untuk mengakomodasi hal tersebut. Bahasa pemrograman inilah yang akhirnya berkembang dan diberi nama JavaScript, walaupun tidak ada hubungan bahasa antara Java dengan JavaScript.

JavaScript bisa digunakan untuk banyak tujuan, misalnya untuk membuat efek rollover baik di gambar maupun teks, dan yang penting juga adalah untuk membuat AJAX. JavaScript adalah bahasa yang digunakan untuk AJAX.

JavaScript merupakan bahasa pemrograman web yang berbentuk kumpulan skrip dan berfungsi untuk memberikan tampilan yang tampak lebih interaktif pada dokumen web. Dengan kata lain bahasa ini adalah bahasa pemrograman untuk memberikan kemampuan tambahan kedalam bahasa pemrograman HTML (Hypertext Markup Language) dengan pengeksekusian perintah pada sisi client dan bukan pada sisi server seperti PHP dan bahasa pemrograman web lain yang bersifat server side scripting atau bahasa pemrograman yang memerlukan server sebagai penerjemah script sebelum ditampilkan pada client.

Pada hakekatnya bahasa pemrograman javascript berisi skrip yang penulisanya diselipkan disebuah dokumen html dan bersifat *client side scripting*, sehingga javascript tidak memerlukan sebuah webserver sebagai penerjemah untuk mengeksekusi skrip skripnya.

Kode JavaScript biasanya dituliskan dalam bentuk fungsi yang ditaruh di tag <head> yang dibuka dengan tag <script type="teks/javascript">.

```
script type="teks/javascript">
alert("Halo Dunia!");
</script>
```

Kode JavaScript juga bisa diletakkan di file tersendiri yang berekstensi .js (singkatan dari JavaScript). Untuk memanggil kode JavaScript yang terdapat di file sendiri, di bagian awal <head> harus ditentukan dahulu nama file .js yang dimaksud menggunakan contoh kode seperti berikut:

```
<script type="teks/javascript" src="alamat.js">
</script>
```

Script ini akan dieksekusi ketika dipanggil (biasanya berbentuk function) atau dipanggil berdasarkan trigger pada event tertentu. Peletakkan script di head akan menjamin skript di-load terlebih dahulu sebelum digunakan (dipanggil).

```
<html>
```

```
<head>
<script type="teks/javascript">...
</script>
</head>
</html>
```

Script ini dieksekusi ketika halaman di-load sampai di bagian <body>.Ketika menempatkan script pada bagian <body> berarti antara isi dan JavaScript dijadikan satu bagian.

```
<html>
<head>
</head>
<body>
<script type="teks/javascript">
...
</script>
</body>
</html>
```

Jumlah JavaScript di <head> dan <body> yang ditempatkan pada dokumen tidak terbatas.

Terkadang ada yang menginginkan menjalankan JavaScript yang sama dalam beberapa kali pada halaman yang berbeda, tetapi tidak mau disibukkan jika

harus menulis ulang script yang diinginkan di setiap halaman. Maka JavaScript dapat ditulis di file secara eksternal. Jadi, antara dokumen HTML dan JavaScript dipisahkan, kemudian berkas tersebut dipanggil dari dokument HTML. Berkas JavaScript tersebut disimpan dengan ekstensi .js.

JavaScript : js/xxx.js document.write("pesan ini tampil ketika halaman diload");

Untuk menggunakan eksternal JavaScript (.js) dipakai atribut "src" pada tag <script> pada halaman HTML-nya.

```
<html>
<head>
</head>
<body>
<script src="xxx.js">
</script>
<p>Script di atas berada di berkas "xx.js" (eksternal) </p>
</body>
</html>
```