BAB II

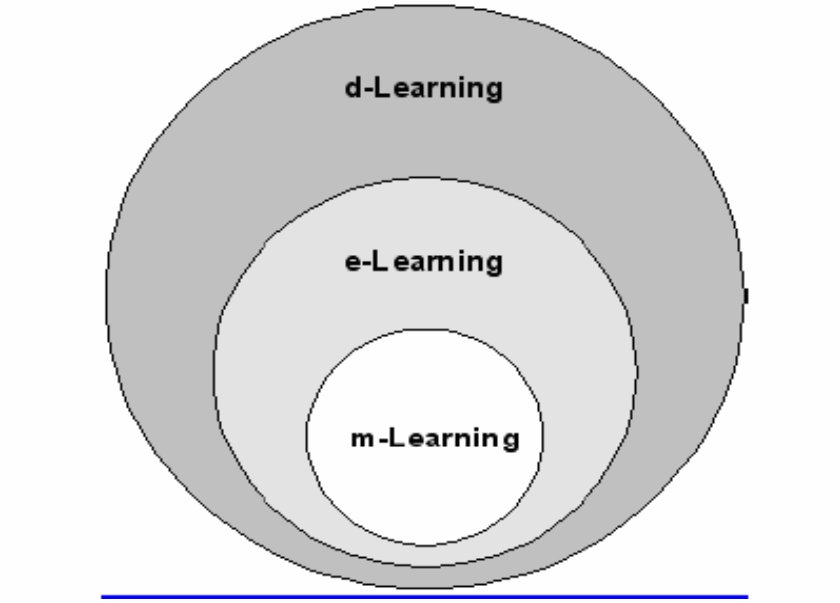
LANDASAN TEORI

Dari perumusan masalah pada sub bab 1.2, dalam pemecahan masalahnya diperlukan beberapa teori pendukung seperti :

2.1 Mobile Learning

Mobile learning atau *M-learning* adalah sebuah pembelajaran jarak jauh dengan menggunakan *portable technologies*. Dengan kata lain mobile learning dapat mengurangi batasan tempat belajar. Beberapa contoh *personal technologies*, yaitu: *Personal Digital Asistant* (PDA), *smart phone*, dan lain-lain. Sedangkan jenis komunikasi yang didukung *mobile learning* adalah GSM, GPRS, WI-FI. (Wikipedia, 2008)

Istilah mobile learning (m-Learning) mengacu kepada penggunaan perangkat/divais teknologi informasi (TI) genggam dan bergerak, seperti PDA, telepon genggam, laptop dan tablet PC, dalam pengajaran dan pembelajaran. M-Learning merupakan bagian dari electronic learning (e-Learning) sehingga, dengan sendirinya, juga merupakan bagian dari distance learning (d-Learning). (Tamim, 2007)



Gambar 2.1. Skema dari bentuk m-Learning

Beberapa kemampuan penting yang harus disediakan oleh perangkat pembelajaran m-Learning adalah adanya kemampuan untuk terkoneksi ke peralatan lain (terutama komputer), kemampuan menyajikan informasi pembelajaran dan kemampuan untuk merealisasikan komunikasi bilateral antara pengajar dan pembelajar. M-Learning adalah pembelajaran yang unik karena pembelajar dapat mengakses materi pembelajaran, arahan dan aplikasi yang berkaitan dengan pembelajaran, kapan-pun dan dimana-pun. Hal ini akan meningkatkan perhatian pada materi pembelajaran, membuat pembelajaran menjadi pervasif, dan dapat mendorong motivasi pembelajar kepada pembelajaran sepanjang hayat (lifelong learning). Selain itu, dibandingkan pembelajaran konvensional, m-Learning memungkinkan adanya lebih banyak kesempatan untuk kolaborasi secara ad hoc dan berinteraksi secara informal diantara pembelajar.

2.2 Shareable Content Object Reference Model (SCORM).

2.2.1 Pendahuluan SCORM.

SCORM dikembangkan pertama kali oleh Advanced Distributed Learning (ADL) pada tahun 1997, sebuah lembaga yang dibentuk oleh United State Departement of Defence(DoD), Department Pertahanan Amerika Serikat.

Pada umumnya SCORM tidak selalu memberikan *E-Learning* sesuatu yang baru, SCORM hanya menyatukan *E-Learning* kepada standard, petunjuk dan spesifikasi tertentu tentang bagaimana suatu *E-Learning* bekerja. Jadi, SCORM adalah kumpulan standard, petunjuk dan spesifikasi untuk membangun *web-based E-Learning*. SCORM membentuk komunikasi antara *client side content* dengan *host* sistem atau yang biasa disebut *run-time environment*. SCORM juga didefinisikan tentang bagaimana suatu kontent dibentuk kedalam file .zip.

Terdapat tiga (3) acuan utama untuk SCORM, yaitu pertama sistem harus dengan mudah memberikan petunjuk yang dapat dimengerti dan diimplementasikan oleh pengembang *E-Learning*. Kedua, sistem harus dengan mudah dipakai dan dimengerti, dan digunakan oleh sebanyak mungkin pengguna. Ketiga, sistem harus mengizinkan perubahan model yang dikehendaki oleh pengembang.

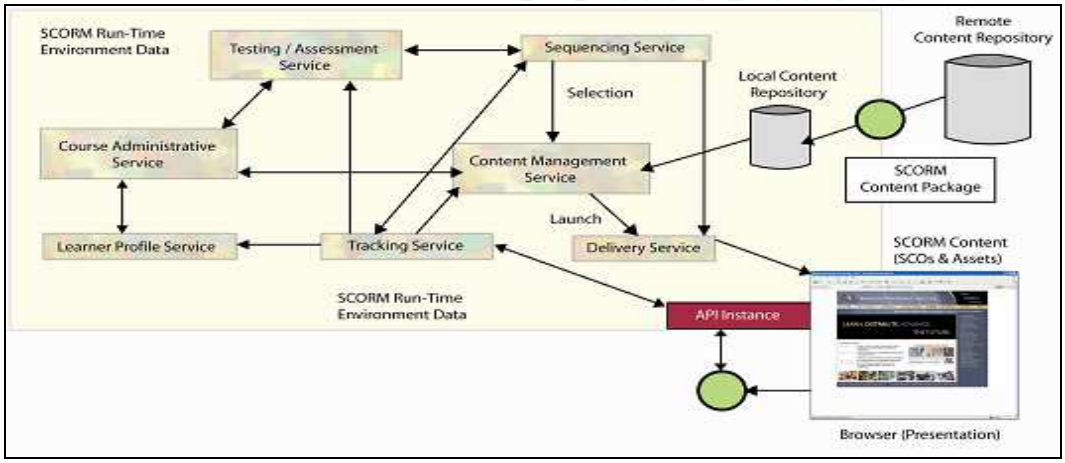
Untuk membantu mendapatkan suatu model SCORM, terdapat enam (6) persyaratan *starting point* untuk SCORM, keenam persyaratan ini dikenal dengan *the ilities*, keenam persyaratan tersebut adalah.

1. Accessibility : Membuat isi pembelajaran dapat dengan mudah diakses dari lokasi yang berbeda.

2. Adaptability : Memungkinkan aplikasi dapat diubah sesuai dengan kebutuhan pengembang atau organisasi.
3. Affordability : Mengurangi waktu dan biaya untuk pengembangan dan pendistribusian.
4. Durability : Desain pembelajaran tidak terlalu banyak terjadi perubahan desain, dan pemrograman dalam rangka penyesuaian dengan teknologi.
5. Interoperability : Desain pembelajaran mendukung multiplatform.
6. Reusability : Bagian dari desain pembelajaran dapat dengan mudah untuk digunakan dalam aplikasi lain.

SCORM juga mengalami perubahan sejak diciptakan pertama kali, saat ini terdapat tiga (3) kali perubahan dari yang pertama, SCORM 1.1, SCORM 1.2, dan SCORM 2004.

Jika kaitkan antara SCORM dengan Learning Management System (LMS), maka SCORM mendifinisikan LMS lebih sempit, yaitu SCORM terfokus pada LMS sejauh mana LMS berhubungan untuk mengirimkan isi dari *web-based learning* dan interaksinya dengan pemakai LMS.



Gambar 2.2 : Kaitan SCORM dengan LMS

(Sumber : SCORM 2004 : Overview, Section 1 hal 7)

Terdapat banyak LMS yang juga menyediakan mekanisme untuk mendaftarkan pengguna, untuk mengikuti mata pelajaran secara langsung berupa mengambil dokumen yang sesuai dengan *course* pengguna tersebut. Dokumen tersebut dapat berupa file text (.pdf, .html) , slide presentasi (.ppt), dan juga file multimedia (.flv).

2.2.2 SCORM 2004.

Seperti yang disebutkan sebelumnya, SCORM sebenarnya adalah kumpulan dari dokumen yang saling berhubungan. Seperti ilustrasi dibawah ini.

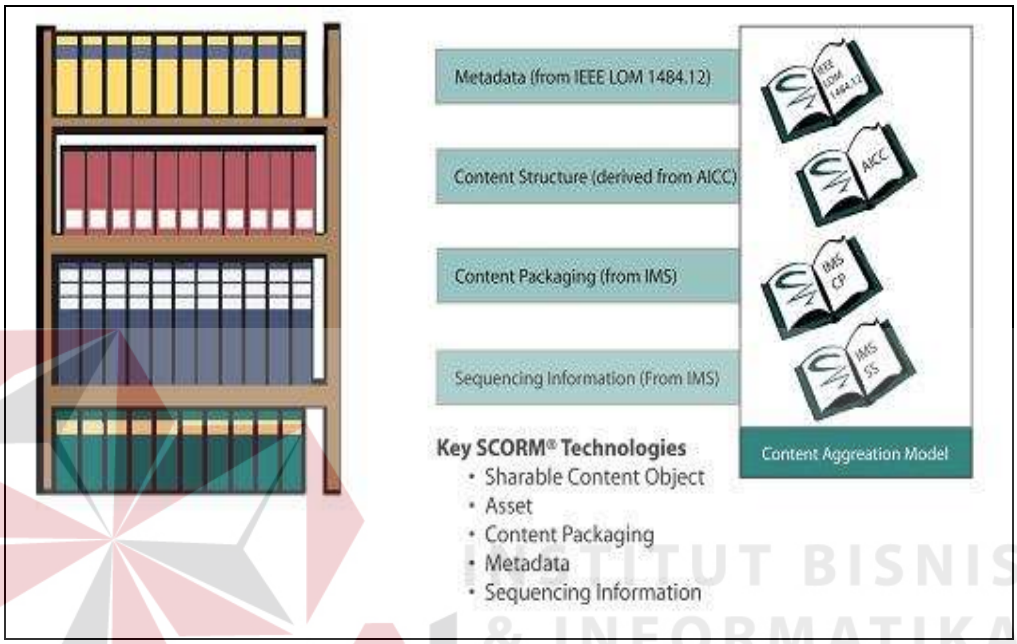
Gambar 2.3 : SCORM 2004

(Sumber : http://www.learningcircuits.org/NR/rdonlyres/52A06F64-40EF-475A-A9EF-5B65BA774B8B/6976/SCORM.jpg)

Dalam SCORM 2004 terdapat dokumen inti yang menyusun SCORM, dokumen inti tersebut adalah

1. Content Aggregation Model (CAM).

SCORM sering juga dideskripsikan sebagai kumpulan buku pada rak buku, CAM sebagai sekumpulan buku pada baris bawah pada rak buku, seperti ilustrasi berikut ini.



Gambar 2.4: Ilustrasi CAM

(Sumber : SCORM 2004 : Content Aggregation Model, Section 1 hal 3)

SCORM CAM merepresentasikan klasifikasi untuk pengembang dan pemakai untuk mengumpulkan sumber daya pembelajaran, sehingga dapat menyajikan pelajaran (*content*) sesuai dengan keinginan pengguna atau pengembang.

Pada dasarnya SCORM CAM terdiri dari 4 susunan utama yaitu :

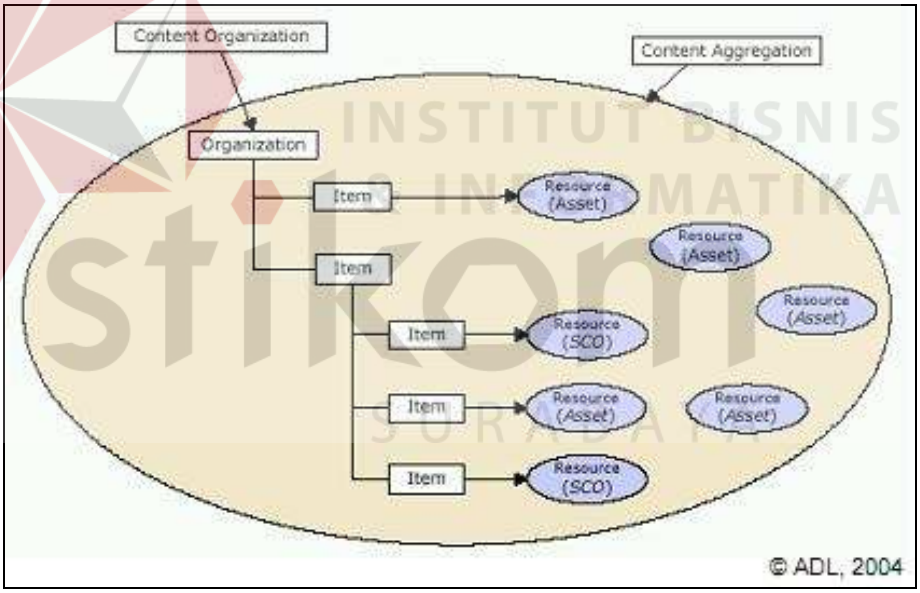
a) Content Model : Mendefinisikan SCORM komponen yang menyusun sistem pembelajaran dan menjelaskan bagaimana komponen tersebut saling berhubungan. Komponen dari *content model* adalah :

(1) Asset : Merupakan blok dasar pembentuk *learning resource*. Asset adalah representasi dari data secara elektronik seperti text, gambar, suara, dll.

(2) Shareable Content Object (SCO) : Merupakan koleksi dari satu atau lebih asset yang menggambarkan *learning resource* yang dapat di jalankan.

(3) Content Organization : Gambaran atau mapping yang mendefinisikan urutan penggunaan *content* melalui aktifitas yang terstruktur.

Berikut ini adalah ilustrasi dari Content Model.

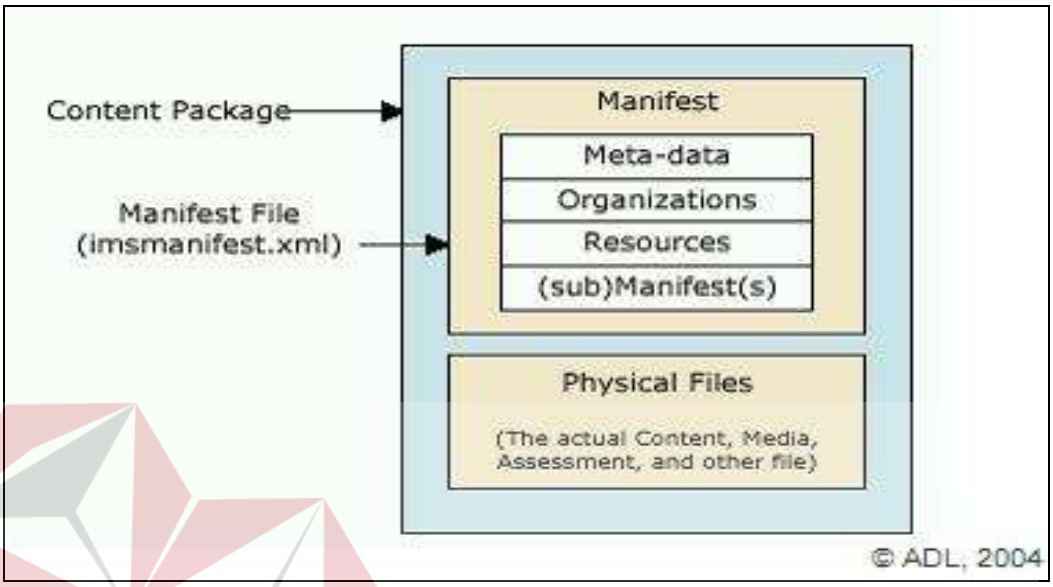


Gambar 2.5 : Ilustrasi Content Model

b) Content Packaging : Berisi semua koleksi dari semua file yang diperlukan untuk menjalankan mata pelajaran pada format standard.

c) Metadata : suatu mekanisme untuk mendeskripsikan instance secara spesifik dari komponen dalam *content model*.

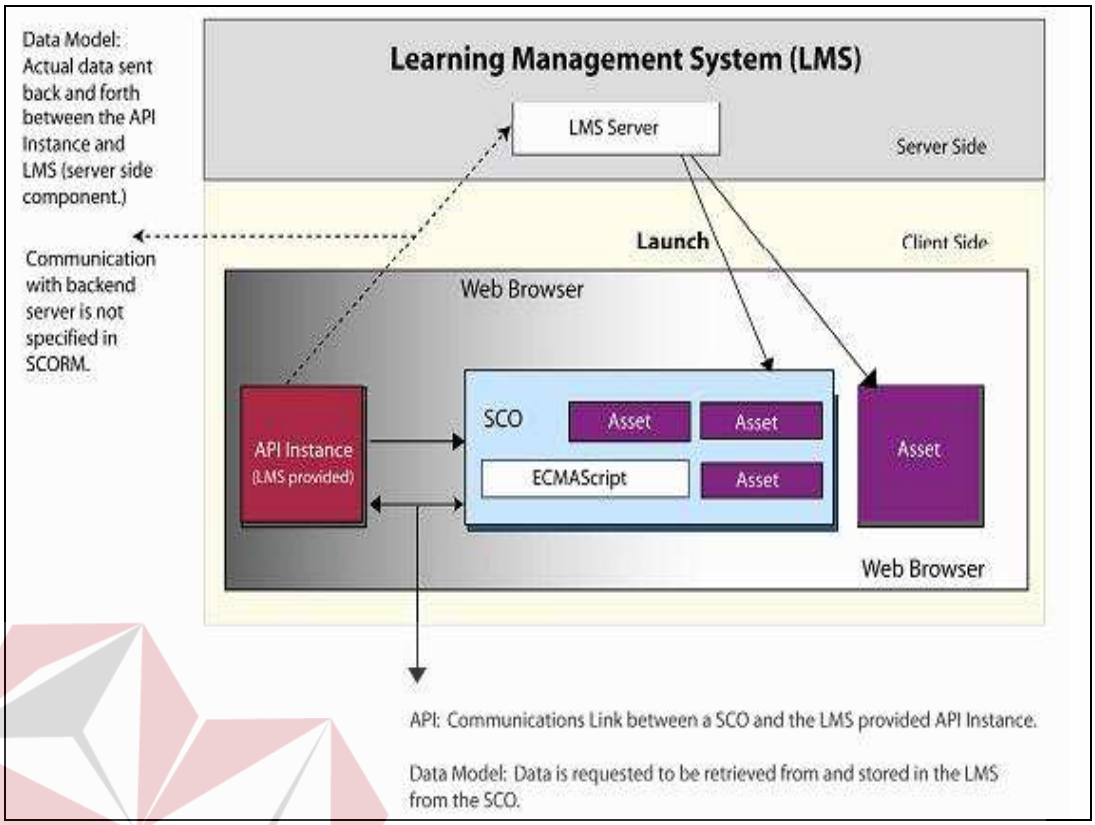
d) Manifest : Juga dikenal sebagai IMS Manifest, berupa dokumen XML yang mendeskripsikan semua *content*. Berikut ini adalah ilustrasi hubungan antara content packaging, metadata, dan manifest.



Gambar 2.6 : Ilustrasi hubungan content packaging, metadata dan manifest

2. Run-Time Environtment.

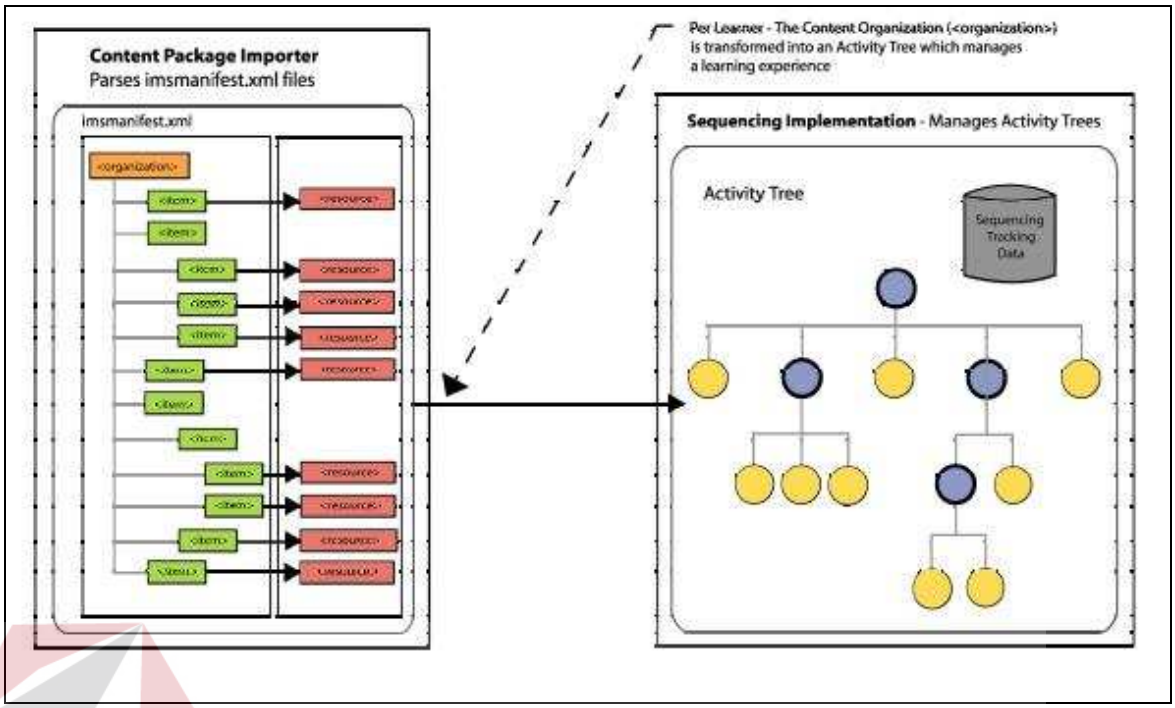
Dokumen paling penting kedua dalam SCORM 2004 adalah SCORM Run-Time Environtment atau biasa disingkat dengan RTE. Dokumen ini menjelaskan bagaimana LMS menjalankan SCO dan berkomunikasi dengan SCO. Berikut ini adalah ilustrasi yang mengambarkan hubungan LMS dan SCO.



Gambar 2.7 : Hubungan LMS, SCO dan Asset.

3. Sequencing and Navigation.(SN)

Sequence and Navigation adalah dokumen terbaru yang ditambahkan dalam SCORM 2004. SN menjelaskan aktifitas yang akan dilakukan oleh pengguna sesuai dengan langkah-langkah yang ditentukan secara konsisten. Konsep utama dalam SN adalah *activity tree*, yang diterjemahkan dari IMS manifest. *Activity tree* sendiri digunakan menjelaskan hirarki pembelajaran dari proses pembelajaran, berikut ini adalah ilustrasi hubungan antara *activity tree* dengan IMS.



Gambar 2.8 : hubungan antara *IMS manifest* dengan *activity tree*

2.3. J2ME

J2ME merupakan subset dari *Java 2 Standart Edition* (*J2SE*) yang masing-masing merupakan bagian dari *Java 2 Platform*. J2SE menyediakan *runtime environment*, sedangkan J2ME menyediakan berbagai kebutuhan *Application Programming Interface* (*API*) untuk semua aplikasi *mobile*.

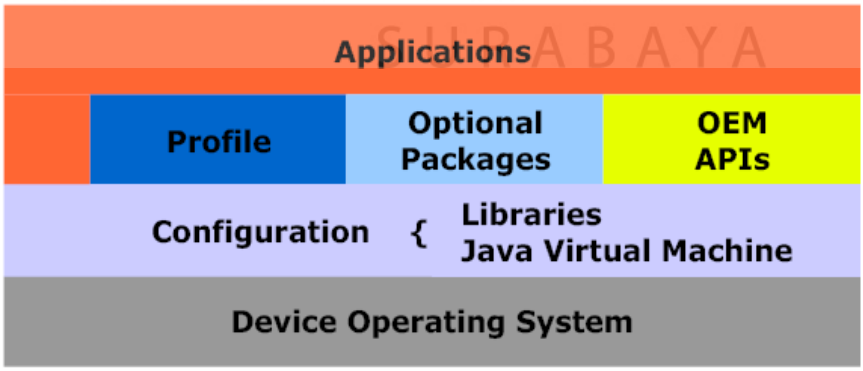
Java sudah menjadi bahasa pemrograman berorientasi objek yang universal. Karena alasan utama inilah aplikasi Kamus Dinamis ini menggunakan teknologi J2ME. Aplikasi yang sekali dibuat pada *J2ME specification* dapat digunakan diberbagai perangkat yang mendukung teknologi Java yang saat ini mencapai jutaan perangkat, tanpa mengeksploitasi kemampuan dasar dari perangkat yang bersangkutan.

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

Dengan perkembangbiakan perangkat mobile konsumer dari telepon, PDA, kotak permainan ke peralatan rumah, Java menyedikan suatu lingkungan yang portable untuk mengembangkan dan menjalankan aplikasi pada perangkat ini.

Program J2ME, seperti semua program JAVA adalah diterjemahkan oleh VM. Program-program tersebut dikompile ke dalam bytecode dan diterjemahkan dengan Java Virtual Machine (JVM). Ini berarti bahwa program-program tersebut tidak berhubungan langsung dengan perangkat. J2ME menyediakan suatu interface yang sesuai dengan perangkat. Aplikasi-aplikasi tersebut tidak harus dikompile ulang supaya mampu dijalankan pada mesin yang berbeda.

Inti dariJ2ME terletak pada configuration dan profile-profile. Suatu configuration menggambarkan lingkungan runtime dasar dari suatu sistem J2ME. Ia menggambarkan core library, virtual machine, fitur keamanan dan jaringan.



Gambar 2.9. Arsitektur dari J2ME

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

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

2.4. Unified Modelling Language (UML)

Sejarah UML menurut Boogs (2002), UML adalah buah pikiran dari Grady Booch, James Rumbaugh, dan Ivar Jacobson. Mereka bekerja dalam organisasi yang terpisah antara tahun 80-an sampai dengan awal 90-an. Mereka merencanakan sebuah metodelogi untuk analis dan design yang berorientasi obyek. Pada pertengahan 90-an mereka meminjam ide-ide dari yang lainnya, sehingga mereka merencanakan menyelesaikan pekerjaan mereka secara bersama-sama. Menurut Sholiq (2005), *Unified Modeling Language* (UML) menyediakan notasi yang cukup tangguh, untuk membangun sistem dari tahap analisa sampai ke tahap perancangan. UML adalah notasi pemodelan standar industri untuk sistem berorientasi obyek, dan adalah juga sebagai platform untuk mempercepat proses pengembangan aplikasi.

UML merupakan sistem arsitektur yang bekerja dalam *Object-Oriented Analysis and Design* (OOAD) dengan satu bahasa yang konsisten untuk menentukan, visualisasi, mengkonstruksi, dan mendokumentasikan *artifact* (sepotong informasi yang digunakan atau dihasilkan dalam suatu proses rekayasa perangkat lunak, dapat berupa model, deskripsi, atau perangkat lunak) yang terdapat dalam sistem perangkat lunak. UML merupakan bahasa pemodelan yang paling sukses dari tiga metode Object Oriented yang telah ada sebelumnya, yaitu

Booch, *Object Modelling Technique* (OMT), dan *Object-Oriented Software Engineering* (OOSE).

Keutamaan dari UML adalah diagram-diagram yang ada pada UML ditambah dengan kemampuan dokumentasi. *Data flow diagram* dan tipe diagram lain yang tidak terdapat dalam UML tidak termasuk dalam paradigma object-oriented. Activity diagram dan collaboration diagram yang terdapat dalam UML menggantikan data flow diagram. Activity diagram juga sangat bermanfaat untuk membuat *workflow*

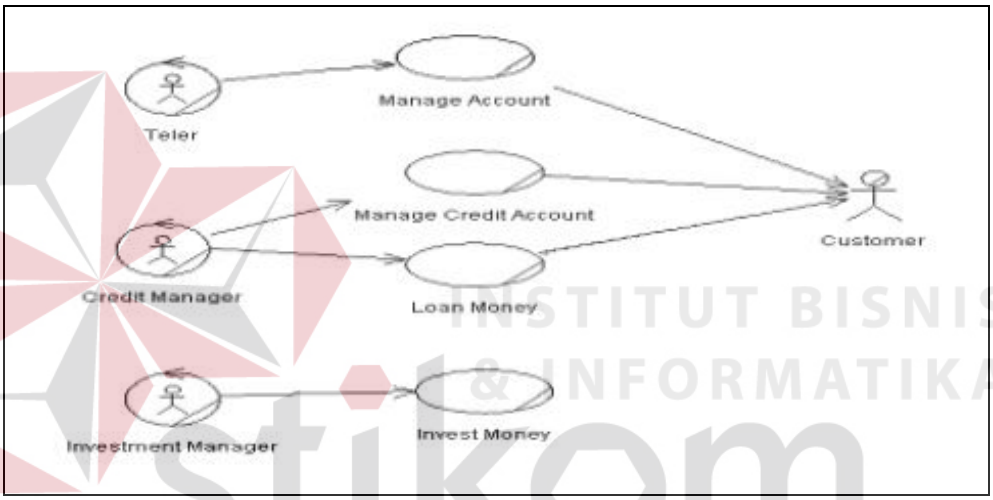
UML merupakan sistem arsitektur yang bekerja dalam *Object-Oriented Analysis and Design* (OOAD) dengan satu bahasa yang konsisten untuk menentukan, visualisasi, mengkonstruksi, dan mendokumentasikan *artifact* (sepotong informasi yang digunakan atau dihasilkan dalam suatu proses rekayasa perangkat lunak, dapat berupa model, deskripsi, atau perangkat lunak) yang terdapat dalam sistem perangkat lunak.

UML terdiri atas sejumlah elemen-elemen grafik yang mengkombinasikan ke dalam bentuk diagram. Tujuan dari diagram-diagram ini adalah untuk menghasilkan multiple view dari sistem, dan kumpulan dari view disebut model. Model UML dari suatu sistem suatu saat seperti sebuah model skala dari bangunan. Penting untuk diperhatikan bahwa model UML menjelaskan apa yang diajukan sistem untuk dikerjakan. Bukan bagaimana cara mengimplementasikannya.

Untuk membuat suatu model, UML memiliki diagram grafis sebagai berikut :

1. Diagram Use Case Bisnis (*Business Use Case Diagram*)

   Menurut Boggs dan Boggs (2002:13), *business use case diagram* digunakan untuk menggambarkan apa yang dilakukan oleh organisasi secara keseluruhan. Diagram ini menjawab pertanyaan "apa yang bisnis lakukan dan mengapa harus membangun sistem". Diagram ini digunakan selama pemodelan aktifitas bisnis berlangsung, dan mengatur kontek sistem untuk membentuk pondasi dalam pembentukan use case.

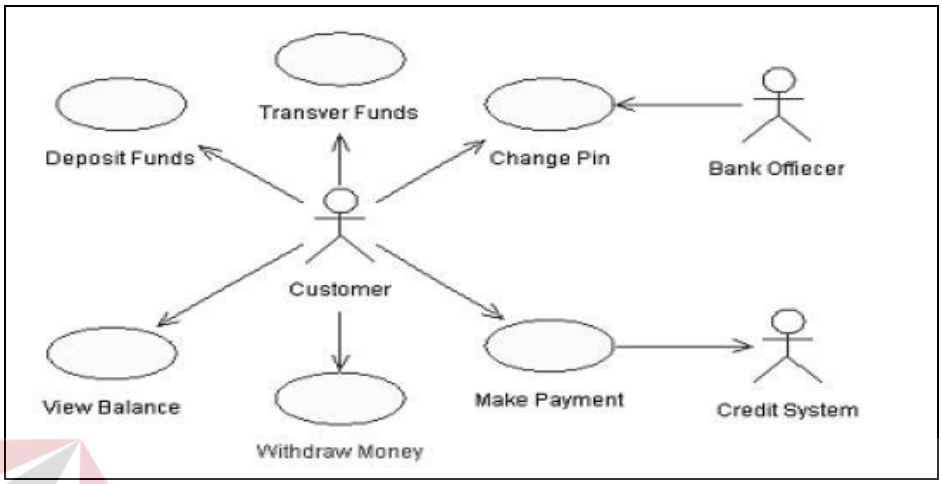


Gambar 2.10 Business Use Case Diagram

2. Diagram Use Case (*Use Case Diagram*)

   Menurut Boggs dan Boggs (2002:13), *use case diagram* menjelaskan tingkah laku sistem yaitu apa yang dilakukan oleh sistem. Diagram ini berisi interaksi antara aktor dengan *use cas*. Dimana aktor dapat berupa orang, peralatan, atau sistem lain yang berinteraksi dengan sistem yang sedang dibangun. Use case menggambarkan fungsionalitas sistem atau persyaratan-persyaratan yang harus dipenuhi sistem dari pandangan pemakai. Jika diagram use case bisnis tidak memperhatikan apakah proses dilakukan secara otomatis

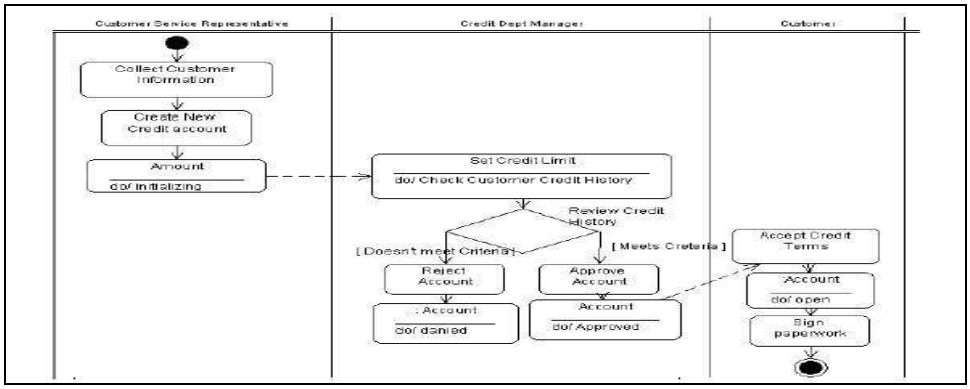
terkomputerisasi, maka diagram use case berfokus hanya pada proses otomatisasi saja.



Gambar 2.11 Use Case Diagram

3. Diagram Aktifitas (*Activity Diagram*)

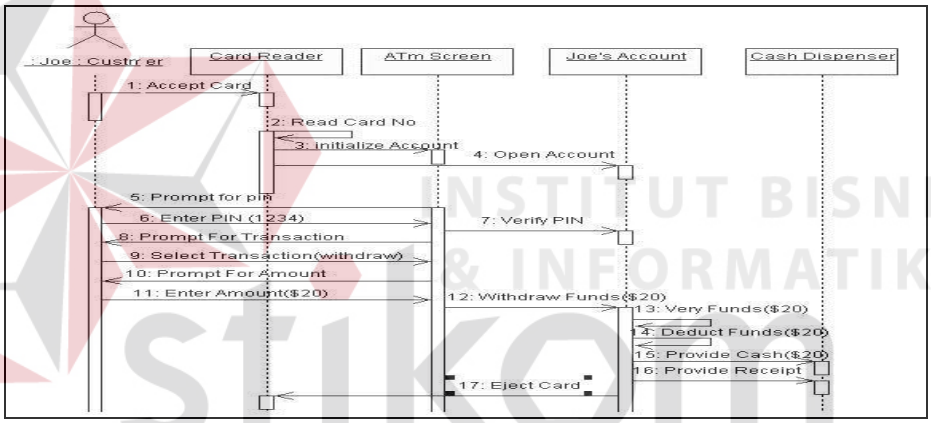
Menurut Boggs dan Boggs (2002:13), diagram aktivitas menggambarkan aliran fungsionalitas sistem. Pada tahap pemodelan bisnis, aktivitas dapat digunakan untuk menunjukkan aliran kerja bisnis (*business workflow*). Dapat juga digunakan untuk menggambarkan aliran kejadian (*flow of event*) dalam use case. Diagram aktifitas tampak seperti pada Gambar 2.12.



Gambar 2.12 Activity Diagram

4. Diagram Sekuensial (*Sequence Diagram*)

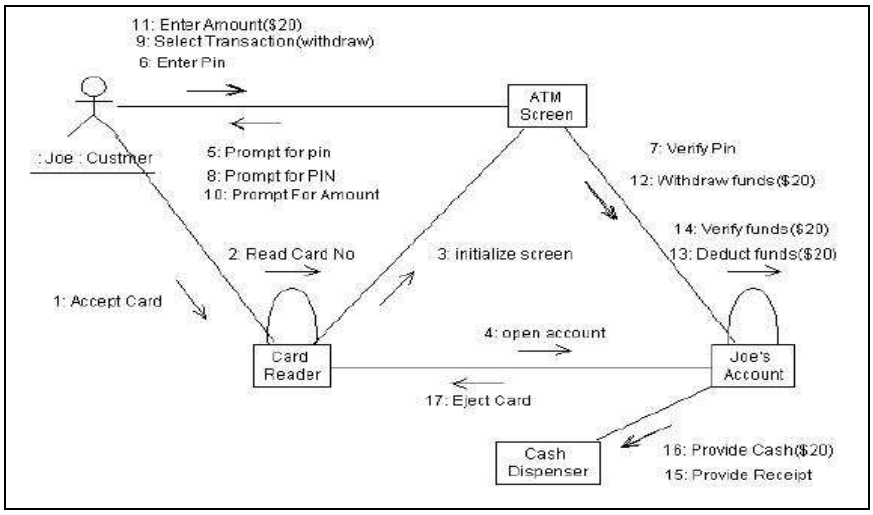
Menurut Boggs dan Boggs (2002:13), diagram sekuensial (*sequence diagram*) digunakan untuk menunjukkan aliran fungsionalitas dalam use case. Misalkan, dalam use case menarik uang, mempunyai beberapa kemungkinan, seperti penarikan uang secara normal, percobaan penarikan dana tanpa dana yang tersedia, percobaan penarikan dengan penggunaan PIN yang salah, dan lainnya. Diagram sekuensial penarikan 20 dollar (tanpa adanya kesalahan seperti no PIN atau penghitungan). Diagram ini menunjukkan aliran proses dalam use case menarik uang. Diagram sekuensial tampak pada Gambar 2.13.



Gambar 2.13 Sequence Diagram

5. Diagram Kolaborasi (*Collaboration Diagram*)

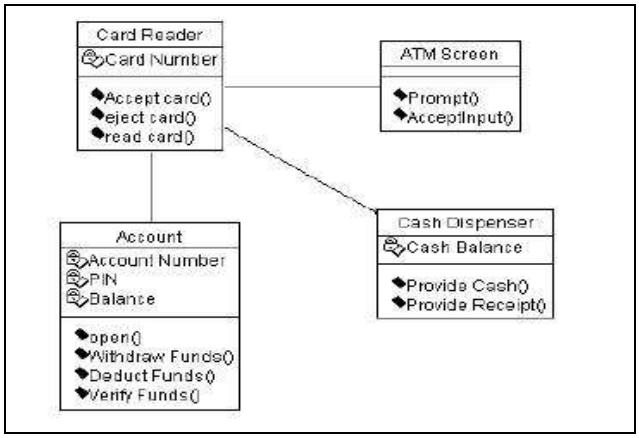
Menurut Boggs dan Boggs (2002:17), menggambarkan informasi seperti *sequence diagram* yaitu elemen-elemen yang bekerja bersama-sama dalam suatu sistem untuk mencapai tujuan. Pada diagram sekuensial keseluruhan interaksi berdasarkan urutan waktu, tapi pada diagram kolaborasi ini ditunjukkan interaksi antara obyek dan aktor tanpa keterangan waktu. Diagram kolaborasi menggambarkan hal yang sama dengan diagram sekuensial seperti yang ditunjukkan pada Gambar 2.14.



Gambar 2.14 Diagram Kolaborasi

6. Diagram Kelas (*Class Diagram*)

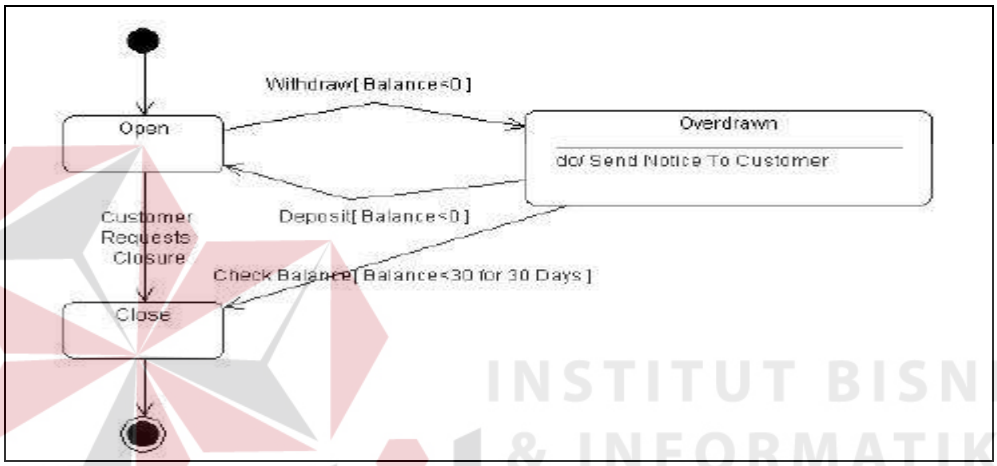
Menurut Boggs dan Boggs (2002:17), *class diagram* menunjukan interaksi antar kelas dalam sistem. Misalkan, nomor account milik Joe adalah sebuah obyek dari kelas Account. Kelas mengandung informasi dan tingkah laku (behavior) yang berkaitan dengan informasi tersebut. Kelas account mengandung nomor PIN pengguna dan tingkah laku untuk mengecek PIN tersebut. Sebuah kelas pada diagram kelas dibuat untuk setiap tipe obyek pada diagram sekuensial atau diagram kolaborasi.



Gambar 2.15 Class Diagram

7. Diagram Keadaan (*statechart diagram*)

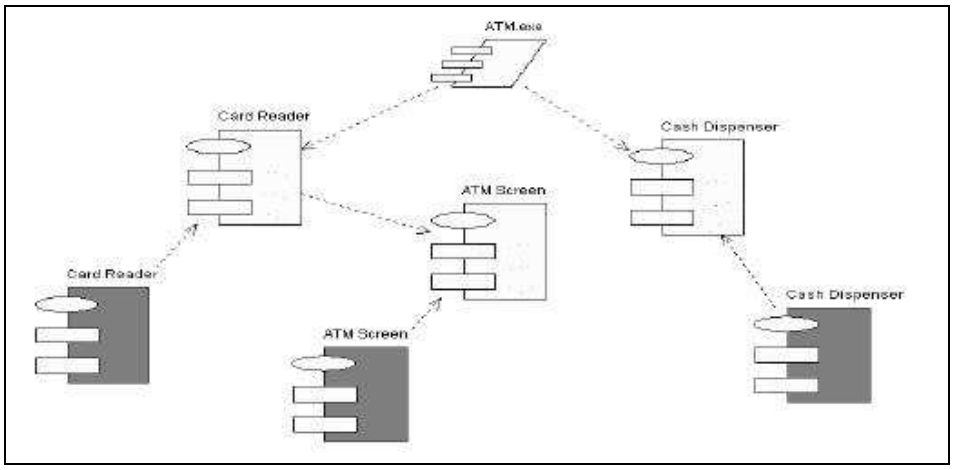
   Menurut Boggs dan Boggs (2002:17), *statechart diagram* menyediakan sebuah cara untuk memodelkan bermacam-macam keadaan yang mungkin dialami oleh sebuah obyek. Jika dalam diagram kelas menunjukkan gambaran statis kelas-kelas dan relasinya, diagram keadaan digunakan untuk memodelkan tingkah laku dinamik sistem.



Gambar 2.16 Statechart Diagram

8. Diagram Komponen (*component diagram*).

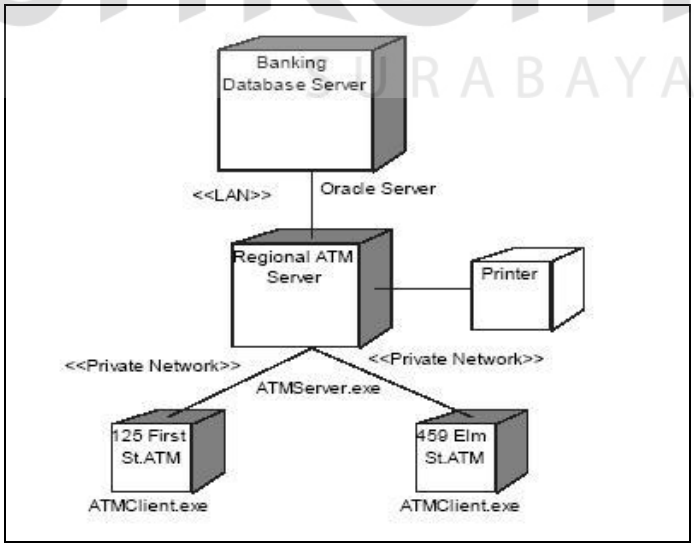
   Menurut Boggs dan Boggs (2002:19), *component diagram* menunjukkan gambaran fisik dari model, seperti komponen *software* dalam sistem beserta *relationship*-nya. Ada dua macam tipe komponen dalam *component diagram* yaitu *executable components* dan *code libraries.* Gambar 2.10 menunjukkan *component diagram* untuk sistem ATM. Garis yang menghubungkan komponen dalam *component diagram* menunjukkan *compile time* dan *run time dependencies*.



Gambar 2.17 Component Diagram

9. Diagram Penyebaran (*deployment diagram*)

Menurut Boggs dan Boggs (2002:21), *deployment diagram* menggambarkan *layout* fisik jaringan di mana berbagai komponen berada. Gambar 2.17 menunjukkan *deployment diagram* untuk sistem ATM. Pada sistem ATM tersebut terdapat banyak sub-sistem yang dijalankan pada peralatan fisik yang terpisah.



Gambar 2.18 Deployment Diagram

2.5. Hypertext Preprocessor (PHP)

PHP pada mulanya ditulis sebagai sebuah kumpulan dari CGI dengan menggunakan bahasa pemrograman C oleh *programmer* bernama Rasmus Lerdorf. Progammer asal Greenland ini membuat PHP pada tahun 1994 untuk menggantikan sebagaian kceil kumpulan *script* dengan Perl yang digunakan untuk maintenance halaman web miliknya. Lerdoff mengawali menciptakan PHP untuk menampilkan *resume* miliknya dan mengumpulkan beberapa data, seperti berapa banyak lalu lintas data yang diterima dalam halaman web miliknya. (Welling,2001).

PHP adalah *server side scripting environment* yang dapat digunakan untuk membuat dan menjalankan aplikasi–aplikasi di *web server* agar menjadi lebih interaktif dan *programmable*. Dengan PHP aplikasi–aplikasi yang ada di web server benar–benar akan dijalankan di web server tanpa mengharuskan adanya tambahan atau syarat tertentu untuk sisi *client* (*web browser*). PHP biasanya dijadikan sebagai *module* dalam suatu web server agar bisa mengeksekusi file–file PHP yang tersedia di web server. PHP dapat berjalan di hampir seluruh platform, *open source* dan berlisensi GNU *Public License* (GPL). (Welling, 2001).

Pada aplikasi ini, PHP digunakan sebagai server untuk menyediakan data yang didapat dari database secara real time kepada user yang mengakses aplikasi membutuhkan data dari server.