BAB II

TINJAUAN PUSTAKA

2.1 *Framework*

2.1.1 Pengertian *Framework*

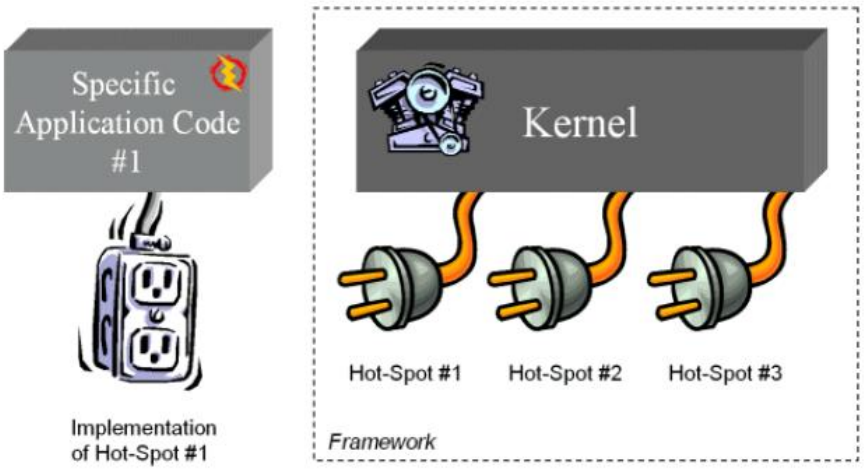
Framework merupakan perangkat lunak yang mulai menjadi pilihan untuk membuat suatu aplikasi (Andresta, 2008). Kemudahan-kemudahan yang diberikan menarik orang-orang untuk menggunakannya. Hal ini tidak terlepas dari tingkat efektifitas dan efisiensinya yang lebih baik dalam proses pengembangan suatu perangkat lunak.

Framework adalah sekumpulan perintah/fungsi dasar yang dapat membantu dalam menyelesaikan proses-proses yang lebih kompleks (Visikom, 2009). *Framework* adalah suatu aplikasi yang dapat digunakan ulang untuk membuat bermacam-macam aplikasi (Jhonson, 2009). *Framework* merupakan kumpulan beberapa kelas abstrak pada domain tertentu sehingga pengembang yang menggunakan *Framework* harus melengkapi kelas abstrak tersebut menjadi perangkat lunak yang diinginkan (Andresta, 2008).

Framework merupakan rancangan sistem yang dapat digunakan ulang. Di dalamnya terdapat interaksi kumpulan objek tertentu. *Framework* mendeskripsikan bagaimana hubungan dan interaksi objek-objek tersebut beserta antar muka dan aliran kembali antar objek tersebut.

Terdapat tiga karakteristik utama program berorientasi objek dalam *Framework* yaitu abstraksi data (*data abstraction*), *polymorphism*, dan pewarisan (*inheritance*). Data abstrak adalah representasi antarmuka yang implementasinya dapat berubah. *Polymorfisme* adalah kemampuan variabel untuk menyimpan nilai dengan tipe yang bermacam-macam, sedangkan pewarisan mempermudah dalam pembuatan komponen baru.

Framework adalah sebuah mesin yang membutuhkan *power* untuk dapat hidup. Mesin *Framework* ini mempunyai *plug-plug* yang disebut *hotspot* dari *Framework*. *Hotspot* ini merupakan bagian yang akan diubah menjadi kelas-kelas abstrak. Untuk dapat hidup tiap *hotspot* ini harus diberikan tenaga berupa kode aplikasi yang akan digunakan kernel *Framework*, yaitu bagian yang tidak berubah.



Gambar 2.1 *Framework* Sebagai Sebuah Mesin (Andresta, 2008)

2.1.2 Karakteristik *Framework*

Karakteristik *Framework* adalah pemodulan, guna ulang, perluasan, dan *inversion of control*

2.1.2.1 Pemodulan (*Modularity*)

Framework terdiri dari beberapa kelas abstrak dengan antarmuka tertentu. Detail implementasi dapat dienkapsulasi dan efek perubahan implementasi dapat dilokalisasi. Lokalisasi juga dapat membantu memahami dan melakukan perawatan terhadap perangkat lunak.

2.1.2.2 Guna Ulang

Framework merupakan satu dari beberapa teknik guna ulang. *Framework* mengguna ulang analisis, rancangan dan kode implementasi.

1. Analisis

Guna ulang analisis mendeskripsikan berbagai jenis objek yang penting dan menyediakan kosa kata yang membahas domain masalah yang dihadapi.

2. Rancangan

Guna ulang rancangan (*design*) dapat dilihat dalam bentuk pola (*pattern*). Pola (*pattern*) merepresentasikan kumpulan solusi yang sama yang digunakan untuk menyelesaikan permasalahan dalam pengembangan perangkat lunak dalam konteks tertentu.

3. Kode Implementasi

Framework mengguna ulang kode karena akan mempermudah membangun sebuah aplikasi dari komponen-komponen pustaka yang tersedia. Alasan lain *Framework* mengguna ulang kode karena sebuah komponen baru dapat dengan mudah diturunkan dari *superclass* yang abstrak.

2.1.2.3 Perluasan (Extensibility)

Aspek perluasan diperlukan untuk mengubah fitur dan layanan pada aplikasi baru yang dibuat sesuai dengan kebutuhan. Untuk aspek perluasan, ada lokasi pada sebuah *framework* di mana fitur aplikasi yang dibuat dihubungkan dengan *framework*. Lokasi tersebut dinamakan *hook*.

Hook pada sebuah framework berada pada *hot spot*, yaitu bagian *framework* yang dapat berubah. *Hot spot* merupakan kelas-kelas abstrak atau metode - metode yang harus diimplementasikan. *Framework* bukanlah program yang dapat dieksekusi (*executable*). Untuk menghasilkan program yang dapat dieksekusi, *developer* harus menginstansiasi *framework* dengan mengimplementasikan kode - kode untuk aplikasi pada setiap *hot spot*. Setelah setiap *hot spot* diinstansiasi barulah *framework* dapat menggunakan kelas - kelas tersebut.

Tetapi ada juga bagian-bagian tertentu dari *framework* yang tidak dapat diubah. Bagian tersebut merupakan *kernel* dari *framework*, atau dapat juga disebut *frozen spot framework*. Tidak seperti *hot spot*, *frozen spot* merupakan kumpulan kode yang telah diimplementasikan pada *framework* yang kemudian akan memanggil satu atau lebih *hot spot* yang telah diimplementasikan oleh pengembang aplikasi yang menggunakan *framework*. *Kernel* (*frozen spot*) tidak akan berubah,

konstan dan merupakan bagian yang selalu ada pada setiap instansiasi *framework*.

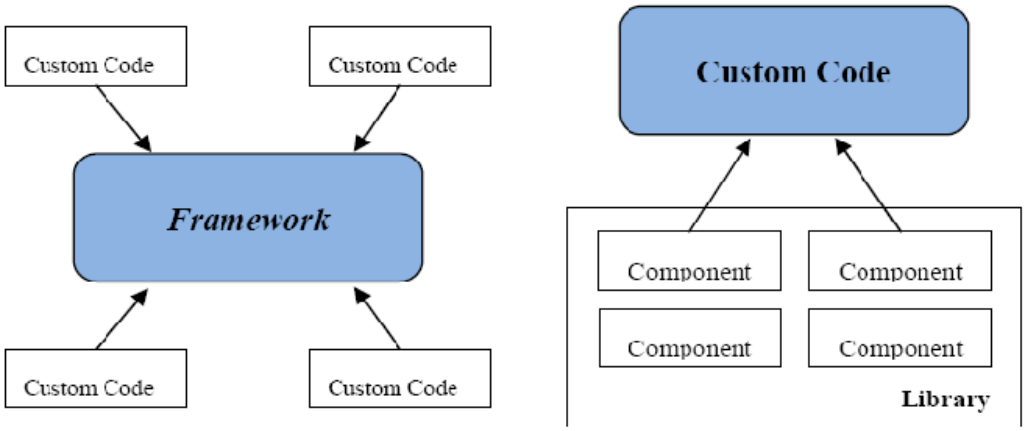
2.1.2.4 *Inversion of Control*

Pada umumnya, *developer* yang mengembangkan aplikasi menggunakan pustaka (*library)* dengan menulis program utama yang memanggil komponen-komponen tersebut ketika dibutuhkan. *Developer* memutuskan kapan memanggil komponen tersebut dan bertanggung jawab terhadap struktur dan kendali program secara keseluruhan. Hal tersebut berbeda dengan *Framework* di mana yang diguna ulang adalah program utamanya. *Developer* memutuskan apa saja yang ditambahkan pada program utama tersebut. Yang terpenting dalam hal ini adalah bahwa kode yang dibuat oleh *developer* dipanggil oleh kode *Framework* yang digunakan. *Framework* menentukan struktur dan kendali program secara keseluruhan. Perbedaan tersebut dapat dilihat pada Tabel 2.1 dan Gambar 2.2.

Tabel 2.1 Perbedaan *Framework* dengan *Library*

Aspek	*Library*	*Framework*
Bagian aplikasi yang Dikembangkan	Program utama (main program)	Subkelas dan komponen
Alur kendali	Ditentukan pengembang aplikasi	Ditentukan *Framework* (pengembang *Framework*)

Kode buatan pengembang aplikasi (*custom code*)	*Custom code* memanggil library *code*	*Framework code* memanggil *custom code*



Gambar 2.2 Perbedaan *framework* dengan *library*

2.1.3 Struktur *Framework*

Secara umum, *framework* menggunakan struktur MVC (*Model*, *View*, *Controller*) (Visikom, 2009)**.**

Jika digambarkan terlihat seperti di bawah ini :

Input > Processing > Output = Controller > Model > View

2.1.3.1 *Model*

Mencakup semua proses yang terkait dengan pemanggilan struktur data baik berupa pemanggilan fungsi, proses *input*, maupun pencetakan *output* ke dalam *browser*.

2.1.3.2 *Controller*

Mencakup semua proses yang terkait dengan pemanggilan *database* dan kapsulisasi proses-proses utama.

2.1.3.3 *View*

Semua yang berhubungan dengan desain antarmuka atau yang terkait *layout output*.

2.1.4 Klasifikasi *Framework*

Framework diklasifikasikan berdasarkan beberapa aspek seperti lingkup, teknik pengembangan dan tingkat generalitas.

2.1.4.1 Lingkup

Menurut lingkupnya *Framework* dibagi menjadi 3 bagian yaitu:

1. Sistem *Infrastruktur Framework*

Framework menyederhanakan pengembangan infrastruktur untuk suatu sistem yang *portable* dan efisiensi untuk sistem operasi dan sistem komunikasi dan juga pengembangan antarmuka pengguna.

2. *Middleware integratiom Framework*

Framework digunakan dalam proses integrasi aplikasi. *Framework* dirancang untuk meningkatkan kemampuan pengembang dalam hal

permodulan, penggunaan ulang, serta perluasan infrastruktur perangkat lunak agar dapat bekerja pada lingkungan terdistribusi.

3. *Enterprise application Framework*

Framework ini ditujukan untuk aktivitas bisnis. Perbedaan dengan *Framework* yang lain adalah terletak pada pembiayaannya yang besar untuk pengembangannya atau membelinya dari vendor. Namun *Framework* ini dapat mengembalikan investasi (*return of investment)* dikarenakan *Framework* jenis ini mendukung pengembangan aplikasi untuk pengguna akhir (*end user)*.

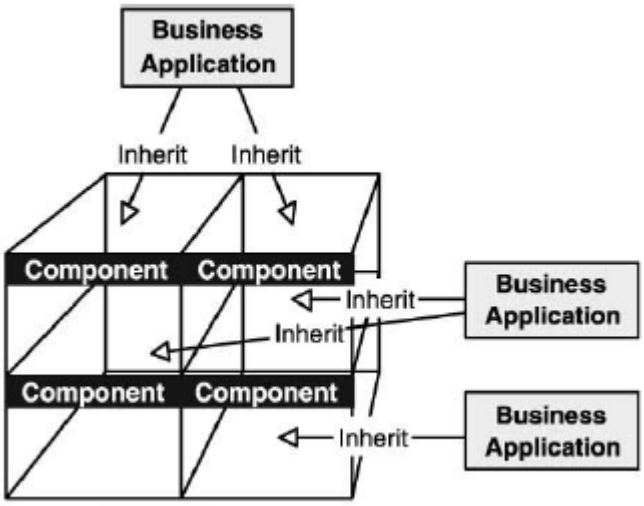
2.1.4.2 Teknik Pengembangan

Framework diklasifikasikan menjadi dua berdasarkan teknik pengembangnya yaitu *white-box Framework* atau dikenal dengan *architecture-driven Framework* dan *black-box Framework* yang dikenal dengan data *driven Framework* (Jhonson, 2009).

1. *White-box Framework*

Instansiasi *Framework* ini hanya dapat dilakukan dengan menciptakan kelas- kelas baru. Kelas-kelas tersebut beserta kode implementasinya dapat ditambahkan dengan pewarisan atau komposisi. Arsitektur *white-box Framework* harus terdokumentasi dengan baik karena pengetahuan yang mendalam mengenai detail arsitektur *Framework*

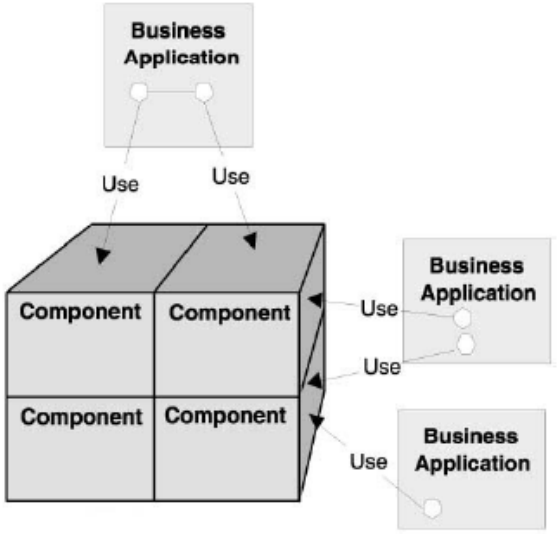
sangat diperlukan untuk mengembangkan aplikasi sesuai dengan keinginan pengembang.



Gambar 2.3 *White Box Framework* (Andresta, 2008)

2. *Black-box Framework*

Black-box Framework menyembunyikan struktur internalnya. Pengguna hanya mengetahui deskripsi umum penggunaan *Framework* dan *hotspot* yang disediakan. Mekanisme yang disediakan untuk *instanisasi framework* ini hanya dengan komposisi sehingga pengguna tidak harus mempelajari detail internal *Framework*.



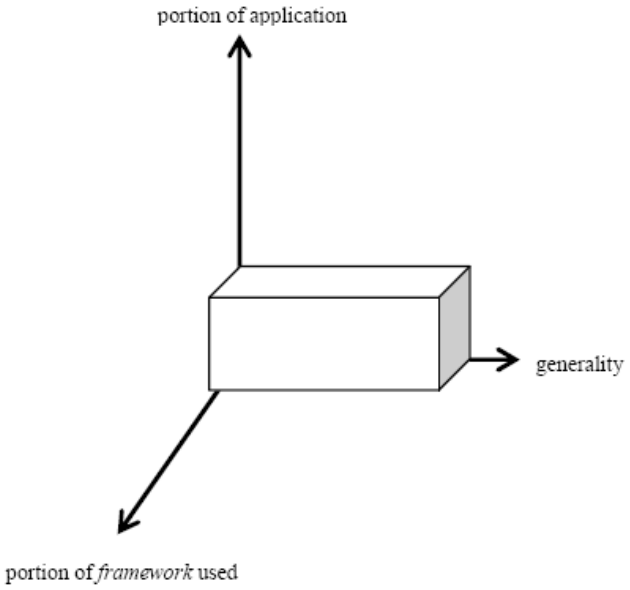
Gambar 2.4 *Black box Framework* (Andresta, 2008)

2.1.4.3 Tingkat Generalitas

Menurut tingkat generalitasnya, *Framework* dibagi 2 yaitu yaitu Horizontal *Framework* dan Vertikal *Framework*.

1. Horizontal *Framework*

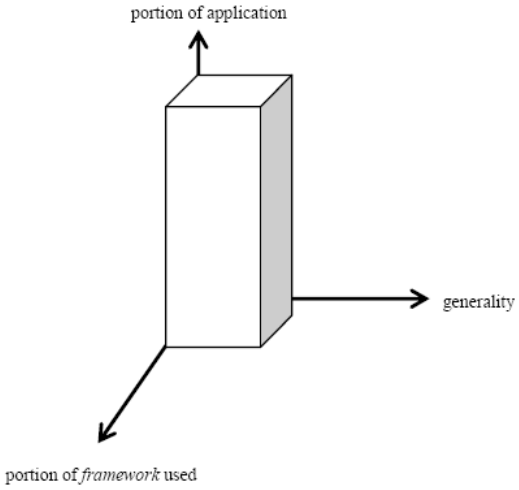
Framework ini bersifat umum. Sehingga dapat digunakan untuk membuat berbagai macam aplikasi. *Framework* ini digunakan untuk mengembangkan antar muka dari aplikasi pada area yang cukup luas di industri perangkat lunak. Contoh perangkat lunak ini adalah GUI *toolkit*.



Gambar 2.5 Horizontal *Framework*

2. Vertikal *Framework*

Framework ini dikenal dengan aplikasi *Framework*. Aplikasi ini ditujukan untuk pembuatan aplikasi yang spesifik pada masalah tertentu saja. Misalnya saja aplikasi untuk sistem akademik dimana aplikasi tersebut di khususkan dalam aplikasi sistem akademik. Fitur pengembangan yang diberikan lebih lengkap dikarenakan pengembang terlebih dahulu melakukan analisis dari *user requirement* tentang masalah-masalah dalam pengembangan aplikasi tersebut.



Gambar 2.6 Vertikal *Framework*

2.2 Sistem Akademik Sekolah (SAS)

Sistem Akademik Sekolah (SAS) adalah contoh aplikasi sistem informasi sekolah yang lebih memfokuskan fungsinya pada kegiatan akademik yang berhubungan dengan kegiatan belajar mengajar, seperti penanganan absen, penanganan nilai, pengumpulan tugas dan lain-lain.

Tabel 2.2 Fitur-fitur dalam Sistem Akademik Sekolah (SAS)

No	Nama Fitur	Keterangan
1	Absensi	Mencatat dan menampilkan jumlah kehadiran siswa
2	Nilai	Nilai-nilai yang didapat siswa ditampilkan agar siswa dan orang tua dapat melihat *transparansi* dari nilai yang didapat siswa.

3	Silabus	Fitur ini memberikan kemudahan bagi siswa untuk melihat resume mata pelajaran yang di ajarkan sekolah. Siswa bisa mendapatkan bahan yang dibutuhkan untuk belajar selain catatan selama siswa dikelas
4	Jadwal	Fitur ini memberikan jadwal dari mata pelajaran dari masing-masing kelas
5	Tugas	Selain memberikan tugas saat tatap muka, guru dapat mempublikasikan tugasnya melalui sistem. Siswa yang tidak hadir dapat melihat tugas yang diberikan.
6	Kalender Akademik	Kalendar akademik menampilkan lamanya jadwal belajar mengajar selama satu semester, waktu pelaksanaan Ujian Tengah Semester dan Ujian Akhir Semester, hari libur nasional, kegiatan – kegiatan yang dilaksanakan sekolah dan hal penting lainnya yang perlu diketahui dalam satu tahun ajaran.

2.3 *Unified Process* (RUP)

Rational unfied process (RUP) merupakan suatu metode dalam pengembangan perangkat lunak dengan mengumpulkan berbagai *best practises* (Taryana, 2007). Ciri utama metode ini adalah mengunakan *use-ase driven* dan pendekatan iteratif dalam siklus pengembangan perangkat lunak. RUP menggunakan konsep *object oriented*

dengan aktifitas yang berfokus pada pengembangan model dengan menggunakan *Unfied Model Language*. RUP memiliki 2 bagian yaitu Dimensi pertama dan Dimensi kedua.

Dimensi pertama digambarkan secara horizontal. Dimensi ini mewakili aspek-aspek dinamis dari pengembangan perangkat lunak. Dimensi ini merupakan fase daur hidup RUP. Dimensi ini terdiri atas *Inception, Elaboration, Construction,* dan *Transition*.
Dimensi kedua digambarkan secara Vertikal. Dimensi ini mewakili dari statis dari proses perangkat lunak. Dimensi ini merupakan bagian-bagian dan cara kerja RUP. Terdiri atas 3 elemen penting yaitu *who, what dan how.* RUP terdiri atas *Business Modelling, Requirement, Analysis and Design, Implementation, Test Deployment, Configuration and Change Management, Project Management, Encirontment*.

Pada penggunaan kedua standar tersebut di atas yang berorientasi objek (*Object Oriented*) memiliki manfaat yaitu :

1. *Improve productivity*

Standard ini dapat memanfaatkan kembali komponen-komponen yang telah tersedia/dibuat sehingga dapat meningkatkan produktifitas.

2. *Deliver high quality system*

Kualitas sistem informasi dapat ditingkatkan sebagai sistem yang dibuat pada komponen-komponen yang telah teruji (*well-tested* dan *well-proven*) sehingga dapat mempercepat *delivery* sistem informasi yang dibuat dengan kualitas yang tinggi.

3. *Lower maintenance cost*

Standard ini dapat membantu untuk menyakinkan dampak perubahan yang terlokalisasi dan masalah dapat dengan mudah terdeteksi sehingga hasilnya biaya pemeliharaan dapat dioptimalkan atau lebih rendah dengan pengembangan informasi tanpa standard yang jelas.

4. *Facilitate reuse*

Standard ini memiliki kemampuan yang mengembangkan komponen-komponen yang dapat digunakan kembali untuk pengembangan aplikasi yang lainnya.

5. *Manage complexity*

Standard ini mudah untuk mengatur dan memonitor semua proses dari semua tahapan yang ada sehingga suatu pengembangan sistem informasi yang amat kompleks dapat dilakukan dengan aman dan sesuai dengan harapan semua manajer proyek IT/IS yakni *deliver good quality software within cost and schedule time and the users accepted*.

2.3.1 Daur Hidup RUP

Daur hidup RUP terdiri atas 4 fase yaitu :

1. *Inception*
2. Elaboration
3. Construction
4. *Transition*

2.3.1.1 *Inception*

Hal-hal yang perlu dilakukan dalam fase ini adalah :

1. Menentukan ruang lingkup proyek
2. Membuat *Business Case*
3. Menentukan Tipe model dalam proses pengembangan proses

2.3.1.2 *Elaboration*

Hal-hal yang perlu dilakukan dalam fase ini adalah :

1. Menganalisa *problem domain*
2. Mengembangkan perencanaan proyek
3. Menganalis resiko
4. Pengembangan perencanaan arsitektural
5. Pengimplementasian *use case*

2.3.1.3 *Contruction*

Hal-hal yang perlu dilakukan dalam fase ini adalah *coding*

2.3.1.4 *Transition*

Hal-hal yang perlu dilakukan dalam fase ini adalah :

1. Testing
2. Membuat Documentasi
3. Membuat rencana peluncuran produk

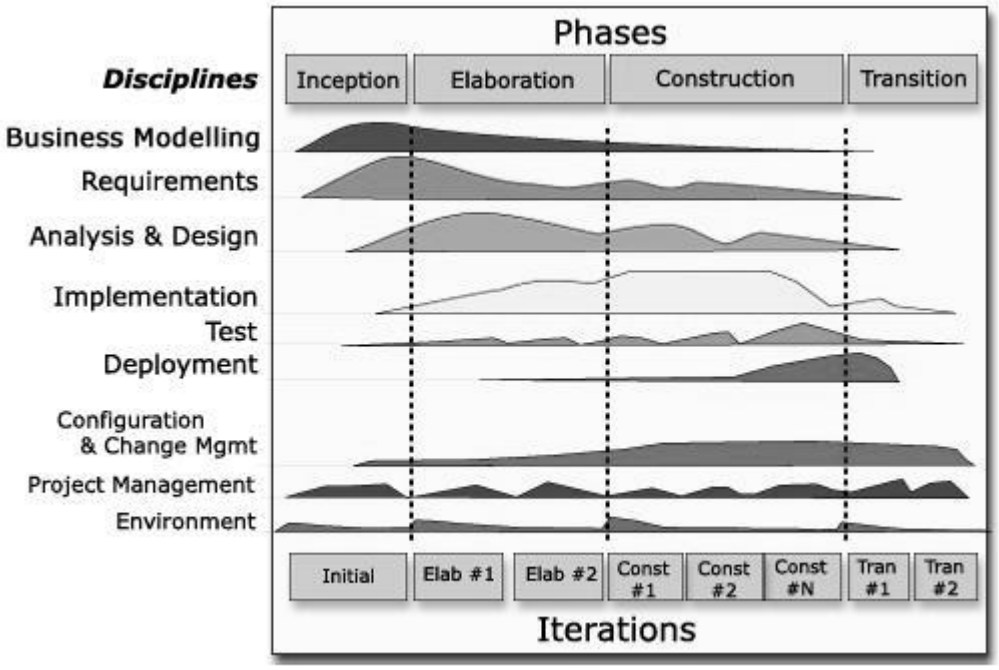
2.3.2 Bagian-Bagian dan Cara Kerja RUP

Elemen-elemen isi dari metode RUP terdiri atas :

1. *Who* (Peranan)
2. *What* (Hasil kerja)
3. *How* (tugas-tugas)

Dalam setiap iterasi atau pengulangan tugas-tugasnya dikategorikan menjadi 8 bagian yang terdiri atas :

1. *Business Modelling*
2. *Requirement*
3. *Analysis dan Design*
4. *Implementation*
5. *Test Deployment*
6. *Configuration* dan *Change Manegement*
7. *Project Management*
8. *Environment*



Gambar 2.7 Arsitektur *Rational Unified Process*

2.4 PHP

PHP merupakan singkatan dari *Hypertext Preprocessor*. PHP dikembangkan pertama kali tahun 1995 oleh Rasmus Lerdorf yang merupakan salah satu anggota group Apache. PHP pertama kali didesain sebagai alat *tracking* pengunjung *website* Lerdorf (Dza, 2008).

Kemudian, fungsinya diperlebar dan dihubungkan dengan Apache. PHP dikembangkan sepenuhnya sebagai bahasa skrip *server side programming* (dijalankan pada sisi *server*). PHP bersifat *open source* sehingga dapat digunakan dengan gratis.

PHP mempunyai kemampuan dapat mengakses *database* dan diintegrasikan dengan HTML. PHP lebih popular dalam jumlah pemakai dibandingkan dengan modul Perl, CGI dan ASP (Dza, 2008).

PHP semakin popular kerena memiliki beberapa kelebihan, diantaranya adalah sebagai berikut (Andi, 2007) :

1. Mudah dibuat dan dijalankan.
2. Mampu berjalan pada sistem operasi yang berbeda-beda.
3. Dapat berjalan pada *web server* yang berbeda-beda.
4. PHP bersifat *open source.*
5. Dapat diletakkan dalam tag HTML.

PHP shell adalah sebuah *shell* layaknya dalam *Linux* dan *Commant Prompt* dalam *windows*, hanya saja *shell* ini dibungkus dalam *script* PHP. Dengan PHP *shell* hampir semua perintah *shell* dapat dijalankan menggunakan *browser* sebagai media *consule*-nya. Untuk menjalankan PHP *shell* dibutuhkan file PHP *shell.*

Perintah Linux dijalankan pada PHP

1. ***Shell_exec*** :

   string **shell_exec** (string $cmd)

   Contoh :

   $perintah=” ls –l “;

Shell_exec($perintah);

2. ***System* :**

$perintah “ping localhost ”;

System($perintah);

2.5 MySQL

MySQL adalah perangkat lunak yang menangani masalah basis data. Banyak orang menyebut MySQL sebagai RDBMS (*Relational Database Management System*) (Dza, 2008). Kegunaan MySQL adalah menyimpan data ke dalam sebuah tabel yang terdapat dalam suatu *database*.

Terdapat beberapa keunggulan MySQL di antaranya sebagai berikut (Dza, 2008) :

- MySQL tersedia untuk berbagai *platform* Sistem Operasi.
- MySQL memiliki banyak fitur yang dapat digunakan dalam aplikasi web, misalnya klausa LIMIT digunakan untuk melakukan *paging* (penomoran halaman).