BAB II

LANDASAN TEORI

2.1 Sistem Informasi

Pengertian sistem informasi tidak bisa dilepaskan dari pengertian sistem dan informasi. Definisi dari sistem adalah sekelompok dua atau lebih komponen-komponen yang saling berkaitan (*interrelated*) atau subsistem-subsistem yang bersatu untuk mencapai tujuan yang sama (*common purpose*). Sedangkan definisi dari informasi adalah data yang diambil kembali, diolah, atau sebaliknya digunakan sebagai dasar untuk peramalan atau pengambilan keputusan. Sumber dari informasi adalah data. Data adalah fakta dan angka yang tidak sedang digunakan pada proses keputusan, dan biasanya berbentuk catatan historis yang dicatatkan dan diarsipkan tanpa maksud untuk segera diambil kembali untuk pengambilan keputusan. Secara lugas sistem informasi didefinisikan sebagai suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan-laporan yang diperlukan oleh pihak luar itu (Senn, 1989).

Sistem informasi memiliki tiga fungsi dasar :

1. Menerima data (*input*)
2. Mengubah data menjadi informasi (proses)
3. Untuk memproduksi dan mengkomunikasikan informasi ke dalam *timely fashion* bagi *user* untuk membuat

keputusan (*output*). Sebagai contoh, banyak bank dan institusi keuangan yang menggunakan sistem informasi untuk membantu menentukan apakah nasabah diperbolehkan untuk melakukan pinjaman.

2.1.1 Komponen Sistem Informasi

Sistem informasi mempunyai enam buah komponen, yaitu *input*, model, *output*, teknologi, basis data, dan kontrol. Keenam komponen ini harus ada bersama-sama dan membentuk satu kesatuan. Jika satu atau lebih komponen tersebut tidak ada, maka sistem informasi tidak akan dapat melakukan fungsinya, yaitu pengolahan data dan tidak dapat mencapai tujuannya, yaitu menghasilkan informasi yang relevan, tepat waktu, dan akurat. Komponen-komponen dari sistem ini dapat digambarkan sebagai berikut ini:

1. *Input*

   Input merupakan data yang masuk ke dalam sistem informasi. Sistem sistem informasi tidak akan dapat menghasilkan *output* jika tidak mempunyai komponen *input*.

2. *Output*

   Produk dari sistem informasi adalah *output* berupa informasi yang berguna bagi para pemakainya. *Output* dari sistem informasi dibuat dengan menggunakan data yang ada di basis data dan diproses menggunakan model tertentu.

3. Basis data

   Basis data adalah kumpulan dari data yang saling berhubungan satu dengan yang lainnya, tersimpan di

perangkat keras komputer dan digunakan perangkat lunak untuk memanipulasinya.

4. Model

   Model yang digunakan di sistem informasi dapat berupa model logika yang menunjukkan suatu proses perbandingan logika atau model matematik yang menunjukkan perhitungan matematika.

5. Teknologi

   Teknologi merupakan komponen yang penting di sistem informasi. Teknologi dapat dikelompokkan ke dalam dua macam kategori, yaitu teknologi sistem komputer (perangkat keras dan perangkat lunak) dan teknologi sistem telekomunikasi.

6. Kontrol

   Kontrol ini digunakan untuk menjamin bahwa informasi yang dihasilkan oleh sistem informasi sifatnya akurat.

2.1.2 Sistem Informasi Akademik Tugas Akhir dan Kerja Praktek

Sistem Informasi Akademik Tugas Akhir(TA) dan Kerja Praktek(KP) adalah suatu sistem informasi yang bertugas menangani informasi berkaitan layanan Kerja Praktek atau Tugas Akhir mahasiswa.

Sistem informasi TA dan KP ini mempunyai keuntungan sebagai berikut:

1. Peningkatan akurasi

Manusia sering berbuat kesalahan terutama untuk masalah membaca tulisan tangan. Dengan bantuan komputer maka tulisan menjadi jelas dan mudah dibaca.

2. Penghematan waktu

Proses administrasi KP dan TA jadi lebih cepat, karena sistem sudah memberikan kemudahan dalam pembuatan surat kp,jadwal ujian kp dan pendadaran.

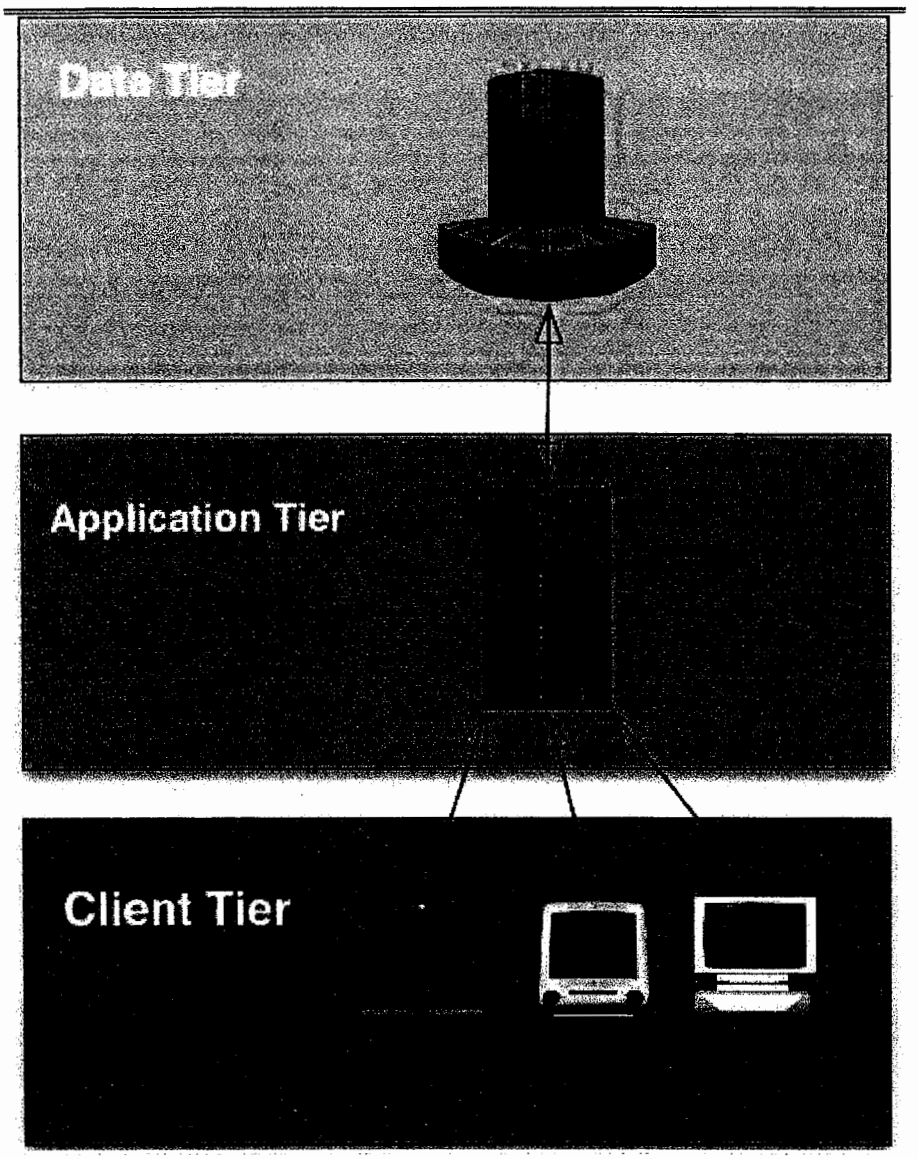
2.2 Sistem Informasi Berbasis *Web*

Saat ini komputer dan piranti pendukungnya telah masuk dalam setiap aspek kehidupan dan pekerjaan. Komputer tersebut dapat berdiri sendiri ataupun membentuk jaringan komputer dan *Internet* yang menghubungkan berbagai tempat di dunia. *Interconnected Network* atau *Internet* adalah sebuah sistem komunikasi global yang menghubungkan komputer-komputer dan jaringan-jaringan komputer di seluruh dunia, baik secara langsung maupun tidak langsung. *Web* adalah fasilitas *hypertext* (sistem yang menghubungkan informasi satu ke informasi yang lain melalui suatu link) untuk menampilkan data berupa teks, gambar, bunyi, animasi dan data multimedia lainnya yang diantaranya data tersebut saling berhubungan satu sama lain. Untuk memudahkan membaca data dan informasi pada *web* dapat menggunakan *web browser* seperti *Microsoft*

Internet Explorer, Netscape, Mozila Firefox, Safari, dan lain-lain.

2.2.1 Aplikasi Berbasis *Web*

Sebagian besar dari kita tentu sudah mengenal dan sering menggunakan *web browser* untuk mengakses *Internet*. Secara sederhana, aplikasi yang berbasis *web* adalah aplikasi yang hanya membutuhkan *web browser* pada sisi *client* untuk mengakses informasi yang disimpan di *server* pusat. Di lingkungan akademis, sistem seperti ini dikenal juga dengan aplikasi yang bersifat *Three Tier Application*.



Gambar 2.1 *Three Tier Application*

Pada contoh diatas, terlihat bahwa *Database* pada sisi *Data Tier*, *Web server* pada sisi *Application Tier* dan *Web Browser* pada sisi *Client Tier*. Data *Tier* dan *Application Tier* di-*install* pada mesin *server*, sedangkan pada *client* cukup di-*install Web Browser*. Data *Tier* program (MySQL, Oracle, SQL Server maupun *Database*

Management System / DBMS lainnya), bertugas mengelola semua data, sedangkan *Application Tier* program bertugas memberikan layanan *web* bagi para *client* yang akan mengakses informasi menggunakan *web browser*.

2.3 Ruby on Rails

2.3.1 Pengenalan *Ruby*

Ruby adalah bahasa pemrograman yang berorientasi objek. Bahasa ini mengkombinasikan sintak-sintak yang terinspirasi dari *Perl* dengan *Smalltalk*.

Ruby membaca *source code* baris perbaris kemudian dieksekusi, oleh karena itu *Ruby* termasuk *Interpreted Language*. Beberapa contoh *Interpreted language* lainya adalah *Perl*, *Phyton* dan *JavaScript*.

2.3.1.1 Sejarah

Ruby diciptakan oleh Yukihiro Matsumoto atau biasa disapa "Matz".

> *Ruby di ciptakan 24 February 1993. Pada waktu itu saya sedang berbicara dengan teman saya tentang bahasa scripting. Sebagai penggemar bahasa object oriented selama 15 tahun, saya merasa pemrograman OO juga cocok untuk scripting.*
>
> *Saya melakukan riset di Internet, tetapi kandidat yang saya temukan, Perl dan Phyton, tidak sesuai dengan harapan saya. Saya menginginkan bahasa yang lebih powerful dari Perl dan lebih object oriented dari Phyton.*

Kemudian saya teringat impian saya dan memutuskan untuk mendesain bahasa sendiri. Pada mulanya saya hanya bermain-main, tetapi lama kelamaan berkembang menjadi tools yang sanggup menggantikan Perl. Saya menamakannya Ruby-diambil dari nama batu berharga-dan merilisnya tahun 1995. Percaya atau tidak saat ini Ruby lebih populer ketimbang Phyton di Jepang. Saya berharap nantinya bahasa ini dapat berkembang ke seluruh dunia

(Yukihiro Matsumoto, a.k.a. "Matz")

2.3.1.2 Pemrograman Berorientasi Objek

Ruby adalah bahasa pemrograman yang berorientasi object (Object Oriented). Pemrograman berorientasi objek memiliki fokus terhadap konsep pemrograman objek, yang memiliki fungsi dan atribut yang digunakan untuk memanipulasi terhadap atribut tersebut. Konsep dasar object Oriented adalah enkapsulasi, pewarisan dan polimorfisme.

2.3.1.2.1 Enkapsulasi

Enkapsulasi adalah proses pemaketan data dengan fungsi. Enkapsulasi mencegah objek luar melihat implementasi detail dari sebuah objek.

2.3.1.2.2 Pewarisan (*Inheritance*)

Pewarisan merupakan suatu sifat yang memungkinkan kita membuat kelas dengan menggunakan kelas yang sudah ada. Pewarisan sering disebut sebagai bentuk spesialisasi atau generalisasi atas objek.

2.3.1.2.3 Polimorfisme (*Polimorfism*)

Polimorfisme mendeskripsikan sifat dari suatu objek yang bervariasi bergantung pada input.

2.3.1.3 Gem

Gem adalah paket library dari *Ruby*. *Gem* dikelola dengan perintah *gem*. Dengan perintah tersebut *gem* dapat di install, remove dan dicari. Ruby repository berada di http://rubyforge.org/projects/rubygems. Salah satu gem yang terkenal adalah Rails.

2.3.2 Pengenalan *Ruby on Rails*

2.3.2.1 Sejarah

Pada akhir tahun 2003, David Heinemeier Hansson dan 37Signals mulai bekerja pada project management solusi berbasis *web*. Hanssons mencoba menggunakan *PHP*, tapi terbentur dengan kekurangan yang dimiliki oleh *PHP*. Banyak programmer PHP yang mengalami masalah yang sama, karena terus-terusan mengulang *code* yang sama dalam membangun sistem.

Hansson mencari alternatif lain, kemudian menemukan bahasa Ruby. Dengan menggunakan Ruby Hansson mengembangkan project berbasis *web* yang pertama yang dinamakan Basecamp. Ketika mengerjakan Basecamp, Hansson menyadari banyak *code* yang tulis dapat di ekstrak dalam sebuah framework. Pada Juli tahun 2004 Ruby on Rails dirilis.

2.3.2.2 Ruby on Rails (RoR)

Ruby on Rails, selanjutnya disingkat *RoR*, merupakan *framework* untuk aplikasi *web* dengan menggunakan bahasa *Ruby*. Dalam membuat suatu *web* aplikasi, *RoR* memiliki struktur atau arsitektur aplikasi *MVC*.

2.3.2.2.1 MVC (Model, View, Controller)

MVC (*Model View Controller*) adalah arsitekur yang biasa di gunakan dalam pembuatan *software*. Model ini memisahkan *business logic* dari *user interface* sehingga aplikasi menjadi lebih mudah untuk dimodifikasi. Pada arsitektur ini *Model* mewakili informasi (data) dari aplikasi, *View* merupakan elemen dari *user interface* dan *Controller* mengatur komunikasi antara *model* dan *user interface*.

1. **Model**

   Struktur ini bertanggung jawab dalam menjaga keadaan dari suatu aplikasi, disebut juga sebagai lapisan data sebelum data dikirim ke *Controller* atau basis data. *Model* digunakan untuk proses *insert, update, dan delete* dari tempet penyimpanan data, seperti yang terjadi dalam relasional basis data.

2. **View**

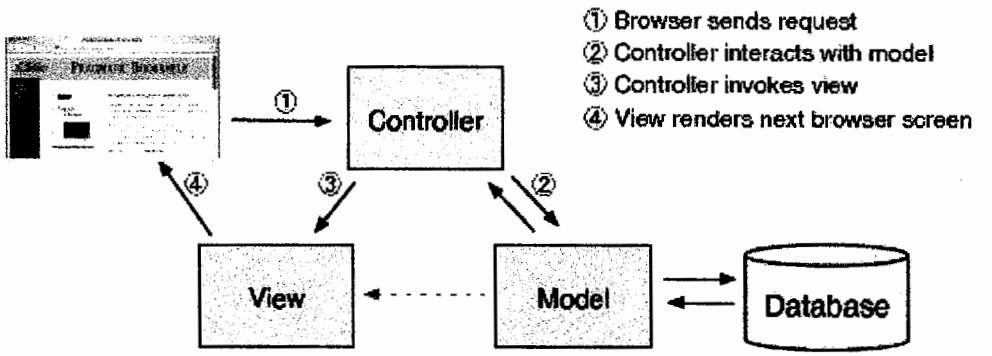
   Struktur ini bertanggung jawab untuk menghasilkan suatu *user interface*, umumnya didasarkan pada data yang ada dimodel yang akan ditampilkan pada *Browser*.

3. Controller

Komponen yang menerima permintaan atau perintah dari browser dan menjalankan perintah spesifik dari user, meng-interaksikannya dengan model kemudian menampilkan hasil proses kembali ke *View* pada browser user.

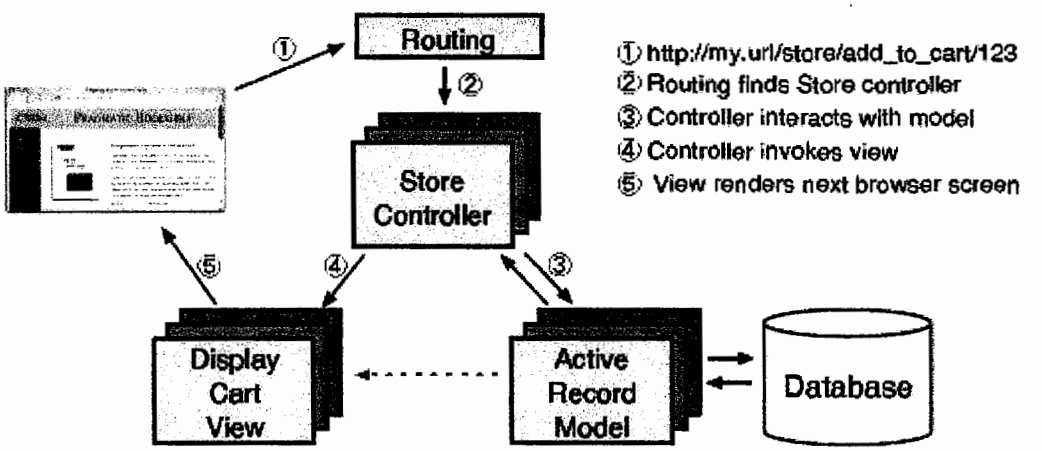
Controller merupakan *business logic* dalam aplikasi *RoR* yang menghubungkan interaksi antara user, view dan model. Fungsi lain dari *Controller* yaitu:

1. Bertanggung jawab memproses perintah eksternal ke internal *actions*. Hal ini mampu menangani URLs yang *user-friendly*.
2. Melakukan manajemen terhadap *caching* yang mampu mempercepat kinerja kerja aplikasi.
3. Melakukan manajemen terhadap Modul Pembantu *(Helper Modules)* yang menjalankan kemampuan pola *View* tanpa memperbesar bagian kode.
4. Melakukan manajemen terhadap *session*, sehingga user bisa secara terus menerus menggunakan aplikasi sesuai dengan *session* mereka, tanpa harus melakukan identifikasi atau *login* jika ingin mengakses setiap halaman.



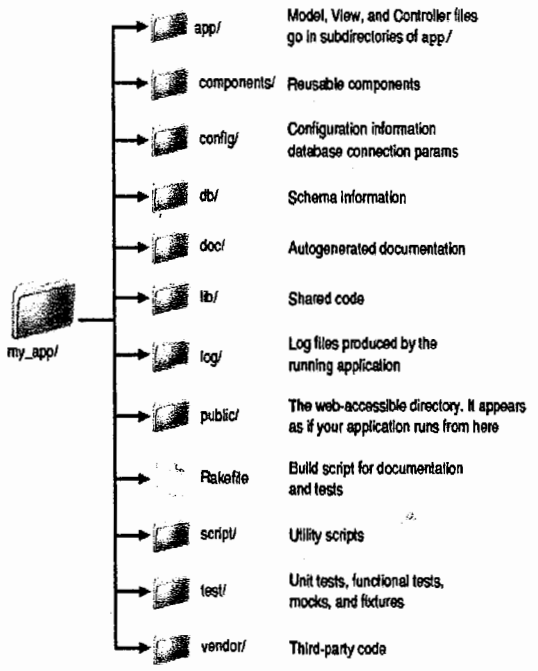
Gambar 2.2 Arsitektur Model-View-Controller

Didalam aplikasi *RoR*, perintah input pertama kali di kirim ke router yang menjalankan semuanya. Didalam router terjadi identifikasi dari beberapa bagian method atau *action* dibeberapa tempat dalam kode *controller*. Action yang terjadi memungkinkan untuk membaca data selama perintah berlangsung yang memungkinkan terjadinya interaksi dengan *model* sehingga menyebabkan *action* lainya dijalankan. Dalam hal ini *action* mempersiapkan informasi untuk *view* yang akan ditampilkan ke user.



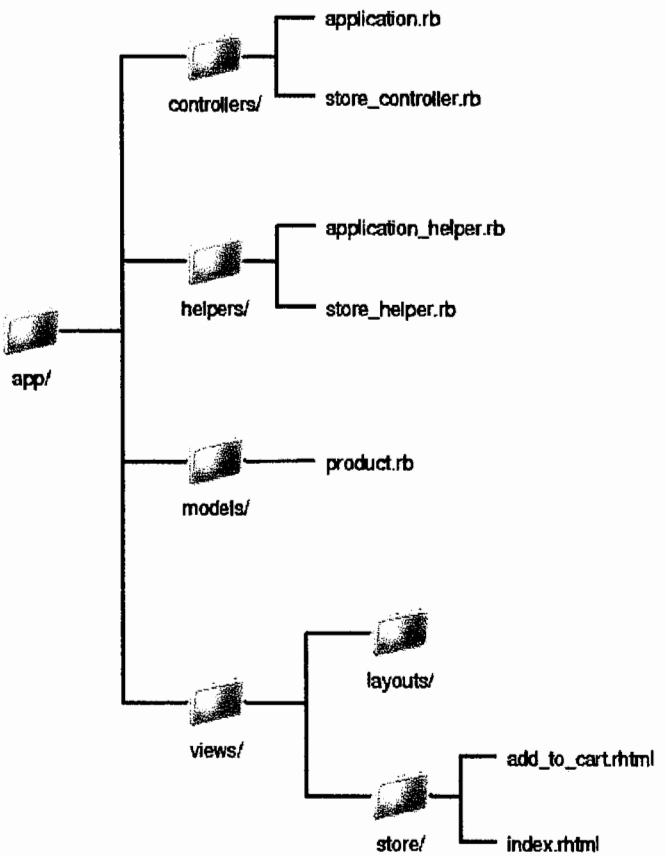
Gambar 2.3 Interaksi RoR dengan MVC

Aplikasi struktur Model-View-Controller disimpan dalam struktur direktori *RoR* yang dapat dilihat sebagai berikut:



Gambar 2.4 Struktur direktori pada suatu aplikasi *Ruby On Rails*

Dapat dilihat bahwa Model-*View-Controller* disimpan dan subdirektori *app*, jika direktori *app* diurai maka hasilnya sebagai berikut :



Gambar 2.5 Subdirektori *app*

2.3.2.2.2 Prinsip Dasar RoR

RoR mendukung beberapa prinsip aplikasi yang membuatnya lebih menonjol dari *framework* lain. Prinsip-prinsip itu adalah:

1. ***Convention Over Configuration***

Konsep ini mengacu pada fakta bahwa RoR diasumsikan kumpulan sejumlah aplikasi *web* yang khas yang sudah menjadi standar. Pada *Framework* lain seperti *Java-based Struts* atau *Python-based Zope*, perlu suatu konfigurasi yang panjang sebelum memulainya. Informasi konfigurasi biasanya disimpan di dalam file XML yang besar dan sulit untuk di-*maintain*, atau wajib mengulangi seluruh proses konfigurasi pada saat mulai suatu proyek yang baru.

Heinemeier Hansson dengan sengaja menciptakan RoR sedemikian sehingga tidak memerlukan konfigurasi berlebihan, sepanjang beberapa konvensi yang standar diikuti.

2. ***Don't Repeat Yourself***

RoR mendukung prinsip DRY (*Don't Repeat Yourself*) *programming*. Ketika memutuskan untuk mengubah perilaku suatu aplikasi yang didasarkan pada prinsip DRY, seharusnya tidak perlu memodifikasi terlalu banyak sintak.

Contoh bagaimana RoR mendukung prinsip DRY adalah bahwa, tidak seperti *Java*, RoR tidak memaksa untuk mengulangi definisi *database* di dalam aplikasi yang dibuat.

RoR juga menerapkan teknik Ajax (*Asynchronous Javascript and XML*). Ajax adalah satu pendekatan

yang mengizinkan aplikasi *web* untuk menggantikan isi di dalam *browser* secara dinamis. Misalnya, untuk menukar data sebuah *form* tanpa harus mengunduh seluruh halaman dari *server*. Para pengembang sering kali harus menyalin sintak selagi menciptakan aplikasi-aplikasi Ajax. RoR membuatnya mudah untuk memperlakukan masing-masing generasi *browser* tanpa harus menyalin setiap sintak untuk masing-masing *browser*.

3. ***Agile Development***

Pada pendekatan pengembangan *software* tradisional pembuatan sketsa perencanaan biasanya lebih panjang dan kaku. Pendekatan ini biasanya memakai pendekatan aplikasi dari bawah-keatas, langkah pertamanya berdasarkan data.

Sebaliknya, metode *Agile Development* menggunakan pendekatan *adaptive*. Pendekatan *adaptive* ini hanya membutuhkan sejumlah kecil tim sebagai pengembang. Para pengembang *Agile* juga merancang aplikasinya dari atas-kebawah yang dimulai dari desain *layout*.

Menurut Patrick Lentz (2007) ada beberapa contoh ilustrasi RoR menganut metode *Agile Development*.

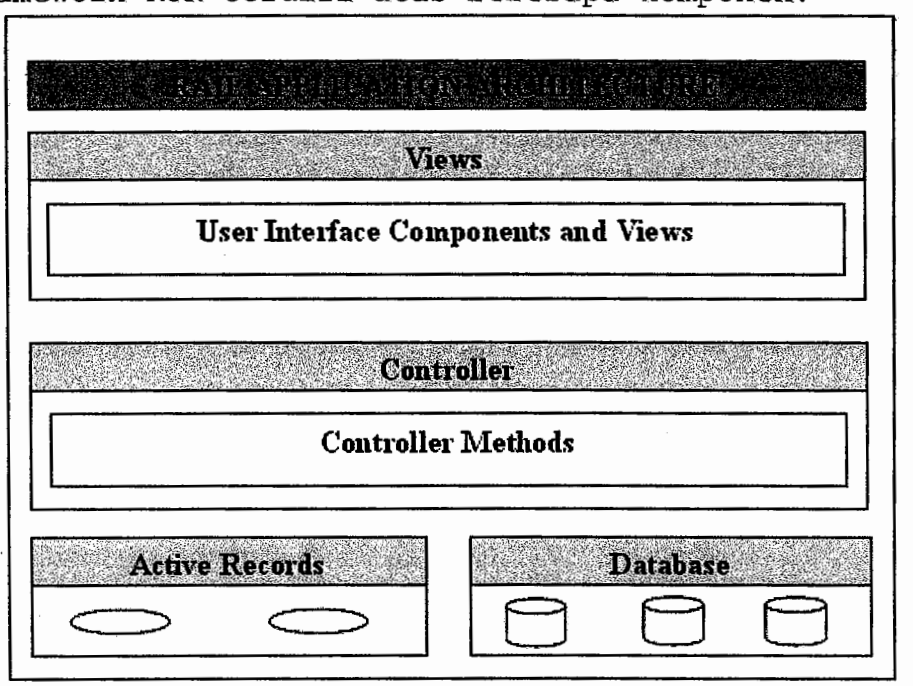
a. Pekerjaaan mendesain aplikasi RoR dapat dimulai sebelum memutuskan mengenai database. Tidak perlu membuat ulang layout tersebut ketika menambahkan fungsi ke desain, semuanya telah diatur secara dinamis berdasarkan kebutuhan.

b. Tidak seperti sintak yang dibuat dalam bahasa C atau Java, aplikasi RoR tidak memerlukan kompilasi sehingga mempercepat proses pengembangan.
c. RoR menyediakan otomatisasi uji coba sintak aplikasi pada proses pengembangan.
d. *Refactoring* (*rewriting code with an emphasis on optimization*).

Oleh karena prinsip-prinsip di atas, RoR merupakan *framework* yang benar-benar menghemat waktu dalam usaha para pengembang membuat sebuah aplikasi *web* berbasis RoR.

2.3.2.2.3 *Arsitektur RoR*

Framework RoR terdiri atas beberapa komponen:

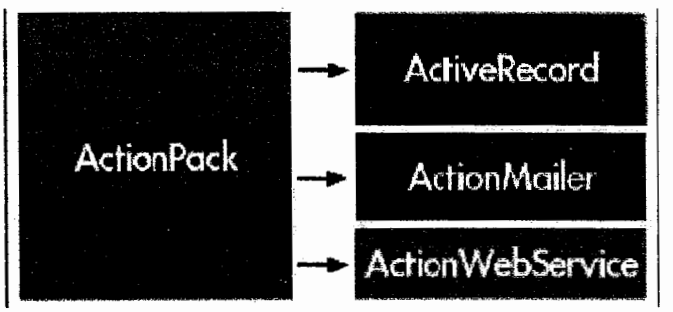


Gambar 2.6 Arsitektur RoR

2.3.2.2.3.1 Active Record

Active Record merupakan *Object Relational Mapping* (OMR) *Layer* pada *RoR*. Didalám OMR tabel dipetakan ke *Class*, baris ke objek dan kolom ke objek atribut. *Active Record* digunakan untuk meminimalisasi jumlah konfigurasi objek relational basi data yang dibentuk, mebebaskan *programmer* untuk bekerja menggunakan *business logic*, jika suatu *web form* menggandung data yang direlasikan ke suatu objek bisnis, *Active Record* dapat mengekstraknya ke dalam *Model*. *Active Record* mendukung validasi data *Model* sehingga jika bentuk data tidak sesuai dengan validasi, View pada *RoR* dapat menampilkan dan membentuk pola error hanya dengan satu baris kode saja.

2.3.2.2.3.2 Action Pack



Gambar 2.7 Action Pack

Komponen ini dapat merupakan jantung dari *framework* RoR. Terdiri atas dua sub-komponen:

a. **ActionController**, modul ini memastikan pemakaian aksi yang tepat berdasarkan *web-request* setiap pengguna. Aksi ini diambil berdasarkan URL; semisal

/books/list akan memanggil metode 'list' pada *BookController-class*.

b. **ActionView**, komponen ini digunakan oleh ActionPack untuk mengembalikan *request-view*. ActionView menggunakan *template* RHTML, RJS dan RXML. *Template* ini dapat digunakan untuk men-generate HTML/JS/XML.

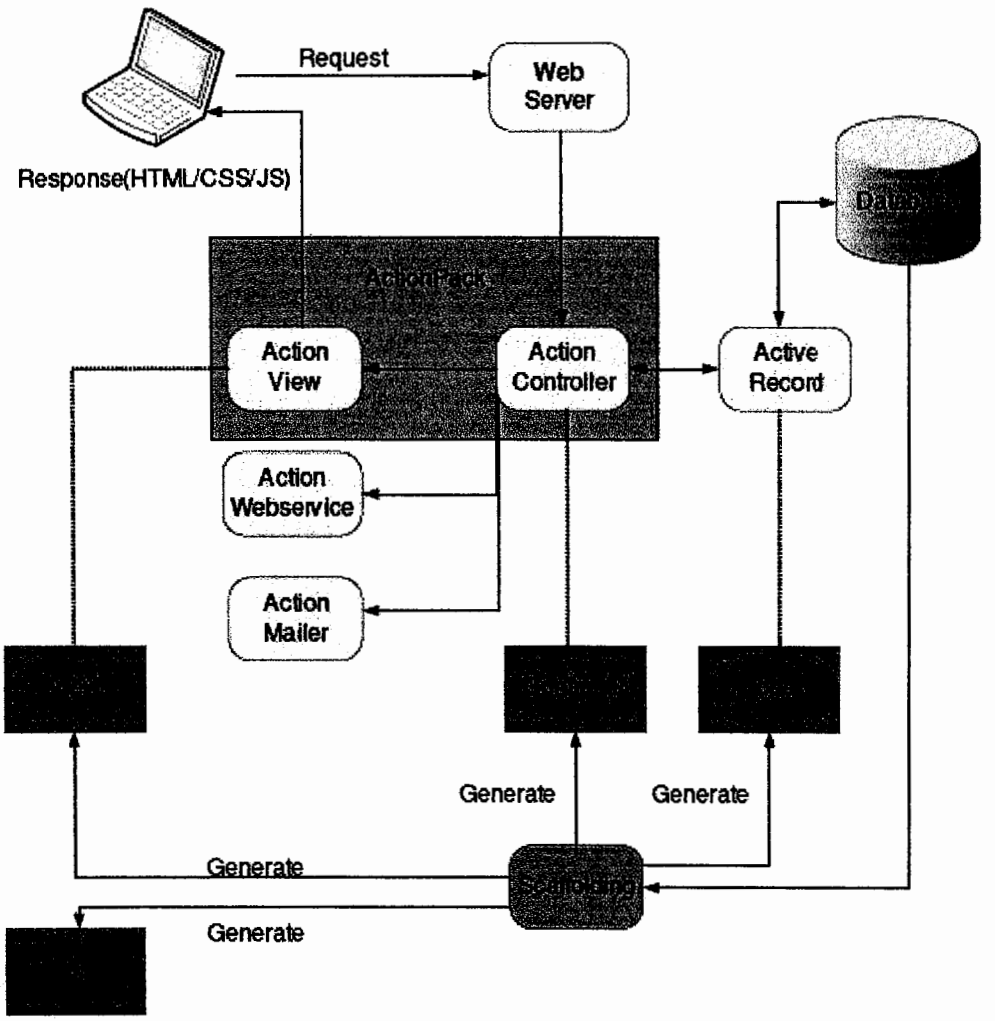
2.3.2.2.3.3 ActionMailer

Disediakan RoR untuk komunikasi melalui email.

2.3.2.2.3.4 ActionWebService

Dukungan RoR terhadap XML-RPC dan SOAP *webservice*.

2.3.2.2.3.5 Aliran Aplikasi RoR



Gambar 2.8 Ilustrasi aliran applikasi RoR menggunakan komponen-komponen di atas.

2.3.2.2.3.6 Scaffolding

Scaffolding digunakan untuk men-*generate* sintak secara cepat untuk *layout* termasuk Model, View, Controller dan Test class berdasarkan entitas database. Model *generate* sintak adalah model *write-only*. Ketika men-*generate* sintak untuk suatu entitas database,

generator akan mengkonfirmasi semua file yang akan *overwrite*. Hal ini memungkinkan untuk men-*generate* hanya sebagiannya untuk mempercepat proses pengembangan.

2.3.2.2.4 *Plugin*

Plugin adalah tambahan modul program dari RoR yang bertujuan untuk mempermudah pembuatan aplikasi. Plugin di-*install dengan perintah ruby script/plugin install [nama_plugin]* dan di-*remove* dengan perintah *ruby script/plugin remove [nama_plugin]*. *Plugin* diletakan di dalam direktori vendor. Plugin diunduh di SVN atau GITHUB.

2.4 *CSS (Cascading Style Sheet)*

2.4.1 Pengertian *CSS (Cascading Style Sheet)*

Teknologi dibalik *style sheets* disebut CSS. CSS adalah singkatan dari *Cascading Style Sheets*. CSS adalah sebuah bahasa yang mendefinisikan konstruksi sebuah style seperti *fonts*, warna dan posisi yang mana biasa digunakan untuk menggambarkan bagaimana informasi pada halaman *web* di format dan ditamplikan. CSS digunakan dalam kode HTML untuk menciptakan kumpulan *style* yang akan memperluas kemampuan HTML.

Dapat diartikan pula bahwa CSS adalah "gaya" dalam pemrograman HTML yang meminimalkan penulisan tag yang berulang-ulang. CSS style dapat disimpan di halaman *web* HTML atau pada file style sheet terpisah. *Style sheet* mengandung perintah-perintah tentang style, dimana perintah-perintah tersebut akan mengaplikasikan style

pada elemen-elemen tipe yang diiberikan. Ketika digunakan secara terpisah, maka perintah-perintah dalam style sheet ditempatkan pada external dokumen style sheet dengan extention file *.css*. *Style rule* adalah sebuah instruksi tentang formatting yang dapat diaplikasikan kepada elemen dalam halaman *web*, seperti paragraf text atau sebuah *link*. *Style rule* terdiri dari satu atau lebih properti dan property tersebut mengasosiasikan nilai - nilai. *Internal stlye sheet* ditempatkan langsung di dalam sebuah halaman *web*, sebaliknya *external style* sheet terdapat pada dokumen yang terpisah dan membutukan link yang sederhana untuk menuju pada halaman *web* melalui *"tag"* khusus. *"Cascading"* adalah sebuah bagian dari singkatan CSS ysng menunjuk pada cara dimana perintah-perintah dalam style sheet diaplikasikan pada elemen-elemen di dokumen HTML. Dengan *Style Sheet* dapat mengontrol seluruh *layout* dari *web* site , dan jika mengubah tampilan dari *web* site cukup dengan memodifikasi *style sheet*.

2.4.2 Kategori Style Sheets

Terdapat 3 ketegori style sheets :

a. *Inline Style Sheet* (di dalam elemen HTML)

Inline Style Sheet dilakukan dengan menambahkan attribut style kedalam tag HTML-nya. Berlaku hanya pada tag dimana style ditambahkan.

Contoh :

```
<p style="font-size: 25pt;
fontweight: bold; font-style:italic;
color:red;">
</p>
```

b. Emmbedded Style Sheet (di dalam dokumen HTML)

Emmbedded Style Sheet diletakan pada tag <head>. Berlaku hanya untuk halaman dimana style didefinisikan.

Contoh :

```
<head>
<title> Example Page </title>
<style type="text/css">
h1 { font-size: 16pt; font-weight:bold;}
p {color:blue;}
</style>
</head>
```

c. *External Style Sheet* (di file external)

External Style Sheet diletakan diluar dokumen HTML-nya. Menggunakan properti <link> untuk menghubungkan halaman HTML-nya
dengan file .css nya.

Contoh :

```
<link href="myfreetemplates.css"
rel="stylesheet" type="text/css" />
```