BAB II

TINJAUAN PUSTAKA

2.1 Sistem Informasi

Telah diketahui bahwa informasi merupakan hal yang sangat penting bagi manajemen di dalam pengambilan keputusan. Informasi dapat diperoleh dari Sistem Informasi (*information system*) atau disebut juga *processing system.*

Untuk menghasilkan sebuah informasi, maka dibutuhkan sistem yang disebut Sistem Informasi. Beberapa definisi Sistem Informasi dari beberapa sumber sebagai berikut:

> *"Sistem Informasi dapat didefinisikan sebagai seperangkat komponen yang saling terkait bekerja sama untuk mengumpulkan atau mengambil, memproses, menyimpan, dan menyebarkan informasi untuk mendukung pengambilan keputusan, koordinasi, dan pengendalian dalam suatu organisasi"* (Laudon, 2010);
>
> *"Sistem Informasi merupakan aktifitas mengumpulkan, proses, menyimpan, menganalisis, dan menyebarkan informasi untuk tujuan tertentu"* (Rainer, 2011);
>
> *"Sistem Informasi merupakan kumpulan prosedur organisasi yang pada saat dilaksanakan akan memberikan informasi bagi pengambil keputusan dan atau untuk mengendalikan informasi"* (Ladjamudin, 2005);

2.1.1 Komponen Sistem Informasi

Untuk mendukung lancarnya suatu Sistem Informasi dibutuhkan beberapa komponen yang fungsinya sangat vital di dalam Sistem Informasi. Komponen-komponen Sistem Informasi tersebut adalah sebagai berikut (Jogiyanto, 2003):

1) *Input* adalah semua data yang dimasukan ke dalam Sistem Informasi. Dalam hal ini yang termasuk dalam *input* adalah dokumen, formulir dan *file*.

2) Dokumen tersebut dikumpulkan dan dikonfirmasikan ke suatu bentuk sehingga dapat diterima oleh pengolah yang meliputi : pencatataan, penyimpanan, pengujian dan pengkodeaan.
3) Proses merupakan kumpulan prosedur yang akan memanipulasi input yang kemudian akan disimpan dalam bagian basisdata dan seterusnya akan diolah menjadi suatu *output* yang akan digunakan oleh si penerima. Komponen ini dalam tugasnya akan mengubah segala masukkan menjadi keluaran yang terdiri dari : manusia, metode dan prosedur, peralatan komputer dan penyimpanan data.
4) *Output* merupakan semua keluaran atau hasil dari *model* yang sudah diolah menjadi suatu informasi yang berguna dan dapat dipakai penerima. Komponen ini akan berhubungan langsung dengan pemakai Sistem Informasi.
5) Teknologi merupakan bagian yang berfungsi untuk memasukan input, mengolah input dan menghasilkan keluaran. Ada tiga bagian dalam teknologi ini yang meliputi perangkat keras, perangkat lunak dan perangkat manusia.
6) Basisdata merupakan kumpulan data yang saling berhubungan yang disimpan dalam perangkat keras komputer dan akan diolah menggunakan perangkat lunak.
7) Kendali merupakan semua tindakan yang diambil untuk menjaga Sistem Informasi tersebut agar bisa berjalan dengan lancar dan tidak mengalami gangguan.

2.1.2 Manfaat Sistem Informasi

Sistem Informasi dapat memberikan manfaat bagi suatu organisasi, perusahaan dan lainnya. Adapun manfaat dari Sistem Informasi adalah sebagai berikut (Jogiyanto, 2003) :

1) Organisasi menggunakan Sistem Informasi untuk mengolah transaksi, mengurangi biaya dan menghasilkan pendapatan sebagai salah satu pelayanan mereka.
2) Bank menggunakan Sistem Informasi untuk mengolah cek-cek nasabah dan membuat berbagai laporan rekening koran dan transaksi yang terjadi.
3) Perusahaan menggunakan Sistem Informasi untuk mempertahankan persediaan pada tingkat paling rendah agar konsisten dengan jenis barang yang tersedia.

2.1.3 Pemakai Sistem Informasi

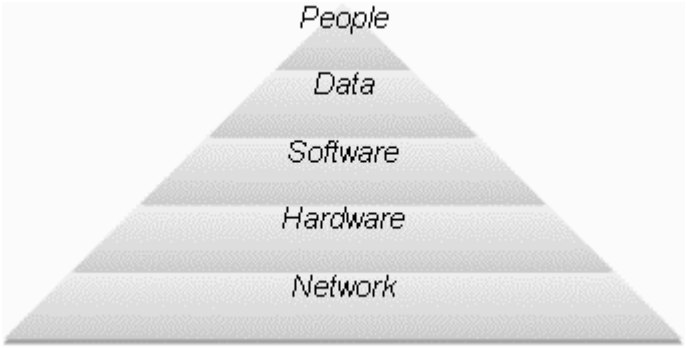
Sebagian besar Sistem Informasi berlandaskan komputer terdapat di dalam suatu organisasi dalam berbagai jenis. Anggota organisasi adalah pemakai informasi yang dihasilkan sistem tersebut termasuk manajer yang bertanggungjawab atas pengalokasian sumber daya untuk pengembangan dan pengoperasian perusahaan.

Tetapi bukan hanya manajer yang memperoleh manfaat dari semua ini. *Non* manajer dan staf ahli juga menggunakan *output*-nya. Para pemakai Sistem Informasi juga termasuk yang berada di luar organisasi.

2.1.4 Sumber Daya Sistem Informasi

Sumberdaya informasi tidak hanya sekedar data dan informasi, melainkan mencakup pula perangkat keras, *peripheral*, perangkat lunak, para spesialis informasi dan para pemakai informasi.

Secara umum Sistem Informasi merupakan kombinasi dari orang (*people*), perangkat keras (*hardware*), perangkat lunak (*software*), jaringan komunikasi (*communications network*) dan sumber data yang dihimpun, ditransformasi dan mengalami proses peangaliran dalam suatu organisasi (Kristanto, 2008).



Gambar 2.1 : Sumber Daya Sistem Informasi
(Kristanto, 2008)

2.1.5 Peranan Sistem Informasi Dalam Bisnis

Sistem Informasi memiliki 3 (tiga) peran yang vital dalam suatu bisnis, antara lain sebagai berikut (Kristanto, 2008) :

1) Mendukung Operasi Bisnis

   Kemampuan Sistem Informasi untuk menggabung dan mengintegrasikan informasi menjadi faktor pendukung operasi bisnis.

2) Mendukung dalam Pengambilan Keputusan Manajerial

Sistem Informasi dapat membantu seorang manajer untuk mengidentifikasi dan mengevaluasi hasil dari keputusan sebelumnya. Sistem Informasi membantu manajer agar dapat membuat keputusan dengan lebih baik, lebih cepat dan lebih banyak informasi yang dihasilkan.

3) Meraih Keuntungan Strategik

Sistem Informasi di desain untuk suatu tujuan yang strategik yang akan membantu perusahaan/organisasi dalam meraih keuntungan pasar yang kompetitif.



Gambar 2.2 : Peranan Sistem Informasi
(Kristanto, 2008)

2.1.6 Jenis-jenis Sistem Informasi

Ditinjau dari aplikasinya dan penggunaan dalam berbagai bidang, Sistem Informasi dapat dibagi menjadi beberapa bagian, yaitu (Kristanto, 2008) :

1) *Routine Processing System* (RPS)

Routine Processing System digunakan untuk melayani berbagai kebutuhan yang telah terdefinisi dan terjadwal secara rutin.

2) *Decision Support System* (DSS)

Decision Support System digunakan untuk melayani kebutuhan yang tidak dapat didefinisikan dengan baik dan biasanya terjadi pada saat perancangan.

3) *Classical Management Information System* (CMIS)

   Classical Management Information System digunakan untuk melayani kebutuhan pembuatan pelaporan kegiatan yang telah terjadwal dan terdefinisi dengan baik.

4) *Real Time Information System* (RTIS)

   Real Time Information System digunakan untuk melayani kegiatan yang mempunyai sifat harus direspon dengan cepat.

5) *Distributed Data Processing System* (DDPS)

   Distributed Data Processing System digunakan untuk melayani kebutuhan yang tersebar secara geografis dengan sumber daya yang tersebar.

6) *Transaction Processing System* (TPS)

   Transaction Processing System digunakan untuk melayani kegiatan yang bersifat transaksional yaitu membawa perubahan terhadap kondisi sistem yang ada.

2.2 Arsitektur *Enterprise*

2.2.1 *Enterprise*

Berikut beberapa pengertian tentang *enterprise*:

1) Organisasi yang mendukung lingkungan bisnis dan misi yang telah ditetapkan.
2) Berfungsinya keseluruhan komponen organisasi yang beroperasi di bawah *control* dari organisasi.

Enterprise bukan hanya organisasi yang berorientasi pada laba/keuntungan (*profit oriented*) tetapi juga organisasi nirlaba seperti institusi pendidikan. *Enterprise* dapat berupa organisasi secara utuh atau bagian dari organisasi tersebut. (Foorthuis, 2012*)*

2.2.2 Arsitektur (*Architecture*)

Pengertian arsitektur di sini tidak hanya terbatas pada pengertian umum yang berhubungan konstruksi fisik, tetapi juga pada konteks bisnis dan arsitektur untuk rekayasa perangkat lunak, berikut beberapa pengertian yang berhubungan arsitektur (Foorthuis, 2012) :

1) Arsitektur (*Architecture*) merupakan komponen-komponen sebuah sistem yang terdiri dari jaringan, perangkat keras dan lunak yang distrukturkan.
2) Rancangan keseluruhan jenis konstruksi baik fisik maupun konteks, nyata atau maya.
3) Dari pengertian di atas dapat diambil suatu kesimpulan bahwa arsitektur pada dasarnya menggambarkan bentuk konstruksi sistem yang diwujudkan dalam sebuah model (*blueprint*) yang dilihat dari beberapa sudut pandang.

2.2.3 Arsitektur Informasi (*Information Architecture*)

Merupakan pengorganisasian sejumlah data yang digunakan atau dihasilkan oleh organisasi yang berhubungan dengan tujuan bisnis organisasi. Arsitektur informasi merupakan sebuah representasi grafis dari perencanaan sumber daya untuk kebutuhan bisnis. Arsitektur informasi juga merupakan sebuah

blue print dimana Sistem Informasi saat ini dan yang akan datang dikembangkan (Foorthuis, 2012).

Arsitektur informasi adalah sebuah sistem yang mengelola data serta penerapan dari proses bisnis yang telah didefinisikan, sehingga sebelum organisasi mendefinisikan kebutuhan informasi yang akan digunakan untuk menjalankan roda organisasinya, maka terlebih dahulu harus mendefinisikan data, proses bisnis dan sistem aplikasinya (Foorthuis, 2012).

2.2.4 Definisi Arsitektur *Enterprise* (*Enterprise Architecture*)

Arsitektur *enterprise* merupakan sebuah cara untuk menyusun elemen-elemen Sistem Informasi enterprise dimana bisa merupakan sekumpulan model dan hubungan antar elemen *enterprise* yang digunakan dalam merencanakan, mendesain, dan merealisasikan suatu struktur *enterprise*, proses bisnis, Sistem Informasi, dan infrastruktur yang terkait di dalamnya (Hadiana, 2013).

Beberapa definisi lain dari *Enterprise Architecture* antara lain:

1) *Enterprise Architecture* adalah perangkat primer yang digunakan untuk mempersingkat waktu penilaian pengaruh, analisa *tradeoff*, perubahan arah rencana strategis dan reaksi taktis. *Enterprise Architecture* terdiri dari dokumen seperti gambar, diagram, dokumen tekstual, model dan menggunakan berbagai metode bisnis yang menjelaskan seperti apa Sistem Informasi yang diperlukan oleh organisasi (Surendro, 2007).
2) Representasi deskriptif (*model*) yang relevan untuk menggambarkan sebuah *enterprise* dan apa saja yang harus dihasilkan guna memenuhi kebutuhan manajemen atau organisasi (Foorthuis, 2012).

3) *Blueprint* pemetaan hubungan antar komponen dan semua orang yang bekerja di dalam perusahaan secara konsisten untuk meningkatkan kerja sama/kolaborasi, serta koordinasi diantaranya (Ward, 2002).
4) Mekanisme untuk memastikan sumber daya teknologi informasi suatu organisasi dapat sejalan dengan strategi dari organisasi tersebut (Foorthuis, 2012).

2.2.5 Keuntungan Arsitektur *Enterprise*

Salah satu tujuan dari penerapan arsitektur *enterprise* adalah menciptakan keselarasan antara bisnis dan teknologi informasi bagi kebutuhan organisasi.

Terdapat beberapa keuntungan dari arsitektur *enterprise*, antara lain (Esmaeil, 2012) :

1) Dapat menangkap fakta tentang misi, fungsi, dan landasan bisnis dalam bentuk yang dipahami untuk mendukung perencanaan dan pengambilan keputusan.
2) Dapat memperbaiki komunikasi di antara organisasi Teknologi Informasi dan bisnis dalam perusahaan menggunakan kosakata standar.
3) Fokus pada penggunaan strategi dari teknologi untuk pengelolaan informasi perusahaan yang baik dan meningkatkan konsistensi, akurasi, tepat waktu, integritas, kualitas, ketersedian, akses, dan berbagi informasi pengelolaan Teknologi Informasi .
4) Mencapai skala ekonomi dengan menyediakan mekanisme berbagi layanan di seluruh bagian perusahaan.
5) Mempercepat integrasi sistem yang eksis, migrasi, dan yang baru.

6) Memastikan pemenuhan hukum dan regulasi.

2.2.6 Fungsi Arsitektur *Enterprise*

Arsitektur E*nterprise* secara fundamental merupakan sebuah evaluasi dan gambaran tentang orang, proses, dan sumber daya. Pendekatan arsitektur *enterprise* didasarkan pada teori-teori organisasi dan sistem sehingga diketahui bagaimana *enterprise* distrukturkan dan bagaimana fungsi sistem dan aktivitas dalam *enterprise*.

Terdapat beberapa fungsi dari arsitektur *enterprise*, antara lain adalah sebagai berikut (Esmaeil, 2012) :

1) Menjabarkan hubungan / kaitan antara tujuan organisasi dengan Sistem Informasi dan komunikasi.
2) Mendukung pengambilan keputusan investasi.
3) Memanfaatkan teknologi informasi dan komunikasi untuk mendukung aktivitas operasi organisasi sekaligus upaya mengurangi redudansi antara Sistem Informasi dan komunikasi, menggunakan kembali informasi dan komponen software, pemilihan solusi dan teknologi baru secara efektif.
4) Meningkatkan kemampuan integrasi data antar bagian dalam organisasi :
    a) Pengembangan standar-standar dalam Sistem Informasi dan komunikasi.
    b) Mengurangi jumlah antarmuka antar aplikasi.

2.2.7 Faktor-Faktor Arsitektur *Enterprise*

Beberapa faktor arsitektur *enterprise* memberikan standar penting yang terdiri dari kumpulan standar proses informasi *enterprise*. Standar proses

informasi *enterprise* merupakan acuan dari perspektif bisnis yang akan dikembangkan.

Faktor-faktor yang menjadi alasan sebuah organisasi mengembangkan Arsitektur *Enterprise* (Esmaeil , 2012) :

1) *Alignment*/Keselarasan

   Pengembangan tersebut selaras dengan tujuan dan keinginan organisasi.

2) *Integration/Integrasi*

   Pemanfaatan sumber daya informasi sesuai dengan standar dan dikelola oleh organisasi secara konsisten dan menyeluruh.

3) *Change/Perubahan*

   Mengelola dan mengantisipasi perubahan dari semua aspek orgnisasi.

4) *Time-to-Market*

   Meminimalisasi pengembangan sistem, pembuatan program, pembaharuan *frame time,* dan kebutuhan sumber daya.

5) *Convergence*/Fokus tujuan

   Mengarah kepada standarisasi produk teknologi informasi seperti: *Technial Reference Model* (TRM).

2.2.8 Prinsip-Prinsip Arsitektur *Enterprise*

Prinsip arsitektur *enterprise* digunakan sebagai ukuran dalam menilai keberhasilan dari pengembangan arsitektur *enterprise* oleh organisasi.

Berikut ini adalah penjelasan prinsip-prinisip tersebut (Surendro, 2009):

1) Prinsip *Enterprise*

Pengembangan arsitektur yang dilakukan diharapkan mendukung seluruh bagian organisasi, termasuk unit-unit organisasi yang membutuhkan.

2) Prinsip Teknologi Informasi

   Lebih mengarahkan konsistensi penggunaan Teknologi Informasi pada seluruh bagian organisasi, termasuk unit-unit organisasi yang akan menggunakan.

3) Prinsip Arsitektur

   Merancang arsitektur sistem berdasarkan kebutuhan proses bisnis dan bagaimana mengimplementasikannya.

2.2.9 Tujuan Manajemen Arsitektur *Enterprise*

Beberapa tujuan dari manajemen arsitektur *enterprise* antara lain adalah sebagai berikut (Esmaeil 2012) :

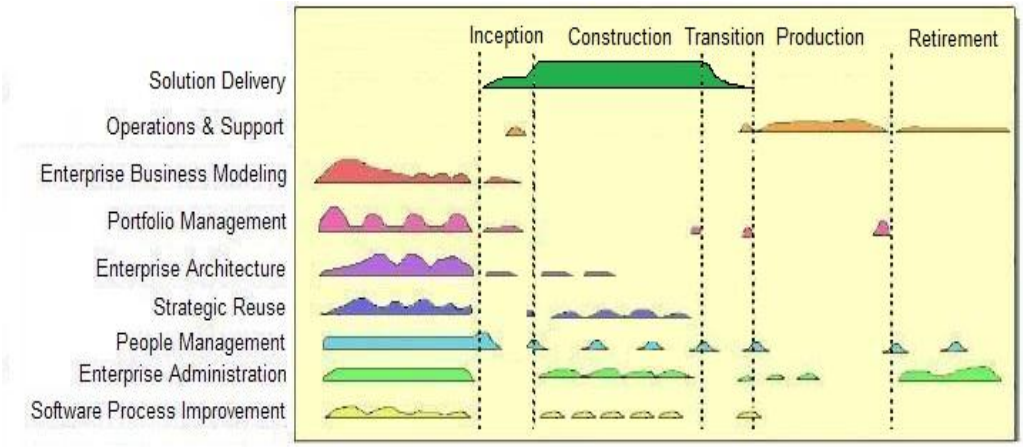
1) Strategi dan orientasi Bisnis, merupakan *enabler* yang menyediakan model bisnis yang baru dengan memperhatikan pengaruh Teknologi Informasi terhadap bisnis.
2) Kemampuan beradaptasi dan berkesinambungan terhadap perubahan organisasi, dan pertumbuhan dalam pasar, bisnis, dan teknologi yang dinamis, yang berkelanjutan dengan prinsip dan struktur arsitektur yang terus berlaku.
3) Efisiensi dan Efektif, strategi berbasis arsitektur dengan berorientasi sukses untuk mengembangkan dan mengimplementasikan Teknologi Informasi dengan pengaruh yang kuat dan menjamin pemenuhanan standar organisasi.
4) Transparansi dan Komunikasi, melibatkan sejumlah orang dengan komposisi yang berbeda dari manajemen hingga ke pakar Teknologi Informasi .

2.3 Pendekatan Teknik dan Metodologi

2.3.1 Metode *Enterprise Unified Process* (EUP)

Proses perangkat lunak yang efektif sangat penting bagi keberhasilan sebuah organisasi. *Unified Process* (UP) adalah suatu kerangka kerja dari proses perangkat lunak yang *instantiated*, baik *Rational Unified Process* (RUP) maupun *Enterprise Unified Process* (EUP).

Ruang lingkup dari RUP adalah dari pengembangan sistem/*software*, sedangkan EUP perluasan dari RUP untuk menjadi siklus penuh teknologi informasi yang meliputi siklus hidup sistem secara lengkap (multi sistem) dengan menambahkan dua fase baru yaitu *production* dan *retirement* serta menambahkan tujuh *enterprise discipliness* (Ambler, 2005).



Gambar 2.3 : *The Lifecycle Enterprise Unified Process* (EUP)
(Ambler, 2005)

2.3.2 Fase-Fase EUP

Fase-fase dalam EUP terdiri dari enam fase (Ambler, 2005), yaitu :

1) ***Inception Phase***, Tahapan ini merupakan tahapan paling awal dimana aktivitas penilaian terhadap sebuah proyek perangkat lunak dilakukan.

Tujuannya adalah untuk mendapatkan kesepakatan dari *stakeholder* sehubungan dengan tujuan dan dana proyek. Tahap ini juga digunakan untuk mendapatkan gambaran umum kebutuhan, persyaratan dan fungsi-fungsi utama perangkat lunak.

2) ***Constraction Phase***, Tujuan dari tahapan ini adalah membangun perangkat lunak sampai dengan saat perangkat lunak tersebut siap digunakan. Titik berat tahapan ini adalah pada penentuan tingkat prioritas kebutuhan / persyaratan, melengkapi spesifikasinya, analisis lebih dalam, disain solusi yang memenuhi kebutuhan dan persyaratan, pengkodean dan pengujian perangkat lunak. Jika dimungkinkan versi awal dari perangkat lunak diuji cobakan untuk mendapatkan masukan dari pengguna.
3) ***Transition Phase***, Tahap ini difokuskan pada bagaimana menyampaikan perangkat lunak yang sudah jadi pada pengguna. Perangkat lunak akan secara resmi diuji oleh baik oleh penguji (*tester*) yang kompeten maupun oleh pengguna. Beberapa aktivitas seperti pemindahan pusat data dan pelatihan pengguna dan staf pendukung harus dilakukan pada tahap ini.
4) ***Production Phase***, fase ini dilakukan setelah melakukan *deployment* terhadap sistem untuk memastikan bahwa sistem berjalan dengan baik (*monitoring*), mengelola permintaan perubahan dan perbaikan sistem jika terjadi kerusakan.
5) ***Retirement Phase***, fase terakhir ini mengenai penghapusan sistem yang lama yang sudah tidak terpakai dan digantikan dengan sistem baru sehingga tidak terjadi penumpukan pengoperasian terhadap pengguna sistem.

2.3.3 Disiplin EUP

EUP mencakup tujuh disiplin *enterprise management* yang mendukung *cross-system* untuk menjadi kunci kesuksesan teknologi informasi (Ambler, 2005).

Berikut ini tujuh disiplin *enterprise management* EUP :

1) ***Enterprise Business Modeling***, disiplin ini untuk mengeksplorasi struktur dan proses bisnis perusahaan. Ini memberikan pemahaman umum *business activities*, *customer*, dan *supplier* bisnis. *Enterprise business modeling* membantu mengidentifikasi masalah dan sistem yang akan diotomatiskan. Disiplin ini memiliki komponen untuk menjelaskan setiap fasenya, yaitu :
   a) *Enterprise Vision* adalah sebuah pernyataan dari tujuan utama dari sebuah organisasi.
   b) *Enterprise Mission Statement* adalah sebuah pernyataan dari strategi yang harus diikuti untuk mencapai *enterprise vision*.
   c) *Enterprise Business Process Model* adalah menangkap proses bisnis fundamental, entitas dan alur kerja
2) ***Portfolio Management***, disiplin ini untuk membantu meningkatkan efisiensi dan efektivitas Teknologi Informasi dalam suatu organisasi dengan memilih dan kemudian mengelola proyek pengembangan perangkat lunak yang layak. Pada disiplin ini terdapat *Inventory Current System* yaitu kegiatan menginventarisasi sistem saat ini yang ada di perusahaan. Tujuan kegiatan ini adalah untuk mengetahui sistem yang sudah ada dalam perusahaan.
3) ***Enterprise Architecture***, disiplin ini membahas masalah arsitektur keseluruhan yang terkait dengan organisasi, membantu memastikan

konsistensi sistem dan aplikasi untuk memudahkan upaya arsitektur. Disiplin ini memeliki beberapa *workflow* diantaranya :

a) *Define Architectural Requirements* adalah mendefinisikan kebutuhan arsitektur perusahaan meliputi arsitektur data, aplikasi dan teknologi.

b) *Define Candidate Architecture* adalah untuk mengidentifikasi dan kemudian memvalidasi kandidat entitas data, kandidat aplikasi dan kandidat teknologi yang merupakan blok pembangunan arsitektur yang akan dibangun perusahaan.

4) ***Strategic Reuse***, disiplin ini mempromosikan pengembangan dan penggunaan kembali aset Teknologi Informasi yang sudah ada, tujuannya adalah untuk memungkinkan pengembangkan aplikasi yang lebih efisien dalam hal penggunaan komponen yang dapat digunakan bersama.

5) ***People Management***, disiplin ini menjelaskan proses pengorganisasian pelatihan, pemantauan, dan mengarahkan orang-orang untuk keberhasilan proyek-proyek dalam organisasi.

6) ***Enterprise Administration***, disiplin ini menentukan bagaimana suatu organisasi menciptakan, memelihara, mengelola data aset fisik dan menyebarkan informasi.

7) ***Software Process Improvement***, disiplin ini menjelaskan kebutuhan metode pengembangan *software* yang diusulkan. Tujuannya adalah untuk sinkronisasi dengan perencanaan arsitektur *enterprise*.

2.3.4 Keuntungan EUP

Ada beberapa keuntungan pengembangan arsitektur *enterprise* dengan menggunakan pendekatan EUP yaitu (Ambler, 2005) :

1) Meningkatkan kualitas Teknologi Informasi pada suatu perusahaan dengan cara melakukan *cross-system* terhadap semua kebutuhan dengan manambahkan disiplin *Enterprise Management* berupa *Enterprise Business Modelling, Portfolio Management, Enterprise Architecture, Strategic Reuse, People Management, Enterprise Administration* dan *Software Process Improvement*.
2) Pendekatan EUP mendukung secara eksplisit pengembangan infrastruktur multisistem seperti pengembangan organisasi secara luas.
3) *Risk focused*, EUP memungkinkan para pengembang arsitektur *enterprise* untuk bisa mengetahui risiko vital di setiap awal tahapan pengerjaan. Dengan demikian faktor-faktor yang sekiranya mempunyai resiko yang paling vital bisa lebih mendapat perhatian terlebih dahulu, sehingga nantinya tidak mengganggu proses pengembangan arsitektur *enterprise* selanjutnya. Selain itu EUP juga menghendaki agar risiko di setiap fase bisa segera diselesaikan pada fase itu juga, sehingga tidak menghambat fase-fase selanjutnya.
4) Meningkatkan produktivitas *end-user* dengan menambahkan fase baru yaitu *production* dan *retirement*. Pada fase tersebut dilakukan *monitoring* terhadap sistem dan penghapusan sistem yang tidak terpakai sehingga dilakukan *software process improvement.*
5) Perubahan *user requirement* lebih mudah diterapkan. Hal ini terdapat pada *fase transition*.

2.4 Pemodelan Bisnis (*Business Modeling*)

Pemodelan bisnis (*business modeling*) adalah studi tentang organisasi. Ketika sedang melakukan pemodelan bisnis, kita sedang menguji struktur

organisasi, memperhatikan peranan-peranan di dalam organisasi, dan bagaimana mereka terhubungkan satu dengan lainnya. Juga menguji aliran kerja (*workflow*) dalam organisasi, proses utama di dalam organisasi, bagaimana mereka bekerja, seberapa efektif dan efisien cara kerja yang mereka lakukan. Demikian juga akan dilakukan pengujian entitas yang ada di luar organisasi, individu atau perusahaan lain yang saling berhubungan dengan bisnis organisasi (Sholiq, 2006).

Singkatnya, pemodelan bisnis mencoba memahami apa yang ada di dalam dan di luar bisnis, dan bagaimana yang ada di dalam dan di luar organisasi berkomunikasi satu sama lain. Informasi-informasi akan didokumentasikan ke suatu model bisnis.

Pertimbangan untuk melakukan pemodelan bisnis, antara lain:

1) Untuk memperoleh pemahaman yang lebih baik tentang visi organisasi dan sistem perangkat lunaknya.
2) Sebagai sarana melakukan rekayasa ulang proses bisnis organisasi.
3) Sebagai sarana pelatihan tentang proses bisnis organisasi

Adapun konsep pemodelan bisnis antara lain (Sholiq, 2006) :

1) *Actor* bisnis (*business actor*)
2) Pekerja bisnis (*business worker*)
3) *Use-case* bisnis (*business use-case*)
4) Relasi Assosiasi (*Association*) dan Generalisasi (*Generalization*)
5) Entitas bisnis (*business entity*)
6) Diagram *use-case* bisnis (*business use-case diagram*)
7) Diagram aktivitas (*activity diagram*)
8) Unit Organisasi (*organization unit*)

2.4.1 UML (*Unified Modelling Language*)

UML adalah keluarga notasi grafis yang didukung oleh *meta-model* tunggal, yang membantu pendeskripsian dan desain sistem perangkat lunak, khususnya sistem yang dibangun menggunakan pemrograman berorientasi objek. (Ali, 2003).

UML menjadi bahasa yang bisa digunakan untuk berkomunikasi dalam perspektif objek antara *user* dengan *developer*, antara *developer* dengan *developer*, antara *developer* analisis dengan *developer* disain, dan antara *developer* disain dengan *developer* pemrograman. UML lahir dari penggabungan banyak bahasa pemodelan grafis berorientasi objek yang berkembang pesat pada akhir 1980-an dan awal 1990-an. UML memungkinkan *developer* melakukan permodelan secara *visual*, yaitu penekanan pada penggambaran, bukan didominasi oleh narasi. Permodelan *visual* membantu untuk menangkap struktur dan kelakuan dari objek, mempermudah penggambaran interaksi antara elemen dalam sistem, dan mempertahankan konsistensi antara rancangan dan implementasi dalam pemrograman.

Diagram berbentuk grafik yang menunjukkan simbol elemen model yang disusun untuk mengilustrasikan bagian atau aspek tertentu dari sistem. Sebuah diagram merupakan bagian dari suatu *view* tertentu dan ketika digambarkan biasanya dialokasikan untuk *view* tertentu. Adapun jenis-jenis diagram ditunjukkan pada Tabel 2.1

Tabel 2.1 Jenis-jenis Diagram dalam UML

No	Nama	Keterangan
1.	*Use Case Diagram*	Menggambarkan sejumlah *external actors* dan hubungannya ke *use case* yang diberikan oleh sistem. *Use case* adalah deskripsi fungsi yang

No	Nama	Keterangan
		disediakan oleh sistem dalam bentuk teks sebagai dokumentasi dari *use case* symbol namun dapat juga dilakukan dalam *activity diagrams*.
2.	*Class Diagram*	Menggambarkan struktur *statis class* di dalam sistem. *Class* merepresentasikan sesuatu yang ditangani oleh sistem. *Class* dapat berhubungan dengan yang lain melalui berbagai cara yaitu *associated* (terhubung satu sama lain), *dependent* (satu *class* tergantung/menggunakan *class* yang lain), *specialed* (satu *class* merupakan spesialisasi dari *class* lainnya), atau *package* (grup bersama sebagai satu unit). Sebuah sistem biasanya mempunyai beberapa *class diagram.*
3	*State Diagram*	Mendeskripsikan implementasi dan ketergantungan modul. Komponen yang merupakan tipe lainnya dari *code module* diperlihatkan dengan struktur dan ketergantungannya juga alokasi sumber daya komponen dan informasi administrative lainnya. *View* ini digambarkan dalam *component view* dan digunakan untuk pengembang (*developer*).
4.	*Sequence Diagram*	Menggambarkan semua *state* (kondisi) yang dimiliki oleh suatu objek dari suatu *class* dan keadaan yang menyebabkan *state* berubah. Kejadian dapat berupa objek lain yang mengirim pesan. *State class* tidak digambarkan untuk semua *class*, hanya yang mempunyai sejumlah *state* yang terdefinisi dengan baik dan kondisi *class* berubah oleh *state* yang berbeda.
5.	*Collaboration Diagram*	Menggambarkan kolaborasi dinamis antara sejumlah objek. Kegunaanya untuk menunjukkan rangkaian pesan yang dikirim antara objek juga interaksi antara objek, sesuatu yang terjadi pada titik tertentu dalam eksekusi sistem.
6.	*Activity Diagram*	Menggambarkan rangkaian aliran dari aktivitas, digunakan untuk mendeskripsikan aktifitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktifitas lainnya seperti *use case* atau interaksi.
7.	*Component Diagram*	Menggambarkan struktur fisik kode dari komponen. Komponen dapat berupa *source code*, komponen *biner*, atau *executable component*. Sebuah komponen berisi informasi

No	Nama	Keterangan
		tentang *logic class* atau *class* yang diimplementasikan sehingga membuat pemetaan dari *logical view* ke *component view*.
8.	*Deployment Diagram*	Menggambarkan arsitektur fisik dari perangkat keras dan perangkat lunak sistem, menunjukkan hubungan komputer dengan perangkat (*nodes*) satu sama lain dan jenis hubungannya. Di dalam *nodes, executeable component* dan objek yang dialokasikan untuk memperlihatkan unit perangkat lunak yang dieksekusi oleh *node* tertentu dan ketergantungan komponen.

Tujuan utama dalam mendesain UML (Ali, 2003) :

1) Menyediakan bagi *users*, bahasa pemodelan *visual* yang ekspresif, yang siap digunakan, sehingga *users* dapat membangun dan mengembangkan model yang berarti.
2) Menyediakan mekanisme yang luas dan spesifik untuk menyampaikan konsep utama.
3) Tidak terikat dalam menentukan bahasa pemograman dan proses pengembangan.
4) Menyediakan sebuah landasan formal dalam memahami bahasa pemodelan.
5) Mendukung konsep pengembangan *level* tinggi.
6) Mengintegrasikan teknis-teknis dan metodologi-metodologi yang terbaik.

Proses menggambarkan *blueprint* suatu sistem secara grafis, terdiri dari komponen-komponen, *interfaces*, dan koneksi-koneksi yang ada dalam sistem tersebut, agar mudah dipahami dan dikomunikasikan.

Visual Modeling dapat membantu untuk menampilkan elemen-elemen yang penting secara *detail* dari suatu masalah yang kompleks dan menyaring untuk kemudian membuang elemen-elemen yang tidak penting (Suhendar, 2002).

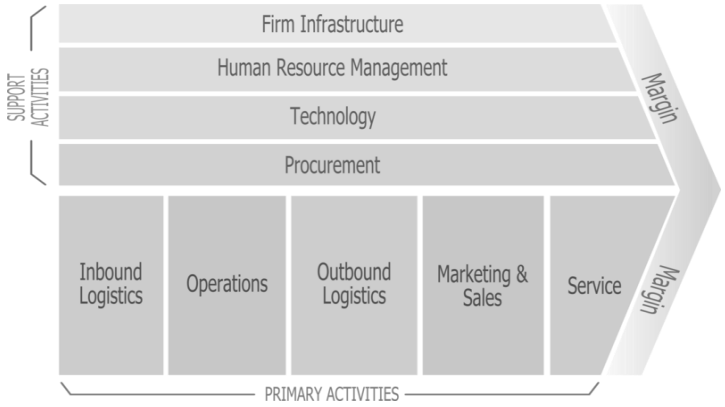
UML dapat digunakan untuk memodelkan berbagai jenis sistem, seperti sistem *software*, sistem *hardware* dan organisasi. Secara umum, UML juga dapat digunakan untuk menggambarkan model dalam proses *software development* dan *business modeling*.

Pemodelan bisnis menggambarkan semua *workflow* yang terjadi dalam suatu organisasi. Secara formal, didefinisikan pemodelan bisnis sebagai segala teknik pemodelan yang digunakan untuk menggambarkan model sebuah bisnis. Pemodelan bisnis dapat digunakan untuk meninjau, meningkatkan, dan membuat sebuah bisnis. UML digunakan sebagai bahasa pemodelan pada beberapa *tools* pengembangan perangkat lunak, karena UML adalah bahasa grafis yang terdiri dari sekumpulan notasi dan semantik. *Tools* yang menggunakan UML sebagai bahasanya adalah *Rational Rose* dan *Enterprise Architecture.*

UML sendiri bukanlah sebuah metode, tapi UML sendiri didisain awalnya untuk bisa cocok dengan metode pengembangan perangkat lunak berorientasi objek terdahulu (*Booch*, *Objectory* dll). Sejak UML dikembangkan, beberapa metode telah disusun ulang untuk mendapat keuntungan dari notasi dan metode baru yang telah dibuat berbasiskan UML.

2.5 *Value Chain* Michael E. Porter

Fungsi dari *value added chain* menurut Porter, yaitu untuk mendeskripsikan cara melihat bisnis sebagai rantai aktifitas yang mengubah *input* menjadi *output* sehingga memiliki nilai bagi pelanggan (Porter, 1985).



Gambar 2.4 *Value Added Chain* Porter, Michael E.
(Porter, 1985)

Seperti pada gambar 2.4, *Value chain* terbagi dalam dua kategori, yaitu:

1) *Primary Activities*, (*line functi*ons) merupakan aktifitas utama dari organisasi yang melibatkan aktifitas-aktifitas sebagai berikut:
   a) *Inbound Logistics*, pada bagian ini terkait dengan penerimaan, penyimpanan, dan pendistribusian input menjadi produk.
   b) *Operations*, semua aktifitas yang terkait dengan pengubahan input menjadi bentuk akhir dari produk, seperti produksi, pembuatan, pemaketan, perawatan peralatan, fasilitas, operasi, jaminan kualitas, proteksi terhadap lingkungan.
   c) *Outbond Logistics*, aktifitas yang terkait dengan pengumpulan, penyimpanan, distribusi secara fisik atau pelayanan terhadap pelanggan.
   d) *Marketing and Sales*, aktifitas yang terkait dengan pembelian produk dan layanan oleh pengguna dan mendorong untuk dapat membeli produk yang dibuat. Memiliki rantai nilai khusus, antara lain :
      (1) *Marketing management*
      (2) *Advertising*

(3) *Sales force administration*

(4) *Sales force operations*

(5) *Technical literature*

(6) *Promotion*

e) *Service*, aktifitas yang terkait dengan penyediaan layanan untuk meningkatkan atau merawat nilai dari suatu produk, seperti instalasi, perbaikan, pelatihan, suplai bahan, perawatan dan perbaikan bimbingan teknis.

2) *Secondary Activities*, (*staff* atau fungsi *overhead*) merupakan aktifitas pendukung yang membantu aktifitas utama. *Secondary activities* melibatkan beberapa bagian/fungsi, antara lain :

a) *Firm Infrastructure*, merupakan aktifitas, biaya, dan aset yang berhubungan dengan manajemen umum, *accounting*, keuangan, keamanan dan keselamatan Sistem Informasi, serta fungsi lainnya.

b) *Human Resources Management*, terdiri dari aktifitas yang terlibat seperti penerimaan, dengar pendapat, pelatihan, pengembangan, dan kompensasi untuk semua tipe personil, dan mengembangkan tingkat keahlian pekerja.

c) *Research, Technology, and System Development*, aktifitas yang terkait dengan biaya yang berhubungan dengan produk, perbaikan proses, perancangan peralatan, pengembangan perangkat lunak komputer, sistem telekomunikasi, kapabilitas basisdata baru, dan pengembangan dukungan sistem berbantuan komputer.

d) *Procurement,* terkait dengan fungsi pembelian input yang digunakan dalam *value chain* organisasi.

2.6 Aplikasi Portofolio

Tidak seperti pada model konsep traditional portofolio yang hanya mempertemukan hubungan antara sistem aplikasi yang satu dengan yang lainnya, serta bagaimana tugas dan ruang lingkup antar sistem didefinisikan, aplikasi portofolio merupakan sebuah model perkiraan kebutuhan sistem aplikasi yang didasarkan pada kebutuhan bisnis disertai dengan definisi apa dan bagaimana sistem aplikasi tersebut memberikan kontribusinya terhadap usaha-usaha pencapaian tujuan bisnis organisasi.

Tabel 2.2. merupakan matriks aplikasi portofolio yang terdiri dari empat kuadran, yaitu *strategic, key operational, support* dan *high potential* (Ward, 2002).

Tabel 2.2 Matriks Aplikasi Portofolio

STRATEGIC	***HIGH POTENTIAL***
Aplikasi-aplikasi kritis untuk menunjang perkembangan strategi bisnis organisasi di masa yang akan datang.	Aplikasi-aplikasi yang mungkin dibutuhkan oleh organisasi untuk keberhasilan dimasa yang akan datang, namun belum dibuktikan.
Aplikasi-aplikasi masa kini yang dibutuhkan oleh organisasi agar dapat menjalankan roda bisnisnya.	Aplikasi-aplikasi yang bersifat valuable tetapi tidak kritis.
KEY OPERATIONAL	***SUPPORT***

1) ***Strategic***

Berisi aplikasi-aplikasi yang secara kritis dibutuhkan untuk keberhasilan bisnis pada masa yang akan datang. Aplikasi ini dibuat untuk mendukung perubahan dan perkembangan organisasi dan bisnisnya.

2) ***Key Operational***

Berisi aplikasi-aplikasi operasional yang ada saat ini, dan dibutuhkan untuk mendukung operasional organisasi dan lebih bersifat sangat penting agar roda bisnis organisasi dapat berjalan.

3) ***Support Applications***

Berisi aplikasi yang dapat mendukung dan meningkatkan efisiensi bisnis dan efektifitas operasional.

4) ***High Potential***

Berisi aplikasi-aplikasi yang bersifat inovatif yang mungkin dapat memperbesar peluang peningkatan keuntungan dimasa yang akan datang, tapi belum dapat dibuktikan.

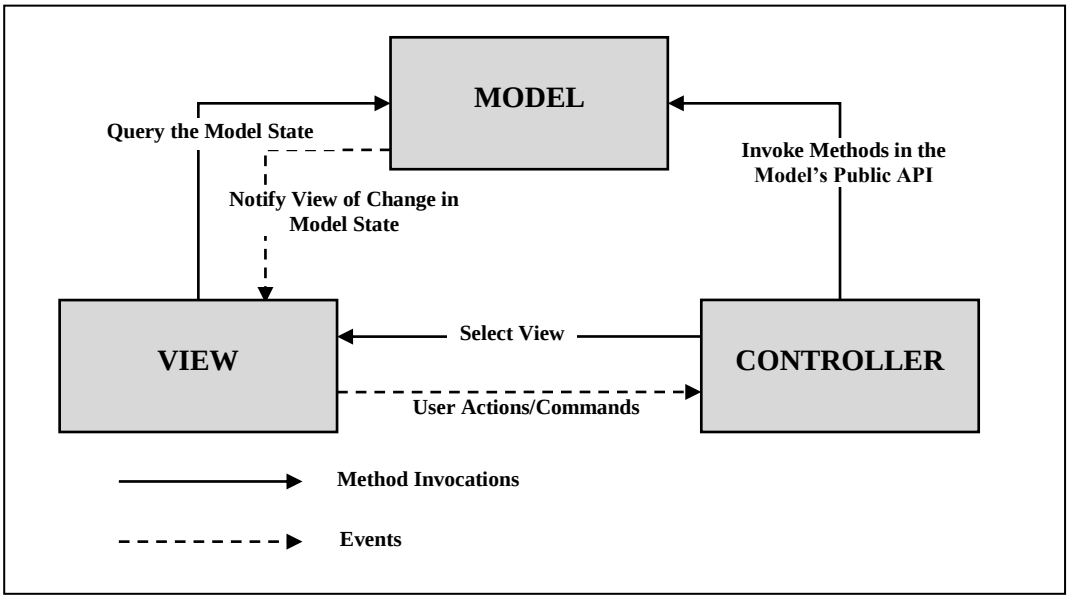
2.7 Arsitektur MVC (*Model-View-Controller*)

Model-View-Controller (MVC) adalah sebuah konsep yang diperkenalkan oleh penemu *Smalltalk* Trygve Reenskaug, untuk meng-enkapsulasi data bersama dengan pemrosesan (*model*), mengisolasi dari proses manipulasi (*controller*) dan tampilan (*view*) untuk direpresentasikan pada sebuah *user interface* (Balani dkk., 2002).

MVC mengikuti pendekatan yang paling umum dari *Layering*. *Layering* hanyalah sebuah logika yang membagi kode kita ke dalam fungsi di kelas yang berbeda. Pendekatan ini mudah dikenal dan yang paling banyak diterima. Keuntungan utama dalam pendekatan ini adalah penggunaan ulang (*reusability*) kode (Balani dkk., 2002) .

Definisi teknis dari arsitektur MVC dibagi menjadi tiga lapisan (Balani dkk., 2002) :

1) *Model*, digunakan untuk mengelola informasi dan memberitahu pengamat ketika ada perubahan informasi. Hanya model yang mengandung data dan fungsi yang berhubungan dengan pemrosesan data. Sebuah model meringkas lebih dari sekedar data dan fungsi yang beroperasi di dalamnya. Pendekatan model yang digunakan untuk komputer model atau abstraksi dari beberapa proses dunia nyata. Hal ini tidak hanya menangkap keadaan proses atau sistem, tetapi bagaimana sistem bekerja. Sebagai contoh, *programmer* dapat menentukan model yang menjembatani komputasi *back-end* dengan *front-end* GUI (*Graphical User Interface*).
2) *View*, bertanggung jawab untuk pemetaan grafis ke sebuah perangkat. *View* biasanya memiliki hubungan 1-1 dengan sebuah permukaan layar dan tahu bagaimana untuk membuatnya. *View* melekat pada model dan me-*render* isinya ke permukaan layar. Selain itu, ketika model berubah, *view* secara otomatis menggambar ulang bagian layar yang terkena perubahan untuk menunjukkan perubahan tersebut. Terdapat kemungkinan beberapa view pada model yang sama dan masing-masing *view* tersebut dapat me-*render* isi model untuk permukaan tampilan yang berbeda.
3) *Controller*, menerima input dari pengguna dan menginstruksikan model dan *view* untuk melakukan aksi berdasarkan masukan tersebut. Sehingga, *controller* bertanggung jawab untuk pemetaan aksi pengguna akhir terhadap *respon* aplikasi. Sebagai contoh, ketika pengguna mengklik tombol atau memilih *item* menu, *controller* bertanggung jawab untuk menentukan bagaimana aplikasi seharusnya merespon. Model, *view* dan *controller* sangat erat terkait, oleh karena itu, mereka harus merujuk satu sama lain.



Gambar 2.5 Hubungan antara *Model, View*, dan *Controller*

(Balani dkk., 2002)

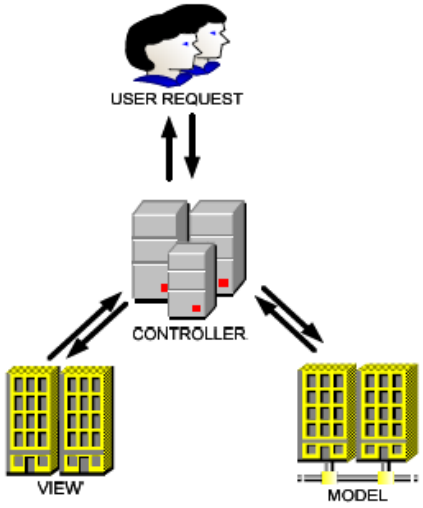
Arsitektur MVC memiliki manfaat yaitu pemisahan antara model dan *view* memungkinkan beberapa *view* menggunakan model yang sama. Akibatnya, komponen model sebuah aplikasi lebih mudah untuk diterapkan, diuji, dan dipelihara, karena semua akses ke model berjalan melalui komponen ini (Balani dkk., 2002).

2.8 *Framework Codelgniter*

Framework Codelgniter adalah sebuah *framework* PHP yang dapat membantu pengembangan aplikasi berbasis web berbasis PHP (Basuki, 2010).

Framework Codelgniter yang dibuat berdasarkan kaidah *Model-View-Controler* (MVC). Dengan menggunakan MVC, memungkinkan pemisahan antara *layer application-logic* dan *presentation*. Sehingga, dalam sebuah tim pengembangan, seorang *programmer* bisa konsentrasi pada *core-system*, sedangkan *web designer* bisa berkonsentrasi pada tampilan *web*. Menariknya

script PHP, query MySQL, Javascript dan CSS bisa saling terpisah, tidak dibuat dalam suatu *script* dengan ukuran yang sangat besar yang membutuhkan *resource* besar pula untuk mengeksekusinya. Akan demikian akan mudah di-*maintenance* dan dikembangkan lebih lanjut.



Gambar 2.6. *Framework Codelgniter*

(Basuki, 2010).

Ketika datang sebuah *user request*, maka permintaan tersebut akan ditangani oleh *controller*, kemudian *controller* akan memanggil model jika memang diperlukan operasi *database*. Hasil dari *query* oleh model dikembalikan ke *controller*. Selanjutnya *controller* akan memanggil *view* dan mengkombinasikan dengan hasil *query model*. Hasil akhir dari operasi ini akan di tampilkan ke *browser* dan bisa dilihat oleh *user*. Dalam konteks *Codelgniter* dan aplikasi berbasis *web*, maka penerapan konsep MVC mengakibatkan kode *program* dapat dibagi menjadi tiga kategori, yaitu :

1) *Model*, kode program (berupa OOP *class*) yang digunakan untuk memanipulasi *database*.

2) *View*, berupa *template* html/xhtml atau php untuk menampilkan data pada *browser*.
3) *Controller*, kode *program* (berupa OOP *class*) yang digunakan untuk mengkontrol aliran aplikasi (*model* dan *view*)

2.9 Penelitian Terkait

Berikut adalah beberapa penelitian yang mempunyai keterkaitan dengan penelitian yang dilakukan beserta pembahasan dan hasil dari penelitian tersebut :

1) *The Analysis of Academic Information System Design* of *IPDC Dili – Timor Leste by using EAP Approach* (Soares dkk., 2013).

   Dalam penelitian ini menggunakan pendekatan EAP, dan lebih fokus kepada arsitektur sistem utama pada perguruan tinggi. Pemodelan menggunakan E-R *Diagram* dan DFD sebagai pemodelan sistemnya. Tidak dijelaskan perencanaan penerapan aplikasi tersebut.
2) Perancangan Model *Enterprise Architecture* Dengan TOGAF *Architecture Development Method* (Yunis dkk., 2009).

   Dalam penelitian ini peneliti membuat rancang bangun arsitektur *enterprise* untuk menciptakan keselarasan antara bisnis dan teknologi informasi bagi kebutuhan organisasi, dengan menggunakan *framework* TOGAF ADM. Hasil akhir dari penelitian membuat sebuah arsitektur *enterprise* yang pada nantinya bisa dijadikan oleh organisasi untuk mencapai tujuan strategisnya. Dari penelitian tersebut peneliti menekankan pada pembahasan arsitektur data, arsitektur aplikasi dan arsitektur teknologi dan belum menggambarkan

sebuah *framework* yang terintegrasi pada setiap bagian atau divisi pada perguruan tinggi.

3) *Analysis Of Binus International University Information Technology Business Architecture* (Lukianto dkk., 2011)

   Dalam penelitian ini peneliti menggunakan TOGAF *framework* digabungkan dengan COBIT (*Control Objectives for Information and Related Technology*), dimana COBIT adalah suatu metodologi yang memberikan kerangka dasar untuk menciptakan Sistem Informasi yang sesuai dengan kebutuhan organisasi. Selain itu COBIT dapat digunakan untuk melakukan pengukuran (*maturity level*) dari Sistem Informasi suatu organisasi. Dengan menggunakan perpaduan prinsip-prinsip dalam TOGAF ADM dan COBIT peneliti tersebut menganailis arsitektur yang ada pada organisasi dan menilainya dengan kerangka COBIT dan mengajukan sebuah arsitektur baru. Pada tahapan perancangan arsitektur tidak tampak penggunaan kerangka kerja TOGAF dan integrasinya.

4) Model *Enterprise* Arsitektur untuk Perguruan Tinggi di Indonesia (Yunis dkk., 2009).

   Dalam penelitian ini peneliti menggunakan model arsitektur *enterprise* perguruan tinggi yang digunakan dalam penelitian secara keseluruhan menekankan bagaimana cara penerapan TOGAF ADM sebagai salah satu metode yang bisa digunakan untuk melakukan pengembangan arsitektur enterprise. TOGAF ADM dikombinasikan dengan metode pengembangan sistem *Rational Unified Process* (RUP) dengan tujuan untuk mendapatkan rancangan atau *blueprint* dari Sistem Informasi perguruan tinggi yang

nantinya akan dikembangkan. Dari penelitian tersebut hanya tergambar penekanan pada model arsitektur secara umum dan belum menggambarkan intergrasi dari masing-masing bagian antara fungsi bisnis utama dan fungsi bisnis pendukung pada perguruan tinggi.

5) Perencanaan Arsitektur *Enterprise* Perguruan Tinggi Dengan Menggunakan *Enterprise Unified Process* (EUP) (Farid, 2009)

   Dalam penelitian ini peneliti membuat perencanaan arsitektur *enterprise* untuk menciptakan keselarasan antara bisnis dan teknologi informasi bagi kebutuhan organisasi, dengan menggunakan metode EUP. Dari penelitian tersebut peneliti menekankan pada pembahasan arsitektur data, arsitektur aplikasi dan arsitektur teknologi dan belum menggambarkan sebuah *framework* yang terintegrasi pada setiap bagian atau divisi pada perguruan tinggi.

Setelah mempelajari penelitian-penelitian sebelumnya, saya melakukan penelitian sejenis dengan menggunakan metode EUP, yang akan menghasilkan *blueprint* artistektur sistem informasi perguruan tinggi, *roadmap* (urutan) aplikasi yang harus dibuat serta keterhubungan dari fungsi bisnis, data, aplikasi dan teknologi, yang diperlukan oleh *stakeholder* dalam membangun sebuah sistem informasi perguruan tinggi yang baik.