BAB 2

LANDASAN TEORI

2.1 *Electronic Learning (e-Learning)*

2.1.1 Pengertian *e-Learning*

e-Learning merupakan suatu jenis belajar mengajar yang memungkinkan tersampaikannya bahan ajar ke siswa dengan menggunakan media Internet, intranet atau media jaringan komputer lain (Hartley, 2001).

e-Learning adalah proses belajar secara efektif yang dihasilkan dengan cara menggabungkan penyampaian materi secara digital yang terdiri dari dukungan dan layanan dalam belajar (Vaughan W*all*er, 2001).

LearnFrame.Com dalam *Glossary of e-Learning Terms* (Glossary, 2001) menyatakan suatu definisi yang lebih luas bahwa *e-Learning* adalah sistem pendidikan yang menggunakan aplikasi elektronik untuk medukung belajar mengajar dengan media Internet, jaringan komputer, maupun komputer *standalone*.

Matthew Comerchero dalam *E-Learning Concepts and Techniques* (Bloomsburg, 2006) mendefinisikan bahwa *e-Learning* adalah saranan pendidikan yang mencakup motivasi diri sendiri, komunikasi, efisiensi dan teknologi. Karena ada keterbatasan dalam interaksi sosial, siswa harus menjaga diri mereka tetap termotivasi. *e-Learning* efisien karena mengeliminasi jarak dan arus pulang-pergi. Jarak dieliminasi karena isi dari *e-Learning* didesain dengan media yang dapat

diakses dari terminal komputer yang memiliki peralatan yang sesuai dan sarana teknologi lainnya yang dapat mengakses jaringan atau Internet.

Dari definisi-definisi yang muncul dapat kita disimpulkan bahwa sistem atau konsep pendidikan yang memanfaatkan teknologi informasi dalam proses belajar mengajar dapat disebut sebagai suatu *e-Learning* (Wahono, 2001, p1).

2.1.2 Fitur *e-Learning*

e-Learning memiliki fitur-fitur sebagai berikut (Clark & Mayer, 2008, p10) :

1. Konten yang relevan dengan tujuan belajar.
2. Menggunakan metode intruksional seperti contoh dan praktek untuk membantu belajar.
3. Menggunakan elemen media seperti kalimat dan gambar untuk mendistribusikan konten dan metode belajar.
4. Pembelajaran dapat secara langsung dengan instruktur (*synchronous*) ataupun secara individu (*asynchronous*).
5. Membangun wawasan dan teknik baru yang dihubungkan dengan tujuan belajar.

2.1.3 Elemen *e-Learning*

Definisi *e-Learning* memiliki beberapa elemen tentang apa, bagaimana dan mengapa dari *e-Learning* (Clark & Mayer, 2008, p10):

1. Apa. *e-Learning* memasukkan baik konten, yaitu informasi, dan metode intruksional, yaitu teknik, yang membantu orang mempelajari konten belajar.

2. Bagaimana. *e-Learning* didistribusikan melalui komputer dalam bentuk kalimat dan gambar. Pendistribusiannya dapat dalam bentuk *asynchronous* yang didesain untuk belajar secara individu dan dalam *synchronous* yang didesain dengan bimbingan dari instruktur secara langsung.
3. Mengapa. *e-Learning* ditujukan untuk membantu pelajar mencapai tujuan belajarnya atau melakukan pekerjaannya.

2.1.4 Aspek Penting dalam *e-Learning*

Berikut ini beberapa aspek penting dalam *e-Learning* :

1. *e-Learning* menciptakan solusi belajar formal dan informal.

   Salah satu kesalahan berpikir tentang *e-Learning* adalah *e-Learning* hanya menciptakan sistem belajar secara formal, seperti dalam bentuk kursus. Namun faktanya adalah saat ini 80% pembelajaran didapat secara informal. Banyak orang saat beraktivitas sehari-hari dan menghadapai suatu masalah membutuhkan solusi secepatnya. Dalam hal ini, *e-Learning* haruslah memiliki karakteristik berikut :

   a. *Just in time* : tersedia untuk pengguna ketika mereka membutuhkannya untuk menyelesaikan tugasnya.

   b. *On Demand* : tersedia setiap saat.

   c. *Bite Sized* : tersedia dalam ukuran yang kecil agar dapat digunakan secara cepat.

2. *e-Learning* menyediakan akses ke berbagai macam sumber pembelajaran baik itu konten maupun manusia.

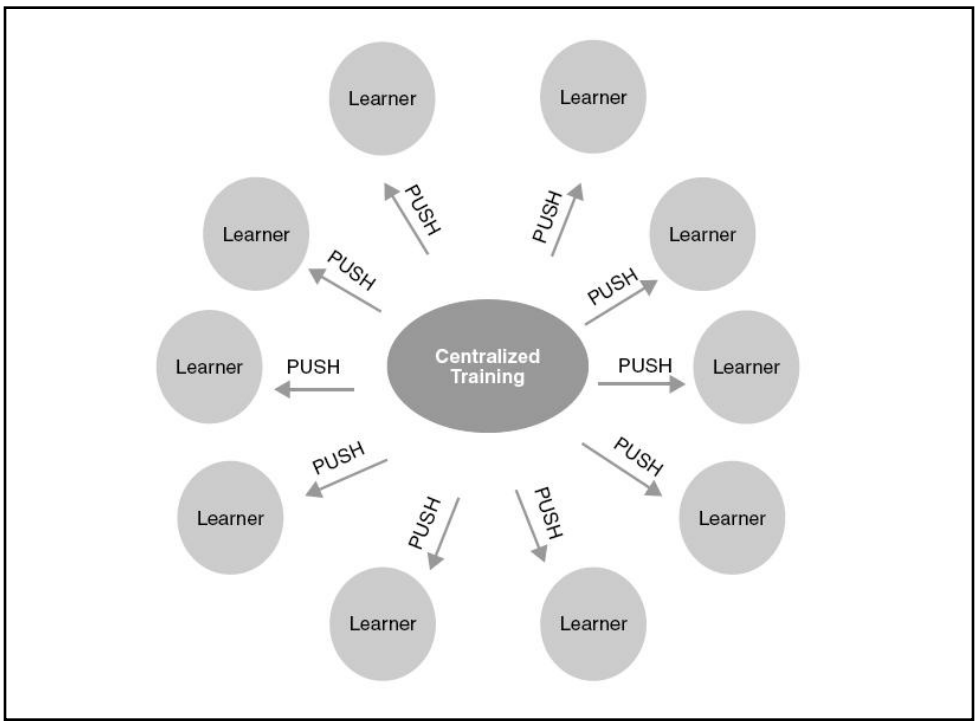
Kesalahan lainnya dalam berpikir tentang *e-Learning* bahwa *e-Learning* hanya membuat konten saja. Sebenarnya *e-Learning* adalah sebuah aktivitas sosial. *E-Learning* menyediakan pengalaman belajar yang kuat melalui komunitas *online* pengguna *e-Learning*. Karena manusia adalah makhluk sosial jadi ada banyak kesempatan untuk berkomunikasi, berkolaborasi dan berbagi ilmu antara sesama pengguna *e-Learning*.

3. *e-Learning* mendukung sekelompok orang atau kelompok untuk belajar bersama.

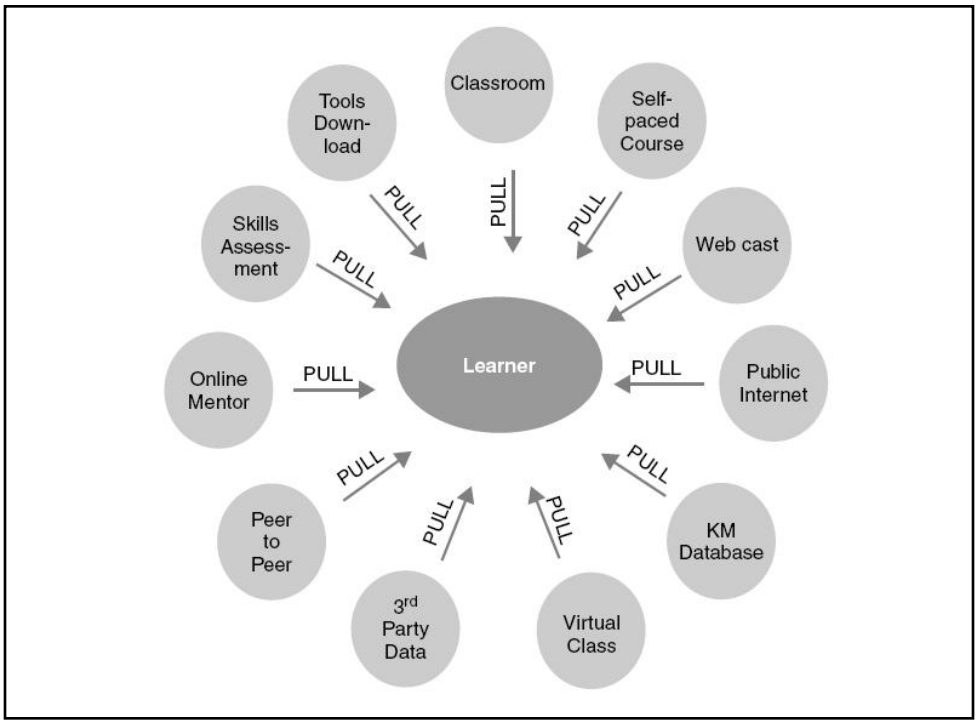
E-Learning bukan aktivitas individu saja, tetapi juga mendukung sekelompok orang atau kelompok untuk belajar bersama, baik untuk berkomunikasi, berkolaborasi, berbagi ilmu, dan membentuk sebuah komunitas *online* yang dapat dilakukan secara langsung (*synchronous*) atau tidak langsung (*asynchronous*).

4. *e-Learning* membawa pembelajaran kepada pelajar bukan pelajar ke pembelajaran.

Bentuk pembelajaran tradisional bahwa pelajar harus pergi keluar untuk mencari pembelajaran mereka sendiri. Sedangkan model *e-Learning* disebut juga *Pull Model of Learning* (Knight, 2005, p.11).



Gambar 2.1 *Push Model of Learning* (Morrison, 2003, p 26)



Gambar 2.2 *Pull Model of Learning* (Morrison, 2003, p 27)

2.1.5 Keuntungan *e-Learning*

Keuntungan menggunakan *e-Learning* diantaranya sebagai berikut (Wahono, 2005, p. 2):

1. Fleksibel karena siswa dapat belajar kapan saja, di mana saja, dan dengan tipe pembelajaran yang berbeda-beda.
2. Menghemat waktu proses belajar mengajar
3. Mengurangi biaya perjalanan
4. Menghemat biaya pendidikan secara keseluruhan (infrastruktur, peralatan, buku-buku)
5. Menjangkau wilayah geografis yang lebih luas
6. Melatih pembelajar lebih mandiri dalam mendapatkan ilmu pengetahuan

2.1.6 Kelemahan *e-Learning*

Kelemahan menggunakan *e-Learning* diantaranya sebagai berikut (Rosenberg, 2006):

1. Karena *e-Learning* menggunakan teknologi informasi, tidak semua orang terutama orang yang masih awam dapat menggunakannya dengan baik.
2. Membuat *e-Learning* yang interaktif dan sesuai dengan keinginan pengguna membutuhkan *programming* yang sulit, sehingga pembuatannya cukup lama.
3. *e-Learning* membutuhkan infrastruktur yang baik sehingga membutuhkan biaya awal yang cukup tinggi.
4. Tidak semua orang mau menggunakan *e-Learning* sebagai media belajar.

2.1.7 Arsitektur *e-Learning*

Berikut ini merupakan arsitektur e-*Learning* yang dijelaskan dalam table 2.1 :

Tabel 2.1 Tiga Arsitektur *e-Learning* (Clark & Mayer, 2008, p27)

Arsitektur	View	Interaktif	Penggunaan
Receptive	Akuisisi Informasi	Rendah	Penginformasian
Directive	Penguatan respons	Medium	Melakukan prosedur seperti penggunaan software
Guided Discovery	Konstruksi wawasan	Tinggi	Melakukan strategi seperti pemecahan masalah

2.2 Qt

Qt merupakan C++ *toolkit* yang ditujukan untuk pengembangan aplikasi *Graphics User Interface*(GUI) *cross-platform*. Qt mempunyai kemampuan untuk dapat digunakan pada sistem operasi Windows, Mac, Linux, dan sistem operasi berbasis Unix lainnya dengan menggunakan pendekatan "*write once, compile anywhere*"(anonim1). Beberapa keuntungan menggunakan Qt:

1. Qt merupakan software *Open source.*
2. Kemampuan *cross-platform*, Qt dapat berjalan di sistem operasi Unix/ Linux maupun Windows.
3. Dukungan *User Interface* yang lengkap

2.2.1 Fitur Utama Qt

Ada beberapa fitur utama yang terdapat pada Qt .(anonim2).

1. *Signal and Slots*

Dalam Qt *Signal and Slots* digunakan untuk komunikasi antar *object*. *Signal* dibangkitkan jika suatu *event* terjadi, baik yang disebabkan oleh interaksi *user* maupun dari internal program. *Slot* merupakan fungsi yang dipanggil sebagai respon dari *signal* tertentu.

2. *Layout Management*

Layout Management berperan pada desain *interface*. *Class layout* Qt didesain agar mudah digunakan dan dimengerti. Ada beberapa *Class* yang digunakan untuk mengatur *layout*. Qt menyediakan QtDesigner, sebuah *tool* untuk mendesain GUI secara visual, yang akan menghasilkan kode *layout*.(Blanchette, Jasmin p20)

3. *Internationalization*

Internationalization merupakan proses membuat aplikasi dapat digunakan oleh orang di negara lain. Qt mendukung banyak bahasa.

2.2.2 *Tools* Qt

Qt memiliki *tools* yang membantu dalam pengembangan, beberapa tools tersebut antara lain(anonim3) :

1. Qt Designer

   Merupakan *tools* untuk mendesain dan membangun GUI dari komponen Qt. Qt Designer memungkinkan pembuatan *wid*gets dan *dialog* menggunakan *on-screen forms*.

2. Qt Assistant

   Merupakan *tools* yang berisi dokumentasi Qt

3. Qt Linguist

   Tools yang mendukung translasi aplikasi ke dalam bahasa lokal

2.3 *eXtensible Markup Languange* (XML)

2.3.1 Pengertian XML

Menurut Charles F. Goldfarb(2000, p1), XML merupakan subset dari *Standart Generalized Markup Language*(SGML).

XML, merupakan format text yang sederhana, sangat fleksibel yang diturunkan dari SGML(anonim4).

2.3.2 Tujuan Perancangan XML

Menurut Erik T. Ray(2003, p13), *World Wide Web Consortium* (W3C) pada pertengahan 1990 memulai mengerjakan *markup language* yang mengkombinasikan fleksibilitas dari SGML dengan kesederhanaan dari *HyperText Markup Languange* (HTML). Tujuan pembuatan XML:

1. *Application-Specific Markup Languages*

   XML memberitahukan bagaimana pengguna dapat membuat *markup elements* sendiri. Pengguna dapat membuat *markup language* untuk mengekspresikan informasi dengan cara yang terbaik. Hal ini berarti dapat ada banyak *markup language* dalam jumlah yang tak terbatas.

2. *Unambiguous Structure*

   Tidak boleh ada dua cara mengintrepretasikan *names, order* dan hirarki dari elemen pada dokumen XML. Hal ini mengurangi kesalahan dan kompleksitas kode.

3. Presentasi disimpan ditempat lain

   Agar dokumen memiliki fleksibilitas format *output*, informasi mengenai *style* disimpan di luar dokumen. XML memungkinkan hal ini dengan menggunakan *stylesheets* yang mengandung informasi mengenai *styl*e.

4. Sederhana

   XML harus sederhana sehingga dapat mendapat sambutan yang luas.

5. Pengecekan kesalahan yang maksimal

   Spesifikasi XML mengatakan *file* yang *well-formed* harus memenuhi kebutuhan minimal *syntax*.

2.3.3 Pemrosesan XML

Menurut Charles F. Goldfarb(2000, p38), ada 4 komponen dalam pemrosesan XML:

1. *Parsing*

   Parsing memisahkan *textual representation* dari sebuah dokumen dan merubahnya menjadi kumpulan objek konseptual. Pemroses XML disebut *XML parser*.

2. APIs

   XML memiliki standarisasi *Aplication Processing Interfaces*(API).

a. DOM

W3C menetapkan standarisasi sebuah API untuk XML. Disebut dengan *Document Object Model*(DOM). DOM dapat digunakan untuk membaca dan menulis XML.

b. SAX

*Simple API for XML(*SAX), dikembangkan oleh XML-DEV. SAX merupakan *event-based* API, *parser* ini memungkinkan aplikasi mengerjakan sedikit bagian dari keseluruhan data yang ditemukan oleh *parser* pada setiap *event* pada dokumen.

2.4 *eXtensible Stylesheet Languange Transformations* (XSLT)

2.4.1 Pengertian XSLT

Menurut Michael Kay(2004, p9), *eXtensible Stylesheet Language Transformation*(XSLT) merupakan bahasa untuk merubah struktur dan konten dari dokumen XML.

Menurut Doug Tidwell(2001, p7), *eXtensible Stylesheet Language Transformation*(XSLT) merupakan rekomendasi resmi dari *World Wide Web Consortium* (W3C). Bahasa yang fleksibel dan kuat untuk melakukan transformasi pada dokumen XML kedalam bentuk lain. Bentuk lain dapat berupa dokumen HTML, dokumen XML lainnya, *Portable Document Format*(PDF), dan lain-lain.

2.4.2 Versi XSLT

Setiap elemen teratas dari file XSL harus menyebutkan versi dari XSL. Versi XSLT yang terakhir direkomendasikan oleh W3C adalah versi 1.0, yang direkomendasikan tanggal 16 November 1999. Versi 2.0 sampai sekarang masih dalam tahap penyempurnaan. Terakhir W3C mengeluarkan *progress report* untuk versi 2.0 pada tanggal 15 November 2002 (anonim5).

2.4.3 *Parser* Transformasi XSLT

Sama seperti dokumen XML, dokumen XSLT juga harus di-*parse*. Proses *parsing* ini dapat terjadi di *client*, dapat pula di *server*. Untuk *parsing* di *server*, terdapat beberapa *parser* yang mendukungnya, seperti Xalan Java (anonim6). *Parser* Xalan Java ini harus di-*install* di *server* agar proses transformasi dapat berjalan. Namun, cara seperti itu tidak direkomendasikan, karena dapat memberatkan kerja *server*. Rekomendasinya agar *user* memakai *browser* yang mendukung XSLT, artinya memiliki *built in parser* untuk XSLT (anonim7), seperti :

1. AIX Netscape

   Platform : AIX

2. InDelv XML *browser*

   Platform : Java 1.1.6 ke atas

3. Mozilla version 1.2 beta

   Platform : Linux, Win32, MacOS

4. X-Smiles version 0.6

   Platform : Java

6. Microsoft Internet Explorer 6.0

   Platform : Windows98/NT/2000

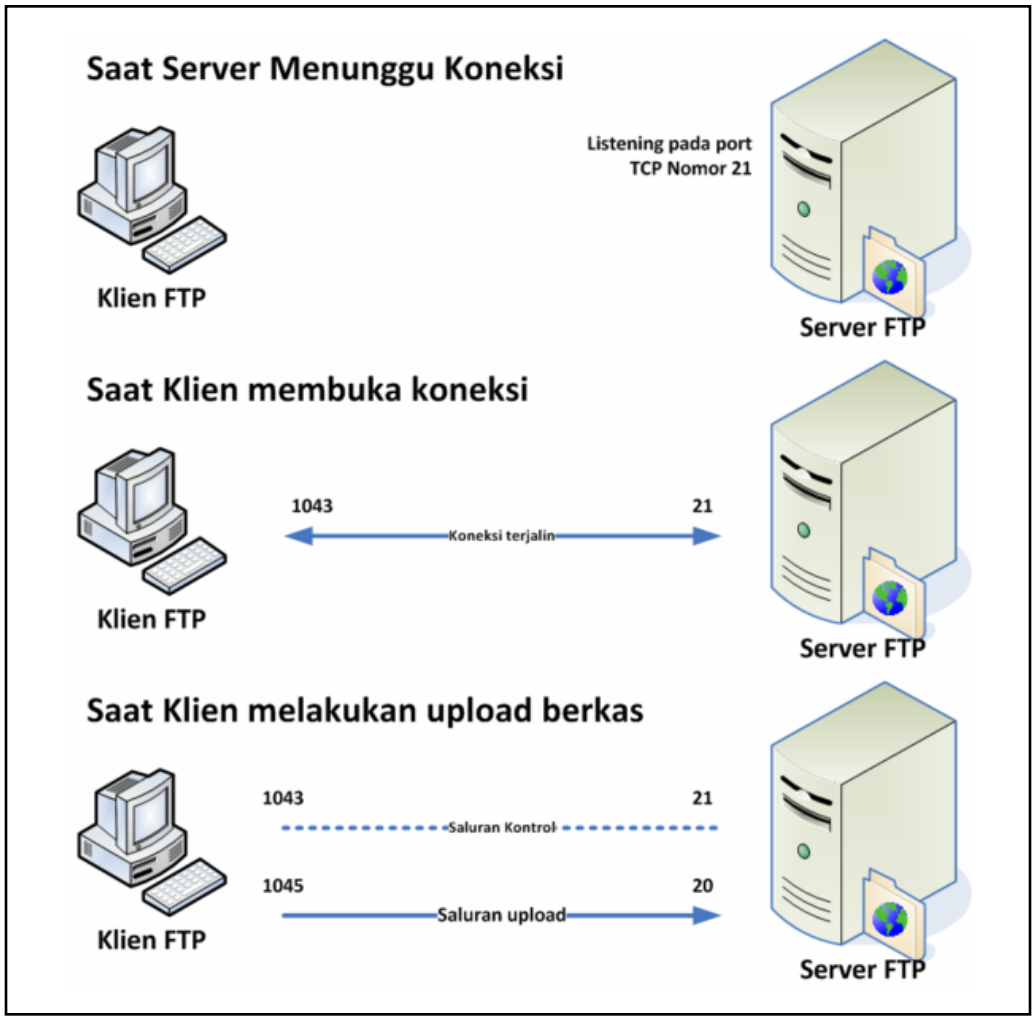
2.5 *File Transfer Protocol* (FTP)

FTP adalah sebuah protokol Internet yang berjalan di dalam lapisan aplikasi yang merupakan standar untuk pentransferan berkas (file) computer antar mesin-mesin dalam sebuah *Internetwork*(anonim8). FTP didefinisikan sebagai sebuah protokol untuk mengirim dan menerima file antara *host*(dalam *Advanced Research Project Agency Network*(ARPANET), dengan fungsi utama dari FTP adalah mengirim dan menerima file degan efisien dan handal antara *host* dan mengijinkan penggunaan yang nyaman dari kemampuan untuk penyimpanan file secara *remote.*

Tujuan dari FTP [RFC959] adalah:

1. Untuk mempromosikan *share* file (yang dapat berupa program komputer dan data).
2. Untuk memacu penggunaan komputer *remote* secara tidak langsung atau implisit (via perangkat lunak).
3. Untuk melindungi pengguna dari variasi sistem penyimpanan file diantara *host.*
4. Untuk mengirim dan menerima data secara handal dan efisien.

FTP adalah bagian dari protokol TCP/ IP dan menggunakan TCP sebagai *transport layer*.



Gambar 2.3 Cara Kerja FTP

2.6 *HyperText Transfer Protocol* (HTTP)

HTTP adalah protokol yang dipergunakan untuk mentransfer dokumen dalam *World Wide Web* (WWW). Protokol ini adalah protokol ringan, tidak berstatus dan generik yang dapat dipergunakan berbagai macam tipe dokumen(anonim9).

2.6.1 HTTP *Request*

Permintaan-permintaan dari *client* ke *server* berisikan informasi tentang macam-macam data yang *user* inginkan. Salah satu item informasi yang dienkapsulasi pada permintaan HTTP adalah sebuah nama *method*. Ini memberitahu *server* macam-macam permintaan yang dibuat, sebagaimana sisa pesan dari *client* diformat. Ada tiga protokol yang memungkinkan digunakan : *GET*, *POST*, dan *PUT* (anonim10).

1. *GET*

GET adalah *method* HTTP paling sederhana dan digunakan sebagian besar untuk meminta *resource* tertentu dari *server*, apakah berupa halaman *web*, file gambar grafis, atau sebuah dokumen, dan lain-lain.

GET dapat juga digunakan untuk mengirim data di atas *server*, meskipun demikian hal itu mempunyai batasan-batasan. Jumlah total karakter yang dapat dienkapsulasi ke dalam permintaan *GET* adalah terbatas, sehingga untuk situasi dimana banyak data perlu dikirimkan ke *server*, tidak semua pesan dapat disampaikan.

Batasan lain *method* permintaan *GET* ketika mengirim data adalah data yang Anda kirim menggunakan *method* ini ditambahkan pada *Uniform Resource Locator*(URL) yang Anda kirim ke *server*. (Untuk sekarang, asumsikan URL sebagai alamat unik yang akan Anda kirim ke *server* sebagai penandaan lokasi yang Anda minta). Salah satu permasalahannya adalah URL dari beberapa permintaan yang Anda inginkan ditampilkan pada

bar *browser* pada beberapa *browser*. Hal ini berarti, bahwa beberapa data sensitif seperti *password* atau informasi kontak (*contact information*) dapat terlihat oleh siapapun.

Keuntungan dari penggunaan *GET* dalam pengiriman data di atas *server* adalah permintaan URL dari permintaan *GET* dapat di-*bookmark* oleh *browser*. Hal ini berarti bahwa *user* dapat dengan mudah mem-*bookmark* permintaannya dan mengakses setiap saat dari pada melalui proses tiap waktu. Hal ini juga dapat membahayakan, jika *bookmark* secara fungsional bukan merupakan sesuatu yang Anda inginkan pada *user* Anda, sebagai gantinya menggunakan *method* lain.

2. *POST*

Jenis lain dari *method* permintaan yang pasti akan digunakan adalah permintaan *POST*. Jenis permintaan ini didesain seperti *browser* dapat membuat permintaan kompleks dari *server*. Mereka didesain sehingga *user*, melalui *browser*, dapat mengirim banyak data ke *server*. Form kompleks secara umum dicapai dengan menggunakan permintaan *POST*, sebagaimana form sederhana yang memerlukan proses *upload* file ke *server*.

Satu perbedaan yang nyata antara *method GET* dan *POST* terletak pada cara mengirimkan data ke *server*. Seperti yang dinyatakan sebelumnya, *GET* hanya menambahkan data ke URL yang akan mengirim. *POST*, di sisi lain, mengenkapsulasi atau menyembunyikan data di dalam *body* pesan (*message body*) yang dikirim. Ketika *server* menerima permintaan dan menentukan

bahwa itu merupakan sebuah permintaan *POST*, dapat dilihat dari *body* pesan data tersebut.

3. PUT

Berfungsi untuk meng-*upload* representasi dari sumber tertentu.

2.6.2 HTTP *Response*

HTTP me-*response* dari *server* yang berisi *headers* dan *body* pesan, seperti yang permintaan HTTP lakukan. Mereka menggunakan kumpulan *header* yang berbeda, meskipun demikian disini kita tidak perlu terlalu dalam membahasnya secara detail. Cukup dengan mengatakan bahwa *headers* berisi informasi tentang protokol HTTP yang digunakan pada *server*, sebagaimana tipe dari isi yang dienkapsulasi ke dalam *body* pesan. Nilai dari tipe isi adalah *Multipurpose Internet Mail Extension*(MIME)-*type*. Ini akan memberitahu *browser* jika pesan berisi HTML, gambar, atau tipe lainnya.

2.7 Rekayasa Perangkat Lunak

Rekayasa perangkat lunak adalah disiplin ilmu yang membahas semua aspek produksi perangkat lunak, mulai dari tahap awal spesifikasi sistem sampai pemeliharaan sistem setelah digunakan. (Sommerville, 2003). Pada definisi ini, ada dua istilah kunci yaitu:

1. Disiplin rekayasa, perekayasa membuat suatu alat bekerja. Menerapkan teori, metode, dan alat bantu yang sesuai, selain itu mereka menggunakannya

dengan selektif dan selalu mencoba mencari solusi terhadap permasalahan, walaupun tidak ada teori atau metode yang mendukung. Perekayasa juga menyadari bahwa mereka harus bekerja dalam batasan organisasi dan keuangan, sehingga mereka berusaha mencari solusi dalam batasan-batasan ini.

2. Semua aspek produksi perangkat lunak, rekayasa perangkat lunak tidak hanya berhubungan dengan proses teknis dari pengembangan perangkat lunak tetapi juga dengan kegiatan seperti manajemen proyek perangkat lunak dan pengembangan alat bantu, metode, dan teori untuk mendukung produksi perangkat lunak.

Secara umum, rekayasa perangkat lunak memakai pendekatan sistematis dan terorganisasi terhadap pekerjaan mereka karena cara ini seringkali paling efektif untuk menghasilkan perangkat lunak berkualitas tinggi. Namun demikian, rekayasa ini sebenarnya mencakup masalah pemilihan metode yang paling sesuai untuk satu set keadaan dan pendekatan yang lebih kreatif, informal terhadap pengembangan yang mungkin efektif pada beberapa keadaan. Pengembangan informal sangat cocok untuk pengembangan sistem *e-commerce web* membutuhkan gabungan keahlian perangkat lunak dan perancangan grafis.

2.7.1 Proses Perangkat Lunak

Proses perangkat lunak adalah serangkaian kegiatan-kegiatan dan hasil-hasil relevannya yang menghasilkan perangkat lunak. Kegiatan-kegiatan ini sebagian besar dilakukan perekayasa perangkat lunak. Ada

empat kegiatan proses dasar yang umum bagi seluruh kegiatan proses perangkat lunak. Kegiatan-kegiatan ini adalah :

1. Spesifikasi perangkat lunak, fungsionalitas perangkat lunak dan batasan kemampuan operasinya harus didefinisikan
2. Pengembangan perangkat lunak, perangkat lunak yang memenuhi spesifikasi tersebut harus diproduksi.
3. Validasi perangkat lunak, perangkat lunak harus divalidasi untuk menjamin bahwa perangkat lunak melakukan apa yang diinginkan oleh pelanggan.
4. Evolusi perangkat lunak, perangkat lunak harus berkembang untuk memenuhi kebutuhan pelanggan yang berubah-ubah.

Proses perangkat lunak yang berbeda mengatur kegiatan ini dengan cara berbeda dan dijelaskan dengan tingkat kerincian yang berbeda pula. Waktu kegiatan bervariasi, sebagaimana hasilnya. Pengaturan yang berbeda dapat menggunakan proses yang berbeda untuk menghasilkan produk dengan jenis yang sama. Namun demikian, untuk beberapa jenis aplikasi tertentu, beberapa proses lebih sesuai dari yang lainnya jika digunakan proses yang tidak sesuai, maka kualitas penggunaan produk perangkat lunak yang akan dikembangkan tersebut mungkin berkurang.

2.7.2 Model Proses Perangkat Lunak

Model proses pengembangan perangkat lunak adalah sebagai berikut :

1. Model air terjun (*waterfall*). Model ini mengambil kegiatan proses dasar seperti spesifikasi, pengembangan, validasi dan evolusi, dan merepresentasikannya sebagai fase-fase proses yang berbeda seperti spesifikasi persyaratan, perancangan perangkat lunak, implemetasi, pengujian dan seterusnya.
2. Pengembangan evolusioner. Pendekatan ini berhimpitan dengan kegiatan spesifikasi, pengembangan, dan validasi. Suatu sistem awal dikembangkan dengan cepat dari spesifikasi abstrak. Sistem ini kemudian diperbaiki dengan masukan dari pelanggan untuk menghasilkan sistem yang memuaskan bagi kebutuhan pelanggan.
3. Pengembangan sistem formal. Pendekatan ini didasarkan atas pembuatan spesifik sistem matematis dan pentransformasian spesifikasi, dengan memakai metode matematis untuk membangun program. Verifikasi komponen sistem dilakukan dengan membuat argumen matematis yang disesuaikan dengan spesifikasi.

Pengembangan berdasarkan pemakaian ulang. Pendekatan ini didasarkan atas adanya komponen yang dapat dipakai untuk jumlah yang signifikan . Proses pengembangan sistem terfokus pada integrasi komponen-komponen ini ke dalam suatu sistem dan bukan mengembangkan dari awal.

2.8 Interaksi Manusia dan Komputer

Interaksi manusia dan komputer adalah disiplin ilmu yang berhubungan dengan perancangan, evaluasi, dan implementasi sistem komputer interaktif untuk digunakan oleh manusia, serta studi fenomena-fenomena besar yang

berhubungan dengannya. Ilmu ini secara khusus menitikberatkan pada perancangan dan evaluasi antar pemakai (*user interface*).

2.8.1 *Eight Golder Rules User Interface Design*

Menurut Shneiderman (1998, p.74-75), ada delapan aturan emas perancangan antarmuka yang harus diperhatikan dalam perancangan dialog.

Delapan aturan emas (Eight Golden Rules) tersebut adalah :

1. Berusaha untuk konsisten

   Dalam hal merancang tampilan harus selalu berusaha untuk konsisten.

2. Memungkinkan *frequent users* menggunakan *shortcuts*.

   Pengguna yang sudah sering menggunakan aplikasi lebih menginginkan kecepatan dalam mengakses fungsi-fungsi yang diinginkan. Sehingga disediakan tombol-tombol spesial penyingkat untuk memudahkan user langsung berinteraksi dengan fungsi yang diinginkannya.

3. Memberikan umpan balik yang informatif.

   Umpan balik harus diberikan untuk memberikan informasi kepada user sesuai dengan aksi(action) yang dilakukan, sehingga user mengetahui aksi apa yang telah dan akan dilakukan dengan adanya umpan balik ini. Umpan balik dapat berupa konfirmasi atau informasi atas suatu aksi.

4. Merancang dialog yang memberikan penutupan (keadaan akhir).

   Umpan balik atas akhir dari suatu proses-proses dan aksi sangat membantu dan juga pengguna mendapat signal untuk melanjutkan aksi lainnya.

5. Memberikan pencegahan kesalahan dan penanganan kesalahan yang sederhana.

   Sistem dirancang sedemikian rupa sehingga dapat mencegah pengguna dalam membuat kesalahan. Bila terjadi kesalahan sistem harus dapat memberikan instruksi sederhana, konstruktif, dan spesifik untuk perbaikan.

6. Memungkinkan pembalikan aksi yang mudah.

   User kadang tidak sengaja melakukan aksi yang tidak diinginkan, untuk itu user ingin melakukan pembatalan. Sistem harus dapat memberikan fungsi pembatalan ini sehingga user akan merasa lebih aman dan tidak takut dalam mencoba dan memakai sistem tersebut.

7. Mendukung internal *locus of control.*

   User yang berpengalaman sangat menginginkan kontrol yang kuat pada sistem sehingga user merasa menguasai sistem tersebut. Sitem yang tidak terduga dan sulit dalam melakukan aksi, akan menyulitkan *user*.

8. Mengurangi beban ingatan jangka pendek.

   Keterbatasan memori pada manusia harus mampu diatasi oleh sistem sehingga tidak banyak membuat *user* melakukan proses penyimpanan memori.