BAB 2 LANDASAN TEORI

2.1 Aplikasi Wiki

Wiki adalah sebuah situs web (atau koleksi dokumen hiperteks lainnya) yang memperbolehkan penggunanya menambah atau mengubah isi situs tersebut. Istilah ini juga dapat merujuk kepada software kolaboratif yang digunakan untuk menciptakan situs web semacam itu. Wiki (dengan huruf besar 'W') and WikiWikiWeb kadang digunakan untuk merujuk kepada Portland Pattern Repository, wiki yang paling pertama diciptakan. Pendukung penggunaan ini mengusulkan penggunaan huruf kecil 'w' untuk membedakan istilah generik yang sedang dibicarakan di sini. Wiki wiki berasal dari istilah bahasa Hawaii untuk "cepat" atau "super-cepat".

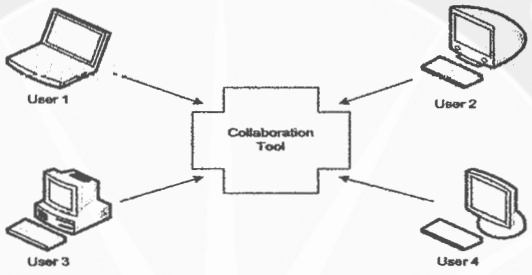
Kadang istilah wikiwiki atau WikiWiki digunakan daripada wiki yang artinya siapapun, termasuk Anda, dapat mengubah atau menyunting isi situs yang ada. Istilah-istilah ini dapat digunakan secara berganti-ganti meskipun perbedaan pandangan mengenai kapitalisasi juga dapat diterapkan dalam cara yang mirip.

2.2 Penulisan Buku Secara Kolaboratif

Kolaboratif dapat diartikan sebagai proses, percakapan dan perilaku yang berhubungan dengan kerjasama atau kolaborasi antar individuals dalam rangka mencapai tujuan bersama. Kolaborasi tidak hanya sekedar menempatkan para individu, tetapi diatur pula bagaimana mengkoordinasikan mereka agar dapat bekerjasama. Gambar 2.1 menunjukkan konsep penulisan buku secara kolaboratif. Pengguna berkolaborasi dengan

tool yang tersedia melalui jaringan internet untuk membuat suatu buku yang telah ditentukan oleh pemilik (*author*) buku.

Kerjasama atau kolaborasi dalam penulisan buku ini adalah suatu proses yang menggambarkan adanya interaksi antar individu dalam penulisan isi buku untuk membentuk sebuah buku yang berguna bagi siapa saja. Dimana setiap individu dapat saling membantu dengan memberi masukkan yang mendukung proses penulisan isi dari suatu buku.



Gambar 2.1 Konsep Penulisan Buku Secara Kolaboratif

2.3 Internet

Internet merupakan suatu jaringan komputer yang terhubung dengan ratusan bahkan ribuan jaringan komputer lain di seluruh dunia. Setiap orang di seluruh dunia dengan profesi yang berbeda dapat menggunakan internet sesuai dengan kebutuhannya masing-masing. Internet selalu berhubungan dengan *World Wide Web* (WWW), karena segala informasi dalam internet disajikan dalam bentuk *web site*. WWW adalah layanan yang paling sering digunakan dan memiliki perkembangan yang sangat cepat karena layanan ini dapat menerima informasi dalam berbagai format (*multimedia*). WWW merupakan suatu kumpulan informasi pada beberapa *server* komputer yang terhubung satu sama lain dalam jaringan internet. Informasi dalam *web* mempunyai *link* yang menghubungkan informasi tersebut ke informasi lain dalam jaringan

internet. Sistem yang menghubungkan informasi satu ke informasi lain melalui suatu link disebut *hypertext*.

Hubungan situs jaringan sebagai sumber informasi dan pengguna dalam WWW adalah hubungan *Client-Server*. Saat pengguna mengirimkan permintaan ke situs jaringan, *web server* akan menerima permintaan tersebut dan kemudian memberikan tanggapan kembali kepada pengguna. Sementara hubungan tersebut berlangsung, *web server* menunggu permintaan layanan dari pengguna lain.

2.4 Rekayasa Perangkat Lunak

Sejak tahun 1950-an perangkat lunak mulai dikembangkan oleh manusia untuk membantu memecahkan berbagai macam masalah yang bervariasi dari masalah sehari-hari sampai ke masalah yang kompleks. Karena masalah yang diselesaikan oleh perangkat lunak semakin bervariasi dan kompleks maka perangkat lunak dikembangkan dengan menggunakan paradigma tertentu. Paradigma tersebut disebut paradigma rekayasa perangkat lunak.

Paradigma rekayasa perangkat lunak mempunyai beberapa metode. Paradigma yang dipakai pada pengembangan perangkat lunak dipengaruhi oleh jenis perangkat lunak yang akan dikembangkan. Paradigma rekayasa perangkat lunak klasik yang banyak digunakan adalah paradigma *waterfall*, atau *classic life cycle*. Paradigma ini menggunakan model yang sistematis dan sekuensial yang dimulai pada tingkat sistem yang diteruskan dengan analisis, perancangan, pengkodean, pengujian, dan perawatan. Metodologi yang dapat digunakan dalam paradigma tersebut adalah terstruktur (tradisional) atau *object oriented*. Pada metodologi

terstruktur pendekatannya berfokus pada fungsi dari sistem, sedangkan pada *object oriented* berpusat pada objek yang menggabungkan atribut dan *method*-nya.

2.5 Konsep Dasar *Object-Oriented*

Pada perangkat lunak berorientasi objek, algoritma dan struktur dibungkus bersama sebagai sebuah objek, yang mempunyai sekumpulan atribut dan/atau method. Bahasa pemrograman berorientasi objek menyediakan mekanisme enkapsulasi, pewarisan dan polimorfisme.

a. Enkapsulasi

   Enkapsulasi dapat dianggap sebagai bungkusan pelindung program dan data yang sedang diolah. Pembungkus ini mendefinisikan perilaku dan melindungi program dan data agar tidak dapat diakses sembarangan oleh program lain. Keampuhan dari enkapsulasi program adalah setiap orang tahu cara mengaksesnya, sehingga dapat menggunakannya tanpa harus mengerti cara implementasinya secara terperinci.

b. Pewarisan

   Pewarisan mengijinkan suatu objek dibuat berdasarkan objek lain. Pewarisan juga mengijinkan untuk berbagi dan menggunakan kembali atribut dan *method*.

c. Polimorfisme

   Polimorfisme, berarti satu objek dengan banyak bentuk, adalah konsep sederhana yang memperbolehkan *method* memiliki beberapa implementasi yang dipilih berdasarkan tipe objek yang dilewatkan pada pengerjaan *method*.

2.6 Unified Modeling Language (UML)

Unified Modeling Language (UML) merupakan bahasa *universal* untuk pemodelan perangkat lunak dan hal tersebut digunakan untuk mengekspresikan model yang beraneka ragam dan dengan tujuan yang berbeda. Notasi UML menjadi notasi standar untuk pembangunan perangkat lunak berorientasi objek. UML digunakan untuk mendeskripsikan dan memodelkan pada setiap fase pengembangan perangkat lunak.

Tujuan utama pada perancangan dengan menggunakan UML adalah sebagai berikut :

1. Membuat pengguna siap menggunakan bahasa pemodelan visual yang ekspersif sehingga dapat dikembangkan dan ditukar dengan objek yang berarti.
2. Menyediakan mekanisme *extensibility* dan spesialisasi untuk menyampaikan inti konsep.
3. Menjadi bebas dari bahasa pemrograman dan pengembangan proses yang khusus.
4. Menyediakan dasar formal untuk mengerti bahasa pemodelan.
5. Mendorong perkembangan pemasaran *tools* berorientasi objek.
6. Mendukung konsep pengembangan level tinggi.
7. Menggabungkan praktik dan metodologi yang terbaik.

2.6.1 Use Case Diagram

Use case diagram menjelaskan manfaat sistem jika dilihat menurut pandangan orang yang berada diluar sistem (*actor*). Diagram ini menunjukkan fungsionalitas suatu sistem atau kelas dan bagaimana sistem berinteraksi dengan dunia luar. *Use case diagram*

menggambarkan kebutuhan (*requirements*) dengan melihat bagaimana sistem digunakan dan siapa penggunanya.

Diagram use case ini terdiri dari :

a. Aktor

Aktor yaitu representasi pemakai sistem (manusia atau sistem yang lain). *Actor* menyatakan peranan yang dimainkan pengguna saat melakukan interaksi dengan sistem.

b. Use Case

Use case adalah representasi fungsionalitas atau layanan yang diberikan sistem kepada pemakai. *Use case* merupakan deskripsi sederetan aksi yang dilakukan sistem untuk mendapatkan hasil tertentu, yaitu sederetan aksi yang menyatakan interaksi antara sistem dengan sesuatu diluar sistem.

c. Asosiasi antara aktor dan *use case*

Asosiasi antara aktor A dengan *use case* X berarti aktor A akan menjalankan *use case* X dan memperoleh hasil dari *use case* X tersebut. Asosiasi antar aktor dan *use case* dapat digambarkan dengan garis tidak terputus-putus, dengan anak panah opsional pada ujung yang bersentuhan dengan *use case*, di mana anak panah opsional tersebut mengarah ke *use case*, seperti pada gambar 2.2.

Gambar 2.2 Asosiasi antara aktor dan *use case*

Antar *use case* dalam *use case diagram* dapat memiliki tiga macam relasi, yaitu *include*, *extend*, dan generalisasi.

a. Relasi *extend* digunakan untuk menunjukkan bahwa suatu *use case* menyediakan fungsionalitas tambahan yang mungkin diperlukan dalam *use case* lain. Sebagai contoh, pada gambar 2.3, *use case Print* memiliki relasi *extends* dengan *use case Display*. Hal ini berarti pada titik tertentu saat aktor mengeksekusi *use case Display*, aktor secara opsional dapat menjalankan *use case Print*.

Gambar 2.3 Relasi *extend* pada *use case diagram*

b. Relasi *include* digunakan untuk menunjukkan bahwa suatu *use case* harus melewati *use case* lain agar dapat berjalan. Sebagai contoh, pada gambar 2.4, *use case* Edit Data memiliki relasi *include* dengan *use case* Login. Hal ini berarti agar *use case* Edit Data dapat dijalankan, *use case* Login harus dijalankan lebih dahulu.

Gambar 2.4 Relasi *include* pada *use case diagram*

c. Relasi generalisasi antar *use case* A dan *use case* B, dimana *use case* B adalah turunan dari *use case* A berarti bahwa *use case* B memiliki semua atribut dan perilaku *use case* A, dan *use case* B dapat memiliki atribut atau perilaku tambahan. Sebagai contoh, pada gambar 2.5 *use case* Buka Deposito merupakan turunan dari *use case* Buka Rekening. Hal ini berarti semua atribut dan perilaku yang dilakukan pada Buka Rekening akan dimiliki oleh

use case Buka Deposito, dan *use case* Buka Deposito dapat memiliki atribut atau perilaku tambahan.

Gambar 2.5 Relasi generalisasi pada *use case diagram*

2.6.2 Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek dalam sistem dan *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atas dimensi vertikal (waktu) dan dimensi horizontal (objek-objek terkait). *Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang memicu aktivitas tersebut, proses apa saja yang terjadi secara internal dan *output* apa yang dihasilkan. Masing-masing objek, memiliki *lifeline* vertikal. *Message* digambarkan dengan garis berpanah dari satu objek ke objek lainnya.

2.6.3 Collaboration Diagram

Collaboration diagram menggambarkan interaksi antar objek seperti *sequence diagram*, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian *message*. Setiap *message* memiliki *sequence number*, di mana *message* dari level tertinggi memiliki nomor 1. Messages dari level yang sama memiliki prefiks yang sama.

2.6.4 Class Diagram

Class diagram menggambarkan tipe-tipe objek pada sistem dan berbagai macam relasi statis yang ada di antara mereka. *Class diagram* menunjukkan *class-class*

yang ada, termasuk atribut-atribut dan method-methodnya.

2.7 XML dan HTML

XML singkatan dari *eXtensible Markup Language* yang sudah menjadi bagian penting bagi programmer yang akan mengembangkan web services. Hal ini karena XML dibangun dengan kemampuan melakukan transfer data antarplatform. XML juga memiliki kemampuan integrasi data disamping pertukaran data antar platform. Untuk web statis yang sederhana, programmer seringkali hanya menggunakan HTML untuk membangunnya. Sedangkan jika web tersebut kompleks ataupun membutuhkan konten yang dinamis, bahasa HTML akan dikombinasikan dengan bahasa pemrograman internet *server-side* seperti PHP,ASP, dan lain-lain. Namun setelah diparser oleh web server, program akan diubah menjadi HTML untuk dikirim ke browser client.

Bahasa XML tidak sama dengan HTML. HTML didesain untuk menampilkan data dan berfokus bagaimana data tersebut ditampilkan, sedangkan XML didesain untuk membawa data, mendeskripsikan data dan berfokus pada apakah data itu. Secara sederhana, XML yang mendefinisikan data dan HTML yang menampilkan data. Kesamaan dari kedua bahasa tersebut yakni sama-sama menggunakan tag.

Tag - tag yang digunakan pada dokumen XML sifatnya *not predefined,* ditentukan sediri. XML dapat menggunakan DTD atau XSD untuk mendefinisikan data supaya menjadi XML Valid sedangkan XML yang hanya berdasarkan sintaks yang benar saja tanpa memperhatikan isi data disebut sebagai XML Well Formed. Untuk dapat

mengatur tampilan isi data pada XML, digunakan suatu bahasa stylesheet yaitu XSL yang akan mendeskripsikan bagaimana XML tersebut ditampilkan atau menghasilkan data dalam format HTML atau text dan sebagainya.

2.8 XSD (XML Schema Definition)

Skema XML adalah sebuah alternative DTD. Skema XML mendeskripsikan struktur dari dokumen XML dan sering disebut *XML Schema definition (XSD)*. Menurut Siswoutomo (2004), tujuan utama dari skema XML adalah mendefinisikan *legal building blocks* dari sebuah dokumen XML. Jadi, sebuah skema XML :

1. Mendefinisikan elemen yang dapat muncul di dokumen.
2. Mendefinisikan atribut yang dapat muncul di dokumen.
3. Mendefinisikan elemen mana yang merupakan anak elemen.
4. Mendefinisikan apakah elemen dapat berisi teks atau kosong.
5. Mendefinisikan nilai default dan nilai tetap untuk elemen dan atribut.
6. Mendefinisikan tipe data dari elemen maupun atribut.

Sangat besar kemungkinan Skema XML akan menjadi yang paling banyak digunakan di aplikasi berbasis web dan menggantikan DTD. Alasannya sebagai berikut :

1. Skema XML mudah dikembangkan.
2. Skema XML lebih kaya dan berguna daripada DTD.
3. Skema XML ditulis menggunakan XML.
4. Skema XML mendukung tipe data.
5. Skema XML mendukung namespaces.

2.9 XSL (Extensible Stylesheet Language)

XSL merupakan Extensible Stylesheet Language. XSL dibangun karena adanya kebutuhan XML berbasis stylesheet language. XML tidak menggunakan predefined tag karena nama dan makna tag ditentukan sediri. Dengan demikian, dapat dikatakan tag tersebut *not well understood*. Misalkan, jika menggunakan tag <table>, browser tidak akan mengerti bagaimana menampilkannya meskipun jika dalam HTML berarti menampilkan tabel. Oleh karena itu, dibutuhkan sesuatu yang ditambahkan ke dokumen XML yang mendeskripsikan bagaimana dokumen tersebut ditampilkan, itulah XSL. XSL lebih dari sekedar stylesheet language. Ia terdiri dari atas tiga bagian :

1. XSLT adalah bahasa untuk melakukan transform dokumen XML.
2. Xpath adalah bahasa untuk mendefinisikan bagian dari dokumen XML.
3. XSL-FO adalah bahasa untuk memformat dokumen XML.

Bayangkan XSL adalah suatu set bahasa yang dapat mentransform XML ke XHTML, memfilter dan mensortir data XML, mendefinisikan bagian dari dokumen XML, memformat data XML berdasarkan nilanya (misalkan yang bernilai negatif diberi warna merah), menghasilkan data XML ke berbagai media berbeda seperti layar monitor, kertas atau suara.

2.10 Tools dan Teknologi yang Digunakan

2.10.1 Teknologi .NET

Microsoft Visual Studio .NET adalah sebuah *platform* untuk membangun, menjalankan dan meningkatkan

generasi lanjut dari aplikasi terdistribusi. *Microsoft Visual Studio .NET* memperluas *client*, *server* dan *service-service* yang terdiri atas:

1. Sebuah model pemrograman yang memungkinkan *developer* membangun aplikasi dan layanan web *XML*.
2. Sekumpulan *XML Web services* seperti *Microsoft .NET My Services*, yang membantu *developer* menghasilkan aplikasi yang sederhana dan terpadu.
3. Serangkaian *server* termasuk *Microsoft Windows Server 2003*, *Microsoft SQL Server* dan *Microsoft BizTalk Server* yang terintegrasi, untuk menjalankan, mengoperasikan dan mengelola aplikasi dan layanan berbasis web.
4. *Tool-tool* pengembang yang menyediakan IDE (*Integrated Development Environment*) untuk memaksimalkan produktivitas pengembangan menggunakan *.NET Framework*.
5. Piranti lunak *client*, seperti *Windows XP*, *Windows CE* dan *Microsoft Office XP* yang membantu pengembang untuk menyebarkan dan mengelola aplikasinya.

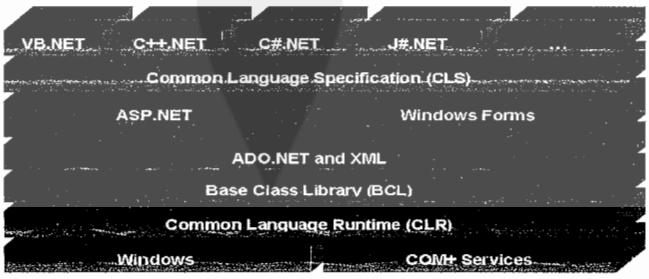
2.10.2 Framework .NET

Framework .NET seperti terlihat pada gambar 2.6 adalah lingkungan untuk membangun, menyebarkan */deploying*, dan menjalankan aplikasi dan layanan berbasis web. *.Net Framework* disusun oleh dua komponen utama, yaitu *Common Language Runtime (CLR)* dan *.NET Framework Class Library* termasuk aplikasi *Console*, *Windows GUI*, *ASP.NET*, layanan web *XML* dan layanan *Windows*.

Visual Studio.NET dibangun menggunakan fondasi *.NET Framework*, menyediakan lingkungan yang cerdas, mudah dikembangkan untuk membangun, menyebarkan dan menjalankan aplikasi dan layanan web XML yang terdistribusi. *.NET Framework* memisahkan *platform* sistem operasi menjadi dua lapisan, yaitu lapisan pemrograman dan lapisan eksekusi.

Tujuan dari *.NET Framework* adalah :

1. Menyediakan lingkungan pemrograman berorientasi objek, apakah kode objek disimpan dan dijalankan secara lokal, dijalankan secara lokal tetapi disebarkan melalui internet atau dijalankan secara *remote* (dijalankan dari suatu tempat).
2. Menyediakan lingkungan untuk menjalankan suatu kode yang menjamin keamanan saat kode dijalankan.
3. Menyediakan lingkungan untuk menjalankan suatu kode yang dapat mengeliminasi masalah performa dari lingkungan *scripted* dan *interpreted*.
4. Menyediakan lingkungan untuk menjalankan suatu kode yang meminimalkan konflik pada *deployment* dan *versioning* perangkat lunak.
5. Menyatukan model-model pemrograman dengan didukung oleh banyak bahasa dan membuat berbagai tipe aplikasi.



Gambar 2.6 Arsitekur .NET *Framework*