BAB II

LANDASAN TEORI

2.1. Tinjauan Pustaka

2.1.1. Konsep Dasar Sistem Informasi

Menurut Sutabri (2012:38), ”Sistem informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian yang mendukung fungsi operasi organisasi yang bersifat manajerial dengan kegiatan setrategi dari suatu organisasi untuk dapat menyediakan laporan-laporan yang diperlukan oleh pihak luar tertentu”.

Sistem informasi terdiri dari komponen-komponen yang disebut dengan istilah blok bangun *(building block),* yang terdiri dari blok masukan, blok model, blok keluaran, blok teknologi, blok basis data, dan blok kendali. Sebagai suatu sistem, keenam blok tersebut saling berinteraksi satu dengan yang lain membentuk satu kesatuan untuk mencapai sasaran.

1. Blok masukan *(input block)*

Input mewakili data yang masuk ke dalam sistem informasi. Yang dimaksud dengan input di sini termasuk metode dan media untuk menangkap data yang akan dimasukan, yang dapat berupa dokumen-dokumen dasar.

2. Blok model *(model block)*

Blok ini terdiri dari kombinasi prosedur, logika, dan model matematika yang akan memanipulasi data input dan yang tersimpan di basis data dengan cara yang sudah tertentu untuk menghasilkan keluaran yang dinginkan.

3. Blok keluaran *(output block)*

Produk dari sistem informasi adalah keluaran yang merupakan informasi yang berkualitas dan dokumentasi yang berguna untuk semua tingkat manajemen serta semua pemakai sistem.

4. Blok teknologi *(technology block)*

Teknologi merupakan *tool box* dalam sistem informasi. Teknologi digunakan untuk menerima input, menjalankan model, menyimpan dan mengakses data, menghasilkan dan mengirimkan keluaran dan membantu pengendalian sistem secara keseluruhan.

5. Blok basis data *(database block)*

Basis data *(database)* merupakan kumpulan data yang saling berkaitan dan erhubungan satu dengan yang lainnya, tersimpan di perangkat keras komputer dan perangkat lunak digunakan untuk memanipulasinya. Data perlu disimpan dalam basis data untuk keperluan penyediaan informasi lebih lanjut. Data di dalam basis data perlu diorganisasikan sedemikian rupa supaya informasi yang dihasikan berkualitas.

Banyak hal yang dapat merusak sistem informasi, seperti bencana alam, api air, debu, kecurangan-kecurangan, kegagalan pada sistem itu sendiri, ketidak efisienan, sabotase, dan lain sebagainya. Beberapa pengendalian perlu dirancang dan diterapkan untuk meyakinkan bahwa hal-hal yang dapat merusak sistem dicegah dan bila terlanjur terjadi maka kesalahan-kesalahan dapat dengan cepat diatasi.

2.1.2. PHP (*Personal Home Page*)

PHP dibuat pertama kali oleh seorang perekayasa perangkat lunak (*software engineering*) yang bernama Rasmus Lerdoff. Rasmus Lerdoff mebuat halaman *web* PHP pertamanya pada tahun 1994. PHP4 dengan versi-versi akhir menuju PHP5 sudah menukung pemrograman berorientasi objek.

Menurut Rossa dan Shalahudin (2011:85) "PHP merupakan bahasa pemrograman yang digunakan untuk pemrograman yang dilakukan untuk pemrograman *web*".

2.1.3. *Database MySQL*

Menurut Kadir (2008:348) "*MySQL* adalah salah satu jenis *database server* yang sangat terkenal". Kepopulerannya disebabkan *MySQL* menggunakan *SQL (Structured Query Language)* sebagai bahasa dasar untuk mengakses *database*nya. Selain itu, ia bersifat *Open Source* (tidak perlu membayar untuk menggunakannya) pada berbagai *platform* (kecuali untuk jenis *Enterprise*, yang bersifat komersial).

2.1.4. *Adobe Dreamweaver*

Menurut Hadi (2008:27) Adobe Dreamweaver adalah "program aplikasi web editor untuk membuat dan mendesain *web* dengan mudah dan tepat yang terpopuler saat ini."

Adobe Dreamweaver CS6 merupakan salah satu aplikasi yang digunakan untuk mendesain sekaligus melakukan pemrograman *web*.

Adobe Dreamweaver CS6 memberikan fasilitas untuk mengedit HTML secara *visual*. Aplikasi ini menyertakan banyak perangkat yang berkaitan dengan pengkodean dan fitur seperti HTML, CSS, hingga *JavaScript*. Selain itu. Aplikasi ini juga memungkinkan pengeditan *JavaSript*, XML, dan dokumen teks lainnya secara langsung.

2.1.5. Konsep Dasar Model Pengembangan Sistem

Menurut Rossa dan Shalahuddin (2011:28), model *waterfall* adalah "model air terjun menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau terurut dimulai dari analisa, desain, pengkodean, ".

Pendekatan alur hidup perangkat lunak secara sekuensial itu sendiri terdiri dari:

1. Analisis kebutuhan perangkat lunak

   Proses pengumpulan kebutuhan dilakukan secara intensif untuk menspesifikasikan kebutuhan perangkat lunak agar dapat dipahami perangkat lunak seperti apa yang dibutuhkan oleh *user*. Spesifikasi kebutuhan perangkat lunak pada tahap ini perlu untuk didokumentasikan.

2. Desain

   Desain perangkat lunak adalah proses multi langkah yang fokus pada desain pembuatan program perangkat lunak termasuk struktur data, arsitektur perangkat lunak, representasi antar muka, dan prosedur pengkodean.Tahap ini mentranslasi kebutuhan-kebutuhan perangkat lunak dari tahap analisis kebutuhan ke representasi desain agar dapat diimplementasikan menjadi

program pada tahap selanjutnya.

Desain perangkat lunak yang dihasilkan pada tahap ini juga perlu didokumentasikan.

3. Pembuatan Kode Program

   Pada tahap ini, desain yang sudah dibuat harus ditranslasikan ke dalam program perangkat lunak sehingga menghasilkan program komputer yang sesuai dengan desain.

4. Pengujian

   Pengujian difokuskan pada perangkat lunak dari segi logika dan fungsional dan memastikan bahwa semua bagian sudah diuji. Hal ini dilakukan untuk meminimalisir kesalahan (*error*) dan mematikan keluaran yang dihasilkan sesuai dengan yang diinginkan.

5. Pendukung (*support*)

   Tidak menutup kemungkinan perangkat lunak mengalami perubahan ketika sudah dikirimkan *user*. Perubahan bisa karena adanya kesalahan yang muncul dan tidak terdeteksi saat pengujian atau perangkat lunak harus beradaptasi dengan lingkungan baru. Tahap pendukung atau pemeliharaan dapat mengulangi proses pengembangan mulai dari analisis spesifikasi untuk perubahan perangkat lunak yang sudah ada, tapi tidak untuk membuat perangkat lunak baru.

Dari kenyataan model air terjun (*waterfall*) sangat jarang dapat dilakukan sesuai alurnya karena beberapa sebab. Berikut menurut Rossa dan Shalahuddin (2011:28) :

1. Perubahan spesifikasi perangkat lunak terjadi di tengah alur pengembangan.
2. Sangat sulit bagi pelanggan untuk mendefinisikan semua spesifikasi di awal alur pengembangan. Pelanggan sering kali butuh contoh (*prototype*) untuk menjabarkan spesifikasi kebutuhan sistem lebih lanjut.
3. Pelanggan tidak mungkin bersabar mengakomodasi perubahan yang diperlukan di akhir alur pengembangan.

Dengan berbagai kelemahan yang dimiliki model air terjun (*waterfall*) tapi model ini telah menjadi dasar dari model-model yang lain dalam melakukan perbaikan model pengembangan perangkat lunak. Model air terjun (*waterfall*) sangat cocok digunakan kebutuhan pelanggan sudah sangat dipahami dan kemungkinan terjadinya kebutuhan selama pengembangan perangkat lunak kecil. Hal positif dari model air terjun adalah struktur tahap pengembangan sistem jelas, dokumentasi hasil di setiap tahap pengembangan, dan sebuah tahap dijalankan setelah tahap sebelumnya selesai dijalankan, sehingga tidak ada tumpang tindih pelaksanaan tahap.

2.1.6. *Unified Modelling Language* (UML)

Menurut Rossa dan Shalahuddin (2011:118) "UML (*Unified Modelling Language*) merupakan bahasa *visual* untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung". UML (*Unified Modelling Language*) telah menjadi standar dalam merancang suatu sistem visualisasi dan mendokumentasikan sistem piranti lunak. Para pengembang sistem berorientasi objek menggunakan bahasa model untuk menggambarkan, membangun dan mendokumentasikan sistem yang sedang dirancang. UML

(*Unified Modelling Language)* memungkinkan para anggota tim untuk bekerja sama dengan model yang sama dalam merancang suatu sistem yang akan dibuat, dikarenakan dapat mempermudah bagi perancangan dalam merancangnya.

Menurut Sukamto dan Shalahdudin (2011:120) pada UML (*Unified Modelling Language)* macam-macam diagram dikelompokkan dalam kategori. Pembagian dan macam-macam diagram tersebut antara lain:

A. Diagram *Use Case*

Use case atau diagram *use case* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu. Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefinisian yang disebut *actor* dan *use case.*

1. *Actor* merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari *actor* adalah gambar orang, tapi *actor* belum tentu merupakan orang.
2. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau *actor.*

B. Diagram Aktivitas (*Activity Diagram*)

Diagam aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis.

Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan *actor*, jadi aktivitas yang dapat dilakukan oleh sistem. Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut:

1. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
2. Urutan atau pengelompokkan tampilan dari sistem atau *user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
3. Rancangan penguji dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.

C. Diagram Komponen (*Component Diagram*)

Diagram komponen atau *component diagram* dibuat untuk menunjukkan organisasi dan ketergantungan diantara kumpulan komponen dalam sebuah sistem. Diagram komponen fokus pada komponen sistem yang dibutuhkan dan ada di dalam sistem. Diagram komponen juga dapat digunakan untuk memodelkan hal-hal berikut:

1. *Source code* program perangkat lunak
2. Komponen *executable* yang dilepas ke *user*
3. Basis data secara fisik
4. Sistem yang harus beradaptasi dengan sistem lain.
5. *Framework system*, *framework* pada perangkat lunak merupakan kerangka kerja yang dibuat untuk memudahkan pengembangan dan pemeliharaan aplikasi, contohnya seperti *Struts* dari *Apache* yang menggunakan prinsip

desain *Model-View-Controller* (MVC) dimana *source code* program dikelompokkan berdasarkan fungsinya sebagai berikut, dimana *controller* berisi *sour code* yang menangani *request* dan validasi, model berisi *sour code* yang menangani manipulasi data dan *business logic*, dan *view* berisi *source code* yang menangani tampilan.

Komponen dasar yang biasanya ada dalam suatu sistem adalah sebagai berikut:

1. Komponen *user interface* yang menangani tampilan.
2. Komponen *business processing* yang menangani fungsi proses bisnis.
3. Komponen data yang menangani menipulasi data
4. Komponen *security* yang menangaani keamanan sistem.

D. Diagram Deployment (*Deployment Diagram*)

Diagram deployment atau *deployment diagram* bersifat statis, diagram ini memperlihatkan konfigurasi saat aplikasi dijalankan (*run-time*), memuat simpul-simpul beserta komponen-komponen yang ada di dalamnya. Diagram deployment berhubungan erat dengan diagram komponen dimana diagram ini memuat satu atau lebih komponen-komponen. Diagram ini sangan berguna saat aplikasi kita berlaku sebagai aplikasi yang dijalankan pada banyak mesin (*distributed computing*). Diagram deployment juga dapat digunakan untuk memodelkan hal-hal berikut:

1. Sistem tambahan (*embedded system*) yang menggambarkan rancangan *device, node, hardware*.
2. Sistem *client* atau *server*
3. Sistem terdistribusi resmi
4. Rekayasa ulang aplikasi

2.1.7. *Entity Relationship Diagram (ERD)*

> Menurut Simarmata (2006:63) mengemukakan bahwa:
> ERD (*Entity Relationship Diagram*) adalah alat pemodelan data utama dan akan membantu mengorganisasi data suatu proyek ke dalam entitas-entitas dan menentukan hubungan antar entitas. Proses memungkinkan analisis menghasilkan struktur basis data yang baik sehingga dapat disimpan dan diambil secara efisien.

Model ERD (*Entity Relationship Diagram*)diperkenalkan pada tahun 1976 oleh P.P.Chen.

Kegunaan ERD adalah:

1. Dapat menggambarkan hubungan antar *entity* dengan jelas
2. Dapat menggambarkan batasan jumlah *entity* dan partisipan antar *entity*.
3. Mudah dimengerti oleh pemakai.
4. Mudah disajikan oleh perancang *database*.

Komponen-komponen yang terdapat di dalam *Entity Relational* model:

1. Entitas

   Suatu kumpulan objek atau sesuatu yang dapat dibedakan atau dapat didefinisikan secara unik. Kumpulan entitas yang sejenis disebut *entity set*.

   a. *Entity* yang bersifat fisik, yaitu *entity* yang dapat dilihat

      Contohnya: rumah, kendaraan, mahasiswa, dosen, dan lain-lain.

   b. *Entity* yang bersifat konsep atau logika, yaitu *entity* yang tidak dapat dilihat. Contohnya: pekerjaan, rencana, dan lain-lain.

   c. Simbol yang digunakan untuk *entity* adalah persegi panjang.

2. *Relationship*

   a. Hubungan yang terjadi antara satu *entity* atau lebih *entity*.

   b *Relationship* tidak mempunyai keberadaan fisik, kecuali yang

mewarisi hubungan antara *entity* tersebut.

c. *Relationship set* adalah kumpulan *relationship* yang sejenis.

d. Simbol yang digunakan adalah bentuk belah ketupat, *diamond*.

3. Atribut

a. Karakteristik *entity* atau *relationship* yang menyediakan penjelasan detail tentang atau *relationship* tersebut.

b. *Atribut value* adalah suatu data *actual* atau informasi yang disimpan disuatu atribut di dalam suatu *entity* atau *relationship*. Terdapat 2 jenis atribut yaitu:

1) Identifer (*key*) untuk menentukan *entity* secara unik

2) Descriptor (*ninkey attribute*) untuk menentukan karakteristik dari suatu *entity* yang tidak unik.

4. *Indicator Type*

a. *Indicator Type Associative Object* berungsi sebagai suatu objek dan suatu *relationship*.

b. *Indicator Type Subpertype* terdiri dari suatu objek dan satu subkategori atau lebih yang dihubungkan dengan satu *relatioship* yang tidak sama.

c. *Cardinality Ratio* atau *Mapping Cardinallity* adalah menjelaskan banyaknya *entity* yang bersesuaian dengan *entity* yang lain melalui *relationship*.

Jenis-jenis *Cardinality Ratio*:

1) *One to one* (1:1) adalah hubungan satu *entity* dengan satu *entity*.

2) *One to Many* (1:M) adalah hubungan satu *entity* dengan banyak *entity* atau banyak *entity* dengan satu *entity*.

3) *Many to Many* (M:N) adalah hubungan banyak *entity* dengan banyak *entity*

d. Derajat *Relationship* menyatakan jumlah *entity* yang berpatisipasi di dalam suatu *relationship*.

1) Derajat satu (*Unary degree*) adalah derajat yag memiliki satu *relationship* untuk satu buah *entity*

2) Derajat dua (*Binary degree*) dalah derajat yang memiliki satu *relationship* untuk dua buah *entity*.

3) Derajat tiga (*Tenary degree*) derajat yang memiliki satu *relationship* untuk tiga atau lebih *entity*.

5. *Participation constraint* menjelaskan apakah keberadaan suatu *entity* tergantung pada hubungan dengan *entity* lain. Terdapat dua macam *participation constraint* yaitu:

a. *Total Participation*, yaitu keberadaan suatu *entity* tegantung pada hubungannya dengan *entity* lain dan *Entity Relationship* digambarkan dua garis penghubung antara *entity* dan *relationship*.

b. *Partial Perticipation*, yaitu keberadaan suatu *entity* tidak tergantung pada hubungan dengan *entity* lain dalam *Entity Relationship* digambarkan dengan satu garis penghubung.

6. *Representasi* dari *entity set*, *entity set* dipresentasikan dalam bentuk *table* dan nama yang *unique* setiap *table* terdiri dari jumlah kolom masing-masing kolom diberi nama *unique* pula. *Entity set* terbagi atas:

a. *Strong entity set*, *entity set* yang satu atau lebih atributnya digunakan oleh *entity set* lain sebagai *key* digambarkan dengan empat persegi panjang.

b. *Weak entity set, entity set* yang dependent terhadap *strong entity set*, keadaan *entity set* tersebut tergantung keberadaan *entity* lain. *Entity* lain itu disebut *indentifying owner* dan relationshipnya disebut *identifying relationship*, *weak entity* selalu mempunyai *total participation constraint* dengan *identifying owner weak entity* digambarkan dengan persegi bertumpuk.

2.2. Penelitian Terkait

Menurut Choliviana, dkk (2012:1) mengemukakan bahwa:
Melalui media *website*, organisasi dapat memberikan informasi sistem pendaftaran kepada anggota dengan cepat dan mudah.Manfaat dari penerapan pendaftaran berbasis web ini akan memberikan gambaran tentang bagaimana teknik sistem registrasi yang dibutuhkan dalam perkembangan teknologi saat ini.

Menurut Hendarti, dkk (2009:155) mengemukakan bahwa:
Dengan penerapan sistem informasi akan dengan cepat mengetahui apa saja yang dibutuhkan dan yang terjadi pada perusahaan dalam waktu singkat. Sehingga dengan adanya informasi maka pihak perusahaan dapat mengambil keputusan yang tepat atas apa yang terjadi. Dan pada akhirnya dapat memotong banyak biaya yang tidak diperlukan dan memperbesar keuntungan perusahaan. Gaya hidup masyarakat saat ini terutama kaum muda, sangat menyukai hal-hal yang serba instan. Artinya mereka suka dengan segala sesuatu yang praktis, cepat, efektif, dan efisien. Maka dari itu, dibuatnya sistem informasi pada suatu perusahaan dapat memenuhi semua tuntutan masyarakat saat ini, dimana mereka dapat mengakses informasi komunitas apa saja tanpa banyak terkendala jarak dan waktu.