BAB II

LANDASAN TEORI

2.1. Tinjauan Pustaka

Dalam proses pengolahan data dan penyediaan informasi akademik sekolah perlu digunakannya suatu alat bantu dalam hal ini berupa aplikasi komputerisasi yang dapat mempermudah dan mempercepat sistem informasi akademik pada SMK Tamansiswa Cikampek.

A. Konsep Dasar Sistem Informasi

Menurut Autanto dalam Puspita dan Anggadini (2011:15) mengemukakan bahwa "Sistem informasi merupakan komponen-komponen dari subsistem yang saling berhubungan dan bekerja sama secara harmonis untuk mencapai satu tujuan yaitu mengolah data menjadi informasi".

Menurut Kristanto (2008:13) "Sebuah sistem informasi merupakan kumpulan dari perangkat keras dan perangkat lunak komputer serta perangkat manusia yang akan mengolah data menggunakan perangkat keras dan perangkat lunak tersebut."

Menurut Burch dan Grudnitski dalam Puspita dan Anggadini (2011:20) mengemukakan bahwa sistem informasi terdari dari komponen-komponen yang disebutnya dengan istilah blok bangunan (*buiding block*), yaitu:

1. Blok Masukan (*Input Block*)

   Input yang mewakili data yang masuk ke dalam sistem informasi. Input disini dapat termasuk metode-metode dan media untuk menangkap data yang akan dimasukkan, yang dapat berupa dokumen-dokumen dasar.

2. Blok Model (*Model Block*)

   Terdiri dari kombinasi prosedur, logika dan model matematik yang akan memanipulasi data input dan data yang tersimpan dalam basis data dengan cara yang sudah tertentu untuk menghasilkan keluaran yang diinginkan.

3. Blok Keluaran (*Output Block*)

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

4. Blok Teknologi (*Technology Block*)

   Teknologi merupakan "kotak alat" dalam sistem informasi yang digunakan untuk menerima input, menjadikan model, menyimpan dan mengakses data, menghasilkan dan mengirim keluaran serta membantu pengendalian dari sistem secara keseluruhan. Teknologi terdiri dari tiga bagian utama, yaitu teknisi (*humanware* atau *brainware*), perangkat lunak (*software*) dan perangkat keras (*hardware*).

5. Blok Basis Data (*Database Block*)

   Merupakan kumpulan dari data yang saling berhubungan satu dengan yang lainnya, ytersimpan diperangkat keras komputer dan digunakan perangkat lunak untuk memanipulasinya.

6. Blok Kendali (*Control Block*)

   Beberapa pengendalian perlu dirancang untuk mencegah kerusakan, kegagalan sistem yang mungkin terjadi.

B. *Unified Modelling Language*

Dalam suatu proses pengembangan perangkat lunak, merancang adalah menemukan suatu cara untuk menyelesaikan masalah. Salah satu *tool* untuk merancang pengembangan perangkat lunak adalah UML.

Menurut Nugroho (2010:6) "UML (*Unified Modeling Language*) adalah 'bahasa' pemodelan untuk sistem atau perangkat lunak yang berparadigma 'berorientasikan objek'".

UML sendiri juga memberikan standar penulisan sebuah sistem *blue print*, yang meliputi konsep bisnis proses, penulisan kelas-kelas dalam bahasa program yang spesifik, skema *database*, dan komponen-komponen yang diperlukan dalam sistem *software*.

UML mempunyai sejumlah elemen grafis yang bisa dikombinasikan menjadi diagram. Karena ini merupakan sebuah bahasa, UML mempunyai aturan untuk menggabungkan elemen-elemen tersebut. Beberapa diagram UML yaitu:

1. *Use Case Diagram*

   Use Case Diagram secara defitif sesungguhnya merupakan "urutan aksi-aksi (sering) disebut juga sebagai *flow of events*) yang akan dilakukan sistem atau perangkat lunak untuk memberikan hasil atau nilai tertentu pada masing-masing *actor*" (Nugroho, 2010:86).

   Menurut Whitten dalam Widodo (2011:21) mengartikan bahwa "*Use Case* sebagai urutan langkah-langkah yang secara tindakan saling terkait (skenario), baik terotomatisasi maupun secara manual, untuk tujuan melengkapi satu tugas bisnis tunggal".

Use Case bekerja dengan cara mendeskripsikan tipikal antara pengguna sebuah sistem dengan sistemnya sendirimelalui sebuah cerita bagaimana sebuah sistem dipakai. Urutan langkah-langkah yang menerangkan antara pengguna dan sistem disebut *scenario*. Setiap *scenario* mendeskripsikan urutan kejadian. Setiap urutan siinisialisasi oleh orang, sistem yang lain, perangkat keras atau urutan waktu. Dengan demikian secara singkat bisa dikatakan.

2. *Activity Diagram*

   Menurut Rosa dan Shalahudin (2014:161) "Activity diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak". *Activity Diagram* mempunyai peran seperti halnya *flowchart*, akan tetapi perbedaannya dengan *flowchart* adalah *activity diagram* bisa mendukung perilaku pararel sedangkan *flowchart* tidak bisa.

3. *Component Diagram*

   Diagram komponen atau *component diagram* dibuat untuk menunjukan organisasi dan ketergantungan diantara komponen dalam sebuah sistem. Diagram komponen fokus pada komponen sistem yang dibutuhkan dan ada di dalam sistem. (Rosa dan Shalahudin, 2014:148).

   Component diagram sangat berkaitan dengan *interface*. *Interface* dalam konsep UML adalah serangkaian operasi yang menspesifikasikan perilaku sebuah *class*. *Component diagram* mengandung *component*, *interface* dan *relationship*. *Component diagram* merepresentasikan dunia riil *item* yaitu

component software. *Component software* menetap di komputer dan bukan di benak para analis.

4. Deployment Diagram

   Deployment Diagram menunjukkan konfigurasi atau tata letak sebuah sistem secara fisik, menampakkan bagian-bagian *software* yang berjalan pada bagian-bagian *hardware*. *Deployment Diagram* menyediakan bagaimana gambaran bagaimana sistem secara fisik akan terlihat. (Rosa dan Shalahudin, 2014:148).

C. *Entity Relationship Diagram (ERD)*

Model ER menggambarkan data yang terlibat dalam organisasi, hubungan objek serta dapat digunakan untuk mengembangkan desain awal database. *Entity Relationship Diagram* menggambarkan lebih sistematis himpunan entitas dan himpunan relasi yang masing-masing dilengkapi dengan atribut-atribut yang merepresentasikan seluruh fakta dari ‘dunia nyata’. (Fathansyah, 2012:81)

Model ER sangat penting terutama perannya dalam desain basis data. Model ER menyediakan konsep yang memungkinkan untuk berpindah dari deskripsi apa yang pengguna inginkan pada basis data, untuk menjelaskan lebih rinci dan dapat diimplementasikan dalam DBMS (*Database Management System*).

Model ER digambarkan dengan ERD (*Entity Relationship Diagram*). ERD adalah kesatuan bentuk logika yang dipakai untuk analisa dan desain *database*. ERD menggambarkan arti dari aspek data. Di dalam pembuatan ERD perlu diperhatikan penentuan suatu konsep apakah merupakan suatu *entity*, *atribut*, atau *relationship*. Menurut (Fathansyah, 2012 : 82) eberapa komponen ERD yang digunakan dalam merancang suatu sistem yaitu:

1. Entitas (*Entity*)

   Entitas (*Entity*) adalah objek yang ada dan dapat dibedakan dengan objek lainnya. Sering lebih berguna untuk mengidentifikasi koleksi entitas-entitas yang serupa yang disebut himpunan entitas (*entity set*). Himpunan entitas adalah kumpulan entitas bertipe sama.

2. *Relationship*

   Relationship adalah asosiasi di antara dua entitas atau lebih. Sebagaimana dengan entitas, kumpulan relationship serupa disebut himpunan relationship (*relationship set*). Relasi menunjukkan adanya hubungan di antara sejumlah entitas yang berasal dari himpunan entitas yang berbeda.

3. Atribut

   Atribut adalah karakteristik dari *entity* atau *relationship*, yang menyediakan penjelasan detail tentang *entity* atau *relationship* tersebut. Atribut terdiri dari beberapa jenis yaitu:

   a. *Key Attribut*e

      Atribut yang digunakan untuk menentukan suatu *entity* secara unik.

   b. *Attribute Simple*

      Atribut yang bernilai tunggal.

   c. *Attribute Multivalue*

      Atribut yang memiliki sekelompok nilai untuk setiap *instan entity*.

   d. *Attribute Composite*

Suatu atribut yang terdiri dari beberapa atribut yang lebih kecil yang mempunyai arti tertentu.

e. *Attribute Derivative*

Suatu atribut yang dihasilkan dari atribut yang lain

4. Kardinalitas (Derajat relasi)

Kardinalitas relasi menunjukkan jumlah maksimum entitas yang dapat berelasi dengan entitas pada himpunan entitas yang lain. Kardinalitas relasi merujuk kepada hubungan maksimum yang terjadi dari himpunan entitas yang satu ke himpunan entitas yang lain dan begitu juga sebaliknya. Kardinalitas di antara dua himpunan entitas (misalnya A dan B) dapat berupa:

a. *One to One* (1:1)

Setiap entitas pada himpunan entitas A berhubungan dengan paling banyak dengan satu entitas pada himpunan entitas begitu juga sebaliknya setiap entitas pada himpunan entitas B berhubungan dengan paling banyak dengan satu entitas pada himpunan entitas A.

b. *One to Many* (1:N)

Setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B, tetapi tidak sebaliknya, dimana setiap entitas pada himpunan entitas B berhubungan dengan paling banyak dengan satu entitas pada himpunan entitas A.

c. *Many to One* (N:1)

Setiap entitas pada himpunan entitas A berhubungan dengan paling banyak dengan satu entitas pada himpunan entitas B, tetapi tidak sebaliknya, dimana setiap entitas pada himpunan entitas A berhubungan dengan paling banyak satu entitas pada himpunan entitas B.

d. *Many to Many* (N:N)

setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B, demikian juga sebaliknya, di mana setiap entitas pada himpunan entitas B dapat berhubungan dengan banyak entitas pada himpunan entitas A.

D. *Logical Record Stucture* (*LRS*)

> Menurut Wulandari (2013:15) menyatakan bahwa :
> *Logical Record Structure* dibentuk dengan nomor dan tipe *record*, beberapa tipe *record* digambarkan oleh kotak-kotak persegi panjang dan dengan nama yang unik. *LRS* terdiri dari *link-link* diantara tipe *record*. *Link* ini menunjukan arah dari satu tipe *record* lainnya. banyak *link* dari *LRS* yang diberi tanda *field-field* yang kelihatan pada kedua *link type record*.

Berikut tahapan transformasi ERD ke LRS :

1. *Entity Relationship Diagram* harus diubah ke bentuk *LRS* (struktur *record* secara logik). Dari bentuk *LRS* inilah yang nantinya dapat ditransformasikan ke bentuk relasi tabel.

   Dalam kaitannya dengan konversi ke *LRS*, untuk perubahan yang terjadi mengikuti aturan-aturan sebagai berikut :

   a. Setiap entitas diubah ke bentuk kotak dengan nama entitas, berada di luar kotak dan atribut berada di dalam kotak.

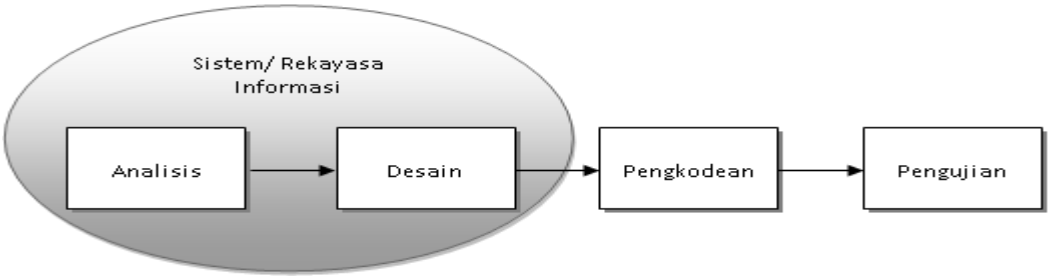
b. Seluruh *relationship* kadang disatukan, dalam sebuah kotak bersama entitas, kadang sebuah kotak bersama-sama dengan entitas, kadang disatukan dalam sebuah kotak tersendiri.

c. Konversi *LRS* ke relasi tabel adalah bentuk pernyataan data secara grafis dimensi, yang terdiri dari kolom dan baris. Relasi adalah bentuk visual dari sebuah *file*, dan tiap *truple* dalam sebuah *field*, atau yang dalam bentuk lingkaran diagram *Entity Relationship* dikenal dengan sebutan atribut.

E. Konsep Dasar Model Pengembangan Sistem

Menurut Rosa dan Shalahuddin (2014:31) "Model *waterfall* adalah model SDLC (*System Development Life Cycle*) yang paling sederhana. Model ini hanya cocok untuk pengembangan perangkat lunak dengan spesifikasi yang tidak berubah-ubah".

Model SDLC air terjun (*waterfall*) sering juga disebut model sekuensial linier atau alur hidup klasik. Model air terjun menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau terurut dari mulai analisis, desain, pengodean, pengujian dan tahap pendukung. Berikut adalah gambar model air terjun:



Sumber: Rosa dan Shalahudin (2014:29)

Gambar II.1. ilustrasi model *waterfall*

Pembuatan model *waterfall* tentu tidak terlepas dari beberapa tahapan yang harus dikerjakan secara terstruktur. Lebih jelas tahapan-tahapan pembuatan model *waterfall* adalah sebagai berikut :

1. Analisa Kebutuhan Perangkat Lunak

   Proses pengumpulan kebutuhan dilakukan secara intensif untuk mespesifikasikan kebutuhan perangkat lunak agar dapat dipahami perangkat lunak seperti apa yang dibutuhkan pengguna. Spesifikasi kebutuhan perangkat lunak pada tahap ini perlu 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 perangkat lunak dari tahap analisis kebutuhan ke representasi desain agar dapat diimplementasikan menjadi program pada tahap selanjutnya. Desain prangkat lunak yang dihasilkan pada tahap ini juga perlu didokumentasikan.

3. Penulisan Kode Program

   Desain harus ditranslasikan kedalam program perangkat lunak. Hasil dari tahap ini adalah program komputer sesuai dengan desain yang telah dibuat pada tahap desain.

4. Pengujian

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

5. Pendukung (*support*) atau pemeliharaan (*maintenance*)

   Tidak menutup kemungkinan sebuah perangkat lunak mengalami perubahan ketika sudah dikirimkan ke pengguna. Perubahan bisa terjadi karena adanya kesalahan yang muncul dan tidak terdeteksi saat pengujian. 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 yang terjadi sangat jarang model *waterfall* dapat dilakukan sesuai alurnya karena sebab berikut :

1. Perubahan spesifikasi perangkat lunak terjadi ditengah 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.

Model *waterfall* sangat cocok digunakan untuk kebutuhan pelanggan sudah sangat dipahami dan kemungkinan terjadi perubahan kebutuhan selama

pengembangan perangkat lunak kecil. Hal positif dari model *waterfall* adalah struktur tahap pengembangan sistem jelas, dokumentasi dihasilkan di setiap tahap pengembangan, dan sebuah tahap dijalankan setelah tahap sebelumnya selesai dijalankan.

2.2. Penelitian Terkait

Berikut ini beberapa tinjauan penelitian terdahulu (jurnal) yang dapat memperkuat alasan dibuatnya sistem informasi akademik pada SMK Tamansiswa Cikampek, diantaranya:

> Menurut Riyadi, dkk dalam jurnalnya (2009:2) dengan dibangunnya sistem informasi berbasis *website* diharapkan dapat memberikan kemudahan dalam akltivitas-aktivitas akademik khususnya seperti proses pencarian data guru yang dibutuhkan oleh siswa atau orang tua/wali tidak perlu lagi mengantri di tata usaha dan pencarian materi ajar dapat dilakukan kapan saja.

Menurut Dengen dan Marisa dalam jurnalnya (2009:29) sistem informasi akademik berbasis *web* ini dirancang sebagai solusi untuk mengelola bagian akademik dalam penyajian laporan nilai serta keaktifan siswa dan sistem informasi ini bersifat *intern*.

Berdasarkan tinjauan penelitian terkait diatas, dengan menggunakan dan memanfaatkan Sistem Informasi Akademik Berbasis *Web* diharapkan guru, siswa, orang tua siswa, dan calon siswa SMK Tamansiswa Cikampek memperoleh beberapa keuntungan diantaranya kemudahan mencari informasi, akses informasi menjadi cepat, perhitungan menjadi akurat, mengurangi penggunaan kertas. Banyak biaya yang dapat dipangkas, dan fleksibilitas bertambah. Dengan

pemanfaatan informasi berbasis *web* di harapkan dapat meningkatkan kualitas pendidikan di Indonesia.