BAB II

LANDASAN TEORI

2.1. Tinjauan Pustaka

A. Konsep Dasar Sistem Informasi

Menurut Zakiyudin (2011:9), sistem informasi adalah suatu sistem didalam suatu organisasi mempertemukan kebutuhan pengolahan transaksi, mendukung operasi, dan bersifat manajerial dan kegiatan strategi dari suatu organisasi dan pihak luar tertentu dengan laporan yang diperlukan.

Menurut Sutabri (2012:38), sistem Informasi adalah suatu sistem didalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian yang mendukung fungsi operasi organisasi yang bersifat manajerial dengan kegiatan strategi dari suatu organisasi untuk dapat menyediakan laporan – laporan yang diperlukan oleh pihak tertentu.

B. Pendaftaran Siswa Baru

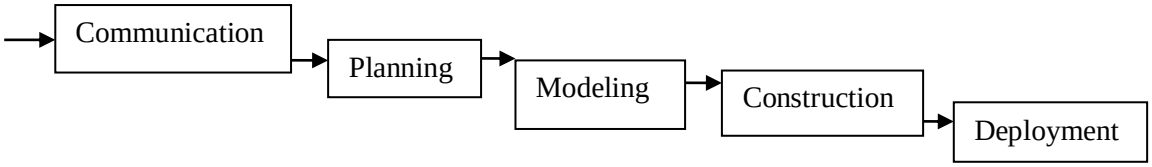
Pengertian pendaftaran disini pada dasarnya hanya untuk memperlancar dan mempermudah dalam proses pendaftaran siswa siswi baru, pendataan dan pembagian kelas seorang siswa siswi. Sehingga dapat terorganisir, teratur dengan cepat dan tepat dengan beberapa persyaratan yang telah ditentukan oleh sekolah. Proses pendaftaran siswa baru merupakan salah satu kewajiban pihak sekolah dan Dinas Pendidikan setiap tahun ajaran baru.

C. Metode Waterfall

1. Konsep Dasar Model Pengembangan Sistem

Model *waterfall* adalah model klasik yang bersifat sistematis, berurutan dalam membangun *software.* Berikut ini ada dua gambaran dari *waterfall* model.

Fase-fase dalam model *waterfall* menurut referensi pressman.



Sumber: Pressman (2010:29)

Gambar II.1.

Waterfall Pressman

a. *Communication*

Langkah ini merupakan analisis terhadap kebutuhan *software,* dan tahap untuk mengadakan pengumpulan data dengan melakukan pertemuan dengan pihak sekolah, maupun mengumpulkan data-data tambahan baik yang ada di jurnal, artikel, maupun dari internet.

b. *Planning*

Proses *planning* merupakan lanjutan dari proses *communication* (*analysis requirement)*. Tahapan ini akan menghasilkan dokumen *user requirement* atau bisa dikatakan sebagai data yang berhubungan dengan keinginan *user* dalam pembuatan *software*, termasuk rencana yang akan dilakukan.

c. *Modeling*

Proses *modeling* ini akan menerjemahkan syarat kebutuhan ke sebuah perancangan *software* yang dapat diperkirakan sebelum dibuat *coding*. Proses ini berfokus pada rancangan struktur data. Arsitektur *software,*

representasi *interface*, dan detail (algoritma) procedural. Tahapan ini akan menghasilkan dokumen yang disebut *software requirement*.

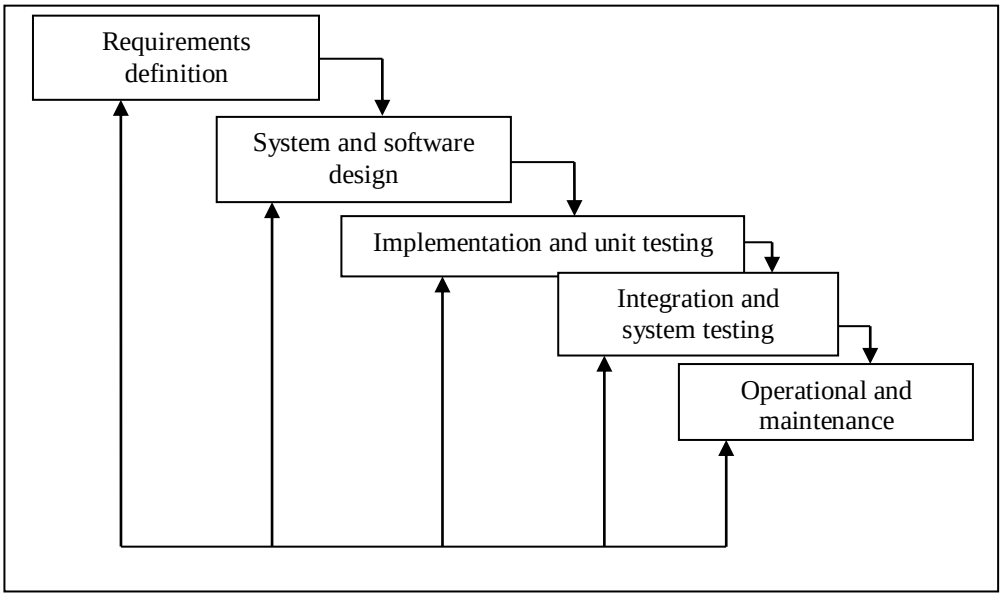
d. *Construction*

Construction merupakan proses membuat kode. *Coding* atau pengkodean merupakan penerjemahan desain dalam bahasa yang bisa di kenali oleh komputer. *Programmer* akan menerjemahkan transaksi yang diminta oleh *user*. Tahapan inilah yang merupakan tahapan secara nyata dalam mengerjakan suatu *software*, artinya penggunaan komputer akan dimaksimalkan dalam tahapan ini. Setelah pengkodean selesai maka akan dilakukan *testing* terhadap sistem yang telah dibuat tadi. Tujuan *testing* adalah menemukan kesalahan-kesalahan terhadap sistem tersebut untuk demikian bisa diperbaiki.

e. *Deployment*

Tahapan ini bisa dikatakan final dalam pembuatan sebuah *software* atau sistem. Setelah melakukan analisis, desain dan pengkodean maka sistem yang sudah jadi akan digunakan oleh *user*. Kemudian *software* yang telah dibuat harus dilakukan pemeliharaan secara berkala.

Sedangkan fase-fase model waterfall menurut referensi Sommerville (2011) :



Sumber: Sommerville (2011:30)

Gambar II.2.

Waterfall Sommerville

1. *Requirements Analysis and Definition*

   Mengumpulkan kebutuhan secara lengkap kemudian di analisis dan didefinisikan kebutuhan yang harus dipenuhi oleh *software* yang akan di bangun. Hal ini sangat penting, mengingat *software* harus dapat berinteraksi dengan elemen-elemen yang lain seperti *hardware*, *database*, dsb. Tahap ini sering disebut dengan *Project Definition*.

2. *System and Software Design*

   Proses pencarian kebutuhan diintensifkan dan difokuskan pada *software.* Untuk mengetahui sifat dari program yang akan dibuat, maka para *software engineer* harus mengerti tentang domain informasi dari *software*, misalnya fungsi yang dibutuhkan, *user interface*, dsb. Dari dua aktivitas

tersebut (pencarian kebutuhan system dan *software)* harus didokumentasikan dan ditunjukkan kepada *user.*

Proses *software design* untuk mengubah kebutuhan-kebutuhan di atas menjadi representasi ke dalam bentuk "*blueprint*" *software* sebelum *coding* dimulai. Desain harus dapat mengimplementasikan kebutuhan yang telah disebutkan pada tahap sebelumnya. Seperti dua aktifitas sebelumnya, maka proses ini juga harus didokumentasikan sebagai konfigurasi dari *software.*

3. *Implementation and Unit Testing*

   Desain program diterjemahkan ke dalam kode-kode dengan menggunakan bahasa pemrograman yang sudah ditentukan. Program yang dibangun langsung diuji baik secara unit.

4. *Integration and System Testing*

   Untuk dapat dimengerti oleh mesin, dalam hal ini adalah komputer, maka desain tadi harus diubah bentuknya menjadi bentuk yang dapat dimengerti oleh mesin, yaitu ke dalam bahasa pemrograman melalui proses *coding*. Tahap ini merupakan implementasi dari tahap design yang secara teknis nantinya dikerjakan oleh *programmer*. Penyatuan unit-unit program kemudian diuji secara keseluruhan (*system testing*).

5. *Operation and Maintenance*

   Sesuatu yang dibuat haruslah diujicobakan. Demikian juga dengan *software.* Semua fungsi-fungsi *software* harus diujicobakan, agar *software* bebas dari *error*, dan hasilnya harus benar-benar sesuai dengan kebutuhan yang sudah didefinisikan sebelumnya.

Pemeliharaan suatu *software* diperlukan, termasuk didalamnya adalah pengembangan, karena *software* yang dibuat tidak selamanya hanya seperti itu. Ketika dijalankan mungkin saja masih ada *error* kecil yang tidak ditemukan sebelumnya, atau ada penambahan fitur-fitur yang belum ada pada *software* tersebut. Pengembangan diperlukan ketika adanya perubahan dari eksternal perusahaan seperti ketika ada pergantian sistem operasi, atau perangkat lainnya.

Kelebihan dari model ini adalah selain karena pengaplikasian menggunakan model ini mudah, kelebihan dari model ini adalah ketika semua kebutuhan sistem dapat didefinisikan secara utuh, eksplisit, dan benar di awal proyek, maka *Software Engineering* (SE) dapat berjalan dengan baik dan tanpa masalah. Meskipun seringkali kebutuhan sistem tidak dapat didefinisikan se-eksplisit yang diinginkan, tetapi paling tidak, problem pada kebutuhan di awal proyek lebih ekonomis dalam hal uang (lebih murah), usaha, dan waktu yang terbuang lebih sedikit jika dibandingkan problem yang muncul pada tahap-tahap selanjutnya.

Kekurangan yang utama dari model ini adalah kesulitan dalam mengakomodasi perubahan setelah proses dijalani. Fase sebelumnya harus lengkap dan selesai sebelum mengerjakan fase berikutnya.

D. Metode Terstruktur

Definisi program adalah penyataan yang disusun menjadi satu kesatuan prosedur yang berupa urutan langkah yang disusun secara logis dan sistematis untuk menyelesaikan masalah.

Sedangkan definisi pemrograman adalah proses mengimplementasikan urutan langkah untuk menyelesaikan suatu masalah dengab bahasa pemrograman dan terstruktur dapat berarti terpola, bentuk yang mengikuti aturan tertentu, juga dapat berarti sesuatu yang sistematis. Jadi Pemrograman Terstruktur adalah metode untuk mengorganisasikan dan membuat kode-kode program supaya mudah untuk dimengerti, mudah diuji dan dimodifikasi secara terpola dan sistematis.

Dengan ide pemrograman terstruktur diharapkan dapat membantu manajemen *source code* (kode program) sehingga program mudah untuk dikelola bagi kepentingan selanjutnya. Sedangkan tujuan utama pemrograman terstruktur adalah agar program-program besar menjadi lebih mudah ditelusuri alur logikanya, mudah untuk dimodifikasi (dikembangkan) dan mudah pula untuk ditemukan bagian yang salah ketika program sedang diuji. Selain itu prinsip utama pemrograman terstruktur adalah jika suatu proses telah sampai pada suatu baris tertentu, maka proses selanjutnya tidak boleh melompat mundur ke baris sebelumnya, kecuali untuk proses berulang *(repetition/looping)*, dan program diformulasikan ke dalam modul-modul dan secara hirarkis.

Untuk membantu dan menjadi dihasilkannya pemrograman yang terstruktur dapat digunakan beberapa metode/alat di antaranya adalah:

a) Desain *Top Down*

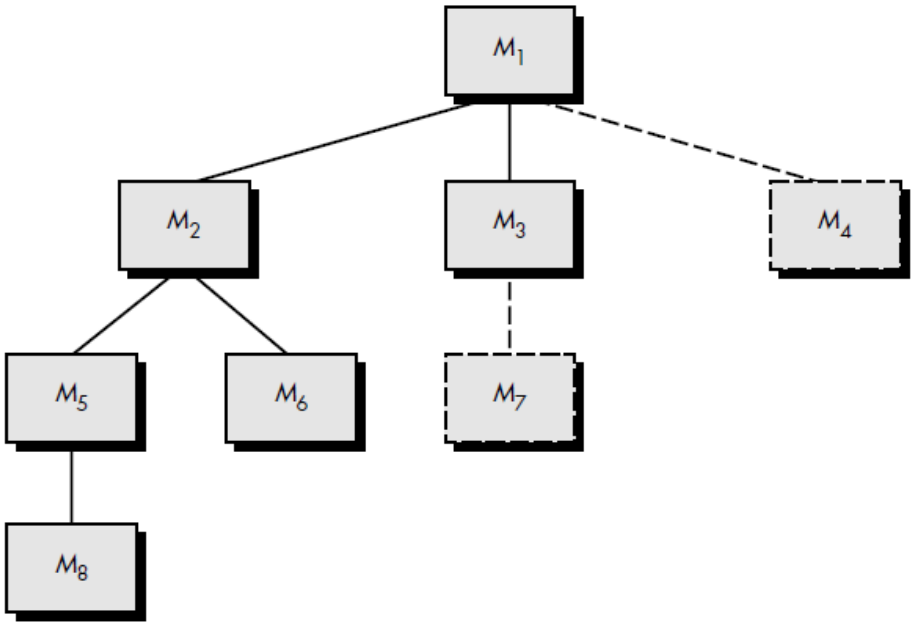
Penyusunan program dengan disain *Top Down* mengacu kepada tujuan program secara menyeluruh, bukan pada bagaimana cara mencapainya. Setelah tujuan terdefinisi dengan jelas, barulah dibuat garis besar proses yang akan dilaksanakan program. Kemudian secara bertahap garis besar proses diuraikan

menjadi tahapan yang lebih terperinci, sehingga akhirnya semua program selesai. Pada disain *Top-Down* akan dihasilkan modul-modul dengan ciri sebagai berikut :

1) Modul mengimplementasikan proses tunggal, logis dan dapat berdiri sendiri dan mudah dipahami.
2) Modul harus independen. Sebuah modul ditulis tidak boleh bergantung pada implementasi di modul lainnya.
3) Modul akan relatif pendek, disarankan kode modul tidak lebih dari satu halaman.

Keuntungan dari disain *Top-Down* adalah bahwa setiap tahap pemrograman yang ada menjadi sederhana, karena setiap tingkat mengabaikan detail dari tingkat yang lebih rendah. Dengan cara ini, sebuah proses yang kompleks dipecah kedalam sebuah urutan langkah yang relatif mudah, hasil akhir adalah berupa sebuah modul yang lebih sederhana.

Metode perancangan program dengan disain *Top Down* adalah sebagai berikut :



Sumber: Pressman (2010 : 59)

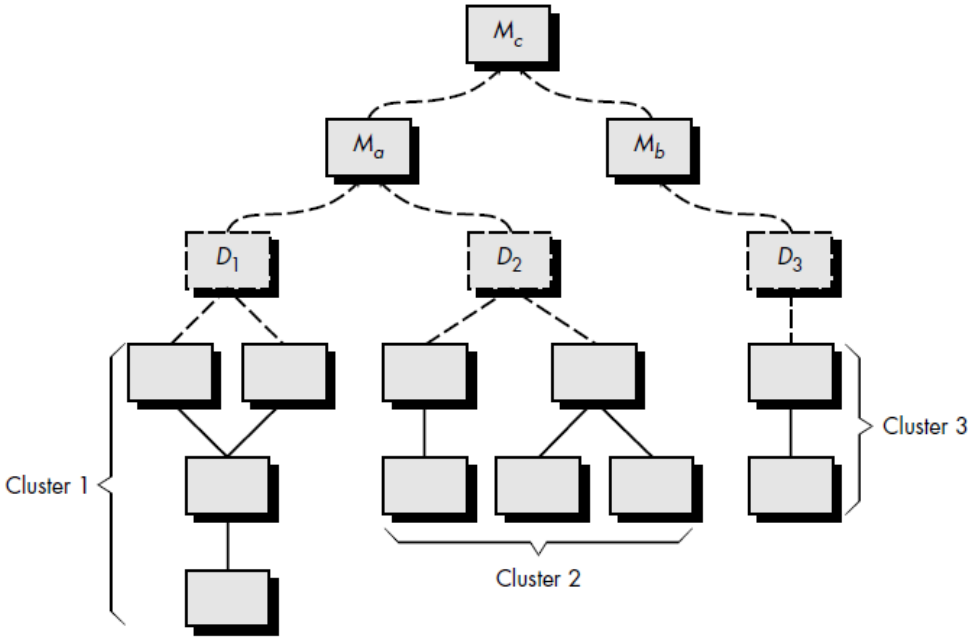
Gambar II.3.
Desain *Top Down*

b) Desain *Bottom Up*

Penyusunan program dengan disain *bottom-up* dilakukan dengan menyelesaikan semua pengkodean untuk modul-modul tingkat paling rendah, kemudian diuji. Jika benar, maka diintegrasikan ke tingkat berikutnya dan diuji lagi. Pengujian Bottom-Up membutuhkan beberapa program kendali, yang memanggil modul dan memasukkan data penambahan signifikan didalam progam.

Pada dasarnya, disain *Bottom-Up* merupakan kebalikan dari disain *Top-Down*, hanya saja pendefinisian tujuan dilakukan tetap pada awal penyusunan program.

Metode perancangan program dengan disain *Bottom Up* adalah sebagai berikut :



Sumber: Pressman (2010 : 78)

Gambar II.4.

Metode *Bottom UP*

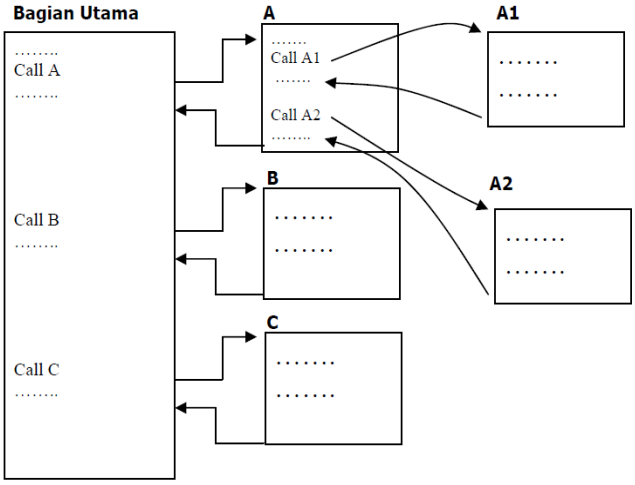
c) Disain *Modular*

Penyusunan program dengan disain modular dilakukan dengan memecah program kedalam beberapa modul, dimana setiap modul menunjukkan fungsi dan tugas tunggal. Dengan membagi masalah kedalam beberapa modul, maka masalah akan menjadi sederhana dan mudah dipahami. Disain dengan metode modular merupakan implementasi dan pengembangan dari disain *Top-Down*.

Setiap program memiliki sebuah modul program utama, yang mengontrol semua proses yang terjadi, termasuk mengirimkan kontrol program kepada sub-modul utuk melakukan suatu fungsi tertentu.

Pemrograman modular diterapkan dengan menggunakan *sub-routine*, yaitu sebuah kumpulan perintah yang melakukan tugas pemrosesan yang terbatas.

Metode perancangan disain program dengan metode *modular* adalah sebagai berikut :



Sumber: Pressman (2010 : 92)

Gambar II.5.

Disain *Modular*

Hal lain yang perlu diperhatikan di dalam struktur pemrograman, yaitu masalah penulisan program yang interaktif dan program yang efisien. Program dikatan interaktif, jika penulisan tersebut dapat dipergunakan oleh pemakai secara mudah serta pemakai dapat mengerti tentang proses yang sedang dilakukan oleh program. Terdapat beberapa kiat dalam membuat program yang interaktif, yaitu:

1) Program harus dapat melakukan validasi terhadap setiap data yang masuk.
2) Program harus dapat mengecek setiap kemungkinan yang penting, yang akan muncul pada data masukan

3) Buatlah format masukan sesederhana mungkin

4) Buatlah program yang dapat memberikan tanda bahwa data yang dikehendaki sudah terpenuhi

5) Berikan label keterangan pada setiap keluaran dan pesan apabila terjadi kesalahan

6) Berikan pesan, apabila terdapat program yang melakukan suatu proses yang memerlukan suatu waktu tunggu

Selain program yang interaktif, kita juga harus menciptakan suatu konsep penulisan yang baik dan benar, yang disebut sebagai efisiensi program.

1. Pengenalan *Personal Home Page* (PHP)

Anhar (2010:3), "PHP merupakan bahasa pemrograman *web server-side* yang bersifat *open source*. PHP merupakan *script* yang terintegrasi dengan HTML dan berbeda pada *server* (*server-side* HTML *embedded scripting*). PHP adalah *script* yang digunakan untuk membuat halaman *website* yang dinamis". Pemrograman PHP sangat cocok dikembangkan dalam lingkungan *web*, karena PHP bisa dilekatkan pada *script* HTML atau sebaliknya. PHP dikhususkan untuk pengembangan *web* dinamis. Hal tersebut bergantung pada permintaan *client browser*-nya (bisa menggunakan *Browser Opera, Internet Explorer, Mozzila* dan lain-lain). Pada umumnya, pembuat web dinamis berhubungan erat dengan *database* sebagai sumber data yang akan ditampilkan.

PHP tergolong juga sebagai bahasa pemrograman yang berbasis *server* (*server side scripting*). Ini berarti bahwa semua *script* PHP diletakkan di *server* dan diterjemahkan oleh web *server* terlebih dahulu.

2. C*ascading Style Sheet* (CSS)

Menurut Riyanto (2011:54), "*Cascading Style Sheet* (CSS) merupakan *template* yang mengontrol pemformatan tag HTML (*HyperText Markup Language*) pada sebuah halaman *web"*. Konsep *style sheet* mirip dengan *template* pada Microsoft Word. Anda dapat mengubah tampilan pada dokumen Word dengan mengubah format pada *style* dokumen. Begitu pula anda dapat mengubah tampilan halaman *web* dengan mengubah format pada *tag* HTML tertentu melalui *style sheet*, untuk selanjutnya mengganti spesifikasi *default* dari *browser* untuk *tag-tag* tersebut.

3. MySQL

Menurut Anhar (2010:45) "MySQL (*My Structure Query Language)* adalah salah satu *DataBase Management System* (DBMS) yang berfungsi untuk mengolah *database* menggunakan bahasa SQL".

Menurut Prasetio (2012:181) "MySQL adalah sebuah *server database open source* yang sepertinya paling populer keberadaaannya. MySQL umumnya digunakan bersamaan dengan *script* PHP untuk membuat *aplikasi server* yang dinamis dan *powerful*."

4. XAMPP

Menurut Imansyah (2010:4) "Xampp adalah installer yang membundel Apache, PHP dan MySQL untuk windows dalam satu paket. Dengan menginstal XAMPP, Anda bisa menjadikan komputer sebagai server."

E. UML (*Unifield Modelling Language*)

Menurut Rossa dan shalahudin (2011:117) untuk memodelkan perangkat lunak yang menggunakan pemograman *procedural* atau structural, kemudian juga ada *State Transition Diagram* yang digunakan untuk memodelkan system real time (waktu nyata). UML muncul karena adanya kebutuhan pemodelan *visual* untuk menspesifikasikan, menggambarkan, membangun dan dokumentasi dari system perankat lunak. UML merupakan bahasa *visual* untuk pemodelan dan komunikasi mengenai sebuah system dengan menggunakan diagram dan *teks-teks* pendukung. Menurut Rossa dan Shalahuddin (2011:62) Pemograman terstruktur adalah konsep atau paradigm atau sudut pandang pemograman yang membagi-bagi program berdasarkan fungsi-fungsi atau prosedur-prosedur yang dibutuhkan program komputer. Modul-modul (pembagian program) biasanya dibuat dengan mengelompokan fungsi-fungsi dan posedur-prosedur yang diperlukan sebuah proses tertentu. Oleh karna itu pemodelan pada pemograman terstruktur lebih focus bagaimana memodelkan data dan fungsi-fungsi atau prosedur-prosedur yang harus dibuat. Jenis paradigm pemograman yang digunakan dapat dideteksi dari bahasa pemograman apa yang digunakan untuk membuat program, baru setelah itu ditentukan paradigm pemograman apa yang akan digunakan. Adapun pemograman terstruktur yang akan dibahas yaitu :

1. *Use Case Diagram*

Menurut Rossa dan Shalahuddin (2011:130) *Use case diagram* merupakan pemodelan untuk melakukan system informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih actor dengan system informasi yang akan dibuat. Secara kasar, *use case* diguanakan untuk mengetahui

fungsi apa saja yang ada didalam sebuah sistem informasi dan siapa saja yang berhak menggunkan fungsi-fungsi itu. Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami.

2. *Activity Diagram*

Menurut Rossa dan Shalahudin (2011:134) *Activity diagram* yang menggambarkan aliran kerja atau aktivitas dari sebuah Sistem atau proses bisnis. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas system bukan apa yang digunakan oleh actor, jadi aktivitas yang dapat dilakukan oleh sistem.

3. *Component Diagram*

Menurut Shalahudin (2013:148), "*Component diagram* dibuat untuk menunjukan organisasi dan ketergantungan diantara kumpulan komponen dalam sebuah sistem. Diagram komponen berfokus pada komponen sistem yang dibutuhkan dan ada didalam sistem."

Diagram komponen juga dapat digunakan untuk memodelkan hal-hal berikut:

A. *Source code* program perangkat lunak.

B. Komponen *executable* yang dilepas ke *user*.

C. Basis data secara fisik.

D. Sistem yang harus beradaptasi dengan sistem lain.

E. *Framework* sistem pada perangkat lunak merupakan kerangka kerja yang dibuat untuk memudahkan pengembangan dan pemeliharaan aplikasi.

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

A. Komponen *user interface* yang menangani tampilan.

B. Komponen *bussines procesiing* yang menangani fungsi-fungsi proses bisnis.

C. Komponen *data* yang menangani manipulasi data.

D. Komponen *security* yang menangani keamanan sistem.

4. *Deployment Diagram*

Menurut Shalahudin (2013:154), "*Deployment diagram* menunjukan konfigurasi komponen dalam proses eksekusi aplikasi.

Diagram *deployment* juga dapat digunakan dalam memodelkan hal-hal sebagai berikut:

A. Sistem tambahan (*embedded system*) yang menggambarkan rancangan *device*, *node* dan *hardware*.

B. Sistem *client server*.

C. Sistem terdistribusi murni.

D. Rekayasa ulang aplikasi.

Hardware adalah *node* yaitu nama untuk semua jenis sumber komputasi. Ada dua tipe *node* yaitu *processor* dan *device*. *Processor* adalah *node* yang bisa mengeksekusi sebuah komponen sedangkan *device* tidak bisa mengeksekusi.

Deployment diagram menggambarkan detail bagaimana komponen dideploy dalam infrastruktur sistem, dimana komponen akan terletak (pada mesin, *server* atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi *server*, dan hal-hal lain yang bersifak fisikal.

F. ERD (*Entity Relationship Diagram*)

Menurut Rosa dan Shalahuddin (2010:212) ERD (*Entity Relation Diagram*) adalah merupakan sebuah diagram yang digunakan untuk merancang

hubungan antara tabel-tabel dalam basis data. ERD merupakan dokumen data perusahaan dalam ringkasan cara dengan mengidentifikasi tipe data *entity* dan hubungan antar *entity*. Atribut merupakan properti atau bagian dari suatu *entity*. *Relationship* menggambarkan hubungan antar *entity*.

ERD terbagi tiga komponen, yaitu entitas (*entity*), atribut (*attribute*), dan relasi atau hubungan (*relation*). Secara garis besar entitas merupakan dasar yang terlibat dalam sistem. Atribut atau *field* berperan sebagai penjelas dari entitas, dan relasi atau hubungan menunjukkan hubungan yang terjadi antara dua entitas.

1. Entitas (*Entity*)

Entitas (*entity*) menunjukkan objek-objek dasar yang terkait didalam sistem. Objek dasar dapat berupa orang, benda atau hal lain yang keterangannya perlu disimpan dalam basis data. Untuk menggambarkan entitas dilakukan dengan mengikuti aturan-aturan sebgai beirkut:

a) Entitas dinyatakan dengan *symbol* persegi panjang.

b) Nama entitas dapat berupa kata bedan tunggal.

c) Nama entitas sedapat mungkin menggunakan anma yang mudah dipahami dan menyatakan maknanya dengan jelas.

2. Atribut (*attribute*)

Atribut juga sering disebut sebagai property (*property*), merupakan keterangan-keterangan yang terkait pada sebuah entitas yang perlu disimpan sebagai basis data. Atribut berfungsi sebagai penjelas sebuah entitas untuk menggambarkan atribut yang dilkaukan dengan mengikuti aturan sebagai berikut:

a) Atribut dinyatakan dengan *symbol* elipps.

b) Nama atribut dituliskan dalam *symbol* elipps.

c) Nama atribut berupa kata benda tunggal.

d) Nama atribut sedapat mungkin menggunakannama yang mudah dipahami dan dapat menyatakan maknanya dengan jelas.

e) Atribut dihubungkan denan entitas yang bersesuaian dengan menggunakan garis.

3. Relasi

Relasi atau hubungan adalah kejadian atau transaksi diantara dua entitas yang keterangannya perlu disimpan dalam basis data. Aturan penggambaran relasi antar entity:

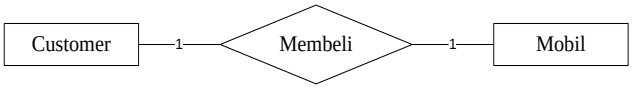
1. Relasi dinyatakan dengan *symbol* belah ketupat.
2. Nama relasi dituliskan didalam *symbol* belah ketupat.
3. Relasi menghubungkan dua entitas.
4. Nama relasi menggunakan kata kerja aktif (diawali awalan me-) tunggal.
5. Nama relasi sedapat mungkin menggunakan nama yang mudah dipahami dan dapat menyatakan maknanya dengan jelas.

4. Derajat Relasi (*Cardinalitas)*

Derajat Relasi (*Cardinalitas)* adalah menjelaskan batasan pada jumlah *entity* yang berhubungan melalui sebuah relasi yang ada. Pemetaan kardinal dapat dikategorikan menjadi tiga macam yaitu:

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

   Yaitu hubungan antara *entity* pertama dapat berhubungan dengan satu *entity* kedua dan *entity* kedua dapat berhubungan dengan *entity* pertama paling banyak satu *entity*. Contoh:



Gambar II.6.
Bagan Relasi Satu ke Satu

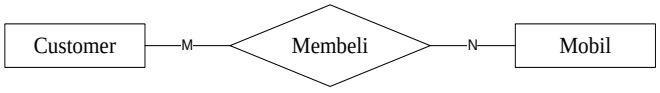
2. *One to Many* (1:M)

Entity pertama dapat berhubungan dengan sejumlah *entity* kedua, tetapi satu *entity* kedua hanya dapat berhubungan dengan satu *entity* kedua. Contoh:

Gambar II.7.
Bagan Relasi Satu ke Banyak

3. *Many to Many* (M:N)

Entitas pertama dapat berhubungan dengan banyak pada *entity* kedua, demikian pula sebaiknya. Contoh:



Gambar II.8.
Bagan Relasi Banyak ke Banyak

2.2. Penelitian Terkait

Teknologi aplikasi sistem informasi merupakan sebuah teknologi interaktif yang dapat digunakan sebagai sarana informasi yang dapat diakses oleh semua pihak, baik pihak umum maupun pihak yang mempunyai hak akses (Sukarno Putra, dkk. ISSN: 2302-5700 – http://ijns.org).

sistem informasi sebagai pendukung untuk mengambil keputusan yang tepat berdasarkan data-data yang tersedia, sehingga kinerja pengolahan data pada sekolah tersebut dapat lebih ditingkatkan (Hidayat, dkk. 2013:1).