BAB II

LANDASAN TEORI

2.1. Konsep Dasar Sistem

2.1.1. Pengertian Sistem

Menurut Jogiyanto dalam (Tabrani, 2014) pendekatan sistem yang lebih menekankan pada prosedur mendefinisikan "Suatu sistem adalah suatu jaringan kerja dari prosedur–prosedur yang saling berhubungan, berkumpul bersama–sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran yang tertentu", sedangkan pendekatan sistem yang lebih menekankan pada elemen atau komponen mendefinisikan "Sistem adalah kumpulan dari elemen–elemen yang berinteraksi untuk mencapai suatu tujuan tertentu".

Menurut Susanto (2013:22), "Sistem adalah kumpulan/group dari sub sistem/bagian/komponen apapun baik fisik ataupun non fisik yang saling berhubungan satu sama lain dan bekerja sama secara harmonis untuk mencapai satu tujuan tertentu".

Adapun sistem menurut Moscove dalam Zaki (2013:2), "Suatu sistem adalah suatu entity (kesatuan) yang terdiri dari bagian-bagian yang saling berhubungan (subsistem) untuk mencapai tujuan-tujuan tertentu".

Sutabri memberikan pengertian sistem sebagai sekelompok unsur-unsur yang erat hubungannya satu dengan yang lain, yang berfungsi bersama-sama untuk mencapai tujuan tertentu Sutabri dalam (Herliana & Rasyid, 2016)

2.1.2. Basis Data

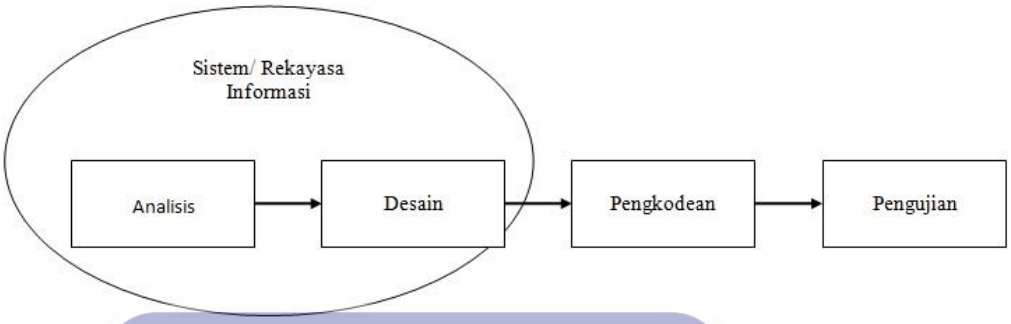
Menurut Indrajani (2015:70), "basis data adalah kumpulan data yang saling berhubungan secara logis dan didesain untuk mendapatkan data yang dibutuhkan oleh suatu organisasi". Selain itu, basis data merupakan koleksi dari data-data yang terorganisasi dengan cara sedemikian rupa sehingga data tersebut mudah disimpan dan dimanupulasi. Sebuah sistem basis data dapat memiliki beberapa basis data. Setiap basis data memiliki sejumlah objek basis data (seperti tabel, indeks, dan lain-lain). Disamping berisi atau menyimpan data, setiap basis data juga mengandung atau menyimpan definisi struktur.

Aplikasi yang digunakan pada basis data yaitu DataBase Management Sistem (DBMS) yang merupakan kumpulan program aplikasi yang digunakan untuk membuat dan mengelola basis data. DBMS berisi suatu koleksi data dan satu set program untuk mengakses data. DBMS merupakan perangkat lunak *(software)* yang menentukan bagaimana data tersebut diorganisasi, disimpan, diubah, dan diambil kembali. Perangkat lunak ini juga menerapkan mekanisme pengamanan data, pemakaian data bersama dan konsistensi data.

2.1.3. Model Pengembangan Perankat Lunak

Menurut Sukamto dan Shalahuddin (2013:25), "Rekayasa Perangkat Lunak merupakan terjemahan dari *Software Engineering*, sedikit mengalami pergeseran makna di relita dunia *industri*, bisnis, pendidikan maupun kurikulum Teknologi Informasi (TI) di tanah air. Di *industri*, *tester* dan *programmer* sering salah kaprah menyandang gelar *Software Engineer*".

Metode yang digunakan pada pengembangan perangkat lunak ini menggunakan model *waterfall* Sukamto dan Shalahudin (2013:28), yang terbagi menjadi lima tahapan, yaitu:



Sumber : Sukamto & Shalahudin, 2013:29

Gambar. II. 1 Ilutrasi Model Waterfall

1. Analisis kebutuhan perangkat lunak

   Proses pengumpulan kebutuhan dilakukan secara intensif untuk menspesifikasikan kebutuhan perangkat lunak seperti apa yang dibutuhkan oleh *user*. Spesifikasi perangkat lunak pada tahap iniperlu didokumentasikan.

2. Desain

   Desain perangkat lunak adalah proses multi langkah yang fokus pada desain pembuatan perangkat lunak termasuk struktur data, arsitektur perangkat lunak, representasi anatarmuka, dan prosedur pengodean. Tahap ini mentransaksi kebutuhan perangkat lunak dari tahap analis 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

   Desain harus ditranslasikan ke 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 dan fungsional dan memastikan bahwa bagian sudah diuji. Hal ini dilakukan untuk meminimalisir kesalahan *(error)* dan memastikan keluaran yang dihasilkan sesuai yang diinginkan menggunakan *black box.*

5. Pendukung *(Support)*

   Tidak menutup kemungkinan sebuah perangkat lunak mengalami perubahan ketika sudah dikirimkan ke *user*. Perubahan bisa terjadi 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 pengambangan mulai dari analis spesifikasi untuk perubahan perangkat lunak yang sudah ada, tapi tidak untuk membuat perngkat lunak baru.

Keuntungan menggunakan metode *waterfall* adalah prosesnya lebih terstruktur, hal ini membuat kualitas *software* baik dan tetap terjaga. Dari sisi *user* juga lebih menguntungkan, karena dapat merencanakan dan menyiapkan kebutuhan data dan proses yang diperlukan sejak awal. Penjadwalan juga menjadi lebih menentu, karena jadwal setiap proses dapat ditentukan secara pasti. Sehingga dapat dilihat jelas target penyelesaian pengembangan program.

Dengan adanya urutan yang pasti, dapat dilihat pula perkembangan untuk setiap tahap secara pasti. Dari sisi lain, model ini merupakan jenis model yang bersifat dokumen lengkap sehingga proses pemeliharaan dapat dilakukan dengan mudah.

Kelemahan menggunakan metode *waterfall* adalah bersifat kaku, sehingga sulit melakukan perubahan di tengah proses. Jika terdapat kekurangan proses/prosedur dari tahap sebelumnya, maka tahapan pengembangan harus dilakukan mulai dari awal lagi. Hal ini akan memakan waktu yang lebih lama. Karena jika proses sebelumnya belum selesai sampai akhir, maka proses selanjutnya juga tidak dapat berjalan. Oleh karena itu, jika terdapat kekurangan dalam permintaan *user* maka proses pengembangan harus dimulai kembali dari awal. Karena itu, dapat dikatakan proses pengembangan *software* dengan metode *waterfall* bersifat lambat.

2,1,4. Persediaan

> Menurut (Sari, 2018) "Persediaan merupakan salah satu perkiraan yang terpenting dalam sebuah perusahaan. Bagi perusahaan, persediaan merupakan *asset* yang cukup besar nilainya. Keberadaan persediaan dalam sebuah perusahaan mengandung implikasi dilihat dari ada atau tidaknya persediaan. Jika persediaan yang tersedia cukup besar maka dampaknya juga biaya yang dibutuhkan untuk menjaga keberadaan persediaan tidak dapat dihindari. Sebaliknya jika persediaan tidak tersedia, maka implikasi ke proses produksi dan penjualan akan menjadi terganggu. Keberadaan persediaan mempengaruhi neraca dan laporan laba rugi.
>
> Persediaan merupakan salah satu aktiva lancar yang harus dikelola dengan baik, terutama untuk perusahaan-perusahaan yang memiliki persediaan barang dagangan. Persediaan yang dimiliki perusahaan akan dapat ditentukan harga perolehan persediaan dan nilai persediaan akan disajikan di neraca. Dalam menghitung nilai persediaan perusahaan dapat mengunakan tiga metode yaitu Metode FIFO, LIFO dan Average.

2.2. Teori Pendukung

2.2.1. UML *(Unifield Modeling Language)*\

1. Konsep Dasar

"UML *(Unified Modelling Language)* adalah bahasa pemodelan untuk membangun perangkat lunak yang dibangun dengan menggunakan teknik pemrograman berorientasi objek. UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung" menurut (Retnoningsih, 2015)

Menurut Fowler dalam (Malau & Ariyanto, 2014) "*Unifield Modelling Language* (UML) adalah keluarga notasi grafis yang didukung oleh *meta-model* tunggal, yang membantu pendeskripsian dan desain system perangkat lunak, khususnya sistem yang dibangun menggunakan pemrograman berorientasi objek".

a. *Use Case* Diagram

Mendeskripsikan interaksi tipikal antara para pengguna sistem dengan sistem itu sendiri, dengan memberi sebuah narasi tentang bagaimana sistem tersebut digunakan.

b. *Activity* Diagram

Activity diagram adalah teknik untuk menggambarkan logika *prosedural*, proses bisnis, dan jalur kerja. Dalam beberapa hal, diagram ini memainkan peran mirip sebuah diagram alir, tetapi perbedaan prinsip antara diagram ini dan notasi diagram alir adalah diagram ini mendukung *behavior* paralel.

c. *Class* diagram

Class diagram menggambarkan jenis objek dalam sistem dan berbagai jenis hubungan statis yang ada di antara mereka.*Class* diagram juga menunjukkan

sifat-sifat dan operasi dari sebuah kelas dan kendala yang erlaku untuk cara objek yang terhubung.

d. *Sequence* Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar system (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence* diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah - langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu.

2. Simbol-Simbol yang digunakan *Use Case*

a. *Actor:* Menunjukan *user* yang akan menggunakan sistem baru.

b. *Use Case:* Menunjukan proses yang terjadi pada sistem baru.

c. *Association:* Menunjukan hubungan antara *actor* dengan *actor* lainnya antar *use case.*

d. *Include:* Menunjukan bahwa *use caese* adalah sumber secara *eksplisit.*

3. Simbol-Simbol yang digunakan *Activity* Diagram

a. *Activity:* Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain.

b. *Action: State* dari sistem yang mencerminkan eksekusi dari suatu aksi.

c. *Initial Node:* Bagaimana objek dibentuk atau diawali.

d. *Activity Final Node:* Bagaimana objek dibentuk dan diakhiri.

e. *Decision:* Digunakan untuk menggambarkan suatu keputusan / tindakan yang harus diambil pada kondisi tertentu.

f. *Line Connector:* Digunakan untuk menghubungkan satu simbol dengan simbol lainnya.

4. Simbol-Simbol yang digunakan *Class* diagram

a. Kelas: Kelas pada stuktur sistem.

b. Antarmuka *(Interface):* Sama dengan konsep *interface* dalam pemrograman berorientasi objek.

c. Asosiasi *(Association) :* Relasi antar kelas dengan makna umum, asosiasi biasanya juga di sertai dengan *multiplicty*.

d. Asosiasi berarah *(Directed Association):* Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi berarah biasanya juga disertai dengan *multiplicity*.

f. Generalisasi *(Generalization):* Relasi antar kelas dengan makna generalisasi-spesialisasi (Umum-khusus)

g. Kebergantungan *(Dependency):* Relasi antar kelas dengan makna kebergantungan antar kelas.

h. Agregasi *(Aggregation):* Relasi antar kelas dengan makna semua-bagian *(Whole-part)*

5. Simbol-Simbol yang digunakan *Sequence* diagram

a. *Actor:* Menggambarkan orang yang berinteraksi dengan sistem.

b. *Entity Class:* Menggambarkan hubungan kegiatan yang akan dilakukan.

c. *Boundary Class:* Menggambarkan sebuah penggambaran dari form.

d. *Control Class:* Menggambarkan penghubung antara boundary dengan tabel.

e. *Lifeline:* Menggambarkan tempat mulai dan berakhirnya sebuah pesan.

f. *Line Message:* Menggambarkan pengiriman pesan.

6. Aturan Main

Aturan mainnya sangatlah sederhana. Berikut beberapa aturan main dalam pembuatan Diagram *Use Case*, yaitu:

a. Ada judul yang merepresentasikan penggambaran Diagram *Use Case*.

b. Penggambaran diagram *use case* selalu diawali/diinisiasi oleh *actor*.

c. *Actor* selalu berada di luar sistem *(system boundary).*

d. Posisi penggambaran *use case* dibuat berurutan *(sequence)* untuk memudahkan pembacaan. Tiap urutan disebut skenario.

e. Garis penghubung *(association, include, extend)* diperbolehkan untuk saling bersinggungan satu sama lain asalkan jelas.

7. Tahapan Proses Pembuatan

a. Tentukan siapa saja yang akan menjadi actor

b. Tentukan Use-Case terkait dengan model system

c. Definisikan Asosiasi dari setiap aktor dan use-case

d. Evaluasi setiap actor dan setiap use case untuk mendapatkan kemungkinan perbaikan.

e. Evaluasi setiap use case untuk dependensi <>.

f. Evaluasi setiap actor dan setiap use case untuk generalisasi.

2.2.2. ERD *(Entity Relationship Diagram)*

1. Konsep Dasar

Menurut Sukamto dan Shalahuddin dalam (Lubis, 2016) "ERD Dikembangkan berdasarkan teori himpunan dalam bidang matematika. ERD digunakan untuk pemodelan basis data relasional. Sehingga jika penyimpanan basis data menggunakan OODBMS maka perancangan data tidak perlu menggunakan ERD".

ERD *(Entity Relationship Diagram)* merupakan suatu model untuk menjelaskan hubungan antar data dalam basis data berdasarkan objek-objek dasar data yang mempunyai hubungan antar relasi. Variasi dari suatu kardinalias akan sangat menentukan bentuk konversi tabel ERD. Peran kardinalitas sangat diperlukan untuk mempertegas perbedaan dari setiap pemodelan diagram E-R.

2. Simbol-simbol yang Digunakan

Simbol-simbol dalam ERD *(Entity Relationship Diagram)* adalah sebagai berikut:

a. Entitas: suatu yang nyata atau abstrak yang mempunyai karakteristik dimana kita akan menyimpan data.
b. Atribut: ciri umum semua atau sebagian besar instansi pada entitas tertentu.
c. Relasi: hubungan alamiah yang terjadi antara satu atau lebih entitas.
d. Link: garis penghubung atribut dengan kumpulan entitas dan kumpulan entitas dengan relasi.

3. Komponen *Entity Relationship Diagram* (ERD)

ERD terbagi atas 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 menunjukan hubungan yang terjadi antara dua entitas.

2.2.3. LRS *(Logical Record Structure)*

1. Konsep Dasar

Menurut Hasugian dan Shidiq dalam (Tasiati & Hellyana, 2013) mengemukakan bahwa: *Logical Record Structure* (LRS) merupakan sebuah model sistem yang digambarkan dengan sebuah *diagram*-ER akan mengikuti pola/aturan pemodelan tertentu dalam kaitannya dengan konversi ke LRS, maka perubahan yang terjadi adalah mengikuti aturan-aturan.

2. Simbol-simbol yang digunakan

a. *Entity*: objek yang mewakili sesuatu yang nyata dan dapat dibedakan dari sesuatu yang lain .

b. Relasi: Hubungan antara sejumlah entitas yang berasal dari himpunan entitas yang berbeda.

3. Kardinalitas Relasi

a. Satu ke satu *(One to One)*

Setiap elemen dari Entitas A berhubungan paling banyak dengan elemen pada Entitas B. Demikian juga sebaliknya setiap elemen B erhubungan paling banyak satu elemen pada Entitas A.

b. Satu ke banyak *(One to Many)*

Setiap elemen dari Entitas A berhubungan dengan maksimal banyak elemen pada Entitas B. Dan sebaliknya setiap elemen dari Entitas B berhubungan dengan paling banyak satu elemen di Entitas A.

c. Banyak ke satu *(Many to One)*

Setiap elemen dari Entitas A berhubungan paling banyak dengan satu elemen pada Entitas B. Dan sebaliknya setiap elemen dari Entitas B behubungan dengan maksimal banyak elemen di entitas A.

d. Banyak ke banyak (Many to Many)

Setiap elemen dari Entitas A berhubungan maksimal banyak elemen pada Entitas B demikian sebaliknya.