

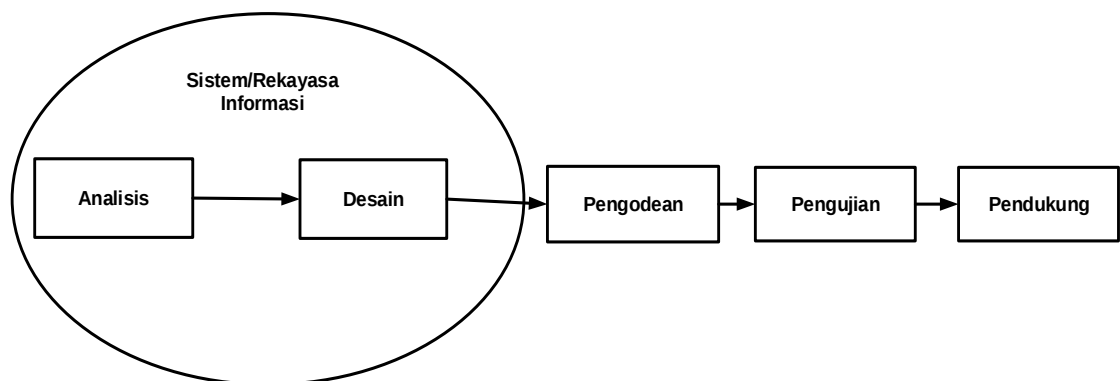
BAB II

LANDASAN TEORI

2.1. Tinjauan Pustaka

A. Konsep Dasar Sistem Informasi

Menurut Rosa A.S dan M. Shalahudin (2016:28) “model *SDLC* air terjun (*waterfall*) sering juga disebut model sequensial linier (*sequential linear*) atau alur hidup klasik (*classic life cycle*)”. Model air terjun menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau terurut dimulai dari analisis, desain, pengkodean, pengujian, dan tahap pendukung (*support*). Berikut adalah gambar model air terjun :



Sumber: Sukanto dan. Shalahudin (2016:28)

Gambar II.1. Waterfall

1. Analisis kebutuhan perangkat lunak

Proses pengumpulan kebutuhan dilakukan secara intensif untuk menspesifikasikan kebutuhan 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 antarmuka, dan prosedur pengkodean. Tahap ini mentranslasi kebutuhan perangkat lunak dari tahap analisis kebutuhan kerepresentasi 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 dalam 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 fungsional dan memastikan bahwa semua bagian mudah 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 *user*. Perubahan bisa terjadi karena adanya kesalahan yang muncul dan tidak terdeteksi saat pengujian atau saat 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.

B. Konsep Dasar Sistem Persediaan Barang

Menurut Rangkuti (2012:07) menyatakan bahwa persediaan adalah bahan-bahan, bagian yang disediakan, dan bahan-bahan dalam proses yang terdapat dalam perusahaan untuk proses produksi, serta barang-barang jadi atau produk yang disediakan untuk memenuhi permintaan dari konsumen atau pelanggan setiap waktu. Penyebab timbulnya persediaan adalah sebagai berikut.

1. Mekanisme pemenuhan atas permintaan

Permintaan terhadap suatu barang tidak dapat dipenuhi seketika bila barang tersebut tidak tersedia sebelumnya. Untuk menyiapkan barang ini diperlukan waktu untuk pembuatan dan pengiriman, maka adanya persediaan merupakan hal yang sulit dihindarkan.

2. Keinginan untuk meredam ketidakpastian

Ketidakpastian terjadi akibat: permintaan yang bervariasi dan tidak pasti dalam jumlah maupun waktu kedatangan, waktu pembuatan yang cenderung tidak konstan antara satu produk dengan produk berikutnya, waktu tenggang (lead time) yang cenderung tidak pasti karena banyak faktor yang tidak dapat dikendalikan. Ketidakpastian ini dapat diredam dengan mengadakan persediaan.

3. Keinginan melakukan spekulasi yang bertujuan mendapatkan keuntungan besar dari kenaikan harga di masa mendatang.

C. Konsep Dasar Pemrograman

Menurut Al-Bahra (2010:259) “Pemrograman terstruktur adalah cara pengorganisasian dan pengkodean program-program yang kompleks dan membuat program tersebut lebih mudah dipahami dan dimodifikasi”.

Langkah-langkah dalam pengembangan sistem adalah sebagai berikut :

1. Definisikan Masalah

Pemahaman permasalahan merupakan suatu hal yang sangat penting bagi si pemogram. Si pemogram perlu memahami permasalahan yang dihadapi dan yang akan diselesaikan oleh pemesan program, agar hasil pendefinisian masalah tidak menyimpang dari masalah yang sedang dihadapi.

2. Merancang *Outline* Pemecahan Masalah

Dalam merancang *outline* pemecahan masalah pemogram perlu membuat rincian proses secara rinci (meliputi proses apa saja yang terjadi pada menu utama, lalu masing-masing submenu dalam menu utama tersebut akan mengerjakan berapa proses, setiap proses yang terjadi berinteraksi dengan *file* apa saja, dan berapa *file* yang terlibat dalam proses tersebut).

3. Pengkodean/Menulis Program

Tujuan pengkodean adalah terjadinya efisiensi terhadap memori yang akan digunakan, efisiensi perintah dalam setiap modul program, serta efisiensi terhadap penggunaan fasilitas *input* dan *output* yang akan berpengaruh langsung terhadap prosesor ataupun pengguna.

4. Eksekusi Program

Dalam mengeksekusi program maka perlu memperhatikan beberapa factor seperti jenis bahasa pemograman yang dipilih apakah sanggup mengangkat

data yang akan diproses dan spesifikasi perangkat keras yang digunakan agar tidak terjadi kesalahan.

5. Dokumentasi Dan Pemeliharaan Program

Pemeliharaan program berfungsi untuk menjabarkan aktivitas dari hasil analisis terhadap sistem dan dilakukan setelah program diimplementasikan dan telah digunakan beberapa saat oleh pemakai. Pemeliharaan mencakup format tampilan yang disesuaikan, fungsi-fungsi yang tidak sesuai oleh pengguna dan adaptasi dengan spesifikasi prosesor yang baru. Komponen Dalam Pemrograman Terstruktur adalah sebagai berikut :

a. Pemrograman Top-Down dan *Bottom-Up*

Teknik *Top-Down* sebagai suatu masalah yang kompleks dibagi-bagi kedalam beberapa kelompok masalah yang lebih kecil. Sedangkan Teknik *Bottom-Up* dilakukan dengan menggabungkan prosedur-prosedur yang ada menjadi satu kesatuan program guna menyelesaikan masalah.

b. Pemrograman Modular

Teknik pemrograman modular yang digunakan untuk mengimplementasikan langkah-langkah pemecahan masalah pada kelompok masalah. Modul Program sebagai sekumpulan instruksi yang memiliki operasi-operasi dan data yang didefinisikan.

c. Struktur Kontrol

Teknik pemrograman yang menggunakan tiga struktur yaitu dengan struktur urut, struktur seleksi dan struktur pengulangan dalam menyelesaikan masalah. Untuk mendukung dalam penyusunan skripsi ini, penulis menggunakan *software* bahasa pemrograman.

D. Bahasa Pemograman

Menurut Yulikuspartono (2009:29), bahasa pemograman merupakan sederetan instruksi atau statement dalam bahasa yang dimengerti oleh komputer yang bersangkutan. Dan berikut contoh bahasa pemograman beserta sumbernya.

a. PHP

Menurut Peranginangin (2011:2) mengatakan bahwa “singkatan dari (*Hypertext Preprocessor*) yang digunakan sebagai bahasa *scriptserver-side* dalam pengembangan *web* yang disisipkan pada dokumen HTML”.

b. Pengenalan MySQL

Menurut Nugroho (2008:91) memberikan batasan bahwa “*Mysql* (*My Structured Query Language*) atau yang biasa dibaca ***mai-se-kuel*** adalah sebuah program pembuat dan pengelola database atau yang sering disebut dengan DBMS (*Database Management System*), sifat dari DBMS ini adalah *Open Source*”.

c. Pengenalan HTML

Menurut Junaedi (2010:19) mengatakan bahwa “suatu format data yang digunakan untuk membuat dokumen *hypertext* yang dapat dieksekusi dari satu *platform* komputer ke *platform* komputer lainnya tanpa perlu melakukan suatu perubahan apapun dengan suatu alat tertentu”.

d. Pengenalan **Web Browser**

Menurut Sutarman (2007:261) mengatakan bahwa “suatu program yang dirancang untuk mengambil informasi dari suatu *server* komputer pada jaringan *internet*”. Namun ada juga sistem yang diakses di *web browser* tanpa menggunakan *internet* namun hanya *intranet*. Dalam arti interanet, sistem tersebut hanya bisa diakses oleh lingkungan dalam saja.

Menurut Nugroho (2013:15) Peralatan pendukung merupakan alat yang digunakan untuk menggambarkan bentuk logika model dari suatu sistem dengan menggunakan simbol-simbol, lambang-lambang, diagram-diagram yang menunjukkan secara tepat arti dan fungsinya. Adapun peralatan pendukung (*tools system*) yang dijelaskan sebagai model sistem yang akan dirancang adalah sebagai berikut :

E. UML (*Unified Modelling Language*)

Menurut Fowler (2010:1) “*Unified Modelling Language* (UML) adalah keluarga notasi grafis yang didukung oleh meta-model tunggal, yang membantu pendeskripsian dan desain sistem perangkat lunak, khususnya sistem yang dibangun menggunakan pemrograman berorientasi objek (OOP)”. Menurut Fowler (2010:109-139) terdapat beberapa tahapan dalam UML seperti berikut:

a. Use case diagram

Use case adalah teknik untuk merekam persyaratan fungsional sebuah sistem. *Use case* 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*.

Menurut Munawar (2010:109) Activity Diagram adalah teknik untuk mendeskripsikan logika procedural, proses bisnis dan aliran kerja dalam banyak kasus. *Activity Diagram* mempunyai peran seperti halnya Flowchart, akan

tetapi perbedaannya dengan *Flowchart* adalah *Activity Diagram* bisa mendukung perilaku paralel sedangkan *Flowchart* tidak bisa.

c. *Component diagram*

Component Diagram tidak seperti ikon, komponen tidak menggunakan notasi yang asing dengan kita. Komponen terhubung melalui antarmuka yang digunakan dan dibutuhkan, sering menggunakan notasi bola dan soket seperti halnya *class diagram*". Anda dapat juga memecah komponen dengan menggunakan *composite structure diagram*.

d. *Deployment diagram*

Deployment diagram menunjukkan susunan fisik sebuah sistem, menunjukkan bagian perangkat lunak mana yang berjalan pada perangkat keras mana".

F. ERD (*Entity Relationship Diagram*)

Menurut Al-Bahra (2010:142) "ERD adalah suatu model jaringan yang menggunakan susunan data yang di simpan dalam sistem secara abstrak". *Entity Relation Diagram* (ERD) digunakan untuk menggambarkan hubungan antara *datastore* yang ada di dalam diagram hubungan data antara lain:

a. *Entity* (entitas)

Entity di gambarkan dengan sebuah bentuk persegi panjang. *Entity* adalah sesuatu apa saja yang di dalam sistem, nyata maupun abstrak dimana data tersimpan atau dimana terdapat data. Entitas diberi nama dengan kata benda dan dapat di kelompokkan dalam empat jenis nama, yaitu orang, benda, lokasi, kejadian (terdapat unsure waktu didalamnya).

b. *Relationship* (hubungan)

Relationship dapat digambarkan dengan sebuah bentuk belah ketupat. *Relationship* adalah hubungan alamiah yang terjadi antara *entitas*. Pada umumnya *relationship* diberi nama dengan kata kerja, sehingga memudahkan untuk melakukan pembacaan relasinya. Penggambaran hubungan yang terjadi adalah sebuah bentuk belah ketupat dihubungkan dengan dua bentuk persegi panjang.

c. *Atribut*

Atribut adalah sifat atau karakteristik dari tiap *entitas* maupun tiap *relationship*. Menjelaskan apa yang sebenarnya yang dimaksud *entitas* maupun *relationship*, sehingga atribut adalah *element* dari setiap *entitas* dan *relationship*.

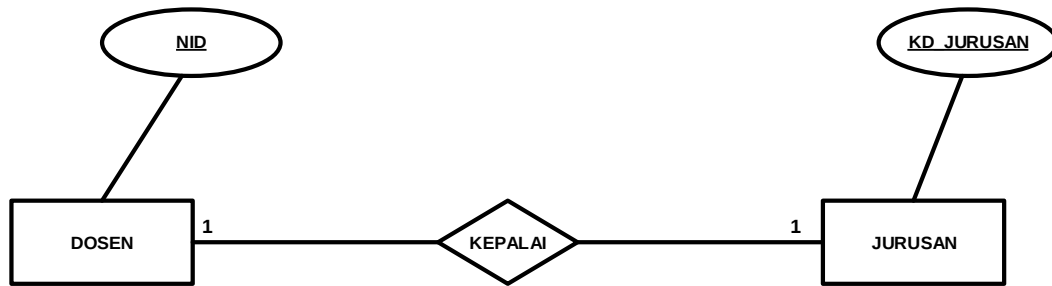
d. Alur

Alur memiliki fungsi untuk menghubungkan atribut dengan entitas dan entitas dengan relasi. Dan berbentuk garis.

Terdapat 3 macam relasi dalam ERD, yaitu:

1. *Satu ke satu (one to one 1 : 1)*

Tingkat hubungan satu ke satu, dinyatakan dengan suatu kejadian pada *entitas* pertama, hanya mempunyai satu hubungan dengan satu kejadian pada *entitas* yang kedua sebaliknya.



Menurut Al-Bahra (2010:160)

Gambar II.2 Diagram Kardinalitas One To One

2. Satu banyak (*one to many* atau $1 : M$)

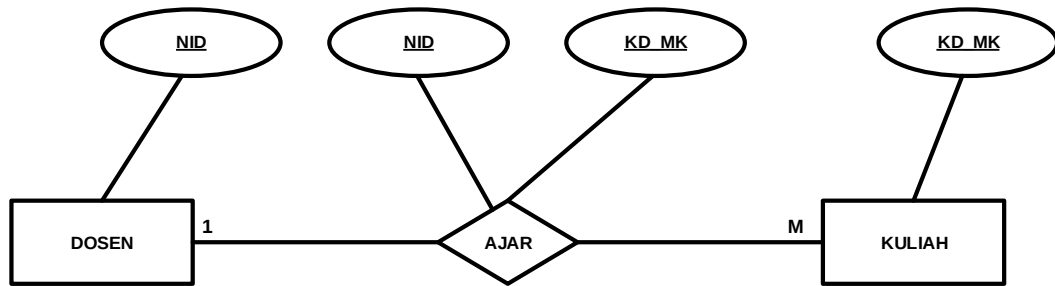
Tingkat hubungan satu banyak adalah sama dengan banyak ke satu. Untuk satu kejadian pada entitas yang pertama dapat mempunyai banyak hubungan dengan kejadian pada entitas yang kedua. Sebaliknya satu kejadian pada entitas kedua hanya dapat mempunyai satu hubungan dengan satu kejadian pada entitas yang pertama.

a. *One to Many* (satu ke banyak)

Satu *tupel* pada entitas A dapat berhubungan dengan banyak *tupelo* pada entitas B, tetapi tidak sebaliknya, dimana setiap *tupel* pada entitas B, berhubungan dengan paling banyak satu *tupel* pada entitas A.

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

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

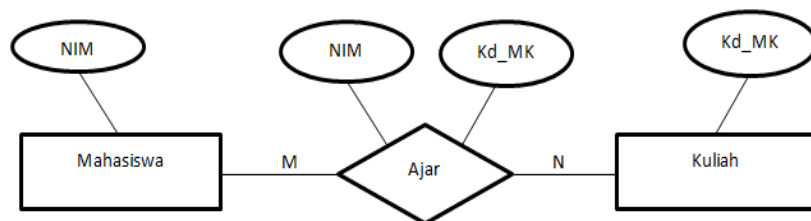


Menurut Al-Bahra (2010:165)

Gambar II.3 Diagram Kardinalitas One To Many

3. Banyak ke banyak (*Many to Many* atau *M : M*)

Tingkat hubungan banyak ke banyak terjadi jika tiap kejadian pada sebuah *entitas* akan mempunyai banyak hubungan dengan kejadian pada *entitas* lainnya. Baim dilihat dari si *entitas* yang pertama, maupun dilihat dari sisi yang kedua.

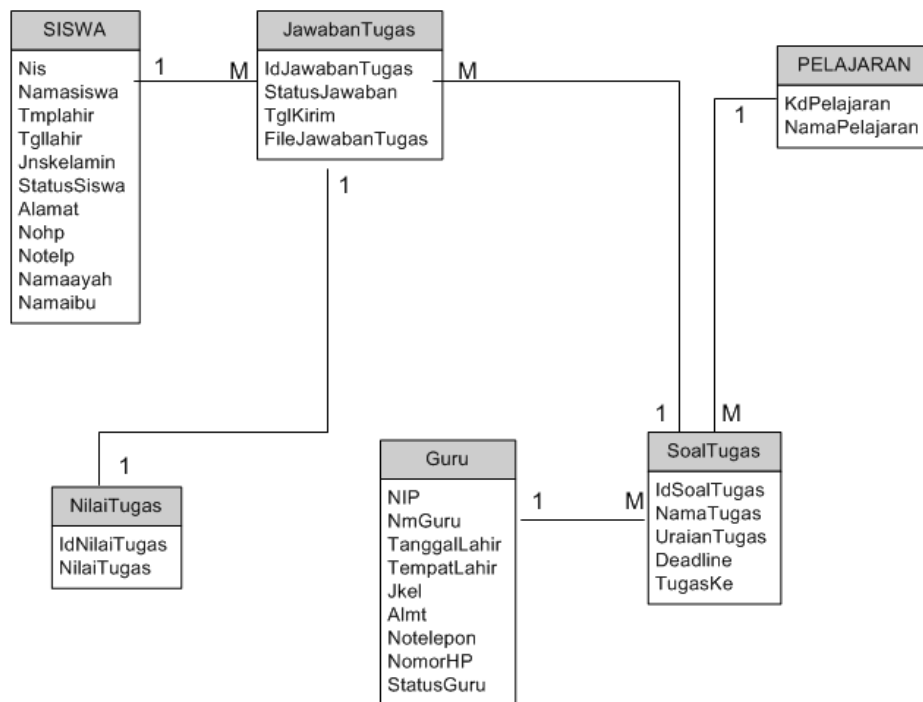


Menurut Al-Bahra (2010:169)

Gambar II.4 Diagram Kardinalitas Many To Many

G. Logical Record Structure (LRS)

Menurut Purnomo (2008:15) "pengertian LRS adalah representasi dari struktur *record-record* pada tabel-tabel yang terbentuk dari hasil antar himpunan entitas yang digunakan untuk Menentukan kardinalitas, jumlah *table* dan *Foreign Key (FK)*". Berikut adalah contoh gambar LRS :



Sumber Purnomo (2008:15)

Gambar II.5 Contoh LRS

H. Pengujian Web (*Black Box Testing*)

Menurut Pressman (2010:3) ‘‘Pengujian *Black Box* adalah pengujian aspek fundamental sistem tanpa memperhatikan struktur logika internal perangkat lunak’’. Metode ini digunakan untuk mengetahui apakah perangkat lunak berfungsi dengan benar. Pengujian *black box* merupakan metode perancangan data uji yang didasarkan pada spesifikasi perangkat lunak. Data uji dieksekusi pada perangkat lunak dan kemudian keluar dari perangkat lunak dicek apakah telah sesuai yang diharapkan. Pengujian *Black Box* berusaha menemukan kesalahan dalam kategori :

1. Fungsi-fungsi yang tidak benar atau hilang

Mengoreksi kesalahan dalam input data hilang dan kurang benar akibat kurang nya *script*.

2. Kesalahan *interface*

Kesalahan yang terjadi saat program dijalankan akan bisa di uji dengan black box agar kembali ke *interface* yang diharapkan.

3. Kesalahan dalam struktur data atau akses *database* eksternal

Menganalisi kesalahan yang akan muncul akibat pemanggilan field pada table di *database*.

4. Kesalahan kinerja.

Black box menganalisa kesalahan kinerja program yang tidak sesuai yang diharapkan.

5. *Inisialisasi* dan kesalahan terminasi

Inisialisasi dan *terminasi* memberikan hasil maksimal terhadap program sesuai dengan alur program.

2.2. Penelitian Terkait

Menurut Tahkiq dkk (2012:01) Selama ini proses arus data pengelolaan Persediaan barang antar sub sistem belum terintegrasi, pencatatan data Transaksi Persediaan Barang pada buku agenda yang mengakibatkan terjadinya penumpukan catatan sehingga ketika data persediaan barang diperlukan proses pencarian tidak bisa dilakukan dengan cepat, dan secara fisik media penyimpanan arsip pemesanan dan penerimaan memerlukan media yang luas dan besar, contohnya lemari, akibatnya terjadinya penumpukan arsip.

Menurut Cahyana dkk (2012:1) Dalam hal ini PT. Putera Agung Setia masih menggunakan sistem manual, semua transaksi persediaan barang dicatat dan kemudian dilaporkan, dengan adanya sistem semacam ini banyak permasalahan yang terjadi terutama pada sistem pelaporan yang manual

seperti ini. Untuk itu, menghindari pencatatan persediaan yang manual, dibutuhkan suatu wadah layanan sistem informasi persediaan barang yang dapat mempermudah manajemen perusahaan memperoleh informasi tentang laporan-laporan yang tersimpan dalam suatu database pengolahan persediaan barang mulai dari pemesanan barang sampai ke proses produksi