# BAB II

# LANDASAN TEORI

## 2.1 Persediaan dan Pengendalian Persediaan

Menurut (Herjanto, 2008)**,** persediaan adalah bahan baku atau barang yang disimpan yang akan digunakan untuk proses memenuhi tujuan tertentu, misalnya untuk digunakan dalam proses produksi atau perakitan, untuk dijual kembali, atau untuk suku cadang dari suatu peralatan atau mesin. Persediaan dapat berupa bahan mentah, bahan pembantu, barang dalam proses, barang jadi ataupun suku cadang. Bisa dikatakan tidak ada perusahaan yang beroperasi tanpa persediaan, meskipun sebenarnya persediaan hanyalah suatu sumber dana yang menganggur, karena sebelum persediaan digunakan berarti dana yang terikat didalamnya tidak dapat digunakan untuk keperluan yang lain.

Persediaan sebagai salah satu aset penting dalam perusahaan karena biasanya mempunyai nilai yang cukup besar serta mempunyai pengaruh terhadap besar kecilnya biaya operasi perencanaan dan pengendalian persediaan merupakan suatu kegiatan penting yang mendapat perhatian khusus dari manajemen perusahaan. Setiap bagian dari perusahaan dapat memandang persediaan dari berbagai sisi yang berbeda. Bagian pemasaran misalnya, menghendaki tingkat persediaan yang tinggi agar dapat melayani permintaan pelanggan sebaik mungkin. Bagian pembelian cenderung untuk membeli barang dalam jumlah yang besar dengan tujuan untuk memperoleh diskon sehingga harga per unit bisa lebih rendah. Demikian juga bagian produksi, menghendaki tingkat persediaan yang besar untuk mencegah terhentinya proses produksi karena

kekurangan bahan. Di pihak lain, bagian keuangan memilih untuk memiliki persediaan yang serendah mungkin agar dapat memperkecil investasi dalam persediaan dan biaya pergudangan.

Sistem pengendalian persediaan dapat didefinisikan sebagai serangkaian kebijakan pengendalian untuk menentukan tingkat persediaan yang harus dijaga, kapan pesanan untuk menambah persediaan harus dilakukan dan berapa besar pesanan harus diadakan. Sistem ini menentukan dan menjamin tersedianya persediaan yang tepat dalam kuantitas dan waktu yang tepat.

Mengendalikan persediaan yang tepat bukan hal yang mudah. Apabila jumlah persediaan terlalu besar mengakibatkan timbulnya dana menganggur yang besar (yang tertanam dalam persediaan), meningkatnya biaya penyimpanan, dan resiko kerusakan barang yang lebih besar. Namun, jika persediaan terlalu sedikit mengakibatkan resiko terjadinya kekurangan persediaan (*stockout*) karena seringkali bahan/barang tidak dapat didatangkan secara mendadak dan sebesar yang dibutuhkan yang menyebabkan terhentinya proses produksi, tertundanya penjualan, bahkan hilangnya pelanggan. Oleh karena itu pengendalian persediaan harus dilakukan sedemikian rupa agar dapat melayani kebutuhan bahan/barang dengan tepat dan dengan biaya rendah.

## 2.2 Fungsi Persediaan

Menurut (Herjanto, 2008)**,** beberapa fungsi penting persediaan dalam memenuhi kebutuhan perusahaan, sebagai berikut :

1. Menghilangkan resiko keterlambatan pengiriman bahan baku atau barang yang dibutuhkan perusahaan.

2. Menghilangkan resiko jika *material* yang dipesan tidak baik sehingga harus dikembalikan.
3. Menghilangkan resiko terhadap kenaikan harga barang atau inflasi.
4. Untuk menyimpan bahan baku yang dihasilkan secara musiman sehingga perusahaan tidak akan kesulitan jika bahan baku itu tidak tersedia di pasaran.
5. Mendapatkan keuntungan dari pembelian berdasarkan diskon kuantitas.
6. Memberikan pelayanan kepada pelanggan dengan tersedianya barang yang diperlukan.

Persediaan dikelompokan ke dalam empat jenis, yaitu :

1. *Fluctuation Stock*, merupakan persediaan yang dimaksudkan untuk menjaga terjadinya fluktuasi permintaan yang tidak diperkirakan sebelumnya, dan untuk mengatasi bila terjadi kesalahan/penyimpangan dalam perkiraan penjualan, waktu produksi, atau pengiriman barang.
2. *Anticipation Stock*, merupakan persediaan untuk menghadapi permintaan yang dapat diramalkan, misalnya pada musim permintaan tinggi, tetapi kapasitas produksi pada saat itu tidak mampu memenuhi permintaan. Persediaan ini juga dimaksudkan untuk menjaga kemungkinan sukarnya diperoleh bahan baku sehingga tidak mengakibatkan terhentinya produksi.
3. *Lot-size Inventory*, merupakan persediaan yang diadakan dalam jumlah yang lebih besar daripada kebutuhan pada saat itu. Persediaan dilakukan untuk mendapatkan keuntungan dari harga barang (berupa diskon) karena membeli dalam jumlah yang besar, atau untuk mendapatkan penghematan biaya pengangkutan per unit yang lebih rendah.

4. *Pipeline Inventory*, merupakan persediaan yang dalam proses pengiriman dari tempat asal ke tempat dimana barang itu akan digunakan. Misalnya, barang yang dikirim dari pabrik menuju tempat penjualan, yang dapat memakan waktu beberapa hari atau minggu.

## 2.3 *Make to Stock*

Menurut (Gasperz, 2002), perusahaan yang memilih strategi *make to stock* akan memiliki *inventory* yang terdiri dari produk akhir *(finished product)* untuk dapat dikirim dengan segera apabila ada permintaan dari pelanggan. Dalam strategi *make to stock*, siklus waktu dimulai ketika produsen menspesifikasikan produk, memperoleh bahan baku, dan memproduksi produk akhir untuk disimpan dalam *stock*, produsen akan mengambil produk itu dari *stock* dan mengirimkannya kepada pemesan.

Perusahaan yang menggunakan strategi *make to stock* memiliki resiko yang tinggi berkaitan dengan investasi *inventory*, karena pesanan pelanggan secara aktual tidak dapat diidentifikasi secara tepat dalam proses produksi. Permintaan aktual dari pelanggan hanya dapat diramalkan, dimana seringkali tingkat aktual dari produksi hanya berkolerasi rendah dengan pesanan pelanggan aktual yang diterima.

## 2.4 Peramalan atau *Forecasting*

Menurut (Santoso, 2009), definisi peramalan atau *forecasting* sendiri sebenarnya beragam, berikut beberapa definisinya :

a. Perkiraan munculnya sebuah kejadian di masa depan, berdasarkan data yang ada di masa lampau.
b. Proses menganalisis data historis dan data saat ini untuk menentukan *trend*.

c. Proses estimasi dalam situasi yang tidak diketahui.
d. Pernyataan yang dibuat tentang masa depan.
e. Penggunaan ilmu dan teknologi untuk memperkirakan situasi di masa depan.
f. Upaya sistematis untuk mengatisipasi kejadian atau kondisi di masa depan.

Dari beberapa definisi di atas, dapat disimpulkan bahwa peramalan/*forecasting* berkaitan dengan upaya memperkirakan apa yang terjadi di masa depan, berbasis pada metode ilmiah (ilmu dan teknologi) serta dilakukan secara sistematis.

Rentang waktu kegiatan peramalan sangat bervariasi. Ada yang melakukan peramalan secara rutin, seperti prediksi persediaan barang untuk item-item barang yang secara kontinyu dibeli dan disimpan, dalam hal ini peramalan dilakukan dalam jangka pendek dapat bulanan, mingguan bahkan harian. Namun ada pula peramalan yang mempunyai rentang waktu lama sampai bertahun–tahun. Peramalan atau *forecasting* dari sudut horison waktu dapat dibagi menjadi :

1. Jangka Pendek *(Short Term)*

   Jangka pendek meliputi kurun waktu mulai dari satu hari sampai satu musim atau dapat sampai satu tahun. Oleh karena itu waktu peramalan sangat singkat, maka data historis (terdahulu) masih relevan untuk dijadikan bahan pembuatan prediksi. Contoh kegiatan peramalan dalam jangka pendek adalah penentuan pemesanan kembali persediaan barang (*Reorder Point*) atau dalam kegiatan pengelolaan kas.

2. Jangka Menengah (*Medium Term*)

   Jangka menengah meliputi kurun waktu dari satu musim (kuartal, triwulan atau yang lain) sampai dua tahun. Kegiatan peramalan dalam jangka menengah

masih menggunakan metode kuantitatif dan kualitatif, karena data historis masa lalu dianggap masih cukup relevan untuk memprediksi masa datang. Contoh kegiatan peramalan dalam jangka waktu menengah adalah perencanaan kapasitas produksi atau perencanaan penjualan produk dalam satu atau dua tahun ke depan.

3. Jangka Panjang (*Long Term*)

   Jangka panjang meliputi peralaman untuk kurun waktu lima tahun. Kegiatan peramalan jangka panjang pada umumnya berdasarkan pada intuisi dan pengalaman seseorang, walaupun banyak pula perusahaan yang tetap menggunakan data historis dan kuantitatif untuk melakukan peramalan yang bersifat *long term.* Contoh kegiatan peramalan jangka panjang adalah kegiatan penelitian dan pengembangan (R&D), perencanaan lokasi pabrik atau perencanaan pembuatan pabrik baru.

### 2.4.1 Tahapan Peramalan atau *Forecasting*

Menurut (Santoso, 2009), agar hasil peramalan dapat secara efektif menjawab masalah yang ada, kegiatan *forecasting* sebaiknya mengikuti tahapan berikut ini:

a. Perumusan masalah dan pengumpulan data

   Menentukan masalah tentang apa yang akan diprediksi, formulasi masalah yang jelas akan menuntun pada ketapatan jenis dan banyaknya data yang akan dikumpulkan. Dapat saja masalah telah ditetapkan namun data yang relevan tidak tersedia, hal ini akan memaksa diadakannya perumusan ulang atau mengubah metode peramalan/*forecasting.*

b. Persiapan data

Setelah masalah dirumuskan dan data telah terkumpul, tahap selanjutnya adalah menyiapkan data hingga dapat diproses dengan benar.

c. Membangun model

Setelah data dianggap memadai dan siap dilakukan kegiatan prediksi, proses selanjutnya adalah memilih (model) metode yang tepat untuk melakukan *forecasting* pada data tersebut.

d. Implementasi model

Setelah metode peramalan ditetapkan, maka model dapat diterapkan pada data dan dapat dilakukan prediksi pada data untuk beberapa periode ke depan.

e. Evaluasi *forecasting*

Hasil *forecasting* yang telah ada kemudian dibandingkan dengan data aktual. Tentu saja tidak ada metode peramalan yang dapat memprediksi data di masa depan secara tepat, yang ada adalah ketepatan prediksi. Untuk itu pengukuran kesalahan *forecasting* dilakukan untuk melihat apakah metode yang telah digunakan sudah memadai untuk memprediksi sebuah data.

### 2.4.2 Jenis Data pada Kegiatan Peramalan

Menurut (Santoso, 2009), data yang akan diprediksi secara umum dapat dibagi menjadi dua tipe yakni data kualitatif dan data kuantitatif. Tidak semua data yang akan digunakan untuk kegiatan prediksi harus berupa angka. Ada data yang berupa pendapat manajer, saran dari para ahli dibidang tertentu, masukan dari konsumen atau para wiraniaga. Data seperti itu berupa kalimat atau ringkasan pernyataan yang tidak semuanya harus direpresentasikan dalam bentuk angka. Walaupun data kualitatif berguna dalam mempertimbangkan sebuah *decision*

namun karena bentuk datanya tidak memungkinkan diolah dengan metode kuantitatif. Jenis data lain adalah data kuantitatif yakni data berupa angka. Data ini dibagi menjadi dua bagian yakni :

1. Data *time series*

   Data *time series* adalah data yang ditampilkan berdasarkan waktu, seperti data bulanan, data harian, data mingguan atau jenis waktu yang lain. Contoh data penjualan bulanan sepeda motor di daerah A dari tahun 2000–2007.

2. Data *cross–sectional*

   Data *cross–sectional* adalah data yang tidak berdasarkan waktu tertentu, namun data pada satu (titik) waktu tertentu. Contoh data biaya promosi di sepuluh area pemasaran produk X selama bulan Januari 2008.

### 2.4.3 Model Data pada Kegiatan Peramalan

Menurut (Santoso, 2009), model data pada kegiatan peramalan meliputi:

1. Data stasioner

   Adalah data dimana rata–rata nilainya tidak berubah dari waktu ke waktu atau dapat dikatakan data bersifat stabil. Jika data stasioner dengan ciri rata–rata nilai data adalah sama maka peramalan nilai mendatang adalah melihat rata-ratanya. Berikut ini beberapa metode yang dapat dilakukan untuk prediksi dengan data berpola stasioner:

   a. *Naive Methods*.

   *b. Simple Averaging Methods.*

   *c. Moving Averages.*

   *d. Autoregressive Moving Average* (*ARIMA*).

2. Data tidak stasioner

Adalah data yang didapati adanya *trend* atau pola *seasonal* (pengaruh musim). Berikut ini metode yang digunakan untuk data yang tidak stasioner:

a. Data dengan adanya pola *trend*.

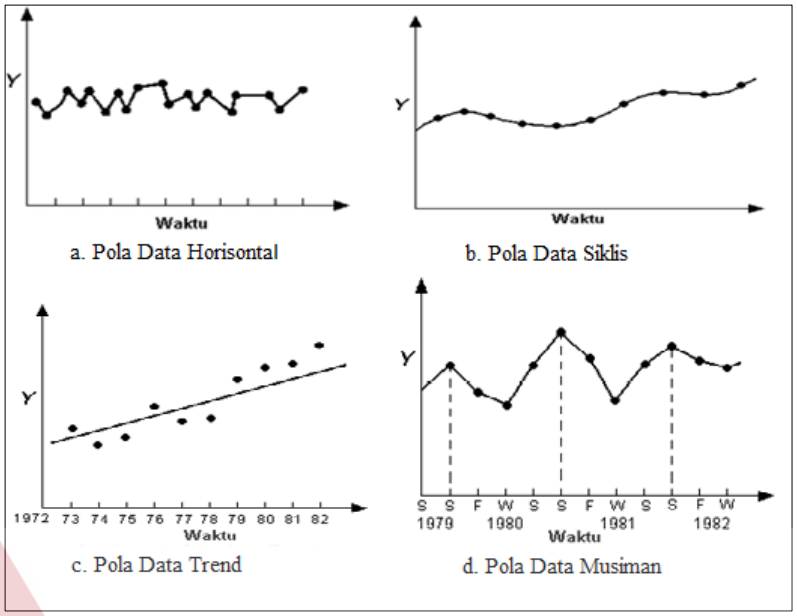
*Trend* ditandai dengan adanya kecenderungan arah data bergerak menaik (*growth*), atau menurun (*decline*) pada jangka panjang. Metode *forecasting* pada situasi ini adalah metode *Double Exponential Smoothing* (*BROWN*), *Exponential Smoothing Holt*, regresi sederhana, *ARIMA* (metode B*ox-Jenkins*).

b. Data dengan adanya pengaruh *seasonal*.

*Seasonal* ditandai dengan adanya pola perubahan yang berulang secara otomatis dari tahun ke tahun. Contoh *seasonal* permintaan buku saat siswa mulai masuk sekolah, permintaan payung saat musim hujan, produksi buah–buahan menurut musimnya dan lainnya. Metode *forecasting* dengan adanya pola *seasonal* adalah dekomposisi data, *Exponential Smoothing Winters* dan *ARIMA*.

c. Data dengan adanya pengaruh siklis

Siklis adalah fluktuasi bergelombang data yang terjadi di sekitar garis *trend*. Pola siklis agak sulit diprediksi karena pola yang ada cenderung tidak stabil. Metode *forecasting* yang digunakan adalah dekomposisi data, model–model ekonometrik, regresi berganda dan *ARIMA*.




Gambar 2.1 Pola Data
(*Sumber: Metode dan Aplikasi peramalan, Makridakis, S)*

## 2.5 Metode *Single Exponential Smoothing* (*SES*)

Menurut (Arsyad, 2001), *Exponential Smoothing* adalah suatu prosedur yang mengulang perhitungan secara terus–menerus dengan menggunakan data terbaru. Teknik ini banyak dipergunakan bila peramalan bulanan atau mingguan diperlukan untuk barang-barang dalam jumlah besar. Secara sederhana pemulusan eksponensial adalah nilai ramalan lama ( $\bar{Y}t$ ) ditambah α (*alpha*) dikalikan dengan tingkat kesalahan ($Yt-\bar{Y}t$) dari ramalan yang lama α berfungsi sebagai faktor penimbang. Secara matematis, persamaan pemulusan eksponensial dapat dilihat pada persamaan (2.1).

$$\bar{Y}t+1 = \alpha Yt + (1 - \alpha)\bar{Y}t \quad \text{.....(2.1)}$$

Dimana :

$\bar{Y}t+1$ = nilai ramalan untuk periode berikutnya

α = konstanta pemulusan ($0 < \alpha < 1$)

Yt = data baru atau nilai Y yang sebenarnya pada periode t

$\bar{Y}t$ = nilai pemulusan yang lama atau rata–rata yang dimuluskan hingga periode t-1

## 2.6 Metode *Double Exponential Smoothing* (*HOLT*)

Menurut (Arsyad, 2001), *Double exponential smoothing* (*HOLT*) memperhalus *trend* dan slopenya secara langsung dengan menggunakan konstanta–konstanta pemulusan yang berbeda. Tiga persamaan yang digunakan dalam teknik ini adalah sebagai berikut:

1. Rangkaian pemulusan secara eksponensial

$$A_t = \alpha\, Y_t + (1 - \alpha)(A_{t-1} + T_{t-1}) \quad \text{.....(2.2)}$$

2. Estimasi *trend*

$$T_t = \beta\,(A_t - A_{t-1}) + (1 - \beta)\,T_{t-1} \quad \text{.....(2.3)}$$

3. Ramalan pada periode $p$

$$\hat{Y}_{t+P} = A_t + pT_1 \quad \text{.....(2.4)}$$

Dimana :

$A_t$ = nilai baru yang telah dimuluskan

α = konstanta pemulusan untuk data ($0 \leq \alpha \leq 1$)

$Y_t$ = data yang baru atau yang sebenarnya pada periode *t*

$\beta$ = konstanta pemulusan untuk estimasi *trend* ($0 \leq \beta \leq 1$)

$T_t$ = estimasi *trend*

$p$ = periode yang diramalkan

$\hat{Y}_{t+P}$ = nilai ramalan pada periode $p$

## 2.7 Metode *Triple Exponential Smoothing* (*WINTER*)

Menurut (Makridakis, Wheelwright, & McGee, 1999), Metode *WINTER* didasarkan atas tiga persamaan persamaan yaitu satu untuk unsur stasioner, satu untuk *trend*, dan satu untuk musiman. Hal ini serupa dengan metode *HOLT*, dengan satu persamaan tambahan untuk mengatasi musiman.

Menurut (Arsyad, 2001), model linear tiga parameter dan pemulusan eksponensial musiman *WINTER* mungkin dapat mengurangi kesalahan peramalan. Suatu persamaan tambahan digunakan untuk mengestimasi adanya pengaruh faktor musim. Estimasi tersebut dinyatakan dalam suatu indeks musiman dan dihitung dengan persamaan 2.7. Alasan mengapa $Y_t$ dibagi dengan $A_t$ adalah untuk menyatakan nilainya sebagai suatu indeks, supaya dapat dihitung rata–ratanya dengan indeks musiman yang dihaluskan sampai periode *t-1*. Keempat persamaan yang digunakan dalam Metode WINTER adalah sebagai berikut :

1. Pemulusan eksponensial

$$A_t = \alpha \frac{Y_t}{S_{t-L}} + (1- \alpha)( A_{t-1}+T_{t-1}) \quad \text{.....(2.5)}$$

2. Estimasi *trend*

$$T_t = \beta (A_t+A_{t-1}) + (1- \beta) T_{t-1} \quad \text{.....(2.6)}$$

3. Estimasi musiman

$$S_t = \mu \frac{Y_t}{A_t} + (1 - \mu) S_{t-L} \quad \text{.....(2.7)}$$

4. Ramalan pada periode $p$ di masa datang

$$\hat{Y}_{t+P} = (A_t + p\, T_t)\, S_{t-L+P} \quad \text{.....(2.8)}$$

Dimana :

$A_t$ = nilai pemulusan yang baru

$\alpha$ = konstanta pemulusan untuk data ($0 \leq \alpha \leq 1$)

$Y_t$ = data yang baru atau yang sebenarnya pada periode *t*

$\beta$ = konstanta pemulusan untuk estimasi *trend* ($0 \leq \beta \leq 1$)

$T_t$ = estimasi *trend*

μ = konstanta pemulusan untuk estimasi musiman($0 \leq \mu \leq 1$)

$S_t$ = estimasi musiman

$p$ = periode yang diramalkan

$L$ = panjangnya musim

$\hat{Y}_{t+P}$ = ramalan pada periode *p*

## 2.8 Pengukuran Kesalahan Peramalan

Menurut (Arsyad, 2001), Berikut ini cara-cara untuk mengevaluasi teknik peramalan :

a. *Mean Absolute Deviation* (*MAD*)

Simpangan absolut rata–rata atau *mean absolute deviation* (*MAD*) adalah mengukur akurasi peramalan dengan merata–ratakan kesalahan peramalan (nilai absolutnya). Berikut rumus dari *mean absolute deviation :*

$$\text{MAD} = \frac{\sum_{t=1}^{n}(Y_t - \hat{Y}_t)}{n} \quad \text{.....(2.9)}$$

Dimana :

n : periode

$Y_t$ : nilai sebenarnya pada periode *t*

$\hat{Y}_t$ : nilai peramalan pada periode *t*

*b.* *Mean Squared Error* (*MSE*)

*Mean squared error* mengukur setiap kesalahan atau residual dikuadratkan kemudian dijumlahkan dan dibagi dengan jumlah observasi. Pendekatan ini menghukum suatu kesalahan peramalan yang besar dikuadratkan. Berikut rumus dari *mean squared error* :

$$MSE = \frac{\sum_{t=1}^{n}(Y_t - \hat{Y}_t)^2}{n} \quad \text{.....(2.10)}$$

Dimana :

n : periode

$Y_t$ : nilai sebenarnya pada periode *t*

$\hat{Y}_t$ : nilai peramalan pada periode *t*

*c.* *Mean Absolute Percentage (MAPE)*

*Mean absolute percentage error* (*MAPE*) dihitung dengan menemukan kesalahan absolut setiap periode kemudian membaginya dengan nilai observasi pada periode tersebut dan akhirnya merata–ratakan persentase absolut ini. Berikut rumus dari *Mean absolute percentage error* :

$$MAPE = \frac{\sum_{t=1}^{n} \frac{|Y_t - \hat{Y}_t|}{Y_t}}{n} \quad \text{.....(2.11)}$$

Dimana :

n : periode

$Y_t$ : nilai sebenarnya pada periode *t*

$\hat{Y}_t$ : nilai peramalan pada periode *t*

d. *Mean Percentage Error* (*MPE*)

*MPE* dihitung dengan cara menemukan kesalahan setiap periode, kemudian membaginya dengan nilai sebenarnya pada periode tersebut dan kemudian merata–ratakan persentase kesalahan tersebut. Berikut rumus dari *mean percentage error* :

$$\text{MPE} = \frac{\sum_{t=1}^{n} \frac{(Y_t - \hat{Y}_t)}{Y_t}}{n} \quad \text{.....(2.12)}$$

Dimana :

n : periode

$Y_t$ : nilai sebenarnya pada periode *t*

$\hat{Y}_t$ : nilai peramalan pada periode *t*

## 2.9 *Reorder Point* (*ROP*)

Menurut (Rangkuti, 2007), *Reorder point* (*ROP*) terjadi apabila jumlah persediaan yang terdapat dalam stok berkurang terus. Dengan demikian kita harus menentukan berapa banyak batas minimal tingkat persediaan yang harus dipertimbangkan sehingga tidak terjadi kekurangan persediaan. Jumlah yang diharapkan tersebut dihitung selama masa tenggang. Mungkin dapat juga ditambahkan dengan *safety stock* yang biasanya mengacu pada probabilitas atau kemungkinan terjadinya kekurangan stok selama masa tenggang. *ROP* atau biasa disebut dengan batas/titik jumlah pemesanan kembali termasuk permintaan yang diinginkan atau dibutuhkan selama masa tenggang misalnya suatu tambahan/ekstra stok. Berikut merupakan rumus dari *Reorder Point* (*ROP*) :

ROP = d X L+ SS.....(2.13)

Dimana :

ROP = titik pemesanan ulang

d = tingkat kebutuhan per unit waktu

L = waktu tenggang

SS = *safety stock*

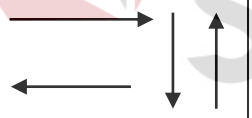
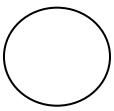
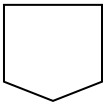
## 2.10 Desain

Menurut (Jogiyanto, 2005), sebagai dasar identifikasi dapat digunakan dokumen sistem bagian alir formulir (*paperwork flowchart* atau *form flowchart*) bila dokumentasi ini dimiliki oleh perusahaan. Berikut merupakan simbol-simbol dalam sistem maupun *data flow diagram*.

### 2.10.1 *Flowchart*

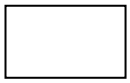
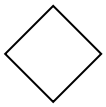
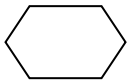
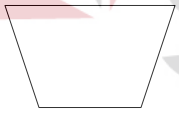
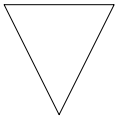
*a. Flow Direction Symbols*

Tabel 2.1 *Flow Direction Symbols*

| | | |
|---|---|---|
|  | *Flow* | Penghubung antara prosedur atau proses. |
|  | *Connector* | Simbol keluar dan masuk prosedur atau proses dalam halaman yang sama. |
|  | *Off-Line Connector* | Simbol keluar dan masuk prosedur atau proses dalam halaman yang lain. |

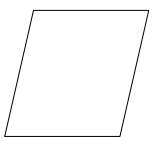
*b. Processing Symbols*

Tabel 2.2 *Processing Symbol*

| | | |
|---|---|---|
|  | *Process* | Penghubung antara prosedur atau proses. |
|  | *Decision* | Untuk menyatakan suatu tindakan (proses) yang tidak dilakukan komputer. |
|  | *Preparation* | Untuk menyatakan penyediaan tempat penyimpanan suatu pengolahan untuk memberi harga awal. |
|  | *Terminal* | Untuk menyatakan permulaan atau akhir suatu program. |
|  | *Manual-input* | Memasukkan data secara manual dengan menggunakan *online keyboard*. |
|  | *manual* | Untuk menyatakan suatu tindakan yang tidak dilakukan oleh komputer. |
|  | *offline-storage* | Untuk menunjukkan bahwa data dalam simbol ini akan disimpan ke suatu media tertentu. |

*c. Input / Output Symbols*

Tabel 2.3 *Input / Output Symbol*

| | | |
|---|---|---|
|  | *Input-output* | Simbol yang menyatakan proses *input* dan *output* tanpa tergantung dengan jenis peralatannya |

INSTITUT BISNIS
& INFORMATIKA
stikom
SURABAYA

### 2.10.3 *Entity Relationship Diagram*

*Entity relationalship diagram* dibagi menjadi dua model yaitu :

a. *Conceptual data model* (*CDM*)

Merupakan jenis model data yang menggambarkan hubungan antar tabel secara konseptual.

b. *Physical data model* (*PDM*)

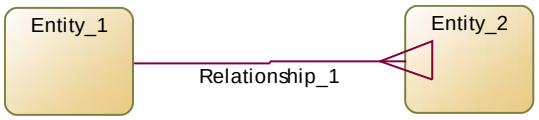
Merupakan jenis model data yang menggambarkan hubungan antar tabel secara fisikal.

Ada empat jenis relasi dalam *ERD* antara lain :

a. Relasi *one-to-one* (1:1) menyatakan bahwa setiap entitas pada tiap entitas A paling banyak berpasangan dengan satu entitas pada tipe entitas B.

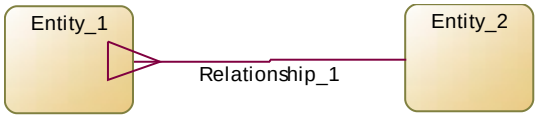

Gambar 2.2 Hubungan *One-to-one*

b. Relasi *one-to-many* (1:M) menyatakan bahwa setiap entitas pada tipe entitas A bisa berpasangan dengan banyak entitas pasa tipe entitas B, sedangkan setiap entitas pada B hanya berpasangan dengan satu entitas pada entitas B.



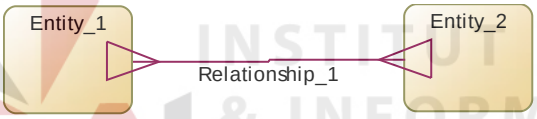

Gambar 2.3 Hubungan *One-to-many*

c. Relasi *many-to-one* (M:1) menyatakan bahwa setiap entitas pada tipe entitas A paling banyak berpasangan dengan satu entitas pada tipe entitas B dan setiap entitas B bisa berpasangan dengan banyak entitas pada tipe entitas A.




Gambar 2.4 Hubungan *Many-to-one*

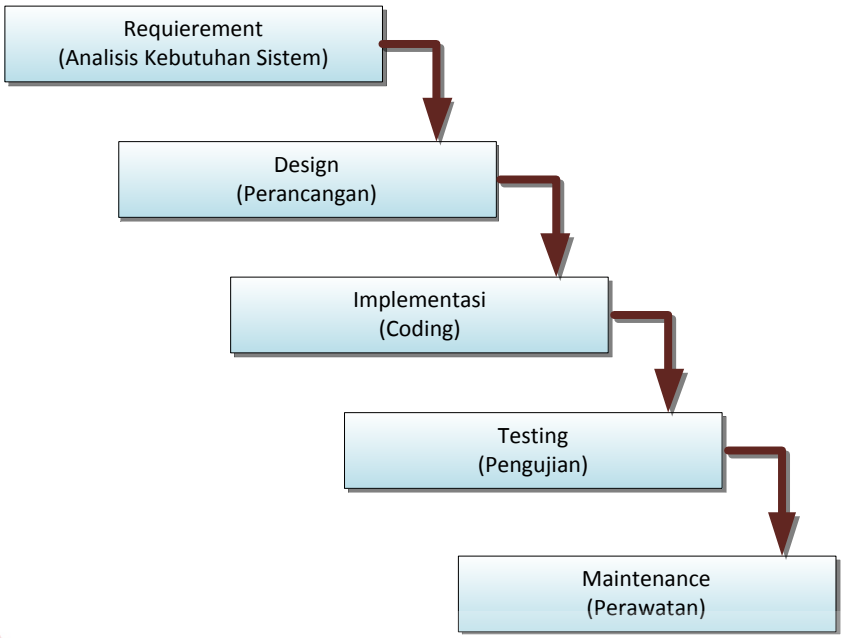
d. Relasi many-to-many (M:M) menyatakan bahwa setiap entitas pasa suatu tipe entitas A bisa berpasangan dengan banyak entitas pada entitas B dan sebaliknya.




Gambar 2.5 Hubungan *Many-to-many*

## 2.11 *System Development Life Cycle*

Menurut (Presman, 2007), model *system development life cycle* (*SDLC*) ini biasa disebut juga dengan model *waterfall* atau disebut juga *classic life cycle*. Tahapan–tahapannya adalah *requirement* (analisis sistem), *analysis* (analisis kebutuhan sistem), *design* (perancangan), *coding* (implementasi), *testing* (pengujian) dan *maintenance* (perawatan). Model ini memungkinkan proses pengembangan lebih terlihat pada gambar 2.6.




Gambar 2.6 *System Development Life Cycle* (*SDLC*) Model *Waterfall*

Dibawah ini merupakan penjelasan dari *system development life cycle* (*SDLC*) model *waterfall*.

a. *Requirement*

Pada tahap ini dilakukan analisa untuk mengetahui kebutuhan yang akan dibutuhkan. Kebutuhan itu sendiri dibedakan menjadi tiga jenis yaitu kebutuhan teknologi, kebutuhan informasi dan kebutuhan *user*. Kebutuhan teknologi dilakukan untuk menganalisa kebutuhan teknologi yang diperlukan dalam pengembangan suatu sistem. Kebutuhan informasi misalnya seperti informasi mengenai visi dan misi perusahaan, sejarah perusahaan, latar belakang perusahaan. Kebutuhan *user* untuk menganalisa terkait kebutuhan *user* dan kategori *user*. Analisa biaya dan resiko dalam tahap ini diperhitungkan biaya yang akan dikeluarkan seperti biaya implementasi, *testing* dan *maintenance*.

b. *Design*

Dari hasil analisa kebutuhan sistem tersebut akan dibuat sebuah *design database*, *DFD, ERD, user interface* dan jaringan yang dibutuhkan untuk sistem. Selain itu juga dilakukan perancangan struktur data, arsitektur perangkat lunak, detil prosedur dan karakteristik tampilan yang akan disajikan.

c. *Implementation*

Tahap ini merupakan tahap yang mengkonversi hasil perancangan sebelumnya ke dalam sebuah bahasa pemograman yang dimengerti oleh komputer. Kemudian komputer akan menjalankan fungsi–fungsi yang telah didefinisikan sehingga mampu memberikan layanan kepada pengguna.

d. *Testing*

Tahap ini dilakukan untuk mengetahui kesesuaian sistem yang berjalan sesuai prosedur atau tidak dan memastikan sistem terhindar dari *error* yang terjadi. *Testing* juga digunakan untuk memastikan kesesuaian dalam proses *input*, sehingga dapat menghasilkan *output* yang sesuai. Terdapat 2 metode pengujian yang dapat digunakan yaitu metode *black box* dan *white box*. Pada pengujian menggunakan *black box* ini dengan menekankan pada fungsionalitas dari sebuah perangkat lunak tanpa harus mengetahui bagaimana struktur di dalam perangkat lunak tersebut. Sedangkan pengujian menggunakan *white box* yaitu menguji struktur internal perangkat lunak dengan melakukan pengujian pada algoritma yang digunakan oleh perangkat lunak.

e. *Maintenance*

Tahap ini merupakan tahap terakhir dalam model *waterfall*. Aplikasi yang sudah dibuat akan dilakukan pemeliharaan atau perawatan. Pemeliharaan atau perawatan tersebut termasuk dalam memperbaiki kesalahan–kesalahan yang terjadi yang tidak ditemukan pada langkah sebelumnya. Tahap terakhir ini tidak dilakukan karena merupakan batasan dari tugas akhir.

## 2.12 Pengujian *Black Box*

Menurut (Pressman, 2007), pengujian adalah proses eksekusi suatu program dengan maksud menemukan kesalahan. Teknik pengujian *black box* adalah yang paling lazim selama integrasi. Pengujian *black box* digunakan untuk memperlihatkan bahwa fungsi–fungsi perangkat lunak adalah operasional bahwa *input* diterima dengan baik dan *output* dapat dihasilkan dengan tepat. Menurut (Romeo, 2003), metode uji coba *black box* memfokuskan pada keperluan fungsional dari *software*. Karena itu uji coba *black box* memungkinkan pengembang *software* untuk membuat himpunan kondisi *input* yang akan melatih seluruh syarat-syarat fungsional suatu program.

Uji coba *black box* berusaha untuk menemukan kesalahan dalam beberapa kategori diantaranya:

1. Fungsi-fungsi yang salah atau hilang
2. Kesalahan *interface*
3. Kesalahan dalam struktur data atau akses *database* eksternal
4. Kesalahan performa
5. Kesalahan inisialisasi dan terminasi

Tidak seperti metode *white box* yang dilaksanakan diawal proses, uji coba *black box* diaplikasikan dibeberapa tahapan berikutnya. Karena uji coba *black box* dengan sengaja mengabaikan struktur kontrol, sehingga perhatiannya difokuskan pada informasi domain. Uji coba desain untuk dapat menjawab pertanyaan berikut:

1. Bagaimana validitas fungsionalnya diuji ?
2. Jenis *input* seperti apa yang akan menghasilkan kasus uji yang baik ?
3. Apakah sistem secara khusus sensitif terhadap nilai *input* tertentu ?
4. Bagaimana batasan–batasan kelas data diisolasi ?
5. Berapa rasio data dan jumlah data yang dapat ditoleransi oleh sistem ?
6. Apa akibat yang akan timbul dari kombinasi spesifikasi data pada operasi sistem ?

Dengan mengaplikasikan uji coba *black box*, diharapkan dapat menghasilkan sekumpulan kasus uji yang memenuhi kriteria berikut:

1. Kasus uji yang berkurang jika jumlahnya lebih dari satu maka jumlah dari uji kasus tambahan harus didesain untuk mencapai uji coba yang cukup beralasan.
2. Kasus uji yang memberitahukan sesuatu tentang keberadaan atau tidaknya suatu jenis kesalahan, daripada kesalahan yang terhubung hanya dengan suatu uji coba spesifik.

Walaupun didesain untuk menemukan kesalahan, uji coba *black box* digunakan untuk mendemonstrasikan fungsi *software* yang dioperasikan, apakah *input* diterima dengan benar, dan *output* yang dihasilkan benar, apakah integritas informasi eksternal terpelihara. *Black box* testing dilakukan tanpa pengetahuan

detil struktur internal dari suatu sistem atau komponen yang dites. *Black box* testing juga disebut sebagai *behavioral* testing, *specification-based* testing, *input* atau *output* atau *functional* testing.

*Black box* testing berfokus pada kebutuhan fungsional pada *software*, berdasarkan pada spesifikasi kebutuhan dari *software*. Dengan adanya *black box* testing, perekayasa *software* dapat menggunakan sekumpulan kondisi masukan yang dapat secara penuh memeriksa keseluruhan kebutuhan fungsional pada suatu program.

*Black box* testing bukan teknik alternatif daripada *white box* testing. Lebih daripada itu, ia merupakan pendekatan pelengkap dalam mencakup *error* dengan kelas yang berbeda dari metode *white box* testing.

Kategori *error* yang akan diketahui melalui *black box* testing adalah:

1. Fungsi yang hilang atau tidak benar
2. *Error* dari antar-muka
3. *Error* dari struktur data atau akses eksternal *database*
4. *Error* kinerja atau tingkah laku
5. *Error* dari inisialisasi dan terminasi