BAB II
LANDASAN TEORI

Pada bab ini membahas mengenai teori-teori yang melandasi penyusunan tugas akhir ini. Teori-teori tersebut terdiri dari penjelasan *Forecasting*, penjelasan singkat *Exponential Smoothing* Tunggal, Regresi Sederhana, dan Perencanaan Produksi.

2.1 *Forecasting*

Berdasarkan pendapat Stenvenson dalam bukunya yang berjudul "Operations Management 11th Edition" yang menerangkan bahwa *forecasting* adalah latar belakang yang mendasari proses pengambilan keputusan yang bertujuan untuk memprediksi permintaan di masa yang akan datang. Pentingnya kedudukan *forecasting* dalam manajemen operasi adalah mempengaruhi tingkat akurasi suatu permintaan dengan kapasitas yang dapat dipenuhi. Hal tersebut dicerminkan pada saat mengambil keputusan yang dipengaruhi oleh dana yang dianggarkan, material / bahan baku produksi yang dibutuhkan dan pencapaian produksi yang merupakan perpaduan dari kedua hal sebelumnya.

Terdapat 2 (dua) aspek utama yang dapat mempengaruhi laju *forecasting* dalam suatu kegiatan produksi, yaitu tingkat permintaan dan akurasi. Tingkat permintaan memiliki fungsi beberapa variasi struktural, diantaranya tren atau variasi musiman. Sedangkan akurasi berfungsi untuk mengukur validitas suatu *forecasting* atau perencanaan dengan hasil dari forcasting tersebut yang dipengaruhi berbagai faktor pendukung maupun penghambat.

Forecasting dibuat dalam jangka waktu tertentu, baik itu jangka pendek maupun jangka panjang. Biasanya *forecasting* yang dibuat untuk jangka waktu yang pendek berhubungan dengan operasi yang sedang berlangsung, sedangkan *forecasting* yang dibuat untuk jangka waktu panjang digunakan sebagai alat perencanaan yang penting. *Forecasting* dengan jangka waktu panjang cenderung membutuhkan waktu yang cukup lama untuk mengembangkannya, menyusunnya hingga melaksanaknnya, terlebih ketika berkesinambungan dengan produk dan/atau layanan baru, peralatan baru, fasilitas baru serta lain hal.

Forecasting adalah dasar untuk merencanakan penganggaran, perencanaan kapasitas, penjualan, produksi dan persediaan, personil, pembelian, serta banyak hal lainnya lagi. *Forecasting* memainkan peran penting dalam proses perencanaan karena dapat memungkinkan manajer untuk mengantisipasi apa yang dapat terjadi (perkiraan / dugaan) masa depan sehingga perencanaan yang sesuai dapat disusun berdasarkan perkiraan / dugaan tersebut .

Fungsi atau kegunaan dari *Forecasting* diantaranya mempermudah seorang manajer dalam merencanakan dan membuat keputusan pada suatu kegiatan bisnis serta bagaimana kegiatan bisnis yang telah direncanakan tersebut dapat dilaksanakan. Oleh karena 2 (dua) fungsi tersebut, umumnya *forecasting* dibuat untuk jangka waktu panjang, terutama dalam suatu kegiatan bisnis yang menangah ke atas. Keunggulan lainnya dari *forecasting* adalah memprediksi keuntungan, pendapatan, biaya, produktivitas perubahan, harga dan ketersediaan energi dan bahan baku, suku bunga, gerakan kunci indikator ekonomi (misalnya, bruto produk domestik, inflasi, pemerintah meminjam), dan harga saham dan obligasi. Terdapat beberapa metode *forecasting* yang dapat digunakan sesuai dengan kebutuhan diantaranya :

2.1.1 *Exponential Smoothing* Tunggal

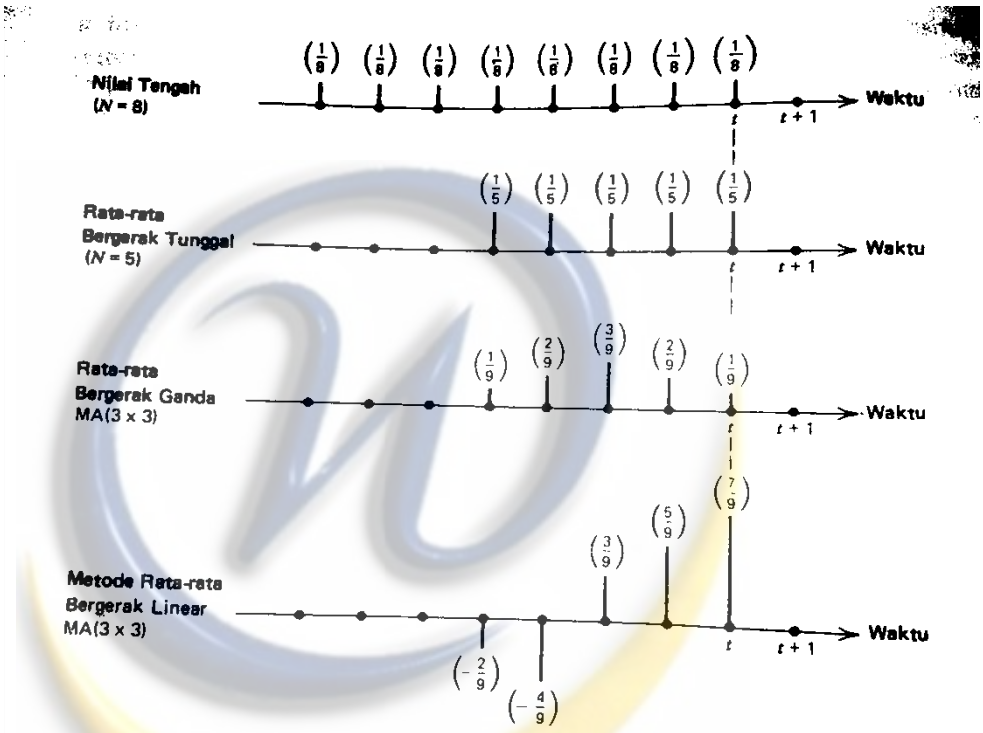
Metode *Exponential Smoothing* terdiri atas tunggal, ganda, dan metode yang lebih rumit. Semuanya mempunyai sifat lebih besar dibanding nilai observasi yang lebih lama. Dalam metode *Averaging*, bobot yang dikenakan pada nilai-nilai observasi merupakan hasil sampingan dari sistem MA tertentu yang diambil. Tetapi dalam Metode *Exponential Smoothing* terdapat satu atau lebih parameter pemulusan yang ditentukan secara eksplisit, dan hasil pilihan ini menentukan bobot yang dikenakan pada nilai observasi.

Kasus yang paling sederhana dari *Exponential Smoothing* tunggal dapat dikembangkan dari persamaan berikut:

$$F_{t+1} = F_t + \left(\frac{X_t}{N} - \frac{X_{t-N}}{N}\right) \quad \text{..... (1)}$$

Misalkan observasi yang lama X_{t-N} tidak tersedia sehingga tempatnya harus digantikan dengan suatu nilai pendekatan periode yang sebelumnya F_t. Dengan melakukan substitusi ini persamaan tersebut menjadi persamaan sebagai berikut:

$$F_{t+1} = F_t + \left(\frac{X_t}{N} - \frac{X_t}{N}\right) \quad \text{..... (2)}$$



Sumber: buku terjemahan "Forecastting 2nd Edition" halaman 80

Gambar 2.1 Contoh Pembobotan yang diberikan kepada data masa lalu

$$F_{t+1} = \left(\frac{1}{N}\right) X_t + \left(1 - \frac{1}{N}\right) F_t \quad \text{..... (3)}$$

Perhatikan bahwa jika datanya stationer, maka substitusi persamaan (3) merupakan pendekatan yang cukup baik, namun bila terdapat trend metode SES yang dijelaskan di sini tidak cukup baik.

Dari persamaan (3) dapat dilihat bahwa ramalan ini ($F_t + 1$) didasarkan atas pembobotan observasi yang terakhir yang suatu nilai bobot (1/N) dan pembobotan ramalan yang terakhir sebelumnya (F_t) dengan suatu bobot [1 –

(1/*N*)]. Karena *N* merupakan suatu bilangan positif, 1/*N* akan menjadi suatu konstanta antara nol (jika *N* tak terhingga) dan 1 (jika *N* = 1). Dengan mengganti 1/*N* dengan α, persamaan diatas menjadi:

$$F_{t+1} = \alpha X_t + (1-\alpha)F_t \quad \text{..... (4)}$$

Persamaan (4) merupakan bentuk umum yang digunakan dalam menghitung ramalan dengan metode *Exponential Smoothing*. Metode ini banyak mengurangi masalah penyimpangan data, karena tidak perlu lagi menyimpan semua data historis atau sebagian dari padanya. Agaknya hanya observasi terakhir, ramalan terakhir, dan suatu nilai α yang harus disimpan.

Implikasi *Exponential Smoothing* dapat dilihat dengan lebih baik bila persamaan (4) diperluas dengan mengganti *F* dengan komponennya sebagai berikut:

$$\begin{aligned} F_{t+1} &= \alpha X_t + (1-\alpha)[\alpha X_{t-1} + (1-\alpha)F_{t-1}] \\ &= \alpha X_t + \alpha(1-a)X_{t-1} + (1-\alpha)^2 F_{t-1} \quad \text{..... (5)} \end{aligned}$$

Jika proses substitusi ini diulangi dengan mengganti F_{t-1} dengan komponennya, F_{t-2} dengan komponennya, dan seterusnya, hasilnya adalah persamaan berikut:

$$\begin{aligned} F_{t+1} = \alpha X_t + \alpha(1-\alpha)X_{t-1} + \alpha(1-\alpha)^2 X_{t-2} + \alpha(1-\alpha)^3 X_{t-3} + \\ \alpha(1-\alpha)^4 X_{t-4} + \alpha(1-\alpha)^5 X_{t-5} + \cdots + \alpha(1-\alpha)^{N-1} X_{t-(N-1)} + \\ (1-\alpha)^N F_{t-(N-1)} \quad \text{..... (6)} \end{aligned}$$

Misalkan α = 0,2; 0,4; 0,6; atau 0,8. Maka bobot yang diberikan pada nilai obsevasi masa lalu akan menjadi sebagai berikut:

Tabel 2.1 Bobot Nilai Observasi

Bobot yang diberikan pada:	α = 0,2	α = 0,4	α = 0,6	α = 0,8
X_t	0,2	0,4	0,6	0,8
X_{t-1}	0,16	0,24	0,24	0,16
X_{t-2}	0,128	0,144	0,096	0,032
X_{t-3}	0,1074	0,0864	0,0384	0,0064
X_{t-4}	$(0,2)(0,8)^4$	$(0,4)(0,6)^4$	$(0,6)(0,4)^4$	$(0,8)(0,2)^2$

Jika bobot ini diplot, dapat dilihat bahwa bobot tersebut menurun secara eksponensial, dari sana nama *Exponential Smoothing* muncul. (Perlu dikemukanan bahwa walaupun tujuan adalah menentukan α yang meminimumkan MSE pada kelompok data pengujian, penaksiran yang terjadi dalam Expnential Smoothing adalah masalah non-linear.)

Cara lain untuk menuliskan persamaan (4) adalah dengan susunan sebagai berikut:

$$F_{t+1} = F_t + \alpha(X_t - F_t) \quad \text{..... (7)}$$

Secara sederhana

$$F_{t+1} = F_t + \alpha(e_t) \quad \text{..... (8)}$$

Dimana e_t adalah kesalahan ramalan (nilai sebenarnya dikurangi ramalan) untuk periode *t*. Dari dua bentuk F_{t+1} ini dapat dilihat bahwa ramalan yang dihasilkan dari SES secara sederhana merupakan ramalan yang lalu ditambah suatu penyesuaian untuk kesalahan yang terjadi pada ramalan terakhir. Dalam bentuk ini terbukti bahwa jika α mempunyai nilai mendekati 1, maka ramalan yang baru akan mencakup penyesuaian kesalahan yang besar pada ramalan sebelumnya. Sebaliknya, jika α mendekati 0, maka ramalan yang baru

akan mencakup penyesuaian yang sangat kecil. Jadi, pengaruh besar kecilnya α benar-benar analog (dalam arah yang berlawanan) dengan pengaruh memasukan jumlah pengamatan yang kecil atau besar pada perhitungan rata-rata bergerak. Perlu juga di perhatikan bahwa *Exponential Smoothing* tunggal akan selalu mengikuti setiap trend dalam data yang sebenarnya, karena yang dapat dilakukannya tidak lebih dari mengatur ramalan mendatang dengan suatu persentase dari kesalahan yang terakhir.

Persamaan (7) dan persamaan (8) mengandung prinsip dasar dari umpan balik yang negatif, karena persamaan ini berperan sebagai proses kontrol yang dilakukan oleh alat otomatis seperti termostat, pilot otomatis, dan sebagainya. Kesalahan ramalan masa lalu dipakai untuk mengoreksi ramalan mendatang pada arah yang berlawanan dengan kesalahan tersebut. Penyesuaian tersebut tetap berlangsung sampai kesalahannya dikoreksi. Prinsip ini sama dengan prinsip alat pengendali otomatis yang mengarah kepada kesetimbangan begitu terjadi penyimpangan (kesalahan). Prinsip ini, yang tampaknya sederhana, memainkan peranan yang sangat penting dalam peramalan. Jika digunakan secara tepat prinsip ini dapat digunakan untuk mengembangkan suatu proses mengatur diri sendiri (*self-adjusting process*) yang dapat mengoreksi kesalahan peramalan secara otomatis.

Tabel 2.2 Peramalan Pengiriman Alat Pembuka Kaleng

Bulan	Periode Waktu	Nilai Pengamatan (pengiriman)	Nilai Pemulusan Eksponensial (α)		
			0,1	0,5	0,9
Jan	1	200,0	-	-	-
Feb	2	135,0	200,0	200,0	200,0
Mar	3	195,0	193,5	167,5	141,5
Apr	4	197,5	193,7	181,3	189,7
Mei	5	310,0	194,0	189,4	196,7
Jun	6	175,0	205,6	249,7	298,7
Jul	7	155,0	202,6	212,3	187,4
Agst	8	130,0	197,8	183,7	158,2
Sept	9	220,0	191,0	156,8	132,8
Okt	10	277,5	193,9	188,4	211,3
Nov	11	235,0	202,3	233,0	270,9
Des	12	-	205,6	234,0	238,6
			Periode Pengujian		

Analisis Kesalahan	2-11 α = 0,1	2-11 α = 0,5	2-11 α = 0,9
Nilai Tengah Kesalahan	5,56	6,80	4,29
Nilai Tengah Kesalahan Absolut	47,76	56,94	61,32
Nilai Tengah Kesalahan Persentase Absolut (MAPE)	24,58	29,20	30,81
Deviasi Standar Kesalahan (Tak Berbias)	61,53	69,13	74,69
Nilai Tengah Kesalahan Kuadrat (MSE)	3438,33	4347,24	5039,37
Statistik Durbin-Watson	1,57	1,84	2,30
Statistik *U* dari Theil	0,81	0,92	0,98
Rata-rata Batting dari McLaughlin	319,12	307,64	301,79

Tabel 2.2 menunjukan hasil pemulusan eksponensial dari pengiriman alat pembuka kaleng dengan menggunakan nilai α 0,1; 0,5; 0,9. *Forecasting* dengan menggunakan *Exponential Smoothing* tunggal dapat dilakukan dengan menggunakan persamaan (4). Sebagai contoh, pada tabel 2.2 ramalan untuk periode 12 (Desember) bila α = 0,1 dihitung sebagai berikut :

$$F_{12} = \alpha X_{11} + (1-\alpha)F_{11}$$
$$= (0{,}1)(235{,}0) + (0{,}9)(202{,}3)$$
$$= 205{,}6$$

Demikian pula, bila α = 0,9 persamaan (4) memberikan peramalan untuk periode 12

$$F_{12} = (0{,}9)(235{,}0) + (0{,}1)(270{,}9) = 238{,}6$$

Perhatikan bahwa pemilihan α mempunyai pengaruh yang besar pada ramalan Desember, dan nilai MAPE untuk periode 2 sampai 11 berkisar dari 24,58 persen (untuk α = 0,1) dan 30,81 persen (untuk α = 0,9).

Exponential Smoothing tunggal memerlukan sedikit penyimpangan data dan perhitungan. Oleh karena itu metode ini menarik jika diperlukan peramalan untuk sejumlah besar item. Salah satu hal yang perlu diperhatikan berkaitan dengan tahap inisialisasi SES. Sebagai contoh, untuk dapat memulai sistem *forecasting* SES kita memerlukan F_1 karena

$F_2 = \alpha X_1 + (1-\alpha)F_1$.

Karena itu nilai F_1 tidak diketahui, kita dapat menggunakan nilai observasi pertama (X_1) sebagai ramalan pertama $(F_1 = X_1)$ dan kemudian dilanjutkan dengan menggunakan persamaan (4). Ini merupakan salah satu

metode inisialisasi. Kemungkinan lain adalah merata-ratakan empat atau lima nilai pertama dalam kelompok data, dan menggunakannyya sebagai ramalan pertama. Dari persamaan (6) bahwa ralaman awal memainkan peran dalam semua ramalan selanjutnya. Suku terakhir pada persamaan (6) adalah
$(1-\alpha)^N F_{1-(N-1)}$.

2.1.2 Regresi Sederhana

Istilah regresi sederhana akan dikaitkan dengan setiap regresi dari suatu ukuran *Y* tunggal (variabel tidak bebas) terhadap ukurang *X* tunggal (variabel bebas). Secara umum akan melibatkan himpunan *n* pasangan hasil pengamatan (tabel 2.3) yang dinyatakan sebagai berikut:

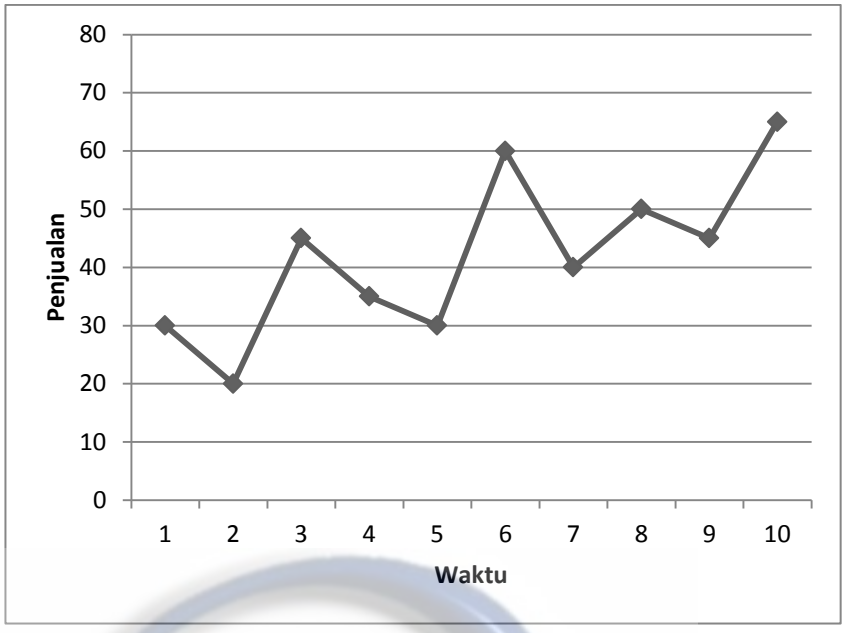
$$\{X_i, Y_i\}\ untuk\ i = 1,2,3,4, \dots, n \quad \text{..... (9)}$$

Tabel 2.3 Data Penjualan selama 10 Periode

(X_i) Periode	(Y_i) Penjualan	(X_i Y_i) Pasangan Data
1	30	(1 , 30)
2	20	(2 , 20)
3	45	(3 , 45)
4	35	(4 , 35)
5	30	(5 , 30)
6	60	(6 , 60)
7	40	(7 , 40)
8	50	(8 , 50)
9	45	(9 , 45)
10	65	(10 , 65)

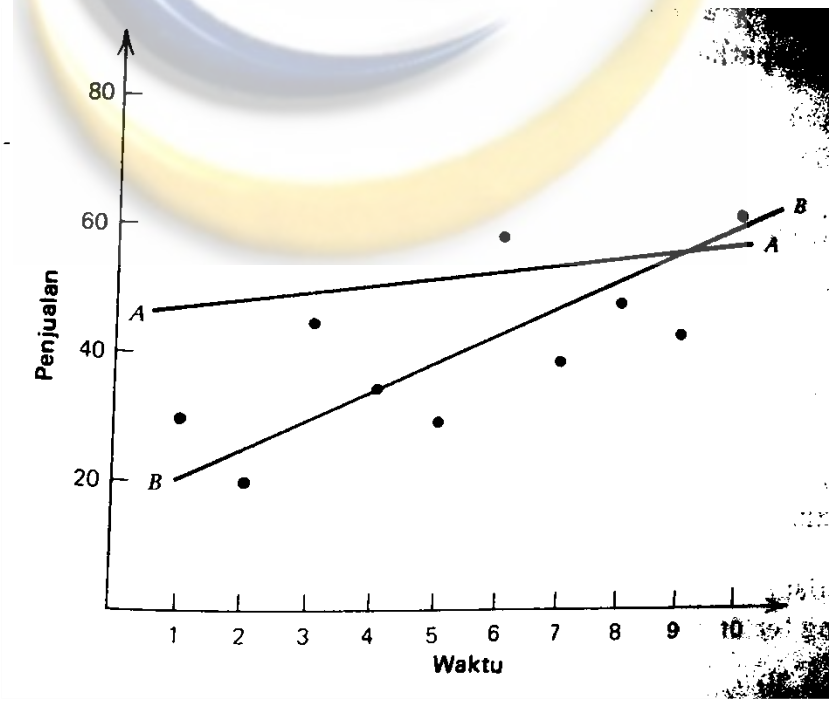
Setiap pasangan dapat digambarkan sebagai suatu titik, dan berdasarkan perjanjian, nilai-nilai *Y* dinyatakan pada sumbu vertikal (ordinat) sedangkan nilai-nilai *X* dinyatakan pada sumbu horizontal (absis) seperti yang ditunjukan pada gambar 2.10.

Hubungan linear antara *Y* dan *X* dan masalah penentuan *"kesesuaian terbaik"* (*best fitting*) dari suatu garis lurus (linear) melalui titik-titik yang sudah digambarkan.



Gambar 2.2 Grafik hubungan antara penjualan dan waktu

Y kedua-duanya merupakan ukuran-ukuran ekomoni, sebagai contoh bahwa kemungkinan *Y* dapat dijelaskan oleh *X*. Tabel 2.3 dan gambar 2.11 menyajikan data dan gambar titik hubungan antara penjualan dan waktu. Untuk mengenal regresi, silahkan perhatikan Gambar 2.3. Garis AA maupun baris BB tampak tidak cocok dengan data pengamatan.

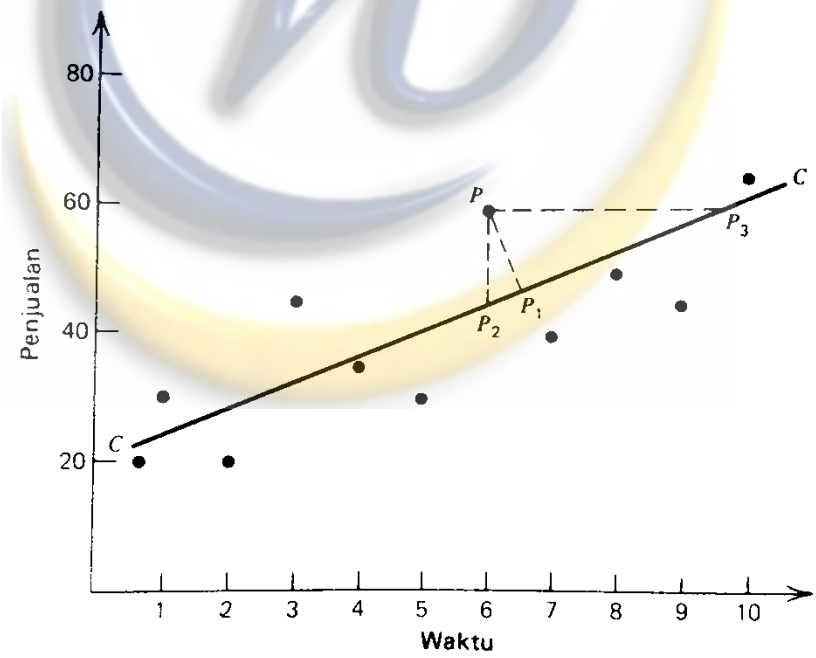


Sumber: buku terjemahan "Forecastting 2nd Edition" halaman 177

Gambar 2.3 Garis lurus yang cocok melalui data penjualan

Bagaimana garis-garis yang berbeda dapat ditaksir titik-titik *goodness of fit*-nya? Gambar 2.4 menunjukan beberapa cara yang lain untuk mengevaluasi *goodness of fit* atau *badness of fit*, *erorr of fit* untuk titik P dan garis CC dapat ditetapkan sebagai berikut:

1. Jarak tegak lurus dari P ke CC (garis PP_1),
2. Panjang garis horisontal dari P ke CC (garis PP_3),
3. Panjang garis vertikal dari P ke CC (garis PP_2),
4. Nilai mutlak panjang garis dari P ke garis CC (secara vertikal atau pun secara horisontal),
5. Kuadrat panjang garis dari P ke garis CC (secara vertika, horisontal atau pun tegak lurus),
6. Variasi bobot dari setiap hal di atas (dalam hal ini panjang garis dari setiap titip seperti P ditetapkan suatu bobot terpisah tatau *importance*).



Sumber: buku terjemahan "Forecastting 2nd Edition" halaman 178

Gambar 2.4 Cara mengukur garis berdasarkan error

Nampaknya sudah jelas bahwa *goodness of fit* membutuhkan suatu pemilihan dari berbagai arti kata *error*. Terdapat suatu prosedur regresi yang

dikenal sebagai MAD (*mean absoulte deviation*) dan terdapat beberapa variasi prosedur kuadrat terkecil terbobot atau terdiskonto (*discounted or weighted least sssquares*). Akan tetapi kita akan memusatkan perhatian pada bentuk konvensional yang dikenal sebagai kuadrat terkecil biasa (*ordinary least squares = OLS*) atau biasa disingkat LS.

Prinsip perhitungannya adalah secara langsung. Jika kita menggunakan *Y* sebagai variabel tidak bebas dan *X = t* sebagai variabel bebas, maka tujuan yang akan kita capai adalah mendapatkan persamaan garis lurus:

$$\gamma_t = a + bt \quad \text{..... (10)}$$

Demikian sehingga untuk setiap nilai waktu *t* tertentu, kesalahan kuadrat:

$$(Y_t - \gamma_t)^2 = e_t^2 \quad \text{..... (11)}$$

Jika dijumlahkan akan menghasilkan total minimum. Ini merupakan prosedur LS dan kesalahan dinyatakan sebagai panjang garis vertikal dari titik tertentu ke garis $\gamma_t = a + bt$.

2.2 Perencanaan Produksi

Perencanaan produksi menurut Bock yang dikutip oleh Ashok Marwaha dalam Tesisnya yang berjudul "Production Planning and Inventory Control Modelling in a Compos*ite* Textile Mill" adalah:

> *"Production Planning involves setting production levels for several periods in the future and assigning general responsibility to provide a data for making decisions on the size and composition of the labor force, capital equipment and plant additions, and planned inventory levels. The ability to meet demand levels generated by possible alternative sales programs is also a function of production planning."*

Rencana produksi yang digunakan untuk berbagai tujuan. Satu contoh adalah rencana penggunaan produksi untuk membantu menentukan jumlah peralatan modal baru yang akan dibeli di masa depan. Dalam contoh ini rencana meliputi lima, delapan, atau bahkan sepuluh tahun kedepan akan diperlukan dan akan menunjukkan pekerjaan produksi dilakukan dan modal yang diperlukan untuk menyelesaikan pekerjaan ini.

Pada saat yang sama yang rencana produksi meliputi beberapa tahun berikutnya diperlukan, rencana lain yang meliputi jangka waktu lebih pendek juga mungkin diperlukan. Rencana ini mungkin mencakup hanya beberapa bulan berikutnya dan dapat digunakan untuk mengatur tingkat agregat produksi untuk memenuhi permintaan rencana dan direncanakan tingkat inventaris masa depan.

Penjadwalan produksi biasanya meliputi satu waktu yang lebih singkat dari pada perencanaan produksi. Jadwal produksi menentukan bagaimana persyaratan produksi dalam beberapa minggu ke depan akan ditugaskan ke Departemen spesifik, proses, mesin, dan operator untuk memenuhi tenggang waktu yang diharapkan oleh Departemen penjualan dan tingkat inventaris yang diinginkan.

2.3 Pemodelan UML

Unified Modelling Language (UML) adalah sebuah "bahasa" yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan standar untuk merancang model sebuah sistem.

Dengan menggunakan UML kita dapat membuat model untuk semua aplikasi piranti lunak, dimana aplikasi tersebut berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa-bahasa berorientasi objek seperti PHP, C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C.

Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (*Object-Oriented Design*), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Enginerring*).

Sejarah UML sendiri cukup panjang. Sampai era tahun 1990 puluhan metodologi permodelan berorientasi objek telah bermunculan di dunia. Diantaranya adalah: metodologi booch, metodologi coad, metodologi OOSE, metodologi OMT, metodologi shlaer-mellor, metodologi wirfs-brock, dsb. Pada tahun 1995 di-*release* draft pertama dari UML (versi 0.8). Sejak tahun 1996 pengembangan tersebut dikoordinasikan oleh **Object management Group** (OMG – http://www.omg.org). Tahun 1997 UML versi 1.1 muncul, dan pada saat itu versi terbaru adalah versi 1.5 yang dirilis pada bulan maret 2003. Booch, Rumbaugh dan Jacobson menyusun tiga buku serial tentang UML pada tahun 1999. Sejak saat itulah UML telah menjelma menjadi standar bahasa pemodelan untuk aplikasi berorientasi objek.

Dari berbagai penjelasan rumit yang terdapat di dokumen dan buku-buku UML. Sebenarnya dasar UML bisa kita rangkumkan dalam gambar berikut:

Major Area	*View*	*Diagrams*	*Main Concepts*
structural	static view	class diagram	class, association, generalization, dependency, realization, interface
	use case view	use case diagram	use case, actor, association, extend, include, use case generalization
	implementation view	component diagram	component, interface, dependency, realization
	deployment view	deployment diagram	node, component, dependency, location
dynamic	state machine view	statechart diagram	state, event, transition, action
	activity view	activity diagram	state, activity, completion transition, fork, join
	interaction view	sequence diagram	interaction, object, message, activation
		collaboration diagram	collaboration, interaction, collaboration role, message
model management	model management view	class diagram	package, subsystem, model
extensibility	all	all	constraint, stereotype, tagged values

Sumber: ilmukomputer “Pengantar Unified Modeling Language (UML)” halaman 3

Gambar 2.5 Rangkuman Konsep dasar UML

Abstraksi konsep dasar UML yang terdiri dari *structural classification*, *dynamic behavior*, dan *model management*, bisa dipahami dengan mudah apabila melihat gambar diatas dari *Diagrams*. *Main concepts* bisa kita pandang sebagai *term* yang akan muncul pada saat kita membuat diagram. Dan view adalah kategori dari diagram tersebut.

2.3.1 *Use Case Diagram*

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditentukan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah *use case diagram* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case diagram* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu.

Use case diagram dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem. Sebuah *use case diagram* dapat meng-*include* fungsionalitas *use case* laini sebagaimana bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang ter-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal. Sebuah *use case* dapat di-*include* oleh sebuah dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan menarik keluar fungsionalitas yang *common*.

Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.

2.3.2 *Class Diagram*

Class diagram adalah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desian berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut

(metoda/fungsi). *Class diagram* menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain.

Class diagram memiliki tiga area pokok:

1. Nama (dan *stereotype*)
2. Atribut
3. Metoda

Atribut dan metoda dapat memiliki salah satu sifat berikut:

1. *Private,* tidak dapat dipanggil dari luar class yang bersangkutan
2. *Protected,* hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya
3. *Public,* dapat dipanggil oleh siapa saja

Class dapat merupakan implementasi dari sebuah *interface*, yaitu *class* abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class*. Dengan demikian *interface* mendukung resolusi metoda pada saat *run-time*. Sesuai dengan perkembangan *class model, class* dapat dikelompokan menjadi *package*. Kita juga dapat membuat diagram yang terdiri dari atas *package*.

Hubungan antar *class*:

1. Assosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain. Panah *navigability* menunjukan arah *query* antar *class*.
2. Agregasi, yaitu hubungan yang menyatakan bagian (“terdiri atas..”).
3. Pewarisan, yaitu hubungan hirarkis antar *class*. *Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.
4. Hubungan dinamis, yaitu rangkaian pesan (*message*) yang di-*passing* dari satu *class* kepada *class* lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence diagram*.

2.3.3 *Activity Diagram*

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses peralel yang mungkin terjadi pada berbagai eksekusi. *Activity diagram* merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas. Sams seperti *state*, standar UML menggunakan segiempat dengan sudut membuat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan behavior pada kondisi tertentu. Untuk mengilustrasikan proses-proses parelel (*fork* dan *join*) dengan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal. *Activity diagram* dapat dibagi menjadi beberapa *object swimlanei* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.

2.3.4 *Sequence Diagram*

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* antar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respon dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dair apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan.

Masing-masing objek, termasuk aktor, memiliki *lifetime* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya.

Pada fase desain berikutnya, *message* akan dipetakan menjadi operasi/metoda dari *class*. *Activation bar* menunjukan lamanya eksekusi sebuah proses, biasanya dengan diterimanya sebuah *message*. Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan *icon* khusus untuk objek *boundary, controller* dan *persistent entity*.

2.4 DBMS Oracle

Database adalah kumpulan data yang diperlakukan sebagai sebuah unit. Tujuan dari *database* adalah untuk menyimpan dan mendapatkan informasi yang berkaitan. Sebuah *database server* adalah kunci manajemen informasi. Secara umum, *server* mampu menangangi jumlah data yang cukup besar dalam lingkungan *multiuser* sehingga dapat menangani banyak pengguna dalam waktu yang bersamaan dalam mengakses data. Sebuah *database server* juga dapat mencegah akses yang tidak memiliki otoritasi dan menyediakan solusi yang efisien untuk pemulihan dari kesalahan.

Oracle *database* adalah *database* pertama yang di desain untuk komputasi bersekala perusahaan, lebih flesibel dan lebih efektif untuk mengelola informasi dan aplikasi. Komputasi bersekala perusahaan membuat lingkungan standar industri yang besar, penyimpanan standar dan banyak server. Dengan arsitektur serperti itu, setiap sistem dapat secara cepat mengakses komponen data. Tidak perlu menyediakan *hardware* tambahan untuk mendukung beban data, karena data dapat dengan mudah di tambahkan atau di alokasikan dari lingkungan sumber daya yang dibutuhkan.

Database memiliki struktur fisik dan strukturl *logic*. Karena struktur fisik dan struktur *logic* terpisah, data disimpan secara fisik dan dapat di olah tanpa mempengaruhi struktur penyimpanan *logic*.

2.4.1 Komputasi Grid

Komputasi grid adalah sebuah arsitektur teknologi informasi (TI) yang memperlihatkan lebih fleksibel dan sistem informasi perusahaan yang lebih murah. Dengan komputasi grid, kelompok-kelompok yang berdiri sendiri,

hardware standar dan komponen *software* dapat terhubung dan bergabung berdasarkan perubahan bisnis yang dibutuhkan.

Mode dari komputasi grid menyelesaikan beberapa masalah umum yang dihadapi oleh perusahaan IT:

1. Aplikasi kurang mampu memanfaatkan sumber daya *hardware* yang ada
2. *Monolithic*, sistem yang berat sangat mahal dan sulit untuk di-*maintain* atau di rubah
3. Pemisahan dan penyatuan informasi tidak dapat dimanfaatkan sepenuhnya oleh perusahaan

2.4.2 Arsitektur Aplikasi

Ada dua arsitektur umum dari *database*, yaitu *client/server* dan *multitier*. Serperti komputasi *internet* yang menjadi lebih populer di lingkungan komputasi, banyak *database management system* berpindah ke lingkungan *multitier*.

Sistem *database* oracle memiliki kelebihan dapat dengan mudah diproses sebagai arsitektur *client/server*. Pada arsitektur ini, sistem *database* memiliki dua bagian: *front-end* atau *client*, dan *back-end* atau *server*.

Client adalah aplikasi *database* yang menginisiasikan permintaan dari pengguna untuk dioperasikan/dilakukan oleh *database server*. Permintaa-perminttan, proses-proses, dan data dikelola oleh *server*. *Client* dapat mengoptimalkan pekerjaan *server* tersebut. Sebagai contoh, *client* mungkin tidak membutuhkan kapasitas penyimpanan yang cukup besar, atau mungkin memiliki kelebihan dari grafik. Biasanya, *client* berjalan pada komputer yang berbeda dari *database server*. *Client* dapat berjalan secara bersama-sama pada satu *server*.

Server menjalankan *software oracle database* dan menangani permintaan *client* secara bersamaan, berbagi akses data. *Server* menerima dan memproses permintaan yang berasal dari aplikasi *client*. Komputer yang menjadi *server* dapat mengoptimalkan tugasnya. Sebagai contoh, komputer *server* memiliki kapasitas penyimpanan yang cukup besar dan prosesor yang cepat.