BAB 2

LANDASAN TEORI

2.1 Pemeliharaan (Maintenance)

Definisi pemeliharaan menurut O'Connor (2001,p407) adalah suatu kegiatan untuk memelihara dan menjaga fasilitas yang ada serta meperbaiki. Melakukan penyesuaian atau pengantian yang diperlukan untuk mendapatkan suatu kondisi operasi produksi agar sesuai dengan perencanaan yang ada.

Perawatan adalah sebuah operasi atau aktivitas yang harus dilakukan secara berkala dengan tujuan untuk mempercepat pergantian kerusakan peralatan dengan *resources* yang ada. Perawatan juga ditujukan untuk mengembalikan suatu sistem pada kondisinya agar dapat berfungsi sebagaimana mestinya, memperpanjang usia kegunaan mesin, dan menekan *failure* sekecil mungkin.(sumber: http://www.ittelkom.ac.id/library)

Menurut Jr.Patton(1995,p23) Pengertian maintanace secara umum yaitu serangkaian aktivitas (baik bersifat teknis dan *administrative*) yang di perlukan mempertahankan dan menjaga suatu produk atau system tetap berada pada dalam kondisi aman, ekonomis, efisien dan pengoperasian optimal. Aktivitas perawatan sangat diperlukan karena :

- Setiap peralatan punya umur pengunaan (*useful life*). Suatu saat dapat mengalami kegagalan dan kerusakan.
- Kita dapat mengetahui dengan tepat kapan peralatan akan mengalami kerusakan
- Manusia selalu berusaha untuk meningkatkan umur penggunaan dengan melakukan perawatan (*maintenance*).

Yang menjadi musuh utama bagian perawatan adalah *breakdown, deterioration*, dan konsekuensi dari semua tipe kejadian yang tidak terencana.

Maintenance sangat berperan penting dalam kegiatan produksi dari suatu perusahaan yang menyangkut kelancaran dan kemacetan produksi, volume produksi serta agar produk dapat diproduksi dan di terima konsumen tepat pada waktunya dan menjaga agar tidak ada sumber daya yang menganggur karena kerusakan (*downtime*) pada mesin sewaktu proses produksi sehingga dapt meminimalkan biaya kehilangan produksi.

2.2 Jenis pemeliharaan

Kegiatan pemeliharaan dapat dibedakan menjadi 3 jenis yaitu *corrective maintenance* (*breakdown maintenance*), *preventive maintenance* dan *total productive maintenance* (perawatan keseluruhan).

2.2.1 Corrective maintenance

Menurut pendapat O'connor (2001,p401) *Corective maintenance* merupakan kegiatan perawatan yang dilakukan setelah mesin atau fasilitas produksi mengalami gangguan atau kerusakan sehingga tidak dapat berfungsi dengan baik. Aktivita *Corrective maintenance* sering disebut aktivitas perbaikan. *Corrective maintenance* biasanya tidak dapat kita rencanakan telebih dahulu karena kita hanya bisa memperbaikinya setelah terjadi kerusakan, bahkan terkadang perbaikan tersebut bisa tertunda dan terlambat.

Perbaikan yang dilakukan akibat terjadinya kerusakan dapat terjadi akibat tidak dilakukannya *preventive maintenance* maupun telah diterapkan *preventive maintenance*, akan tetapi sampai pada suatu waktu tertentu fasilitas produksi atau peralatan yang ada tetap rusak. Dalam hal ini, *corrective maintenance* bresifat perbaikan menunggu sampai

kerusakan terjadi dahulu, kemudian baru diperbaiki agar fasilitas produksi maupun peralatan yang ada dapat dipergunakan kembali dalam proses produksi sehingga operasi dalam proses produksi dapat berjalan lancer dan kembali normal.

Apabila perushaan hanya mengambil tindakan *corrective maintenance* saja, maka terdapat factor ketidakpastian akan lancarnya fasilitas dalam proses produksi maupun peralatannya sehingga menimbulkan efek-efek yang dapat menghambat kegiatan produksi jikalau terjadi gangguan kerusakan tiba-tiba pada fasilitas produksi perushaan.

Tindakan *corrective maintenance* ini keliatannya lebih murah biayanya dibandingkan tindakan *preventive maintenance*. Tentu saja pernyataan ini benar selama gangguan kerusakan belum terjadi pada fasilitas maupun peralatan ketika proses produksi berlangsug. Namun, saat kerusakan terjadi selama proses produksi maka biaya perawatan akan mengalami peningkatan akibat terhentinya proses produksi. Selain itu biaya-biaya perawatan dan pemeliharaan akan membengkak pada saat terjadinya kerusakan tersebut. Dengan demikian, dapat disimpulkan bahwa tindakan *corrective maintenance* lebih memusatkan permasalahan setelah permasalahan itu terjadi, bukan menganalisa masalah untuk mencegah agar tidak terjadi.

Menurut Pendapat O'connor (2001,p401) *Corrective maintenance* dapat dihitung dengan MTTR (*mean time to repair*) dimana time to repair memiliki aktivitas yang biasanya dibagi menjadi 3 group:

- *Preparation time*: Waktu yang dibutuhkan untuk persiapan seperti mencari orang untuk perkerjaan , travel, peralatan sudah dipenuhi atau belum dan tes perlengkapannya.

- *Active Maintenance Time*: Waktu yang di perlukan untuk melakukan pekerjaan tersebut. Meliputi waktu mempelajari repair chart sebelum actual repair dimulai dan waktu yang dihabiskan menverifikasi bahwa kerusakan tersebut di perbaiki.
- *Delay Time*: Waktu yang dibutuhkan untuk menunggu komponen dalam mesin unutk diperbaiki.

Corrective maintenance merupakan studi dalam menentukan tindakan yang diperlukan untuk mengatasi kerusakan-kerusakan atau kemacetan yang terjadi berulang kali. Tindakan perawatan ini bertujuan untuk mencegah terjadinya kerusakana yang sama. Prosedur ini ditetapkan pada peralatan atau mesin yang sewaktu waktu dapat terjadi kerusakannya.

Pada umumnya usaha untuk mengatasi kerusakan itu dapat dilakuakn dengan cara sebagai berikut:

1. Mencatat data trouble/kerusakan, kemudian melakukan peningkatan peralatan sehingga kerusakan yang sama tidak terjadi lagi.
2. Meng-improve peralatan sehingga mejadi lebih mudah.
3. Merubah proses.
4. Merancang kembali komponen yang gagal.
5. Mengganti dengan komponen yang baru
6. Meningkatakan prosedur perawatan preventif.
7. Meninjau kembali dan merubah sistem pengoperasian.

Dengan demikian, didapatkan kesimpulan bahwa pemeliharaan korektif memusatkan permasalahan setelah permasalahan itu terjadi, bukan menganalisa masalah untuk mencegahnya agar tidak terjadi.

2.2.2 Perventive Maintenance

Menurut pendapat Ebelling(1997,189), Pervetive maintence adalah pemeliharaan yang dilakukan tejadwal, umunya secara periodik, dimana sejumlah tugas pemeliharaan seperti inspeksi, perbaikan, penggantian, pembersihan, pelumasan dan penyesuaian dilaksanakan.

Dengan adanya preventive Maintenance diharapkan semua mesin yang ada akan terjamin kelancaran proses kerjanya shingga tidak ada yang terhambat dalm proses kerjanya sehingga tidak ada yang terhambat dalam proses produksinya dan bisa selalu dalam keadaan optimal.

2.2.3 Total productive maintenance

Secara teoritis, total biaya pemeliharaan dapat digambarkan bahwa biaya pemeliharaan korektif akan berbanding terbalik dengan pemeliharaan. Pemeliharaan secara produktivitas dapat dilakukan dengan jalan berikut (Tampubolon, 2004, p253):

1. Mendesain mesin atau peralatan yang memiliki reabilitas tinggi, mudah di operasikan dan mudah dipelihara.
2. Analisa biaya investasi untuk mesin atau peralatan dengan pelayanan (service) dari pemasok dan biaya-biaya pemeliharaannya.
3. Mengembangkan perencanaan pemeliharaan *perventif* yang dapat dimanfaatkan secara praktis oleh operator, bagian pemeliharaan, dan teknisi.
4. Melatih pekerja untuk mengoperasikan mesin atau peralatan, termasuk cara memeliharanya.

2.3 Konsep-Konsep Pemeliharaan

2.3.1 Konsep Hubungan Waktu dalam Maintenance

Keterangan istilah dalam maintenance :

1. *Up time*

   Waktu (*period of time*) dimana mesin/peralatan ada dalam kondisi baik sehingga dapat melakukan fungsi seperti seharusnya (melakukan fungsi dalam kondisi yang ditetapkan dan dengan maintenance yang ditetapkan pula)

2. *Down Time*

   Waktu (*period of time*) diamana mesin/peralatan tidak berada dalam kondisi untuk dapat melakukan fungsinya. *Downtime* dihitung mulai saat mesin tidak berfungsi sampai mesin kembali dalam keadaan dapat berfungsi seperti seharusnya, setelah dilakukan perbaikan.

3. *Operating Time*

   Waktu (*period of time*) dimana mesin melakukan fungsi seperti seharusnya

   $$OPERATING\ TIME < UP\ TIME$$

4. *Standby Time*

   Waktu (*period of time*) dimana mesin berada dalam kondisi untuk dapat berfungsi seperti seharusnya, tetapi mesin tidak dioperasikan.

   $$Up\ time = Operating\ Time + Standby\ Time$$

5. *Maintenance Time*

   Waktu dimana kegiatan maintenance dilakukan termasuk *delay-delay* yang terjadi selama pelaksanaan kegiatan.

6. *Active Maintenance Time*

Bagian dari maintenance time, dimana kegiatan/pekerjaan *maintenance* benar-benar dilakukan

7. *Logistic Time*

   Waktu dalam *downtime*, dimana kegiatan *maintenance* belum dapat dimulai karena alasan logistik.

8. *Adminstrative Time*

   Waktu dalam *downtime*, dimana kegiatan *maintenance* belum dapat dimulai karena alasan *administrative.*

9. *Corrective Maintenance Time*

   Waktu dalam active maintenance time, dimana dilakukan kegiatan *corrective maintenance.*

10. *Perventive Maintenance Time*

    Waktu dalam active maintenance time, dimana dilakukan *kegiatan preventive maintenance.*

    (sumber: Jr.Patton(1995,p124-125))

2.3.2 Konsep *Breakdown Time*

Menurut Pendapat Jr.Patton(1995,p130),*Breakdown* dapat didefinisikan sebagai berhentinya mesin pada saat produksi yang melibatkan engineering dalam perbaikan, biasanya mengganti sparepart yang rusak, dan lamanya waktu lebih dari 5 menit (berdasarkan *OPI-Overall Performance Index*).

Downtime mesin merupakan waktu menganggur atau lama waktu dimana unit tidak dapat lagi menjalankan fungsinya sesuai dengan yang diharapkan. Hal ini terjadi apabila suatu unit mengalami masalah seperti kerusakan mesin yang dapat mengganggu kinerja mesin secara keseluruhan termasuk kualitas produk yang dihasilkan atau

kecepatan produksinya sehingga membutuhkan waktu tertentu untuk mengembalikan fungsi unit-unit tersebut pada konsis semula,

Unsur-Unsur dalam *Downtime* :

1. *Maintenance Delay*

   Waktu yang dibutuhkan untuk menunggu ketersediaan sumber daya *maintenance* untuk melakukan proses perbaikan. Sumber daya maintenance dapat berupa alat bantu, teknisi, alat tes, komponen pengganti, dan lain-lain.

2. *Supply delay*

   Waktu yang dibutuhkan untuk *personel maintenance* untuk memperoleh komponen yang dibutuhkan dalam proses perbaikan. Terdiri dari lead time *administrative, lead time* produksi, dan waktu transportasi komponen pada lokasi perbaikan.

3. *Acces time*

   Waktu untuk mendapatkan akses ke komponen yang mengalami kerusakan.

4. *Diagnoses time*

   Waktu yang dibutuhkan untuk menentukan penyebab kerusakan dan langkah perbaikan yang harus ditempuh untuk memperbaiki kerusakan.

5. Repaire or replacement unit

   Waktu aktual yang dibutuhkan untuk menyelesaikan proses pemulihan setelah permasalahan dapat diidentifikasikan dan akses ke komponen rusak dapat dicapai.

6. Verification and alignment

   Waktu untuk memastikan bahwa fungsi daripada suatu unit telah kembali pada kondisi operasi semula.

2.3.3 Konsep *Reliability(*Kehandalan)

Yang dimaksud dengan kehandalan adalah:

1. Peluang sebuah komponen atau sistem akan dapat beroperasi sesuai fungsi yang diinginkan untuk suatu periode waktu tertentu ketika digunakan dibawah kondisi operasi yang telah di tetapkan. Ebeling, (1997, p5)
2. Peluang sebuah komponen, sub-sistem atau sistem melakukan fungsinya dengan baik, seperti yang dipersyaratkan, dalam kurun waktu tertentu dan dalam kondisi operasi tertentu pula. (http://www.migas-indonesia.com/files/article/Reliability_an_Introductory_Note.doc, Ahmad Taufik)

2.3.4 Konsep Availability

Menurut Ebeling(1997,p5), Availability adalah probabilitas komponen atau sistem dapat beroperasi sesuai dengan fungsinya pada kondisi operasi normalnnya apabila tindakan perawatan pencegahan dan pemeriksaan dalam arti availability merupakan proporsi waktu teoritis yang tersedia untuk komponen dalam system dapat beroperasi dengan baik.

2.3.5 Konsep Maintainability

Menurut Ebeling (1997, p6) definisi maintainability adalh probalitas bahwa suatu kompone yang rusak akan diperbaiki dalam jangka waktu (T), dimana pemeliharaan (maintanability) dilakukan sesuai dengan ketentuan yang ada.

Menurut pendapat O'Connor (2001, p401) kebanyakan sistem *engineered* itu dipelihara (*dimaintain*), sistem akan diperbaiki kalau terjadi kerusakan dan pemeliharaan akan dibentuk pada sistem tersebut untuk menjaga pengoperasian yang ada dalam sistem pemeliharaan ini (*system maintainability*).

2.4 Fungsi Distribusi Kerusakan

Menurut Ebeling (1997,12), Distribusi kerusakan merupakan ekspresi matematis usia dan pola kerusakan mesin atau peralatan. Kerusakn setiap mesin akan mempengaruhi kedekatan yang digunakan dalm menguji kesesuaian dan menghitung parameter fungsi distribusi kerusakan.

Pada umunya, karakteristik dari kerusakan setiap mesin tidaklah sama terutama jika dioperasikan dalam kondisi lingkungan yang berbeda. Suatu peralatan atau mesin yang memiliki karakteristik dan dioperasikan dalam kondisi yang sama juga mungkin akan memberikan nilai selang waktu antar kerusakan yang berbeda.

2.5 Fungsi Distribusi Kumulatif

Fungsi distributisi kumulatif merupakan fungsi yang menggambarkan probabilitas sebelum waktu t. Probabilitas suatu system atau peralatan mengalami kegagalan dalam beroperasi sebelum waktu t, yang merupakan fungsi dari waktu yang secara matematis dapat dinyatakan sebagai:

$$F(t) = \int f(t)dt \text{ untuk } t \geq 0$$

Keterangan

F(t): Fungsi *distributive* kumulatif

F(t): Fungsi Kepadatan Peluang

Jika $t = \infty$ maka F(t) =1

2.6 Fungsi Kehandalan

Berdasarkan pendapat dari Ebeling (1997, p23) kehandalan merupakan probabilitas sistem atau komponen akan berfungsi hingga waktu tertentu (t). Pengertian

fungsi kehandalan adalah probabilitas suatu sistem atau komponen akan beroperasi dengan baik tanpa mengalami kerusakan pada suatu periode waktu t dalam kondisi operasional yang telah ditetapkan. Probabilitas kerusakan dari suatu fungsi waktu dapat dinyatakan sebagai berikut:

F (t) = P ($T \leq t$), dimana:

T = variable acak continiu yang menyatakan saat terjadinya kegagalan

F(t) = probabilitas bahwa kerusakan terjadi sebelum waktu T = t (fungsi distribusi).

Reliability diuraikan sebagai berikut:

R (t) = P(T>t), dimana:

R (t) = distribusi kehandalan, probabilitas bahwa kegagalan tidak akan terjadi sebelum t, atau probabilitas bahwa waktu kerusakan lebih besar atau sama dengan t.

2.7 Laju kerusakan

Laju kerusakan (*failure rate*) dari suatu peralatan atau mesin pada waktu t adalah probabilitas dimana peralatan mengalami kegagalan atau kerusakan dalam suatu interval waktu berikutnya yang diberikan dan diketahui kondisinya baik pada awal interval, sehingga dianggap sebagai suatu probabilitas konsional. Notasinya adalah $\lambda(t)$ atau R(t)

2.8 Distribusi Kerusakan

Pendekatan yang digunakan untuk mencari kecocokan antara distribusi keandalan dengan data kerusakan terbagi 2 cara yaitu:

1. Menurunkan distribusi kehandalan secara empiris langsung dari data kerusakan. Dengan kata lain kita menemukan model matematis untuk kehandalan, laju kerusakan, dan rata-rata waktu kerusakan secara langsung berdasarkan pada data kerusakan. Cara ini disebut nonparametic method. Hal ini di karenakan metode

ini tidak dibutuhkan spesifikasi dari distribusi secara teoritis tertentu dan selain itu juga tidak membutuhkan penaksiran dari parameter untuk distribusi.

2. Mengidentifikasi sebuah distribusi kehandalan secara teoritis, menarik parameter, dan kemudian melakukan uji kesesuaian distribusi. Metode ini akan menggunakan distribusi teoritis dengan tingkat kecocokan tertinggi dan data kerusakan sebagai model distribusi reliabilitas yang digunakan untuk menghitung kehandalan, laju kerusakan dan rata-rata waktu kerusakan.

Berdasarkan kenyataan bahwa hampir semua data kerusakan umum memiliki kecocokan yang tinggi terhadap suatu distribusi tertentu, maka cara kedua umumnya lebih disukai dari pada cara pertama.Cara kedua juga memiliki bebrapa keunggulan Ebeling, (1997,p358), yaitu:

1. Model Empiris tidak menyediakan informasi di luar range dari data sampel, sedangkan dalam model distribusi teoritis, ekstrapolasi melebihi range data sampel adalah mungkin untuk dilakukan.
2. Yang ingin diprediksi adalah data secara keseluruhan, bukan hanya terbatas pada sampel saja karena sampel hanya merupakan sebagian kecil dari populasiyang diambil secara acak, sehingga model kerusakan tidak cukup, bila hanya dibentuk berdasarkan data sampel saja.
3. Distribusi teoritis dapat juga digunakan untuk menggambarkan berbagai laju kerusakan.
4. Ukuran sampel yang kecil menyediakan informasi yang sedikit mengenai proses kegagalan. Akan tetapi jika sampel konsisten terhadap distribusi teoritis, maka hasil prediksi yang lebih kuat dapat diperoleh.

5. Distribusi teoritis lebih mudah untuk digunakan dalam menganalisa proses kegagalan kompleks.

Terdapat 4 macam distribusi yang digunakan agar dapat mengetahui pola data yang terbentuk. Distribusi tersebut antara lain : distribusi *Weibull*, *Exponential*, normal dan lognormal.

Distribusi kerusakan merupakan ekspresi matematis usia dan pola kerusakan mesin atau peralatan. Karakteristik kerusakan setiap peralatan/mesin akan mempengaruhi kedekatan yang digunakan dalam menguji kesesuaian dan menghitung parameter fungsi distribusi kerusakan. Pada umumnya, karakteristik dari kerusakan stiap mesin tidaklah sama terutama jika dioperasikan dalam kondisi lingkungan yang berbeda. Suatu peralatan maupun mesin yang memiliki karakteristik dan diperasikan dalam kondisi yang sama juga mungkin akan memberikan nilai selang waktu antar kerusakan yang berlainan. Suatu kondisi yang berhubungan dengan kebijakan perawatan seperti kebijakan perawatan pencegahan (*preventive*) memerlukan informasi tentang selang waktu suatu mesin akan mengalami kerusakan lagi. Biasanya saat terjadi perubahan kondisi mesin dari kondisi bagus menjadi rusak lagi, tidak dapat diketahui dengan pasti. Akan tetapi, dapat diketahui probabilitas terjadinya perubahan tersebut.

2.8.1 Distribusi *Weibull*

Menurut Ebeling(1997,p59), Distribusi *Weibull* merupakan distribusi yang paling banyak digunakan untuk waktu kerusakan karena distribusi ini baik digunakan unutk laju kerusakan yang meningkat maupun laju kerusakan yang menurun.

Terdapat dua parameter yang digunakan dalam distribusi ini yaitu θ yang disebut parameter skala (scale parameter) dan β yang disebut dengan parameter bentuk (shape parameter).

Menurut Ebeling(1997,p59), Fungsi reliability yang terdapat dalam distribusi *Weibull* yaitu :

$$\textit{Reability function}: R(t) = e^{\left(\frac{t}{\theta}\right)^{p}}$$

Dimana $\theta > 0, \beta > 0$, dant>0

Menurut Ebeling, dalam distribusi *weibull* yang menetukan tingkat kerusakan dan pola data yang terbentuk adalah parameter β. Nilai-nilai β yang menunjukkan laju kerusakan terdapat pada table berikut:

Tabel 2.1 Nilai parameter bentuk (β) Distirbusi weibull

Nilai	**Laju Kerusakan**
$0<\beta<1$	Laju kerusakan menurun (decreasing failure rate) DFR
$\beta=1$	Laju kerusakan konstan (constant failure rate) CFR Distribusi Exponential
$1<\beta<2$	Laju kerusakan meningkat (increasing failure rate) IFR kurva berbentuk konkaf
$\beta=2$	laju kerusakan linier (linier failure rate) LFR distribusi Rayleigh
$\beta>2$	Laju kerusakan meningkat (incrkaeasing failure rate) IFR kurva berbentuk konveks
$3\leq\beta\leq4$	Laju kerusakan meningkat (incrkaeasing failure rate) IFR kurva berbentuk simetris Distribusi normal

Jika parameter β mempengaruhi laju kerusakan maka parameter θ mempengaruhi nilai tengah dari pola data.

2.8.2 Distribusi Exponensial

Distibusi eksponential digunakan untuk menghitung keandalan dari distribusi kerusakan yang memiliki laju kerusakan konstan. Distribusi ini mempunyai laju kerusakan yang tetap terhadap waktu, dengan kata lian probabilitas terjadinya kerusakan tidak tergantung pada umur alat. Distribusi ini merupakan distribusi yang paling mudah untuk dianalisa.

Parameter yang digunakan dalam distribusi Exponential adalah λ, yang menunjukkan rata-rata kedatangan kerusakan yang terjadi.

Fungsi *reliability* yang terdapat dalam distribusi *eksponential* yaitu (Ebeling, 1997, p41):

$$\text{Reliability Function: } R(t) = e^{-\lambda t}$$

Dimana $t > 0$, $\lambda > 0$

2.8.3 Distribusi *Normal*

Distribusi Normal cocok untuk digunakan dalam memodelkan fenomena keausan (kelelahan) atau kondisi *wear out* dari suatu item. Sebenarnya distribusi ini bukanlah distribusi reliabilitas murni karena variable acaknya memiliki range antara minus tak hingga sampai plus tak hingga. Akan tetapi, karena hampir untuk semua nilai μ dan σ, peluang untuk variabel acak yang memiliki nilai negative dapat diabaikan, maka distribusi normal dapat digunakan sebagai pendekatan yang baik untuk proses kegagalan.

Parameter yang digunakan adalah μ (nilai tengah) dan σ (standar deviasi). Karena hubungannya dengan distribusi lognormal, distribusi ini dapat juga digunakan untuk menganalisa probabilitas lognormal.

Fungsi *reliability* yang terdapat dalam distribusi Normal yaitu (Ebeling, 1997,p69):

$$Reliability\, R(t) = \phi\left(\frac{t-\mu}{\sigma}\right)$$

Dimana $\mu > 0, \sigma > 0$ dan t>0

2.8.4 Distribusi *Lognormal*

Menurut pendapat Ebeling (1997,p74),Distribusi lognormal memiliki parameter bentuk (shape pharameter= s), dan Parameter lokasi (*location* parameter = t_{med}) yang merupakan nilai tengah dari waktu kerusakan. Distribusi ini dimengerti hanya untuk t positif dan lebih sesuai daripada distribusi normal dalam hal kerusakan. Seperti halnya distribusi weibull, lognormal ini dapat mempunyai berbagai bentuk. Seringkali dijumpai bahwa data yang sesuai dengan distribusi Weibull sesuai pula dengan distribusi Lognormal.

Fungsi reliability yang terdapat pada distribusi lognormal yaitu:

$$Reliability\ function : R(t)=1-\phi\left(\frac{1}{s}\ln\frac{t}{t_{med}}\right)$$

Dimana $s > 0, t_{med} > 0$ dan $t > 0$

Fungsi distribusi *Lognormal* dinyatakan dengan:

$$f(t) = \frac{1}{\sqrt{2\pi}st}\exp\left[-\frac{1}{2s^2}\left(\ln\frac{t}{t_{med}}\right)^2\right] \quad \text{dengan } t \geq 0$$

2.9 Indetifikasi Kerusakan Distribusi

Menurut Ebeling (1997,p367),Pengidentifikasian distribusi dapat dilakukan dalam 2 tahap, yaitu Index Of Fit (r) dan *Goodness of Fit Test*.

2.9.1 *Index Of Fit*

Dengan metode *Least Square Curve Fitting*, dicari nilai *index of fit* (r) atau korelasi antara t; (atau ln t;) sebagai x dengan y yang merupakan fungsi dari distribusi teoritis terhadap x. Kemudian distribusi yang terpilih adalah distribusi yang nilai *index of fit* (r) terbesar distribusi dengan nilai r yang terbesar akan dipilih untuk diuji dengan menggunakan *Goodness of Fit Test*.

Rumus umum yang terdapat dalam metode *Least Square Curve Fitting* adalah:

$$F(t_i) = \frac{i - 0.3}{n + 0.4}$$

Dimana: i = data waktu ke-t

n = jumlah data kerusakan

$$Index\ of\ Fit\,(r) = \frac{n\sum x_i y_i - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right)}{\sqrt{\left[n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2\right]\left[n\sum_{i=1}^{n} y_i^2 - \left(\sum_{i=1}^{n} y_i\right)^2\right]}}$$

Dimana n = Jumlah kerusakan yang terjadi

Gradien:

1. Untuk Distribusi *Weibull*, Normal, Lognormal

$$b = \frac{n\sum_{i=1}^{n} x_i y_i - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right)}{n\sum_{i=1}^{n} x_i^2 \left(\sum_{i=1}^{n} x_i\right)}$$

2. Untuk Distribusi *Exponential*

$$b = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2}$$

$$\textit{Intersep}:\ a = \bar{y} - b\bar{x}$$

Dalam menetukan distribusi yang hendak digunakan untuk menghitung MTTF, MTTR dan Reliability, proses yang harus dilakukan adalah mencari nilai r untuk masing-masing distribusi sehingga didapatkan nilai r terbesar yang kemudian akan diuji lagi menurut hipotesa distribusinya.

sumber: Ebeling (1997, p367).

Dibawah ini adalah rumus-rumus mencari nilai r, yaitu:

1. Distribusi *Weibull*

$$r_{weibull} = \frac{n\sum_{i=1}^{n} x_i y_i - \left(\sum_{1=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right)}{\sqrt{\left[n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2\right]\left[n\sum_{i=n}^{n} y_i^2 - \left(\sum_{i=1}^{n} y_i\right)^2\right]}}$$

Keterangan:

$x_i = \ln(t_i)$

$y_i = \ln\left[\ln\left(\frac{1}{1-F(t_i)}\right)\right]$

t_i adalah data ke-i

Parameter : $\beta = b$ dan $\theta = e^{-\left(\frac{a}{b}\right)}$

2. Distribusi Eksponential

$$r_{eksponential} = \frac{n\sum_{i=1}^{n} x_i y_i - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right)}{\sqrt{\left[n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2\right]\left[n\sum_{i=1}^{n} y_i^2 - \left(\sum_{i=1}^{n} y_i\right)^2\right]}}$$

Keterangan:

$x_i = t_i$

$y_i = \ln\left[\ln\left(\frac{1}{1-F(t_i)}\right)\right]$

ti adalah data ke-i

parameter: $\lambda = b$

3. Distribusi normal

$$r_{normal} = \frac{n\sum_{i=1}^{n} x_i z_i - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} z_i\right)}{\sqrt{\left[n\sum_{i=1}^{n} x_1^2 - \left(\sum_{i=1}^{n} x_i\right)^2\right]\left[n\sum_{i=1}^{n} z_i^2 - \left(\sum_{i=1}^{n} z_i\right)^2\right]}}$$

Keterangan:

$x_i = t_i$

$z_i = \phi^{-1}[F(t_i)]$ (diperoleh dari tabel $\phi(z)$ dilampiri

ti adalah data ke-i

parameter : $\sigma = \frac{1}{b}$ dan $\mu = -\left(\frac{a}{b}\right)$

4. Distribusi Lognormal

$$r_{lognormal} = \frac{n\sum_{i=1}^{n} x_i z_i - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} z_i\right)}{\sqrt{\left[n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2\right]\left[n\sum_{i=1}^{n} z_i^2 - \left(\sum_{i=1}^{n} z_i\right)^2\right]}}$$

Keterangan:

$x_i = \ln(ti)$

$z_i = \Phi^{-1}[F(t_i)]$ diporelah dari tabel $\phi(z)$ dilampiri

t_i adalah data ke-i

Parameter : $s = \frac{1}{b}$ dan $t_{med} = e^{-sa}$

2.9.2 Uji kebaikan suai (goodness of Fit)

Menurut Ebeling (1997,399), Tahap selanjutnya setelah dilakukan perhitungan *index of fit* adalah pengujian *goodness of fit* untuk nilai *index of fit* yang terbesar. Dilakukan dengan membandingkan antara hipotesis nol (Ho) dan hipotesis alternative (H_1). Ho menyatakan bahwa waktu kerusakan tidak berasal dari distribusi tertentu.

Pengujian ini merupakan perhitungan statistik yang didasarkan pada sampel waktu kerusakan. Statistik ini kemudian dibandingkan dengan nilai kritik yang diperoleh dari table. Secara umum, apabila pengujian statistik ini berada di luar nilai kritik, maka Ho diterima. Sebaliknya, maka H1 yang diterima. Ada 2 jenis goodness-of-fit test, yaitu:

1. Uji umum (general tests)

   Digunakan untuk menguji beberapa distribusi.

   Terdiri dari: uji Chi-Square.

2. Uji khusus (spesific tests)

Digunakan hanya untuk menguji 1 jenis distribusi. Nilai kritis tergantung dari derajat kepercayaan (α) pengujian sampel yang ada. Terdiri dari :

a. Mann's Test untuk distribusi Weibull

b. Barlett's Test untuk distribusi Exponential,

c. Kolmogorov-Smirnov Test untuk Distribusi Normal dan Lognormal.

Ketika suatu distribusi data waktu kerusakan telah diasumsikan sebelumnya, dimana asumsi tersebut bisa ditentukan melalui bentuk umum atau bentuk dari plot data dalam suatu grafik (bisa dalam bentuk minitab). Validasi dari asumsi distribusi dapat diketahui melalui suatu pengujian. Hasil pengujian tersebut mempunyai dua kemungkinan, yaitu asumsi bahwa distribusi bisa diterima atau ditolak.

2.9.2.1 Mann's Test untuk pengujian Distribusi weibull

Menurut Ebeling (1997,p400-401) hipotesa untuk melakukan uji ini adalah:

H_0 : Data kerusakan berdistribusi Weibull

H_1 : Data kerusakan tidak berdistribusi Weibull

Uji statistikanya adalah:

$$M = \frac{k_1 \sum_{i=k1+1}^{r-1} \left[\frac{(\ln t_{i+1} - \ln t_i)}{M_i} \right]}{k_2 \sum_{i=1}^{k1} \left[\frac{(\ln t_{i+1} - \ln t_i)}{M_i} \right]}$$

$$M_i = Z_{i+1} - Z_i$$

$$Z_i = \ln\left[-\ln\left(1 - \frac{i - 0.5}{n + 0.25} \right) \right]$$

Keterangan:

t_i = data waktu kerusakan yang ke-i

X_i = ln(ti)

n = banyaknya data

M_i = nilai pendekatan Mann untuk data ke-i

$M_{\alpha,k1,k2}$ = nilai Mtabel untuk distribusi weibull (lihat distirbusi F), dengan $v_1 = k_1$ dan $v_2 = k_2$

k_1 = r/2

k_2 = (r-1)/2 (bilangan bulat terbesar yang lebih kecil dari (r/2)

H_0 diterima jika M_{hitung} jatuh dibawah wilayah kritis:

$$M_{hitung} < M_{tabel(\alpha,k1,k2)}$$

2.9.2.2 Barlett's Test untuk pengujian Distribusi Exponential

Menurut Ebeling (1997, p399) hipotesa untuk melakukan uji ini adalah:

H0 : Data kerusakan distribusi Eksponential

H1 : Data kerusakan tidak berdistribusi Eksponential

Uji statistiknya adalah:

$$B = \frac{2r\left[\ln\left(\frac{1}{R}\right)\sum_{i=1}^{r} t_i - \left(\frac{1}{R}\right)\sum_{i=1}^{r} \ln t_i\right]}{1+\frac{(r+1)}{6r}}$$

Keterangan:

t_i = data waktu kerusakan ke-i

r = jumlah kerusakan

B = nilai uji statistik untuk uji Barlett's Test

Jika $X^2_{\frac{1-\alpha}{2},r-1} < B < X^2_{\frac{\alpha}{2},r-1}$ maka H_0 diterima

2.9.2.3 *Kolmogorov-Smirnov* untuk pengujian Distribusi Normal maupun lognormal

Menurut Ebeling 1997,(p402-404) Hipotesa untuk melakukan uji ini adalah:

H_0 : Data kerusakan berdistribusi Normal atau lognormal

H_1: Data kerusakan tidak berdistribusi Normal dan lognormal

Uji statistiknya adalah : Dn = max $\{D_1, D_2\}$

Dimana,

$$D_1 = \max_{1<i<n} \left\{ \Phi\left(\frac{t_i - \bar{t}}{s} \right) - \frac{i-1}{n} \right\} \qquad D_2 = \max_{1<i<n} \left\{ \frac{i}{n} - \Phi\left(\frac{t_i - \bar{t}}{s} \right) \right\}$$

$$\bar{t} = \sum_{i=1}^{n} \frac{\ln t_i}{n} \quad \text{dan } s^2 = \frac{\sum_{i=1}^{n} \left(\ln t_i - \bar{t} \right)^2}{n-1}$$

Keterangan:

ti= data waktu kerusakan ke-i

$\bar{t}$ = rata-rata data waktu kerusakan

s = standar deviasi

n = banyaknya kerusakan

jika , $D_n < D_{kritis}$ maka terima h_0

b. Distibusi Eksponensial

$$MTTF = \frac{1}{\lambda}$$

c. Distibusi Normal

$$MTTF = \mu$$

d. Distribusi Lognormal

$$MTTF = t_{med}.\, e^{\frac{s^2}{2}}$$

2.11 Nilai tengah dari Distribusi Perbaikan (*Mean time to Repaire*)

Dalam menghitung rata-rata atau penentuan nilai tengah dari fungsi probabilitas untuk waktu perbaikkan, sangatlah perlu diperhatikan distribusi data perbaikkannya. Penentuan untuk menguji ini dilakukan dengan cara yang sama dengan yang sudah dijelskan sebelumnya. Menurut Ebeling (1997,p192), MTTR diperoleh dengan rumus:

$$MTTR = \int_0^\infty th(t)dt = \int_0^\infty \left(1 - H(t)\right)dt \quad \text{dimana,}$$

h(t) = fungsi kepadatan peluang untuk data waktu perbaikan (TTR)

H(t) = fungsi distribusi kumulatif untuk data waktu perbaikan (TTR)

Perhitungan nilai MTTR unuk masing-masing distribusi, yaitu

a. Distribusi Weibull (Ebeling (1997,p59)

$$MTTF = \theta \cdot \Gamma\left(1+\frac{1}{\beta}\right)$$

Nilai $\Gamma\left(1+\frac{1}{\beta}\right)$ didapat dari:

$\Gamma(x)$ = tabel fungsi gamma (lihat lampiran)

b. Distibusi Eksponensial

$$MTTF = \frac{1}{\lambda}$$

c. Distibusi Normal

$$MTTF = \mu$$

d. Distribusi Lognormal

$$MTTF = t_{med}.\, e^{\frac{s^2}{2}}$$

2.12 Reliabilitas dengan *Preventive Maintenance* dan Tanpa *Preventive Maintenance*

Peningkatan kehandalan dapat ditempuh dengan cara perawatan pencegahan. Menurut Ebeling (1997, p204),Perawatan pencegahan dapat mengurangi pengaruh wear out dan menunjukkan hasil yang signifikan terhadap umur mesin. Model kehandalan berikut ini mengasumsikan sistem kembali ke kondisi baru setelah mengalami perawatan pencegahan. Keandalan pada saat t dinyatakan sebagai berikut :

$R_m(t) = R(t)$ untuk $0 \leq t < T$

$R_m(t) = R(T)^n.R(t-T)$ untuk $T \leq t < 2T$

Secara umum persamaanya adalah :

$R_m(t) = R(T)^n.R(t-nT)$ untuk $nT \leq t < (n+1)T$ dan n=1,2,3,..dst

Dimana :

T adalah selang waktu *preventive maintenance*

T adalah waktu operasional mesin

n merupakan jumlah perawatan

Rm(t) adalah *reliability* dengan *Preventive Maintenance*

R(T)n adalah probabilitas kehandalan hingga n selang waktu perawatan

R(t-nT) adalah probabilitas kehandalan untukwaktu t-nTdari tindakan *preventive* yang terakhir.

Untuk komponen yang memiliki laju kerusakan yang konstan : $R(t) = e^{-\lambda t}$ maka dapat menggunakan persamaan dibawah ini :

$R_m(t) = (e^{-\lambda t})^n e^{-\lambda t (t-nT)}$

$R_m(t) = e^{-\lambda nt}.e^{-\lambda t}.e^{\lambda nt}$

$R_m(t) = e^{-\lambda t}$

$R_m(t) = R(t)$

Berdasarkan rumus di atas, ini membuktikan bahwa jika pola kerusakan berdistribusi eksponensial atau memiliki laju kerusakan konstan, bila dilakukan *preventive maintenance* tidak akan memberikan dampak apapun. Hal ini disebabkan karena tidak terjadinya peningkatan reliability seperti yang diharapkan, karena $R_m(t) = R(t)$.

Untuk komponen yang memiliki distribusi lognormal maka dapat menggunakan persamaan dibawah ini :

$$R(T) = 1-\Phi\left(\frac{1}{s}\ln\frac{t}{t_{med}}\right)$$

$$R(T)^n = \left[1-\Phi\left(\frac{1}{s}\ln\frac{t}{t_{med}}\right)\right]^n$$

$$R(t-nT) = 1-\Phi\left(\frac{1}{s}\ln\frac{t-nt}{t_{med}}\right)$$

Untuk komponen yang memiliki distribusi normal maka dapat menggunakan persamaan dibawah ini:

$$R\,(T) = 1-\Phi\left(\frac{t-\mu}{\sigma}\right)$$

$$R\,(T)^n = 1-\Phi\left(\frac{t-\mu}{\sigma}\right)^n$$

$$R(t\text{-}nT) = 1\text{-}\Phi\left(\frac{(t-nT)-\mu}{\sigma}\right)$$

Sedangkan untuk komponen yang memiliki distribusi weibull maka dapat menggunakan persamaan dibawah ini :

$$Rm(t) = \exp\left[-n\left(\frac{T}{\theta}\right)^{\beta}\right]\exp\left[-\left(\frac{t-nT}{\theta}\right)^{\beta}\right] \text{ untuk } nT \leq t<(n+1)T$$

Untuk masing-masing distribusi yang ingin diukur peningkatan *reliability*-nya, dapat menggunakan persamaan sebagai berikut:

$$\text{Peningkatan } \textit{Reliability} = \frac{Rm(t)-R(t)}{R(t)} x100\%$$

2.13 Sistem Informasi

2.13.1 Pengertian Sistem

Sistem Menurut Mathiasen et al. (2000, p9), sistem adalah kumpulan dari komponen yang mengimplemntasikan persyaratan *model, function, interface*. Sedangkan McLeod (2001,p11) definisi sistem adalah sekelompok elemen yang salaing terintegrasi dengan maksud yang sama untuk mencapai suatu tujuan. Menurut O'Brien (2004,p8) adalah suatu kelompok dari elemen-elemen yang saling berhubungan dan berinteraksi satu sama lain dan menciptakan suatu keutuhan yang utuh. Elemen-elemen ini bekerja sama untuk mencapai suatu tujuan bersama dengan menerima *input* dan memproduksi *output* dalam proses yang terorganisir.

Sistem memiliki tiga komponen dasar dan 2 komponen tambahan yang saling berinterkasi, yaitu:

- *Input* (masukan)

  Merupakan sekumpulan data baik dari dalam organisasi maupun dari luar organisasi yang akan digunakan dalam proses sistem informasi.

- *Process* (Proses)

  Merupakan proses transformasi yang mengubah *input* menjadi *output*. Contohnya mencakup proses manufaktur, perhitungan matematis dan lain sebagainya..

- Output (keluaran)

  Merupakan elemen yang telah melalui proses transformasi. Contohnya mencakup jasa, produk dan informasi.

- Feedback (Umpan balik):

  Merupakan output yang di kembalikan kepada orang-orang dalam organisasi untuk membantu mengevaluasi input.

- Subsistem :

  Merupakan sebagian dari sistem yang mempunyai fungsi khusus. Masing-masing subsistem ini memiliki komponen input, proses, output, dan feedback.

2.13.2 Pengertian Data dan Informasi

Data terdiri dari fakta-fakta dan angka-angka yang relatif tidak berarti bagi pemakai. Saat data diproses, ia dapat diubah menjadi informasi. Sedangkan pengertian infomasi menurut McLeod (2001, p15) adalah data yang telah diproses, atau data yang memiliki arti dan siap untuk dipakai. Informasi juga bisa dapat diartikan sebagai data

yang diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi yang menerimanya.

Informasi sangat dibutuhkan karena informasi merupakan dasar dalam mengambil keputusan dalam perusahaan. Pengolahan informasi adalah salah satu elemen kunci dalam sistem konseptual. Pengolahan informasi dapat meliputi elemen-elemen komputer, elemen-elemen non-komputer, atau kombiansi keduanya.

2.13.3 Pengertian Sistem Informasi

Berdasarkan pengertian yang diberikan Whitten et al (2004,p12), Sistem informasi adalah susunan dari manusia, data, berbagai proses, dan teknologi informasi yang saling berinteraksi untuk mengumpulkan, mengolah, menyimpan, dan menyediakan output informasi yang dibutuhkan untuk mendukung sebuah oraganisasi. Menurut O'Brien (2004, p8), mengungkapkan bahwa sistem informasi bergantung pada kerangka kerjanya yang terdiri dari manusia, software, data, jaringan dan hardware. Sedangkan Pengertian Sistem Informasi Menurut McLeod (2001,p4), adalah suatu kombinasi yang terorganisasi dari manusia, peranti lunak, perangkata keras, jaringan komunikasi, dan sumber daya data yang mengumpulkan, mentransformasikan, serta menyebarkan informasi di dalam sebuah organisasi.

Menurut pendapat ahli lainnya, infomasi adalah data yang telah diproses menjadi bentuk yang memiliki arti bagi penerima dan dapat berupa fakta, suatu nilai yang bermanfaat atau prospek keputusan. Jadi ada suatu proses transformasi data menjadi suatu informasi (input-proses-output). Dari definisi yang disebutkan, informasi dapat disimpulkan sebagai data yang telah diolah yang mempunyai arti dalam pengambilan keputusan bagi pihak yang bersangkutan..

Adapun komponen-komponen dari sistem informasi adalah metode kerja (*work practices*), informasi (*information*), manusia (people), teknologi informasi (information technologies).

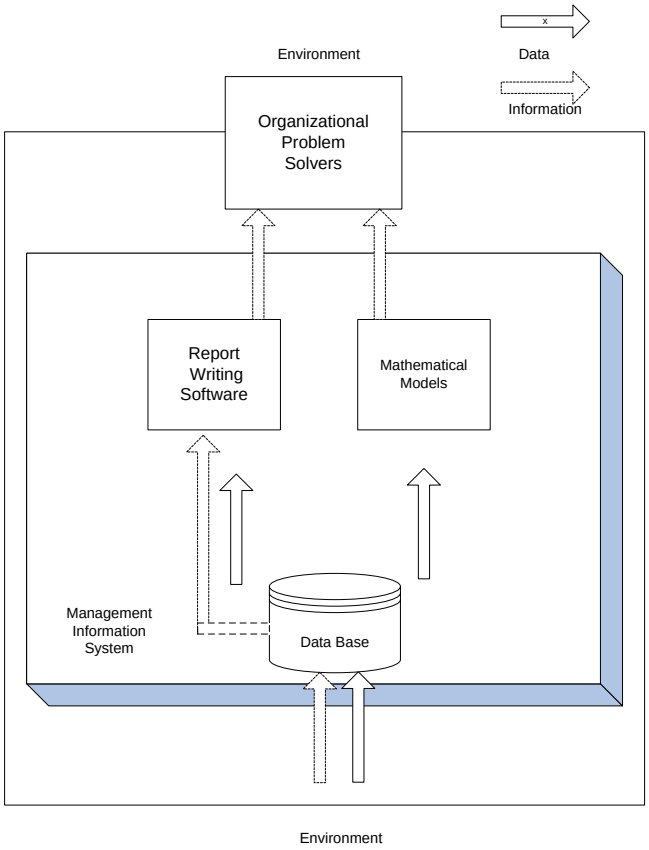
Alasan diperlukannya sistem informasi dalam suatu organisasi ialah sebagai berikut:

1. Untuk mensinkronisasi aktivitas-aktivitas dalm organisasi sehingga semua sumber daya dapat dimanfaatkan seefektif mungkin.
2. Perkembangan teknologi yang semakin kompleks.
3. Semakin pendeknya waktu untuk pengambilan keputusan.
4. Lingkungan bisnis yang semakin kompetitif.
5. Pengaruh kondisi ekonomi internasional.
6. Meningkatnya kompleksitas dari aktivitas bisnis / organisasi.

2.13.4 Sistem Informasi Manajemen

McLeod (2001,p239), mendefinisikan Sistem Manajemen Informasi sebagai sistem berbasis komputer yang menyediakan infrmasi bagi pengguna yang memiliki kepentingan yang sama yaitu pengambilan keputusan untuk menyelesaikan masalah yang dihadapi oleh organisasi / Perusahaan. Output dari SIM adalah berupa laporan periodik, laporan khusus, dan perhitungan matematis.

Model SIM menjelaskan bahwa data dan informasi didapat dari lingkungan. Database digunakan oleh software untuk menghasilkan laporan dan model matematis juga digunakan untuk menghasilkan perhitungan yang akan digunakan oleh pengambil keputusan dalam organisasi baik berupa manager maupun non manager. Aliran data dan informasi dibedakan untuk menunjukkan yang mana yang merupakan input dan output dari komponen sistem.



Sumber: Mcleod(2001,p240)

:Gambar 2.1 Model SIM

2.14 Analisis Perancangan sistem informasi Berorientasi Objek

Menurut Whitten et. Al. (2004, p31), *Object Oriented Analysis and Design* (OOAD) merupakan suatu kumpulan alat dan teknik untuk membangun suatu teknik yang menggunakan teknologi objek untuk membangun sebuah sistem dan perangkat lunaknya. Sedangkan yang dimaksud dengan teknologi objek itu sendiri adalah teknologi perangkat lunak yang mendefinisikan sebuah sistem dalam istilah objek yang menggabungkan data dan perilakunya.

Pendekatan yang berorientasi objek ini menggunakan objek dan class sebagai konsepnya . Pengertian objek yaitu suatu entitas yang memiliki indetitas, status, dan perilaku (Mathiassen et al., 2000,p4). Dengan menggunakan objek maka sistem dapat

mengatur apa saja yang dapat dilakukan terhadap entitas tersebut. Sedangkan pengertian class adalah deskripsi dari kumpulan objek yang memiliki struktur , pola perilaku,atribut yang sama. Contoh dari class misalnya sekumpulan entitas pelanggan yang berbeda menjadi sebuah class pembeli, masing-masing objek didalamnya memiliki identitas nama dan alamat yang pastinya bisa berbeda.

OOAD dibangun dari empat prinsip umum untuk analisis dan desain, yaitu :

a. Desain isi sebuah sistem;
b. Memperjelas pertimbangan arsitektur;
c. Menggunakan kembali pola yang menjelaskan ide-ide desain yang baik;
d. Mengadaptasikan metode ke setiap pengembangan**.**

Pembuatan skripsi ini menggunakan *Unified Modelling Languange* (UML) yang merupakan salah satu konsep *Object Oriented Analysis and Design*.

2.15 Unified Modelling Language (UML)

Menurut Booch (1999,p1) *Unified Modeling Language* (UML) adalah penerus dari *object-oriented analysis and design* (OOA&D) *methods* yang muncul pada akhir ’80 an dan awal ’90 an. UML secara langsung menggabungkan *methods* dari Booch, Rumbaugh (OMT), dan Jacobson, tetapi menjadi lebih berkembang daripada itu semua. UML berkembang melalui sebuah proses standarisasi yang dilakukan oleh OMG (Object Management Group) dan sekarang memakai standar dari OMG.

UML disebut sebagai bahasa untuk permodelan, bukan sebuah *method*. Hampir semua *methods* mengandung, paling tidak dalam beberapa prinsip, dua dari sebuah bahasa permodelan dan sebuah proses. Bahasa permodelan tersebut (terutama yang

berbasis grafis) adalah sebuah notasi yang menggunakan methods untuk mengekspresikan sebuah rancangan. Proses tersebut adalah tuntunan yang mereka lakukan dalam setiap langkah untuk merancang sesuatu.

2.15.1 Pengertian UML

Unified Modelling Language (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah system (http://www.ittelkom.ac.id/library).

2.15.2 Sejarah UML

UML adalah sebuah bahasa yag berdasrkan grafik/gambar untuk menvisualisasikan, menspesifikasikan, membangun dan pendokumentasian dari sebuah sistem pengembangan software berbasis OO(Object-Oriented). UML sendiri juga memberikan standar penulisan sebuah sistem blue print, yang meliputi konsep bisnis proses, penulisan kelas-kelas dalam bahasa program yang spesifik, skema database, dan komponen-komponen yang diperlukan dalam sistem software(http://omg.org).

Pendekatan analisa dan perancangan dengan menggunakan model OO mulai diperkenalkan sekitar pertengahan 1970 hingga akhir 1980 di karenakan pada saat itu aplikasi software sudah meningkat dan mulai komplek. Jumlah yang menggunakan metoda OO mulai diujicobakan diaplikasikan antara 1989 hingga 1994, seperti halnya oleh Grady Booch dari Rational Software Co., dikenal dengan OOSE (*Object-Oriented Software Engineering*), serta James Rumbaugh dari general Electric, dikenal dengan OMT (*Object Modelling Technique*).

Kelemahan saat itu disadari oleh Booch maupun Rumbaugh adalah tidak adanya standar penggunaan model yang berbasis OO, ketika mereka bertemu ditemani rekan lainnya Ivar Jacobson dari Objectory mulai mendiskusikan untuk mengadopsi masing-masing pendekatan metoda OO untuk membuat suatu model bahasa yang uniform/seragam yang disebut UML (Unified Modeling Language) dan dapat diguakan seluruh dunia.

Secara resmi bahasa UML dimulai pada bulan oktober 1994, ketika Rumbaugh bergabung booch untuk membuat sebuah project pendekatan metoda yang uniform/seragam dari masing-masing metoda mereka. Saat itu baru dikembangkan draft metoda UML version 0.8 dan diselesaikan serta di release pada bulan oktober 1995. Bersamaan dengan saat itu, Jacobson bergabung dan UML tersebut diperkaya ruang lingkupnya dengan metoda OOSE sehingga muncul release 0.9 pada buan juni 1996. Hingga saat ini sejak juni 1998 UML version 1.3 telah diperkaya dan direspons oleh OMG (Object Management Group), Anderson Consulting, Ericson, Platinum technology, Object Time Limited, dll serta dipelihara oleh OMG yang dipimpin Cris Kobryn.

UML adalah standar dunia yang dibuat oleh Object Management Group (OMG), sebuah badan yang bertugas mengeluarkan standar-standar teknologi object-oriented dan software component. (sumber: http://www.scribd.com/doc/2584053/Pengenalan-UML)

2.15.3 Konsepsi Dasar UML

UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat 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 C++, Java, C# atau VB.NET. (sumber: http://www.ittelkom.ac.id/library/)

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

Untuk menguasai UML, ada dua hal yang harus kita perhatikan:

1. Menguasai pembuatan diagram UML
2. Menguasai langkah-langkah dalam analisa dan pengembangan dengan UML

2.15.4 Diagram UML

Menurut Whitten (2004,p441-442), UML menawarkan 9 diagram yang di kelompokkan menjadi lima perspektif yang berbeda untuk memodelkan *system* informasi.

Kelompok 1: *Use-case Model Diagram*

Diagram *use-case* secara grafis menggambarkan interaksi antara sistem, sistem eksternal dan pengguna. Dengan kata lain, secara grafis mendeskripsikan siapa yang akan menggunakan dan dalam cara apa pengguna mengharapkan interaksi dengan sistem itu

Kelompok 2: Diagram Struktur *Statis*

UML memberikan dua diagram untuk struktur static model dari sistem informasi. Dua diagram yaitu

1. *Class Diagrams*
2. *Object Diagram*

Kelompok 3: Diagram interaksi

Model diagram interaksi terdiri dari suatu set dari objek, hubungan dan pesan yang dikirimkan antara mereka. Terdiri dari diagram:

1. *Sequence diagrams*
2. *Collaboration diagrams*

Kelompok 4: *State Diagrams*

State diagram juga merupakan model *dynamic behaviour* dari suatu system. UML memiliki suatu diagram untuk memodelkan behaviour yang kompleks dari suatu particulary objek dan suatu diagram untuk memodelkan tingkah laku dari suatu use-case atau methods. Diagram terdiri dari:

1. *Statechart diagrams*
2. *Activity diagrams*

Kelompok 5 : *Implementation Diagrams*

Implementation diagram juga memberikan contoh struktur dari sistem informasi. Pada *implementation diagram* terdiri dari:

1. *Component Diagrams*
2. *Deployment Diagrams*

2.15.4.1 *Use Case Diagram*

Menurut schmuller(1999,p10) *Use case diagram* mendeskripsikan suatu tingkah laku sistem dari suatu sudut pandang pengguna. Sedangkan menurut Whitten (2004,p441), Diagram use-case secara grafis menggambarkan interaksi antara sistem, sistem eksternal dan pengguna. Dengan kata lain, secara grafis mendeskripsikan siapa yang akan menggunakan dan dalam cara apa pengguna mengharapkan interaksi dengan sistem itu.

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* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal. Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara 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* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.

Suatu u*se case* diagram memberikan gambaran dari sebuah sistem, seperti yang terlihat pada Gambar. Bagi para pembangun sistem, *use case* diagram merupakan teknik sekaligus alat yang berguna dalam menentukan kebutuhan sebuah sistem agar dapat digunakan oleh semua orang, bukan hanya oleh mereka yang mengerti bidang komputer.

Tugas *use case* adalah memodelkan sebuah sistem dari sudut pandang para penggunanya.

Use case diagram sangat penting untuk memodelkan perilaku dari sistem, subsistem atau kelas. Setiap *use case* menunjukkan satu set *use case*, *actor* dan hubungan diantaranya.

Actor : merupakan representasi dari siapa saja yang berinteraksi dengan *use case* dalam sebuah sistem.

Use case : merupakan deskripsi suatu set aksi yang dikerjakan oleh sistem.

Transition : merupakan penghubung *actor* dan *use case*.

2.15.4.2 Class Diagram

Menurut Schmuller(1999,p63), *Class diagram* menjelaskan suatu kumpulan dari class dan hubungan stuktural mereka. UML memiliki class diagram; yang merupakan pusat penjelasan dalam OOAD. adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). *Class* diagram menggambarkan struktur dan deskripsi kelas, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain.

Class memiliki tiga area pokok :

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

Atribut dan metoda dapat memiliki salah satu sifat berikut :

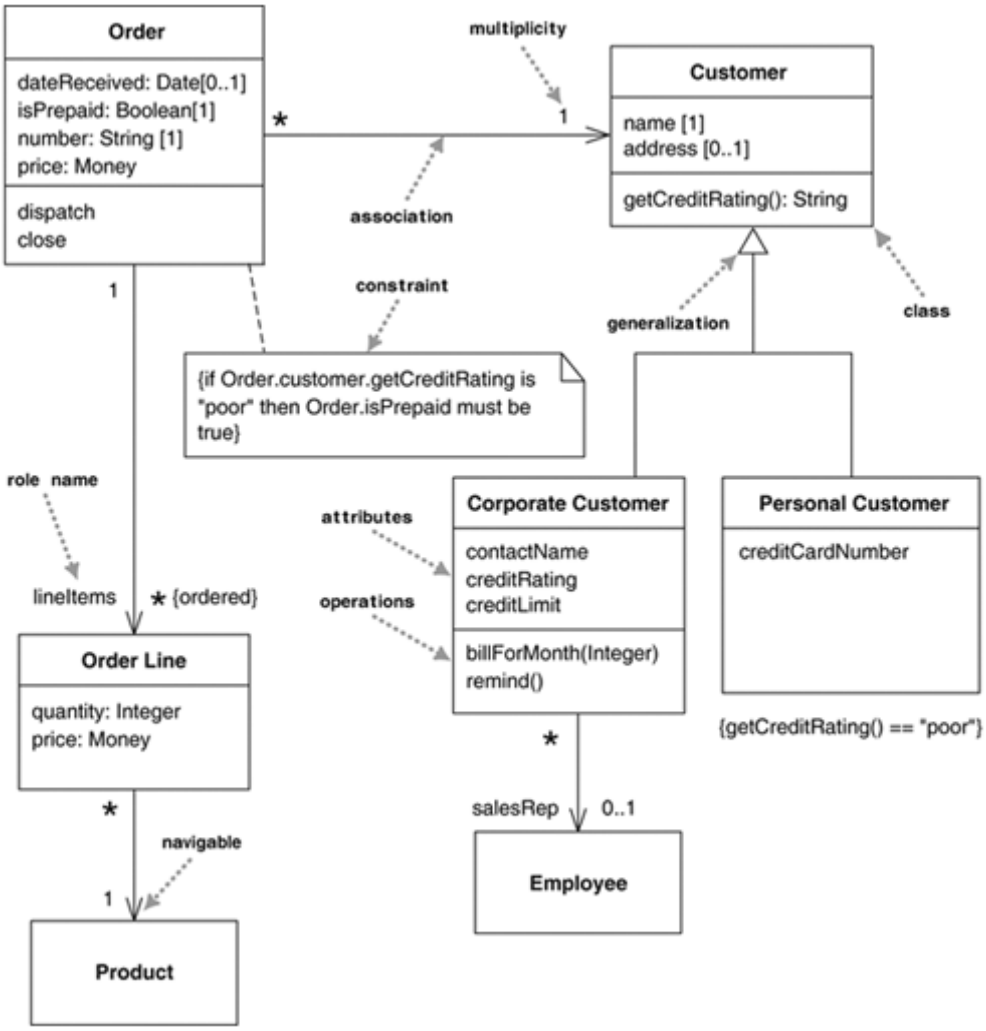
1. *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan
2. *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya
3. Public, dapat dipanggil oleh siapa saja

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 dikelompokkan menjadi paket. Kita juga dapat membuat diagram yang terdiri atas paket.

Hubungan Antar *Class*

1. Asosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui keberadaan *class* lain. Panah navigasi(***navigability)*** menunjukkan 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* yang akan dijelaskan kemudian.



Gambar2.3: Contoh Class Diagram

(Sumber Gambar: http://lecturer.ukdw.ac.id/willysr/pspl-ti/pengantar_uml.pdf)

Suatu *class* diagram menggambarkan struktur dari objek sistem, seperti yang terlihat pada Gambar. Hubungan antar *class* ada tiga yaitu :

(a) *Dependency*

Dependency merupakan hubungan yang menyatakan ketergantungan antara satu objek dengan objek yang lain. Apabila sebuah perubahan dilakukan

maka akan mengakibatkan perubahan bagi objek lain yang menggunakannya

(b) *Association*

Association adalah hubungan terstruktur antara objek yang satu dengan objek yang lain. *Aggregation* merupakan *association* antara dua *class* yang menunjukkan bahwa dua *class* tersebut berada pada level yang setara.

(c) *Generalization*

Generalization merupakan hubungan antara benda yang bersifat umum (yang disebut dengan *superclass* atau *parent*) dengan benda yang lebih spesifik (yang disebut dengan *subclass* atau *child*).

(Sumber http://lecturer.ukdw.ac.id/willysr/pspl-ti/pengantar_uml.pdf)

2.15.4.3 Object Diagram

Menurut Whitten et. al. (2004, p673), *Object* diagram merupakan suatu diagram yang menyerupai *class* diagram. Diagram ini menggambarkan contoh objek yang nyata dan memperlihatkan nilai atribut-atributnya saat ini. Diagram ini membantu pengembang sistem dengan memberi gambaran keadaan objek pada suatu saat. Diagram ini tidak digunakan sesering *class* diagram, tetapi jika *object* diagram ini digunakan, akan sangat membantu pengembang dalam memahami struktur dari sistem

2.15.4.4 Sequence Diagram

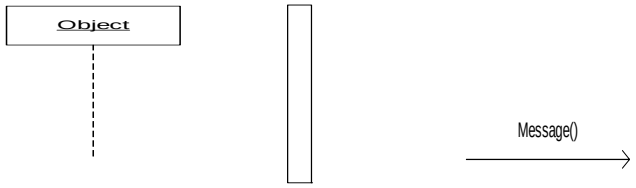
Menurut Booch(1999,245),*Sequence* diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence* diagram terdiri atas dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequenc*e

diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari 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 *lifeline* 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* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message*.

Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan *icon* khusus untuk objek *boundary*, *controller* dan *persistent entity*.

Simbol-simbol yang digunakan dalam *sequence* diagram yaitu (Booch, 1999, p247) :



Object Lifeline *Activation* *Message*

Gambar 2.4 Simbol-simbol *Sequence* Diagram

(Sumber: Booch , 1999,p247)

(a) *Object Lifeline* : Objek yang terlibat dalam operasi

(b) *Activation* : Lama waktu sebuah objek bekerja

(c) *Message* : Pengiriman pesan dari satu objek ke objek lain

2.15.4.5 Statechart Diagram

Menurut Booch (1999,p331)*Statechart* diagram menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari stimuli yang diterima. Pada umumnya *statechart* diagram menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart* diagram).

Dalam UML, *state* digambarkan berbentuk segiempat dengan sudut membulat dan memiliki nama sesuai kondisinya saat itu. Transisi antar *state* umumnya memiliki kondisi *guard* yang merupakan syarat terjadinya transisi yang bersangkutan, dituliskan dalam kurung siku. *Action* yang dilakukan sebagai akibat dari *event* tertentu dituliskan dengan diawali garis miring.

Titik awal dan akhir digambarkan berbentuk lingkaran berwarna penuh dan berwarna setengah.

Simbol-simbol yang digunakan dalam membuat sebuah s*tatechart* diagram yaitu:

Initial state *Final state* *State* *Transition*

Gambar 2.5 Simbol-simbol *Statechart* Diagram

(sumber: Booch , 1999, p333)

(a) *Initial state* : titik awal proses

(b) *final state* : kondisi yang sedang terjadi

(c) *state* : *state* aksi yang mengeksekusi sebuah aksi

(d) *transition* : atribut yang sedang dijalankan sebuah kelas

2.15.4.6 Collaboration Diagram

Elemen-elemen dari sebuah sistem yang bekerja sama untuk memenuhi tujuan sistem, dan setiap bahasa perancangan harus mempunyai cara untuk mewujudkan hal tersebut. *Collaboration* diagram dirancang untuk memenuhi syarat tersebut (Schmuller, 1999, p13).

Menurut Schmuller(1999,p120) Suatu *Colaboration diagram* adalah perpanjangan dari objek diagram. Sedangkan suatu objek diagram memperlihatkan objek dan hubungan mereka dengan yang lain. Sebagai tambahan untuk assosiasi antara objek, diagram kolaborasi menunjukkan pesan bahwa objek mengirim satu dengan yang lain.

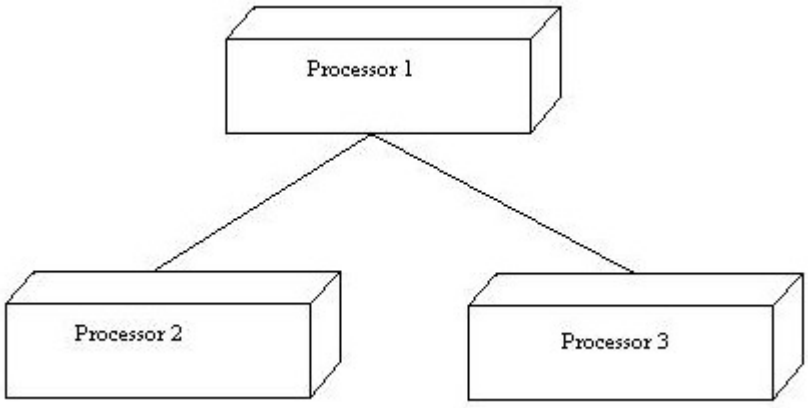
2.15.4.7 Component Diagram

Component diagram dapat menggambarkan hubungan organisasi di antara kumpulan komponen dan menunjukkan sudut pandang implementasi dari sistem. *Component* diagram adalah *class* diagram yang lebih menekankan terhadap komponen sistem. Schmuller(1999, p13).

Menurut Whitten et. al. (2004, p673)**,** *component* diagram adalah diagram yang menggambarkan hubungan organisasi di antara kumpulan komponen dan ketergantungan di antara komponen *software* yang digunakan di dalam sistem. *Component* diagram ini dapat digunakan untuk menunjukkan coding yang dibuat didalam sistem dibagi ke dalam modul-modul atau ke dalam komponen-komponen.

2.15.4.8 Deployment Diagram

Deployment diagram menggambarkan arsitektur dari sistem komputer, komputer dan alat-alatnya, menunjukkan koneksi yang terjadi didalam sistem komputer tersebut antara satu dengan yang lainnya dan menunjukkan software yang digunakan didalam mesin komputer. Setiap komputer akan digambarkan sebagai sebuah kotak, hubungan antara komputer yang satu dengan yang lainnya di gambarkan dengan garis, seperti yang terlihat pada Gambar 2.7. (Schmuller,1999,p13).



Gambar 2.6 *Deployment diagram*

(sumber gambar: Schmuller (1999,14))

2.15.5 Langkah-Langkah Penggunaan UML

Berikut ini adalah tips pengembangan piranti lunak dengan menggunakan UML:

1. Buatlah daftar *business process* dari level tertinggi untuk mendefinisikan aktivitas dan proses yang mungkin muncul.

2. Petakan *use case* untuk tiap *business process* untuk mendefinisikan dengan tepat fungsionalitas yang harus disediakan oleh sistem. Kemudian perhalus *use case* diagram dan lengkapi dengan *requirement*, *constraints* dan catatan-catatan lain.
3. Buatlah *deployment* diagram secara kasar untuk mendefinisikan arsitektur fisik sistem.
4. Definisikan *requirement* lain (*non-fungsional*, *security* dan sebagainya) yang juga harus disediakan oleh sistem.
5. Berdasarkan *use case* diagram, mulailah membuat *activity* diagram.
6. Definisikan objek-objek level atas (*package* atau domain) dan buatlah *sequence* dan/atau *collaboration* diagram untuk tiap alir pekerjaan. Jika sebuah *use case* memiliki kemungkinan alir normal dan error, buatlah satu diagram untuk masing-masing alir.
7. Buarlah rancangan *user interface* model yang menyediakan antarmuka bagi pengguna untuk menjalankan skenario *use case*.
8. Berdasarkan model-model yang sudah ada, buatlah *class* diagram. Setiap *package* atau domain dipecah menjadi hirarki *class* lengkap dengan atribut dan metodanya. Akan lebih baik jika untuk setiap *class* dibuat unit tes untuk menguji fungsionalitas *class* dan interaksi dengan *class* lain.
9. Setelah *class* diagram dibuat, kita dapat melihat kemungkinan pengelompokan *class* menjadi komponen-komponen. Karena itu buatlah komponen diagram pada tahap ini. Juga, definisikan tes integrasi untuk setiap komponen meyakinkan ia berinteraksi dengan baik.

10. Perhalus *deployment* diagram yang sudah dibuat. Detilkan kemampuan dan *requirement* piranti lunak, sistem operasi, jaringan, dan sebagainya. Petakan komponen ke dalam node.
11. Mulailah membangun sistem. Ada dua pendekatan yang dapat digunakan :
    - Pendekatan *use case*, dengan meng-*assign* setiap *use case* kepada tim pengembang tertentu untuk mengembangkan unit *code* yang lengkap dengan tes.
    - Pendekatan komponen, yaitu meng-*assign* setiap komponen kepada tim pengembang tertentu.
12. Lakukan uji modul dan uji integrasi serta perbaiki model berserta *code*-nya. Model harus selalu sesuai dengan *code* yang aktual.
13. Piranti lunak siap dirilis.

2.16 nteraksi Manusia dan Komputer

Interaksi Manusia dan Komputer atau dalam istilah bahasa inggris *Human Computer Interaction* (HCI) juga dapat diartikan sebagai suatu ilmu yang mempelajari tentang bagaimana mendesain, mengevaluasi, dan mengimplementasikan sistem komputer yang interaktif sehingga dapat digunakan oleh manusia dengan mudah. Interaksi adalah komunikasi 2 arah antara manusia (user) dan sistem komputer. (http://idhaclassroom.com/2007/09/15/artikel-terbaru/interaksi-manusia-dan-komputer.html)

Menurut Shneiderman (1998, p18), ada lima kriteria yang harus dipenuhi dalam interaksi antara manusia dan komputer, yaitu :

a. *Time to Learn*

Lamanya waktu yang diperlukan bagi pengguna untuk dapat mempelajari serta bagaiman menggunakan *command* yang berhubungan dengan tugas.

b. *Speed of Performance*

Lamanya waktu yang dibutuhkan hingga tugas diselesaikan.

c. *Rate of Errors by Users*

Banyaknya jumlah dan jenis dari kesalahan yang dibuat pengguna sewaktu melaksanakan tugas.

d. *Retention Overtime*

Seberapa baik pengguna memelihara pengetahuan masalah setelah satu jam, satu hari, atau satu minggu.

e. *Subjective Satisfication*

Seberapa banyak pengguna (*users*) menyukai penggunaan aspek sistem yang berbeda.

Shneiderman (1998, pp74-75), juga menyatakan bahwa dalam merancang suatu sistem interaksi manusia dengan komputer haruslah memperhatikan delapan aturan umum yang disebut *Eight Golden Rules of Interactive Design* :

a. Berusaha keras untuk konsisten

Urutan aksi harus konsisten dalam situasi yang sama, seperti penggunaan istilah, warna, tampilan dan jenis huruf yang sama.

b. Memungkinkan pengguna yang sering menggunakan *shortcuts*

Penggunaan *shortcuts* untuk mengurangi jumlah interaksi dan meningkatkan kecepatan tampilan.

c. Memberikan umpan balik yang informatif

Respon balik harus diberikan untuk memberikan informasi kepada konsumen sesuai dengan aksi yang dilakukannya. Konsumen akan mengetahui aksi yang telah dan akan dilakukan dari respon balik ini. Respon bisa berupa konfirmasi atau informasi atas dasar suatu aksi.

d. Merancang dialog untuk menghasilkan keadaan akhir

Urutan aksi harus diatur dalam grup dimana ada awal, tengah, dan akhir dengan adanya umpan balik yang dapat memberikan pilihan untuk melanjutkan grup aksi berikutnya.

e. Memberikan penanganan kesalahan yang sederhana

Sistem harus dirancang agar pemakai tidak membuat kesalahan yang serius. Jika pemakai melakukan kesalahan, sistem harus bisa mendeteksi kesalahan dan memberikan instruksi yang sederhana, membangun dan khusus untuk melakukan perbaikan.

f. Mengijinkan pembalikan aksi (*undo*) dengan mudah

Aksi harus dapat dibalik jika memungkinkan. Ciri ini mengurangi kegelisahan, karena pemakai tahu bahwa kesalahan dapat diperbaiki sehingga mendorong penjelajahan pilihan yang tidak biasa dipakai.

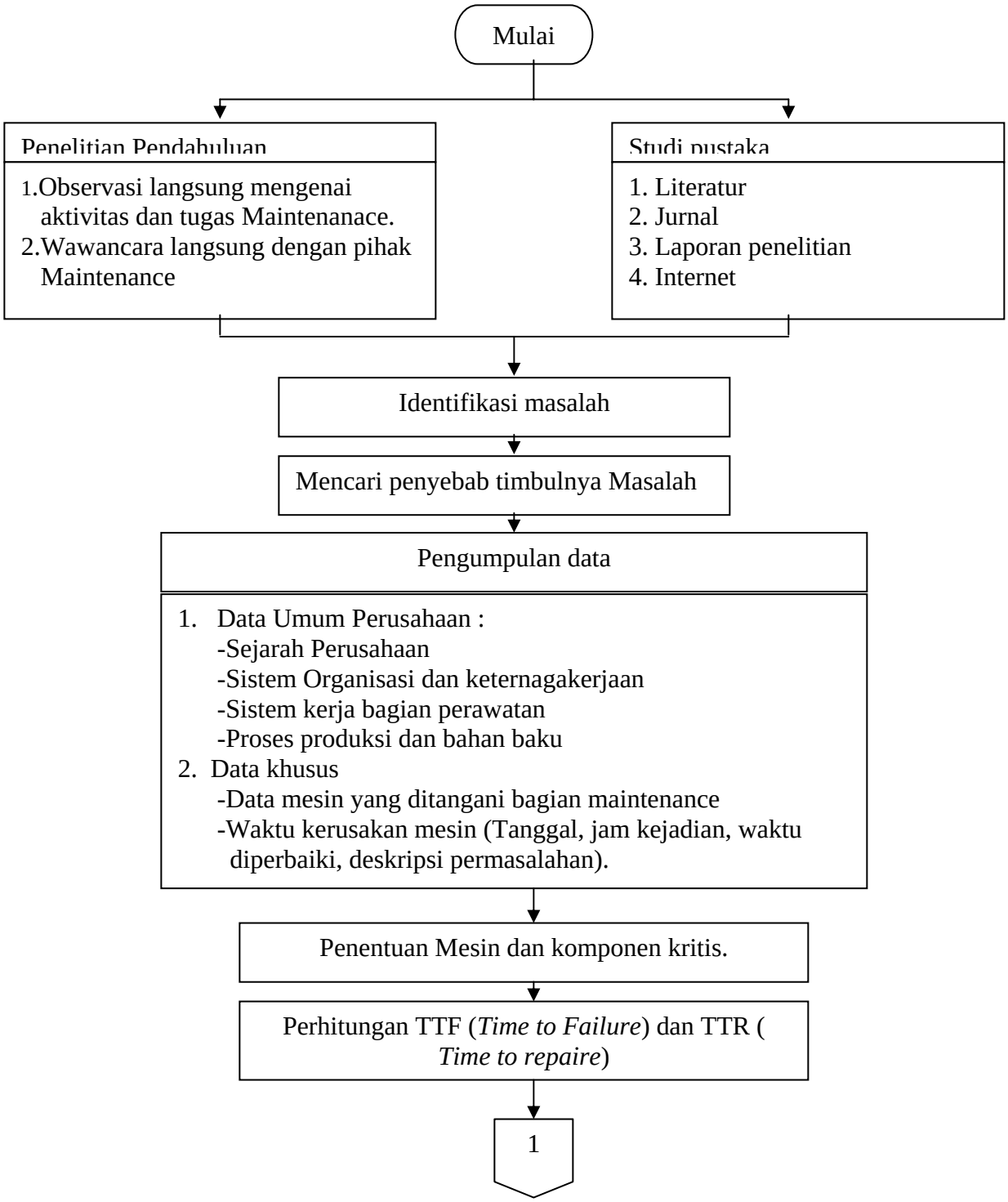
g. Mendukung *internal focus of control*

Pengguna berinisiatif dalam melakukan aksi daripada menunggu respon dari sistem untuk beraksi.

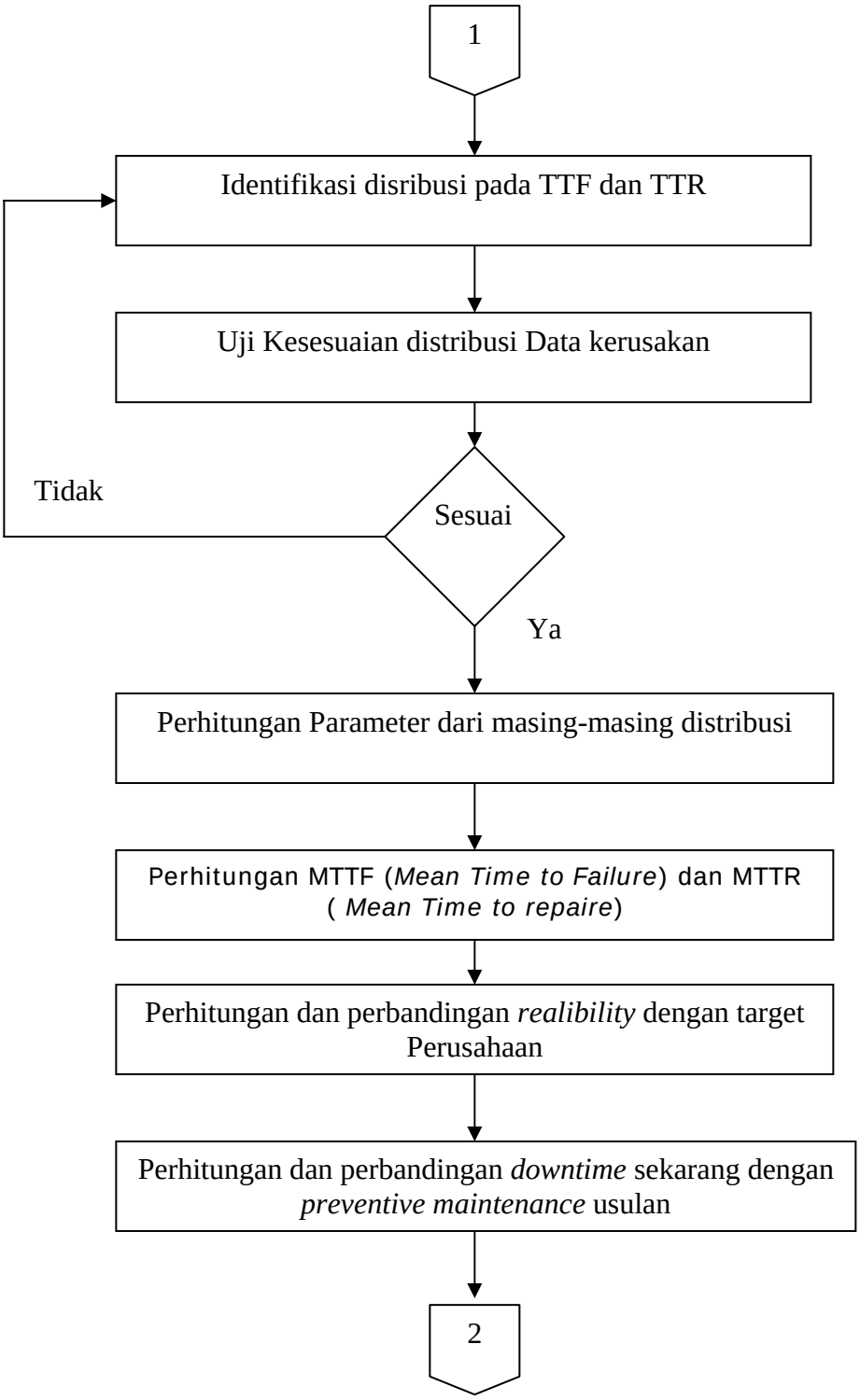
h. Mengurangi beban ingatan jangka pendek

Keterbatasan ingatan pada manusia harus ditanggulangi oleh program dengan tidak banyak membuat pengguna untuk melakukan proses penyimpanan memori.

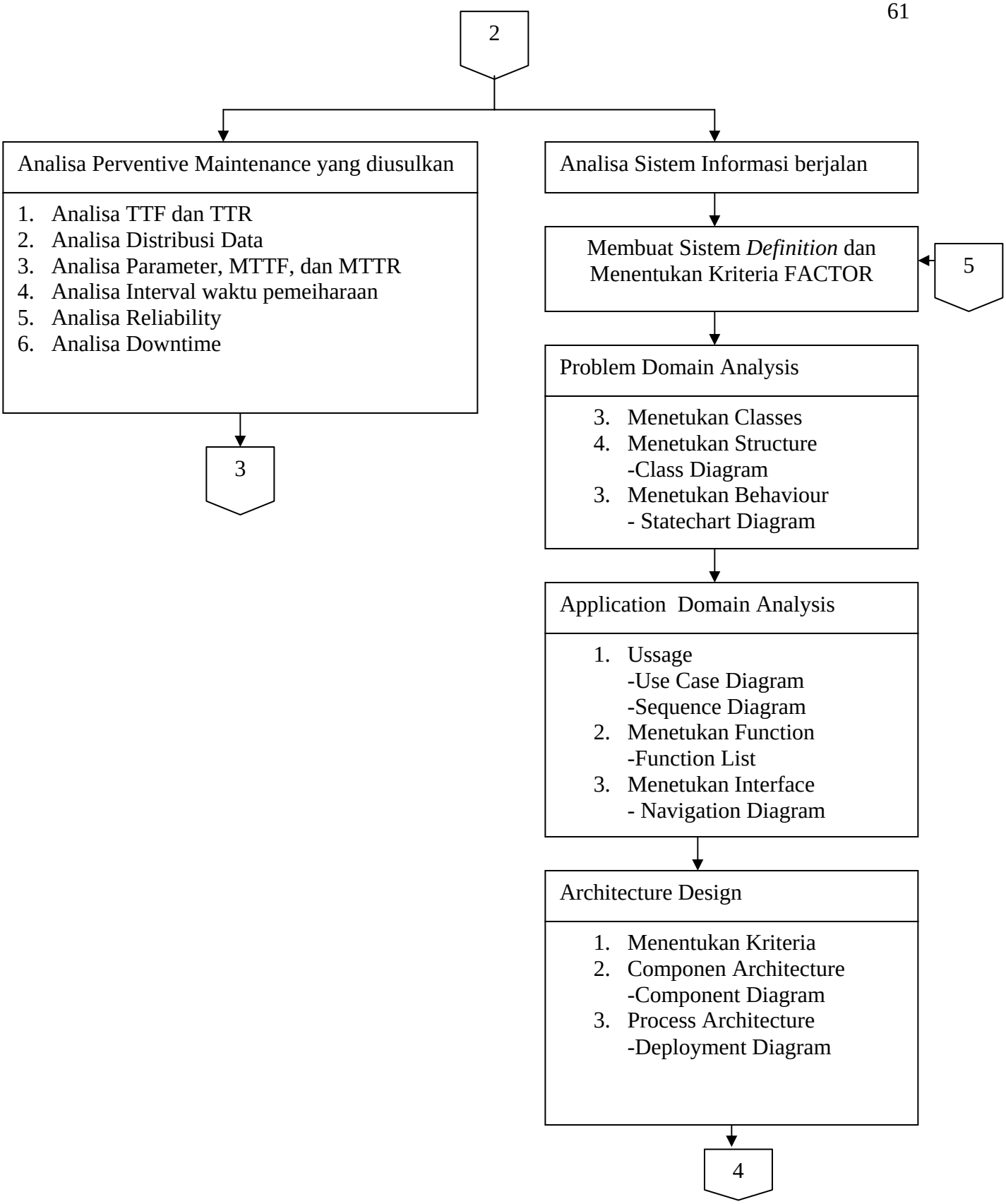
2.17 Kerangka Pikir



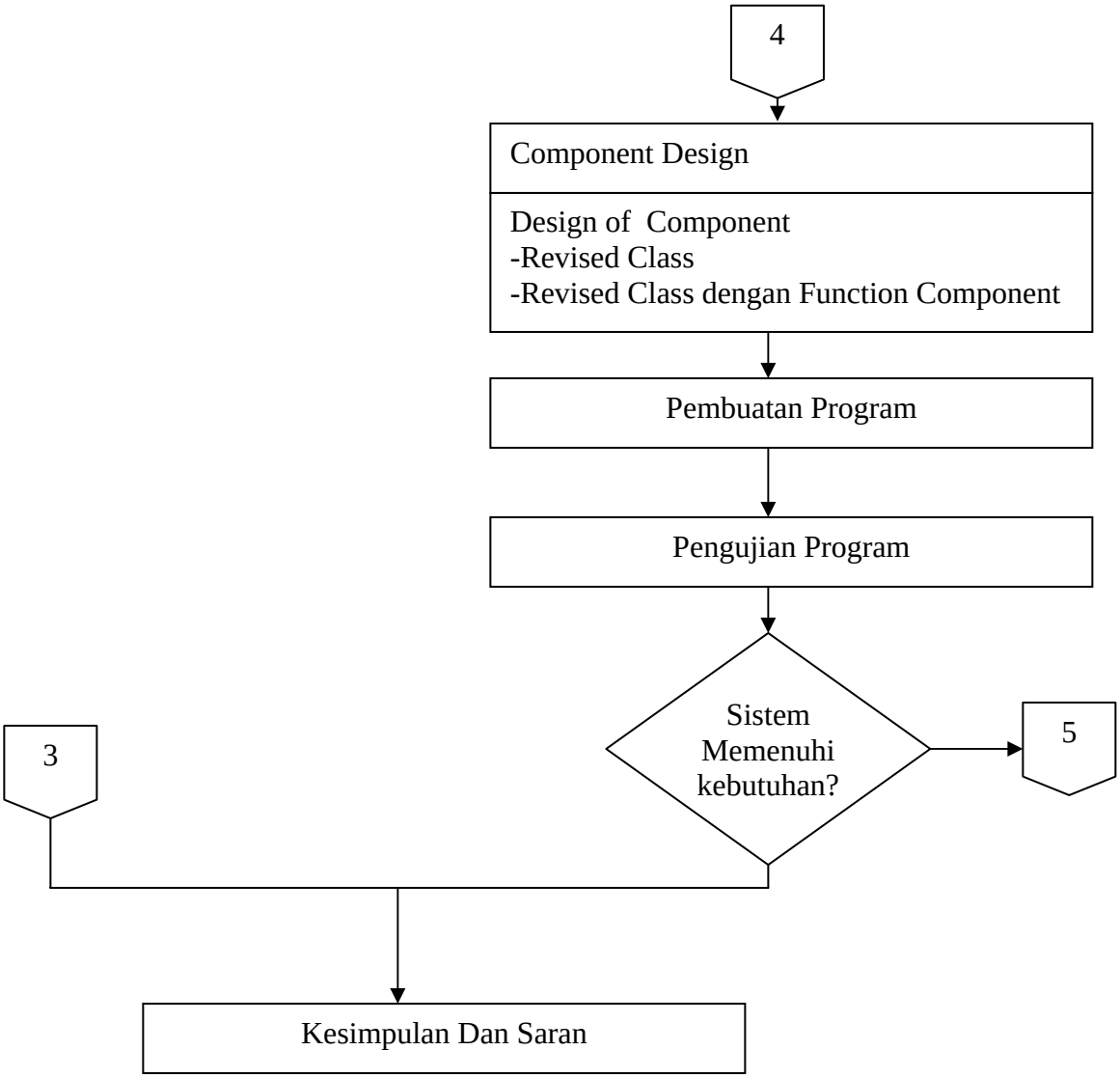
Gambar 2.1 Diagram Alir Model Metodologi Pemecahan Masalah



Gambar 2.2 Diagram Alir Model Metodologi Pemecahan Masalah (Lanjutan)



Gambar 2.3 Diagram Alir Model Metodologi Pemecahan Masalah (Lanjutan)



Gambar 2.3 Diagram Alir Model Metodologi Pemecahan Masalah (Lanjutan)