# BAB II

# TINJAUAN PUSTAKA

## 2.1 Sistem Ekstraksi DEA Unit Purifikasi Kilang Polypropylene

Wahyudin (2010) menjelaskan fungsi sistem ekstraksi DEA adalah untuk menurunkan kadungan *carbonyl sulfide* yang ada pada bahan baku *polypropylene* hingga menjadi 5 ppmmol maksimum, Sedangkan *hydrogen sulfide* diturunkan kandungannya dari 6000 menjadi 10 ppm.

Sistem ekstraksi DEA terdiri dari dari tiga kolom yaitu :

- Dua kolom liquid-liquid extraction yaitu C-201(primary DEA extractor) dan C-202 (secondary DEA extractor), dimana bahan baku *polypropylene* secara berlawanan arah dikontakan dengan larutan DEA. Untuk memperluas kontak, didalam kolom terdapat 3$^{rd}$ stage polypropylene pall ring 1 ½ inchi.
- Satu kolom DEA regenerator (C-203) yang berfungsi untuk melucuti *carbonyl sulfide* dan *hydrogen sulfide* dalam larutan DEA dengan temperatur 120$^{o}$C dan tekanan 0,5 kg/cm$^{2}$g. Untuk membantu memisahkannya, di dalam C-203 terdapat 20 tray, setiap tray mempunyai 48 valve tray dengan material stainless steel. Pada tray nomor 1 s/d 5 dilapisi dengan monel.

Pada kolom C-201 dan C-202 terjadi proses ekstrkasi cair cair sedangkan pada kolom C-203 terjadi proses distilasi. Reaksi yang terjadi pada sistem ekstraksi DEA adalah :

- Reaksi pada *extractor :*
  $(CH_2CH_2OH)_2NH + 2COS \rightarrow (CH_2CH_2OCOSH)_2NH$
- Reaksi pada regenerator :
  $(CH_2CH_2OCOSH)_2NH + 2\ H_2O \rightarrow (CH_2CH_2OH)_2NH + 2H_2S + 2CO_2$

### 2.1.1 Ekstraksi Cair - Cair

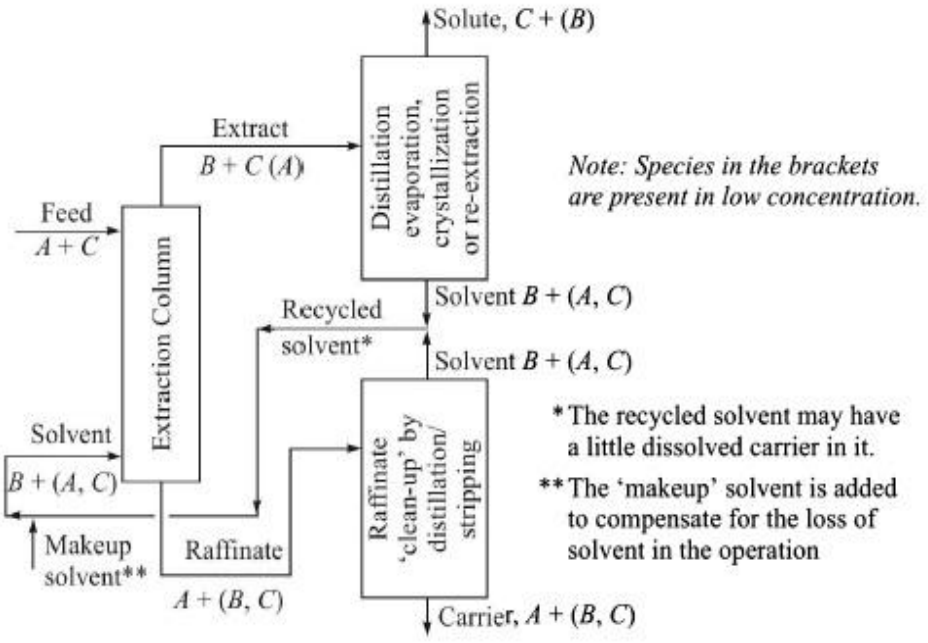
(Dutta, 2009:422-423) menjelaskan bahwa ekstraksi cair-cair merupakan suatu operasi transfer massa dimana suatu larutan yang terdiri dari campuran antara zat terlarut (solute) dan cairan pembawa (diluen), dikontakkan dengan

suatu pelarut yang disebut *solvent* untuk memisahkan antara solute dan diluen. Dua fase liquid yang berbeda densitas selanjutnya dipisahkan. Fase yang banyak mengandung solute disebut dengan ekstrak, sedangkan fase yang mengandung sedikit solute disebut rafinat. Rafinat terdiri dari cairan pembawa, *solvent* dan sedikit solute.

Proses ekstraksi secara umum tediri dari 4 tahap yaitu :

- Mengontakkan campuran dengan *solvent.*
- Memisahkan fase ekstrak dan fase rafinat yang mempunyai densitas yang berbeda
- Pemisahan dan pembaruan solute dari fase ekstrak ( menggunakan proses distilasi, evaporasi, kristalisasi, dll.).
- Pemisahan dan pembaruan *solvent* dari setiap fase (biasanya menggunakan proses distilasi).

Ekstraksi terutama digunakan bila pemisahan campuran dengan cara distilasi tidak mungkin dilakukan atau tidak ekonomis. Proses eksraksi dapat dilihat pada gambar 1.




*Sumber : Principles of Mass Transfer and Separation Process (Dutta, 2009: 423)*

Gambar 1 Proses ekstrasi cair-cair

Menurut (Dutta, 2009 : 424-436) syarat syarat pelarut yang baik sehingga proses esktraksi berjalan baik adalah :

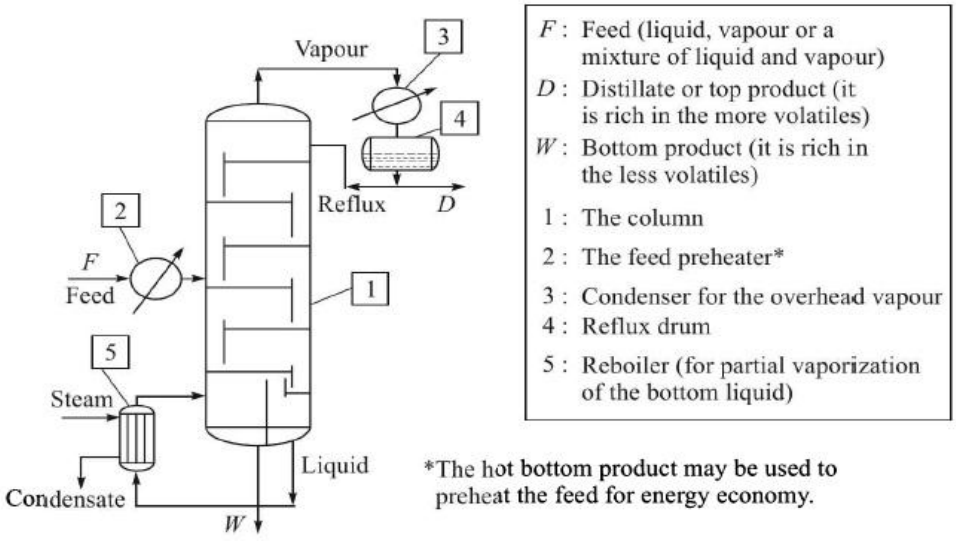
1. Kemampuan melarutkan yang tinggi komponen zat terlarut didalam campuran.
2. Bisa untuk diambil kembali.
3. Perbedaan berat jenis antara ekstrak dan rafinat lebih besar.
4. Pelarut dan larutan yang diekstraksi harus tidak mudah campur.
5. Tidak mudah bereaksi dengan zat yang akan diekstraksi.
6. Tidak merusak alat secara korosi.
7. Tidak mudah terbakar, tidak beracun, dan harganya relatif murah.

Adapun pertimbangan pemakaian proses ekstraksi sebagai proses pemisahan antara lain :

1. Komponen larutan senstif terhadap pemanasan
2. Titik didih komponen-komponen dalam campuran berdekatan.
3. Kemudahan menguap (volatility) komponen hampir sama.
4.

### 2.1.2 Distilasi

Pada gambar 2 pada dilihat proses distilasi menggunakan menara distilasi.




*Sumber :Principles of Mass Transfer and Separation Process(Dutta, 2010:320)*

Gambar 2 Proses distilasi

Distilasi atau penyulingan adalah suatu metode pemisahan dua campuran atau lebih berdasarkan perbedaan kecepatan atau kemudahan menguap (volatilitas) bahan. Dalam distilasi, campuran zat dididihkan sehingga menguap. Zat yang memiliki titik didih lebih rendah akan menguap lebih dulu dan uap ini kemudian didinginkan kembali ke dalam bentuk cairan melalui proses kondensasi..Distilasi dilaksanakan dengan rangakaian alat berupa kolom/menara yang terdiri dari piring (*plate tower/tray*) sehingga dengan proses perpindahan panas komponen dapat menguap, terkondensasi, dan dipisahkan secara bertahap berdasarkan volatilitasnya (Dutta, 2010: 319).

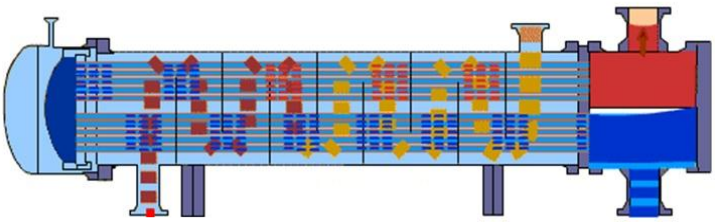
Selain proses ektraksi cair-cair dan distilasi, di dalam sistem ekstraksi DEA terdapat alat lain yang menunjang proses pada sistem ekstraksi DEA seperti Kondensor, *boiler, reboiler* dan pompa

### 2.1.3 Kondensor

Cahoyo, dkk (2013) menjelaskan beberapa hal yang berhubungan dengan kondensor sebagai berikut :

**Pengertian Kondensor**

Kondensor merupakan alat penukar kalor pada sistem refrigerasi yang berfungsi untuk melepaskan kalor ke lingkungan. Kondensor banyak digunakan dalam kehidupan kehidupan sehari-hari baik itu dalam industri rumah tangga, industri otomotif, maupun dalam industri farmasi dan obat-obatan. Di Indonesia sendiri, kondensor bukanlah hal yang asing. Dalam industri kimia kondensor banyak kita jumpai pada menara pemisahan yang melibatkan perpindahan panas dalam prosesnya. Di dalam sistem kompresi uap *(vapor compression)* kondensor adalah suatu komponen yang berfungsi untuk merubah fase *Refrigerant* dari uap bertekanan tinggi menjadi cairan bertekanan tinggi atau dengan kata lain pada kondensor ini terjadi proses kondensasi. *Refrigerant* yang telah berubah menjadi cair tersebut kemudian dialirkan ke *evaporator* melalui pompa.



*Sumber : Cahyono,dkk. (2013)*

Gambar 3 Kondensor pada sistem kompresi uap

**Pengertian Kondensasi**

Kondensasi berasal dari bahasa latin yaitu *condensare* yang berarti membuat tertutup. Kondensasi merupakan perubahan wujud zat dari gas atau uap menjadi zat cair. Kondensasi terjadi pada pemampatan atau pendinginan jika tercapai tekanan maksimum dan suhu di bawah suhu kritis. Kondensasi terjadi ketika uap didinginkan menjadi cairan, tetapi dapat juga terjadi bila sebuah uap dikompresi (yaitu tekanan ditingkatkan) menjadi cairan, atau mengalami kombinasi dari pendinginan dan kompresi.

**Cara Kerja Kondensor**

Uap panas yang masuk ke kondensor dengan temperatur yang tinggi dan bertekanan yang merupakan hasil proses dari turbin. Kemudian uap panas masuk ke dalam *Suction Pipe* dan kemudian mengalir dalam tube. Dalam tube, uap panas didinginkan dengan media pendingin air yang dialirkan melewati sisi luar tube, kemudian keluar melalui *Discharge Pipe* dengan temperatur yang sudah turun. Prinsip kondensasi di kondensor adalah menjaga tekanan uap *superheat refrigerant* yang masuk ke kondensor pada tekanan tertentu kemudian suhu *Refrigerant*nya diturunkan dengan membuang sebagian kalornya ke medium pendingin yang digunakan di kondensor. Sebagai medium pendingin digunakan udara dan air atau gabungan keduanya. Dalam perancangan ini akan digunakan air sebagai media pendingin.Pada proses pendinginan *(cooling)* cairan *Refrigerant* yang menguap di dalam pipa-pipa *cooling coil (evaporator)* telah menyerap panas sehingga berubah wujudnya menjadi gas dingin dengan kondisi superheat pada saat meninggalkan *Cooling Coil*. Panas yang telah diserap oleh *Refrigerant*

ini harus dibuang atau dipindahkan ke suatu medium lain sebelum ia dapat kembali diubah wujudnya menjadi cair untuk dapat mengulang siklusnya kembali.

**Komponen Utama dari Kondensor**

Kondensor pada umumnya memiliki beberapa komponen utama, dimana masing-masing komponen memiliki fungsinya tersendiri. Adapun komponen-komponen utama dari kondensor adalah sebagai berikut:

1. *Suction Pipe* dan *Discharge Pipe* (Pipa saluran masuk dan pipa saluran keluar).

a. *Suction Pipe*

*Suction Pipe* adalah pipa saluran masuk untuk masuknya media pendingin ke dalam kondensor,yang mana media pendingin itu berupa fluida cair yang bertekanan yang merupakan hasil dari pemampatan di kompresor.

*b. Discharge Pipe*

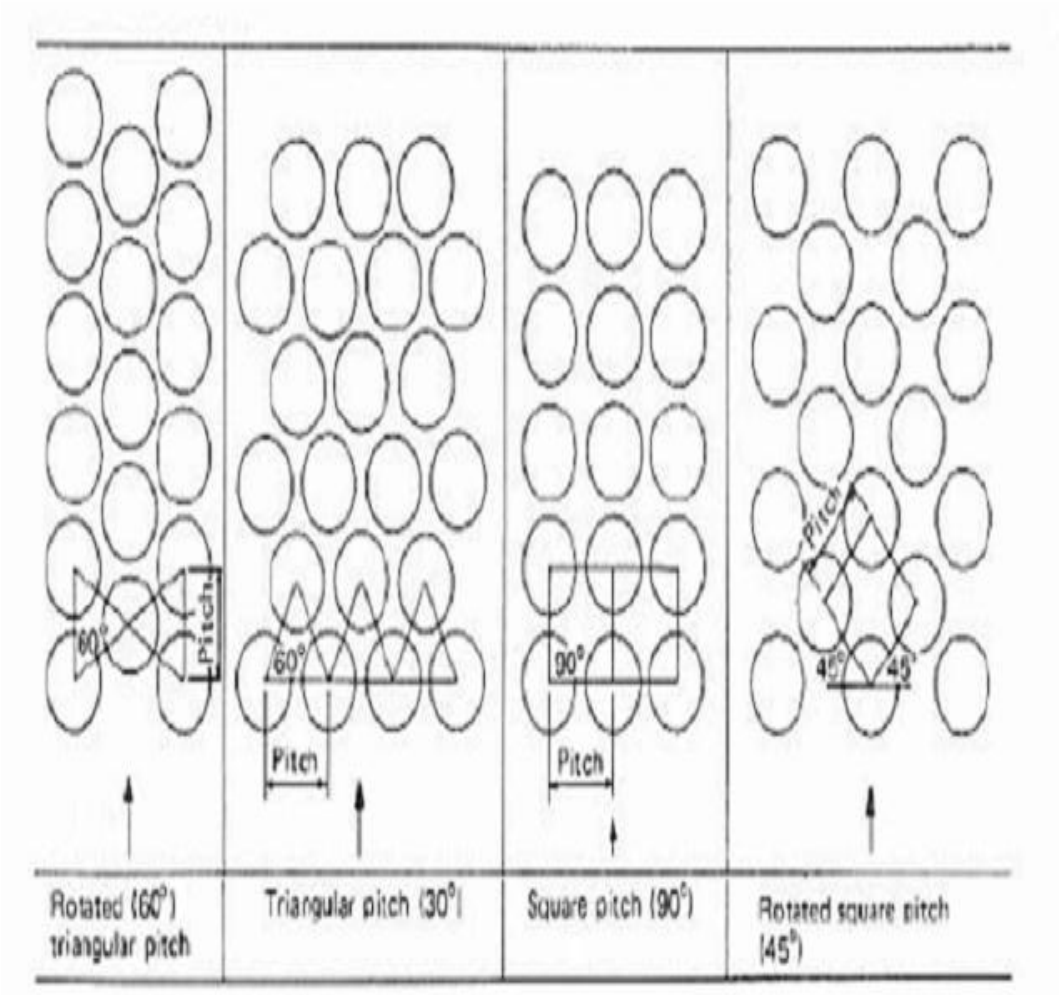
*Discharge pipe* adalah pipa saluran keluar *Refrigerant* dari kompresor melalui tube ke tangki receiver.

2. *Tube* ( Pipa dalam Kondensor )

*Tube* adalah pipa aliran yang dilalui *Refrigerant* yang bertekanan dan panas yang merupakan hasil dari turbin melalui *suction pipe* dan akan disalurkan ke discharge pipe dan kemudian diterima oleh tangki *receiver*. Umumnya terdapat empat susunan *tube* yaitu, *Triangular* ($30^o$), *Rotate square* ($60^o$), *Square* ($90^o$), *Rotate square* ($45^o$).

Susunan *triangular* memberikan nilai perpindahan panas yang lebih baik bila dibandingkan dengan susunan *rotate* square dan *square* karena dengan susunan *triangular* dapat menghasilkan turbulensi yang tinggi, namun begitu *tube* yang disusun secara *triangular* akan menghasilkan *pressure drop* (penurunan tekanan) yang lebih tinggi dari pada susunan *rotate square* dan *square*. Apabila fluida yang digunakan memiliki tingkat *fouling* yang tinggi dan memerlukan pembersihan secara mekanik (*mechanical cleaning*) susunan *tube* secara riangular tidak digunakan, sebaiknya digunakan susunan *square*, apabila jenis *cleaning*

yang digunakan adalah *chemical cleaning,* maka susunan tube secara triangular dapat diperimbangkan kembali, mengingat untuk *chemical cleaning* tidak memerlukan akses jalur ruang (*acess lanes*) yang lebih seperti pada *mechanical cleaning.*




*Sumber : Cahyono,dkk. (2013)*

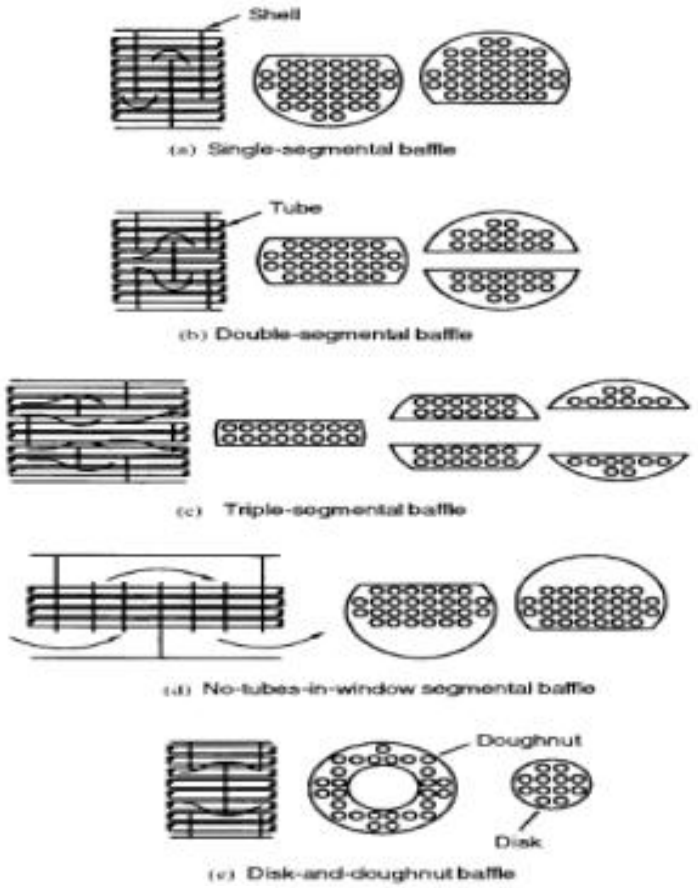
Gambar 4 Lay-Out pada Tube

3. *Water Box*

Ruang air pendingin (*Refrigerant)* yang terbuat dari baja karbon

4. *Buffle*

*Buffle* merupakan jarak bagi antar *tube.*




*Sumber : Cahyono,dkk. (2013)*

Gambar 5 Jenis – jenis *buffle* yang ada pada *tube*

### 2.1.4 *Boiler*

Sudrazat (2013) menjelaskan beberapa hal yang berhubungan dengan *boiler* sebagai berikut :

**Pengertian *Boiler***

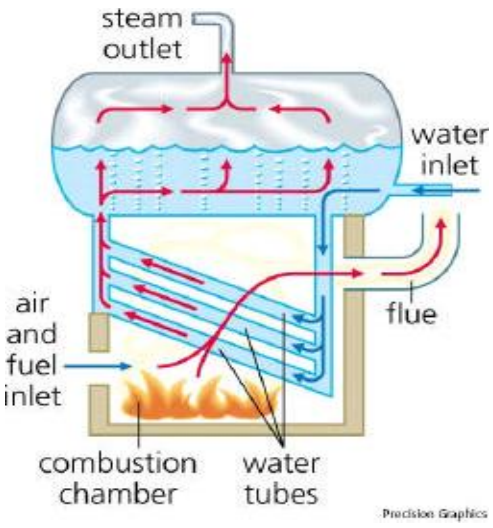
*Boiler* adalah bejana tertutup dimana panas pembakaran dialirkan ke air sampai terbentuk air panas atau *steam*. Air panas atau *steam* pada tekanan tertentu kemudian digunakan untuk mengalirkan panas ke suatu proses. Sistem *boiler* terdiri dari: sistem air umpan, sistem *steam* dan sistem bahan bakar. Sistem air umpan menyediakan air untuk *boiler* secara otomatis sesuai dengan kebutuhan

*steam*. Berbagai kran disediakan untuk keperluan perawatan dan perbaikan. Sistem *steam* mengumpulkan dan mengontrol produksi *steam* dalam *boiler*. *Steam* dialirkan melalui sistem pemipaan ke titik pengguna. Pada keseluruhan sistem, tekanan *steam* diatur menggunakan kran dan dipantau dengan alat pemantau tekanan. Sistem bahan bakar adalah semua peralatan yang digunakan untuk menyediakan bahan bakar untuk menghasilkan panas yang dibutuhkan. Peralatan yang diperlukan pada sistem bahan bakar tergantung pada jenis bahan bakar yang digunakan pada sistem.Air yang disuplai ke *boiler* untuk dirubah menjadi *steam* disebut air umpan. Dua sumber air umpan adalah:

(1) Kondensat atau *steam* yang mengembun yang kembali dari proses

(2) Air makeup (air baku yang sudah diolah) yang harus diumpankan dari luar ruang *boiler* dan *plant* proses.

**Bagian-bagian *boiler***




*Sumber : Sudrazat (2013)*

Gambar 6 Water Tube Boliler

- Ruang bakar/furnace merupakan tempat berlangsungnya pembakaran.
- Alat pembakar/burner merupakan tempat bercampurnya bahan bakar dengan udara dan melakukan pembakaran.
- Permukaan penguap/*steam*ing surface, berfungsi menangkap energi kalor dari gas dan meneruskannya ke air sehingga air menjadi uap.
- Cerobong/stack berfungsi sebagai saluran pembuangan gas asap dan menarik api.
- Drum uap/*steam* drum, berfungsi sebagai pengumpul uap, pemisah uap dan tempat pemasukan air.

**2.1.5 R*eboiler***

Rahman (2013) menjelaskan beberapa hal yang berhubungan dengan r*eboiler* sebagai berikut :

**Pengertian Re*boiler***

Re*boiler* adalah salah satu exchanger panas atau satu perlengkapan destilasi kolom yang memberikan perpindahan panas. Re*boiler* digunakan untuk menguapkan cairan yang masuk sehingga uap yang dihasilkan masuk kembali dan naik ke kolom, dan cairan sisanya akan tertinggal dibagian bawah kolom sebagai resiud. Tangki re*boiler* vertical dan horizontal bekerja dengan sirkulasi natural, dimana aliran yang mengalir ke re*boiler* disebabkan oleh ketidakseimbangan tekanan hidrostatik antara cairan didalam tower dan campuran didalam tube re*boiler*.

Jenis re*boiler*:

1. Kettle re*boiler*s
2. Thermosyphon re*boiler*s
3. Fired re*boiler*s
4. Forced sirkulasi re*boiler*s

**Sistem Kerja**

Dua fluida mengalir dengan temperatur awal yang berbeda mengalir sepanjang heat exchangers. Satu aliran mengalir sepanjang tabung sedangkan arus lain pada bagian luar tabung tetapi masih di dalam shell. Panas ditransfer dari satu fluida ke fluida lainnya melalui dinding tabung, baik dari sisi tabung menuju shell atau sebaliknya. Fluida bisa merupakan cairan atau gas pada sisi shell maupun pada sisi tabung. Dalam tujuan memindahkan panas secara efisien, suatu area perpindahan kalor yang besar harus digunakan, oleh karena itu terdapat banyak tabung. Dengan cara ini, panas yang dibuang dapat disimpan untuk digunakan. Hal ini adalah suatu jalan yang baik untuk memelihara energi.

Heat exchanger yang berfasa tunggal (cairan atau gas) pada setiap sisi dapat disebut heat exchanger berfasa satu atau berfasa tunggal. Heat exchanger berfasa dua dapat digunakan untuk memanaskan cairan dan mendidihkannya sehingga menjadi gas (uap air), terkadang disebut *boiler*, atau mendinginkan uap air untuk dikondensasikan menjadi bentuk cairan (condenser), pada umumnya

perubahan fase yang terjadi berada pada sisi shell. *Boiler* didalam mesin uap lokomotif biasanya cukup besar, yang pada umumnya shell and tube heat exchanger terbentuk silinder. Pada pembangkit tenaga listrik yang besar dengan *steam*-driven turbin, shell and tube condenser digunakan untuk mengkondensasikan uap air yang keluar turbin ke dalam bentuk air yang dapat didaur ulang kembali menjadi uap air, yang mungkin pada shell and tube tipe *boiler*.

### 2.1.7 Pompa

Surya (2009) menjelaskan beberapa hal yang berhubungan dengan pompa sebagai berikut :

**Pengertian Pompa**

Pompa adalah suatu alat atau mesin yang digunakan untuk memindahkan cairan dari suatu tempat ke tempat yang lain melalui suatu media perpipaan dengan cara menambahkan energi pada cairan yang dipindahkan dan berlangsung secara terus menerus.Pompa beroperasi dengan prinsip membuat perbedaan tekanan antara bagian masuk (*suction*) dengan bagian keluar (*discharge*). Dengan kata lain, pompa berfungsi mengubah tenaga mekanis dari suatu sumber tenaga (penggerak) menjadi tenaga kinetis (kecepatan), dimana tenaga ini berguna untuk mengalirkan cairan dan mengatasi hambatan yang ada sepanjang pengaliran. Hambatan-hambatan pengaliran itu dapat berupa perbedaan tekanan, perbedaan ketinggian atau hambatan gesek.

**Klasifikasi Pompa**

Secara umum, pompa dapat diklasifikasikan menjadi 2 bagian yaitu :

1. Pompa kerja positif (*positive displacement pump*)

Pada pompa kerja positif, kenaikan tekanan cairan di dalam pompa disebabkan oleh pengecilan volume ruangan yang ditempati cairan tersebut. Adanya elemen yang bergerak dalam ruangan tersebut menyebabkan volume ruangan akan membesar atau mengecil sesuai dengan gerakan elemen tersebut. Contoh pompa kerja positif adalah pompa rotary

***Pompa Rotari***

Pompa rotari adalah pompa perpindahan positif dimana energi mekanis ditransmisikan dari mesin penggerak ke cairan dengan menggunakan elemen yang berputar (rotor) di dalam rumah pompa (*casing*).

Beberapa pompa rotari yang banyak ditemukan antara lain sebagai berikut :

a. Pompa roda gigi luar

   Rotornya berupa sepasang roda gigi yang berputar di dalam rumah pompa. Roda gigi itu dapat berupa gigi heliks-tunggal, heliks-ganda atau gigi lurus.

b. Pompa roda gigi dalam

   Mempunyai rotor yang berupa roda gigi dalam yang berpasangan dengan roda gigi luar yang bebas (*idler*).

c. Pompa kam dan piston, disebut juga pompa *plunyer rotary*

   Terdiri dari lengan eksentrik dan lengan bercelah pada bagian atasnya.

d. Pompa cuping (pompa *lobe*)

   Mempunyai dua rotor atau lebih dengan dua, tiga, empat cuping atau lebih pada masing-masing rotor.

e. Pompa sekrup

   Mempunyai satu, dua, tiga sekrup yang berputar dalam rumah pompa yang diam.

f. Pompa vane

   Rotornya berupa elemen berputar yang dipasang eksentrik dengan rumah pompa. Pada keliling rotor terdapat alur-alur yang diisi bilah-bilah sudu yang dapat bergerak bebas. Ketika rotor diputar sudu-sudu bergerak dalam arah radial akibat gaya sentrifugal, sehingga salah satu ujung sudu selalu kontak dengan permukaan dalam rumah pompa membentuk sekat-sekat ruangan di dalam pompa.

Adapun keuntungan-keuntungan dari pompa rotari adalah sebagai berikut :

1. Pompa rotari banyak digunakan pada pemompaan cairan yang viskositasnya lebih tinggi dari air.
2. Aliran yang dihasilkan hampir merata (*uniform*), karena putaran rotor relatif konstan.

2. Pompa Kerja Dinamis (*non-positive displacement pump*)

Pompa dinamik adalah pompa yang bekerja dengan volume ruang yang tetap. *Head* yang dibangkitkan merupakan perubahan energi kinetik fluida yang bergerak karena dorongan oleh sudu-sudu impeler yang berputar dalam rumah pompa. Contoh dari pompa kerja dinamis adalah pompa sentrifugal.

***Pompa Sentrifugal***

Pompa sentrifugal merupakan pompa yang paling banyak digunakan karena mempunyai bentuk yang sederhana dan harga yang relatif murah. Pada pompa sentrifugal, energi penggerak dari luar diberikan kepada poros yang kemudian digunakan untuk menggerakkan baling-baling yang disebut impeler.

Berdasarkan jenis aliran dalam impeler, pompa sentrifugal dibedakan menjadi 3 bagian, yaitu sebagai berikut :

1. Pompa aliran radial
2. Pompa aliran aksial
3. Pompa aliran campur (mixed flow)

Beberapa keuntungan dari pompa sentrifugal (Lazarkiewics, 1965) adalah sebagai berikut :

1. Gerakan impeler yang kontinyu menyebabkan aliran tunak (*steady-state*) dan tidak berpulsa
2. Keandalan operasi tinggi disebabkan gerakan elemen yang sederhana dan tidak adanya katup-katup
3. Kemampuan untuk beroperasi pada putaran tinggi, yang dapat dikopel dengan motor listrik, motor bakar atau turbin uap
4. Ukuran kecil sehingga hanya membutuhkan ruang yang kecil, lebih ringan dan biaya instalasi ringan
5. Harga murah dan biaya perawatan murah

Sedangkan beberapa elemahan / kekurangan dari pompa sentrifugal, adalah sebagai berikut :

1. Pompa *single stage* tidak dapat menghasilkan tekanan yang tinggi kecuali dengan putaran > 10.000 rpm. Pompa *multistage* / tekanan tinggi sangat mahal terutama dari bahan anti korosi.
2. Efisiensi dapat turun drastis, jika tidak digunakan pada debit sekitar puncak efisiensi.
3. Pompa sentrifugal tidak *self priming* (tidak bisa langsung menghisap pada saat pompa tidak ada cairan) dan performace akan turun drastis dengan meningkatnya kekentalan cairan.

**Prinsip Kerja Alat**

1. Prinsip Kerja Pompa Rotari

Prinsip kerja pompa rotari adalah energi mekanis ditransmisikan dari mesin penggerak ke cairan dengan menggunakan elemen yang berputar (rotor) di dalam rumah pompa (*casing*). Pada waktu rotor berputar di dalam rumah pompa, akan terbentuk kantong-kantong yang mula-mula volumenya besar (pada sisi isap) kemudian volumenya berkurang (pada sisi tekan) sehingga fluida akan tertekan keluar.

2. Prinsip Kerja Pompa Sentrifugal

Pada pompa sentrifugal, energi penggerak dari luar diberikan kepada poros yang kemudian digunakan untuk menggerakkan impeler. Impeler memutar cairan yang masuk ke dalam pompa sehingga mengakibatkan energi tekanan dan energi kinetik cairan bertambah. Cairan akan terlempar ke luar akibat gaya sentrifugal yang ditimbulkan gerakan impeler. Cairan yang keluar dari impeler ditampung oleh saluran berbentuk volut (spiral) di sekeliling impeler dan disalurkan ke luar pompa melalui difuser. Di dalam difuser ini sebagian energi kecepatan akan diubah menjadi energi tekanan.

## 2.2 Sistem, Model dan Simulasi

Siswoyo (2010) menjelaskan beberapa hal yang berhubungan dengan sistem,model, dan simulasi sebagai berikut :

### 2.2.1 Sistem

Sistem adalah sekumpulan obyek yang tergabung dalam suatu interaksi dan interdependensi yang teratur. Sistem dibedakan menjadi dua tipe yaitu sistem diskrit dan sistem kontinyu. Adapun komponen-komponen yang ada pada sistem adalah :

- Entitas – objek yang sedang diamati dari sistem
- Atribut – identitas dari entitas
- Aktivitas – suatu masa yang mewakili proses suatu entitas
- Status – kumpulan variabel yg dibutuhkan untuk menggambarkan sistem
- Kejadian – Kejadian yg mengubah status system

### 2.2.2 Model

Model merupakan penyederhanaan dari sistem yang akan dipelajari. Model sangat beragam, bisa dalam bentuk ikon, analog atau simbol. Model ikon meniru sistem nyata secara fisik, seperti globe (model dunia), planetarium (model system ruang angkasa), dan lain-lain. Model analog meniru sistem hanya dari perilakunya. Model simbol tidak meniru sistem secara fisik, atau tidak memodelkan perilaku sistem, tapi memodelkan sistem berdasarkan logikanya. Logika bisa bervariasi mulai dari intuisi ke bahasa verbal atau logika matematik. Karena model analisis simulasi harus dapat diimplementasikan pada komputer, maka model simulasi harus eksplisit, yaitu harus sebagai model simbolik paling tidak untuk level aliran logika.

**Model simbolik dapat diklasifikasikan menjadi :**

1. Model preskriptif atau deskriptif. Model preskriptif digunakan untuk mendefinisikan dan mengoptimalkan permasalahan. Model deskriptif menggambarkan sistem berdasarkan perilakunya dan permasalahan optimasi diserahkan ke analisis berikutnya.

2. Model diskrit atau kontinu. Pengklasifikasian model menjadi diskrit dan kontinu didasarkan pada variabelnya. Perbedaan paling penting dalam kedua

model adalah waktu. Jika revisi terhadap model terjadi secara kontinu berdasarkan waktu, maka model itu diklasifikasikan sebagai model kontinu.

3. Model probabilistik atau deterministik. Pembedaan kedua model ini juga didasarkan pada variabel model. Jika ada variabel acak, model kita klasifikasikan sebagai model probabilistik. Jika tidak, model merupakan klasifikasi model deterministik.

4. Model statis atau dinamis. Pembedaan kedua model ini juga didasarkan pada variabel model. Jika variabel model berubah sesuai dengan waktu, maka model digolongkan sebagai model dinamis.

5. Model loop terbuka atau tertutup. Pengklasifikasian model kedalam bentuk loop terbuka atau tertutup didasarkan pada struktur model. Pada model terbuka, output dari model tidak menjadi umpan balik untuk memperbaiki input. Sebaliknya adalah model loop tertutup.

### 2.2.3 Simulasi

Simulasi adalah suatu prosedur kuantitatif, yang menggambarkan sebuah sistem, dengan mengembangkan sebuah model dari sistem tersebut dan melakukan sederetan uji coba untuk memperkirakan perilaku sistem pada kurun waktu tertentu.

**Langkah-langkah Model Simulasi :**

1. Formulasikan Masalah & Buat Rencana Pemecahannya
2. Kumpulkan data dan Definisikan modelnya
3. Uji Validitas (utk Model)
4. Buat Program Komputer
5. Jalankan programnya
6. Uji Validitas
7. Rancang Percobaan
8. Jalankan Produksi
9. Analisis Data Output
10. Penyimpanan hasil dan Program yang dipakai

**Bahasa Simulasi**

Pemrograman model simulasi dapat dilakukan menggunakan bahasa umum komputer (general purposes language) atau menggunakan bahasa simulasi. Untuk memilih bahasa simulasi yang cocok, kita harus mempelajari beberapa bahasa simulasi, melihat dan memahami kelebihan dan kekurangan dari masing-masingnya, sehingga kita melakukan pemilihan yang tepat saat kita perlu menggunakan bahasa simulasi. Satu bahasa simulasi tidak dapat menjadi alat yang tepat untuk semua kasus permodelan simulasi.

**Karakteristik Bahasa Simulasi**

Struktur dinamis dan statis bahasa simulasi menyediakan kebutuhan jelas untuk mengeksekusi model simulasi. Beberapa sifat bahasa simulasi lainnya dibutuhkan atau sangat diinginkan untuk penggunaan efektif analisis simulasi sebagai teknik pembantu pengambilan keputusan.

- Pengembangan kode model. Kebanyakan bahasa simulasi masih membutuhkan pemasukan pernyataan kode untuk menciptakan kode model, tetapi kemampuan grafik mikrokomputer telah memungkinkan input grafik. Cara ini paling sesuai untuk bahasa yang fokus pada aliran objek melalui elemen atau blok model.
- Debugging model. Begitu mode simulasi sudah dikodekan menggunakan bahasa simulasi yang dipilih, langkah selanjutnya adalah debugging kode sehingga model simulasi berjalan ke penghentian normal. Syntax errors (kesalahan sintaks) adalah permasalahan pertama dalam proses, dan analisis untuk mendeteksi ini sudah ditanam dalam bahasa simulasi umumnya. Kesulitan berikutnya yang dihadapi adalah perbaikan kesalahan selama eksekusi kode. Analisis bahasa simulasi umumnya tidak sesuai secara total dengan permasalahan ini. Setelah menemukan kesalahan seperti ini, program berhenti dan tidak memberikan alasan dalam bentuk logika model kenapa program berhenti.

- Penurunan variabel acak. Untuk kebanyakan simulasi probabilistik, kemampuan mengekstrak sampel acak dari distribusi probabilitas tertentu sangat penting. Bahasa simulasi melakukannya dengan mudah.
- Pengumpulan statistik. Penjalanan model simulasi tanpa mengumpulkan data ukuran kinerja sistem sama saja dengan tidak melakukan pengamatan pada sistem dunia nyata yang sedang berlangsung. Pengamat ada selama operasi sistem dunia nyata tetapi tidak mengamati dan mencatat apa yang terjadi. Bahasa simulasi harus memungkinkan pengguna dengan mudah menspesifikasikan beragam statistik yang dikumpulkan selama eksekusi model. Juga untuk membantu interpretasi output simulasi, kemampuan penggambaran grafik dan inferensi statistik diperlukan.
- Disain percobaan. Karena analisis simulasi bersifat deskriptif, kesuksesan aplikasinya tergantung pada percobaan model. Rancangan percobaan efektif dan efisien benar-benar meningkatkan kualitas solusi yang didapatkan dari model simulasi.
- Animasi grafis dan output dinamis. Kemampuan menggunakan bahasa simulasi pada mikrokomputer memungkinkan kemampuan grafis mesin ini untuk mengilustrasikan penjalanan mode simulasi atau outputnya. Ilustrasi objek yang mengalir melalui elemen model disebut sebagai animasi. Animasi biasanya menggunakan monitor berwarna dan dengan mudah mengenali simbol objek dan elemen model. Dengan mengamati aliran seperti itu, analisis dapat memperhatikan penyebaba permasalahan operasi dan dapat memperbaikinya. Animasi model akan memperlambat eksekusi model. Oleh akrena itu, animasi biasanya hanya dilakukan pada mikrokomputer cepat dengan memori besar.

**Pemilihan Bahasa Simulasi**

Beberapa hal yang perlu diperhatikan dalam pemilihan bahasa simulasi adalah kemudahan untuk dipelajari, kemudahan menjelaskan pada orang yang bukan teknik, biaya, kode standar untuk semua komputer dan cakupan permasalahan yang dapat ditangani oleh bahasa. Pada umumnya, semakin mirip

elemen bahasa simulasi dengan elemen dunia nyata, semakin mudah elemen itu dipelajari. Kemudahan menjelaskan fungsi bahasa simulasi ke manajer yang mengeluarkan dana untuk pembelian perangkat lunak dan yang tidak memahami secara teknis juga digunakan dalam memilih bahasa simulasi. Ada beberapa sumber di internet yang bisa digunakan untuk mengikuti perkembangan bahasa simulasi. Ada banyak sekali bahasa simulasi yang ada dan dapat dikembangkan untuk membuat program dalam suatu model perhitungan matematis. Semua memiliki kelebihan dan kelemahan masing masing. Beberapa bahasa simulasi yang terkenal dapat dilihat pada tabel 2.1.

Tabel 1 Software Libraries

| **Software Libraries** | |
|---|---|
| Bahasa Simulasi | Penjelesan |
| The Numerical Algorithms Group Ltd | Arsip *Algoritma* numerik |
| Netlib | Arsip *Algoritma* numeric |
| C++SIM | C++ libreries untuk simulasi sistem kejadian diskrit |
| Mathtools | Suatu portal yang menyediakan akses gratis untuk MATLAB, Excel, C, C++, Fortran and *Java*. |
| CSIM18 | Mesquite CSIM berorientasi proses, general purpose simulation toolkit yang ditulis dengan fungsi umum bahasa C. The toolkit memungkinkan programmer menciptakandan mengimplementasikan model orientasi proses dan simulasi kejadian diskrit. |
| Warped | WARPED adalah domain umum Time Warp simulation kernel ditulis dalam C++. The distribution includes a plug-in sequential kernel to support comparative analysis to parallel executions. Primary development has been on Linux-based Pentium PCs, Sun Workstations, a 4-processor Sun SparcCenter 1000, and the Intel Paragon. |
| SimTools, Version 2.7 | Review beberapa alat simulasi |

| Bahasa Simulasi | Penjelasan |
|---|---|
| OpEMCSS | The Operational Evaluation Modelling for Context-Sensitive Systems (OpEMCSS) adalah tambahan Extend simulation environment. Memungkinkan bagi pemakai untuk merepresentasikan sistem adaptif kompleks relatif lebih mudah. |
| baseSim | iBright adalah evolusi baseSim Simulation Components (pertama dikembangkan oleh solutionsBase, sekarang oleh defunct) adalah kelompok Visual Components dirancang sebagai komplemen dan perluasan fungsi Borland Delphi v.4.0/5.0. Menyediakan alat untuk pemodelan simulasi sederhana maupun kejadian diskrit kompleks. |
| SSS | A library (coded in C) untuk simulasi sistem kejadian diskrit oleh M. A. Pollatschek |
| TomasWeb | TomasWeb memberikan simulasi orientasi objek yang diimplementasikan dalam Delphi 5 and 6. berbasis pendekatan orientasi proses. metode pendiskripsian, dimana beberapa kejadian (perubahan status) dikombinasikan kedalam proses tunggal. Oleh karenanya, tools ini mendukung simulasi orientasi proses. TomasWeb dikembangkan terutama untuk pendidikan dan penelitian. Perangkat lunak ini gratis, tapi memerlukan Borland's Delphi. |
| HighMAST object-oriented simulation library | HighPoint Software Systems menawarkan simulasi orientasi objek. Ditulis dalam C# , dan terdiri dari 200+ classes, 70+ interfaces. HighMAST framework dibangun sebagai open architecture library on Microsoft's capable .NET technology base. |
| Code by Law and Kelton | Contoh-contoh Code dalam C and FORTRAN dari buku "Simulation Modelling and Analysis, by A.V. Law and W.D. Kelton |

| Bahasa Simulasi | Penjelesan |
|---|---|
| Dex | Dex, the Dynamic Experimentation toolkit, bertujuan utnuk menyediakan kecepatan, fleksibel dan mudah digunakan untuk pengembangan, analisis dan visualisasi simulasi multi dinamis. Kernel and utility classes terdiri dari kerangka kerja yang dapat digunakan dalam C++ atau kombinasi dengan bahasa DEx, bahasa khusus domain berbasis C++ dirancang untuk percepatan prootip. Dikompile dalam Linux dan tersedia gratis di internet. |
| *Java*SIM | Versi *Java* C++SIM |
| Simulations in *Java* | Arsip Sim *Java* |

*Sumber : Siswoyo (2010)*

**Elemen Simulasi**

Adapun elemen-elemen yang ada pada proses simulasi adalah:

- Analisis simulasi merupakan teknik pemodelan deskriptif, karena itu tidak ada formulasi permasalahan dan langkah penyelesaian eksplisit yang merupakan bagian integral dari model optimasi.
- Meskipun tidak ada langkah eksplisit, paling tidak kita dapat menggunakan elemen simulasi berikut dalam perancangan model simulasi:
    - ✓ Formulasi permasalahan
    - ✓ Pengumpulan dan analisis data
    - ✓ Pengembangan model
    - ✓ Verifikasi dan validasi model
    - ✓ Percobaan dan optimasi model
    - ✓ Implementasi hasil simulasi

Formulasi Masalah merupakan suatu langkah yang sangat penting dalam perancangan model simulasi. Formulasi masalah yang tidak tepat tidak akan mungkin menghasilkan model yang tepat (akurat). Formulasi masalah merupakan suatu kegiatan untuk memilih satu permasalahan yang dianggap paling penting

untuk diselesaikan saat itu dari sekian banyak permasalahan. Hal-hal berikut diungkapkan dalam formulasi masalah:

- Identifikasi keputusan dan variabel tidak dapat dikontrol
- Spesifikasi pembatas variabel keputusan
- Mendefinisikan ukuran kinerja sistem dan fungsi tujuan
- Mengembangkan model struktur awal yang menghubungkan variabel sistem dan ukuran kinerja

**Variabel dan Pembatas**

Setelah programmer memutuskan permasalahan yang akan diselesaikan dalam model simulasi, maka langkah selanjutnya adalah menentukan variable yang mendefinisikan sistem dan outputnya. Variabel dapat dikategorikan sebagai variabel eksogenus dan endogenus. Variabel eksogenus kadang-kadang disebut juga sebagai variabel input sedangkan variabel endogenus disebut juga sebagai variabel output.

- Variabel eksogenus : ada di luar sistem dan tidak terikat dengan model.
- Variabel endogenus : ada dalam sistem dan merupakan fungsi variabel eksogenus. Variabel eksogenus terdiri dari variabel yang dapat dikontrol dan tidak dapat dikontrol.

Variabel eksogenus yang dapat dikontrol dapat dimanipulasi pengambil keputusan, sedangkan variabel eksogenus yang tidak dapat dikontrol tidak dapat dimanipulasi pengambil keputusan. Jadi untuk menyelesaikan permasalahan yang dihadapi manajemen, menggunakan model simulasi, mereka hanya dapat memanipulasi variabel eksogenus yang dapat dikontrol. Penentuan variabel sebagai terkontrol atau tidak tergantung dari kemampuan pengambil keputusan mengendalikan sumber daya. Variabel eksogenus yang dapat dikontrol kadang-kadang disebut dengan variabel keputusan. Variabel eksogenus yang tidak dapat dikontrol kadang-kadang disebut dengan parameter sistem. Sumber daya yang membatasi dalam mencapai tujuan juga harus didefinisikan dengan tepat. Permasalahan timbul karena adanya batasan-batasan dalam sistem.

**Pengukuran Kinerja Sistem dan Fungsi Tujuan**

Ukuran kinerja sistem bisa lebih dari satu. Pengoptimalan salah satunya bisa saling bertentangan dengan ukuran kinerja lainnya. Pengambil keputusan harus dapat memilih ukuran kinerja yang paling tepat untuk tujuan optimasi. Detail model tergantung dari tujuan pengembangan model dan kontribusi marjinal penambahan detail. Kompleksitas model ditentukan secara subjektif, coba-coba yang diturunkan dari perkiraan biaya marjinal yang harus dikeluurkan untuk mendapatkan data dan relasi dalam model terhadap akurasi tambahan yang dapat diberikan.

**Pengumpulan Data**

Data diperlukan untuk percobaan model. Verifikasi dan validasi model dapat dilakukan dengan adanya data. Dalam validasi dan verifikasi, analisis menguji seberapa dekat model yang dibuat dapat meniru sistem aslinya dengan membandingkan output model dengan kinerja sistem. Output akan diperoleh jika simulasi dijalankan untuk data tertentu. data bisa diperoleh dengan pengamatan dan pelaporan pribadi, atau dengan membangkitkan bilangan acak jika data historisnya sudah ada. Cara kedua ini khususnya digunakan untuk model probabilistik. Ukuran sampel tergantung dari biaya yang bersedia dikeluarkan untuk keakuratan tertentu.

**Pengembangan Model**

Pemahaman yang baik akan sistem sebenarnya sangat diperlukan dalam membentuk model dan merupakan hal yang sulit juga untuk dilakukan. Tidak ada pendekatan yang baku dalam membentuk model. Ada dua pendekatan yang dapat kita gunakan, yaitu pendekatan aliran fisik dan perubahan status. Dalam pendekatan aliran fisik, pemrosesan atau perpindahan entiti secara fisik ditunjukkan dalam model. Keberadaan entiti ini dilacak dalam sistem selama proses penjalanan simulasi, untuk mengetahui entiti sedang diproses dimana dan pencabangan aturan keputusan untuk menentukan rutenya. Diagram alur entiti dan pemrosesan elemen sistem memberikan representasi sistem darimana model dan pemrograman komputernya dikembangkan. Dalam pendekatan perubahan

status, kita memerlukan variabel status (termasuk dalam klasifikasi variabel endogenus) dan kejadian. Kita dapat bedakan model yang akan kita bangun ke dalam model konseptual, logika dan simulasi. Penggolongan ini akan memudahkan dalam membentuk model simulasinya.

- Model konseptual : menggambarkan sistem secara konsep, dapat secara verbal atau menggunakan grafik.
- Model logika : menerjemahkan model konseptual ke dalam bentuk suatu diagram alur atau *Algoritma*.
- Model simulasi : menerjemahkan model logika ke dalam program computer

**Verifikasi dan validasi Model**

Verifikasi dan validasi dilakukan untuk ketiga model (konseptual, logika dan simulasi).

- Model valid jika ukuran outputnya sangat dekat dengan ukuran sistem nyata yang sesuai.
- Validasi menunjukkan seberapa akurat model memprediksi kejadian mendatang.
- Prediksi kejadian masa mendatang harus didahului prediksi nilai variabel input.
- Percobaan model dan optimasi
- Analisis output : statistik
- Analisis output : analisis terminating dan analisis keseimbangan.
- Analisis terminating : penjalanan model diakhiri dengan beberapa kejadian spesifik.
- Percobaan model :
    - ✓ desain percobaan klasik (ANOVA)
    - ✓ Metodologi respon permukaan *(response surface methodology*)

## 2.3 *Algoritma*

Menurut Ngoen (2008) kata *Algoritma* berasal dari latinisasi nama seorang ahli matematika dari Uzbekistan Al_Khawārizmi (hidup sekitar abad ke-9),

sebagaimana tercantum pada terjemahan karyanya dalam bahasa latin dari abad ke-12 "Algorithmi de numero Indorum". Pada awalnya kata algoritma adalah istilah yang merujuk kepada aturan-aturan aritmetis untuk menyelesaikan persoalan dengan menggunakan bilangan numerik arab. Pada abad ke-18, istilah ini berkembang menjadi *Algoritma*, yang mencakup semua prosedur atau urutan langkah yang jelas dan diperlukan untuk menyelesaikan suatu permasalahan. Masalah timbul pada saat akan menuangkan bagaimana proses yang harus dilalui dalam suatu/sebuah sistem (program) bagi komputer sehingga pada saat eksekusinya, komputer dapat bekerja seperti yang diharapkan. Programer komputer akan lebih nyaman menuangkan prosedur komputasinya atau urutan langkah proses dengan terlebih dahulu membuat gambaran (diagram alur) diatas kertas.

*Algoritma* memegang peranan penting dalam bidang pemrograman. Sebegitu pentingnya suatu *Algoritma*, sehingga perlu dipahami konsep dasar *Algoritma*. Apalagi untuk seorang programer, tentu diperlukan suatu *Algoritma* sehingga dapat membuat program yang lebih efektif dan efisien. Bagi kebanyakan orang, *Algoritma* sangat membantu dalam memahami konsep logika pemrograman. *Algoritma* adalah kumpulan instruksi yang dibuat secara jelas untuk menunjukan langkah-langkah penyelesaian suatu masalah. Pada umumnya *Algoritma* kurang lebih sama dengan suatu prosedur yang sering dilakukan setiap hari, misalnya prosedur untuk mengganti ban bocor/pecah, prosedur pemakaian telepon umum, prosedur membuat kue dan lain-lain. Dalam bidang komputer, misalnya EDP (Elektronik Data Processing) atau MIS (Management Information System), *Algoritma* sering dimanfaatkan untuk menyelesaikan suatu masalah atau untuk proses pengambilan keputusan. Seorang sistem analisis (analisyst system) tentunya menggunakan *Algoritma* untuk merancang suatu sistem. Bagi seorang programer, *Algoritma* digunakan untuk membuat modul-modul program.Guna memahami suatu *Algoritma*, harus dimiliki pengetahuan dasar matematika karena pada dasarnya *Algoritma* lahir dari konsep logika matematika. Disini yang perlu dilatih adalah kemampuan logikanya agar benar-benar bisa menyusun langkah-langkah penyelesaian masalah dengan baik.

Konsep Dasar *Algoritma*

*Algoritma* adalah kumpulan instruksi/perintah yang dibuat secara jelas dan sistematis berdasarkan urutan yang logis (logika) untuk penyelesaian suatu masalah (French,C.S.,1984) menyatakan sejumlah konsep yang mempunyai relevansi dengan masalah rancangan program yaitu kemampuan komputer, kesulitan dan ketepatan. Penerapan dari konsep tersebut biasanya digunakan dalam rancangan *Algoritma*. Dalam merancang sebuah *Algoritma*, Fletcher (1991) memberikan beberapa cara atau metode yaitu kumpulan perintah, ekspresi, tabel instruksi, program komputer, kode semu dan flow chart, sedangkan Knuth (1973) menyarankan *Algoritma* fundamental. Untuk keperluan matematika dan program komputer metode yang sering digunakan yaitu:

1. Diagram Alir (Flow Chart)
2. Kode Semu (Pseudo Code)
3. *Algoritma* Fundamental

Knuth (1973) menyatakan 5 komponen utama dalam *Algoritma* yaitu finiteness, definiteness, input, output dan effectiveness. Sehingga dalam merancang sebuah *Algoritma* ada 3 (tiga) komponen yang harus ada yaitu:

1. Komponen masukan (input)

   Komponen ini biasanya terdiri dari pemilihan variable, jenis variable, tipe variable, konstanta dan parameter (dalam fungsi).
2. Komponen keluaran (output)

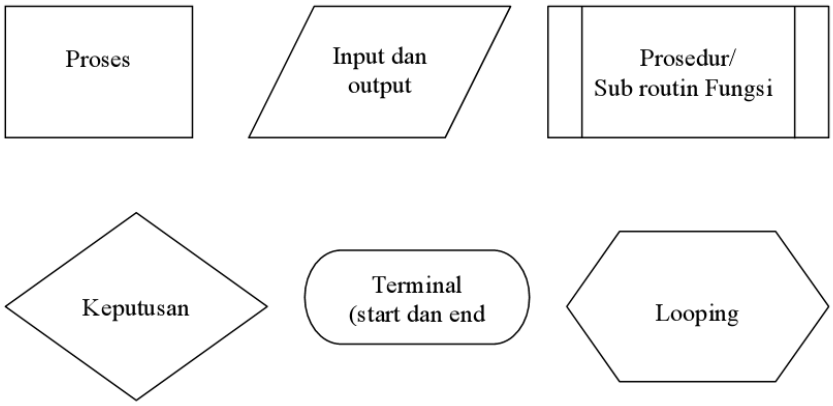
   Komponen ini merupakan tujuan dari perancangan *Algoritma* dan program. Permasalahan yang diselesaikan dalam *Algoritma* dan program harus ditampilkan dalam komponen keluaran. Karakteristik keluaran yang baik adalah benar (menjawab) permasalahan dan tampilan yang ramah (Frendly).
3. Komponen proses (processing)

   Komponen ini merupakan bagian utama dan terpenting dalam merancang sebuah *Algoritma*. Dalam bagian ini terdapat logika masalah, logika *Algoritma*

(sintaksis dan semantik), rumusan, metode (rekursi, perbandingan, penggabungan, pengurangan dan lain-lain).

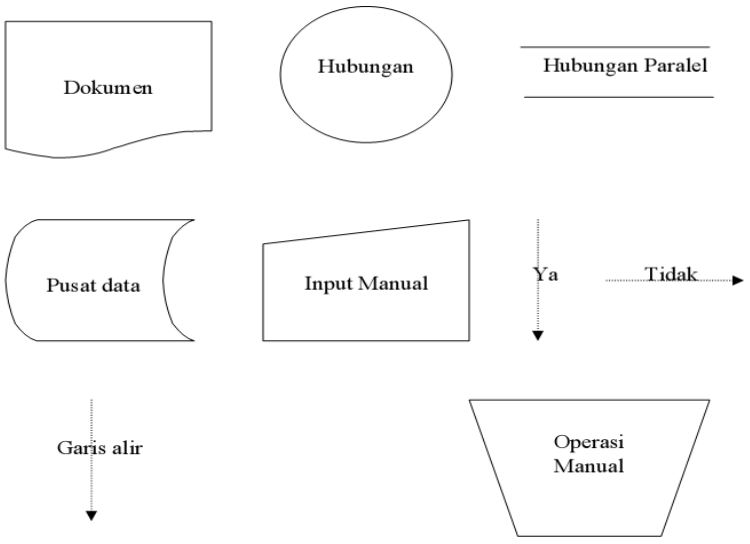
Diagram Alir

Jogiyanto dalam Hapsari (2012: 19-20) menjelaskan *flowchart* adalah suatu bagan (chart) yang menggambarkan arus logika dari data yang akan diproses dalam suatu program dari awal sampai akhir. *Algoritma* ini menggunakan sejumlah simbol untuk menyatakan kegiatan-kegiatan secara keseluruhan. Simbol dan artinya dalam diagram alir ditunjukkan pada gambar di bawah ini 7 dan 8.




*Sumber : TuThev (2013)*

Gambar 7 Simbol dan arti Diagram Alir




*Sumber : TuThev (2013)*

Gambar 8 Simbol dan arti Diagram Alir (lanjutan)

Kode Semu

TuThey (2013) menjelaskan dalam merancang sebuah *Algoritma* menggunakan kode semu, komponen-komponen input, output dan proses harus terdefinisi secara jelas. Disamping itu beberapa ketentuan dan aturan pendefinisian memang secara baku tidak ditemukan dalam beberapa buku literatur, namun aturan-aturan yang di ajukan dibawah ini akan membantu mempermudah perancangan *Algoritma* dan evaluasi serta analisis *Algoritma*.

Aturan-aturan tersebut :

1. Kode semu harus dimulai dengan judul.
   Aturan ini secara mudah dapat dimengerti fungsi dan manfaatnya. Judul harus dapat menjelaskan spesifikasi masalah yang dirancang *Algoritmanya*. Penulisannya dapat dengan huruf kapital semuanya atau tidak.
2. Kode semu harus ditulis dengan nomor yang menunjukkan urutan-urutan langkah-langkah dalam *Algoritma*.
3. Pendeklarasian variabel, konstanta, parameter, rumus dan pernyataan harus sederhana

Contoh. 1.1

Bandingkan kedua *Algoritma* ini.

Masalah : Mencari akar-akar persamaan non linear dengan metode bagi dua.

A. Kode semu yang dirancang tidak menggunakan aturan.

Penyelesaian:

1. Formulasikan sebuah persaman non linier
2. Cari nilai bawah $x_b$ yang menyebabkan nilai fungsi $f(x_b)$ positif atau negatif, kemudian cari nilai atas $x_a$ yang menyebabkan nilai fungsi $f(x_a)$ berlawanan (positif negatif) dengan nilai bawah.
3. Bandingkan nilai $f(x_b)$ dengan $f(x_a)$
4. Jika $f(x_b).f(x_a) > 0$ maka ulangi langkah 2
5. Jika $f(x_b).f(x_a) < 0$ maka bagi dua interval $x_b$ dengan $x_a$ . Ulangi langkah 3
6. Jika $f(x_b).f(x_a) = 0$ maka iterasi berhenti, akar-akar persamaan x diperoleh

B. Kode semu yang dirancang menggunakan aturan

Penyelesaian:

*Algoritma* Bagi Dua

1. Formulasikan masalah f(x)
2. Cari taksiran bawah ($x_b$) dan taksiran atas ($x_a$)
3. Bandingkan dan evaluasi, jika $f(x_b).f(x_a) > 0$ maka ulangi langkah 2
4. Jika $f(x_b).f(x_a) < 0$ maka bagi dua interval dengan $(x_b + x_a) / 2$, kembali bandingkan

   dan evaluasi.
5. Jika $f(x_b).f(x_a) = 0$ maka iterasi berhenti, akar-akar persamaan x diperoleh

*Algoritma* Fundamental

Knuth (1973) menyajikan format *Algoritma* yang dapat digunakan secara bebas untuk berbagai bahasa pemrograman, artinya dapat dengan mudah diimplementasikan menggunakan Pascal, C, Fortran, PL atau BASIC. Secara umum notasi dan aturan yang digunakan sebagai berikut :

1. Nama/judul *Algoritma* harus ditulis dengan huruf kapital

   Contoh : *Algoritma* BAGI DUA
2. Berikan komentar dan penjelasan pendahuluan. Penjelasan secara singkat tentang *Algoritma*.

   Contoh : *Algoritma* BAGI DUA

   Mencari akar persamaan dengan taksiran pertama $x_b$ dan $x_a$
3. Langkah-langkah.

   *Algoritma* tersusun menurut nomor langkah-langkah diawali dengan '[.....]' untuk memberikan keterangan tentang langkah tersebut.

   Contoh : 1. [formulasikan f(x)]
4. Komentar (*comments*). Komentar untuk penjelasan bagi pembaca ditulis dengan tanda (.....)
5. Pernyataan dan struktur Kontrol

   Pernyataan adalah perintah yang terdapat didalam algoritm, sedangkan struktur kontrol untuk mengendalikan pernyataan yang digunakan. Pernyataan dan struktur kontrol terdiri dari :

   a. Perintah pemberian nilai menggunakan $\leftarrow\rightarrow$, $\leftarrow$

      Contoh : A$\leftarrow$B (artinya A = B)

X ← 0 (artinya x bernilai 0)

X ←→Y (artinya x dan y saling tukar)

b. Pernyataan IF

IF kondisi

Then.....

IF kondisi

Then.....

.....

else.....

c. Pernyataan Case

Perintah ini untuk menyeleksi pilihan tertentu. Bentuknya :

Select Case (ekspresi)

Case nilai 1 :

Case nilai 2 :

…

Case nilai n :

Default :

d. Pernyataan Repeat

Perintah pengulangan digunakan dengan bentuk :

Repeat for indeks = barisan nilai

Repeat while ekspresi logika

Repeat for indeks = barisan nilai while ekspresi logika

e. Pernyataan Goto dan Exitloop

Perintah untuk melompat ke langkah yang telah ditentukan dan keluar dari pengulangan.

Bentuknya :

Goto step.....

Exitloop

f. Pernyataan Exit

Perintah untuk menghentikan *Algoritma*.

6. Nama-nama variabel harus ditulis dengan huruf besar

7. Input dan output

   Data dapat dimasukkan melalui variabel dengan pernyataan READ dengan bentuk: Read : NAMA VARIABEL

   Untuk mencetak pesan-pesan/tulisan (diapit dengan tanda kutip) dan juga variabel

   digunakan pernyataan : Write : tulisan dan atau nama variabel

8. Prosedur

   Bentuk prosedur digunakan untuk modul *Algoritma* yang berdiri sendiri untuk menyelesaikan masalah tertentu. Pemakaian prosedur untuk masalah sederhana, sedangkan *Algoritma* untuk masalah umum. Bentuk yang digunakan :

   Procedure nama prosedur

9. Fungsi

   Sama dengan prosedur menggunakan bentuk : Function nama fungsi

## 2.4 *Java*

*Java* sebagai bahasa pemrograman tingkat tinggi yang berorientasi objek, atau lazim disebut dengan istilah Object Oriented Programming (OOP) . *Java* awalnya dibuat oleh James Gosling saat masih bergabung di Sun Microsystems saat ini merupakan bagian dari Oracle dan dirilis tahun 1995. Bahasa ini banyak mengadopsi sintaksis yang terdapat pada C dan C++ namun dengan sintaksis model objek yang lebih sederhana serta dukungan rutin-rutin aras bawah yang minimal (Purnomo, 2007). Aplikasi-aplikasi berbasis *Java* umumnya dikompilasi ke dalam p-code (*bytecode*) dan dapat dijalankan pada berbagai Mesin Virtual *Java* (JVM). *Java* merupakan bahasa pemrograman yang bersifat umum/non-spesifik (*general purpose*), dan secara khusus didisain untuk memanfaatkan dependensi implementasi seminimal mungkin. Karena fungsionalitasnya yang memungkinkan aplikasi *Java* mampu berjalan di beberapa platform sistem operasi yang berbeda. Saat ini *Java* merupakan bahasa pemrograman yang paling populer digunakan, dan secara luas dimanfaatkan dalam pengembangan berbagai jenis perangkat lunak aplikasi ataupun aplikasi berbasis web.

Versi Awal

Bedasarkan (http://id.wikipedia.org/wiki/Java) Versi awal *Java* pada tahun 1996 sudah merupakan versi release sehingga dinamakan *Java* Versi 1.0. *Java* versi ini menyertakan banyak paket standar awal yang terus dikembangkan pada versi selanjutnya:

- *Java*.lang: Peruntukan kelas elemen-elemen dasar.
- *Java*.io: Peruntukan kelas *input* dan *output*, termasuk penggunaan berkas.
- *Java*.util: Peruntukan kelas pelengkap seperti kelas struktur data dan kelas kelas penanggalan.
- *Java*.net: Peruntukan kelas TCP/IP, yang memungkinkan berkomunikasi dengan komputer lain menggunakan jaringan TCP/IP.
- *Java*.awt: Kelas dasar untuk aplikasi antarmuka dengan pengguna (GUI)
- *Java*.applet: Kelas dasar aplikasi antar muka untuk diterapkan pada penjelajah web.

Karakeristik dan Kelebihan *Java*

Purnomo (2007 : 3) menjelaskan karakteristik dan kelebihan bahasa pemrograman *Java* sebagai berikut.

- Sederhana

  Java merupakan suatu bahasa sederhana yang memiliki sintaks yang sama dengan bahasa pemrograman C++, namun dengan memperbaiki beberapa kekurangan dari C++, seperti mengurangi kompleksitas beberapa fitur, menambah fungsi, serta menghilangkan beberapa hal yang menyebabkan ketidakstabilan sistem pada C++.
- *Multiplatform*. Kelebihan utama dari *Java* ialah dapat dijalankan di beberapa *platform* / sistem operasi komputer, sesuai dengan prinsip *tulis sekali, jalankan di mana saja*. Dengan kelebihan ini pemrogram cukup menulis sebuah program *Java* dan dikompilasi (diubah, dari bahasa yang dimengerti manusia menjadi bahasa mesin / *bytecode*) sekali lalu hasilnya dapat dijalankan di atas beberapa platform tanpa perubahan. Kelebihan ini memungkinkan sebuah program berbasis *Java* dikerjakan diatas operating system Linux tetapi dijalankan dengan baik di atas Microsoft Windows.

Platform yang didukung sampai saat ini adalah Microsoft Windows, Linux, Mac OS dan Sun Solaris. Penyebabnya adalah setiap sistem operasi menggunakan programnya sendiri-sendiri (yang dapat diunduh dari situs *Java*) untuk meninterpretasikan *bytecode* tersebut.

- Berorientasi Objek

  Bahasa pemrograman saat ini mengarah ke bahasa pemrograman berorientasi objek. Rancangan pada data (objek) dan antarmuka.

- Kuat

  Program yang dibuat dengan java dapat dipercaya dalam berbagai hal, karena java banyak menekankan pada pengecekan awal untuk menghindari kemungkinan terjadi masalah, pengecekan pada saat *run-time*, dan mengurangi kemungkinan timbulnya masalah.

- Aman

  *Java* memungkinakan untuk membuat program yang bebas virus dan sistem yang bebas dari kerusakan, karena java membuat sistem yang mekanisme keamanannya benar benar kuat.

- *Portable*

  Tidak seperti pada C dan C++, spesifikasi java tidak terdapat aspek yang bergantung pada lingkungan implementasi. Ukuran tipe data primitif telah ditentukan sejak awal. Missal ,“int” selalu berarti sebuah integer 32 bit dll.

- *MultiThread*

  *Multithreading* adalah kemampuan suatu program komputer untuk melakukan beberapa pekerjaan sekaligus. *Thread* dalam *java* juga memiliki kemampuan untuk memanfaatkan kelebihan dari multiprosesor jika sistem operasi yang digunakan mendukung multiprosesor

- *Interpreter*

  *Interpreter java* dapat mengeksekusi kode byte java secara langsung pada setiap mesin yang terdapat interpreter dan setiap run-time *java.*

- Perpustakaan Kelas Yang Lengkap, *Java* terkenal dengan kelengkapan *library*/perpustakaan (kumpulan program program yang disertakan dalam pemrograman *Java*) yang sangat memudahkan dalam penggunaan oleh para

pemrogram untuk membangun aplikasinya. Kelengkapan perpustakaan ini ditambah dengan keberadaan komunitas *Java* yang besar yang terus menerus membuat perpustakaan-perpustakaan baru untuk melingkupi seluruh kebutuhan pembangunan aplikasi.

- Bergaya C++, memiliki sintaks seperti bahasa pemrograman C++ sehingga menarik banyak pemrogram C++ untuk pindah ke *Java*. Saat ini pengguna *Java* sangat banyak, sebagian besar adalah pemrogram C++ yang pindah ke *Java*. Universitas-universitas di Amerika Serikat juga mulai berpindah dengan mengajarkan *Java* kepada murid-murid yang baru karena lebih mudah dipahami oleh murid dan dapat berguna juga bagi mereka yang bukan mengambil jurusan komputer.
- Pengumpulan sampah otomatis, memiliki fasilitas pengaturan penggunaan memori sehingga para pemrogram tidak perlu melakukan pengaturan memori secara langsung (seperti halnya dalam bahasa C++ yang dipakai secara luas).

Kekurangan *Java*

Kekurangan bahasa pemrograman *Java* menurut (http://id.wikipedia.org/wiki/Java) adalah :

- Tulis sekali, jalankan di mana saja - Masih ada beberapa hal yang tidak kompatibel antara *platform* satu dengan *platform* lain. Untuk J2SE, misalnya *SWT-AWT bridge* yang sampai sekarang tidak berfungsi pada Mac OS X.
- Mudah didekompilasi. Dekompilasi adalah proses membalikkan dari kode jadi menjadi kode sumber. Ini dimungkinkan karena kode jadi *Java* merupakan *bytecode* yang menyimpan banyak atribut bahasa tingkat tinggi, seperti nama-nama kelas, metode, dan tipe data. Hal yang sama juga terjadi pada Microsoft .NET Platform. Dengan demikian, *Algoritma* yang digunakan program akan lebih sulit disembunyikan dan mudah dibajak/di*reverse-engineer*.
- Penggunaan memori yang banyak. Penggunaan memori untuk program berbasis *Java* jauh lebih besar daripada bahasa tingkat tinggi generasi

sebelumnya seperti C/C++ dan Pascal (lebih spesifik lagi, Delphi dan Object Pascal). Biasanya ini bukan merupakan masalah bagi pihak yang menggunakan teknologi terbaru (karena trend memori terpasang makin murah), tetapi menjadi masalah bagi mereka yang masih harus berkutat dengan mesin komputer berumur lebih dari 4 tahun.

Contoh Program Sederhana

Contoh program Hello World yang ditulis menggunakan bahasa pemrograman *Java* adalah sebagai berikut:

```
// Outputs "Hello, world!" and then exits
public class HelloWorld {
   public static void main(String args[]) {
System.out.println("Hello, world!");
 }
```

Tahap Kompilasi

Pada (http://id.wikipedia.org/wiki/Java) dijelaskan Tahap-tahap kompilasi pada bahasa pemrograman *Java* adalah :

1. Tulis / Ubah. Pemrogram menulis program dan menyimpannya di media dalam bentuk berkas '.*Java*'.
2. Kompilasi. Pengkompilasi membentuk *bytecodes* dari program menjadi bentuk berkas '.class'.
3. Muat. Pemuat kelas memuat *bytecodes* ke memori.
4. Verifikasi. Peng-verifikasi memastikan *bytecodes* tidak mengganggu sistem keamanan *Java*.
5. Jalankan. Penerjemah menerjemahkan *bytecodes* ke bahasa mesin.tidak bisa di pakai‘’

Karena *java* merupakan bahasa pemrograman tingkat tinggi, maka untuk mempermudah proses pembuatan program, java mempunyai *Integrated Development Environment* guna mempermudah programmer dalam membuat aplikasi.

*Integrated Development Environment*

Dalam (http://id.wikipedia.org/wiki/Java) dijelaskan *Integrated Development Enviroment* merupakan aplikasi pembantu yang memudahkan kita dalam menulis *Java*script. Banyak pihak telah membuat IDE (*Integrated Development Environment* - Lingkungan Pengembangan Terintegrasi) untuk *Java*. Yang populer saat ini antara lain:

- Dr. *Java*, program gratis yang dikembangkan oleh Universitas Rice, Amerika Serikat
- BlueJ, program gratis yang dikembangkan oleh Universitas Monash, Australia
- NetBeans (open source- Common Development and Distribution License (CDDL))

NetBeans disponsori Sun Microsystems, dan versi terkininya memilki Matisse, sebuah GUI Editor yang menurut pendapat umum merupakan yang terbaik.

- Eclipse JDT (open source- Eclipse Public License)

Eclipse dibuat dari kerja sama antara perusahaan-perusahaan anggota 'Eclipse Foundation' (beserta individu-individu lain). Banyak nama besar yang ikut dalam 'Eclipse Foundation', termasuk IBM, BEA, Intel, Nokia, Borland. Eclipse bersaing langsung dengan Netbeans IDE. Plugin tambahan pada Eclipse jauh lebih banyak dan bervariasi dibandingkan IDE lainnya.

- IntelliJ IDEA (commercial, free 30-day trial)
- Oracle JDeveloper (free)
- Xinox JCreator (ada versi berbayar maupun free)

JCreator ditulis dalam C/C++ sehingga lebih cepat (dan menggunakan memori lebih sedikit) dari kebanyakan.