BAB 2
LANDASAN TEORI

2.1 Kecerdasan Buatan

Kecerdasan buatan berasal dari Bahasa Inggris "*Artificial Intelligence*" atau disingkat AI, yang terdiri dari dua kata yaitu *intelligence* adalah kata sifat yang berarti cerdas, sedangkan *artificial* artinya buatan. Kecerdasan buatan yang dimaksud di sini merujuk pada mesin yang mampu berfikir, menimbang tindakan yang akan diambil, dan mampu mengambil keputusan seperti yang dilakukan oleh manusia. Berikut adalah definisi kecerdasan buatan menurut beberapa ahli:

a. Alan Turing, ahli matematika berkembangsaan Inggris yang dijuluki bapak komputer modern dan pembongkar sandi Nazi dalam era Perang Dunia II 1950, menetapkan definisi AI : "Jika komputer tidak dapat dibedakan dengan manusia saat berbincang melalui terminal komputer, maka bisa dikatakan itu cerdas, mempunyai kecerdasan".
b. John McCarthy dari Stanford mendefinisikan kecerdasan sebagai "kemampuan mencapai sukses dalam menyelesaikan suatu permasalahan".
c. Herbert Alexander Simon (2001) : "AI merupakan kawasan penelitian, aplikasi, dan instruksi yang terkait dengan pemrograman komputer untuk melakukan sesuatu hal yang dalam pandangan manusia adalah cerdas".
d. Rich and Knight (1991): "AI merupakan sebuah studi tentang bagaimana membuat komputer melakukan hal-hal yang pada saat ini dapat dilakukan lebih baik oleh manusia".
e. Encyclopedia Britannica: "AI merupakan cabang ilmu komputer yang dalam merepresentasikan pengetahuan lebih banyak menggunakan bentuk simbol-simbol daripada bilangan dan memproses informasi berdasarkan metode heuristis atau dengan berdasarkan sejumlah aturan".
f. Winston dan Prendergast (1984), tujuan kecerdasan buatan adalah membuat mesin menjadi lebih pintar (tujuan utama), memahami apa itu kecerdasan (tujuan ilmiah), dan membuat mesin lebih bermanfaat (tujuan *enterpreneurial*).

Berdasarkan definisi tersebut, maka kecerdasan buatan menawarkan media maupun uji teori tentang kecerdasan. Teori-teori itu nantinya dapat dinyatakan dalam bahasa pemrograman dan eksekusinya dapat dibuktikan pada komputer nyata.

Program konvensional hanya dapat menyelesaikan persoalan yang diprogram secara spesifik. Jika ada informasi baru, sebuah program konvensional harus diubah untuk menyesuaikan diri dengan informasi baru tersebut. Hal ini tidak hanya menyebabkan boros waktu, namun juga dapat menyebabkan terjadinya *error*. Sebaliknya, kecerdasan buatan memungkinkan komputer untuk berfikir atau menalar dan menirukan proses belajar manusia sehingga informasi baru dapat diserap sebagai pengetahuan, pengalaman dan proses pembelajaran serta dapat diinginakan sebagai acuan di masa-masa yang akan datang.

Dari sini dapat dikatakan bahwa cerdas adalah memiliki pengetahuan, pengalaman, dan penalaran untuk membuat keputusan dan mengambil tindakan. Jadi, agar mesin bisa cerdas (bertindak sperti manusia), maka harus diberi bekal pengetahuan dan bekal kemampuan untuk menalar. (T. Sutojo, 2011).

2.1.1 Sejarah Kecerdasan Buatan

Selama bertahun-tahun para filsuf berusaha mempelajari kecerdasan buatan yang dimiliki manusia. Dari pemikiran tersebut lahirlah AI sebagai cabang ilmu yang berusaha mempelajari dan meniru kecerdasan manusia. Sejak saat itu para peneliti mulai memikirkan perkembangan AI sehingga teori-teori dan prinsip-prinsipnya berkembang terus hingga sekarang. Secara lengkap sejarah perkembangan AI dapat dikemukakan sebagai berikut:

a. Abad ke-17 -19

   Abad ini merupakan titik awal perkembanan kecerdasan buatan. Hal ini ditandai oleh penemuan-penemuan berikut :

   a) Rané Descartes menggemukakan bahwa semua tidak ada yang pasti, kecuali kenyataan bahwa seseorang bisa befikir.
   b) Blaise Pascal berhasil menciptakan mesin penghitung digital mekanis pertama pada 1642.

c) Charles Babbage dan Ada Lovelace berhasil membuat mesin penghubung mekanis yang dapat diprogram.
d) Bertand Russel dan Alfred North Whitehead menerbitkan buku Principia Mathematica, yang merombak logika formal.
e) Walter Pitts menerbitkan "Kalkulus Logis" pada 1943, yang merupakan fondasi untuk JST.

b. Pada 1950-1970

Tahun-tahun ini merupakan tahun pembuka bagi kecerdasan buatan, di mana para ilmuwan dan peneliti mulai memikirkan cara agar mesin dapat melakukan pekerjaannya seperti yang dilakukan oleh manusia. Hal ini ditandai oleh beberapa penemuan-penemuan berikut :

a) Pada Februari 1951, University of Manchester telah berhasil mengembangkan komputer elektronik pertama di dunia yang diberi nama "Ferranti Mark I"
b) Pada 1951 sebuah program permainan catur dibuat oleh Dietrich Prinz.
c) Alan Turing, seorang matematikawan Inggris pertama kali mengusulkan adanya tes untuk bisa melihat tidaknya sebuah mesin dikatakan cerdas. Hasil tes tersebut kemudian dikenal dengan *Turing Test*, di mana mesin tersebut menyamar seolah-olah sebagai seseorang di dalam suatu permainan yang mampu memberikan respons terhadap serangkaian pertanyaan yang diajukan. Turing beranggapan bahwa jika mesin dapat membuat seseorang percaya bahwa dirinya mampu berkomunikasi dengan orang lain, maka dapat dikatakan bahwa mesin tersebut cerdas (seperti layaknya manusia).
d) John McCarthy membuat istilah "kecerdasan buatan" pada konferensi pertama yang diselesaikan untuk pokok persoalan ini, pada 1956. Dia juga menemukan bahasa pemrograman Lisp.

e) ELIZA deprogram oleh Joseph Weinzenbaum (1967). Program ini mampu melakukan terapi terhadap pasien dengan memberikan beberapa pertanyaan.

f) Alain Colmerauer mengembangkan bahasa komputer Prolog.

g) Selama 1960-an dan 1970-an, Joel Moses mendemonstrasikan kekuatan pertimbangan simbolis untuk mengintegrasikan masalah di dalam program Macsyma, program berbasis pengetahuan yang sukses pertama kali dalam bidang matematika.

h) Marvin Minsky dan Seymour Papert menerbitkan Perceptrons, yang mendemonstrasikan JST sederhana.

i) Ted Shortliffe mendemonstraasikkan kekuatan sistem berbasis aturan untuk representasi pengetahuan dan inferensi dalam diagnosis dan terapi medis yang kadang kala disebut sebagai sistem pakar pertama.

c. Pada 1980-2004

Pada 1980-an, JST digunakan secara meluas dengan algoritma perambatan balik. Paul John Werbos adalah orang pertama yang menjelaskannya pada 1974. Perolehan besar dalam berbagai bidang AI dan demonstrasi berbagai macam aplikasi berlangsung pada 1990-an.

Deep Blue adalah komputer pertama yang memenangkan sebuah permainan catur melawan seorang juara dunia (Garry Kasparov) dalam waktu standar sebuah turnamen catur. Kemenangan pertamanya (dalam pertandingan atau babak pertama) terjadi pada 10 Februari 1996 dan merupakan permainan yang sangat terkenal. Namun Kasparov kemudian memenangkan 3 pertandingan lainnya dan memperoleh hasil remis pada 2 pertandingan selanjutnya sehingga mengalahkan *Deep Blue* dengan hasil 4-2. *Deep Blue* lalu di-*upgrade* lagi secara besar-besaran dan kembali bertanding melawan Kasparov pada Mei 1997. Dalam pertandingan enam babak tersebut *Deep Blue* menang dengan hasil 3,5-2,5. Babak terakhirnya berakhir pada 11 Mei. *Deep Blue* menjadi komputer pertama yang mengalahkan juara dunia

bertahan. Saat ini *Deep Blue* sudah "dipensiunkan" dan dipajang di Museum Nasional Sejaarah Amerika (*National Museum of America History*), Amerika Serikat.

Defense Advanced Research Project Agency (DARPA) menyatakan bahwa biaya yang disimpan melalui penerapan AI untuk penjadwalan dalam Perang Teluk pertama telah mengganti seluruh investasi dalam penelitian AI sejak 1950 pada pemerintah AS. Tantangan hebat DARPA, yang dimulai pada 2004 adalah sebuah pacuan untuk hadiah $2 juta di mana kendaaraan dikemudikan sendiri tanpa komunikasi dengan manusia, menggunakan GPS, komputer, dan susunan sensor yang canggih, melintas beberapa ratus mil daerah gurun yang menantang.

2.1.2 Kecerdasan Buatan dan Kecerdasan Alami

Jika dibandingkan dengan kecerdasan alami (kecerdasan yang dimiliki oleh manusia), kecerdasan buatan memiliki keuntungan komersial, antara lain :

a. Kecerdasan buatan lebih bersifat permanen. Kecerdasan alami akan cepat mengalami perubahan. Kemampuan kecerdasan buatan tidak akan pernah berubah selama programnya tidak diubah oleh programmer. Berbeda dengan kecerdasan alami karena sifat manusia yang subjektif, pelupa, dan makin lama makin tua hingga kemampuan berfikirnya berkurang seiring bertambahnya waktu, kemampuan kecerdasan alami cenderung tidak permanen.
b. Kecerdasan buatan lebih mudah diduplikasi dibandingkan kecerdasan alami.
c. Kecerdasan buatan lebih murah dibandingkan kecerdasan alami.
d. Kecerdasan buatan bersifat konsisten sedangkan kecerdasan alami bisa berubah-ubah.
e. Kecerdasan buatan dapat didokumentasikan. Solusi dan keputusan yang dibuat oleh kecerdasan buatan dapat didokumentasikan dengan mudah karena disimpan di dalam *hardsisk* dan pencarian datanya relatif lebih mudah dilacak. Sedangkan untuk kecerdasan alami, hal ini sangat sulit.

f. Kecerdasan buatan dapat mengerjakan pekerjaan lebih cepat dibanding dengan kecerdasan alami.

Sementara itu, kecerdasan alami memberikan keuntungan sebagai berikut:

a. Kreatif. Pengetahuan seorang manusia selalu bertambah seiring dengan perkembangan waktu. Sifat bosan manusia pun mengakibatkan harus berfikir kreatif untuk mencari solusi-solusi terbaru. Berbeda dengan kecerdasan buatan, penambahan pengetahuan harus dilakukan pada sistem yang telah dibangun
b. Kecerdasan alami memungkinkan orang menggunakan pengalaman secara langsung. Sedangkan pada kecerdasan buatan harus bekerja dengan input-input simbolik.
c. Pemikiran manusia dapat digunakan secara luas. Sedangkan kecerdasan buatan sangat terbatas.

2.1.3 Subdisiplin Ilmu dalam Kecerdasan Buatan

Persoalan-persoalan yang ditangani oleh kecerdasan buatan makin lama makin berkembang sehingga memungkinkan bagi kecerdasan buatan untuk merambah ke bidang ilmu yang lain. Hal ini disebabkan karakteriktik cerdas sudah mulai dibutuhkan di berbagai disiplin ilmu teknologi. Sebagai contoh, perkawinan antara psikologi kecerdasan buatan melahirkan sebuah area yang dikenal dengan nama *cognition & psycolinguistics*. Perpaduan antara teknik elektro dengan kecerdasan buatan melahirkan berbagai ilmu baru seperti pengolahan citra, teori kendali, serta pengenalan pola dan robotika. Sistem Pendukung Keputusan dan Sistem Informasi Manajemen adalah hasil kontribusi dari kecerdasan buatan. Keadaan seperti ini melahirkan fenomena baru yang cukup rumit untuk mengklasifikasikan kecerdasan buatan menurut disiplin ilmu yang menggunakannya. Oleh karena itu, klasifikasi subdisiplin ilmu dalam kecerdasan buatan berdasarkan pada hasil perkawinan antara kecerdasan buatan dengan bidang ilmu lainnya. Pembahasan selanjutnya adalah tentang beberapa subdisiplin ilmu dalam kecerdasan buatan (T. Sutojo, 2011).

a. Sistem Pakar (*Expert System*)

Sistem pakar adalah suatu sistem yang dirancang untuk dapat menirukan keahlian seorang pakar dalam menjawab pertanyaan dan memecahkan suatu masalah. Sistem pakar akan memberikan pemecahan suatu masalah yang didapat dari dialog dengan pengguna. Dengan bantuan sistem pakar seseorang yang bukan ahli/pakar dapat menjawab pertanyaan, menyelesaikan masalah serta mengambil keputusan yang biasanya dilakukan oleh seorang pakar.

b. Pengolahan Bahasa Alami (*Natural Language Processing*)

Pengolahan Bahasa Alami adalah pembuatan program yang memiliki kemampuan untuk memahami bahasa manusia. Pada prinsipnya bahasa alami adalah suatu bentuk representasi dari suatu pesan yang ingin dikomunikasikan antarmanusia. Bentuk utama representasinya adalah berupa suara/ucapan (*spoken/language*), tetapi sering pula dinyatakan dalam bentuk tulisan. Dengan pengolahan bahasa alami ini diharapkan pengguna dapat berkomunikasi dengan komputer dengan menggunakan bahasa sehari-hari. Tujuannya adalah melakukan proses pembuatan model komputasi dari bahasa sehingga dapat terjadi suatu interaksi antara manusia dengan komputer dengan perantara bahasa alami. Diantara contohnya adalah *Google Translate* dan *Text to Speach.*

c. Pengenalan Ucapan (*Speech Recognition*)

Pengenalan ucapan, atau yang sering disebut dengan *Automatic Speech Recognition* (ASR), adalah suatu pengembangan teknik dan sistem yang memungkinkan komputer untuk menerima masukan berupa kata yang diucapkan.

d. Robotika dan Sistem Sensor (*Robotics and Sensory System*)

Robotika adalah ilmu pengetahuan dan tekhnologi rekayasa robot, dan desain, menufaktur, aplikasi dan disposisi struktural. Robotika berhubungan dengan elektronik, mekanik dan perangkat lunak. Istilah robot berawal dari bahasa Cheko "**robota**" yang berarti pekerja yang tidak mengenal lelah atau bosan. Sedangkan terminologi, arti yang paling tepat dengan istilah robot adalah "**Sistem atau alat yang digunakan untuk menggantikan kinerja manusia secara otomatis**". Robot adalah mesin yang mampu melaksanakan fungsi fisik manusia secara terbatas. Agar bias menggunakan *software intelligence*, robot harus bisa menerima input lingkungannya. Ini berarti ia harus menggunakan sensor untuk mendeteksi posisi yang ada di sekitarnya.

e. *Computer Vision*

Computer Vision adalah salah satu cabang ilmu pengetahuan yang mempelajari bagaimana computer dapat mengenali objek yang diamati atau diobservasi. *Computer vision* mencoba untuk dapat menginterpretasikan gambar atau objek-objek tampak melalui computer dan mendeskripsikannya sehingga menghasilkan informasi yang berguna.

f. *Intelligent Computer-Aided Instruction*

Komputer dapat digunakan sebagai tutor yang dapat melatih dan mengajar.

g. *Game Playing*

Dalam kamus bahasa Indonesia, *Game* adalah permainan, yaitu kegiatan yang kompleks yang di dalamnya terdapat peraturan, *play*, dan budaya. Dalam permainan terdapat peraturan yang bertujuan untuk membatasi perilaku pemain sekaligus menentukan permainan.

2.1.4 Lingkup Kecerdasan Buatan pada Aplikasi Komersial

Permasalahan manusia semakin bertambah seiring dengan perkembangan zaman. Oleh karena itu, bantuan dari kemajuan teknologi sangat dibutuhkan untuk mengatasi permasalahan tersebut, khususnya teknologi kecerdasan buatan. Dalam kehidupan manusia, kecerdasan buatan dapat membantu dalam berbagai bidang berikut (T. Sutojo, 2011).

a. Bidang Komunikasi

Komputer digunakan untuk member komando suara dari pengguna. Contohnya adalah aplikasi *Microsoft Voice* yang berbasis bahasa Inggris.

Ketika pengguna mengatakan "Mulai kalkulator" dengan intonasi dan tata bahasa yang sesuai, komputer segera membuka aplikasi kalkulator. Jika komando suara yang diberikan sesuai dengan daftar perintah yang tersedia, aplikasi akan memastikan komando suara dengan menampilkan tulisan "Aplikasi Anda meminta saya untuk 'mulai kalkulator'?". Untuk melakukan verifikasi, pengguna cukup mengatakan "Lakukan" dan komputer akan langsung beroperasi.

Komputer digunakan untuk menuliskan kata-kata yang didekte oleh pengguna. Hal ini banyak dimanfaatkan dalam pembuatan laporan atau penelitian. Contohnya adalah aplikasi *Microsoft Dictation* yang merupakan aplikasi yang dapat menuliskan apa yang diucapkan oleh pengguna secara otomatis.

b. Bidang Kesehatan

Komputer untuk membantu para penyandang cacat dalam beraktifitas. Contohnya adalah aplikasi Antarmuka Suara Pengguna atau *Voice User Intervace* (VUI) yang menggunakan teknologi pengenal ucapan di mana pengendalian saklar lampu, misalnya, tidak perlu dilakukan secara manual dengan menggerakkan saklar, tetapi cukup dengan mengeluarkan perintah

dalam bentuk ucapan sebagai saklar. Metode ini membantu manusia secara fisik tidak dapat menggerakkan saklar karena cacat pada tangannya misalnya. Penerapan VUI ini tidak hanya untuk lampu saja, tetapi bisa juga untuk aplikas-aplikasi kontrol yang lain.

Melalui sistem pakar, komputer digunakan untuk mendiagnosis penyakit, melakukan monitoring kondisi pasien, dan memberikan *treatment* yang cocok.

c. Bidang Lalu Lintas Udara

Komputer untuk mengatur lalu lintas udara atau yang dikenal dengan *Air Traffic Controllers* (ATC) yang dipakai oleh para pilot untuk mendapatkan keterangan mengenai keadaan lalu lintas udara seperti radar, cuaca dan navigasi.

d. Bidang Pertanian

Komputer untuk mengontrol robot yang melakukan kontrol terhadap penyiraman tanaman, pemantauan hama, pemilihan hasil panen dan tugas-tugas lain yang ada dalam dunia pertanian.

e. Pabrik

Komputer untuk mengontrol robot yang harus melakukan suatu pekerjaan yang bisa berbahaya oleh manusia, pekerjaan-pekerjaan yang membosankan, pekerjaan-pekerjaan yang memerlukan ketelitian tinggi, pekerjaan-pekerjaan yang memerlukan pengawasan dan pekerjaan-pekerjaan *maintenance*.

f. *Soft Computing*

Istilah *soft computing* pertama kali dicetuskan pada 1990 oleh Prof. L.A. Zadeh dari *Berkeley University* berkaitan dengan ide untuk mendirikan *Berkeley Intiative in Soft Computer* (BISC). *Soft computing* adalah

kumpulan teknik komputasi dalam ilmu komputer, yang berusaha untuk mempelajari, memodelkan dan menganalisis suatu fenomena tertentu guna mengeksploitasi adanya toleransi terhadap ketidaktepatan, ketidakpastian dan kebenaran parsial untuk dapat diselesaikan dengan mudah, *robustness*, dan biaya penyelesaiannya murah. *Soft computing* berbeda dengan *conventional* (*hard*) *computing*, yang memungkinkan toleransi terhadap input, proses, dan output yang bersifat tidak akurat (*imprecision*), tidak pasti (*uncertainty*), dan setengah benar (*partial truth*). *Soft computing* berusaha untuk mengintegrasikan beberapa paradigma model perhitungan meliputi *artificial neural network, fuzzy logic* dan *genetic algorithms*.

2.2 *Jaringan Saraf Tiruan (JST)*

JST adalah paradigma pengolahan informasi yang terinspirasi oleh sistem saraf secara biologis, seperti proses informasi pada otak manusia. Elemen kunci dari paradigma ini adalah struktur dari sistem pengolahan informasi yang terdiri dari sejumlah besar elemen pemrosesan yang saling berhubungan (*neuron*), bekerja serentak untuk menyelesaikan masalah tertentu. Cara kerja JST sama seperti cara kerja manusia, yaitu belajar melalui contoh. Sebuah JST dikonfigurasikan untuk aplikasi tertentu, seperti pengenalan pola atau aplikasi data, melalui proses pembelajaran. Belajar dalam sistem biologis melibatkan penyesuaian terhadap koneksi *synaptic* yang ada antara *neuron*. Hal ini berlaku juga untuk JST. (T. Sutojo, 2011).

2.2.1 Kemampuan JST

JST mempunyai kemampuan yang sangat luar biasa untuk mendapatkan informasi dari data yang rumit atau tidak tepat, mampu menyelesaikan permasalahan yang tidak terstruktur dan sulit didefinisikan, dapat belajar dari pengalaman, mampu mengakuisisi pengetahuan walaupun tidak ada kepastian, mampu melakukan generalisasi dan ekstrasi dari suatu pola data tertentu, dapat menciptakan suatu pola pengetahuan melalui pengaturan diri atau kemampuan belajar (*self organizing*), mampu memilih suatu input data ke dalam kategori

tertentu yang sudah ditetapkan (klasifikasi), mampu menggambarkan objek secara keseluruhan maupun hanya diberikan sebagian data dari objek tersebut (asosiasi), mempunyai kemampuan mengolah data-data iniput tanpa harus mempunyai target (*self organizing*) dan mampu menemukan jawaban terbaik sehingga mampu meminimilasi fungsi biaya (optimasi).

Kelebihan-kelebihan yang diberikan oleh JST antara lain:

a. Belajar *Adaptive*

   Kemampuan untuk mampu mempelajari bagaimana melakukan pekerjaan berdasarkan data yang diberikan untuk pelatihan atau pengalaman awal.

b. *Self-Organization*

   Sebuah JST dapat membuat organisasi sendiri atau representasi dari informasi yang diterimanya selama waktu belajar.

c. *Real Time Operation*

   Perhitungan JST dapat dilakukan secara paralel sehingga perangkat keras yang dirancang dan diproduksi secara khusus dapat mengambil keuntungan dari kemampuan ini.

Selain mempunyai kelebihan-kelebihan tersebut, JST juga mempunyai kelemahan-kelemahan berikut:

a. Tidak efektif jika digunakan untuk melakukan operasi-operasi numerik dengan persisi tinggi.
b. Tidak efisien jika digunakan untuk melakukan operasi algoritma aritmatik, operasi logika dan simbolis.
c. Untuk beroperasi JST butuh pelatihan sehingga bila jumlah datanya besar, waktu yang digunakan untuk proses pelatihan sangat lama.

Kata kunci dari semua ini adalah “JST tidak melakukan mukjizat, tetapi jika digunakan dengan bijaksana, JST dapat memberikan hasil yang luar biasa”. (T. Sutojo, 2011).

2.2.2 JST Versus Pemrograman Biasa

Untuk pemecahan masalah, JST mengambil pendekatan yang berbeda dibandingkan dengan pemrograman konvensional. Pemrograman konvensional menggunakan pendekatan algoritmik, yaitu dengan mengikuti seperangkat instruksi untuk memecahkan masalah. Instruksi-instruksi ini selanjutnya dikonversi ke program bahasa tingkat tinggi dan kemudian ke dalam kode mesin yang komputer dapat mengerti. Mesin ini benar-benar dapat diprediksi jika ada yang tidak beres disebabkan oleh kesalahan perangkat lunak atau perangkat keras. Tanpa langkah-langkah spesifik tersebut (seperangkat instruksi) pemrograman konvensional tidak bisa menyelesaikan masalah menggunakan pemrograman konvensional, kita harus mengerti maslah tersebut dan kita harus mengetahui cara menyelesaikan masalah tersebut. Hal inilah yang membatasi kemampuan pemecahan masalah menggunakan pemrograman konvesional.

JST menemukan cara menyelesaikan masalah dengan sendirinya sehingga kita bisa melakukan hal-hal yang kita tidak tahu persis bagaimana melakukannya. Selain itu, kita bisa menyelesaikan masalah tanpa harus mengetahui terlebih dahulu bagaimana menyelesaikan masalah tersebut. Namun, kelemahannya adalah karena jaringan menemukan cara untuk memecahkan masalah dengan sendirinya, pengoperasiannya tidak dapat diprediksi.

JST dan pemrograman konvensional tidak saling bersaing, tetapi saling melengkapi. Ada masalah-masalah yang harus diselesaikan menggunakan pendekatan algoritmik seperti operasi aritmatika dan masalah-masalah yang harus diselesaikan menggukan JST. Bahkan, ada juga masalah-masalah besar yang terpaksa menggunaka kombinasi dari keduanya. (T. Sutojo, 2011).

2.2.3 Aplikasi JST

Hingga kini terdapat lebih dari 20 metode JST. Masing-masing metode menggunakan arsitektur, fungsi aktifasi dan perhitungan berbeda-beda dalam prosesnya. Aplikasi yang sudah berhasil ditemukan antara lain (Jong Jek Siang, 2011).

1. Klasifikasi

   Model yang bisa digunakan untuk aplikasi ini antara lain *Adaline,* LVQ, *Backpropagation* dan lain-lain.

2. Pengenalan Pola

   Model yang bisa digunakan untuk aplikasi ini antara lain *Adaptive Resonance Theory (ART), LVQ, Backpropagation, Neocognitron*, dan lain-lain.

3. Peramalan

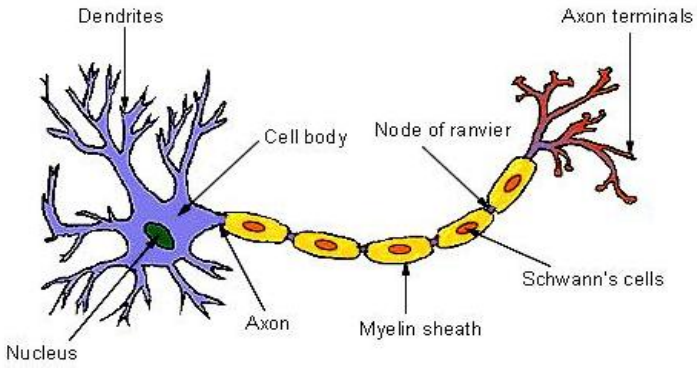
   Model yang bisa digunakan untuk aplikasi ini antara lain *Adaline, Madaline, Backpropagation*, dan lain-lain.

4. Optimasi

   Model yang bisa digunakan untuk apliaksi ini antara lain *Adaline, Hopfield, Boltzman, Backpropagation*, dan lain-lain.

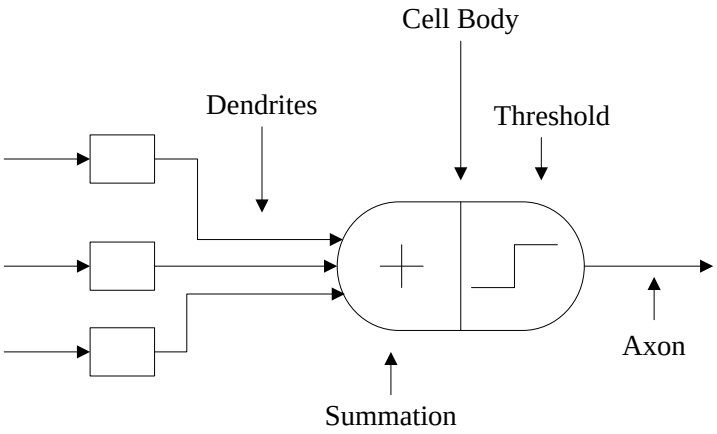
2.2.4 Model Biologi

JST muncul setelah pengenalan *neuron* disederhanakan oleh McCulloch dan Pitts pada 1943 (McCulloch dan Pitts, 1943). *Neuron* ini disajikan sebagai model *neuron* biologis dan sebagai komponen konseptual untuk rangkaian yang dapat melakukan tugas-tugas komputasi. Dalam otak manusia (Gambar 2.1), sebuah *neuron* (sel saraf) tertentu mengumpulkan sinyal berupa rangsangan dari *neuron* lain melalui *dendrite.* Sinyal yang datang dan diterima oleh *dendrite* akan dijumlahkan (*summation*) dan dikirim melalui *axon* ke *dendrite* akhir yang bersentuhan dengan *dendrite* dari *neuron* yang lain. Sinyal ini akan diterima oleh *neuron* lain jika memenuhi nilai *threshold* tertentu. Dalam hal ini, *neuron* dikatakan teraktifasi. Cara kerja otak mannusia ini dapat disederhanakan menjadi model *neuron* (Gambar 2.2). pembelajaran pada otak manusia terjadi ketika ada hubungan antara satu *neuron* dengan *neuron* lain yang terjadi secara adaptif dan berlangsung secara dinamis.



Gambar 2.1 Komponen *Neuron*

(Internet, 2012)



Gambar 2.2 Model *Neuron*

(T. Sutojo, 2011)

2.2.5 Model Matematika

Terdapat tiga komponen dasar penting ketika membuat sebuah model fungsional dari *neuron* biologis. Pertama, sinapsis *neuron* dimodelkan sebagai bobot. Kekuatan hubungan antara masukan dan *neuron* ditentukan oleh nilai bobot. Nilai bobot negatif mencerminkan koneksi hambat, sedangkan nilai-nilai positif menandakan koneksi rangsang sel. Komponen kedua adalah penjumlah semua masukan yang dimodifikasi oleh masing-masing bobot. Kegiatan ini disebut sebagai kombinasi linear. Komponen ketiga bertindak sebagai fungsi kontrol aktifitas amplitudo output dari *neuron*.

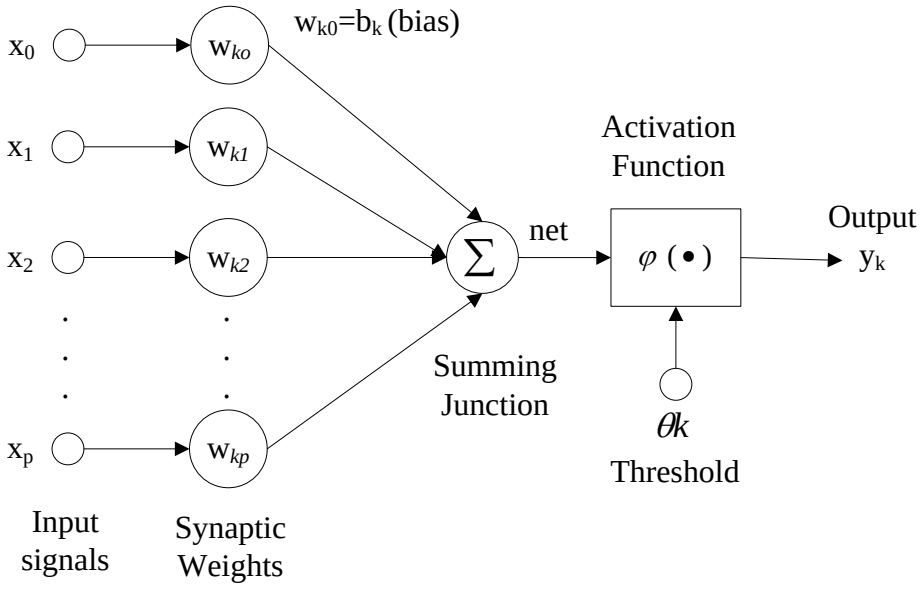
JST merupakan generalisasi model matematis dengan beberapa asumsi berikut.

a. Pemrosesan informasi terjadi pada *neuron*.
b. Sinyal dikirimkan di antara *neuron-neuron* melalui penghubung *dendrite* dan *axon*.
c. Penghubung antarelemen memiliki bobot yang akan menambah atau mengurangi sinyal.
d. Untuk menentukan output, setiap *neuron* memiliki fungsi aktifasi yang dikenakan pada jumlah semua input-nya. Besar output akan dibandingkan dengan nilai *threshold* tertentu.

Berdasarkan model matematis tersebut, baik tidaknya suatu model JST ditentukan oleh hal-hal berikut:

a. Arsitektur jaringan, yaitu sebuah arsitektur yang menentukan pola antar*neuron*.
b. Metode pembelajaran (*learning method*), yaitu metode untuk menentukan dan mengubah bobot.
c. Fungsi aktifasi.

Secara matematis, proses ini dijelaskan dalam Gambar 2.3



Gambar 2.3 Model Matematis dari JST

(T. Sutojo, 2011)

Dari model ini aktifitas interval *neuron* dapat ditunjukkan sebagai berikut :

$$net = \sum_{j=1}^{p} w_{kj} x_j \tag{2.1}$$

(T. Sutojo, 2011)

Setelah *net* melewati fungsi aktifasi tertentu, output *neuron* adalah y_k.

2.2.6 Arsitektur Jaringan

Seperti yang telah disebutkan sebelumnya, baik tidaknya suatu model JST salah satunya ditentukan oleh hubungan antarneuron atau yang biasa disebut sebagai arsitektur jaringan. *Neuron-neuron* tersebut terkumpul dalam lapisan-lapisan yang disebut *neuron layer*. Lapisan-lapisan penyusun JST dibagi menjadi tiga, yaitu:

a. Lapisan Input (*Input Layer*)

   Unit-unit dalam lapisan input disebut unit-unit input yang bertugas menerima pola inputan dari luar yang menggambarkan suatu permasalahan.

b. Lapisan Tersembunyi (*Hidden Layer*)

   Unit-unit dalam lapisan tersembunyi disebut unit-unit tersembunyi yang nilai output-nya tidak dapat diamati secara langsung.

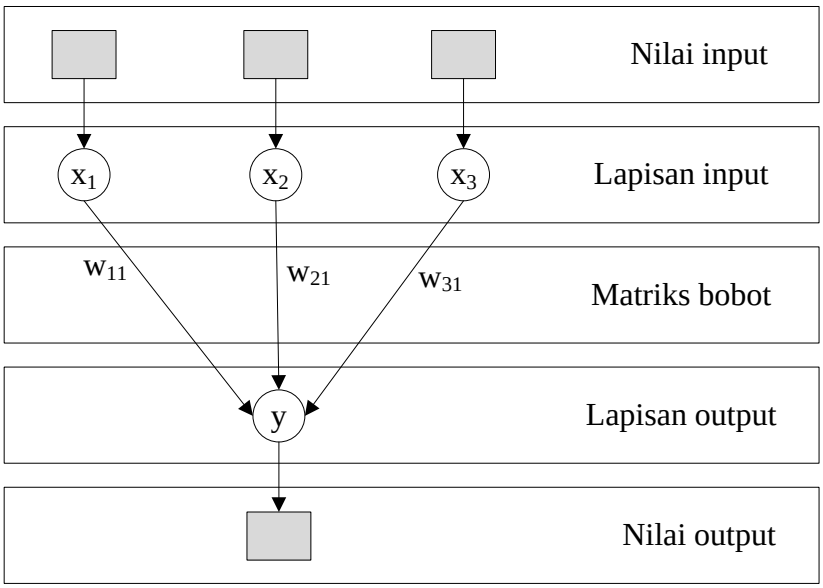
c. Lapisan Output (*Output Layer*)

   Unit-unit dalam lapisan output disebut unit-unit output, yang merupakan solusi JST terhadap suatu permasalahan.

Arsitektur jaringan yang sering digunakan dalam JST antara lain jaringan lapisan tunggal, jaringan lapisan banyak, dan jaringan lapisan kompetitif dengan masing-masing penjelasan sebagai berikut:

a. Jaringan Lapisan Tunggal

Jaringan dengan lapisan tunggal terdiri dari 1 lapisan input dan 1 lapisan output (Gambar 2.4). Setiap unit dalam lapisan input selalu terhubung dengan setiap unit yang terdapat pada lapisan output. Jaringan ini menerima input kemudian mengolahnya menjadi output tanpa melewati lapisan tersembunyi. Contoh JST yang menggunakan jaringan lapisan tunggal adalah *Adaline*, *Hopfield*, dan *Perceptron*.



Gambar 2.4 JST dengan lapisan tunggal

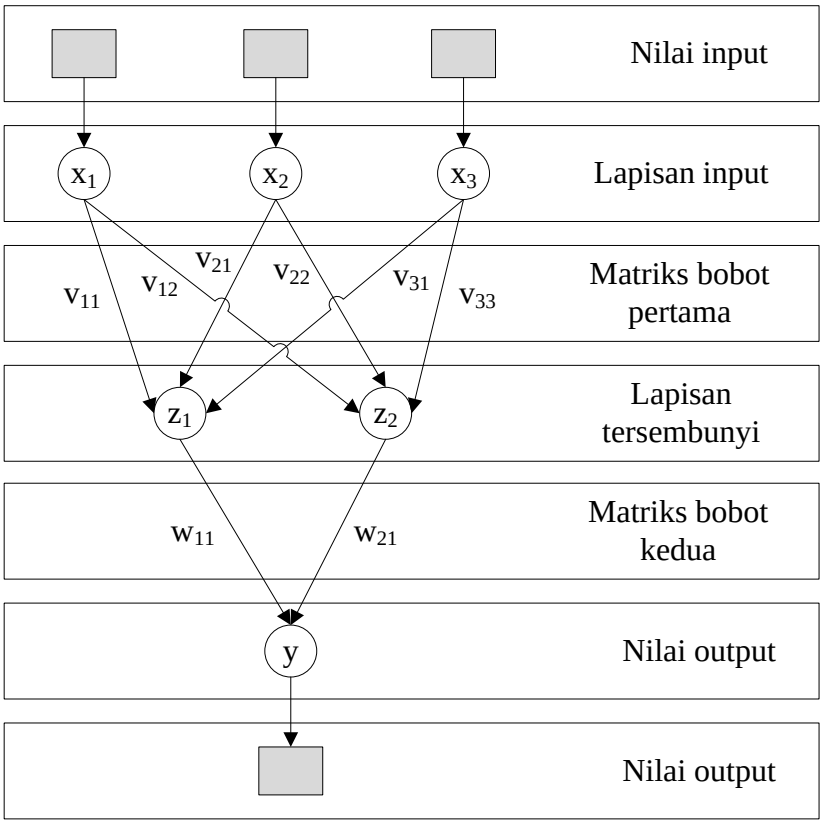
(T. Sutojo, 2011)

Pada Gambar 2.4 tersebut lapisan input memiliki 3 unit *neuron*, yaitu x_1, x_2 dan x_3 yang terhubung langsung dengan lapisan output yang memiliki 2 unit *neuron*, yaitu y_1 dan y_2. Hubungan neuron-neuron pada lapisan tersebut ditentukan oleh bobot yang bersesuaian w_{11}, w_{22}, dan w_{31}.

b. Jaringan Lapisan Banyak

Jaringan banyak mempunyai 3 jenis lapisan, yaitu lapisan input, lapisan tersembunyi dan lapisan output (Gambar 2.5). Jaringan ini dapat menyelesaikan permasalahan yang lebih kompleks dibandingkan dengan

jaringan lapisan tunggal. Contoh JST yang menggunakan jaringan lapisan banyak adalah *Madaline, Backpropagation,* dan *Neocognitron.*



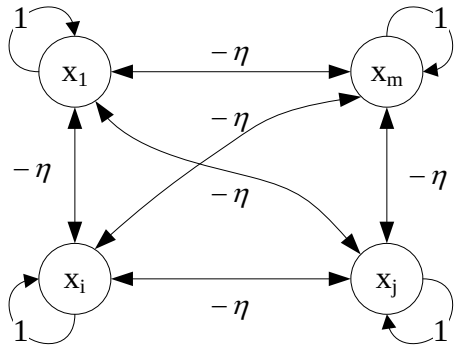
Gambar 2.5 JST dengan lapisan banyak

(T. Sutojo, 2011)

Pada Gambar 2.5, lapisan input mamiliki 3 unit *neuron*, yaitu x_1, x_2 dan x_3 yang terhubung langsung dengan lapisan tersembunyi yang memiliki 2 unit *neuron* tersembunyi, yaitu z_1 dan z_2. Hubungan *neuron-neuron* pada lapisan input dan lapisan output tersebut ditentukan oleh bobot v_{11}, v_{12}, v_{21}, v_{22}, v_{31} dan v_{32}. Kemudian, 2 unit *neuron* tersembunyi z_1 dan z_2 terhubung langsung dengan lapisan output yang memiliki 1 unit *neuron* Y yang besarnya ditentukan oleh bobot w_{11} dan w_{21}.

c. Jaringan Lapisan Kompetitif

Jaringan ini memiliki bobot yang telah ditentukan dan tidak memiliki pelatihan (Gambar 2.6). Jaringan ini digunakan untuk mengetahui *neuron* pemenang dari sejumlah *neuron* yang ada. Akibatnya, pada jaringan ini sekumpulan *neuron* bersaing untuk mendapatkan hak menjadi aktif. Nilai bobot setiap *neuron* untuk dirinya sendiri adalah 1, sedangkan untuk *neuron* lainnya bernilai *random* negatif. Contoh JST yang menggunakan dengan lapisan kompetitif adalah LVQ.



Gambar 2.6 JST dengan lapisan kompetitif

(T. Sutojo, 2011)

2.2.7 Fungsi Aktivasi

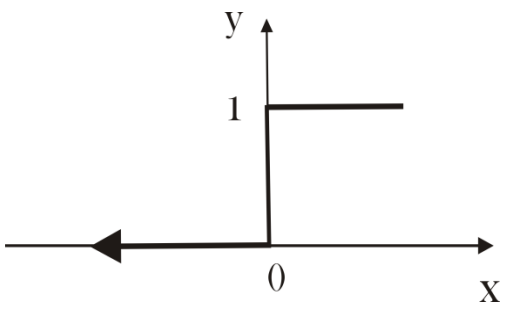
Perilaku dari JST ditentukan oleh bobot dan input-output fungsi aktifasi yang ditetapkan. Beberapa fungsi aktifasi yang sering digunakan dalam JST adalah sebagai berikut (T. Sutojo, 2011) :

a. Fungsi Undak Biner *Hard Limit*

Fungsi undak biner ini biasanya digunakan oleh jaringan lapisan tunggal untuk mengonversi nilai input dari suatu veriabel yang bernilai kontinu ke suatu nilai output biner (0 atau 1) (Gambar 2.7). secara matematis, fungsi untuk biner (*Hard Limit*) dituliskan sebagai berikut.

$$y = \begin{cases} 0, \text{jika } x \leq 0 \\ 1, \text{jika } x > 0 \end{cases} \quad (2.2)$$

(T. Sutojo, 2011)



Gambar 2.7 Fungsi Undak Biner *Hard Limit*

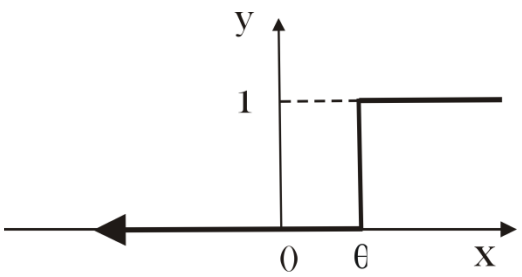
(T. Sutojo, 2011)

b. Fungsi Undak Biner *Threshold*

Berbeda dengan fungsi undak biner *Hard Limit*, fungsi Undak Biner *Threshold* menggunakan nilai ambang θ sebagai batasnya (Gambar 2.8). Secara matematis, fungsi undak biner *threshold* dituliskan sebagai berikut.

$$y = \begin{cases} 0, \text{jika } x < 0 \\ 1, \text{jika } x \geq 0 \end{cases} \quad (2.3)$$

(T. Sutojo, 2011)



Gambar 2.8 Fungsi Undak Biner *Threshold*

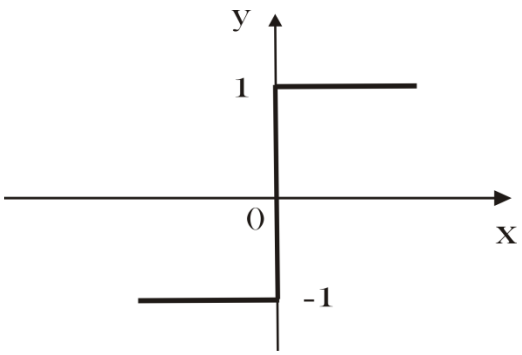
(T. Sutojo, 2011)

c. Fungsi Bipolar *Symetric Hard Limit*

Fungsi Bipolar *Symetric Hard Limit* mempunyai output yang bernilai 1, 0, atau -1 (Gambar 2.9). Secara matematis, fungsi *Symetric Hard Limit* dituliskan sebagai berikut.

$$y = \begin{cases} 1, \text{jika } x > 0 \\ 0, \text{jika } x = 0 \\ -1, \text{jika } x < 0 \end{cases} \quad (2.4)$$

(T. Sutojo, 2011)



Gambar 2.9 Fungsi Bipolar Symetric *Hard Limit*

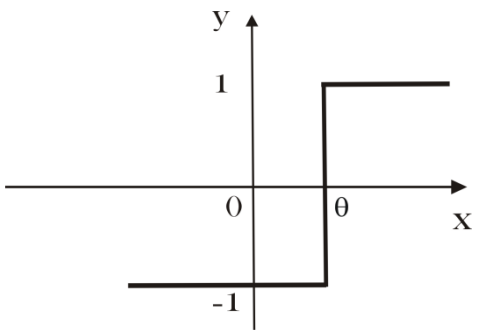
(T. Sutojo, 2011)

d. Fungsi Bipolar dengan *Threshold*

Fungsi Bipolar dengan *Threshold* mempunyai output yang bernilai 1, 0, atau -1 (Gambar 2.10) untuk batas nilai ambang θ tertentu. Secara matematis, Fungsi Bipolar dengan *Threshold* dituliskan sebagai berikut.

$$y = \begin{cases} 1, \text{jika } x > \theta \\ 0, \text{jika } x = \theta \\ -1, \text{jika } x < \theta \end{cases} \quad (2.5)$$

(T. Sutojo, 2011)



Gambar 2.10 Fungsi Bipolar dengan *Threshold*

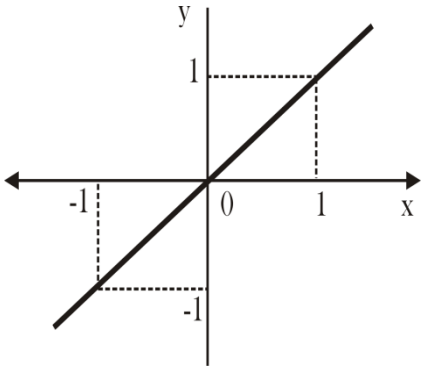
(T. Sutojo, 2011)

e. Fungsi *Linear* (Identitas)

Nilai input dan nilai output pada fungsi linear adalah sama (Gambar 6.11). Secara matematis, fungsi linear dituliskan sebagai berikut.

$$y = x \tag{2.6}$$

(T. Sutojo, 2011)



Gambar 2.11 Fungsi Linear (identitas)

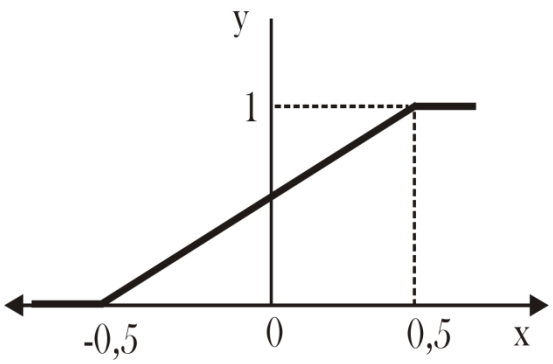
(T. Sutojo, 2011)

f. Fungsi *Saturating Linear*

Fungsi ini akan bernilai 1 jika input-nya lebih dari $\frac{1}{2}$. Jika nilai input terletak antara -$\frac{1}{2}$ dan $\frac{1}{2}$ maka output-nya akan bernilai sama dengan nilai tambah $\frac{1}{2}$. Jika nilai input-nya kurang dari -$\frac{1}{2}$ maka fungsi bernilai 0 (Gambar 2.12). Secara sistematis, fungsi *satirating linear* dituliskan sebagai berikut.

$$y = \begin{cases} 1, \text{jika } x \geq 0.5 \\ x + 0.5, \text{jika } -0.5 \leq x \leq 0.5 \\ -1, \text{jika } x \leq -0.5 \end{cases} \tag{2.7}$$

(T. Sutojo, 2011)



Gambar 2.12 Fungsi Saturating Linear

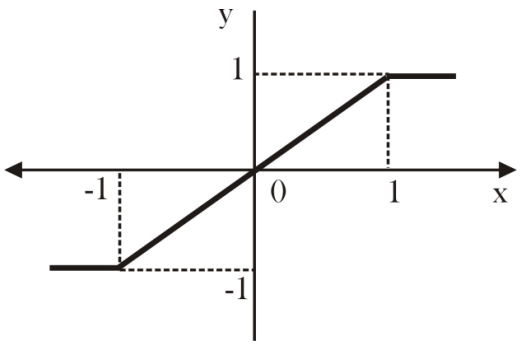
(T. Sutojo, 2011)

g. Fungsi *Symetric Saturating Linear*

Fungsi ini akan bernilai 1 jika input-nya lebih dari 1. Jika nilai input terletak antara -1 dan 1 maka output-nya akan bernilai sama dengan nilai output-nya. Sedangkan jika input-nya kurang dari -1 maka fungsinya akan bernilai -1 (Gambar 2.13). Secara matematis, fungsi *symmetric saturating linear* dituliskan sebagai berikut.

$$y = \begin{cases} 1, \text{jika } x \geq 1 \\ x, \text{jika } -1 \leq x \leq 1 \\ -1, \text{jika } x \leq -1 \end{cases} \quad (2.8)$$

(T. Sutojo, 2011)



Gambar 2.13 Fungsi Symetric Saturating Linear

(T. Sutojo, 2011)

h. Fungsi Sigmoid Biner

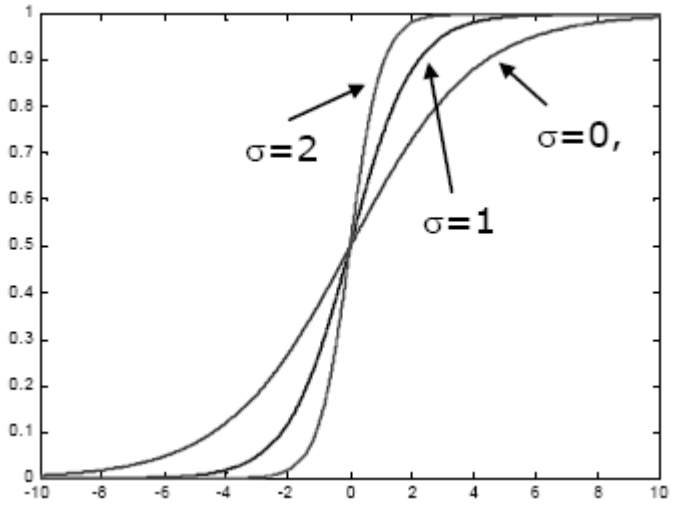
Biasanya fungsi ini digunakan untuk JST yang dilatih menggunakan metode *Backpropagation*. JST yang membutuhkan nilai output yang terletak antara 0 sampai 1 sering kali menggunakan fungsi sigmoid biner karena fungsi ini memiliki nilai pada *range* 0 sampai 1 (Gambar 2.14). Secara matematis, fungsi sigmoid biner dituliskan sebagai berikut.

$$y = f(x) = \frac{1}{1+e^{-\sigma x}} \tag{2.9}$$

(T. Sutojo, 2011)

$$\text{dengan: } y = f'(x) = \sigma f(x)[1 - f(x)] \tag{2.10}$$

(T. Sutojo, 2011)



Gambar 2.14 Fungsi Sigmoid Biner

(T. Sutojo, 2011)

i. Fungsi Sigmoid Bipolar

Output dari fungsi sigmoid bipolar memiliki range antara 1 sampai -1 (Gambar 2.15). Secara matematis, fungsi sigmoid bipolar dirumuskan sebagai berikut :

$$y = f(x) = \frac{1 - e^{-x}}{1 + e^{-x}} \tag{2.11}$$

(T. Sutojo, 2011)

$$\text{dengan: } y = f'(x) = \frac{\sigma}{2}[1 + f(x)][1 - f(x)] \qquad (2.12)$$

(T. Sutojo, 2011)

Fungsi ini hampir sama dengan fungsi *hyperbolic tangent*. Keduanya memiliki range antara -1 sampai 1. Secara matematis, fungsi *hyperbolic tangent* dituliskan sebagai berikut.

$$y = f(x) = \frac{e - e^{-x}}{e^{x} + e^{-x}} \qquad (2.13)$$

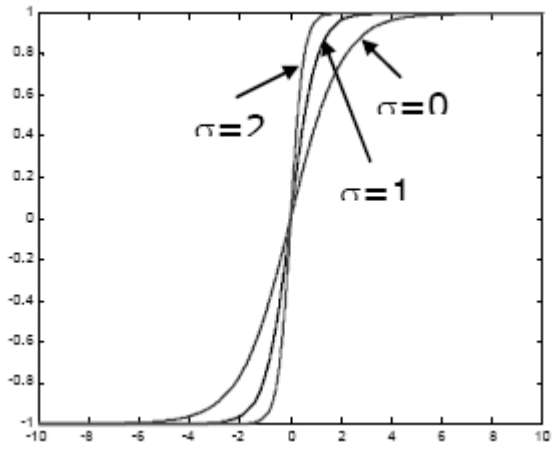
(T. Sutojo, 2011)

$$\text{atau } y = f(x) = \frac{1 - e^{-zx}}{1 + e^{-x}} \qquad (2.14)$$

(T. Sutojo, 2011)

$$\text{dengan: } y = f'(x) = [1 + f(x)][1 - f(x)] \qquad (2.15)$$

(T. Sutojo, 2011)



Gambar 2.15 Fungsi Sigmoid Biner

(T. Sutojo, 2011)

2.2.8 Model *Neuron* McCulloch Pitts

Pada 1943 S. Warren McCulloch, seorang ahli saraf dan Walter Pitts, seorang ahli logika, mencoba memahami bagaimana otak dapat menghasilkan pola yang sangat kompleks dengan menggunakan sel-sel dasar yang terhubung bersama-sama. Sel-sel otak disebut *neuron* dasar, kemudian McCulloch dan Pitts memberikan model yang sangat sederhana dari sebuah *neuron*. Model *neuron* McCulloch-Pitts merupakan model JST pertama yang ditemukan dan telah membuat kontribusi penting bagi perkembangan JST sampai sekarang. McCulloch dan Pitts menyimpulkan bahwa kombinasi beberapa *neuron* sederhana menjadi sebuah sistem *neural* akan meningkatkan kemampuan komputasinya. Bobot dalam jaringan yang diusulkan oleh McCulloch dan Pitts diatur untuk melakukan fungsi logika sederhana. Fungsi aktifasi yang dipakai adalah fungsi *threshold* .

2.1.1 Algoritma Pembelajaran dengan Supervisi (*Supervised*)

Pada pembelajaran terawasi, metode ini digunakan jika output yang diharapkan telah diketahui sebelumnya. Biasanya pembelajaran dilakukan dengan menggunakan data yang telah ada. Pada proses pembelajaran, satu pola input akan diberikan ke satu *neuron* pada lapisan input. Pola ini dirambatkan di sepanjang jaringan saraf hingga sampai ke *neuron* pada lapisan output. Lapisan output ini akan membangkitkan pola output yang nantinya akan dicocokan dengan pola output targetnya. Apabila terjadi perbedaan antara pola output hasil pembelajaran dengan pola target, maka di sini muncul *error*. Apabila nilai *error* masih cukup besar, mengindikasikan bahwa masih perlu dilakukan lebih banyak pembelajaran lagi. Diantara model JST dengan pembelajaran terawasi adalah model *Hebbian*, *Perceptron*, *Adaline, Boltzam, Hopfield, Backpropagation*, dan lain-lain.

a. Hebb Rule

   Model *neuron* McCulloch-Pitts akan mengalami kesulitan apabila berhadapan dengan fungsi-fungsi yang kompleks. Hal ini terjadi karena dalam menentukan bobot w dan nilai ambang θ harus dilakukan secara

analitik atau menggunakan cara coba-coba. Pada 1949, D.O. Hebb memperkenalkan cara menghitung bobot w dan bias secara interatif dengan memanfaatkan model pembelajaran dengan supervisi sehingga bobot dengan w dan bias dapat dihitung secara otomatis tanpa harus melakukan cara coba-coba. Model Hebb merupakan model jaringan tertua yang menggunakan model pembelajaran supervisi. Fungsi aktifasi yang digunakan adalah fungsi biner atau bipolar.

b. *Perceptron*

Model jaringan *perceptron* merupakan model yang paling baik pada saat itu. Model ini ditemukan oleh Rosenblatt (1962) dan Minsky – Papert (1969). Fungsi aktifasi yang digunakan bukan fungsi biner (atau bipolar), tetapi memiliki kemungkinan -1, 0 atau 1.

c. *Adaptive Linear Neuron (Adaline)*

Model *Adaline* ditemukan oleh Widrow dan Hoff (1960). Arsitekturnya mirip dengan *perceptron*. Beberapa masukan dan sebuah bias (unit masukan tambahan) dihubungkan langsung dengan sebuah *neuron* keluaran. Bobot dimodifikasi dengan aturan delta, sering juga disebut *Least Mean Square (LMS)*.

$$\text{net} = \sum_{i=1}^{i} x_i w_{i+} + b \qquad (2.16)$$

(Jong Jek Siang, 2004)

$$y = f(\text{net}) = \text{net} = \sum_{i=1}^{i} x_i w_{i+} + b \qquad (2.17)$$

(Jong Jek Siang, 2004)

Kuadrat selisih antara target (t) dan keluaran jaringan f(net) merupakan *error* yang terjadi. Dalam aturan delta, bobot dimodifikasi sedemikian hingga *error*-nya minimum.

$$E = (t - f(\text{net}))^2 = (t - (\sum_{i=1}^{i} x_i w_i + b))^2 \qquad (2.18)$$

(Jong Jek Siang, 2004)

E merupakan fungsi bobot w_i. Penurunan E tercepat terjadi pada arah

$$\frac{\partial E}{\partial w_i} = -2\left(t - \left(\sum_{i=1}^{i} x_i w_i\right)x_i\right) = -2(t-y)x_i \qquad (2.19)$$

(Jong Jek Siang, 2004)

Maka perubahan bobot adalah :

$$\Delta w_i = \alpha(t-y) \qquad (2.20)$$

(Jong Jek Siang, 2004)

Dan perubahan bias adalah :

$$\Delta b_i = \alpha(t-y) \qquad (2.21)$$

(Jong Jek Siang, 2004)

dengan α merupakan bilangan positif kecil (umumnya diambil = 0,1)

Algoritma Pelatihan *Adaline* (Jong Jek Siang, 2004)

1. Inisialisasi semua bobot dan bias (umumnya $w_1 = b = 0$)
   Tentukan laju pemahaman (α). Untuk penyederhanaan biasanya α diberi nilai kecil (α =0.1)
2. Selama $max_i \Delta w_i > batas\ toleransi$:
   a. Set aktifasi unit masukan $x_i = s_i$ (i=1,…,n)
   b. Hitung respon unit keluaran dan fungsi aktifasi berdasarkan persamaan (2.16) dan (2.17)
   c. Perbaiki bobot dan bias yang mengandung kesalahan (y≠t) berdasarkan persamaan (2.20) dan (2.21)

Algoritma Pengujian *Adaline* (Jong Jek Siang, 2004)

1. Inisialisasi semua bobot dan bias dengan bobot dan bias hasil pelatihan.
2. Untuk setiap input masukan bipolar x, lakukan :
   a. Set aktifasi unit masukan $x_1 = s_1$ (i=1, ….n)
   b. Hitung *net* vektor keluaran dengan persamaan (2.16)
   c. Kenakan fungsi aktifasi dengan persamaan (2.17)

d. *Backpropagation*

Backpropagation pertama kali diperkenalkan oleh Rumelhart, Hinton dan William pada tahun 1986, kemudian Rumelhart dan Mc Clelland mengembangkannya pada tahun 1988. Algoritma *Backpropagation* untuk *neural network* umumnya diterapkan pada *perceptron* berlapis banyak (*multilayer perceptrons*). *Perceptron* paling tidak mempunyai bagian input, bagian output dan beberapa lapis yang berada di antara input dan output. Lapis di tengah ini yang juga dikenal dengan lapis tersembunyi (*hidden layers*), bisa satu, dua, tiga dan seterusnya. Dalam praktek, banyaknya *hidden layer* paling banyak adalah tiga lapis. Dengan tiga lapis ini hampir semua permasalahan dalam dunia industri telah bisa diselesaikan. Output lapis terakhir dari hidden *layer* langsung dipakai sebagai output dari *neural network*.

Ada tiga tahap yang harus dilakukan dalam pelatihan jaringan, yaitu tahap perambatan maju (*forward propagation*), tahap perambatan-balik *(backward propagation),* dan tahap perubahan bobot dan bias. .

Algoritma Pelatihan *Backpropagation* (T. Sutojo, 2011)

1. Insialisasi bobot (ambil nilai *random* yang cukup kecil)
2. Selama kondisi berhenti bernilai salah, kerjakan:

   Tahap 1: Perambatan Maju (*Forward propagation*)

a. Setiap unit input (x_i, i=1,2,3,…,n) menerima sinyal xi dan meneruskan sinyal tersebut ke semua unit pada lapisan tersembunyi.

b. Setiap unit tersembunyi (z_j, j=1,2,3,…,p) menjumlahkan bobot sinyal input dengan persamaan berikut,

$$z_{inj} = v_{0j} \sum_{i=1}^{n} x_i v_{ij} \quad (2.22)$$

Dan menerapkan fungsi aktifasi untuk menghitung sinyal output-nya:

$$z_j = f(z_{inj}) = v_{0j} \sum_{i=1}^{n} x_i v_{ij} \quad (2.23)$$

Biasanya fungsi aktifasi yang digunakan adalah fungsi sigmoid, kemudian mengirimkan sinyal tersebut ke semua unit output.

c. Setiap unit output (y_k, k=1,2,3,…,m) menjumlahkan bobot sinyal input.

$$y_{ink} = w_{0k} \sum_{i=1}^{n} z_i w_{ij} \quad (2.24)$$

Dan menerapkan fungsi aktifasi untuk menghitung sinyal output-nya:

$$y_k = f(y_{ink}) = w_{0k} \sum_{i=1}^{n} z_i v_{ij} \quad (2.25)$$

Tahap 2: Perambatan Balik *(Back propagation)*

a. Setiap unit output (y_k, k=1,2,3,…,m) menerima pola target yang sesuai dengan pola input pelatihan, kemudian hitung eror dengan persamaan berikut.

$$\delta_k = (t_k - y_k) f'(y_{ink}) \quad (2.26)$$

f' adalah turunan dari fungsi aktifasi kemudian hitung koreksi bobot dengan persamaan berikut,

$$\Delta w_{jk} = \alpha \delta_k z_j \quad (2.27)$$

b. Setiap unit tersembunyi (z_j, j=1,2,3,…,p) menjumlahkan delta input-nya:

$$\delta_{inj} = \sum_{k=1}^{m} \delta_i w_{ij} \quad (2.28)$$

Untuk menghitung informasi *error*, kalikan nilai ini dengan turunan dari fungsi aktifasinya:

$$\delta_j = \delta_{inj} f(z_{inj}) \quad (2.29)$$

kemudian hitung koreksi bobot dengan persamaan berikut:

$$\Delta v_{jk} = \alpha \delta_j x_j \quad (2.30)$$

Setelah itu, hitung koreksi bias dengan persamaan berikut:

$$\Delta v_{0j} = \alpha \delta_j \quad (2.31)$$

Tahap 3: Perubahan Bobot dan Bias

a. Setiap unit output (y_k, k=1,2,3,…,m) dilakukan perubahan bobot dan bias (j=0,1,2,…,p) dengan persamaan berikut. (2.32)

$$w_{jk} \text{ (baru)} = w_{jk} \text{ (lama)} + \Delta w_{jk}$$

Setiap unit tersembunyi (z_j, j=1,2,3,…,p) dilakukan perubahan bobot dan bias (i=0,1,2,…,n) dengan persamaan berikut.

$$v_{ij} \text{ (baru)} = v_{ij} \text{ (lama)} + \Delta v_{ij} \quad (2.33)$$

b. Tes kondisi berhenti.

Algoritma Pengujian *Backpropagation* (T. Sutojo, 2011)

1. Gunakan bobot dan bias *input layer* (v_{ij}) dan *hidden layer* (w_{ij}) hasil pelatihan.
2. Setiap unit input (x_i, i=1,2,3,…,n) menerima sinyal x_i dan meneruskan sinyal tersebut ke semua unit pada lapisan tersembunyi.
3. Setiap unit tersembunyi (z_j, j=1,2,3,…,p) menjumlahkan bobot sinyal input dengan persamaan (2.22)

Dan menerapkan fungsi aktifasi untuk menghitung sinyal output-nya dengan persamaa (2.23)

4. Setiap unit output (y_k, k=1,2,3,…,m) menjumlahkan bobot sinyal input dengan persamaan (2.24)

   Dan menerapkan fungsi aktifasi untuk menghitung sinyal output-nya dengan persamaa (2.25)

2.1.2 Algoritma Pembelajaran Tanpa Supervisi (*Unsupervised*)

Pada metode pembelajaran yang tidak terawasi, tidak memerlukan target output. Pada metode ini tidak dapat ditentukan hasil seperti apa yang diharapkan selama proses pembelajaran. Selama proses pembelajaran, nilai bobot disusun dalam suatu *range* tertentu tergantung pada nilai input yang diberikan. Tujuan pembelajaran ini adalah mengelompokkan unit-unit yang hampir sama dalam suatu area tertentu. Pembelajaran seperti ini biasanya sangat cocok untuk pengelompokkan (klasifikasi) pola. Pada jaringan ini, suatu lapisan yang berisi *neuron-neuron* akan menyusun dirinya sendiri berdasarkan input nilai tertentu dalam suatu kelompok yang dikenal dengan istilah *cluster*. Selama proses penyusunan diri, *cluster* yang memiliki vektor bobot paling cocok dengan pola input akan terpilih sebagai pemenang. Diantara metode JST yang masuk kategori ini antara lain adalah : *Competitive*, *Hebbian*, *Kohonen*, *LVQ*, *Neocognitron*, dan lain-lain.

Berdasarkan strategi pembelajaran tesebut, ada beberapa metode yang dapat dikategorikan ke dalam kedua jenis pembelajaran supervisi dan tanpa supervisi.

2.2 Peramalan

Menurut Assauri, peramalan adalah kegiatan untuk memperkirakan apa yang akan terjadi di masa yang akan datang. Sedangkan ramalan adalah suatu situasi atau kondisi yang diperkirakan akan terjadi pada masa yang akan datang. Menurut Khotler, peramalan menjadikan pengelolaan dari suatu variabel di masa datang

akan terlihat, sehingga mempermudah dalam perencanaan-perencanaan untuk periode yang akan datang.

Setiap kebijakan perusahaan tidak akan terlepas dari usaha untuk meningkatkan kesejahteraan masyarakat atau meningkatkan keberhasilan perusahaan untuk mencapai tujuan pada masa yang akan datang, dimana kebijakan tersebut dilaksanakan. Oleh karena itu perlu dilihat dan dikaji situasi dan kondisi pada saat kebijakan tersebut dilaksanakan. Usaha untuk melihat dan mengkaji situasi dan kondisi tersebut tidak terlepas dari kegiatan peramalan.

Di dalam usaha mengetahui atau melihat perkembangan di masa depan, peramalan dibutuhkan untuk menentukan kapan suatu peristiwa akan terjadi atau suatu kebutuhan akan timbul, sehingga dapat dipersiapkan kebijakan yang perlu dilakukan. Selain itu ramalan dibutuhkan untuk memberikan informasi kepada pimpinan sebagai dasar untuk membuat suatu keputusan.

Menurut Assauri, metode peramalan adalah cara memperkirakan secara kuantitatif apa yang akan terjadi pada masa depan, berdasarkan pada data yang relevan pada masa lalu. Oleh karena metode peramalan didasarkan atas data yang relevan pada masa lalu, maka metode peramalan ini dipergunakan dalam peramalan yang objektif.

Menurut Henke dan Reitch, terdapat dua langkah dasar yang harus dilakukan dalam membuat atau menghasilkan suatu peramalan yang akurat dan berguna. Langkah dasar yang pertama adalah pengumpulan data yang relevan dengan tujuan peramalan yang dimaksud dan menurut informasi – informasi yang dapat menghasilkan peramalan yang akurat. Langkah dasar yang kedua adalah memilih metode peramalan yang tepat yang akan digunakan dalam mengolah informasi yang terkandung dalam data yang telah dikumpulkan.

2.2.1 Jenis-jenis Peramalan

Menurut Assauri, peramalan dapat dibedakan dari beberapa segi tergantung dari cara melihatnya, yaitu dilihat dari jangka waktu ramalan dan dilihat dari sifat ramalan.

Jika dilihat dari jangka waktu ramalan yang disusun, maka ramalan dapat dibedakan atas dua macam, yaitu:

a. Peramalan jangka panjang, yaitu peramalan yang dilakukan untuk penyusunan hasil ramalan yang jangka waktunya lebih dari satu setengah tahun atau tiga semester.
b. Peramalan jangka pendek, yaitu peramalan yang dilakukan untuk penyusunan hasil ramalan dengan jangka waktu yang kurang dari satu setengah tahun, atau tiga semester.

Berdasarkan sifat ramalan yang telah disusun, maka peramalan dapat dibedakan atas dua macam, yaitu :

a. Peramalan kualitatif, yaitu peramalan yang didasarkan atas data kualitatif pada masa lalu. Hasil peramalan yang dibuat sangat tergantung pada orang yang menyusunnya. Hal ini penting karena hasil peramalan tersebut ditentukan berdasarkan pemikiran yang bersifat intuisi, *judgement* atau pendapat, dan pengetahuan serta pengalaman dari penyusunnya.
b. Peramalan kuantitatif, yaitu peramalan yang didasarkan atas data kuantitatif pada masa lalu. Hasil peramalan yang dibuat sangat tergantung pada metode yang dipergunakan dalam peramalan tersebut.

Metode peramalan kuantitatif dapat dibagi menjadi dua tipe, *causal* dan *time series.* Metode peramalan *causal* meliputi faktor-faktor yang berhubungan dengan variabel yang diprediksi seperti analisis regresi. Peramalan *time series* merupakan metode kuantitatif untuk menganalisis data masa lampau yang telah dikumpulkan secara teratur menggunakan teknik yang tepat. Hasilnya dapat dijadikan acuan untuk peramalan nilai di masa yang akan datang (Makridakis, 1999).

Model deret berkala dapat digunakan dengan mudah untuk meramal, sedang model kausal lebih berhasil untuk pengambilan keputusan dan kebijakan. Peramalan harus mendasarkan analisisnya pada pola data yang ada. Empat pola data yang lazim ditemui dalam peramalan (Materi Statistika, UGM):

a. Pola Horizontal

Pola ini terjadi bila data berfluktuasi di sekitar rata-ratanya. Produk yang penjualannya tidak meningkat atau menurun selama waktu tertentu termasuk jenis ini. Struktur datanya dapat digambarkan sebagai berikut ini.

b. Pola Musimam

Pola musiman terjadi bila nilai data dipengaruhi oleh faktor musiman (misalnya kuartal tahun tertentu, bulanan atau hari-hari pada minggu tertentu). Struktur datanya dapat digambarkan sebagai berikut ini.

c. Pola Siklis

Pola ini terjadi bila data dipengaruhi oleh fluktuasi jangka panjang seperti yang berhubungan dengan siklus bisnis.

d. Pola *Trend*

Pola *Trend* terjadi bila ada kenaikan atau penurunan sekuler jangka panjang dalam data.

2.2.2 Metode Peramalan

Model kuantitatif intrinsik sering disebut sebagai model-model deret waktu (*Time Series model*). Model deret waktu yang populer dan umum diterapkan dalam peramalan permintaan adalah rata-rata bergerak (*Moving Averages*), pemulusan eksponensial (*Exponential Smoothing*), dan proyeksi kecenderungan (*Trend Projection*). Model kuantitatif ekstrinsik sering disebut juga sebagai model kausal, dan yang umum digunakan adalah model regresi (*Regression Causal model*) (Gaspersz, 1998).

2.2.3 Ukuran Akurasi Peramalan

Model-model peramalan yang dilakukan kemudian divalidasi menggunakan sejumlah indikator. Indikator-indikator yang umum digunakan adalah rata-rata penyimpangan absolut (*Mean Absolute Deviation*), rata-rata kuadrat terkecil

(*Mean Square Error*), rata-rata persentase kesalahan absolut (*Mean Absolute Percentage Error*), validasi peramalan *(Tracking Signal),* dan pengujian kestabilan (*Moving Range).*

a. *Mean Absolute Deviation (MAD)*

Metode untuk mengevaluasi metode peramalan menggunakan jumlah dari kesalahan-kesalahan yang absolut. MAD mengukur ketepatan ramalan dengan merata-rata kesalahan dugaan (nilai absolut masing-masing kesalahan). MAD berguna ketika mengukur kesalahan ramalan dalam unit yang sama sebagai deret asli.

b. *Mean Square Error (MSE)*

MSE adalah metode lain untuk mengevaluasi metode peramalan. Masing-masing kesalahan atau sisa dikuadratkan. Kemudian dijumlahkan dan ditambahkan dengan jumlah observasi. Pendekatan ini mengatur kesalahan peramalan yang besar karena kesalahan-kesalahan itu dikuadratkan. Metode itu menghasilkan kesalahan-kesalahan sedang yang kemungkinan lebih baik untuk kesalahan kecil, tetapi kadang menghasilkan perbedaan yang besar.

c. *Mean Absolute Percentage Error (MAPE)*

MAPE dihitung dengan menggunakan kesalahan absolut pada tiap periode dibagi dengan nilai observasi yang nyata untuk periode itu. Kemudian, merata-rata kesalahan persentasi absolut tersebut. Pendekatan ini berguna ketika ukuran atau besar variabel ramalan itu penting dalam mengevaluasi ketepatan ramalan. MAPE mengindikasi seberapa besar kesalahan dalam meramal yang dibandingkan dengan nilai nyata.

d. *Tracking Signal*

Validasi peramalan dilakukan dengan *Tracking Signal*. *Tracking Signal* adalah suatu ukuran bagaimana baiknya suatu peramalan memperkirakan nilai-nilai aktual. *Tracking signal* yang positif menunjukan bahwa nilai

aktual permintaan lebih besar daripada ramalan, sedangkan *tracking signal* yang negatif berarti nilai aktual permintaan lebih kecil daripada ramalan.

e. Moving Range (MR)

Peta *MR* dirancang untuk membandingkan nilai permintaan aktual dengan nilai peramalan. Data permintaan aktual dibandingkan dengan nilai peramalan pada periode yang sama. Peta tersebut dikembangkan ke periode yang akan datang hingga dapat dibandingkan data peramalan dengan permintaan aktual. Peta *MR* digunakan untuk pengujian kestabilan sistem sebab-akibat yang mempengaruhi permintaan.

2.1.1 Kegunaan Peramalan

Kegunaan peramalan dapat dilihat pada saat pengambilan keputusan. Setiap orang selalu dihadapkan pada masalah pengambilan keputusan. Keputusan yang baik adalah keputusan yang didasarkan atas pertimbangan apa yang akan terjadi pada waktu keputusan itu dilaksanakan, berorientasi ke depan. Apabila kurang tepat ramalan yang disusun atau yang kita buat, maka makin kurang baiklah keputusan yang kita ambil. Oleh karena itu masalah pengambilan keputusan merupakan masalah yang selalu kita hadapi.

Peranan peramalan cukup penting, baik dalam penelitian, perencanaan maupun dalam pengambilan keputusan. Oleh karena itu, ketepatan dari ramalan tersebut merupakan hal yang sangat penting. Walaupun demikian perlu disadari bahwa suatu peramalan adalah tetap ramalan, di mana selalu ada unsur kesalahannya dan belum pernah mencapai efisiensi 100 % tepat. Sehingga yang penting diperhatikan adalah usaha untuk memperkecil kemungkinan kesalahannya tersebut. Akhirnya, baik tidaknya suatu ramalan yang disusun sangat tergantung pada pihak-pihak yang melakukannya, langkah-langkah peramalan yang dilakukannya, dan metode yang dipergunakan.

2.2 Metodologi Pembuatan Perangkat Lunak

Tahap pengembangan perangkat lunak menggunakan metode *waterfall* sering juga disebut model sekuensial *linier* (*sequential linear*) atau alur hidup klasik (*classic life cycle*). Model *waterfall* menyediakan pendekatan alur hidup perangkat lunak secara *sequential* atau terurut dimulai dari analisis, *design*, pengkodean, pengujian dan tahap pendukung (*support*) (Rosa dan M. Salahudin, 2011).

a. Analisis kebutuhan perangkat lunak

   Proses pengumpulan kebutuhan dilakukan secara intensif untuk men-spesifikasikan kebutuhan perangkat lunak agar dapat dipahami perangkat lunak seperti apa yang dibutuhkan oleh *user*. Spesifikasi kebutuhan perangkat lunak pada tahap ini perlu untuk didokumentasikan.

b. Perancangan sistem (*Design*)

   Desain perangkat lunak adalah proses multilangkah yang fokus pada desain pembuatan program perangkat lunak, representasi perangkat lunak termasuk struktur data, arsitektur perangkat lunak, representasi antarmuka dan prosedur pengkodean. Tahap ini mentranslasi kebutuhan perangkat lunak dari tahap analisis kebutuhan ke representasi desain agar dapat diimplementasikan menjadi program pada tahap selanjutnya. Desain perangkat lunak yang dihasilkan pada tahap selanjutnya. Desain perangkat lunak yang dihasilkan pada tahap ini juga perlu didokumentasikan.

c. Pembuatan kode program

   Desain harus ditranslasikan ke dalam program perangkat lunak. Hasil dari tahap ini adalah program komputer sesuai dengan desain yang telah dibuat pada tahap desain.

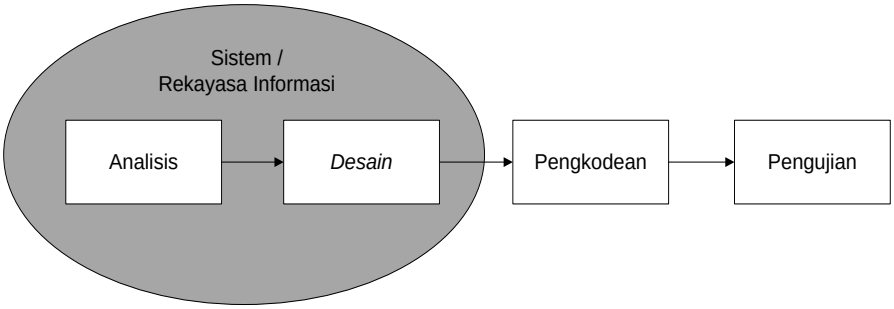
d. Pengujian

   Pengujian fokus pada perangkat lunak dari segi logik dan fungsional dan memastikan bahwa semua bagian sudah diuji. Hal ini dilakukan untuk

meminimalisir kesalahan (*error*) dan memastikan keluaran yang dihasilkan sesuai dengan keinginan.

e. Pendukung atau pemeliharaan (*Maintenance*)

Tidak menutup kemungkinan sebuah perangkat lunak mengalami perubahan ketika sudah dikirimkan ke pengguna(*user*). Perubahan bisa terjadi karena adanya kesalahan yang muncul dan tidak terdeteksi pada saat pengujian atau perangkat lunak harus beradaptasi dengan liungkungan baru. Tahap pendukung atau pemeliharaan dapat mengulangi proses pengembangan mulai dari analisis spesifikasi untuk perubahan perangkat lunak yang sudah ada, tetapi tidak untuk membuat perangkat lunak baru. Gambar 2.16 menunjukan paradigm *waterfall*.



Gambar 2.16 Alur Metodologi *Waterfall*
(Presman, 2001)

2.3 *Object Oriented Programming (OOP)*

OOP yang dijelaskan di bagian ini *meliputi class, object, atribut*, dan m*ethod.* Keempat bagian tersebut merupakan konsep dasar dari OOP.

Class adalah kumpulan objek yang memiliki atribut yang sama, *class* dapat dikatakan sebagai *template* untuk membuat objek atau *proptotipe* (*blue print*) yang mendefinisikan variable-varibael dan metode-metode secara umum. Objek merupakan hasil instansiasi dari *class.* Objek disebut juga *instances.* Proses pembuatan objek disebut *instantiation*.

Atribute adalah data yang membedakan antara objek yang satu dengan objek yang lain. Di dalam *class, atribute* disebut juga sebagai variabel.

Method adalah *statement* dalam suatu *class* yang meng-*handle* suatu suatu *task*. Cara objek berkomunikasi dengan objek lain adalah dengan menggunakan *method*.

2.4 Konsep Basis Data

Sistem basis data adalah suatu sistem menyusun dan mengelola *record-record* menggunakan komputer untuk menyimpan atau merekam serta memelihara data opersional lengkap sebuah organisasi/perusahaan sehingga mampu menyediakan informasi yang optimal yang diperlukan pemakai untuk proses mengambil keputusan.

Berikut adalah beberapa pengertian dari basis data yang dikembangkan atas dasar sudut pandang yang berbeda, yaitu:

a. Basis data adalah kumpulan data-data (*file*) *non-redudant* yang saling terkait satu sama lainnya (dinyatakan oleh atribut-atribut kunci dari tabel-tabelnya/ struktur data dan relasi-relasi) di dalam usaha membentuk bangunan informasi yang penting (*enterprise*).
b. Basis data adalah himpunan kelompok data (*file*/arsip) yang saling berhubungan dan diorganisasikan sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah.
c. Basis data adalah kumpulan data yang saling berhubungan dan disimpan bersama sedemikian rupa tanpa pengulangan yang tidak perlu (*redundancy*) untuk memenuhi berbagai kebutuhan.
d. Basis data adalah kumpulan *file*, *table*, arsip yang saling berhubungan dan disimpan di dalam media penyimpanan elektronik.

Menurut Connoly dan Begg, basis data dapat diartikan sebagai sekumpulan data atau entitas tentang suatu benda beserta deskripsinya yang berhubungan satu sama lain secara logika. Basis data dirancang dan diorganisasikan untuk memnuhi logika dan informasi pada suatu organisasi. Semua data di dalam basis data saling terintegrasi sehingga jumlah duplikasi dapat diminimalkan.

Untuk mengelola basis data diperlukan perangkat lunak yang disebut *Database Management System* (DBMS). DBMS adalah perangkat lunak sistem

yang memungkinkan para pemakai membuat, memelihara, mengontrol, dan mengakses basis data dengan cara yang praktis dan efisien.

Pengelolaan basis data secara fisik tidak dilakukan oleh pemakai secara langsung, tetapi ditangani oleh sebuah perangkat lunak (sistem) yang khusus atau spesifik. Perangkat lunak ini disebut Sistem Manajemen Basis Data (SMBD) atau sering juga disebut sebagai DBMS (*Database Manajemen System*). SMBD adalah perangkat lunak yang memungkinkan para pemakai untuk mendeskripsikan, membuat, memelihara, serta mengendalikan akses ke basis data.

2.4.1 Tujuan Basis Data

Tujuan dari basis data adalah:

a. Kemudahan dan kecepatan dalam pengambilan kembali data/arsip
b. Efisiensi ruang dan waktu
c. Keakuratan data
d. Ketersediaan untuk proses pengambilan data yang diperlukan tiap saat
e. Kelengkapan data-data yang diperlukan atau yang tersimpan
f. Keamanan data
g. Kebersamaan

2.4.2 Keuntungan Basis Data

Keuntungan dari basis data adalah :

a. Mereduksi redudansi yang akibatnya mengurangi inkonsistensi.
b. Data dapat di-*share* antar aplikasi.
c. Standarisasi data dapat dilakukan.
d. Batasan *security* dapat diterapkan.
e. Mengelola integritas (keterjaminan akurasi) data.
f. Menyeimbangkan kebutuhan yang saling konflik.
g. Independensi data (objektif DBS) : kekebalan aplikasi terhadap perubahan struktur penyimpanan dan teknik pengaksesan data (basis data harus dapat berkembang tanpa mempengaruhi aplikasi yang ada).

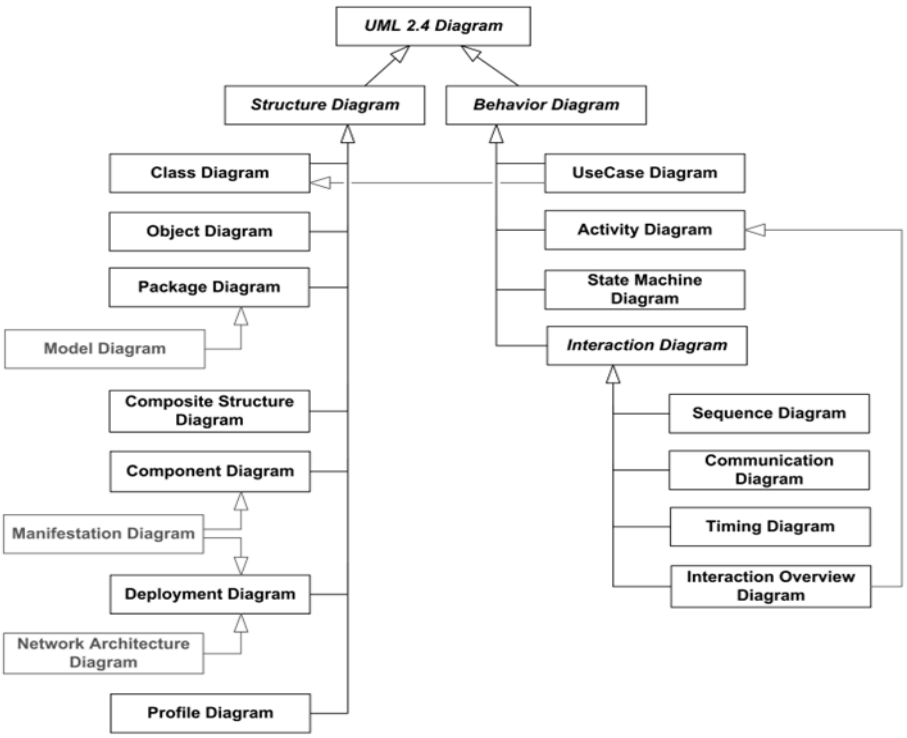
2.5 *Unified Modelling Language (UML)*

Dalam sebuah pembuatan sistem dibutuhkan suatu bentuk model untuk menggambarkan hasil analisis dan perancangan sistem. *Tools* pemodelan yang digunakan dalam dokumen ini adalah UML. UML adalah bahasa untuk menjelaskan, menggambarkan, merancang, dan mendokumentasikan artifak dari sistem software serta pemodelan bisnis dan sistem *non* software lainnya. (Larman, Graig : 2004). Adapun pertimbangan menggunakan UML adalah sebagai berikut :

a. UML merupakan alat pemodelan yang telah menjadi notasi standar dalam melakukan analisis dan perancangan sistem yang berorientasi objek.
b. UML dapat digunakan pada berbagai tahap pengembangan perangkat lunak dan pada berbagai teknologi implementasi sistem.
c. Menambah wawasan keilmuan dan teknologi.

Pada UML versi 2.4 terdapat 2 kategori diagram (lihat Gambar 2.17) sebagai berikut:

a. *Structure Diagram* yaitu kumpulan diagram untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
b. *Behaviour Diagram* yaitu kumpulan diagram untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.



Gambar 2.17 Gambar Diagram UML

c. *Use Case Diagram*

Use case diagram merupakan permodelan untuk menggambarkan kelakuan (*behavior*) sistem yang akan dibangun. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibangun. *Use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Ada dua hal utama pada *use case* yaitu aktor dan *use case*.

1. Aktor merupakan orang, proses atau sistem lain yang berinteraksi dengan sistem yang akan dibangun.
2. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antarunit atau aktor

d. *Activity Diagram*

Activity diagram menggambarkan berbagai alir aktifitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang

mungkin terjadi, dan bagaimana aktifitas berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

Activity diagram merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu, *activity diagram* tidak menggambarkan *behaviour internal* sebuah sistem secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktifitas dari level atas secara umum.

Sebuah aktifitas dapat direalisasikan oleh satu *use case* atau lebih. Aktifitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktifitas.

Activity diagram digambar menggunakan segiempat dengan sudut membulat untuk menggambarkan aktifitas. *Decision* untuk menggambarkan *behaviour* pada kondisi tertentu. *Activity diagram* dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktifitas tertentu.

e. *Sequence Diagram*

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem berupa pesan yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respon dari sebuah kejadian untuk menghasilkan output tertentu. Diawali dari apa yang men-*trigger* aktifitas 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 metoda dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message*.

f. *Class Diagram*

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan perancangan berorientasi objek. *Class* menggambarkan keadaan (atribut) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metode). Sebuah Kelas memiliki struktur sebagai berikut:

1) Nama Kelas/*Stereotype*
2) *Atribute*, yaitu variabel-variabel yang dimiliki suatu *class*.
3) *Method*, yaitu fungsi-fungsi yang dimiliki oleh suatu *class*.

Class Diagram menggambarkan struktur sistem dari segi pendefinisian *class-class* yang akan dibuat untuk membangun sistem.

Class diagram mempunyai 3 macam *relationalships* (hubungan), sebagai berikut :

1) *Association*

   Assosiation adalah hubungan antara bagian dari dua *class*. Terjadi *association* antara dua *class* jika salah satu bagian dari *class* mengetahui yang lainnya dalam melakukan suatu kegiatan. Di dalam diagram, sebuah *association* adalah penghubung yang menghubungkan dua *class*.

2) *Aggregation*

   Aggregation adalah *association* dimana salah satu *class*-nya merupakan bagian dari suatu kumpulan. *Aggregation* memiliki titik pusat yang mencakup keseluruhan bagian.

3) *Generalization*

Generalization adalah suatu hubungan turunan dengan mengasumsikan satu *class* merupakan suatu *Super Class* dari kelas yang lain. *Generalization* memiliki tingkatan yang berpusat pada *Super Class*.

2.6 Pembangunan Perangkat Lunak

Pembanguna perangkat lunak menggunakan .NET Framework, bahasa pemrograman C#, dengan editor Microsoft Visual Studio 2010, Library ZedGraph, dan database MySQL dengan package xampp.

2.6.1 *Dot NET Framework*

Microsoft mengumumkan C# pada tahun 2000 bersamaan dengan pengumuman *platform Dot NET Framework*. *Dot NET Framework* adalah *platform* yang merupakan perwujudan teknologi modern. *Dot NET Framework* diciptakan untuk dapat memecahkan masalah yang banyak dihadapi dunia pemrograman secara lebih efisien. Metodologi perancangan berorientasi objek dan konsep *software* sebagai komponen tertanam dengan kuat di *Dot NET Framework*.

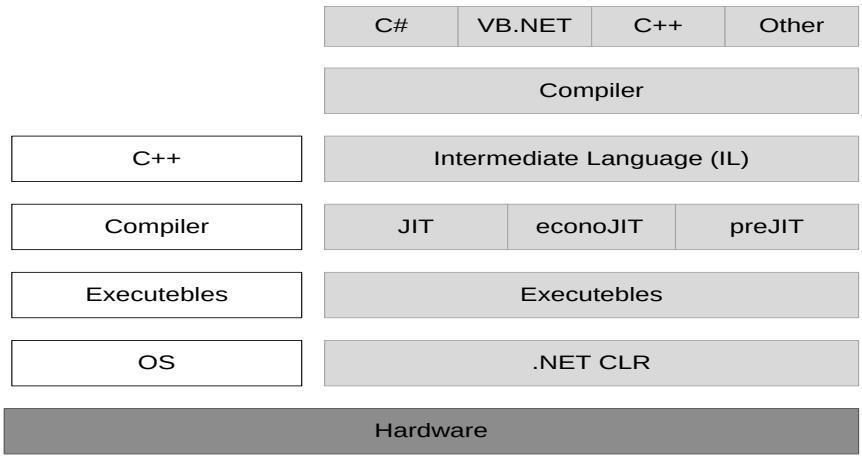
Dua bagian penting dari *Dot NET Framework* adalah *Common Language Runtime* (CLR) dan *Class Library Dot NET*.

a. *CLR*

CLR adalah bagian dari *Framework* .NET yang mengelola program-program .NET yang dijalankan. *CLR* inilah yang mengatur hal-hal seperti pengalokasian memori, pengecekan *type*, dan keamanan.

Dalam .NET, program tidak dikompilasi kedalam *file executable* (.exe) melainkan dikompilasi ke *Intermediate Language* (IL), oleh karena itu program-program .NET dapat dijalankan di sistem operasi apa saja asalkan pada sistem operasi yang bersangkutan terpasang .NET Framework. Program .NET dengan CLR bisa dianalogikan seperti program Java dengan

JVM. Cara kerjanya adalah ketika program .NET dijalankan, CLR akan memanggil *Just In Time Coompiler (JIT)* untuk mengompile IL dalam program tersebut agar menjadi kode mesin di sistem operasi yang bersangkutan. Keseluruhan program tersebut tidak di-*compile* oleh JIT sekaligus, melainkan sesuai kebutuhan. Hal inilah yang membuat program-program .NET dapat berjalan dengan kecepatan tinggi karena program tersebut dijalankan secara *native* pada sistem operasi yang bersangkutan. IL juga merupakan salah satu faktor yang memungkinkan tercapainya *cross language interperability*, yaitu kemampuan sebuah program untuk berinteraksi dengan program yang dibuat menggunakan bahasa lain.



Gambar 2.18 Struktur CLR

b. *Class Library Dot NET*

Di dalam Framework .NET terdapat banyak *class library* mulai dari *class* yang berisi fungsi-fungsi matematika, *class* yang berhubungan dengan keamanan, *class* untuk membuat program dengan *Graphic User Interface* (GUI) sampai pada *class* yang berhubungan dengan input/output.

2.6.2 Bahasa Pemrograman C#

C# (dibaca: C sharp atau see sharp) sering dianggap sebagai bahasa penerus C++ atau versi canggih dari C++, karena ada anggapan bahawa tanda # adalah

perpaduan dari 4 buah tanda tambah (+) yang disusun sedemikian rupa sehingga membentuk tanda #. Akan tetapi terlepas dari benar tidaknya anggapan tersebut, C# adalah sebuah bahasa pemrograman yang berorientasi objek yang dikembangkan oleh Microsoft dan menjadi salah satu bahasa pemrograman yang mendukung .Net *programming* melalui Visual Studio.

C# didasarkan pada bahasa pemrograman C++, C# juga memiliki kemiripan dengan beberapa bahasa pemrograman seperti Visual Basic, Java, Delphi, dan tentu saja C++. C# memiliki kemudahan *syntax* seperti Visual Basic dan tentu saja ketangguhan seperti Java dan C++. Kemiripan-kemiripan ini tentuya memudahkan *programmer* dari berbagai latar belakang bahasa pemrograman tidak perlu waktu yang lama untuk menguasainya karena bagaimanapun juga C# lebih sederhana dibandingkan bahasa pemrograman seperti C++ dan Java (Wahana Komputer, 2008).

2.6.3 Microsoft Visual Studio 2010

Microsoft Visual Studio 2010 adalah "alat" bagi para pengembang software yang sarat dengan fungsi-fungsi yang memanjakan para pengembang software untuk menyelesaikan proyek besar maupun proyek kecil untuk teknologi .Net. Rahasia kenyamanan Visual Studio .Net adalah pada *Integrated Development Environment* (IDE)-nya.

IDE adalah sebuah *tools* di mana *developper* program dapat mengembangkan programnya dengan lebih mudah. Fasilitas-fasilitas IDE adalah:

a. Pemberian warna yang berbeda untuk setiap *syntax* dan *keyword.*
b. *Intellisense* (*autocomplete*), fasilitas ini menolong *developer* untuk melengkapi secara otomatis semua penulisan objek, properti, *method*, dan *keyword* begitu Anda memberi tanda titik (.) pada program sebagai tanda pemisah antara objek, properti, *method*, atau yang lain.
c. Manajemen proyek dan solusi yang terdiri atas berbagai macam *file*.
d. Kemudahan membuata *user interface* dengan fasilitas *drag* dan *drop*.

e. Tab properti yang memudahkan *programmer* untuk melakukan pengaturan properti-properti dari berbagai macam kontrol seperti kontrol window atau web.
f. Fasilitas *debug* program yang memudahkan *programmer* dapat melihat apa yang terjadi sewaktu program berjalan.
g. *Hot compiler,* fasilitas ini melakukan pemeriksaan terhadap kode program yang ditulis *programmer*. Jika terdapat kesalahan pada kode tersebut maka kode akan diberi tanda.
h. *Help* yang dinamik, di mana pun kursor berada. Jika menekan tombol *help* (F1) maka MSDN akan menyediakan topik *help* berdasarkan *keyword* yang ada pada baris kursor.
i. Kompilasi dan membangun aplikasi dengan mudah dan cepat.
j. Eksekusi program dengan atau tanpa *debugger*.
k. *Deployment* apliaksi yang mudah, bisa lewat disk atau internet.

2.6.4 *Library ZedGraph*

ZedGraph is a class library, Windows Forms UserControl, and ASP web-accessible control for creating 2D line, bar, and pie graphs of arbitrary datasets. The classes provide a high degree of flexibility - almost every aspect of the graph can be user-modified. At the same time, usage of the classes is kept simple by providing default values for all of the graph attributes. The classes include code for choosing appropriate scale ranges and step sizes based on the range of data values being plotted. Moreover, ZedGraph is compatible with .NET 2.0, and VS .NET 2005. (www.codeproject.com/A-flexible-charting-library-for-NET.htm. Waktu akses Senin, 17 Desember 2012 pukul 10:35:11).

2.6.5 MySQL

MySQL dibuat dan dikembangkan oleh MySQL AB yang berada di Swedia. MySQL dapat digunakan untuk membuat dan mengola *database* beserta isinya. Kita dapat memanfaatkan MySQL untuk menambahkan, mengubah dan menghapus data yang berada dalam *database*. MySQL merupakan sistem

manajemen *database* yang bersifat *relational*. Artinya data-data yang dikelola dalam *database* akan diletakkan pada beberapa tabel yang terpisah sehingga manipulasi data akan menjadi jauh lebih cepat. MySQL dapat digunakan untuk mengelola *database* mulai dari yang kecil sampai dengan yang sangat besar. MySQL juga dapat menjalankan perintah-perintah *Structured Query Language* (SQL) untuk mengelola *database-database* yang ada di dalamnya. Hingga kini, MySQL sudah berkembang hingga versi 5.5.27 dan bersifat *open source* dan sudah mendukung *trigger* untuk memudahkan pengelolaan tabel dalam *database*. *Tool* MySQL menggunakan *package* xampp versi 1.8.1 dan SQLYog versi 9.10 Beta l sebagai *interface* untuk mengelolah *store procedure*.