BAB II

TINJAUAN PUSTAKA

II.1 Landasan Teori

II.1.1 Image Recognition

Citra (*image*) adalah gambar pada bidang dua dimensi yang dihasilkan dari gambar analog dua dimensi yang kontinyu menjadi gambar diskrit melalui proses digitasi.

Citra sebagai keluaran dari suatu sistem perekam data dapat bersifat :

1. Optik berupa foto
2. Analog berupa sinyal video seperti gambar pada TV
3. Digital yang dapat langsung disimpan pada storage device

Sebagai hasil keluaran, citra dapat berupa :

1. Citra diam *(still image)*
2. Citra bergerak *(moving image)*

Contoh citra :



Gambar II.1 Gambar Diam (Still Image)

Recognition yaitu suatu tindakan untuk mengenali sesuatu, baik benda maupun lainnya.

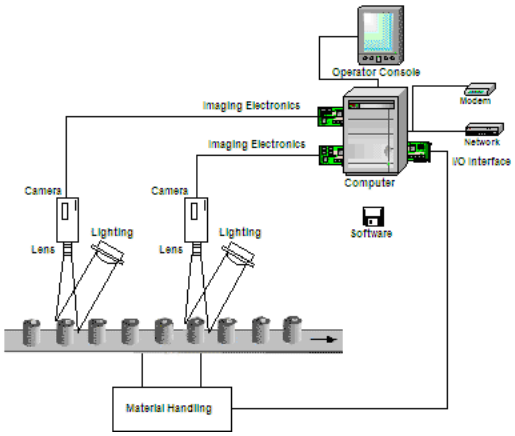
Image recognition adalah kemampuan mesin atau program untuk mengidentifikasi pola dan bentuk dalam suatu gambar dan mengkonversikannya ke format yang dapat dibaca oleh mesin.

II.1.2 Pengenalan Pola (*Pattern Recognition*)

Pengenalan pola adalah disiplin ilmu yang mengklasifikasikan *object* berdasar *image*, berat atau parameter-parameter yang telah ditentukan ke dalam sejumlah kategori atau kelas. Pengenalan pola meliputi berbagai aplikasi dan implementasi dalam kasus-kasus *real world*. Contoh aplikasi yang menerapkan pengenalan pola adalah sebagai berikut:

II.1.2.1 Machine Vision

Pengenalan pola menjadi dasar dari sistem mesin ini. Mesin ini menangkap sebuah atau sekelompok *object* dengan kamera dan selanjutnya dianalisa untuk di deskripsikan *object* atau benda tersebut.



Gambar II.2 Machine Vision

II.1.2.2 Character Recognition (OCR)

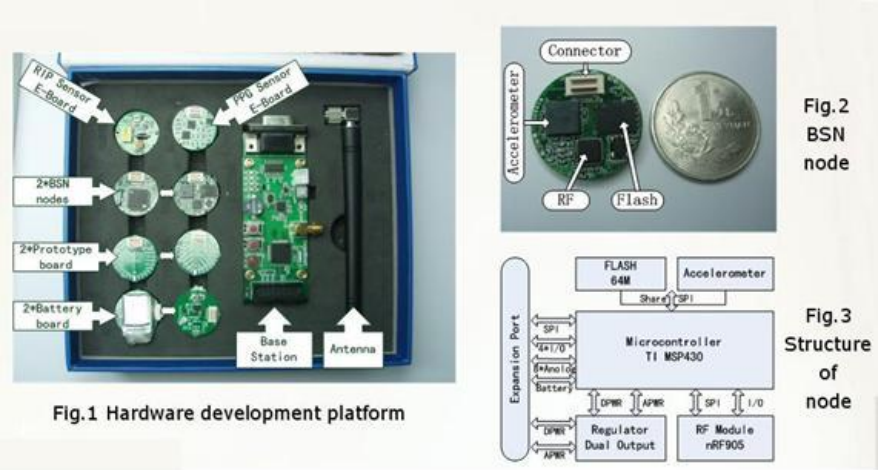
Proses konversi *scan* gambar mesin-cetak atau teks tulisan tangan (angka, huruf, dan simbol) ke dalam format komputer-*processable* juga dikenal sebagai pengenalan karakter optik (OCR). Sebuah sistem OCR tipikal mengandung tiga komponen logis: pemindai gambar, perangkat lunak OCR dan *hardware*, dan *interface output*. Pemindai gambar optik menangkap gambar teks untuk diakui. Teks gambar diproses dengan perangkat lunak OCR dan *hardware*. Proses ini melibatkan tiga operasi: analisis dokumen (gambar penggalian karakter individu), mengenali gambar tersebut (berdasarkan bentuk), dan pengolahan kontekstual (baik untuk memperbaiki *misclassifications* dilakukan oleh algoritma pengakuan atau untuk membatasi pengakuan pilihan). *Interface output* bertanggung jawab untuk komunikasi hasil sistem OCR ke dunia luar.



Gambar II.3 Character Recognition (OCR)

II.1.2.3 Computer Aided Diagnosis

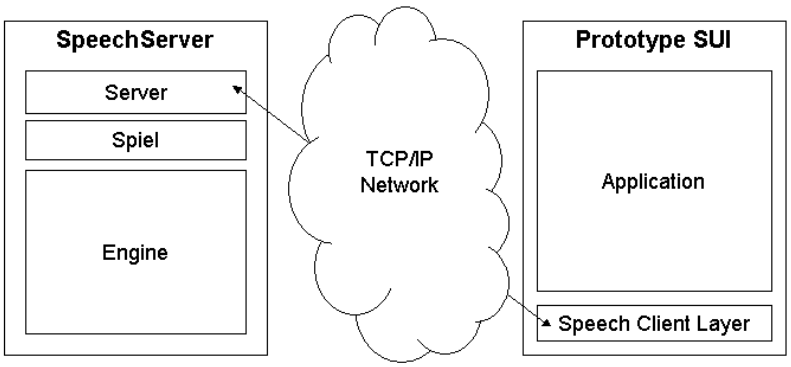
Definisi *Computer Aided Diagnosis* (CAD), ditemukan dalam sastra, dapat berupa sebuah diagnosis yang dibuat oleh seorang ahli radiologi menggunakan output dari skema terkomputerisasi untuk gambar otomatis analisis sebagai alat bantu diagnostik. Adapun maksud dari CAD yaitu untuk mendukung dan bukan menggantikan manusia sebagai pengamat dalam analisis gambar radiografi.



Gambar II.4 Computer Aided Diagnosis (CAD)

II.1.2.4 Speech Recognition

Pengenalan pola suara adalah salah satu aplikasi yang berkembang saat ini. Sistem ini mengijinkan kita untuk berkomunikasi antara manusia dengan memasukkan data ke komputer. Meningkatkan efisiensi industri manufaktur, mengontrol mesin dengan berbicara pada mesin itu. Secara proses, kata yang diucapkan adalah digital (berubah menjadi urutan angka) dan cocok dengan kamus kode untuk mengidentifikasi kata-kata. Kebanyakan sistem harus "dilatih," sampel membutuhkan dari semua kata yang sebenarnya yang akan digunakan oleh pengguna sistem. Kata-kata sampel digital, disimpan di komputer dan digunakan untuk perbandingan untuk kata-kata yang baru. Sistem canggih lainnya memerlukan suara sampel, tetapi bukan dari setiap kata. Sistem ini menggunakan sampel suara bersama dengan kamus kosakata yang lebih besar untuk mencocokkan kata-kata yang masuk. Namun sistem lain bertujuan untuk menjadi "pembicara-independen" yaitu mereka akan mengenali kata-kata dalam kosakata mereka dari speaker tanpa pelatihan. Dikatakan (tahun 1994) bahwa komputer perlu sesuatu seperti 1000 kali lebih cepat sebelum kosakata besar (beberapa ribu kata-kata), speaker-independen, pengenalan suara suara yang terhubung akan layak.



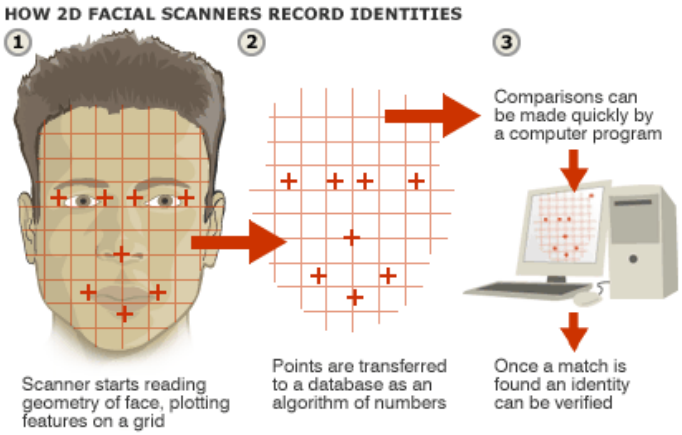
Gambar II.5 Speech Recognition

II.1.2.5 Face Recognition

Subyek pengenalan wajah seumur dengan visi komputer, baik karena kepentingan praktis dari topik dan keteoritisan dari ilmuwan kognitif. Meskipun kenyataan bahwa metode lain untuk identifikasi (seperti sidik jari atau *scan* iris) dapat lebih akurat, pengenalan wajah selalu tetap menjadi fokus utama penelitian karena non-invasif sifat dan karena utama metode penduduk identifikasi orang.

Mungkin contoh awal dari suatu sistem pengenalan wajah adalah Kohonen, yang menunjukkan bahwa jaringan saraf sederhana dapat melakukan pengenalan wajah dengan selaras dan normalisasi gambar wajah. Sistem yang bekerja pada jenis jaringan ini yaitu dengan menghitung gambaran wajah eigenvektor yang autokorelasi dengan matriks, eigenvektor ini sekarang dikenal sebagai "*eigenfaces*" Pada tahun berikutnya banyak peneliti mencoba skema pengenalan wajah didasarkan pada tepi, jarak antar-fitur, dan pendekatan jaringan saraf.

Pada perkembangannya, pengenalan wajah adalah sebuah sistem yang mengenali gambar wajah manusia yang biasanya digunakan dalam otomatisasi dan *security* sebuah industri.



Gambar II.6 Face Recognition

II.1.2.6 Biometrics

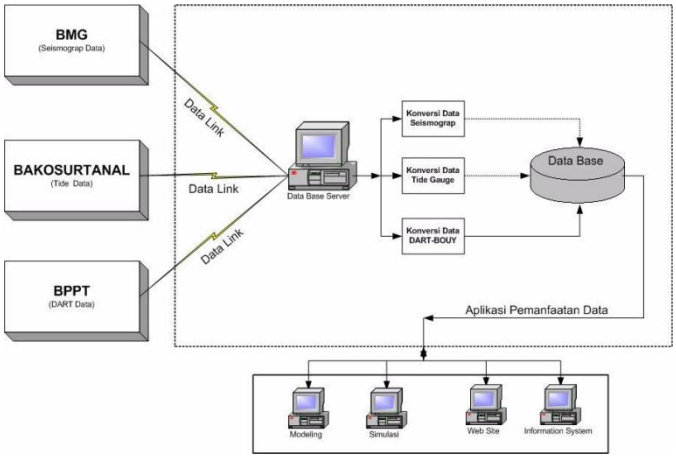
Biometrics di sektor teknologi tinggi mengacu pada kelas tertentu dalam teknologi identifikasi. Teknologi ini menggunakan karakter individu biologis yang unik dalam menentukan identitas seseorang. *Biometric* berguna untuk mengenali suatu pola mahluk hidup yang dihubungkan dengan parameter – parameter psikologi maupun tingkah laku. Ciri-ciri yang dianggap termasuk seperti sidik jari, pola retina dan iris, karakteristik wajah dan banyak lagi.



Gambar II.7 Biometrics

II.1.2.7 Image Database Retrieval

Image Database Retrieval adalah sistem komputer untuk *browsing*, mencari dan mengambil gambar dari *database* gambar digital. Sebagian besar metode tradisional dan umum dari pengambilan gambar menggunakan beberapa metode untuk menambahkan metadata seperti *captioning*, kata kunci, atau deskripsi gambar sehingga pengambilan yang dapat dilakukan melalui kata-kata penjelasan. Penjelasan gambar manual memakan waktu, tenaga dan mahal. Selain itu, peningkatan aplikasi web sosial dan web semantik telah menginspirasi pengembangan beberapa alat berbasis web gambar penjelasan.



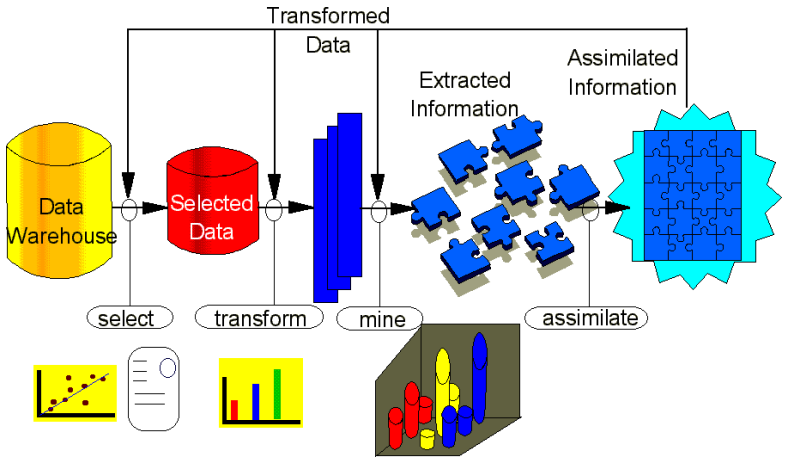
Gambar II.8 Image Database Retrieval

II.1.2.8 Data Mining

Data mining adalah proses penggalian pola dari data. *Data mining* menjadi alat yang semakin penting untuk mengubah data menjadi informasi. Hal ini umumnya digunakan dalam berbagai profil praktek, seperti pemasaran, pengawasan, deteksi penipuan dan penemuan ilmiah.

Data mining dapat digunakan untuk menemukan pola dalam data tetapi sering dilakukan hanya pada sampel data. Proses *mining* tidak akan efektif jika sampel bukan merupakan representasi yang baik dari tubuh utama dari data. *Data mining* tidak dapat menemukan pola-pola yang mungkin ada dalam tubuh lebih besar dari data jika pola-pola yang tidak hadir dalam sampel yang "*mined*". Ketidakmampuan untuk menemukan pola bisa menjadi penyebab untuk beberapa perselisihan antara pelanggan dan penyedia layanan. Oleh karena itu *data mining* tidak sangat mudah, namun mungkin berguna jika cukup representatif sampel data dikumpulkan. Penemuan pola tertentu dalam satu set data tertentu tidak berarti bahwa suatu pola ditemukan di tempat lain pada data yang lebih besar dari sampel

yang diambil. Sebuah bagian penting dari proses ini adalah verifikasi dan validasi pola pada data sampel lainnya.

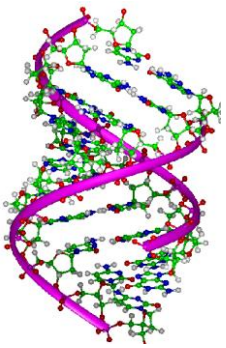


Gambar II.9 Data Mining

II.1.2.9 Bioinformatics

Bioinformatics adalah penggunaan matematika, statistik dan metode komputer untuk menganalisis biologi, biokimia, dan data biofisik. Karena *bioinformatics* adalah bidang, muda cepat berkembang, namun juga memiliki sejumlah definisi yang kredibel lainnya. Hal ini juga dapat didefinisikan sebagai ilmu pengetahuan dan teknologi pembelajaran, mengelola, dan pengolahan informasi biologis. *Bioinformatics* sering difokuskan pada perolehan data yang berorientasi biologis, mengorganisir informasi ini ke *database*, mengembangkan metode untuk mendapatkan informasi yang berguna dari *database* tersebut, dan merumuskan metode untuk mengintegrasikan data yang terkait dari sumber yang berbeda. Database komputer dan algoritma yang dikembangkan untuk mempercepat dan meningkatkan penelitian biologi.

Bioinformatics dapat membantu menjawab pertanyaan seperti apakah gen baru dianalisis adalah serupa dengan gen apapun sebelumnya dikenal, apakah urutan protein yang dapat menunjukkan bagaimana fungsi protein, dan apakah gen diaktifkan pada sel kanker berbeda dari yang diaktifkan dalam yang sehat sel.



Gambar II.10 Bioinformatics

II.1.2.10 Image Recognition Methods

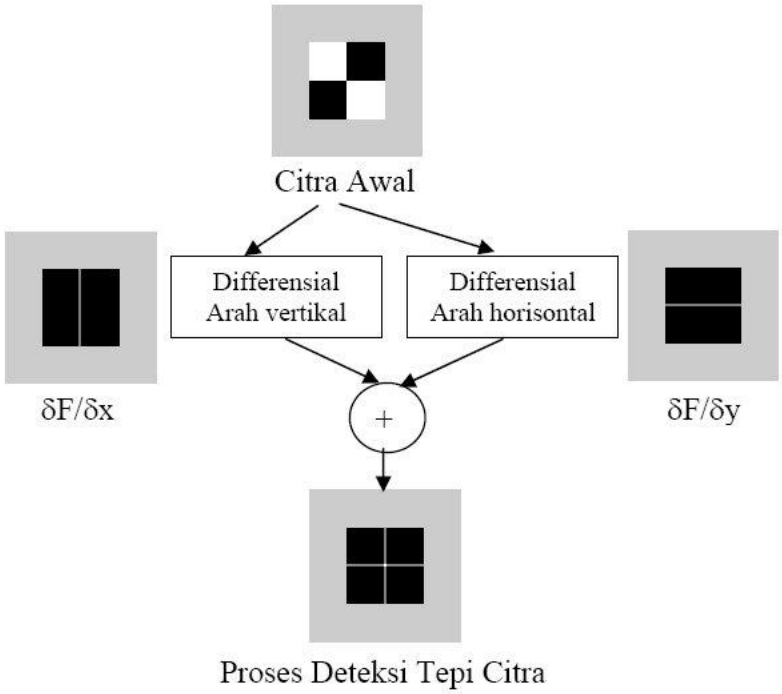
Pengenalan pola ini mengelompokkan data numerik dan simbolik (termasuk citra) secara otomatis oleh mesin (dalam hal ini komputer). Tujuan pengelompokan adalah untuk mengenali suatu objek di dalam citra.

Adapun metode yang digunakan untuk mengenali citra tersebut yaitu :

A. Edge Detection

Deteksi tepi *(edge detection)* pada suatu citra adalah suatu proses yang menghasilkan tepi-tepi dari obyek-obyek citra, Tepi-tepi ini akan menandai bagian detail citra.

Suatu titik (x,y) dikatakan sebagai tepi *(edge)* dari suatu citra bila titik tersebut mempunyai perbedaan yang tinggi dengan tetangganya. Gambar II.11 menggambarkan bagaimana tepi suatu gambar diperoleh.



Gambar II.11 Edge Detection

Macam-macam metode untuk proses deteksi tepi, antara lain:

1. Metode Robert

Metode pendektesian tepi dengan mencari perbedaan (*differensial*) pada arah horisontal dan perbedaan pada arah vertikal, dengan ditambahkan proses konversi biner setelah dilakukan perbedaan. Agar mendapatkan tepi-tepi yang lebih baik, maka konversi biner dilakukan dengan meratakan distribusi warna hitam dan putih atau dengan kata lain objek gambar yang akan digunakan untuk metode ini sebaiknya adalah gambar hitam putih. Kelebihan operator ini yaitu cepat dan mudah untuk menghitung pengukuran gradien 2-D spasial pada gambar. Pixel nilai di setiap titik di output mutlak mewakili besarnya estimasi gradien spasial citra input pada saat itu.

Operator terdiri dari sepasang kernel 2 × 2 konvolusi. Satu kernel yang lain hanya diputar oleh 90 °. Ini sangat mirip dengan operator Sobel.

+1	0
0	-1

Gx

0	+1
-1	0

Gy

Gambar II.12 Robert's Method

Kernel ini dirancang untuk merespon secara maksimal untuk tepi berjalan pada 45 ° ke jaringan pixel, satu kernel untuk masing-masing dua orientasi tegak lurus. Kernel dapat diterapkan secara terpisah dengan citra input, untuk menghasilkan pengukuran yang terpisah dari komponen gradien di setiap orientasi (menyebut Gx dan Gy). Ini kemudian dapat digabungkan bersama-sama untuk mencari besarnya nilai mutlak gradien pada setiap titik dan orientasi gradien itu. Besarnya gradien diberikan oleh:

$$|G| = \sqrt{Gx^2 + Gy^2} \quad \text{.....(1)}$$

walaupun biasanya, sebuah besaran perkiraan dihitung dengan menggunakan:

$$|G| = |Gx| + |Gy| \quad \text{.....(2)}$$

yang lebih cepat untuk menghitung.

Sudut orientasi tepi menimbulkan gradien spasial (relatif terhadap orientasi grid pixel) diberikan oleh:

$$\theta = \arctan(Gy/Gx) - 3\pi/4 \quad \text{.....(3)}$$

2. Metode Prewitt

Metode ini adalah pengembangan metode robert dengan menggunakan filter HPF yang diberi satu angka nol penyangga. Metode ini mengambil prinsip dari fungsi laplacian yang dikenal sebagai fungsi untuk membangkitkan HPF. Sehingga tepi-tepi yang dihasilkan lebih banyak dari metode Robert.

$S_x =$

-1	0	1
-1	0	1
-1	0	1

$S_y =$

1	1	1
0	0	0
-1	-1	-1

Gambar II.13 Prewitt Method

Filter HPF menciptakan sebuah gambar di mana tepi (perubahan tajam nilai gray level) yang akan ditampilkan. Hanya ukuran 3x3 filter dapat digunakan dengan filter ini. Filter ini menggunakan dua 3x3 template untuk menghitung nilai Prewitt gradien seperti yang ditunjukkan di bawah ini:

Template:

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$ sebagai X.....(4)

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$ sebagai Y.....(5)

Gunakan template untuk jendela filter 3x3.

$$\begin{bmatrix} a1 & a2 & a3 \\ a4 & a5 & a6 \\ a7 & a8 & a9 \end{bmatrix}$$ 3x3 filter jendela.....(6)

dimana a1 .. a9 adalah tingkat keabuan dari setiap piksel di jendela filter.

X = -1 * a1 + 1 * a3 - 1 * a4 + 1 * A6 - 1 * A7 + 1 * A9.....(7)

Y = 1 * a1 + 1 * a2 + 1 * a3 - 1 * A7 - 1 * A8 - 1 * A9.....(8)

Prewitt Gradient = sqrt (X * X + Y * Y).....(9)

3. Metode Sobel

Metode ini juga merupakan pengembangan metode robert. Kelebihan dari metode sobel ini adalah kemampuan untuk mengurangi *noise* sebelum melakukan perhitungan deteksi tepi sehingga tepi-tepi yang dihasilkan lebih banyak dibanding 2 metode sebelumnya. Operator terdiri dari sepasang kernel 3 × 3 konvolusi seperti yang ditunjukkan pada Gambar 1. Satu kernel yang lain hanya diputar oleh 90 °.

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

Gambar II.14 Sobel Method

Kernel ini dirancang untuk merespon secara maksimal untuk tepi berjalan secara vertikal dan horizontal relatif terhadap grid pixel, satu kernel untuk masing-masing dua orientasi tegak lurus. Kernel dapat diterapkan secara terpisah dengan citra input, untuk menghasilkan pengukuran yang terpisah dari komponen gradien di setiap orientasi (menyebut Gx dan Gy). Ini kemudian dapat digabungkan bersama-sama untuk mencari besarnya nilai mutlak gradien pada setiap titik dan orientasi gradien itu. Besarnya gradien diberikan oleh:

$$|G| = \sqrt{Gx^2 + Gy^2} \quad \text{.....(10)}$$

Biasanya, sebuah perkiraan besarnya dihitung dengan menggunakan:

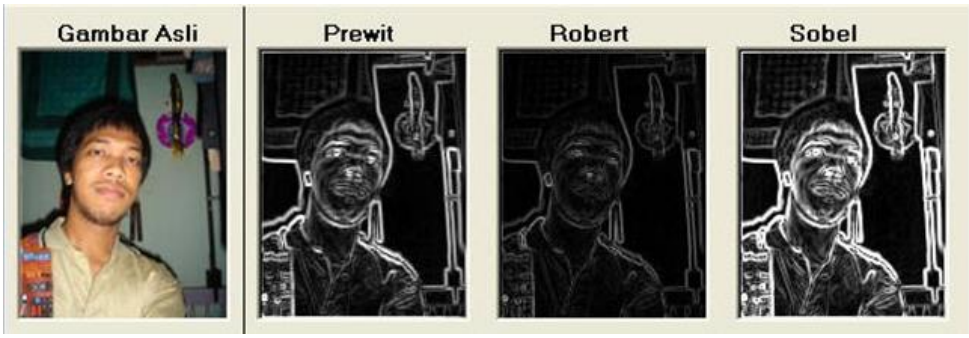
$$|G| = |Gx| + |Gy| \quad \text{.....(11)}$$

yang lebih cepat untuk menghitung.

Sudut orientasi tepi (relatif ke grid pixel) menimbulkan gradien spasial diberikan oleh:

$$\theta = \arctan(Gy/Gx) \quad \text{.....(12)}$$

Melihat dari perbandingan 3 metode yaitu Roberts, Prewitt, dan Sobel, maka output yang dihasilkan :



Gambar II.15 Output Prewitt, Sobel, dan Robert Method

II.1.3 Artificial Intelligence

Artificial Intelligence adalah bagian dari ilmu komputer yang mempelajari bagaimana membuat mesin (komputer) dapat melakukan pekerjaan seperti dan sebaik yang dilakukan oleh manusia bahkan bisa lebih baik daripada yang dilakukan manusia.

Menurut John McCarthy, 1956, AI :

Untuk mengetahui dan memodelkan proses – proses berpikir manusia dan mendesain mesin agar dapat menirukan perilaku manusia. Cerdas = memiliki pengetahuan + pengalaman, penalaran (bagaimana membuat keputusan & mengambil tindakan), moral yang baik

Manusia cerdas (pandai) dapat menyelesaikan permasalahan karena manusia mempunyai pengetahuan & pengalaman. Agar mesin bisa cerdas (bertindak seperti & sebaik manusia) maka harus diberi bekal pengetahuan & mempunyai kemampuan untuk menalar. 2 bagian utama yg dibutuhkan untuk aplikasi kecerdasan buatan :

a. Basis pengetahuan (*knowledge base*): berisi fakta-fakta, teori, pemikiran & hubungan antara satu dengan lainnya.

b. Motor inferensi (*inference engine*) : kemampuan menarik kesimpulan berdasarkan pengalaman motor inferensi, basis pengetahuan, input, masalah, pertanyaan, dan lainnya serta output, jawaban, dan solusi.

II.1.4 Jaringan Syaraf Tiruan (JST)

Jaringan Syaraf Tiruan adalah paradigma pemrosesan suatu informasi yang terinspirasi oleh sistim sel syaraf biologi, sama seperti otak yang memproses

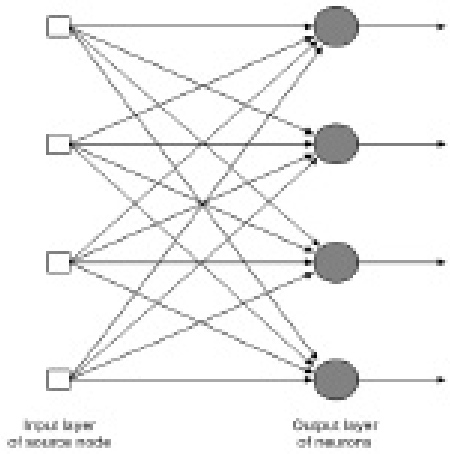
suatu informasi. Elemen mendasar dari paradigma tersebut adalah struktur yang baru dari sistem pemrosesan informasi. Jaringan Syaraf Tiruan, seperti manusia, belajar dari suatu contoh. Jaringan Syaraf Tiruan dibentuk untuk memecahkan suatu masalah tertentu seperti pengenalan pola atau klasifikasi karena proses pembelajaran.

Jaringan syaraf tiruan berkembang secara pesat pada beberapa tahun terakhir. Jaringan syaraf tiruan telah dikembangkan sebelum adanya suatu komputer konvensional yang canggih dan terus berkembang walaupun pernah mengalami masa vakum selama beberapa tahun.

Adapun arsitektur jaringan syaraf tiruan yang sering digunakan yaitu :

II.1.4.1 Jaringan Syaraf Perceptron

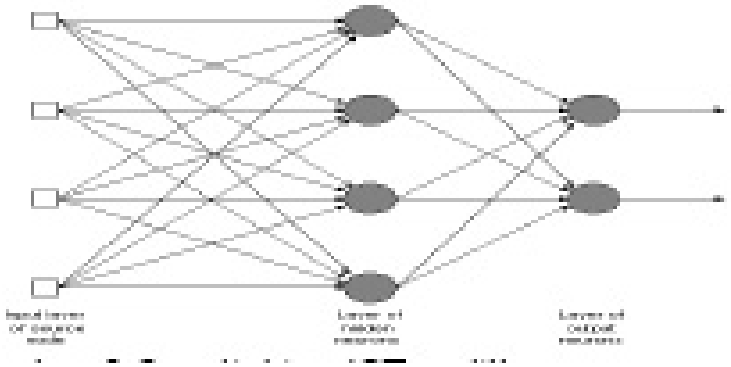
Jaringan jenis ini hanya terdiri dari layer input dan layer output saja. Input yang masuk melalui *vector* input akan langsung diproses dan kemudian ditentukan target *output* dari hasil pengolahan.



Gambar II.16 Perceptron

II.1.4.2 Jaringan Syaraf Multilayer

Jaringan jenis ini mempunyai lapisan yang terletak di antara input layer dan output layer yang lebih dikenal sebagai hidden layer, jumlah lapisan hidden layer ini minimal satu buah dan maksimalnya tidak ada batasan.



Gambar II.17 Multilayer

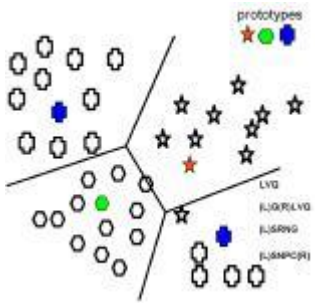
Umumnya JST jenis ini lebih mampu untuk menyelesaikan masalah yang lebih rumit dibandingkan dengan JST perceptron terdiri dari empat *node input*, empat *node hidden*, dan dua *node output*.

A. Learning Vector Quantization (LVQ)

LVQ merupakan jaringan syaraf dengan tipe arsitektur jaringan lapis-tunggal umpan-maju (*Single Layer Feedforward*) yang terdiri atas unit masukan (Xn) dan unit keluaran (Ym) .LVQ adalah versi pembelajaran terawasi dari vektor kuantisasi yang dapat digunakan ketika kita telah diberi label data *input*. Teknik belajar menggunakan kelas informasi untuk memposisikan ulang vektor Voronoi, sehingga untuk meningkatkan kualitas classifier keputusan daerah. Ini adalah proses dua tahap - sebuah SOM diikuti oleh LVQ: Hal ini terutama berguna untuk masalah klasifikasi pola. Langkah pertama adalah fitur seleksi - identifikasi tanpa pengawasan dari serangkaian fitur yang cukup kecil di mana isi informasi penting

dari data input terkonsentrasi. Langkah kedua adalah klasifikasi mana domain fitur yang ditugaskan untuk kelas masing-masing. Pendekatan LVQ dasar cukup intuitif. Hal ini didasarkan pada SOM terlatih standar dengan input vektor (x) dan bobot / Voronoi vektor (WJ).

Faktor baru yang menunjukkan input data telah terkait informasi kelas. Ini memungkinkan kita untuk menggunakan label dikenal klasifikasi masukan untuk menemukan yang terbaik klasifikasi label untuk setiap WJ, yaitu untuk setiap sel Voronoi.



Gambar II.18 Learning Vector Quantization

II.1.4.3 Jaringan Syaraf Recurrent

Jika pada jenis perceptron dan multilayer terdapat jenis arsitektur yang simetris dimana tiap layer-nya terhubung ke layer yang lebih atas, maka pada arsitektur jenis ini akan dijumpai suatu node yang mempunyai link ke dirinya sendiri ataupun link ke node yang lain dalam satu layer bahkan ada juga yang dari input layer langsung menuju ke output layer tanpa melalui output layer. JST jenis ini lebih mirip pada jaringan syaraf yang dimiliki manusia dimana struktur jaringannya tidak tersruktur. Umumnya JST jenis ini lebih sulit untuk diimplementasikan dari pada jenis arsitektur yang lainnya.

II.1.5 Search Engine

Search engine disebut juga dengan mesin pencari, dimana sistem yang ada pada sistem tersebut diolah melalui satu atau sekelompok komputer yang berfungsi untuk melakukan pencarian data. Data yang ada pada mesin ini dikumpulkan oleh mereka melalui suatu metoda tertentu, dan diambil dari seluruh server yang dapat mereka akses. Jika dilakukan pencarian melalui *search engine* ini, maka pencarian yang dilakukan sebenarnya adalah pada *database* yang telah terkumpul di dalam mesin tersebut.

Berdasarkan cara mengumpulkan data halaman-halaman web, mesin pencari dapat di kelompokkan menjadi 4 kategori yaitu :

II.1.5.1 Human Organized Search Engine

Mesin pencari yang di kelola sepenuhnya oleh tangan manusia. Mesin pencari ini menggunakan metode dengan memilah-milih informasi yang relevan dan dikelompokan sedemikian rupa sehingga lebih bermakna dan bermanfaaat bagi penggunannya. Situs ini dalam prakteknya memperkerjakan para pakar dalam bidang-bidang tertentu, kemudian para pakar tersebut dapat mengkelompokan situs-situs tertentu sesuai dengan bidangnnya atau kategori situs itu sendiri.

Tabel II.1 Human Organized Search Engine

Search Engine	Tahun Peluncuran	Alamat Situs
Yahoo!	1994	www.yahoo.com
LooKSmart	1996	www.looksmart.com
About	1987	www.about.com
Open Directory	1998	Dmoz.org

II.1.5.2 Computer Created Search Engine

Search engine kategori ini banyak memiliki kelebihan karena banyak menyajikan inforrmasi walaupun kadang-kadang ada beberapa informasi yang tidak relevan tidak seperti yang kita inginkan. *Search engine* ini telah menggunakan *software* laba-laba atau *spider software* yang berfungsi menyusup pada situs-situs tertentu, kemudian mengumpulkan data serta mengelompokan dengan sedikit bantuan tangan manusia.

Tabel II.2 Computer Created Search Engine

Search Engine	Tahun Peluncuran	Alamat Situs
WebCrawler	1994	www.webcrawler.com
Excite	1995	www.excite.com
Inktomi	1996	www.inktomi.com
Northen Light	1997	www.nothernlight.com
Fast Search	1999	www.altheweb.com

II.1.5.3 Hybrid Seacrh Engine

Merupakan gabungan antara tangan manusia dengan komputer, sehingga menghasilkan hasil pencarian yang relative akurat. Peran manusia dalam hal ini adalah sebagai penelaah dalam proses pengkoleksian *database* halaman web. Sebenarnya tipe ini lah yanag paling mudah pembuatannya menurut saya karena dapat di desain sesuai dengan keinginan kita.

Tabel II.3 Hybrid Search Engine

Search Engine	Tahun Peluncuran	Alamat Situs
Lycos	1995	www.lycos.com
Alta Vista	1995	www.altavista.com
HotBot	1996	www.hotbot.com
Snap	1997	www.snap.com
Direct Hit	1998	www.directthit.com
Google	1998	www.google.com

Go	1999	go.com
Go To	1997	www.goto.com

II.1.5.4 MetaCrawler/Metasearch

Merupakan perantara dari mesin pencari yang sebenarnya. Mesin ini hanya akan mengirimkan permintaan pencarian ke berbagai mesin pencari serta menampilkan hasilnya satu di layer browser sehingga akan menampilkan banyak sekali hasil dari berbagai mesin pencari yang ada.

Tabel II.4 MetaCrawler

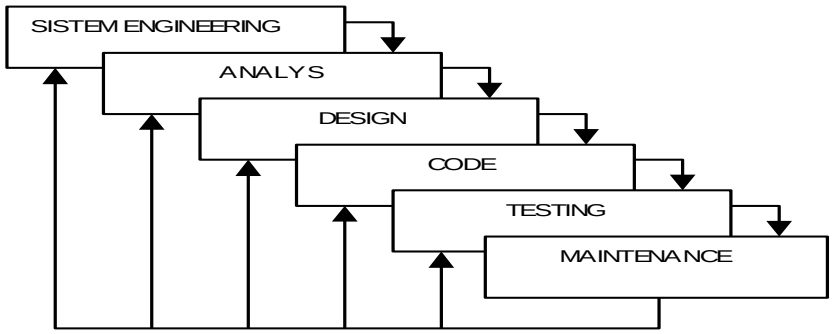
Search Engine	Tahun Peluncuran	Alamat Situs
SavvySearch	1995	savvysearch.com
Dogpile	Tidak teridentifikasi	www.dogpile.com
The Big Hub	Tidak teridentifikasi	www.infind.com
C4 Total Search	Tidak teridentifikasi	www.snap.com

II.1.6 Metode Pengembangan Perangkat Lunak

Metode pengembangan perangkat lunak memiliki beberapa jenis model proses dalam membangun suatu software. Adapun macamnya yaitu :

II.1.6.1 Linear SequentialModel/ Waterfall Model

Model ini adalah model klasik yang bersifat sistematis, berurutan dalam membangun *software*. Berikut ini gambaran dari *waterfall model*. Fase-fase dalam Waterfall Model menurut referensi Sommerville:



Gambar II.19 Waterfall Model

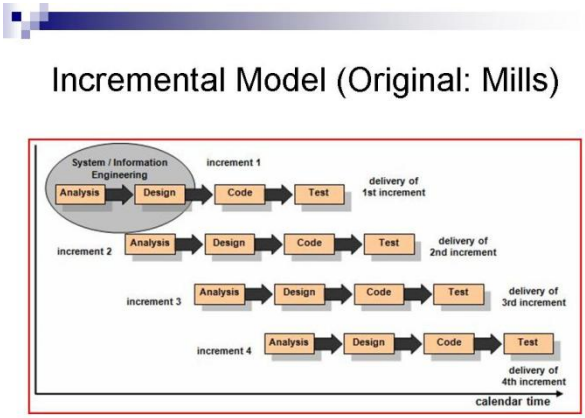
1. *Requirements analysis and definition*: Mengumpulkan kebutuhan secara lengkap kemudian kemudian dianalisis dan didefinisikan kebutuhan yang harus dipenuhi oleh program yang akan dibangun. Fase ini harus dikerjakan secara lengkap untuk bisa menghasilkan desain yang lengkap.
2. *Sistem and software design*: Desain dikerjakan setelah kebutuhan selesai dikumpulkan secara lengkap.
3. *Implementation and unit testing*: desain program diterjemahkan ke dalam kode-kode dengan menggunakan bahasa pemrograman yang sudah ditentukan. Program yang dibangun langsung diuji baik secara unit.
4. *Integration and sistem testing*: Penyatuan unit-unit program kemudian diuji secara keseluruhan (sistem testing).
5. *Operation and maintenance*: mengoperasikan program dilingkungannya dan melakukan pemeliharaan, seperti penyesuaian atau perubahan karena adaptasi dengan situasi sebenarnya. Kekurangan yang utama dari model ini adalah kesulitan dalam mengakomodasi perubahan setelah proses dijalani. Fase sebelumnya harus lengkap dan selesai sebelum mengerjakan fase berikutnya.

II.1.6.2 Evolutionary Software Process Models

Bersifat iteratif/ mengandung perulangan. Hasil proses berupa produk yang makin lama makin lengkap sampai versi terlengkap dihasilkan sebagai produk akhir dari proses.

Dua model dalam *evolutionary software process* model adalah:

A. Incremental Model (Original: Mills)



Gambar II.20 Incremental Model

1. Kombinasikan element-element dari waterfall dengan sifat iterasi/perulangan.
2. Element-element dalam waterfall dikerjakan dengan hasil berupa produk dengan spesifikasi tertentu, kemudian proses dimulai dari fase pertama hingga akhir dan menghasilkan produk dengan spesifikasi yang lebih lengkap dari yang sebelumnya. Demikian seterusnya hingga semua spesifikasi memenuhi kebutuhan yang ditetapkan oleh pengguna.
3. Produk hasil increment pertama biasanya produk inti (core product), yaitu produk yang memenuhi kebutuhan dasar. Produk tersebut digunakan oleh pengguna atau menjalani review/pengecekan detil. Hasil review tersebut menjadi

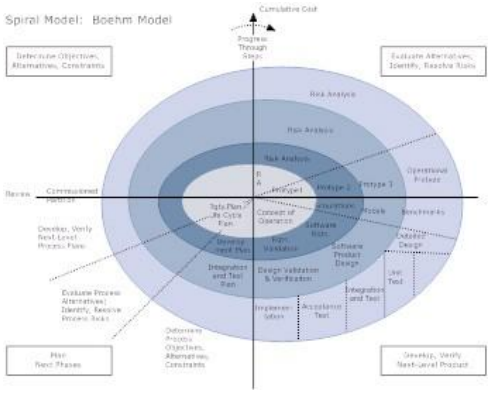
bekal untuk pembangunan pada increment berikutnya. Hal ini terus dikerjakan sampai produk yang komplit dihasilkan.

4. Model ini cocok jika jumlah anggota tim pengembang/pembangun PL tidak cukup.

5. Mampu mengakomodasi perubahan secara fleksibel.

6. Produk yang dihasilkan pada increment pertama bukanlah prototype, tapi produk yang sudah bisa berfungsi dengan spesifikasi dasar.

B. Spiral Model (Original: Boehm)



Gambar II.21 Spiral Model

Proses digambarkan sebagai spiral. Setiap loop mewakili satu fase dari *softwareprocess*. Loop paling dalam berfokus pada kelayakan dari sistem, loop selanjutnya tentang definisi dari kebutuhan, loop berikutnya berkaitan dengan desain sistem dan seterusnya. Setiap Loop dibagi menjadi beberapa sektor :

1. *Objective settings* (Menentukan Tujuan)

Menentukan tujuan dari fase yang ditentukan. Batasan-batasan pada proses dan produk sudah diketahui. Perencanaan sudah disiapkan. Resiko dari proyek

sudah diketahui. Alternatif strategi sudah disiapkan berdasarkan resiko-resiko yang diketahui, dan sudah direncanakan.

2. *Risk assessment and reduction* (Penanganan dan pengurangan resiko)

Setiap resiko dianalisis secara detil pada sektor ini. Langkah-langkah penanganan dilakukan, misalnya membuat *prototype* untuk mengetahui ketidakcocokan kebutuhan.

3. *Development and Validation* (Pembangunan dan pengujian)

Setelah evaluasi resiko, maka model pengembangan sistem dipilih. Misalnya jika resiko *user interface* dominan, maka membuat *prototype user interface*. Jika bagian keamanan yang bermasalah, maka menggunakan model formal dengan perhitungan matematis, dan jika masalahnya adalah integrasi sistem model *waterfall* lebih cocok.

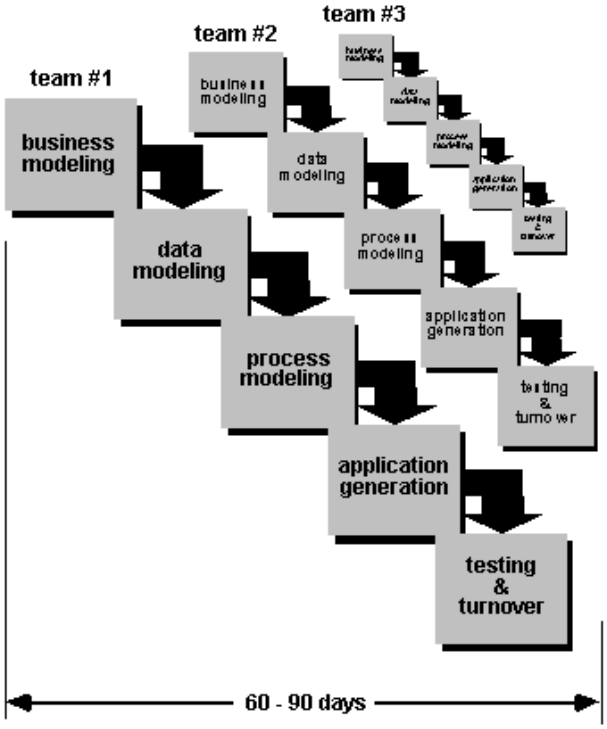
4. *Planning* (Perencanaan)

Proyek dievaluasi atau ditinjau-ulang dan diputuskan untuk terus ke fase loop selanjutnya atau tidak. Jika melanjutkan ke fase berikutnya rencana untuk loop selanjutnya.

C. RAD (Rapid Application Development)

RAD adalah model proses pembangunan PL yang incremental. RAD menekankan pada siklus pembangunan yang pendek/singkat. RAD mengadopsi model waterfall dan pembangunan dalam waktu singkat dicapai dengan menerapkan *component based construction*. Waktu yang singkat adalah batasan yang penting untuk model ini.

Jika kebutuhan lengkap dan jelas maka waktu yang dibutuhkan untuk menyelesaikan secara komplit software yang dibuat adalah misalnya 60 sampai 90 hari.



Gambar II.22 RAD

Fase-fase di atas menggambarkan proses dalam model RAD. Sistem dibagi-bagi menjadi beberapa modul dan dikerjakan dalam waktu yang hampir bersamaan dalam batasan waktu yang sudah ditentukan.

1. *Business modelling* : menjawab pertanyaan-pertanyaan: informasi apa yang mengendalikan proses bisnis, informasi apa yang dihasilkan, siapa yang menghasilkan informasi, kemana informasi itu diberikan, siapa yang mengolah informasi.

2. *Data modelling*: aliran informasi yang sudah didefinisikan, disusun menjadi sekumpulan objek data. Ditentukan karakteristik/atribut dan hubungan antar objek-objek tersebut,
3. P*rocess Modelling* : objek data yang sudah didefinisikan diubah menjadi aliran informasi yang diperlukan untukmenjalankan fungsi-fungsi bisnis.
4. *Application Generation*: RAD menggunakan component program yang sudah ada atau membuat component yang bisa digunakan lagi, selama diperlukan.
5. *Testing and Turnover* : karena menggunakan component yang sudah ada, maka kebanyakan component sudah melalui uji atau testing. Namun component baru dan interface harus tetap diuji.

D. Prototyping Model

Kadang-kadang klien hanya memberikan beberapa kebutuhan umum *software* tanpa detil *input*, proses atau detil *output*. Di lain waktu mungkin dimana tim pembangun (*developer*) tidak yakin terhadap efisiensi dari algoritma yang digunakan, tingkat adaptasi terhadap sistem operasi atau rancangan form *user interface*. Ketika situasi seperti ini terjadi model *prototyping* sangat membantu proses pembangunan *software*.

Proses pada model prototyping yang digambarkan pada gambar II.23, bisa dijelaskan sebagai berikut:

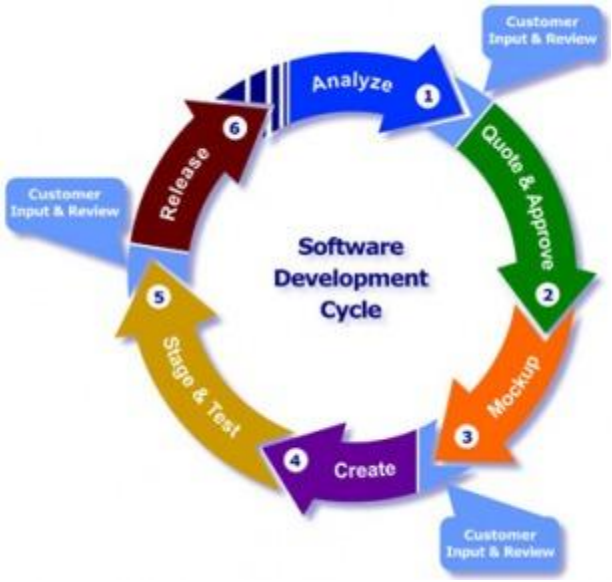
1. Pengumpulan kebutuhan: *developer* dan klien bertemu dan menentukan tujuan umum, kebutuhan yang diketahui dan gambaran bagian-bagian

yang akan dibutuhkan berikutnya. Detail kebutuhan mungkin tidak dibicarakan disini, pada awal pengumpulan kebutuhan.

2. Perancangan : perancangan dilakukan cepat dan rancangan mewakili semua aspek *software* yang diketahui, dan rancangan ini menjadi dasar pembuatan *prototype*.
3. Evaluasi *prototype*: klien mengevaluasi *prototype* yang dibuat dan digunakan untuk memperjelas kebutuhan *software*. Perulangan ketiga proses ini terus berlangsung hingga semua kebutuhan terpenuhi. *Prototype-prototype* dibuat untuk memuaskan kebutuhan klien dan untuk memahami kebutuhan klien lebih baik. *Prototype* yang dibuat dapat dimanfaatkan kembali untuk membangun *software* lebih cepat, namun tidak semua *prototype* bisa dimanfaatkan. Sekalipun *prototype* memudahkan komunikasi antar *developer* dan klien, membuat klien mendapat gambaran awal dari *prototype* , membantu mendapatkan kebutuhan detil lebih baik namun demikian *prototype* juga menimbulkan masalah:
    a. Dalam membuat prototype banyak hal yang diabaikan seperti efisiensi, kualitas, kemudahan dipelihara/dikembangkan, dan kecocokan dengan lingkungan yang sebenarnya. Jika klien merasa cocok dengan *prototype* yang disajikan dan berkeras terhadap produk tersebut, maka developer harus kerja keras untuk mewujudkan produk tersebut menjadi lebih baik, sesuai kualitas yang seharusnya.

b. *Developer* biasanya melakukan kompromi dalam beberapa hal karena harus membuat *prototype* dalam waktu singkat. Mungkin sistem operasi yang tidak sesuai, bahasa pemrograman yang berbeda, atau algoritma yang lebih sederhana.

Agar model ini bisa berjalan dengan baik, perlu disepakati bersama oleh klien dan *developer* bahwa *prototype* yang dibangun merupakan alat untuk mendefinisikan kebutuhan *software*.



Gambar II.23 Prototyping Model

E. Extreme Programming (XP) Model

Model proses ini diciptakan dan dikembangkan oleh Kent Beck. Model ini adalah model proses yang terbaru dalam dunia rekayasa perangkat lunak dan mencoba menjawab kesulitan dalam pengembangan *software* yang rumit dan sulit dalam implementasi.

Menurut Kent Beck XP adalah : "*A lightweight, efficient, low-risk, flexible, predictable, scientific and fun way to develop software*".

Suatu model yang menekankan pada:

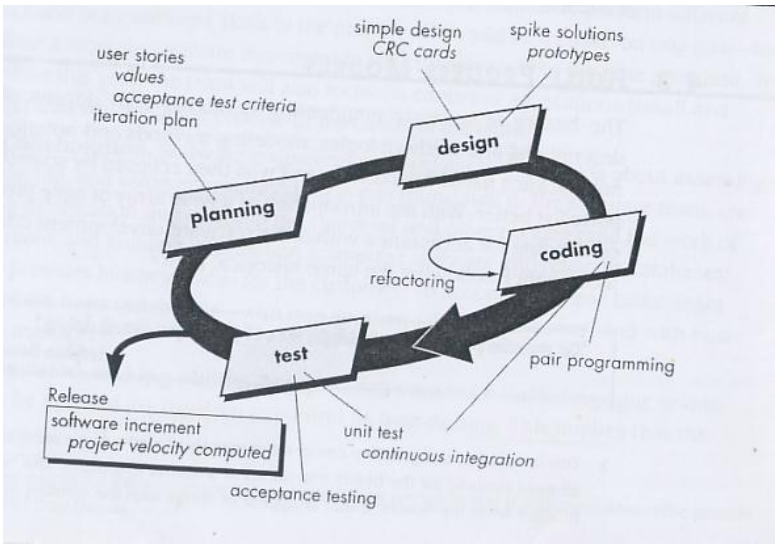
1. Keterlibatan user secara langsung
2. Pengujian
3. *pay-as-you-go design*

Adapun empat nilai penting dari XP

1. *Communication* / Komunikasi : komunikasi antara developer dan klien sering menjadi masalah. Karena itu komunikasi dalam XP dibangun dengan melakukan pemrograman berpasangan (pair programming). Developer didampingi oleh pihak klien dalam melakukan coding dan unit testing sehingga klien bisa terlibat langsung dalam pemrograman sambil berkomunikasi dengan developer. Selain itu perkiraan beban tugas juga diperhitungkan.
2. *Simplicity* / sederhana: Menekankan pada kesederhanaan dalam pengkodean:

   "*What is the simplest thing that could possibly work?*". Lebih baik melakukan hal yang sederhana dan mengembangkannya besok jika diperlukan. Komunikasi yang lebih banyak mempermudah, dan rancangan yang sederhana mengurangi penjelasan.
3. *Feedback* / Masukan / Tanggapan: Setiap *feedback* ditanggapi dengan melakukan tes, unit test atau sistem integration dan jangan menunda karena biaya akan membengkak (uang, tenaga, waktu).

4. *Courage* / Berani: Banyak ide baru dan berani mencobanya, berani mengerjakan kembali dan setiap kali kesalahan ditemukan, langsung diperbaiki.



Gambar II.24 XP Model

F. Object Oriented Programing(OOP)

Object-Oriented Programming (OOP) adalah sebuah pendekatan untuk pengembangan / *development* suatu *software* dimana dalam struktur *software* tersebut didasarkan kepada interaksi *object* dalam penyelesaian suatu proses/tugas. Interaksi tersebut mengambil form dari pesan-pesan dan mengirimkannya kembali antar *object* tersebut. *Object* akan merespon pesan tersebut menjadi sebuah tindakan / *action* atau metode. Jika kita mencoba melihat bagaimana tugas disekitar kita diselesaikan, kita akan mengetahui bahwa kita berinteraksi dalam sebuah *object-oriented world*. Jika akan bepergian kita pasti berinteraksi dengan *object* mobil. Sebagai sebuah *object*, mobil berisi *object-object* lain yang berinteraksi untuk melakukan tugasnya membawa kita.

Object-oriented programs terdiri dari *objects* yang berinteraksi satu sama lainnya untuk menyelesaikan sebuah tugas. Seperti dunia nyata, *users* dari *software* programs dilibatkan dari logika proses untuk menyelesaikan tugas. Contoh, ketika kamu mencetak sebuah halaman di *word processor*, kamu berarti melakukan inisialisasi tindakan dengan mengklik tombol printer. Kemudian kamu hanya menunggu respon apakah *job* tersebut sukses atau gagal, sedangkan proses terjadi internal tanpa kita ketahui. Tentunya setelah kamu menekan tombol printer, maka secara simultan *object* tombol tersebut berinteraksi dengan *object* printer untuk menyelesaikan *job* tersebut.

Adapun karakteristik yang dimiliki OOP yaitu :

a. Objects

Object adalah sebuah *structure* yang menggabungkan data dan prosedur untuk bekerja bersama-sama.

b. Abstraction

Ketika kamu berinteraksi dengan *object-object* di dunia ini, kamu sering hanya konsentrasi dengan sebuah bagian dari propertiesnya. Tanpa kemampuan untuk mensarikan/*abstract* atau menyaring untuk dibuang properties *object* yang asing / tidak ada hubungannya, maka kamu akan menemukan kesulitan untuk memproses informasi yang kebanyakan membombarding kamu.

Ketika membangun *objects* dalam aplikasi OOP, adalah penting untuk menggabungkan konsep *abstraction* ini. Jika kamu membangun aplikasi *shipping*, kamu harus membangun *object* produk dengan atribut seperti ukuran dan berat. Warna adalah contoh informasi yang tidak ada hubungannya dan harus dibuang.

Tetapi ketika kamu membangun *order-entry application*, warna menjadi penting dan harus termasuk atribut *object* produk.

c. Encapsulation

Ciri penting lainnya dari OOP adalah *encapsulation*. *Encapsulation* adalah sebuah proses dimana tidak ada akses langsung ke data yang diberikan, bahkan *hidden*. Jika kamu ingin mendapat data, kamu harus berinteraksi dengan *object* yang bertanggung jawab atas data tersebut.

d. Polymorphism

Polymorphisms adalah kemampuan 2 buah *object* yang berbeda untuk merespon pesan permintaan yang sama dalam suatu cara yang unik. Contohnya, kamu dapat mengirim pesan *print* ke *object* printer yang akan mencetak pada printer, dan kamu juga dapat mengirim pesan yang sama ke *object screen* yang akan menuliskan pada *screen* monitor. Dalam OOP, kamu menerapkan tipe *polymorphism* melalui proses yang disebut *overloading*. Kamu dapat mengimplementasikan metode yang berbeda pada sebuah *object* yang mempunyai nama yang sama.

e. Inheritance

Penggunaan *inheritance* dalam OOP untuk mengklasifikasikan *objects* dalam program sesuai karakteristik umum dan fungsinya. Hal ini akan membuat pekerjaan bersama *object* lebih mudah dan lebih intuitif. Hal ini juga membuat programming lebih mudah karena memungkinkan kita untuk mengkombinasikan

karakteristik umum kedalam *object parent* dan mewariskan karakteristik ini ke *child object*.

G. UML (User Modeling Language)

a. Pengertian UML

UML *(Unified Modeling Language)* adalah sebuah bahasa untuk menetukan, visualisasi, kontruksi, dan mendokumentasikan *artifact* (bagian dari informasi yang digunakan atau dihasilkan dalam suatu proses pembuatan perangkat lunak. Artifact dapat berupa model, deskripsi atau perangkat lunak) dari sistem perangkat lunak, seperti pada pemodelan bisnis dan sistem non perangkat lunak lainnya.

UML merupakan suatu kumpulan teknik terbaik yang telah terbukti sukses dalam memodelkan sistem yang besar dan kompleks. UML tidak hanya digunakan dalam proses pemodelan perangkat lunak, namun hampir dalam semua bidang yang membutuhkan pemodelan.

b. Bagian-bagian UML

Bagian-bagian utama dari UML adalah view, diagram, model element, dan *general mechanism*.

1. View

View digunakan untuk melihat sistem yang dimodelkan dari beberapa aspek yang berbeda. *View* bukan melihat grafik, tapi merupakan suatu abstraksi yang berisi sejumlah diagram. Beberapa jenis *view* dalam UML antara lain: *use case view, logical view, component view, concurrency view, dan deployment view.*

a) *Use case view*

Mendeskripsikan fungsionalitas sistem yang seharusnya dilakukan sesuai yang diinginkan *external actors*. *Actor* yang berinteraksi dengan sistem dapat berupa user atau sistem lainnya. *View* ini digambarkan dalam *use case diagrams* dan kadang-kadang dengan *activity diagrams*. *View* ini digunakan terutama untuk pelanggan, perancang (*designer*), pengembang (*developer*), dan penguji sistem (*tester*).

b) *Logical view*

Mendeskripsikan bagaimana fungsionalitas dari sistem, struktur statis (*class, object,dan relationship*) dan kolaborasi dinamis yang terjadi ketika *object* mengirim pesan ke *object* lain dalam suatu fungsi tertentu. *View* ini digambarkan dalam class diagrams untuk struktur statis dan dalam *state, sequence, collaboration, dan activity diagram* untuk model dinamisnya. View ini digunakan untuk perancang (*designer*) dan pengembang (*developer*).

c) *Component view*

Mendeskripsikan implementasi dan ketergantungan modul. Komponen yang merupakan tipe lainnya dari code module diperlihatkan dengan struktur dan ketergantungannya juga alokasi sumber daya komponen dan informasi administrative lainnya. View ini digambarkan dalam *component view* dan digunakan untuk pengembang (*developer*).

d) Concurrency view

Membagi sistem ke dalam proses dan prosesor. *View* ini digambarkan dalam diagram dinamis *(state, sequence, collaboration, dan activity diagrams)* dan diagram implementasi *(component dan deployment diagrams)* serta digunakan untuk pengembang (*developer*), pengintegrasi (*integrator*), dan penguji (*tester*).

e) Deployment view

Mendeskripsikan fisik dari sistem seperti komputer dan perangkat (*nodes*) dan bagaimana hubungannya dengan lainnya. *View* ini digambarkan dalam *deployment diagrams* dan digunakan untuk pengembang (*developer*), pengintegrasi (*integrator*), dan penguji (*tester*).

2. Diagram

Diagram berbentuk grafik yang menunjukkan simbol elemen model yang disusun untuk mengilustrasikan bagian atau aspek tertentu dari sistem. Sebuah diagram merupakan bagian dari suatu *view* tertentu dan ketika digambarkan biasanya dialokasikan untuk *view* tertentu. Adapun jenis diagram antara lain :

a) Use Case Diagram

Use case adalah abstraksi dari interaksi antara sistem dan actor. *Use case* bekerja dengan cara mendeskripsikan tipe interaksi antara *user* sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. *Use case* merupakan konstruksi untuk mendeskripsikan bagaimana sistem akan terlihat di mata *user*. Sedangkan *use case* diagram

memfasilitasi komunikasi diantara analis dan pengguna serta antara analis dan *client*.

b) Class Diagram

Class adalah dekripsi kelompok obyek-obyek dengan *property*, perilaku (operasi) dan relasi yang sama. Sehingga dengan adanya *class* diagram dapat memberikan pandangan global atas sebuah sistem. Hal tersebut tercermin dari class- class yang ada dan relasinya satu dengan yang lainnya. Sebuah sistem biasanya mempunyai beberapa *class* diagram. *Class* diagram sangat membantu dalam visualisasi struktur kelas dari suatu sistem.

c) Component Diagram

Component software merupakan bagian fisik dari sebuah sistem, karena menetap di komputer tidak berada di benak para analis. Komponen merupakan implementasi *software* dari sebuah atau lebih *class*. Komponen dapat berupa *source code*, komponen biner, atau *executable component*. Sebuah komponen berisi informasi tentang *logic class* atau *class* yang diimplementasikan sehingga membuat pemetaan dari *logical view* ke *component view*. Sehingga component diagram merepresentasikan dunia nyata yaitu *component software* yang mengandung *component, interface dan relationship*.

d) Deployment Diagram

Menggambarkan tata letak sebuah sistem secara fisik, menampakkan bagian-bagian *software* yang berjalan pada bagian-bagian *hardware*,

menunjukkan hubungan komputer dengan perangkat (*nodes*) satu sama lain dan jenis hubungannya. Di dalam *nodes*, *executeable component* dan *object* yang dialokasikan untuk memperlihatkan unit perangkat lunak yang dieksekusi oleh *node* tertentu dan ketergantungan komponen.

e) State Diagram

Menggambarkan semua *state* (kondisi) yang dimiliki oleh suatu *object* dari suatu *class* dan keadaan yang menyebabkan *state* berubah. Kejadian dapat berupa *object* lain yang mengirim pesan. *State class* tidak digambarkan untuk semua *class*, hanya yang mempunyai sejumlah *state* yang terdefinisi dengan baik dan kondisi *class* berubah oleh *state* yang berbeda.

f) Sequence Diagram

Sequence diagram digunakan untuk menggambarkan perilaku pada sebuah *scenario*. Kegunaannya untuk menunjukkan rangkaian pesan yang dikirim antara *object* juga interaksi antara *object*, sesuatu yang terjadi pada titik tertentu dalam eksekusi sistem.

g) Collaboration Diagram

Menggambarkan kolaborasi dinamis seperti *sequence* diagrams. Dalam menunjukkan pertukaran pesan, *collaboration* diagrams menggambarkan *object* dan hubungannya (mengacu ke konteks). Jika penekannya pada waktu atau urutan gunakan *sequence* diagrams, tapi jika penekanannya pada konteks gunakan *collaboration* diagram.

h) Activity Diagram

Menggambarkan rangkaian aliran dari aktivitas, digunakan untuk mendeskripsikan aktifitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktifitas lainnya seperti *use case* atau interaksi.

II.1.7 Pembangun Perangkat Lunak

II.1.7.1 Sejarah Java

Bahasa pemrograman Java pertama lahir dari *The Green Project*, yang berjalan selama 18 bulan, dari awal tahun 1991 hingga musim panas 1992. Proyek tersebut belum menggunakan versi yang dinamakan Oak. Proyek ini dimotori oleh Patrick Naughton, Mike Sheridan, James Gosling dan Bill Joy, beserta sembilan pemrogram lainnya dari Sun Microsistems. Salah satu hasil proyek ini adalah maskot Duke yang dibuat oleh Joe Palrang.

Pertemuan proyek berlangsung di sebuah gedung perkantoran Sand Hill Road di Menlo Park. Sekitar musim panas 1992 proyek ini ditutup dengan menghasilkan sebuah program Java Oak pertama, yang ditujukan sebagai pengendali sebuah peralatan dengan teknologi layar sentuh (*touch screen*), seperti pada PDA sekarang ini. Teknologi baru ini dinamai "*7" (Star Seven).

Setelah era Star Seven selesai, sebuah anak perusahaan TV kabel tertarik ditambah beberapa orang dari proyek *The Green Project*. Mereka memusatkan kegiatannya pada sebuah ruangan kantor di 100 Hamilton Avenue, Palo Alto.

Perusahaan baru ini bertambah maju, jumlah karyawan meningkat dalam waktu singkat dari 13 menjadi 70 orang. Pada rentang waktu ini juga ditetapkan

pemakaian Internet sebagai medium yang menjembatani kerja dan ide di antara mereka. Pada awal tahun 1990-an, Internet masih merupakan rintisan, yang dipakai hanya di kalangan akademisi dan militer.

Mereka menjadikan perambah (*browser*) Mosaic sebagai landasan awal untuk membuat perambah Java pertama yang dinamai *Web Runner*, terinsipirasi dari film 1980-an, *Blade Runner*. Pada perkembangan rilis pertama, *Web Runner* berganti nama menjadi *Hot Java*.

Pada sekitar bulan Maret 1995, untuk pertama kali kode sumber Java versi 1.0a2 dibuka. Kesuksesan mereka diikuti dengan untuk pemberitaan pertama kali pada surat kabar San Jose Mercury News pada tanggal 23 Mei 1995.

Sayang terjadi perpecahan di antara mereka suatu hari pada pukul 04.00 di sebuah ruangan hotel Sheraton Palace. Tiga dari pimpinan utama proyek, Eric Schmidt dan George Paolini dari Sun Microsistems bersama Marc Andreessen, membentuk Netscape.

Nama Oak, diambil dari pohon oak yang tumbuh di depan jendela ruangan kerja "bapak java", James Gosling. Nama Oak ini tidak dipakai untuk versi release Java karena sebuah perangkat lunak sudah terdaftar dengan merek dagang tersebut, sehingga diambil nama penggantinya menjadi "Java". Nama ini diambil dari kopi murni yang digiling langsung dari biji (kopi tubruk) kesukaan Gosling.

II.1.7.2 J2ME *(Java 2 Micro Edition)*

J2ME adalah bagian dari *Java 2 Standard Edition (*J2SE) yang digunakan untuk implementasi pada perangkat genggam seperti seperti hand phone, PDA (*Personal Digital Assistant),* Palm, PocketPC, Smart-Phone hingga PDA-Phone

yang tidak mampu mendukung secara penuh implementasi menggunakan J2SE. Perangkat tersebut memiliki jumlah memori yang terbatas, menghabiskan sedikit daya dari baterei, layar yang kecil dan bandwith jaringan yang rendah.

J2ME diterjemahkan oleh VM. Program dikompile ke dalam bytecode dan diterjemahkan denga Java Virtual Machine (JVM). Ini berarti bahwa program-program tersebut tidak berhubungan langsung dengan perangkat. J2ME menyediakan suatu interface yang sesuai dengan perangkat. Aplikasi-aplikasi tersebut tidak harus dikompile ulang supaya mampu dijalankan pada mesin yang berbeda.

II.1.7.3 Netbeans

NetBeans IDE adalah IDE open source yang ditulis sepenuhnya dengan bahasa pemrograman Java menggunakan platform NetBeans. NetBeans IDE mendukung pengembangan semua tipe aplikasi Java (J2SE, web, EJB, dan aplikasi mobile). Fitur lainnya adalah sistem proyek berbasis Ant, kontrol versi, dan refactoring.

Pada versi NetBeans IDE 5.5.1 yang dirilis Mei 2007 telah disediakan fitur-fitur Java EE yang sudah ada (termasuk Java Persistence support, EJB-3 dan JAX-WS). Sementara paket tambahannya, NetBeans Enterprise Pack mendukung pengembangan aplikasi perusahaan Java EE 5, meliputi alat desain visual SOA, skema XML, web service dan pemodelan UML. NetBeans C/C++ Pack mendukung proyek C/C++.

Semua fungsi IDE disediakan oleh modul-modul. Tiap modul menyediakan fungsi yang didefinisikan dengan baik, seperti dukungan untuk bahasa pemrograman Java, editing, atau dukungan bagi CVS.

II.1.7.4 WAMP Server

WampServer merupakan lingkungan pengembangan web Windows. Hal ini memungkinkan untuk membuat aplikasi web dengan Apache, PHP dan *database* MySQL. Hal ini juga dilengkapi dengan PHPMyAdmin untuk dengan mudah mengelola *database* .

WampServer menginstal secara otomatis (*installer*), dan penggunaannya sangat intuitif. Anda akan dapat menyetel server tanpa menyentuh pengaturan file.

WampServer merupakan satu-satunya solusi paket yang akan memungkinkan untuk mereproduksi server produksi. WampServer juga memiliki *trayicon* untuk mengelola server dan pengaturannya.



Gambar II.25 WAMP Server

II.1.7.5 Adobe Dreamweaver

Adobe Dreamweaver (dulunya Macromedia Dreamweaver) adalah pengembangan web aplikasi awalnya dibuat oleh Macromedia , dan sekarang dikembangkan oleh Adobe Sistems , yang mengakuisisi Macromedia tahun 2005. Dreamweaver tersedia untuk Mac dan Windows sistem operasi . versi terbaru telah dimasukkan dukungan untuk web teknologi seperti CSS , JavaScript , dan berbagai server-side scripting bahasa dan kerangka termasuk ASP , ColdFusion , dan PHP .

Adapun fitur-fitur yang disediakan oleh Dreamweaver yaitu :

Dreamweaver yang WYSIWYG mode dapat menyembunyikan HTML code rincian halaman dari pengguna, sehingga memungkinkan untuk non-coders untuk membuat halaman web dan situs.

Dreamweaver memungkinkan pengguna untuk melihat situs di web browser secara lokal-instal. Ia juga memiliki alat manajemen situs seperti FTP / SFTP dan WebDAV mentransfer file dan fitur sinkronisasi, kemampuan untuk mencari dan mengganti baris teks atau kode oleh istilah pencarian dan ekspresi reguler di seluruh situs, dan fitur template yang memungkinkan satu sumber *update* kode bersama dan tata letak di seluruh situs tanpa server-side termasuk atau scripting. Panel perilaku juga memungkinkan penggunaan JavaScript dasar tanpa pengetahuan coding, dan integrasi dengan Adobe kerangka AJAX sigap menawarkan akses mudah ke konten dinamis yang dihasilkan dan antarmuka.

Dreamweaver dapat menggunakan pihak ketiga "*Extensions*" untuk memperluas fungsi inti dari aplikasi, yang setiap pengembang web bisa menulis (terutama dalam HTML dan JavaScript).

II.1.8 Media transmisi

Media transmisi adalah media yang dapat mentransmisikan data. Data-data pada jaringan dapat ditransmisikan melalui 3 media :

a. *Copper* media (media tembaga)

b. *Optical* Media (media optik)

c. *Wireless* Media (media tanpa kabel)

II.1.8.1 Copper Media

Copper media merupakan semua media transmisi data yang terbuat dari bahan tembaga. Orang biasanya menyebut dengan nama kabel. Data yang dikirim melalui kabel, bentuknya adalah sinyal-sinyal listrik (tegangan atau arus) digital. Jenis-jenis kabel yang dipakai sebagai transmisi data pada jaringan :

1. Koaksial
2. STP
3. UTP

1. Kabel Koaksial

Kabel ini sering digunakan sebagai kabel antena TV. Disebut juga sebagai kabel BNC (*Bayonet Naur Connector*). Kabel ini merupakan kabel yang paling banyak digunakan pada LAN, karena memiliki perlindungan terhadap derau yang lebih tinggi, murah, dan mampu mengirimkan data dengan kecepatan standar .Ada

2 jenis yaitu RG-58 (10Base2) dan RG-8 (10Base5). Ada 3 jenis konektor pada kabel Coaxial, yaitu T konektor, I konektor (socket) dan BNC konektor.

Keuntungan menggunakan kabel koaksial adalah murah dan jarak jangkauannya cukup jauh. Kekurangannya adalah susah pada saat instalasi. Untuk saat ini kabel koaksial sudah tidak direkomendasikan lagi intuk instalasi jaringan

2. Twisted Pair

Twisted Pair terdiri dari 2 jenis :

a) Unshielded Twisted Pair (UTP)

b) Shielded Twisted Pair (STP)

3. Kabel STP (Shielded Twisted Pair)

Keuntungan menggunakan kabel STP adalah lebih tahan terhadap interferensi gelombang elektromagnetik baik dari dalam maupun dari luar. Kekurangannya adalah mahal, susah pada saat instalasi (terutama masalah grounding), dan jarak jangkauannya hanya 100m

3.1 Kabel UTP (Unshielded Twisted Pair)

Keuntungan menggunakan kabel UTP adalah murah dan mudah diinstalasi. Kekurangannya adalah rentan terhadap interferensi gelombang elektromagnetik, dan jarak jangkauannya hanya 100m

3.2 Optical Media

Ada tiga jenis kabel *fiber optic* yang biasanya digunakan, yaitu single mode, multimode dan plastic optical fiber yang berfungsi sebagai petunjuk cahaya dari ujung kabel ke ujung kabel lainnya. Dari *transmitter receiver*, yang

mengubah pulsa elektronik ke cahaya dan sebaliknya, dalam bentuk *light-emitting diode* ataupun laser.

Kabel *fiber optic* single mode merupakan fiber glass tunggal dengan diameter 8.3 sampai 10 mikrometer, memiliki satu jenis transmisi yang dapat mengantarkan data berkapasitas besar dengan kecepatan tinggi untuk jarak jauh, dan membutuhkan sumber cahaya dengan lebar spektrum yang lebih kecil. Kemampuan kabel jenis single mode dalam mengantarkan transmisi adalah 50 kali lebih cepat dari kabel jenis multimode, karena memiliki *core* yang lebih kecil sehingga dapat menghilangkan setiap distorsi dan pulsa cahaya yang tumpang tindih.

Kabel *fiber optic* multimode terbuat dari fiberglass dengan diameter lebih besar, yaitu 50 sampai dengan 100 mikrometer yang dapat mengantarkan data berkapasitas besar dengan kecepatan tinggi untuk jarak menengah. Apabila jarak yang ditempuh lebih dari 3000 kaki, akan terjadi distorsi sinyal pada sisi penerima yang mengakibatkan transmisi data menjadi tidak akurat. Sedang *plastic optical fiber* adalah kabel berbasis plastik terbaru yang menjamin tingkat performa yang sama dengan *fiber glass* dalam jarak pendek dengan biaya yang jauh lebih murah.

Saat ini, *fiber optic* telah digunakan sebagai standar kabel data dalam biding physical layer telekomunikasi atau jaringan, seperti perangkat TV kabel, juga sistem keamanan yang menggunakan *Closed Circuit Television (CCTV)*, dan lain sebagainya

Bahan dasar dari optical media adalah kaca dengan ukuran yang sangat kecil (skala mikron).Biasanya dikenal dengan nama *fibre optic* (serat optic).Data yang dilewatkan pada medium ini dalam bentuk cahaya (laser atau inframerah)

3.3 Wireless Network

Saat ini sudah banyak digunakan jaringan tanpa kabel (*wireless network*), transmisi data menggunakan sinar infra merah atau gelombang mikro untuk menghantarkan data. Walaupun kedengarannya praktis, namun kendala yang dihadapi disini adalah masalah jarak, *bandwidth*, dan mahalnya biaya. Namun demikian untuk kebutuhan LAN di dalam gedung, saat ini sudah dikembangkan teknologi wireless untuk Active Hub (*Wireless Access Point*) dan Wireless LAN Card (pengganti NIC), sehingga bisa mengurangi semrawutnya kabel transmisi data pada jaringan komputer. *Wireless Access Point* juga bisa digabungkan (*up-link*) dengan ActiveHub dari jaringan yang sudah ada. Media transmisi wireless menggunakan gelombang radio frekuensi tinggi. Biasanya gelombang elektromagnetik dengan frekuensi 2.4 Ghz dan 5 Ghz. Data-data digital yang dikirim melalui wireless ini akan dimodulasikan ke dalam gelombang elektromagnetik ini.

II.1.9 GPRS

GPRS *(General Packet Radio Service)* adalah suatu teknologi yang memungkinkan pengiriman dan penerimaan data lebih cepat jika dibandingkan dengan penggunaan teknologi *Circuit Switch Data* atau CSD. Sering disebut pula dengan teknologi 2,5G

Sistem GPRS dapat digunakan untuk transfer data (dalam bentuk paket data) yang berkaitan dengan e-mail, data gambar (MMS), dan penelusuran (browsing) internet. Layanan GPRS dipasang pada jenis ponsel tipe GSM dan IS-136, walaupun jaringan GPRS saat ini terpisah dari GSM.

GPRS merupakan sistem transmisi berbasis paket untuk GSM yang menggunakan prinsip *'tunnelling'*. Ia menawarkan laju data yang lebih tinggi. Laju datanya secara kasar sampai 160 kbps dibandingkan dengan 9,6kbps yang dapat disediakan oleh rangkaian tersakelar GSM. Kanal-kanal radio ganda dapat dialokasikan bagi seorang pengguna dan kanal yang sama dapat pula digunakan secara berbagi (*'sharing'*) di antara beberapa pengguna sehingga menjadi sangat efisien.

Dalam teorinya GPRS menjanjikan kecepatan mulai dari 56 kbps sampai 115 kbps, sehingga memungkinkan akses internet, pengiriman data multimedia ke komputer, notebook dan handheld computer. Namun, dalam implementasinya, hal tersebut sangat tergantung faktor-faktor sebagai berikut:

1. Konfigurasi dan alokasi time slot pada level BTS
2. Software yang dipergunakan
3. Dukungan fitur dan aplikasi ponsel yang digunakan.