

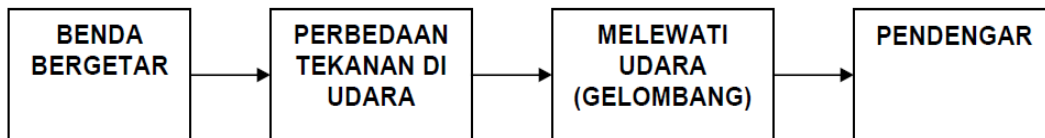
## BAB II

### LANDASAN TEORI

#### 2.1 Pengertian suara

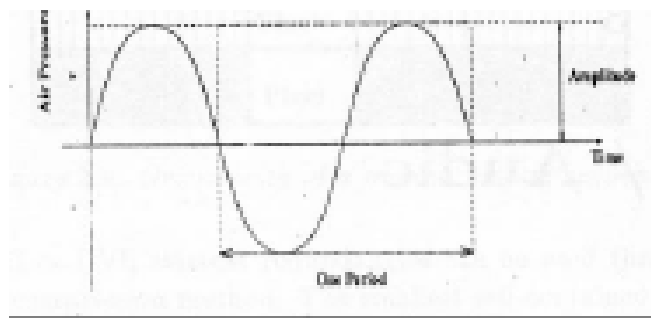
##### 2.1.1 Suara

Suara adalah sesuatu yang dihasilkan oleh getaran yang berasal dari benda bergerak, sesuatu yang menghasilkan bunyi, dan suara adalah sebuah tekanan gelombang udara, maka memiliki nilai kontinu terhadap waktu (*analog*).



Gambar 2.1. Skema Suara

Suara berhubungan erat dengan rasa “mendengar”. Suara/bunyi biasanya merambat melalui udara. Suara/bunyi tidak bisa merambat melalui ruang hampa. Gelombang suara bervariasi dalam tingkatan tekanan suara (*amplitudo*) dan dalam frekuensi. Jumlah waktu yang diperlukan untuk terjadinya suatu getaran atau gelombang dinamakan perioda ( $T$ ). Sedangkan jumlah gelombang yang terjadi setiap detik dinamakan frekuensi ( $f$ ) dengan satuan m/dt (Hz).



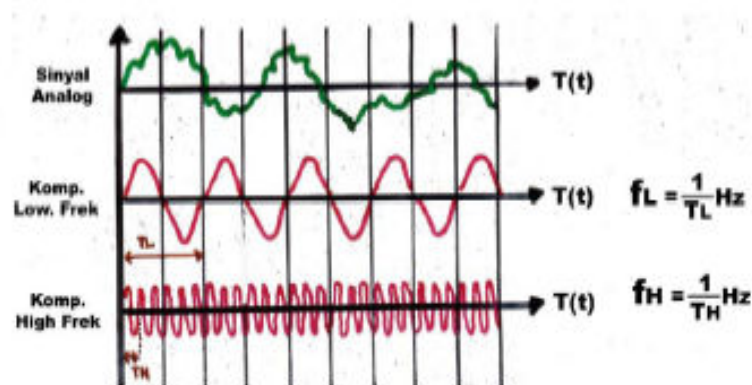
Gambar 2.2. Gelombang suara

### 2.1.2 Frekuensi

Frekuensi adalah jumlah gelombang yang terjadi setiap detik. Jenis-jenis frekuensi ada 4, sebagai berikut:

1. *Infra sound*                      0 - 20 Hz
2. Pendengaran manusia            20 Hz – 20 KHz
3. *Ultra sound*                      20 KHz – 1 GHz
4. *Hyper sound*                      1 GHz – 10 THz

Manusia bersuara dengan frekuensi 50 Hz – 10 KHz, Sinyal suara musik memiliki frekuensi 20 Hz – 20 KHz, Sistem multimedia menggunakan suara yang berada dalam jarak pendengaran manusia. Suara yang berada pada jarak pendengaran manusia disebut “*AUDIO*” dan gelombangnya sebagai “*ACCOUSTIC SIGNALS*”. Suara di luar jarak pendengaran manusia dapat dikatakan sebagai “*NOISE*” (getaran yang tidak teratur dan tidak berurutan dalam berbagai frekuensi, tidak dapat didengar manusia).



Gambar 2.3. Signal analog terdiri dari sebuah frekuensi sinusoidal dimana amplitudonya serta fasenya berubah secara “relative” antara satu dengan lainnya

### 2.1.3 Amplitudo

Amplitudo adalah keras lemahnya bunyi atau tinggi rendahnya gelombang. Satuan amplitudo adalah desibel (db), bunyi dapat merusak telinga jika tingkat volumenya lebih besar dari 85 db dan pada ukuran 130 db akan mampu membuat hancur gendang telinga.

### 2.1.4 Velocity

*Velocity* adalah kecepatan perambatan gelombang bunyi sampai ke telinga pendengar. Satuan yang digunakan : m/s pada udara kering dengan suhu 20° C (68° F) m kecepatan rambat suara sekitar 343 m/s.[5]

### 2.1.5 Audio Digital

*Audio digital* merupakan versi *digital* dari suara analog. Pengubahan suara analog menjadi suara *digital* membutuhkan suatu alat yang disebut *Analog To Digital Converter* (ADC). ADC akan mengubah amplitudo sebuah gelombang analog ke dalam waktu interval (*sampling*) sehingga menghasilkan representasi *digital* dari suara. *Sampling* adalah melakukan pencuplikan amplitudo gelombang suara pada tiap satu satuan waktu.

Berlawanan dengan ADC, *Digital to Analog Converter* (DAC) akan mengubah suara *digital* ke alat suara analog (*speaker*). *Audio digital* merupakan representasi dari suara asli (*original sound*). Dengan kata lain, *audio digital* merupakan sampel suara. Kualitas perekaman digital tergantung pada seberapa sering sampel diambil (angka *sampling* atau frekuensi dihitung dalam KiloHertz atau seribu sampel per detik). Tiga frekuensi *sampling* yang paling sering digunakan

dalam multimedia adalah kualitas CD 44.1 kHz, 22.05 kHz, 11.25 kHz dengan ukuran sampel 8 bit dan 16 bit. Ukuran sampel 8 bit menyediakan 256 unit deskripsi jarak dinamis atau amplitudo (level suara dalam satu waktu).

Ketika pengambilan *sampling* gelombang dengan ADC mempunyai kendali terhadap dua variabel, yaitu:

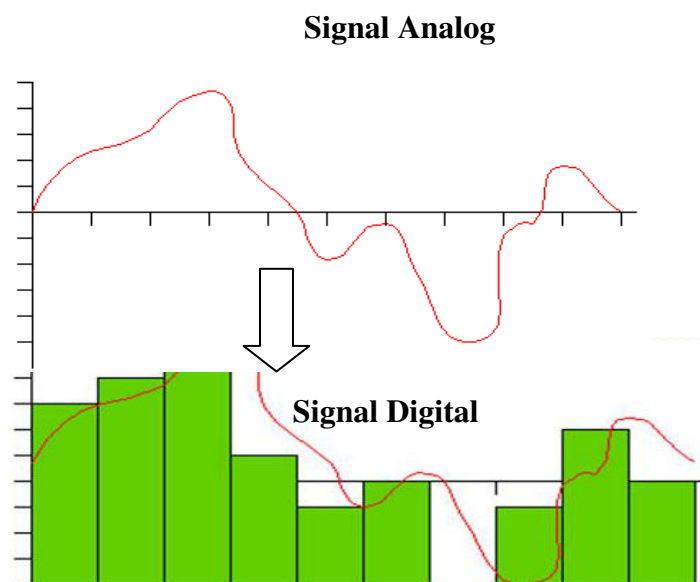
1. *Sampling Rate*

*Sampling rate* adalah mengendalikan berapa banyak sampel yang akan diambil per-detiknya.

2. *Sampling Precision*

*Sampling precision* adalah mengendalikan berapa banyak gradasi (tingkat kuantisasi) yang dimungkinkan ketika mengambil sampel.

Dengan gambar sebagai berikut:



Gambar 2.4. Gelombang suara analog menjadi digital

### 2.1.6 Wave form audio (WAV)

WAV adalah format audio standar *Microsoft* dan *IBM* untuk *PC*. WAV biasanya menggunakan coding PCM (*Pulse Code Modulation*). WAV adalah data tidak terkompres sehingga seluruh sampel audio disimpan semuanya di *harddisk*. *Software* yang dapat menciptakan WAV dari analog *sound* misalnya *windows sound recorder*. WAV jarang digunakan di internet karena ukurannya yang relative besar. Maksimum ukuran file WAV adalah 2GB.[1]

## 2.2 Sinyal Suara Ucapan

Sinyal suara ucapan manusia dapat dipandang sebagai sinyal yang berubah lambat terhadap waktu (*slowly time varying signal*), jika diamati pada selang waktu yang singkat yaitu 5-100 ms. Pada selang waktu tersebut, katakarakteristik sinyal suara ucapan dapat dianggap *stasioner*. Untuk selang waktu yang lebih panjang (dengan orde 0.2 detik atau lebih), karakteristik sinyal berubah untuk merefleksikan suara berbeda yang diucapkan.

### 2.2.1. Klasifikasi Berdasarkan Sinyal Eksitasi

Berdasarkan sinyal eksitasi yang dihasilkan pada proses produksi suara, sinyal suara ucapan dapat dibagi menjadi tiga bagian yaitu *silence*, *unvoiced*, dan *voiced*:

1. Sinyal *silence* : sinyal pada saat tidak terjadi proses produksi suara ucapan, dan sinyal yang diterima oleh pendengar dianggap sebagai bising latar belakang.
2. Sinyal *unvoiced* : terjadi pada saat pita suara tidak bergetar, dimana sinyal eksitasi berupa sinyal random.

3. Sinyal *voiced* : terjadi jika pita suara bergetar, yaitu pada saat sinyal eksitasi berupa sinyal pulsa kuasi-periodik. Selama terjadinya sinyal *voiced* ini, pita suara bergetar pada frekuensi fundamental inilah yang dikenal sebagai *pitch* dari suara tersebut.

### 2.2.2. Analisis Sinyal Ucapan

Informasi yang terdapat di dalam sebuah sinyal ucapan dapat dianalisis dengan berbagai cara. Beberapa peneliti telah membagi beberapa level pendekatan untuk menggambarkan informasi tersebut, yaitu *level akustik*, *fonetik*, dan *fonologi*.

#### 1. *Level Akustik*

Sinyal ucapan merupakan variasi tekanan udara yang dihasilkan oleh sistem artikulasi. Untuk menganalisa aspek-aspek akustik dari sebuah sinyal ucapan, dapat dilakukan dengan transformasi dari bentuk sinyal ucapan menjadi sinyal listrik dengan menggunakan transduser seperti *microphone*, telepon, dan sebagainya. Setelah melalui berbagai pengolahan sinyal digital, maka akan diperoleh informasi yang menunjukkan sifat-sifat akustik dari sinyal ucapan tersebut yang meliputi frekuensi fundamental, intensitas, dan distribusi energi spektral.

#### 2. *Level Fonetik*

Level ini menggambarkan bagaimana suatu sinyal suara yang diproduksi oleh organ-organ di dalam tubuh manusia.

### 3. *Level Fonologi*

Di dalam level ini, dikenal istilah fonem yang merupakan unit terkecil yang membentuk sebuah kalimat atau ucapan. Deskripsi ini memuat informasi durasi, intensitas, dan *pitch* dari fonem-fonem yang membangun kalimat tersebut.[8]

#### 2.2.3 **Preemphasis**

*Preemphasis* merupakan salah satu jenis *filter* yang sering digunakan sebelum sebuah *signal* diproses lebih lanjut. *Filter* ini mempertahankan frekuensi-frekuensi tinggi pada sebuah spektrum, yang umumnya tereliminasi pada saat proses produksi suara. Tujuan dari *preemphasis filter* ini adalah:

1. Mengurangi *noise ratio* pada sinyal, sehingga dapat meningkatkan kualitas sinyal.
2. Untuk mendapatkan bentuk spektral frekuensi sinyal bicara yang lebih halus.
3. Menyeimbangkan spektrum dari sinyal suara.

Pada saat memproduksi sinyal suara, glotis manusia menghasilkan sekitar -12 dB *octave slope*. Namun ketika energi akustik tersebut dikeluarkan melalui bibir, terjadi peningkatan sebesar +6 dB. Sehingga sinyal yang terekam oleh *microphone* adalah sekitar -6 dB *octave slope*. Dimana bentuk spektral yang relatif bernilai tinggi untuk daerah rendah dan cenderung turun secara tajam untuk daerah frekuensi diatas 2000 Hz. Filter *preemphasis* didasari oleh hubungan *input/output* dalam domain waktu yang dinyatakan dalam persamaan seperti berikut:

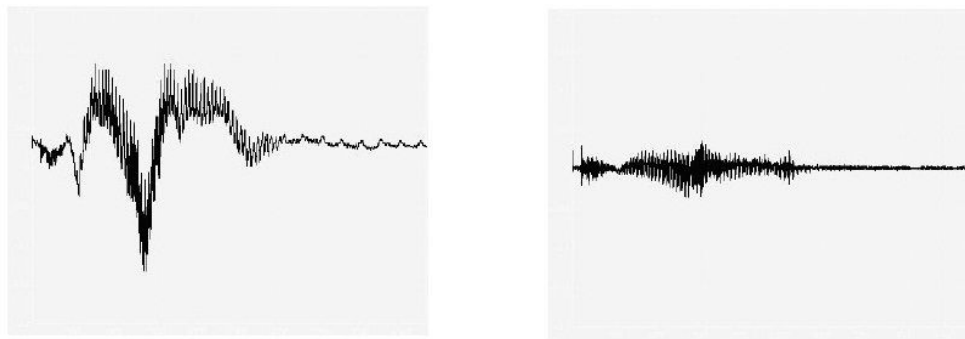
$$y(n) = s(n) - \alpha s(n-1) \dots\dots\dots (1)$$

Dimana:

$y(n)$  = Sinyal hasil *preemphasis*

$s(n)$  = Sinyal sebelum *preemphasis*

$\alpha$  merupakan konstanta *filter preemphasis*, biasanya bernilai 0.97. Dalam bentuk dasar operator  $s$  sebagai unit *filter*, persamaan diatas akan memberikan sebuah *transfer function filter preemphasis* seperti berikut:



a. Sebelum *preemphasis*

b. Sesudah *preemphasis*

Gambar 2.5. Sebelum dan sesudah *preemphasis*

#### 2.2.4 Frame blocking

*Frame Blocking* adalah pembagian suara menjadi beberapa frame yang nantinya dapat memudahkan dalam perhitungan dan analisa suara, satu frame terdiri dari beberapa sampel tergantung tiap berapa detik suara akan disampel dan berapa besar frekuensi samplingnya. Panjang *frame* yang digunakan, sangat mempengaruhi keberhasilan dalam analisa *spektral*. Di satu sisi, ukuran dari *frame* harus sepanjang mungkin untuk dapat menunjukkan *resolusi* frekuensi yang baik. Tetapi di lain sisi, ukuran *frame* juga harus cukup pendek untuk dapat menunjukkan *resolusi* waktu yang baik. Refresentasi fungsi frame blocking sebagai berikut:

$$x(n) = y(M + n)$$



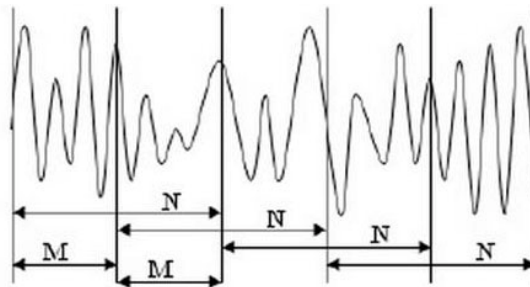
dimana,

$x(n)$ =sinyal sesudah di *frame blocking*

$y$ =sinyal hasil *preemphasis*

$M$ = *overlapping frame*

$n = 1, 2, 3, \dots$



Gambar 2.6. Bentuk sinyal yang di *frame blocking*

Proses *framing* ini dilakukan terus sampai seluruh *signal* dapat terproses. Selain itu, proses ini umumnya dilakukan secara *overlapping* untuk setiap *frame*-nya. Panjang daerah *overlap* yang umum digunakan adalah kurang lebih 30% sampai 50% dari panjang *frame*.

### 2.2.5 Windowing

Suara yang di potong-potong menjadi beberapa *frame* membuat data suara menjadi *discontinue*, hal ini mengakibatkan kesalahan data proses *fourier transform*. Agar tidak terjadi kesalahan data pada proses *fourier transform* maka sampel suara yang telah dibagi menjadi beberapa *frame* perlu dijadikan suara kontinu dengan cara mengalikan tiap *frame* dengan *window* tertentu. Jenis *window* yang dipakai pada proses ini adalah *window hamming*. Berikut ini adalah representasi fungsi *window* terhadap *signal* suara yang diinputkan.

$$\tilde{x}(n) = x(n)w(n)$$

rumus window hamming sebagai berikut:

$$w[n] = 0.54 - 0.46 \cos(2\pi n / (N-1)) \quad \dots\dots\dots (2)$$

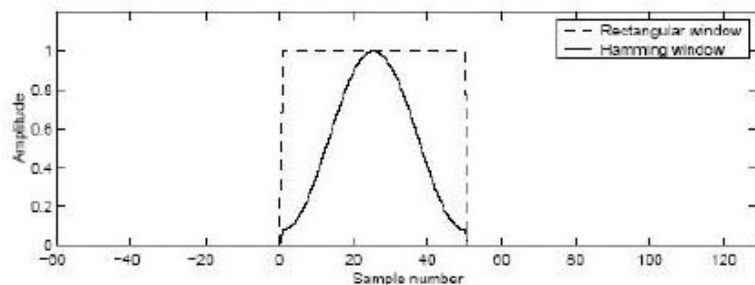
dimana,

$$n = 1, 2, 3, \dots$$

$$\pi = 3.14$$

$$N = \text{panjang frame}$$

Fungsi *window* yang paling sering digunakan dalam aplikasi *voice recognition* adalah *Window Hamming*. Fungsi *window hamming* ini menghasilkan *sidelobe level* yang tidak terlalu tinggi (kurang lebih -43 dB), selain itu *noise* yang dihasilkan pun tidak terlalu besar (kurang lebih 1.36 BINS).[10]



Gambar 2.7. Bentuk gelombang dari *window hamming*

### 2.3 Fast Fourier Transform (FFT)

*Fast fourier transform* adalah suatu metode yang sangat efisien untuk menyelesaikan *transformasi fourier diskrit* yang banyak dipakai untuk keperluan analisa sinyal seperti pemfilteran, *analisa korelasi*, dan *analisa spektrum*. Ada 2 (dua) jenis algoritma FFT yaitu algoritma *fast fourier transform decimation in time* (FFT DIT) dan algoritma *fast fourier transform decimation in frequency* (FFT DIF). Dimana pada FFT DIT inputan disusun/dikelompokkan menjadi kelompok ganjil dan

kelompok genap. Sedangkan pada FFT DIF inputan tetap tetapi output disusun/dikelompokkan menjadi kelompok ganjil dan kelompok genap.

Bentuk rumusannya sebagai berikut:

$$X[k] = \sum_{n=1}^{N-1} x(n) W_N^{kn} \dots\dots\dots (3)$$

Dimana :

$X[k]$  = Merupakan *magnitude frekuensi*

$x(n)$  = Nilai sampel sinyal

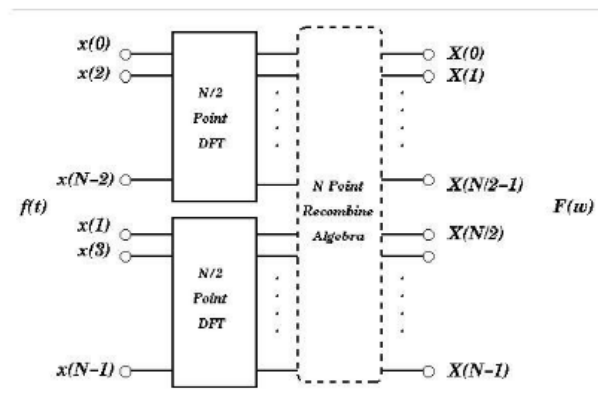
$$W = \frac{1}{N} \cos(2\pi k n / N) - j \sin(2\pi k n / N)$$

$N$  = jumlah sinyal yang akan diproses

$k$  = sinyal yang diproses

$$\text{Gunakan rumus } \left| [R^2 + I^2]^{1/2} \right|$$

FFT dilakukan dengan membagi  $N$  buah titik pada transformasi *fourier diskrit* menjadi 2, masing-masing  $(N/2)$  titik *transformasi*. Proses memecah menjadi 2 bagian ini diteruskan dengan membagi  $(N/2)$  titik menjadi  $(N/4)$  dan seterusnya hingga diperoleh titik minimum.[7]

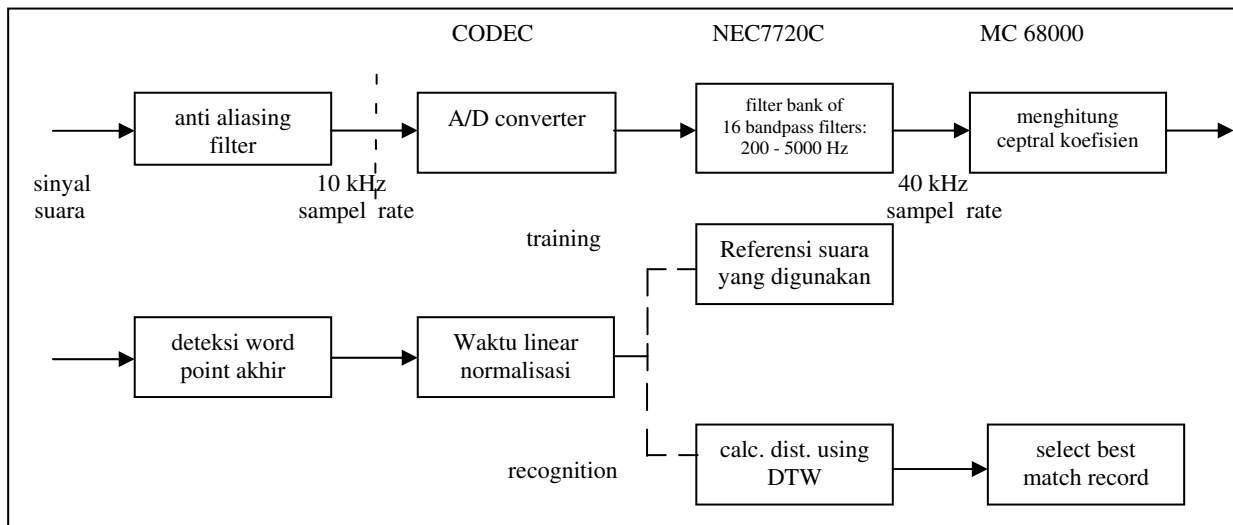


Gambar 2.8. Pembagian sinyal suara menjadi dua kelompok

## 2.4 Dynamic Time Warping (DTW)

Satu masalah yang cukup rumit dalam pengenalan suara (*voice recognition*) adalah proses perekaman yang terjadi seringkali berbeda durasinya, biarpun kata atau kalimat yang diucapkan sama. Bahkan untuk satu suku kata yang sama atau vocal yang sama seringkali proses perekaman terjadi dalam durasi yang berbeda. Sebagai akibatnya proses pencocokan antara sinyal uji dengan sinyal *record* suara di *database* seringkali tidak menghasilkan nilai yang optimal.

Sebuah teknik yang cukup populer di awal perkembangan teknologi pengolahan sinyal pengenalan suara (*voice recognition*) adalah dengan memanfaatkan sebuah teknik *Dynamic Time Warping* (DTW) yang juga lebih dikenal sebagai *dynamic programming*. Teknik ini ditujukan untuk mengakomodasi perbedaan waktu antara proses perekaman saat pengujian dengan suara yang tersedia pada *record* suara di *database*. Prinsip dasarnya adalah dengan memberikan sebuah rentang '*steps*' dalam ruang (dalam hal ini sebuah frame-frame waktu dalam sampel, frame-frame waktu dalam *database*) dan digunakan untuk mempertemukan lintasan yang menunjukkan *local match* terbesar (kemiripan) antara time frame yang lurus. Total '*similarity cost*' yang diperoleh dengan algoritma ini merupakan sebuah indikasi seberapa bagus *sample* suara dan *record* suara di *database* ini memiliki kesamaan, yang selanjutnya akan dipilih *best-matching record* suara di *database*.



Gambar 2.9. Blok diagram dari sistem recognition dengan metode DTW[12].

#### 2.4.1 Dynamic Time Warping Algorithm

*Dynamic Time Warping algorithm* [Sakoe , H. & S. Chiba-8] adalah algoritma yang menghitung *optimal warping path* antara dua waktu. Algoritma ini menghitung baik antara nilai *warping path* dari dua waktu dan jaraknya. Misalnya, memiliki dua barisan numerik:

$$\begin{aligned} \{A\} &= (a_1, a_2, \dots, a_n) \\ \{B\} &= (b_1, b_2, \dots, b_n) \dots\dots\dots (4) \end{aligned}$$

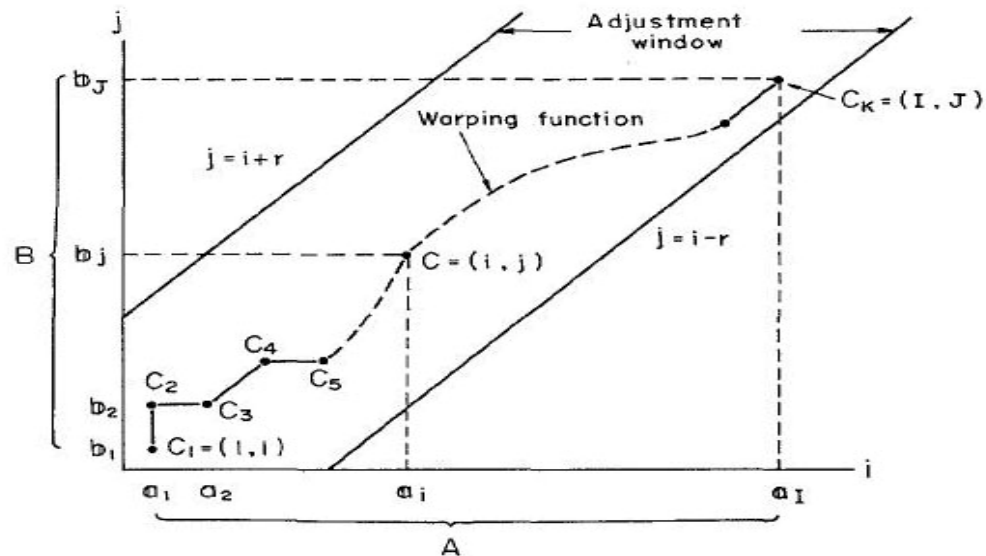
dengan pemisalan ini, maka dapat dikatakan bahwa panjang dua barisan ini bisa saja berbeda. Dimana masing-masing barisan A dan B dikembangkan sepanjang sumbu i dan sumbu j. Dimana pola suara ini adalah dari pola suara yang sama, perbedaan waktu antara barisan A dan B dapat digambarkan berdasarkan urutan point  $c = (i, j)$  :

$$F = c(1), c(2), \dots, c(k) \dots\dots\dots (5)$$

Dimana

$$c(k) = (i(k), j(k))$$

Urutan ini dapat dianggap mewakili suatu fungsi pemetaan dari waktu barisan A dan waktu barisan B, hal itu disebut *warping path* sebuah fungsi ketika tidak ada perbedaan waktu.



Gambar 2.10. Fungsi warping dan definisi pengaturan windows

Barisan fungsi *warping* bertepatan dengan diagonal baris  $j = i$ , ini menyimpang jauh dari garis diagonal sebagai perbedaan waktunya.[6]

$$[i(k) - j(k)] \leq r \quad \dots\dots\dots (6)$$

Dimana  $r$  adalah bilangan bulat positif yang tepat disebut *signal-window*,  $k$  adalah jumlah titik sinyal *warping path*. Kondisi ini sesuai dengan fakta bahwa fluktuasi jarak waktu dalam kasus-kasus biasa tidak pernah ada perbedaan jarak waktu terlalu berlebihan.

Algoritma ini memulai dengan penghitungan jarak lokal antara elemen dari barisan menggunakan tipe jarak yang berbeda. Frekuensi yang paling banyak menggunakan metode untuk penghitungan jarak adalah jarak absolut antar nilai dua

elemen. Jika dalam matriks maka dapat ditulis dengan memiliki  $i$  garis dan  $j$  kolom, secara umum:

$$d(c) = d(i,j) = |a_i - b_j| \dots\dots\dots (7)$$

Mulai dengan matrik jarak lokal, kemudian minimum jarak pada *warping*  $F$  menjadi fungsi sebagai berikut :

$$E(F) = \sum_{k=1}^K d(c(k)) \cdot w(k) \dots\dots\dots (8)$$

Dimana  $w(k)$  adalah koefisien bobot non-negatif yang diperkenalkan untuk memungkinkan  $E(F)$  mengukur *fleksibel* karakteristik) dan merupakan jarak yang minimum untuk fungsi *warping*  $F$ . Pencapaian nilai yang minimum saat  $F$  fungsi *warping* ditentukan perbedaan waktu menjadi optimal. Nilai jarak minimum ini dapat dianggap sebagai jarak antara barisan A dan B, masih tersisa setelah eliminasi perbedaan waktu antara barisan A dan B, dan tentu saja diperkirakan waktu akan stabil dengan sumbu fluktuasi. Berdasarkan pertimbangan ini, waktu normalisasi jarak antara dua barisan suara A dan B didefinisikan sebagai berikut:

$$D(A,B) = \min_F \left[ \frac{\sum_{k=1}^K d(c(k)) \cdot w(k)}{\sum_{k=1}^K w(k)} \right] \dots\dots\dots (9)$$

Dimana penyebut  $\sum w(k)$  yang digunakan untuk mengkompensasi pengaruh  $K$  (jumlah titik pada fungsi  $F$  *warping*). Persamaan (9) tidak lebih dari sebuah fundamental definisi jarak waktu normal. Efektif karakteristik ini sangat tergantung pada pengukuran spesifikasi fungsi *warping path* dan definisi koefisien

pembobotan. Pengukuran karakteristik dari normalisasi jarak waktu akan bervariasi, menurut sifat barisan suara (terutama waktu sumbu ekspresi barisan suara) yang harus diselesaikan. Oleh karena itu, masalah sekarang terbatas pada kasus yang paling umum di mana ada dua kondisi sebagai berikut:

1. Kondisi barisan suara adalah waktu sampel yang umum dan periode sampling konstan.
2. Kondisi dimana pengetahuan tentang bagian dari barisan suara berisi informasi yang penting.

Dalam hal ini, adalah wajar untuk mempertimbangkan setiap bagian dari barisan suara mengandung jumlah informasi linguistik yang sama. Dimana  $w(k)$  adalah elemen yang dimiliki *warping path* dan  $k$  adalah jumlahnya. Penghitungannya dibuat untuk dua barisan diperlihatkan pada gambar dibawah dan *warping path* diberi *highlight*.

	-2	10	-10	15	-13	20	-5	14	2
3	5	12	25	37	53	70	78	89	90
-13	16	28	15	43	37	70	78	105	104
14	32	20	39	16	43	43	62	62	74
-7	37	37	23	38	22	49	45	66	71
9	48	38	42	29	44	33	47	50	57
-2	48	50	46	46	40	55	36	52	54

Gambar 2.11. Warping path

#### 2.4.1.1 Pembatasan Fungsi warping

Fungsi *warping*  $F$ , yang didefinisikan oleh persamaan (5) adalah model jarak waktu *fluktuasi* dalam barisan suara. Dalam kata lain, fungsi  $F$  bila dilihat sebagai pemetaan dari jarak waktu barisan A ke jarak waktu barisan B, harus menjaga



struktur *linguistik* penting dalam barisan jarak waktu A dan sebaliknya. Barisan suara yang penting pada jarak waktu struktur adalah *kontinuitas*, *monotonisitas* (atau pembatasan waktu relatif dalam sebuah suara), pembatasan kecepatan *parameter akustik transisi* dalam suara, dan sebagainya. Kondisi ini dapat diwujudkan sebagai pembatasan berikut pada fungsi warping  $F(c(k)=(i(k),j(k)))$ .

Ada tiga kondisi yang menentukan pada DTW algoritma yang meyakinkan konvergensi cepat:

1. *Monotony-path* yang tidak pernah ada kembalian, yang berarti antara index  $i$  dan  $j$  digunakan untuk menyebrang barisan tidak pernah berkurang.

$$i(k-1) \leq i(k) \text{ dan } j(k-1) \leq j(k).$$

2. *Continuity-path* berkembang yang secara berangsur-angsur, tahap per tahap, yang berarti index  $i$  dan  $j$  naik dengan maksimum kenaikan 1 unit setiap langkahnya. Sebagai hasil dari kedua pembatasan memiliki hubungan antara dua titik berturut-turut sebagai berikut:

$$c(k-1) = \begin{cases} (i(k), j(k)-1), \\ (i(k)-1, j(k)-1), \dots\dots \\ \text{or } (i(k)-1, j(k)). \end{cases} \quad (10)$$

3. *Boundary-path* mulai dari pojok kiri bawah dan berakhir pada pojok kanan atas.

Karena prinsip optimasi dalam *dynamic programming* diimplementasikan pada teknik “*backward*”, mengidentifikasi *warping path* menggunakan tipe struktur

dinamis yang disebut stack. Seperti *algorithm dynamic programming* lainnya. *Dynamic Time Warping* (DTW) memiliki kompleksitas polinomial. Ketika barisan memiliki banyak elemen, minimal ada dua ketidaknyamanan:

1. Mengingat matriks yang besar,
2. Menampilkan banyak perhitungan jarak.

Ada perbaikan dalam standar *Dynamic Time Warping Algorithm* yang merangkum dua masalah diatas dengan nama: FastDTW (*Fast Dynamic Time Warping*) [Stan Salvador, Philip Chan - 6]. Solusi yang ditawarkan berisi pembagian jarak matriks ke dalam 2,4,8,16,dst. Dengan cara ini, perhitungan jarak diperlihatkan pada matriks yang lebih kecil dan *warping path* digunakan saat menggabungkan dari matriks kecil tadi.

#### 2.4.1.2 Prinsip-prinsip Dynamic Time Warping Algorithm

1. Prinsip Optimalitas: jika solusi total optimal, maka bagian solusi sampai tahap ke-k juga optimal. Pada *Dynamic Time Warping Algorithm*, rangkaian keputusan yang optimal dibuat dengan menggunakan Prinsip *Optimalitas*.
2. Prinsip optimalitas berarti bahwa jika bekerja dari tahap k ke tahap k + 1, dapat menggunakan hasil optimal dari tahap k tanpa harus kembali ke tahap awal. Jika pada setiap tahap menghitung ongkos (*cost*), maka dapat dirumuskan bahwa:

$$N = \sum_{k=1}^K w(k) \dots\dots\dots (11)$$

Koefesien normalisasi adalah ketetapan dari fungsi *warping path* maka dapat di lakukan persamaan:

$$D(A, B) = \frac{1}{N} \min_F \left[ \sum_{k=1}^K d(c(k)) \cdot w(k) \right] \dots\dots\dots (12)$$

3. Dengan prinsip optimalitas ini dijamin bahwa pengambilan keputusan pada suatu tahap adalah keputusan yang benar untuk tahap-tahap selanjutnya.

#### **2.4.1.3 Karakteristik Persoalan *Dynamic Time Warping Algorithm***

1. Persoalan dapat dibagi menjadi beberapa tahap (*stage*), yang pada setiap tahap hanya diambil satu keputusan.
2. Masing-masing tahap terdiri dari sejumlah status (*state*) yang berhubungan dengan tahap tersebut. Secara umum, status merupakan bermacam kemungkinan masukan yang ada pada tahap tersebut.
3. Hasil dari keputusan yang diambil pada setiap tahap ditransformasikan dari status yang bersangkutan ke status berikutnya pada tahap berikutnya.
4. Ongkos (*cost*) pada suatu tahap meningkat secara teratur (*steadily*) dengan bertambahnya jumlah tahapan.
5. Ongkos pada suatu tahap bergantung pada ongkos tahap-tahap yang sudah berjalan dan ongkos pada tahap tersebut.
6. Keputusan terbaik pada suatu tahap bersifat *independent* terhadap keputusan yang dilakukan pada tahap sebelumnya.
7. Adanya hubungan rekursif yang mengidentifikasikan keputusan terbaik untuk setiap status pada tahap k memberikan keputusan terbaik untuk setiap status pada tahap k+1 telah diketahui. Rumus tersebut menjadi:

### 2.4.2 Menggunakan *Dynamic Time Warping Algorithm* dalam pengenalan suara

*Vocal Signal Analysis*. Suara merambat melalui udara sebagai gelombang longitudinal dengan kecepatan yang tergantung densitas udara. Cara yang paling mudah untuk merepresentasikan suara adalah dengan grafik sinusoidal. Grafik tersebut merepresentasikan variasi dari tekan udara tergantung waktunya.

Ada tiga hal yang membentuk gelombang suara, yaitu amplitudo, frekuensi, dan fase. Amplitudo diukur menggunakan satuan decibels (DB), pengukuran dilakukan dengan mengikuti fungsi logaritma sebagai standar suara. Pengukuran amplitudo menggunakan satuan decibels (DB) sangat penting karena ini representasi langsung bagaimana suara dirasakan oleh orang. Frekuensi adalah banyaknya gelombang per-detik, biasa diukur menggunakan skala Hertz (Hz). Kemudian, fase mengukur posisi dari awal gelombang sinus. Kondisi awal :

$$g(1,1) = 2d(1,1) \quad \dots\dots\dots (13)$$

Persamaan *Dynamic Time Warping Algorithm*:

$$g(i,j) = \min \begin{cases} g(i,j-1) + d(i,j) \\ g(i-1,j-1) + 2d(i,j) \\ g(i-1,j) + d(i,j) \end{cases} \quad \dots\dots\dots (14)$$

Membatasi kondisi (*Signal-Window*)

$$j - r \leq i \leq j + r$$

Jarak sisa normalisasi

$$D(A,B) = \frac{1}{N} g(I, J)$$

$$N = I + J. \quad \dots\dots\dots (15)$$

Untuk membuat suara menjadi kurva sinusoidal, digunakanlah teorema *Fourier*. *Word detection*, teknologi sekarang ini bisa mengidentifikasi secara akurat awal dan akhir satu kata diucapkan dalam audio streaming, tergantung pada proses sinyal yang berbeda dengan waktu. Dengan mengevaluasi energi dan rata-rata magnitude dalam waktu yang singkat dan menghitung rata-rata *zero-crossing rate*. Menetapkan poin awal dan akhir merupakan masalah sederhana jika rekaman audio dilakukan dalam kondisi yang ideal. Dalam kasus ini, rasio *signal-noise*-nya tinggi karena mudah untuk menentukan lokasi dalam stream yang terdiri dari sinyal valid dengan analisis sampel. Dalam kondisi sebenarnya tidak-lah sesederhana itu, *background-noise* memiliki intensitas yang signifikan dan dapat mengganggu proses isolasi kata dalam stream.

Algoritma yang paling baik untuk mengisolasi kata adalah *Rabiner-Lamel Algorithm*. Jika mempertimbangkan *signal-window* {s1, s2, ..., sn} dimana n adalah jumlah sampel dari *window* dan si, i = 1, n adalah ekspresi numerik dari sampel, energi yang berasosiasi dengan *signal-window*:

$$E(n) = \frac{1}{n} \sum_{i=1}^n s_i^2 \quad \dots\dots\dots (16)$$

Rata-rata *zero-crossing rate*:

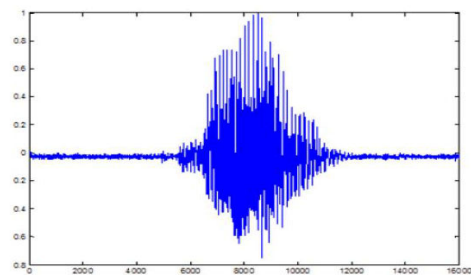
$$ZCR(n) = \sum_{i=1}^{n-1} \text{sign}(s_i) \cdot \text{sign}(s_{i+1}) \quad \dots\dots\dots (17)$$

where  $\text{sign}(s_i) = \begin{cases} 1 & \text{if } s_i > 0 \\ 0 & \text{if } s_i < 0 \end{cases}$ .

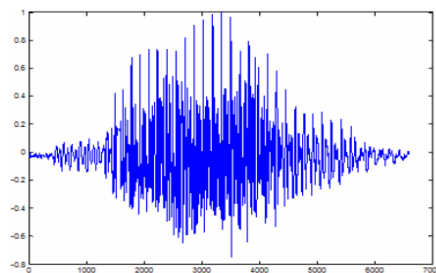
Metode menggunakan tiga numerik level: dua untuk energi (superior, inferior) dan satu untuk rata-rata *zero crossing rate*.

### 2.4.3 Menggunakan *Dynamic Time Warping Algorithm* dalam *Word Identification*

Identifikasi kata bisa dilakukan dengan perbandingan langsung numerik form dari sinyal atau dari spektrogram sinyal. Proses perbandingan dalam dua kasus di atas keduanya memiliki perbedaan panjang jarak dan non-linear. *Dynamic Time Warping Algorithm* sukses dalam mengurutkan penyelesaian masalah dengan menemukan hubungan *warping path* ke jarak optimal antara dua jarak yang berbeda panjang. Ada beberapa kekhususan ketika algoritma ini digunakan dalam dua kasus: dibawah ini adalah contoh perbandingan suara sinyal A dan sinyal B.



Gambar 2.12. Sinyal suara A ‘ucapan’



Gambar 2.13. Sinyal suara B “ucapan”

Keterangan gambar A dan B :

1. Perbandingan langsung dari bentuk numerik atau sinyal. Dalam kasus ini, untuk setiap jarak numerik, sebuah jarak baru dibuat, maka jarak yang memiliki dimensi yang lebih kecil. Jarak numerik bisa memiliki ribuan nilai numerik, ketika subsekuens bisa memiliki ratusan. Mengurangi jumlah nilai numerik bisa dilakukan dengan menghilangkan yang ada diantara ekstrim poin. Proses pengurangan jarak numerik ini tidak diperkenankan menggabungkan bentuknya dan proses ini bisa membawa ke pengurangan pengenalan presisi. Tetapi, membuat nilainya naik dalam hal kecepatan, presisi pada faktanya meningkat dengan memperluas jumlah kata dalam kamus.
2. Representasi sinyal spektrogram dan mengaplikasikannya pada DTW algorithm untuk perbandingan spektrogram. Metode ini terdiri dari pembagian numerik sinyal dalam banyak *window* (interval) yang akan *overlap*. Setiap window, angka real di interval, *transformasi* akan dihitung dan disimpan dalam matriks: spektrogram suara. Parameter yang digunakan akan sama untuk semua operasi penghitungan, yaitu panjang *window*, panjang *transformasi*, dan panjang *overlap window* untuk dua *window* berturut-turut. *Transformasi* secara simetris berhubungan dengan pada pusat dan bilangan kompleks dari setengah konjugat bilangan kompleks dari bilangan simetris pada setengah pertama. Dalam kenyataan ini, hanya nilai dari setengah pertama yang disimpan, maka spektrogram akan menjadi matriks bilangan kompleks, banyaknya baris sama dengan setengah dari panjang *transformasi* dan banyaknya kolom tergantung dari

panjangnya suara. DTW akan diaplikasikan dalam matriks bilangan real yang dihasilkan dari konjugasi nilai spektogram.[11]

Istilah–istilah yang ada di metode *Dynamic time warping* diantaranya sebagai berikut:

- a. *Optimal warping path* :Jarak nilai efektif yang dapat dicapai.
- b. *Fluktuasi* :Segala hal yang bisa dilihat di dalam sebuah grafik.
- c. *Normalisasi* :Suatu teknik untuk mengorganisasikan data ke dalam table-tabel.
- d. *Signal window* :Arus data yang mengalir melalui jalur transmisi.
- e. *Stage* :Tingkatan, level.
- f. *State* :Penjelasan dari kondisi internal yang mengindikasikan apa yang baru–baru ini telah dikerjakan dan tugas apa yang akan dikerjakan selanjutnya.
- g. *Signal vocal analysis* :Menganalisa sinyal suara yang diucapkan.
- h. *Signal noise* :Sinyal tidak diketahui yang masuk ketika melakukan pengambilan sampel suara.
- i. *Word detection* :Melacak kondisi tertentu yang mempengaruhi sistem komputer atau data yang dipakai untuk mengerjakan satu unit data yang diproses.
- j. *Transformasi* :Perubahan atau pergantian bentuk.



- k. *Zero crossing rate* : Adalah tingkat tanda perubahan sepanjang sinyal tingkat dimana perubahan sinyal dari positif ke negatif atau sebaliknya.

## 2.5 Perancangan Sistem

Menurut Harianto Kristianto, 1992:145 perancangan sistem dapat didefinisikan sebagai penggambaran, perencanaan, dan pembuatan sketsa atau pengaturan dari beberapa elemen yang terpisah ke dalam satu kesatuan yang utuh. Suatu sistem akan di analisis dan di rancang oleh satu orang atau satu tim yang disebut analisis sistem. Perancangan sistem adalah kombinasi atau seri dari proses yang mengangkat aktifitas-aktifitas :

- a. Identifikasi suatu *problem*
- b. Analisa suatu *problem*
- c. Menyelesaikan suatu *problem*

Dengan kata lain perancangan sistem adalah proses pengamatan terhadap suatu badan usaha dengan tujuan dapat mengetahui situasi operasinya dan apakah badan usaha tersebut memerlukan suatu perbaikan atau tidak.

### 2.5.1 Tujuan Perancangan Sistem

Menurut Harianto Kristianto, 1992:147 tahap perancangan sistem mempunyai 2 tujuan , yaitu :

- a. Untuk memenuhi kebutuhan kepada pemakai sistem.

- b. Untuk memberikan gambaran yang jelas dan rancangan bangun yang lengkap kepada pemrogram komputer dan ahli–ahli teknik lainnya yang terlibat.

Agar dapat mencapai tujuan yang kedua yang lebih condong pada desain sistem yang terinci yaitu pembuatan rancang bangun yang jelas dan lengkap untuk digunakan dalam pembuatan program komputer, maka analisa sistem harus dapat mencapai sasaran sebagai berikut :

- a. Perancangan harus berguna, mudah di pahami dan mudah digunakan.
- b. Perancangan sistem harus dapat mendukung tujuan utama perusahaan atau yayasan dengan lebih mendefinisikan pada tahap perencanaan sistem yang selanjutnya pada tahap analisa sistem.
- c. Perancangan sistem harus lebih efisien dan efektif untuk dapat mendukung pengolahan data, identifikasi dan verifikasi data yang akan dilakukan oleh komputer.
- d. Perancangan sistem harus dapat mempersiapkan bangunan yang terdiri oleh masing-masing komponen dari sistem informasi.

### **2.5.2 Analisa Terstruktur**

Analisa terstruktur merupakan suatu pendekatan umum yang memungkinkan terbentuknya pemahaman analisa terhadap komponen–komponen sistem secara bertahap, yang dituju sebenarnya adalah mengorganisasikan tugas–tugas yang berkaitan dengan penetapan kebutuhan, sehingga sistem terstruktur di berlakukan terhadap penyusunan dan pembentukan proses, di lakukan terhadap

penyertaan dari semua paparan rinci yang relevan tentang sistem, pembentukan solusi yang sebaik mungkin dan menyediakan komunikasi yang dapat memberikan fasilitas komunikasi antara analisa dan pemakai. Kegiatan yang dilakukan pada analisa ini adalah : menganalisa sistem yang ada, mempelajari dan mengetahui apa yang dikerjakan oleh sistem yang ada. Menspesifikasikan sistem yaitu menspesifikasikan masukan yang digunakan, database yang ada, proses yang dilakukan dan keluaran yang dihasilkan.

Tujuan dari analisa adalah menentukan kebutuhan pemakai secara akurat di dalam analisa terstruktur terdapat pendekatan yang dipakai, antara lain : Pendekatan *Top Down*, yaitu memecah masalah ke dalam bagian-bagian terkecil atau perlevel, sehingga mudah untuk diselesaikan, Pendekatan modul, yaitu membagi sistem ke dalam modul yang dapat beroperasi tanpa ketergantungan. Menggunakan alat-alat bantu dalam bentuk grafik dan teks, sehingga mudah untuk di mengerti dan di koreksi bila terjadi perubahan.

## **2.6 Basis Data**

Basis data (*database*), atau sering pula dieja basisdata, adalah kumpulan informasi yang disimpan di dalam komputer secara sistematis sehingga dapat diperiksa menggunakan suatu program komputer untuk memperoleh informasi dari basis data tersebut. Perangkat lunak yang digunakan untuk mengelola dan memanggil kueri (*query*) basis data disebut sistem manajemen basis data (*database management system*, DBMS). Sistem basis data dipelajari dalam ilmu informasi.

Istilah "basis data" berawal dari ilmu komputer. Meskipun kemudian artinya semakin luas, memasukkan hal-hal di luar bidang elektronika, artikel ini mengenai basis data komputer. Catatan yang mirip dengan basis data sebenarnya sudah ada sebelum revolusi industri yaitu dalam bentuk buku besar, kuitansi dan kumpulan data yang berhubungan dengan bisnis.

Konsep dasar dari basis data adalah kumpulan dari catatan-catatan, atau potongan dari pengetahuan. Sebuah basis data memiliki penjelasan terstruktur dari jenis fakta yang tersimpan di dalamnya: penjelasan ini disebut skema. Skema menggambarkan obyek yang diwakili suatu basis data, dan hubungan di antara obyek tersebut. Ada banyak cara untuk mengorganisasi skema, atau memodelkan struktur basis data: ini dikenal sebagai model basis data atau model data. Model yang umum digunakan sekarang adalah model relasional, yang menurut istilah layman mewakili semua informasi dalam bentuk tabel-tabel yang saling berhubungan dimana setiap tabel terdiri dari baris dan kolom (definisi yang sebenarnya menggunakan terminologi matematika). Dalam model ini, hubungan antar tabel diwakili dengan menggunakan nilai yang sama antar tabel. Model yang lain seperti model hierarkis dan model jaringan menggunakan cara yang lebih eksplisit untuk mewakili hubungan antar tabel.

Istilah *basis data* mengacu pada koleksi dari data-data yang saling berhubungan, dan perangkat lunaknya seharusnya mengacu sebagai *sistem manajemen basis data* (*database management system/DBMS*). Jika konteksnya sudah

jelas, banyak *administrator* dan *programmer* menggunakan istilah basis data untuk kedua arti tersebut.

Perangkat lunak basis data yang banyak digunakan dalam pemrograman dan merupakan perangkat basis data aras tinggi (*high level*):

- |                                       |                          |
|---------------------------------------|--------------------------|
| 1. <i>DB2</i>                         | 14. <i>FoxPro</i>        |
| 2. <i>Microsoft SQL Server</i>        | 15. <i>Visual FoxPro</i> |
| 3. <i>Oracle</i>                      | 16. <i>Arago</i>         |
| 4. <i>Sybase</i>                      | 17. <i>Force</i>         |
| 5. <i>Interbase</i>                   | 18. <i>Recital</i>       |
| 6. <i>XBase</i>                       | 19. <i>dbFast</i>        |
| 7. <i>Firebird</i>                    | 20. <i>dbXL</i>          |
| 8. <i>MySQL</i>                       | 21. <i>Quicksilver</i>   |
| 9. <i>PostgreSQL</i>                  | 22. <i>Clipper</i>       |
| 10. <i>Microsoft Access</i>           | 23. <i>FlagShip</i>      |
| 11. <i>dBase III</i>                  | 24. <i>Harbour</i>       |
| 12. <i>Paradox</i>                    | 25. <i>Visual dBase</i>  |
| 13. <i>Lotus Smart Suite Approach</i> |                          |

Selain perangkat lunak di atas, terdapat juga perangkat lunak pemrograman basis data aras rendah (*low level*), diantaranya:

1. *Btrieve*
2. *Tsunami Record Manager*

DBMS (*Database Management System*) adalah *software* yang menangani semua akses ke basis data. Secara konsep apa yang terjadi adalah sebagai berikut :

1. User melakukan pengaksesan basis data untuk informasi yang diperlukannya menggunakan suatu bahasa manipulasi data, biasanya disebut SQL.
2. DBMS menerima *request* dari user dan menganalisa request tersebut
3. DBMS memeriksa skema *eksternal user*, pemetaan eksternal/konseptual, skema konseptual, pemetaan konseptual/internal, dan struktur penyimpanan.
4. DBMS mengeksekusi operasi-operasi yang diperlukan untuk memenuhi permintaan *user*.

### 2.6.1 Bahasa Basis Data

*DBMS* merupakan perantara bagi pemakai dengan basis data dalam disk. Cara berkomunikasi/berinteraksi antara pemakai dengan basis data tersebut diatur dalam suatu bahasa khusus yang ditetapkan oleh perusahaan pembuat *DBMS*. Bahasa tersebut dapat disebut sebagai bahasa basis data yang terdiri atas sejumlah perintah yang diformulasikan oleh *user* dan diproses oleh *DBMS* untuk melakukan suatu aksi atau pekerjaan tertentu.

Ada 3 bahasa yang digunakan dalam basis data yaitu :

1. DDL (*Data Definition Language* )

Merupakan bahasa definisi data yang digunakan untuk membuat dan *memanage* objek *database* seperti *database*, tabel dan *view*.

## 2. DML (*Data Manipulation Language*)

Merupakan bahasa manipulasi data yang digunakan untuk memanipulasi data pada objek *database* seperti tabel.

## 3. DCL (*Data Control Language*)

Merupakan bahasa yang digunakan untuk mengendalikan pengaksesan data.

### 2.6.2 Flowchart dan Simbol–Simbolnya

Flowchart adalah penggambaran secara grafik dari langkah-langkah dan urutan-urutan prosedur dari suatu program. Flowchart menolong seorang analis dan programmer untuk memecahkan suatu masalah dan keadaan dari segmen-segmen yang lebih kecil serta menolong dalam menganalisis alternatif lain dalam pengoperasian.

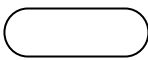


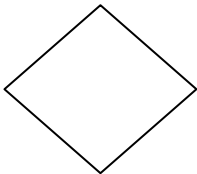
Flowchart biasanya mempermudah penyelesaian suatu masalah khususnya masalah yang perlu dipelajari dan dievaluasi lebih lanjut. Bila seorang analis dan programmer akan membuat flowchart, maka disini ada beberapa petunjuk yang harus di perhatikan, antara lain :

- a. Flowchart digambarkan dari halaman atas kebawah dan dari kiri ke kanan.
- b. Kapan aktifitas di mulai dan berakhir harus ditentukan dengan jelas.
- c. Aktivitas yang digambarkan harus didefinisikan secara hati-hati dan definisi ini harus dimengerti oleh pembacanya.
- d. Setiap langkah dari aktifitas harus diuraikan dengan menggunakan eskripsi kata kerja.
- e. Setiap langkah dari aktifitas harus berada pada urutan yang benar.




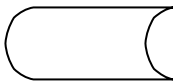
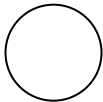
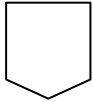
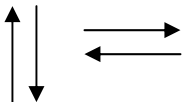
- f. Lingkup dan jarak dari aktifitas yang sedang digambarkan harus ditelusuri dengan hati-hati. Percabangan-percabangan yang memotong aktifitas yang sedang digambarkan tidak perlu digambarkan pada flowchart yang sama.
- g. Simbol konektor harus digunakan dan percabangannya diletakkan pada halaman yang terpisah atau dihilangkan seluruhnya bila percabangan tidak berkaitan dengan sistem.

Berikut ini ditampilkan daftar simbol-simbol yang sering dipakai pada flowchart beserta keterangannya :

*Tabel 2.1 Simbol Flowchart*

No	Simbol Flowchart	Nama	Keterangan
1		<i>Terminal</i>	Tampil pada awal Diagram Alur (berisi kata “Start”) atau pada akhir proses (berisi kata “End”).
2		<i>Prosesing</i>	Satu atau beberapa himpunan penugasan yang akan dilaksanakan secara berurutan.
3		<i>Predefined Process</i>	Rincian operasi berada di tempat lain
4		<i>Decision</i>	Ada dua alternatif yang dapat ditentukan untuk melaksanakan jalur pada Diagram Alur. Jalur yang harus diikuti dipilih pada saat pelaksanaan Algoritma dengan meng-adakan percobaan apakah langkah-langkah



			yang ditetapkan di dalam bagan sudah terpenuhi atau belum.
5		<i>Data Input/Output</i>	Data yang akan dibaca dan dimasukkan ke dalam memori komputer dari suatu alat input atau data dan harus melewati memori untuk dike-luarkan dari alat output.
6		<i>Manual Input</i>	Pengoperasian data yang dimasukkan masih secara manual.
7		<i>Document</i>	Data yang akan dibaca dari memori komputer dan sudah menjadi sebuah dokumen.
8		<i>Data Stored</i>	Digunakan sebagai tempat untuk penyimpanan data.
9		<i>Connector</i>	Tanda untuk memisahkan Diagram Alur menjadi beberapa halaman. Ditulis di tempat/halaman untuk menyambung bagian Diagram Alur yang terputus.
10		<i>Off – Page Connector</i>	Tanda untuk memisahkan Diagram Alur untuk menyambung dan menuju ke halaman berikutnya.
11		<i>Flowline</i>	Menunjukkan bagan instruksi alur kerja.

## 2.7 Data Flow Diagram

Data Flow Diagram (DFD) adalah sebuah diagram yang menggunakan notasi untuk menggambarkan arus data dari sistem. Penggunaan notasi ini sangat membantu dalam komunikasi dengan pemakai sistem untuk memahami sistem secara logika.

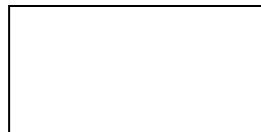
Penggunaan DFD sebagai *modelling tools* dipopulerkan oleh Tom De Marco (1978) dan Gane & Sarson (1979) dengan menggunakan pendekatan metoda analisis sistem terstruktur (*structured system analysis methode*).

### 2.7.1 Simbol-simbol DFD

Beberapa simbol yang digunakan pada DFD adalah sebagai berikut :

#### 1. External Entity

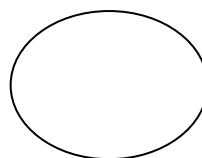
Entity bisa berupa orang atau organisasi yang berada diluar sistem yang memberikan data kepada sistem atau yang menerima informasi dari sistem.



Gambar 2.14. Simbol eksternal entity

#### 2. Proses

Menggambaran apa yang dilakukan oleh sistem. Berfungsi mentransformasikan satu atau beberapa data masukan menjadi satu atau beberapa data keluaran sesuai dengan spesifikasi yang diinginkan.

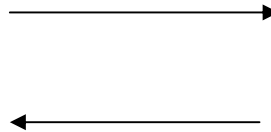


Gambar 2.15. Simbol proses

### 3. Data Flow

Menggambarkan aliran data dari suatu entity ke entity lainnya. Arah panah menggambarkan aliran data. Aliran data bisa terjadi antara :

- a. Dua proses yang berurutan
- b. Dari data store ke proses dan sebaliknya.
- c. Dari source ke proses
- d. Dari proses ke link.



Gambar 2.16. Simbol data flow

### 4. Data Store

Menggambarkan tempat penyimpanan data. Proses dapat mengambil data dari atau memberi data ke store.



Gambar 2.17. Simbol Data Store

#### 2.7.2 Level Proses Pada DFD

Data Flow Diagram disusun berdasarkan tingkatan yang disebut levelisasi DFD.

##### 1. Diagram Konteks.

Diagram konteks merupakan diagram yang paling tinggi, terdiri dari suatu Proses dan menggambarkan ruang lingkup sistem.

## 2. Diagram Zero.

Merupakan diagram diantara diagram konteks dan diagram detail yang menggambarkan proses utama dari data flow diagram.

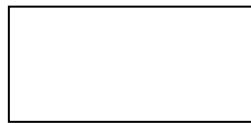
## 2.8 Entity Relationship Diagram

Merupakan suatu struktur diagram yang menggambarkan persepsi dari pemakai dan berisi obyek-obyek dasar yang disebut entity dan entity-entity tersebut saling berhubungan.

### 2.8.1 Komponen Pada Entity Relationship Diagram

#### 1. Entity

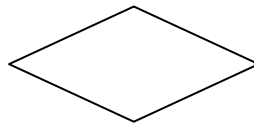
Segala sesuatu yang bisa dijelaskan dengan data, kelompok benda atau obyek dan diberi nama dengan kata benda.



Gambar 2.18. Simbol Entity

#### 2. Relationship

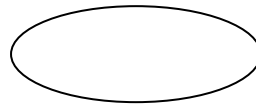
Asosiasi antara satu atau beberapa entity, diberi nama dengan kata kerja.



Gambar 2.19. Simbol Relationship

### 3. Attribute

Properti atau karakteristik suatu entity atau relationship yang mempunyai penjelasan detail tentang entity atau relationship tersebut.



Gambar 2.20. Simbol atribut

#### 2.8.2 Jenis Relationship

##### 1. *One to one* ( 1 : 1 )

Hubungan antar sebuah atribut dengan atribut lainnya dalam satu *file* yang sama mempunyai hubungan satu lawan satu.

##### 2. *Many to one* ( M : 1 atau 1 : M )

Hubungan antar satu atribut dengan atribut lainnya dalam satu *file* yang sama mempunyai hubungan satu lawan banyak.

##### 3. *Many to many* ( N : M )

Hubungan antar satu atribut dengan atribut lainnya dalam satu *file* yang sama mempunyai hubungan banyak lawan banyak.

#### 2.8.3 Kamus Data

Kamus data adalah katalog fakta tentang data dan kebutuhan-kebutuhan informasi dari suatu sistem informasi. kamus data merupakan sebuah daftar yang terorganisasi dari elemen data yang berhubungan dengan sistem, dengan definisi yang teliti sehingga pemakai dan analisis sistem akan memiliki pemahaman yang umum mengenai *input*, *output*, komponen penyimpanan serta *kalkulasi intermediate*.

Pendefinisian data tersebut dilakukan dengan menggunakan notasi yang umum digunakan dalam menganalisa sistem yaitu dengan menggunakan sejumlah simbol. Kamus data biasanya dipelihara secara otomatis oleh sistem manajemen *database*.

Cara mendefinisikan kamus data yaitu :

- a. Menggambarkan arti aliran data atau penyimpanan yang ditunjuk dalam DFD.
- b. Menggabungkan komponen dari kumpulan data yang mengalir yaitu kumpulan komponen yang mungkin bisa dipecah lagi menjadi data elementer.
- c. Menggambarkan data yang tersimpan.
- d. Menentukan nilai dibagian elementer dari informasi yang relevan di DFD dan data *store*nya

## **2.9 Metode Pengembangan Perangkat Lunak**

Merupakan sebuah model *chaos* yang menggambarkan “perkembangan P/L sebagai sebuah kesatuan dari pemakai ke pengembang dan ke teknologi” disebut dengan “*Prescriptive*” karena Menentukan sekumpulan elemen proses (Aktivitas, Aksi, tugas, produk kerja, jaminan kualitas, dll untuk setiap proyek), setiap model proses juga menentukan alur kerjanya.

Pada saat kerja bergerak maju menuju sebuah sistem yang lengkap, keadaan yang digambarkan secara rekursif diaplikasikan kepada kebutuhan pemakai dan spesifikasi teknis P/L pengembang Saat ini, *prescriptive* memberikan jawaban secara

definitive untuk masalah pengembangan P/L dalam setiap perubahan lingkungan komputasi

### 2.9.1 Model Sekuensial Linear

Pada Tugas akhir ini menggunakan metode penggunaan perangkat lunak siklus kehidupan klasik atau model air terjun. Model ini mengusulkan sebuah pendekatan kepada perkembangan *software* yang sistematis dan sekuensial yang mulai pada tingkat dan kemajuan sistem pada seluruh analisis, desain, kode, pengujian, dan pemeliharaan. Dimodelkan setelah siklus rekayasa konvensional, model sekuensial linier melingkupi aktivitas-aktivitas sebagai berikut :

#### 1. Rekayasa dan pemodelan sistem/informasi

Karena sistem merupakan bagian dari sebuah sistem yang lebih besar, kerja dimulai dengan membangun syarat dari semua elemen sistem dan mengalokasikan beberapa subset dari kebutuhan ke *software* tersebut. Pandangan sistem ini penting ketika *software* harus berhubungan dengan elemen-elemen yang lain seperti *software*, manusia, dan database. Rekayasa dan analisis sistem menyangkut pengumpulan kebutuhan pada tingkat sistem dengan sejumlah kecil analisis serta desain tingkat puncak. Rekayasa informasi mencakup juga pengumpulan kebutuhan pada tingkat bisnis strategis dan tingkat area bisnis.

#### 2. Analisis kebutuhan *Software*

Proses pengumpulan kebutuhan diintensifkan dan difokuskan, khususnya pada *software*. Untuk memahami sifat program yang dibangun, analis harus memahami

domain informasi, tingkah laku, dan *interface* yang diperlukan. Kebutuhan baik untuk sistem maupun *software* didokumentasikan dan dilihat lagi dengan pelanggan.

### 3. Desain

Desain *software* sebenarnya adalah proses multi langkah yang berfokus pada empat atribut sebuah program yang berbeda; struktur data, arsitektur *software*, representasi *interface*, dan detail (algoritma) prosedural. Proses desain menterjemahkan syarat/kebutuhan ke dalam sebuah representasi *software* yang dapat diperkirakan demi kualitas sebelum dimulai pemunculan kode. Sebagaimana persyaratan, desain didokumentasikan dan menjadi bagian dari konfigurasi *software*.

### 4. Generasi Kode

Desain harus diterjemahkan kedalam bentuk mesin yang bias dibaca. Langkah pembuatan kode melakukan tugas ini. Jika desain dilakukan dengan cara yang lengkap, pembuatan kode dapat diselesaikan secara mekanis.

### 5. Pengujian

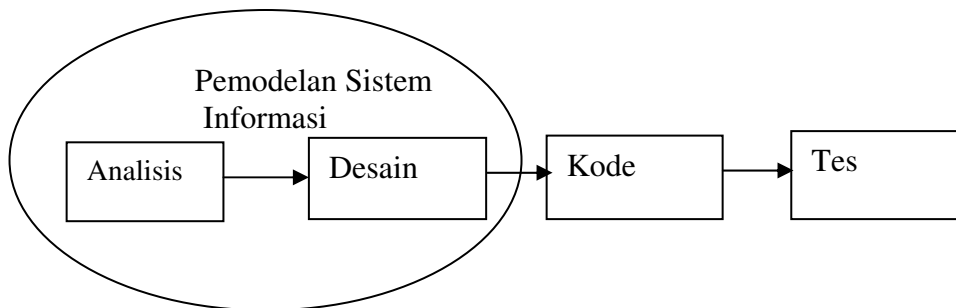
Sekali program dibuat, pengujian program dimulai. Proses pengujian berfokus pada logika internal *software*, memastikan bahwa semua pernyataan sudah diuji, dan pada eksternal fungsional, yaitu mengarahkan pengujian untuk menemukan kesalahan-kesalahan dan memastikan bahwa input yang dibatasi akan memberikan hasil aktual yang sesuai dengan hasil yang dibutuhkan.

### 6. Pemeliharaan

*Software* akan mengalami perubahan setelah disampaikan kepada pelanggan (perkecualian yang mungkin adalah *software* yang dilekatkan). Perubahan akan



terjadi karena kesalahan–kesalahan ditentukan, karena *software* harus disesuaikan untuk mengakomodasi perubahan–perubahan di dalam lingkungan eksternalnya (contohnya perubahan yang dibutuhkan sebagai akibat dari perangkat peripheral atau sistem operasi yang baru), atau karena pelanggan membutuhkan perkembangan fungsional atau unjuk kerja. Pemeliharaan *software* mengaplikasikan lagi setiap fase program sebelumnya dan tidak membuat yang baru lagi.



Gambar 2.21. Model Sekuensial Linear

Masalah yang kadang terjadi ketika model sekuensial linier diaplikasikan adalah :

1. Jarang sekali proyek nyata mengikuti aliran sekuensial yang dianjurkan oleh model. Meskipun model linier bisa mengakomodasi iterasi, model ini melakukannya dengan cara tidak langsung. Sebagai hasilnya, perubahan–perubahan dapat menyebabkan keraguan pada saat tim proyek berjalan.

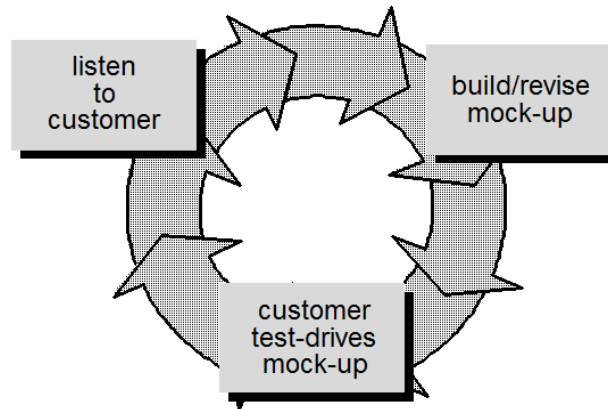
2. Kadang–kadang sulit bagi pelanggan untuk menyatakan semua kebutuhannya secara eksplisit. Model linier sekuensial memerlukan hal ini dan mengalami kesulitan untuk mengakomodasi ketidakpastian natural yang ada pada bagian awal beberapa proyek. Pemodelan Sistem Informasi

3. Pelanggan harus bersifat sabar. Sebuah versi kerja dari program-program kerja itu tidak akan diperoleh sampai akhir waktu proyek dilalui. Sebuah kesalahan besar, jika tidak terdeteksi sampai program yang bekerja tersebut dikaji ulang, bisa menjadi petaka.

4. Pengembang sering melakukan penundaan yang tidak perlu. Sifat alami dari siklus kehidupan klasik membawa kepada *blocking state* di mana banyak anggota tim proyek harus menunggu tim yang lain untuk melengkapi tugas yang saling memiliki ketergantungan. Blocking state cenderung menjadi lebih lazim pada awal dan akhir sebuah proses sekuensial linier.[4]

### **2.9.2 Model Prototype**

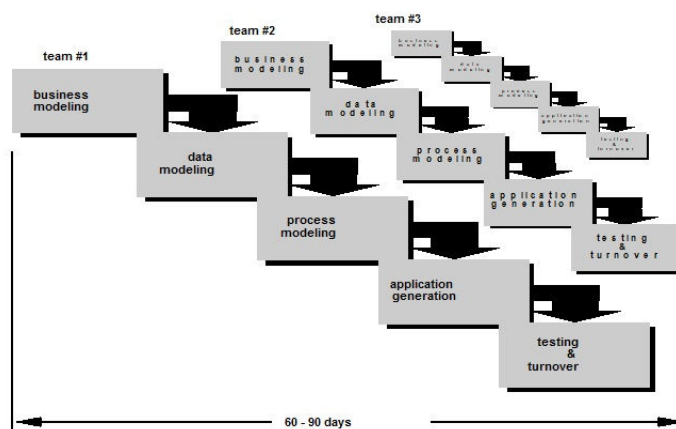
Model *prototype* dibangun dari mengumpulkan berbagai kebutuhan, kemudian tim pengembang akan bertemu dengan pelanggan untuk menentukan tujuan dari perangkat lunak, dan mengidentifikasi kebutuhan-kebutuhan yang telah diketahui oleh pelanggan, dan batasan-batasan apa saja yang dapat dikategorikan sebagai tugas utama. Hasilnya akan dibangun rancangan sementara yang mewakili berbagai aspek dari perangkat lunak yang kelak akan digunakan oleh pelanggan/pengguna (seperti bentuk pendekatan input yang digunakan dan bentuk output). Idealnya model *prototype* melayani sebuah mekanisme untuk mengidentifikasi kebutuhan perangkat lunak. Dimana jika nantinya sebuah model *prototype* berhasil dibuat, seorang *developer* harus berusaha mendayagunakan tools yang ada (semisal, *report generator*, *windows manager*) dapat bekerja dengan baik (cepat).



Gambar 2.22. Model Prototype

### 2.9.3 Model RAD

*Rapid Application Development* (RAD) adalah sebuah model proses perkembangan perangkat lunak sekuensial linier yang menekankan siklus perkembangan yang sangat pendek. Model RAD ini merupakan sebuah adaptasi “kecepatan tinggi” dari model sequensial linier dimana perkembangan cepat dicapai dengan menggunakan pendekatan konstruksi berbasis komponen.



Gambar 2.23. Model RAD

#### 2.9.4 Model Spiral

Model spiral (*spiral model*) adalah model proses *software* yang *evolusioner* yang merangkai sifat iteratif dari prototipe dengan cara kontrol dan aspek sistematis dari model sekuensial linier. Model ini berpotensi untuk pengembangan versi pertambahan *software* secara cepat. Di dalam model spiral, *software* dikembangkan di dalam suatu deretan pertambahan. Selama awal iterasi, rilis inkremental bisa merupakan sebuah model atau prototipe kertas. Selama iterasi berikutnya, sedikit demi sedikit dihasilkan versi sistem rekayasa yang lebih lengkap.

Model spiral dibagi menjadi sejumlah aktifitas kerangka kerja, disebut juga wilayah tugas, di antara tiga sampai enam wilayah tugas, yaitu :

1. Komunikasi Pelanggan

Tugas–tugas yang dibutuhkan untuk membangun komunikasi yang efektif di antara pengembangan dan pelanggan.

2. Perencanaan

Tugas–tugas yang dibutuhkan untuk mendefinisikan sumber–sumber daya, ketepatan waktu, dan proyek informasi lain yang berhubungan.

3. Analisis Risiko

Tugas–tugas yang dibutuhkan untuk menaksir risiko–risiko, baik manajemen maupun teknis.

4. Perekayasaan

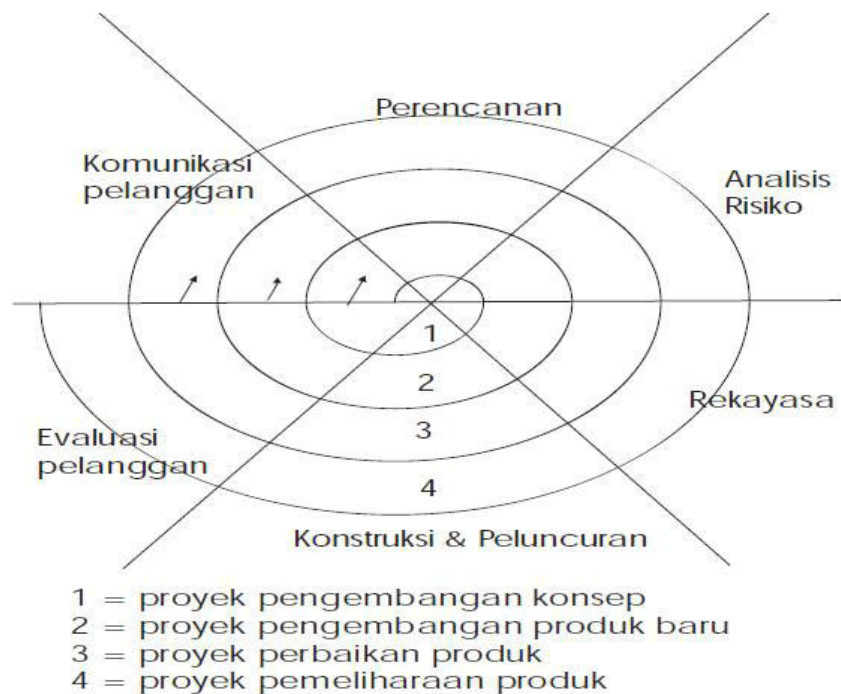
Tugas–tugas yang dibutuhkan untuk membangun satu atau lebih representasi dari aplikasi tersebut.

## 5. Konstruksi dan peluncuran

Tugas-tugas yang dibutuhkan untuk mengkonstruksi, menguji, memasang (*instal*) dan memberikan pelayanan kepada pemakai (contohnya pelatihan dan dokumentasi).

## 6. Evaluasi pelanggan

Tugas-tugas yang dibutuhkan untuk memperoleh umpan balik dari pelanggan dengan didasarkan pada evaluasi representasi *software*, yang dibuat selama masa perekayasaan, dan diimplementasikan selama masa pemasangan.[8]



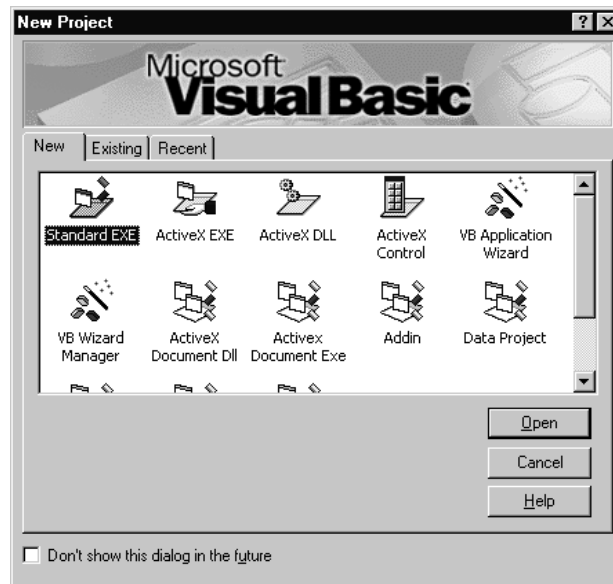
Gambar 2.24. Model Spiral

## 2.10 Visual Basic

Visual Basic 6.0 diperkenalkan pada tahun 1998 seiring dengan semakin meluasnya pengguna sistem operasi Windows 98. Sebagaimana versi 5.0, versi 6.0 juga merupakan bagian dari paket Visual Studio 6.0 bersama bahasa pemrograman C dan *FoxPro*. Berbagai fasilitas ditemukan dalam VB 6.0, misalnya akses *database* dengan teknologi baru yaitu ADO (*Activex Data Objects*), kemampuan internet yang diperluas, elemen dan data control yang semakin banyak, kemampuan untuk membuat control sendiri, fasilitas dan alat bantu yang disebut *wizard*, dan berbagai kemampuan lainnya. Dengan berbagai fasilitas yang begitu banyak dan kemampuan yang sangat luas, VB 6.0 menjadi bahasa pemrograman visual yang paling banyak digunakan bahkan hingga saat ini.

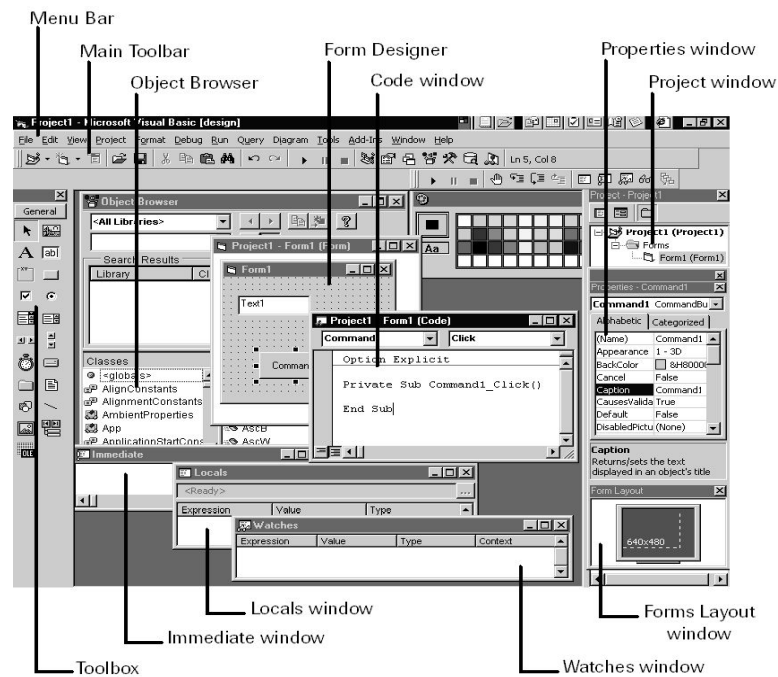
### 2.10.1 Pengenalan IDE Visual Basic

IDE adalah (*Integrated Development Environment*), atau juga disebut sebagai *Integrated Design/Debugging Environment*, adalah perangkat lunak komputer yang berfungsi untuk membantu pemrograman dalam mengembangkan perangkat lunak. Singkatnya, IDE merupakan suatu lingkungan pengembangan aplikasi yang terintegrasi, lengkapnya dengan beragam tools atau utilitas pendukung. Saat IDE Visual Basic aktif maka akan dihadapkan kepada suatu pilihan terhadap jenis *Project* yang ingin anda buat.



Gambar 2.25. Dialog box new *project* visual basic

Window Visual Basic 6 menggunakan model MDI (*Multiple Document Interface*). Berikut ini adalah gambar yang menunjukkan bagian-bagian dan nama-nama window yang dapat tampil pada IDE Visual Basic.



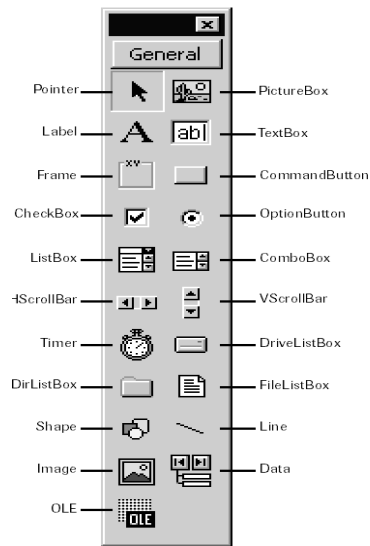
Gambar 2.26. IDE Visual Basic dengan window yang terbuka.

Penjelasan–penjelasan pada gambar IDE visual basic sebagai berikut:

1. *Menu Bar*, digunakan untuk memilih tugas-tugas tertentu seperti menyimpan *project*, membuka *project*, dll.
2. *Main Toolbar*, digunakan untuk melakukan tugas-tugas tertentu dengan cepat.
3. *Window Project*, window ini berisi gambaran dari semua modul yang terdapat dalam aplikasi dan dapat menggunakan icon *Toggle Folders* untuk menampilkan modul-modul dalam window tersebut secara di group atau berurut.
4. *Window Form Designer*, window ini merupakan tempat untuk merancang user *interface* dari aplikasi, jadi window ini menyerupai kanvas bagi seorang pelukis.
5. *Window Toolbox*, Window ini berisi komponen-komponen yang dapat digunakan untuk mengembangkan user *interface*.
6. *Window Code*, merupakan tempat untuk menulis koding.
7. *Window Properties*, merupakan daftar properti-properti *object* yang sedang terpilih. Sebagai contohnya anda dapat mengubah warna tulisan (*foreground*) dan warna latarbelakang (*background*).
8. *Window Color Palette*, adalah fasilitas cepat untuk mengubah warna suatu *object*.
9. *Window Form Layout*, akan menunjukkan bagaimana form bersangkutan ditampilkan ketika runtime.

*Window Toolbox* merupakan window yang sangat penting. Dari window ini mengambil komponen-komponen (*object*) yang akan ditanamkan pada form untuk membentuk *user interface*.





Gambar 2.27. Toolbox Visual Basic

Keterangan gambar diatas sebagai berikut:

1. *Pointer* bukan merupakan suatu kontrol; gunakan icon ini ketika ingin memilih kontrol yang sudah berada pada form.
2. *PictureBox* adalah kontrol yang digunakan untuk menampilkan *image* dengan *format*: BMP, DIB (bitmap), ICO (*icon*), CUR (*cursor*), WMF (*metafile*), EMF (*enhanced metafile*), GIF, dan JPEG.
3. *Label* adalah kontrol yang digunakan untuk menampilkan teks yang tidak dapat diperbaiki oleh pemakai.
4. *TextBox* adalah kontrol yang mengandung string yang dapat diperbaiki oleh pemakai, dapat berupa satu baris tunggal, atau banyak baris.
5. *Frame* adalah kontrol yang digunakan sebagai kontainer bagi kontrol lainnya.
6. *CommandButton* merupakan kontrol hampir ditemukan pada setiap form, dan digunakan untuk membangkitkan event proses tertentu ketika pemakai melakukan klik padanya.

7. *CheckBox* digunakan untuk pilihan yang isinya bernilai *yes/no*, *true/false*.
8. *OptionButton* sering digunakan lebih dari satu sebagai pilihan terhadap beberapa option yang hanya dapat dipilih satu.
9. *ListBox* mengandung sejumlah item, dan user dapat memilih lebih dari satu (bergantung pada property *MultiSelect*).
10. *ComboBox* merupakan kombinasi dari *TextBox* dan suatu *ListBox* dimana memasukkan data dapat dilakukan dengan mengetikkan maupun pemilihan.
11. *HScrollBar* dan *VScrollBar* digunakan untuk membentuk *scrollbar* berdiri sendiri.
12. *Timer* digunakan untuk proses *background* yang diaktifkan berdasarkan interval waktu tertentu. Merupakan kontrol non-visual.
13. *DriveListBox*, *DirListBox*, dan *FileListBox* sering digunakan untuk membentuk *dialog box* yang berkaitan dengan *file*.
14. *Shape* dan *Line* digunakan untuk menampilkan bentuk seperti garis, persegi, bulatan, oval.
15. *Image* berfungsi menyerupai *image box*, tetapi tidak dapat digunakan sebagai kontainer bagi kontrol lainnya. Sesuatu yang perlu diketahui bahwa kontrol *image* menggunakan resource yang lebih kecil dibandingkan dengan *PictureBox*
16. *Data* digunakan untuk *data binding*
17. *OLE* dapat digunakan sebagai tempat bagi program eksternal seperti *Microsoft Excel*, *Word*, dll.[9]