

## **BAB 2**

### **LANDASAN TEORI**

#### ***2.1. Pattern Recognition***

Pengenalan pola (*pattern recognition*) dapat diartikan sebagai proses klasifikasi dari objek atau pola menjadi beberapa kategori atau kelas. Dan bertujuan untuk pengambilan keputusan (Theodoridis and Koutroumbas 2006, 1).

Pola adalah bentuk atau model (atau, lebih abstrak, suatu set peraturan) yang dapat dipakai untuk membuat atau untuk menghasilkan suatu atau bagian dari sesuatu, khususnya jika sesuatu yang ditimbulkan mempunyai sejenis pola dasar yang dapat ditunjukkan atau terlihat, yang mana dapat dikatakan mempertunjukkan pola.

Deteksi pola dasar disebut pengenalan pola. Untuk dapat mengenali suara dan pemilik suara (*speaker*), diperlukan suatu pengenalan pola yang dapat mengenal dan mencocokkan pola yang diinput dan pola yang telah disimpan di dalam *database*.

Riset untuk pengenalan suara otomatis sudah lama dilakukan, dimulai dari tahun 1960 yang awalnya terinspirasi oleh film fiksi yang berjudul *A Space Odyssey*. Pengenalan pola digunakan untuk mengenali objek kompleks dari bentuk sifat objek yang akan dikenali ciri-ciri dari objeknya. Pengenalan pola secara formal dapat dideskripsikan sebagai sebuah proses yang menerima pola atau sinyal berdasarkan

hasil pengukuran yang kemudian diklasifikasikan ke dalam satu atau lebih kategori atau kelas tertentu (Haykin 1999, 67)

Metode klasifikasi yang digunakan pada sistem pengenalan pola memiliki dua jenis pendekatan. Pendekatan statistik dan pendekatan struktural (atau sintaktik). Pengenalan pola statistik berdasarkan pada karakteristik statistikal dari pola-pola yang ada dengan asumsi bahwa pola-pola tersebut dihasilkan oleh sebuah sistem probabilistik. Pengenalan pola struktural berdasarkan pada hubungan struktural dari fitur dari setiap pola.

Sebuah sistem pengenalan pola terdiri dari sensor yang mengumpulkan pola yang akan diproses dan mengukur variabel dari setiap pola, *pre-processing* yang menghilangkan *noise* dalam data, mekanisme ekstraksi fitur untuk mendapatkan informasi numerik atau simbolik dari pola-pola tersebut, model pembelajaran yang mempelajari pemetaan antara fitur dan kelompok pola, metode klasifikasi yang memisah-misahkan pola-pola tersebut ke dalam kategori berdasarkan fitur dan model pembelajaran, dan *post-processing* yang mengevaluasi benar tidaknya hasil yang didapat.

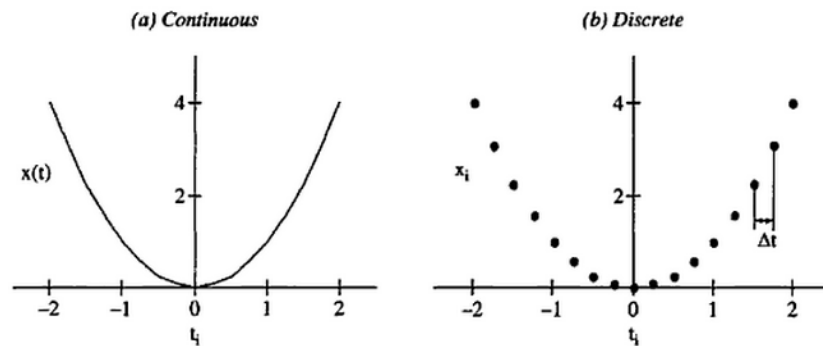
Pengenalan pola merupakan bidang dalam pembelajaran mesin dan dapat diartikan sebagai tindakan mengambil data mentah dan bertindak berdasarkan klasifikasi data. Dengan demikian, hal tersebut merupakan himpunan kaidah bagi pembelajaran yang diawasi (*supervised learning*).

## 2.2. Sinyal dan sinyal percakapan

### 2.2.1. Sinyal

Sebuah sinyal adalah variasi dari variabel seperti gelombang tekanan udara dari suara, warna dari gambar, kedalaman sebuah permukaan, temperatur dari tubuh, tegangan atau arus dari sebuah konduktor atau sistem biologis, cahaya, sinyal elektromagnetik radio, harga-harga barang, atau volume dan berat dari suatu objek. Sebuah sinyal membawa informasi mengenai satu atau lebih atribut mengenai status, komposisi, arah pergerakan dan tujuan dari sumber. Dapat dikatakan, sebuah sinyal adalah sebuah media untuk membawa informasi mengenai keadaan masa lalu, masa sekarang, dan masa yang akan datang dari suatu variabel (Vaseghi 2007, 3). Sinyal dapat diklarifikasikan menjadi beberapa jenis yaitu, sinyal waktu, sinyal nilai, sinyal random, dan sinyal non-random.

Pada umumnya variabel independen untuk sinyal satu dimensi adalah waktu. Jika variabel independennya kontinu, maka sinyal tersebut disebut sebagai sinyal waktu kontinu (*continuous-time signal*). Jika variabel independennya diskrit, maka sinyal tersebut disebut sebagai sinyal waktu diskrit (*discrete-time signal*). Sinyal waktu kontinu didefinisikan setiap waktu  $t$  dalam sebuah interval yang biasanya tidak terbatas, sedangkan sinyal waktu diskrit didefinisikan pada waktu diskrit, dan biasanya berupa urutan angka.



**Gambar 2. 1 (a) Sinyal Waktu Kontinu dan (b) Sinyal Waktu Diskrit**  
**Sumber: (Santamarina 2005)**

Sinyal waktu kontinu dengan amplitudo kontinu biasanya disebut sebagai sinyal analog. Contoh sinyal analog adalah sinyal suara. Sinyal waktu diskrit dengan amplitudo bernilai diskrit yang direpresentasikan oleh digit angka yang terbatas (*finite*), biasanya disebut sebagai sinyal digital.

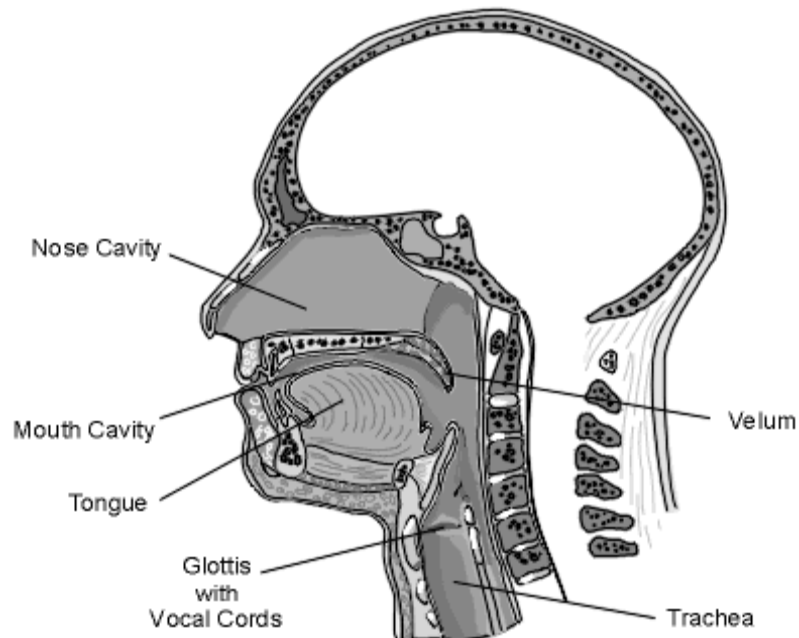
Berdasarkan jenis frekuensinya, sinyal terbagi menjadi sinyal *stationary* dan sinyal *non-stationary*. Frekuensi dalam sinyal *stationary* tidak berubah dan selalu berulang dalam waktu, sedangkan frekuensi dalam sinyal *non-stationary* berubah-ubah dalam waktu.

### 2.2.2. Sinyal percakapan

Sinyal percakapan adalah sinyal yang dihasilkan dari suara manusia sewaktu melakukan percakapan. Sinyal percakapan merupakan kombinasi kompleks dari variasi tekanan udara yang melewati pita suara dan *vocal tract*, yaitu mulut, lidah, gigi, bibir, dan langit-langit mulut. *Speech* (wicara) dihasilkan dari sebuah kerjasama antara *lungs* (paru-paru), *glottis* (dengan *vocal cords*) dan

*articulation tract* (*mouth* atau mulut dan *nose cavity*/rongga hidung). Sinyal suara terdiri dari serangkaian suara yang masing–masing menyimpan sepotong informasi. Berdasarkan cara menghasilkannya, suara dapat dibagi menjadi *voiced* dan *unvoiced*. *Voiced sounds* atau suara ucapan dihasilkan dari getaran pita suara, sedangkan *unvoiced sounds* dihasilkan dari gesekan antara udara dengan *vocal tract*.

Sinyal percakapan memiliki beberapa karakteristik, seperti *pitch* dan intensitas suara yang berguna dalam melakukan analisis sinyal suara. *Pitch* adalah frekuensi dari sinyal atau yang sering disebut intonasi. Intensitas suara adalah tingkat kekuatan suara.



**Gambar 2. 2 Produksi Suara Manusia**  
**Sumber: (Holmes and Holmes 2001, 18)**

*Impuls* tekanan pada umumnya disebut sebagai *pitch impulses* dan frekuensi sinyal tekanan adalah *pitch frequency* atau *fundamental frequency*. Sederet *impuls* (fungsi tekanan suara) dihasilkan oleh pita suara untuk sebuah suara. Hal ini merupakan bagian dari sinyal *voice* (suara) yang mendefinisikan *speech melody* (melodi wicara). Ketika berbicara dengan *pitch* yang stabil, suara sinyal wicara *monotonous* tetapi dalam kasus normal sebuah perubahan permanen pada frekuensi terjadi. *Impuls pitch* merangsang udara dalam mulut dan untuk suara tertentu (*nasals*) juga merangsang *nasal cavity* (rongga hidung). Ketika rongga beresonansi, timbul radiasi sebuah gelombang suara yang merupakan sinyal percakapan. Kedua rongga beraksi sebagai *resonators* dengan karakteristik frekuensi resonansi masing-masing yang disebut *formant frequencies*, sehingga *formant* merupakan variasi resonansi yang dihasilkan oleh *vocal tract*. Pada saat rongga mulut mengalami perubahan besar, dihasilkan beragam pola ucapan suara yang berbeda. Di dalam kasus *unvoiced sounds*, keluaran pada *vocal tract* lebih menyerupai *noise* atau derau.

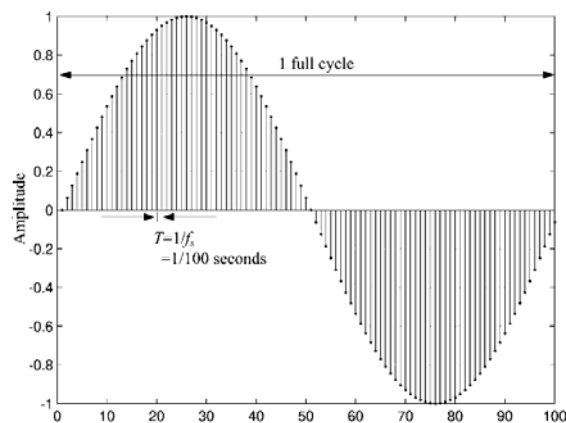
### 2.2.3. Bit Rate

*Bit rate* menyatakan jumlah *bit* yang digunakan setiap unit waktu dalam *file suara*. Sebuah suara digital terdiri dari *word* yang berisi 0 atau 1. Pada *word 2 bit*, terdapat 4 kemungkinan nilai, yaitu 00, 01, 10, dan 11. Pada *word 3 bit*, terdapat 8 kemungkinan nilai, yaitu 000, 111, 001, 010, 011, 100, 101, dan 110. Pada *word n bit*, terdapat  $2^n$  kemungkinan nilai. Jika *word 8 bit* digunakan untuk menyatakan volume, maka terdapat 256 kemungkinan nilai volume,

berkisar dari 0 sampai dengan 255. Semakin tinggi *bit rate*, semakin akurat resolusi dari volume *file* suara tersebut.

#### 2.2.4. Sampling Rate

*Sampling rate* (biasa disebut juga *sampling frequency*) menyatakan jumlah *sample* per detik yang diambil dari sinyal kontinu untuk membuat sinyal diskrit. Untuk sinyal *time-domain*, *sampling rate* dapat diukur dalam *hertz* (Hz). Kebalikan dari *sampling rate* adalah *sampling period* yang menyatakan selang waktu di antara setiap *sample* (Antoniou 2006, 3) .



**Gambar 2. 3 Sampling Sinyal**  
**Sumber: (Park 2009, 21)**

Semakin tinggi *sampling rate*, maka semakin akurat resolusi *file* suara tersebut. Sebagai contoh, suara 16 bit dan 44,1 KHz bermakna suara tersebut di-*sampling* 44.100 kali per detik dan diukur dengan akurasi 16 bit.

Sinyal suara yang hanya berisi suara manusia (*speech signal*) dapat di-*sampling* pada nilai yang jauh lebih rendah. Dalam kebanyakan kasus, hampir

semua energi dalam suara tersimpan dalam rentang 0-4000 Hz sehingga *sampling* cukup dilakukan pada 8000 Hz (Vaseghi 2007, 163). Hal ini didasarkan pada teori *Nyquist-Shannon* yang menyebutkan bahwa untuk mencegah hilangnya informasi dalam sebuah konversi sinyal kontinu ke diskrit, *sampling* minimal harus dua kali lebih besar dari sinyal asli (Shannon 1949).

#### **2.2.5. Metode Signal Processing**

Metode pemrosesan sinyal menyediakan sebuah variasi alat untuk memodelkan, menganalisis, membuat, mensintetiskan, dan mengenali sinyal. Metode pemrosesan sinyal telah berevolusi dalam kompleksitas algoritma yang bertujuan untuk mengoptimalkan utilisasi dari informasi yang tersedia untuk mencapai performa terbaik. Secara umum kebutuhan komputasi meningkat terkadang secara eksponensial, seiring dengan kompleksitas algoritma. Tetapi biaya implementasi dari metode pemrosesan sinyal terbaru telah cukup memadai dengan *trend* yang konsisten beberapa tahun terakhir. Dengan peningkatan berkelanjutan dalam performa, ditambah dengan pengurangan biaya secara bersamaan, dari perangkat keras pemrosesan sinyal, pemrosesan sinyal digital (*Digital Signal Processing*) dapat dikategorikan menjadi empat kategori umum, yaitu pemrosesan sinyal berbasis transformasi, pemrosesan sinyal berbasis model, pemrosesan sinyal statistik bayesian, dan jaringan saraf tiruan.



### 2.3. Ekstraksi Fitur

Setelah pengambilan sampel, sinyal percakapan diketahui masih relatif berulang, sehingga *speech coders/decoders (codecs)* didesain untuk mengekstraksi representasi yang kompak (*compact representation*) yang cukup untuk rekonstruksi dari sinyal dengan kualitas tinggi. Dalam sistem pengenalan suara, representasi yang kompak ini juga diperlukan. Algoritma pemrosesan sinyal digunakan untuk mengekstrak vektor fitur, mempertahankan informasi yang diperlukan untuk mengenali percakapan dan membuang sisanya. Langkah ini sering disebut sebagai Ekstraksi Fitur (*Feature Extraction*). (Thiran, Marqués and Bourlard 2010, 29). Fitur dari sebuah sistem pengenalan pola yang baik harus bersifat alamiah, dapat diukur dengan mudah, tidak berubah dari waktu ke waktu atau terpengaruh oleh kondisi kesehatan pengguna, tidak terpengaruh oleh *noise*, dan tidak dapat ditiru oleh orang lain.

Pengenalan suara pada dasarnya bergantung pada pengenalan rangkaian fenomena yang bergantung pada bentuk suara. Pada umumnya pendekatan umum untuk ekstraksi fitur adalah mengekstraksi representasi halus dari kepadatan kekuatan *spectrum* sinyal (karakteristik dari respon filter frekuensi), biasanya diperkirakan dari analisis *frame* yang sepanjang 20-30 ms. Beberapa alat pemrosesan sinyal sering digunakan pada implementasi ekstraksi fitur. Alat tersebut termasuk transformasi fourier waktu singkat (*short-time fourier transform*) yang dapat digunakan untuk memperoleh kekuatan dan fase *spectrum* dari analisa *frame* singkat. Alat lainnya yang biasa digunakan adalah *Linear Predictive Coding*

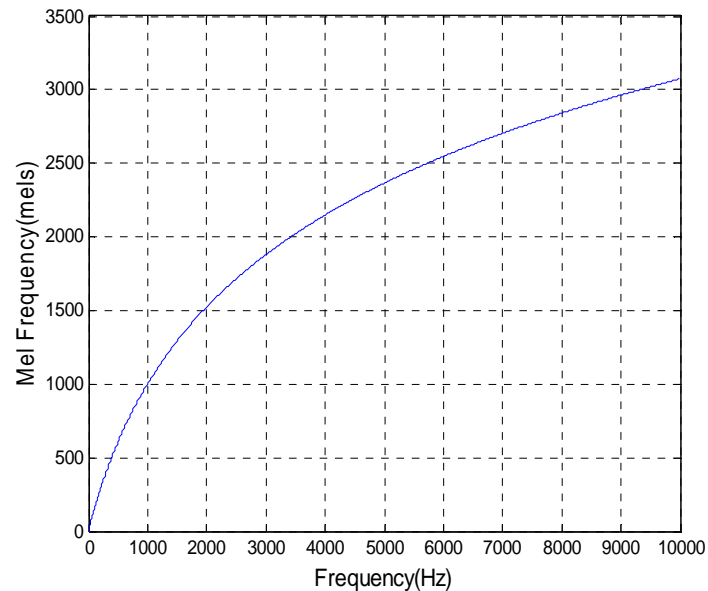
(LPC) yang mana kita modelkan daerah vokal dengan filter *all-pole*. Alat lainnya adalah *cepstrum*, yang dihitung sebagai *inverse short time fourier transform* dari logaritma pangkat dari *spectrum*. Elemen urutan rendah dari *cepstrum vector* merupakan pendekatan yang baik dari bagian filter yang baik dari sebuah model.

Perkembangan pengetahuan kita akan sistem pendengaran manusia telah membuat beberapa model dari resolusi frekuensi nonlinear, dan kehalusan dari pendengaran sering digunakan. Salah satunya adalah *Mel-Frequency Cepstrum Coefficients (MFCCs)*.

### 2.3.1. *Mel-Frequency Cepstrum Coefficients (MFCCs)*

*Mel-frequency Cepstrum coefficient* merupakan metode yang paling dikenal dan paling banyak digunakan pada bidang ekstraksi fitur suara. *MFC (Mel-Frequency Cepstrum)* memetakan komponen frekuensi dengan menggunakan skala *Mel* yang dimodelkan berdasarkan persepsi suara dari kuping manusia. *Mel-frequency cepstrum* mewakili *spectrum* jangka pendek dari suara menggunakan *linear cosine transform* dari *log* dari sebuah *spectrum* skala *Mel*. Rumus dari skala *Mel* adalah :

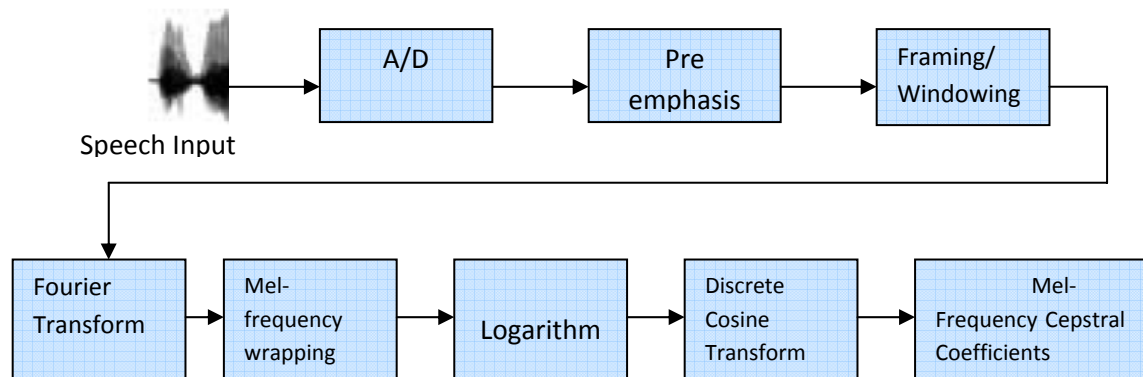
$$M = 2595 \log_{10} \left( \frac{f}{700} + 1 \right) \quad (2.1)$$



**Gambar 2. 4 Gambar Skala Mel**

**Sumber: (Vaseghi 2007, 518)**

*MFCC* merupakan parameter domain frekuensi yang lebih konsisten dan akurat daripada fitur domain waktu. Langkah – langkah untuk menghitung *MFCC*: *Fast Fourier Transform*, memfilter dengan filter *Mel* dan *cosine transform* dari vektor *log* energi. *MFCC* dimulai dihitung dengan mengambil *windowed frame* dari sinyal suara, lalu menggunakan *Fast Fourier Transform (FFT)* untuk memperoleh parameter tertentu dan kemudian diubah menjadi skala *Mel* untuk meperoleh vektor fitur yang mewakili amplitudo terkompres secara logaritmik dan informasi frekuensi yang sederhana. Kemudian dihitung dengan mengaplikasikan *discrete cosine transform* kepada *log* dari *Mel-filter bank*. Hasilnya adalah fitur yang menggambarkan bentuk spektral dari sinyal (Muda, Begam and Elamvazuthi 2010).



**Gambar 2. 5 Diagram dari Mel-Frequency Cepstral Coefficient**

**Sumber: (Muda, Begam and Elamvazuthi 2010)**

### 2.3.2. *Pre-emphasis*

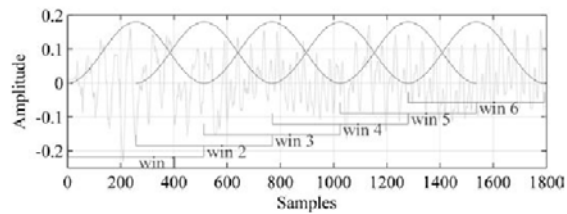
*Pre-emphasis* memproses pengiriman sinyal melalui filter yang menekankan terhadap frekuensi tinggi. Proses *Pre-emphasis* akan meningkatkan energi sinyal pada frekuensi tinggi.

$$Y[n] = X[n] - aX[n-1] \quad (2.2)$$

Jika dimisalkan  $a=0.95$ , maka 95% dari suatu sampel diduga berasal dari sampel sebelumnya.

### 2.3.3. *Framing*

Proses untuk memisahkan sampel yang telah diperoleh dari *analog to digital conversion (ADC)* menjadi beberapa *frame* dengan panjang sekitar 20-30 ms. Sinyal suara dibagi menjadi beberapa *frame* sebanyak N sampel. *Frame* yang bersebelahan dipisahkan sebesar M ( $M < N$ ) nilai normal yang biasa digunakan adalah  $M=128$  dan  $N=256$



**Gambar 2. 6 Framing Signal**

Sumber: (Park 2009, 69)

#### 2.3.4. Windowing

Langkah berikutnya adalah melakukan proses *window* pada setiap bagian sinyal yang telah dibuat pada proses *framing*. Hal ini dilakukan untuk meminimalkan diskontinuitas pada bagian awal dan akhir sinyal dan mengintegrasikan garis-garis frekuensi terdekat.

Jika didefinisikan sebuah *window*  $w(n)$  dan sinyal tiap bagian adalah  $x(n)$  maka sinyal hasil proses *windowing* ini adalah sebagai berikut.

$$Y(n) = X(n) \cdot W(n), \quad 0 \leq n \leq N-1 \quad (2.3)$$

Adapun  $N$  adalah jumlah sampel dari setiap *frame*. Model *window* yang paling sering digunakan adalah *Hamming Window* yang direpresentasikan dengan persamaan berikut.

$$W(n) = 0.54 - 0.46 \cos\left(\frac{2\pi n}{N-1}\right) \quad 0 \leq n \leq N-1 \quad (2.4)$$

Hasil yang didapatkan lalu dikalikan dengan sampel.

### 2.3.5. Fast Fourier Transform

*Fast Fourier Transform (FFT)* yang ditemukan tahun 1965 merupakan pengembangan dari *Fourier Transform (FT)*. Penemu *FT* adalah *J. Fourier* pada tahun 1822. *FT* membagi sebuah sinyal menjadi frekuensi yang berbeda-beda dalam fungsi eksponensial yang kompleks.

$$X(f) = \int_{-\infty}^{\infty} x(t) \cdot e^{-2\pi i f t} dt \quad (2.5)$$

Adapun  $t$  menyatakan waktu,  $f$  menyatakan frekuensi,  $X(f)$  menyatakan sinyal dalam domain frekuensi, dan  $x(t)$  menyatakan sinyal dalam domain waktu.

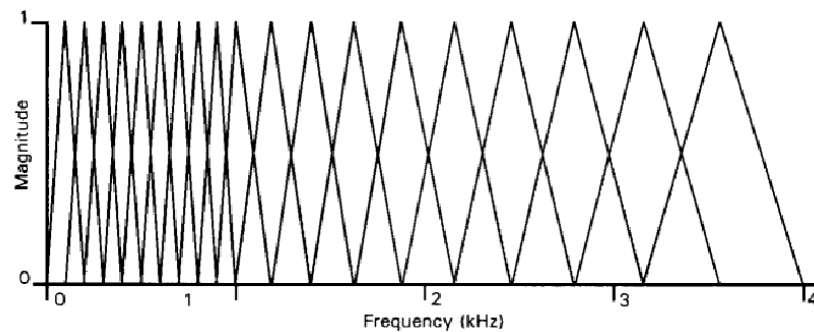
Untuk pemrosesan sinyal diskrit, sebuah algoritma baru yang disebut *Discrete Fourier Transform (DFT)* diciptakan. *DFT* memiliki rumus sebagai berikut.

$$X_k = \sum_{n=0}^{N-1} x_n e^{\frac{2\pi i}{N} n k} \quad k = 0, \dots, N-1 \quad (2.6)$$

Adapun  $N$  menyatakan jumlah sampel. Persamaan (2.5) memiliki kompleksitas algoritma  $O(N^2)$ . *FFT* membagi sampel  $N$  menjadi dua buah  $N_1$  dan  $N_2$  secara rekursif bersama perkalian dengan hasil yang didapatkan lalu dikalikan dengan sampel  $e^{\frac{2\pi i}{N} n k}$ . Hal ini membuat *FFT* hanya memiliki kompleksitas  $O(N \log N)$ . Dalam pemrosesan sinyal suara, *FFT* akan mengubah sinyal suara dalam domain waktu menjadi domain frekuensi.

### 2.3.6. Mel Filter Bank Processing

Jangkauan frekuensi dalam *spectrum FFT* sangat lebar. Sinyal suara juga tidak mengikuti skala linear. Maka filter kemiringan menurut skala *Mel* yang ditunjukkan pada 2.7 lalu diaplikasikan.



**Gambar 2. 7 Mel scale filter bank**  
**Sumber: (Holmes and Holmes 2001, 161)**

Gambar di atas menunjukkan rangkaian filter segitiga yang digunakan untuk menghitung jumlah berat dari filter komponen spektral sehingga hasil dari proses mendekati skala *Mel*. Setiap ketinggian filter *frequency response* adalah bentuk segitiga dan setara pada penggabungan pusat frekuensi dan penurunan secara linear menuju nol pada pusat frekuensi dari dua filter besebelahan. Setelah itu untuk menghitung *Mel* dari sebuah frekuensi  $f$  dalam Hz digunakan persamaan berikut.

$$m = 1127 \ln\left(\frac{f}{700} + 1\right) \quad (2.7)$$

### 2.3.7. Discrete Cosine Transform

Setelah melalui *Mel filter*. *Log Mel spectrum* perlu untuk diubah menjadi domain waktu menggunakan *Discrete Cosine Transform* (DCT). Hasil dari konversi inilah yang disebut sebagai *Mel Frequency Cepstrum Coefficient*. Kumpulan dari koefisien ini disebut sebagai vektor akustik (*acoustic vectors*) yang akan digunakan sebagai nilai yang mewakili sinyal suara.

$$X(k) = 2 \sum_{m=0}^{N-1-k} x(m) \cos \frac{\pi k (2m+1)}{2N} \quad 0 \leq k \leq N-1 \quad (2.8)$$

## 2.4. Hidden Markov Model

Model *Markov* Tersembunyi atau lebih dikenal sebagai *Hidden Markov Model* (HMM) adalah sebuah model statistik dari sebuah sistem yang diasumsikan sebuah *Markov Process* dengan parameter yang tak diketahui, dan tantangannya adalah menentukan parameter-parameter tersembunyi (*hidden*) dari parameter-parameter yang dapat diamati. Parameter-parameter yang ditentukan kemudian dapat digunakan untuk analisis yang lebih jauh, misalnya untuk aplikasi *Pattern Recognition* (Rabiner 1989). Sebuah HMM dapat dianggap sebagai sebuah *Bayesian Network* dinamis yang paling sederhana.

Pada model *Markov* umum, *state*-nya langsung dapat diamati, oleh karena itu probabilitas transisi *state* menjadi satu-satunya parameter. Di dalam Model *Markov* yang tersembunyi (*Hidden*) , *state*-nya tidak dapat diamati secara langsung, akan tetapi yang dapat diamati adalah variabel-variabel yang terpengaruh oleh *state*. Setiap *state* memiliki distribusi probabilitas atas *token-token output* yang mungkin



muncul. Oleh karena itu rangkaian *token* yang dihasilkan oleh *HMM* untuk memberikan informasi tentang sekuens *state-state*.

Dalam *HMM* ada dua buah asumsi independen pertama variabel tersembunyi  $t^{th}$  dengan variabel tersembunyi  $(t-1)^{th}$  adalah independen dari variabel sebelumnya atau  $P(Q_t | Q_{t-1}, Q_{t-2}, \dots, Q_1, O_1) = P(Q_t | Q_{t-1})$ , lalu kedua observasi  $t^{th}$  dengan variabel tersembunyi  $t^{th}$  adalah independen dengan variabel yang lainnya atau  $P(Q_t | Q_t, O_t, Q_{t-1}, Q_{t-2}, \dots, Q_{t+1}, Q_{t+2}, Q_t, Q_{t-1}, Q_{t-2}, \dots, Q_1, O_1) = P(Q_t | Q_t)$

Dari persamaan diatas.  $Q_t$  adalah variabel diskrit acak dengan  $N$  nilai kemungkinan. Kemudian dimisalkan  $P(Q_t | Q_{t-1})$  sebagai matriks transisi  $A = \{a_{i,j}\} = P(Q_t = j | Q_{t-1} = i)$ . Kemudian  $t = 1$  dideskripsikan dengan distribusi awal  $\pi_i = P(Q_1 = i)$ . Kemudian dapat dikatakan situasi berada pada *state*  $j$  pada waktu  $t$  jika  $Q_t = j$ .

Rangkaian observasi dideskripsikan sebagai  $O = (O_1 = o_1, \dots, O_T = o_T)$ .

Probabilitas dari vektor observasi pada waktu  $i$  dan  $j$  dideskripsikan sebagai

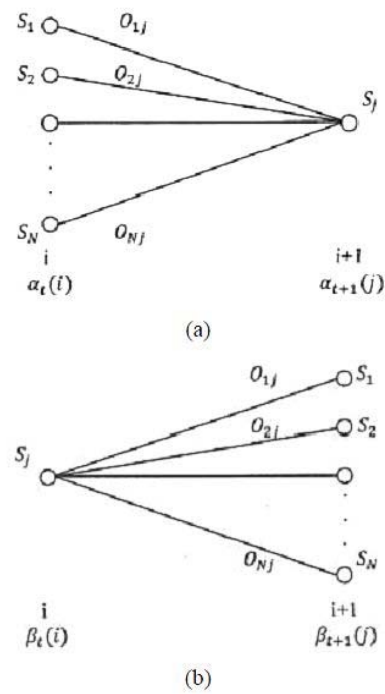
$b_j(o_t) = p(O_t = o_t | Q_t = j)$ . Distribusi lengkap dari parameter untuk semua observasi dapat dideskripsikan sebagai  $B = \{b_j(\cdot)\}$  [3].

Kita dapat mendeskripsikan himpunan lengkap dari parameter *HMM* dengan model sebagai berikut.

$$\lambda = (A, B, \pi) \quad (2.9)$$

Setiap data dari suara yang diambil akan dimodelkan dengan *forward-backward algorithm* atau yang lebih dikenal dengan *Baum-Welch algorithm*. *Forward variable* didefinisikan sebagai observasi sebagian dinotasikan dengan  $O_1, O_2, \dots, O_t$  dan *state*  $S_i$  pada waktu  $t$ , dengan  $\lambda$  dan  $\alpha$  sebagai  $\alpha(t)$ . *Backward variable* didefinisikan sebagai observasi sebagian dari *state probability*  $t+1$  sampai *state* sekarang, di mana *state*  $S_i$  pada waktu  $t$ , dengan  $\lambda$  dan  $\alpha$  sebagai  $\beta(t)$ . Rangkaian *state probability* ini dapat dinotasikan sebagai berikut.

$$P(O|\lambda) = \sum_{t=1}^N \alpha_t(t) \beta_t(t) = \sum_{t=1}^N \alpha_t(t) \quad (2.10)$$

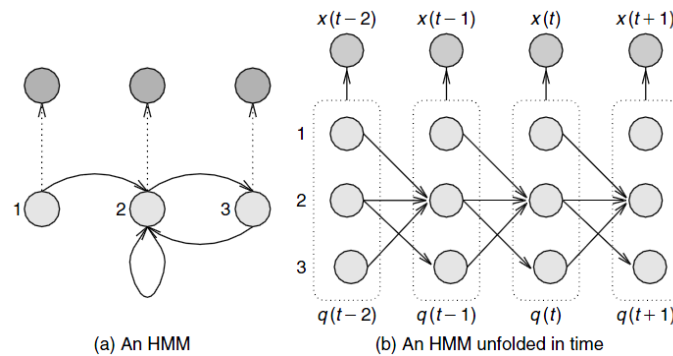


**Gambar 2. 8 ilustrasi Baum-Welch algorithm. (a) Forward variable  $\alpha_{t+1}$  (b) Backward variable  $\beta_{t+1}$**   
**Sumber: (Rabiner 1989, 262)**

Probabilitas pada state  $S_i$  pada waktu  $t$  dengan observasi rangkaian  $O$  dan

HMM  $\lambda$  (pour and Farokhi 2009) dapat dinotasikan sebagai berikut.

$$\gamma_t = \frac{\alpha_t(O) \beta_t(O)}{P(O|\lambda)} \quad (2.11)$$



**Gambar 2. 9** dua jenis penggambaran *HMM*

**Sumber:** (Thiran, Marqués and Bourlard 2010, 29)

Dalam *Hidden Markov Model*, terdapat tiga masalah dasar, yaitu

a. Masalah Evaluasi (*Evaluation Problem*)

Masalah Evaluasi adalah masalah di mana probabilitas  $P(O|\lambda)$  dari serangkaian pengamatan ( $O=o_1, o_2, \dots, o_T$ ) yang dibangun oleh model *HMM* ( $\lambda$ ) belum diketahui. Probabilitas tersebut dapat dihitung dengan menggunakan algoritma *forward* (*forward algorithm*)

b. Masalah Pengkodean (*Decoding Problem*)

Masalah pengkodean adalah masalah di mana bagian dari rangkaian *state* dalam model *HMM* ( $\lambda$ ) yang menghasilkan pengamatan ( $O=o_1, o_2, \dots, o_T$ ) belum diketahui.

c. Masalah pembelajaran (*Learning Problem*)

Masalah pembelajaran adalah masalah di mana parameter model harus disesuaikan berdasarkan model *HMM* ( $\lambda$ ) dan serangkaian pengamatan  $O=o_1, o_2, \dots, o_T$  agar nilai probabilitas  $P(O|\lambda)$  maksimal. Untuk mendapatkan model *HMM* tersebut, digunakan algoritma *k-means* (*k-means algorithm*).

### 2.4.1. Algoritma *Forward*

Untuk menghitung probabilitas dari serangkaian pengamatan ( $O=o_1, o_2, \dots, o_T$ ) yang dibangun oleh model  $HMM \lambda = (A, B, \pi)$ , diperlukan algoritma *forward*. Berikut ini adalah langkah-langkah dalam algoritma *forward*.

#### 1. Inisialisasi

$$\alpha_1(i) = \pi_i b_i(o_1), \quad 1 \leq i \leq N \quad (2.12)$$

di mana :

$\alpha$  adalah variabel *forward*

$\pi_1$  adalah distribusi *state* awal

$b_i$  adalah probabilitas *state* ke  $i$

$o_1$  adalah pengamatan ke 1

$N$  adalah jumlah *state*

#### 2. Induksi

$$\alpha_{t+1}(j) = \left[ \sum_{i=1}^N \alpha_t(i) a_{ij} \right] b_j(o_{t+1}), \quad 1 \leq t \leq T-1, 1 \leq j \leq N \quad (2.13)$$

di mana :

$t$  adalah waktu

$T$  adalah jumlah pengamatan

$a_{ij}$  adalah probabilitas transisi  $i$  ke  $j$

### 3. Terminasi

$$P(O|\lambda) = \sum_{t=1}^T \alpha_T(t) \quad (2.14)$$

di mana :

$P(O|\lambda)$  adalah probabilitas dari pengamatan  $O$  yang dibangun oleh model

$HMM(\lambda)$

#### 2.4.2. Algoritma *k-means*

Algoritma *k-means* berguna untuk mendapatkan model  $HMM \lambda=(A, B, \pi)$  agar mendapatkan nilai probabilitas  $P(O|\lambda)$  yang maksimal. Dalam menjalani proses *learning* pada algoritma *k-means* ini, dibutuhkan serangkaian data dan sejumlah pengamatan. Berikut ini adalah algoritma *k-means* yang terdiri dari 6 langkah.

- a. Pilihlah *means* inisial secara acak dari data yang akan diolah

$$means(k) = data(rand(i), rand(j)) \quad 1 \leq i \leq w, 1 \leq j \leq T, 1 \leq k \leq N \quad (2.15)$$

di mana :

$data(i,j)$  adalah data pada matriks  $i, j$

$rand()$  adalah fungsi acak

$w$  adalah jumlah baris data

$T$  adalah jumlah kolom data

$N$  adalah jumlah *state*

- b. Berikan nilai klasifikasi pada setiap unit dari data berdasarkan *euclidean distance* terkecil dari *means*.

$$Euclidean Distance = \sqrt{(x_t - means(k))^2} \quad 1 \leq t \leq w, 1 \leq j \leq T, 1 \leq k \leq T \quad (2.16)$$

- c. Hitung *means* baru berdasarkan klasifikasi baru.

$$MeansBaru(k) = \frac{\text{jumlah nilai data}(k)}{\text{jumlah data}(k)}, 1 \leq k \leq N \quad (2.17)$$

- d. Hitung error antara *means* dengan *means* baru

$$Error = \sqrt{\frac{\sum_{k=1}^N (Meansbaru(k) - Means(k))^2}{N}} \quad (2.18)$$

- e. Jika *error* lebih besar dari batas yang ditentukan, ulangi langkah b sampai e
- f. Hitung nilai-nilai pada model *HMM*  $\lambda = (A, B, \pi)$ , dengan menggunakan klasifikasi akhir.

## 2.5. Speaker Recognition

*Speaker recognition* atau yang juga dikenal dengan *voice recognition* adalah sistem yang dapat mengenali identitas seseorang dari suaranya (Holmes and Holmes 2001, 219). Sistem ini mengekstrak fitur dari suara, memodelkannya, dan

menggunakan model tersebut untuk membedakan seseorang berdasarkan suaranya. *Speaker recognition* sering disamakan dengan *speech recognition*, padahal keduanya memiliki definisi yang berbeda. *Speaker recognition* mengenali siapa yang berbicara sementara *speech recognition* mengenali apa yang diucapkan.

*Speaker recognition* terdiri dari dua tahap yaitu tahap pelatihan dan tahap pengenalan (evaluasi). Pada tahap pelatihan, suara pembicara akan direkam dan fitur di dalamnya akan diekstrak untuk mendapatkan *voice print*, *template*, atau model suara. Pada tahap evaluasi, suara pembicara dicocokkan dengan *template* atau model.

Terdapat beberapa faktor yang harus diperhatikan dalam evaluasi sistem *speaker recognition* :

- Kualitas suara (jumlah *channel* dan karakteristik mikrofon, tingkat *noise*, variasi suara pada tahap pelatihan dan evaluasi)
- Metode input (*text-dependent* atau *text-independent*)
- Lama suara (durasi dan jumlah sesi suara pada tahap pelatihan dan evaluasi)
- Jumlah subjek dalam tahap pelatihan

Berdasarkan fungsinya, *speaker recognition* terbagi menjadi *speaker verification* (atau *voice authentication*) dan *speaker identification* (Holmes and Holmes 2001, 219).

### **2.5.1. *Speaker Identification***



Sebuah sistem *speaker identification* bertujuan untuk mengidentifikasi siapa yang sedang berbicara dari sekumpulan set suara yang telah dikenali dan tidak terdapat klaim identitas dari pengguna. Dalam *speaker identification*, suara pengguna dicocokkan dengan semua *template* yang ada 1 banding N *template*. Ruang lingkup *speaker identification* terbagi dua yaitu membedakan setiap *speaker* dalam sebuah percakapan dan mengidentifikasi suara seseorang berdasarkan data yang telah dimasukkan sebelumnya di mana suara orang tersebut termasuk di dalamnya. *Speaker identification* dapat digunakan dalam investigasi polisi seperti pencarian suara seorang tersangka pada *database* pelaku kriminal untuk mengenali identitas tersangka.

### 2.5.2. *Speaker Verification*

Sebuah sistem *speaker verification* bertujuan untuk mengesahkan identitas seseorang berdasarkan suaranya. Ketika pengguna memberikan *input* suaranya, ia juga memasukkan identitas yang diklaimnya. Dalam *speaker verification*, suara pengguna hanya dicocokkan dengan 1 *template* yang diklaim ketika proses pemasukan 1 banding 1. *Speaker verification* biasa digunakan dalam aplikasi yang membutuhkan akses keamanan seperti aplikasi sekuritas yang menggunakan *input* suara sebagai pengganti *password* dan *PIN* (*Personal Identification Number*).

Dalam *speaker verification* dikenal dua jenis kesalahan yaitu *false rejection* di mana pengguna yang mengklaim sebagai dirinya sendiri ditolak oleh

sistem dan *false acceptance* di mana peniru yang mengklaim sebagai pengguna lain diterima oleh sistem.

## **2.6. Interaksi Manusia dan Komputer**

Manusia adalah makhluk yang senang berinteraksi dengan manusia lain. Oleh karena itulah maka penggunaan komputer dibuat sesederhana mungkin sehingga manusia dapat berinteraksi dengan komputer semudah berinteraksi dengan manusia lain. Proses penyederhanaan ini dikenal dengan *interactive user interface*.

### **2.6.1. Program Interaktif**

Ada lima kriteria yang harus dimiliki oleh suatu program sehingga dapat berinteraksi dengan baik dan bersifat *user friendly* (Shneiderman, 2005, p16). Lima kriteria tersebut adalah sebagai berikut.

1. Memiliki waktu belajar yang relatif singkat.
2. Mampu memberikan informasi yang diperlukan dengan cepat.
3. Mudah untuk dioperasikan oleh *user*.
4. Kemudahan untuk mengingat program tersebut walaupun telah lama tidak mengoperasikannya.
5. Kepuasan pribadi.

### **2.6.2. Delapan Aturan Emas**

Menurut Shneiderman (2005, p74-75) terdapat delapan aturan dalam merancang sistem interaksi manusia dan komputer yang baik yaitu sebagai berikut.

1. Bertahan untuk konsistensi.
2. Memperbolehkan *user* untuk memakai *shortcut*.
3. Memberikan umpan balik yang informatif.
4. Pengorganisasian yang baik sehingga *user* mengetahui kapan awal dan akhir dari suatu *action*.
5. Pengguna mampu mengetahui dan memperbaiki kesalahan dengan mudah.
6. Dapat dilakukan pembalikan *action*.
7. *User* mampu aktif dalam mengambil langkah selanjutnya bukan hanya merespon pesan yang muncul.
8. Mengurangi beban ingatan jangka pendek bagi pemakai sehingga perancangannya harus sederhana.

## **2.7. Waterfall Model**

Dalam membuat suatu program aplikasi terdapat beberapa paradigma atau model proses, satu diantaranya adalah *Waterfall Model* atau yang biasa dikenal dengan *Classic Life Cycle*. Model ini mendukung pendekatan yang berurutan dan

sistematis terhadap pengembangan program aplikasi yang dimulai dengan spesifikasi konsumen, dan perkembangan melalui tahap perencanaan, pemodelan, pembuatan dan pengaplikasian dalam dukungan aplikasi yang lengkap. Model ini disebut dengan *waterfall* karena tahap demi tahap yang dilalui harus menunggu selesainya tahap sebelumnya dan berjalan berurutan (Pressman 2005, 79).

Terdapat 6 tahapan dalam model *Waterfall*. Berikut adalah penjelasan dari tahap-tahap yang dilakukan di dalam model ini.

- Analisis dan perancangan sistem

Permodelan ini diawali dengan mencari kebutuhan dari keseluruhan sistem yang akan diaplikasikan ke dalam program aplikasi. Hal ini sangat penting, mengingat program aplikasi harus dapat berinteraksi dengan elemen-elemen yang lain seperti *hardware*, *database*, dsb. Tahap ini sering disebut dengan *Project Definition*.

- Analisis kebutuhan

Proses pengumpulan elemen sistem ditingkatkan dan dipusatkan secara khusus untuk mengerti karakteristik dari program yang akan dibuat. Kebutuhan maupun sistem harus dibicarakan bersama dengan pelanggan.

- Perancangan

Proses perancangan menerjemahkan kebutuhan elemen sistem yang direpresentasikan ke dalam sebuah *blueprint* yang dapat diperkirakan kualitasnya sebelum dilakukan pengkodean.

- Pengkodean

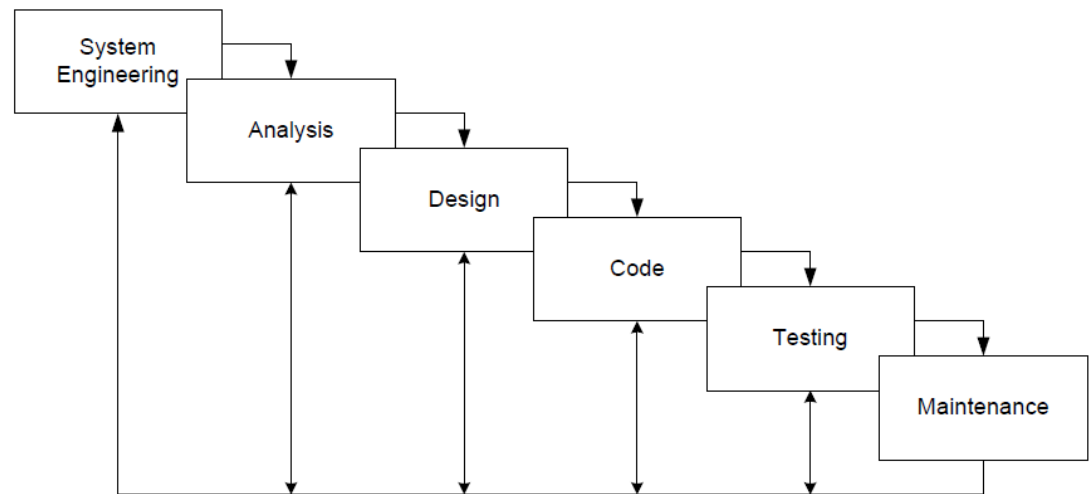
Untuk dapat dimengerti oleh komputer, maka desain tadi harus diubah bentuknya menjadi bentuk yang dapat dimengerti oleh komputer, yaitu ke dalam bahasa pemrograman melalui proses pengkodean. Tahap ini merupakan implementasi dari tahap desain yang secara teknis nantinya dikerjakan oleh *programmer*.

- Pengujian

Pengujian dilakukan untuk memastikan hasilnya sesuai dengan kebutuhan yang sudah didefinisikan sebelumnya.

- Pemeliharaan

Pemeliharaan suatu program diperlukan karena program yang dibuat tidak selamanya hanya seperti itu. Pengembangan merupakan salah satu bagian dari pemeliharaan. Pengembangan terjadi untuk memperbaiki *error* kecil yang tidak ditemukan sebelumnya, atau ada penambahan fitur-fitur yang belum ada pada program tersebut.



**Gambar 2. 10 Model waterfall**  
**Sumber: (Pressman 2005)**

Model ini menjadi terkenal karena pengaplikasian yang mudah, dan ketika semua kebutuhan sistem dapat didefinisikan secara utuh, eksplisit, dan benar di awal proyek, maka pembuatan program dapat berjalan dengan baik dan tanpa masalah. Tetapi karena model ini melakukan pendekatan secara berurutan, maka ketika suatu tahap terhambat, tahap selanjutnya tidak dapat dikerjakan dengan baik dan hal itu menjadi salah satu kekurangan dari model ini. Kesimpulannya adalah ketika suatu proyek skalanya sedang mengarah kecil cocok menggunakan model ini, akan tetapi bila mengerjakan proyek yang besar, tampaknya akan kesulitan jika menggunakan model *waterfall* ini.

## 2.8. C#

*C#* adalah sebuah bahasa pemrograman yang berorientasi pada objek yang dikembangkan oleh *Microsoft* dan menjadi bahasa pemrograman utama dari *Microsoft* yang mendukung *.Net programming* melalui *Visual Studio*. *C#* menekankan fitur-fitur yang cepat, dengan inovasi terbaru. Mudah dan efisien digunakan untuk menulis program untuk lingkungan komputasi perusahaan.

*C#* dibuat oleh *Microsoft* pada akhir tahun 1990an dan merupakan bagian dari strategi *.Net* dari *Microsoft*. *C#* pertama kali dikeluarkan pada pertengahan tahun 2000. Perancang utama dari *C#* adalah *Anders Hejlsberg*. *Hejlsberg* merupakan seorang pakar bahasa pemrograman terkemuka, dengan beberapa pencapaian yang luar biasa. Salah satunya adalah pengembangan *Turbo Pascal* yang menjadi standar untuk semua kompiler masa kini.



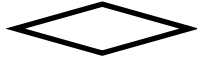



*C#* sangat berhubungan erat dengan *C*, *C++* dan *Java*. Hal ini dikarenakan mereka bertiga merupakan bahasa pemrograman yang paling banyak dipakai di dunia. Terlebih pada saat *C#* diciptakan, hampir semua programmer profesional mengetahui *C*, *C++* dan *Java*. Dengan membangun *C#* pada fondasi yang solid maka *C#* menawarkan kemudahan migrasi dari ketiga bahasa tersebut. (Schildt 2010).

## **2.9. Flow Chart**

*Flow chart* juga dikenal sebagai *flow diagram* atau *work flow*. *Flow chart* digunakan untuk menunjukkan informasi yang saling berhubungan (kejadian, langkah-langkah dalam sebuah proses, aliran uang atau data, dll). *Flow chart* juga

sering digunakan untuk membantu memecahkan masalah. Juga dapat digunakan untuk mendefinisikan dan melihat efek dari alur kerja (Fowler and Stanwick 2004, 428).

Simbol-simbol utama yang digunakan dalam *Flow chart* adalah

1.  berupa proses atau aktivitas, fungsi, tugas
2.  untuk merubah proses, start, stop, pause, interrupt
3.  untuk menunjukan pilihan. Berupa pertanyaan atau penentuan sebuah keputusan.
4.  untuk data. Dapat berupa proses input/ouput
5.  untuk inisialisasi data
6.  Untuk menghubungkan komponen dan menunjukan arah.