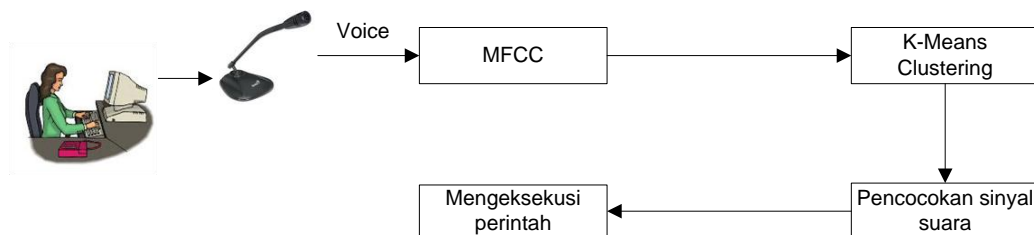


BAB 3

ANALISIS DAN PERANCANGAN SISTEM

3.1 Deskripsi Umum Sistem

Proses *voice command* pada sistem dapat dijelaskan sebagai berikut. Pertama-tama, pengguna menentukan kata yang akan dijadikan sebagai perintah untuk membuka aplikasi tertentu kemudian mengucapkannya pada *microphone*, dan memilih aplikasi apa yang akan dibuka oleh perintah tersebut. Sistem akan menyimpan data tersebut dalam sebuah database. Setelah itu barulah pengguna mengucapkan kata yang menjadi perintah tadi. Sistem akan mencocokkan sinyal suara yang masuk dengan data yang terdapat dalam template. Jika data sinyal sama, maka komputer akan mengeksekusi aplikasi yang telah ditentukan sebelumnya.



Gambar 3.1 Gambaran Umum Sistem Aplikasi Voice Command

Gambar 3.1 diatas menunjukan tentang gambaran umum sistem. Setelah menginputkan kata, selanjutnya masuk ke dalam tahap MFCC (*Mel Frequency Cepstrum Coefficients*). MFCC (*Mel Frequency Cepstrum Coefficients*) feature

extraction mengkonversikan *signal* suara kedalam beberapa vektor data berguna bagi proses pengenalan suara. Dalam MFCC (*Mel Frequency Cepstrum Coefficients*) sendiri terdapat tujuh tahapan yaitu *Pre Emphasize*, *Frame Blocking*, *Windowing*, *Fast Fourier Transform*, *Mel Frequency Warping*, *Discrete Cosine Transform*, dan *Cepstral liftering* ,yang telah dijelaskan pada bab sebelumnya. Data hasil MFCC (*Mel Frequency Cepstrum Coefficients*)*Feature extraction* kemudian akan memasuki tahap *K-Means Clustering* tahap ini membuat beberapa vektor pusat sebagai wakil dari keseluruhan vektor data yang ada. Tahap mencocokkan sinyal adalah tahap perhitungan *probabilitas* kemiripan pola dari tiap-tiap model sinyal yang mempunyai *probabilitas* kemiripan yang tertinggi dan berdasarkan pada *template*.

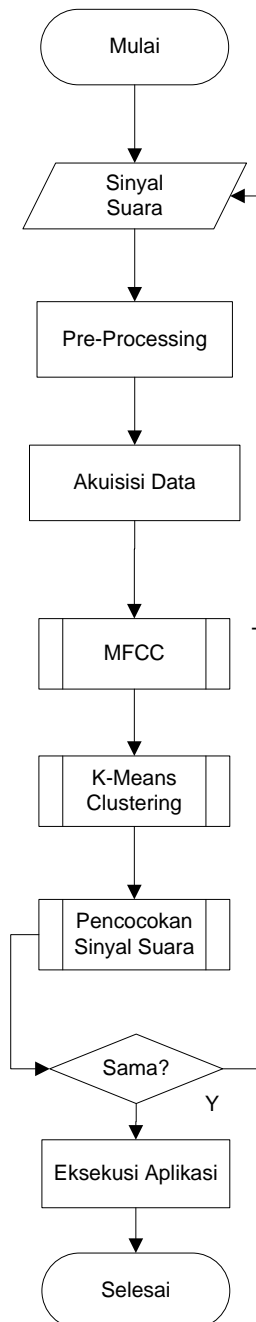
3.2 Analisis Sistem

Analisis sistem dapat didefinisikan sebagai penguraian dari suatu sistem yang utuh kedalam bagian-bagian komponennya dengan maksud untuk mengidentifikasi dan mengevaluasi permasalahan-permasalahan, hambatan-hambatan yang terjadi dan kebutuhan-kebutuhan yang diharapkan sehingga dapat diusulkan perbaikan-perbaikannya.

Dalam proses pembuatan suatu sistem mutlak dilakukan penelitian dan penganalisaan tentang sistem yang akan dibangun, berikut adalah beberapa analisis yang dilakukan untuk membangun perangkat lunak untuk membuka aplikasi pada komputer dengan perintah suara menggunakan metode *Mel Frequency Cepstrum Coefficients* (MFCC).

3.2.1 Analisis Pengenalan Suara

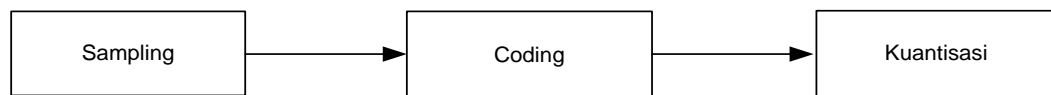
Dalam aplikasi ini terdapat 4 tahapan yaitu, MFCC, *K-Means Clustering*, pencocokan sinyal, dan eksekusi program. Secara umum program mengikuti alur berikut ini :



Gambar 3.2 Gambaran kerja sistem Aplikasi Voice Command

3.2.1.1 Pre-Processing

Sinyal suara yang akan diproses bersifat analog sehingga jika akan dilakukan pengolahan secara digital, sinyal suara tersebut harus dikonversi menjadi sinyal digital, berupa urutan angka dengan tingkat presisi tertentu yang dinamakan *analog to digital conversion* dengan menggunakan *analog-to-digital converter* (ADC). Konsep Kerja ADC terdiri dari tiga proses :



Gambar 3.3 Konsep Kerja ADC (Analog To Digital Converter)

Keterangan konsep kerja ADC :

1. *Sampling* adalah konversi sinyal kontinu dalam domain waktu menjadi sinyal diskrit, melalui proses *sampling* sinyal pada selang waktu tertentu. Sehingga jika $x_0(t)$ adalah sinyal input, maka outputnya adalah $x_0(nT)$, dengan T adalah interval *sampling*.
2. Kuantisasi adalah proses untuk membulatkan nilai data kedalam bilangan bilangan tertentu yang telah ditentukan terlebih dahulu.
3. Coding, pada proses ini, tiap nilai diskrit yang telah didapat, direpresentasikan dengan angka binary n-bit.

3.2.1.2 Akuisisi Data

Data berupa sinyal suara diperoleh dengan cara merekam suara melalui mikrofon yang dihubungkan dengan komputer. Perekaman suara di dalam aplikasi

menggunakan frekuensi sampling standar 8000Hz. Suara dengan format .wav ini bisa menggunakan 16 *bits/sample* dan 1 untuk *channel mono*. Durasi suara yang direkam apabila lebih pendek lebih mudah untuk diambil perbedaannya. Dalam analisis ini digunakan contoh durasi rekaman yang diambil adalah 2 detik.

$$X = FS \times dt(\text{detik}) \times \left(\frac{\text{bit}}{8}\right) \times j \quad (3.1)$$

dimana :

X = data sampling sinyal

Fs = frekuensi sampling

dt = durasi rekaman (detik)

bit = jumlah bit resolusi

j = 1 untuk *mono* atau 2 untuk *stereo*

Perhitungan pada proses akuisi data untuk pengambilan sampling adalah :

$$8000 \times 2 \times (16/8) \times 1 = 32000 \text{ byte}$$

Untuk menghitung sample rate, gunakan cara $\frac{F_s}{T_s}$ dengan time 2s maka

didapatkan :

$$\text{Sample rate} = \frac{F_s}{T_s} = \frac{32000}{2} = 16000 \text{ Hz}$$

Untuk mendapatkan sample point dengan waktu pengambilan data setiap

20 ms maka didapatkan :

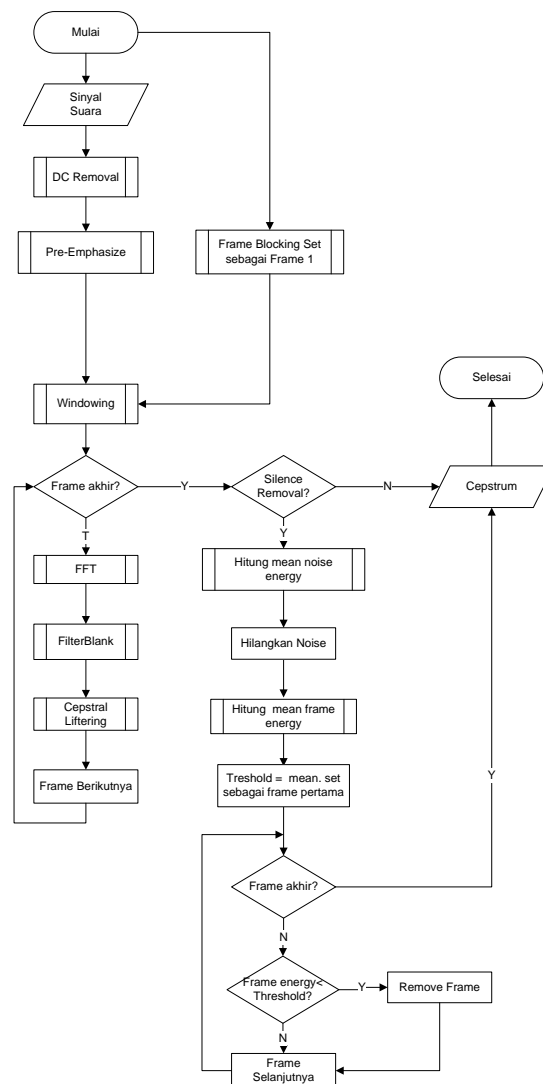
Sample point = sample rate * waktu pengambilan data

$$16000 * 0,02 = 320 \text{ sample points}$$

Untuk penelitian kali ini, digunakan jumlah contoh data sebanyak 8. Data tersebut adalah : (140, 116, 136, 160, 180, 190, 195,180).

3.2.1.3 MFCC

Metode *Mel Frequency Cepstrum Coefficients* (MFCC) ini menggunakan beberapa parameter yang akan berperan penting dalam menentukan tingkat keberhasilan pengenalan *signal* suara. Berikut ini adalah keseluruhan proses MFCC *Feature extraction* :



Gambar 3.4 Proses metode *Mel Frequency Cepstrum Coefficients* (MFCC)

3.2.1.3.1 DC Removal

Fungsi ini menghitung nilai rata-rata dari data *sample* suara, dan mengurangkannya untuk setiap *sample* pada window. Tujuannya adalah untuk memperoleh normalisasi dari data suara input.

Rumus :

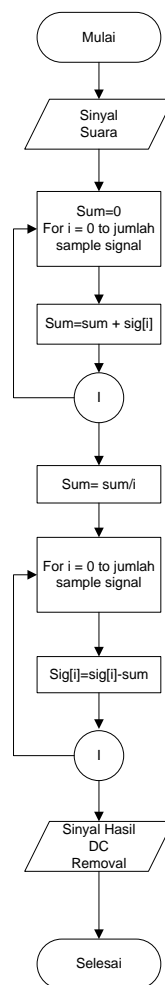
$$D[i] = s[i] - \frac{\sum_{i=1}^n s[i]}{n} \quad (3.2)$$

$D[i]$ = hasil *signal* ke -i setelah dilakukan DC Removal

$S[i]$ = *signal* awal ke-i

N = jumlah *sample*, $n > 0$

berikut in adalah flowchart dari proses DC removal :



Gambar 3.5 Flowchart proses DC Removal

untuk sampel data sinyal, dibawah ini :

(140, 116, 136, 160, 180, 190, 195, 180)

Hitung DC Removal :

$$\frac{140 + 116 + 136 + 160 + 180 + 190 + 195 + 180}{8} = 162,125$$

Setelah nilai rata-rata diketahui, kurangkan nilai Sinyal awal dengan hasil

DC Removal sehingga nilai sinyal menjadi :

$$n_0 = 140 - 162,125 = -22,125$$

$$n_1 = 116 - 162,125 = -46,125$$

$$n_2 = 136 - 162,125 = -26,125$$

$$n_3 = 160 - 162,125 = -2,125$$

$$n_4 = 180 - 162,125 = 17,875$$

$$n_5 = 190 - 162,125 = 27,875$$

$$n_6 = 195 - 162,125 = 32,875$$

$$n_7 = 180 - 162,125 = 17,875$$

Sehingga data sinyal setelah dilakukan pre-emphasis adalah :

(-22.125 , - 46.125 , - 26.125 , - 2.125 , 17.875 , 27.875 , 32.875 , 17.875)

3.2.1.3.2 *Pre-emphasize*

Pre-emphasis dilakukan untuk memperbaiki *signal* dari gangguan *noise*, sehingga dapat meningkatkan tingkat akurasi dari proses *feature extraction*.

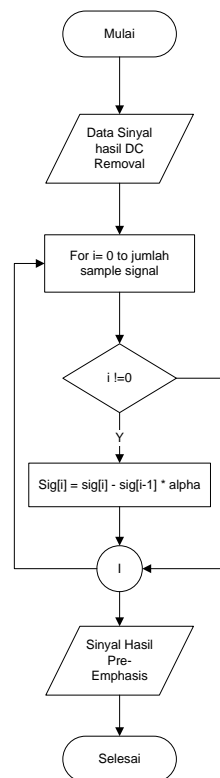
Default dari nilai α yang digunakan dalam proses *pre-emphasis filtering* adalah 0,97.

$$y[n] = s[n] - \alpha s[n-1] \quad \dots\dots\dots(3.3)$$

$y[n]$ = *signal hasil pre – emphasize filter*

$s[n]$ = *signal sebelum pre – emphasize filter*

Berikut ini adalah flowchart proses *pre-emphasis* :



Gambar 3.6 Proses Pre-emphasis

Sample Sinyal :

$signal = (-22.125, -46.125, -26.125, -2.125, 17.875, 27.875, 32.875, 17.875)$ dengan $\alpha = 0,97$

$$Y_0 = -22,125$$

$$Y_1 = (-46,125) - (-22,125 * 0,97) = -24,6$$

$$Y_2 = (-26,125) - (-46,125 * 0,97) = 18,6$$

$$Y_3 = (-2,125) - (-26,125 * 0,97) = 23,2$$

$$Y_4 = 17,875 - (-2,125 * 0,97) = 19,9$$

$$Y_5 = 27,875 - (17,875 * 0,97) = 10,5$$

$$Y_6 = 32,875 - (27,875 * 0,97) = 5,8$$

$$Y_7 = 17,875 - (32,875 * 0,97) = 14$$

Data sinyal baru adalah data sinyal sebelum proses *pre-emphasis* ditambah dengan data hasil *pre-emphasis* diatas. Sehingga sinyal setelah *pre-emphasis* :

$$N_n = N_n + Y_n \quad (3.4)$$

$$N_0 = -22,125 + (-22,125) = -44,25$$

Begitu juga dengan ketujuh data yang lainnya, sehingga didapatkan nilai sinyal setelah *pre-emphasis* adalah :

$$(-44.25, -70,7, -7,5, 21.075, 37.7, 38.3, 38.6, 31.8)$$

3.2.1.3.3 *Frame Blocking*

Hasil perekaman suara merupakan sinyal analog yang berada dalam domain waktu yang bersifat variant time, yaitu suatu fungsi yang bergantung waktu. Oleh karena itu sinyal tersebut harus dipotong-potong dalam slot-slot waktu tertentu agar dapat dianggap invariant. Sinyal suara dipotong sepanjang 20 milidetik .Setiap potongan tersebut disebut *frame*.

Untuk menghitung jumlah Frame digunakan rumus :

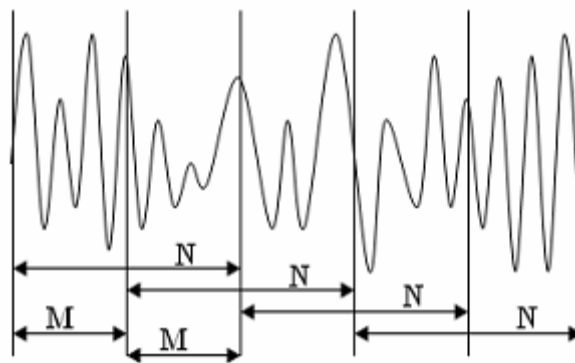
$$\text{Jumlah frame} = ((I - N) / M) + 1 \quad (3.5)$$

I = Sample rate

N = Sample point (Sample rate * waktu framing (s))

$$M = N/2$$

Potongan *frame* digambarkan seperti gambar dibawah ini :



Gambar 3.7 Frame Blocking

Berikut ini adalah algoritma dari proses framing :

```

Procedure framing
{menghitung jumlah frame dalam satu sinyal
i.s : panjang sinyal, waktu sampling telah diketahui
f.s : jumlah frame}
kamus
N, M, I, Jumframe :integer
Ts = 0.02
Algoritma
I ← Fs / Ts;
N ← I * Ts
M ← N/2
Begin
    jumFrame ← ((I-N)/M)+1;
end

```

Dengan waktu $T_s = 20 \text{ ms}$, dan sampel rate = 16000 Hz,

maka didapatkan Frame size (N)

$$N = 16000 * 0,02 = 320 \text{ Sample Point}$$

Dan M = 160

Sehingga Jumlah Frame :

$$((I-N) M) + 1 = ((16000-320) / 160) + 1 = 99$$

3.2.1.3.4 Windowing

Setelah melakukan framing, selanjutnya dilakukan proses windowing . berikut ini adalah algoritma proses Windowing :

```

Procedure windowing
{menghitung sinyal windowing
i.s: Sinyal hasil pre-emphasis telah diketahui
f.s : sinyal hasil windowing diketahui}
Kamus
Xo : double
N,N :int
Algoritma
For n=0 to N do
    Begin
        W[n] = 0,54 - 0,46 cos (( 2*Pi*n)/(N-1));
        X[n] = X[n] * W[n]
    End
endfor

```

Dengan menggunakan rumus $w(n) = 0,54 - 0,46 \cos\left(\frac{2\pi n}{N-1}\right)$ diperoleh hasil sebagai berikut :

$$w_0 = 0,54 - 0,46 \cos\left(\frac{2 * 3.14 * 0}{320 - 1}\right) = 0,08$$

$$\text{Sehingga, } X_0 = -44,25 * 0.08 = -3,54$$

Dengan cara yang sama lakukan pada titik yang lain, dan diperoleh nilai

(-3.54, -5.6, -0.6 , 1.68, 3.016 , 3.06 , 3.08, 2.54)

3.2.1.3.5 Fast Fourier Transform (FFT)

Analisa berdasarkan *fourier transform* sama artinya dengan analisa spektrum, karena *fourier transform* merubah *signal* digital dari time domain ke frekuensi domain. FFT dilakukan dengan membagi N buah titik pada transformasi diskrit menjadi 2, masing masing (N/2) titik transformasi. Proses memecah menjadi (N/4) dan seterusnya hingga diperoleh titik minimum.

FFT (Fast Fourier Transform) adalah teknik perhitungan cepat dari DFT. FFT adalah DFT dengan teknik perhitungan yang cepat dengan memanfaatkan sifat periodikal dari transformasi fourier. Perhatikan definisi dari FFT :

$$F(k) = \sum_{n=1}^N f(n).e^{-j2\pi knT/N} \quad (3.5)$$

Atau dapat dituliskan dengan :

$$F(k) = \sum_{n=1}^N f(n) \cos(2\pi knT/N) - j \sum_{n=1}^N f(n) \sin(2\pi knT/N) \quad (3.6)$$

Untuk melihat nilai hasil FFT digunakan rumus

$$|f(u)| = [R^2 + I^2]^{1/2} \quad (3.7)$$

Berikut adalah Algoritma FFT :

```

Procedure FFT
{menghitung sinyal hasil FFT
i.s : banyaknya data dan sinyal hasil windowing diketahui
f.s : nilai hasil FFT}
Kamus
I,n :integer
A,b :double
Algoritma
For i= n div 2 do
Begin
    Z[n] = 0
    For n=0 to N do
        begin
            A ← X[i] * cos (2*Pi*n*i)/ N
            b ← X[i] * Sin (2*Pi*n*i)/ N
        end
    end for
    Z[n] ← sqrt (a^2 +b^2)/N
End
For i= n div 2 +1 to n do
    begin
        x←i-((0+n)div 2)
        y←((0+n)div 2)-x
        Z[n]←Z[y]
    end
end for

```

Diketahui sinyal hasil windowing :

(-3.54, -5.6, -0.6 , 1.68, 3.016 , 3.06 , 3.08, 2.54)

Untuk F_0 , maka diperoleh perhitungan sebagai berikut :

$$(f_0) = \frac{1}{8}[-3,54 (\cos(\frac{2\pi * 0 * 0}{8}))] - j \sin \frac{2\pi * 0 * 0}{8} +$$

$$[-5,6 (\cos(\frac{2\pi * 0 * 1}{8}))] - j \sin \frac{2\pi * 0 * 1}{8} +$$

$$[-0,6 (\cos(\frac{2\pi * 0 * 2}{8}))] - j \sin \frac{2\pi * 0 * 2}{8} +$$

$$[1,68 (\cos(\frac{2\pi * 0 * 3}{8}))] - j \sin \frac{2\pi * 0 * 3}{8} +$$

$$[3,016 (\cos(\frac{2\pi * 0 * 4}{8}))] - j \sin \frac{2\pi * 0 * 4}{8} +$$

$$[3,06 (\cos(\frac{2\pi * 0 * 5}{8}))] - j \sin \frac{2\pi * 0 * 5}{8} +$$

$$[3,08 (\cos(\frac{2\pi * 0 * 6}{8}))] - j \sin \frac{2\pi * 0 * 6}{8} +$$

$$[2,54 (\cos(\frac{2\pi * 0 * 7}{8}))] - j \sin \frac{2\pi * 0 * 7}{8} = 0,4545 + 0j = 0,4545$$

Gunakan rumus $|[R^2 + I^2]^{1/2}| = |0,4545| = 0,4545$

Maka $F_0 = 0,4545$ Dengan cara yang sama lakukan kepada ketujuh data sinyal lainnya. Sehingga diperoleh data sinyal hasil FFT adalah :

$$(0,4545, 0,321, 0,612, 0,102, 1,546, 1,50, 1,51, 1,21)$$

3.2.1.3.6 Filterbank

Magnitude hasil dari proses FFT selanjutnya akan melalui tahap *filterbank*.

$$Y[i] = \sum_{j=1}^N S[j] H_i[j] \quad (3.7)$$

N = jumlah magnitude spectrum

$S[j]$ = magnitude spectrum pada frekuensi j

$H_i[j]$ = koefisien *filterbank* pada frekuensi j ($1 \leq i \leq M$)

M= jumlah Channel dalam *filterbank*

Untuk mendapatkan H_i digunakan rumus :

$$H_i = \frac{2595 * \log(1 + \frac{f}{700})}{\frac{Si}{2}}$$

Berikut ini adalah Algoritma untuk proses FilterBank

```

Procedure Filterbank
{ menghitung nilai Mel Spectrum
i.s : Sinyal FFT
f.s : Sinyal Hasil filterbank}
kamus
H : double
I,N :integer
Algoritma
For i = 0 to n do
    Begin
        H[i] ← (2595 * log ( 1+ 1000/700))/ (X[i]/2);
        S[i] ← H[i] * X[i];
    end
End for

```

Diketahui :

$$S_0 = 0.4545$$

$$H_0 = \frac{2595 * \log(1 + \frac{0}{700})}{\frac{0.4545}{2}} = 4400, 37$$

$$\text{Maka didapat } S_0 = 0.4545 * 4400.37 = 1999,96$$

Dengan cara yang sama maka didapatkan sinyal hasil filterbank adalah :

$$(1999.96, 246.4, 418.7, 316.0, 211.4, 216.2, 117.3, 181.1)$$

3.2.1.3.7 Discrete Cosine Transform (DCT)

Proses ini merupakan langkah akhir dari *feature extraction*. Hasil dari DCT ini adalah fitur-fitur yang dibutuhkan oleh penulis untuk melakukan proses analisa terhadap pengenalan suara tersebut. Menggunakan rumus :

$$\tau_n = \sum_{k=1}^K (\log S_k) \cos\left[n\left(k - \frac{1}{2}\right)\frac{\pi}{K}\right] \quad (3.8)$$

S_k = keluaran dari proses *filterbank* pada indeks k

K = jumlah koefisien yang diharapkan

Berikut ini adalah algoritma untuk proses DCT :

```

Procedure DCT (k : integer, dct : float)
Kamus
    K : integer
    fbank: float;
Algoritma
    For(int n=0; n<=k ; n++)
    begin
        Sum = 0.0;
        Sum +=fbank[k-1]*cos(n*(k-0,5) * (PI/FilterNum));
        k+1;
    end
    End for;

```

Koef(k) = 8

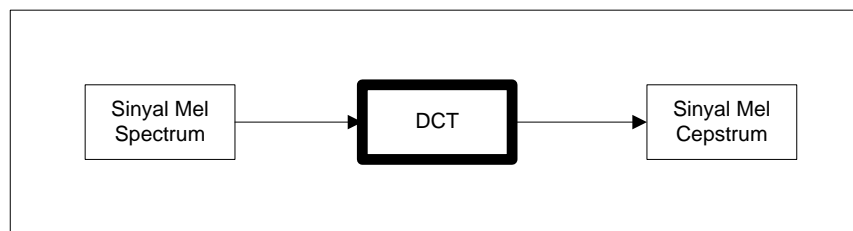
Untuk n=0 maka,

$$\begin{aligned}
 n_0 = & \log(1999,9) \cos\left[0 \left(0 - \frac{1}{2}\right) \frac{3,14}{8}\right] + \log(246,4) \cos\left[0 \left(1 - \frac{1}{2}\right) \frac{3,14}{8}\right] + \\
 & \log(418,7) \cos\left[0 \left(2 - \frac{1}{2}\right) \frac{3,14}{8}\right] + \log(316,1) \cos\left[0 \left(3 - \frac{1}{2}\right) \frac{3,14}{8}\right] + \\
 & \log(211,4) \cos\left[0 \left(4 - \frac{1}{2}\right) \frac{3,14}{8}\right] + \log(216,2) \cos\left[0 \left(5 - \frac{1}{2}\right) \frac{3,14}{8}\right] +
 \end{aligned}$$

$$\log(117.3) \cos \left[0 \left(6 - \frac{1}{2} \right) \frac{3.14}{8} \right] + \log(181.1) \cos \left[0 \left(7 - \frac{1}{2} \right) \frac{3.14}{8} \right]$$

$$= 19.8$$

Jadi $n_0 = 19,8$



3.2.1.3.8 Cepstral liftering

Hasil dari fungsi DCT adalah cepstrum yang sebenarnya sudah merupakan hasil akhir dari proses *feature extraction*. Tetapi, untuk meningkatkan kualitas pengenalan, maka cepstrum hasil dari DCT harus mengalami *cepstral liftering*

$$w[n] = \left\{ 1 + \frac{L}{2} \sin \left(\frac{n\pi}{L} \right) \right\} \quad (3.9)$$

L = jumlah cepstral coefficients

N = index dari cepstral coefficients

```

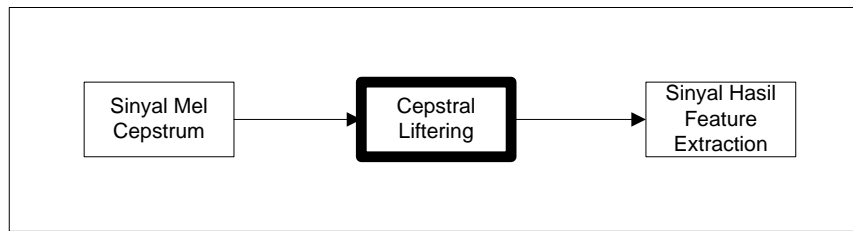
Procedure LifteringCoefficients ( int type, int n;k)
Kamus
    Int k;
    Double h, dct;
    Integer n;
Algoritma
    H= n/2.0;
    For (k=1; k<= n; k++)
    begin
        Lifter_function[k-1] = dct * ( h ( sin ((n +1) *pi)/ (k-
        1)
    end
    End for; }
  
```

Diketahui : nilai DCT = 19,8

Untuk w_0 , maka

$$w_0 = 19,8 * \frac{8}{2} * \sin \frac{\pi}{7} = 0.62$$

Hasil diatas merupakan hasil dari feature extraction.



3.2.1.3.9 Remove Silence

Dengan demikian selesailah proses *Mel Frequency Cepstrum Coefficients* (MFCC) *feature extraction*. Namun hasilnya masih mengandung *frame-frame* silence. *Frame* semacam ini dapat mengganggu pengenalan kata. Oleh karena itu sebaiknya *frame-frame* ini harus dihilangkan.

Frame energy sendiri didapatkan dari *frame* ke-0 dari hasil *feature extraction*. *Energy* tersebut akan dijadikan acuan untuk apakah sebuah *frame* tergolong sebagai suatu *noise*. Apabila energi suatu *frame* kurang dari nilai rata-rata, maka tergolong sebagai *noise* dan akan segera dihilangkan. Perhitungan *noise removal* ini adalah sebagai berikut :

$$Noise = \frac{\sum_{i=1}^n s[i]}{n} \quad (3.8)$$

Noise = rata-rata *energy frame*

$$s[i] = \text{signal frame ke-}i$$

Berikut ini adalah algoritma dari proses remove silence :

```

Procedure meanFrame
Kamus
    Double temp = 0, temp2=0;
    N, i : integer
Algoritma
    If (n>2)
        For (int i=0; i<dct[i]: i++)
            begin
                Temp2=dct[i]);
            end
            Temp+=temp2;
        End
        end for;
    Temp/=n;
    End if

```

Diketahui : $\text{signal} = \{ 0.62, 0.103, 0.225, 0.324, 0.214, 0.211, 0.15, 0.701 \}$

Noise

$$= \frac{(0.62 + 0.643 + 0.825 + 0.794 + 0.764 + 0.771 + 0.815 + 0.781)}{8}$$

$$= 0.751$$

Hilangkan noise, yaitu nilai yang kurang dari nilai rata-rata diatas, maka didapatkan nilai :

$$(0.225, 0.324, 0.214, 0.211, 0.15, 0.701)$$

Hitung rata-rata energi tiap frame. Dalam analisis ini terdapat 99 frame. Setelah itu hitung rata-rata nilai dari keseluruhan frame, nilai inilah yang akan dijadikan treshold. Nilai frame yang lebih kecil dari nilai treshold akan dihilangkan.

```

Procedure removeSilence
Kamus
    Double temp = 0, temp2=0;
    N, i : integer
Algoritma
    If (n>2)
        For (int i=0; i<n ; i++)
            begin
                                Temp2=meanFrame[i]);
            end

            Temp+=temp2;
            End
        end for;
    Temp/=n;
    End if

```

3.2.1.4 K-Means Clustering

Metode yang digunakan dalam melakukan *Clustering* terhadap hasil dari *feature extraction* adalah K-Means Clustering.

Pada fungsi *Clustering*, inputan sebenarnya merupakan cepstrums hasil dari *feature extraction*. Terdapat proses dalam fungsi *Clustering* untuk mendapatkan vektor pusat. Pencarian vektor pusat dilakukan secara berulang ulang sehingga didapatkan vektor pusat yang mewakili seluruh vektor hasil *feature extraction*.

K-Means ini menggunakan nilai rata-rata yang diambil dari setiap cluster. Untuk menghitung rata-rata dari setiap cluster

$$C_k = \text{Mean}(\text{Anggota}_k) \quad (3.11)$$

C_k adalah nilai rata-rata dari cluster K (contoh C_1 adalah nilai rata-rata dari cluster yang pertama). Anggota_k adalah semua anggota dari kluster K.

Untuk memilih anggota dari suatu cluster, caranya adalah dengan menghitung selisih antara data dan setiap nilai rata-rata cluster. Cluster yang nilai rata-ratanya memiliki selisih terkecil dengan data tersebut, merupakan cluster dimana data tersebut dikategorisasikan. Secara matematis seperti berikut.

$$T = \text{Argmin}_{k \in K}(\text{selisih}(x, C_k)) \quad (3.12)$$

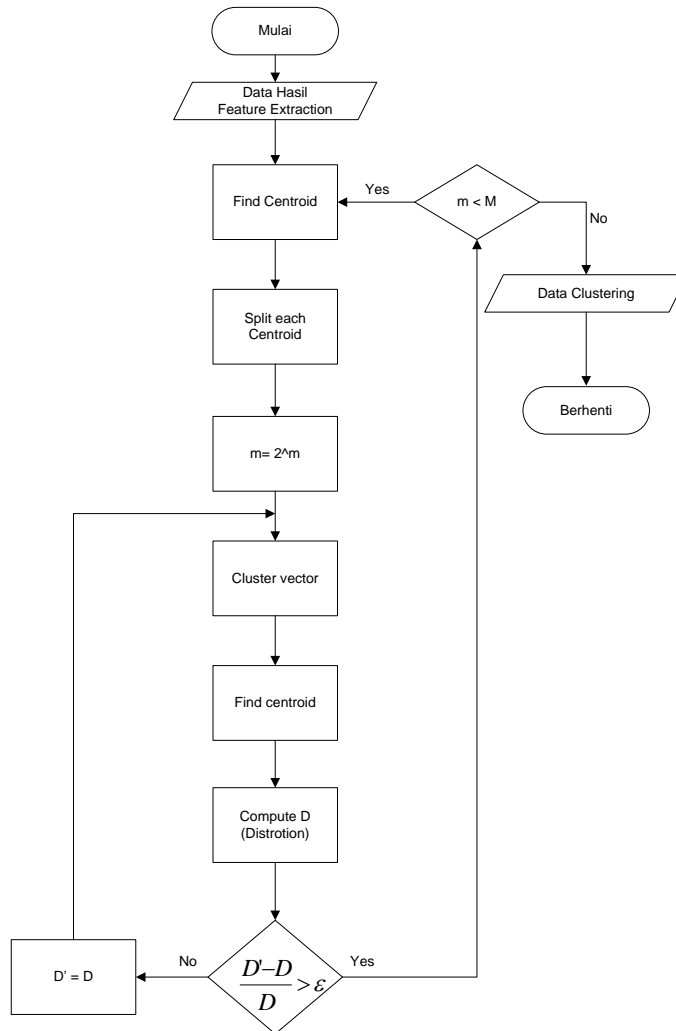
Berikut ini adalah penjelasan dari algoritma K_Means Clustering :

1. Tentukan K
2. Tentukan centroid secara random
3. Hitung jarak setiap data ke masing-masing centroid
4. Setiap data memilih centroid terdekat
5. Tentukan posisi centroid baru dengan cara menghitung nilai rata-rata dari data-data yang terletak pada centroid yang sama.
6. Jika posisi centroid baru dengan centroid lama tidak sama ulangi langkah ke 3.

Untuk penentuan jumlah klustering, akan menentukan nilai homogenitas data dan menentukan waktu pemrosesan. Untuk setiap kluster, akan ditentukan centroidnya menggunakan urutan langkah di atas.

Dalam analisis ini akan digunakan kluster sebanyak 20 kluster. Dalam analisis sebelumnya telah diketahui bahwa jumlah titik point sampel sebanyak 320 point, sehingga dalam satu kluster akan terdapat 16 titik data. Dari 16 data itu akan dihitung nilai rata-rata ,menggunakan rumus 3.11 . setelah mendapatkan nilai centroid yang tepat maka proses clustering berakhir.

Berikut ini adalah flowchart untuk proses *K-Means Clustering* :



Gambar 3.8 Flowchart *K-Means Clustering*

Contoh terdapat empat titik sebagai berikut :

A(5,3) , B(-1,1), C(1,-2), D(-3,-2) akan dilakukan clustering menjadi 2

kelompok (k=2). Langkah-langkahnya adalah sebagai berikut :

1. Langkah pertama

Tabel 3.1 Koordinat Centroid

Cluster	Koordinat dari centroid	
	X_1	X_2
(AB)	$\frac{5 + (-1)}{2} = 2$	$\frac{3 + 1}{2} = 2$
(CD)	$\frac{5 + (-3)}{2} = -1$	$\frac{-2 + (-2)}{2} = -2$

2. Langkah kedua

Lakukan perhitungan jarak dengan euclidean dari masing-masing item dari centroid (pusat) cluster dan tandai kembali setiap item berdasarkan kedekatan group.

$$d^2(A, (AB)) = (5 - 2)^2 + (3 - 2)^2 = 10$$

$$d^2(A, (CD)) = (5 - 2)^2 + (3 + 2)^2 = 61$$

A lebih dekat dengan cluster (AB) dibandingkan dengan cluster (CD).

$$d^2(B, (AB)) = (-1 + 2)^2 + (1 + 2)^2 = 10$$

$$d^2(B, (CD)) = (-1 + 1)^2 + (1 + 2)^2 = 9$$

B, lebih dekat dengan CD sehingga B akan ditandai menjadi anggota pada cluster (CD), sehingga membentuk cluster baru (BCD), maka koordinat dari pusat cluster terupdate sebagai berikut :

Tabel 3.2 Koordinat Centroid terupdate

Cluster	Koordinat dari centroid	
	X_1	X_2
(A)	5	3
(CD)	-1	-1

3. Langkah ke tiga

Lakukan check untuk setiap item untuk ditandai kembali, menggunakan cara seperti di tahapan ke 2.

Tabel 3.3 Koordinat Centroid baru

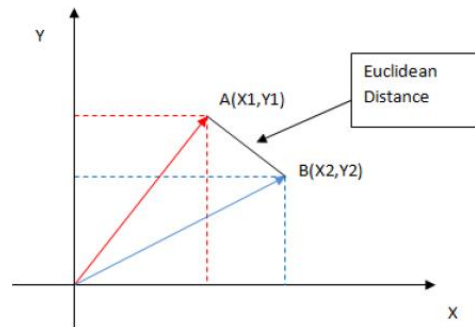
Cluster	Koordinat dari centroid			
	A	B	C	D
(A)	0	40	41	89
(BCD)	52	4	5	5

Kita lihat setiap item yang baru telah ditandai untuk cluster berdasarkan centroid (pusat) terdekat maka proses telah dihentikan. Sehingga dengan $k=2$ cluster maka terbentuk cluster sebagai berikut : A dan (BCD).

3.2.1.5 Pencocokan *Signal*

Untuk dapat membuka aplikasi yang diinginkan, maka data *signal* baru yang masuk akan dicocokkan dengan data yang telah ada dalam database sebelumnya. Setiap vektor dari model yang diujicobakan, dibandingkan dan dihitung *euclidean distance*-nya dengan semua vektor yang ada pada salah satu model database secara bergantian. Kemudian diambil *distance* yang paling minimum antara sebuah vektor pada model yang diuji cobakan dengan semua

vektor yang ada pada salah satu model database. Sehingga didapatkan N minimum *distance*. Berikut ini adalah gambar perhitungan distance :



Gambar 3.9 Perhitungan Distance

$$d(A, B) = \sqrt{(X1 - X2)^2 + (Y1 - Y2)^2}$$

Hitung minimal distance dari setiap codebook yang ada. Disitulah dianggap sebagai kemiripan sinyal.

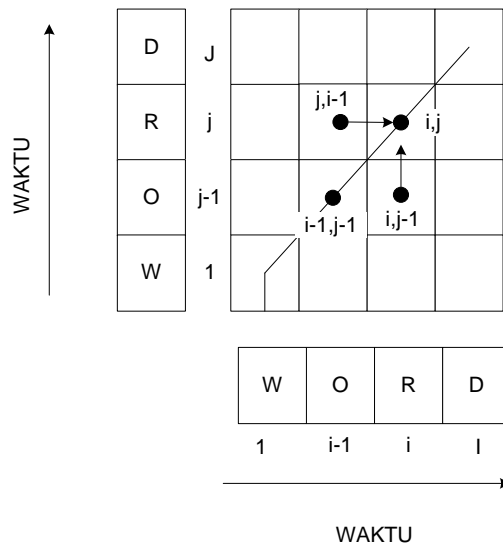
Berikut ini adalah algoritma untuk proses perhitungan *Distance* :

```

Prcedure distance;
Kamus
i,j: integer
D (0,0)=0
algoritma
For i = 1 to i
    D (i, 0) = D (i-1,0) + 1
EndFor
For j = 1 to J
    D (0, j) = D (0, j-1) + 1
EndFor
For i = 1 to i
    For j = 1 to j
        c1 = D (i-1, j-1) + d (i, j,i-1, j-1)
        c2 = D (i-1, j) + 1
        c3 = D (i, j-1) + 1
        D (i, j) = min (c1, c2, c3)
    EndFor
EndFor

```

Berikut ini adalah gambaran mengetahui jarak *warping path*/jarak terbaik dari kata word :



Gambar 3.10 Jarak terbaik

3.2.1.6 Eksekusi aplikasi

Output dari program ini adalah mengeksekusi aplikasi yang diperintahkan oleh pengguna sesuai dengan database perintah yang telah disediakan sebelumnya. Jadi, apabila suatu kata sudah dapat teridentifikasi, maka kata tersebut nantinya akan dicocokkan pada database perintah yang sudah ada.

```
ShellExecute(NULL,"open",path,NULL, NULL, SW_SHOWNORMAL);
```

Code diatas akan menghasilkan terbukanya suatu aplikasi tertentu yang ditunjukan oleh path dan option yang dipilih adalah membuka secara normal.

3.2.2 Deskripsi Kebutuhan Sistem

Sebelum membangun sebuah sistem perlu dilakukan analisis kebutuhan sistem untuk menjamin bahwa sistem yang dibuat sesuai dengan kebutuhan pengguna dan layak untuk dikembangkan. Tahapan analisis kebutuhan sistem dapat dirinci menjadi beberapa tahap guna mempermudah proses analisis secara keseluruhan. Tahapan-tahapan ini sangat penting untuk menjamin keberhasilan pengembangan sistem secara keseluruhan.

3.2.2.1 Deskripsi kebutuhan antarmuka *eksternal*

Analisis kebutuhan antarmuka *eksternal* akan diuraikan secara rinci untuk keperluan perancangan perangkat lunak. Kebutuhan antarmuka *eksternal* tersebut meliputi antarmuka pemakai, antarmuka perangkat keras, dan antarmuka perangkat lunak.

3.2.2.2 Antarmuka Pemakai

Perangkat lunak *voice command* ini menunjukkan proses *feature extraction* MFCC dan proses pencocokan suara dengan data yang ada.

3.2.2.3 Antarmuka Perangkat Keras

Pembangunan perangkat lunak *voice command* ini memerlukan beberapa perangkat keras seperti :

1. Processor yang digunakan Intel core i3
2. Memory yang digunakan 3GB 8000MHz

3. Harddisk, sebagai media storage yang digunakan 160 GB
4. *Microfone*
5. *Speaker*
6. *Keyboard*

3.2.2.4 Antarmuka Perangkat Lunak

Adapun perangkat lunak pendukung pembangunan perangkat lunak *voice command* ini antara lain :

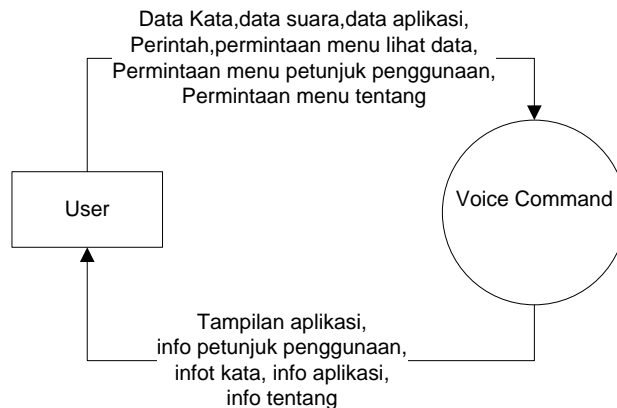
1. Sistem Operasi *Windows Seven*.
2. *Microsoft Visual Basic 6*, digunakan untuk pengkodean sistem.
3. *Microsoft Access* digunakan untuk menyimpan data kata dan perintah.
4. *Rational Rose 2003*, digunakan untuk memodelkan sistem.

3.2.3 Kebutuhan Fungsional

Kebutuhan fungsional dianalisis dengan memodelkan sistem. Pemodelan yang digunakan untuk memodelkan perangkat lunak *voice command* ini adalah pemodelan tersrstruktur. Perangkat lunak ini dimodelkan menggunakan DFD (*Data Flow Diagram*). *Tools* yang digunakan adalah diagram konteks , DFD dan spesifikasi proses yang dibuat menggunakan *Microsoft Visio 2007* sebagai perangkat lunak yang digunakan.

3.2.3.1 Diagram Konteks

Diagram konteks ini menggambarkan sistem aplikasi *voice command*. Berikut ini adalah gambar diagram konteks untuk aplikasi *voice command* :



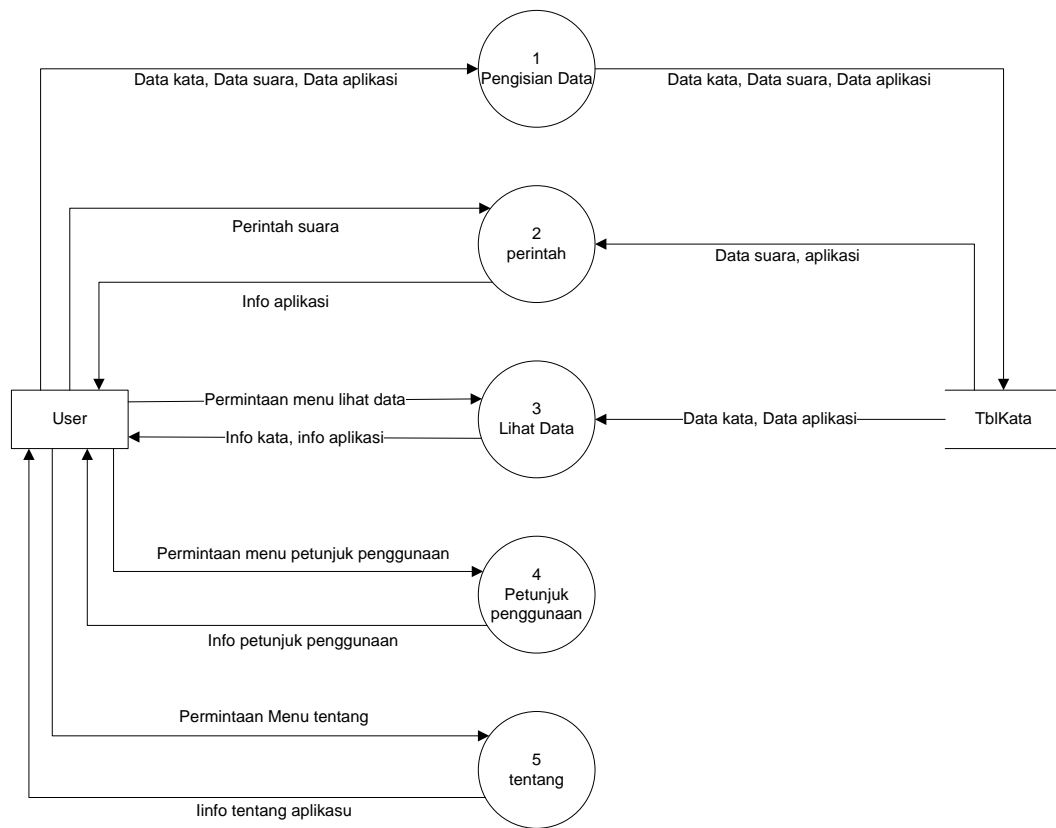
Gambar 3.11 Diagram Konteks Aplikasi *Voice command*

3.2.3.2 DFD (Data Flow Diagram)

DFD sering digunakan untuk menggambarkan suatu sistem yang telah ada atau sistem baru yang akan dikembangkan secara logika tanpa mempertimbangkan lingkungan fisik dimana data tersebut mengalir atau lingkungan fisik dimana data tersebut akan disimpan

1. DFD Level 1 Aplikasi *Voice command*

Pada DFD Level 1 ini terdapat 5 proses yaitu, Pengisian Data, Perintah, Lihat Data, Petunjuk Penggunaan, dan Tentang. Berikut ini adalah gambar untuk DFD Level 1 pada aplikasi *voice command*.



Gambar 3.12 DFD Level 1 Aplikasi *Voice command*

Berikut ini adalah penjelasan bagi setiap proses yang terdapat dalam DFD level 1 :

1 : Pengisian Data

Proses ini dibuat untuk menambahkan data baru pada database, seperti data kata, data suara dan data aplikasi.

2 : Perintah

Proses ini bertujuan untuk memulai menggunakan aplikasi. Memulai untuk memberikan perintah pada aplikasi untuk membuka program.

3: Lihat Data

Proses ini digunakan untuk melihat data kata dan data aplikasi yang telah dimasukan sebelumnya oleh pengguna.

4: Petunjuk Penggunaan

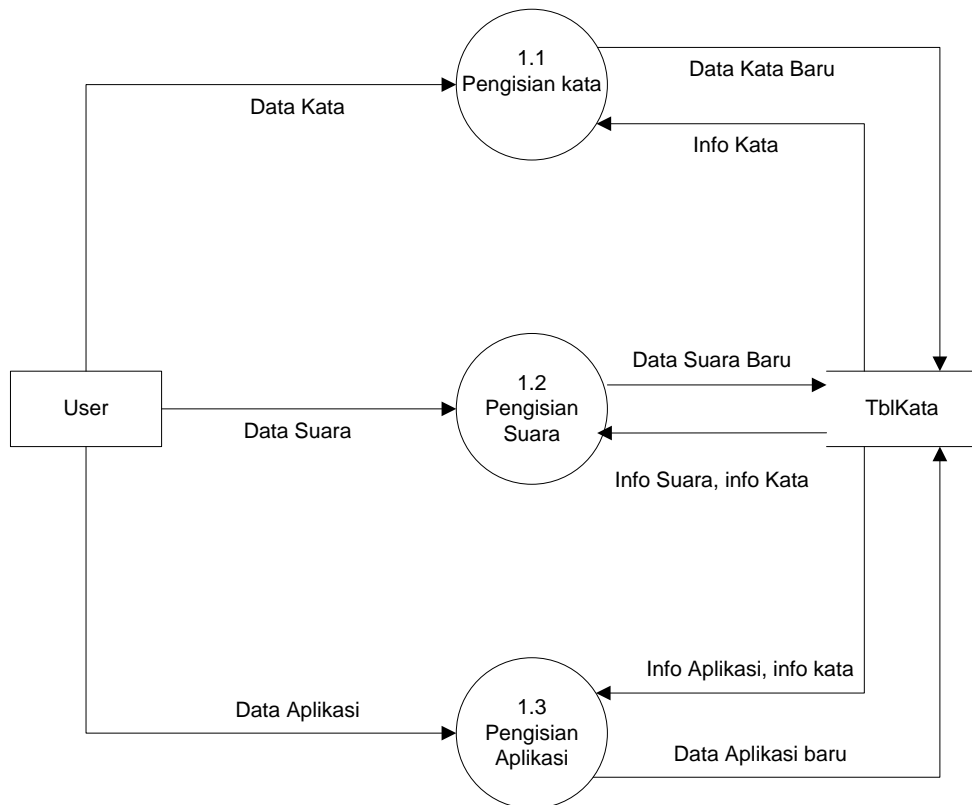
Proses ini digunakan untuk menampilkan form berisi petunjuk penggunaan aplikasi *voice command*.

5: Tentang

Proses ini digunakan untuk melihat tentang aplikasi.

2. DFD Level 2 Proses Pengisian Data

Pada DFD level 2 ini terdapat tiga proses, yaitu pengisian kata, pengisian suara dan pengisian aplikasi. Berikut ini adalah gambar untuk DFD Level 2 Proses File :



Gambar 3.13 DFD Level 2 Proses Pengisian Data

Berikut ini adalah penjelasan proses-proses pada DFD Level2 Proses Pengisian Data :

1.1 : Pengisian kata

Proses ini digunakan untuk menyimpan data kata baru pada database.

1.2 : Pengisian Suara

Proses ini digunakan untuk menyimpan data suara yang baru dimasukan oleh user kedalam database

1.3 : Pengisian Aplikasi

Proses ini digunakan untuk menyimpan data aplikasi yang akan dieksekusi ketika proses pencocokan suara berjalan dengan benar.

3.2.3.3 Spesifikasi Proses

Adapun Spesifikasi proses pada aplikasi *voice command* ini adalah sebagai berikut :

Tabel 3.4 Spesifikasi proses Aplikasi *Voice command*

no	Proses	Keterangan
1	No Proses	1
	Nama Proses	Pengisian Data
	Deskripsi	Memasukan data baru untuk aplikasi <i>voice command</i>
	Input	Data kata, data suara, data aplikasi
	Output	Data suara baru, data kata baru, data aplikasi baru
	Proses	Pengisian data kata, data suara dan data aplikasi
	Logika Proses	1. Menu pengisian data telah dipilih 2. Proses telah menerima data kata, data suara dan data aplikasi 3. Jika user memilih simpan, proses akan menyimpan data kedalam database
2	No Proses	2
	Nama Proses	Perintah
	Deskripsi	Melakukan proses eksekusi perintah
	Input	Perintah suara
	Output	Info aplikasi
	Proses	Sistem mengeksekusi perintah, sesuai dengan perintah user
	Logika Proses	1. Proses menerima sinyal suara 2. Proses melakukan ekstraksi fitur pada sinyal suara menggunakan MFCC 3. Proses melakukan Clustering pada sinyal cepstrum hasil ekstraksi fitur 4. Proses melakukan pencocokan suara tersebut dengan suara yang ada didalam database. 5. Jika data suara cocok, maka proses akan mengeksekusi perintah sesuai dengan perintah suara.
3	No Proses	3
	Nama Proses	Lihat Data
	Deskripsi	Melihat data kata dan data aplikasi
	Input	Menu lihat data
	Output	Info kata, info aplikasi
	Proses	Menampilkan informasi data kata dan data aplikasi
	Logika Proses	1. Menu lihat data telah dipilih 2. Proses menampilkan data kata dan data aplikasi
4	No Proses	4

no	Proses	Keterangan
4	Nama Proses	Petunjuk Penggunaan
	Deskripsi	Menampilkan data petunjuk penggunaan
	Input	menu petunjuk penggunaan
	Output	info petunjuk penggunaan
	Proses	Sistem menampilkan informasi petunjuk penggunaan aplikasi.
	Logika Proses	1. Proses menerima perintah petunjuk penggunaan dari pengguna 2. Proses akan menampilkan informasi petunjuk penggunaan aplikasi
5	No Proses	5.
	Nama Proses	Tentang
	Deskripsi	Menampilkan data tentang aplikasi
	Input	menu tentang
	Output	info tentang aplikasi
	Proses	Sistem menampilkan informasi tentang pembuat aplikasi
	Logika Proses	1. Proses menerima perintah tentang dari pengguna 2. Proses akan menampilkan informasi tentang aplikasi.

3.2.3.4 Kamus Data

Kamus data adalah suatu daftar data elemen yang terorganisir dengan definisi yang tetap dan sesuai dengan sistem, sehingga user dan analis sistem mempunyai pengertian yang sama tentang input, output, dan komponen data store. Berikut ini adalah kamus data untuk aplikasi Voice Command :

Tabel 3.5 Kamus Data Aplikasi Voice command

Nama Aliran Data	Data Kata
Deskripsi	Data kata masukan user
Type	Varchar
Range	[A...Z a..z]
Nama Aliran Data	Data Suara
Deskripsi	Data suara hasil perekaman user
Type	Varchar
Range	[0..1]
Nama Aliran Data	Data Aplikasi
Deskripsi	Data aplikasi masukan pengguna
Type	Varchar
Range	[A..Z a..z]

3.2.3.5 Struktur Tabel

Struktur tabel merupakan urutan isi atau data yang berada dalam suatu record. Struktur tabel digunakan sebagai suatu alat bantu dalam menyelesaikan program. Pada perancangan perangkat lunak yang dibangun perlu untuk menjelaskan struktur tabel yang mempengaruhi jalannya perangkat lunak atau aplikasi yang dibangun.

No	Nama Field	Type	Ukuran	Keterangan
1	id	Auto number		Id data
2	kata	text	50	Data kata
3	suara	text	255	Data suara
4	aplikasi	text	255	Data aplikasi

3.2.4 Deskripsi kebutuhan Non Fungsional

Berikut ini adalah kebutuhan Non Fungsional yang dibutuhkan oleh sistem:

Tabel 3.6 Deskripsi Kebutuhan Non Fungsional

Kriteria	Tuntutan
Performansi	Harus dapat mendeteksi suara dengan baik, dan dapat mengurangi <i>noise</i> .
	Perangkat lunak yang dibuat dapat dioperasikan pada komputer berspesifikasi minimal Intel Pentium 3 atau yang setara dengan jumlah RAM minimal 256 MB
Batasan memory	Maksimal jumlah memori yang digunakan oleh perangkat lunak tidak boleh melebihi 50 MB
Antar muka	Tulisan pesan dan menu perintah yang ditampilkan harus cukup jelas terbaca oleh pengguna dalam keadaan terang maupun gelap dengan menggunakan warna tulisan dan latar belakang yang tingkat kontrasnya tinggi dengan jenis huruf Arial berukuran minimal 12 poin.
	Modus grafis yang digunakan adalah VGA dengan

	resolusi minimal 640*480 dengan kedalaman warna 8 bit atau 256 warna
	Terdapat berbagai operasi dalam satu tampilan

3.2.5 Atribut Kualitas Perangkat Lunak

Berikut ini menjelaskan tentang kualitas perangkat lunak yang dibangun :

Tabel 3.7 Atribut Kualitas Perangkat Lunak

Kriteria Kualitas	Tuntutan Kualitas
Keandalan	Perangkat lunak dapat dijalankan pada komputer dengan spesifikasi rendah
	Perangkat lunak dapat digunakan untuk membuka aplikasi yang terinstall di Windows.
Ketersediaan	Bahasa pemrograman yang digunakan adalah bahasa yang kecil dan memungkinkan untuk dikembangkan.
	Menggunakan antarmuka perangkat keras yang sudah standar dan tersedia banyak dipasaran yaitu <i>microfone</i> .
Kepemindahan	Perangkat lunak dibuat dengan bahasa pemrograman yang dapat bekerja di berbagai arsitektur komputer.

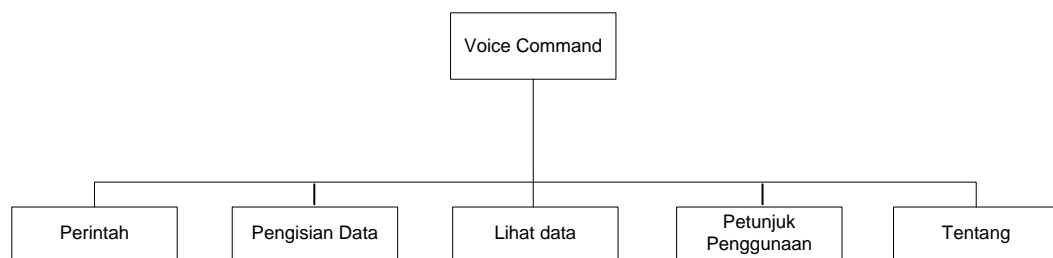
3.2.6 Batasan Perancangan

Batasan-batasan pada tahap perancangan perangkat lunak *voice command* ini adalah :

1. Aplikasi ini hanya dapat membuka aplikasi yang telah terinstall di dalam komputer.
2. Perancangan yang dikembangkan meliputi perancangan prosedural, arsitektur dan interface.

3.3 Perancangan Struktur Program

Struktur program merepresentasikan organisasi komponen program atau modul secara hirarki. Notasi yang digunakan merepresentasikan hirarki tersebut menggunakan diagram pohon. Fungsi pada struktur direpresentasikan dengan simbol persegi, input dan output digambarkan dengan anak panah. Adapun struktur program untuk perangkat lunak pembuka aplikasi dengan perintah suara adalah sebagai berikut :



Gambar 3.14 Struktur Program *Voice command*

3.3.1 Perancangan Antarmuka

Perancangan antarmuka merupakan sebuah penggambaran, perencanaan, dan pembuatan sketsa atau pengaturan dari beberapa elemen yang terpisah ke dalam satu kesatuan yang utuh dan berfungsi. Adapun perancangan antarmuka perangkat lunak *voice command* adalah sebagai berikut :

1. Perancangan Antarmuka *Form* Aplikasi

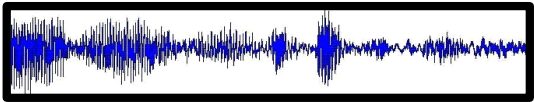
a. Desain *Form* Aplikasi

Form aplikasi merupakan *form* yang digunakan sebagai tampilan pada saat pengguna membuka aplikasi. Berikut ini adalah desain tampilan *form* aplikasi :

F01

Voice Command

Perintah
Pengisian Data
Lihat Data
Petunjuk Penggunaan
Tentang



Image

ON

Keluar

- Klik perintah untuk memulai aplikasi
- Klik pengisian data untuk menuju F02
- Klik lihat data untuk menuju F03
- Klik Petunjuk penggunaan untuk menuju F04
- Klik Tentang untuk menuju F05
- Klik keluar untuk menutup aplikasi

Keterangan :

Nama *Form* : F01

Ukuran layer : Default Window Size

Jenis Font : Arial

Background : Blue

Gambar 3.15 Perancangan Antarmuka *form* aplikasi

b. Deskripsi Objek *Form* Aplikasi

Berikut ini adalah deskripsi objek *form* aplikasi :

Tabel 3.8 Deskripsi Objek *form* Aplikasi

Objek	Jenis	Keterangan
Pengisian data	Button	Menuju F02
Perintah	Button	Digunakan untuk memulai melakukan perintah
Lihat Data	Button	Menuju ke F03

Objek	Jenis	Keterangan
Petunjuk Penggunaan	Button	Menuju ke F04
Tentang	Button	Menuju ke F05
image	Button	Menggambarkan signal suara
Voice Graph	Image	Pada aplikasi, suara akan diproses ketika sinyal suara masuk melebihi batas voice graph sebesar 2000 dari maksimum nilai 50000.
On	Button	Mengaktifkan dan menonaktifkan aplikasi

2. Perancangan antarmuka Pengisian Data

a. Desain *Form* Pengisian Data

User dapat menambahkan data kata , data suara dan data aplikasi dalam form ini untuk kemudian disimpan kedalam tabel. Berikut ini adalah gambaran form pengisian data :

F02

Voice Command

Masukan kata :

Aplikasi :

Rekam Suara :

Mulai

Berhenti

Simpan

Batal

- Masukan data kata ke dalam text area masukan kata
- Klik browse untuk memilih data aplikasi yang akan dibuka
- Klik mulai untuk memulai perekaman
- Klik berhenti untuk memberhentikan perekaman
- Klik simpan untuk menyimpan data yang telah dimasukan
- Klik batal untuk membatalkan penambahan data

Keterangan :

Nama *Form* : F02

Ukuran layer : Default Window Size

Jenis Font : Arial

Background : Blue

Gambar 3.16 Perancangan Antarmuka *form* Pengisian data

b. Deskripsi Objek

Berikut ini adalah deskripsi objek dari *form* pengisian data :

Objek	Jenis	Keterangan
Masukan kata	text	Digunakan untuk memasukkann data kata baru
browse	button	Digunakan untuk memilih aplikasi
mulai	button	Digunakan untuk memulai perekaman
Berhenti	button	Digunakan untuk mmengakhiri perekaman
Simpan	button	Digunakan untuk menyimpan data kedalam database
Batal	button	Digunakan untuk membatalkan pengisian data

3. Perancangan antarmuka lihat data

a. Desain *Form* lihat data

User dapat melihat kata dan aplikasi apa saja yang telah tersimpan dalam database aplikasi.

F03

Voice Command

Kata	Perintah

● Klik keluar untuk menutup form

Keluar

Keterangan :
 Nama *Form* : F03
 Ukuran layer : Default Window Size
 Jenis Font : Arial
 Background : Blue

Gambar 3.17 Perancangan Antarmuka proses lihat data

Berikut ini adalah deskripsi objek dari *form* lihat data :

Tabel 3.10 Deskripsi Objek *form* Lihat Data

Objek	Jenis	Keterangan
Keluar	Button	Digunakan untuk menutup form

4. Perancangan antarmuka petunjuk penggunaan

a. Desain *Form* petunjuk penggunaan

Untuk mempermudah user dalam memakai aplikasi. Maka disediakan menu petunjuk penggunaan. Berikut adalah tampilannya :

F04		<ul style="list-style-type: none"> • Klik tutup untuk menutup <i>form</i>
<div style="border: 1px solid black; padding: 10px; text-align: center;"> <p>Petunjuk penggunaan</p> <div style="border: 1px solid black; width: 100px; height: 100px; margin: 0 auto; position: relative;"> <div style="position: absolute; right: -20px; top: 50%; transform: translateY(-50%); width: 10px; height: 100px; border: 1px solid black;"></div> <div style="position: absolute; right: -10px; top: 50%; transform: translateY(-50%); width: 10px; height: 10px; border: 1px solid black;"></div> </div> <div style="border: 1px solid black; width: 100px; height: 20px; margin: 0 auto; text-align: center;">Tutup</div> </div>		
<p>Keterangan :</p> <p>Nama <i>Form</i> : F04</p> <p>Ukuran layer : Default Window Size</p> <p>Jenis Font : Arial</p> <p>Background : Blue</p>		

Gambar 3.18 Perancangan Antarmuka Petunjuk Penggunaan

b. Deskripsi objek

Berikut ini adalah deskripsi objek dari *form* petunjuk penggunaan

Tabel 3.11 Deskripsi Objek *form* Petunjuk Penggunaan

Objek	Jenis	Keterangan
Petunjuk penggunaan	Pdf	
Tutup	button	

5. Perancangan antarmuka tentang

a. Desain *Form* tentang

Berikut ini adalah tampilan untuk *form* tentang . *Form* ini berisi tentang pembuat aplikasi.

F05		<ul style="list-style-type: none"> Klik tutup untuk menutup <i>form</i>
<div style="border: 1px solid black; padding: 10px; text-align: center;"> <p>Voice Command</p> <p>Program untuk membuka Aplikasi pada komputer Anna Dara Andriana 10107275 UNIVERSITAS KOMPUTER INDONESIA 2007</p> <p>OK</p> </div>		
<p>Keterangan :</p> <p>Nama <i>Form</i> : F05</p> <p>Ukuran layer : Default Window Size</p> <p>Jenis Font : Arial</p> <p>Background : Blue</p>		

Gambar 3.19 Perancangan antarmuka Tentang

b. Deskripsi Objek

Berikut ini adalah deskripsi objek untuk tampilan tentang :

Tabel 3.12 Deskripsi Objek *form* Tentang

Objek	Jenis	Keterangan
Tentang	Memo	
Ok	button	

6. Perancangan antarmuka Pesan penyimpanan kata berhasil

a. Desain *Form* pesan pengisian kata kosong

Pesan ini akan muncul ketika user tidak mengisi kata pada form pengisian data. Berikut ini gambaran perancangan pesan :

The diagram shows a Windows Alert message box with the following components:

- Title Bar:** Labeled "M01".
- Message Area:** Contains the text "Kata tidak boleh kosong" (Word cannot be empty) and an "OK" button.
- Instructions:** A bullet point stating "Klik ok untuk menutup pesan" (Click ok to close the message).
- Notes Section:**
 - Keterangan :
 - Nama *Form* : M01
 - Ukuran layer : Windows Alert
 - Jenis Font : Arial
 - Background : Windows default

Gambar 3.20 Perancangan pesan pengisian kata kosong

b. Deskripsi Objek

Berikut ini adalah deskripsi objek dari tampilan pesan pengisian kata kosong:

Tabel 3.13 Deskripsi Objek *form* pesan penyimpanan kata kosong

Objek	Jenis	Keterangan
Ok	button	

7. Perancangan antarmuka Pesan pengisian data

a. Desain *Form* pesan pengisian data

Ketika user menyimpan data pada form pengisian data akan muncul pesan sebagai berikut:

M02		<ul style="list-style-type: none"> Klik ok untuk menyimpan data
<div style="border: 1px solid black; padding: 10px; text-align: center;"> <p>Anda Yakin Untuk Menyimpan Data?</p> <p>OK</p> </div>		
<p>Keterangan :</p> <p>Nama <i>Form</i> : M02</p> <p>Ukuran layer : Windows Alert</p> <p>Jenis Font : Arial</p> <p>Background : Windows default</p>		

Gambar 3.21 Perancangan pesan pengisian data

b. Deskripsi Objek

Berikut ini adalah deskripsi objek dari pesan pengisian data :

Tabel 3.14 Deskripsi Objek *form* penyimpanan suara berhasil

Objek	Jenis	Keterangan
Ok	button	

8. Perancangan antarmuka Pesan penyimpanan suara

1. Desain *Form* pesan penyimpanan suara

Pesan ini akan muncul ketika user menyimpan data . berikut ini adalah gambaran perancangan pesan penyimpanan suara berhasil :

M03		<ul style="list-style-type: none"> Klik ok untuk menutup pesan
<div style="border: 1px solid black; padding: 10px; text-align: center;"> <p>Suara.wav telah tersimpan</p> <div style="border: 1px solid black; padding: 5px; margin: 10px auto; width: 50px;">OK</div> </div>		
Keterangan : Nama <i>Form</i> : M03 Ukuran layer : Windows Alert Jenis Font : Arial Background : White		

Gambar 3.22 Perancangan pesan penyimpanan suara

2. Deskripsi Objek

Berikut ini adalah deskripsi objek dari pesan penyimpanan suara :

Tabel 3.15 Deskripsi Objek *form* pesan penyimpanan suara

Objek	Jenis	Keterangan
Ok	button	

9. Perancangan antarmuka Pesan keluar aplikasi

1. Desain *Form* pesan keluar aplikasi

Ketika user akan menutup aplikasi, sistem akan menampilkan pesan seperti berikut ini :

M04		<ul style="list-style-type: none"> • Klik ok untuk keluar dari aplikasi • Klik cancel untuk membatalkannya
<p>Yakin untuk keluar dari aplikasi?</p> <div> <input type="button" value="OK"/> <input type="button" value="Cancel"/> </div>		
<p>Keterangan :</p> <p>Nama <i>Form</i> : M04</p> <p>Ukuran layer : Windows Alert</p> <p>Jenis Font : Arial</p> <p>Background : White</p>		

Gambar 3.23 Perancangan pesan keluar aplikasi

2. Deskripsi Objek

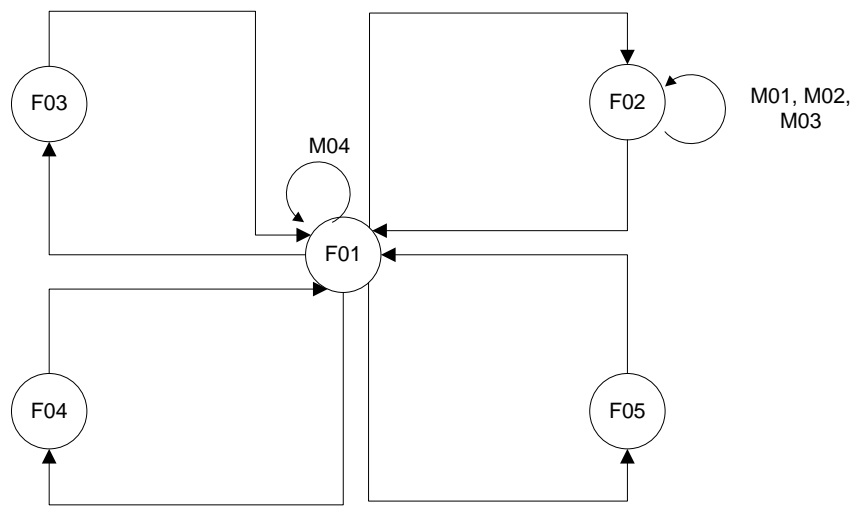
berikut ini adalah deskripsi objek dari pesan keluar aplikasi :

Tabel 3.16 Deskripsi Objek *form* pesan keluar aplikasi

Objek	Jenis	Keterangan
Ok	button	
Cancel	button	

3.3.2 Jaringan Semantik

Berikut ini adalah gambar jaringan semantik yang menggambarkan hubungan antar modul perangkat lunak *voice command* :



Gambar 3.24 Jaringan Semantik Aplikasi *Voice command*