BAB II
TINJAUAN PUSTAKA

II.1 Profil Instansi

Salah satu perwujudan dari pelaksanan Otonomi Daerah sebagaimana diatur dalam Peraturan Pemeritah Nomor 25 Tahun 2000 tentang Kewenangan Provinsi sebagai Daerah Otonom, maka telah ditetapkan Peraturan Daerah Nomor 15 Tahun 2000 tanggal 12 Desember 2000 tentang Dinas Daerah Provinsi Jawa Barat, salah satu diantaranya telah terbentuk Dinas Pariwisata dan Kebudayaan Provinsi Jawa Barat yang merupakan penggabungan dari 4 (empat) instansi yaitu eks Dinas Pariwisata Jawa Barat, Kanwil Pariwisata, Dinas Pendidikan Bagian Kebudayaan dan Kanwil Pendidikan Bagian Kebudayaan.

Setelah terbentuk Dinas Pariwisata dan Kebudayaan Pronvinsi Jawa Barat sebagaimana tertuang dalam Peraturan Daerah Nomor 15 Tahun 2000, ditetapkan bahwa Dinas Pariwisata dan Kebudayaan Provinsi Jawa Barat merupakan instansi teknis yang melaksanakan sebagai urusan pemerintahan dan pembangunaan di bidang Pariwisata dan Kebudayaan di Jawa Barat.

Selanjutnya dengan berdasarkan Peraturan Daerah Nomor 15 Tahun 2000, maka upaya meningkatkan kelancaran pelaksanaan tugas dinas telah diatur Keputusan Gubernur Jawa Barat Nomor 55 Tahun 2001 tanggal 4 Desember 2001. Kemudian pada tahun 2002, Pemerintah Daerah Provinsi Jawa Barat mengeluarkan Peraturan Daerah Provinsi Jawa Barat No. 5 tahun 2002 tentang perubahan atas Peraturan Daerah Provinsi Jawa Barat No.15 tahun 2000 tentang Dinas Daerah Propinsi Jawa Barat.

II.1.1 Visi dan Misi Instansi

II.1.1.1 Visi

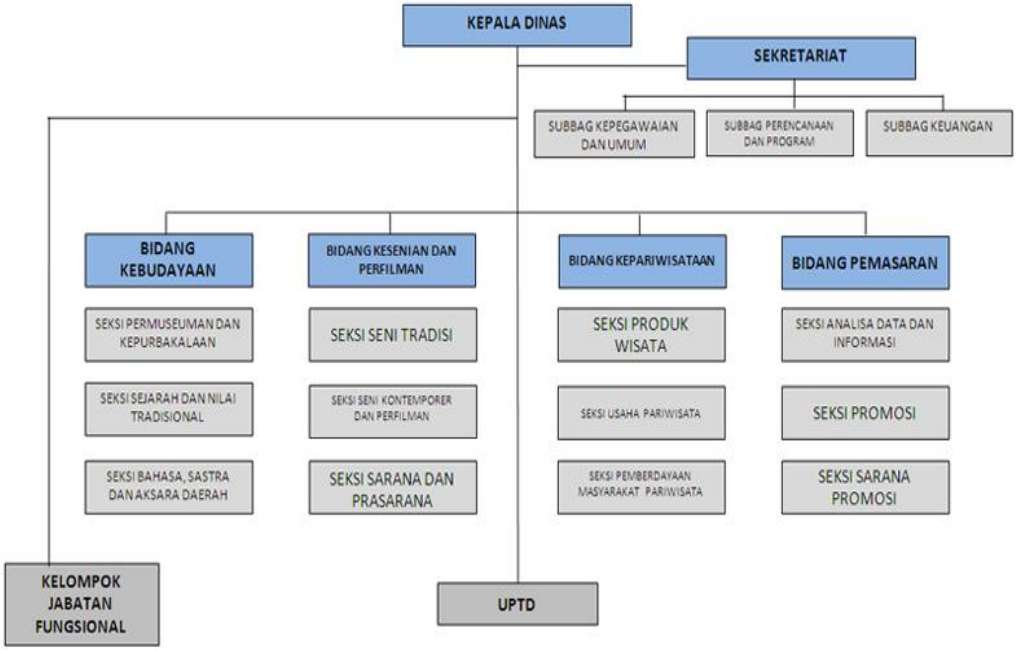
”Dinas Pariwisata dan Kebudayaan Provinsi Jawa Barat sebagai motor penggerak terwujudnya Jawa Barat sebagai daerah Budaya dan tujuan wisata andalan.”

II.1.1.2 Misi

1. Pembinaan, pelestarian dan pengembangan asset budaya yang mendukung upaya pengembangan Pariwisata Jawa Barat
2. Mengefektifkan kebudayaan sebagai asset daerah yang mendukung kepada pengembangan Usaha Jasa Pariwisata
3. Mempromosikan Kepariwisataan Jawa Barat
4. Meningkatkan sumber daya manusia Kebudayaan dan Kepariwisataan
5. Memuliakan nilai-nilai budaya yang terkandung dalam aspek jarahnitra dan tradisi Jawa Barat

II.1.2 Struktur Organisasi Instansi

Dinas Pariwisata dan Kebudayaan Provinsi Jawa Barat merupakan salah satu Lembaga Daerah di lingkungan Pemerintah Provinsi Jawa Barat, dimana terdapat struktur organisasinya seperti pada gambar II.1 dan gambar II.2.



Gambar II.1 Struktur Organisasi Dinas Pariwisata dan Kebudayaan Provinsi Jawa Barat.



Gambar II.2 Struktur Organisasi Dinas Pariwisata dan Kebudayaan Provinsi Jawa Barat.

II.1.3 Deskripsi Kerja Struktur Organisasi Instansi

Dalam melakukan penelitian tugas akhir ini, penelitian dilakukan pada bagian sub dinas bidang kesenian dan perfilman dan bidang kebudayaan. Berikut ini merupakan deskripsi kerja dari sub dinas bidang kesenian dan perfilman dan bidang kebudayaan:

a) BIDANG KESENIAN DAN PERFILMAN

Tugas :

Menyelenggarakan pengkajian bahan kebijakan teknis dan melestarikan, mengembangkan dan memanfaatkan kesenian meliputi seni tradisi, seni kontemporer dan perfilman, prasarana dan sarana kesenian.

Fungsi :

1. Penyelenggarakan pengkajian rencana dan program kesenian tradisional, seni kontemporer dan perfilman, serta prasarana dan sarana.
2. Penyelenggaraan pengkajian bahan petunjuk teknis pelestarian pengembangan dan pemanfaatan seni tradisional, seni kontemporer dan perfilman, serta prasarana dan sarana.
3. Penyelenggaraan pelestarian, pengembangan dan pemanfaatan seni tradisional, seni kontemporer dan perfilman, serta prasarana dan sarana.

4. Penyelenggaraan pengkajian bahan dan fasilitasi pelestarian, pengembangan dan pemanfaatan seni tradisional, seni kontemporer dan perfilman, serta prasarana dan sarana.
5. Penyelenggaraan pengkajian bahan dan fasilitasi hak atas kekayaan intelektual seni tradisional, seni kontemporer dan perfilman.

b) BIDANG KEBUDAYAAN

Tugas :

Menyelenggarakan pengkajian bahan kebijakan teknis, fasilitas dan penyelenggaraan pelestarian, pengembangan dan pemanfaatan kebudayaan meliputi permuseuman, peninggalan sejarah dan kepurbakalaan, kesejarahan, nilai tradisional dan kebahasaan daerah.

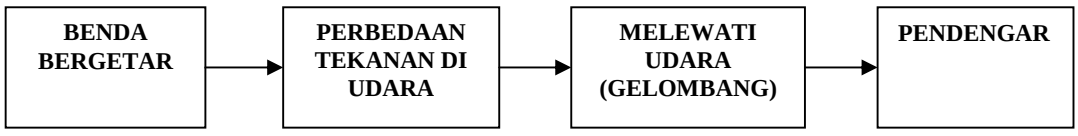
Fungsi :

1. Penyelenggaraan kebijakan teknis kebudayaan.
2. Penyelenggaraan pengkajian rencana dan program kebudayaan.
3. Penyelenggaraan pengkajian bahan pedoman dan petunjuk teknis pelestarian, pengembangan dan pemanfaatan kebudayaan.
4. Penyelenggaraan pengkajian bahan dan memfasilitasi pelestarian, pengembangan dan pemanfaatan kebudayaan.
5. Penyelenggaraan pelestarian, pemanfaatan dan pengembangan kebudayaan.

II.2 Landasan Teori

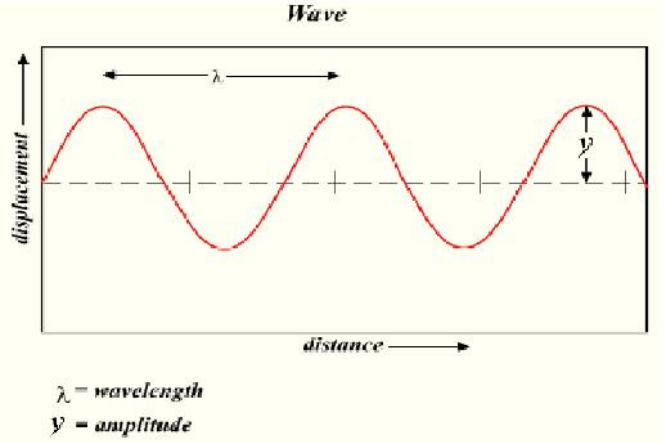
II.2.1 Suara

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 II.3 Skema suara

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 ferkuensi (f) dengan satuan m/dt (Hz).



Gambar II.4 Gelombang suara

II.2.2 Frekuensi

Frekuensi adalah jumlah gelombang yang terjadi setiap detik. Jenis-jenis frekuensi ada 4, sebagi 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 diluar jarak pendengaran manusia dapat dikatakan sebagai "*NOISE*" (getaran yang tidak teratur dan tidak berurutan dalam berbagai frekuensi, tidak dapat didengar manusia). Didalam istilah musik, *pitch* berarti sama seperti frekuensi.

II.2.3 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.

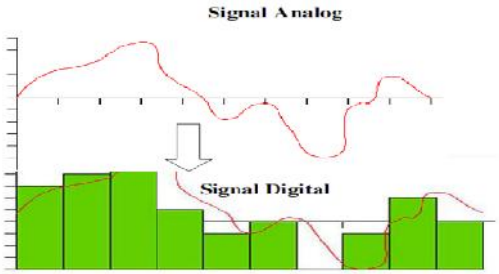
II.2.4 *Velocity*

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

II.2.5 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). Ukuran sampel 8 bit menyediakan 256 unit deskripsi jarak dinamis atau amplitudo (level suara dalam satu waktu).



Gambar II.5 Gelombang suara analog menjadi digital

II.2.6 *Wave form audio* (WAV)

WAV atau WAVE singkatan dari *Waveform Audio Format* merupakan standar format file *audio* untuk penyimpanan file audio dalam PC yang berbasis *Microsoft* dan IBM. WAV merupakan jenis dari format *bitstream* **RIFF** dimana datanya tersimpan di dalam "*Chunks*" dan file WAV pun kompatibel pada komputer berbasis Macintosh dan Amigo dimana menggunakan format **IFF** dan **AIFF**. Karena WAV berjalan dalam sistem Windows dan IBM maka semua data tersimpan dalam *Little-Endian*. Format RIFF bertindak sebagai "Pembungkus" pada berbagai jenis *codec* kompresi audio. Format ini adalah format utama dari *audio* dalam sistem Windows.

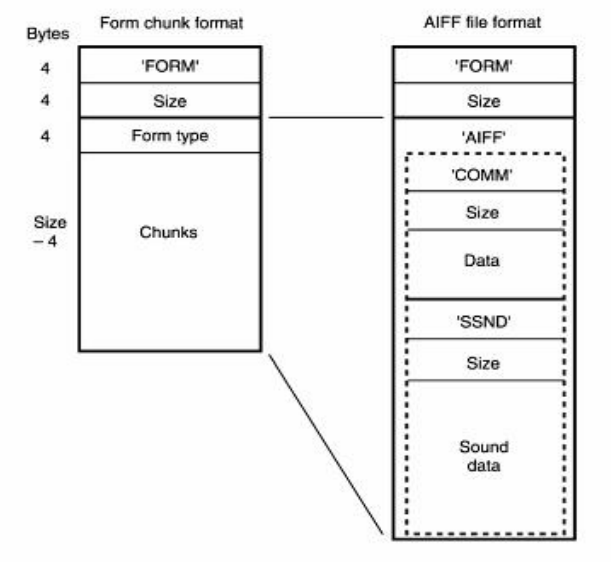
File WAV memiliki file yang tidak terkompres yang terdapat pada format *Pulse Code Modulation* (PCM). *Audio* PCM merupakan standar file *audio* dalam format CD pada 44.100 *samples* per detik, dan setiap satu *sample* memiliki 16 bit. Karena PCM tidak dimanfaatkan, karena sifatnya yang *lossless*, dimana menampung seluruh *track sample audio*, karena itu pengguna yang profesional atau peneliti audio menggunakan format WAV sebagai format audio dengan kualitas yang maksimal. WAV yang dapat diedit dan dimanipulasi dengan mudah menggunakan bantuan perangkat lunak.

II.2.6.1 Struktur File WAV

File WAV menggunakan struktur standar RIFF yang mengelompokkan isi file (sampel format, sampel audio digital, dan lain sebagainya) menjadi "*Chunk*" yang terpisah, setiap bagian mempunyai header dan *byte* data masing-masing. Header *Chunk* menetapkan jenis dan ukuran dari *byte* data *Chunk*. Dengan metoda pengaturan seperti ini maka program yang tidak mengenali jenis *Chunk* yang khusus dapat dengan mudah melewati bagian *Chunk* ini dan melanjutkan langkah memproses *Chunk* yang dikenalnya.

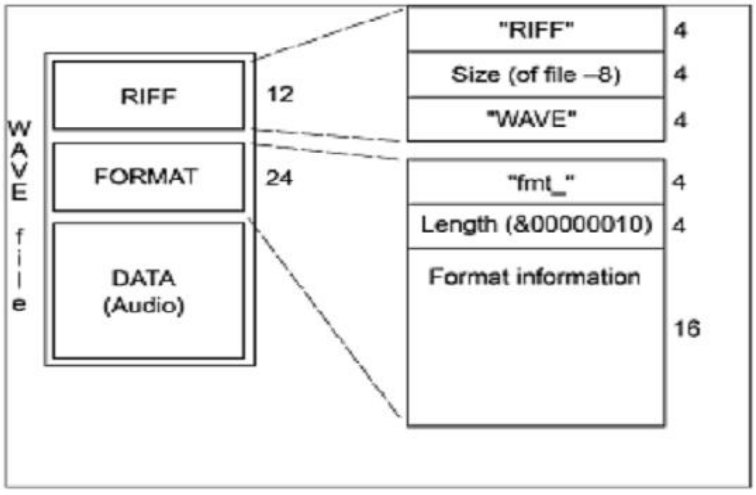
Jenis *Chunk* tertentu mungkin terdiri atas sub-*Chunk*. *Chunk* pada file RIFF merupakan suatu *string* yang harus diatur untuk tiap kata. Ini berarti ukuran total dari *Chunk* harus merupakan kelipatan dari 2 *byte* (seperti 2, 4, 6, 8 dan seterusnya). Jika suatu *Chunk* terdiri atas jumlah *byte* yang ganjil maka

harus dilakukan penambahan *byte* (extra padding *byte*) dengan menambahkan sebuah nilai nol pada *byte* data terakhir. Extra padding *byte* ini tidak ikut dihitung pada ukuran *Chunk*. Oleh karena itu sebuah program harus selalu melakukan pengaturan kata untuk menentukan ukuran nilai dari header sebuah *Chunk* untuk mengkalkulasi offset dari *Chunk* berikutnya. fomat file dari setiap isi audio *WAVE* dapat dilihat pada gambar II.6.



Gambar II.6 Format File WAVE

Pada gambar II.7, adalah struktur untuk setiap *Chunk* RIFF pada setiap file audio *WAVE*.



Gambar II.7 Struktur *Chunk* RIFF

II.2.6.2 *Header* File WAV

Header file WAV mengikuti struktur format file RIFF standar. Delapan *byte* pertama dalam file adalah *header chunk* RIFF standar yang mempunyai *chunk* ID "RIFF" dan ukuran *chunk* didapat dengan mengurangkan ukuran file dengan *8 byte* yang digunakan sebagai *header*. Empat *byte* data yaitu kata "RIFF" menunjukkan bahwa file tersebut merupakan file RIFF. File WAV selalu menggunakan kata "WAVE" untuk membedakannya dengan jenis file RIFF lainnya sekaligus digunakan untuk mendefinisikan bahwa file tersebut merupakan file audio *Waveform*.

Tabel II.1 Nilai Jenis *Chunk* RIFF

Offset	Ukuran	Deskripsi	Nilai
0x00	4	Chunk ID	"RIFF" (0x52494646)
0x04	4	Ukuran Data Chunk	(ukuran *file*) – 8
0x08	4	Jenis RIFF	"WAVE" (0x57415645)

II.2.6.3 *Chunk* File WAV

Ada beberapa jenis *chunk* untuk menyatakan file WAV. Kebanyakan file WAV hanya terdiri atas 2 buah *chunk*, yaitu *Chunk* Format dan *Chunk* Data. Dua jenis *chunk* ini diperlukan untuk menggambarkan format dari sampel digital audio. Meskipun tidak diperlukan untuk spesifikasi file WAV yang resmi, lebih baik menempatkan *Chunk* Format sebelum *Chunk* Data. Kebanyakan program membaca *chunk* tersebut dengan urutan di atas d an jauh lebih mudah dilakukan *streaming digital* audio dari sumber yang membacanya secara lambat dan linear seperti Internet.

Jika *Chunk* Format lebih dulu ditempatkan sebelum *Chunk* Data maka semua data dan format harus di-stream terlebih dahulu sebelum dilakukan *playback*. Format *Chunk* RIFF untuk file *WAVE* dapat dilihat seperti tabel II.2.

Tabel II.2 Format *Chunk* RIFF

Offset	Ukuran (byte)	Deskripsi
0x00	4	Chunk ID
0x04	4	Ukuran Data Chunk
0x08	Byte Data Chunk	

II.2.6.4 *Chunk* Format

Chunk format terdiri atas informasi tentang bagaimana suatu data *Waveform* disimpan dan cara untuk dimainkan kembali, termasuk jenis kompresi yang digunakan, jumlah kanal, laju pencuplikan (*sampling rate*), jumlah bit tiap sampel dan atribut lainnya. *Chunk* format ini ditandai dengan *chunk* ID "fmt ". Setiap *Chunk* RIFF audio *WAVE* juga memiliki nilai–nilai *Chunk* yang berbeda, pada tabel II.3 berikut adalah nilai – nilai *Chunk* Format pada file *WAVE*.

Tabel II.3 Nilai-Nilai *Chunk* Format File WAV

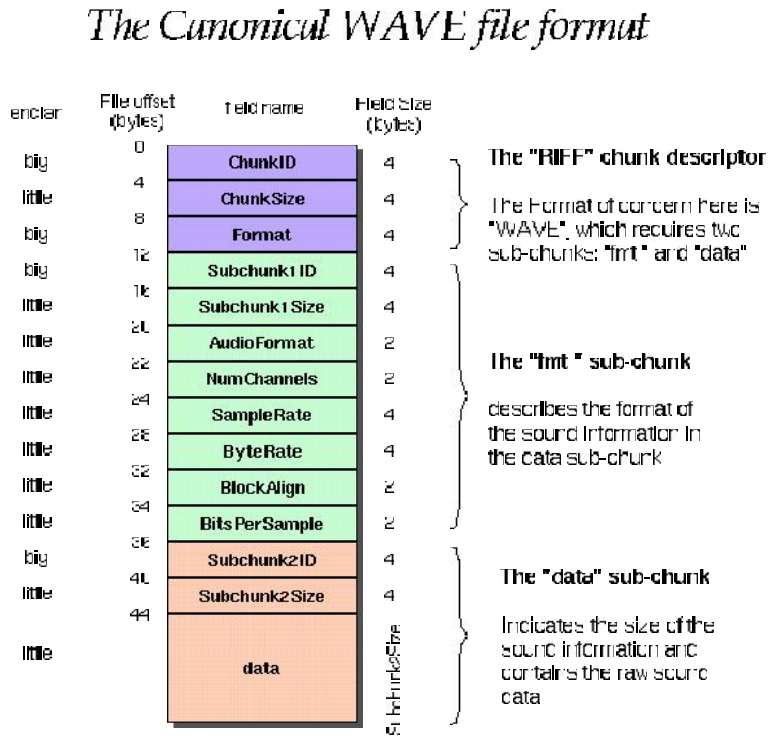
Offset	Ukuran	Deskripsi	Nilai
0x00	4	Chunk ID	"fmt" (0x666D7420)
0x04	4	Ukuran Data Chunk	16 + extra format bytes
0x08	2	Kode Kompresi	1 – 65,535
0x0a	2	Jumlah Kanal	1 – 65,535
0x0c	4	Laju Pencuplikan	1 – 0xFFFFFFFF
0x10	4	Jumlah rata-rata byte tiap detik	1 – 0xFFFFFFFF
0x14	2	Block align	1 – 65,535
0x16	2	Bit Significant tiap sample	2 – 65,535

II.2.6.5 *Chunk* Data

Chunk ini ditandai dengan adanya *string* "data". *Chunk* Data pada file WAV terdiri atas sampel digital audio yang mana dapat di-*decode* kembali menggunakan metode kompresi yang dinyatakan dalam *chunk* format WAV. Jika kode kompresinya adalah 1 (jenis PCM tidak terkompresi), maka "Data WAV" terdiri atas nilai sampel mentah (*raw sample value*).

II.2.6.6 Format WAV PCM

Jenis format WAV ini merupakan jenis file WAV yang paling umum dan hampir dikenal oleh setiap program. Format WAV PCM (*Pulse Code Modulation*) adalah file WAV yang tidak terkompresi, akibatnya ukuran file sangat besar jika file mempunyai durasi yang panjang. Untuk bentuk umum file audio wave mulai dari Chunk ID hingga data dapat dilihat pada gambar II.8.



Gambar II.8 Format File *Audio* WAVE

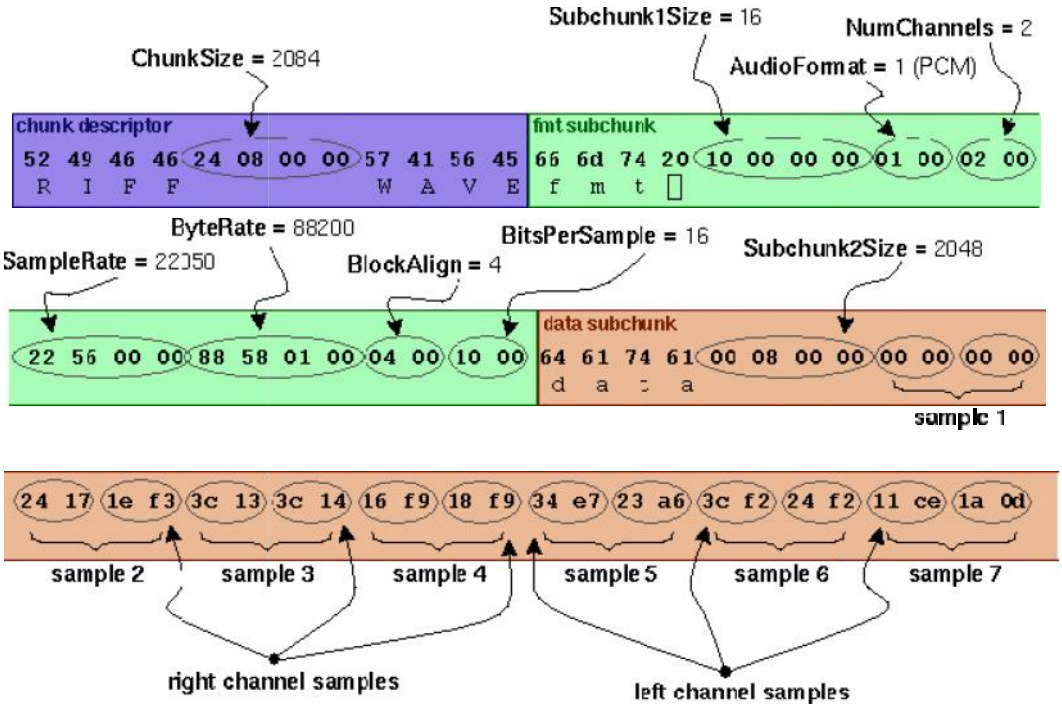
keterangan:

- : *Chunk* RIFF, format yang terdapat di sini adalah WAVE dimana terdiri dari sub-*Chunk* “fmt” dan “data”.
- : Sub *Chunk* “fmt” menggambarkan format dari informasi *sound* pada sub *Chunk* data
- : Sub *Chunk* “data” menandai ukuran dari informasi data dan berisi data *sound*

Sebagai contoh, disini kita ambil file *audio* wav 72 bit yang diubah kedalam *hexadecimal*.

```
52 49 46 46 24 08 00 00 57 41 56 45 66 6d 74 20 10 00 00 00 01 00 02 00
22 56 00 00 88 58 01 00 04 00 10 00 64 61 74 61 00 08 00 00 00 00 00 00
24 17 1e f3 3c 13 3c 14 16 f9 18 f9 34 e7 23 a6 3c f2 24 f2 11 ce 1a 0d
```

Penjelasan dari file audio wav 72 bit diatas untuk interpretasi dari setiap nilai *hexadecimal* dapat dilihat pada gambar II.9.



Gambar II.9 Intepretasi File Audio *wav*

II.2.7 *Sound Recognition*

Sound Recognition merupakan sebuah teknologi terapan yang mengubah sinyal-sinyal digital yang berasal dari bunyi menjadi instruksi atau aksi tertentu. Teknologi *sound recognition* mampu menganalisis sebuah masukan berupa sinyal digital ke dalam bentuk besaran frekuensi. Di dalam istilah musik, *pitch* / nada berarti sama seperti frekuensi. Frekuensi yang terdapat pada sinyal digital ini sangat bervariasi. Dapat dibedakan ada suara bernada tinggi, dan suara bernada rendah. Perbedaan ini disebabkan oleh *pitch* dari gelombang suara tersebut, semakin rapat gelombang, akan semakin tinggi nada suara yang dihasilkan.

Secara garis besar, cara kerja *sound recognition* ialah sinyal suara audio yang diterima melalui mikrofon diambil salah satu komponennya yaitu frekuensi. Frekuensi tersebut dapat direpresentasikan dalam bentuk spektrum. Dengan mentransformasikan sinyal dari domain waktu kedalam domain frekuensi, menggunakan rumus matematika Fourier, akan diperoleh sekumpulan informasi tentang frekuensi yang terkandung dalam sinyal tersebut. Pemisahan ini dapat mencirikan suatu objek suara, sehingga suara dapat dikenali.

II.2.8 *Fast Fourier Transform*

Perhitungan DFT secara langsung dalam komputerisasi dapat menyebabkan proses perhitungan yang sangat lama. Hal itu disebabkan karena dengan DFT, dibutuhkan N^2 perkalian bilanngan kompleks. Karena itu dibutuhkan cara lain untuk menghitung DFT dengan cepat. Hal itu dapat dilakukan dengan menggunakan algoritma *Fast Fourier Transform* (FFT) dimana FFT menghilangkan proses perhitungan yang kembar dalam DFT.

DFT dapat dituliskan sebagai berikut : [2]

$$X[k] = \sum_{k=0}^{N-1} x[j] W_N^{nk} \qquad (2.1)$$

Dimana $W = e^{-2\pi i/N}$ (2.2)

X[k] = representasi domain frekuensi dari deret waktu sinyal 'n'. Rumus ini menghasilkan satu bilangan kompleks (bilangan yang terdiri dari bilangan riil dan imajiner) H_n untuk setiap n.

N = jumlah sinyal yang akan diproses.

k = sampel (dalam domain waktu).

W = konstanta kompleks.

n = komponen frekuensi n; n = 0,1,2, ... ,N-1.

x[j] = nilai sampel sinyal.

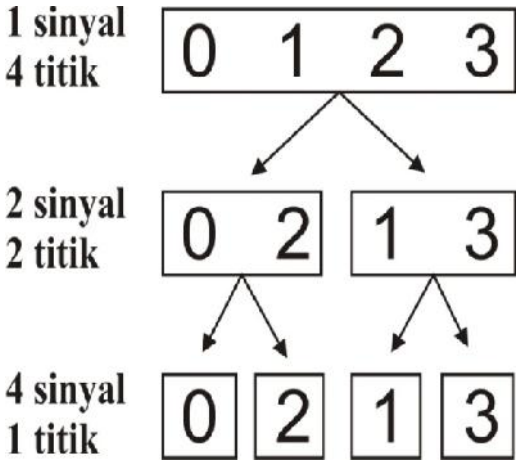
i = satuan imajiner.

Dengan kata lain, vektor dari $x[j]$ dikalikan dengan matriks yang lain (n,k) yang merupakan elemen konstan W untuk kelipatan n x k. Perkalian matriks menghasilkan hasil vector yang komponennya adalah $X[k]$. Matriks ini jelas membutuhkan perkalian matriks N^2 perkalian kompleks. Ditambah sejumlah operasi kecil untuk menghasilkan pangkat yang diperlukan W. Jadi, Tranformasi Fourier Diskrit tampaknya menjadi $O(N^2)$ proses. Namun sebenarnya Transformasi Fourier Diskrit dapat dihitung dalam operasi $O(N \log_2 N)$ dengan algoritma yang disebut Fast Fourier Transform.

Salah satu "penemuan kembali" dari FFT, yaitu Danielson dan Lanczos pada tahun 1942, menyediakan salah satu turunan paling jelas dari algoritma fft. Pada prinsipnya algoritma ini adalah memecah *N*-titik menjadi dua (*N*/2)-titik,

kemudian memecah tiap (N/2)-titik menjadi dua (N/4)-titik, begitu seterusnya sampai hanya terdapat 1 titik.

Misalkan sinyal x[n] terdiri dari N-titik. Kita akan memecah (desimasi) sinyal ini menjadi dua bagian yang masing-masing terdiri dari (N/2)-titik, yaitu satu adalah kumpulan dari nilai-nilai berindeks genap dan satu kumpulan lagi adalah kumpulan dari nilai-nilai berindeks ganjil. Demikian seterusnya. Contoh untuk desimasi 4 titik diperlihatkan pada gambar II.10. Jika terdapat N titik maka akan menghasilkan $^{2}\log N$ tingkat sampai mendapat 1 titik. Untuk N = 4, berarti memerlukan $^{2}\log 4$ = 2 tingkat, untuk N = 512 memerlukan 7 tingkat, untuk N = 4096 memerlukan 12 tingkat, dan seterusnya.



Gambar II.10 Desimasi untuk 4 titik

Jika telah melakukan desimasi seperti gambar II.10, tidak ada salahnya melakukan proses yang lebih mudah. Cara lain untuk mendapatkan desimasi sampai 1 titik adalah dengan melakukan pembalikan bit (*bit reversal*), seperti gambar. Untuk 4 titik, berarti 1 desimal direpresentasikan dengan 2 bit. Desimal, pertama-tama, diurutkan mulai dari desimal 0 sampai desimal 3. Kemudian, dilakukan representasi bit untuk tiap-tiap desimal tersebut. Setelah itu, bit-bit tersebut dibalik (yaitu ditulis kembali namun dengan urutan dari kanan ke kiri). Langkah terakhir adalah merepresentasikan kembali bit-bit tersebut kedalam desimal. Hasil akhir pada gambar II.10 sama dengan hasil akhir pada gambar II.11. [2]

Tabel II.4 Pembalikan Bit untuk 4 Titik

Urutan normal		Urutan setelah pembalikan	
Desimal	Biner	Desimal	Biner
0	00	0	00
1	01	2	10
2	10	1	01
3	11	3	11

Konsep perhitungan FFT sering disebut dengan metode kupu-kupu (*Butterfly method*)

Diberikan : {x[0],x[1],x[2],..,x[N-1]}

Indeks genap : {x[0],x[2],x[4],..,x[N-2]}

Indeks ganjil : {x[1],x[3],x[5],..,x[N-1]}

Kita telah mengetahui DFT untuk x[j] seperti persamaan (2.1) dan kita akan menuliskan kembali persamaan tersebut karena kita telah memecah menjadi dua bagian, yaitu bagian genap dan ganjil, sebagai berikut:

Dapat dibuktikan pada persamaan berikut:

$$X[k] = \sum_{j=0}^{N-1} e^{\frac{2\pi ijk}{N}} x[j]$$

$$= \sum_{j=0}^{N/2-1} e^{\frac{2\pi ijk/N}{N}} x[2j] + \sum_{j=0}^{N/2-1} e^{\frac{2\pi ik(2j+1)/N}{N}} x[2j + 1] \qquad (2.3)$$

$$= \sum_{j=0}^{N/2-1} e^{\frac{2\pi ikj}{N/2}} x[2j] + W_N^{'k} \sum_{j=0}^{N/2-1} e^{\frac{2\pi ikj}{N/2}} x[2j+1]$$

$$= \sum_{j=0}^{N/2-1} x[2j]\, W_{N/2}^{'kj} + W_N^{'k} \sum_{j=0}^{N/2-1} x[2j+1]\, W_{N/2}^{'kj}$$

X[k] : representasi domain frekuensi dari deret waktu sinyal 'n'. Rumus ini menghasilkan satu bilangan kompleks (bilangan yang terdiri dari bilangan riil dan imajiner) H_n untuk setiap n.

W : konstanta kompleks

N : jumlah sinyal yang akan diproses

k : frekuensi komponen k; k = 0,1, ... N-1

x : deret waktu sinyal (data)

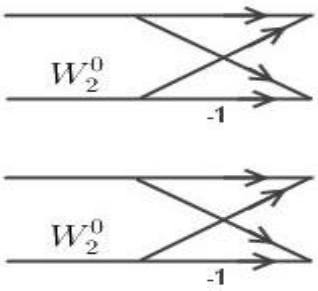
j : sampel (dalam domain waktu)

i : satuan imajiner

Rumus diatas merupakan rumus untuk memecah menjadi (N/2)-titik dan faktor W_N^{rk} menjadi bobot (*weight*) untuk mendapatkan hasil *N*-titik. Proses tersebut, untuk *N* = 4 dapat digambarkan seperti gambar II.13.

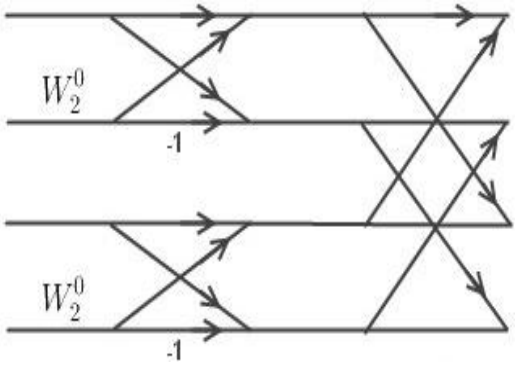
Gambar II.13 dapat dimengerti dengan melihat struktur dasar seperti ditunjukan pada gambar II.14. Setiap variable yang melewati tanda panah sama dengan variable tersebut dikalikan dengan bobot yang ditunjukan pada tanda panah tersebut dan variable pada pertemuan dua ujung panah adalah penjumlahan berbobot (*weighted sum*) variabel-variabel pada panah tersebut. Berikut contoh langkah-langkah menyelesaikan metode *butterfly* dengan 4-titik.

1. Buat 2 buah 2-titik *butterfly*



Gambar II.11 Dua buah 2-titik *butterfly*

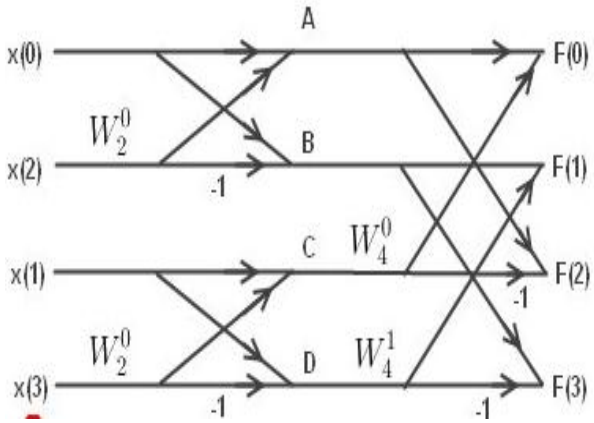
2. Selanjutnya perpanjang garis dan hubungkan bagian atas ke bawah dan bagian atas kebawah *butterfly*.



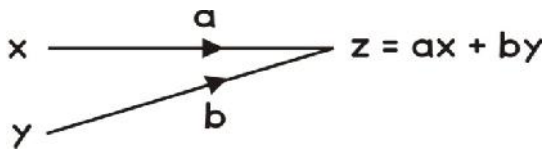
Tahap 1 Tahap 2

Gambar II.12 Penggabungan 2-titik *butterfly*

3. Berikan label untuk nilai input dan output. Beri label pada bagian bawah diagram dengan W. Tahap 1 memiliki basis W_2, dan tahap 2 memiliki basis W_4. Hal ini akan terus berlanjut dalam mode biner 2,4,8,16 untuk tahap-tahap selanjutnya. Nilai masukan berupa hasil dari pembalikan bit (*bit reversal*).



Gambar II.13 Metode Butterfly 4-titik



Gambar II.14 Struktur dasar metode *butterfly*

4. Keluaran dari metode *butterfly* dapat dilihat pada persamaan berikut

   Tahap 1:

$$A = x(0) + W_2^{r0}\, x(2)$$
$$B = x(0) - W_2^{r0}\, x(2)$$
$$C = x(1) + W_2^{r0}\, x(3)$$
$$D = x(1) - W_2^{r0}\, x(3)$$

   Tahap 2:

$$X(0) = A + W_4^{r0}\, C$$
$$X(1) = B + W_4^{r1}\, D$$
$$X(2) = A - W_4^{r0}\, C$$

$$X(3) = B - W_4^1 D$$

Subtitusi kembali nilai A, B, C, D:

$$\begin{aligned}
X(0) &= x(0) + W_2^0 x(2) + W_4^0\left(x(1) + W_2^0 x(3)\right) \\
&= x(0) + W_2^0 x(2) + W_4^0 x(1) + W_4^0 W_2^0 x(3) \\
X(1) &= x(0) - W_2^0 x(2) + W_4^1\left(x(1) - W_2^0 x(3)\right) \\
&= x(0) - W_2^0 x(2) + W_4^1 x(1) - W_4^1 W_2^0 x(3) \\
X(2) &= x(0) + W_2^0 x(2) - W_4^0\left(x(1) + W_2^0 x(3)\right) \\
&= x(0) + W_2^0 x(2) - W_4^0 x(1) - W_4^0 W_2^0 x(3) \\
X(3) &= x(0) - W_2^0 x(2) - W_4^1\left(x(1) + W_2^0 x(3)\right) \\
&= x(0) - W_2^0 x(2) - W_4^1 x(1) - W_4^1 W_2^0 x(3)
\end{aligned}$$

W adalah konstanta kompleks yang sama seperti dalam (2.2), X_k^e menunjukkan komponen k dari Transformasi Fourier dengan panjang *N*/2 yang terbentuk dari komponen x_k asli adalah transformasi yang sesuai dengan panjang *N*/2 yang terbentuk dari komponen ganjil (odd). Perhatikan juga bahwa k pada baris terakhir (2.3) bervariasi dari 0 sampai *N*, bukan hanya untuk *N*/2. Namun demikian, transformasi X_k^e dan X_k^o yang periodik di *k* dengan panjang *N* /2. Jadi setiap perulangan melalui dua siklus untuk mendapatkan X_k.

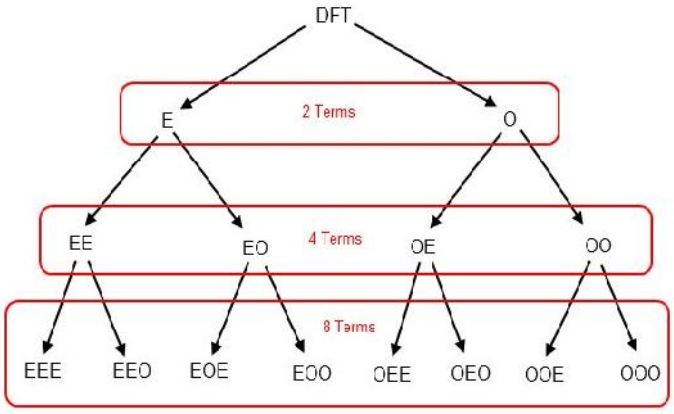
Hal yang menakjubkan mengenai *Danielson-Lanczos Lemma* adalah dapat digunakan secara rekursif. Setelah mengurangi masalah komputasi X_k dengan komputasi X_k^e dan X_k^o, kita bisa melakukan pengurangan yang sama dari X_k^e untuk masalah perhitungan transformasi dari *N*/4 komponen genap data masukan dan *N*/4 data komponen ganjil. Dengan kata lain, kita dapat mendefinisikan X_k^e dan X_k^o menjadi Transformasi Diskrit dari titik-titik yang genap-genap (even-even) dan genap-ganjil (even-odd) dari pembagian data yang berurutan. Meskipun ada cara lain untuk menangani masalah ini, sejauh ini hal termudah adalah mendefinisikan nilai asli N dengan integer kelipatan 2. Bahkan, penggunaan FFT direkomendasikan hanya dengan N bilangan kelipatan 2. Dengan pembatasan pada *N* ini, jelas bahwa kita dapat terus menerapkan *Danielson-Lanczos Lemma* sampai kita memperoleh pembagian semua data ke Transformasi Fourier dengan panjang 1 yang merupakan operasi identitas yang salinan satu

nomor input ke dalam satu slot output. Dengan kata lain, untuk setiap pola $\log_2 N$ e (ganjil) dan o (genap), terdapat transformasi satu titik yang merupakan satu dari angka masukan x_n.

$$X_k^{eoeeeoeo\ldots oeo} = x_n \quad \text{untuk beberapa n} \qquad (2.4)$$

(Tentu saja transformasi satu titik sebenarnya tidak tergantung pada *k*, karena periodik dalam periode k dengan periode 1)

Cara berikutnya adalah mencari tahu nilai n mana yang berkorespondensi dengan pola e dan o dalam persamaan (2.4). Jawabannya ialah dengan cara membalik pola e dan o, maka biarkan e = 0 and o = 1, dan akan didapatkan nilai n dalam bentuk biner. Hal ini dapat dilakukan karena pembagian urutan data menjadi genap dan ganjil merupakan pengujian dari bit orde rendah (paling signifikan) yang berurutan dari n. Gagasan pembalikan bit dapat dimanfaatkan dalam cara yang sangat cerdas yang, bersama dengan Danielson-Lanczos.



Gambar II.15 Pengubahan urutan array (di sini dengan panjang 8) dengan pembalikan bit, (a) antara dua array, dibandingkan (b) di tempat Bit penataan kembali pembalikan adalah bagian penting dari algoritma Fast Fourier Transform (FFT).

Lemma, membuat FFT lebih praktis, sebagai contoh jika menggunakan vektor asli data f dan mengatur ulang ke dalam urutan bit-terbalik (lihat Gambar II.15), sehingga angka-angka individu tidak berada di urutan j, tetapi dari jumlah yang diperoleh dari pembalikan bit j. Sehingga pencatatan pada aplikasi secara rekursif dari Danielson-Lanczos Lemma menjadi sangat sederhana.

Pada algoritma ini menggabungkan pasangan berdekatan dengan mendapatkan dua titik transformasi, kemudian menggabungkan pasangan berdekatan untuk mendapatkan 4-titik transformasi, dan sebagainya, sampai bagian pertama dan kedua dari himpunan seluruh data digabungkan menjadi transformasi akhir. Setiap kombinasi menggunakan urutan operasi *N*, yang menunjukan kombinasi $N \log_2 N$, sehingga seluruh algoritma merupakan urutan $N \log_2 N$ (dengan asumsi, bahwa proses pengurutan bit tidak lebih besar dari $N \log_2 N$).

Struktur algoritma FFT ini, memiliki dua bagian. Bagian pertama yaitu mengubah urutan data kedalam bit-terbalik. Untungnya hal ini tidak membutuhkan penyimpanan tambahan, karena hanya melibatkan pasangan swapping elemen. (Jika k_1 adalah kebalikan bit k_2, maka k_2 adalah kebalikan bit k_1). Bagian kedua memiliki perulangan luar yang dieksekusi sebanyak $\log_2 N$ kali, pada gilirannya, mengubah panjang 2, 4, 8, ... , N. Untuk setiap tahap dari proses ini, terdapat dua loop bersarang dalam rentang subtransforms yang telah dihitung dan elemen untuk setiap transformasi dari implementasi Danielson-Lanczos Lemma. Operasi ini dibuat lebih efisien dengan membatasi panggilan eksternal untuk trigonometri sinus dan cosinus ke loop luar, dengan jumlah perhitungan sebanyak $\log_2 N$ kali. Perhitungan sinus dan cosinus dari berbagai sudut adalah melalui hubungan perulangan sederhana dalam loop bagian dalam.

Jumlah input bilangan kompleks dinyatakan dalam (nn), array data (data[1..2*nn]), dan isign, yang harus ditetapkan baik ± 1 dan merupakan tanda i dalam eksponensial persamaan (2.3).

Ketika isign di set dengan -1, Rutinitas FFT akan menghitung invers transform (2.4) kecuali bahwa hal itu tidak dikalikan dengan faktor normalisasi 1/*N* yang muncul dalam persamaan tersebut.

$$x_k = \frac{1}{N} \sum_{n=0}^{N-1} X_n \, e^{2\pi i k n / N} \qquad (2.4)$$

Perhatikan bahwa argumen nn adalah jumlah poin data yang kompleks. Panjang sebenarnya dari array nyata (data [1 .. 2 * nn]) adalah 2 kali nn, dengan setiap nilai kompleks menempati dua lokasi berurutan. Dengan kata lain, data[1]

adalah bagian nyata dari f_0, Data[2] adalah bagian imajiner dari f_0, dan seterusnya sampai data [2*nn-1], yang merupakan bagian nyata dari f_{N-1}, dan data [2*nn], yang merupakan bagian imajiner f_{N-1}. Rutinitas FFT memberikan kembali F_n yang dikemas dalam dalam nn bilangan kompleks.

Bagian nyata dan imajiner bagian dari komponen frekuensi nol F_0 dalam data[1] dan data[2]; frekuensi positif terkecil nol memiliki bagian real dan imajiner dalam data[3] dan data[4]; frekuensi terkecil (dalam amplitudo) negatif bukan nol memiliki bagian real dan imajiner dalam data[2*nn-1 dan data[2*nn]. Peningkatan amplitudo frekuensi positif disimpan dalam pasangan real-imajiner data[5], data[6] hingga data[nn-1], data [nn]. Peningkatan amplitudo frekuensi negatif disimpan dalam data[2 * nn-3], data[2*nn-2] disimpan dalam data[nn +3], data[nn +4]. Akhirnya, pasangan data[nn +1], data[nn +2] mengandung bagian real dan imajiner dari satu titik alias yang berisi frekuensi yang paling positif dan frekuensi yang paling negatif. [8]

```
#include <math.h> #define SWAP(a,b) tempr=(a);(a)=(b);(b)=tempr
void four1(float data[], unsigned long nn, int isign)
{
unsigned long n,mmax,m,j,istep,i;
double wtemp,wr,wpr,wpi,wi,theta;
float tempr,tempi;
n=nn << 1;
j=1;
for (i=1;i<n;i+=2) {                    //bagian pembalikan bit.
        if (j > i) {
        SWAP(data[j],data[i]);          //Menukar dua bilangan kompleks.
        SWAP(data[j+1],data[i+1]);
        } m=nn;
        while (m >= 2 && j > m) {
        j-=m;
        m >>= 1;
        }
        j+=m;
}
//Inilah bagian awal dari rutinitas algoritma FFT Danielson-Lanczos.
mmax=2;
while (n > mmax) {      //perulangan luar yang dieksekusi selama log2 nn  kali.
        istep=mmax << 1;
theta=isign*(6.28318530717959/mmax);            //inisialisasi perulangan trigonometri.
wtemp=sin(0.5*theta);
 wpr = -2.0*wtemp*wtemp;
        wpi=sin(theta);
        wr=1.0;
        wi=0.0;
        for (m=1;m<mmax;m+=2) { //Berikut adalah dua loop bersarang dalam.
```

```
                    for (i=m;i<=n;i+=istep) {
                        j=i+mmax;                    // formula Danielson-Lanczos:
                        tempr=wr*data[j]-wi*data[j+1];
                        tempi=wr*data[j+1]+wi*data[j];
                        data[j]=data[i]-tempr;
                        data[j+1]=data[i+1]-tempi;
                        data[i] += tempr;
                        data[i+1] += tempi;
                    }
                    wr=(wtemp=wr)*wpr-wi*wpi+wr;     //perulangan trigonometri
                    wi=wi*wpr+wtemp*wpi+wi;
                }
                mmax=istep;
            }
        }
```

II.2.9 Kecapi

Kecapi merupakan alat musik tradisional yang berasal dari daerah Jawa Barat. Kecapi merupakan salah satu jenis instrumen kordofon. Kordofon berasal dari bahasa inggris *chordophone*, *chord* berarti dawai, senar, atau tali; sedangkan *phone* berarti bunyi. Sebagaimana dijelaskan pada Ensiklopedi Musik bahwa: "Kecapi berupa instrumen yang masuk dalam keluarga kordofon, yaitu instrumen yang menggunakan dawai, dawai itu dikencangkan untuk menghasilkan ragam bunyi yang tentu dan pasti". [1]



Gambar II.16 Alat Musik Kecapi

Sebagai karya cipta budaya sunda yang lahir di Jawa Barat yang dahulu dikenal dengan sebutan Kerajaan Sunda, Kecapi merupakan alat musik tradisional Sunda yang sudah dikenal sejak jaman kerajaan pajajaran yaitu pada pemerintahan Sri Baduga Maharaja Nyakrawati yaitu pada tahun 1482-1521M.

Di wilayah budaya Sunda dikenal dengan nama *kacapi*. Terdapat beberapa jenis *kacapi*, yang bentuk dan ukurannya berbeda-beda. Yang paling banyak dikenal adalah *kacapi indung* atau *kacapi gelung*. Kacapi jenis ini yang biasa dipakai dalam ensambel *tembang Sunda*. Kacapi gelung juga yang paling besar di

antara *kacapi* Sunda, panjangnya sekitar 1,5 meter, lebar sekitar 30 cm, dan tinggi sekitar 40 cm. Jumlah kawatnya 18, yang dilaras dalam 5-nada (*pentatonik*) yaitu Da, Mi, Na, Ti, La, sehingga ranah nadanya hampir 4 oktaf.

Sekitar tahun 1950-an ada seniman terkenal, Mang Koko, yang membuat *kacapi* sedikit lebih lebar, hingga dawainya berjumlah 20. *Kacapi* ini kadang disebut *kacapi* modern, kadang disebut *kacapi wanda anyar* ("wajah baru"). *Kacapi* ini kemudian sangat populer, terutama karena diajarkan di sekolah kesenian yang pertama di Jawa Barat, Konservatori Karawitan, di mana Mang Koko adalah juga direkturnya (yang kedua, setelah *Machyar Kusumadinata*).[6]

II.2.9.1 Bentuk Kecapi

Bentuk kecapi dapat dibedakan menjadi dua yaitu:

1. Kecapi *parahu*



Gambar II.17 Kecapi *parahu*

Suatu kotak resonansi yang bagian bawahnya diberi lubang resonansi untuk memungkinkan suara keluar. Sisi-sisi jenis *kacapi* ini dibentuk sedemikian rupa sehingga menyerupai perahu. Di masa lalu, kacapi ini dibuat langsung dari bongkahan kayu dengan memahatnya.

2. Kecapi *sitter*



Gambar II.18 Kecapi *sitter*

Merupakan kotak resonansi dengan bidang rata yang sejajar. Serupa dengan kacapi parahu, lubangnya ditempatkan pada bagian bawah. Sisi bagian atas dan bawahnya membentuk trapesium.

II.2.9.2 Fungsi Kecapi

Menurut fungsinya dalam mengiringi musik, kacapi dapat dimainkan sebagai:



Gambar II.19 *Kacapi indung* dan *kacapi rincik*

1. *Kacapi indung* atau *kacapi induk*

   Kacapi indung memimpin musik dengan cara memberikan intro, *bridges*, dan *interlude*, juga menentukan tempo. Untuk tujuan ini, digunakan sebuah *kacapi* besar dengan 18 atau 20 dawai.

2. *Kacapi rincik atau kacapi anak*

   Kacapi rincik memperkaya iringan musik dengan cara mengisi ruang antar nada dengan frekuensi-frekuensi tinggi, khususnya dalam lagu-lagu yang bermetrum tetap seperti dalam *kacapi suling* atau *Sekar Panambih*. Jumlah dawai(senar) yang digunakan untuk kacapi rincik pada umumnya berjumlah lima belas. Pada kacapi rincik jangkauan suara dari dawai lebih tinggi satu oktaf dari *kacapi indung*.

II.2.9.3 Teknik Memainkan Kecapi

Pada Waditra tradisional sunda seperti halnya instrumen musik barat, sudah dipastikan sama-sama memiliki tatacara dalam memainkannya, baik dalam instrumental maupun iringan lagu. Begitu pula halnya dengan instrumen kecapi yang biasa digunakan di dalam sajian musik/karawitan Sunda, memiliki tatacara yang biasa dilakukan di dalam membunyikannya. Dalam karawitan Sunda istilah tatacara membunyikan sebuah instrumen musik, dikenal dengan istilah teknik. Teknik petikan yang biasa digunakan dalam memainkan

instrument kecapi pada umumnya terdapat 3 macam yaitu *disintreuk-ditoel, dijeungkalan,* dan *beulit kacang.* Teknik permainan tersebut menggunakan 2 sampai 4 jari yaitu 2 jari untuk tangan kiri (ibu jari dan telunjuk) dan 2 jari untuk tangan kanan. Namun terdapat berbaga perkembangan dalam permainan kecapi yang dilakukan oleh seniman diantaranya yaitu oleh Mang Koko.

Dengan adanya eksplorasi yang dilakukan oleh Mang Koko, maka teknik petian yang tadinya hanya menggunakan 4 jari, kini bisa dioptimalkan menjadi 8 jari. 8 jari tersebut terdiri atas 4 jari tangan kiri (ibu jari, telunjuk, jari tengah dan jari manis) dan 4 jari pada tangan kanan (ibu jari, telunjuk, jari tengah dan jari manis). Dari pengoptimalan jari yang dilakukan oleh Mang Koko, terdapat beberapa perkembangan teknik permainan kecapi. Perkembangan yang dilakukan oleh Mang Koko yaitu dengan menambah 2 teknk permainan kecapi, *diranggeum* dan *dijambret*. Berikut beberapa macam teknik memainkan kecapi:

1. Teknik *sintreuk-toel* (kanan *disintreuk* - kiri *ditoel*)

   Sintreuk-toel adalah teknik petikan kacapi dengan menggunakan dua jari yaitu telunjuk kanan dan telunjuk kiri. Posisi dan gerakan jarinya yaitu: satu telunjuk kanan melipat ke daiam, ujung kukunya menyentuh senar dengan gerakan nyintreuk (menjentik); dan dua telunjuk kiri agak lengkung ke bawah, ujung kukunya menyentuh senar dengan gerakan noel (sentuhan dengan ujung jari), sehingga gerakan dari kedua jari itu menghasilkan komposisi nada (gending) yang diinginkan. Gerakan tersebut ada yang searah dalam nada *gembyang* (oktaf) atau *kempyung* (akor), ada yang berlawanan dengan nada yang berlainan, dan ada pula yang seperti saling bersahutan antara telunjuk kanan dan kiri. Fungsi dari masing-masing jari di atas ada yang sama-sama sebagai penyaji melodi, ada pula yang telunjuk kanan sebagai penyaji melodi serta telunjuk kiri sebagai penyaji bass dan lain-lain. Artinya tergantung pada kebutuhan musiknya.

2. Teknik *dijeungkalan*

   Nama dari teknik *dijeungkalan* diambil karen posisi jari (ibu jari dan telunjuk) seperti sedang menjengkal.

3. Teknik *beulit kacang*

   Disebut beulit kacang karena posisi jari dari tangan kiri berbelit-belit. Teknik ini merupakan pengembangan dari teknik *dijeungkalan* dan *ditoel*.

4. Teknik *diranggeum*

   Teknik diranggeum merupakan pengembangan dari teknik penjarian dijengkalan. Nama dari teknik *diranggeum* diambil karena posisi jari semacam *ngaranggeum* (mengambil sesuatu dengan lima jari).

5. Teknik *dijambret*

   Pada teknik *dijambret* ini, petikan kacapi yang posisi dan gerakan jarinya terutama jari-jari tangan kanan, seperti menjambret yaitu membunyikan tiga buah nada secara bersamaan, dengan menggunakan ibu jari, telunjuk, dan jari tengah. Sedangkan posisi dan gerakan tangan kiri (ibu jari dan telunjuk) seperti *ngajeungkalan*. Fungsi dari kedua tangan tersebut masing-masing sebagai penyaji iringan (tangan kanan) dan penyaji bass (tangan kiri). Teknik dijambret biasanya digunakan untuk mengiringi lagu-lagu Sunda yang berirama mars (tempo cepat).

II.2.9.4 Tangga Nada

Dalam khasanah musik tradisional dan non-tradisional dikenal istilah tangga nada. Tangga nada merupakan hasil perpaduan atau susunan nada-nada. Bentuk tangga nada antara musik satu dengan musik lainnya, tentunya memiliki perbedaan dan persamaan. Tangga nada yang banyak digunakan, yaitu tangga nada pentatonis dan diatonic. Tangga nada pentatonis, yaitu tangga nada yang terdiri atas lima susunan nada. Tangga nada pentatonis banyak digunakan pada musik tradisional. Adapun tangga nada diatonic, yaitu tangga nada yang memiliki deretan nada yang berjarak 1 dan $\frac{1}{2}$. Tangga nada ini terdiri atas tangga nada *mayor* dan *minor*. Tangga nada diatonis *mayor* memiliki daretan nada yang berjarak 1,1, $\frac{1}{2}$,1,1,1, $\frac{1}{2}$. Adapun tangga nada diatonis *minor* memiliki jarak nada 1, $\frac{1}{2}$,1,1, $\frac{1}{2}$,1,1.

Pada musik daerah di Indonesia, digunakan beberapa jenis tangga nada atau didalam pengertian Karawitan Sunda disebut dengan Laras. Setiap tangga nada memiliki jumlah nada dan karakter yang berbeda-beda. Di Jawa Barat ada tangga nada salendro, pelog degung, dan madenda, di Jawa Tengah terdapat tangga nada salendro dan pelog.

1. Tangga Nada Salendro

   Tangga nada salendro terdiri atas lima susunan nada. Jarak antara nada yang satu dengan nada lainnya hampir sama sehingga tangga nada ini biasa disebut *salendro padantara*. Artinya, memiliki jarak interval yang sama. Tangga nada ini dapat digambarkan seperti berikut.

   1 . . 2 . . 3 . . 4 . . 5 . . 1

   Da Mi Na Ti La Da

   Tangga nada salendro biasanya digunakan pada musik daerah Jawa Barat. Jawa Tengah, dan Jawa Timur.

2. Tangga Nada Pelog Degung

   Tangga nada Pelog Degung terdiri atas lima susunan nada. Pada khasanah musik Jawa Barat tangga nada pelog degung ini lebih banyak digunakan.

   1 2 . . 3 . . . . 4 5 1

   Da Mi Na Ti La Da

3. Tangga Nada Madenda

   Tangga nada pelog degung dan tangga nada madenda jika dibandingkan terlihat adanya perbedaan, yaitu terletak pada interval beberapa nada. Untuk lebih jelasnya perhatikan gambar berikut.

   5 4 . . 3 . . . . 2 1 . . . . . 5 pelog degung

   5 . . 4 3 . . . . 2 1 . . . . . 5 medenda

   Tangga nada ini biasanya digunakan pada musik daerah Jawa Barat.

II.2.10 Android

II.2.10.1 Sejarah Android

Android adalah sebuah sistem operasi untuk perangkat lunak *mobile* berbasis linux yang mencakup sistem operasi, middleware dan aplikasi. Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka. Awalnya, Google Inc. membeli Android Inc. yang merupakan pendatang baru yang membuat peranti lunak untuk ponsel/*smartphone*. Kemudian untuk mengembangkan Android, dibentuklah Open Handset Alliance, konsorsium dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi, termasuk Google, Htc, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia.

Pada saat perilisan perdana Android, 5 November 2007, Android bersama Open Handset Alliance menyatakan mendukung pengembangan *open source* pada perangkat *mobile*. Di lain pihak, Google merilis kode-kode android dibawah lisensi *Apache*, sebuah lisensi perangkat lunak dan open *platform* perangkat seluler.

Di dunia ini terdapat dua jenis distributor sistem operasi android. Pertama yang mendapat dukungan penuh dari Google atau Google Mail *Service*s (GMS) dan kedua adalah yang benar-benar bebas distribusinya tanpa dukungan langsung Google atau dikenal sebagai Open Handset Distribution (OHD).

Sekitar September 2007 Google mengenalkan Nexus One, salah satu jenis *smartphone* yang menggunakan android sebagai sistem operasinya. Telpon seluler ini diproduksi oleh HTC Corporation dan tersedia di pasaran pada 5 Januari 2010. Pada 9 Desember 2008, diumumkan anggota baru yang bergabung dalam program kerja Android AEM Holdings, Atheros Communication, diproduksi oleh Asustek Komputer Inc, Garmin Ltd, Softbank, Sony Ericsson, Toshiba Corp, dan Vodafone Group Plc. Seiring pembentukan Open Handset Alliance, OHA mengumumkan produk perdana mereka, android perangkat *mobile* yang merupakan modifikasi kernel Linux 2.6. sejak Android dirilis telah dilakukan berbagai pembaruan berupa perbaikan bug dan penambahan fitur baru.

Pada masa saat ini kebanyakan vendor-vendor *smartphone* sudah memproduksi *smartphone* berbasis android, vendor-vendor itu antara lain HTC, Motorola, Samsung, LG, HKC, Huawei, Archos, Webstation Camangi, Dell, Nexus, SciPhone, WayteQ, Sony Ericsson, LG, Acer, Philips, T-*Mobile*, Nexian, IMO, Asus dan masih banyak lagi vendeor *smartphone* didunia yang memproduksi android. Hal ini karena android adalah sistem oerasi yang *open source* sehingga bebas didistribusikan dan dipakai oleh vendor manapun.

Tidak hanya menjadi sistem operasi di *smartphone*, saat ini android menjadi pesaing utama dari Apple pada sistem operasi Table PC. Pesatnya pertumbuhan Android selain faktor yang disebutkan diatas adalah karena android itu sendiri adalah *platform* sangat lengkap baik itu sistem operasi, aplikasi dan Tool Development, Market aplikasi android serta dukungan yang sangat tinggi dari komunitas *Open source* di dunia, sehingga android terus berkembang pesat dari segi teknologi maupun dari segi jumlah *device* yang ada di dunia. [3]

II.2.10.2 *The Dalvik Virtual Machine* (DVM)

Salah satu elemen kunci dari android adalah *Dalvik Virtual Machine* (DVM). Android berjalan di *Dalvik Virtual Machine* (DVM) bukan di Java *Virtual Machine* (JVM), terdapat banyak persamaan antara dengan Java *Virtual Machine* (JVM) seperti Java ME (*Java Mobile Edition*), tetapi android menggunakan *Virtual Machine* sendiri yang ditambahkan beberapa fitur dan dirancang sesuai kebutuhan untuk memastikan bahwa beberapa fitur-fitur berjalan lebih efisien pada perangkat *mobile*.

Dalvik Virtual Machine (DVM) adalah "*register bases*" sementara java *Virtual Machine* (JVM) bersifat "*stack based*", DVM didesain dan ditulis oleh Dan Bornsten dan beberapa enginers Google lainnya. Jadi bisa dikatakan "*Dalvik* equal (Java) = False" *Dalvik Virtual Machine* menggunakan kernel linux untuk menangani fungsionalitas tingkat rendah termasuk keamanan, *threading*, dan proses serta manajemen memori. Ini memungkinkan kita untuk menulis aplikasi C/C++ sama halnya seperti pada OS Linux kebanyakan. Meskipun dalam

kenyataannya kita harus banyak memahami Arsitektur dan proses sistem dari kernel linux yang digunakan dalam Android tersebut.

Semua *hardware* yang berbasis android dijalankan dengan menggunakan *Virtual Machine* untuk eksekusi aplikasi, pengembang tidak perlu khawatir tentang implementasi pernagkat keras tertentu. *Dalvik Virtual Machine* mengeksekusi *Executable* file, sebuah format yang dioptimalkan untuk memastikan memori yang digunakan sangat kecil. *The Executable* file diciptakan dengan mengubah kelas bahasa java dan dikompilasi menggunakan *tools* yang disediakan dalam SDK Android. [3]

II.2.10.3 Arsitektur Android

Secara garis besar arsitektur android dapat dijelas dan digambarkan sebagai berikut :

1. *Applications* dan *Widgets*

   Applications dan *Widgets* ini adalah layer dimana berhubungan dengan aplikasi dan biasanya download aplikasi kemudian lakukan instalasi dan jalankan aplikasi tersebut, di layer inilah terdapat seperti aplikasi inti termasuk klien email, program SMS, kalender, peta, *browser*, kontak, dan lain-lain. Semua aplikasi ditulis menggunakan bahasa pemrograman Java.

2. *Applications Frameworks*

   Android adalah "*Open Development Platform*" yaitu android menawarkan kepada pengembang atau member kemampuan kepada pengembangan untuk membangun aplikasi yang bagus dan inovatif. Pengembang bebas untuk mengakses perangkat keras, akses informasi *resources*, menjalankan serive *background*, mengatur alarm, dan menambahkan tambahan seperti status *notification*s dan masih banyak lagi. Pengembang memiliki akses penuh menuju API *Framework* seperti yang dilakukan oleh aplikasi yang kategori inti. Arsitektur aplikasi dirancang supaya dengan mudah dapat menggunakan komponen yang sudah digunakan (*reuse*).

Sehingga bisa disimpulkan Application Framework adalah layer dimana para pembuat aplikasi melakukan pengembangan/pembuatan aplikasi yang akan dijalankan di sistem android, karena pada layer inilah aplikasi dapat dirancang, seperti *content-providers* yang berupa sms dan panggilan telepon.

Komponen-komponen yang termasuk didalam *application Framework* adalah sebagai berikut :

a. Views

b. Content Provider

c. Resource Manager

d. Notification Manager

e. Activity Manager

3. *Libraries*

Libraries adalah layer dimana fitur-fitur android berada biasanya para pembuat aplikasi kebanyakan mengakses *libraries* untuk menjalankan aplikasinya. Berjalan diatas kernel, layer ini meliputi berbagai library C/C++ inti seperti Libc dan SSL, serta :

a. *Libraries* media untuk pemutar media audio dan video
b. *Libraries* untuk manajemen tampilan
c. *Libraries Graphics* mencakup SGL dan OpenGL untuk grafis 2D dan 3D
d. *Libraries SQLite* untuk dukungan database
e. *Libraries* SSL dan *WebKit* terintegrasi dengan web *browser* dan security
f. *Libraries Live Webcore* mencakup modern web *browser* dengan engine embedded web view

4. *Android Run Time*

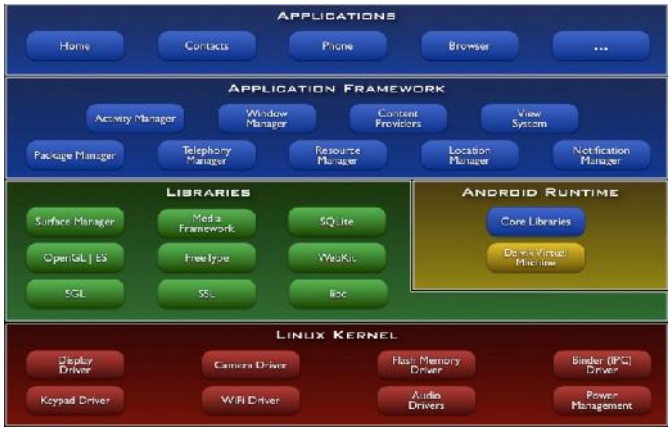
Layer yang membuat aplikasi android dapat dijalankan dimana dalam prosesnya menggunakan implementasi Linux. *Dalvik Virtual Machine* (DVM) merupakan mesin yang membentuk dasar kerangka aplikasi Android. Didalam *Android Run Time* dibagi menjadi dua bagian yaitu:

a. *Core Libraries* : aplikasi android dibangun dalam bahasa java, sementara *Dalvik* sebagai virtual mesin bukan *Java Virtual Machine*, sehingga diperlukan sebuah *libraries* yang berfungsi untuk menterjemahkan bahasa Java/C yang dihandle oleh Core *Libraries*

b. *Dalvik Virtual Machine* : Virtual mesin yang berbasis *register* yang dioptimalkan untuk menjalankan fungsi-fungsi secara efisien dimana merupakan pengembangan yang mampu membuat linux kernel untuk *threading* dan manajemen tingkat rendah.

5. Linux kernel

   Linux kernel adalah layer dimana inti dari operating sistem dari android itu sendiri, berisi file-file sistem yang mengatur sistem *processing*, *memory*, *resources*, *drivers*, dan sistem-sistem operanting andoroid lainnya. Linux kernel yang digunakan android adalah linux kernel *release* 2.6. [3]



Gambar II.20 Arsitektur android

II.2.10.4 Fundamental Aplikasi

Aplikasi android ditulis dalam bahasa pemrograman java, kode java dikompilasi bersama data file *resource* yang dibutuhkan oleh aplikasi dimana prosesnya dipaket oleh *tools* yang dinamakan "apt *tools*" ke dalam paket android sehingga menghasilkan file dengan ekstensi apk. File apk itulah yang sebenernya disebut dengan aplikasi yang dapat diinstal di perangkat *mobile* nantinya. Berikut beberapa jenis komponen pada aplikasi android yaitu:

1. *Activities*

   Suatu *activity* akan menyajikan *user interface* (UI) kepada pengguna, sehingga pengguna dapat melakukan interaksi. Sebuah aplikasi android bisa jadi hanya memiliki satu *activity*, tetapi umumnya aplikasi memiliki banyak *activity* tergantung pada tujuan aplikasi dan desain dari aplikasi tersebut. Satu *activity* biasanya akan dipakai untuk menampilkan aplikasi atau yang bertindak sebagai *user interface* (UI) saat aplikasi diperlihatkan kepada *user*. Untuk pindah dari satu *activity* ke *activity* lain dapat dilakukan dengan satu even misalnya click tombol, memilih opesi atau menggunakan triggers tertentu. Secara hirarki sebuah window *activity* dinyatakan dengan method *Activity*.setContentView(). ContentView adalah objek yang berada pada root hirarki.

2. *Service*

   Service tidak memiliki visual *user interface* (UI), tetapi *service* berjalan secara *background*, sebagai contoh dalam memainkan music, *service* mungkin memainkan music atau mengambil data dari jaringan, tetapi setiap *service* haruslah berada dalam kelas induknya. Misalnya media player sedang memutar lagu dari *list* yang ada. Aplikasi ini akan memiliki dua atau lebih *activity* yang memungkinkan *user* untuk memilih lagu atau menulis sms sambil *player* sedang jalan untuk menjaga *music* tetapi dijalankan, *activity player* dapat menjalankan *service* untuk membuat aplikasi tetap berjalan. *Service* dijalankan pada *thread* utama dari proses aplikasi.

3. *Broadcast Receiver*

   Broadcast Receiver berfungsi menerima dan bereaksi untuk menyampaikan notifikasi. Contoh *broadcast* seperti notifikasi zona waktu berubah, baterai *low*, gambar telah selesai diambil oleh kamera, atau pengubah referensi bahasa yang digunakan. Aplikasi juga dapat menginisiasi *broadcast* misalnya memberikan informasi pada aplikasi lain bahwa ada data yang telah didownload ke perangkat dan siap untuk digunakan *Broadcast Receiver* tidak memiliki *user interface* (UI) tetapi

memiliki sebuah *activity* untuk merespon informasil yang mereka terima atau mungkin menggunakan *notification* manager untuk memberitahu kepada pengguna seperti lampu latar atau *viberating* (getaran) perangkat dan lain sebagainya.

4. *Content Provider*

   Content Provider membuat kumpulan aplikasi data secara spesifik sehingga bisa digunakan oleh aplikasi lain. Data disimpan dalam file sistem seperti database SQLite.*Content Provider* menyediakan cara untuk mengakses data yang dibutuhkan oleh suatu *activity*, misalnya ketika menggunakan aplikasi yang membutuhkan peta (MAP) atau aplikasi yang membutuhkan untuk mengkases data kontak dan navigasi, maka disinilah fungsi *content provider*. [3]

II.2.10.5 Versi Android

Telepon pertama yang memakai sistem operasi android adalah HTC Dream yang dirilis pada 22 Oktober 2008. Pada penghujung 2010 diperkirakan hamper semua vendor seluler didunia menggunakan android sebagai operating sistem. Adapun versi-versi android yang pernah dirilis adalah sebagai berikut :

1. Android versi 1.1

   Pada 9 Maret 2009, Google merilis Android versi 1.1. android versi ini dilengkapi dengan pembaruan estis pada aplikasi, jam, *alarm*, *voice search* (pencarian suara), pengiriman pesan dengan Gmail, dan pemberitahuan email.

2. Android versi 1.5 (*cupcake*)

   Pada pertengahan 2009, google kembali merilis telepon seluluer dengan menggunakan Android dan SDK (*Software Development Kit*) dengan versi 1.5 (*Cupcake*). Terdapat beberapa pembaruan termasuk penambahan beberapa fitur dalam seluler versi ini yakni kemampuan merekam dan menonton video dengan modus kamera, mengupload video Youtube dan gambar ke Picasa langsung dari telepon, dukungan *Bluetooth* A2DP, kemampuan terhubung secara otomatis ke headset

Bluetooth, dan *keyboard* pada layer yang dapat disesuaikan oleh sistem.

3. Android versi 1.6 (*Donut*)

   Donut (versi 1.6) dirilis pada September dengan menampilkan proses pencarian yang lebih baik disbanding sebelumnya, pengguna *baterai indicator* dan *control applet* VPN. Fitur lainnya adalah galeri yang memungkinkan pengguna untuk memilih foto yang akan dihapus, kamera, camcorder dan galeri yang diintegrasikan, CD/EVDO, 802.1x, VPN, Gestures, dan *Text-to-speech engine*, kemampuan *dial* kontak, teknologi *text to change speech* (tidak tersedia pada semua ponsel), pengadaan resolusi VWGA.

4. Android versi 2.0/2.1 (*Eclai*r)

   Pada 3 Desember 2009 kembali diluncurkan ponsel Android dengan versi 2.0/2.1 (*Éclair*), perubahan yang dilakukan adalah pengoptimalan *hardware*, peningkatan Google Maps 3,1,2, perubahan UI dengan *browser* baru dan dukungan HTML 5, daftar kontak yang baru, dukungan *flash* untuk kamera 3.2 MP, *digital Zoom*, dan *Bluetooth* 2.1.

5. Android versi 2.2 (Froyo: *Frozen Yoghurt*)

   Pada bulan Mei 2010 android 2.2 Rev 1 diluncurkan. Android inilah yang sangat banyak sekarang dipasaran, salah satunya adalah dipakai di Samsung FX tab yang sudah ada dipasaran. Fitur yang tersedia di android versi ini sudah kompleks sekali diantaranya adalah :

   a. Kerangka aplikasi memungkinkan penggunaan dan penghapusan komponen yang tersedia
   b. *Dalvik Virtual Machine* dioptimalkan untuk perangkat *mobile*.
   c. Grafik: grafik di 2D dan 3D berdasarkan *libraries* OpenGL
   d. SQLite: untuk penyimpanan data.
   e. Mendukung media: audio, video, dan berbagai format gambar (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF).
   f. GSM, *Bluetooth*, EDGE, 3G, dan WiFi (tergantung *Hardware*)
   g. Kamera, *Global Positioning System* (GPS), kompas, dan accelerometer (tergantung *Hardware*)

6. Android versi 2.3 (*Gingerbread*)

   Android versi 2.3 diluncurkan pada desember 2010, hal-hal yang direvisi dari versi sebelumnya adalah kemampuan sebagai berikut :

   a. SIP-*based* VoIP
   b. *Near Field Communications* (NFC)
   c. *Gyroscope* dan sensor
   d. *Multiple cameras support*
   e. *Mixable audio effects*
   f. *Download manager*

Demikian beberapa versi android yang sudah dirilis sampai sekarang, kemungkinan besar versi tersebut akan terus berkembang seiring dengan kebutuhan yang sangat kompleks dibidang penggunaan *smartphone*.[3]

II.2.10.6 Android SDK

Android SDK adalah *tools* API (*Application Programming Interface*) yang diperlukan untuk mulai mengembangkan aplikasi pada *platform* android menggunakan bahasa pemrograman Java. Android merupakan subset perangkat lunak untuk ponsel yang meliputi sistem operasi, *middleware* dan aplikasi kunci yang di *release* oleh Google. Saat ini disediakan Android SDK (*Software Development Kit*) sebagai alat bantu dan API untuk mulai mengembangkan aplikasi pada *platform* android menggunakan bahasa pemrograman Java. Sebagai *platform* aplikasi-netral, android dapat memberi anda kesempatan untuk membuat aplikasi yang kita butuhkan yang bukan merupakan aplikasi bawaan *Handphone* / *Smartphone*. Beberapa fitur-fitur android yang paling penting adalah :

1. *Framework* : aplikasi yang mendukung pengganti komponen dan reusable.
2. Mesin *Virtual Dalvik* dioptimalkan untuk perangkat *mobile*
3. *Integrated Browser* berdasarkan engine *open source WebKit*.
4. Grafis yang dioptimalkan dan didukung oleh *libraries* grafis 2D, grafis 3D berdasarkan spesifikasi opengl ES 1,0 (Opsional Ekselerasi *hardware*)
5. SQLite untuk penyimpanan data.

6. Media *Support* yang mendukung audio, video, dan gambar (MPEG4, H.264, MP3, AAC, AMR, JPG, PING, GIF), GSM Telephony (tergantung *Hardware*)
7. *Bluetooth*, EDGE, 3G, dan WiFi (tergantung *hardware*)
8. Kamera, GPS, Kompas, dan *Accelerometer* (tergantung *hardware*)
9. Lingkungan *Development* yang lengkap dan termasuk pernagkat *emulator, tools* untuk *debugging,* profil dan kinerja memori, dan *plugin* untuk IDE Eclipse.
10. Native Code, digunakan untuk aplikasi yang melakukan komputasi yang sangat kritis dan membutuhkan banyak daya komputasi. Komponen native code dengan C atau C++ bisa diterapkan di Android untuk alasan efisiensi. Sehingga komputasi yang intensif sebaiknya menggunakan C atau C++. Android NDK bisa dipakai untuk mengembangkan *native component*. NDK ini merupakan pelengkap *Software Development Kit* (SDK) Android dan memiliki beberapa *library* yang bisa digunakan untuk mengembangkan *library* C/C++.

Berikut beberapa *Class-Class* yang digunakan pada Android:

1. Activity

   Merupakan hal yang ada pada aplikasi Android. Sebuah activity, biasanya berupa satu layar dalam aplikasi. Activity ini dinyatakan dengan sebuah *class* tunggal yang merupakan perpanjangan (*extends*) dari *class* Activity.
2. Intent Receiver

   Android menggunakan *class* Intent untuk melakukan aksi, berubah dari layar ke layar yang lain. Dua hal utama yang perlu diperhatikan pada struktur *class* Intent adalah aksi yang dilakukan dan data yang dibawa.
3. Content Providers

   Merupakan sebuah *class* yang berisi *method-method* yang memperbolehkan aplikasi lain menyimpan dan mengambil data yang di-*handle* oleh sebuah content provider. Content provider ini berguna ketika kita merancang aplikasi yang dibuat dapat di-share untuk aplikasi lainnya.

4. View

   Pokok penyusun tampilan grafis disebut View. Tiap View di deskripsikan oleh objek View yang tugasnya adalah menggambarkan area kontak dan meng-*handel* event di area tersebut. View merupakan class dasar untuk objek *widget* yang berinteraksi dengan user. Contoh *widget* adalah button, checkbox, textview.

5. MediaPlayer/MediaRecorder

   Merupakan metode strandar untuk memanipulasi audio. Tapi hanya bisa untuk file audio berbasis file atau *stream data*. Ini akan menciptakan *thread* sendiri untuk pemrosesannya. SoundPool menggunakan *framework* ini.

6. AudioTrack/AudioRecord

   Menyediakan akses *direct* ke raw audio. Berguna untuk memanipulasi audio di memory. Akan menuliskan *buffer* ketika memainkan audio atau penggunaan lainnya yang tidak membutuhkan file atau stream. Metode ini tidak membuat *thread* sendiri untuk pemrosesan

7. AudioFormat

   Kelas AudioFormat digunakan untuk mengakses sejumlah format audio dan konstanta konfigurasi channel. AudioFormat biasanya digunakan dalam AudioTrack dan AudioRecord. [3]

II.2.11 *Tools-Tools* yang digunakan

II.2.11.1 *Unified Modelling Language* (UML)

UML (*Unified Modelling Language*) adalah salah satu alat bantu yang sangat handal di dunia pengembangan sistem yang berorientasi obyek. Hal ini disebabkan UML menyediakan bahasa pemodelan visual yang memungkinkan bagi pengembang sistem untuk membuat cetak biru atas visi mereka dalam bentuk yang baku, mudah dimengerti serta dilengkapi dengan mekanisme yang efektif untuk berbagi (*sharing*) dan mengkomunikasikan rancangan mereka dengan yang lain.

UML merupakan kesatuan dari bahasa pemodelan yang dikembangkan oleh Booch, *Object Modeling Technique* (OMT) dan *Object Oriented Software Engineering* (OOSE). Metode Booch dari Grady Booch sangat terkenal dengan nama metode *Design Object Oriented*. Metode ini menjadikan proses analisis dan *design* ke dalam empat tahapan iterative, yaitu: identifikasi kelas-kelas dan obyek-obyek, identifikasi semantik dari hubungan objek dan kelas tersebut, perincian *interface* dan implementasi. Keunggulan metode Booch adalah pada detil dan kayanya dengan notasi dan elemen. Pemodelan OMT yang dikembangkan oleh Rumbaugh didasarkan pada analisis terstruktur pemodelan entity-relationship. Tahapan utama dalam metodologi ini adalah analisis, *design* sistem, *design* obyek dan implementasi. Keunggulan metode ini adalah dalam penotasian yang mendukung semua konsep OO. Metode OOSE dari Jacobson lebih member penekanan pada use case. OOSE memiliki tiga tahapan yaitu membuat model requirement dan analisis, *design* dan implementasi, dan model pengujian (*test model*). Keunggulan metode ini adalah mudah dipelajari karena memiliki notasi yang sederhana namun mencakup seluruh tahapan dalam rekayasa perangkat lunak.

Design UML, metode Booch, OMT dan OOSE digabungkan dengan membuang elemen-elemen yang tidak praktis ditambah dengan elemen-elemen dari metode lain yang lebih efektif dan elemen-elemen baru yang belum ada pada metode terdahulu sehingga UML lebih ekspresif dan seragam dari pada metode lainnya. [5]

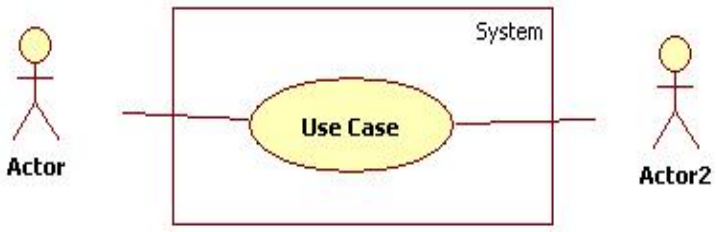
II.2.11.1.1 Use Case Diagram

Use Case adalah deskripsi fungsi dari sebuah sistem dari perspektif pengguna. *Use Case* bekerja dengan cara mendeskripsikan tipikal interaksi antara *user* (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Urutan langkah-langkah yang menerangkan antara pengguna dan sistem disebut *scenario*. Setiap scenario mendeskripsikan urutan kejadian. Setiap urutan diinisialisasi oleh orang, sistem yang lain, perangkat keras atau urutan waktu. Dengan demikian secara singkat bisa

dikatakan *use case* adalah serangkaian scenario yang digabungkan bersama-sama oleh tujuan umum pengguna.

Dalam pembicaraan tentang *use case*, pengguna biasanya disebut dengan aktor. Aktor adalah sebuah peran yang bisa dimainkan oleh pengguna dalam interaksinya dengan sistem.

Model *use case* adalah bagian dari model *requirement* (Jacob et all, 1992). Termasuk disini adalah problem domain *object* model dan penjelasan tentang *user interface*. *Use case* memberikan spesifikasi fungsi-fungsi yang ditawarkan oleh sistem dari perspektif *user*. [5]



Gambar II.21 *Use case model*

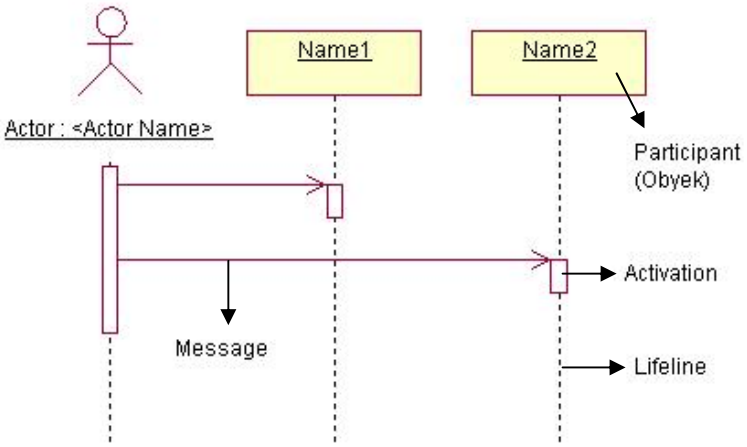
II.2.11.1.2 Sequence Diagram

Sequence diagram digunakan untuk menggambarkan perilaku pada sebuah skenario. Diagram ini menunjukkan sejumlah contoh obyek dan *message* yang diletakkan diantara obyek-obyek ini di dalam *use case*.

Sequence diagram menambahkan dimensi waktu pada interaksi diantara obyek. Pada diagram ini *participant* diletakkan di atas dan waktu ditunjukkan dari atas ke bawah. *Life line participant* diurutkan dari setiap *participant*. Kotak kecil pada *lifelin*e menyatakan *activation*, yaitu menjalankan salah satu operation dari *participant*. State bisa ditambahkan dengan menambahkannya sepanjang life line.

Message (sederhana, *synchronou*s atau *asynchronous*) adalah tanda panah yang menghubungkan suatu *life line* ke *life line* yang lain. Lokasi *life line* dalam dimensi vertikal mewakili urutan waktu dalam *sequence* diagram. *Message* yang pertama terjadi adalah yang paling dekat dengan bagian atas diagram dan yang terjadi belakangan adalah yang dekat dengan bagian bawah.

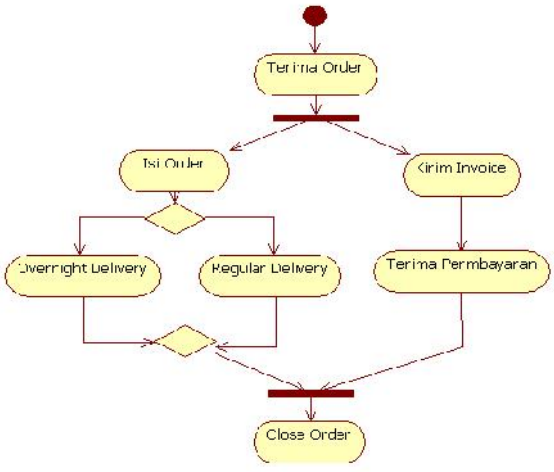
Pada beberapa sistem, operasi bisa dilakukan kepada dirinya sendiri. Hal ini disebut dengan rekursif. Untuk melukiskannya digunakan anak panah dari *activation* kembali ke dirinya sendiri, dan sebuah kotak kecil diletakkan pada bagian atas dari *activation*. [5]



Gambar II.22 Contoh *sequence diagram*

II.2.11.1.3 Activity Diagram

Activity diagram seperti sebuah *flow chart*. *Activity* diagram menunjukkan tahapan, pengambilan keputusan dan percabangan. Diagram ini sangat berguna untuk menunjukkan *operation* sebuah obyek dan proses bisnis. Kelebihan *activity* diagram dibandingkan *flowchart* adalah kemampuannya dalam menampilkan aktivitas parallel. [9]

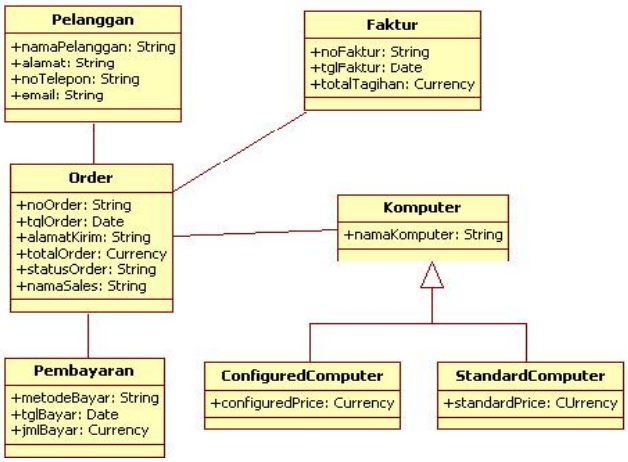


Gambar II.23 Contoh *activity diagram*

II.2.11.1.4 Class Diagram

Kotak adalah notasi UML untuk *class*. Nama, *attribute*, *operation* dan *responsibility* dari *class* ada pada kotak tersebut. *Stereotype* bisa dipergunakan untuk mengorganisasikan daftar *attribute* dan *operation*. Dalam beberapa kasus, kadangkala hanya perlu ditampilkan sebagian saja dari *attribute* dan *operation*.

Tipe *attribute* dan nilai default bisa dimunculkan sebagaimana pada *operation*. Untuk mengurangi ambiguitas pada pendeskripsian *class*, *constraint* bisa ditambahkan. Bahkan kalau perlu bisa ditambahkan *attached* notes ke dalam kotak tersebut. [5]



Gambar II.24 Contoh *class diagram*

II.2.11.2 Eclipse

Eclipse adalah sebuah IDE (*Integrated Development Environment*) untuk mengembangkan perangkat lunak dan dapat dijalankan di semua *platform* (*platform- independent*). Berikut ini adalah sifat dari Eclipse:

1. *Multi-platform*
2. Target sistem operasi Eclipse adalah Microsoft Windows, Linux, Solaris, AIX, HP-UX dan Mac OS X.
3. *Mulit-language*
4. Eclipse dikembangkan dengan bahasa pemrograman Java, akan tetapi Eclipse mendukung pengembangan aplikasi berbasis bahasa pemrograman lainnya, seperti C/C++, Cobol, Python, Perl, PHP, dan lain sebagainya.

5. *Multi-role*
6. Selain sebagai IDE untuk pengembangan aplikasi, Eclipse pun bisa digunakan untuk aktivitas dalam siklus pengembangan perangkat lunak, seperti dokumentasi, test perangkat lunak, pengembangan web, dan lain sebagainya.

Eclipse pada saat ini merupakan salah satu IDE favorit dikarenakan gratis dan *open source*, yang berarti setiap orang boleh melihat kode pemrograman perangkat lunak ini. Selain itu, kelebihan dari Eclipse yang membuatnya populer adalah kemampuannya untuk dapat dikembangkan oleh pengguna dengan komponen yang dinamakan *plug-in*. Macam–macam plugin yang tersedia di Eclipse :

1. Subversion
2. m2eclipse
3. Glassfish Integration
4. Jboss Tools
5. STS (SpringSource Tool Suite)
6. EGIT (Git Eclipse Plugin)
7. Mercurial Eclipse (Mercurial Eclipse Plugin)
8. Windows Builder