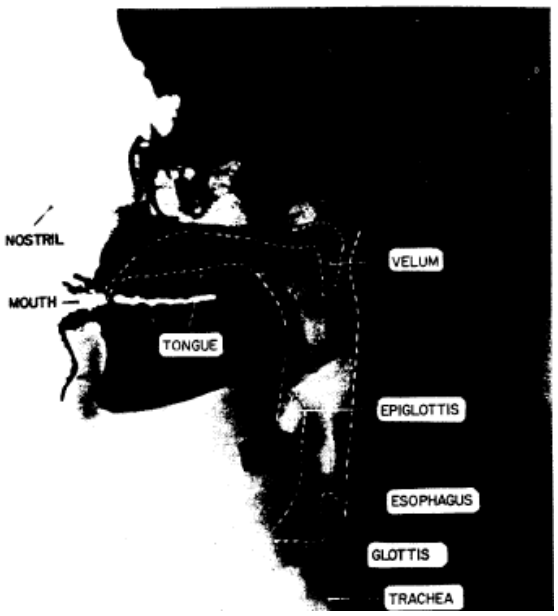
BAB 2

LANDASAN TEORI

2.1 Pengenalan Suara pada Manusia

Sinyal percakapan adalah sinyal yang dihasilkan dari suara manusia sewaktu melakukan percakapan. Sinyal percakapan merupakan kombinasi kompleks dari variasi tekanan udara yang melewati pita suara dan *vocal tract*, yaitu mulut, lidah, gigi, bibir, dan langit-langit mulut. *Speech* (wicara) dihasilkan dari sebuah kerjasama antara *lungs* (paru-paru), *glottis* (dengan *vocal cords*) dan *articulation tract* (*mouth* atau mulut dan *nose cavity*/rongga hidung). Sinyal suara terdiri dari serangkaian suara yang masing–masing menyimpan sepotong informasi. Berdasarkan cara menghasilkannya, suara dapat dibagi menjadi *voiced* dan *unvoiced*. *Voiced sounds* atau suara ucapan dihasilkan dari getaran pita suara, sedangkan *unvoiced sounds* dihasilkan dari gesekan antara udara dengan *vocal tract* (berawal di awal bukaan pita suara atau glottis dan berakhir di bibir).

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



Gambar 2.1 Proses Produksi Suara [5]

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

yang berbeda. Di dalam kasus *unvoiced sounds*, keluaran pada *vocal tract* lebih menyerupai *noise* atau derau.

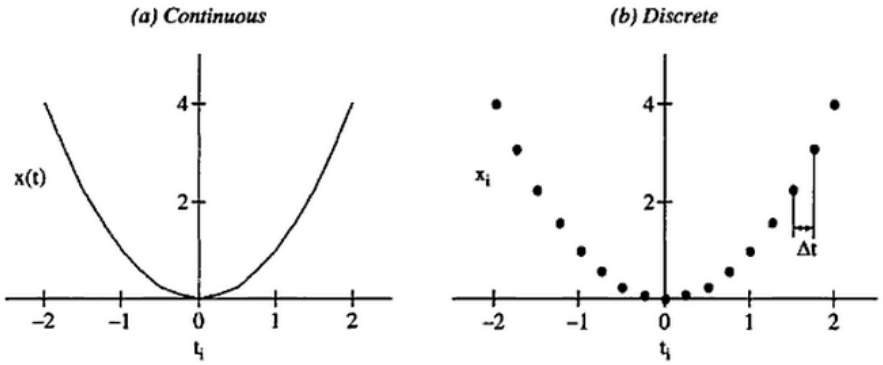
2.2 Sinyal Analog dan Sinyal Digital

Sinyal adalah besaran fisis yang berubah menurut waktu, ruang atau variabel-variabel bebas lainnya. Contoh sinyal : sinyal ucapan.

Biasanya sinyal ini berbentuk tanda-tanda, lampu-lampu, suara-suara, dan lain-lain. Dalam istilah teknik, sinyal itu ialah besaran yang berubah dalam waktu dan atau dalam ruang, dan membawa suatu informasi.

Secara sistematis, sinyal adalah fungsi dari satu atau lebih variabel independen. Proses ini dilakukan melalui pemodelan sinyal.

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



Gambar 2.2 Sinyal Waktu Kontinyu dan (b) Sinyal Waktu Diskrit

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

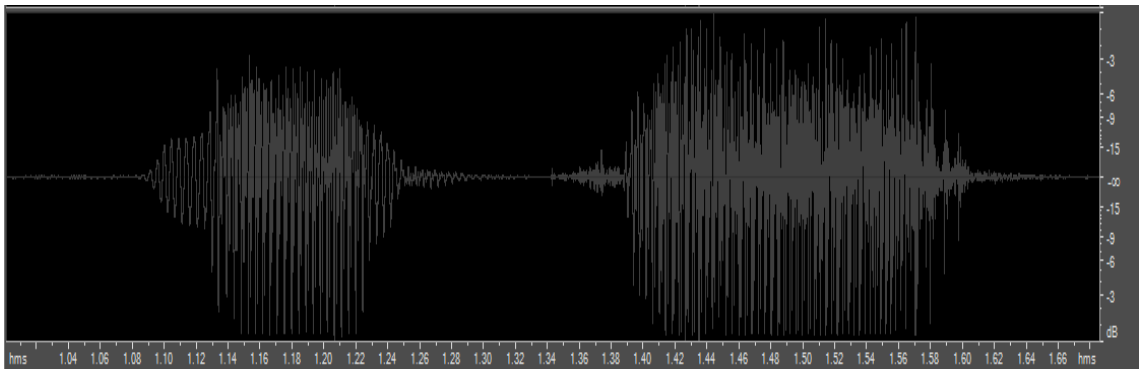
2.2.1 Sinyal Analog

Sinyal analog atau sinyal waktu kontinyu adalah sinyal yang memiliki nilai real pada setiap waktu. Sinyal kontinyu merupakan suatu sinyal yang berbentuk gelombang sinusoidal dan merupakan variabel yang berdiri sendiri. Pada sinyal kontinyu, variabel indipendent (yang berdiri sendiri) terjadi terus-menerus dan kemudian sinyal dinyatakan sebagai sebuah kesatuan nilai dari variabel independent. Dengan menggunakan sinyal analog, maka jangkauan transmisi data dapat mencapai jarak yang jauh, tetapi sinyal ini mudah terpengaruh oleh noise. Gelombang pada sinyal analog yang umumnya berbentuk gelombang sinus memiliki tiga variable dasar, yaitu amplitudo, frekuensi dan *phase.*

- Amplitudo merupakan ukuran tinggi rendahnya tegangan dari sinyal analog.

- Frekuensi adalah jumlah gelombang sinyal analog dalam satuan detik.
- *Phase* adalah besar sudut dari sinyal analog pada saat tertentu.

Salah satu contoh sinyal suara yang paling mudah adalah suara.



Gambar 2.3 Bentuk Sinyal Suara

2.2.2 Sinyal Digital

Sinyal digital merupakan hasil teknologi yang dapat mengubah sinyal menjadi kombinasi urutan bilangan 0 dan 1 (juga dengan biner), sehingga tidak mudah terpengaruh oleh derau, proses informasinya pun mudah, cepat dan akurat, tetapi transmisi dengan sinyal digital hanya mencapai jarak jangkau pengiriman data yang relatif dekat. Biasanya sinyal ini juga dikenal dengan sinyal diskrit. Sinyal yang mempunyai dua keadaan ini biasa disebut dengan bit. Bit merupakan istilah khas pada sinyal digital. Sebuah bit dapat berupa nol (0) atau satu (1). Kemungkinan nilai untuk sebuah bit adalah 2 buah (2^1). Kemungkinan nilai untuk 2 bit adalah sebanyak 4 (2^2), berupa 00, 01, 10, dan 11. Secara umum, jumlah kemungkinan nilai yang terbentuk oleh kombinasi n bit adalah sebesar 2^n buah.

System digital merupakan bentuk *sampling* dari sitem analog digital pada dasarnya di *code*-kan dalam bentuk *biner* (atau *Hexa*) besarnya nilai suatu sistem

digital dibatasi oleh lebarnya/jumlah bit *(bandwidth)*. Jumlah bit juga sangat mempengaruhi nilai akurasi sistem digital.

Signal digital ini memiliki berbagai keistimewaan yang unik yang tidak dapat ditemukan pada teknologi analog yaitu :

- Mampu mengirimkan informasi dengan kecepatan cahaya yang dapat membuat informasi dapat dikirim dengan kecepatan tinggi.
- Penggunaan yang berulang-ulang terhadap informasi tidak mempengaruhi kualitas dan kuantitas informsi itu sendiri.
- Informasi dapat dengan mudah diproses dan dimodifikasi ke dalam berbagai bentuk.
- Dapat memproses informasi dalam jumlah yang sangat besar dan mengirimnya secara interaktif.

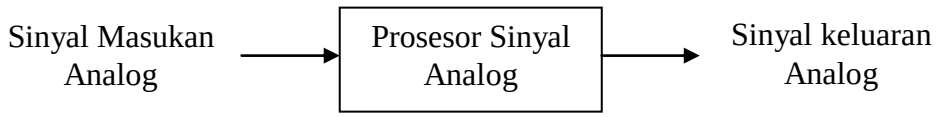
Pengolahan sinyal digital memerlukan komponen-komponen digital, *register*, *counter*, *decoder*, mikroprosessor, mikrokontroler dan sebagainya.

Saat ini pengolahan sinyal banyak dilakukan secara digital, karena kelebihannya antara lain :

1. Untuk menyimpan hasil pengolahan, sinyal digital lebih mudah dibandingkan sinyal analog. Untuk menyimpan sinyal digital dapat menggunakan media digital seperti CD, DVD, *Flash Disk, Hardisk*. Sedangkan media penyimpanan sinyal analog adalah pita tape magnetik.
2. Lebih kebal terhadap *noise* karena bekerja pada level ‘0’ dan ‘1’.
3. Lebih kebal terhadap perubahan temperatur.
4. Lebih mudah pemrosesannya.

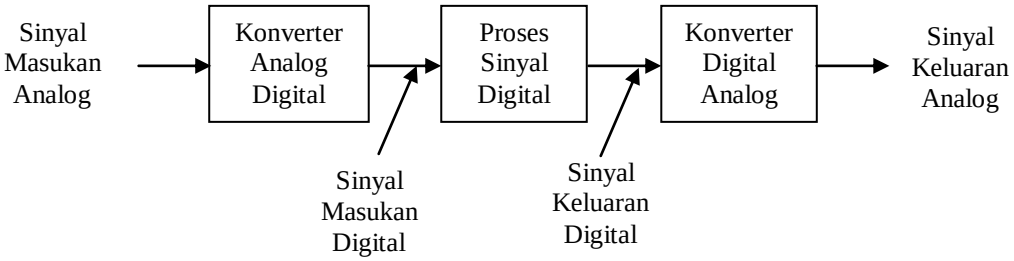
2.2.3 Elemen Dasar Sistem Pemrosesan Sinyal Digital

Sebagian besar sinyal-sinyal yang ditemukan dalam sains dan teknologi adalah analog yaitu sinyal-sinyal yang merupakan fungsi dari suatu variabel kontinyu, seperti waktu dan ruang, dan biasanya mengambil nilai-nilai dalam interval yang kontinyu. Sinyal-sinyal seperti itu dapat diproses secara langsung dengan sistem analog yang tepat atau penggandaan frekuensi yang bermaksud mengubah karakteristiknya atau mengambil beberapa informasi yang diinginkan. Itu artinya sinyal telah diproses secara langsung dalam bentuk analognya, dengan sinyal masukan maupun keluarannya adalah sinyal analog. Proses ini diilustrasikan pada gambar berikut:



Gambar 2.4 Sistem Proses Sinyal Analog

Pemrosesan sinyal digital menyediakan suatu metode alternatif untuk pemrosesan sinyal analog, seperti diilustrasikan pada gambar berikut:



Gambar 2.5 Sistem Proses Sinyal Digital

Untuk melakukan pemrosesan sinyal digital, diperlukan suatu *interface* yang dinamakan *Analog to Digital Converter* (ADC). Keluaran pengkoversian ADC adalah sinyal digital yang cocok dengan masukan terhadap prosesor digital.

Untuk pemakaian dengan keluaran digital dari prosesor sinyal digital akan disampaikan kepada pemakai dalam bentuk analog. Untuk itu, diperlukan sebuah *interface* lain untuk mengubah sinyal digital menjadi sinyal analog, yang dinamakan dengan *Digital to Analog Converter* (DAC). Namun pada beberapa aplikasi, pengkonversian DAC tidak diperlukan, dikarenakan aplikasi tersebut hanya melakukan analisis sinyal dengan informasi yang ingin disampaikan dalam bentuk digital. Program sistem aplikasi yang akan dibangun merupakan salah satunya.

2.2.4 *Bit Rate*

Dalam telekomunikasi dan komputasi, bit rate (kadang-kadang ditulis *bitrate*, data *rate* atau sebagai variabel R [16]) adalah jumlah bit yang disampaikan atau diproses per unit waktu.

Tingkat bit dihitung menggunakan bit per detik (bit/s atau bps) unit, sering bersama dengan awalan SI seperti kilo-(kbit/s atau kbps), mega-(Mbit/s atau Mbps), giga-(Gbit/s atau Gbps) atau tera-(Tbit/s atau Tbps). Perhatikan bahwa, tidak seperti banyak lainnya yang berkaitan dengan komputer unit, 1 kbit/s secara tradisional didefinisikan sebagai 1.000-bit/s, bukan 1.024-bit/s, dll, juga sebelum 1999 ketika SI prefiks diperkenalkan untuk unit informasi dalam standar IEC 60027-2. Huruf besar K seperti dalam Kbit/s atau Kbps tidak boleh digunakan [10].

Singkatan resmi untuk "bit per detik" adalah "bit/s" (bukan "bita/detik", lihat gaya penulisan untuk unit SI). Dalam konteks tidak formal singkatan "b/s" atau

"bps" sering digunakan, meskipun ini kebingungan risiko dengan "byte per detik" ("B/s", "Bps"). 1 Byte/s (Bps atau B/s) sesuai dengan 8-bit/s (bps atau b/s) [10].

Bit rates		
Decimal prefixes (SI)		
Name	**Symbol**	**Multiple**
kilobit per second	kbit/s	10^3
megabit per second	Mbit/s	10^6
gigabit per second	Gbit/s	10^9
terabit per second	Tbit/s	10^{12}
Binary prefixes (IEC 60027-2)		
kibibit per second	Kibit/s	2^{10}
mebibit per second	Mibit/s	2^{20}
gibibit per second	Gibit/s	2^{30}
tebibit per second	Tibit/s	2^{40}

Gambar 2.6 *Bit rates* [10]

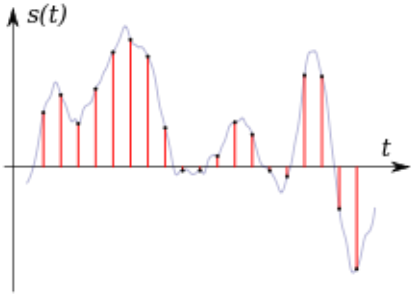
Dua faktor yang mempengaruhi kapasitas informasi *(Bit Rate)*, yaitu lebar pita frekuensi *(bandwidth)* dan kualitas saluran (dinyatakan dengan *Signal to Noise Ratio*, S/N atau SNR) [9].

2.2.5 *Sampling Rate*

Sampling rate, atau frekuensi *sampling* (f_s) mendefinisikan jumlah sampel per unit waktu (biasanya detik) diambil dari sinyal kontinu untuk membuat sinyal diskrit. Untuk waktu-domain sinyal, unit untuk *sampling rate* adalah hertz (detik terbalik, 1/s, s-1), kadang-kadang tercatat sebagai Sa/s (sampel per detik). Kebalikan dari frekuensi *sampling* adalah periode *sampling* atau *sampling* interval, yang adalah waktu antara sampel [8].

Pada saat merekam suara, sirkuit ADC *(Analog to Digital Converter)* yang terdapat di sound card akan mengukur dan menangkap sinyal elektrik yang dikirim secara terus menerus . Frekuensi dari sinyal elektrik yang dapat ditangkap

oleh sirkuit tersebut kemudian disimpan sebagai data biner yang disebut dengan data sample. Jumlah data sample yang dapat diambil setiap 1 detik disebut dengan *sampling rate.*



Gambar 2.7 Sinyal analog (biru muda) dengan sinyal sampel (merah) dengan jarak tetap atau *sampling rate* [8].

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

2.3 Konversi Sinyal Analog Menjadi Sinyal Digital

Sebagian besar sinyal-sinyal, seperti sinyal seismik, sinyal radar, dan khususnya sinyal suara merupakan sinyal analog. Untuk memproses sinyal analog dengan alat digital tidak dapat langsung diproses begitu saja. Langkah pertama ialah harus dikonversi dulu menjadi sinyal digital, yaitu mengkonversi menjadi suatu deret angka yang mempunyai presisi terbatas. Proses ini dinamakan konversi analog ke digital (A/D) dengan alat yang dinamakan *Analog to Digital Converter* (ADC).

Secara konsepsi, konversi sinyal dari analog ke digital mempunyai tiga langkah, yaitu:

1. Pencuplikan *(Sampling)*

   Ini adalah konversi suatu sinyal waktu kontinyu menjadi suatu sinyal waktu diskrit yang diperoleh dengan mengambil cuplikan *(sample)* sinyal waktu kontinyu pada saat waktu diskrit.

$$x(n) = x_a(nT), \qquad -\infty < n < \infty \tag{1}$$

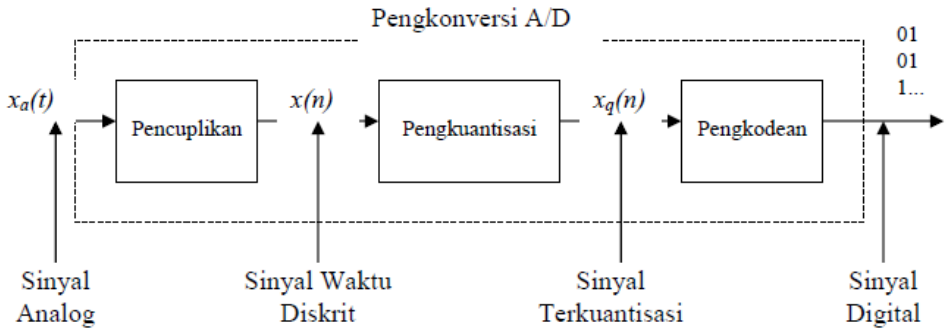
   Dengan $x(n)$ adalah sinyal waktu diskrit yang diperoleh dengan mengambil cuplikan-cuplikan sinyal analog $x_a(t)$ setiap *T* detik. Selang waktu *T* antara cuplikan yang berurutan dinamakan periode pencuplikan dan kebalikannya $\frac{1}{T} = F_s$ dinamakan laju pencuplikan.

2. Kuantisasi

   Ini adalah konversi sinyal yang bernilai kontinyu waktu diskrit menjadi sinyal bernilai diskrit, waktu diskrit (digital). Nilai setiap cuplikan sinyal digambarkan dengan suatu nilai terpilih dari himpunan berhingga nilai-nilai yang mungkin. Selisih antara cuplikan $x(n)$ yang tidak terkuantisasi dan keluaran $x_q(n)$ yang terkuantisasi dinamakan Galat Kuantisasi *(Quantization Error)*.

3. Pengkodean *(Coding)*

   Dalam proses pengkodean, setiap nilai diskrit $x_q(n)$ digambarkan dengan suatu barisan biner-b.



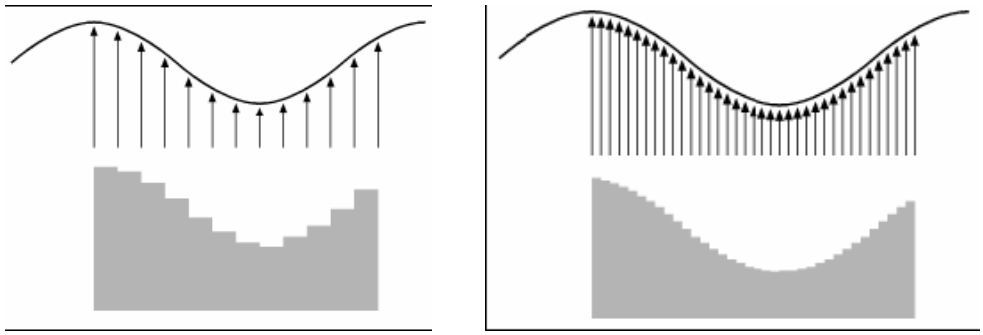
Gambar 2.8 Bagian Dasar Konversi Analog Ke Digital

Perlu dicatat bahwa data-data digital yang dihasilkan ADC hanyalah merupakan pendekatan proporsional terhadap masukan analog. Hal ini karena tidak mungkin melakukan konversi secara sempurna berkaitan dengan kenyataan bahwa informasi digital berubah dalam *step-step,* sedangkan analog berubahnya secara kontinyu.

Misalnya ADC dengan resolusi 8 bit menghasilkan bilangan 0 sampai dengan 255 (256 bilangan dan 255 *step),* dengan demikian tidak mungkin menyajikan semua kemungkinan nilai-nilai analog. Jika sekarang resolusinya menjadi 20 bit maka akan terdapat 1.048.575 *step,* semakin banyak kemungkinan nilai-nilai analog yang bisa disajikan. Penting untuk diingat, bagaimanapun juga pada sebuah *step* terdapat tak terhingga kemungkinan nilai-nilai analog untuk sembarang ADC yang dapat diperoleh di dunia ini. Sehingga apa yang dibuat manusia *(Human made)* tidak akan pernah bisa menyamai kondisi dunia nyata.

ADC *(Analog to Digital Converter)* memiliki 2 karakter prinsip, yaitu kecepatan *sampling* dan resolusi. Kecepatan *sampling* suatu ADC menyatakan seberapa sering sinyal analog dikonversikan ke bentuk sinyal digital pada selang

waktu tertentu. Kecepatan *sampling* biasanya dinyatakan dalam *sample per second* (SPS).



Gambar 2.9 ADC dengan kecepatan *sampling* rendah dan kecepatan *sampling* tinggi

Resolusi ADC menentukan ketelitian nilai hasil konversi ADC. Sebagai contoh: ADC 8 bit akan memiliki *output* 8 bit data digital, ini berarti sinyal *input* dapat dinyatakan dalam 255 (2n–1) nilai diskrit. ADC 12 bit memiliki 12 bit output data digital, ini berarti sinyal input dapat dinyatakan dalam 4096 nilai diskrit. Dari contoh diatas ADC 12 bit akan memberikan ketelitian nilai hasil konversi yang jauh lebih baik daripada ADC 8 bit.

Prinsip kerja ADC adalah mengkonversi sinyal analog ke dalam bentuk besaran yang merupakan rasio perbandingan sinyal input dan tegangan referensi. Sebagai contoh, bila tegangan referensi 5 volt, tegangan input 3 volt, rasio input terhadap referensi adalah 60%. Jadi, jika menggunakan ADC 8 bit dengan skala maksimum 255, akan didapatkan sinyal digital sebesar 60% x 255 = 153 (bentuk decimal) atau 10011001 (bentuk biner).

signal = *(sample/max_value) * reference_voltage*
= (153/255) * 5
= 3 Volts

2.4 Jenis File Audio Digital

Setiap bentuk file audio memiliki kelebihan dan kekurangan masing-masing. Format file audio tersebut dapat dirubah sesuai dengan kebutuhan. Format file audio bermacam-macam, diantaranya:

a. WAP *(Wave),* format file ini merupakan dasar dari format audio file yang memiliki kualitas suara terbaik, hanya saja file ini membutuhkan tempat penyimpanan yang besar. Pemilihan format ini sangat tepat apabila membutuhkan kualitas audio yang baik dan memiliki tempat penyimpanan yang besar. Format file ini mendukung untuk mono atau stereo [14].

b. Apple AIFF (.AIF, .SND), format ini adalah format audio standar milik Apple Computer. Seperti WAV milik Windows, AIFF mendukung untuk fasilitas mono atau stereo, 16-bit atau 8-bit [14].

c. Dialogic ADPCM (.VOX), format Dialogic ADPCM ini biasanya ditemui pada aplikasi telepon. Format ini hanya dapat menyimpan audio mono 16-bit, dan seperti format ADPCM lainnya file ini dapat dikompres hingga 4-bit [14].

d. DiamondWare Digitized (.DWD), ini adalah format audio yang digunakan oleh perangkat DiamondWare's Sound, biasanya format ini digunakan oleh para programmer untuk menghasilkan audio interaktif yang diaplikasikan pada game dan multimedia. Format ini juga medukung baik mono maupun stereo [14].

e. MPEG Audio Player 3 (.MP3), ini merupakan format audio file yang banyak diminati oleh para pengguna komputer, karena disamping kualitas yang

dihasilkan baik file ini juga tidak memerlukan tempat penimpanan yang besar [14].

f. Real Media (.RM), format audio ini biasanya dapat ditemukan pada jaringan internet [14].

g. Sound Blaster (.VOC), ini adalah format audio file dari Sound Blaster dan format file suara dari Sound Blaster Pro. Format ini hanya mendukung 8-bit audio, mono hingga 44.1 KHz, dan stereo hingga 22 KHz [14].

h. *Advance Audio Coding (AAC).* Sepuluh tahun sejak ditemukannya MP3, sering pula disebut MP4. Apel merupakan vendor yang paling getol menggunakan file suara berlisensi ini. Apel juga merupakan pengembang dari file AAC ini yang bisa dijalankan di iTunes, QuickTime 6, iPod, dan seterusnya [15].

i. Musik Digital (MIDI) merupakan standar untuk menghubungkan komputer dengan instrumen musik elektronik dan pemrosesan efek khusus. Format suara instrumen ini di perkenalkan pada tahun1983 oleh perusahaan musik elektrik seperti Roland, Yamaha dan Korg. Format MIDI bersifat sangat kompak dengan ukurannya yang kecil, suara yang di hasilkan oleh MIDI dengan dukungan sound card yang memiliki *synthesizer* (penghasil suara elektrik) sangatlah mirip dengan organ elektrik yang bisa memainkan berbagai alatmusik atau peranti elektronik lainnya, tetapi tidak cocok untuk hasil konversi dari suara analog karena tidak terlaluakurat. File dengan format ini berukuran kecil dan sering digunakan dalam ponsel sebagai ringtone [15].

2.5 Ekstraksi Fitur

2.5.1 Analisis LPC

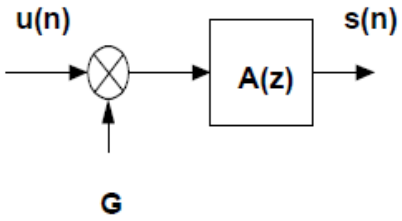
Representasi sinyal suara dalam bentuk parameter merupakan cara representasi sinyal yang cukup rumit karena dari sinyal yang ada akan dihitung secara matematis parameter sinyal yang mengandung informasi sinyal. Hal ini dapat diselesaikan dengan menggunakan fitur algoritma LPC.

Teori LPC yang dipakai dalam analisa suara, telah dipahami dengan baik dalam beberapa tahun yang lalu. Ada beberapa alasan yang menjadikan teori LPC telah banyak digunakan dalam sistem pengenalan suara, antara lain: [5]

1. LPC membuktikan suatu model yang baik untuk pengenalan suara, yaitu memberikan parameter model yang tepat untuk sinyal suara, dapat dilihat pada spektrum koefisien peramalan yang mirip dengan spektrum sinyal aslinya.
2. Perhitungan yang dibutuhkan untuk mencari parameter sinyal suara relatif lebih singkat dibandingkan dengan metode lainnya.
3. Metode untuk mendapat parameter-parameter sinyal ucapan, seperti jalur formant dan amplitudo.

2.5.2 Persamaan LPC

Pemodelan sinyal suara dengan LPC terlihat seperti Gambar 2.14.



Gambar 2.10 Model LPC dari suara

Berdasarkan model pada Gambar 2.14, hubungan antara $s(n)$ dan $u(n)$ adalah:

$$s(n) = \sum_{k=1}^{p} a_k s(n-k) + Gu(n) \quad (2.1)$$

2.5.3 Proses LPC

2.5.4 *Pre-emphasis* [5],[20]

Preemphasis adalah suatu proses produksi suara manusia, radiasi pada bibir dan lidah ketika proses *phonation* mengakibatkan komponen frekuensi tingginya. *Preemphasis* terhadap cuplikan sinyal dengan peersamaan *preemphasizer:*

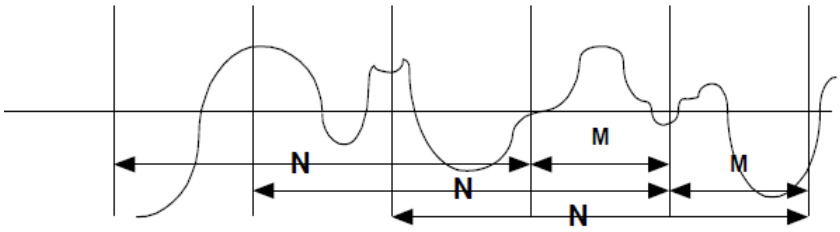
$$s(n) = s(n) - as(n-1) \quad (2.2)$$

dengan $s(n)$ adalah sampel ke-n dan harga *a* yang paling sering digunakan adalah 0,95.

2.5.5 *Framming*

Pada tahap ini sinyal hasil *preemphasis* dikelompokkan ke dalam bingkai-bingkai dengan ukuran masing-masing bingkai N data. Bingkai ini berurutan

dengan pemisahan antara kedua bingkai sebesar M data. Biasanya M=1/3 N. Blok pemilihan pemilihan bingkai dari sinyal suara dapat dilihat pada Gambar 2.11.



Gambar 2.11 Blok pemilihan bingkai dari sinyal suara

Membagi hasil *preemphasis* $s(n)$ ke dalam *frame-frame* yang masing-masing memuat *N* buah sampel yang dipisahkan sejauh *M* buah sampel. Semakin $M \ll N$ semakin baik perkiraan spektral LPC dari *frame* ke *frame.*

$$s(n) = M < N \tag{2.3}$$

Bingkai pertama berisi n data pertama sinyal suara. Bingkai ke-2 dimulai dari data ke M pada bingkai pertama, sehingga terdapat penumpukan bingkai sejauh N-M buah data. Demikian juga dengan bingkai ke-3 dimulai dari data ke-2M bingkai pertama (atau data ke-M dari bingkai ke-2), sehingga terdapat penumpukan bingkai sejauh N-2M dengan bingkai pertama. Proses ini berlangsung terus sampai seluruh data sinyal suara dibingkaikan.

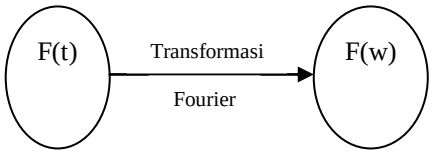
2.5.6 *Windowing*

Melakukan *windowing* terhadap setiap *frame* yang telah dibentuk untuk meminimalkan diskontinuitas pada ujung awal dan ujung akhir setiap *frame* dengan persamaan *Hamming Window* untuk sampel ke-n adalah:

$$W(n) = 0{,}54 - 0{,}46 * \cos\left(\frac{2\pi n}{N-1}\right),\ 0 \leq n \leq N-1 \tag{2.4}$$

2.5.7 *Transformasi Diskrit Fourier*

Transformasi Fourier adalah suatu model transformasi yang memindahkan domain spasial atau domain waktu menjadi domain frekuensi.



Gambar 2.12 Transformasi *Fourier*

Transformasi fourier, yang ditemukan oleh Joseph Fourier, sering digunakan untuk mentransformasikan sebuah sinyal domain waktu yang bersifat kontinyu menjadi sinyal domain frekuensi.

Transformasi Diskrit Fourier (DFT) atau yang sering disebut sebagai transformasi fourier terbatas adalah sebuah transformasi fourier yang seringkali digunakan dalam pemrosesan sinyal dan bidang-bidang yang berkaitan dengan analisis frekuensi yang terkandung di dalam sebuah sinyal, untuk menyelesaikan persamaan differensial parsial, dan untuk operasi seperti konvolusi.

DFT diperkenalkan sebagai aproksimasi metode numerik untuk mengerjakan fungsi transformasi Fourier. DFT $X(m)$ dari sebuah sinyal waktu yang diskrit $x(n)$ dapat ditulis sebagai berikut:

$$X(m) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi nm/N} \qquad (2.5)$$

Dari persamaan 2.1, komponen dari bilangan natural dipisahkan menjadi bagian riil dan imajinernya menggunakan persamaan Euler, yaitu:

$$e^{-j\theta} = \cos(\theta) + j\sin(\theta) \qquad (2.6)$$

Dengan menggabungkan persamaan 2.1 dan 2.2 maka didapat persamaan DFT yang baru, yaitu:

$$X(m) = \sum_{n=0}^{N-1} x(n)[\cos\left(\frac{2\pi nm}{N}\right) - j\sin\left(\frac{2\pi nm}{N}\right)] \qquad (2.7)$$

Seperti yang terlihat pada persamaan 2.1 dan 2.3, kalkulus tidak lagi dibutuhkan untuk menghitung DFT. Dengan fungsi penjumlahan yang terbatas, tidak dijumpai kesulitan dengan fungsi yang tak terbatas bersifat kontinyu. Dalam bidang proses sinyal digital, sinyal dan spektra diproses hanya dalam bentuk sampel, sehingga DFT adalah sesuatu yang benar-benar dipakai untuk memprosesnya. Implementasi dari DFT dapat menggunakan transformasi Fourier cepat atau *Fast Fourier Transform*.

2.5.8 *Fast Fourier Transform (FFT)*

Fast Fourier Transform adalah suatu algoritma komputasi optimal yang mengimplemetasikan *Discreet Fourier Transform* (DFT) dengan teknik perhitungan yang cepat serta memanfaatkan sifat periodikal dari transformasi *fourier*. FFT merupakan operasi matematika yang bertujuan untuk dekomposisi dari suatu sinyal domain waktu ke sinyal domain frekuensi.

Konsep utama algoritma ini adalah mengubah sinyal suara yang berbasis waktu menjadi berbasis frekuensi dengan membagi masalah menjadi beberapa upa masalah yang lebih kecil. Kemudian, setiap upa masalah diselesaikan dengan cara melakukan pencocokan pola digital suara [4]. Sinyal suara yang diurai menjadi data berbentuk frekuensi memiliki informasi yang lebih mudah diolah dan juga karena hasil yang ingin dijadikan acuan adalah frekuensi dasar *user*.

DFT dilakukan dengan mengimplementasikan sebuah transformasi, dengan panjang vektor *N* berdasarkan rumus :

$$F(u) = \frac{1}{N} \Sigma \frac{x=N-1}{x=0} f(x) \exp\left[-\frac{2j\pi ux}{N}\right]$$

(2.4)

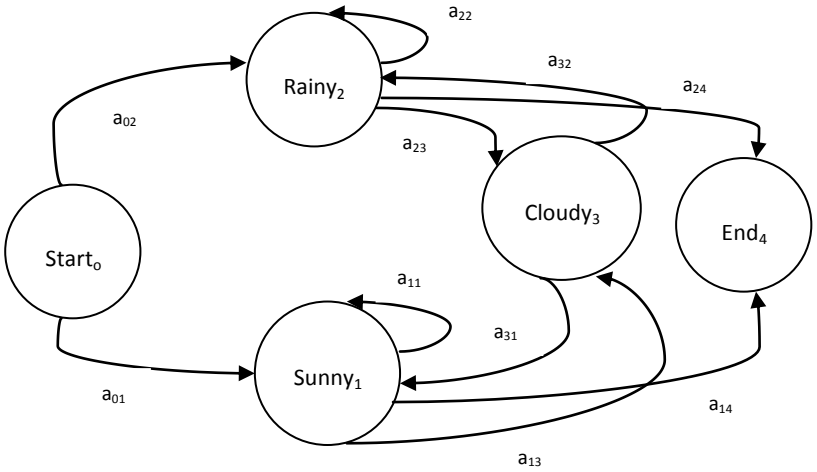
$$F(u) = \frac{1}{N} \Sigma \frac{x=N-1}{x=0} f(x)(\cos\left(\frac{2\pi ux}{N}\right) - fsin(\frac{2\pi ux}{N})) \qquad (2.8)$$

2.6 Pemodelan dengan *Markov Models*

2.6.1 Rantai Makov

Rantai markov merupakan suatu struktur yang terdiri dari entitas-entitas stationer yang disebut keadaan *(state). Markov Chain* merupakan perluasan dari *finite automaton*. *Finite automaton* sendiri adalah kumpulan *state* yang transisi antar *state*-nya dilakukan berdasarkan masukan observasi. Pada *Markov Chain*, setiap busur antar *state* berisi probabilitas yang mengindikasikan kemungkinan jalur tersebut akan diambil. Jumlah probabilitas semua busur yang keluar dari sebuah simpul adalah satu.

Analisa rantai Markov tidak memberikan keputusan rekomendasi, melainkan hanya informasi probabilitas mengenai situasi keputusan yang dapat membantu pengambil keputusan untuk mengambil keputusan. Dengan demikian, analisa rantai Markov bukanlah teknik optimisasi, tetapi adalah teknik deskriptif yang menghasilkan informasi probabilitas di masa mendatang.



Gambar 2.13 *Rantai Markov Left-Right Model*

2.6.2 Definisi *Hidden Markov Model (HMM)*

Penerapan teori HMM untuk pengenalan kata bukan lagi hal yang baru. Teori dasar HMM telah dipublikasikan oleh Baum pada awal tahun 70-an dan telah diaplikasikan oleh Baker dan Jelinek di IBM pada tahun 70-an. Tetapi perkembangan pesat dari HMM baru terjadi beberapa tahun terakhir ini. Hal ini disebabkan :

1. Teori dasar dari HMM telah dipublikasikan pada jurnal matematika namun tidak dibaca oleh insinyur yang bekerja pada pemrosesan sinyal.
2. Penerapan teori langsung ke pengenalan kata tidak memberi tutorial yang cukup untuk diterapkan ke penelitian mandiri [5].

Hidden Markov Model (HMM) merupakan salah satu model stokastik (model memiliki komponen *input random*, dan menghasilkan *output* yang *random* pula) yang banyak menarik perhatian akhir-akhir ini. HMM terdiri atas sebuah sinyal

yang dimodelkan sebagai sebuah rantai Markov keadaan terhingga dan sebuah observasi yang dimodelkan sesuai proses observasi pada rantai Markov.

HMM telah diperkenalkan dan dipelajari sejak akhir tahun 60-an dan awal tahun 70-an. Metode statisitik HMM semakin populer pada dekade terakhir ini karena model tersebut kaya akan struktur matematika dan mengandung teori dasar yang bisa digunakan untuk beberapa aplikasi yang penting. Penerapan dari HMM meliputi pengenalan ucapan (*speech recognition*), *target tracking*, komunikasi digital, teknik biomedik dan keuangan.

HMM didefinisikan sebagai sekumpulan lima parameter (N, M, A, B, π). Jika dianggap $\lambda=\{A, B, \pi\}$ maka HMM mempunyai parameter tertentu N dan M. Ciri-ciri HMM adalah: [17][18][19]

- Observasi diketahui tetapi urutan keadaan (state) tidak diketahui sehingga disebut hidden.
- Observasi adalah fungsi probabilitas keadaan.
- Perpindahan keadaan adalah dalam bentuk probabilitas.

2.6.3 Parameter Distribusi

HMM mempunyai parameter-parameter distribusi sebagai berikut : [5], [20]

1. Probabilitas Transisi

$$A = \{a_ij\} , a_{ij} = Pr(X_{t+1} = q_j \mid X_t = q_i) , 1 \leq j,i \leq N \quad (2.9)$$

2. Probabilitas observasi

$$B = \{ b_i\} , b_i(k) = Pr(O_t = V_k \mid X_t = q_{it}) , \quad (2.10)$$

3. Distribusi keadaan awal

$$\pi = \{ \pi_i\} , \pi_i = Pr(X_o = q_i) \quad (2.11)$$

Sedangkan parameter tertentu HMM ada dua yaitu N dan M.

1. N, jumlah keadaan model. Dinotasikan himpunan terbatas untuk keadaan yang mungkin adalah Q = {q_1, ……., q_N}
2. M, jumlah dari simbol observasi/keadaan, ukuran huruf diskret. Simbol observasi berhubungan dengan keluaran fisik dari sistem yang dimodelkan. Dinotasikan himpunan terbatas untuk observasi yang mungkin adalah V= {V_1, ……..,V_M}.

Secara tradisional, parameter-parameter HMM ditaksir berdasarkan kriteria *maximum likelihood* (ML) dan algoritma *Baum-Welch* (EM = Expectation Modification).

2.6.4 Tiga Masalah Dasar *Hidden Markov Model (HMM)*

Ada 3 masalah dasar HMM yang harus dipecahkan untuk model yang diterapkan di dunia nyata, yaitu : [5], [20]

1. Menghitung $P(O \mid \lambda)$ bila diberikan urutan observasi $O = O_1O_2 \ldots O_T$ dan sebuah model $\lambda=(A, B, \pi)$.
2. Memilih urutan keadaan yang paling optimal yang berhubungan dengan $Q = q_1q_2 \ldots q_T$ bila diberikan urutan observasi $O = O_1O_2 \ldots O_T$ dan sebuah model $\lambda=(A, B, \pi)$.
3. Mengatur parameter λ agar $P(O \mid \lambda)$ maksimal.

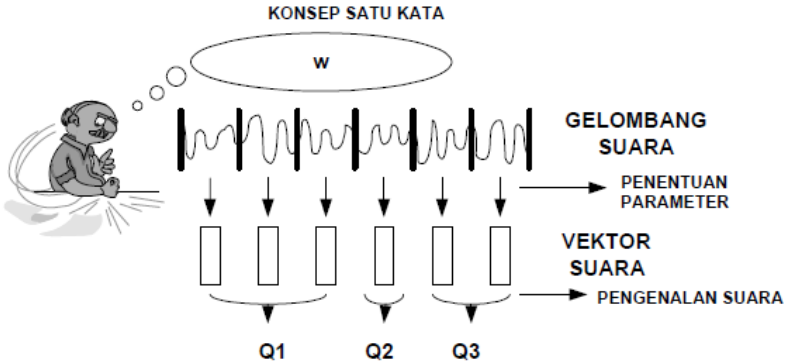
Untuk memecahkan masalah pertama dengan menggunakan algoritma Maju-Mundur. Masalah kedua dipecahkan dengan menggunakan algoritma Viterbi. Sedangkan masalah ketiga dipecahkan dengan algoritma Baum-welch.

2.6.5 Penentuan Parameter Secara Umum

Cara merepresentasi sinyal suara agar dapat dianalisa lebih lanjut dapat dilakukan dengan mempresentasikan sinyal suara ke dalam suatu bentuk. Suatu sinyal suara pada intinya dapat dipresentasikan ke dalam bentuk:

- Representasi sinyal suara dalam bentuk gelombang
- Representasi sinyal suara dalam bentuk parameter

Untuk representasi suara dalam bentuk gelombang dengan menampilkan pola-pola gelombang suara yang ada. Sedangkan representasi sinyal suara dalam bentuk parameter merupakan cara representasi sinyal yang cukup rumit karena dari sinyal yang ada akan dihitung secara matematis parameter sinyal yang mengandung informasi sinyal. Representasi sinyal suara ke dalam parameter memberikan hasil yang lebih baik daripada representasi sinyal suara dalam bentuk gelombang. Penentuan parameter HMM secara umum dapat dilihat pada Gambar 2.13.



Gambar 2.14 Penentuan Parameter secara Umum

Diasumsikan sinyal suara merupakan pesan yang dienkoding dengan urutan satu atau lebih simbol. Urutan tersebut adalah parameter vektor suara yang diasumsikan merupakan representasi gelombang suara dengan periode tertentu.

Sebuah gelombang suara dibagi-bagi dengan lama periode tertentu misalnya 10 ms, jadi untuk frekuensi sampling 8000 Hz akan mempunyai 80 titik per periode 10 ms. Representasi gelombang suara menjadi vektor suara menggunakan fitur LPC (Linear Predictive Coding).

Pada penentuan parameter HMM, diasumsikan bahwa urutan vektor observasi setiap kata merupakan model Markov. Pada gambar terlihat gelombang suara direperesentasikan ke bentuk parameter vektor suara yang terdiri dari 6 observasi dengan masing-masing observasi mempunyai panjang urutan tertentu.

Kata terisolasi adalah kata yang berdiri sendiri atau tidak dalam suatu rangkaian dengan kata yang lain. Sistem penentuan parameter HMM kata terisolasi menggunakan asumsi bahwa kata diucapkan oleh pembicara tunggal. Sebelum melakukan penentuan parameter maka untuk setiap v kata, harus dibuat parameter HMM sebanyak λ^v, sehingga parameter model (A,B,π) dapat mendekati himpunan kemungkinan vektor observasi dari kata ke-v.

Kata yang akan dikenali masuk ke sistem berupa sinyal suara dengan format *.wav* yang direkam dengan frekuensi *sampling* 500 Hz, 8 bit, stereo.

2.7 Pencarian *Binary Search*

Binary Search adalah algoritma pencarian yang lebih efisien daripada algoritma *Sequential Search*. Hal ini dikarenakan algoritma ini tidak perlu menjelajahi setiap elemen dari tabel. Kerugiannya adalah algoritma ini hanya bisa digunakan pada tabel yang elemennya sudah terurut baik menaik maupun menurun.

Pada intinya, algoritma ini menggunakan prinsip *divide and conquer*, dimana sebuah masalah atau tujuan diselesaikan dengan cara mempartisi masalah menjadi bagian yang lebih kecil. Algoritma ini membagi sebuah tabel menjadi dua dan memproses satu bagian dari tabel itu saja.

2.7.1 Proses *Searching* Algoritma *Binary Search*

Proses *searching binary* membagi tabel atau data menjadi dua bagian, dengan inisialisasi posisi awal = 0, posisi akhir = n untuk mendapatkan nilai tengah.

Data tengah = (posisi awal+posisi akhir)/2.

Selanjutanya bandingkan data yang dicari dengan data tengah, jika lebih kecil, proses dilanjutkan dengan posisi akhir = posisi tengah-1.

Jika lebih besar, proses dilanjutkan dengan posisi awal+1, dan diproses berulang hingga data tengah = data yang dicari.

2.8 *Object Oriented Programming (OOP)*

Object Oriented Programming (OOP) merupakan metodologi dalam pemrograman yang diciptakan untuk memodelkan kasus-kasus nyata ke dalam suatu objek.

Dalam OOP data dan fungsi-fungsi yang akan mengoperasikannya digabungkan menjadi satu kesatuan yang dapat disebut sebagai objek. Proses perancangan atau desain dalam suatu pemrograman merupakan proses yang tidak terpisah dari proses yang mendahului, yaitu analisis dan proses yang mengikutinya. Pembahasan mengenai orientasi objek tidak akan terlepas dari

konsep objek seperti inheritance atau penurunan, encapsulation atau pembungkusan, dan polymorphism atau kebanyakrupaan. Berikut penjelasannya:

1. *Class* dan Objek

   Dapat didefinisikan bahwa class merupakan struktur data dari suatu objek, lebih jelasnya adalah sebuah bentuk dasar atau blueprint yang mendefinisikan variabel method umum pada semua objek dari beberapa macam. Objek sendiri adalah kumpulan variabel dan fungsi yang dihasilkan dari template khusus atau disebut class. objek adalah elemen pada saat run-time yang akan diciptakan, dimanipulasi, dan dihancurkan ketika eksekusi. Ada pun class merupakan definisi statik dari himpunan objek yang mungkin diciptakan sebagai instantiasi dari class. Sederhananya adalah kumpulan objek yang mempunyai atribut sama. Dengan demikian, pada saat run-time maka yang kita miliki adalah objek [22].

2. *Encapsulation*

   Encapsulation adalah penyatuan anatara proses-proses yang terjadi dan sejumlah data yang dikelola ke dalam suatu bentuk kelompok. Pengelompokkan tersebut adalah dengan dibentuknya suatu objek secara abstrak *(abstraction).*

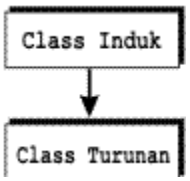
3. *Abstraction*

   Abstraction memiliki pengertian bahwa dalam mengembangkan perangkat lunak seorang programmer akan menggunakan atau menciptakan sejumlah objek dimana objek tersebut akan ada dan berperilaku pada sistem.

4. Inheritance

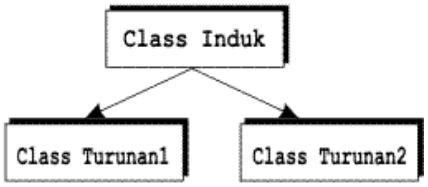
Inheritance atau pewarisan adalah kalimat yang pasti ada di dalam pemerograman berorientasi objek disebut juga dengan istilah reusable. Ketika kita menggunakan kembali atau mengganti method dari class yang sudah ada, serta ketika menambahkan field instant dan method baru, maka pada saat itulah Anda bekerja dengan inheritance. Konsep ini merupakan konsep yang fundamental dalam orientasi objek dan harus digunakan dengan baik, ada beberapa macam Inheritance atau pewarisan yang ada di dalam OOP *Single Inheritance* dan *Multple Inheritance*.

- *Single Inheritance*



Gambar 2.15 *Single Inheritance*

- *Multiple Inheritance*



Gambar 2.16 *Multiple Inheritance*

dalam penggunaan *inheritance class* Induk akan mewariskan semua atribut yang ia miliki sehingga *class* bawahnya akan memilki atribut yang

sama dengan *class* induk, bahkan dapat memilki kebabasan untuk memilki atribut berbeda dengan *class* induknya [22].

5. *Polymorphism*

   Polymorphism adalah suatu konsep dimana beberapa objek dapat memiliki metode yang sama dengan aksi yang berbeda.

2.9 *Unified Modeling Language (UML)*

UML adalah bahasa pemodelan standar yang terdiri dari serangkaian diagram, dikembangkan untuk membantu sistem dan pengembang perangkat lunak menyelesaikan tugas-tugas berikut: Spesifikasi; Visualisasi; Desain arsitektur; Konstruksi; Simulasi dan Pengujian serta Dokumentasi [12].

UML pada awalnya dikembangkan dengan ide mempromosikan komunikasi dan produktivitas antara pengembang sistem berorientasi objek, tetapi kekuatan tampak jelas dari UML telah menyebabkannya untuk membuat terobosan ke dalam setiap jenis sistem dan pengembangan perangkat lunak [12].

2.9.1. Konsep Dasar UML [23]

Dari berbagai penjelasan rumit yang terdapat di dokumen dan buku-buku UML. Sebenarnya konsepsi dasar UML bisa kita rangkumkan dalam gambar dibawah.

Major Area	*View*	*Diagrams*	*Main Concepts*
structural	static view	class diagram	class, association, generalization, dependency, realization, interface
	use case view	use case diagram	use case, actor, association, extend, include, use case generalization
	implementation view	component diagram	component, interface, dependency, realization
	deployment view	deployment diagram	node, component, dependency, location
dynamic	state machine view	statechart diagram	state, event, transition, action
	activity view	activity diagram	state, activity, completion transition, fork, join
	interaction view	sequence diagram	interaction, object, message, activation
		collaboration diagram	collaboration, interaction, collaboration role, message
model management	model management view	class diagram	package, subsystem, model
extensibility	all	all	constraint, stereotype, tagged values

Gambar 2.17 Konsep Dasar UML

Seperti juga tercantum pada gambar di atas UML mendefinisikan diagram-diagram sebagai berikut:

1. *Use Case Diagram*

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya.

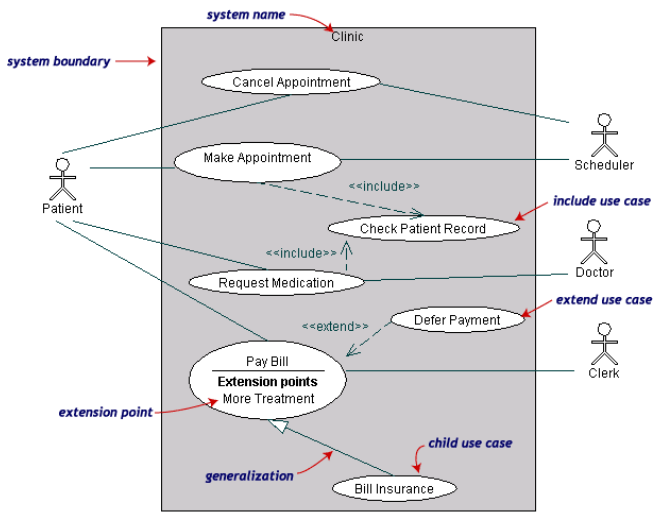
Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu.

Use case diagram dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem.

Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal. Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common.*

Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar u*se case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.

Contoh *use case diagram:*



Gambar 2.18 Contoh *Use Case Diagram*

2. Activity Diagram

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

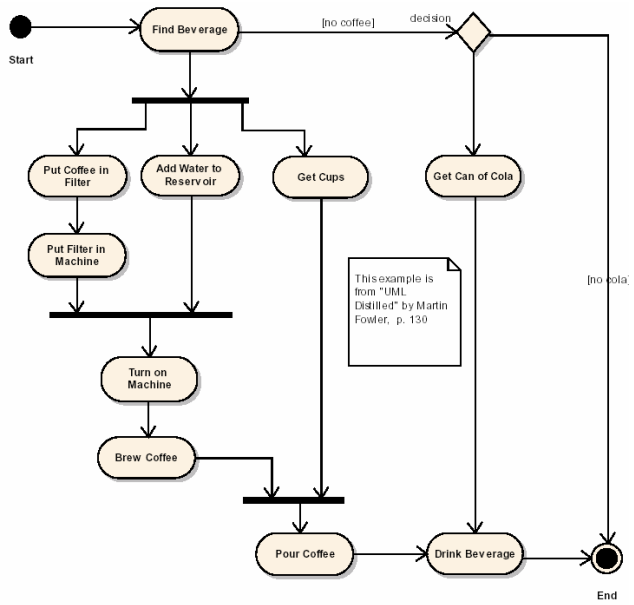
Activity diagram merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya *(internal processing).* Oleh karena itu *activity diagram* tidak menggambarkan *behaviour internal* sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas.

Sama seperti *state*, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan *behaviour* pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel *(fork dan join)* digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal.

Activity diagram dapat dibagi menjadi beberapa object *swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.

Contoh *activity diagram* tanpa *swimlane*:



Gambar 2.19 Contoh *Activity Diagram* Tanpa *Swimlane*

3. Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang men-trigger aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan.

Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi operasi/metoda dari *class. Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message*.

Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan *icon* khusus untuk objek *boundary, controller* dan *persistent entity*.

4. Class Diagram

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi).

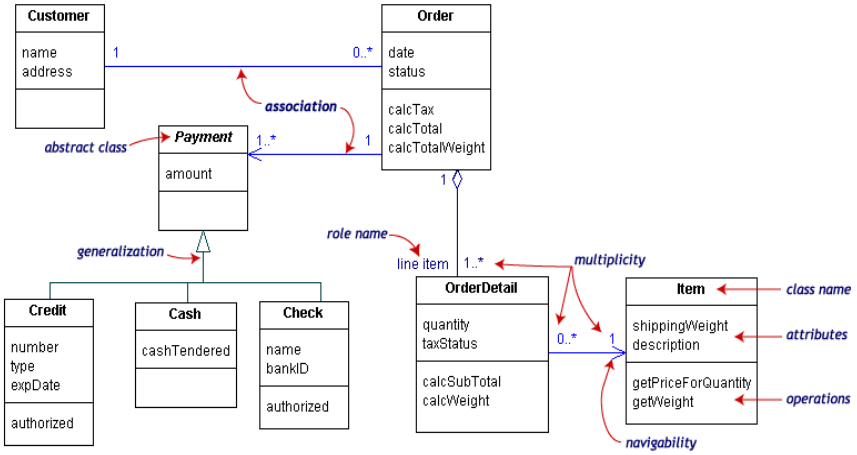
Class diagram menggambarkan struktur dan deskripsi *class, package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain.

Class memiliki tiga area pokok, yaitu: Nama (dan stereotype), Atribut dan Metoda.

Atribut dan metoda dapat memiliki salah satu sifat berikut :

- *Private,* tidak dapat dipanggil dari luar *class* yang bersangkutan.
- *Protected,* hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya.

- *Public*, dapat dipanggil oleh siapa saja.



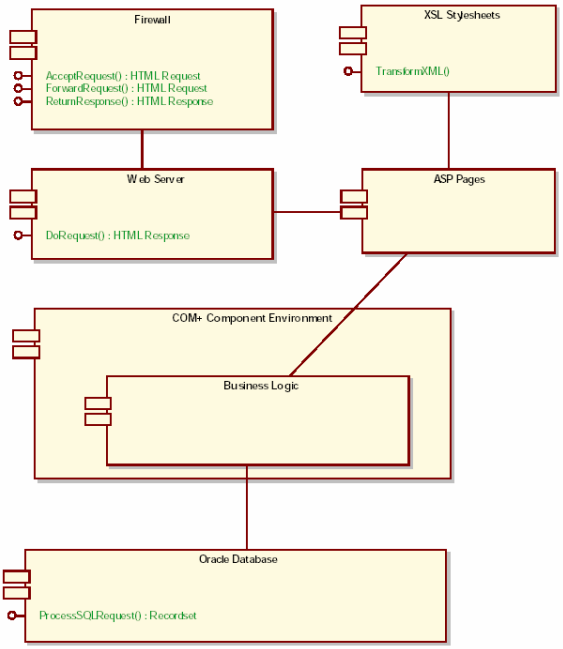
Gambar 2.20 Contoh *Class Diagram*

5. *Component Diagram*

Component diagram menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan *(dependency)* di antaranya.

Komponen piranti lunak adalah modul berisi *code*, baik berisi *source code* maupun *binary code*, baik *library* maupun *executable*, baik yang muncul pada *compile time*, *link time*, maupun *run time*. Umumnya komponen terbentuk dari beberapa *class* dan/atau *package*, tapi dapat juga dari komponen-komponen yang lebih kecil. Komponen dapat juga berupa *interface*, yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain.

Contoh *component diagram:*



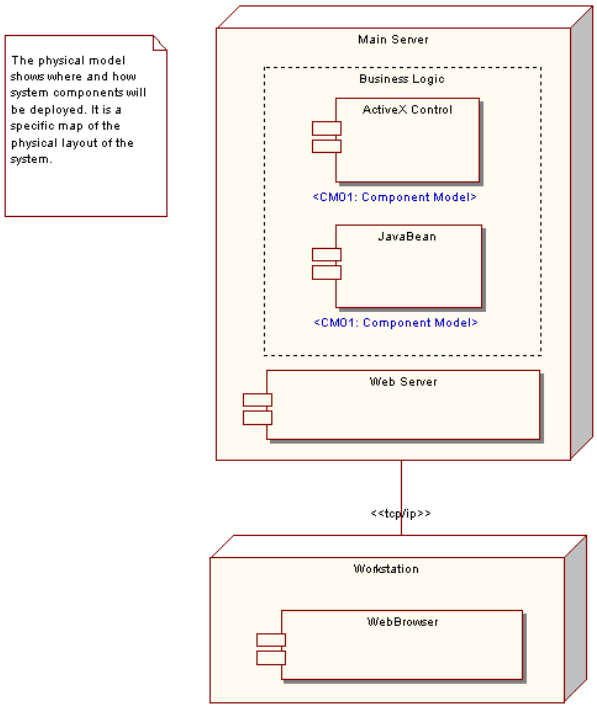
Gambar 2.21 Contoh *Component Diagram*

6. *Deployment Diagram*

Deployment/physical diagram menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal.

Sebuah *node* adalah server, *workstation*, atau piranti keras lain yang digunakan untuk men-*deploy* komponen dalam lingkungan sebenarnya. Hubungan antar *node* (misalnya TCP/IP) dan *requirement* dapat juga didefinisikan dalam diagram ini.

Contoh *deployment diagram* :



Gambar 2.22 Contoh *Deployment Diagram*

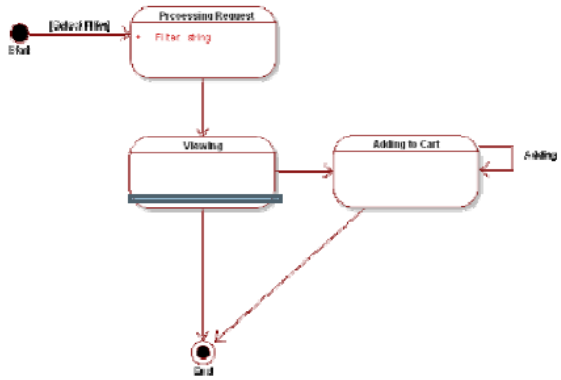
7. State Diagram

Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari *stimuli* yang diterima. Pada umumnya *statechart diagram* menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart diagram*).

Dalam UML, *state* digambarkan berbentuk segiempat dengan sudut membulat dan memiliki nama sesuai kondisinya saat itu. Transisi antar *state* umumnya memiliki kondisi *guard* yang merupakan syarat terjadinya transisi yang bersangkutan, dituliskan dalam kurung siku. *Action* yang dilakukan sebagai akibat dari *event* tertentu dituliskan dengan diawali garis miring.

Titik awal dan akhir digambarkan berbentuk lingkaran berwarna penuh dan berwarna setengah.

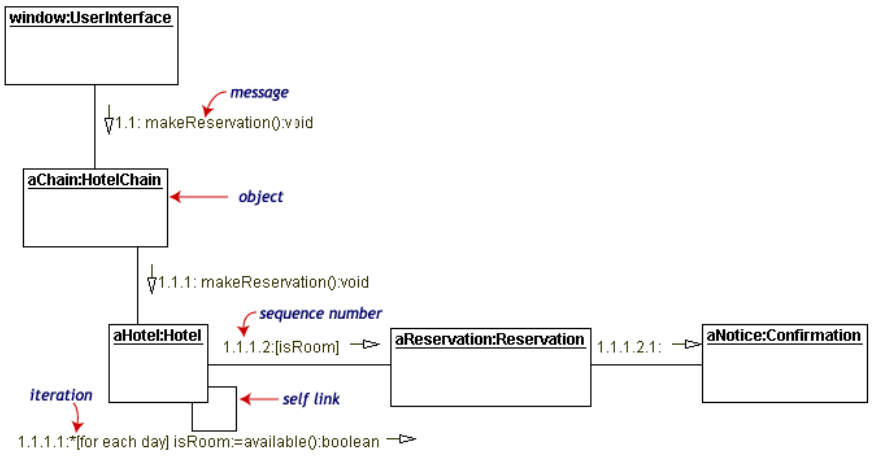
Contoh s*tatechart diagram:*



Gambar 2.23 Contoh *Statechart Diagram*

8. Collaboration Diagram

Collaboration diagram juga menggambarkan interaksi antar objek seperti *sequence diagram*, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian message. Setiap message memiliki *sequence number*, di mana *message* dari level tertinggi memiliki nomor 1. *Messages* dari level yang sama memiliki prefiks yang sama.



Gambar 2.24 Contoh *Collaboration Diagram*

2.10 Android [11]

Android adalah sistem operasi yang berbasis Linux untuk telepon seluler seperti telepon pintar dan komputer *tablet*. Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri untuk digunakan oleh bermacam peranti bergerak. Awalnya, Google Inc. membeli Android Inc., pendatang baru yang membuat peranti lunak untuk ponsel. 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.

Telepon pertama yang memakai sistem operasi Android adalah HTC Dream, yang dirilis pada 22 Oktober 2008. Pada penghujung tahun 2009 diperkirakan di dunia ini paling sedikit terdapat 18 jenis telepon seluler yang menggunakan Android. Adapun versi-versi Android yang pernah dirilis adalah sebagai berikut:

- Android versi 1.1

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

- Android versi 1.5 *(Cupcake)*

  Pada pertengahan Mei 2009, Google kembali merilis telepon seluler dengan menggunakan Android dan SDK *(Software Development Kit)* dengan versi 1.5 *(Cupcake)*. Terdapat beberapa pembaruan termasuk juga penambahan beberapa fitur dalam seluler versi ini yakni kemampuan merekam dan

menonton *video* dengan modus kamera, mengunggah *video* ke Youtube dan gambar ke Picasa langsung dari telepon, dukungan *Bluetooth* A2DP, kemampuan terhubung secara otomatis ke *headset Bluetooth*, animasi layar, dan *keyboard* pada layar yang dapat disesuaikan dengan sistem.

- Android versi 1.6 *(Donut)*

  Donut (versi 1.6) dirilis pada September dengan menampilkan proses pencarian yang lebih baik dibanding sebelumnya, penggunaan baterai indikator dan kontrol applet VPN. Fitur lainnya adalah galeri yang memungkinkan pengguna untuk memilih foto yang akan dihapus; kamera, *camcorder* dan galeri yang dintegrasikan; CDMA / 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.

- Android versi 2.0/2.1 *(Eclair)*

  Pada 3 Desember 2009 kembali diluncurkan ponsel Android dengan versi 2.0/2.1 *(Eclair),* perubahan yang dilakukan adalah pengoptimalan hardware, peningkatan Google Maps 3.1.2, perubahan UI dengan *browser* baru dan dukungan HTML5, daftar kontak yang baru, dukungan *flash* untuk kamera 3,2 MP, digital *zoom,* dan *Bluetooth 2.1*.

- Android versi 2.2 *(Froyo: Fozen Yoghurt)*

  Pada 20 Mei 2010, Android versi 2.2 *(Froyo)* diluncurkan. Perubahan-perubahan umumnya terhadap versi-versi sebelumnya antara lain dukungan *Adobe Flash 10.1*, kecepatan kinerja dan aplikasi 2 sampai 5 kali lebih cepat,

intergrasi V8 *JavaScript engine* yang dipakai Google Chrome yang mempercepat kemampuan *rendering* pada *browser*, pemasangan aplikasi dalam SD Card, kemampuan *WiFi Hotspot portabel*, dan kemampuan *auto update* dalam aplikasi Android Market.

- Android versi 2.3 *(Gingerbread)*

  Pada 6 Desember 2010, Android versi 2.3 *(Gingerbread)* diluncurkan. Perubahan-perubahan umum yang didapat dari Android versi ini antara lain peningkatan kemampuan permainan *(gaming)*, peningkatan fungsi *copy paste*, layar antar muka *(User Interface)* didesain ulang, dukungan format video VP8 dan WebM, efek audio baru *(reverb, equalization, headphone virtualization, dan bass boost)*, dukungan kemampuan *Near Field Communication (NFC)*, dan dukungan jumlah kamera yang lebih dari satu.

- Android versi 3.0/3.1 *(Honeycomb)*

  Android *Honeycomb* dirancang khusus untuk tablet. Android versi ini mendukung ukuran layar yang lebih besar. *User Interface* pada *Honeycomb* juga berbeda karena sudah didesain untuk tablet. *Honeycomb* juga mendukung multi prosesor dan juga akselerasi perangkat keras *(hardware)* untuk grafis. Tablet pertama yang dibuat dengan menjalankan *Honeycomb* adalah Motorola Xoom.

- Android versi 4.0 *(ICS : Ice Cream Sandwich)*

  Diumumkan pada tanggal 19 Oktober 2011, membawa fitur *Honeycomb* untuk *smartphone* dan menambahkan fitur baru termasuk membuka kunci dengan pengenalan wajah, jaringan data pemantauan penggunaan dan kontrol,

terpadu kontak jaringan sosial, perangkat tambahan fotografi, mencari *email* secara *offline*, dan berbagi informasi dengan menggunakan *Near Field Communication (NFC)*.

2.11 Java [21]

Sebagai sebuah bahasa pemrograman, Java dapat membuat seluruh bentuk aplikasi, desktop, web dan lainnya, sebagaimana dibuat dengan menggunakan bahasa pemrograman konvensional yang lain. Java adalah bahasa pemrograman yang berorientasi objek (OOP) dan dapat dijalankan pada berbagai *platform* sistem operasi. Perkembangan Java tidak hanya terfokus pada satu sistem operasi, tetapi dikembangkan untuk berbagai sistem operasi dan bersifat *open source.*

Sebagai sebuah peralatan pembangun, teknologi Java menyediakan banyak tools : compiler, interpreter, penyusun dokumentasi, paket kelas dan sebagainya.

Aplikasi dengan teknologi Java secara umum adalah aplikasi serba guna yang dapat dijalankan pada seluruh mesin yang memiliki *Java Runtime Environment* (JRE).

Terdapat dua komponen utama dari *Deployment Environment*. Yang pertama adalah JRE, yang terdapat pada paket J2SDK, mengandung kelas–kelas untuk semua paket teknologi Java yang meliputi kelas dasar dari Java, komponen GUI dan sebagainya. Komponen yang lain terdapat pada Web Browser. Hampir seluruh Web Browser komersial menyediakan *interpreter* dan *runtime environment* dari teknologi Java.

2.11.1 Karakterisktik Java [21]

Berdasarkan *white paper* resmi dari SUN, Java memiliki karakteristik berikut:

1. Sederhana

   Bahasa pemrograman Java menggunakan sintaks mirip dengan C++ namun sintaks pada Java telah banyak diperbaiki terutama menghilangkan penggunaan *pointer* yang rumit dan *multiple inheritance*. Java juga menggunakan *automatic memory allocation* dan *memory garbage collection.*

2. Berorientasi objek *(Object Oriented)*

   Java mengunakan pemrograman berorientasi objek yang membuat program dapat dibuat secara modular dan dapat dipergunakan kembali. Pemrograman berorientasi objek memodelkan dunia nyata kedalam objek dan melakukan interaksi antar objek-objek tersebut.

3. Dapat didistribusi dengan mudah

   Java dibuat untuk membuat aplikasi terdistribusi secara mudah dengan adanya *libraries networking* yang terintegrasi pada Java.

4. *Interpreter*

   Program Java dijalankan menggunakan interpreter yaitu *Java Virtual Machine* (JVM). Hal ini menyebabkan *source code* Java yang telah dikompilasi menjadi Java *bytecodes* dapat dijalankan pada *platform* yang berbeda-beda.

5. *Robust*

   Java mempuyai reliabilitas yang tinggi. *Compiler* pada Java mempunyai kemampuan mendeteksi *error* secara lebih teliti dibandingkan bahasa

pemrograman lain. Java mempunyai *runtime-Exception handling* untuk membantu mengatasi *error* pada pemrograman.

6. Aman

   Sebagai bahasa pemrograman untuk aplikasi internet dan terdistribusi, Java memiliki beberapa mekanisme keamanan untuk menjaga aplikasi tidak digunakan untuk merusak sistem komputer yang menjalankan aplikasi tersebut.

7. *Architecture Neutral*

   Program Java merupakan *platform independent*. Program cukup mempunyai satu buah versi yang dapat dijalankan pada *platform* yang berbeda dengan *Java Virtual Machine.*

8. *Portabel*

   Source code maupun program Java dapat dengan mudah dibawa ke *platform* yang berbeda-beda tanpa harus dikompilasi ulang.

9. *Performance*

   Performance pada Java sering dikatakan kurang tinggi. Namun *performance* Java dapat ditingkatkan menggunakan kompilasi Java lain seperti buatan *Inprise, Microsoft* ataupun *Symantec* yang menggunakan *Just In Time Compilers* (JIT).

10. *Multithreaded*

    Java mempunyai kemampuan untuk membuat suatu program yang dapat melakukan beberapa pekerjaan secara sekaligus dan simultan.

11. Dinamis

    Java didesain untuk dapat dijalankan pada lingkungan yang dinamis. Perubahan pada suatu class dengan menambahkan *properties* ataupun *method* dapat dilakukan tanpa menggangu program yang menggunakan *class* tersebut.

2.12 Skala Pengukuran

Skala pengukuran merupakan kesepakatan yang digunakan sebagai acuan untuk menentukan panjang pendeknya interval yang ada dalam alat ukur, sehingga alat ukur tersebut bila digunakan dalam pengukuran akan menghasilkan data kuantitatif [25]. Dengan skala pengukuran ini, maka nilai variabel yang diukur 10 dengan instrumen tertentu dapat dinyatakan dalam bentuk angka, sehingga akan lebih akurat, efisien, dan komunikatif. Berbagai skala yang dapat digunakan untuk penelitian adalah [25]:

1. Skala *Likert*

Skala *Likert* digunakan untuk mengukur sikap, pendapat, dan persepsi seseorang atau sekelompok orang tentang fenomena sosial. Dengan skala *Likert*, maka variabel yang akan diukur dijabarkan menjadi indikator variabel. Kemudian indikator tersebut dijadikan sebagai titik tolak untuk menyusun item–item instrumen yang dapat berupa pernyataan atau pertanyaan. Jawaban setiap item instrumen yang menggunakan skala likert mempunyai gradasi dari sangat positif sampai sangat negatif, yang dapat berupa kata–kata antara lain: sangat setuju

dengan skor 5, setuju dengan skor 4, ragu–ragu dengan skor 3, tidak setuju dengan skor 2, sangat tidak setuju dengan skor1.

Instrument peneletian yang menggunakan skla likert dapat dibuat dalam bentuk *checklist* ataupun pilihan ganda.

Contoh :

Berilah jawaban pertanyaan berikut sesuai dengan pernyataan anda, dengan cara memberi tanda (√) pada kolom yang tersedia.

Tabel 2.1 Tabel pertanyaan skala likert

No.	Pertanyaan	Jawaban				
		SS	ST	RG	TS	STS
1. 2.	Prosedur kerja yang baru itu akan segera diterapkan di perusahaan anda.		√			

SS = Sangat Setuju diberi skor 5

ST = Setuju diberi skor 4

RG = Ragu-ragu diberi skor 3

TS = Tidak Setuju diberi skor 2

STS = Sangat Tidak Setuju diberi skor 1

Kemudian dengan teknik pengumpulan data angket, maka instrument tersebut misalnya diberikan kepada 100 orang karyawan yang diambil secara random. Dari 100 orang pegawai setelah dilakukan analis misalnya:

25 Orang menjawab SS

40 Orang menjawab ST

5 Orang menjawab RG

20 Orang menjawab TS

10 Orang menjawab STS

Berdasarkan data tersebut 65 orang (40+25) atau 65% karyawan menjawab setuju dan sangat setuju. Jadi kesimpulannya mayoritas karyawan setuju adanya metode kerja baru.

Data interval tersebut juga dapat dianalisis dengan menghitung rata-rata jawaban berdasarkan skoring setiap jawaban dari responden. Berdasarkan skor yang telah ditetapkan dapat dihitung sebagai berikut :

Jumlah skor untuk 25 orang yang menjawab SS = 25 x 5= 125

Jumlah skor untuk 40 orang yang menjawab ST = 40 x 4= 160

Jumlah skor untuk 5 orang yang menjawab RG = 5 x 3 = 15

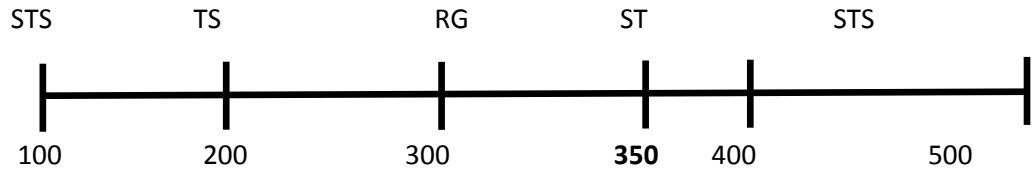
Jumlah skor untuk 20 orang yang menjawab TS = 20 x 2= 40

Jumlah skor untuk 10 orang yang menjawab STS = 10 x 1= 10

Jumlah total = 350

Jumlah skor ideal (kriterium) untuk seluruh item = 5 x 100 = 500 (seandainya semua menjawab SS). Jumlah skor yang diperoleh dari penelitian 350. Jadi berdasarkan data tu maka tingkat persetujuan terhadap metode kerja baru itu = (350 : 500) x 100% = 70% dari yang diharapkan (100%)

Secara kontinum dapat digambarkan seperti berikut :



Gambar 2.25 Gambar Interpretasi Skor Persetujuan Metode Kerja Baru

Jadi berdasarkan data yang diperoleh dari 100 responden maka rata-rata 350 terletak pada daerah setuju.

2. Skala *Guttman*

Skala pengukuran dengan tipe ini, akan didapat jawaban yang tegas, yaitu ”ya–tidak”, ”benar–salah”, ”pernah–tidak pernah”,”positif–negatif” dan lain–lain. Data yang diperoleh dapat berupa data interval atau rasio 11 dikhotomi (dua alternatif). Jadi kalau pada skala *likert* terdapat 3, 4, 5, 6, 7 interval, dari kata “sangat setuju” sampai “sangat tidak setuju”, maka pada skala Guttman hanya ada dua interval yaitu ”setuju” atau ”tidak setuju”.

Penelitian menggunakan skala Guttman dilakukan bila ingin mendapatkan jawaban yang tegas terhadap suatu permasalahan yang ditanyakan.

Contoh :

1. Bagaimana pendapat anda, bila orang itu menjabat pimpinan di perusahaan ini ?
   a. Setuju
   b. Tidak setuju

2. Pernahkah pimpinan melakukan pemeriksaan di ruang kerja anda ?
    a. Tidak pernah
    b. Pernah

3. Rating Scale

Dari ketiga skala pengukuran seperti yang telah dikemukan, data yang diperoleh semuanya adalah data kualitatif yang kemudian dikuantitatifkan. Tetapi dengan *rating scale* data mentah yang diperoleh berupa angka kemudian ditafsirkan dalam pengertian kualitatif. Yang penting bagi penyusun instrumen dengan *rating scale* adalah harus dapat mengartikan setiap angka yang diberikan pada alternatif jawaban pada setiap item instrumen.

Contoh :

Seberapa baik data ruang kerja yang ada di Perusahaan A ?

Berilah jawaban dengan angka :

4. Bila tata ruang itu sangat baik
3. Bila tata ruang itu cukup baik
2. Bila tata ruang itu kurang baik
1. Bila tata ruang itu tidak baik

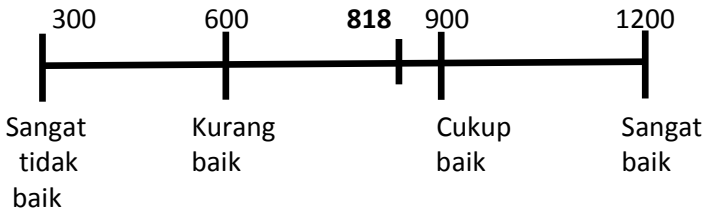
Jawaban dengan melingkari nomor jawaban yang tersedia sesuai dengan keadaan yang sebenarnya

Tabel 2.2 Tabel pertanyaan skala *rating scale*

No. Item	Pertanyaan tentang tata ruang kantor	Interval Jawaban			
1.	Penataan meja kerja sehingga arus kerja menjadi pendek	4	3	2	1
2.	Pencahayaan alam tiap ruangan	4	3	2	1
3.	Pencahayaan buatan/listrik tiap ruang sesuai dengan kebutuhan	4	3	2	1
4.	Warna lantai sehingga tidak menimbulkan pantulan cahaya yang dapat mengganggu pegawai	4	3	2	1
5.	Sirkulasi udara setiap ruangan	4	3	2	1
6.	Keserasian warna alat-alat kantor, perabot dengan ruangan	4	3	2	1
7.	Penempatan lemari arsip	4	3	2	1
8.	Penempatan ruangan pimpinan	4	3	2	1
9.	Meningkatkan keakraban sesama pegawai	4	3	2	1
10.	Kebersihan ruangan	4	3	2	1

Jumlah skor kriterium (bila setiap butir mendapat skor tertinggi) = 4 x 10 x 30 = 1200. Untuk ini skor tertinggi tiap butir = 4, jumlah butir = 10 dan jumlah responden = 30.

Jumlah skor hasil pengumpulan data = 818. Dengan demikian kualitas tata ruang kantor lembaga A menurut persepsi 30 responden itu 818 : 1200 = 68% dari kriteria yang ditetapkan. Hal ini secara kontinum dapat dibuat kategori sebagai berikut.



Gambar 2.26 Gambar Interpretasi Skor Persetujuan Tata Ruang Baru

Nilai 818 termasuk dalam kategori interval “kurang baik dan cukup baik”. Tetapi lebih mendekati cukup baik.

4. Semantic Deferential

Skala pengukuran yang berbentuk *Semantic Deferential* dikembangkan oleh Osgood. Skala ini juga digunakan untuk mengukur sikap, hanya bentuknya tidak pilihan ganda maupun *checklist*, tetapi tersusun dalam satu garis kontinum yang jawabannya sangat positifnya terletak dibagian kanan garis, dan jawabannya sangat negatif terletak 12 di bagian kiri garis, atau sebaliknya.

Data yang diperoleh adalah data interval, dan biasanya skala ini digunakan untuk mengukur sikap atau karakteristik tertentu yang dipunyai oleh seseorang.