BAB 2

LANDASAN TEORI

2.1 Sinyal Analog dan Sinyal Digital

Secara umum, sinyal didefinisikan sebagai suatu besaran fisis yang merupakan fungsi waktu, ruangan atau beberapa variabel. Menurut *Stoneytiti,* sinyal adalah kuantitas terukur yang rentang waktuny atau spasial yang bervariasi. Sebuah sinyal dapat dinyatakan sebagai fungsi dari waktu dan frekuensi.

Sinyal analog bekerja dengan mentransmisikan suara dan gambar dalam bentuk gelombang kontinu (continous varying). Dua parameter/karakteristik terpenting yang dimiliki oleh isyarat analog adalah amplitudo dan frekuensi. Isyarat analog biasanya dinyatakan dengan gelombang sinus, mengingat gelombang sinus merupakan dasar untuk semua bentuk isyarat analog. Hal ini didasarkan kenyataan bahwa berdasarkan analisis *fourier*, suatu sinyal analog dapat diperoleh dari perpaduan sejumlah gelombang sinus. 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 variabel dasar, yaitu amplitudo, frekuensi dan phasa.

- Amplitudo merupakan ukuran tinggi rendahnya tegangan dari sinyal analog.
- Frekuensi adalah jumlah gelombang sinyal analog dalam satuan detik.
- Phasa adalah besar sudut dari sinyal analog pada saat tertentu.

Salah satu contoh sinyal analog yang paling mudah adalah suara.

Sinyal digital merupakan hasil teknologi yang dapat mengubah signal 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 diskret. 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.

Sistem digital merupakan bentuk sampling dari sitem analog. digital pada dasarnya di kodekan 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.

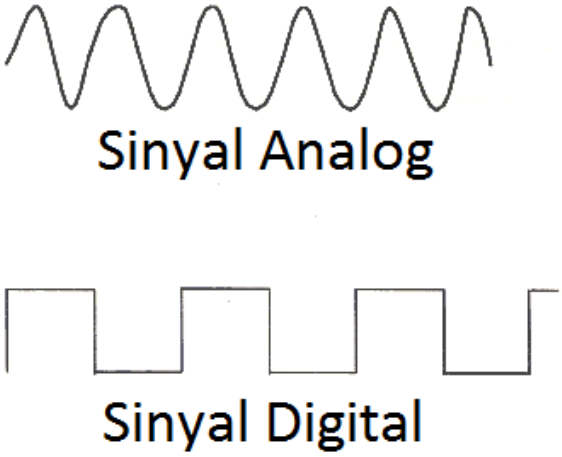
Sinyal digital ini memiliki berbagai keistimewaan yang unik yang tidak dapat ditemukan pada teknologi analog yaitu :

- 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.



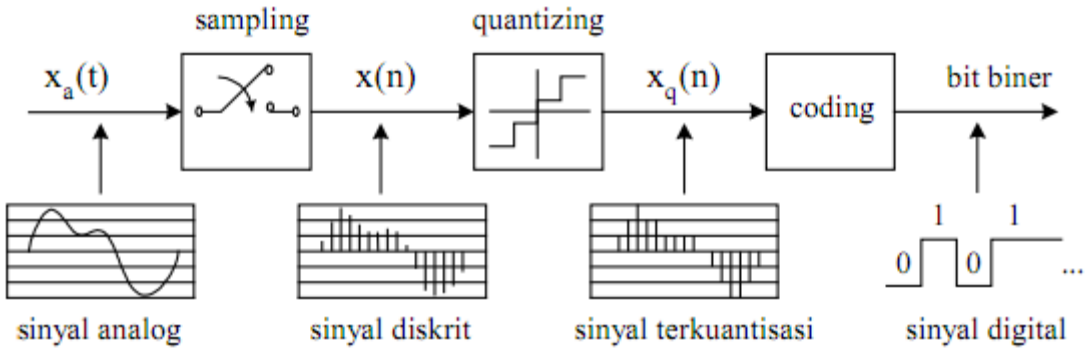
Gambar 2.1 Bentuk Sinyal Analog dan Sinyal Digital

Mengacu pada pendapat Stephen Cook(Cornelius Arianto, 2010),Ada dua faktor penting selama proses sinyal analog diubah menjadi sinyal digital. Pertama adalah "*sample rate*", atau seberapa sering untuk merekam nilai-nilai tegangan. Kedua, adalah

"*bit* per sampel", atau seberapa akurat nilai dicatat. Item ketiga adalah jumlah saluran (mono atau stereo), tetapi untuk aplikasi yang paling ASR(*Automatic Speech Recognition*) mono sudah cukup. Pengembang harus bereksperimen dengan nilai yang berbeda untuk menentukan apa yang terbaik dengan algoritma mereka.

2.2 Konsep Dasar ADC (*Analog to Digital Converter*)

Sebuah ADC (*Analog to Digital Converter*) berfungsi untuk mengkodekan tegangan sinyal analog waktu kontinu ke bentuk sederetan bit digital waktu diskrit sehingga sinyal tersebut dapat diolah oleh komputer. Proses konversi tersebut dapat digambarkan sebagai proses 3 langkah



Gambar 2.2 Proses konversi Sinyal Analog ke Sinyal Digital

2.2.1 *Sampling* (pencuplikan)

Merupakan konversi suatu sinyal analog waktu-kontinu, xa(t), menjadi sinyal waktu-diskrit bernilai kontinu, x(n), yang diperoleh dengan mengambil "cuplikan" sinyal waktu kontinu pada saat waktu diskrit. Secara matematis dapat ditulis :

$$x(n) = xa(nT)$$

Dimana :

T = interval pencuplikan (detik) n = bilangan bulat, $-\infty < n < \infty$

2.2.2 *Quantizing* (kuantisasi)

Merupakan konversi sinyal waktu-diskrit bernilai-kontinu, x(n), menjadi sinyal waktu-diskrit bernilai-diskrit, $x_q(n)$. Nilai setiap waktu kontinu dikuantisasi atau dinilai dengan tegangan pembanding yang terdekat. Selisih antara cuplikan x(n) dan sinyal terkuantisasi xq(n) dinamakan error kuantisasi. Tegangan sinyal input pada skala penuh dibagi menjadi 2^N tingkatan. Dimana N merupakan resolusi bit ADC (jumlah kedudukan tegangan pembanding yang ada). Untuk N = 3 bit, maka daerah tegangan input pada skala penuh akan dibagi menjadi : $2^N = 2^3 = 8$ tingkatan (level tegangan pembanding) .

2.2.3 Coding (pengkodean)

Setiap level tegangan pembanding diko an dekan ke dalam barisan bit biner. Untuk N = 3 bit, maka level tegangan pembanding = 8 tingkatan. Kedelapan tingkatan tersebut dikodekan sebagai bit-bit 000, 001, 010, 011, 100, 101, 110, dan 111.

2.3 Hidden Markov Model

Prinsip umum Hidden Markov Model (Sri Mulyana,2008) adalah memodelkan simbol kedalam sebuah mesin finite state, sehingga diketahui simbol apa yang dapat mewakili sebuah parameter vektor dari sebuah kata dimasukkan kedalam mesin, dan diestimasi berulang–ulang hingga dihasilkan parameter vektor atau observasi ot dengan mean dan kovarian yang konvergen untuk setiap statenya .

Pada implementasinya sistem pengenalan suara berbasis hidden markov model dibagi menjadi beberapa bagian sebagai berikut :

1. Data preparasi : pembentukan parameter vektor (observasi)
2. Training : inisialisasi dan estimasi parameter vector
3. Testing : pengenalan

2.3.1 Observasi

Pemisahaan kata menjadi simbol yang dilafalkan (phone) menghasilkan rangakaian observasi **o** untuk setiap kejadian yang mungkin pada saat transisi antar state. Aggap suara sebagai sebuah rangkaian vektor suara atau observasi, yang didefinisikan sebagai berikut :

$$O = o_1,o_2,o_3,....,o_T$$

Dimana ot adalah vektor suara yang diobservasi pada saat t. Observasi pada dasarnya menentukan nilai dari persamaan berikut

$$\text{Argmax}\{P(W_i,O)\}$$

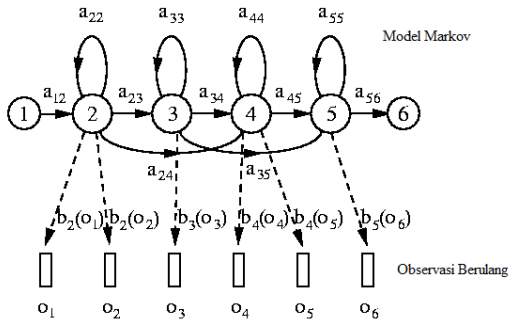
Dimana W_i adalah pengucapan yang ke-i, probabilitas ini tidak dapat dihitung secara langsung tetapi dapat dihitung dengan menggunakan aturan Bayes

$$P(W_i|O) = \frac{P(O|W_i)P(W_i)}{P(O)}$$

Maka, prioritas kemungkinan $P(W_i)$ sangat tergantung pada $P(O|W_i)$.

Dalam pengenalan suara berbasis hmm, diasumsikan bahwa rangkaian vektor observasi berkorespondensi dengan masing masing word yang dihasilkan oleh markov model . Markov model adalah mesin finite state yang mengalami perubahan state sekali setiap satuan waktu t pada saat state j dimasuki, vektor suara ot dihasilkan berdasarkan

nilai kemungkinan $b_j(o_t)$. Selanjutnya transisi antara state i ke state j juga merupakan probabilitas diskrit a_{ij}. Gambar di bawah menunjukkan contoh dari proses ini dimana lima model state berupa rangakaian state X = 1,2,3,4,5,6 untuk membangun urutan o_1 sampai o_6.



Gambar 2.3 Bentuk State Hidden Markov Model

Untuk membangun rangkaian observasi O dengan jumlah state 6. probabilitas diskrit untuk transisi dari state i ke state j ditentukan oleh a_{ij} sedangkan $b_j(o_t)$ adalah probabilitas yang membentuk observasi pada saat t (o_t) untuk state j

Probabilitas ***O*** dibangun oleh model M yang melalui seluruh urutan state X dihitung sabagai hasil perkalian antara kemungkinan transisi dan kemungkinan hasil. Jadi untuk rangkaian state X pada gambar 2.3.

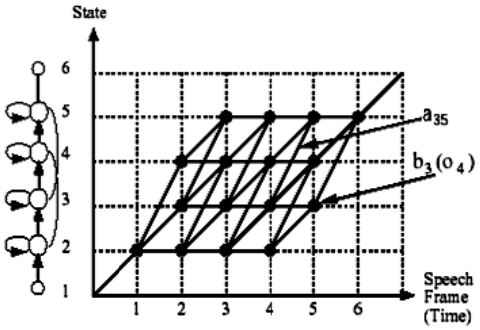
$$P(O,X|M) = a_{12}b_2(o_1)a_{22}b_2(o_2)a_{23}b_3(o_3)...$$

Meskipun demikian hanya rangkaian observasi ***O*** yang diketahui dan rangkaian state X yang mendasari adalah ters an an embunyi. Itu mengaAnpa ini disebut hidden markov model.

$$P(O|M) = \sum_{x} a_{x(0)x(1)} \prod_{t=1}^{T} b_{z(t)}(o_t) a_{x(t)x(t+1)}$$

2.3.2 Inisialisasi

Inisialisasi dapat dilakukan dengan menggunakan algoritma viterbi untuk menemukan jalur terbaik dalam sebuah matrik dimana dimensi vertikal merepresentasikan state-state hmm dan dimensi horisontal merepresentasikan frame suara. Masing masing titik pada gambar dibawah menunjukkan kemungkinan terhadap frame saat itu dan daerah antar titik menunjukkan kemungkinan transisi.



Gambar 2.4 Alur algoritma viterbi **(**Ruvinna, 2011**)**

Untuk mencari urutan state setiap observasi pada frame suara diamana a_{35} menunjukkan kemungkinan transisi dari state 3 ke state 5 dan $b_3(o_4)$ adalah probabilitas pembentukan observasi o_3 pada state 3

Kemungkian masing masing jalur dihitung dengan menjumlah kemungkinan transisi dan kemungkinan keluaran sepanjang path. Pada waktu t masing masing bagian path diketahui untuk semua state i. dapat dihitung dengan persamaan di bawah

Konsep path ini sangat berguna untuk suara kontinyu pada umumnya.

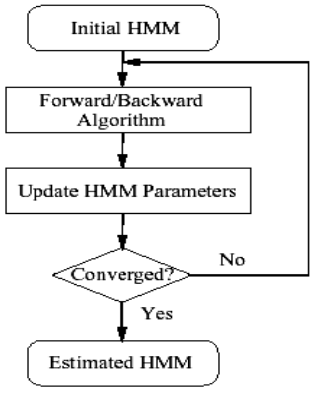
2.3.3 Estimasi

Proses estimasi dilakukan dengan menggunakan metode *Baum-Welch Re-estimation*. Formula Baum-Welch re-estimasi untuk mean dan kovarian pada masing masing state HMM adalah

dan

Estimasi dilakukan terhadap mean dan varian hmm yang mana distribusi keluaran masing masing state adalah komponen gausian, didefinisikan sebagai berikut :

Parameter vektor akan diestimasi dengan menggunakan algoritma *foreward-backward* hingga diperoleh nilai probabilitas P(O|M) terbesar berdasarkan observasi pada masing masing state. Perhitungan algoritma Baum-Welch dilakukan berdasarkan diagram alur berikut :



Gambar 2.5 Diagram Alur Estimasi (Ruvinna, 2011)

Estimasi dilakukan terhadap parameter vector pada initial HMM dengan menggunakan metode forward/backward hingga diperoleh parameter vektor yang konvergen (tidak dapt diestimasi lagi). Kriteria update adalah nilai probabiltias observasi terhadap model P(O|M) lebih tinggi dari nilai iterasi sebelumnya.

Nilai kemungkinan foreward $\alpha_j(t)$ untuk beberapa model M dan N state didefinisaikan sebagai

$$\alpha_j(t) = P(o_1, \dots, o_t, x(t) = j \mid M)$$

kemungkinan ini dapat dihitung berdasarkan rumus :

$$\alpha_j(t) = \left[\sum_{i=2}^{N-1} \alpha_i(t-1)a_{ij}\right] b_j(t)$$

sedangkan nilai kemungkinan backward $\beta_j(t)$ untuk model M dan N state didefinisikan sebagai $\beta_j(t) = P(o_{t+1}, \dots, o_T, x(t) = j \mid M)$

dan dapat dihitung dengan persamaan :

$$\beta_i(t) = \sum_{i=2}^{N-1} a_{ij} b_j(o_{t+1})\, \beta_j(t+1)$$

berdasarkan persamaan

$$\alpha_j(t)\beta_j(t) = P(O, x(t) = j|M)$$

maka didapat persamaan untuk menentukan nilai probabilitas $L_j(t)$ sebagai berikut :

$$\begin{aligned} L_j(t) &= P(x(t) = j|O, M) \\ &= \frac{P(O, x(t) = j|M)}{P(O|M)} \\ &= \frac{1}{P}\alpha_j(t)\beta_j(t) \end{aligned}$$

dimana P = P(*O*|*M*).

Algoritma untuk membentuk re-estimasi parameter hmm dengan Baum-Welch re-estimasi adalah sebagai berikut :

1. Untuk setiap vektor parameter/matrik, alokasikan storage untuk pembilang dan penyebut formula Baum-Welch sebagai acumulator.
2. Hitung kemungkinan foreward dan backward untuk semua state j pada waktu t.
3. Untuk setiap state j dan waktu t, gunakan probabiltas Lj(t) dan vektor observasi saat ini o_t untuk merubah acumulator pada state itu.
4. Gunakan nilai acumulator terakhir untuk menghitung nilai parameter yang baru.

Jika nilai P = P($O|M$) iterasi saat ini kurang dari iterasi sebelumnya maka berhenti jika tidak ulangi langkah diatas dengan menggunakan nilai parameter yang baru.

2.4 A/D dan HMM

Dalam permasalahan ini konversi Sinyal Analog menjadi Sinyal Digital (A/D) akan telah menghasilkan nilai dalam bentuk biner yang dimana akan digunakan dalam Hidden Markov Model, dengan data yang telah didapatkan akan diperbandingkan dengan seluruh kemungkinan yang ada dan akan diulang hingga mendapatkan hasil yang maksimal dengan kemungkinan error yang terkecil.

2.5 Data Learning

Setelah dirubah signalnya maka saatnya untuk membuat pembelajaran tentang apa yang dimasukkan, menurut Evandro Gouvêa pada tahun 2008, sinyal digital yang diterima memiliki 8 input untuk tiap huruf yang diberikan akan divariasikan dengan 3 hidden layer dan hanya menghasilkan satu output,berikut adalah salah satu contohnya:

Kata : CAT THESE RAT THAT

Memiliki grammar sebagai berikut : <start> K AE T DH IY Z R AE T DH AE T <end>

Setiap state memiliki kemungkinan untuk dituliskan, berikut adalah pemaparan pertama untuk 3 hidden layer dengan HMM sebagai inisialisasi awal

K(sil,AE) 0 1 2
AE(K,T) 3 4 5
T(AE,DH) 6 7 8
DH(T,IY) 9 10 11
IY(DH,Z) 12 13 14
Z(IY,R) 15 16 17
R(Z,AE) 18 19 20
AE(R,T) 21 22 23
DH(T,AE) 24 25 26
AE(DH,T) 27 28 29
T(AE,sil) 30 31 32

Setiap state selalu dihitung state baru, sehingga seluruhnya adalah kemungkinan yang akan terjadi dalam hal ini akan menghasilkan 33 state karena setiap penamaan state dianggap benar dan baru ,selanjutnya akan dilakukan Reduksi pertama dengan mengambil ulang state yang sudah diulang pada hidden layer pertama dengan 2 variasi yang berbeda

K(sil,AE) 0 1 2
AE(K,T) 3 4 5
T(AE,DH) 6 7 8
DH(T,IY) 9 10 11
IY(DH,Z) 12 13 14
Z(IY,R) 15 16 17
R(Z,AE) 18 19 20
AE(R,T) 21 22 5
DH(T,AE) 9 23 24
AE(DH,T) 25 26 5
T(AE,sil) 6 27 28

Dalam hal ini, setiap state yang sudah diulang di reduksi contoh AE(K,T) dengan AE(R,T) maka layer ketiga akan diubah sesuai dengan nilai di layer ketiga yang pertama kali,dengan cara yang sama diulang ke seluruh state sehingga hanya menjadi 29 state saja. Akan dilakukan reduksi tahap ketiga ,

K(sil,AE)	0	1	2
AE(K,T)	3	4	5
T(AE,DH)	6	7	8
DH(T,IY)	9	10	11
IY(DH,Z)	12	13	14
Z(IY,R)	15	16	17
R(Z,AE)	18	19	20
AE(R,T)	21	22	5
DH(T,AE)	9	23	11
AE(DH,T)	21	24	5
T(AE,sil)	6	25	26

Dalam tahap ini apabila ada state yang sama namun dengan pemilihan yang jauh berbeda , maka nilai yang berbeda pada kemungkinan ketiga akan dibuah sama dengan state awal yang sudah keluar dan di definisikan sebelumnya. Dan dalam langkah ini menghasilkan 27 state yang dimana semakian sedikit dan setelah itu akan masuk ke tahap proses estimasi dengan state yang sudah di reduksi dimana kemungkinan terbaiklah yang akan terpilih dan ditampilkan.

2.6 WWW(*World Wide Web*)

Menurut Berners-Lee(1991),Definisi WWW **(** *World Wide Web* **)** adalah suatu ruang informasi yang yang dipakai oleh pengenal global yang disebut *Uniform Resource Identifier* (URI) untuk mengidentifikasi sumber-sumber daya yang

berguna. WWW sering dianggap sama dengan Internet secara keseluruhan, walaupun sebenarnya ia hanyalah bagian daripadanya.

Fungsi WWW adalah menyediakan data dan informasi untuk dapat digunakan bersama. WWW atau *World Wide Web adalah* suatu program yang ditemukan oleh Tim Berners-Lee pada tahun 1991. Awalnya Berners-Lee hanya ingin menemukan cara untuk menyusun arsip-arsip risetnya. Untuk itu, dia mengembangkan suatu sistem untuk keperluan pribadi. Sistem itu adalah program peranti lunak yang diberi nama *Equire*. Dengan program itu, Berners-Lee berhasil menciptakan jaringan terkait antara berbagai arsip sehingga memudahkan informasi yang dibutuhkan. Inilah yang kemudian menjadi dasar dari sebuah revolusi yang dikenal sebagai web.

WWW dikembangkan pertama kali di Pusat Penelitian Fisika Partikel Eropa (CERN), Jenewa, Swiss. Pada tahun 1989 Berners-lee membuat proposal untuk proyek pembuatan hypertext secara global, kemudian pada bulan Oktober 1990, '*World Wide Web*' sudah bisa dijalankan dalam lingkungan CERN. Pada musim panas tahun 1991, WWW resmi digunakan secara luas pada jaringan Internet.

2.7 JSP (Java Server Page)

JSP merupakan perluasan dari spesifikasi Java Servlet, yang dalam web programming yang bersifat server side seperti halnya PHP dan ASP. JSP dapat berupa gabungan antara baris HTML dan fungsi-fungsi dari JSP itu sendiri. Berbeda dengan Servlet yang harus dikompilasi oleh USER menjadi class sebelum dijalankan, JSP tidak perlu dikompilasi oleh USER tapi SERVER yang akan melakukan tugas tersebut. Pada saat user membuat pertama kali atau melakukan modifikasi halaman dan mengeksekusinya pada web browser akan memakan sedikit waktu sebelum ditampilkan.

JSP memiliki 3 fase alur : isialisasi, servis dan dekstruksi. Fafe-fase ini sama dengan method servlet yang diambil dari container yang berbeda : jspInit() untuk inisialisasi fase, _jspService() untuk fase servis, dan jspDestroy() untuk mendestruksi fase.

Meskipun JSP berbasis Java, dan dikendalikan sebagai kode Java oleh servlet, memperbolehkan pengembang untuk menggunakan syntax yang berbeda pada spesifikasi Java 2.0 dan sebagai gantinya menggunakan aturan spesifikasi JSP. Bagian berikut ini menggambarkan syntax JSP dengan lebih detail.

Semua komponen Java Server Pages dapat dibagi menjadi dua kategori umum: elements dan templates data. Element merupakan data yang dapat menghasilkan informasi. Data template merupakan informasi statis yang tercetak dalam bentuk presentasi/tampilan. Ekspresi JSP, <%= new java.util.Date()%> adalah satu-satunya element data yang memanggil data template.

JSP memiliki dua tipe sintak, dua tipe dari authoring JSP ini didukung oleh Container JSP : JSP Style dan XML Style. Memilih salah satu format sintaks hanya bergantung dari *preference* dan standarisasi. Sintaks normal didesain lebih mudah untuk pada pembuat. *XML-compatible* syntax telah disediakan ketika menggunakan *JSP authoring tools*. Bagaimanapun juga, yang lebih sering disediakan adalah sintaks normal karena lebih mudah untuk dibaca dan dimengerti.

Seperti yang telah dijelaskan pada bab sebelumnya, JSP memungkinkan untuk dilihat sebagai HTML atau XML dokumen dengan berdasar pada Script JSP. Scripting JSP element memperbolehkan anda memasukkan kode Java ke dalam Servlet yang akan di-generate dari halaman JSP. Cara termudah untuk membuat dynamic JSP adalah dengan menaruh *scripting element* ke dalam data template. *Sricpt* element terdiri dari:

a. **Scriplet**, berisi statement-statement yang merupakan logika dari suatuproses. Pembatas yang digunakan pada scriplet adalah : <% // statement %>

b. **Deklarasi**, gunanya untuk mendeklarasikan variabel atau method. Untuk deklarasi digunakan pembatas sebagai berikut <%! // variabel // method %> pada JSP, seperti juga Java variabel bersifat strong type, artinya apabila ingin menggunakan suatu variabel maka harus melalui proses mendeklarasian tipe dari variabel tersebut.

c. **Ekspresi**, berguna untuk menampilkan nilai dari suatu variabel atau method. Pembatas yang digunakan pada suatu ekspresi adalah: <%=%>

2.8 Apache Ant

Apache Ant adalah sebuah perangkat lunak untuk mengotomatisasi proses membangun perangkat lunak. Hal ini mirip dengan Membuat tetapi diimplementasikan dengan menggunakan bahasa Java, memerlukan platform Java,dan sangat cocok untuk membangun proyek-proyek Java.

Perbedaan yang paling segera terlihat antara *Ant* dan *Make* adalah Ant yang menggunakan XML untuk menggambarkan proses membangun dan dependensinya, sedangkan *Make* menggunakan format *Makefile*. Secara default file XML bernama build.xml. Ant adalah proyek Apache. Ini adalah perangkat lunak open source, dan dirilis di bawah Lisensi Apache Software.

2.9 JSAPI

JSAPI adalah javaTM Speech API(Application Programming Interface). Dimana memperbolehkan pembuat untuk menyalakan segala perangkat yang berhubungan dengan suara dalam platform apapun yang mendukung JSAPI. JSAPI didefinisikan dengan standar tinggi yang telah diberikan, mudah digunakan, cocok untuk banyak platform. Sistem teknologi *speech* yang dapat dilakukan dalam JSAPI adalah *speech recognition* and *speech synthesis*.

Speech recognition menyediakan komputer dengan kemampuan untuk mendengarkan bahasa lisan dan untuk menentukan apa yang telah dikatakan. Dengan kata lain, itu proses input audio yang berisi kalimat dengan mudah diubah menjadi teks. *Speech Recognition* menyediakan proses kebalikan dari menulis kalimat dari teks yang dihasilkan oleh aplikasi, servlet atau pengguna. Hal ini sering disebut sebagai *text-to-speech technology*.

Perusahaan dan individu dapat manfaat dari berbagai macam aplikasi dari teknologi pidato menggunakan API Java *Speech*. Sebagai contoh, sistem suara interaktif respon adalah alternatif yang menarik untuk pengganti antar muka melalui telepon, sistem dikte dapat jauh lebih cepat dari input diketik untuk banyak pengguna,teknologi berbicara meningkatkan aksesibilitas ke komputer bagi banyak orang dengan keterbatasan fisik.

Tampilan *Speech* memberikan pengembang aplikasi Java kesempatan untuk menerapkan kepribadian yang berbeda dan menarik untuk aplikasi mereka dan untuk membedakan produk mereka. Pengembang aplikasi Java akan memiliki akses ke Negara dengan teknologi *speech recognition* dari perusahaan terkemuka.Dengan API

standar untuk *speech*, pengguna dapat memilih produk yang terbaik *speech* yang memenuhi kebutuhan mereka dan anggaran mereka.

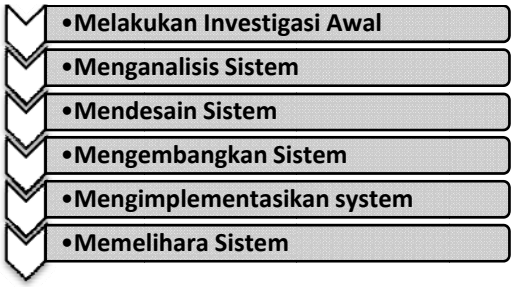
JSAPI dikembangkan melalui proses pengembangan terbuka. Dengan keterlibatan aktif dari perusahaan teknologi pidato terkemuka, dengan masukan dari pengembang aplikasi dan dengan bulan meninjau publik dan komentar, spesifikasitelah mencapai tingkat tinggi keunggulan teknis.

Sebagai spesifikasi untuk teknologi berkembang pesat, Sun akan mendukung dan meningkatkan Java API untuk mempertahankan kemampuan Pidato terkemuka.

JSAPI merupakan ekstensi untuk platform Java. Ekstensi adalah paket kelas yang ditulis dalam bahasa pemrograman Java (dan kode asli yang terkait) yang pengembang aplikasi dapat digunakan untuk memperluas fungsionalitas dari bagian inti dari platform Java.

2.10 Software Development Life Cycle

Menurut Henry C.Lucas Junior(Eny Fitri Asih, 2010), *Software Development Life Cycle*(SDLC) Merupakan suatu bentuk yang digunakan untuk menggambarkan tahapan utama dan langkah-langkah di dalam tahapan dalam proses pengembangannya. Pada sistem life cycle tiap-tiap bagian dari pengembangan sistem dibagi menjadi beberapa tahapan kerja.



Gambar 2.6 Diagram SDLC

Analisis dan desain sistem merupakan prosedur pemecahan maslah yang terdiri dari enam fase untuk meneliti sistem informasi dan meningkatkannya.Keenam fase tersebut membentuk apa yang disebut siklus hidup pengembangan sistem. Siklus hidup pengembangan sistem SDLC adalah proses langkah demi langkah yang diikuti oleh banyak organisasi selama analisis dan desain sistem.

2.10.1 Fase Pertama : Melakukan Investigasi Awal

Empat langkah yang ada pada fase pertama:

Tujuan dari fase pertama ini adalah melakukan analisis awal, mencari alternative solusi, mendeskripsikan biaya dan keuntungn, dan menyerahkan rencana awal dengan beberapa rekomendasi. Empat langkah fase pertama ialah:

1. **Melakukan analisis awal**, anda perlu mencari apa yang menjadi tujuan organisasi dan sifat serta cakupan masalah, selanjutnya melihat apakah masalah yang dipelajari cocok dengan tujuan tersebut.
2. **Mengajukan solusi-solusi alternatif,** Solusi-solusi alternatif dapat diperoleh dengan mewawancarai orang dalm organisasi, klien atau pelanggan yang terpengaruh oleh sistem, pemasok dan konsultan.
3. **Mendeskripsikan biaya dan keuntungan** , anda perlu mendaftarkan biaya maupun keuntungan secara terperinci. Biaya akan tergantung dari keuntungan yang bisa menawarkan penghematan.
4. **Menyerahkan rencana awal**, Semua yang anda temukan digabung dalam suatu laporan tertulis, pembaca laporan ini bisa saja eksekutif yang punya wewenang untuk memutuskan dan menjalankan proyek.

2.10.2 Fase Kedua : Menganalisis Sistem

Tiga langkah dalam menganalisis sistem

Tujuan dari fase kedua ini adalah mengumpulkan data, menganalisis data, dan menuliskan laporan. Dalam fase ini, anda akan mengikuti arahan dari pihak managemen setelah mereka membaca laporan (fase pertama). Pihak manajemen memberi perintah untuk menganalisis atau mepelajari sistem yang sudah ada untuk memahami perbedaan sistem baru dengan sistem yang sudah ada. Tiga langkah pada tahap ini ialah:

1. **Mengumpulkan data**, dalam upaya mengumpulkan data, anda akan meninjau dokumen tertulis, mewawancarai pegawai dan manager, membuat kuesioner dan mengobservasi rang dan proses-proses di tempat kerja.
2. **Menganalisa data,** data yang telah dikumpulkan kemudian dianalisis. Ada banyak piranti analitik yang dapat dipakai, **piranti pemodelan** memungkinkan analisis sistem menampilkan representasi sistem dalam bentuk gambar, misal data flow diagram atau diagram aliran data. Dan Perangkat CASE (*Computer Aided Software Engineering)* adalah program yang mengotomatisasi berbagai aktivitas SDLC. Contoh programnya ialah *Analyst Pro, Visible Analyst* dan *Sistem Architect.*
3. **Menulis laporan**, perlu membuat laporan setelah selesai melakukan analisis. Ada 3 bagian, yang pertama, harus menjelaskan cara bekerja sistem yang sudah ada. Kedua, harus menjelaskan masalah-masalah pada sistem yang ada. Ketiga harus mendeskripsikan ketentuan-ketentuan untuk sistem baru dan memberikan rekomendasi tentang apa yang akan dilakukan selanjutnya.

2.10.3 Fase Ketiga : Mendesain Sistem

Tiga langkah ketika mendesain sistem

Tujuan fase ini adalah membuat desai awal, lalu desain yang detail, dan membuat laporan.

1. **Membuat desain awal**, desin awal mendeskripsikan kpabilitas fungsional secar umum dari sistem sistem informasi yang diusulkan. Perangkat yang digunakan pada fase ini adalah perangkat CASE dan perangkat lunak managemen proyek. Prototyping juga digunakan pada tahap ini,,prototyping adaalah pengguna workstation, perangkat CASE dan aplikasi perangkat lunak lain untuk membuat model kerja dari komponen sistem sehingga sistem baru bisa segera diuji dan dievaluasi.Jadi prototype adalah sistem dengan kemapuan kerja terbatas yang dikembangkan untuk menguji konsep-konsep desain.

2. **Membuat desain yang detail**, desain yang detail menggambarkan bagaimana sistem informasi yang diusulkan mampu memberikan kapabilitas yang digambarkan secara umum dalam desain awal.

3. **Menulis laporan,** semua pekerjaan dala desain awal dan desain yang detail akan dikemas dalam laporan yang terperinci. Anda bisa melakukan persentasi atau diskusi saat menyerahkan laporan ini kepada manajemen senior.

2.10.4 Fase Keempat : Mengembangkan Sistem

Tiga langkah yang diperlukan dalam mengembangkan sistem

1. **Mengembangkan atau mendapatkan perangkat lunak,** analisis sistem harus membuat keputusan yang disebut keputusan “membuat-atau-membeli’. Dalam keputusan tersebut, anda menentukan apakah akan membuat program – menulis

sendiri – atau embelinya, yang artinya hanya tinggal membeli paket perangkat lunak yang sudah ada.

2. **Mendapatkan perangkat lunak, setelah memilih perangkat lunak**, maka selanjutnya meng-uprade perangkat keras untuk menjalankan perangkat lunak tersebut. Namun bisa saja sistem tidak membutuhkan perangkat keras, atau perangkat keras tersebut dapat disewa tanpa harus dibeli.

3. **Menguji sistem,** dengan perangkat lunak dan perangkat keras yang telah diperoleh,maka dilakukan pengujian. Biasanya dilakukan dalam 2 tahap, yaitu :

- **Pengujian unit :** kinerja dari masing-masing bagian diteliti dengan menggunakan data uji (disusun atau sampel). Jika program ditulis sebagai usaha kerja sama dari banyak programmer, maka masing-masing bagian dari program diuji terpisah.
- **Pengujian sistem :** bagian-bagian dihubungkan bersama-sama dengan menggunakan data uji untuk mengetahui apakah bagian-bagian itu dapat bekerja sama. Sistem juga dapat diuji dengan data sesungguhnya dari organisasi.

2.10.5 Fase Kelima : Mengimplementasikan sistem

1. Konversi ke sistem baru, proses transisi dari sistem informasi yang lama ke yang baru, melibatkan konversi perangkat keras, perangkat lunak, dan file. Ada 4 strategi untuk melakukan konversi,yaitu :

- **Implementasi langsung** : pengguna hanya berhenti menggunakan sistem yang lama dan mulai menggunakan yang baru.

• **Implementasi parallel** : Sistem lama dan sistem yang baru berjalan berdampingan sampai sistem baru menunjukkan keandalannya di saat sistem lama tidak berfungsi lagi.

• **Implementasi bertahap** : bagian-bagian dari sistem baru dibuat dalam fase terpisah-entah waktu yang berbeda(parallel) atau sekaligus dalam kelompok-kelompok (langsung).

• **Implementasi pilot** : seluruh sistem dicoba, namun hanya oleh beberapa pengguna. Setelah keandalannya terbukti barulah sistem bisa diimplementasikan pada pengguna lainnya.

2. **Melatih pengguna**, ada banyak piranti yang bisa digunkan membuat pengguna membuat pengguna mengenal sistem baru dengan baik,dari dokumentasi hingga video tape hingga pelatihan diruang kelas secara langsung ataupun satu per satu.

2.10.6 Fase Keenam : Memelihara Sistem

Pemeliharaan sistem ialah menyesuaikan dan meningkatkan sistem dengan cara melakukan audit dan evaluasi secara periodic dan dengan membuat perubahan berdasarkan kondisi-kondisi baru. Meskipun pengkonversian sudah lengkap, bahkan pengguna sudah dilatih, sistem tidak bisa berjalan dengan sendirinya. Inilah tahap dimana sistem harus dimonitor untuk memastikan bahwa sistem itu berhasil. Pemeliharaan tidak hanya menjaga agar mesin tetap berjalan, namun juga meng-upgrade dan meng-update sistem agar bisa mengikuti perkembangan produk, jasa, layanan, peraturan pemerintah, dan ketentuan lain yang baru.

Setelah beberapa saat, biaya pemeliharaan akan meningkat seiring makin banyaknya usaha untuk mempertahankan sistem agar tetap responsive terhadap kebutuhan

pengguna. Dalam beberapa hal, biaya pemeliharaan ini bisa membengkak, menandakan bahwa sekaranglah saat yang tepat untuk memulai lagi SDLC.

2.11 Bentuk Program

Arsitektur program menggunakan konsep :

- Object Oriented Programming
- Procedural Programming

OOP merupakan paradigma pemrograman yang popular saat ini yang telah menggantikan teknik pemrograman berbasis prosedur. OOP. Pemrograman Berorientasi Objek (*Object Oriented Programming*) merupakan pemrograman yang berorientasikan kepada objek., dimana semua data dan fungsi dibungkus dalam class-class atau *object-object*. Setiap object dapat menerima pesan, memproses data, mengirim, menyimpan dan memanipulasi data. Beberapa *object* berinteraksi dengan saling memberikan informasi satu terhadap yang lainnya. Masing-masing *object* harus berisikan informasi mengenai dirinya sendiri dan dapat dihubungkan dengan *Object* yang lain.

Dalam *Object Oriented Programming* ada beberapa istilah yang harus anda pahami:

· ***Object***

Object adalah sesuatu yang bisa dianalogikan dengan benda, orang, tempat, kejadian atau konsep-konsep yang ada di dunia nyata yangdigunakan pada perangkat lunak atau sistem informasi. Contohnya kampus, gedung, mahasiswa, kuliah, registrasi, pembayaran dan yang lainnya.

· ***Class***

Class adalah kumpulan/himpunan *object* dengan atribut/properti yang mirip, prilaku yang mirip, serta hubungan dengan object yang lain dengan cara yang mirip.

· ***Attribut***

Attribut adalah data yang dimiliki oleh *object* dalam kelas.

2.12 *User Interface Design*

Tujuan dari UID (Uni Proboyekti, 2011) adalah merancang interface yang efektif untuk sistem perangkat lunak. Efektif artinya siap digunakan, dan hasilnya sesuai degan kebutuhan. Kebutuhan disini adalah kebutuhan penggunanya. Pengguna sering menilai sistem dari interface, bukan dari fungsinya melainkan dari *user interface*. Jika desain *user interface* tersebut buruk, maka itu sering jadi alasan untuk tidak menggunakan software. Selain itu *interface* yang buruk dapat menyebabkan pengguna membuat kesalahan fatal.

Menurut Shneiderman ,8 (delapan) aturan yang dapat digunakan sebagai petunjuk dasar yang baik untuk merancang suatu *user interface*. Delapan aturan ini disebut dengan *Eight Golden Rules of Interface Design,* yaitu:

a.Konsistensi

Konsistensi dilakukan pada urutan tindakan, perintah, dan istilah yang digunakan pada prompt, menu, serta layar bantuan.

b.Memungkinkan pengguna untuk menggunakan *shortcut*

Ada kebutuhan dari pengguna yang sudah ahli untuk meningkatkan kecepatan interaksi, sehingga diperlukan singkatan, tombol fungsi, perintah tersembunyi, dan fasilitas makro.

c. Memberikan umpan balik yang informatif

Untuk setiap tindakan operator, sebaiknya disertakan suatu sistem umpan balik. Untuk tindakan yang sering dilakukan dan tidak terlalu penting, dapat diberikan umpan balik yang sederhana. Tetapi ketika tindakan merupakan hal yang penting, maka umpan balik sebaiknya lebih substansial. Misalnya muncul suatu suara ketika salah menekan tombol pada waktu input data atau muncul pesan kesalahannya.

d. Merancang dialog untuk menghasilkan suatu penutupan

Urutan tindakan sebaiknya diorganisir dalam suatu kelompok dengan bagian awal, tengah, dan akhir. Umpan balik yang informatif akan meberikan indikasi bahwa cara yang dilakukan sudah benar dan dapat mempersiapkan kelompok tindakan berikutnya.

e. Memberikan penanganan kesalahan yang sederhana

Sedapat mungkin sistem dirancang sehingga pengguna tidak dapat melakukan kesalahan fatal. Jika kesalahan terjadi, sistem dapat mendeteksi kesalahan dengan cepat dan memberikan mekanisme yang sedehana dan mudah dipahami untuk penanganan kesalahan.

f. Mudah kembali ke tindakan sebelumnya

Hal ini dapat mengurangi kekuatiran pengguna karena pengguna mengetahui kesalahan yang dilakukan dapat dibatalkan; sehingga pengguna tidak takut untuk mengekplorasi pilihan-pilihan lain yang belum biasa digunakan.

g. Mendukung tempat pengendali internal (*internal locus of control*)

Pengguna ingin menjadi pengontrol sistem dan sistem akan merespon tindakan yang dilakukan pengguna daripada pengguna merasa bahwa sistem mengontrol pengguna. Sebaiknya sistem dirancang sedemikan rupa sehingga pengguna menjadi inisiator daripada responden.

h. Mengurangi beban ingatan jangka pendek

Keterbatasan ingatan manusia membutuhkan tampilan yang sederhana atau banyak tampilan halaman yang sebaiknya disatukan, serta diberikan cukup waktu pelatihan untuk kode, *mnemonic*, dan urutan tindakan.