BAB II

LANDASAN TEORI

2.1. Kecerdasan Buatan

Kecerdasan Buatan *(Artificial Intelegent)* merupakan cabang dari ilmu komputer yang dalam merepresentasi pengetahuan lebih banyak menggunakan bentuk simbol-simbol daripada bilangan, dan memproses informasi berdasarkan metode heuristic atau dengan berdasarkan sejumlah aturan.

2.1.1. Kecerdasan Buatan Dalam Berbagai Perspektif

Dari perspektif Kecerdasan (Intelligence) AI adalah bagaimana membuat mesin yang "cerdas" dan dapat melakukan hal-hal yang sebelumnya dapat dilakukan oleh manusia Dari perspektif bisnis, AI adalah sekelompok alat bantu (*tools*) yang berdaya guna, dan metodologi yang menggunakan *tool-tool* tersebut guna menyelesaikan masalah-masalah bisnis. Dari perspektif pemrograman (*Programming*), AI termasuk didalamnya adalah studi tentang pemrograman simbolik, pemecahan masalah, proses pencarian (*search*).

2.1.2. Domain Penelitian Dalam Kecerdasan Buatan

Adapun domain-domain dari Penelitian dalam kecerdasan buatan diantaranya adalah :

1. Permainan *(Game)*

Kebanyakan permainan dilakukan dengan menggunakan sekumpulan aturan. Dalam permainan digunakan apa yang disebut dengan pencarian ruang. Teknik untuk menentukan alternatif dalam menyimak problema ruang merupakan sesuatu

yang rumit. Teknik tersebut disebut dengan *heuristic*. Permainan merupakan bidang yang menarik dalam studi *heuristic.*

2. *Natural Language*

Suatu teknologi yang memberikan kemampuan kepada komputer untuk memahami bahasa manusia sehingga pengguna komputer dapat berkomunikasi dengan komputer dengan menggunakan bahasa sehari -hari.

3. Robotik

Robotik dan sistem sensor Sistem sensor, seperti sistem vision, sistem *tactile*, dan sistem pemrosesan sinyal jika dikombinasikan dengan AI, dapat dikategorikan kedalam suatu sistem yang luas yang disebut sistem robotik.

4. Sistem Pakar (*Expert System*)

Sistem pakar (*Expert System*) adalah program penasehat berbasis computer yang mencoba meniru proses berpikir dan pengetahuan dari seorang pakar dalam menyelesaikan masalah-masalah spesifik.

2.1.3. Konsep Dan Definisi Dalam Kecerdasan Buatan

Turing Test merupakan sebuah metode pengujian kecerdasan yang dibuat oleh Alan Turing. Proses uji ini melibatkan seorang penanya (manusia) dan dua obyek yang ditanyai. Yang satu adalah seorang manusia dan satunya adalah sebuah mesin yang akan diuji. Penanya tidak bisa melihat langsung kepada obyek yg ditanyai Penanya diminta untuk membedakan mana jawaban komputer dan mana jawaban manusia berdasarkan jawaban kedua obyek tersebut. Jika penanya tidak dapat membedakan mana jawaban mesin dan mana jawaban manusia maka Turing berpendapat bahwa mesin yang diuji tersebut dapat diasumsikan cerdas.

Dalam pemrosesan simbolik Komputer semula didisain untuk memproses bilangan/angka-angka (pemrosesan numerik). Sementara manusia dalam berpikir dan menyelesaikan masalah lebih bersifat simbolik, tidak didasarkan kepada sejumlah rumus atau melakukan komputasi matematis. Sifat penting dari AI adalah bahwa AI merupakan bagian dari ilmu komputer yang melukan proses secara *simbolik* dan *non-algoritmik* dalam penyelesaian masalah.

Istilah *Heuristic* diambil dari bahasa Yunani yang berarti menemukan Heuristic merupakan suatu strategi untuk melakukan proses pencarian (***search***) ruang problema secara selektif, yang memandu proses pencarian yang kita lakukan disepanjang jalur yang memiliki kemungkinan sukses paling besar.

Penarikan Kesimpulan (Inferencing) dimana AI mecoba membuat mesin memiliki kemampuan berpikir atau mempertimbangkan (*reasoning*) Kemampuan berpikir (*reasoning*) termasuk didalamnya proses penarikan kesimpulan (*inferencing*) berdasarkan fakta-fakta dan aturan dengan menggunakan metode heuristik atau metode pencarian lainnya.

Pencocokan Pola (*Pattern Matching*) AI bekerja dengan metode pencocokan pola (*pattern matching*) yang berusaha untuk menjelaskan obyek, kejadian (*events*) atau proses, dalam hubungan logik atau komputasional.

2.1.4. Hubungan AI Dengan Pengenalan Suara

Pemahaman Ucapan/Suara (*Speech/Voice Understanding*), adalah teknik agar komputer dapat mengenali dan memahami bahasa ucapan. Proses ini mengijinkan seseorang berkomunikasi dengan komputer dengan cara berbicara kepadanya. Istilah “pengenalan suara” mengandung arti bahwa tujuan utamanya

adalah mengenai kata yang diucapkan tanpa harus tahu artinya, di mana bagian itu merupakan tugas "pemahaman suara". Secara umum prosesnya adalah usaha untuk menerjemahkan apa yang diucapkan seorang manusia menjadi kata-kata atau kalimat yang dapat dimengerti oleh komputer.

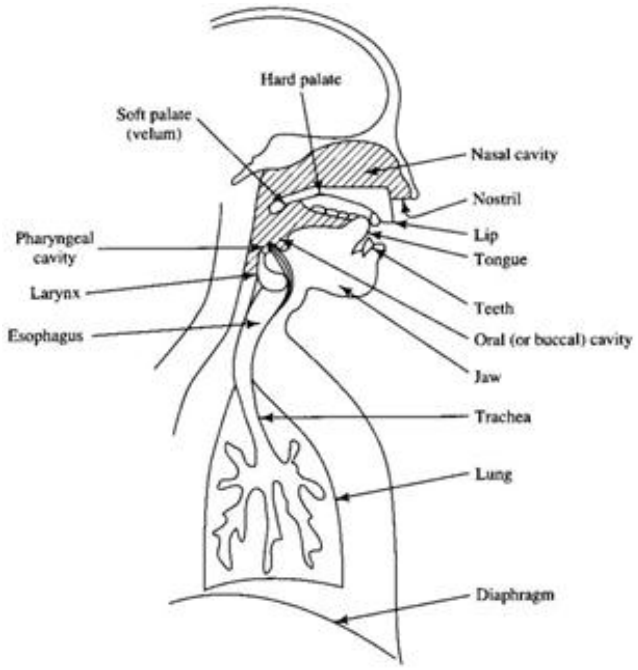
2.2. Pengenalan Suara Pada Manusia

Pada sistem pengenalan suara oleh manusia terdapat tiga organ penting yang saling berhubungan yaitu : telinga yang berperan sebagai transduser dengan menerima sinyal masukan suara dan mengubahnya menjadi sinyal syaraf, jaringan syaraf yang berfungsi mentransmisikan sinyal ke otak, dan otak yang akan mengklasifikasi dan mengidentifikasi informasi yang terkandung dalam sinyal masukan.

2.2.1. Proses Produksi Suara

Proses produksi suara pada manusia dapat dibagi menjadi tiga buah proses fisiologis, yaitu : pembentukan aliran udara dari paru-paru, perubahan aliran udara dari paru-paru menjadi suara, baik *voiced*, maupun *unvoiced* yang dikenal dengan istilah *phonation*, dan artikulasi yaitu proses modulasi/ pengaturan suara menjadi bunyi yang spesifik.

Organ tubuh yang terlibat pada proses produksi suara adalah : paru-paru, tenggorokan (*trachea*), laring (*larynx*), faring (*pharynx*), pita suara (*vocal cord*), rongga mulut (*oral cavity*), rongga hidung (*nasal cavity*), lidah (*tongue*), dan bibir (*lips*), seperti dapat dilihat pada gambar dibawah.



Gambar 2.1 Organ tubuh manusia

Organ tubuh ini dapat dikelompokkan menjadi tiga bagian utama, yaitu : *vocal tract* (berawal di awal bukaan pita suara atau glottis, dan berakhir di bibir), *nasal tract* (dari velum sampai nostril), dan *source generator* (terdiri dari paru-paru, tenggorokan, dan *larynx*). Ukuran *vocal tract* bervariasi untuk setiap individu, namun untuk laki-laki dewasa rata-rata panjangnya sekitar 17 cm. Luas dari vocal tract juga bervariasi antara 0 (ketika seluruhnya tertutup) hingga sekitar 20 cm^2. Ketika *velum*, organ yang memiliki fungsi sebagai pintu penghubung antara *vocal tract* dengan *nasal tract*, terbuka, maka secara akustik *nasal tract* akan bergandengan dengan vocal tract untuk menghasilkan suara *nasal*.

Aliran udara yang dihasilkan dorongan otot paru-paru bersifat konstan. Ketika pita suara dalam keadaan berkontraksi, aliran udara yang lewat membuatnya bergetar. Aliran udara tersebut dipotong-potong oleh gerakan pita suara menjadi sinyal pulsa yang bersifat *quasi-periodik*. Sinyal pulsa tersebut

kemudian mengalami modulasi frekuensi ketika melewati *pharynx*, rongga mulut ataupun pada rongga hidung. Sinyal suara yang dihasilkan pada proses ini dinamakan sinyal *voiced*. Namun, apabila pita suara dalam keadaan relaksasi, maka aliran udara akan berusaha melewati celah sempit pada permulaan vocal tract sehingga alirannya menjadi turbulen, proses ini akan menghasilkan sinyal *unvoiced*. Ketika sumber suara melalui *vocal tract*, kandungan frekuensinya mengalami modulasi sehingga terjadi resonansi pada *vocal tract* yang disebut *formants*. Apabila sinyal suara yang dihasilkan adalah sinyal *voiced*, terutama vokal, maka pada selang waktu yang singkat bentuk *vocal tract* relative konstan (berubah secara lambat) sehingga bentuk *vocal tract* dapat diperkirakan dari bentuk spektral sinyal *voiced*.

Aliran udara yang melewati pita suara dapat dibedakan menjadi *phonation*, bisikan, *frication*, kompresi, vibrasi ataupun kombinasi diantaranya. *Phonated excitation* terjadi bila aliran udara dimodulasi oleh pita suara. *Whispered excitation* dihasilkan oleh aliran udara yang bergerak cepat masuk ke dalam lorong bukaan segitiga kecil antara *arytenoids cartilage* di belakang pita suara yang hampir tertutup. *Frication excitation* dihasilkan oleh desakan di vocal tract. *Compression excitation* dihasilkan akibat pelepasan udara melalui vocal tract yang tertutup dengan tekanan tinggi. *Vibration excitation* disebabkan oleh udara yang dipaksa memasuki rusang selain pita suara, khususnya lidah. Suara yang dihasilkan oleh *Phonated excitation* disebut *voiced.* Suara yang dihasilkan oleh *Phonated excitation* ditambah *frication* disebut *mixed voiced*, sedangkan yang dihasilkan oleh selain itu disebut *unvoiced*. Karakteristik suara tiap individu

bersifat unik karena terdapat perbedaan dalam hal panjang maupun bentuk vocal tract.

2.2.2. Karakteristik Telinga

Telinga terbagi menjadi tiga bagian, yaitu bagian luar, tengah, dan dalam. Pinna, sebagai bagian luar telinga, berfungsi sebagai corong, untuk mengumpulkan sinyal suara menuju *auditory canal* sehingga dapat memberikan kesan arah sinyal suara yang diterima.

Auditory canal adalah struktur berbentuk pipa lurus sepanjang 2,7 cm, dengan diameter sekitar 0,7 cm, yang pada bagian ujungnya terdapat selaput membrane, yaitu gendang telinga. Membran ini merupakan pintu masuk telinga bagian tengah, yaitu ruangan berisi udara dengan volume sebesar 2 cm^3, yang terdiri dari tiga buah tulang, yaitu *malleus* (martil), *incus* (landasan), dan *stapes* (sanggurdi). Bagian ini terhubung dengan tenggorokan melalui *Eustachian tube*. Getaran pada gendang telinga ditransmisikan ke *malleus* melalui *incus*, dan *stapes*, yaitu membentuk *oval window*.

Telinga bagian dalam (*labyrinth*) memiliki tiga bagian, yaitu *vestibule* (ruang pintu masuk), *semicular canal*, dan *cochlea*. *Vestibule* terhubung dengan telinga bagian tengah melalui dua jalur, yaitu *oval window*, dan *round window*. Keduanya tertutup untuk mencegah keluarnya cairan yang mengisi telinga telinga bagian dalam. Pada *cochlea*, yang berstruktur seperti rumah siput, terdapat syaraf pendengaran. Syaraf ini memanjang sampai ke *basilar membrane*. Pada bagian atas *basilar membrane* terdapat *organ of corty* yang memiliki empat baris sel rambut (sekitar 3×10^4 sel seluruhnya).

2.2.3. Proses Pendengaran

Proses pendengaran pada telinga manusia dijelaskan sebagai berikut :

1. Sinyal suara memasuki saluran telinga dan variasi tekanan yang dihasilkannya menekan gendang telinga. Karena sisi bagian dalam dari gendang telinga mempunyai tekanan yang nilainya dijaga konstan maka gendang telinga akan bergetar.
2. Getaran dari gendang telinga disalurkan pada tiga rangkaian tulang yaitu; *martil*, *incus* dan *stapes*. Mekanisme ini dirancang untuk mengkopel variasi suara dari udara luar ke telinga bagian dalam. Karena luas permukaan penampang yang ditekan *stapes* lebih kecil dari luas penampang gendang telinga maka tekanan suara yang sampai ke telinga bagaian dalam bertambah besar.
3. Cairan pada *cochlea* bergetar dengan frekuensi yang sama dengan gelombang yang datang. *Basilar membrane* kemudian memisahkan sinyal berdasarkan frekuensinya. *Basilar membrane* berstruktur kuat dan panjang di daerah sekitar *oval window* namun bersifat lentur pada bagian ujungnya. Frekuensi resonansi yang dihasilkan membrane tersebut berbeda sepanjang dimensi *basilar membrane*. Dimana resonansi frekuensi tinggi terjadi pada bagian bagian *basilar membrane* yang berada dekat dengan *oval window*, sedangkan resonansi frekuensi rendah terjadi pada daerah ujung lainnya. Syaraf yang berada pada mambran kemudian mendeteksi posisi terjadinya resonansi yang juga akan menentukan frekuensi suara yang datang. Ukuran dari *basilar membrane* rata-rata sekitar 35 mm. Dari ukuran panjang tersebut dapat dihasilkan 10 resolusi frekuensi,

sehingga pada setiap 3.5 mm panjang membran terdapat 1 oktaf frekuensi resonansi.

2.2.4. Translator Bahasa Alami ke Bahasa Buatan

Translator bahasa alami ke bahasa buatan, yaitu translator yang mengubah perintah-perintah dalam bahasa alami menjadi bahasa buatan yang dapat dieksekusi oleh mesin atau komputer. Sebagai contoh, translator yang memungkinkan kita memberikan perintah bahasa alami kepada komputer. Dengan sistem seperti ini, pengguna sistem dapat memberikan perintah dengan bahasa sehari-hari, misalnya, untuk menghapus semua file, pengguna cukup memberikan perintah" komputer, tolong hapus semua file !" Translator akan mentranslasikan perintah bahasa alami tersebut menjadi perintah bahasa formal yang dipahami oleh komputer, yaitu " dir *.* < ENTER>".

2.2.5. Sistem *Speech Recognition*

Secara garis besar, cara kerja sistem pengenalan suara ini ialah mula-mula sinyal suara manusia yang diterima dengan menggunakan *microphone* (sinyal *analog*) dicuplik sehingga menjadi sinyal *digital* dengan bantuan sound card pada PC.

2.3. Sinyal suara Ucapan

Sinyal suara ucapan manusia dapat dipandang sebagai sinyal yang berubah lambat terhadap waktu (slowly time varying signal), jika diamati pada selang waktu yang singkat yaitu 5-100 ms. Pada selang waktu tersebut, katakteristik sinyal suara ucapan dapat dianggap stasioner. Untuk selang waktu yang lebih

panjang (dengan orde 0.2 detik atau lebih), karakteristik sinyal berubah untuk merefleksikan suara berbeda yang diucapkan.

2.3.1. Klasifikasi Berdasarkan Sinyal Eksitasi

Berdasarkan sinyal eksitasi yang dihasilkan pada proses produksi suara, sinyal suara ucapan dapat dibagi menjadi tiga bagian yaitu silence, unvoiced, dan voiced:

1. Sinyal *silence* : sinyal pada saat tidak terjadi proses produksi suara ucapan, dan sinyal yang diterima oleh pendengar dianggap sebagai bising latar belakang.
2. Sinyal *unvoiced* : terjadi pada saat pita suara tidak bergetar, dimana sinyal eksitasi berupa sinyal random.
3. Sinyal *voiced* : terjadi jika pita suara bergetar, yaitu pada saat sinyal eksitasi berupa sinyal pulsa kuasi-periodik. Selama terjadinya sinyal voiced ini, pita suara bergetar pada frekuensi fundamental – inilah yang dikenal sebagai *pitch* dari suara tersebut.

2.3.2. Analisis Sinyal Ucapan

Informasi yang terdapat di dalam sebuah sinyal ucapan dapat dianalisis dengan berbagi cara. Beberapa peneliti telah membagi beberapa level pendekatan untuk menggambarkan informasi tersebut, yaitu level akustik, fonetik, fonologi, morfologi, sintatik, dan semantik.

1. Level Akustik

Sinyal ucapan merupakan variasi tekanan udara yang dihasilkan oleh sistem artikulasi. Untuk menganalisa aspek-aspek akustik dari sebuah sinyal ucapan, dapat dilakukan dengan transformasi dari bentuk sinyal ucapan menjadi sinyal

listrik dengan menggunakan tranduser seperti *microphone*, telepon, dan sebagainya. Setelah melalui berbagai pengolahan sinyal digital, maka akan di peroleh informasi yang menunjukkan sifat-sifat akustik dari sinyal ucapan tersebut yang meliputi frekuensi fundamental (F0), intensitas, dan distribusi energi spektral.

2. Level Fonetik

Level ini menggambarkan bagaimana suatu sinyal suara diproduksi oleh organ-organ di dalam tubuh manusia.

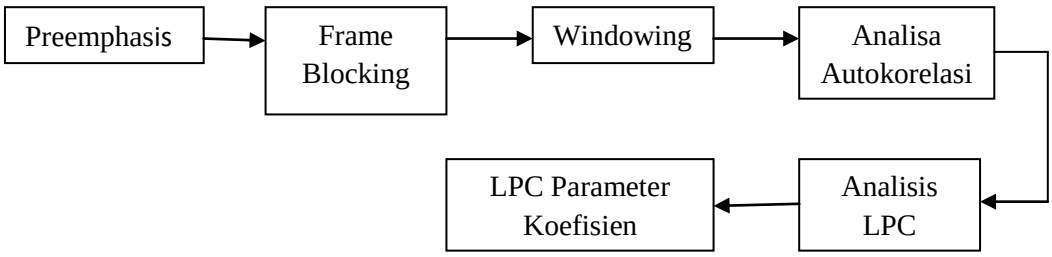
3. Level Fonologi

Di dalam level ini, dikenal istilah fonem yang merupakan unit terkecil yang membentuk sebuah kalimat atau ucapan. Deskripsi ini memuat informasi durasi, intensitas, dan *pitch* dari fonem-fonem yang membangun kalimat tersebut.

2.4. Analisis Sinyal Dengan Metode LPC

Analisis sinyal adalah kegiatan melakukan ekstraksi terhadap informasi yang terdapat di suatu sinyal. *Linear Predictive Coding* (LPC) merupakan salah satu teknik analisis sinyal percakapan yang paling *powerful* dan menyediakan ekstraksi fitur yang berkualitas baik dan efisien untuk digunakan dalam perhitungan. LPC pertama kali digunakan pada tahun 1978 untuk membuat alat sintesis sinyal percakapan. LPC melakukan analisis dengan cara memperkirakan *formant*, memisahkan *formant* dari sinyal, yang dinamakan proses *inverse filtering*, lalu mengestimasi intensitas dan frekuensi dari sinyal percakapan yang tersisa, yang disebut *residue.16* Karena sinyal percakapan bervariasi seiring

waktu, estimasi tersebut dilakukan untuk setiap potongan kecil dari sinyal, yang dinamakan *frame.* Prosedur untuk mendapatkan koefisien LPC diperlihatkan pada gambar dibawah.



Gambar 2.2 Block Diagram LPC

Gambar diatas menunujukan blok diagram LPC yang biasa digunakan dalam sistem pengenalan suara manusia. Langkah-langkah dasar yamg harus dilakukan mengenai blok diagram tersebut adalah :

1. *Preemphasis* : Proses dimana sinyal / speech ucapan sirubah menjadi sinyal
2. *Blocking into frames* : Pada tahap ini sinyal yang telah di preephasis, diblok menjadi beberapa bagian dengan jumlah sample *N*, dan tiap bagian dipisahkan dengan sejummlah M sample.
3. *FrameWindowing* : Tahap berikutnya adalah melakukan proses *window* pada setiap bagian sinyal yang telah sibuat sebelumnya. Hal ini dilakukan untuk meminimalkan pada bagian awal dan akhir sinyal. Jika definisikan sebuah *window w(n)* dan sinyal tiap bagian adalah x(n) maka sinyal hasil proses *windowing*.
4. *Auto Correlation Analisys* : Tiap bagian yang telah diberi *window* kemudian akan dibentuk autokorelasinya.

5. Analisa LPC : Langkah berikutnya adalah analisa LPC dimana semua nilai aotukorelasinya yang telah dihitung pada tahap sebelumnya akan diubah menjadi parameter LPC
6. Pengubahan parameter LPC menjadi koefisien *cepstral* : Parameter LPC yang sangat penting yang bisa diturunkan dari koefisien LPC adalah koefisien *cepstral* LPC, *c(m)*

Adapun langkah-langkah analisis LPC untuk pengenalan suara adalah sebagai berikut.

1. *Preemphasis* terhadap cuplikan sinyal dengan persamaan *preemphasizer:*

$$s(n) = s(n) - as(n-1) \qquad (1)$$

dengan *s*(*n*) adalah sampel ke-n dan harga *a* yang paling sering digunakan ialah 0.95.

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

3. 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*:*

$$W(n) = 0.54 - 0.46 \cos(2\pi n/N-1),\ 0 \leq n \leq N-1 \qquad (2)$$

4. Analisis autokorelasi terhadap setiap *frame* hasil *windowing* x_1 (*n*) dengan persamaan:

$$r_1(m) = \sum_{n=0}^{n-1-m} x_1(n)x_1(n+m) \qquad (3)$$

dengan *m* dimulai dari 0 dan nilai tertinggi dari *m* = *p* adalah orde LPC yang biasa bernilai 8 - 16.

5. Mengubah *p* +1 buah hasil autokorelasi pada masingmasing *frame* menjadi koefisien LPC $a_m = \alpha_m^{(p)}$ untuk *m* = 1,2,..., *p* dengan persamaan dibawah ini:

$$E^{(o)}=r(0) \quad (4)$$

$$k_m=\{r(m)-\Sigma^{m-1}_{j=1}\alpha_j^{(m-1)}r(|m-j|)/E^{(m-1)},\ 1 \le m \le p \quad (5)$$

$$\alpha_m^{(m)}= k_m \quad (6)$$

$$\alpha_j^{(m)} = \alpha_j^{(m-1)}- k_m\ \alpha_{m-j}^{(m-1)},\ 1 \le j \le m-1 \quad (7)$$

$$E^{(m)}=(1- k_m^2)\ E^{(m-1)} \quad (8)$$

dengan $r(0)$adalah hasil autokorelasi, $E_{(m)}$ adalah *error*, k_m adalah koefisien pantulan, $a_j^{(m)}$ adalah koefisien prediksi untuk $1 \le j \le m$.

6. Mengubah parameter LPC a_m ke koefisien cepstral c_m untuk mendapatkan kinerja yang lebih baik dan tahan terhadap *noise*, yaitu dengan persamaan:

$$c_m=a_m+\Sigma^{m-1}_{k=1}(k/m)c_k\ a_{m-k},\ 1 \le m \le p \quad (9)$$

$$c_m= \Sigma^{m-1}_{k=1}(k/m)c_k\ a_{m-k},\ m > p \quad (10)$$

koefisien *cepstral* ini adalah koefisien dari representasi pada *spectrum logaritmis*.

2.5. Metode Jaringan Syaraf Tiruan

JST atau *Artificial Neural Network (ANN)* adalah bagian dari sistem kecerdasan buatan (*Artificial Intelligence*) yang merupakan salah satu representasi buatan dari otak manusia yang selalu mencoba untuk mensimulasikan proses pembelajaran pada otak manusia tersebut. Istilah tiruan atau buatan dimaksudkan karena jaringan syaraf ini diimplementasikan dengan menggunakan program

komputer yang mampu menyelesaikan sejumlah proses perhitungan selama proses pembelajaran. JST dimaksudkan untuk membuat model sistem komputasi yang dapat menirukan cara kerja jaringan syaraf biologis.

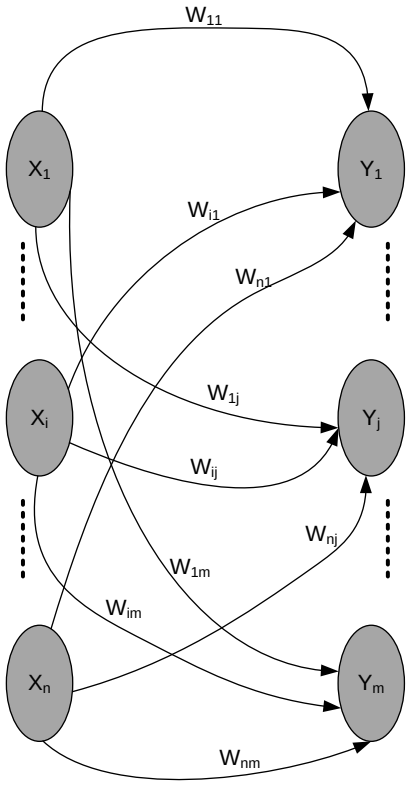
Sebuah JST terdiri dari sejumlah elemen pemroses yang dinamakan *neuron*. Masing-masing *neuron* ini dihubungkan ke *neuron* lainnya dengan suatu bilangan yang dinamakan *weight* atau bobot keterhubungan, yang berisi informasi yang digunakan jaringan untuk menyelesaikan masalah.

2.5.1. Karakteristik Jaringan Syaraf Tiruan

Sebuah jaringan saraf tiruan pada umumnya memiliki karakteristik: arsitektur, algoritma pembelajaran, dan fungsi aktivasi.23 Arsitektur adalah pola koneksi antar neuron. Algoritma pembelajaran adalah metode yang digunakan untuk menentukan bobot keterhubungan. Fungsi aktivasi adalah fungsi yang digunakan neuron untuk memetakan sinyal masukan yang diterima menjadi sinyal keluaran yang akan dikirimkan ke neuron lainnya.

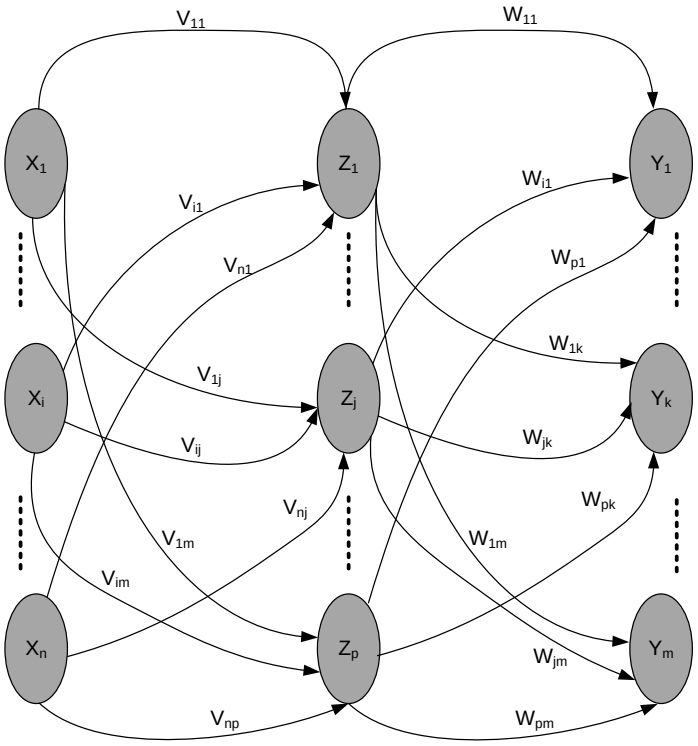
2.5.2. Klasifikasi Jaringan Syaraf Tiruan

Menurut arsitekturnya, jaringan saraf tiruan seringkali diklasifikasikan sebagai jaringan lapis tunggal dan jaringan lapis jamak.24 Jaringan lapis tunggal terdiri dari satu lapis unit masukan dan satu lapis unit keluaran. Arsitektur jaringan lapis tunggal dapat dilihat pada gambar dibawah.



Gambar 2.3. jaringan lapisan tunggal

Jaringan lapis jamak terdiri dari satu lapis unit masukan, *n* lapis unit tersembunyi, dan satu lapis unit keluaran. Arsitektur jaringan lapis jamak dengan satu lapis unit tersembunyi dapat dilihat pada gambar dibawah.



Gambar 2.4. jaringan lapisan jamak

Fungsi aktivasi digunakan untuk menghasilkan sinyal keluaran yang dibatasi pada rentang nilai tertentu, misalnya: antara 0 dengan 1 disebut biner, dan antara -1 dengan 1 disebut bipolar. Ada beberapa jenis fungsi aktivasi yang sering digunakan yaitu: linier, tangga, dan *sigmoid*.

Kehadiran unit-unit tersembunyi dan fungsi aktivasi yang non linier pada jaringan, dapat memberikan kemampuan kepada jaringan untuk menyelesaikan lebih banyak masalah daripada jaringan yang hanya memiliki unit-unit masukan dan unit-unit keluaran.

2.5.3. Pembelajaran Jaringan Syaraf Tiruan

Cara pembelajaran jaringan saraf tiruan dapat dibagi menjadi dua kategori yaitu *supervised* (terarah) dan *unsupervised* (tidak terarah).26 Pembelajaran *supervised* mengasosiasikan vektor-vektor masukan dengan target keluaran,

contohnya: *Hebb*, *Perceptron*, *Adaline*, *Learning Vector Quantization* (LVQ), *BackPropagation*, dan lain-lainnya. Sedangkan pembelajaran *unsupervised* mengelompokkan vektor-vektor masukan yang memiliki sifat mirip menjadi satu keluaran tanpa memperhatikan target keluaran.

Pada penelitian ini akan dilakukan sebuah pemodelan menggunakan Metode Hidden Markov Models (HMM) untuk melakukan pengenalan suara. Pada subbab selanjutnya akan dijelaskan lebih lanjut mengenai JST Model HMM.

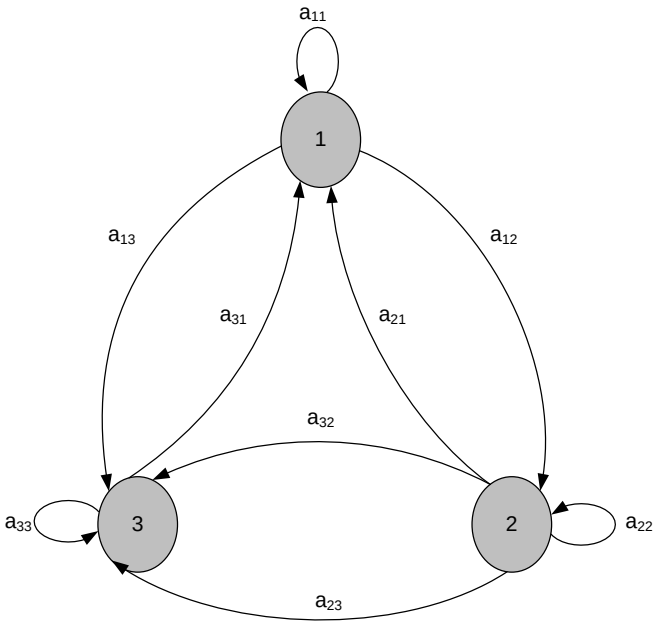
2.6. Pemodelan Dengan *Markov Models*

Misalkan sebuah system dideskripsikan pada setiap waktu sebagai salah satu dari N status {1,2,..,N}. Sistem mungkin mengalami perubahan status dalam satuan waktu distrik (mungkin kembali ke status semula) tergantung kepada aturan peobabilitas setiap status. Misalkan waktu disimbolkan sebagai t, status pada setiap waktu disimbolkan sebagai qt. Deskripsi keseluruhan probabilitas dari system secara umum membutuhkan spesifikasi status saat ini dan status-status sebelumnya.

Untuk kasus khusus dengan spesifikasi waktu diskrit, first orde, dan Markov chain, dapat rumus ketergantungan probabilitas suatu status terhadap status-status sebelumnya disederhanankan hanya tergantung pada tepat satu status sebelumnya.

2.6.1. Rantai Markov

Algoritma HMM (Hidden Markov Model) didasari oleh model matematik yang dikenal dengan rantai markov. Rantai markov secara umum ditunjukkan pada Gambar 2.3



Gambar 2.5 Rantai Markov

Beberapa hal yang dapat dijelaskan rantai markov yaitu :

1. Transisi keadaan dari suatu keadaan tergantung pada keadaan sebelumnya.

$P [q_t = j | q_{t-1} = i, q_{t-2} = k\ldots\ldots] = P [q_t = j | q_{t-1} = I]$

Penyederhanaan lebih lanjut dengan hanya memperhitungkan proses dibagian kanan, bebas terhadap waktu, maka pendekatan aturan probabilitas transisi status sepeti dibawah

2. Transisi keadaan bebas terhadap waktu.

$a_{ij} = P [q_t = j | q_{t-1} = i]$

dengan sifat

aij≥0, unutk setiap j, i

aij=1, untuk setiap I, i=1 sampai N

karena mematuhi batasan *stochastic* standar.

Contoh permasalahan :

Cuaca dalam satu hari dimodelkan ke dalam tiga status : rainy (1), cloudy (2) dan sunny (3). Aturan probabilitas dari setiap transisi status dideskripsikan sebagai berikut

A ={a Cuaca dalam satu hari dimodelkan ke dalam tiga status : rainy (1), cloudy (2) dan sunny (3). Aturan probabilitas dari setiap transisi status dideskripsikan sebagai berikut

$A = \{a_{ij}\} = 0.4\ 0.3\ 0.3$

$0.2\ 0.6\ 0.2$

$0.1\ 0.1\ 0.8$

Probabilitas cuaca untuk 8 hari berturut-turut “sun-sun-sun-rain-rain-sun-cludysun” adalah :

Misalkan O adalah tahap pengamatan O={sunny, sunny, sunny, rainy, rainy, sunny, cloudy, sunny}={3,3,3,1,1,3,2,3}

$P(O|Model) = P[3,3,3,1,1,3,2,3|Model]$

$= P[3]P[3|3]^2P[1|3]P[1|1]$

$= II_3.(a_{33})^2\ a_{31}\ a_{11}\ a_{13}\ a_{32}\ a_{23}$

$= (1.0)(0.8)^2(0.1)(0.4)(0.3)(0.1)(0.2)$

$= 1.536 \times 10^{-4}$

Dengan,

$$II_{3=}p[q_t=I],\ 1< i < N$$

2.6.2. *Hidden Markov Models*

Konsep markov model diperluas menjadi Hidden Markov Model dimana observasi merupakan fungsi probabilitas dari status. Model ini dapat diaplikasikan pada kasus yang proses tidak dapat diobservasi secara langsung (tersembunyi) tetapi bias diobservasi hanya melalui kumpulan proses stokastik yang lain menghasilkan tahapan observasi, misalnya persoalan cointoss dan speech recognition (pengenalan suara).

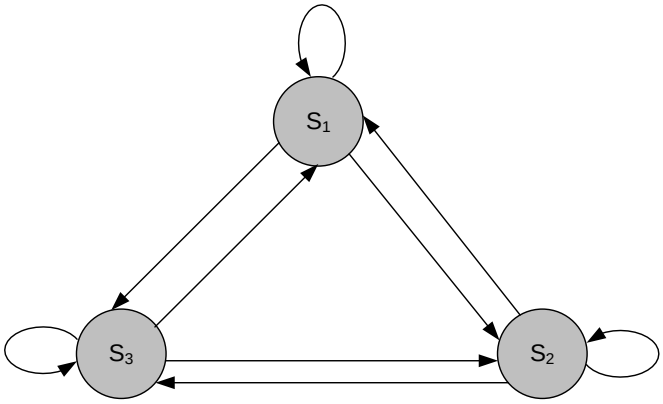
HMM merupakan perluasan dari teori yang lebih sederhana yaitu markov chain. Markov chain merupakan teori yang menjabarkan kemungkinanan terjadinya suatu kejadian berdasarkan rangkaian kejadian sebelumnyadari data-data probabilitas kemunculan setiap kejadian yang mungkin serta probabilitas kemunculan dua kejadian berurutan.

2.6.3. Tipe HMM

HMM dibagi menjadi dua tipe dasar yaitu HMM ergodic dan HMM Kiri Kanan.

1. HMM *ergodic*.

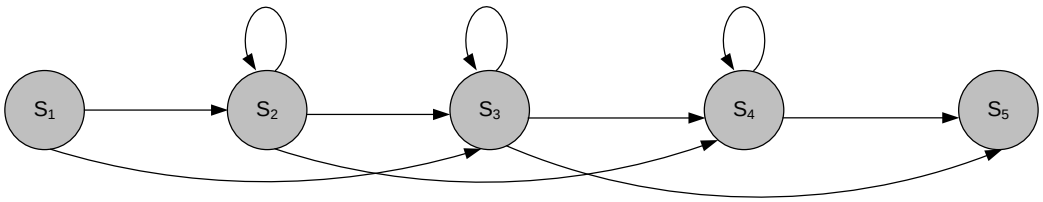
Pada HMM model *ergodic* perpindahan keaadaan satu ke keadaan yang lain semuanya memungkinkan, hal ini ditunjukkan pada



Gambar 2.6 HMM model ergoric

2. HMM kiri-kanan

Pada HMM kiri-kanan perpindahan keadaan hanya dapat berpindah dari kiri kekanan, perpindahan keadaan tidak dapat mundur ke belakang, hal ini ditunjukkan pada



Gambar 2.7 HMM model kiri-kanan

2.6.4. Elemen HMM

Elemen yang terdapat pada HMM yaitu :

1. N, jumlah keadaan *(state)* dalam model
2. M, jumlah sumber observasi yang berbeda tiap keadaan
3. Distribusi keadaan transisi A = {a_{ij}}dengan

   $a_{ij} = P [q_{t+1} = j \mid q_t = j], 1 \leq i, j \leq N$

4. Distribusi probabilitas simbol observasi, B = { bj (k) }dengan

$b_j\,(k) = P\,(\,o_t = v_k \mid q_t = j\,)$, untuk $1 \leq j \leq N$, i $\leq k \leq M$

5. Distribusi keadaan awal $\pi = \{\,\pi_i\,\}$

$\pi_i = P\,[\,q_t = i\,]$ $\qquad 1 \leq i \leq N$

2.6.5. Prinsip Kasus dari HMM

Ada 3 kasus utama untuk menangani, merumuskan suatau HMM yang sukses ini adalah :

Kasus 1 Evaluasi (B)

Dengan :

1. a suatu model $\lambda = (A, B, \pi)$ siap untuk digunakan

2. Urutan pengamatan $O = O_1, O_2, O3,\ldots..,O_{t-1},O_T$

3. Hitung $P(O/\lambda)$ kemungkinan urutan untuk pengamatan model

Kasus 2 pengkodean (B)

Dengan :

1. a suatu model $\lambda = (A, B, \pi)$ siap untuk digunakan

2. Urutan pengamatan $O = O_1, O_2, O3,\ldots..,O_{t-1},O_T$

Aksi : mencari urutan status $Q = q_1, q_2, q_3,\ldots.., q_{t-1}, q_T$ yang pasti hamper dapat dipastikan menghasiilkan pengmatan-pengamatan yang menggunakan model yang diberi

Kasus 3 Pelatihan (B)

Dengan :

1. a suatu model $\lambda = (A, B, \pi)$ siap untuk digunakan

2. Pengamatan latihan $O_1^k, O_2^k, O_3^k,\ldots\ldots O_{T-1}^k, O_T^k$ k dimana dikali 1000 adalah banyaknya contoh-contoh untuk pelatihan model

Aksi : set parameter-parameter model untuk memaksimalkan P(O/λ)

Kasus 1 adalah suatu prosedur evaluasi sebagai kita sedang mencari-cari untuk menemukan kemungkinan menghasilkan pengamatan yang diberi O oleh suatu model yang diberi 1. Ini bisa digunakan unutk menenmukan model terbaik di antara banyak yang menghasilkan pengamatan yang diberi.

Kasus 2 adalah suatu prosedur pengawassandian untuk mendeteksi atau tidak bersembunyi, menyembunyikan urutan status (Negara dari suatu yang diberi pengamatan. Pengamatan-pengamatan itu bisa contoh-contoh pelatihan jika kita ingin belajar perilaku dari tiap status Negara dari aspek yang berbeda, seperti jangka waktu atau spectral karakteristik-karakteristik dari tiap status(Negara. Bebrapa teknik menggunakan jangka waktu status(Negara didalam evaluassi meraka prosedur dan dalam hal ini pengamatan itu akan contoh ujian untuk mendeteksi Negara itu jangka waktu.

Kasus 3 adalah prosedur latihana itu kepada pengoptimalan parameter-parameter model unutk memperoleh yang terbaik model bahwa mewakili menunujukan set tertentu pengamatan-pengamatan skepunyaan satu kesatuan.

2.6.6. Pengenalan Suara Dengan HMM

Sistem pengenalan suara moern secara umum berdasarkan kepada *Hidden Markov Mode*l (HMM). HMM merupakan model statistic dimana mempunyai keluaran rangkaian simbol dan kuantitas. Dengan memiliki sebuah model yang memberikan kemungkinan dari rangkaian akustik data yang telah diobservasi dari sebuah atau banyak kata (rangkaian kata) akan dapat menyebabkan sistem bekerja dengan rangkaian kata tersebut sesuai dengan aplikasi aturan .

$$Pr(word/acoustic) = \frac{pr(acoustic|word)pr(word)}{pr(acoustic)}$$

Dari rangakaian data akustik yang ada, pr(acoustic) adalah konstan data tidak dapat diabaikan. Pr(word) adalah merupakan kemungkinan terbesar dari suatu kata . pr(acoustic|word) masa yang paling terlibat didalam persamaan dan diperoleh dari HMM.

Untuk dapat digunakan sebagai aplikasi pada dunia nyata, tiga masalah yang mendasar harus dapat diselesaikan.

Diberikan :

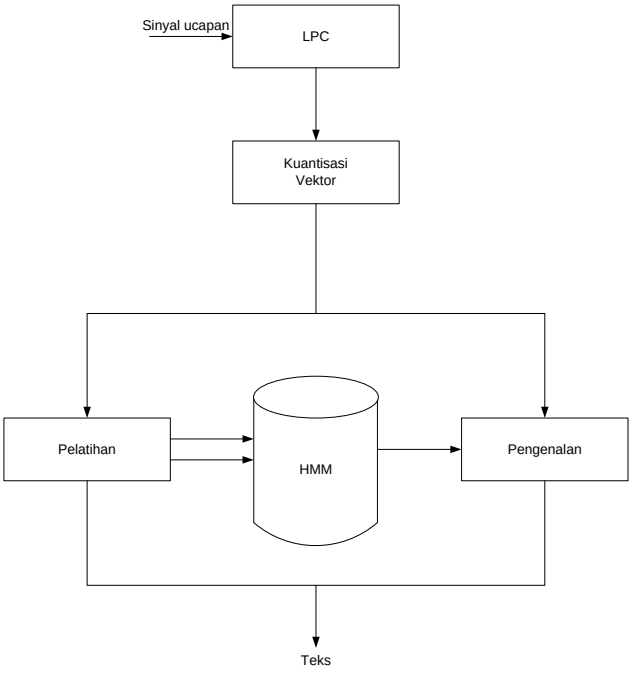
Rangkaian Pengamatan

$O = (O_1 O_2 O_3 \ldots\ldots\ldots\ldots O_T)$

Model $\lambda = (A, B, \pi)$

Jadi O itu adalah sebuah tahapan pengamatan, yang terdiri dari pengamatan pertama sampai pengamatan ke-T, dan Model $\lambda = (A, B, \pi)$ adalah sebuah suatu prosedur evaluasi untuk mencari-cari dan menemukan kemungkinan menghasilkan pengamatan yang diberi O oleh suatu model yang diberi. Ini bisa digunakan untuk menemukan model terbaik di antara banyak yang menghasilkan pengamatan yang diberi.

2.6.7. Arsitektur Metode HMM



Gambar 2.8 Arsitektur Metode HMM

Penjelasan Gambar diatas akan diperjelas sebagai berikut :

1. Linear Predective *(LPC)* berfungsi untuk mengekstraksi sinyal data sinyal yang diinputkan

2. Kuantisasi vector berfungsi unutk proses pengkodean ciri-ciri vector sinyal kedalam bentuk symbol HMM. Pengkodean suara digital dpat diklsifikasikan kedalam 2 buah kategori, yaitu bentuk gelombang coders dan vocoders. Gelombang coders menggunakan algoritma untuk mengkodekan sinyal suara sehingga menghasilkan output system dalam bentuk yagn dapat diperkirakan. Vocoders mengkodekan sinyal suara dengan mengumpulkan parameter-parameter yang digitalkan dan ditransmisikan kedalam receiver.

3. Pelatihan berfungsi untuk pengestimasian ulang parameter HMM dari suatu data berdasarkan data latih yang diajukan pengguna, sehingga hasil estimasi memilki kualitas yang baik.

4. Pengenlan berfungsi melakukan perhitungan probabilitas kemiripan pola dari tiap-tiap model HMM yang mempunyai probabilitas kemiripan yang tertinggi.

2.6.8. *Random Field*

Diberikan $S=\{1,2,\ldots,m\}$ yang menunjukan pixel atau site. Asumsikan Ls himpunan terbatas dari distrik label ubtuk setiap $s \in S$ diberikan maka ruang konfigurasi (configuration spasce) x, dapat didefinisikan untuk setiap $x \in X$ sebagai

$X = \Pi_{Lt}$ *dimana* $x \epsilon X$

Dimana x disebut konfigurasi.

Definisi 1. Sebuah ukuran probablitas (probability measure) atau distribusi pada x sisefinisikan sebagai sebuah vector $v = (v(x))_{x|c}$ serta memilki sua sifat, yaitu :

(i) $v(x) \geq 0 \; V \; x \epsilon X$

(ii) $\Sigma_{x \epsilon X} \, v(x) = 1$

sebuah random field didefiniskan sebagai sebuah distribusi Π pada □ dengan sifat positif mutlak (strictly posisitive), yaitu

$\Pi (x) > 0, \; V \;\; x \epsilon X$

2.6.9. *Markov Random Field*

Definisi 2. Misalkan N menunujukan koleksi himpunan bagian dari S dengan relasi

Sebagai berikut

$N:=\{ N (s) \; C \; S | s \epsilon S\}$

N disebut daerah sekitar (neighbourhood) dari system S jika memenuhi

(i) $s \in N(s)$, $V s \in S$

(ii) Vs, $\epsilon_t s$ dengan $s \epsilon t$ maka $st \approx N(t)$ $t \epsilon N(s)$

Site t ϵ N(s) disebut tetangga *(neigbour)* dari s_t dinotasikan sebagai <s,t> jika s dan t saling bertetanggaan.

Pada kasus regular site grid, himpunan S dapat dianggap sebagai himpunan bagian dari ZxZ, yaitu S={(I,j) ϵ ZxX|-m£$_i$, £m} untuk lattice bujur sangkar dengan ukuran (2m+1)x(2m+1), sedangkan daerah sekitar dari sebuah site (i,j) pada ssebuah grid merupakan site0site disekitar site yang dianggap sebagai titik tengah dengan radius z, yaitu

$$N(i,j):=\{(k,l) \epsilon s|0 < (k-i)^2+(l-j)^2 \approx c^2\}$$

Namun demikian, secara umum radius lingkaran c biasanya tidak digunakan unruk menunjukan order dari daerah sekitar, melainkan digunakan order yang meningkatkan sejalan dengan nilai.

2.6.10. Sifat *Markov (Markovianity)* Pada *Random Fields*

Markov Random Field merupakan generalisasi dari proses Markov yang terdapat pada teori proses stokastik. Perbedaannya terletak pada domain ketergantungan dari random variable yang akan didefinisikan, dimana proses Markov hanya memperlihatkan ketergantungan terhadap waktu pada suatu dimensi saja, sedangkan Markov Random Field menunjukan ketergantungan secara spasial. Defiisi 4. Diberikan sebuah daerah sekitar sembarang N. Random Field P pada X disebut Markov Random Field (MRF) terhadap system daerah

sekitar N jika untuk seluruh site sIS karakteristik lokalnya memenuhi persamaan dibawah ini :

$$P\ (X_s x_s \mid X_r = x_r,\ r \neq s) = P(X_s = x_s \mid X_r = x_r,\ r \in N(s))$$

Yaitu, karakteristik local dari sebuah site tunggal pada nilai-nilai (label) dari daerah sekitarnya saja. Besag membuktikan pada (Besag, 74) bahwa sebarang random field P ditentukan secara unik oleh karakteristik local dari site tunggalnya, persamaan (1). Teori ini indah tapi secara praktis sangat sulit untuk menggunakan random field secara umum, karena metode untuk mendapatkan distribusi gabungan dari nilai karakteristik lokalnya tidak diketahui. Lebih lanjut, bila diberikan sebuah fungsi, sangatlah sulit untuk menunujukan apakah fungsi ini merupakan karakteristik local dari sebuah random field.

2.7. Pemodelan Sistem

Dalam suatu proses pengembangan software, analisa dan rancangan telah merupakan terminologi yang sangat tua. Pada saat masalah ditelusuri dan spesifikasi dinegosiasikan, dapat dikatakan bahwa kita berada pada tahap rancangan. merancang adalah menemukan suatu cara untuk menyelesaikan masalah, salah satu tool/model untuk merancang pengembangan software yang berbasis object-oriented adalah UML. Alasan mengapa UML digunakan adalah, pertama, scalability dimana objek lebih mudah dipakai untuk menggambarkan sistem yang besar dan komplek. Kedua, dynamic modeling, dapat dipakai untuk pemodelan sistem dinamis dan real time. Sebagaimana dalam tulisan pertama, penulis menjelaskan konsep mengenai obyek, OOA&D (Obyek Oriented Analyst/ Design) dan pengenalan UML, maka dalam tulisan kedua ini lebih ditekankan

pada cara bagaimana UML digunakan dalam merancang sebuah pengembangan software yang disertai gambar atau contoh dari sebuah aplikasi.

2.7.1. Basis Data

Basis data (*database*), atau sering pula dieja basisdata, adalah kumpulan informasi yang disimpan di dalam komputer secara sistematik sehingga dapat diperiksa menggunakan suatu program komputer untuk memperoleh informasi dari basis data tersebut. Perangkat lunak yang digunakan untuk mengelola dan memanggil kueri (*query*) basis data disebut sistem manajemen basis data (*database management system*, DBMS). Sistem basis data dipelajari dalam ilmu informasi.

Istilah "basis data" berawal dari ilmu komputer. Meskipun kemudian artinya semakin luas, memasukkan hal-hal di luar bidang elektronika, artikel ini mengenai basis data komputer. Catatan yang mirip dengan basis data sebenarnya sudah ada sebelum revolusi industri yaitu dalam bentuk buku besar, kuitansi dan kumpulan data yang berhubungan dengan bisnis.

Konsep dasar dari basis data adalah kumpulan dari catatan-catatan, atau potongan dari pengetahuan. Sebuah basis data memiliki penjelasan terstruktur dari jenis fakta yang tersimpan di dalamnya: penjelasan ini disebut skema. Skema menggambarkan obyek yang diwakili suatu basis data, dan hubungan di antara obyek tersebut. Ada banyak cara untuk mengorganisasi skema, atau memodelkan struktur basis data: ini dikenal sebagai model basis data atau model data. Model yang umum digunakan sekarang adalah model relasional, yang menurut istilah layman mewakili semua informasi dalam bentuk tabel-tabel yang saling

berhubungan dimana setiap tabel terdiri dari baris dan kolom (definisi yang sebenarnya menggunakan terminologi matematika). Dalam model ini, hubungan antar tabel diwakili denga menggunakan nilai yang sama antar tabel. Model yang lain seperti model hierarkis dan model jaringan menggunakan cara yang lebih eksplisit untuk mewakili hubungan antar tabel.

Istilah *basis data* mengacu pada koleksi dari data-data yang saling berhubungan, dan perangkat lunaknya seharusnya mengacu sebagai *sistem manajemen basis data* (*database management system/DBMS*). Jika konteksnya sudah jelas, banyak administrator dan programer menggunakan istilah basis data untuk kedua arti tersebut.

Perangkat lunak basis data yang banyak digunakan dalam pemrograman dan merupakan perangkat basis data aras tinggi (*high level*):

1. DB2
2. Microsoft SQL Server
3. Oracle
4. Sybase
5. Interbase
6. XBase
7. Firebird
8. MySQL
9. PostgreSQL
10. Microsoft Access
11. dBase III
14. FoxPro
15. Visual FoxPro
16. Arago
17. Force
18. Recital
19. dbFast
20. dbXL
21. Quicksilver
22. Clipper
23. FlagShip
24. Harbour

12. Paradox
13. Lotus Smart Suite Approach
25. Visual dBase

Selain perangkat lunak di atas, terdapat juga perangkat lunak pemrograman basis data aras rendah (*low level*), diantaranya:

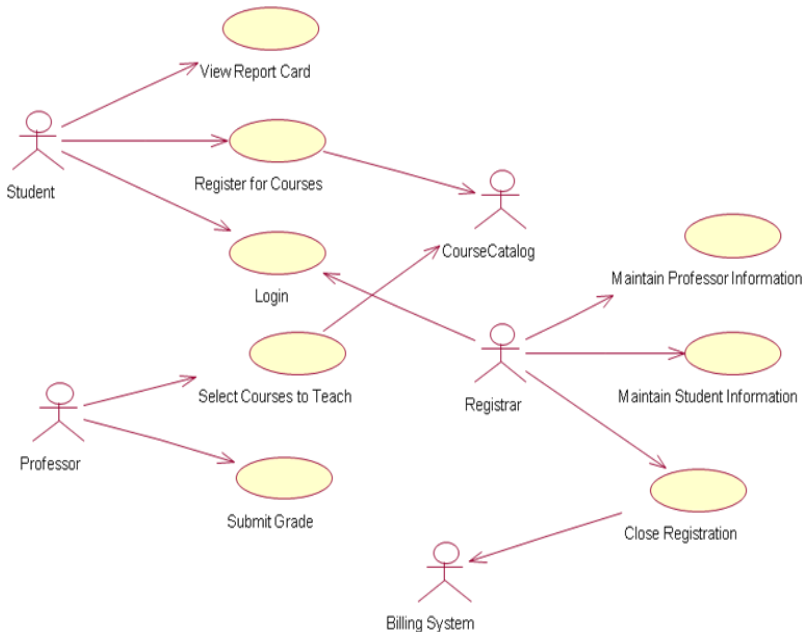
1. Btrieve
2. Tsunami Record Manager

DBMS (Database Management System) adalah software yang menangani semua akses ke basis data. Secara konsep apa yang terjadi adalah sebagai berikut :

1. user melakukan pengaksesan basis data untuk informasi yang diperlukannya menggunakan suatu bahasa manipulasi data, biasanya disebut SQL.
2. DBMS menerima *request* dari user & menganalisa request tersebut
3. DBMS memeriksa skema eksternal user, pemetaan eksternal/konseptual, skema konseptual, pemetaan konseptual/internal, & struktur penyimpanan.
4. DBMS mengeksekusi operasi-operasi yang diperlukan untuk memenuhi permintaan user.

2.7.2. *Use Case Diagram*

Sebuah *use case* menggambarkan suatu urutan interaksi antara satu atau lebih aktor dan sistem. Dalam fase *requirements*, model *use case* mengambarkan sistem sebagai sebuah kotak hitam dan interaksi antara aktor dan sistem dalam suatu bentuk naratif, yang terdiri dari input user dan respon-respon sistem. Setiap use case menggambarkan perilaku sejumlah aspek sistem, tanpa mengurangi struktur internalnya. Selama pembuatan model use case secara pararel juga harus ditetapkan obyek-obyek yang terlibat dalam setiap use case.



Gambar 2.9. 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.

2.7.3. Aktor

Sebuah aktor mencirikan suatu bagian *outside user* atau susunan yang berkaitan dengan user yang berinteraksi dengan sistem [Rumbaugh, Booch, da Jacobson 1999]. Dalam model use case, aktor merupakan satu-satunya kesatuan eksternal yang berinteraksi dengan sistem. Terdapat beberapa variasi bagaimana aktor dibentuk [Fowler dan Scott 1999]. Sebuah aktor sering kali merupakan manusia (*human user*). Pada sejumlah sistem informasi, manusia adalah satu-satunya aktor. Dan mungkin saja dalam sistem informasi, seorang aktor bisa saja

menjadi suatu sistem eksternal. Pada aplikasi real-time dan distribusi, sebuah aktor bisa saja menjadi satu perangkat eksternal I/O atau sebuah alat pengatur waktu. Perangkat eksternal I/O dan pengatur waktu aktor secara khusus lazimnya berada dalam real-time yang tersimpan dalam sistem (*real-time embedded systems*), sistem berinteraksi dengan lingkungan eksternal melalui sensor dan aktuator.

Primary actor (aktor utama) memprakarsai sebuah use case. Jadi, suatu primary aktor memegang peran sebagai proaktif dan yang memulai aksi dalam sistem. Aktor lainnya yang berperan sebagai *secondary* aktor bisa saja terlibat dalam use case dengan menerima *output* dan memberikan *input*. Setidaknya satu actor harus mendapatkan nilai dari use case. Biasanya adalah *primary* aktor (actor utama). Bagaimanapun, dalam *real-time embedded systems*, *primary* aktor dapat berperan sebagai perangkat eksternal I/O atau pengatur waktu, penerima utama dari *use case* bisa menjadi *secondary human* aktor yang menerima sejumlah informasi dari sistem.



Gambar 2.10. aktifitas actor dan use case

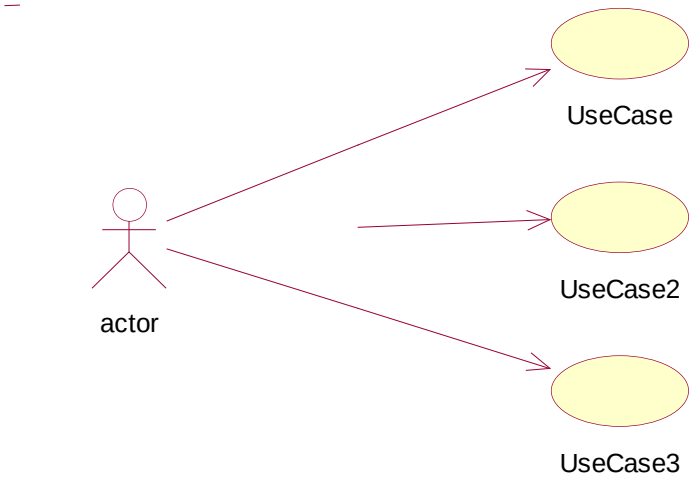
2.7.4. Identifikasi *Use Case*

Sebuah use case dimulai dengan masukan/input dari seorang aktor. Use case merupakan suatu urutan lengkap kejadian-kejadian yang diajukan oleh seorang aktor, dan spesifikasi interaksi antara aktor dengan sistem. Use case yang

sederhana hanya melibatkan satu interaksi/hubungan dengan sebuah aktor, dan use case yang lebih kompleks melibatkan beberapa interaksi dengan aktor. Use cases yang lebih kompleks juga melibatkan lebih dari satu aktor.

Untuk menjabarkan use case dalam sistem, sangat baik bila dimulai dengan memperhatikan aktor dan actions/aksi yang mereka lakukan dalam sistem. Setiap use case menggambarkan suatu urutan interaksi antara aktor dengan sistem. Sebuah use case harus memberikan sejumlah nilai pada satu aktor.

Kemudian, kebutuhan fungsional sistem dijelaskan dalam use case yang merupakan suatu spesifikasi eksternal dari sebuah sistem. Bagaimanapun juga, ketika membuat use case, sangatlah penting menghindari suatu dekomposisi fungsional yang dalam beberapa use case kecil lebih menjelaskan fungsi-fungsi 9/10 individual sistem daripada menjelaskan urutan kejadian yang memberikan hasil yang berguna bagi aktor.



Gambar 2.11. interaksi actor dan use case

Urutan utama use case menjelaskan urutan interaksi yang paling umum antara aktor dan sistem. Dan mungkin saja terdapat cabang-cabang urutan use case utama, yang mengarah pada berkurangnya frekuensi interaksi antara actor dengan sistem. Deviasi-deviasi dari urutan utama hanya dilaksanakan pada beberapa situasi, contohnya jika aktor melakukan kesalahan input pada sistem. Ketergantungan pada aplikasi kebutuhan, alternatif ini memecahkan use case dan kadang-kadang bersatu kembali dengan urutan utama. Cabang-cabang alternative digambarkan juga dalam use case.

2.7.5. Pendokumentasian Model *Use Case*

Use case didokumentasi dalam *use case model* sebagai berikut:

1. *Use Case Name.*

Setiap use case diberi nama.

2. *Summary.*

Deskripsi singkat use case, biasanya satu atau dua kalimat.

3. *Dependency.*

Bagian ini menggambarkan apakah use case yang satu tergantung pada use case yang lain, dalam arti apakah use case tersebut termasuk pada use case yang lain atau malah memperluas use case lain.

4. *Actors*

Bagian ini memberikan nama pada actor dalam use case. Selalu terdapat use case utama (primary use case) yang memulai use case. Disamping itu terdapat juga secondary use case yang terlibat dalam use case. Contohnya, dalam use case Withdraw Funds, ATM Customer adalah actor-nya.

5. *Preconditions*

Satu atau lebih kondisi harus berjalan dengan baik pada permulaan use case; contohnya mesin ATM yang tidak jalan, menampilkan pesan Selamat Datang.

6. *Deskripsi*

Bagian terbesar dari use case merupakan deskripsi naratif dari urutan utama use case yang merupakan urutan yang paling umum dari interaksi antara aktor dan sistem. Deskripsi tersebut dalam bentuk input dari aktor, diikuti oleh respon pada sistem. Sistem ditandai dengan sebuah kotak hitam (black box) yang berkaitan dengan apa yang sistem lakukan dalam merespon input aktor, bukan bagaimana internal melakukannya.

7. *Alternatif-alternatif*

Deskripsi naratif dari alternatif merupakan cabang dari urutan utama. Terdapat beberapa cabang alternatif dari urutan utama. Contohnya, jika rekening customer terdapat dana yang tidak sesuai, akan tampil permohonan maaf dan menolak kartu.

8. *Postcondition*

Kondisi yang selalu terjadi di akhir use case, jika urutan utama telah dilakukan; contohnya dana customer telah ditarik.

9. *Outstanding questions*

Pertanyaan-pertanyaan tentang use case didokumentasikan untuk didiskusikan dengan para user.

2.7.6. *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).

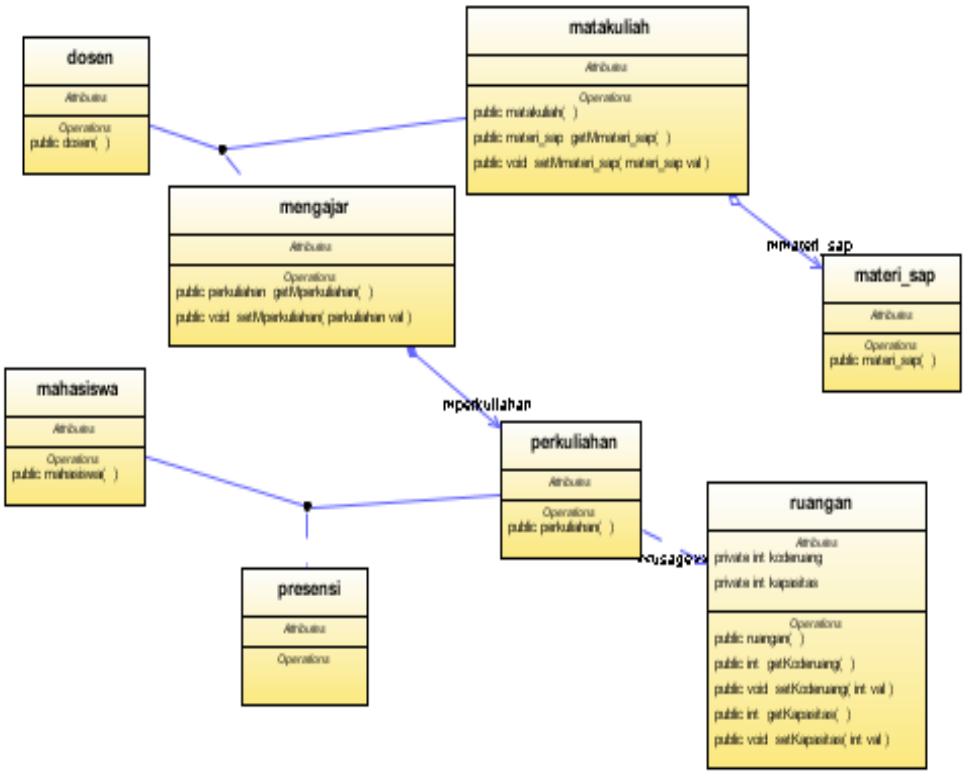
Class diagram menggambarkan struktur dan deskripsi *class, package* dan objek beserta hubungan satu sama lain seperti *containment,* pewarisan, asosiasi, dan lain-lain.

Class memiliki tiga area pokok :

1. Nama(danstereotype)
2. Atribut
3. Metoda

Atribut dan metoda dapat memiliki salah satu sifat berikut :

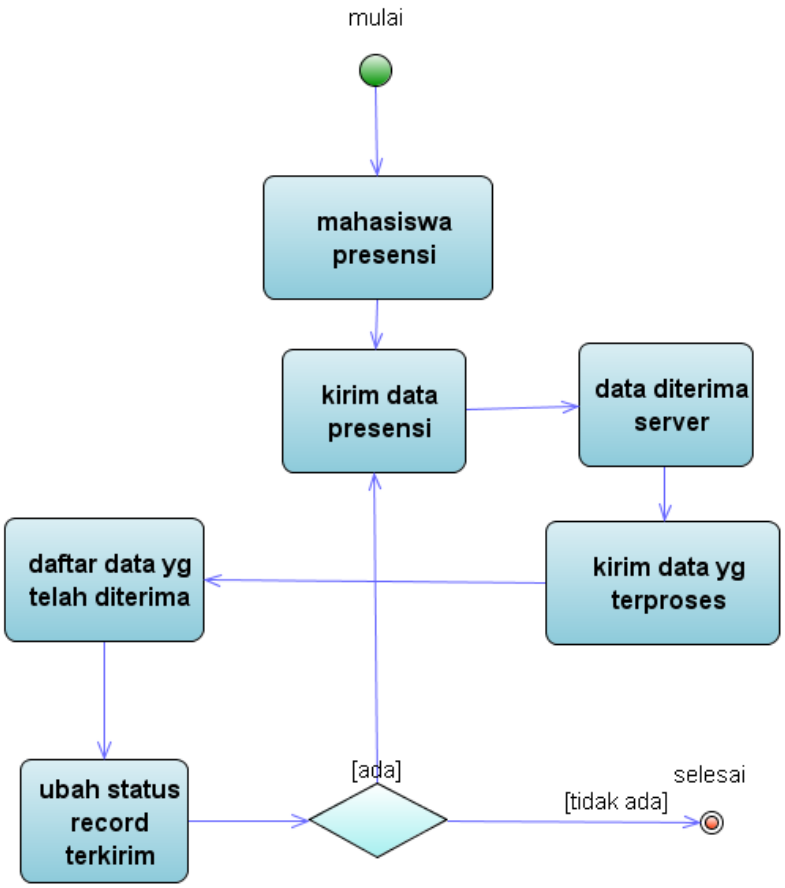
1. Private, tidak dapat dipanggil dari luar class yang bersangkutan
2. Protected, hanya dapat dipanggil oleh class yang bersangkutan dan anak-anak yang mewarisinya
3. Public, dapat dipanggil oleh siapa saja



Gambar 2.12. Class Diagram

2.7.7. *StateChart 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*).



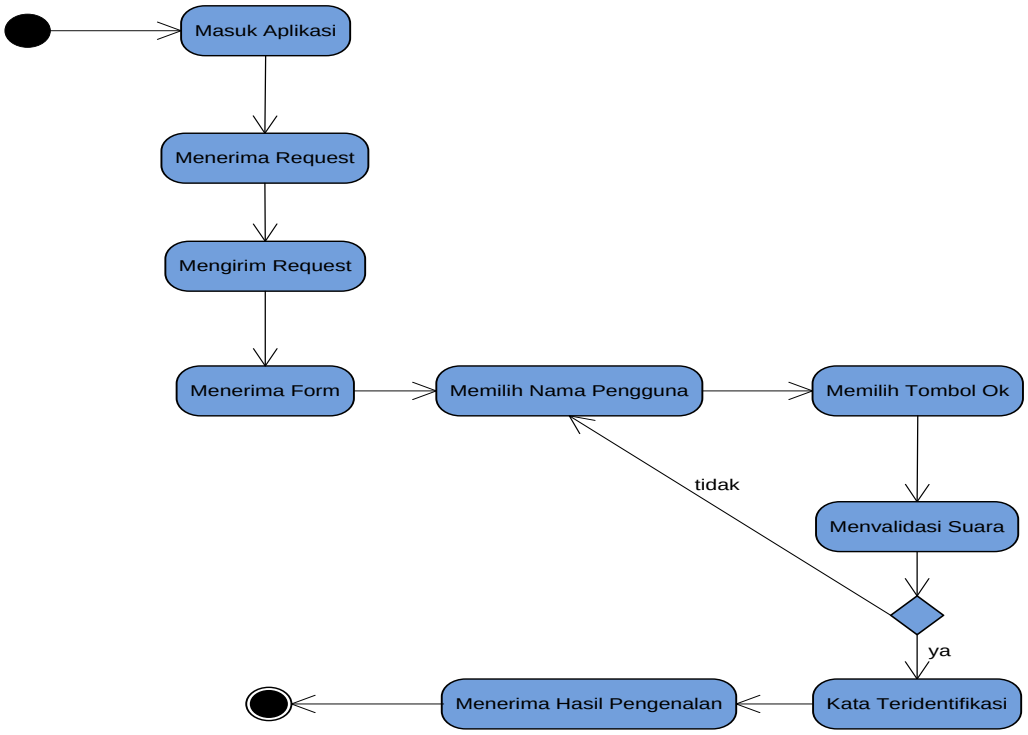
Gambar 2.13. StateChart Diagram

2.7.8. *Activivty Diagram*

Activity diagrams 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.

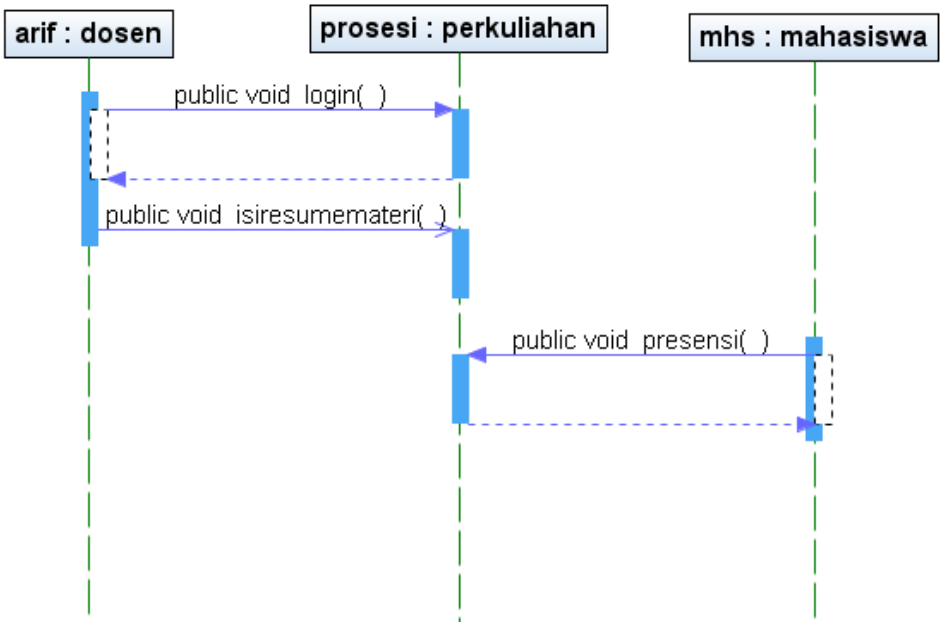


Gambar 2.14. Activity Diagram

2.7.9. *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.



Gambar 2.15. Sequence Diagram

2.8. Metode Pengembangan Perangkat Lunak

Merupakan sebuah model *chaos* yang menggambarkan "perkembangan P/L sebagai sebuah kesatuan dari pemakai ke pengembang dan ke teknologi" disebut dengan "Prescriptive" karena Menentukan sekumpulan elemen proses (Aktivitas, Aksi, tugas, produk kerja, jaminan kualitas, dll untuk setiap proyek), setiap model proses juga menentukan alur kerjanya.

Pada saat kerja bergerak maju menuju sebuah sistem yang lengkap, keadaan yang digambarkan secara rekursif diaplikasikan kepada kebutuhan pemakai dan spesifikasi teknis P/L pengembang Saat ini, prescriptive memberikan jawaban secara definitive untuk masalah pengembangan P/L dalam setiap perubahan lingkungan komputasi

2.8.1. Model Skuensial Linear

Pada Tugas akhir ini menggunakan metode penggunaan perangkat lunak siklus kehidupan klasik atau model air terjun. Model ini mengusulkan sebuah pendekatan kepada perkembangan software yang sistematik dan sekuensial yang mulai pada tingkat dan kemajuan sistem pada seluruh analisis, desain, kode, pengujian, dan pemeliharaan. Dimodelkan setelah siklus rekayasa konvensional, model sekuensial linier melingkupi aktivitas – aktivitas sebagai berikut :

1. Rekayasa dan pemodelan sistem/informasi

Karena sistem merupakan bagian dari sebuah sistem yang lebih besar, kerja dimulai dengan membangun syarat dari semua elemen sistem dan mengalokasikan beberapa subset dari kebutuhan ke software tersebut. Pandangan sistem ini penting ketika software harus berhubungan dengan elemen-elemen yang lain seperti software, manusia, dan database. Rekayasa dan anasisis system menyangkut pengumpulan kebutuhan pada tingkat sistem dengan sejumlah kecil analisis serta disain tingkat puncak. Rekayasa informasi mancakup juga pengumpulan kebutuhan pada tingkat bisnis strategis dan tingkat area bisnis.

2. Analisis kebutuhan Software

Proses pengumpulan kebutuhan diintensifkan dan difokuskan, khusunya pada software. Untuk memahami sifat program yang dibangun, analis harus memahami domain informasi, tingkah laku, unjuk kerja, dan interface yang diperlukan. Kebutuhan baik untuk sistem maupun software didokumentasikan dan dilihat lagi dengan pelanggan.

3. Desain

Desain software sebenarnya adalah proses multi langkah yang berfokus pada empat atribut sebuah program yang berbeda; struktur data, arsitektur software, representasi interface, dan detail (algoritma) prosedural. Proses desain menterjemahkan syarat/kebutuhan ke dalam sebuah representasi software yang dapat diperkirakan demi kualitas sebelum dimulai pemunculan kode. Sebagaimana persyaratan, desain didokumentasikan dan menjadi bagian dari konfigurasi software.

4. Generasi Kode

Desain harus diterjemahkan kedalam bentuk mesin yang bias dibaca. Langkah pembuatan kode melakukan tugas ini. Jika desain dilakukan dengan cara yang lengkap, pembuatan kode dapat diselesaikan secara mekanis.

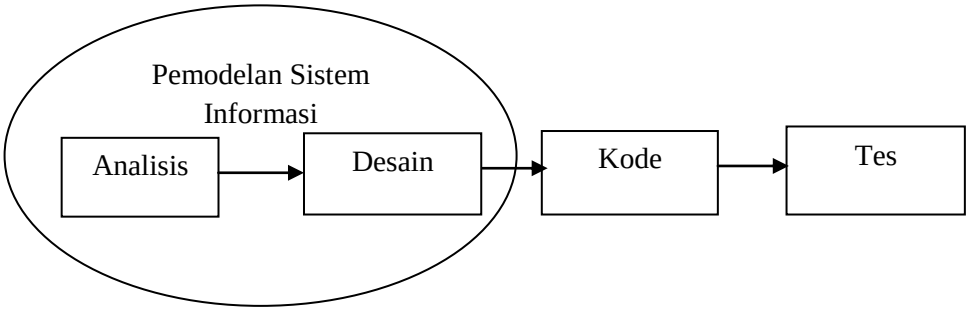
5. Pengujian

Sekali program dibuat, pengujian program dimulai. Proses pengujian berfokus pada logika internal software, memastikan bahwa semua pernyataan sudah diuji, dan pada eksternal fungsional, yaitu mengarahkan pengujian untuk menemukan kesalahan – kesalahan dan memastikan bahwa input yang dibatasi akan memberikan hasil aktual yang sesuai dengan hasil yang dibutuhkan.

6. Pemeliharaan

Software akan mengalami perubahan setelah disampaikan kepada pelanggan (perkecualian yang mungkin adalah software yang dilekatkan). Perubahan akan terjadi karena kesalahan – kesalahan ditentukan, karena software harus disesuaikan untuk mengakomodasi perubahan – perubahan di dalam lingkungan eksternalnya (contohnya perubahan yang dibutuhkan sebagai akibat dari

perangkat peripheral atau sistem operasi yang baru), atau karena pelanggan membutuhkan perkembangan fungsional atau unjuk kerja. Pemeliharaan software mengaplikasikan lagi setiap fase program sebelumnya dan tidak membuat yang baru lagi.



Gambar 2.16. Model Skuensial Linear

Masalah yang kadang terjadi ketika model sekuensial linier diaplikasikan adalah :

1. Jarang sekali proyek nyata mengikuti aliran sekuensial yang dianjurkan oleh model. Meskipun model linier bisa mengakomodasi iterasi, model ini melakukannya dengan cara tidak langsung. Sebagai hasilnya, perubahan – perubahan dapat menyebabkan keraguan pada saat tim proyek berjalan.

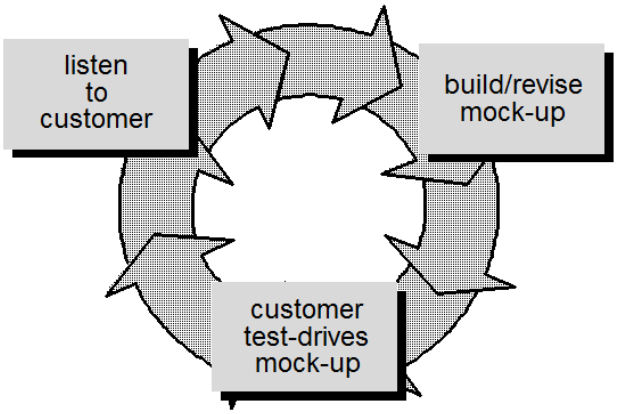
2. Kadang – kadang sulit bagi pelanggan untuk menyatakan semua kebutuhannya secara eksplisit. Model linier sekuensial memerlukan hal ini dan mengalami kesulitan untuk mengakomodasi ketidakpastian natural yang ada pada bagian awal beberapa proyek. Pemodelan Sistem Informasi

3. Pelanggan harus bersifat sabar. Sebuah versi kerja dari program – program kerja itu tidak akan diperoleh sampai akhir waktu proyek dilalui. Sebuah kesalahan besar, jika tidak terdeteksi sampai program yang bekerja tersebut dikaji ulang, bisa menjadi petaka.

4. Pengembang sering melakukan penundan yang tidak perlu. Sifat alami dari siklus kehidupan klasik membawa kepada *blocking state* di mana banyak anggota tim proyek harus menunggu tim yang lain untuk melengkapi tugas yang saling memiliki ketergantungan. Blocking state cenderung menjadi lebih lazim pada awal dan akhir sebuah proses sekuensial linier.

2.8.2. Model Prototype

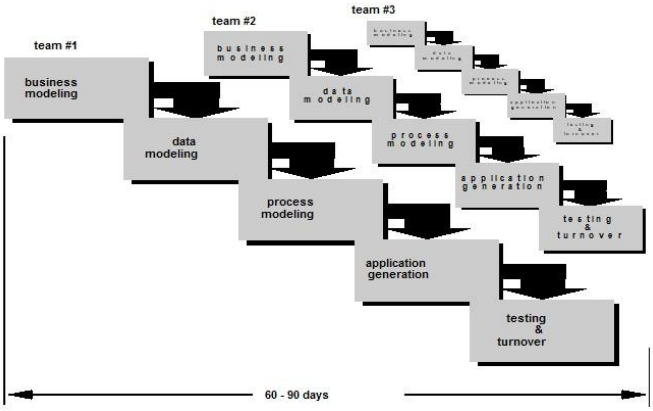
Model prototype dibangun dari mengumpulkan berbagai kebutuhan, kemudian tim pengembang akan nertemu dengan pelanggan untuk menentukan tujuan dari perangkat lunak, dan mengidentifikasi kebutuha-kebutuhan yang telah diketahui oleh pelanggan, dan batasan-batasan apa saja yang dapat dikategorikan sebagai tugas utama. Hasilnya akan dibangun rancangan sementara yang mewakili berbagai aspek dari perangkat lunak yang kelak akan digunakan oleh pelanggan/pengguna (seperti bentuk pendekatan input yang digunakan dan bentuk output). Idealnya model prototype melayani sebuah mekanisme untuk mengidentifikasi kebutuhan perangkat lunak. Dimana jika nantinya sebuah model prototype berhasil dibuat, seorang developer harus berusaha mendayagunakan tools yang ada (semisal, report generator, windows manager) dapat bekerja dengan baik (cepat).



Gambar 2.17. Model Prototype

2.8.3. Model RAD

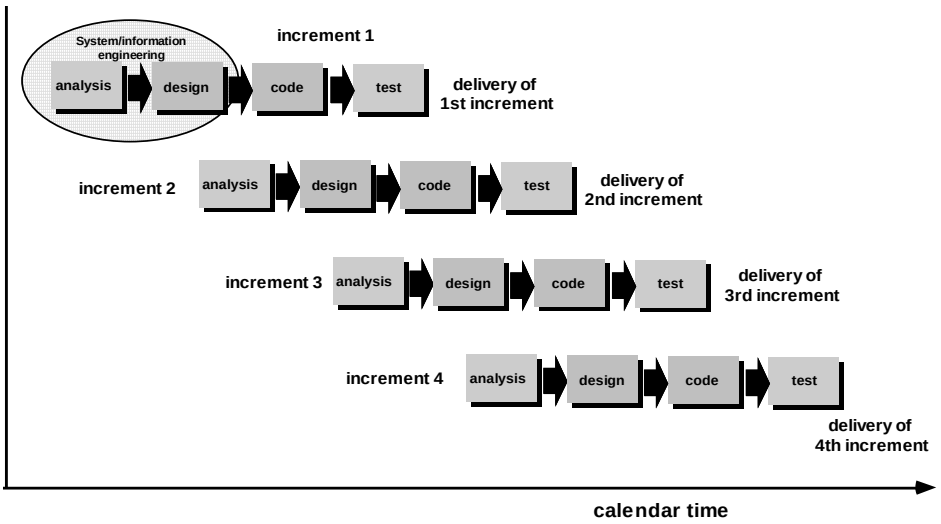
Rapid Aplication Development (RAD) adalah sebuah model proses perkembangan perangkat lunak sekuensial linier yang menekankan siklus perkembangan yang sangat pendek. Model RAD ini merupakan sebuah adaptasi "kecepatan tinggi" dari model sequensial linier dimana perkembangan cepat dicapai dengan menggunakan pendekatan konstruksi berbasis komponen.



Gambar 2.18. Model RAD

2.8.4. Model Evolusioner

Model evolusioner adalah model iterative. Model itu ditandai dengan tingkah laku yang memungkinkan perekayasa perangkat lunak mengembangkan versi perangkat yang lebih lengkap sedikit demi sedikit.



Gambar 2.19. Model Evolusioner

2.9. Masalah-masalah yang Penting Pada Sistem Pengenalan Ucapan

Pada kenyataannya, Sistem Pengenalan Ucapan sampai saat ini masih perku meningkatkan bebrapa hal guna memperbaiki kinerjanya, terutama masalh akurasi. Dalam pengenalan ucapan disebutkan beberapa masalah mengapa sulit meningkatkan akurasi. Masalah tersebut dijelaskan sebagai berikut :

1. Semakin besar ukuran kosakata, semakin tinggi kesalahan dalam pengenalan ucapan. Sebagai contoh, 10 digit dari "nol" sampai "Sembilan" dapat dengan mudah dikenali. Tetapi dengan meningkatkannya ukuran vocabulary menjadi 200, 5000, atau 100000 maka bisa terjadi rata-rata kealahan 3%, 7%, atau 45%

sementara itu, setiap bahasa mempunyai huruf ketika diucapkan sulit untuk dibedakan. Sebagai contoh huruf C, D. J dalam kasus bahasa Indonesia.

2. Tingkat ketergantungan kepada pembicara. Sistem pengenalan ucapan *Speaker Dependence* hanya ditujukan unutk pembicara tunggal. Sedangkan *Speaker Indenpence* ditujukan untuk siapapun. Jenis *Speaker*

Independence sulit untuk dilakukan karena parameter suatu system pengenalan ucapan biasanya dilatih menggunakan *Speaker* tertentu, yang sangat spesifik ucapannya. Hasil penelitian menunjukan bahwa *Speaker Independence* mempunyai kesalahan rata-rata 3 sampai 5 kali lebih besar dibanding *Speaker Dependence.*

3. Penggalan ucapan, *Isolated Speech* berarti ucapan kata tunggal. *Discontinuous Speech* berarti satu kalimat penuh tetapi kata dipilah berdasarkan keadaan diam *(silence)*. *Isolated* dan *Continous Speech* relatif lebih mudah karena pengglan kata dapat dikenali dan kata per kata dapat diucapkan lebih "bersih". Sedangkan *Contionous Speech* lebih sulit dilakukan karena tidak diketahui pengaglan kata dan juga dipengaruhi oleh artikulasi gabungan kata.

4. Persyaratan bahasa. Berbagai pekerjaan sulit untuk dilakukan bilamana tidak memenuhi syarat-syarat penggunaan bahasa. Termasuk didalamnya adalah Sintaks, Simantik dan Grammarnya. Sebagai contoh, *query* terhadapa jasa penerbangan dengan pernyataan 'Pesawat itu berwarna merah', diterima oleh Sintaks, Simantik maupun Grammar. Sedangkan pernyataan 'Pesawat itu sedang marah', diterima Sintaks dan Grammar tetapi ditolak oleh Simantik. Sehingga,

semua struktur bahasa perlu ditinjau dalam membangun sistem pengenalan ucapan.

5. Ucapan Spontanitas. Sistem pengenalan ucapan dapat ditinjau dari pembicara yang sedang membaca skrip atau berbicara secara spontan. Pembicara spontan seperti emm, uh, e dan lain-lain, lebih sulit dikenali.

6. Kondisi lingkungan. Kinerja system pengenalan ucapan dapat juga di uji melalui lingkungan alamiah yang terdapat banyak interferensi, derau dan lain-lain. Sehingga usaha untuk dapat mengenali ucapan dari pembicara yang dimaksudkan dengan suara bersih telah banyak diteliti dengan berbagai metode. Diantaranya, penghilangan derau, penggandaan mikrofon, pembatas lebar pita frekuensi, dan perubahan gaya ucapan.

Bahkan ada kelemahan Sistem Pengenalan Ucapan yang mendasar, yaitu bahasa pembicaraan akan lebih efektif bila diterapkan pada interaksi antara manusia dengan komputer. Hal ini diakibatkan oleh lambatnya komputer merepresentasikan informasi dan sulitnya melakukan *review* dan *edit*. Sehingga perancangan Sistem Pengenalan Ucapan harus mengerti tentang *acoustic memory* dan *prosody*

2.10. Borland Delphi

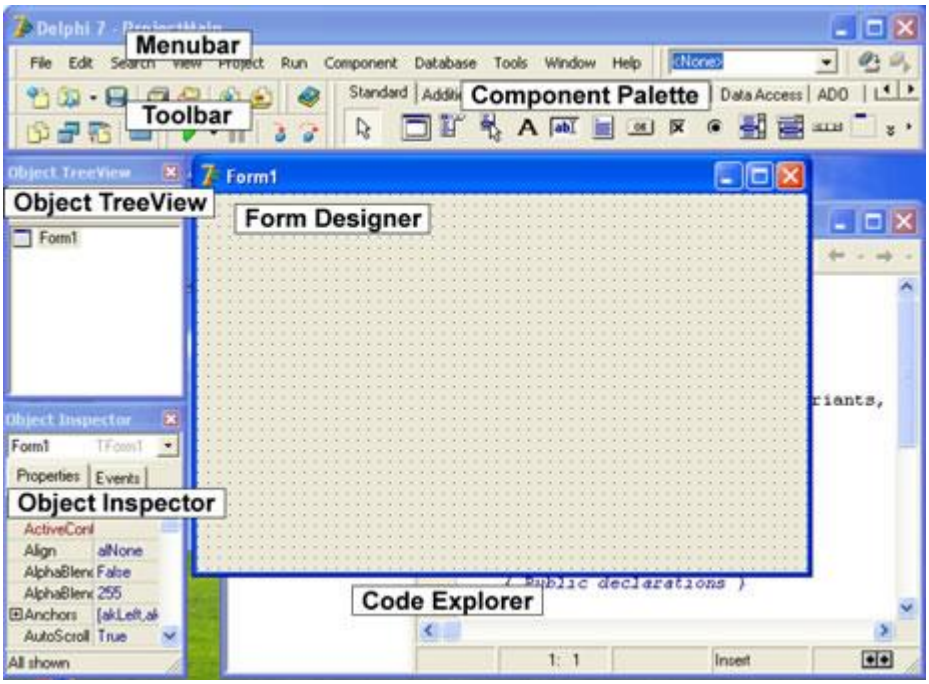
Borland Delphi adalah sebuah alat pengembangan aplikasi-aplikasi untuk sistem operasi Microsoft Windows. Delphi sangat berguna dan mudah digunakan untuk membuat suatu program berbasis GUI (Graphical user interface) atau console (mode teks).

Semua user interface seperti form, tombol (button), dan objek list-list telah disertakan dalam Delphi dalam bentuk komponen atau control. Pengembang dapat dengan mudah menempatkan komponen-komponen tersebut ke dalam form. Pengembang dapat juga menempatkan control ActiveX pada form untuk membuat program-program khusus seperti Browser Web dalam waktu yang cepat. Delphi memungkinkan pengembang untuk merancang keseluruhan interface secara visual, dan dengan cepat dapat diimplementasikan sebuah kode perintah berbasis event (event driven) dengan mengklik mouse. Dengan Delphi, pengembang perangkat lunak dapat membuat program Windows dengan lebih cepat dan lebih mudah dari sebelumnya.

2.10.1. Pengenalan IDE Delphi

IDE adalah sebuah singkatan dari Integrated Development Environment yaitu sebuah lingkungan pengembangan yang terintegrasi, istilah IDE popular untuk menyebut software2 bahasa pemrograman dimana proses pengembangan programmnya mulai dari koding, designing dan debugging dilakukan pada satu framework atau pada satu aplikasi yang terintegrasi.

IDE Delphi tersebut terbagi menjadi tujuh bagian utama, yaitu : Menubar, Toolbar, Component Palette, Form Designer, Code Explorer, Object Treeview dan Object Inspector.



Gambar 2.20. IDE Delphi

Menubar dan Toolbar merupakan dua bagian yang biasanya terdapat pada aplikasi-aplikasi windows lain. Fungsi dari Menubar dan Toolbar ini relatif hampir sama dengan aplikasi window lain, sehingga secara explisit tidak akan dibahas dalam buku ini, melainkan hanya menubar yang biasa digunakan dalam pembuatan program.



Gambar 2.21. Menubar dan Toolbar

Component Palette berisi kumpulan VCL (Visual Component Library) yang berguna dalam desain aplikasi. VCL merupakan pustaka untuk komponen visual, dimana dalam component palette dilambangkan dengan ikon yang

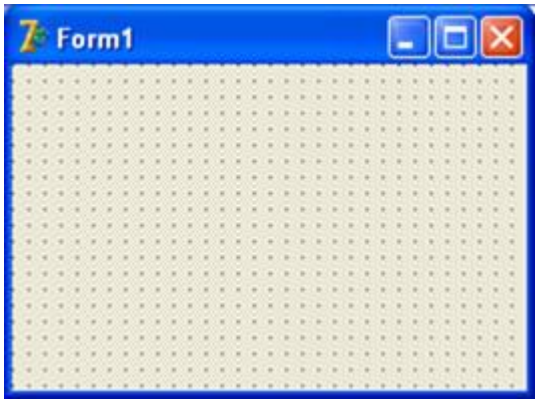
merepresentasikan komponen tersebut. Komponen-komponen VCL pada component palette dikelompokkan ke dalam tab tab, sesuai dengan fungsinya, dengan maksud untuk memudahkan programmer dalam memilih komponen yang diinginkannya.

Disamping VCL ada juga CLX (Component Library for Cross Platform), jika Anda ingin tahu lebih banyak baik mengenai VCL maupun CLX, Anda bisa mendapatkannya pada buku-buku lain yang khusus membahas tentang dasar dasar pemrograman Delphi.



Gambar 2.22. Componnent Pallete

Setiap aplikasi biasanya memiliki jendela atau background interface, yang dalam bahasa pemrograman Delphi atau bahkan dalam bahasa pemrograman lain yang berbasis visual, biasa disebut dengan Form. Form Designer berfungsi sebagai tempat untuk mendesain form untuk aplikasi yang akan kita buat, dan juga sebagai tempat untuk meletakkan komponen-komponen yang kita ambil dari component palette.



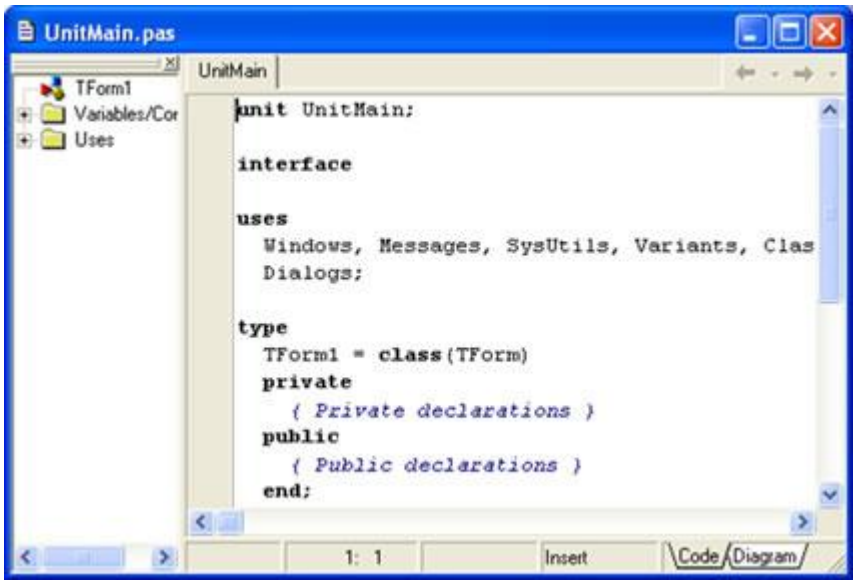
Gambar 2.23. Form Designer

Code Explorer merupakan area di mana kita menuliskan kode program, posisinya secara default terletak dibelakang form. Untuk menampilkan code explorer di depan form, Anda bisa menggunakan tombol F12 pada keyboard. Pada code explorer tersebut, Anda akan melihat kode-kode dalam bahasa pemrograman delphi yang secara otomatis digenerate oleh Delphi, hal ini jelas akan memudahkan atau mempercepat kita dalam menulis program.

Pada code explorer Delphi, ada sebuah fitur yang disebut dengan code completion. Sesuai dengan namanya, code completion berfungsi melengkapi kode yang kita tulis dalam bentuk pilihan/list dari code-code yang bisa kita gunakan, hal ini akan sangat membantu apabila kita lupa terhadap kode tertentu.

Code completion ini secara otomatis akan muncul ketika Anda menekan tombol titik pada keyboard, selain itu untuk menampilkan code completion ini, Anda juga bisa melakukannya dengan menekan kombinasi tombol Ctrl + Space secara bersamaan.

Disamping code completion, code explorer pada delphi juga dilengkapi dengan hint (layer berwarna kuning yang muncul ketika mouse didekatkan pada komponen tertentu). Hint ini, muncul ketika kita menggunakan procedure atau fungsi dan menekan tombol (pada keyboard, pada hint tersebut akan muncul nama dan tipe data dari parameter yang digunakan atau informasi lain yang diperlukan.



Gambar 2.24. Code Explorer

Object TreeView adalah bagian yang berisi daftar komponen yang digunakan dalam form designer. Dengan model tampilan bercabang, akan memudahkan kita dalam menunjuk komponen tertentu terutama jika aplikasi yang dibuat menggunakan banyak komponen.

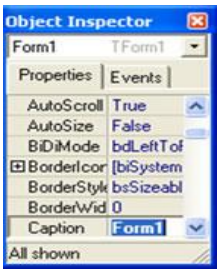


Gambar 2.25. Object TreeView

Object Inspector adalah bagian yang digunakan untuk memanipulasi sifat atau karakteristik dan event dari komponen yang kita gunakan dalam form designer. Jendela object inspector terbagi menjadi dua bagian tab, yaitu tab property dan tab event. Tab property digunakan untuk memanipulasi properti

yang dimiliki oleh komponen tertentu, misalkan ukuran, warna dan caption dari komponen. Sedangkan Tab Event digunakan untuk menangani pemasukan kode pada kejadian tertentu dari suatu komponen, misalnya : kejadian ketika komponen button (tombol) diklik atau onClick.

Sebagaimana yang telah dijelaskan di atas, salah satu fungsi dari object inspector ini adalah untuk mengatur property dari komponen, pengaturan tersebut tidak akan disimpan pada file unit, melainkan akan disimpan pada file form (*.dfm).



Gambar 2.26. Object Inspector

2.10.2. Kelebihan Borland Delphi

Kelebihan-kelebihan yang dapat diambil ketika seorang pengembang perangkat lunak menggunakan Borland Delphi adalah :

1. Delphi mendukung Pemrograman berbasis OOP (Object Oriented Programing)
2. Pengembangan secara Cepat RAD (Rapid Application Development)
3. Menggunakan bahasa tingkat tinggi
4. Hasil dari proses kompilasi berupa sebuah file yang dapat dieksekusi (executable file) sehingga mempermudah dalam pendistribusian program dan mengurangi banyaknya file pendukung dan lain-lain.

2.10.3. Tipe Data Delphi

Borland Delphi memeliki 6 (enam) tipe data, yaitu :

1. Tipe Data Integer atau Ordinal
2. Tipe Data Boolean
3. Tipe Data Floating Point atau Real
4. Tipe Data Enumerated
5. Tipe Data Sub Range
6. Tipe Data String dan Character