BAB 2
LANDASAN TEORI

2.1 Sejarah Singkat Tulisan Arab Melayu

Sejarah tulisan Arab Melayu (yang di Pulau Jawa dikenal dengan huruf *Arab Pegon* dan populer di Malaysia dengan istilah tulisan *Jawi*) di Nusantara adalah bermula semenjak bangsa Melayu menerima agama islam, menggantikan kepercayaan dan agama-agama sebelumnya (Animisme, Hindu, dan Budha). Jika dikaitkan dengan masa masuknya agama Islam di Nusantara, maka adanya huruf Arab Melayu bermula kira-kira sejak tahun 1200 M/ 1300 M[2] .

Beberapa fakta sejarah yang membuktikan dipakainya tulisan Arab Melayu seja 700 / 800 tahun yang lalu adalah.

1. Penemuan batu-batu bersurat, antara lain di Terangganu (Malaysia), Aceh, Riau, Sumatra, dan daerah-daerah lain.
2. Makam para raja melayu di seluruh Nusantara yang menggunakan huruf Arab Melayu di batu nisannya.
3. Perhubungan antarbangsa, khususnya dengan Eropa, yang di antaranya berupa perjanjian raja-raja melayu islam dengan pihak kolonial Inggris telah menggunakan tulisan Arab Melayu.

2.2 Pelambangan Huruf Arab Melayu

Sesuai dengan namanya, huruf-huruf tulisan Arab Melayu tentunya juga mempergunakan huruf Arab Hijaiyah.Namun demikian, tidak semua huruf Arab Hijayiah diserap sebagai huruf Arab Melayu, karena faktor penyesuaian bahasa.Jika huruf Arab Hijaiyah ada yang diberi syakal/tanda baca (terutama Alquran), maka penulisan huruf Arab Melayu tidak menggunakan tanda baca.Oleh karenanya, diperlukan beberapa rumusan dasar, terutama terkait dengan cara membaca tulisan Arab Melayu.

Tabel 2.1 Huruf Arab Hijaiyah

No	Huruf Arab	Huruf Latin	No	Huruf Arab	Huruf Latin
1	ا	Alif	15	ض	Dhad
2	ب	Ba'	16	ط	Tha'
3	ت	Ta'	17	ظ	Zha'
4	ث	Tsa'	18	ع	Ain
5	ج	Jim	19	غ	Ghain
6	ح	Ha'	20	ف	Fa'
7	خ	Kho'	21	ق	Qaf
8	د	Dal	22	ك	Kaf
9	ذ	Dzal	23	ل	Lam
10	ر	Ro'	24	م	Mim
11	ز	Zai'	25	ن	Nun
12	س	Sin	26	و	Wawu
13	ش	Syen	27	ه	Ha'
14	ص	Shad	28	ي	Ya'

Seperti abjad rumawi, jenis huruf dalam abjad huruf Arab Melayu juga memiliki kategori vokal dan konsonan.Vokal Arab Melayu hanya dilambangkan tiga huruf saja yaitu ي (ya'), ا (alif),and و (wau) dengan berbagai variasinya.Konsonan Arab Melayu dibedakan ke dalam tiga kategori yaitu konsonan dari huruf Arab Hijaiyah, huruf jati Arab Melayu, dan konsonan Arab yang menjadi bunyi pinjaman bahasa melayu.

1. Huruf Vokal Arab Melayu.

Tabel 2.2 Huruf Vokal Arab Melayu

Rumawi	A	I	U	E	O
Arab Melayu	ا	ي	و	ي	و

Vokal [i] dan [e] menggunakan lambang yang sama (ي), seperti halnya lambang (و).

2. Huruf Konsonan Arab Melayu.
    a. Konsonan dari huruf Arab Melayu

Tabel 2.3 Konsonan Huruf Arab Melayu

Rumawi	**Arab Melayu**
B	ب
D	د
F	ف
H	ه
J	ج
K	ك
L	ل
M	م
N	ن
Q	ق
R	ر
S	س
T	ت
W	و
Y	ي
Z	ز

b. Huruf jati Arab Melayu

Tabel 2.4 Huruf Jati Arab Melayu

Rumawi	P	G	C	Ny	V	Ng
Arab Melayu	ڤ	ݢ	چ	ڽ	ۏ	ڠ

c. Konsonan Arab Hijaiyah yang menjadi bunyi pinjaman tulisan Arab Melayu.

Tabel 2.5 Huruf Konsonan Hijaiyah Menjadi Bunyi Pinjaman

Rumawi	**Arab Melayu**
Tsa	ث
Ha	ح
Kha	خ
Dzal	ذ
Syin	ش
Shod	ص
Dhad	ض
Tha’	ط
Zha’	ظ
‘Ain	ع
Ghain	غ

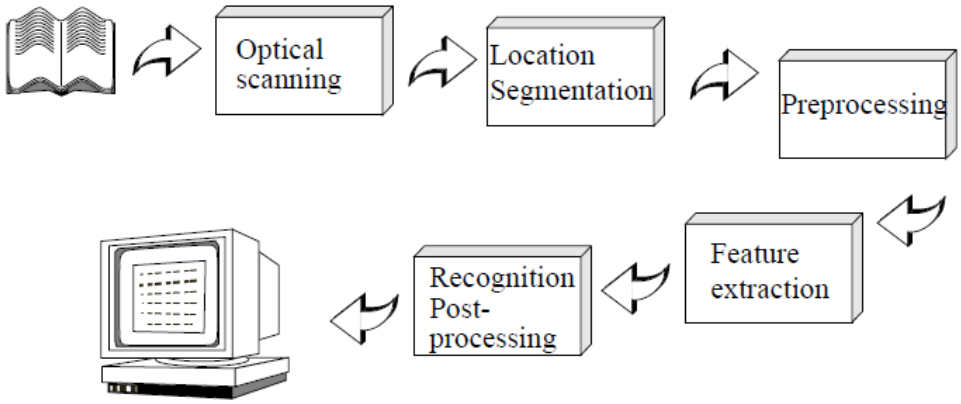
2.3 *Handwriting Recognition* (Pengenalan Tulisan Tangan)

Handwriting Recognition merupakan bagian dari *Optical Character Recognition* (OCR), dimana karakter masukan berasal dari tulisan tangan manusia yang kemudian diubah secara digitalisasi oleh komputer menjadi sebuah citra. OCR adalah sebuah sistem komputer yang dapat membaca huruf, baik yang berasal dari sebuah pencet maupun yang berasal dari tulisan tangan.

OCR adalah aplikasi yang menerjemahkan gambar karakter (*image character*) menjadi bentuk teks dengan cara menyesuaikan pola karakter per baris dengan pola yang telah tersimpan di dalam aplikasi. Hasil dari proses OCR adalah

berupa teks sesuai dengan gambar *output scanner* dimana tingkat keakuratan penerjemahan karakter tergantung dari tingkat kejelasan gambar dan metode yang digunakan.

Berikut ilustrasi dari aplikasi OCR:



Gambar 2.1 Proses Pada OCR (Line Eikvil, 1993)

Prinsip kerja dari aplikasi OCR adalah sebagai berikut.

1. Memasukkan dokumen berisi teks (teks cetakan mesin) ke dalam alat optik (scanner) sehingga didapat sebuah file citra.
2. File citra tersebut diproses menggunakan perangkat lunak aplikasi pengenalan teks, di mana perangkat ini melakukan proses pengenalan terhadap karakter-karakter yang ada pada file citra tersebut.
3. Keluaran dari perangkat lunak aplikasi pengenalan teks ini berupa file teks yang berisi karakter-karakter yang telah dikenali dan siap untuk diolah lebih lanjut.

2.4 Citra Digital

Definisi citra menurut Kamus Webster adalah suatu representasi,kemiripan, atau imitasi dari suatu objek atau benda.Citra dapat dikelompokkan menjadi citra tampak dan citra tak tampak.Banyak contoh citra tampak dalam kehidupan sehari-hari, misalnya foto keluarga, lukisan, apa yang terlihat di layar monitor dan telivisi, serta hologram (citra optis).Sedangkan citra

tak tampak misalnya data gambar dalam file (citra digital) dan citra yang direpresentasikan menjadi fungsi matematis

Citra digital dibentuk oleh kumpulan titik yang dinamakan piksel (*pixel* atau "*picture element"*).Setiap piksel digambarkan sebagai satu kotak kecil.Setiap piksel mempunyai koordinat posisi.

Sebuah piksel mempunyai koordinat berupa (x,y). Koordinat x menyatakan posisi kolom dan koordinat y menyatakan posisi baris.Piksel pojok kiri-atas mempunyai koordinat (0,0) dan piksel pada pojok kanan-bawah mempunyai koordinat (N-1, M-1).

2.5 Pengolahan Citra Digital

Pengolahan citra pada dasarnya dilakukan dengan cara memodifikasi setiap titik dalam citra sesuai keperluan,.Secara garis besar, modifikasi tersebut dikelempokan menjadi:

1. Operasi titik, di mana setiap titik diolah secara tidak gayut terhadap titik-titik yang lain.
2. Operasi global, di mana karakteristik global (biasanya berupa sifat statistik) dari citra digunakan untuk memodifikasi nilai setiap titik.
3. Operasi temporal/berbasis bingkai, di mana sebuah citra diolah dengan cara dikombinasikan dengan citra lain.
4. Operasi geometri, di mana bentuk, ukuran, tau orientasi citra dimodifikasi secara geometris.
5. Operasi banyak titik bertetangga, di mana data dari titik-titik yang bersebelahan (bertetangga) dengan titik yang ditinjau ikut berperan dalam mengubah nilai.
6. Operasi morfologi, yaitu operasi berdasarkan segmen atau bagian dalam citra yang menjadi perhatian.

2.5.1 Jenis Citra

Ada tiga jenis citra yang umum digunakan dalam pemrosesan citra. Ketiga jenis citra tersebut yaitu citra berwarna, citra berskala keabuan, dan citra biner.

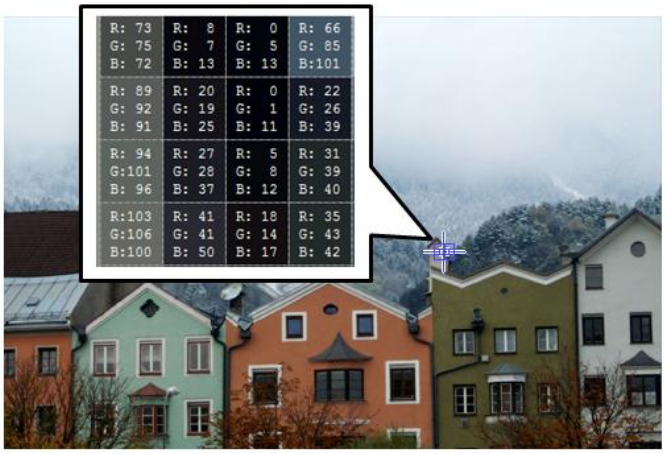
1) Citra Berwarna

Citra berwarna, atau biasa dinamakan citra RGB, merupakan jenis citra yang menyajikan warna dalam bentuk komponen R (merah), G (hijau), dan B (biru). Setiap komponen warna menggunakan 8 bit (nilainya berkisar antara 0 sampai dengan 255). Dengan demikian, kemungkinan warna yang bisa disajikan mencapai 255 x 255 x 255 atau 16.581.375 warna. Tabel 2.6 menunjukkan contoh warna dan nilai R,G, dan B.

Tabel 2.6 Nilai R,G, dan B

Warna	R	G	B
Merah	255	0	0
Hijau	0	255	0
Biru	0	0	255
Hitam	0	0	0
Putih	255	255	255
Kuning	0	255	255

Sebuah warna tidak hanya dinyatakan dengan komposisi R,G dan B tunggal. Pada Tabel 2.4 terlihat bahwa warna merah mempunyai R= 255, G=0, dan B=0.Namun, komposisi R = 254, G=1, B=1 juga berwarna merah.



Gambar 2.2 Citra berwarna dan representasi warnanya. Setiap piksel dinyatakan dengan nilai R, G, dan B

2) Citra Keabuan (*Grayscale*)

Citra gray scale merupakan citra digital yang hanya memiliki satu nilai kanal pada setiap pixelnya, dengan kata lain nilai bagian RED = GREEN =BLUE. Nilai tersebut digunakan untuk menunjukkan tingkat intensitas. Warna yang dimiliki adalah warna dari hitam, keabuan, dan putih. Tingkatan keabuan di sini merupakan warna abu dengan berbagai tingkatan dari hitam hingga mendekati putih.Citra gray scale berikut memiliki kedalaman warna 8 bit (256 kombinasi warna keabuan).



Gambar 2.3 Citra Keabuan

3) Citra Biner (Monokrom)

Citra biner adalah citra digital yang hanya memiliki dua kemungkinan nilai pixel yaitu hitam dan putih.Citra biner juga disebut sebagai citra B&W (black and white) atau citra monokrom.

Pada citra biner, setiap titik bernilai 0 atau 1 dan masing-masing merepresentasikan warna tertentu. Contohnya adalah warna hitam bernilai 0 dan warna putih bernilai 1.Pada standar citra untuk ditampilkan di layar komputer, nilai biner ini berhubungan ada atau tidaknya cahaya yang ditembakkan oleh electron gun yang terdapat di dalam monitor komputer. Angka 0 menyatakan tidak ada cahaya maka warna yang direpresentasikan adalah hitam. Untuk angka 1 terdapat cahaya sehingga warna yang direpresentasikan adalah putih. Standar ini disebut sebagai standar citra

cahaya, sedangkan standar citra tinta/cat adalah berkebalikan karena nilai binertersebut menyatakan ada atau tidaknya tinta.



Gambar 2.4 Citra Biner

2.6 Ekstraksi Ciri atau fitur

Menurut Abdul Kadir dan Adhi Susanto, fitur suatu objek merupakan karakteristik yang melekat pada objek.Fitur bentuk merupakan suatu fitur yang diperoleh melalui bentuk objek dan dapat dinyatakan melaui kontur, area, dan transformasi.

2.6.1 *Connected Component Labeling*

Connected component labeling merupakan teknik yang juga bisa digunakan untuk mengklasifikasikan *region* atau objek dalam citra digital. Teknik ini memanfaatkan teori *connectivity* piksel pada citra. Piksel-piksel dalam *region* disebut *connected* (ada konektifitasnya atau *connectivity*) bila mematuhi aturan kedekata pada piksel . Aturan kedekatan piksel memanfaatkan sifat ketetanggaan piksel . Dengan demikian piksel-piksel yang di katakan terhubung pada dasarnya memiliki sifat kedekatan satu sama lain karena mereka masih memiliki hubungan *neighbourhood* atau ketetanggaan.

Proses *connected component labeling* memeriksa piksel-piksel disekitar citra atau *region*. Citra di-*scan* dari kiri ke kanan, dimulai dari atas citra hingga ke bawah citra untuk mencari piksel-piksel yang saling berdekatan atau terhubung. Menurut menurut Gonzales dan Woods ada dua jenis konektivitas yang digunakan pada citra dua dimensi yaitu 4 konektivitas dan 8 konektivitas.

4 Konektivitas *(4-Connected Neighbors)*

	P(x, y-1)	
P (x-1, y)	P (x,y)	P(x+1, y)
	P(x, y+1)	

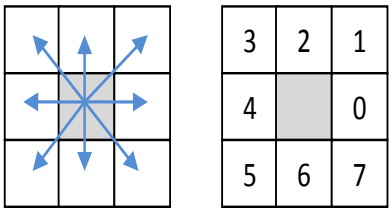
8 Konektivitas *(8-Connected Neighbors)*

P (x-1, y-1)	P(x, y-1)	P (x+1, y-1)
P (x-1, y)	P (x,y)	P(x+1, y)
P (x-1, y+1)	P(x, y+1)	P (x+1, y+1)

Gambar 2.5 Konektivitas pada *connected component labeling*

2.6.2 *Freeman Chain Code*

Rantai kode (*chain code*) merupakan contoh representasi kontur yang mula-mula diperkenalkan oleh Freeman pada tahun 1961. Representasi bentuk dilakukan dengan menggunakan pendekatan 8-ketetanggaan[8] .Kode rantai setiap tetangga piksel dinyatakan dengan sebuah angka sebagaimana terlihat pada gambar 2.6.

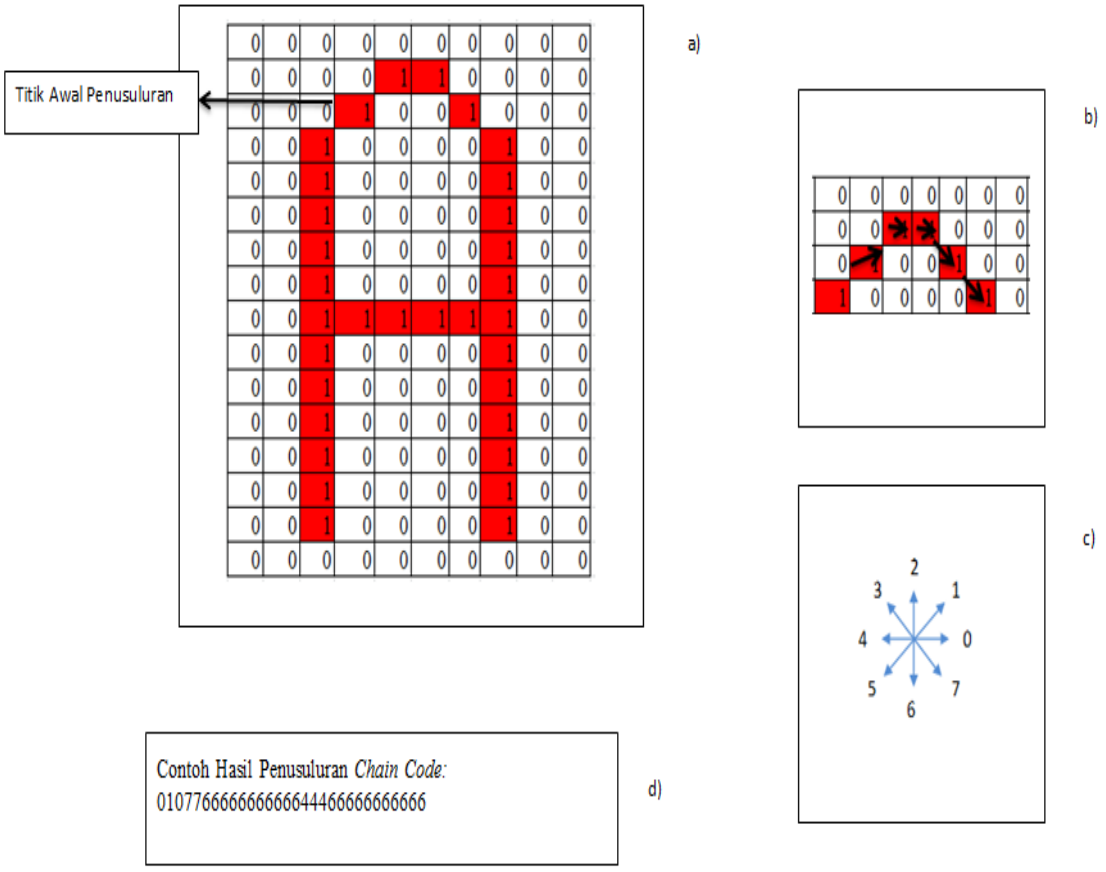


Gambar 2.6 Arah rantai kode beserta kodenya

Mengenali suatu pola dari suatu objek di dalam citra membutuhkan adanya ciri-ciri khusus. Ciri-ciri berguna untuk membedakan antar pola satu objek dengan objek yang lainnya.Ciri yang bagus adalah ciri yang memiliki pembeda pola yang tinggi, sehingga pengelompokkan pola berdasarkan ciri-ciri yang dimiliki dapat menghasilkan keakuratan yang tinggi.Ekstrasi ciri merupakan proses pengambilan suatu ciri-ciri dari suatu objek di dalam citra untuk membedakan objek yang satu

dengan yang lainnya. *Chain code* adalah metode yang melakukan penelusuran piksel-piksel objek dengan panduan arah mata angin.

Teknik *chain-code* dapat digunakan untuk menemukan struktur pembentuk suatu objek.Hasil akhir dari teknik *chain-code* adalah sebuah vektor ciri yang berisi informasi urutan kode chain-code pembentuk objek.



Gambar 2.7 Contoh penelusuran piksel dengan *chain code* dan hasil penelusuran

2.6.2.1 Normalisasi *chain code*

Freeman chain code merepresentasikan dengan lengkap dari bentuk suatu objek yang dapat dihitung setiap fitur objek dari kode rantai freeman.Normalisasi *freeman chain code* bertujuan untuk mengubah panjang kode rantai yang diinginkan.Normalisasi mengubah matriks satu dimensi menjadi matriks dua dimensi.Pada matriks dua dimensi, baris pertama mengandung nilai dari *chain code* dan baris kedua berisi frekuensi nilai tersebut. Kemudian menghapus nilai *chain code* jika memiliki frekuensi 1 dan membangun ulang nilai *chain code*

dengan panjang *chain code* yang ditetapkan. Nilai frekuensi baru code ke-i didapatkan dengan rumus[5]:

$$\boldsymbol{F_t^n = \frac{F_i}{\sum F_i} * 10} \qquad \text{Persamaan 2.1}$$

Dimana:

- F_t^n = Nilai *chain code* yang baru
- F_i = Frekuensi kode ke-i
- $\sum F_i$= total semua frekuensi

2.7 Klasifikasi

Klasifikasi adalah proses untuk menemukan model atau fungsi yang menjelaskan atau membedakan konsep atau kelas data, dengan tujuan untuk dapat memperkirakan kelas dari suatu objek yang labelnya tidak diketahui. Proses klasifikasi biasanya dibagi menjadi dua fase yaitu *fase learning* dan *fase test*. Pada *fase learning*, sebagian data yang telah diketahui kelas datanya diumpankan untuk membentuk model perkiraan. Kemudian pada *fase test* model yang sudah terbentuk diuji dengan sebagian data lainnya untuk mengetahui akurasi dari model tersebut. Bila akurasinya mencukupi model ini dapat dipakai untuk prediksi kelas data yang belum diketahui. Teknik ini dapat memberikan klasifikasi pada data baru dengan memanipulasi data yang ada yang telah diklasifikasi dan dengan menggunakan hasilnya untuk menghitung jarak antara ciri-ciri citra *template* dan citra masukan.

2.7.1 Support Vector Machine (SVM)

Support Vector Machine (SVM) dikembangkan oleh Boser, Guyon, Vapnik, dan pertama kali dipresentasikan pada tahun 1992 di *Annual Workshop on Computational Learning Theory*. Konsep dasar SVM sebenarnya merupakan kombinasi harmonis dari teori-teori komputasi yang telah ada puluhan tahun sebelumnya, seperti margin hyperplane (Duda & Hart tahun 1973, Cover tahun 1965, Vapnik 1964, dsb.), kernel diperkenalkan oleh Aronszajn tahun 1950, dan demikian juga dengan konsep-konsep pendukung yang lain. Akan tetapi hingga

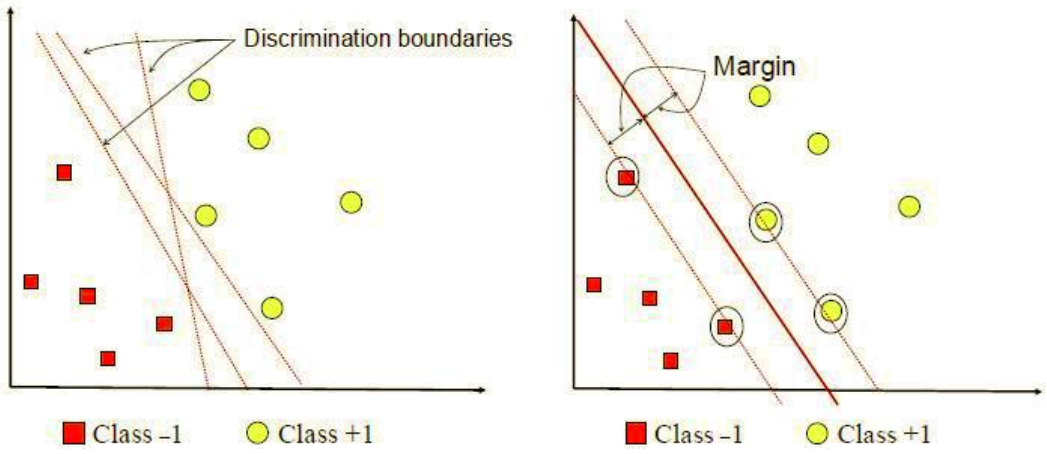
tahun 1992, belum pernah ada upaya merangkaikan komponen-komponen tersebut.

SVM adalah metode *machine learning* yang bekerja atas prinsip *Structural Risk Minimization* (SRM) dengan tujuan menemukan *hyperplane* terbaik yang memisahkan dua buah class pada input *space*. SVM dapat digunakan untuk klasifikasi yang dapat diterapkan pada deteksi tulisan tangan, pengenalan obyek, identifikasi suara dan lain-lain.

Berbeda dengan strategi neural network yang berusaha mencari hyperplane pemisah antar class, SVM berusaha menemukan hyperplane yang terbaik pada input space. Prinsip dasar SVM adalah linear classifier, dan selanjutnya dikembangkan agar dapat bekerja pada problem non-linear. dengan memasukkan konsep kernel trick pada ruang kerja berdimensi tinggi. Perkembangan ini memberikan rangsangan minat penelitian di bidang pattern recognition untuk investigasi potensi kemampuan SVM.

2.7.2 SVM Pada Data Terpisah Secara Linear

Kasus data yang terpisah secara linear adalah kasus yang termudah, seperti yang terlihat pada gambar 2.8.



Gambar 2.8 SVM berusaha menemukan hyperplane terbaik yang memisahkan dua kelas -1 dan +1

Gambar 2.8 memperlihatkan beberapa pattern yang merupakan anggota dari dua buah class : +1 dan -1 yang mempunyai tupel pelatihan 2-D. Pattern yang

tergabung pada class -1 disimbolkan dengan kotak berwarna merah sementara pattern pada class +1 disimbolkan dengan lingkaran berwarna kuning. Masalah klasifikasi dapat diterjemahkan dengan usaha menemukan *hyperplane* yang memisahkan antara kedua kelompok tersebut. *Hyperplane* pemisah terbaik antara kedua *class* dapat ditemukan dengan mengukur margin hyperplane dengan pattern terdekat masing-masing class. Pattern yang paling dekat ini disebut dengan support vector. Garis solid pada gambar 2.8 sebelah kanan menunjukan *hyperplane* terbaik, yaitu yang terletak pada tengah-tengah kedua class, sedangkan titik merah dan kuning yang berada dalam lingkaran hitam adalah *support vector*.
Data yang tersedia dinotasikan sebagai $x \in R^d$ sedangkan label masing-masing dinotasikan $y_i \ni \{-1, +1\}$ untuk $i = 1,2,3, \dots n$ yang mana n adalah banyaknya data. Diasumsikan kedua class dapat terpisah secara sempurna oleh hyperplane berdimensi *d* , yang didefinisikan:

$$\vec{w}.\vec{x} + b = 0 \qquad \text{Persamaan 2.2}$$

Pattern $\overrightarrow{x_i}$ yang termasuk class –1 (sampel negatif) dapat dirumuskan sebagai pattern yang memenuhi pertidaksamaan

$$\vec{w}.\vec{x} + b \leq -1 \qquad \text{Persamaan 2.3}$$

Sedangkan pattern $\overrightarrow{x_i}$ yang termasuk class +1 (sampel positif)

$$\vec{w}.\vec{x} + b \geq +1 \qquad \text{Persamaan 2.4}$$

Keterangan :

w = *vector* bobot

X = nilai masukan atribut

b = bias

Margin terbesar dapat ditemukan dengan memaksimalkan nilai jarak antara jarak dan titik terdekatnya, yaitu $1/||w||$. Hal ini dapat dirumuskan sebagai *Quadratic Programming* (QP) *problem*, yaitu mencari titik minimal persamaan 2.5, dengan memperhatikan constraint persamaan 2.6.

$$min_w = \tau(w) = 1/2 \ ||\mathrm{w}||^2 \qquad \text{Persamaan 2.5}$$

$$y_i(x_i.w + b) - 1 \geq 0 \qquad \text{Persamaan 2.6}$$

Problem ini dapat dipecahkan dengan berbagai teknik komputasi, diantaranya lagrange multiplier yang dinyatakan pada persamaan 2.7.

$$L(w,b,a) = \frac{1}{2\left|\left|w\right|\right|^2} - \sum_{i=1}^{l} \alpha_i (y_i((\overrightarrow{x_i} \,.\, \vec{w} + \, b) - 1))$$

Persamaan 2.7

dengan $i = 1,2,\dots,l$

Dimana α adalah lagrange multiplier, yang bernilai 0 atau positif $\alpha_i \geq 0$. Nilai optimal dari persamaan 2.7 dapat dihitung dengan meminimalkan L terhadap w dan b, dan memaksimalkan L terhadap a_i . Dengan memperhatikan sifat bahwa pada titik optimal gradient L=0 persamaan 2.7 dapat dimodifikasi sebagai maksimasi problem yang hanya mengandung a_i, sebagaimana terlihat pada persamaan 2.8 dan 2.29.

$$\sum_{1=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \overrightarrow{x_i} \,.\, \overrightarrow{x_j} \qquad \text{Persamaan 2.8}$$

$$\alpha \geq 0 \; (i = 1,2,\dots,l) \sum_{i=1}^{l} a_i y_i = 0 \qquad \text{Persamaan 2.9}$$

Dengan demikian, maka akan diperoleh α_i yang kebanyakan bernilai positif yang disebut sebagai *support vector* dan juga memperoleh persamaan 2.10 dan 2.11 sebagai solusi bidang pemisah.

$$w = \Sigma \alpha_i y_i x_i \qquad \text{Persamaan 2.10}$$

$$b = y_k - w^T x_k \qquad \text{Persamaan 2.11}$$

2.7.3 Kasus Data yang Tidak Terpisah Secara Linear

Kasus data yang tidak terpisah secara linear diasumsikan bahwa class pada input space tidak dapat terpisah secara sempurna. Hal ini menyebabkan constraint pada persamaan 2.6 tidak dapat terpenuhi, sehingga optimasisasi tidak dapat dilakukan, untuk mengatasi masalah ini SVM dirumuskan ulang

dengan memperkenalkan teknik softmargin. Dalam softmargin persamaan 2.6 dimodifikasi dengan menggunakan slack variabel sehingga terlihat pada persamaan 2.12.

$$y_i(x_i.w + b) \geq 1 - \varepsilon_i \qquad \text{Persamaan 2.12}$$

Dengan demikian persamaan 2.5 diubah menjadi persamaan 2.13.

$$min_w = \tau(w) = \frac{1}{2||w||^2} + C \sum_{i=l}^{l} \varepsilon_i \qquad \text{Persamaan 2.13}$$

Fitur C digunakan untuk mengontrol *tradeoff* antara margin dan kesalahan klasifikasi ε.

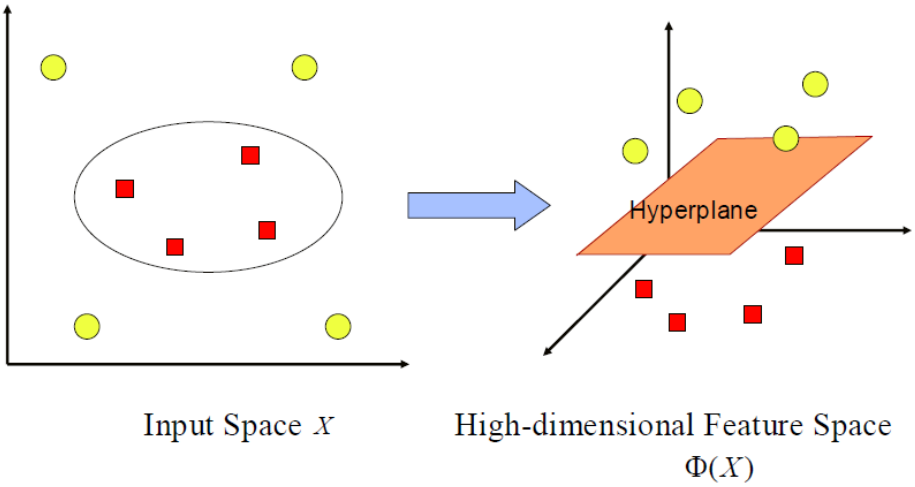
2.7.4 *Kernel Trick* dan *Non-Linear Classification* pada SVM

Pada umumnya masalah dalam domain dunia nyata (*real world problem*) jarang yang bersifat *linear separable*. Kebanyakan bersifat *non linear*. Untuk menyelesaikan problem *non linear*, SVM dimodifikasi dengan memasukkan fungsi *Kernel*.

Dalam non linear SVM, pertama-tama data x dipetakan oleh fungsi Φ (x) ke ruang vektor yang berdimensi lebih tinggi. Pada ruang vektor yang baru ini, hyperplane yang memisahkan kedua class tersebut dapat dikonstruksikan. Hal ini sejalan dengan teori Cover yang menyatakan"*Jika suatu transformasi bersifat non linear dan dimensi dari feature space cukup tinggi, maka data pada input space dapat dipetakan ke feature space yang baru, dimana pattern-pattern tersebut pada probabilitas tinggi dapat dipisahkan secara linear*".

Ilustrasi dari konsep ini dapat dilihat pada gambar 2.9. Pada gambar 2.9 diperlihatkan data pada class kuning dan data pada class merah yang berada pada input space berdimensi dua tidak dapat dipisahkan secara linear. Selanjutnya fungsi Φ memetakan tiap data pada input space tersebut ke ruang vektor baru yang berdimensi lebih tinggi (dimensi 3), dimana kedua class dapat dipisahkan secara linear oleh sebuah hyperplane. Notasi matematika dari mapping ini adalah sebagai berikut.

$$\Phi: \ \Re^d \rightarrow \Re^q \ d < q \qquad \text{Persamaan 2.14}$$



Gambar 2.9 Fungsi Φ memetakan data ke ruang vektor yang berdimensi lebih tinggi, sehingga kedua *class* dapat dipisahkan secara linear oleh sebuah *hyperplane*

Pemetaan ini dilakukan dengan menjaga topologi data, dalam artian dua data yang berjarak dekat pada input space akan berjarak dekat juga pada feature space, sebaliknya dua data yang berjarak jauh pada input space akan juga berjarak jauh pada feature space.

Selanjutnya proses pembelajaran pada SVM dalam menemukan titik-titik support vector, hanya bergantung pada dot product dari data yang sudah ditransformasikan pada ruang baru yang berdimensi lebih tinggi, yaitu $\Phi(x_i).\Phi(x_i)$.

Karena umumnya transformasi Φ ini tidak diketahui, dan sangat sulit untuk difahami secara mudah, maka perhitungan dot product tersebut sesuai teori Mercer dapat digantikan dengan fungsi kernel yang terlihat pada persamaan 2.15.

$$\mathrm{K}(x_i.x_j\Phi(x_i).\Phi(x_i) \qquad \text{Persamaan 2.15}$$

Beberapa kernel yang umum dipakai pada SVM adalah:

1. *Polynomial*

   Kernel tirck polynomial cocok digunakan untuk menyelesaikan masalah klasifikasi, dimana dataetpelatihan sudah normal. *Kernel trick* ini dinyatakan dalam persamaan.

   $$K\left(\overrightarrow{x_i},\overrightarrow{x_j}\right) = (\overrightarrow{x_i}.\overrightarrow{x_j} + 1)^p \qquad \text{Persamaan 2.16}$$

2. *Radial Basis Function* atau *Gaussian*

Kernel trick radial basis function atau *gaussian* merupakan kernel yang paling banyak digunakan untuk menyelesaikan masalah klasifikasi untuk dataset yang tidak terpisah secara linear, dikarenakan akurasi pelatihan dan akurasi prediksi yang sangat baik pada kernel ini, dimana *kernel radial basis function* dinyatakan dalam persamaan 2.17.

$$K(\overrightarrow{x_i}, \overrightarrow{x_j}) = exp\left(-\left|\left|\overrightarrow{x_i} - \overrightarrow{x_j}\right|\right|^2\right)\gamma \quad \text{Persamaan 2.17}$$

3. *Sigmoid*

Kernel sigmoid merupakan *kernel trick* svm yang merupakan pengembangan dari jaringan saraf tiruan, dimana kernel ini dinyatakan dengan persamaan 2.18.

$$K(\overrightarrow{x_i}, \overrightarrow{x_j}) = tanh(\overrightarrow{\alpha x_i}.\overrightarrow{x_j} + \beta) \quad \text{Persamaan 2.18}$$

Kernel trick memberikan beberapa kemudahan, karena dalam proses pembelajaran SVM, untuk menentukan *support vector*, pengguna hanya cukup mengetahui fungsi *kernel trick* yang dipakai, tanpa perlu mengetahui wujud dari fungsi *non-linier*.

Dari keseluruhan *kernel trick* tersebut, *kernel trick radial basis function* merupakan *kernel trick* yang memberikan hasil terbaik pada proses klasifikasi khususnya untuk data yang tidak bisa dipisahkan secara linear.

Selanjutnya hasil klasifikasi dari data $\vec{x}$ diperoleh dari persamaan berikut :

$$f(x) = \sum_{i=1, \overrightarrow{x_i} \in SV}^{n} \alpha_i \; y_i \; K(\overrightarrow{x_i}, \overrightarrow{x_j}) + b \quad Persamaan\ 2.19$$

2.7.5 *Multiclass SVM*

SVM saat pertama kali diperkenalkan oleh Vapnik, hanya dapat mengklasifikasikan data ke dalam dua kelas (klasifikasi biner). Namun, penelitian lebih lanjut untuk mengembangkan SVM sehingga bisa mengklasifikasi data yang memiliki lebih dari dua kelas, terus dilakukan. Ada dua pilihan untuk mengimplementasikan multi class SVM yaitu dengan menggabungkan beberapa SVM biner atau menggabungkan semua data yang terdiri dari beberapa kelas ke

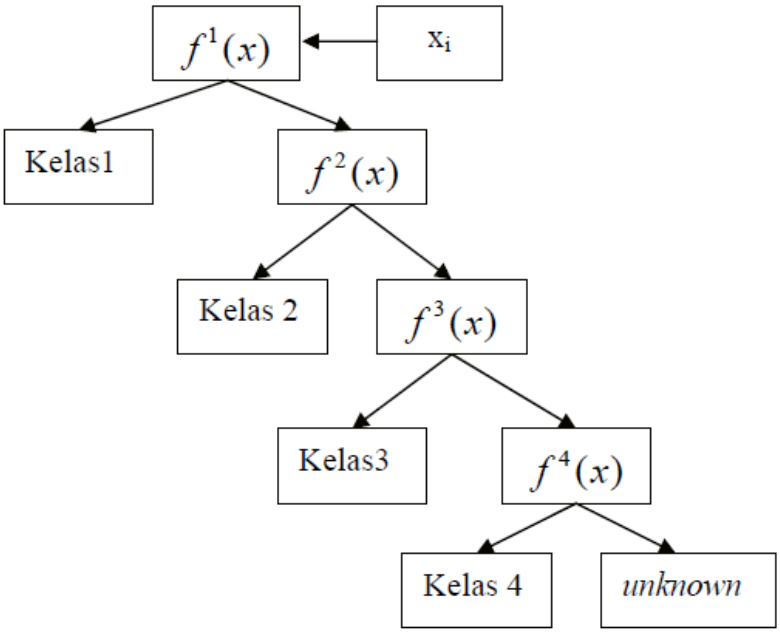
dalam sebuah bentuk permasalah optimasi. Namun, pada pendekatan yang kedua permasalahan optimasi yang harus diselesaikan jauh lebih rumit. Ada dua metode yang umum digunakan untuk mengimplementasikan multi class SVM dengan pendekatan yaitu metode "*one-against-all*" dan metode " *one-against-one*".

2.7.5.1 *One-Against-All*

Dengan menggunakan metode ini, dibangun k buah model SVM biner (k adalah jumlah kelas). Contohnya, terdapat permasalahan klasifikasi dengan 4 buah kelas. Untuk pelatihan digunakan 4 buah SVM biner seperti pada tabel 2.7 dan penggunannya dalam mengklasifikasi data baru dapat dilihat pada gambar 2.10.

Tabel 2.7 Contoh SVM biner dengan metode *One-against-all*

$Y_i = 1$	$Y_i = -1$	**Hipotesis**
Kelas 1	Bukan Kelas 1	$f_1(x) = (w_1)x + b_1$
Kelas 2	Bukan Kelas2	$f_2(x) = (w_2)x + b_2$
Kelas 3	Bukan Kelas 3	$f_3(x) = (w_3)x + b_3$
Kelas 4	Bukan Kelas 4	$f_4(x) = (w_4)x + b_4$



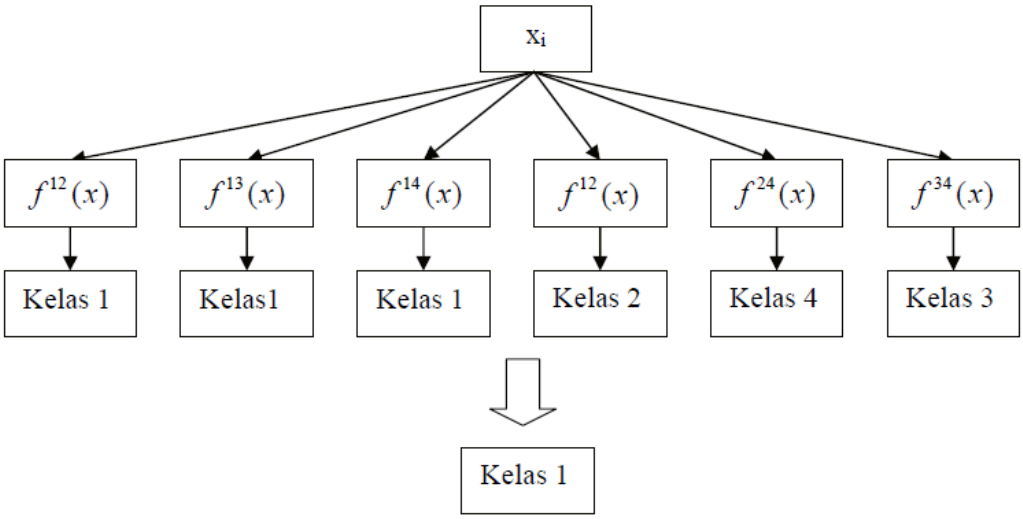
Gambar 2.10 Contoh klasifikasi dengan metode *One-against-all*

2.7.5.2 *One-Against-One*

Dengan menggunakan metode ini, dibangun $\left(\frac{k(k-1)}{2}\right)$buah model klasifikasi biner (k adalah jumlah kelas). Setiap model klasifikasi dilatih pada data dari dua kelas. Terdapat beberapa metode untuk melakukan pengujian setelah keseluruhan k(k−1)/ 2 model klasifikasi selesai dibangun. Salah satunya adalah metode voting.

Tabel 2.8 Contoh SVM biner dengan metode *One-against-one*

Y_i=1	Y_i=-1	Hipotesis
Kelas 1	Kelas 2	$f_{12}(x) = (w_{12})x + b_{12}$
Kelas 1	Kelas 3	$f_{13}(x) = (w_{13})x + b_{13}$
Kelas1	Kelas 4	$f_{14}(x) = (w_{14})x + b_{14}$
Kelas 2	Kelas 3	$f_{23}(x) = (w_{23})x + b_{23}$
Kelas 2	Kelas 4	$f_{24}(x) = (w_{24})x + b_{24}$
Kelas 3	Kelas 4	$f_{34}(x) = (w_{34})x + b_{34}$



Gambar 2.11 Contoh klasifikasi dengan metode *One-against-one*

2.8 Pemrograman Berorientasi Objek

Pemrograman berorientasi objek adalah metode implementasi dimana pemogram diorganisasaikan sebagai komunitas objek yang bekerja sama berinteraksi.Masing-masing merepresentasikan instan suatu kelas yang menyediakn sejumlah layanan yang relatif sederhana.Objek-objek saling berkomunikasi untuk mempertukarkan informasi.Prilaku kompleks diperolah dengan kerjasama antar objek-objek.Kelas-kelas dapat merupakan anggota hirarki yang disatukan lewat hubungan pewarisan.

Tiga aspek penting dalam pemrograman beriorientasi objek:

1. Menggunakan objek-objek bukan algoritma-algoritma sebagai blok-blok bangunnan logik dasar (hirarki "*part of*").
2. Masing-masing objek adalah instan suatu kelas.
3. Kelas-kelas saling berhubungan lewat keterhubungan perwarisan ("*is a*").

Stroustrup mendeskripsikan permograman beriorientasi objek sebagai penggabungan abstraksi data dengan pewarisan. Fitur-fitur penting untuk mendukung abstraksi data adalah *constructor* dan *destructor* untuk penciptaan dan pemusnahan objek, tipe berparameter, *mekanisme exception,object, coercions* dan *iteraror* . Defenisi tersebut mengarah ke bahasa yang mendukung entitas-

entitas independen yang disebut objek. Kelakukan objek harus didefenisikan tipe yang dapat didefenisiskan oleh program menggunakan bahasa.Tipe-tipe ini dapat dihubungkan melalui mekanisme pewarisan.

Citra/image yang telah tertanam mengenai pemrograman adalah membuat file teks kode sumber yang kemudian diberikan ke kompilator. Hal itu merupakan hanya salah satu bagian dari pemrograman.Tujuan nyata dari pemrogram adalah menciptakan kode yang benar (yaitu seluruhnya memenuhi kebutuhan), sementara kompilator hanya sanggup memeriksa sintaks dan membangkitkan kodek objek.

2.9 UML

Kegunaan diagram pada pemodelan adalah untuk formulisasi ekspresi model objek secara koheren, presisi dan mudah dirumuskan. Pemodelan berorientasi objek memerlukan kakas untuk mengekspresikan model. UML menyediakan sejumlah diagran untuk mengekspresikan pemodelan berorientasi objek yang dilakukan.

1. Use case diagram

Diagram use-case digunakan untuk mendeskripsikan apa yang seharusnya dilakukan oleh sistem. Diagram use-case menyediakan cara mendeskripsikan pandangan eksternal terhadap sistem dan interaksi –interaksinya dengan dunia luar. Dengan cara ini, diagram use-case menggantikan diagram konteks pada pendekatan konvensional.

Pemodelan ini biasa dilakukan lewat proses berulang interaksi antara pengembang dan pemakai untuk memperoleh spesifikasi kebutuhan yang sama-sama disepakati.

Pemodelan use-case diciptakan oleh Ivar Jacobson.

Tujuan utama pemodelan use-case adalah:

a. Memutuskan dan mendeskripsikan kebutuhan-kebutuhan fungsional sistem.
b. Memberikan deskripsi jelas dan konsisten dari apa yang seharusnya dilakukan,sehingga model use case digunakan

di seluruh proses pengembangan untuk komunikasi dan menyediakan basis untuk pemodelan berikutnya yang mengacu sistem harus memberikan fungsionalitas yang dimodelkan pada use-case.

c. Menyediakan basis untuk melakukan pengujian sistem yang memverifikasi sistem.Menguji apakah sistem telah memberikan fungsionalitas yang diminta.
d. Menyediakan kemampuan melacak kebutuhan fingsionalitas menjadi kelas-kelas dan operasi-operasi aktual di sistem.Untuk menyederhanakan perubahan dan ekstensi ke sistem dengan mengubah model *use-case* dan kemudian melacak *use-case* yang dipengaruhi ke perancangan dan implementasi sistem.

Elemen diagram use-case adalah:

a) Aktor

Aktor adalah pemakai sistem, dapat berupa manusia atau sistem terotomatisasi lain.Aktor adalah sesuatu atau seseorang yang berinteraksi dengan sistem, yaitu siapa atau apa yang menggunakan sistem.Yang dimaksud dengan berinteraksi adalah aktor mengirim atau menerima pesan ke atau dari sistem, atau mempertukarkan informasi dengan sistem. Dalam perspektif aktor, use-case melakukan sesuatu yang berharga bagi aktor.

Aktor adalah tipe (kelas), bukan instan. Aktor merepresentasikan peran bukan pemakai individu dari sistem.Aktor mempunyai nama. Nama yang dipilih seharusnya menyatakan peran aktor.

b) Use-Case

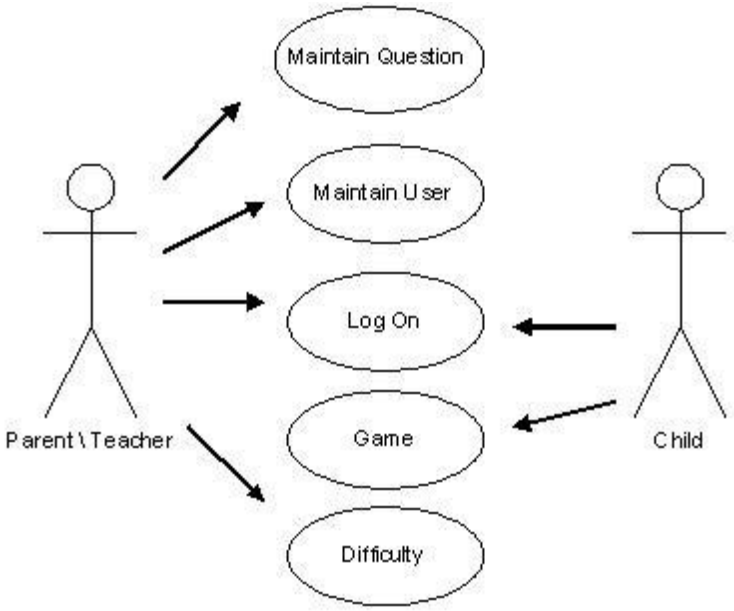
Use-case adalah cara spesifik penggunaan sistem oleh aktor.Ciri-ciri dari *use-case* adalah:

1. Pola perilaku yang harus dipenuhi oleh sistem.
2. Sekuen transaksi terhubung yang dilakukan aktor dan sistem.
3. Memberikan sesuatu yang berharga bagi aktor.

c) Hubungan antar *use-case*.

Keterhubungan antar use-case dengan user-case lain berupa generalisasi antar use-case, yaitu:

1. *Include*, perilaku use-case merupakan bagian dari use-case yang lain.
2. *Extend*, perilaku use-case memperluas perilaku use-case yang lain.



Gambar 2.12 Contoh *Use Case*

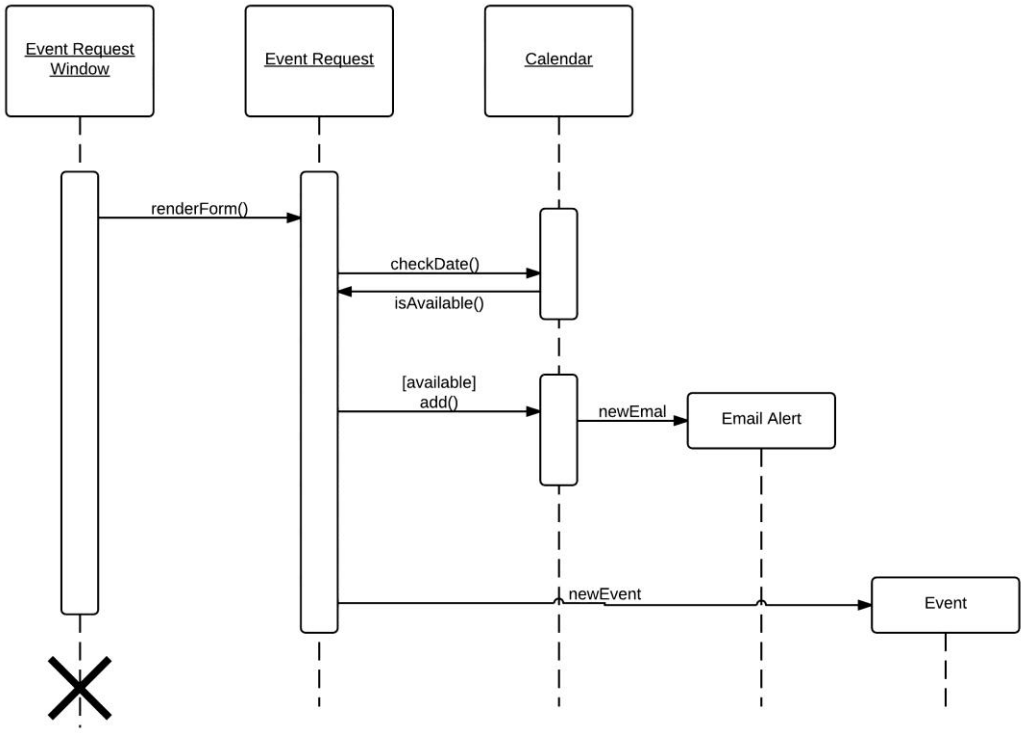
2. *Sequence Diagram*

Diagram sekuen mendeskripsikan komunikasi di antara objek-objek , meliputi pesan-pesan yang ada dan urutan pesan tersebut muncul.

Diagram sekuen digunakan untuk:

1. *Overview* perilaku sistem.
2. Menunjukkan objek-objek yang diperlukan.
3. Mendokumentasikan skenario dari suatu diagram use case

4. Memeriksa jalur-jalur pengaksesan



Gambar 2.13 Contoh *Sequence Diagram*

3. *Colloboration Diagram*

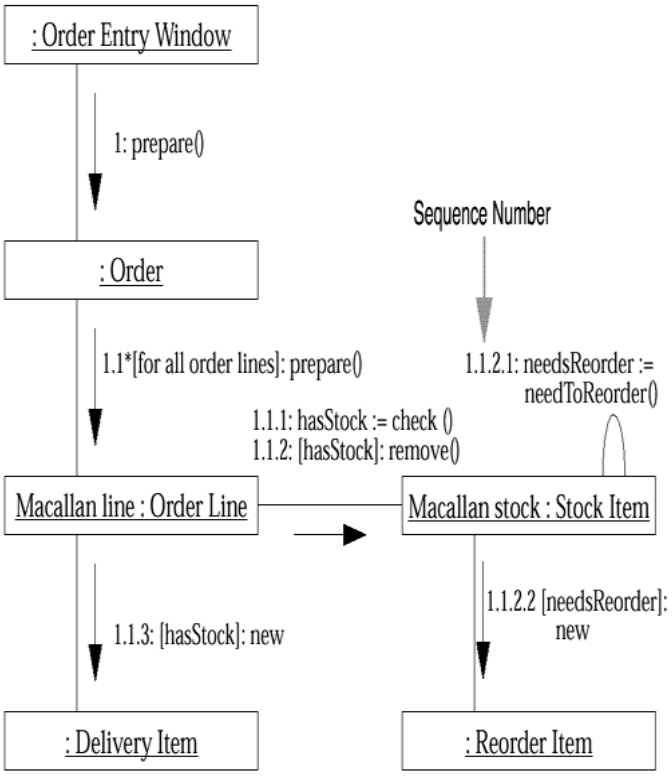
Diagram kolaborasi mendefinisikan peran-peran yang dimainkan ketika satu tugas dilakukan.Peran-peran dimainkan oleh instan-instan yang berinteraksi. Diagram kolaborasi menyatakan komunikasi di antara objek-objek yang menunjukkan pesan-pesan yang ada, urutan pesan dan hubungan antar objek-objek.

Diagram kolaborasi digunakan untuk:

1. Pandangan dalam dari perilaku sistem, berfokus pada link-link di antara objek.
2. Ilustrasi dari suatu diagram use-case.
3. Menyatakan objek-objek yang diperlukan untuk merealisasikan suatu layanan.
4. Memeriksa jalur-jalur pengaksesan.

Kriteria diagram interaksi yang bagus adalah:

1. Fokus pada mengkomunikasi satu aspek dinamisasi sistem.
2. Hanya berisi elemen-elemen yang secara esensi memahamai aspek ini.
3. Menyediakan konsistensi rinci dengan level abstraksinya dan seharusnya hanya mengemukakan hal-hal esensi yang perlu dipahami.
4. Bukan minimalis sehingga informasi ke pembaca mengenai simantiks yang penting tidak kurang.

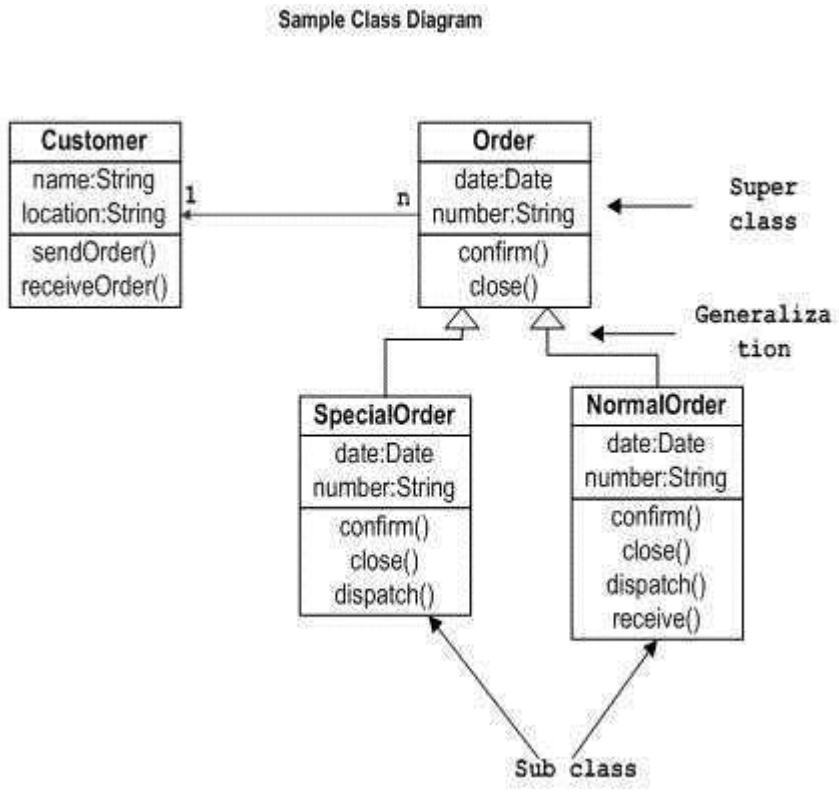


Gambar 2.14 Contoh *Colloboration Diagram*

4. *Class Diagram*

Diagram kelas merupakan diagram paling umum dipakai di semua pemodelan berorientasi objek.Pemodelan kelas merupakan pemodelan paling utama di pendekatan berorientasi objek.Pemodelan kelas menunjukkan kelas-

kelas yang ada di sistem dan hubungan antar kelas-kelas itu, atribut-atribut dan operasi-operasi di kelas-kelas.



Gambar 2.15 Contoh *Class Diagram*

2.10 Java Netbeans

NetBeans adalah suatu tool untuk membuat program dengan menggunakan bahasa pemrograman Java berbasis grafis. Cara membuat program dengan merancang tampilan menggunakan komponent visual dan proses diletakan pada event driven. Netbeans menyediakan sekumpulan perangkat lunak modular yang disebut modul yang dipakai untuk membangun suatu aplikasi. Sebuah modul adalah merupakan arsip Java (Java Archive) yang memuat kelas-kelas Java yang berinteraksi dengan NetBeans Open API . Kemudian rancangan tampilan yang dibuat menggunakan NetBens programnya secara otomatis akan digenerate menjadi kode.

NetBeans memilki IDE (*Integrated Development Environment*), ada juga yang mengatakan *Integrated Design Environment* dan *Integreted Debugging Environment* yakni sebuah program atau alat bantu yang terdiri atas *Editor, Compiler, Debuger* dan *Design* yang terintegrasi dalam satu aplikasi.