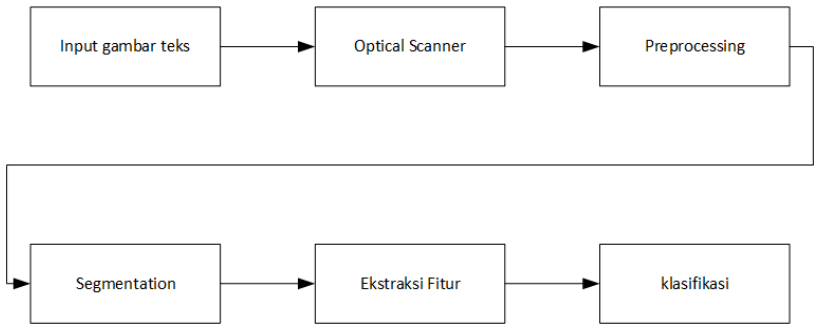
BAB 2
TINJAUAN PUSTAKA

2.1 *Optical Character Recognition*

OCR adalah singkatan dari *Optical Character Recognition*. Teknologi ini memungkinkan untuk mengenali karakter secara otomatis melalui mekanisme optik. Dalam kasus manusia, mata manusia adalah mekanisme optik. Gambar yang dilihat oleh mata adalah input untuk otak. Kemampuan untuk memahami input ini bervariasi pada setiap orang menurut banyak faktor. OCR adalah teknologi yang berfungsi seperti kemampuan membaca manusia. Meskipun OCR tidak mampu bersaing dengan kemampuan membaca manusia [8].

OCR adalah teknologi yang memungkinkan mengonversi berbagai jenis dokumen seperti dokumen kertas yang dipindai, file PDF atau gambar yang diambil oleh kamera digital menjadi data yang dapat diedit dan dicari. Gambar yang diambil oleh kamera digital berbeda dari dokumen atau gambar yang dipindai.

Alur dari sistem OCR biasanya terdapat input, proses, dan output. Untuk input biasanya berupa dokumen digital yang didalamnya terdapat unsur-unsur karakter. Selanjutnya masuk kedalam proses, untuk proses biasanya terdapat preprocessing, ekstraksi fitur, dan masuk pada tahap klasifikasi. Klasifikasi disini adalah untuk menandakan bahwa objek tersebut masuk kedalam salah satu kelas yang telah ditentukan. Hasil keluaran dari klasifikasi adalah huruf yang sesuai dengan ciri dari kelas tersebut. Berikut adalah diagram blok dari sistem OCR.



Gambar 2.1 Diagram Blok OCR

2.2 *Preprocessing*

Perancangan sistem *preprocessing* menjelaskan alur kerja dalam OCR dalam menyiapkan citra sebagai data input yang siap diolah lebih lanjut oleh sistem. Sistem *preprocessing* yang digunakan pada penelitian ini adalah sebagai berikut:

2.2.1 *Grayscale*

Citra *grayscale* adalah citra yang hanya memiliki 1 buah kanal sehingga yang ditampilkan hanyalah nilai intensitas atau dikenal juga dengan istilah derajat keabuan. Karena jenis citra ini hanya memiliki 1 kanal saja, maka citra *grayscale* memiliki tempat penyimpanan yang lebih hemat. Jenis ini disebut juga sebagai 8-bit image karena untuk setiap nilai pikselnya memerlukan penyimpanan sebesar 8-bit [9]. Rumus yang digunakan untuk mengubah citra RGB menjadi citra *grayscale* yaitu "*luma*" atau "*luminance*". Berikut adalah rumus untuk mengubah citra menjadi *grayscale*:

$$Gray = 0.299 * red + 0.587 * green + 0.114 * blue \quad (2.1)$$

2.2.2 *Thresholding*

Citra biner atau citra hitam putih (*black and white image*) adalah citra yang hanya memiliki 2 kemungkinan nilai untuk setiap pikselnya, yaitu

0 atau 1. Nilai 0 akan tampil sebagai warna hitam sedangkan nilai 1 akan tampil sebagai warna putih. Maka dari itu, jenis citra ini hanya membutuhkan 1-bit untuk menyimpan setiap nilai pada setiap pikselnya. Jenis citra ini sering digunakan untuk proses *masking* ataupun proses segmentasi citra [9]. Proses *threshodling* digunakan untuk mengekstrak *foreground* (tinta) dari *background* (kertas) dan mengubah menjadi citra biner. Proses *thresholding* mengubah warna gambar menjadi citra biner (*binary image*) dimana ditentukan sebuah nilai level *threshold* kemudian piksel yang memiliki nilai level dibawah level *threshold* di set menjadi warna putih (1 pada nilai biner) dan nilai diatas nilai *threshold* di set menjadi warna hitam (1 pada nilai biner) [10]. Berikut adalah algoritma biner yang digunakan:

$$g(x,y) = \begin{cases} 1 \; if \; f(x,y) < T \\ 0 \; if \; f(x,y) > T \end{cases} \tag{2.2}$$

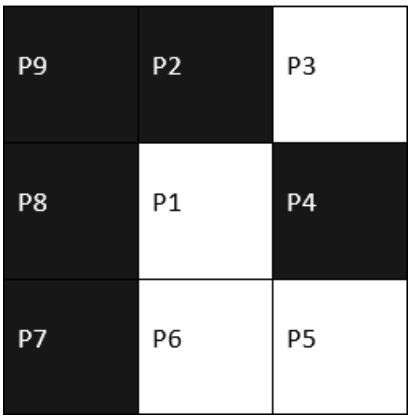
Keterangan:
Gray merupakan nilai hasil *grayscale* dan 128 adalah nilai level threshold.

2.2.3 *Thinning*

Proses *thinning* atau penipisan bertujuan unuk mengurangi suatu objek dari citra biner menjadi ukuran yang minimum dan direduksi menjadi rangka (skeleton). Metode yang digunakan dalam proses *thinning* yaitu menggunakan metode *Zhang Suen*. *Zhang Suen* adalah metode paralel yang terdiri dari dua sub-iterasi. Iterasi pertama berfungsi untuk menghapus titik batas selatan-timur dan verteks barat laut sementara yang lain berfungsi untuk menghapus titik batas barat laut dan verteks tenggara [11]. *Zhang Suen* memiliki beberapa tahapan dalam memproses citra biner:

1. Pertama-tama yaitu citra merupakan hasil dari *thresholding* atau citra berupa citra biner yang hanya memiliki 2 nilai saja yaitu piksel hitam dan piksel putih, dimana piksel hitam untuk latar belakang dan piksel

putih sebagai latar depan dan dengan menggunakan aturan delapan tetangga. Berikut ini adalah gambar aturan delapan tetangga:



Gambar 2.2 Aturan Delapan Tetangga

Dimana P1 adalah pusat piksel dan piksel lainnya adalah tetangga piksel P1. Kemudian nilai 0 untuk latar belakang (hitam) dan nilai 1 untuk latar depan (putih). Jadi bisa dilihat bahwa nilai P1, P3, P5, P6 bernilai 1 dan P2, P4, P7, P8, P9 bernilai 0. Algoritma ini diterapkan pada foreground (piksel putih).

2. N(P1) adalah jumlah tetangga P1 yang memiliki latar depan (1) dengan menggunakan aturan delapan tetangga. Kondisi pertama yaitu $2 \leq N(P1) \leq 6$. Hitung N(P1), jumlah tetangga P1 yang memiliki latar depan (1) dengan menggunakan aturan delapan tetangga. Nilai N(P1) yang mempunyai nilai 1 adalah P3, P5, P6. Nilai N(P1) = 3 maka bernilai *true*.
3. S(P1) adalah jumlah perpindahan dari latar belakang (0) ke latar depan (1) antara 2 piksel, lakukan searah jarum jam. Berikut ini merupakan contoh perpindahan dari latar belakang ke latar depan dengan menggunakan gambar diatas:
   P2 – P3, P3 – P4, P4 – P5, P5 – P6, P6 – P7, P7 – P8, P8 – P9, P9 – P2
   Berikut adalah hasil S(P1) dari gambar diatas:
   P2 – P3 = 0 ke 1 (*true*)

P3 – P4 = 1 ke 0

P4 – P5 = 0 ke 1 (*true*)

P5 – P6 = 1 ke 1

P6 – P7 = 1 ke 0

P7 – P8 = 0 ke 0

P8 – P9 = 0 ke 0

P9 – P2 = 0 ke 0

Jadi, nilai S(P1) adalah 2, bernilai false

4. Langkah selanjutnya yaitu P2 * P4 * P6 = 0, berikut adalah hasil dengan menggunakan gambar 2.2 diatas:
   P2 (0) * P4 (0) * P6 (1) = 0, bernilai *true*
5. Kemudian langkah selanjutnya yaitu P4 * P6 * P8 = 0, berikut adalah hasil dengan menggunakan gambar 2.2 diatas:
   P4 (0) * P6 (1) * P8 (0) = 0, bernilai *true*
6. Setelah tahapan sub iterasi pertama maka tandai piksel tersebut yang nantinya akan dihapus atau diubah nilainya menjadi 0. Ulangi langkah diatas hingga piksel terakhir. Setelah hingga piksel terakhir maka hapus atau ubah piksel yang telah ditandai menjadi 0.
7. Proses sub iterasi kedua sebenarnya hampir sama hanya saja yang membedakannya yaitu ada pada langkah ke-3 dan ke-4. Langkah ke-3 pada sub iterasi kedua yaitu P2 * P4 * P8 = 0, dan langkah ke-4 pada sub iterasi kedua yaitu P2 * P6 * P8 = 0.
8. Sama halnya seperti pada sub iterasi pertama, apabila 4 kondisi bernilai *true* maka tandai piksel tersebut yang nantinya akan dihapus atau diubah nilainya menjadi 0. Ulangi langkah tersebut hingga piksel terakhir. Setelah hingga piksel terakhir maka hapus atau ubah piksel yang telah ditandai menjadi 0.
9. Berikut ini adalah langkah-langkah singkat dari algoritma *Zhang Suen*:
   a. $2 \leq N(P1) \leq 6$
   b. $S(P1) = 1$
   c. P2 * P4 * P6 = 0

d. P4 * P6 * P8 = 0
e. $2 \leq N(P1) \leq 6$
f. S(P1) = 1
g. P2 * P4 * P8 = 0
h. P2 * P6 * P8 = 0

2.2.4 Segmentasi

Proses segmentasi bertujuan untuk membagi citra ke dalam basis elemen sesuai dengan kriteria yang ditentukan. Sehingga citra yang diambil adalah citra yang penti saja. Metode yang digunakan dalam proses segmentasi yaitu metode *Connected Component Labeling*. Connected Component Labeling adalah salah satu operasi paling umum di hampir semua aplikasi pengolahan gambar. Dalam divisi mesin, kebanyakan objek memiliki permukaan. Titik-titik dalam komponen yang terhubung membentuk wilayah kandidat untuk mewakili suatu objek. Objek gambar, yang merupakan komponen, dipisahkan dari gambar latar belakang pada gambar biner [12]. Metode ini memiliki sifat ketetanggaan, ketetanggaan harus memiliki panjang atau jarak 1 unit (langsung antara piksel dengan piksel tanpa ada perantaranya). Ada 2 jenis konektivitas yang digunakan pada citra 2 dimensi [13], yaitu:

1. 4-konektivitas (*4-connected neighbors*)

   Piksel-piksel yang berdekatan dikatakan memiliki hubungan 4-konektivitas jika piksel-piksel tersebut terletak berdampingan secara horizontal dan vertikal N4(P). Kumpulan dari piksel-piksel ini disebut dengan *4 neighbors of P*. Pada konsep 4-connected neighbors bila terdapat 2 piksel bersinggungan secara diagonal maka akan dianggap 2 objek.

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

Gambar 2.3 4 Conected Neighbors

2. 8-konektivitas (*8-connected neighbors*)

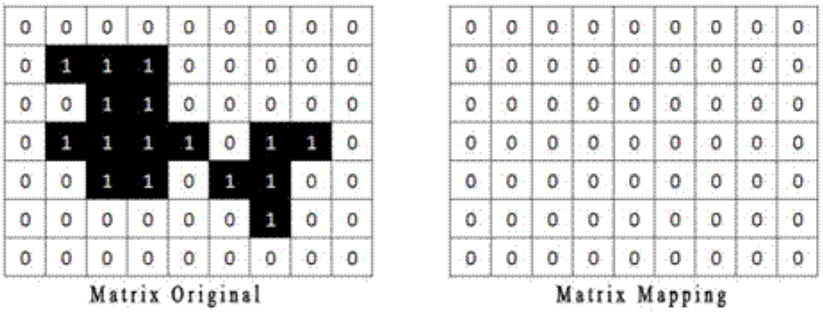
   Piksel-piksel yang berdekatan dikatakan memiliki hubungan 8-konektivitas jika piksel-piksel tersebut terletak berdampingan secara horizontal dan vertikal N8(P) atau disebut juga empat diagonal neighbors. Pada konsep 8-connected neighbors bila terdapat 2 piksel yang bersinggungan baik secara diagonal maupun secara horizontal dan vertikal maka akan dianggap 1 objek.

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 2.4 8 Connected Neighbors

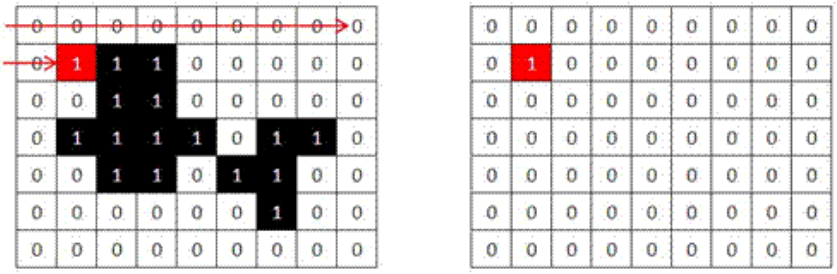
Berikut adalah langkah-langkah dalam metode Connected Component Labeling:

1. Buat dua buah matrix, matrix pertama merepresentasikan objek gambar atau citra biner yang akan diolah dan matrix kedua merupakan matrix tempat meletakkan piksel-piksel terpilih yang disebut dengan matrix mapping.



Gambar 2.5 Matrix Original dan Matrix Mapping

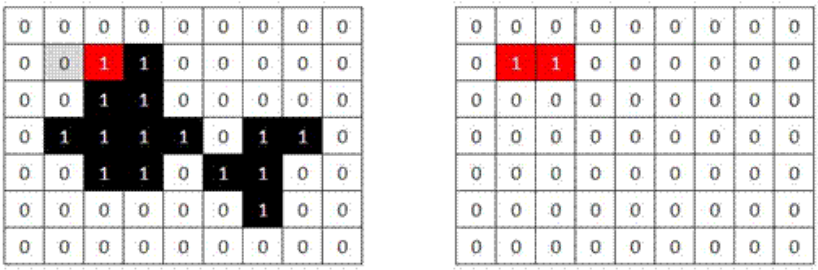
2. Lakukan scanning piksel-piksel terhadap citra foreground mulai dari sisi atas matriks yaitu dari kiri ke kanan dan dari atas ke bawah.



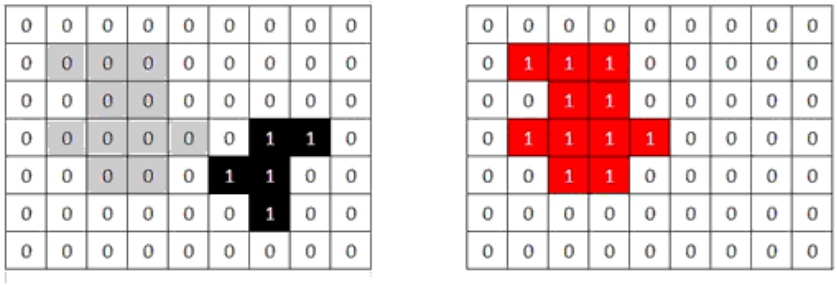
Gambar 2.6 Connected Component Langkah Pertama

Jika ditemukan piksel foreground, maka lakukan pelabelan terhadap piksel tersebut, sebagai contoh label 1 dan pindahkan piksel pertama ke matrix mapping. Selanjutnya ubah nilai piksel pertama pada matriks original yang telah ditemukan tadi dengan nilai 0.

3. Lakukan pendekatan 4-connected neighbors secara berulang-ulang terhadap piksel-piksel yang memiliki kedekatan dan kesamaan nilai intensitas sampai tidak ada lagi kedekatan secara 4-connected neighbors antara piksel-piksel yang telah dilabeli dengan label 1.



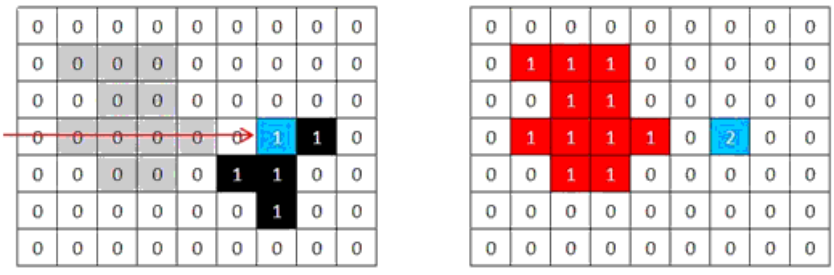
Gambar 2.7 Langkah pertama pemindai nilai



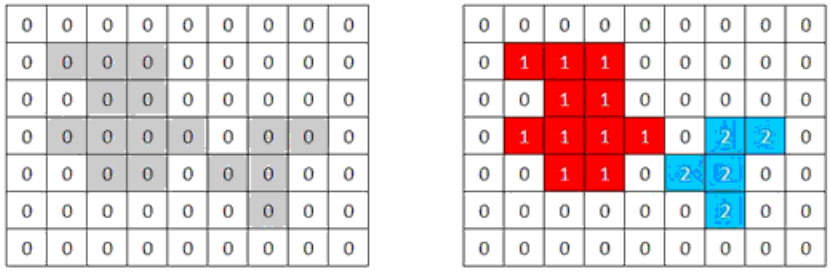
Gambar 2.8 Langkah terakhir pemindahan nilai

Jika sudah tidak ditemukan lagi piksel-piksel tetangga yang terdapat kedekatan secara 4-connected neighbors, maka lakukan proses merging pada matrix mapping.

4. Lakukan lagi scanning terhadap citra foreground untuk mendapatkan objek yang lain, kemudian ulang hingga tidak ada lagi objek yang di *scanning*. Jika sudah selesai maka selesai proses *scanning*, *labeling* dan *merging*.



Gambar 2.9 Scanning objek lain



Gambar 2.10 Scanning telah selesai

2.2.5 *Scaling*

Scaling dilakukan untuk menyamakan setiap ukuran citra karakter hasil segmentasi karena tidak semua citra mempunyai tinggi dan lebar yang sama. Metode yang digunakan adalah bukan metode khusus melainkan menggunakan perbandingan ukuran dari setiap citra hasil segmentasi dengan ukuran citra yang diinginkan.

Berikut ini adalah rumus yang digunakan dalam proses *scaling*:

Menghitung nilai koordinat baris:

$$x = \frac{pb * pp}{pa} \tag{2.3}$$

Keterangan:

x = Nilai piksel baris baru

pb = Ukuran panjang matriks baru

pp = Posisi piksel baris

pa = Ukuran panjang matriks lama

Menghitung nilai koordinat kolom:

$$y = \frac{lb * pp}{la} \tag{2.4}$$

Keterangan:

y = Nilai piksel kolom baru

lb = Ukuran lebar matriks baru

lp = Posisi piksel kolom

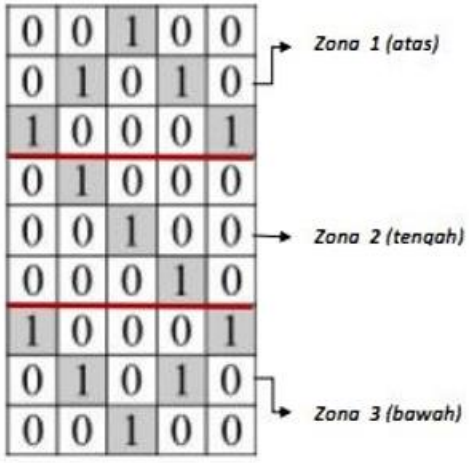
la = Ukuran lebar matriks lama

2.3 Ekstraksi Ciri

Ekstraksi ciri adalah proses pengukuran terhadap data yang telah di normalisasi untuk membentuk sebuah nilai fitur. Nilai fitur digunakan oleh pengklasifikasi untuk mengenali unit masukan dengan unit target keluaran dan memudahkan pengklasifikasian karena nilai ini mudah untuk dibedakan [14]. Pada penelitian ini, metode yang digunakan untuk ekstraksi ciri adalah metode *zoning*.

2.3.1 *Zone Based Feature Extraction*

Zoning adalah salah satu ekstraksi fitur yang paling popular dan sederhana untuk diimplementasikan [14]. Setiap citra dibagi menjadi N x M zona dan dari setiap zona tersebut dihitung nilai fitur sehingga didapatkan fitur dengan panjang N x M. Salah satu cara menghitung nilai fitur setiap zona adalah dengan menghitung jumlah piksel hitam setiap zona dan membaginya dengan jumlah piksel hitam terbanyak pada yang terdapat pada salah satu zona. Contoh pembagian 3 zona pada citra biner dapat dilihat pada gambar 2.11.



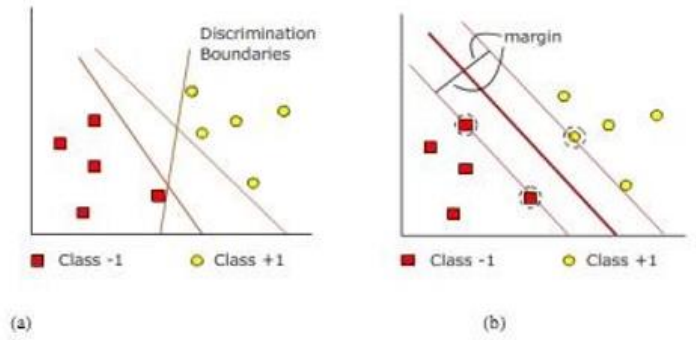
Gambar 2.11 Pembagian Zona pada Citra Biner

Metode ekstraksi fitur berbasis zona memberikan hasil yang baik bahkan ketika langkah sebelum proses tertentu dimulai seperti filtering, smoothing, dan menghapus zona yang tidak dianggap. Konsep metode ekstraksi ciri yang digunakan untuk mengekstraksi fitur untuk klasifikasi yang efisien dan pengenalan yaitu [14]:

1. Hitung *centroid* dari citra.
2. Bagi ke dalam *n* buah zona yang sama besar proporsinya.
3. Hitung jarak antara titik *centroid* dengan koordinat *pixel* yang memiliki nilai.
4. Ulangi langkah 3 untuk *pixel* yang ada di semua zona.
5. Hitung rata-rata dari jarak yang telah didapat pada langkah 3.
6. Ulangi langkah 5 hingga didapat masing-masing rata-rata jarak dari setiap zona.
7. Akhirnya *n* buah fitur akan didapat untuk melakukan klasifikasi dan pengenalan

2.4 *Support Vector Machine* (SVM)

Support Vector Machine (SVM) pertama kali diperkenalkan oleh Valdimir Vanpik pada tahun 1992 sebagai rangkaian harmonisasi konsep-konsep unggulan dalam bidang pengenalan pola [15]. *Support Vector Machine* merupakan metode pembelajaran yang digunakan untuk klasifikasi biner, ide dasarnya adalah mencari hyperplane terbaik yang berfungsi sebagai pemisah dua *class* pada *input space* [15].



Gambar 2.12 SVM Berusaha Menemukan *Hyperplane* Terbaik yang Memisahkan Dua Kelas -1 dan +1

Berbeda dengan pendekatan *neural network* yang berusaha mencari *hyperplane* pemisah antar kelas, SVM mencoba mencari dan memisah *hyperplane* yang terbaik yang berada pada dua kelas, gambar (a) menunjukan pola-pola yang merupakan anggota dari dua buah kelas +1 dan -1, pola yang berada pada anggota -1 kemudian disimbolkan dengan warna merah, sedangkan pola pada +1 warna kuning. Masalah pada klasifikasi dapat dilakukan dengan usaha menemukan garis (*hyperplane*) yang memisahkan kelompok tersebut. Pendefinisian persamaan suatu *hyperplane* pemisah yang dituliskan dengan:

$$w * x_i + b = 0 \quad (2.5)$$

Data x_i yang terbagi dalam dua kelas, yang termasuk kelas -1 (sample negatif) didefinisikan sebagai vektor yang memenuhi pertidaksamaan 2.6. Sedangkan yang termasuk kelas +1 (sample positif) memenuhi persamaan 2.7. Berikut adalah persamaan 2.6 dan 2.7 dibawah ini:

$$w * x_i + b < 0 \text{ untuk } y_i = -1 \quad (2.6)$$

$$w * x_i + b > 0 \text{ untuk } y_i = +1 \quad (2.7)$$

Ket:

x_i = data input

y_i = label yang diberikan

w = nilai dari bidang normal

b = posisi bidang relatif terhadap pusat koordinat

Parameter w dan b adalah parameter yang akan dicari nilainya, bila label y_i = -1, maka pembatas menjadi persamaan 2.8. Apabila label data y_i = +1, maka pembatas menjadi persamaan 2.9. Berikut adalah persamaan 2.8 dan 2.9 dibawah ini:

$$w * x_i + b \leq -1 \quad (2.8)$$

$$w * x_i + b \geq +1 \tag{2.9}$$

Margin terbesar dapat dicari dengan cara memaksimalkan jarak antara bidang pembatas kedua kelas dan titik terdekatnya, yaitu $^{2}/_{|w|}$. Hal ini dirumuskan sebagai permasalahan *Quadratic Programming* (QP) *problem* yaitu mecari titik minimal persamaan 2.10 dengan memperhatikan persamaan 2.11 Berikut:

$$min\tau(w) = \frac{1}{2}\|w\|^2 \tag{2.10}$$

$$y_i(w * x_i + b) - 1 \geq 0, (i = 1, \dots, n) \tag{2.11}$$

Permasalahan ini dapat dipecahkan dengan berbagai teknik komputasi. Lebih mudah diselesaikan dengan mengubah persamaan 2.10 Kedalam fungsi *lagrangian* pada persamaan 2.12 dan menyederhanakannya menjadi persamaan 2.14 berikut:

$$L(w, b, a) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} a_i \left(y_i\big((w^T x_i + b) - 1\big)\right) \tag{2.12}$$

$$L(w, b, a) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} a_i y_i (w^T x_i + b) + \sum_{i=1}^{n} a_i \tag{2.13}$$

Dimana a_i adalah *lagrange multiplier* yang bernilai nol atau positif ($a_i > 0$). Nilai optimal dari persamaan 2.13 dapat dihitung dengan meminimalkan *L* terhadap *w*, *b*, dan *a*. Dapat dilihat pada persamaan 2.14 Sampai 2.16 Berikut:

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{n} a_i y_i x_i = 0 \tag{2.14}$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{n} a_i y_i = 0 \tag{2.15}$$

$$\frac{\partial L}{\partial a} = \sum_{i=1}^{n} a_i y_i (w^T x_i + b) - \sum_{i=1}^{n} a_i = 0 \quad (2.16)$$

Maka masalah *lagrange* untuk klasifikasi dapat dinyatakan pada persamaan 2.17 berikut:

$$Min\ L(w, b, a) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} a_i y_i (w^T x_i + b) - \sum_{i=1}^{n} a_i \quad (2.17)$$

Dengan memperhatikan persamaan 2.18 dan 2.19 berikut:

$$w - \sum_{i=1}^{n} a_i y_i x_i = 0 \quad (2.18)$$

$$\sum_{i=1}^{n} a_i y_i = 0 \quad (2.19)$$

Model persamaan 2.19 diatas merupakan *primal lagrange*. Sedangkan dengan memaksimalkan L terhadap a_i, persamaannya menjadi persamaan 2.20 berikut:

$$Max \sum_{i=1}^{n} a_i - \frac{1}{2} \sum_{i=1,j=1}^{n} a_i a_j y_i y_j{}^T x_i x_j^T \quad (2.20)$$

Dengan memperhatikan persamaan 2.21 berikut:

$$\sum_{i=1}^{n} a_i y_i = 0, a_i \geq 0 (i, j = 1, \dots, n) \quad (2.21)$$

Untuk mencari nilai x_i dan y_i dapat dilakukan ketika sudah didapatkan nilai tiap karakter dari pembobotan tf-idf dan inisialisasi kelas. Hasil dari pembobotan tf-idf diubah kedalam bentuk format data SVM, sedangkan data kelas menjadi label.

Untuk mendapatkan nilai a_i, langkah pertama adalah mengubah setiap abstrak menjadi nilai vektor (*support vector*) = $\left(\frac{x}{y}\right)$, kemudian nilai vektor dari setiap karakter dimasukan ke persamaan 2.20 kernel *trick phi* berikut:

$$\emptyset \begin{bmatrix} x \\ y \end{bmatrix} = \begin{cases} \sqrt{x_n^2 + y_n^2} > 2 \; maka \; \begin{bmatrix} \sqrt{x_n^2 + y_n^2} - x + |x - y| \\ \sqrt{x_n^2 + y_n^2} - y + |x - y| \end{bmatrix} \\ \sqrt{x_n^2 + y_n^2} \leq 2 \; maka \; \begin{bmatrix} x \\ y \end{bmatrix} \end{cases} \quad (2.22)$$

Nilai *x* didapatkan dari persamaan 2.23 kernel linier untuk *x* berikut:

$$\sum_{i=1,j=1}^{n} x_i x_j^T \, , (i, j = 1, \dots, n) \quad (2.23)$$

Nilai *y* didapatkan dari persamaan 2.24 kernel linier untuk *y* berikut:

$$\sum_{i=1,j=1}^{n} y_i y_j^T \, , (i, j = 1, \dots, n) \quad (2.24)$$

Untuk mendapatkan jarak tegak lurus yang optimal dengan mempertimbangkan vektor positif, maka hasil perhitungan drai substitusi nilai *x* dan nilai *y* ke persamaan 2.24 diberi nilai bias = 1. Kemudian cari parameter a_i, dengan terlebih dahulu mencari nilai fungsi setiap abstrak menggunaan persamaan 2.25, lalu mencari nilai a_i, pada persamaan linear menggunakan persamaan 2.26 dengan memperhatikan *i*,*j* = 1,...,*n* berikut:

$$\sum_{i=1,j=1}^{n} a_i S_i^T S_j \quad (2.25)$$

$$\sum_{i=1,j=1}^{n} a_i S_i^T S_j = y_i \quad (2.26)$$

Setelah parameter a_i didapatkan kemudian dimasukkan ke persamaan 2.27 berikut:

$$\hat{\mathrm{W}} = \sum_{i=1}^{n} a_i S_i \quad (2.27)$$

Hasil yang didapatkan menggunakan persamaan 2.28, selanjutnya digunakan persamaan 2.29, untuk mendapatkan nilai *w* dan *b*:

$$y = wx + b \tag{2.28}$$

Sedemikian sehingga didapatkanlah nilai *w* dan *b* atau nilai *hyperplane* untuk mengklasifikasikan kedua kelas.

Sebuah fungsi bisa menjadi fungsi kernel jika memenuhi *Teorema mercer*, yang menyatakan bahwa matriks kernel yang dihasilkan harus bersifat *positive* semi *definite* [16]. Berikut ini adalah beberapa fungsi kernel yang umum digunakan yaitu:

a. Kernel linear

$$K(x_i, x) = {x_i}^T x \tag{2.29}$$

b. Polynomial

$$K(x_i, x) = (\gamma . {x_i}^T x + r)^p, \gamma > 0 \tag{2.30}$$

c. Radial basis Function

$$K(x_i, x) = exp(-\gamma \|x_i - x_2\|^2), \gamma > 0 \tag{2.31}$$

d. Sigmoid kernel

$$K(x_i, x) = \tanh(\gamma {x_i}^T + r) \tag{2.32}$$

Hyperplane yang menjadi pemisah terbaik dapat ditemukan dengan mengukur margin *hyperplane* dan mencari titik maksimalnya. *Margin* adalah jarak antara *hyperplane* dengan *pattern* terdekat dari masing-masing kelas. *Pattern* yang paling dekat tersebut disebut *Support Vector*. Untuk pembahasan suatu kasus yang dapat dipisahkan secara linier, maka dalam hal ini fungsi pemisah yang dicari adalah fungsi linier. Fungsi tersebut dapat didefinisikan sebagai:

$$g(x) \coloneqq sqn(f(x)) \tag{2.33}$$

dengan

$$f(x) = WT\ X + b \tag{2.34}$$

Masalah klasifikasi ini dapat dirumuskan berikut: kita akan menemukan set parameter (*w,b*) sehingga *f*(*x*) = < *w,x*> + *b* = *yi*, untuk semua *i*. Dalam teknik ini kita berusaha menemukan fungsi pemisah (*classfiier*/*hyperplane*) terbaik diantara fungsi yang tidak terbatas jumlahnya untuk memisahkan dua macam objek. *Hyperplane* terbaik adalah *hyperplane* yang terletak ditengah-tengah antara dua set objek dari dua kelas. Mencari *hyperplane* terbaik ekuivalen dengan memaksimalkan margin atau jarak antara dua set objek dari kelas yang berbeda. Jika WX_1 + b = +1 adalah *hyperplane* pendukung dari kelas +1 (WX_1 + b = +1) dan WX_2 + b = -1 *hyperplane* pendukung dari kelas -1 (WX_2 + b = -1), margin antara dua kelas dapat dihitung dengan mencari jarak antara kedua *hyperplane-hyperplane* pendukung dari kedua kelas. Secara spesifik *margin* dihitung dengan cara berikut:

$$(WX_1 + b = +1) - (WX_2 + b = -1) = W(X_1 - X_2) = 2 = > \tag{2.35}$$

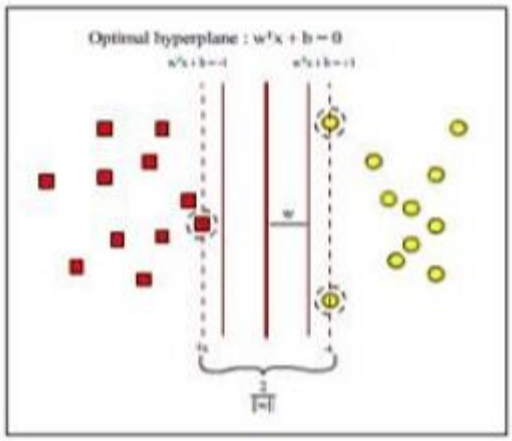
$$LD = \sum_{i=1}^{n} a_i - \frac{1}{2}\sum_{i,j} a_i a_j y_i y_j x_i x_j \tag{2.36}$$

$$syarat\ 1\text{: } \sum_{i=1}^{n} a_i y_i = 0$$

$$syarat\ 2\text{: } a_i \geq 0, i = 1,2, \dots, N$$

xi.xj merupakan *dot product* dua data dalam data latih. *Hyperplane* (batas keputusan atau pemisah).

Gambar 2.13 menunjukan bagaimana SVM bekerja untuk menentukan suatu fungsi pemisah dengan margin yang maksimal [15].



Gambar 2.13 Mencari Fungsi Pemisah Yang Optimal Untuk Objek Yang Dapat Dipisahkan Secara Linier

Untuk membuktikan bahwa memaksimalkan margin antara dua set objek akan meningkatkan probabilitas pengelompokkan secara benar dari data testing. Pada dasarnya jumlah fungsi pemisah tidak terbatas jumlahnya, misalnya dari jumlah yang tak terbatas kita ambil dua fungsi, yaitu $f_1(x)$ dan $f_2(x)$. Fungsi f_1 memiliki margin yang lebih besar daripada f_2. Setelah menemukan dua fungsi ini, sekarang suatu data baru masuk dengan keluaran -1. Maka kita harus mengelompokkan apakah data ini ada dalam kelas -1 atau +1 menggunakan fungsi pemisah yang sudah kita temukan. Dengan mengguankan f_1, kita akan kelompokkan data baru ini di kelas -1 yang berarti kita benar mengelompokkannya. Kemudian dengan f_2 kita akan menempatkannya di kelas +1 yang berarti salah. Dari contoh sederhana ini kita lihat bahwa memperbesar margin bisa meningkatkan probabilitas pengelompokkan suatu data secara benar.

2.4.1 Support Vector Machine untuk Multi-Kelas

Pada awal dikembangkannya SVM pendekatan ini digunakan untuk klasifikasi dua kelas. Pengembangan ke arah persoalan klasifikasi untuk multi kelas masih menjadi perhatian peneliti [15]. Terdapat dua pendekatan utama kuntuk SVM multi kelas, yang pertama kita dapat menemukan dan menggabungkan beberapa fungsi pemisah persoalan klasifikasi dua kelas

untuk menggabungkan beberapa fungsi pemisah persoalan klasifikasi dua kelas untuk menyelesaikan persoalan multi kelas. Kedua, secara langsung menggunakan semua data dari semua kelas dalam satu formulasi persoalan optimasi. Yang termasuk pada pendekatan pertama diama beberapa fungsi untuk problem dua kelas dikembangkan lalu digabung antara lain: satu-lawan-semua (One-Against-All, OAA), dan satu-lawan-satu (One-Against-One, OAO) [15].

2.4.1.1 Metode Satu-Lawan-Semua (*One-Against-All*)

Dengan menggunakan metode ini, dibangun k sebuah model SVM biner (k adalah jumlah kelas). Setiap model klasifikasi ke-i dilatih dengan menggunakan keseluruhan data, untuk mencari solusi, terdapat permasalahan klasifikasi dengan 4 buah kelas. Untuk pelatihan digunakan 4 buah SVM biner seperti pada tabel 2.1 dan penggunaannya dalam mengklasifikasi data baru dapat dilihat pada gambar 2.15.

$$\begin{aligned} &\min_{w^i,b^i,\xi^j} \frac{1}{2}\left(w^i\right)^T w^i + C\sum_t \xi_t^i \\ &s.t\ \left(w^i\right)^T \phi(x_t) + b^i \geq 1 - \xi_t^i \rightarrow y_t = i, \\ &\left(w^i\right)^T \phi(x_t) + b^i \geq -1 + \xi_t^i \rightarrow y_t \neq i, \\ &\xi_t^i \geq 0 \end{aligned} \tag{2.37}$$

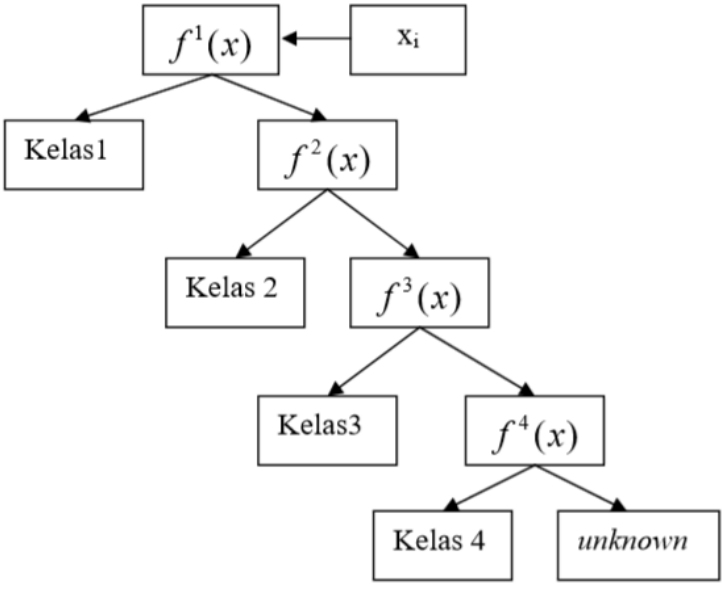
Gambar 2.14 Perhitungan Metode *One Vs All*

Tabel 2.1 Contoh 4 SVM Biner dengan Metode *One Vs All*

y_i = 1	y_i = -1	Hipotesis
Kelas 1	Bukan kelas 1	$f^1(x) = (w^1)x + b^1$
Kelas 2	Bukan kelas 2	$f^2(x) = (w^2)x + b^2$
Kelas 3	Bukan kelas 3	$f^3(x) = (w^3)x + b^3$
Kelas 4	Bukan kelas 4	$f^4(x) = (w^4)x + b^4$

Konsep pada OAA yaitu dimisalkan pada kasus lima kelas, kelas 1, 2, 3, dan 4. Bila akan diujikan $\rho^{(1)}$, semua data dalam kelas 1 diberi label +1 dan data dari kelas lainnya diberi label -1. Pada $\rho^{(2)}$, semua data dalam kelas 2 diberi label +1 dan data dari kelas lainnya diberi label -1 dst hingga data terakhir. Kemudian dicari *hyperplane* dengan algoritma SVM du akelas. Maka akan didapat *hyperplane* untuk masing-masing kelas diatas. Kemudian kelas dari suatu data barus *x* ditentukan berdasarkan nilai terbesar dari *hyperplane* [17]:

$$kelas\ x = \arg\max_{l=1\ldots k} \left(\left(w^{(l)}\right)^T . \emptyset(x) + b^{(l)} \right) \tag{2.38}$$



Gambar 2.15 Contoh Klasifikasi dengan Metode *One Vs All*

2.4.1.2 Metode Satu-Lawan-Satu (*One-Against-One*)

Dengan menggunakan metode ini, dibangun persamaan:

$$\frac{k(k-1)}{2} \tag{2.39}$$

Ket:

k = jumlah kelas dalam SVM

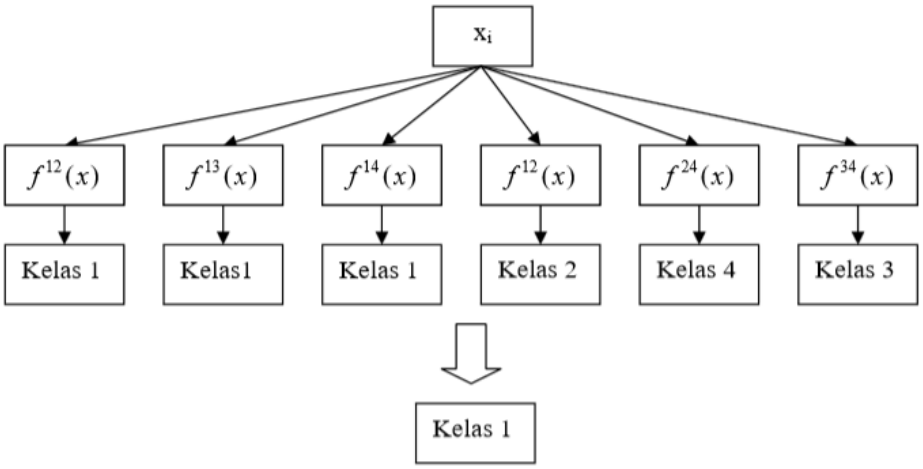
Setiap model klasifikasi dilatih pada data dari dua kelas. Untuk data pelatihan dari kelas ke-i dan kelas ke-j, dilakukan pencarian solusi untuk persoalan optimasi konstrain sebagai berikut:

$$\begin{aligned} &\min_{w^{ij},b^{ij},\xi^{ij}} \frac{1}{2}\left(w^{ij}\right)^T w^{ij} + C\sum_t \xi_t^{ij} \\ &s.t\ \left(w^{ij}\right)^T \phi(x_t) + b^{ij} \geq 1 - \xi_t^{ij} \rightarrow y_t = i, \\ &\left(w^{ij}\right)^T \phi(x_t) + b^{ij} \geq -1 + \xi_t^{ij} \rightarrow y_t = j, \\ &\xi_t^{ij} \geq 0 \end{aligned} \tag{2.40}$$

Terdapat beberapa metode untuk melakukan pengujian setelah keseluruhan k(k-1)/2 model klasifikasi selesai dibangun. Salah satunya adalah metode voting [15].

Tabel 2.2 Contoh 6 SVM Biner dengan Metode *One Vs One*

y_i = 1	y_i = -1	Hipotesis
Kelas 1	Kelas 2	$f^{12}(x) = (w^{12})x + b^{12}$
Kelas 1	Kelas 3	$f^{13}(x) = (w^{13})x + b^{13}$
Kelas 1	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 2.16 Contoh Klasifikasi dengan Metode *One Vs One*

Jika data *x* dimasukan kedalam fungsi hasil pelatihan:

$$f(x) = \left(w^{ij}\right)^T \emptyset(x) + b) \tag{2.41}$$

Dan hasilnya menyatakan *x* adalah kelas *i*, maka suara untuk kelas *i* ditambah satu. Kelas dari data *x* akan ditentukan dari jumlah suara terbanyak. Jika terdapat dua buah kelas yang jumlah suaranya sama, maka kelas yang indeksnya lebih kecil dinyatakan sebagai kelas dari data. Jadi pada pendekatan ini terdapat $\frac{k(k-1)}{2}$ buah persamaan *Quadratic Programming* yang masing-masing memiliki 2*n/k* variabel (*n* adalah jumlah data pelatihan). Contohnya, terdapat permasalahan klasifikasi dengan 4 buah kelas. Oleh karena itu, dugnakan 6 buah SVM biner seperti pada tabel 2 dan contoh penggunaannya dalam memprediksi kelas data baru.

2.5 Sertifikat

Menurut Kamus Besar Bahasa Indonesia (KBBI) arti dari kata sertifikat merupakan tanda atau surat keterangan (pernyataan) tertulis atau tercetak dari orang yang berwenang yang dapat digunakan sebagai bukti kepemilikan atau suatu kejadian [18].

Sertifikat memiliki fungsi untuk membuktikan bahwa seseorang telah mengikuti suatu kegiatan pelatihan, kegiatan workshop, kegiatan seminar, kegiatan kemah pramuka dan kegiatan lainnya yang dilaksanakan dengan maksud memberikan edukasi kepada pesertanya. Bagian-bagian sertifikat tersebut adalah sebagai berikut:

1. Kop sertifikat atau piagam dan logo organisasi
2. Tulisan sertifikat atau piagam
3. Nomor surat
4. Nama peserta
5. Waktu dan tempat
6. Pengesahan

2.6 UML

UML (*Unified Modeling Language*) adalah salah standar mendefinisikan *requirement*, membuat analisis & desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek [19].

UML adalah bahasa grafis untuk mendokumentasi, menspesifikasikan, dan membangun sistem perangkat lunak. UML berorientasi objek, menerapkan banyak leel abstraksi, tidak bergantung proses pengembangan, tidak bergantung bahasa dan teknologi, pemanduan beberapa notasi di beragam metodologi, usaha bersama dari banyak pihak, didukung oleh kakas-kakas yang diintegrasikan lewat XML (XMI). Standar UML dikelola oleh OMG (*Object Management Group*) [20].

Tujuan utama perancangan UML adalah:

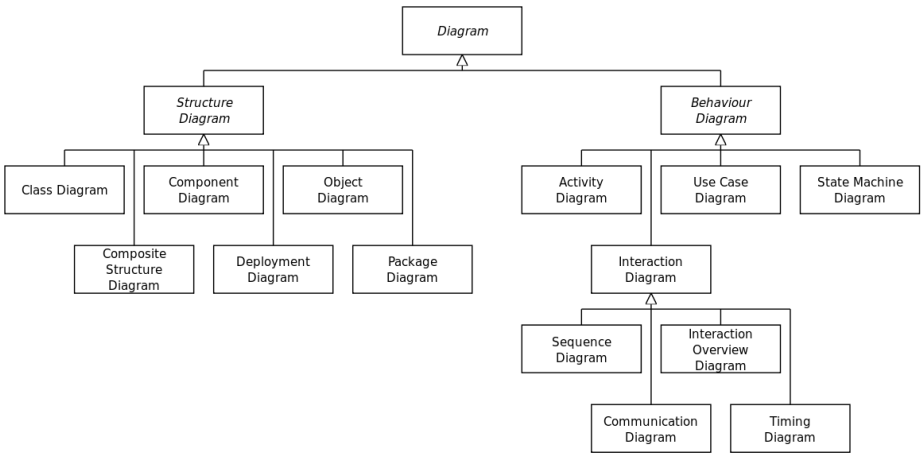
1. Menyediakan bahasa pemodelan visual yang ekspresif dan siap pakai untuk mengembangkan dan pertukran model-model yang berarti
2. Menyediakan mekanisme perluasan dan spesifikasi untuk memperluas konsep-konsep inti
3. Mendukung spesifikasi independen bahasa pemrograman dan proses pengembangan tertentu
4. Menyediakan basis formal untuk pemahaman bahasa pemodelan
5. Mendorong pertumbuhan pasar kakas berorientasi objek

6. Mendukung konsep-konsep pengembangan level lebih tinggi seperti komponen, kolaborasi, framework, dan pattern

Secara fundamental, UML berkaitan dengan penangkapan dan komunikasi pengetahuan.

Konsep-konsep ang diterapkan di UML adalah satu model berisi informasi mengenai sistem (atau domain), model-model berisi elemen-elemen model seperti kelas-kelas, simpul-simpul, paket-paket, dan sebagainya. Satu diagram menunjukkan satu pandangan tertentu dari model.

UML adalah meta model, yaitu UML mendefinisikan jenis-jenis elemen yang dapat digunakan pengembang di model-model UML-nya dan konstrain-konstrain dari penggunaannya. UML menyediakan mekanisme perluasan dan mengakomodasikan konsep-konsep baru dengan meta model yang ditawarkannya.

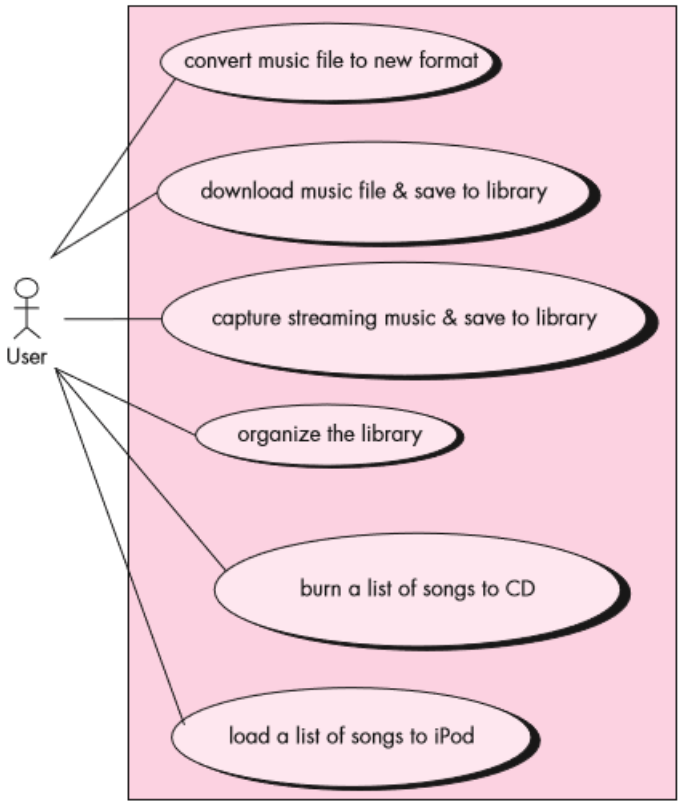


Gambar 2.17 Struktur Diagram UML

2.6.1 *Usecase* Diagram

Diagram *Usecase* (*usecase* diagram) merupakan salah satu diagram untuk memodelkan aspek perilaku sistem masing-masing digram *usecase* menunjukkan sekumpulan *usecase*, aktor, dan hubungannya. Diagram *usecase* adalah penting untuk memvisualisasikan, menspesifikasikan, dan mendokumentasikan kebutuhan perilaku sistem. Diagram-diagram *usecase*

merupakan pusat pemodelan perilaku sistem, subsistem, dan kelas [20]. Berikut adalah contoh dari *Usecase* Diagram pada gambar 2.14:



Gambar 2.18 Contoh *Usecase* Diagram

Usecase adalah interaksi antara aktor eksternal dan sistem, hasil yang dapat diamati oleh aktor, berorientasi pada tujuan, dideskripsikan di diagram *usecase* dan teks. Diagram *usecase* melibatkan:

1. Sistem yaitu sesuatu yang hendak dibangun
2. Aktor, entitas-entitas luar yang berkomunikasi dengan sistem
3. *Usecase* adalah fungsionalitas dipersepsi oleh aktor
4. Reliasi adalah relasi antara aktor dengan *usecase*

Diagram *usecase* digunakan untuk mendeskripsikan apa yang seharusnya dilakukan oleh sistem. Diagram *usecase* menyediakan cara mendeskripsikan pandangan eksternal terhadap sistem dan interaksi-

interaksinya dengan dunia luar. Dengan cara ini, diagram *usecase* 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 *usecase* diciptakan oleh Ivar Jacobson.

Tujuan utama pemodelan *usecase* adalah:

1. Memutuskan dan mendeskripsikan kebutuhan-kebutuhan fungsional sistem
2. Memberikan deskripsi jelas dan konsisten dari apa yang seharusnya dilakukan, sehingga model *usecase* digunakan di seluruh proses pengembangan untuk komunikasi dan menyediakan basis pemodelan berikutnya yang mengacu sistem harus memberikan fungsionalitas yang dimodelkan pada *usecase*
3. Menyediakan basis untuk melakukan pengujian sistem yang memverifikasi sistem. Menguji apakan sistem telah memberikan fungsionalitas yang diminta
4. Menyediakan kemampuan melacak kebutuhan fungsionalitas menjadi kelas-kelas dan operasi-operasi aktual di sistem. Untuk menyederhanakan perubahan dan ekstensi ke sistem dengan mengubah model *usecase* dan kemudian melacak *usecase* yang dipengaruhi ke perancangan dan implementasi sistem

Elemen pada diagram *usecase* adalah:

1. Aktor

   Aktor adalah pemakai sistem, dapat berupa manusia atau sistem terotomatisasi lain. Aktor adalah sesuatau 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 atau dari sistem, atau mempertukarkan informasi dengan sistem. Dalam perspektif aktor, *usecase* melakukan sesuatu yang berharga bagi aktor.

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

Aktor berkomunikasi dengan sistem lewat pengiriman dan penerimaan pesan. *Usecase* selalu diawali oleh aktor yang mengirim pesan, disebut dengan stimulus. Ketika satu *usecase* dilakukan, *usecase* dapat mengirim pesan ke satu aktor atau lebih.

2. *Usecase*

   Usecase adalah cara spesifik penggunaan sistem oleh aktor. Ciri-ciri dari *usecase* adalah:

   a. Pola perilaku yang harus dipenuhi oleh sistem
   b. Sekuen transaksi terhubung yang dilakukan aktor dan sistem
   c. Memberikan sesuatu yang berharga bagi aktor

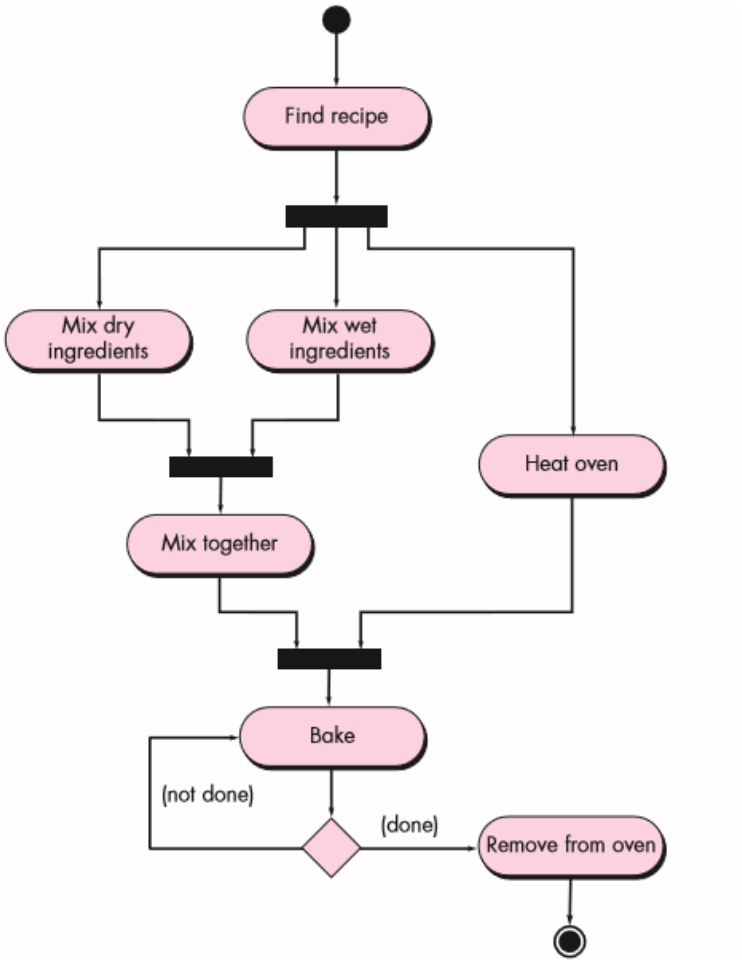
3. Hubungan ketergantungan, generalisasi dan asosiasi

   Keterhubungan antar *usecase* dengan *usecase* lain berupa generalisasi antara *usecase* yaitu:

   a. Include, perilaku *usecase* merupakan bagian dari *usecase* yang lain
   b. Extend, perilaku *usecase* memperluas perilaku *usecase* yang lain

2.6.2 *Activity* Diagram

Diagram aktivitas adalah diagram *flowchart* yang diperluas yang menunjukkan aliran kendali suatu aktivitas ke aktivitas lain. Diagram ini digunakan untuk memodelkan aspek dinamis sistem. Aktivitas adalah eksekusi nonatomik yang berlangsung di *state machine*. Diagram aktivitas mendeskripsikan aksi-aksi dan hasilnya. Diagram aktivitas berupa operasi-operasi dan aktivitas-aktivitas di *usecase* [20]. Berikut adalah contoh dari *Activity* Diagram pada gambar 2.19:



Gambar 2.19 Contoh *Activity* Diagram

Diagram aktivitas dapat digunakan untuk:

1. Pandangan dalam yang dilakukan di operasi
2. Pandangan dalam bagamana objek-objek bekerja
3. Pandangan dalam di aksi-aksi dan pengaruhnya pada objek-objek
4. Pandangan dalam dari suatu *usecase*
5. Logik dari proses bisnis

Diagram aktivitas merupakan jenis khusus dan diagram *statechart*. *State* adalah aksi-aksi yang menuju state berikutnya setelah selesai aksi itu.

Diagram aktivitas berfokus pada aktivitas-aktivitas, potongan-potongan dari proses yang boleh jadi (mungkin) berkorespondensi dengan

metode-metode atau fungsi-fungsi anggota dan pengurutan dari aktivitas-aktivitas ini. Hal ini serupa dengan flowchart. Namun, diagram aktivitas berbeda dari flowchart terutama karena diagram aktivitas secara eksplisit mendukung aktivitas-aktivitas paralel dan sinkronasi aktivitas-aktivitas itu.

Penggunaan utama model aktivitas adalah:

1. Memodelkan *workflow*

   Pembuatan *workflow* adalah sebagai berikut:

   a. Tetapkan fokus dari *workflow*. Untuk sistem yang tidak sepele, tidak mungkin menunjukkan semua *workflow* yang penting di satu diagram.
   b. Pilih objek bisnis yang mempunyai tanggungjawab level tinggi untuk bagian dari *workflow*. Ini dapat berupa sesuatu yang nyata di sistem, atau lebih abstrak. Dalam kasus tertentu dapat digunakan *swimlane* untuk masing-masing bisnis penting.
   c. Identifikasikan precondition dari state awal workflow dan postcondition dari state akhir workflow. Ini penting untuk memodelkan batas dari workflow.
   d. Mulai dari state awal workflow, spesifikasikan aktivitas dan aksi yang ada dan tempatkan sebagai aktivitas di diagram.
   e. Untuk aksi yang rumit, atau sekelompok aksi yang muncul berulang kali, jadikan menjadi satu state aktivitas dan kemudian buat diagram aktivitas tersendiri untuk menggambarkannya.
   f. Buat trasisi yang menghubungkan antar aktivitas.
   g. Jika terdapat objek penting yang terlibat di workflow, maka gambarkan di diagram aktivitas.

2. Memodelkan operasi

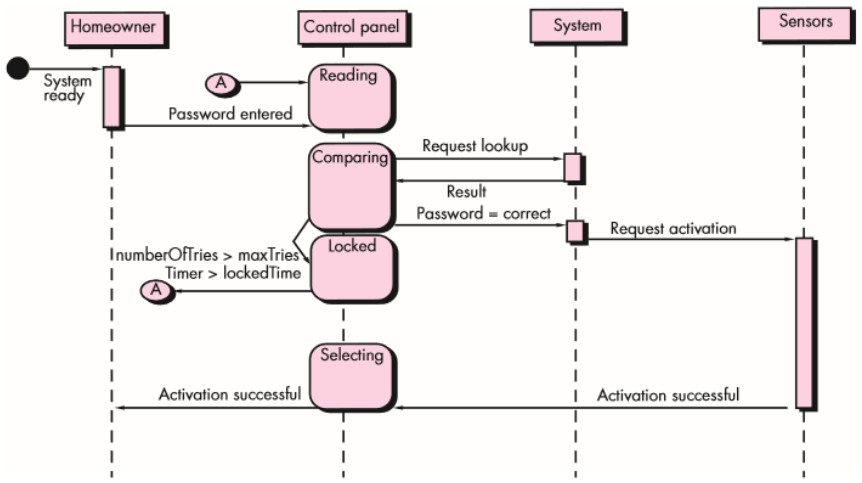
   Diagram aktivitas untuk memodelkan operasi adalah sebagai berikut:

   a. Kumpulkan abstraksi yang terlibat di operasi. Ini meliputi parameter operasi, atribut di kelas dan kelas-kelas tetangga yang terlibat.

b. Identifikasi preconditions dari state awal operasi dan postcondition dari state akhir dari operasi. Juga identifikasi invarian di kelas yang harus selalu dipenuhi selama eksekusi operasi.
c. Dimulai dari state awal operasi, spesifikasikan aktivitas-aktivitas dan aksi-aksi yang terjadi dan buat di diagram aktivitas sebagai state aktivitas atau state aksi.
d. Gunakan pencabangan bila diperlukan untuk menspesifikasikan jalur kondisi dan iterasi.
e. Hanya jika operasi itu dimiliki oleh kelas aktif, gunakan forking dan joining bila perlu untuk menspesifikasikan alur paralel dari kendali.

2.6.3 *Sequence* Diagram

Diagram sekuen mendeskripsikan komunikasi di antara objek-objek, meliputi pesan-pesan yang ada dan urutan pesan tersebut muncul. Berikut adalah contoh *sequence* diagram pada gambar 2.20:



Gambar 2.20 Contoh *Sequence* Diagram

Diagram sekuen digunakan untuk:

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

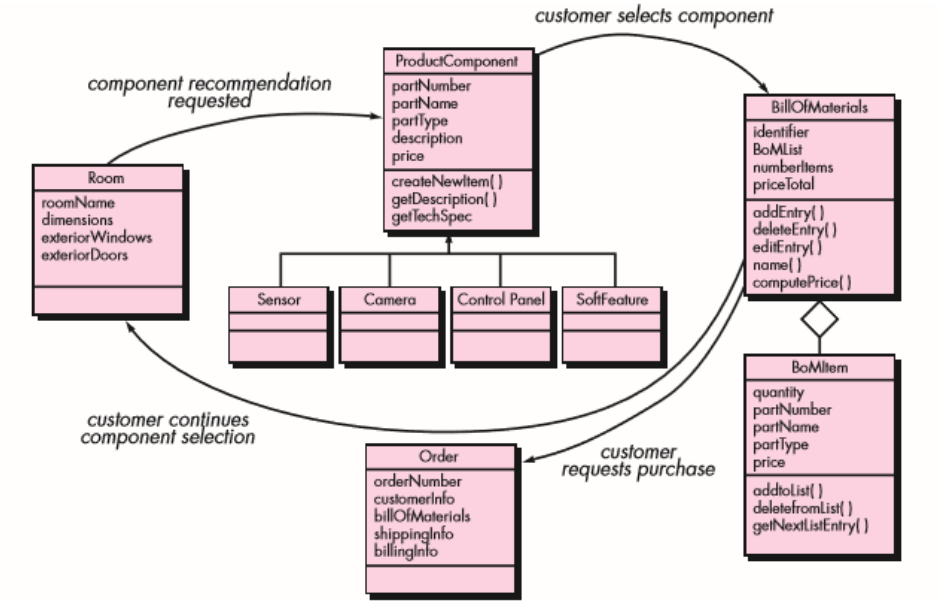
4. Memeriksa jalur-jalur pengaksesan

Diagram sekuen digunakan untuk memodelkan skenario penggunaan. Skenario penggunaan adalah barisan kejadian yang terjadi selama satu eksekusi sistem. Cakupan skenario dapat beragam, dari mulai semua kejadian di sistem atau hanya kejadian pada objek-objek tertentu.skenario menjadi rekaman historis eksekusi sistem atau gagasan eksperimen eksekusi sistem yang diusulkan.

Diagram sekuen menunjukkan objek sebagai garis vertikal dan tiap kejadian sebagai panah horisontal dari objek pengirim ke objek penerima. Waktu berlalu dari atas ke bawah dengan lama waktu tidak relevan. Diagram ini hanya menunjukkan barisan kejadian, bukan pewaktuan nyata. Kecuali untuk sistem waktu nyata yang mengharuskan konstrain barisan kejadian [20].

2.6.4 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 [20]. Berikut adalah contoh dari *class* diagram pada gambar 2.21:



Gambar 2.21 Contoh *Class* Diagram

Diagram kelas menunjukkan aspek statik sistem terutama untuk mendukung kebutuhan fungsional sistem. Kebutuhan fungsional berarti layanan-layanan yang harus disediakan sistem ke pemakai. Meskipun diagram kelas serupa dengan model data, namun kelas-kelas tidak hanya menunjukkan struktur informasi tapi juga mendeskripsikan perilaku. Salah satu maksud diagram kelas adalah untuk mendefinisikan fondasi bagi diagram-diagram lain dimana aspek-aspek lain dari sistem ditunjukkan.

Kelas di diagram kelas dapat secara langsung diimplementasikan di bahasa pemrograman berorientasi objek yang secara langsung mendukung bentukan kelas.

Elemen-elemen esensi di diagram kelas adalah sebagai berikut:

1. Kelas

   Kelas merupakan elemen terpenting di sistem berorientasi objek. Kelas mendeskripsikan satu blok pembangun sistem. Kelas memiliki sejumlah fitur, kita dapat memodelkan multiplisitas, ketampakan, penanda, polymorphism, dan karakteristik-karakteristik lain.

2. Antarmuka (*interfaces*)

   Antarmuka (interfaces) merupakan koleksi operasi yang menspesifikasikan layanan dari suatu kelas atau komponen. Antarmuka mendeskripsikan perilaku tampak secara eksternal dari elemen.

3. Kolaborasi

   Kolaborasi merupakan pendefinisian suatu interaksi dan sekelompok peran dan elemen-elemen lain yang bekerja sama untuk menyediakan suatu perilaku kooperatif yang lebih besar dari penjumlahan seluruh elemen. Kolaborasi memiliki struktur, perilaku dan dimensi. Suatu kelas dapat berpartisipasi di beberapa kolaborasi. Kolaborasi ini merepresentasikan implementasi pola tertentu yang membentuk sistem.

4. Hubungan (*relationship*)

   Hubungan antar kelas di diagram kelas beraneka ragam, yaitu:

   a. Asosiasi

      Asosiasi merepresentasikan hubungan antara instan-instan kelas. Interpretasi dari asosiasi bervariasi sesuai perspektifnya. Secara konseptual, asosiasi merepresentasi hubungan antara kelas-kelas yang terlibat. Pada perspektif spesifikasi, terdapat tanggungjawab untuk mengetahui, dan akan dibuat eksplisit dengan operasi pengaksesan dan pembaruan. Ini berarti terdapat pointer antara satu kelas dan kelas lainnya, tapi ini tersembunyi oleh pengkapsulan. Perspektif implementasi mengimplikasikan keberadaan pointer. Dengan demikian hal esensi untuk mengetahui perspektif yang digunakan saat membangun model agar menginterpretasikan secara benar.

   b. Generalisasi

      Subtype merupakan penambahan penting terhadap ERD (*Entity Relationship Diagram*) di orientasi objek. Fasilitas ini memiliki koerspondensi dengan pewarisan di pemrograman berorientasi objek.

c. *Dependency*

Terdapat delapan *stereotype* terhadap *dependency* diantara kelas dan objek, yaitu:

1) *Bind*, menspesifikasikan sumber menginstaniasi template target menggunakan parameter aktual.
2) *Derive*, menspesifikasikan sumber dapat dikomputasi dari target.
3) *Friend*, menspesifikasikan sumber diberi ketampakan spesial oleh target.
4) *instanceOf*, menspesifikasikan objek sumber adalah instan dari kelas.
5) *Instantiate*, menspesifikasikan sumber menciptakan instan dari target.
6) *Powertype*, menspesifikasikan target adalah powertype dari sumber, powertype adalah classifier yang memiliki objek-objek adalah anak-anak dari induk tertentu.
7) *Refine*, menspesifikasikan sumber adalah derajat abstraksi yang lebih baik dibanding target.
8) *Use*, menspesifikasikan semantiks dari elemen sumber bergantung pada semantiks bagian publik dari target.

2.7 HTML

HyperText Markup Language (HTML) adalah bahasa yang digunakan untuk menulis halaman web. HTML merupakan pengembangan dari standar pemformatan dokumen teks, yaitu *Standard Generalized Markup Language* (SGML). HTML pada dasarnya merupakan dokumen ASCII atau teks biasa, yang dirancang untuk tidak tergantung pada suatu sistem operasi tertentu.

HTML dibuat oleh tim Beners-Lee ketika masih bekerja untuk CERN (CERN adalah lembaga penelitian fisika energi tinggi di Jenewa), dan dipopulerkan pertama kali oleh broser Mosaic. Selama awal tahun 1990, HTML mengalami perkembangan yang sangat pesat. Setiap perkembangan HTML, pasti akan menambahkan kemampuan dan fasilitas yang lebih baik dari versi sebelumnya. Kegunaan bahasa ini ialah untuk memanipulasi broser

sehingga dapat menampilkan informasi yang dapat dibaca oleh pengguna komputer [21].

2.8 CSS

CSS (*Cascading Style Sheet*) adalah suatu bahasa stylesheet yang digunakan untuk mengatur tampilan suatu website, baik tata letaknya, jenis huruf, warna, dan semua yang berhubungan dengan tampilan. Pada umumnya CSS digunakan untuk memformat halaman web yang ditulis dengan HTML atau XHTML [21].

Ada dua cara yang bisa diterapkan untuk menggunakan CSS pada web. Cara yang pertama dengan membuat CSS langsung di dalam satu file HTML (internal/*inline style sheet*). Cara yang kedua dengan memanggil CSS tersebut dari file CSS tersendiri (*external style sheet*).

Berikut ini merupakan contoh dari internal dan eksternal CSS:

```
<head>
      <style type=”text/css”>
            hr (color:sienna;)
            p (margin-left:20px;)
            body (background-image:url(“images/back40.gif”);)
      </style>
</head>
```

Gambar 2.22 Contoh Internal CSS

```
<head>
      <link rel=”stylesheet” type=”text/css” href=”mystyle.css”
/>
<head>
```

Gambar 2.23 Contoh Eksternal CSS

Apabila memakai internal CSS, semua kode CSS dan markup dimasukkan dalam satu file yang sama, sedangkan jika memakai eksternal CSS diperlukan link untuk menghubungkan keduanya.

2.9 JavaScript

JavaScript adalah bahasa script berdasar pada objek yang memperbolehkan pemakai untuk mengendalikan banyak aspek interaksi pemakai pada suatu dokumen HTML. Dimana objek tersebut dapat berupa suat window, frame, URL, dokumen, form, button, atau item yang lain. Yang semuanya mempunyai properti yang saling berhubungan dengannya, dan masing-masing memiliki nama, lokasi, warna nilai, dan atribut lain [21].

2.10 Php

2.10.1 Pengertian Php

Php (*Hypertext Preprocessing*) adalah sebuah bahasa pemrograman script yang paling banyak dipakai saat ini. Php adalah bahasa scripting yang dirancang khusus untuk digunakan di web. Php ini memiliki fitur untuk membantu dalam memprogram tugas-tugas yang diperlukan untuk mengembangkan aplikasi web dinamis [22].

2.10.2 Sejarah Php

Asal-usul PHP tanggal kembali ke 1995 ketika kontraktor pengembangan perangkat lunak independen bernama Rasmus Lerdorf mengembangkan skrip Perl / CGI yang memungkinkan dia untuk mengetahui berapa banyak pengunjung yang membaca resume online-nya. Naskahnya melakukan dua tugas: mencatat informasi pengunjung, dan menampilkan jumlah pengunjung ke halaman web. Karena Web pada saat itu masih merupakan teknologi baru, alat seperti ini tidak ada. Jadi, skrip Lerdorf menghasilkan sedikit ketertarikan. Lerdorf mulai membagikan toolsetnya, yang dijuluki Personal Home Page (PHP).

Kebisingan mendorong Lerdorf untuk terus mengembangkan bahasa, dengan mungkin perubahan awal yang paling penting adalah fitur baru untuk mengubah data yang dimasukkan dalam bentuk HTML ke dalam variabel simbolis, mendorong ekspor ke sistem lain. Untuk mencapai hal ini, ia memilih untuk melanjutkan pengembangan dalam kode C daripada Perl. Penambahan berkelanjutan ke toolset PHP memuncak pada bulan

November 1997 dengan rilis PHP 2.0, atau Personal Home Page / Form Interpreter (PHP / FI). Pelepasan 2,0 disertai dengan sejumlah peningkatan dan perbaikan dari programmer di seluruh dunia.

Rilis PHP baru sangat populer, dan tim inti pengembang segera bergabung dengan Lerdorf. Mereka menyimpan konsep asli menggabungkan kode langsung bersama HTML dan menulis ulang mesin parsing, melahirkan PHP 3.0. Pada rilis Juni 1998 versi 3.0, lebih dari 50.000 pengguna menggunakan PHP untuk meningkatkan halaman Web mereka.

Pembangunan berlanjut pada kecepatan yang sibuk selama dua tahun ke depan, dengan ratusan fungsi ditambahkan dan basis pengguna tumbuh dengan pesat. Pada awal tahun 1999, Netcraft (www.netcraft.com), sebuah perusahaan riset dan analisis Internet, melaporkan perkiraan konservatif dari basis pengguna lebih dari 1 juta, menjadikan PHP salah satu bahasa scripting paling populer di dunia. Popularitasnya bahkan melampaui harapan terbesar para pengembang, dan segera menjadi jelas bahwa pengguna berniat menggunakan PHP untuk menjalankan aplikasi yang jauh lebih besar dari yang diperkirakan sebelumnya. Dua pengembang inti, Zeev Suraski dan Andi Gutmans, mengambil inisiatif untuk sepenuhnya memikirkan kembali cara PHP dioperasikan, yang berpuncak pada penulisan ulang parser PHP, yang dijuluki mesin scripting Zend. Hasil dari pekerjaan ini adalah dalam rilis PHP 4.

2.11 CodeIgniter

CodeIgniter adalah sebuah Application Development Framework (toolkit) bagi orang-orang yang ingin membangun website menggunakan Php. Tujuannya adalah untuk memungkinkan anda mengembangkan proyek-proyek lebih cepat daripada harus menulis kode dari awal, tersedia banyak library untuk tugas-tugas yang biasa diperlukan, serta antarmuka dan struktur logis yang sederhana untuk mengakses library ini. CodeIgniter memungkinkan fokus pada sebuah proyek dengan meminimalkan jumlah kode yang dibutuhkan untuk tugas yang diberikan [23].

CodeIgniter didasarkan pada pola pengembangan Model-View-Controller. MVC adalah pendekatan perangkat lunak yang memisahkan logika aplikasi dari presentasi. Dalam prakteknya, itu memungkinkan halaman web yang dimiliki scripting tersebut minimal karena presentasi terpisah dari scripting Php [24]. Berikut ini adalah penjelasan dari setiap *class*:

1. *Model*, mewakili struktur data. Biasanya *class model* akan berisi fungsi yang membantu mengambil, menyimpan, dan memperbarui informasi dalam database.
2. *View*, informasi yang disajikan kepada pengguna. *View* yang biasanya akan menjadi halaman web, tetapi dalam CodeIgniter, *view* juga bisa menjadi bagian dari sebuah halaman seperti *header* atau *footer*. Hal ini juga dapat menjadi halaman RSS, atau jenis-jenis lain dari "halaman".
3. *Controller*, berfungsi sebagai perantara antara *Model*, *View*, dan *recource* lain yang diperlukan untuk memproses HTTP request dan menghasilkan halaman web.

CodeIgniter memiliki pendekatan yang cukup longgar untuk MVC karena Model tidak selalu diperlukan. Jika anda tidak perlu menambahkan pemisahan, atau menemukan bahwa mempertahankan sebuah model memerlukan kompleksitas lebih dari yang diinginkan. Model bisa diabaikan dan membangun aplikasi dengan minimal menggunakan Controller dan View. CodeIgniter juga memungkinkan untuk memasukkan script sendiri, atau bahkan mengembangkan library inti untuk sistem, mengukinkan untuk bekerja dengan cara yang paling masuk akal.

2.12 Perangkat Lunak Pendukung

2.12.1 MySQL

Banyak situs web dinamis memerlukan database backend. Database dapat berisi informasi yang ditampilkan dari halaman web kepada pengguna, atau bertujuan dari database mungkin untuk menyimpan informasi yang diberikan oleh pengguna. Dalam beberapa aplikasi, database

keduanya menyediakan informasi yang tersedia dan menyimpan informasi baru.

MySQL adalah database yang paling populer untuk digunakan dalam situs web, dikembangkan menjadi cepat dan kecil, khususnya untuk situs web. MySQL sangat populer untuk digunakan dengan situs web yang ditulis dalam bahasa pemrograman PHP, PHP dan MySQL bekerja sama dengan baik [22].

Fitur-fitur MySQL antara lain:

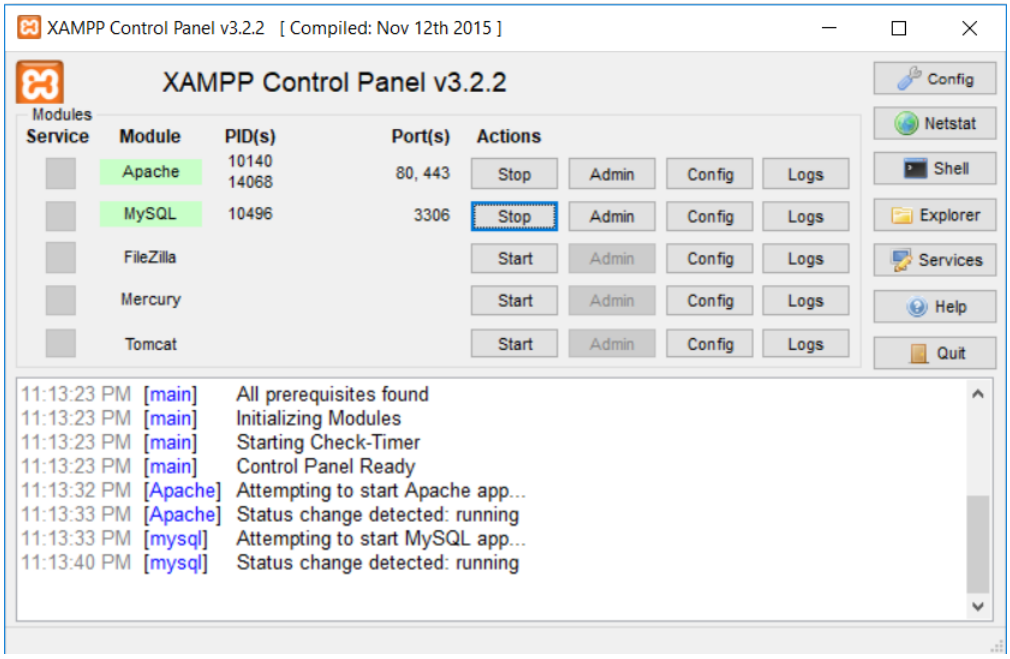
1. ***Relational Database System***. Seperti halnya software database lain yang ada di pasaran, MySQL termasuk RDBMS.
2. **Arsitektur *Client-Server***. MySQL memiliki arsitektur client-server dimana server database MySQL terinstall di server. Client MySQL dapat berada di komputer yang sama dengan server, dan dapat juga di komputer lain yang berkomunikasi dengan server melalui jaringan bahkan internet.
3. **Mengenal perintah SQL standar**. SQL (*Structured Query Language*) merupakan suatu bahasa standar yang berlaku di hampir semua software database. MySQL mendukung SQL versi SQL:2003.
4. **Mendukun Sub Select**. Mulai versi 4.1 MySQL telah mendukung select dalam select (sub select).
5. **Mendukung Views**. MySQL mendukung views sejak versi 5.0.
6. **Mendukung *Stored Prosedured* (SP)**. MySQL mendukung SP sejak versi 5.0.
7. **Mendukung *Triggers***. MySQL mendukung trigger pada versi 5.0 namun masih terbatas. Pengembang MySQL berjanji akan meningkatkan kemampuan trigger pada versi 5.1.
8. Mendukung *replication*.
9. Mendukung transaksi.
10. Mendukung *foreign key*.
11. Tersedia fungsi GIS.
12. Free (bebas didownload).

13. Stabil dan tangguh.
14. Fleksibel dengan berbagai pemrograman.
15. Security yang baik.
16. Dukungan dari banyak komunitas.
17. Perkembangan software yang cukup cepat.

2.12.2 XAMPP

XAMPP adalah kit all-in-one yang populer dalam menginstal apache, MySQL, dan PHP dalam satu prosedur. XAMPP juga menginstal phpMyAdmin, aplikasi web yang dapat digunakan untuk mengelola database MySQL [22].

Menurut situs web XAMPP, XAMPP dimaksudkan sebagai lingkungan pengembangan pada komputer lokal. Sebagai lingkungan pengembangan, XAMPP dikonfigurasi untuk menjadi seterbuka mungkin. XAMPP tidak dimaksudkan untuk penggunaan produksi yang tidak aman sebagai lingkungan produksi. Berikut ini adalah tampilan aplikasi dari XAMPP:



Gambar 2.24 Tampilan Aplikasi XAMPP

Bagian yang penting dari XAMPP:

1. htdoc, merupakan folder untuk meletakkan file yang akan dijalankan, seperti file Php, HTML, dan script lainnya.
2. phpMyAdmin, merupakan bagian untuk mengelola database MySQL yang ada pada komputer.
3. Control Panel, berfungsi untuk mengelola layanan (service) XAMPP, seperti start service (mulai) atau stop service (berhenti).

XAMPP adalah singkatan dari setiap huruf, yaitu:

1. X: Program ini dapat dijalankan di banyak sistem operasi, seperti Windows, Linux, Mac OS, dan Solaris.
2. A: Apache, server aplikasi web. Tugas utama apache adalah untuk menghasilkan halaman web yang benar kepada pengguna terhadap kode Php yang sudah dituliskan oleh pembuat halaman web.
3. M: MySQL, server aplikasi database. SQL (Structured Query Language) merupakan bahasa terstruktur yang difungsikan untuk mengolah database. MySQL dapat digunakan untuk membuat, mengelola database dan isinya.
4. P: Php, bahasa pemrograman web. Bahasa pemrograman Php (Hypertext Preprocessor) adalah bahasa pemrograman untuk membuat web berbasis server-side scripting. Php digunakan untuk membuat halaman web dinamis.
5. P: Perl, bahasa pemrograman untuk semua tujuan, pertama kali dikembangkan oleh Larry Wall, mesin Unix. Perl dirilis pertama kali tanggal 18 Desember 1987 yang ditandai dengan keluarnya Perl 1. Pada versi-versi selanjutnya, Perl juga tersedia untuk berbagai sistem operasi Unix (SunOS, Linux, BSD, HP-UX), juga tersedia untuk sistem operasi seperti DOS, Windows, PowerPC, BeOS, VMS, EBCDIC, dan PocketPC.

2.13 Pengujian Sistem

Pengujian sistem adalah proses pemeriksaan atau evaluasi sistem atau komponen sistem secara manual atau otomatis untuk memverifikasi apakah sistem memenuhi kebutuhan-kebutuhan yang dispesifikasikan atau mengidentifikasi perbedaan-perbedaan antara hasil yang diterapkan dengan hasil yang terjadi [20]. Pengujian seharusnya meliputi tiga konsep berikut.

1. Demonstrasi validitas perangkat lunak pada masing-masing tahap di siklus pengembangan sistem.
2. Penentuan validitas sistem akhir dikaitkan dengan kebutuhan pemakai.
3. Pemeriksa perilaku sistem dengan mengeksekusi sistem pada data sampel pengujian.

Pada dasarnya pengujian diartikan sebagai aktivitas yang dapat atau hanya dilakukan setelah pengkodean (kode program selesai). Namun, pengujian seharusnya dilakukan dalam skala lebih luas. Pengujian dapat dilakukan begitu spesifikasi kebutuhan telah dapat didefinisikan. Evaluasi terhadap spesifikasi dan perancangan juga merupakan Teknik pengujian. Kategori pengujian dapat dikategorikan menjadi dua [20], yaitu:

1. Berdasarkan ketersediaan logic sistem, terdiri dari *Black box* testing dan *White box testing*.
2. Berdasarkan arah pengujian, terdiri dari Pengujian *top down* dan Pengujian *bottom up*.

2.13.1 Pengujian *Black Box*

Konsep *Black box* digunakan untuk merepresentasikan sistem yang cara kerja didalamnya tidak tersedia untuk diinspeksi. Di dalam *black box*, item-item yang diuji dianggap "gelap" karena logikanya tidak diketahui, yang diketahui hanya apa yang masuk dan apa yang keluar dari *black box* [20].

Pada pengujian *black box*, kasus-kasus pengujian berdasarkan pada spesifikasi sistem. Rencana pengujian dapat dimulai sedini mungkin di proses pengembangan perangkat lunak. Teknik pengujian konvensional yang termasuk pengujian "*black box*" adalah sebagai berikut [20]:

1. Graph-based testing
2. Equivalence partitioning
3. Comparison testing
4. Orthogonal array testing

Pada pengujian *black box*, kita mencoba beragam masukan dan memeriksa keluaran yang dihasilkan. Kita dapat mempelajari apa yang dilakukan kotak, tapi tidak mengetahui sama sekali mengenai cara konversi dilakukan. Teknik pengujian *black box* juga dapat digunakan untuk pengujian berbasis scenario, dimana isi dalam sistem mungkin tidak tersedia diinspeksi tapi masukan dan keluaran yang didefinisikan dengan *use case* dan informasi analisis yang lain.

2.13.2 Pengujian Akurasi

Akurasi merupakan seberapa dekat suatu angka hasil pengukuran terhadap angka sebenarnya (true value atau reference value). Tingkat akurasi diperoleh dengan persamaan sebagai berikut.

$$Akurasi = \frac{Jumlah\ Karakter\ Sama}{Jumlah\ Seluruh\ Karakter} * 100\% \qquad (2.42)$$