BAB 2
LANDASAN TEORI

2.1. Penyakit Ikan

Penyakit ikan adalah segala sesutu yang dapat menimbulkan gangguan baik fisik maupun fisiologis pada ikan. Gangguan ini dapat disebabkan oleh organisme lain, kondisi lingkungan atau campur tangan manusia. Sakit adalah suatu kondisi dimana terjadi gangguan atau ketidak normalan fungsi pada ikan baik itu secara fisik maupun fisiologis. Sakit dan penyakit ini dapat disebabkan oleh ketidak serasian yang terjadi di dalam lingkungan atau ekosistem dimana ikan tersebut berada. Dengan kata lain penyakit merupakan interaksi yang tidak serasi antara ikan dengan faktor biotik (organisme) dan faktor *abiotic* (lingkungan). Interaksi yang tidak serasi ini akan menimbulkan stress pada ikan sehingga menyebabkan daya tahan tubuh menurun dan akibatnya mudah timbul berbagai penyakit. [**1**]

Faktor biotik yang merugikan pada ikan di dalam ekosistem dapat dibagi atas tiga kelompok besar yakni :

1. Parasite, yaitu organisme yang hidup dan memperoleh makanan dari inang yang ditumpanginya. Kedalam golongan ini termasuk bakteri protozoa, virus, crustacean (udang renik), cacing, dan jamur.
2. Hama, yaitu organisme yang menggangu atau merusak ikan secara fisik contohnya *Trynix* sp, *Egretta* sp, ular air, dan lain-lain.
3. Predator, yaitu hewan karnifora pemangsa misalnya *Varanus Salvador* (biawak).
4. Kompetitor, yakni organisme yang merupakan pesaing dalam memperolah oksigen, ruang, dan makanan seperti ikan-ikan liar, belut, dan lain-lain.

Faktor non-biotik yang sering juga disebut dengan factor non parasite, terdiri dari beberapa faktor, yakni :

1. Faktor lingkungan, diantra faktor lingkungan yang dapat merugikan kesehatan ikan ialah pH air yang terlalu tinggi atau rendah, kandungan oksigen yang rendah, temperatur yang berubah secara tiba-tiba, adanya gas beracun serta kandungan racun yang berada di dalam air yang berasal dari pestisida, pupuk, limbah pabrik, limbah rumah tangga, dan lain-lain.
2. Pakan, penyakit dapat timbul karena kualitas pakan yang diberikan tidak baik. Gizi rendah, kurang vitamin, pakan terlalu lama disimpan serta pemberian pakan yang tidak tepat.
3. Turunan, penyakit turunan atau genetis dapat berupa bentuk tubuh yang tidak normal dan pertumbuhan yang lambat.

2.1.1. Patologi Parasi Protozoa

Protozoa adalah organisme unisellular (bersel satu) mikroskopik yang sangat kecil dan memiliki struktur yang kompleks yang digunakan untuk pergerakan, pelekatan, dan pelindungan. Parasit ini memiliki beberapa kelompok yang ada pada ikan. Parasit ini dapat berkembang biak pada tubuh inang. Golongan protozoa setidaknya memiliki 7 phylum yang merupakan parasit pada ikan, yaitu : *Phylum Amoebozoa, Phylum Ciliophora, phylum Apicomplexa, Phylum Microspora, Phylum Myxozoa.*

Diantara golongan parasit air tawar, *Ichthiophthirius multifillis* telah menyebabkan banyak masalah dalam budidaya ikan air tawar. Pada kondisi budidaya, spesies protozoa tertentu dapat menyebabkan penyakit yang menghasilkan mortalitas tinggi yang berdampak pada kerugian ekonomi yang cukup besar pada ikan air tawar maupun ikan air laut. [**1**]

2.1.1. Histopatologi

Histopologi adalah cabang dari biologi yang mempelajari tentang struktur jaringan secara detail yang melibatkan pemeriksaan mikroskopis tipis, bagian jaringan yang diwarnai dalam rangka untuk mempelajari struktur dan fungsi,

dalam kasus histopatologi, untuk menentukan perubahan yang mungkin disebabkan oleh patogen dan penyakit.

Histopatologi adalah cabang biologi yang mempelajari kondisi dan fungsi jaringan dalam hubungannya dengan penyakit. Histopatologi sangant penting dalam kaitan dengan diagnosis penyakit karena salah satu pertimbangan dalam penegakan diagnosis adalah hasil pengamantan terhadap jaringan yang diduga terganggu. [**2**]

Histopatologi dapat dilakukan dengan mengambil sample jaringan dari bagian ektoparasit seperti insang, sirip, kulit, dan mata, sedangkan endoparasit seperti usus, ginjal, hepatopankreas, dan lain-lain, dapat juga dengan cara mengamati jaringan setelah kematian terjadi. Dengan membandingkan kondisi jaringan sehat terhadap jaringan sampel dapat diketahui apakah suatu penyakit yang diduga benar-benar menyerang atau tidak.



Gambar 2.1 Sample Penyakit Ikan Trichodina sp [1]

2.2. Pengolahan Citra

Pengolahan citra adalah sebuah ilmu yang mempelajari hal-hal yang berkaitan dengan perbaikan kualitas gambar baik itu meningkatkan kontras, transformasi warna, dan restorasi citra, transformasi gambar seperti rotasi, translasi, skala, dan transformasi geometri, proses pemilihan sebuah citra (feature images) yang optimal untuk tujuan analisis, melakukan penarikan informasi atau dekripsi objek atau pengenalan objek yang terkandung dalam citra, melakukan kompresi atau reduksi sebuah data untuk tujuan penyimpanan data, transmisi data, dan waktu proses data. Input dari pengolahan citra adalah citra, sedangkan output-nya adalah citra hasul pengolahan. [**3**]

2.2.1. Citra Digital

Citra digital adalah citra yang dapat diolah oleh komputer. Citra yang disimpan dalam memori komputer hanyalah sebuah angka-angka yang menunjukkan besar intensitas pada masing-masing *pixel* tersebut. Sebuah citra dapat didefinisikan sebagai fungsi dua dimensi f(x,y), yang dimana x dan y adalah korrdinat spasial, dan amplitude dari f pada sembarang pasangan koordinat (x,y) disebut *intensity* (intensitas) atau *gray level* (level keabuan) dari sebuah citra pada titik tertentu. Ketika (x,y) dan nilai intensitas dari f adalah semua terbatas maka itu disebut digital image (citra digital). Citra digital terdiri dari sejumlah elemen tertentu, setiap elemen memiliki lokasi dan nilai tertentu. Elemen-elemen ini disebut *picture elements*, *image elements*, dan *pixels*. Pixel adalah istilah yang sering digunakan secara umum untuk menyatakan elemen pada citra digital [**3**]

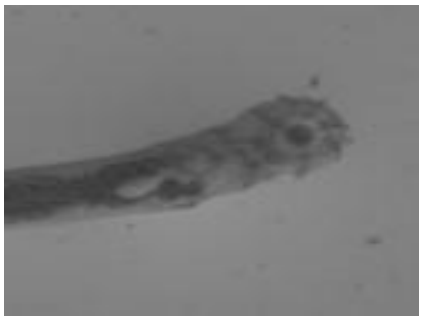
2.2.2. Grayscale

Grayscale adalah citra yang terdiri dari warna hitam, abu-abu, dan putih. Grayscale menunjukkan jumlah warna yang ada dalam satu citra. Makin besar angka *grayscale*, maka citra yang terbentuk makin mendekati kenyataan, biasanya grayscale disebut sebagai gambar hitam putih, dan memiliki 8 bit warna. Untuk menghasilkan warna *grayscale* biasanya nilai warna RGB mod 3.

$$Grayscale = \frac{(R+G+B)}{3} \quad (2.1)$$

Persamaan 2.1 Rumus Grayscale

Pada Gambar 2.2 berikut ini merupakan gambar hasil proses *grayscale*.



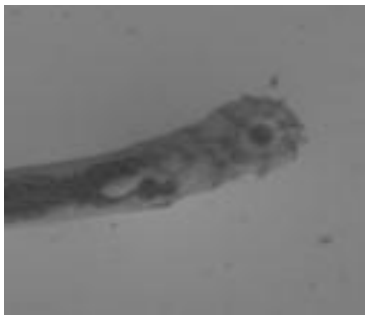
Gambar 2.2 Gambar Hasil Grayscale

2.2.3. Edge Detection Canny

Edge detection canny merupakan algoritma yang digunakan untuk menganalisis suatu citra tepi. Pengembang algoritma *Edge detection canny* adalah John F. Canny pada tahun 1986 dan menggunakan algoritma multi-tahap untuk mendeteksi berbagai tepi pada gambar.

Algoritma Canny berjalan dalam 5 langkah yang terpisah yaitu

a)	Smoothing	:	Menggabungkan gambar untuk menghilangkan noise.
b)	Finding gradien	:	Tepian harus ditandai pada gambar memiliki gradien yang besar.
c)	Non-maksimum-suppresion	:	Hanya maxima lokal yang harus ditandai egde
d)	Double thresholding	:	Tepian yang berpotensi ditentukan oleh thresholding.
e)	Edge tracking by hysteresis	:	Tepian final ditentukan dengan menekan semua sisis yang tidak terhubung dengan tepian yang sangat kuat



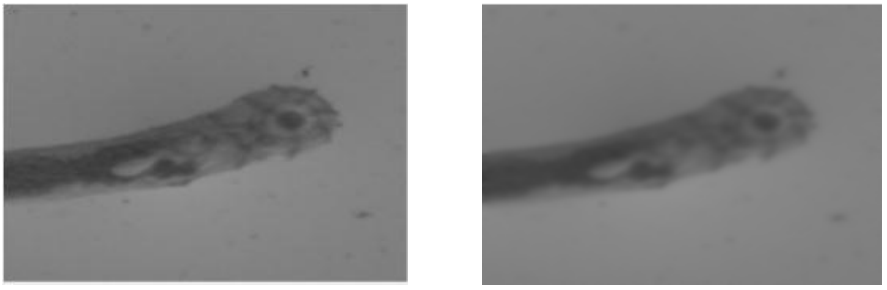
Gambar 2.3 Gambar yang digunakan untuk Edge Detection Canny

Tidak dapat dipungkiri bahwa semua gambar yang diambil akan berisi sejumlah *noise*. Untuk mencegah noise salah dideteksi sebagai tepian, maka noise harus dikurangi. Oleh karena itu pada langkah pertama gambar harus diperhalus dengan menggunakan Gaussian filter. Inti dari *Gaussian filter* adalah standar deviasi dengan σ = 1,4 ditentukan pada persamaan (2.2) di bawah ini.

$$\delta = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \quad (2.2)$$

Persamaan 2.2 Gaussiean filter

Pada persamaan diatas merupakan matriks *Gaussian filter* yang digunakan untuk menghilangkan untuk menghilangkan noise pada gambar.



(a) Original (b) Diperhalus

Gambar 2.4 Gambar Grayscale yang diperhalus

Algoritma Canny pada dasarnya menemukan titik tepi pada gambar grayscale dengan perubahan nilai intensitas yang paling besar, daerah ini

ditemukan dengan menentukan gradien gambar. Gradien pada setiap piksel gambar yang telah diperhalus ditentukan dengan menerapkan operator Sobel. Langkah Kedua adalah memperkirakan gradien pada arah x dan y. Hal tersebut ditunjukkan dalam Persamaan (2.3).

$$M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \; M_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{2.3}$$

Persamaan 2.3 Rumus Operator Sobel

Magnitudo gradien (juga dikenal sebagai kekuatan tepi) dapat ditentukan sebagai jarak Euclidean yang diukur mengukur dengan menerapkan hukum Pythagoras seperti yang ditunjukkan dalam Persamaan (2.4). Yang terkadang disederhanakan dengan menerapkan ukuran jarak Manhattan seperti yang ditunjukkan dalam Persamaan (2.5) untuk mengurangi kompleksitas komputasi.

$$|G| = \sqrt{G_x^3} + \sqrt{G_y^2} \tag{2.4}$$

Persamaan 2.4 Rumus Magnitudo Gradien

$$|G| = |G_x| + \left|G_y\right| \tag{2.5}$$

Persamaan 2.5 Rumus Penyederhanaan Jarak Manhattan

dimana: Gx dan Gy adalah gradien pada masing-masing arah x dan y. Hal ini tampak jelas dari Gambar 3, bahwa gambar dengan gradien yang besar sering menunjukkan tepian yang cukup jelas. Namun, tepian biasanya luas dan dengan demikian tidak dapat menunjukkan persis di mana tepian yang sebenarnya. Untuk menentukan tepian yang sebenarnya ini, arah tepian harus ditentukan dan disimpan seperti ditunjukkan dalam Persamaan (2.6).

$$\theta = arctan\left(\frac{|G_x|}{|G_y|}\right) \quad (2.6)$$

Persamaan 2.6 Rumus Untuk Menentukan Tepian Sobel

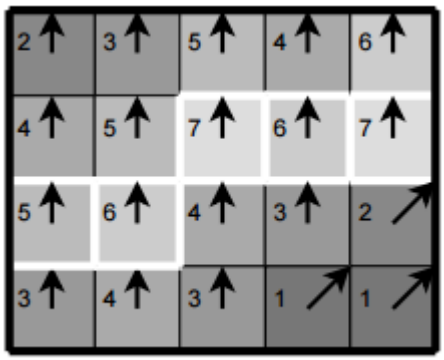


Gambar 2.5 Magnitudo gradien dari penerapan Operator Sobel terhadap gambar yang sudah dihaluskan

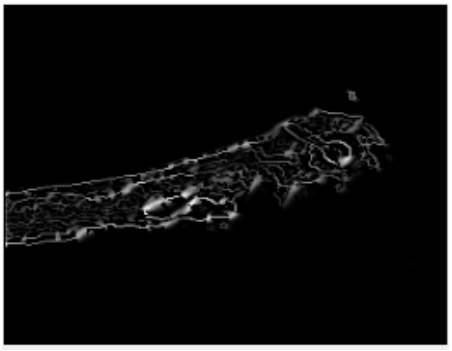
Pada langkah ketiga bertujuan untuk mengkorversikan tepian yang masih blurred pada gambar hasil magnitude gradien hingga menhasilkan tepian yang tajam. Pada dasarnya hal ini dilakukan dengan mempertahankan semua maxima lokal dalam gambar gradien dan menghapus segala sesuatu yang lain. Algoritma adalah untuk setiap piksel pada gambar gradien adalah sebagai berikut:

1. Putar arah gradien θ ke arah 45˚terdekat, kemudian hubungkan dengan 8 titik tetangga yang terhubung dengannya.
2. Bandingkan nilai piksel tepian saat ini dengan nilai piksel tepian dalam arah positif dan negatif gradien. Jika arah gradien adalah utara (θ =90 ◦), bandingkan dengan piksel ke utara dan selatan.
3. Jika nilai piksel tepian saat ini adalah yang terbesar, maka simpan nilai tepian tersebut, namun jika bukan, hapus nilai tersebut.

Sebuah contoh sederhana dari penghapusan non maksimum ditunjukkan pada Gambar 2.6. Hampir semua piksel yang memiliki arah gradien yang menunjukkan arah utara, oleh karena itu mereka dibandingkan dengan piksel atas dan bawah. Piksel yang berubah menjadi maksimal dalam perbandingan ini ditandai dengan warna putih pada perbatasan, sisanya dihapus. Gambar 5 menunjukkan efek pada gambar tes.

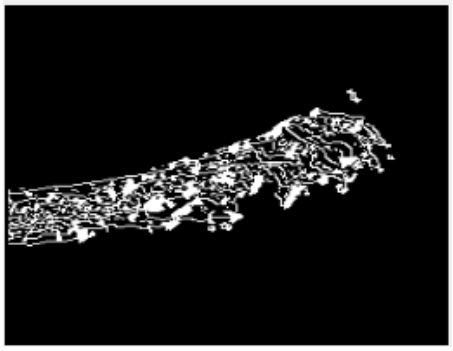


Gambar 2.6 Ilustrasi Penghapusan Non Maksimum



Gambar 2.7 Penghapusan Non Maksimum

Piksel tepian yang tersisa setelah dilakukan penghapusan non maksimum ditandai dengan nilai piksel per piksel yang kuat. Kebanyakan dari titik ini adalah tapian yang nyata pada gambar, akan tetapi beberapa kemungkinan disebabkan oleh noise atau variasi wana karena permukaan yang kasar. Cara paling sederhana untuk membedakannya adalaha menggunakan nilai threshold (ambang batas) sehingga hanya tepian dengan nilai yang kuar yang akan dipertahankan. Disini pada algoritma Canny menggunakan sistem *thresholding* ganda yang dimana *thresholding* tersebut akan dibandingkan dengan *thresolding* yang memiliki nilai yang lebih besar, apabila nilai lebih besar maka nilai thresolding tersebut akan digunakan sebagai acuan menentukan tepian dan nilai thresolding yang kecil akan dihilangkan. [**4**].



Gambar 2.8 Hasil akhir Edge Detection Canny

2.2.4. Normalisasi

Normalisasi merupakan proses merubah dimensi region tiap karakter. Pada proses ini menggunakan proses *Thresholding*, merupakan salah satu operasi yang termasuk ke dalam operasi titik dalam pengolahan citra digital. Operasi ini digunakan untuk mengubah citra dengan format skala keabuan yang dimiliki kemungkinan nilai lebih dari 2 ke citra biner yang memiliki 2 buah nilai (0 dan 1).

Thresholding terdirri dari dua jenis, yaitu:

a. *Thresholding* Tunggal

Thresholding Tunggal merupakan prses *thresholding* yang hanya memiliki sebuah nilai ambang batas atau batasan suatu citra.

$$K_0 \begin{cases} 0, jika\ K_i < ambang\ batas\ citra \\ (0 = hitam) \\ 1, jika\ K_i \geq ambang\ batas\ citra \\ (1 = putih) \end{cases}$$

atau

$$K_0 \begin{cases} 0, jika\ K_i \geq ambang\ batas\ citra \\ (0 = hitam) \\ 1, jika\ K_i < ambang\ batas\ citra \\ (1 = putih) \end{cases}$$

Persamaan 2.7 Fungsi Thresholding Tunggal

b. *Thresholding* Ganda

Meiliki ambang batas atas dan batas bawah. Dilakukan untuk menampilkan titik-titik yang memiliki rentang nilai skala keabuan tertentu.

$$K_0 \begin{cases} 0, jika\ ambang\ bawah \leq K_i \leq ambang\ atas \\ 1, lainnya \end{cases}$$

atau

$$K_0 \begin{cases} 0, jika\ ambang\ bawah \leq K_i \leq ambang\ atas \\ 1, lainnya \end{cases}$$

Persamaan 2.8 Fungsi Thresholding Ganda

2.2.5. Ekstraksi fitur

Ektraksi fitur atau featur extraction merupakan suatu pengambilan ciri / feature dari suatu bentuk yang nantinya nilai yang didapatkan akan dianalisis untuk proses selanjutnya. featur extraction dilakukan dengan cara menghitung jumlah titik atau pixel yang ditemukan dalam setiap pengecekan, dimana pengecekan dilakukan dalam berbagai arah tracing pengecekan pada kordinat kartesial dari citra digital yang dianalisis.

Ektraksi fitur terbagi kedalam tiga macam, yaitu :

a. Ektraksi fitur bentuk

   Bentuk dari suatu objek adalah karakter konfigurasi permukaan yang diwakili oleh garis dan kontur. Fitur bentuk dikategorikan bergantung pada teknik yang digunakan. Kategori tersebut adalah berdasarkan batas (*boundary-based*) dan berdasarkan daerah (*region-based*). Teknik berdasarkan batas (*boundary-based*) menggambarkan bentuk daerah dengan menggunakan karakteristik eksternal, contohnya adalah piksel sepanjang batas objek. Sedangkan teknik berdasarkan daerah (*region-based*) menggambarkan bentuk wilayah dengan menggunakan karakteristik internal, contohnya adalah piksel yang berada dalam suatu wilayah

b. Ektraksi fitur tekstur

   Pada ekstraksi fitur ini, fitur pembeda adalah tekstur yang merupakan karakteristik penentu pada citra. Teknik statistik yang terkenal untuk ekstraksi fitur adalah matriks *gray level co-occurrence.* Teknik tersebut dilakukan dengan melakukan pemindaian untuk mencari jejak derajat keabuan setiap dua buah piksel yang dipisahkan dengan jarak d dan

sudut θ yang tetap. Biasanya sudut yang digunakan adalah 0°, 45°, 90°, dan 135°.

c. Ektraksi fitur warna

Pada ekstraksi fitur warna, ciri pembeda adalah warna. Biasanya ekstraksi fitur ini digunakan pada citra berwarna yang memiliki komposisi warna RGB (*red, green, blue*)

2.3. Kecerdasan Buatan

Kecerdasan Buatan (*Artificial Intelegent*) merupakan cabang dari ilmu komputer yang dalam merepresentasikan sebuah pengetahuan yang lebih banyak menggunakan sebuah bentuk seperti simbol-simbol dari pada menggunakan bilangan, dan memproses sebuah informasi berdasarkan metode heuristic atau dengan berdasarkan berbagai macam aturan.

2.3.1. Kecerdasan Buatan Dalam Beberapa Aspek

Dari aspek Kecerdasan (*Intelligence*) AI merupakan proses pembuatan mesin yang memiliki pemikiran sendiri atau "cerdas" dan dapat melakukan berbagai hal yang sebelumnya dapat dilakukan oleh manusia. Dari aspek bisnis, Ai adalah sekelompok alat bantu yang memiliki berbagai macam fungsi dalam menyelesaikan berbagai permasalahan dalam dunia bisnis. Dari aspek pemrograman (*Programing*), AI merupakan bagian dari sistem itu sendiri agar dapat menjalankan sistem yang sudah dirancang. [6]

2.3.2. Beberapa Penelitian Dalam Kecerdasan Buatan

Adapun penelitian dalam kecerdasan buatan diantaranya yakni :

1. Permainan (*Game*)

Kebanyakan permainnan dilakukan dengan menggunakan sekumpulan aturan. Dalam permainana digunakan apa yang disebut dengan *heuristic*. Permainan merupakan bidang yang menarik dalam studi *heuristic*.

2. Natural Language

Merupakan suatu teknologi yang memberikan kemampuan kepada computer untuk memahami bahasa manusia sehingga pengguna computer dapat berkomunikasi dengan komputer menggunakan bahasa sehari-hari. [**5**]

3. Robotik

Robotic yang bila digabungkan dengan sistem sensor, seperti sistem vision, sistem *tactile*, dan sistem pemrosesan sinyal yang jika digabungkan dengan *Artificial Intelegent* (AI), akan dapat dikategorikan menjadi suatu sistem yang sangat luas yang dapat disebut dengan sistem robotic. [**6**]

4. Sistem Pakar (*Expert System*)

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

2.4. Markov Chains

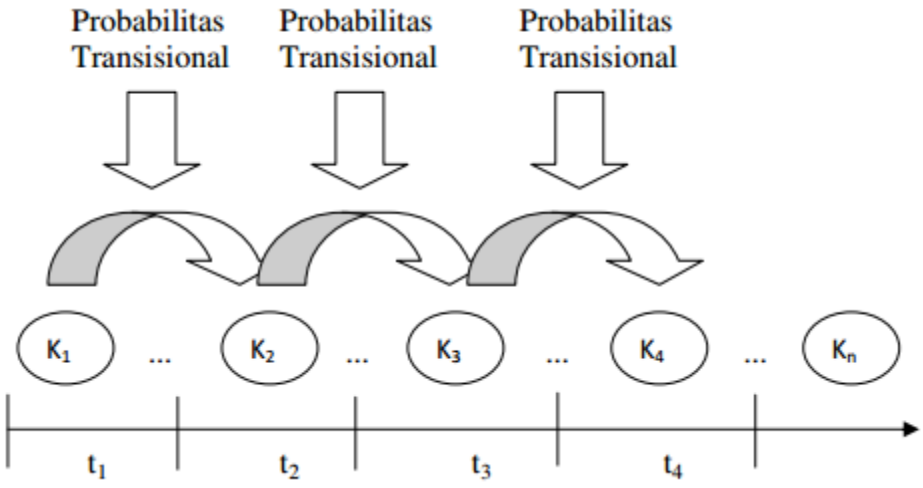
Rantai Markov *(Markov Chains)* adalah merupakan suatu teknik matematika yang biasa digunakan untuk melakukan pemodelan (modeling) berbagai macam sistem dan proses bisnis. Teknik ini dapat digunakan untuk memperkirakan perubahan-perubahan di waktu yang akan datang dalam variabel-variabel dinamis atas dasar perubahan-perubahan dari variabel-variabel dinamis tersebut di waktu yang lalau. Teknik ini dapat digunakan juga untuk menganalisis kejadia-kejadian di waktu-waktu mendatang secara matematis.

Model Rantai Markov *(Markov Chains)* ditemukan oleh seorang ahli Rusia yang bernama Andrey Markov pada tahun 1906, yaitu: "*Untuk setiap waktu t, ketika kejadian adalah* K_t *dan seluruh kejadian sebelumnya adalah* $K_{t(j)}$*… ,* $K_{t(j-n)}$ *yang terjadi dari proses yang diketahui, probabilitas seluruh kejadian*

yang akan datang Kt(j) hanya bergantung pada kejadian $K_{t(j-1)}$*dan tidak bergantung pada kejadian-kejadian sebelumnya yaitu* $K_{t(j-2)}$, $K_{t(j-3)}$,..., $K_{t(j-n)}$.*".*

Rantai Markov *(Markov Chains)* apabia diberikan inputan keadaan dari keadaan saat ini, keadaan akan datang dapat diprediksi dan dapat di lepas dari keadaan di masa lampau. Artinya, deskripsi kondisi saat ini menangkap semua informasi yang mempengaruhi evolusi dari suatu sistem dimasa depan. Dengan kata lain, kondisi masa depan dituju dengan menggunakan prbabilitas bukan dengan determinitas.

Gambaran mengenai rantai Markov ini kemudian dituangkan dalam Gambar 2.9 dimana gerakan-gerakan dari beberapa variabel di masa yang akan datang bisa diprediksi berdasarkan gerakan-gerakan variabel tersebut pada masa lalu. K_{t4} dipengaruhi oleh kejadian K_{t3}, K_{t3} dipengaruhi oleh kejadian K_{t2} dan demikian seterusnya dimana perubahan ini terjadi karena peranan probabilitas transisi (transition probability). Kejadian K_{t2} misalnya, tidak akan mempengaruhi kejadian K_{t4}.



Gambar 2.9 Markoc Chains

Rantai Markov *(Markov Chains)* bermanfaat untuk menghitung probabilitas suatu kejadian teramati yang secra umum dapat dirumuskan sebagai berikut:

$$P(\sigma_i) = P(\sigma_t|\sigma_{t-1}, \sigma_{t-2}, \sigma_{t-3}, \dots, \sigma_{t-n})$$

σ_t adalah kondisi saat ini, dan σ_t adalah kondisi pada waktu tertentu yang berhubungan dengan σ_t. Sedangkan σ_{t-i} adalah kondisi sebelum σ_t. Kemudian dapat diasumsikan bahwa sebelah kanan persamaan bersifat invariant, yaitu, dihipotesiskan dalam keseluruahan sistem, transisi diantara keadaan tertentu tetap sama dalam hubungan probabilistiknya. Berdasarkan asumsi tersebut, dapat terbentuknya suatu stet keadaan probabilitas transisi diantara dua keadaan K_i dan t_j :

$$Probabilitas\ Transisi = P\big(\sigma_t = K_t\big|\sigma_{t-1} = t_j\big), 1 \leq i, j \leq N$$

2.5. Hidden Markov Model (HMM)

Hidden Markov Model (HMM) adalah suatu model probabilitas yang menggambarkan hubungan statistik antara urutan observasi O dan urutan state S yang tidak diobservasi ”hidden”. [7]

Ciri-ciri HMM adalah sebagai berikut:

a) Observasi diketahui tetapi urutan keadaan (*state*) tidak diketahui sehingga disebut *hidden.*
b) Observasi adalah fungsi probabilitas keadaan.
c) Perpindahan keadaan adalah dalam bentuk probabilitas.

2.5.1. Parameter pada HMM

HMM mempunyai parameter-parameter distribusi sebagai berikut :

a. Probabilitas Transisi (A)

Parameter A merupakan probabilitas kedudukan suatu state terhadap semua state yang ada, termasuk kedudukan terhadap state itu sendiri. parameter transisi dapat dituliskan dalam bentuk matriks seperti berikut:

$$A_{ij} = \begin{bmatrix} A_{11} & A_{12} & A_{13} & \cdots & A_{1n} \\ A_{21} & A_{22} & A_{23} & \cdots & A_{2n} \\ A_{31} & A_{32} & A_{33} & \cdots & A_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & A_{n3} & \cdots & A_{nn} \end{bmatrix} \quad (2.7)$$

Persamaan 2.9 Probabilitas Transisi

b. Probabilitas State (B)

Parameter B merupakan probabilitas state atau probabilitas observasi merupakan kemunculan suatu state deretan seluruh state yang ada.. Parameter B dapat dituliskan dalam bentuk matriks seperti berikut :

$$B = \begin{bmatrix} b1 \\ b2 \\ b3 \\ \vdots \\ bn \end{bmatrix} \quad (2.8)$$

Persamaan 2.10 Probabilitas State

c. Distribusi keadaan awal (π)

Parameter phi (π), disebut sebagai parameter awal, merupakan probabilitas kemunculan suatu state di awal. Sama halnya dengan parameter B, parameter π dapat dituliskan dalam bentuk seperti berikut :

$$\pi = \begin{bmatrix} \pi 1 \\ \pi 2 \\ \pi 3 \\ \vdots \\ \pi n \end{bmatrix} \quad (2.9)$$

Persamaan 2.11 Probabilitas Phi ($\boldsymbol{\pi}$)

Sedangkan parameter tertentu HMM ada dua yaitu N dan M:

d. N, jumlah keadaan model. Dinotasikan himpunan terbatas untuk keadaan yang mungkin adalah $Q = \{q_1, \dots, q_N\}$ (2.10)

e. M, jumlah dari simbol observasi/keadaan, ukuran huruf diskret. Simbol observasi berhubungan dengan keluaran fisik dari sistem yang dimodelkan. Dinotasikan himpunan terbatas untuk observasi yang mungkin adalah

$$V = \{V_1, \dots, V_M\} \quad (2.11)$$

Parameter-parameter HMM ditaksir berdasarkan kriteria *maximum likelihood* (ML) dan algoritma *Baum-Welch* (EM = Expectation Modification). [**7**]

2.5.2. Permasalahan pada HMM

Ada 3 masalah dasar HMM yang harus dipecahkan untuk model yang diterapkan di dunia nyata, yaitu [10]:

1. Menghitung $P = (O|\lambda)$ bila diberikan urutan observasi $O = O_1, O_2, \dots, O_T$ dan $\lambda = (A, B, \pi)$.

   Solusi: Cara umum yang biasa digunakan adalah dengan memeriksa setiap kemungkinan urutan *N state* sepanjang *T* (banyaknya observasi). Hal ini tidak mungkin dilakukan karena perhitungannya kurang efisien. Ada prosedur lain yang lebih sederhana dan efisien adalah menggunakan prosedur *forward* dan *backward*.

   a. Prosedur Maju (*forward procedure*)

      Asumsi variabel *forward* $\alpha_t(i)$ pada waktu ke-*t* dan *state* ke-*i* didefinisikan dengan. $\alpha_t(i) = P(O_1, O_2, \dots, O_T, q_t = i \mid \lambda)$. Fungsi peluang *forward* dapat diselesaikan untuk *N state* dan *T* simbol observasi secara induktif dengan langkah sebagai berikut:

      a) Inisialisasi, $\alpha_t(i) = \pi_i b_i(O_1), \quad 1 \le i \le N$ (2.12)

      b) Induksi, $\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i)\alpha_{ij}\right] b_j(O_{i+1})$,

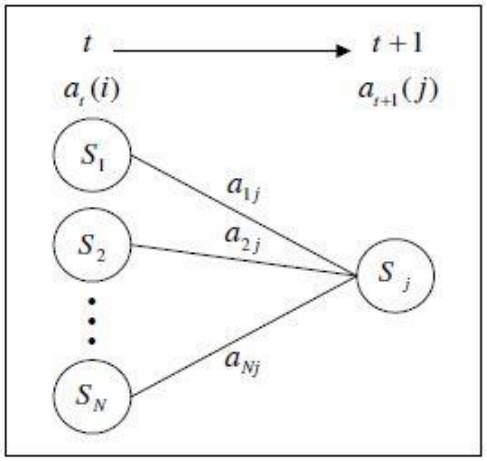
      $1 \le t \le T\text{-}1;\ 1 \le j \le N$ (2.13)

      c) Terminasi, $P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$ (2.14)

      Perhitungan peluang *forward* berdasarkan +pola diagram *trellis*. Terdapat *N* titik tiap *time slot* pada pola, semua kemungkinan deretan *state* digabung sebanyak *N* titik tanpa memperdulikan panjang deretan

observasi. Pada saat t=1, dihitung nilai $O_1(i)$, $1 \leq i \leq N$. Pada waktu t = 2,3,...,T hanya diperlukan perhitungan nilai $\alpha_t(j)$ dimana $1 \leq j \leq N$.

Tiap perhitungan membutuhkan nilai sebelumnya sebanyak N dari $\alpha_{t-1}(i)$ karena tiap N titik hanya dapat dihubungkan dengan N titik pada *time slot* sebelumnya.



Gambar 2.10 Induksi Prosedur Forward

b. Prosedur Mundur (*backward procedure*)

Dengan cara yang sama, variabel *backward* $\beta_t(i)$ pada waktu ke-t dan *state* ke-i didefinisikan dengan $\beta_t(i) = P(O_{t+1}, O_{t+2}, ..., O_T, qt = i \mid \lambda)$. Prosedur *backward* digunakan untuk menghitung $P(O|\lambda)$ secara mundur dari waktu observasi maksimum, t=T sampai waktu observasi awal t=1. Langkah prosedur *backward* adalah sebagai berikut:

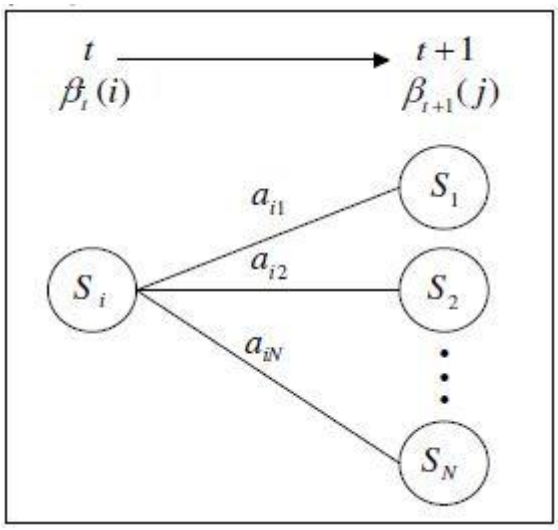
a) Inisialisasi, $\beta_T(i) = 1$; $1 \leq i \leq N$ (2.15)

b) Induksi, $\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_i + 1)\, \beta_t + 1(j)$,

$t = T-1, T-2, ..., 1$ (2.16)

Pada inisialisasi didefinisikan $\beta_T(i)$ =1 untuk semua nilai i. Untuk mendapatkan *state* ke-i waktu ke-t dan deretan observasi pada waktu t+1, maka diasumsikan semua kemungkinan *state* j pada waktu t+1, untuk

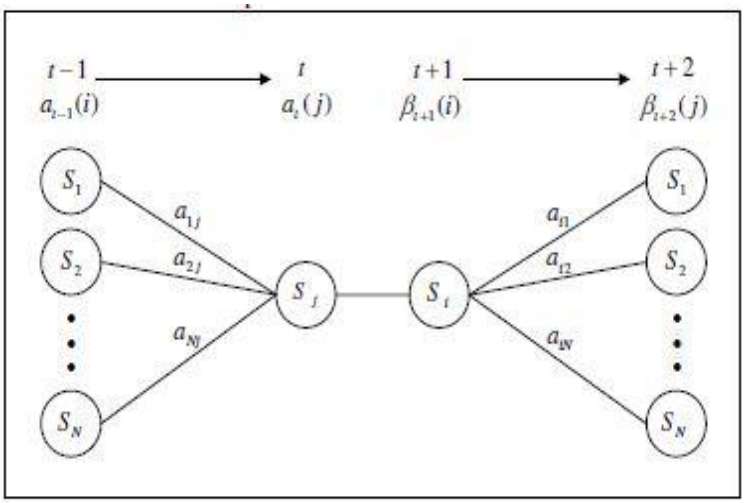
mendapatkan transisi dari *i* ke *j*, a_{ij} dan deretan observasi o_{t+1} pada *state* ke-*j*, $b_j(O_{t+1})$. Kemudian dihitung bagian observasi dari *state j*, $\beta_{t+1}(j)$.



Gambar 2.11 Induksi Prosedur Backward

c. Prosedur Maju – Mundur (*forward – backward procedure*)

Kombinasi dari prosedur *forward* dan *backward* dapat digunakan untuk mendapatkan nilai $P(O|\lambda)$. Peluang berada pada *state* ke-*i* waktu ke-*t* dari *N state* sebelumnya waktu *t*-1 dapat dihitung dengan fungsi peluang *forward* $\alpha_t(i)$. Sedangkan peluang transisi dari *state i* waktu ke-*t* menuju *state j* waktu *t*+1 dan mengambil sebuah simbol observasi pada *state j* adalah $a_{ij}b_j(O_{t+1})$. Fungsi peluang *backward* digunakan untuk menghitung peluang munculnya deretan simbol observasi dari waktu *t*+1 sampai *T*.



Gambar 2.12 Ilustrasi Prosedur Forward-Backward

Secara matematis perhitungan $P(O|\ \lambda)$ menggunakan prosedur *forward-backward* dapat dinyatakan sebagai berikut:

$$P(O|\lambda)= \left[\sum_{i=1}^{N}\left[\sum_{j=1}^{N}\alpha_1(i)a_{ij}.\,b_j(O_i+1)\right]\beta_t+1(j)\right]$$

$$P(O|\lambda)= \sum_{i=1}^{N}\alpha_t(i)\,\beta_t(i) \qquad (2.17)$$

Keterangan :

$\alpha_t(i)$: merupakan hasil dari observasi prosedur *forward* tahap terminasi.

$\beta_t(i)$: merupakan hasil dari observasi prosedur *backward* tahap terminasi.

2. Memilih urutan keadaan yang paling optimal yang berhubungan dengan Q = q_1,q_2 ... q_T bila diberikan urutan observasi O = O_1,O_2, ...,O_T dan sebuah model $\lambda = (A, B, \pi)$.
   Solusi: Inti dari permasalahan 2 adalah mencari deretan *state* yang tersembunyi (*hidden*) untuk suatu deretan observasi yang dihasilkan dari model λ. Deretan *state* yang dicari harus merupakan deretan yang optimal agar dapat dimodelkan pada deretan obervasi dari model λ. Metode yang biasa digunakan untuk menemukan deretan *state* optimal adalah algoritma *viterbi* (*dynamic programming*). Algoritma *viterbi* dapat memaksimalkan nilai $P(O|\ \lambda)$ sehingga memberikan deretan observasi yang optimal. Berdasarkan aturan Bayes, secara matematis $P(Q|O, \lambda)$ dapat dinyatakan sebagai berikut:

$$P(O|\lambda)= \frac{P(Q,O|\lambda)}{P(O|\lambda)} \qquad (2.18)$$

Pada algoritma *viterbi*, peluang maksimum berada pada *state* ke-*i* waktu ke-*t*, yang sebelumnya telah menempuh deretan *state* *q*={*q1*, *q2*, ..., *qt-1*}, jika diberikan deretan observasi *O* dan suatu model λ dinotasikan dengan $\delta_t(i)$.

$$\delta t\,(i) = \max\ P[q_1 q_2 ... q_{t-1}, q_t = i, O_1 O_2 ... O_T \mid \lambda] \quad (2.19)$$

Untuk mendapatkan deretan observasi dibutuhkan urutan dari argumen yang maksimal untuk tiap *t* dan *j*. Hal ini dilakukan melalui sebuah fungsi *array* $\psi_t(j)$.Prosedur selengkapnya dari algoritma viterbi adalah sebagai berikut:

1) Inisialisasi, $\delta_1(i) = \pi_i\ b_i\ (o_1),\ 1 \leq i \leq N;\ \psi_1(i) = 0$ (2.20)

2) Rekursi, $\delta_1(j) = \max_{1 \leq i \leq N} \left[\delta_{t-1}(i) a_{ij}\right] b_j(o_t),\ 2 \leq t \leq T;\ 1 \leq j \leq N$

$\psi_t(j) = \arg\max_{1 \leq i \leq N} \left[\delta_{t-1}(i) a_{ij}\right],\ 2 \leq t \leq T;\ 1 \leq j \leq N$ (2.21)

3) Terminasi, $P^* = \max_{1 \leq i \leq N} [\delta_T(i)]$

$q^*_t = \arg\max_{1 \leq i \leq N} [\delta_T(i)]$ (2.22)

Backtracking, $q^*_t = \psi_{t+1}(q *_{t+1}),\ t = T-1, T-2, \ldots 1$ (2.23)

2.6. Pemrograman Desktop

Pemograman desktop merupakan pemograman yang dilakukan dengan memanipulasi elemen – elemen visual yang dilakukan pada sebuah PC tunggal yang pengoperasiannya tidak bergantung pada PC lain dalam jaringan maupun web. Pemograman desktop dibagi 3, yaitu .Net, Java and Delphi. Bahasa pemograman .Net yaitu Visual Basic (VB), C++ dan C#. Pemograman desktop dapat digunakan tanpa perlu terhubung ke internet (online). Tetapi biasanya harus diinstall terlebih dahulu sebelum digunakan, dan ini memakan space harddisk.

2.7. Bahasa Pemrograman C#

Bahasa pemograman yang dipakai dalam membuat aplikasi ini adalah C#. C#(baca : See-Sharp) adalah bahasa pemograman baru yang diciptkan Microsoft yang digunakan oleh banyak developer .NET untuk mengembangkan aplikasi

dengan platform .NET. C# ini merupakan bahasa pemograman yang sederhana dan simple. C# disebut sederhana dan simple, karna bahasa ini didasarkan kepada bahasa C dan C++. C# juga merupakan Object Oriented Language, dan bisa digunakan untuk membuat berbagai macam aplikasi, pengolahan kata, grafik, spreadsheets, atau bahkan membuat compiler untuk sebuah bahasa pemograman.

2.8. OOP (*Object Oriented Programming*)

Secara spesifik, pengertian berorientasi objek berarti bahwa mengorganisasi perangkat lunak sebagai kumpulan dari objek tertentu yang memiliki struktur data dan perilakunya. Hal ini yang membedakan dengan pemograman konvensional dimana struktur data dan perilaku hanya berhubungan secara terpisah. Terdapat beberapa cara untuk menentukan karakteristik dalam pendekatan berorientasi objek, tetapi secara umum mencakup empat hal, yaitu identifikasi, klasifikasi, polymorphism (polimorfisme) dan inheritance (pewarisan)

2.8.1. Konsep Dasar OOP

Pendekatan berorientasi objek merupakan suatu teknik atau cara pendekatan dalam melihat permasalahan dan sistem (sistem perangkat lunak, sistem informasi atau sistem lainnya). Pendekatan berorientasi objek akan memandang sistem yang akan dikembangkan sebagai suatu kumpulan objek yang berkorespodensi dengan objek-objek dunia nyata. Berikut ini adalah beberapa konsep dasar yang harus dipahami tentang metodologi berorientasi objek

1. Kelas (Class)

   Kelas adalah kumpulan objek-objek dengan karakteristik yang sama. Kelas merupakan definisi statis dan himpunan objek yang sama yang mungkin lahir atau tercipta dan kelas tersebut. Sebuah kelas akan mempunyai sifat (atribut), kelakuan (operasi/metode), hubungan (relationship) dan arti. Suatu kelas dapat diwariskan ke kelas yang baru.
2. Objek (Object)

Objek adalah abstraksi dan sesuatu yang mewakilkan dunia nyata seperti benda, manusia, satuan organisasi, tempat, kejadian, struktur, status atau halhal lain yang bersifat abstrak. Objek merupakan entitas yang mampu menyimpan informasi (status) dan mempunyai operasi (kelakukan) yang dapat diterapkan atau dapat berpengaruh pada status objeknya. Objek mempunyai siklus hidup yang diciptakan, dimanipulasi, dan dihancurkan.

3. Metode (Method)

   Operasi atau metode pada sebuah kelas hampir sama dengan fungsi prosedur pada metodologi struktural. Sebuah kelas boleh memiliki lebih dari satu metode atau operasi yang befungsi untuk memanipulasi objek itu sendiri. Operasi atau metode merupakan fungsi atau transformasi yang dapat dilakukan terhadap objek atau dilakukan oleh objek.

4. Atribut (Attribute)

   Atribut dari sebuah kelas adalah variable global yang dimiliki sebuah kelas. Atribut dapat berupa nilai atau elemen-elemen data yang dimiliki oleh objek dalam kelas objek. Atribut dipunyai secara individual oleh sebuah objek, misalnya berat, jenis, nama dan sebagainya.

5. Abstraksi (Absraction)

   Prinsip untuk merepresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan.

6. Enkapsulasi (Encapsulation)

   Pembungkusan atribut data dan layanan (operasi-operasi) yang dipunyai objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerjanya.

7. Pewarisan (Inheritance)

   Mekanisme yang memungkinkan suatu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dari dirinya.

8. Antarmuka (Interface)

   Antarmuka (interface) sangat mirip dengan kelas, tapi tanpa atribut kelas dan memiliki metode yang dideklarasikan tanpa isi. Deklarasi metode pada sebuah interface dapat diimplementasikan oleh kelas lain.

9. Reusability

   Pemanfaatan kembali objek yang sudah didefinisikan untuk suatu permasalahan pada permasalahan lainnya yang melibatkan objek tersebut.

10. Generalisasi dan Spesialisasi

    Menunjukan hubungan antara kelas dan objek yang umum dengan kelas dan objek yang khusus. Misalnya kelas yang lebih umum (generalisasi) adalah kendaraan darat dan kelas khususnya (spesialisasi) adalah mobil, motor, dan kereta

11. Komunikasi Antar Objek

    Komunikasi Antar Objek dilakukan lewat pesan (message) yang dikirim dari suatu objek ke objek yang lain.

12. Polimorfisme (Polymorphism)

    Kemampuan suatu objek untuk digunakan di banyak tujuan yang berbeda dengan nama yang sama sehingga menghemat baris program.

13. Package

    Package adalah sebuah container atau kemasan yang dapat digunakan untuk mengelompokkan kelas-kelas sehingga memungkinkan beberapa kelas yang bernama sama disimpan dalam package yang berbeda.

2.8.2. Analisis Berorientasi Objek

Analisis berorientasi objek atau Object Oriented Analysis (OOA) adalah tahap untuk menganalisis spesifikasi atau kebutuhan akan sistem yang akan dibangun dengan konsep berorientasi objek, apakah benar kebutuhan yang ada dapat diimplementasikan menjadi sebuah sistem berorientasi objek. OOA biasanya menggunakan kartu CRC (Component, Responsibility, Collaborator) untuk membangun kelas-kelas yang akan digunakan atau menggunakan UML

(Unifed Modeling Language) pada bagian diagram use case, diagram kelas, dan diagram objek. [**8**]

2.8.3. Desain Beroroentasi Objek

Desain berorientasi objek atau Object Oriented Design (OOD) adalah tahapan perantara untuk spesifikasi atau kebutuhan sistem yang akan dibangun dengan konsep berorientasi objek ke desain pemodelan agar lebih mudah diimplementasikan dengan pemograman berorientasikan objek. Pemodelan berorientasi objek biasanya dituangkan dalam dokumentasi perangkat lunak dengan menggunakan perangkat lunak dengan menggunakan perangkat pemodelan berorientasi objek, diantaranya adalah UML (Unified Modeling Language). Kendala dan permasalahan pembangunan sistem berorientasi objek biasanya dapat dikenali dalam tahap ini. [**8**]

2.8.4. Pemodelan

Pemodelan adalah gambaran dari realita yang sederhana dan dituangkan dalam bentuk pemetaan dengan aturan tertentu. Pemodelan dapat menggunakan bentuk yang sama dengan realitas. Pemodelan juga banyak digunakan untuk merencanakan suatu hal agar kegagalan dan resiko yang mungkin terjadi dapat meminimalisir. Pada dunia pembangunan perangkat lunak juga diperlunakan pemodelan. Pemodelan perangkat lunak digunakan untuk mempermudah langkah berikutnya dari pengembangan sebuah sistem sehingga lebih terencana. Pemodelan pada pembangunan perangkat lunak untuk menvisualkan perangkat lunak yang akan dibuat. [**8**]

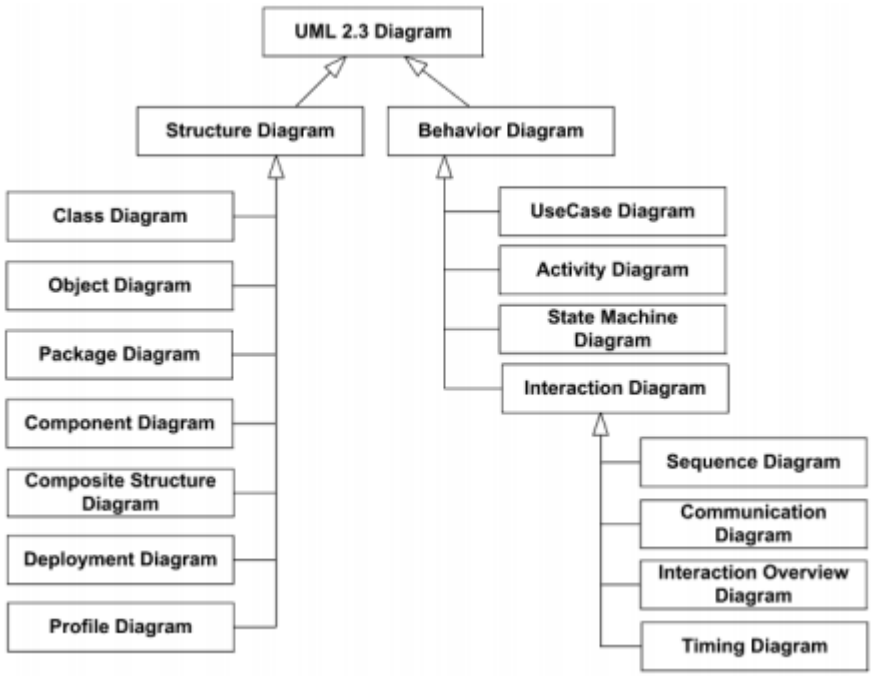
2.9. UML (*Unified Modeling Language)*

UML *(Unified Modelling Language)* adalah sebuah “bahasa” yang telah menjadi standar dalam industri visualisasi, merancang, dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. UML dapat digunakan untuk membuat model untuk semua jenis perangkat lunak, dimana perangkat lunak tersebut dapat berjalan pada piranti

keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan class dan operation dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa berorientasi objek. Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan syntax. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentum, dan UML syntax mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya yaitu : Grady Booch OOD (Object Oriented Design), Jim Rumbaugh OMT (Object Modeling Technique) dan Ivar Jacobson OOSE (Object Oriented Software Engineering). [**9**]

2.9.1. Diagram UML

Pada UML 2.3 terdiri dari 13 macam diagram yang dikelompokan dalam 3 kategori. Pembagian kategori dan macam-macam diagram tersebut dapat dilihat pada Gambar 2.13



Gambar 2.13 Gambar UML

Berikut ini penjelasan singkat dari pembagian kategori tersebut. [**9**]

1. Structure diagrams yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
2. Behavior diagrams yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.
3. Interaction diagrams yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antarsubsistem pada suatu sistem.

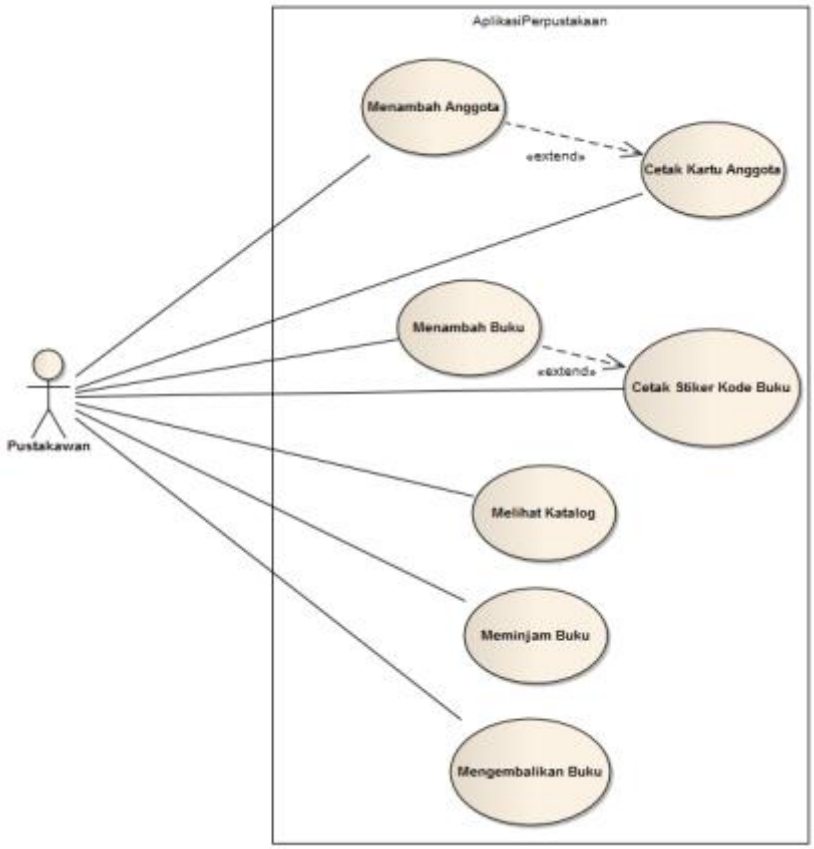
2.9.2. Diagram Use Case (*Use Case Diagram*)

Diagram use case merupakan pemodelan untuk kelakuan (behaviour) sistem informasi yang akan dibuat. Use case mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, use case digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi

itu. Syarat penamaan pada use case adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada use case yaitu pendefinisian apa yang disebut aktor dan use case. [**9**]

1. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
2. Use case merupakan fungsionalitas yang disediakan sistem sebagai unitunit yang saling bertukar pesan antar unit atau aktor

Pada Gambar 2.14 berikut ini merupakan contoh use case diagram dalam pembuatan sistem informasi perpustakaan.



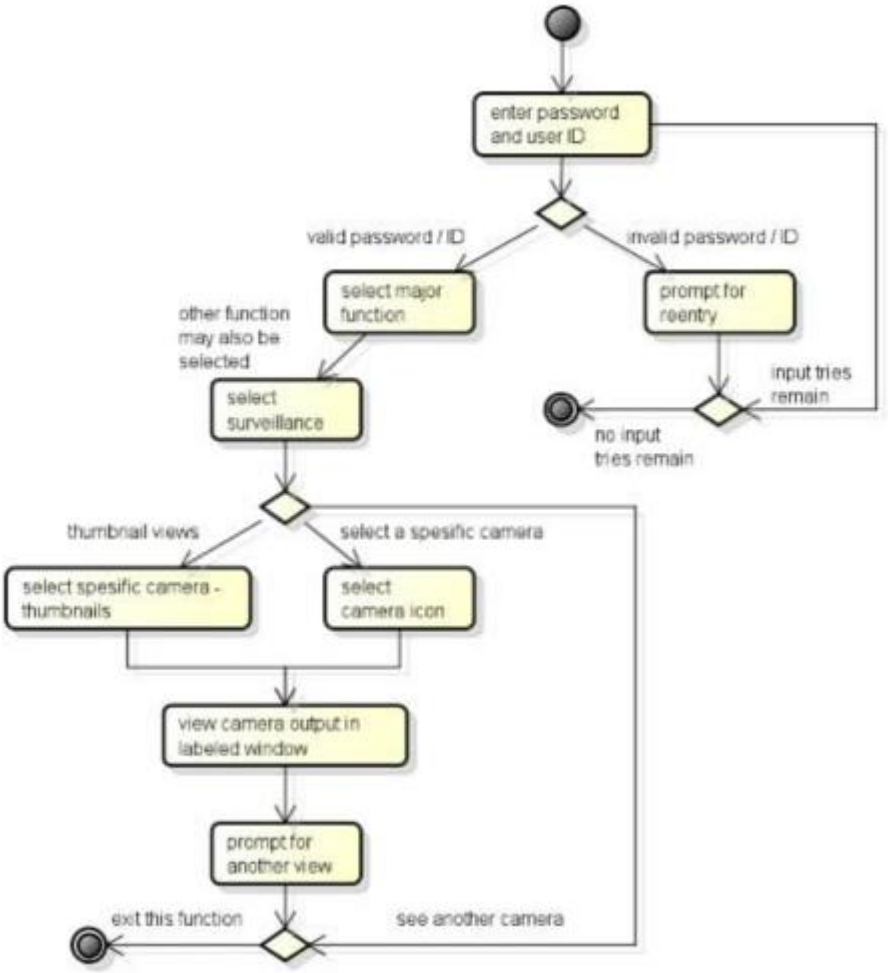
Gambar 2.14 Use Case Diagram

2.9.3. Diagram Aktivitas (*Activity Diagram*)

Diagram aktivitas menggambarkan workflow (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem. Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut:

1. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
2. Urutan atau pengelompokkan tampilan dari sistem/user interface dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
3. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.

Rancangan menu yang ditampilkan pada perangkat lunak. Contoh Activity Diagram dapat dilihat pada Gambar 2.15.

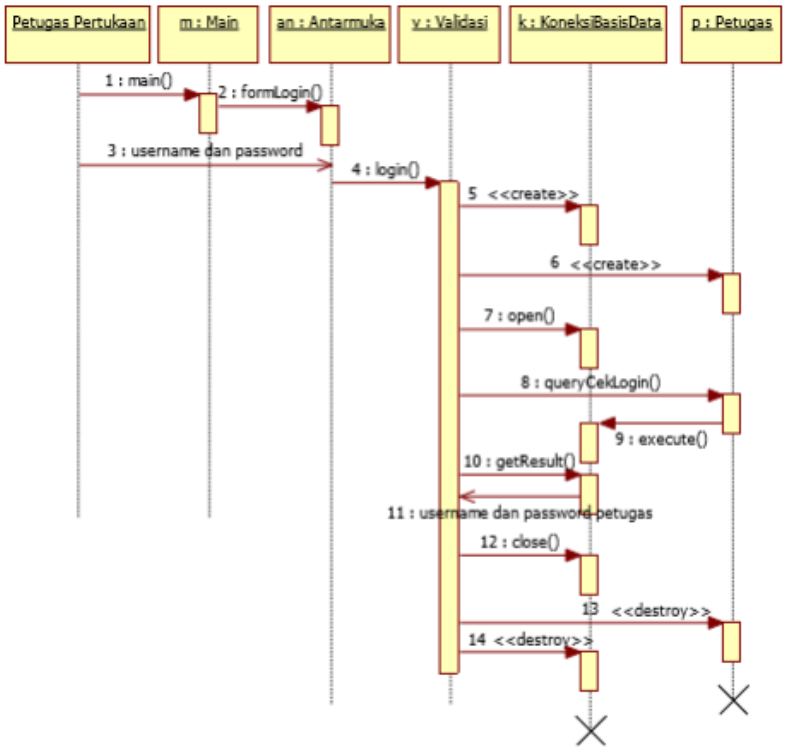


Gambar 2.15 Contoh Activity Diagram

2.9.4. Diagram Sekuen (*Sequence Diagram*)

Diagram sekuen menggambarkan kelakuan pada objek use case dengan mendeskripsikan waktu hidup objek dan message yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambar diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah use case beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu. Membuat diagram sekuen juga dibutuhkan untuk melihat skenario yang ada pada use case. Banyaknya diagram sekuen yang harus digambar adalah minimal sebanyak pendefinisian use case yang memiliki proses sendiri atau yang penting semua use case yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak use case yang didefinisikan maka

diagram sekuen yang harus dibuat juga semakin banyak. Contoh Sequence Diagram dapat dilihat pada Gambar 2.16. [**9**]



Gambar 2.16 Contoh Sequence Diagram

2.9.5. Diagram Kelas (*Class Diagram*)

Diagram kelas atau class diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas mimiliki apa yang disebut atribut dan metode atau operasi. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas. Operasi atau metode adalah fungsifungsi yang dimiliki oleh suatu kelas. Contoh Class Diagram dapat dilihat pada Gambar 2.17



Gambar 2.17 Contoh Class Diagram

Syarat penamaan pada use case adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada use case yaitu pendefinisian apa yang disebut aktor dan use case. [**9**]

1. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem yang akan dibuat diluar sistem yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
2. Use case merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

2.10. Pengujian

Pengujian adalah suatu proses mengevaluasi suatu sistem atau komponen dengan maksud untuk menemukan apakah itu memenuhi persyaratan yang ditentukan atau tidak. Kegiatan ini menghasilkan persyaratan yang sebenarnya, dan perbedaan antara hasil mereka. Dengan kata lain pengujian ialah mengeksekusi sistem untuk mengidentifikasi kesenjangan, kesalahan atau persyaratan yang hilang bertentangan dengan keinginan atau persyaratan yang sebenarnya. Menurut ANSI / IEEE 1059 standar, Pengujian dapat didefinisikan sebagai "Suatu proses menganalisis item perangkat lunak untuk mendeteksi perbedaan antara kondisi yang ada dan diperlukan (yaitu cacat / kesalahan / bug) dan mengevaluasi fitur item perangkat lunak".