BAB 2
LANDASAN TEORI

2.1 Temu Kembali Informasi (Information Retrieval)

Information Retrieval (Temu kembali Informasi) atau sering dikenal juga dengan Information Retrieval system (IRS) merupakan media layanan yang dapat digunakan oleh pengguna untuk mendapatkan informasi atau sumber informasi sesuai dengan pencarían yang dilakukan. Hasil informasi yang diperoleh dengan menggunakan kata kunci (keyword) bisa sesuai atau tidak sesuai dengan keinginan pengguna [5].

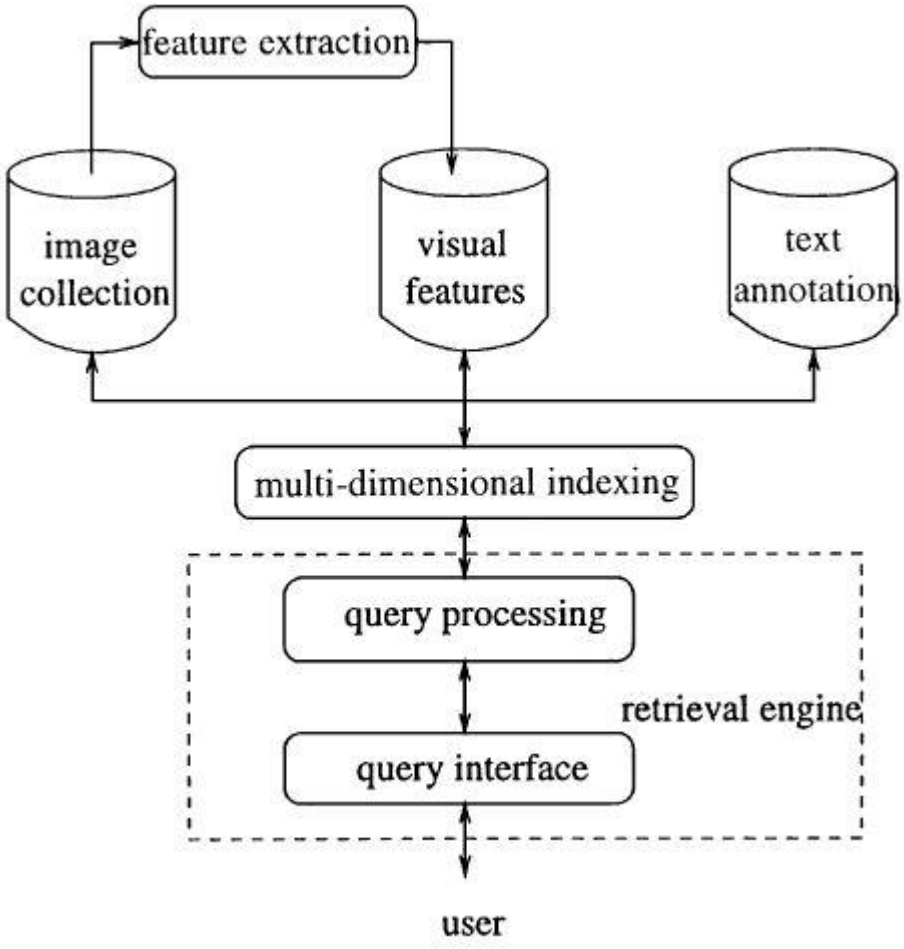
ISO 2382/1 mendefinisikan Information Retrieval (IR) sebagai tindakan, metode dan prosedur untuk menemukan kembali data yang tersimpan, kemudian menyediakan informasi mengenai subyek yang dibutuhkan. Tindakan tersebut mencakup text indexing, inquiry analysis, dan relevance analysis. Data mencakup teks, tabel, gambar, ucapan, dan video.

2.2 Temu Kembali Citra (Image Retrieval)

Image retrieval merupakan teknik yang digunakan untuk mencari citra-citra yang memiliki kemiripan karakteristik dari citra acuan. Setiap citra memiliki informasi fitur yang unik. Dalam hal ini, image retrieval dapat diimplementasikan dengan membandingkan fitur-fitur yang diekstraksi dari suatu citra. Kriteria untuk kemiripan antar citra dapat didasarkan pada fitur – fitur seperti warna, tekstur, bentuk, topologi citra, dan sebagainya. Terdapat dua teknik image retrieval yang berkembang saat ini, yaitu tekstual (manual) dan berdasarkan isi (otomatis).

Teknik *image retrieval* yang pertama, yaitu tekstual, merupakan teknik yang sangat sederhana, yaitu berdasarkan kata kunci yang diberikan untuk tiap citra. Permasalahan dengan teknik ini adalah lamanya waktu pencarian dan adanya ketergantungan terhadap manusia yang sangat tinggi untuk

mendeskripsikan suatu citra. Hal ini menyebabkan terjadinya pendeskripsian yang tidak konsisten. Teknik *image retrieval* yang kedua, berdasarkan isi, adalah teknik yang mengindekskan suatu citra berdasarkan isinya seperti warna, sisi, bentuk, tekstur, informasi spasial, dan sebagainya. Teknik ini sering disebut dengan *Content Based Image Retrieva*l (CBIR). CBIR merupakan jenis *image retrieval* yang paling populer saat ini.



Gambar 2.1 Arsitektur Sistem Image retrieval [6]

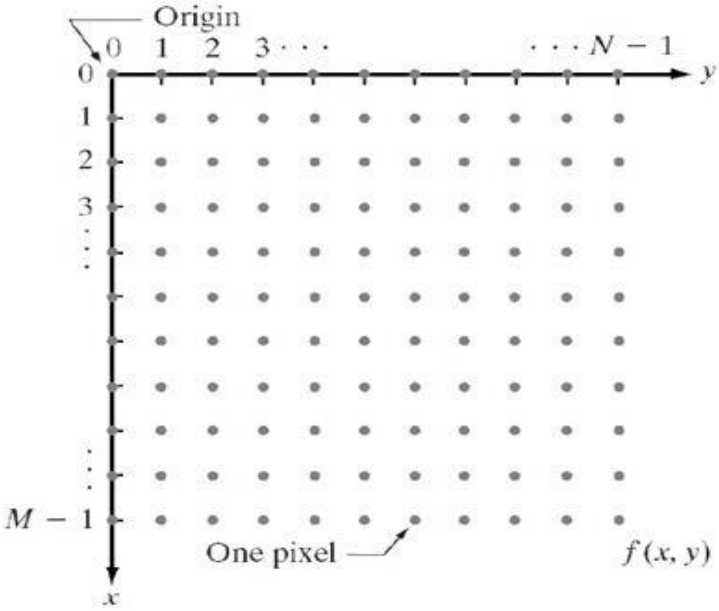
2.3 Pengolahan Citra (Image Processing)

Pengolahan citra merupakan proses pengolahan dan analisis citra yang banyak melibatkan persepsi visual. Proses ini mempunyai ciri data masukan dan informasi keluaran yang berbentuk citra. Istilah pengolahan citra digital secara umum didefinisikan sebagai pemrosesan citra dua dimensi dengan komputer. Dalam definisi yang lebih luas, pengolahan citra digital juga mencakup semua data dua dimensi. Sesuai dengan perkembangan komputer itu sendiri, pengolahan citra mempunyai dua tujuan utama, yaitu sebagai berikut:

1. Memperbaiki kualitas citra, dimana citra yang dihasilkan dapat menampilkan informasi secara jelas. Hal ini berarti manusia sebagai pengolah informasi (human perception).
2. Mengekstraksi informasi ciri yang menonjol pada suatu citra, dimana hasilnya adalah informasi citra dimana manusia mendapatkan informasi ciri dari citra secara numerik atau dengan kata lain computer (mesin) melakukan interprestasi terhadap informasi yang ada pada citra melalui besaran-besaran data yang dapat dibedakan secara jelas (berupa besaran numerik).

2.3.1 Dasar Pengolahan Citra

Suatu Citra dapat didefinisikan sebagai fungsi f(x,y) berukuran M baris dan N kolom, dengan x dan y adalah koordinat spasial dan amplitude f di titik koordinat (x,y) dinamakan intensitas atau tingkat keabuan dari citra pada titik tersebut. Apabila nilai x, y, dan nilai amplitude f secara keseluruhan berhingga (finite) dan bernilai diskrit maka dapat dikatakan bahwa citra tersebut adalah citra digital. Nilai pada suatu irisan antara baris dan kolom (pada posisi x,y) disebut dengan picture elements, pels, image elements, atau pixels. Makin tinggi nomor piksel maka makin gelap (hitam) piksel tersebut. Begitu juga sebaliknya makin rendah nilai piksel tersebut maka makin terang. Sistem yang umum memiliki 256 tingkat kecerahan untuk setiap piksel, yang paling terang adalah 255 dan yang paling gelap adalah 0 [7].



Gambar 2.2 Koordinat Citra

2.3.2 Jenis-jenis Citra [7]

a. Citra Biner

Citra biner adalah citra yang hanya memiliki dua kemungkinan nilai pixel yaitu hitam dan putih. Citra biner juga disebut sebagai citra B&W (*black and white)* atau citra monokrom. Hanya dibutuhkan 1 bit untuk mewakili nilai setiap pixel dari citra biner. Citra biner sering kali muncul sebagai hasil dai proses pengolahan seperti segmentasi, pengembangan morfologi, ataupun *dithering.*

b. Citra *Grayscale*

Citra *grayscale* merupakan citra digital yag hanya memiliki satu nilai kanal pada setiap pixelnya, dengan kata lain nilai bagian *RED = GREEN = BLUE.* Nilai tersebut digunakan untuk menunjukkan tingkat intensitas. Warna yang dimiliki adalah warna dari hitam, keabuan, dan putih. Tingkatan keabuan disini merupakan warna abu dengan berbagai tingkatan dari hitam hingga mendekati putih.

c. Citra berwarna

Citra berwarna adalah citra digital yang setiap pixelnya mengandung informasi warna. Informasi warna ini biasanya dibentuk dari paling sedikit

3 sampel (saluran warna). Saluran warna yang paling umum digunakan adalah RGB.

2.3.3 Pembentukan citra

Pembentukan citra digital (diskrit) melalui beberapa tahapan, yaitu akuisisi citra, sampling, dan kuantisasi [7].

1. Akuisisi citra

   Proses akuisisi citra adalah pemetaan suatu pandangan menjadi citra kontinu dengan menggunakan sensor.
2. *Sampling*

   Sampling merupakan proses pengambilan nilai diskrit koordinat (x.y) dengan melewatkan citra melalui *grid* (celah).
3. Kuantisasi

   Kuantisasi warna merupakan proses untuk melakukan pengurangan jumlah warna dengan mengambil beberapa perwakilan warna yang dapat membedakan satu bagian dengan bagian yang lain pada citra.

2.3.4 Sistem Koordinat Warna

Ada berbagai jenis sistem koordinat warna atau sistem ruang warna yang diciptakan untuk memeuhi tujuan tertentu atau yang diciptakan untuk *platform* perangkat keras tertentu. Berikut adalah beberapa sistem koordinat warna yang ada :

1. RGB

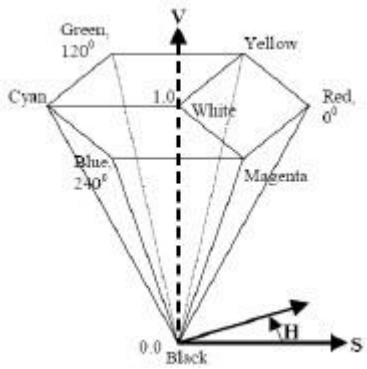
Model warna RGB adalah model warna berdasarkan konsep penambahan kuat cahaya primer yaitu Red, Green dan Blue. Dalam suatu ruang yang sama sekali tidak ada cahaya, maka ruangan tersebut adalah gelap total. Tidak ada signal gelombang cahaya yang diserap oleh mata kita atau RGB (0,0,0). Apabila menambahkan cahaya merah pada ruangan tersebut, maka ruangan akan berubah warna menjadi merah misalnya RGB (255,0,0), semua benda dalam ruangan tersebut hanya dapat terlihat berwarna merah. Demikian apabila cahaya kita ganti dengan hijau atau biru.

2. HSV

Model HSV (Hue Saturatuion Value) menunjukkan ruang warna dalam bentuk tiga komponen utama, yaitu hue, saturation dan value (atau disebut juga brightness). Hue adalah sudut dari 0 sampai 360 derajat. Biasanya 0 adalah merah, 60 derajat adalah kuning, 120 derajat adalah hijau, 180 derajat adalah cyan, 240 derajat adalah biru dan 300 derajat adalah magenta.

Hue menunjukkan jenis warna (seperti merah, biru atau kuning) atau corak warna, yaitu tempat warna tersebut ditemukan dalam spektrum warna. Merah, kuning dan ungu (purple) adalah kata-kata yang menunjukkan hue. Saturasi (saturation) suatu warna adalah ukuran seberapa besar kemurnian dari warna tersebut. Sebagai contoh, suatu warna yang semuanya merah tanpa putih adalah saturasi penuh. Jika ditambahkan putih ke merah, hasilnya lebih berwarna-warni dan warna bergeser dari merah ke merah muda (pink). Hue masih tetap merah tetapi nilai saturasinya berkurang.

Saturasi biasanya bernilai 0 sampai 1 (atau 0% sampai 100%) dan menunjukkan nilai keabu-abuan warna dimana 0 menunjukkan abu-abu dan 1 menunjukkan warna primer murni. Komponen ketiga dari HSV adalah value atau disebut juga intensitas (intensity), yaitu ukuran seberapa besar kecerahan suatu warna atau seberapa besar cahaya datang dari suatu warna. Nilai value dari 0% sampai 100%.



Gambar 2.3 Nilai Hue, Saturation, Value [3]

Perhitungan konversi RGB menjadi HSV dapat dirumuskan sebagai berikut.

$$H = \tan\left[\frac{3(G-B)}{(R-G)+(R+B)}\right] \quad (2.1)$$

$$S = 1 - \frac{\min(R,G,B)}{V} \quad (2.2)$$

$$V = \frac{R+G+B}{3} \quad (2.3)$$

Namun pada rumus di atas, apabila S = 0 maka H tidak dapat ditentukan. Untuk itu diperlukan normalisasi RGB terlebih dahulu dengan rumus berikut.

$$r = \frac{R}{R+G+B} \quad (2.4)$$

$$g = \frac{G}{R+G+B} \quad (2.5)$$

$$b = \frac{B}{R+G+B} \quad (2.6)$$

3. Ruang Warna YCbCr

Ruang warna YCbCr disebut juga ruang warna CCIR601 (International Radio Consultative Committe). Model warna ini dikembangkan untuk mengantisipasi perkembangan informasi berbasiskan video, sehingga model ini banyak digunakan pada video digital. Secara umum dapat dikatakan bahwa model warna ini merupakan bagian dari ruang transmisi video dan televisi. Model warna lain yang mirip dengan YCbCr adalah YUV dan YIQ, perbedaannya terletak bahwa YCbCr adalah system warna digital sedangkan yang lain adalah system warna analog. Model warna YCbCr memisahkan nilai RGB menjadi informasi *luminance* dan *chrominance*. Formulasi konversi RGB - YCbCr ditunjukan sebagai berikut:

$$\begin{aligned} Y &= 0.299900R + 0.58700G + 0.11400B \\ Cb &= -0.16874R - 0.33126G + 0.50000B \\ Cr &= 0.50000R - 0.41869G - 0.08131B \end{aligned} \quad (2.7)$$

matrix diatas adalah merupakan formulasi yang digunakan untuk mengkonversi nilai RGB menjadi informasi *luminance* dan *chrominance*, pada proses pengkonversian ini terjadi sebuah perkalian matrix antara nilai dari setiap komponen warna RGB dengan formula yang tersebut diatas sehingga menjadi sebuah ruang warna baru yaitu Ycbcr.

2.3.5 Format File Citra

Format file standar yang digunakan saat ini terdiri dari beberapa jenis. Format-format ini digunakan dalam menyimpan citra dalam sebuah file. Setiap format memiliki karakteristik masing-masing. Berikut adalah penjelesan beberapa format umum digunakan saat ini [7] :

1. *Bitmap (.bmp)*
   Format .bmp adalah format penyimpanan standar tanpa kompresi yang umum dapat digunakan untuk menyimpan citra biner hingga citra warna. Format ini terdiri dari beberapa jenis yang setiap jenisnya ditentukan dengan jumlah bit yang digunakan untuk menyimpan sebuah nilai pixel.
2. *Tagged image format (*.tiff, .tiff)
   Format .tif merupakan format penyimpanan citra yang dapat digunakan untuk menyimpan citra bitmap hingga citra dengan warna palet terkompresi. Format ini dapat digunakan untuk menyimpan citra yang tidak terkompresi dan juga citra terkompresi.
3. *Portable network Graphics (.png)*
   Format .png adalah format penyimpanan citra terkompresi. Format ini dapat digunakan pada citra grayscale, citra dengan palet warna, dan juga citra *fullcolor*. Format .png juga mampu menyimpan informasi hingga kanal alpha dengan penyimpanan sebesar 1 hingga 16 bit per kanal.
4. *JPEG (.,jpg)*
   .jpg adalah format yang sangat umum digunakan saat ini khususnya untuk transmisi citra. Format ini digunakan untuk menyimpan citra hasil kompresi dengan metode JPEG.
5. MPEG (.mpg)

Format ini digunakan di dunia internet dan diperuntukkan sebagai format penyimpanan citra bergerak (video). Format ini mendukung video dengan kompresi ber-rugi.

6. *Graphic Interchange Format* (.gif)
   Format ini dapat digunakan pada citra warna dengan palet 8 bit. Penggunaan umumnya pada aplikasi web. Kualitas yang rendah meyebabkan format ini tidak terlalu popular di kalangan peneliti pengolahan citra digital.

2.4 Histogram Warna

Histogram Warna adalah suatu proses untuk meratakan histogram agar derajat keabuan dari yang paling rendah (0) sampai dengan yang paling tinggi (255) mempunyai kemunculan yang rata. Histogram Warna suatu citra dibangun dengan menghitung jumlah piksel setiap warna. Perkembangan dalam algoritma ekstraksi masih mengikuti perkembangan yang sama [3]:

1. Pemilihan ruang warna.
2. Kuantisasi ruang warna.
3. Perhitungan histogram.

Setiap langkah ini penting untuk menuju pengembangan algoritma.

2.4.1 Definisi Histogram Warna

Sebuah histogram citra mengacu pada probabilitas fungsi dari intensitas gambar. Hal ini memperluas citra warna untuk menangkap probabilitas bersama dari intensitas tiga ruang warna. Histogram warna didefinisikan sebagai berikut[3].

$$h_{A,B,C}(a,b,c) = N \cdot Prob(A = a, B = b, C = c) \tag{2.8}$$

Dimana A, B, dan C merepresentasikan 3 ruang warna (R,G,B atau HSV) dan N adalah nilai *pixel* dari suatu citra. Secara komputasi, histogram warna dibentuk berdasarkan diskretisasi warna dalam suatu citra dan menghitung jumlah *pixel* setiap warna.

Karena beberapa komputer mewakili citra berwarna mencapai 224 warna, proses ini biasanya membutuhkan kuantisasi ruang warna yang besar. Isu-isu utama tentang penggunaan histogram warna untuk pengindeksan melibatkan pemilihan ruang warna dan kuantisasi dari ruang warna. Ketika ruang warna perseptual yang seragam dipilih, kuantisasi seragam mungkin tepat. Namun jika ruang warna non-seragam yang dipilih, maka kuantisasi non-seragam mungkin diperlukan.

2.4.2 Kuantisasi Warna

Kuantisasi warna merupakan proses untuk melakukan pengurangan jumlah warna dengan mengambil beberapa perwakilan warna yang dapat membedakan satu bagian dengan bagian yang lain pada citra. Proses ini dibutuhkan dikarenakan pada citra 24 bit dengan ruang warna RGB yang mempunyai range warna 0-255 akan memiliki kombinasi warna sebanyak 16.777.216. Dan jika jumlah warna ini dipaksakan untuk dilakukan proses ekstraksi fitur ciri maka akan menghabiskan waktu komputasi (time consuming) dan menimbulkan kesulitan ketika melakukan ekstraksi dan pencocokan vektor cirinya. Masalah tersebut dapat diatasi dengan *color quantization* (kuantisasi warna), yaitu suatu prosedur untuk mengurangi kemungkinan jumlah warna. Dengan cara ini, jumlah warna yang besar tadi bisa dikurangi, sehingga proses yang dibutuhkan akan semakin mudah. [8]

2.4.2.1 Jenis-jenis Kuantisasi [8]

1. Kuantisasi Uniform : mempunyai interval pengelompokan tingkat keabuan yang sama (misal: intensitas 1 s/d 10 diberi nilai 1, intensitas 11 s/d 20 diberi nilai 2, dan seterusnya).
2. Kuantisasi Non-uniform: Kuantisasi yang lebih halus diperlukan terutama pada bagian citra yang menggambarkan detil atau tekstur atau batas suatu wilayah obyek, dan kuantisasi yang lebih kasar diberlakukan pada wilayah yang sama pada bagian obyek.
3. Kuantisasi Tapered: bila ada daerah tingkat keabuan yang sering muncul sebaiknya dikuantisasi secara lebih halus dan diluar batas daerah tersebut dapat dikuantisasi secara lebih kasar.

2.4.3 Normalisasi Histogram

Normalisasi Histogram berguna untuk melihat statistika dari image sehingga Dilakukan agar tampilan histogram pada layar tidak melebihi batas layar, dikarenakan jumlah piksel pada citra sangat besar (hingga ribuan piksel). Cara kerja dari normalisasi histogram adalah membagi setiap nilai dari histogram dengan jumlah pixel dari image.

2.4.4 Histogram Euclidean Distance

Ada beberapa rumus jarak untuk mengukur kemiripan histogram warna. Secara umum, teknik untuk membandingkan distribusi probabilitas, seperti tes kolmogoroff-smirnov tidak sesuai untuk histogram warna. Hal ini karena persepsi visual menentukan kesamaan daripada kedekatan distribusi probabilitas. Pada dasarnya, rumus jarak warna sampai pada ukuran kesamaan antara gambar berdasarkan persepsi konten warna. Untuk itu digunakanlah Histogram *Euclidean Distance* untuk menghitung jarak histogram warnanya.

Misal h dan g mewakili 2 histogram warna. *Euclidean distance* antara histogram warna h dan g dapat dihitung seperti ini [3]:

$$d^2(h,g) = \sum_A \sum_B \sum_C \left(h(a,b,c) - g(a,b,c)\right)^2 \qquad (2.9)$$

Dalam rumus jarak ini, hanya ada perbandingan antara *bins* identik dalam histogram masing-masing. Dua *bins* yang berbeda dapat mewakili perseptual warna yang sama tetapi tidak dibandingkan. Semua *bins* memiliki kontribusi yang sama pada jarak.

2.5 Tools yang digunakan

2.5.1 OOP (Object Oriented Programming)

Metodologi berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya. Metodologi berorientasi objek merupakan suatu cara bagaimana sistem perangkat lunak dibangun melalui pendekatan objek secara sistematis. Metode berorientasi objek didasarkan pada penerapan prinsip-prinsip pengelolaan kompleksitas. Metode berorientasi onjek meliputi rangkaian aktivitas analisis berorientasi objek, perancangan berorientasi objek, pemrograman berorientasi objek, dan pengujian berorientasi objek.

Pada saat ini, metode berorientasi objek banyak dipilih karena metodologi lama banyak menimbulkan masalah seperti adanya kesulitan pada saat mentransformasi hasil dari satu tahap pengembangan ke tahap berikutnya, misalnya pada metode pendekatan terstruktur, jenis aplikasi yang dikembangkan saat ini berbeda dengan masa lalu. Aplikasi yang dikembangkan saat ini sangat beragam (aplikasi bisnis, real-time, utility, dan sebagainya) dengan *platform* yang berbeda-beda, sehingga menimbulkan tuntutan kebutuhan metodologi pengembangan yang dapat mengakomodasi ke semua jenis aplikasi tersebut.[9]

Keuntungan menggunakan metodologi berorientasi objek adalah sebagai berikut[9]:

a. Meningkatkan produktivitas

   Karena kelas dan objek yang ditemukan dalam suatu masalahmasih dapat dipakai ulang untuk masalah lainnya yang melibatkan objek tersebut (*reusable).*

b. Kecepatan pengembangan

   Karena sistem yang dibangun dengan baik dan benar pada saat analisis dan perancangan akan menyebabkan berkurangnya kesalahan pada saat pengkodean.

c. Kemudahan pemeliharaan

Karena dengan model objek, pola-pola yang cenderung tetap dan stabil dapat dipisahkan dan pola-pola yang mungkin sering diubah-ubah.

d. Adanya konsistensi

Karena sifat pewarisan dan penggunaan notasi yang sama pada saat analisis, perancangan maupun pengkodean.

e. Meningkatkan kualitas perangkat lunak

Karena adanya pendekatan pengembangan lebih dekat dengan dunia nyata dan adanya konsistensi pada saat pengambangannya, perangkat lunak yang dihasilkan akan mampu memenuhi kebutuhan pemakai serta mempunyai sedikit kesalahan.

Berikut beberapa contoh bahasa pemrograman yang mendukung pemrograman berorientasi objek[9] :

a. Smalltalk

Smalltalk merupakan salah satu bahasa pemrogramanyang dikembangkan untuk mendukung pemrograman berorientasi objek mulai tahun 1978.

b. Bahasa Pemrograman Eiffel

Eiffel merupakan bahasa pemrograman yang dikembangkan untuk mendukung pemrograman berorientasi objek mulai tahun 1985 oleh Bertrand Meyer dan *compiler* Eiffel selesai pada tahun 1987.

c. Bahasa Pemrograman C++

C++ merupakan pengembangan lebih lanjut bahasa pemrograman C untuk mendukung pemrograman berorientasi objek.

d. Bahasa Pemrograman (web) PHP

PHP dibuat pertama kali oleh seorang perekayasa perangkat (*software engineering)* yang bernama Rasmus Lerdoff.

e. Bahasa Pemrograman Java

Java dikembangkan oleh perusahaan Sun Microsystem. Java menurut definisi dari Sun Microsystem adalah nama untuk sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada komputer *standalone* ataupun pada lingkungan jaringan.

2.5.1.1 Konsep Dasar Berorientasi Objek [9]

a. Objek (*Object*)

   Objek adalah abstraksi dan sesuatu yang mewakili dunia nyata seperti benda, manusia, satuan organisasi, tempat, kejadian, struktur,status, atau hal-hal lain yang bersifat abstrak. Objek merupakan suatu entitas yang mampu menyimpan informasi(status) dan mempunyai operasi yang dapat diterapkan atau dapat berpengaruh pada status objeknya.

b. Kelas (*class*)

   Kelas adalah kumpulan objek-objek dengan karakteristik yang sama. Kelas merupakan definisi statik dan himpunan objek yang sama yang mungkin lahir atau diciptakan dalam kelas tersebut.

c. Pembungkusan (*Encapsulation*)

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

d. Pewarisan (*Inheritance*) dan Generalisasi/Spesialisasi

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

e. Metode

   Operasi atau metode pada sebuah kelas hampir sama dengan fungsi atau prosedur pada metodologi struktural.

f. *Polimorfisme*

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

2.5.2 UML (Unified Modeling Language)

Unified Modelling Language (UML) adalah sekumpulan spesifikasi yang dikeluarkan oleh OMG. UML terbaru adalah UML 2.3 yang terdiri dari 4 macam spesifikasi, yaitu Diagram Interchange Specification, UML, Infrastructure, UML Superstructure, dan Object Constraint Language (OCL). Pada UML 2.3 terdisi 13 macam diagram yang dikelompokkan pada 3 kategori, yaitu [9]:

A. Structure Diagram, yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
    1. Diagram Kelas

       Diagram kelas menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut attribut dan metode atau operasi.
    2. Diagram Objek

       Diagram objek menggambarkan struktur sistem dari segi penamaan objek dan jalannya objek dalam sistem.
    3. Diagram Komponen

       Diagram komponen dibuat untuk menunjukan organisasi dan ketergantungan diantara kumpulan komponen dalam sebuah sistem.
    4. Composite Structure Diagram

       Composite structure diagram baru mulai ada pada UML versi 2.0. Diagram ini dapat digunakan untuk menggambarkan struktur dari bagian-bagian yang saling terhubung maupun mendeskripsikan struktur pada saat berjalan (runtime).
    5. Package Diagram

       Package diagram menyediakan cara mengumpulkan elemen-elemen yang saling terkait dalam diagram UML. Hampir semua diagram dalam UML dapat dikelompokkan menggunakan *package diagram.*
    6. Deployment Diagram

       Deployment menunjukan konfigurasi komponen dalam proses eksekusi aplikasi.

B. Behavior Diagram, yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.
    1. Use Case Diagram

       Use case diagram merupakan pemodelan untuk kelakuan (behavior) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah

interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat.

2. Activity Diagram

   Activity diagram menggambarkan *workflow* atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak.

3. State Machine Diagram

   State machine diagram digunakan untuk menggambarkan perubahan status atau transisi status dari sebuah mesin atau sistem atau objek.

C. Interactions Diagram, yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi antar subsistem pada suatu sistem.

1. Sequence Diagram

   Sequence diagram menggambarkan kelakuan objek pada use case dengan mendeskripsikan waktu hidup objek dan message yang dikirimkan dan diterima antar objek.

2. Diagram Komunikasi

   Diagram komunikasi menggambarkan interaksi antar objek/bagian dalam bentuk urutan pengiriman pesan. Diagram komunikasi merepresentasikan informasi yang diperoleh dari diagram kelas, diagram sekuen, dan diagram *use case* untuk mendeskripsikan gabungan antara struktur statis dan tingkah laku dinamis dari suatu sistem.

3. Timing Diagram

   Timing diagram merupakan diagram yang fokus pada penggambaran terkait batasan waktu.

4. Interaction Overview Diagram

   Interaction overview diagram mirip dengan diagram aktivitas yang berfungsi untuk menggambarkan sekumpulan urutan aktivitas, diagram ini adalah bentuk aktivitas diagram yang setiap titik merepresentasikan diagram interaksi.

2.5.3 Netbeans 8.0

Netbeans merupakan sebuah aplikasi *Integrated Development Environment* (IDE) yang berbasiskan Java dari *Sun Microsystems* yang berjalan di atas *swing*. *Swing* merupakan sebuah teknologi Java untuk pengembangan aplikasi desktop yang dapat berjalan pada berbagai macam platform seperti windows, linux, Mac OS X dan Solaris. Sebuah IDE merupakan lingkup pemrograman yang di integrasikan ke dalam suatu aplikasi perangkat lunak yang menyediakan *Graphic User Interface* (GUI), suatu kode editor atau *text*, suatu *compiler* dan suatu *debugger*.

Netbeans juga digunakan oleh sang programmer untuk menulis, meng-compile, mencari kesalahan dan menyebarkan program netbeans yang ditulis dalam bahasa pemrograman java namun selain itu dapat juga mendukung bahasa pemrograman lainnya dan program ini pun bebas untuk digunakan dan untuk membuat professional *desktop, enterprise, web*, dan *mobile applications* dengan Java language, C/C++, dan bahkan dynamic languages seperti PHP, JavaScript, Groovy, dan Ruby. NetBeans merupakan sebuah proyek *open source* yang sukses dengan pengguna yang sangat luas, komunitas yang terus tumbuh, dan memiliki hampir 100 mitra (dan terus bertambah!). *Sun Microsystems* mendirikan proyek *open source* NetBeans pada bulan Juni 2000 dan terus menjadi sponsor utama. Dan saat ini pun netbeans memiliki 2 produk yaitu Platform Netbeans dan Netbeans IDE. Platform Netbeans merupakan framework yang dapat digunakan kembali (reusable) untuk menyederhanakan pengembangan aplikasi desktop dan Platform NetBeans juga menawarkan layanan-layanan yang umum bagi aplikasi desktop, mengijinkan pengembang untuk fokus ke logika yang spesifik terhadap aplikasi.

Netbeans IDE merupakan sebuah IDE open source yang ditulis sepenuhnya dengan bahasa pemrograman java menggunakan platform netbeans. NetBeans IDE mendukung pengembangan semua tipe aplikasi Java (J2SE, web, EJB, dan aplikasi mobile). Fitur lainnya adalah sistem proyek berbasis Ant, kontrol versi, dan refactoring.

Versi terbaru saat ini adalah NetBeans IDE 8.0 yang dirilis 18 maret 2014. NetBeans IDE 8.0 menyediakan kode analisis yang benar-benar canggih dan editor untuk bekerja dengan teknologi terbaru Java 8. Java SE 8, Java SE Embedded 8, dan Java ME Embedded 8. IDE ini juga memiliki berbagai perangkat tambahan baru yang lebih meningkatkan dukungan untuk Maven dan Java EE dengan PrimeFaces, alat-alat baru untuk HTML5, khususnya untuk AngularJS, dan perbaikan PHP dan C / C + + support.

2.5.4 JDK 8

JDK adalah singkatan dari Java Development Kit yaitu software yang digunakan untuk membangun aplikasi-aplikasi java. Tanpa JDK kita tidak akan bisa membangun atau membuat berbagai macam aplikasi java. JDK ini berguna saat menulis code program. Seperti halnya JRE, JDK juga memiliki JVM di dalamnya. JDK inilah yang merupakan kompiler dari JAVA sendiri, dimana *source code* (kode sumber) dari Java akan dikompilasi menjadi *Byte Code*, yaitu bit code (biner) yang dapat dimengerti oleh mesin. JDK juga akan membentuk sebuah Objek *Code* dari *Source Code*.