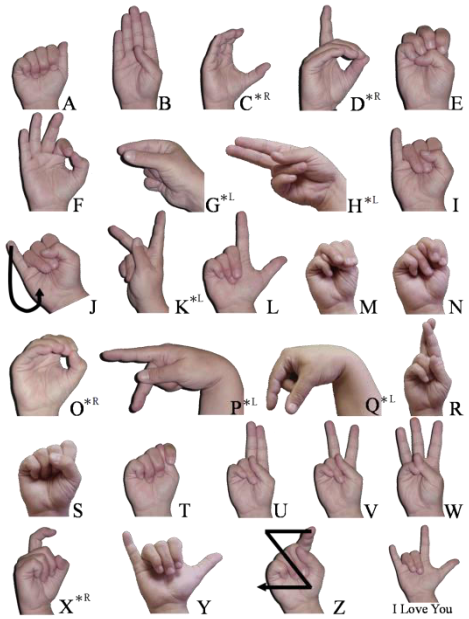
BAB 2
LANDASAN TEORI

2.1 Bahasa Isyarat

Bahasa isyarat adalah salah satu media untuk berkomunikasi yang mengutamakan komunikasi manual, bahasa tubuh dan gerak bibir. Di dunia penggunaan bahasa isyarat dalam setiap negara sangat unik dan bisa saja berbeda – beda, sebahai contoh Inggris dan Amerika menggunakan bahasa isyarat yang berbeda meskipun mempunyai bahasa tertulis yang sama, lain lagi dengan Inggris dan Spanyol meskipun memiliki bahasa tertulis yang berbeda tetapi kedua Negara tersebut menggunakan bahasa isyarat yang sama, karena sampai saat ini belum ada bahasa isyarat internasional yang sukses untuk diterapkan. Di Indonesia, ada dua sistem bahasa isyarat yang berlaku, Bahasa Isyarat Indonesia (BISINDO) dan Sistem Isyarat Bahasa Indonesia (SIBI).



Gambar 2.1 Abjad Bahasa Isyarat Indonesia (BISINDO)



Gambar 2.2 Abjad Sitem Bahasa Isyarat Indonesia (SIBI)

2.2 Kecerdasan Buatan

Kecerdasan buatan (*Artificial inteligence*) adalah salah satu cabang ilmu pengetahuan yang berhubungan dengan pemanfaatan mesin untuk memecahkan persoalan yang rumit dengan cara yang lebih manusiawi. Hal ini biasanya dilakukan dengan mengikuti karakteristik dan analogi berpikir dari kecerdasan manusia, dan menerapkannya sebagai algoritma yang dikenal oleh komputer.

Semakin pesatnya perkembangan teknologi menyebabkan adanya perkembangan dan perluasan lingkup yang membutuhkan kehadiran kecerdasan buatan. Karakteristik cerdas sudah mulai dibutuhkan di berbagai disiplin ilmu dan teknologi. Kecerdasan Buatan tidak hanya merambah di berbagai disiplin ilmu yang lain. Irisan antara psikologi dan kecerdasan buatan melahirkan sebuah area yang dikenal dengan nama *cognition & psycolinguistics*. Irisan antara teknik elektro dengan kecerdasan buatan melahirkan berbagai ilmu seperti pengolahan citra, teori kendali, pengenalan pola dan robotika.

Kecerdasan buatan digunakan untuk menganalisis pemandangan dalam citra dengan perhitungan simbol-simbol yang mewakili isi pemandangan tersebut setelah citra diolah untuk memperoleh ciri khas. Kecerdasan buatan bisa dilihat

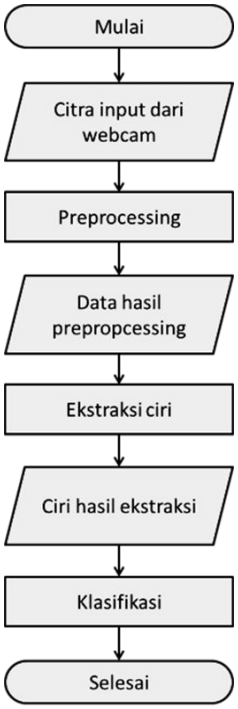
sebagai tiga kesatuan yang terpadu yaitu persepsi, pengertian dan aksi. Persepsi menerjemahkan sinyal dari dunia nyata dalam citra menjadi simbol-simbol yang lebih sederhana, pengertian memanipulasi simbol-simbol tersebut untuk memudahkan penggalian suatu informasi tertentu, dan aksi menerjemahkan simbol-simbol yang telah dimanipulasi menjadi sinyal lain yang dapat merupakan hasil akhir.

2.3 Pengolahan Citra (*Image Processing*)

Untuk melakukan pengenalan suatu objek berdasarkan tekstur yang dimiliki objek tersebut maka dibutuhkanlah suatu teknik untuk pengolahan citra. Pengolahan citra adalah pemrosesan citra dengan menggunakan komputer untuk menghasilkan citra yang kualitasnya lebih baik. Umumnya, operasi-operasi pada pengolahan citra diterapkan pada citra bila :

1. Perbaikan atau memodifikasi citra perlu dilakukan untuk meningkatkan kualitas penampakan atau untuk menonjolkan beberapa aspek informasi yang terkandung di dalam citra,
2. Elemen di dalam citra perlu dikelompokkan, dicocokkan, atau diukur,
3. Sebagian citra perlu digabung dengan bagian citra yang lain.

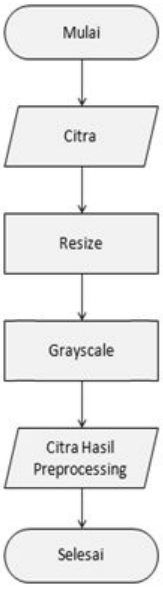
Pengolahan Citra bertujuan memperbaiki kualitas citra agar mudah diinterpretasi oleh manusia atau mesin (dalam hal ini komputer).



Gambar 2.3 Proses Pengolahan Citra

2.3.1 Preprocessing

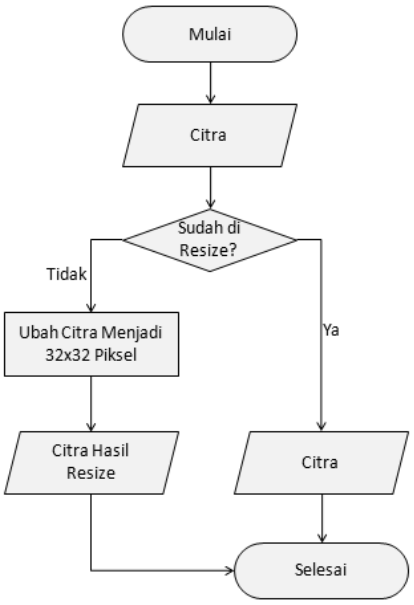
Preprocessing adalah proses untuk memudahkan dalam menghasilkan nilai dari ekstraksi ciri. Pada tahap ini citra inputan yang telah didapat akan dilakukan proses *resize* dan *threshold*. Berikut ini gambar dari proses *preprocessing* :



Gambar 2.4 Proses Preprocessing

2.3.2 Resize

Pada proses preprocessing dilakukan proses *resize* citra. *Resize* adalah proses mengubah resolusi atau ukuran horizontal dan vertikal pada suatu citra, hal ini perlu dilakukan untuk menghasilkan ukuran yang sama pada semua citra yang dimasukkan. Tujuan dari *resize* citra adalah untuk mempercepat proses ekstraksi fitur warna dan fitur tekstur pada gambar yang memiliki dimensi (ukuran) yang besar. Berikut ini gambar dari proses *resize* :

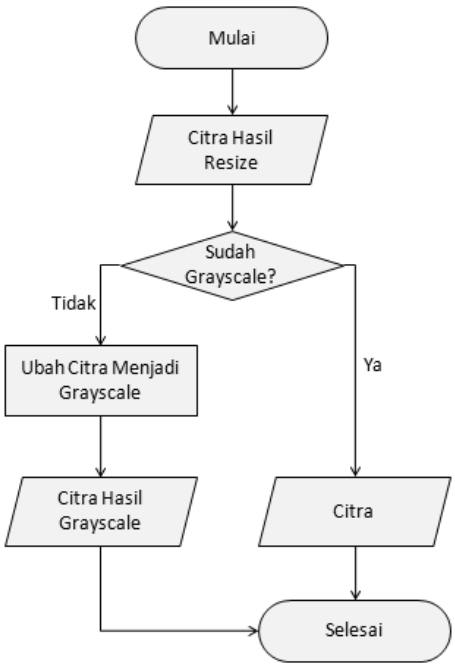


Gambar 2.5 Proses Resize

2.3.3 Threshold

Threshold atau citra biner pada sebuah image digital merupakan proses untuk mengubah warna dimana setiap pikselnya hanya berisikan informasi intensitas warna hitam dan putih. Untuk mendapatkan nilai biner pada citra maka dilakukan proses threshold. Threshold berfungsi untuk memudahkan proses klasifikasi citra agar lebih terfokus terhadap masing-masing piksel.

Berikut ini gambar dari proses threshold :



Gambar 2.6 Proses Threshold

2.4 Elemen Citra

Dalam pengolahan citra objek yang digunakan adalah sebuah citra dari sebuah objek tertentu yang mana citra tersebut mengandung sejumlah elemen dasar. Elemen dasar tersebut di manipulasi dalam pengolahan citra, elemen tersebut adalah:

1. Warna

   Warna adalah persepsi yang dirasakan oleh sistem visual manusia terhadap panjang gelombang cahaya yang dipantulkan oleh objek. Setiap warna mempunyai panjang gelombang. Warna yang diterima oleh tangan merupakan hasil kombinasi cahaya dengan panjang gelombang berbeda. Kombinasi warna yang memberikan rentang warna yang paling lebar adalah *red* (R), *green* (G), *blue* (B).

2. Kecerahan (*brightness*)

   Kecerahan disebut juga intensitas cahaya. Kecerahan pada sebuah piksel (titik) didalam citra bukanlah intensitas yang *rell*, tetapi sebenarnya adalah intensitas rata-rata dari suatu area yang melingkupinya.

3. Kontras (*contrast*)

   Kontras menyatakan sebaran terang dan gelap di dalam sebuah gambar. Citra dengan kontras rendah dicirikan oleh sebagian besar komposisi citranya adalah terang atau sebagian besar gelap. Pada citra dengan kontras yang baik, komposisi gelap dan terang tersebar secara merata.

4. Kontur (*contour*)

   Kontur adalah keadaan yang ditimbulkan oleh perubahan intensitas pada piksel yang bertetangga. Karena adanya perubahan intensitas, tangan manusia dapat mendeteksi tepi objek didalam citra.

5. Bentuk (*shape*)

   Bentuk adalah properti intrinsik dari objek tiga dimensi, dengan pengertian bahwa *shape* merupakan properti intrinsik utama untuk sistem visual manusia. Pada umumnya citra yang dibentuk oleh tangan merupakan citra *dwimatra* (dua dimensi), sedangkan objek yang dilihat umumnya berbentuk *trimatra* (tiga dimensi). Informasi bentuk objek dapat diekstraksi dari citra pada permulaan pra-pengolahan dan segmentasi citra.

6. Tekstur (*texture*)

   Tekstur diartikan sebagai distribusi spasial dari derajat keabuan di dalam sekumpulan piksel-piksel yang bertetangga. Jadi tekstur tidak dapat didefinisikan untuk sebuah piksel. Sistem visual manusia menerima informasi citra sebagai suatu kesatuan. Resolusi citra yang diamati ditentukan oleh skala dimana tekstur tersebut dipersepsi.

7. Waktu dan Pergerakan

   Respon suatu sistem visual tidak hanya berlaku pada faktor ruang, tetapi juga pada faktor waktu. Sebagai contoh, bila citra diam ditampilkan secara cepat, akan berkesan melihat citra yang bergerak.

8. Deteksi dan Pengenalan

   Dalam mendeteksi dan mengenali suatu citra, ternyata tidak hanya sistem visual manusia saja yang bekerja, tetapi juga ikut melibatkan ingatan dan daya pikir manusia.

2.5 Citra analog dan Citra Digital

Secara umum terdapat 2 jenis citra yaitu citra analog dan citra digital. Citra analog adalah citra yang bersifat kontinu, seperti gambar pada monitor televisi, foto sinar-X, foto yang tercetak dikertas foto, lukisan, pemandangan, hasil CT scan, gambar-gambar yang terekam pada pita kaset, dan lain-lain sebagainya.

Citra analog tidak dapat direpresentasikan dalam komputer sehingga tidak dapat diproses dikomputer secara langsung. Oleh sebab itu, agar citra ini dapat diproses dikomputer, proses konversi analog ke digital harus dilakukan terlebih dahulu. Citra analog dihasilkan dari alat-alat analog, *video* kamera analog, kamera foto analog, *Web Cam*, *CT scan*, sensor *ultrasound* pada system USG, dan lain-lain .

Citra digital adalah citra yang dapat diolah oleh komputer dan citra digital yaitu gambar pada bidang dua dimensi. Dalam tinjauan matematis, citra merupakan fungsi kontinu dari intensitas cahaya pada bidang dua dimensi. Ketika sumber cahaya menerangi objek, objek memantulkan kembali sebagian cahaya tersebut. Pantulan ini ditangkap oleh alat-alat pengindera optik, misalnya tangan manusia, kamera, scanner dan sebagainya. Bayangan objek tersebut akan terekam sesuai intensitas pantulan cahaya. Ketika alat optik yang merekam pantulan cahaya itu merupakan mesin digital, misalnya kamera digital, maka citra yang dihasilkan merupakan citra digital. Pada citra digital, kontinuitas intensitas cahaya dikuantisasi sesuai resolusi alat perekam.

2.6 Metode Principal Component Analysis (PCA)

Proses ekstraksi PCA adalah (Principal Component Analysis), dimana hasil proyeksi tersebut diektraksi dengan perhitungan PCA untuk mendapatkan *feature* dari image. *Feature* adalah komponen–komponen penting dari *image-image training* yang didapatkan dari proses *training*. *Feature* inilah yang nanti akan digunakan untuk mengidentifikasikan *image* yang akan dikenali. Dengan demikian pada ekstrasi PCA ini terdapat tahap – tahap untuk menghitung ekstrasi fitur dengan persamaan-persamaan berikut:

2.6.1 Pembentukan matriks data citra

Tahapan awal dari ekstraksi PCA adalah pembentukan matriks yang datanya diambil dari *pixel* setiap citra. Data citra direpresentasikan dalam ukuran matrik m x n, dimana m adalah jumlah citra yang dilatih dan n merupakan dimensi dari citra tersebut, kemudian dilakukan proses ekstraksi menjadi citra dengan dimensi yang lebih kecil yang hasilnya diproyeksikan menjadi sebuah matriks seperti pada persamaan berikut [6].

$$x = \begin{bmatrix} x_{1,1}x_{1,2}\ x_{1,3} \dots & x_{1,n} \\ x_{2,1}x_{2,2}x_{2,3} & x_{2,n} \\ x_{3,1}\ x_{3,2}x_{3,3} & x_{3,n} \\ \dots & \dots \\ x_{m,1}x_{m,2}x_{m,3} \dots & x_{m,n} \end{bmatrix} \quad \text{..... (2.1)}$$

Dimana

x = Matriks citra

Setelah matriks data citra terbentuk, maka proses berikutnya adalah proses perhitungan untuk mencari rata-rata hasil seluruh citra. Pencarian nilai rata-rata ini tujuannya untuk mengetahui *noise* atau persamaan tiap vector yang dapat menganggu keakuratan perhitungan pada PCA, yang dapat dihitung dengan menggunakan rumus: [6].

$$\alpha = \frac{(x_{1,i} + x_{2,i} + x_{3,i} + \dots + x_{m,i})}{m} = \frac{\sum_{j=1}^{m} x_{j,i}}{m} = [\tau_1, \tau_2, \dots \dots \dots, \tau_n] \quad \text{..... (2.2)}$$

Dimana

α = Rata-rata matriks seluruh citra

$x_{1,i\dots}\ x_{m,I}$ = Matrik citra ke-1 sampai citra ke-m

m = Jumlah matriks citra

Terlihat bahwa hasil perhitungan persamaan diatas adalah merupakan sebuah *array* yang memuat nilai n, (τ1, τ2, … … …, τn). *Array* tersebut nantinya akan digunakan untuk mencari nilai rata-rata nol (*zero mean*), setelah

dibentuk menjadi matriks 1xn terlebih dahulu. Namun, ukuran matriks nilai rata-rata tidak berkesesuaian dengan matriks data citra sampel yang memiliki ukuran mxn, maka perlu dilakukan penggandaan nilai rata-rata.

2.6.2 Perhitungan nilai rata-rata nol (Zero Means)

Perhitungan nilai rata-rata nol, berfungsi untuk menghilangkan *noise* yang dapat menganggu keakuratan pada perhitungan PCA. Perhitungan ini dapat dimodelkan menggunakan persamaan.

$D - \alpha = Y$ ……..(2.6)

Dimana :

D = Matriks Vektor Dataset 4x6400

α = Rata-rata matriks vektor dataset 1x6400

Y = Matriks Zero Means 4x6400

2.6.4 Pembentukan matrik kovarian

Hasil perhitungan nilai rata-rata nol digunakan untuk mendapatkan nilai matriks kovarian. Matriks kovarian merupakan matriks reduksi dari proses ekstraksi PCA, yang memiliki dimensi lebih kecil dibandingkan dimensi matriks citra asli. Berikut ini adalah persamaan matriks kovarian.

$Y * Y^I = C$ ……(2.7)

Dimana :

Y = Matriks Zero Means 6x190

Y^I = Transpose Matriks Zero Means 190x6

C = Matriks Kovarian 6x6

2.6.5 Penentuan matriks *eigenvektor*

Setelah matriks kovarian didapatkan maka langkah selanjutnya adalah menentukan matriks *eigenvektor.* Matriks *eigen* dapat dicari menggunakan persamaan: [6]

$$C - Z = |C - \lambda I|$$

$$|C - \lambda I| = 0 \quad \text{……..……..……...(2.9)}$$

Dimana C adalah matriks kovarian. Z adalah matriks *eigenvektor* dengan λsebagai *scalar* pembentuknya dan I sebagai matriks identitas, persamaan tersebut jika diimplementasikan ke bentuk matriks menjadi persamaan sebagai berikut: [6]

$$\begin{bmatrix} C_{11}C_{12} & \dots & C_{1,n} \\ C_{21}C_{2,2} & \dots & C_{2,n} \\ \dots & \dots & \dots \\ C_{m,1}C_{m,2} & \dots & C_{m,n} \end{bmatrix} - \lambda \begin{bmatrix} 1\,0 & \dots & 0 \\ 0\,1 & \dots & 0 \\ \dots & \dots & \dots \\ 0\,0 & \dots & 1 \end{bmatrix} = 0$$

$$\begin{bmatrix} C_{11}C_{12} & \dots & C_{1,n} \\ C_{21}C_{2,2} & \dots & C_{2,n} \\ \dots & \dots & \dots \\ C_{m,1}C_{m,2} & \dots & C_{m,n} \end{bmatrix} - \begin{bmatrix} \lambda\,0 & \dots & 0 \\ 0\,\lambda & \dots & 0 \\ \dots & \dots & \dots \\ 0\,0 & \dots & \lambda \end{bmatrix} = 0$$

$$\begin{bmatrix} C_{11} - \lambda & C_{12} & \dots & C_{1,n} \\ C_{21} & C_{2,2} - \lambda & \dots & C_{2,n} \\ \dots & \dots & \dots & \dots \\ C_{m,1} & C_{m,2} & \dots & C_{m,n} - \lambda \end{bmatrix} = 0 \quad \text{……..……... (2.10)}$$

2.7 Klasifikasi Naïve Bayesian

Naïve bayesian adalah suatu metode pengklasifikasian paling sederhana dengan menggunakan peluang yang ada, dimana diasumsikan bahwa setiap variable X bersifat bebas (*independence*) [4]. Karena asumsi variabel tidak saling terikat, maka didapatkan :

$$P(X \mid Y = y) = \prod_{i=1}^{q} P(X_i \mid Y = y) \qquad (2.7)$$

Terdapat beberapa langkah dalam pengklasifikasian menggunakan metode naive bayesian, berikut adalah langkah - langkahnya :

Training :

1. Hitung rata-rata (*mean*) tiap fitur dalam dataset training dengan.

   $$\mu = \frac{\sum x_i}{n} \quad \text{……(2.8)}$$

   Dimana:

   μ = mean eigenvektor dataset

   n = banyaknya data

$\sum x_i$ = jumlah nilai data

2. Kemudian hitung nilai varian dari dataset training tersebut seperti pada.

$$\sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu)^2 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.9)$$

Dimana:

σ^2= varians

μ= mean

x_i = nilai data

n = banyaknya data

Testing :

1. Hitung probabilitas (*Prior*) tiap kelas yang ada dengan cara menghitung jumlah data tiap kelas dibagi jumlah total data secara keseluruhan.
2. Selanjutnya menghitung densitas probabilitasnya. Fungsi densitas mengekspresikan probabilitas relatif. Data dengan mean μ dan standar deviasi σ, fungsi densitas probabilitasnya adalah :

$$\varphi_{\mu\sigma}(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (2.10)$$

Dimana :

x = data masukan

π = 3,14

σ = standar deviasi

μ = mean

3. Setelah didapatkan nilai densitas probabilitasnya, selanjutnya menghitung posterior masing-masing kelas dengan menggunakan persamaan.

$$posterior = \frac{prior\ x\ likehood}{evidence} \quad (2.11)$$

Atau bisa ditulis

$$Posterior = \frac{P(C_i)x\ P(X1\ldots\ldots\ldots\ldots\ldots Xn|C_i)}{P(X1\ldots\ldots\ldots\ldots\ldots Xn|C_i)} \quad (2.12)$$

4. Setelah didapat nilai posterior masing-masing kelas, maka kelas yang sesuai untuk data masukan adalah kelas yang memiliki nilai posterior terbesar.

2.8 Object Oriented Programming (OOP)

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 penerapann prinsip-prinsip pengelolaan kompleksitas. Metode berorientasi objek meliputi rangkaian aktifitas analisis beorientasi objek, perancangan berorientasi objek, pemrograman berorientasi objek, dan pengujian berorientasi objek .

Pada saat ini, metode berorientais objek banyak dipilih karena metodologi lama banyak menimbulkan masalah seperti adanya kesulitan pada saat mentranformasi 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.

Keuntungan menggunakan metodologi berorientasi objek adalah sebagai berikut :

a. Meningkatkan Produktivitas

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

b. Kecepatan Pengembangan

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

c. Kemudahan Pemeliharaan

   Karena dengan model objek, pola-pola yang cendrung 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 pemograman yang mendukung pemrograman berorientasi objek adalah :

a. Bahasa Pemrograman Eiffel

   Eiffel merupakan bahsa pemrograman yang kembangkan untuk mendukung pemrograman berorientasi objek oleh Bertrand Meyer dan *compiler*.

b. Bahasa Pemrograman Web (PHP)

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

c. Bahasa Pemrograman C#

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

2.9 Pemrograman C#

2.9.1 Sejarah C#

C# (tanda '#' dibaca "Sharp") merupakan bahasa pemograman baru yang diciptakan Microsoft secara khusus sebagai salah satu bahasa pemrograman dalam teknologi .Net sebagai bahasa baru, C# tidak berevolusi dari bahasa C# versi bukan teknologi .Net. dengan demikian C# dapat memaksimalkan kemampuannya tanpa khawatir dengan masalah kompatibilitas dengan versi-versi sebelumnya. Keharusan sebuah perangkat lunak untuk tetap dapat kompatibel dengan versi-versi sebelumnya sebagaimana yang terjadi pada Visual Basic (VB) maupun

C++biasanya menghambat optimalitas kemampuan dari perangkat lunak tersebut [16].

Sejak diluncurkan pada tahun 2000, C# dengan cepat merebut hati progammer C++ bahkan VB. Dengan tata cara penulisan yang mirip C++ dan *interface* mirip VB 6.0 menurut wikipedia, sebuah ensiklopedia gratis di internet pengguna C# .Net pada saat ini sudah melebihi pengguna VB.Net. sementara itu jumlah pengguna bahasa pemrograman lain masih berada dibawah jumlah pengguna VB .Net. masih menurut wikipedia, jumlah buku C# yang terjual pun berada dikisaran 2 hingga 3 kali lebih banyak dari jumlah buku VB yang terjual.
Dari informasi ini dapat disimpulkan bahwa C# merupakan bahasa pemrogrman baru yang sedang berkembang dan dapat diterima dengan baik oleh kebanyakan progammer dan kalangan industri. Di Microsoft sendiri, C# merupakan bahasa pemrograman yang digunakan untuk membuat perangkat lunak yang berteknologi .Net dengan demikian dapat diperkirakan bahwa C# akan menjadi bahasa pemrograman yang akan banyak digunakan di masa-masa mendatang [16].

2.9.2 Konsep Pemrograman Berorientasi Objek

Berikut adalah konsep dasar pemrograman berorientasi objek :

a. Objek (*Object*)

   Objek adalah abtraksi dan sesuatu yang mewakili dunia nyata seperti benda, satuan organisasi, tempat, kejadian, struktur, status, atau hal-hal lain yang bersifat abstrak. Objek merupakan suatu entitas yang mampu menyimpan informasi 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 yang mempunyai 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*

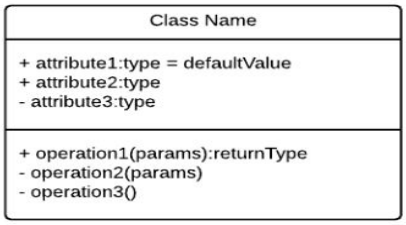
Kemampuan suatu objek untuk digunakan dibanyak tujuan yang berbeda dengan nama yang sama sehingga menghemat baris program.

2.10 Unified Modelling Language (UML)

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 Spesification,* UML *Infrastrukture,* UML *Superstrukture, dan Object Constraint Language* (OCL). Pada UML 2.3 terdiri 13 macam diagram yang dikelompokan pada 3 kategori, yaitu:

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

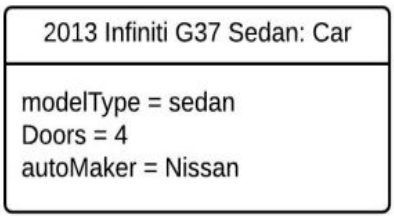
    Diagram kelas menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut attribut dan metode atau operasi.



Gambar 2.7 Diagram Kelas

2. Diagram Objek

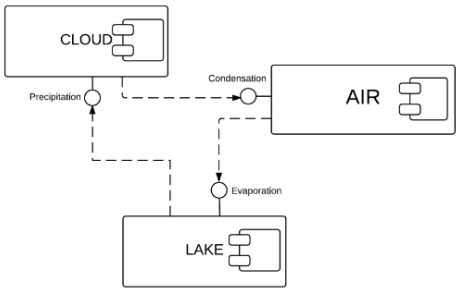
   Digram objek menggambarkan struktur sistem dari segi penamaan objek dan jalannya objek dalam sistem.



Gambar 2.8 Diagram Objek

3. Diagram Komponen

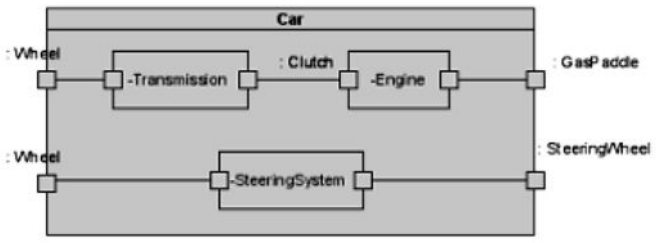
   Diagram komponen dibuat untuk menunjukan organisasi dan ketergantungan diantara kumpulan komponen dalam sebuah sistem.



Gambar 2.9 Diagram Komponen

4. *Composite Structure Diagram*

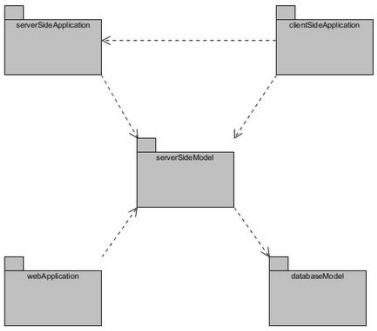
   Composite Structure Diagram baru mulai ada pada UML versi 2.0. diagram ini dapat digunakan untuk menggambarkan struktur dari bagian-bagian yang saling terhubung maupun mendeskripsikan struktur pada saat berjalan (*runtime*).



Gambar 2.10 Composite Structure Diagram

5. *Package Diagram*

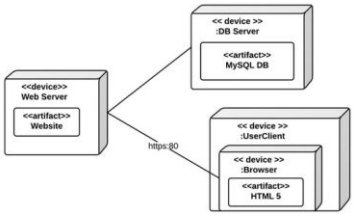
   Package diagram menyediakan cara mengumpulkan elemen-elemen yang saling terkait dalam diagram UML. Hampir semua diagram dalam UML dapat dikelompokan menggunakan *Package Diagram*.



Gambar 2.11 Package Diagram

6. *Deployment Diagram*

   Deployment diagram menunjukan konfigurasi komponen dalam proses eksekusi aplikasi.

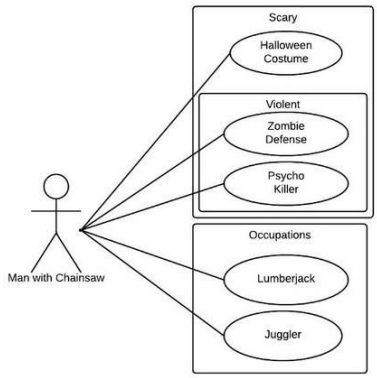


Gambar 2.12 Deployment Diagram

2. *Behavior Diagram*, yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem. Pada *Behavior Diagram* dibagi menjadi 3 bagian :

   1. *Use Case Diagram*

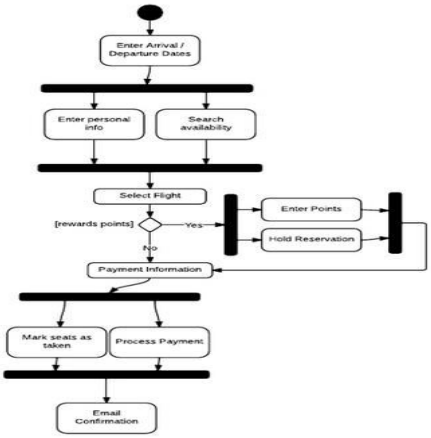
      Use case diagram merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat,. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat.



Gambar 2.13 Use Case Diagram

2. *Activity Diagram*

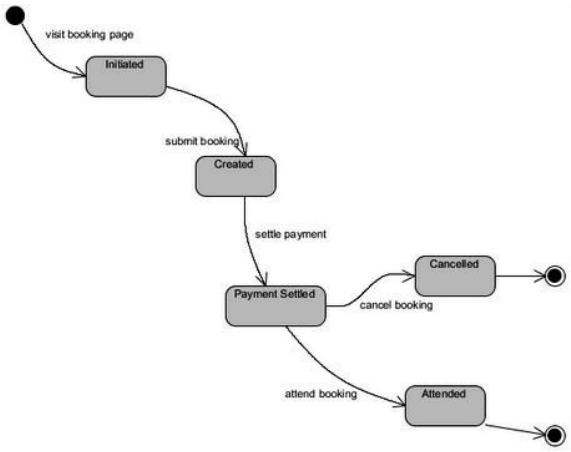
   Activity diagram menggambarkan *workflow* atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak.



Gambar 2.14 Activity Diagram

3. *State Machine Diagram*

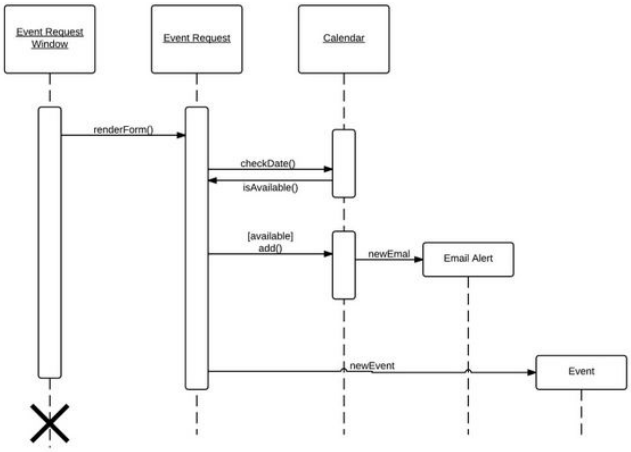
   State machine diagram digunakan untuk menggambarkan perubahan status atau transisi status dari sebuah mesin atau sistem atau objek.



Gambar 2.15 State Machine Diagram

3. *Interactions Diagram*, yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi antar subsistem pada suatu sistem. Pada Interactions Diagram dibagi menjadi 4 bagian :
   1. *Sequence Diagram*

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

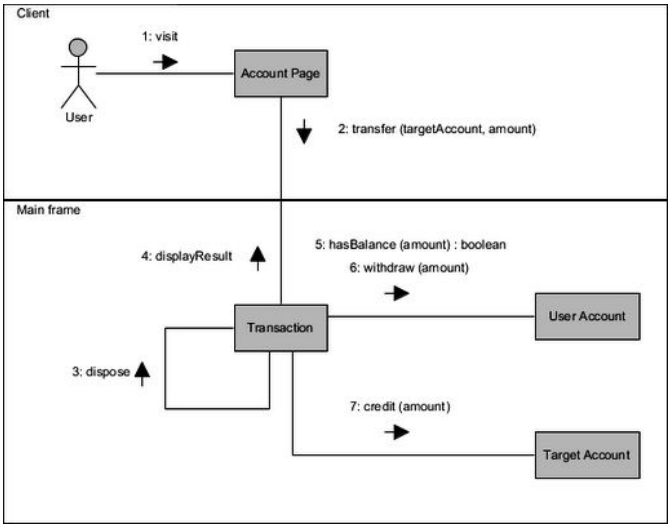


Gambar 2.16 Sequence Diagram

   2. *Communication Diagram*

      Communication diagram menggambarkan interaksi antar bojek/bagian dalam bentuk urutan pengiriman pesan. Diagram komunikasi

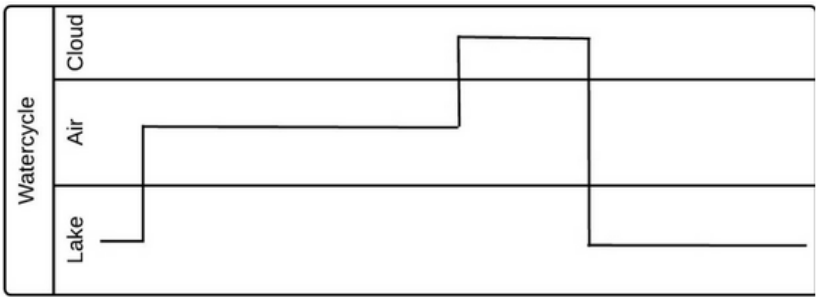
merepresentasikan informasi yang diperoleh dari diagram kelas, diagram *sequence*, dan diagram *use case* untuk mendeskripsikan gabungan antara struktur statis dan tingkah laku dinamis dari suatu sistem.



Gambar 2.17 Communication Diagram

3. *Timing Diagram*

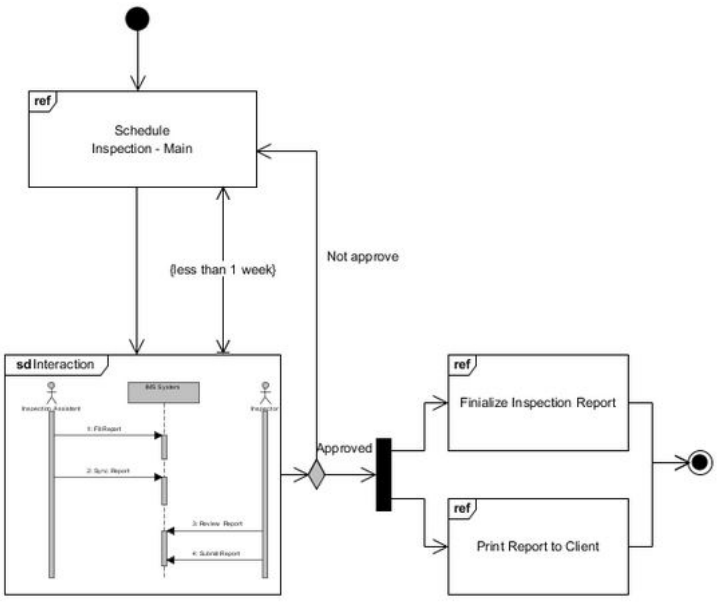
   Timing diagram merupakan diagram yang fokus pada penggambaran terkait batas waktu.



Gambar 2.18 Timing Diagram

4. *Interaction Overview Diagram*

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



Gambar 2.19 Interaction Overview Diagram