BAB 2
TINJAUAN PUSTAKA

2.1 Pengolahan Citra

Pemanfaatan citra digital banyak digunakan dalam berbagai bidang seperti bidang pendidikan, kedokteran, industry dll. Pengolahan citra digital merupakan salah satu disiplin ilmu yang mempelajari hal-hal yang berkaitan dengan perbaikan kualitas gambar(peningkatan kontras, tranformasi, warna, restorasi citra), transformasi gambar(rotasi, translasi, skala, transformasi, geometrik), melakukan pemulihan citra ciri (*feature image*s) yang optimal untuk tujuan analisis, melakukan proses penarikan informasi atau deskripsi objek atau pengenalan objek yang terkandung pada citra, melakukan kompresi atau reduksi data untuk tujuan penyimpanan data, transmisi datam dan waktu proses data,input dai pengolahan citra adalah citra, sedangkan outputnya adalah citra hasil pengolahan.

2.2 Operasi Pengolahan Citra

Banyak operasi yang dilakukan dalam pengolahan citra, umumnya operasi pengolahan citra dapat diklasifikasikan dalam beberapa jenis sebagai berikut [1]:

1. Perbaikan Kualitas Citra (*image enhancement*)

Operasi citra jenis ini bertujuan untuk memperbaiki kualitas citra dengan cara memanipulasi parameter-parameter citra, ciri khusus yang terdapat didalam citra lebih ditonjolkan. Contoh-contoh operasi citra perbaikan citra :

a. Perbaikan kontras gelap/terang.
b. Perbaikan tepian objek (*edge enhancement*).
c. Penajaman (*sharpening*).
d. Penajaman warna semu (*sharpening*).
e. Penapisan derau (*noise filtering*).

2. Pemugaran Citra (*image restoration*)

Operasi citra ini bertujuan untuk menghilangkan/meminimumkan cacat pada citra. Tujuan pemugaran citra hampir sama dengan operasi perbaikan citra,

perbedaanya pada pemugaran citra penyebab degradasi gambar diketahui. Contoh operasi pemugaran citra :

a. Penghilangan kesamaran (*debluring*).
b. Penghilangan derau (*noise*).

3. Pemampatan Citra (*image compression*)

Jenis operasi ini bertujuan agar citra dipresentasikan dalam bentuk yang lebih kompak sehingga memerlukan memori yang lebih sedikit. Hal penting yang harus diperhatikan dalam pemampatan adalah citra yang telah dimampatkan harus tetap mempunyai kualitas gambar yang bagus.

4. Segmentasi Citra (*image segmentation*)

Operasi citra jenis ini bertujuan untuk memecah suatu citra kedalam beberapa segmen dengan suatu kriteria tertentu. Jenis operasi ini berkaitan erat dengan pengolahan pola.

5. Pengorakan Citra (*image analysis*)

Jenis operasi ini bertujuan menghitung besaran kuantitatif dari citra untuk menghasilkan deskripsinya. Teknik pengorakan citra mengekstraksi ciri-ciri tertentu yang membantu dalam identifikasi objek. Proses segmentasi seringkali diperlukan untuk melokalisasi objek yang diinginkan dari sekelilingnya.contoh-contoh operasi pengorakan citra :

a. Pendeteksian tepi objek (*edge detection*)
b. Ekstraksi batas (*boundary*)
c. Representasi daerah (*region*)

6. Rekonstruksi Citra (*image reconstruction*)

Operasi citra jenis ini bertujuan untuk membentuk ulang objek dari beberapa citra hasil proyeksi.operasi rekonstruksi citra banyak digunakan dalam bidang medis. Misalnya foto rontgen dengan sinar X digunakan untuk membentuk ulang gambar organ tubuh.

7. Perubahan Model Warna

Warna adalah persepsi yang dirasakan oleh sistem visual manusia terhadap panjang gelombang cahaya yang dipantulkan oleh objek [1]. Setiap warna mempunyai panjang gelombang yang berbeda. Warna merah mempunyai panjang

gelombang yang paling tinggi, sedangkan warna ungu mempunyai panjang gelombang rendah. Warna-warna yang diterima oleh mata adalah hasil kombinasi cahaya dengan panjang gelombang berbeda. Penelitian memperlihatkan kombinasi warna memberikan rentang warna yang paling lebar adalah red (R), green (G), dan blue (B).

a. Citra RGB yang biasa disebut citra true color, disimpan dalam citra berukuran (m x n) x 3 yang mendefinisikan warna merah (red), hijau (green) dan biru (blue) untuk setiap pikselnya. Warna pada stiap piksel ditentukan berdasarkan kombinasi dari warna red, green, dan blue (RGB). RGB merupakan citra 24 bit dengan komponen merah, hijau dan biru yang masing-masing umumnya bernilai 8 bit sehingga intensitas kecerahan warna sampai 256 level dari kombinasi warnanya kurang dari sekitar 16 juta warna.
b. Citra keabuan

   Citra dengan derajat keabuan berbeda dengan citra RGB, citra ini didefinisikan oleh satu nikai derajat warna. Umumnya bernilai 8 bit sehingga intensitas kecerahan warna sampai 256 leve dan kombinasi warnanya256 varian. Tingka kecerahan paling rendah yaitu 0 untuk warna hitam dan putih bernilai 255. Untuk mengkonversi citra yang memiliki warna RGB ke derajat keabuan bisa menggunakan rumus dibawah ini.

   $X = 0.2 * R + 0.72 * G + 0.07 * B$ (Persamaan 1)

   Dimana :

   X = citra greyscale

   R = red

   G = green

   B = blue

2.3 Tekstur

Tekstur adalah konsep intuitif yang mendeksripsikan tentang sifat kehalusan, kekasaran dan keteraturan dalam suatu daerah /wilayah(*region*). Dalam pengolahan citra digital, tekstur didefiniskan sebagai distribusi spasial dari derajat keabuan di dalam sekumpulan piksel yang bertetangga[7].

Secara umum tekstur mengacu pada pengulangan elemen-elemen tekstur dasar yang disebut primitif atau tekstel(*textur element-texel*) syarat-syarat terbentuknya suatu tekstur antara lain :

1. Adanya pola-pola primitif yang terdiri dari suatu piksel atau lebih. bentuk-bentuk pola primitif ini dapat berupa titik, garis lurus, garis lengkung, luasan, dan lain-lain yang merupakan elemen dasar dari sebuah tekstur.
2. Pola-pola primitif tersebut muncul berulang-ulang dengan interval dan arah tertentu sehingga dapat diprediksi atau ditemukan karakteristik pengulangannya.

Suatu citra memberikan interpretasi tekstur yang berbeda jika dilihat dengan jarak dan sudut yang berbeda. Manusia memandang tekstur berdasarkan deskripsi yang bersifat acak, seperti halus, kasar, teratur,tidak teratur dan sebagainya. Hal ini merupakan deskripsi yang tidak tepat dan non kuantitatif, sehingga diperlukan adanya suatu deskripsi yang kuantititif (matematis) untuk memudahkan analisis[7].

2.4 Analisis Tekstur

Analisis tekstur merupakan dasar dari berbagai macam aplikasi, aplikasi dari analisis tekstur antara lain: penginderaan jarak jauh, pencitraan medis, identifikasi kualitas suatu bahan (kayu, kulit, tekstil, dan lain-lain), dan juga berbagai macam aplikasi lainnya. Pada analisis citra, pengukuran tekstur dikategorikan menjadi lima kategori utama yaitu : statistis, struktur, geometri, model dasar, dan pengolahan sinyal. Pendekatan statistis mempertimbangkan bahwa intensitas dibangkitkan oleh medan acak dua dimensi, metode ini berdasar pada frekuensi ruang. Contoh metode statistis adalah fungsi autokorelasi, *Co-ocurence Matrix***,** transformasi fourier, frekuensi tepi, run length dan lainnya. Teknik struktural berkaitan dengan penyusunan bagian-bagian terkecil suatu citra. Contoh metode struktural adalah model fraktal. Metode geometri berdasar atas perangkat geometri yang ada pada elemen tekstur. Contoh metode model dasar adalah medan acak. Sedangkan metode pengolahan sinyal adalah metode berdasarkan analisis frekuensi seperti tranformasi gabor dan transformasi wavelet[6].

2.5 Grey Level Co-occurrence Matrix

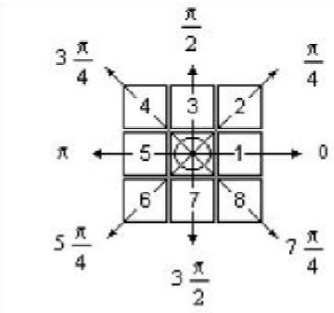
Co-occurance matrix atau matriks ko-okurensi adalah salah satu metode statistik yang dapat digunakan untuk analisis tekstur. Matriks Ko-okurensi dibentuk dari suatu citra dengan melihat pada piksel-piksel yang berpasangan yang memiliki intensitas tertentu. Penggunaan metode ini berdasar pada hipotesis bahwa dalam suatu tekstur akan terjadi keabuan. Misal, d didefinisikan sebagai jarak antara dua posisi piksel, yaitu (x1,y1) dan (x2,y2); dan 0 didefinisikan sebagai sudut diantara keduanya. Maka matriks kookurensi didefinisikan sebagai matriks yang menyatakan distribusi spasial antara dua piksel yang bertetangga yang memiliki intensitas i dan j, yang memiliki jarak d diantara keduanya, matriks kookurensi dinyatakan dengan :

$$P(i,j) = \frac{V(i,j)}{\sum_{i,j=0}^{n-1} V(i,j)} \qquad \text{(Persamaan 2)}$$

Dimana :

- P : Probabilitas munculnya pasangan berderajat keabuan
- $V(i,j)$: Banyaknya pasangan titik-titik dengan tingkat keabuan tertentu

Suatu piksel yang bertetangga yang memiliki jarak d diantara keduanya, dapat terletak di delapan arah yang berlainan, untuk lebih jelasnya dapat dilihat pada **Gambar 0.1**.



Gambar 0.1 Hubungan Ketetanggan Antar Piksel

Setelah memperoleh matriks kookurensi, fitur ciri yang merepresentasikan citra dapat dihitung. Dalam matriks kookurensi, terdapat beberapa ciri tekstur yang dapat diperoleh dari suatu citra yang digunakan sebagai pembeda antara citra dengan kelas tertentu, dengan kelas lainnya[]. Ciri-ciri tersebut adalah :

1. Energi (*Energy*)

Energi adalah fitur untuk mengukur konsentrasi pasangan intensitas pada matriks kookurensi. Nilai energi akan semakin membesar bila pasangan piksel yang memenuhi syarat matriks intensitas kookurensi terkosentrasi pada beberapa koordinat dan mengecil bila letaknya menyebar. Rumus yang digunakan untuk menghitung energi adalah :

Rumus Energi = $\sum_{i,j=0}^{n-1} P^2(i,j)$ (Persamaan 3)

Dimana :

i, dan j adalah keabuan dari resolus 2 piksel yang berdekatan

p(i,j) adalah co-occurrence matrix simetris ternormalisasi.

2. Entropi (*Entropy*)

Entropi menunjukan ketidakteraturan distribusi intensitas suatu citra pada matriks kookurensi persamaanya untuk menghitung entropi adalah :

Rumus Entropy = $\sum_{i,j=0}^{n-1} \mathrm{P}(i,j)(-\ln P(i,j))$ (Persamaan 4)

Dimana :

i dan j adalah sifat keabuan dari resolusi 2 piksel yang berdekatan

p(i,j) adalah co-occurrence matrix simetris ternormalisasi.

3. Kontras (*Contrast*)

Kontras adalah fitur yang digunakan untuk mengukur kekuatan perbedaan intensitas dalam citra. nilai kontras membesar jika variasi intensitas citra tinggi dan menurun bila variasi rendah. Persamaan yang digunakan untuk menghitung kontras adalah :

Rumus Kontras = $\sum_{i,j=0}^{n-1} Pi,j(i-j)^2$ (Persamaan 5)

Dimana :

i dan j adalah sifat keabuan dari resolusi 2 piksel yang berdekatan

p(i,j) adalah prbabilitas kolom(i,j).

4. Homogenitas (*Homogenity*)

Homogenitas digunakan untuk mengukur kehomogenan variasi intensitas citra. Nilai homogenitas akan semakin membesar bila variasi intensitas dalam citra mengecil. Persamaan yang digunakan untuk menghitung homogenitas adalah :

Rumus homogenitas = $\sum_{i,j=0}^{n-1} \frac{P(i,j)}{1+|i-j|}$ (Persamaan 6)

Dimana :

i dan j adalalh sifat keabuan dari 2 piksel yang berdekatan

p(i,j) adalah probabilitas kolom(i,j).

Dalam persamaan diatas, notasi P melambangkan probabilitas, yang bernilai mulai dari nol hingga satu, yaitu nilai elemen dalam matriks kookurensi, sedangkan i dan j, melambangkan pasangan intensitas yang berdekatan, yang dalam matriks kookurensi masing-masing menjadi nomor baris dan nomor kolom. Ciri-ciri dari matriks kookurensi yang memiiki nilai sama atau hampir sama dapat digunakan salah satunya saja, sehingga dapat mengurangi beban komputasi [6].

2.6 Klasifikasi

Klasifikasi merupakan suatu pekerjaan yang melakukan penilaian terhadap suatu obyek data untuk maksud dalam suatu kelas tertentu dari sejumlah kelas yang tersedia[9].

Ada dua pekerjaan utama:

1. Pembangunan model sebagai prototype untuk disimpan sebagai memori.
2. Menggunakan model tersebut untuk melakukan pengenalan/klasifikasi/prediksi pada suatu obyek data lain masuk pada kelas mana.

Contoh aplikasi:

1. Klasifikasi pengenalan jenis hewan spesies baru.
2. Diagnosis penyakit pasien
3. Pengenalan jenis bunga
4. Pemeriksaan kualitas bunga kamboja (bahan teh)
5. Absensi karyawan (dengan sidik jari, mata, wajah, tangan, dsb.)

Algoritma klasifikasi dibagi menjadi dua macam: *erger learner* dan *lazy learner[9]*.

1. Eager learner

a. Didesain untuk melakukan pembacaan/pelatihan/pembelajaran pada data latih untuk dapat memetakan dengan benar setiap vektor masukan ke label kelas keluaranya.
b. Model (berupa bobot/sejumlah nilai kualitas tertentu) disimpan sebagai memori, sedangkan semua data latihnya dibuang.
c. Prediksi dilakukan menggunakan model yang tersimpan, tidak melibat data uji sama sekali.
d. Proses prediksi berjalan dengan cepat.
e. Proses pelatihan lama.

Contoh: *Artificial Neural Network* (ANN), *Support Vector Machine* (SVM), Decision Tree, Bayesian dan sebagainnya.

2. Lazy learner

a. Hanya sedikit melakukan pelatihan (bahkan tidak sama sekali).
b. Hanya menyimpan sebagian atau seluruh data latih, kemudian menggunakan data latih tersebut ketika proses prediksi.
c. Proses prediksi menjadi lama karena model harus membaca kembai semua data latihnya untuk dapat memberikan keluaran label kelas dengan benar pada data uji yang diberikan.
d. Proses pelatihan berjalan dengan cepat.

Contoh: K-Nearest Neihgbor (K-NN), Fuzzy K-Nearest Neighbor (FK-NN), Regresi Linear, dan sebagainya.

2.7 Klasifikasi K-Nearest Neighbor

K-Nearest Neighbor merupakan sebuah metode klasifikasi terhadap sekumpulan data berdasarkan pembelajaran data yang sudah terklasifikasikan sebelumnya. KNN termasuk dalam golongan *supervised* , dimana hasil query *instance* yang baru diklasifikasikan berdasarkan mayoritas kedekatan jarak dari kategori yang ada dalam KNN. Nantinya kelas yang baru dari suatu data akan dipilih berdasarkan grup kelas yang dekat jarak vektornya.[8].

Tujuan dari metode ini adalah mengklasifikasikan obyek baru berdassarkan atribut dan trainning sample. *Classifier* tidak menggunakan model apapun untuk

dicocokan dan hanya berdasarkan pada memori. Diberikan titik *query*, akan ditemukan sejumlah k obyek atau (titik training) yang paling dekat dengan titik *query*. Klasifikasi menggungkan coting terbanyak diantara klasifikasi dari k obyek. Metode *k-nearest neighbor* (KNN) menggunakan klasifikasi ketetanggaan sebagai nilai prediksi dari *query instance* yang baru.

Metode k-nearest neighbor (KNN) sangatlah sederhana, bekerja berdasarkan jarak terpendek dari query instance ke training sample untuk menemukan KNN-nya. *Training sample* diproyeksikan ke ruang berdimensi banyak, dimana masing-masing dimensi merepresentasikan fitur dari data. Ruang ini dibagi menjadi bagian–bagian berdasarkan klasifikasi *trainning sample*. sebuah titik pada ruang ini ditandai kelas c jika kelas c merupakan klasifikasi yang paling banyak ditemukan pada k buah tetangga terdekat dari titik tersebut. Dekat atau jauhnya tetangga biasanya dihitung berdasrkan *Euclidean Distance*.

Jarak *Euclidian* paling sering digunakan menghitung jarak. Jarak *euclidean* berfungsi menguji ukuran yang bisa dgunakan sebagai interpretasi kedekatan jarak antara dua obyek. Yang direpresentasikan sebagai berikut:

$$d(x-y) = \sqrt{\sum_{j=1}^{i}(x_j - y_j)^2} \qquad \text{(Persamaan 7)}$$

Keterangan:

d = jarak data uji ke data pembelajaran.

x_j = data uji ke-j, dengan j = 1, 2, . . . n.

y_j = data pembelejaran ke-j dengan j = 1, 2, . . . n.

Nilai k yang terbaik untuk algoritma ini tergantung pada data. Secara umum, nilai k yang tinggi akan mengurangi efek noise pada klasifikasi, tetapi membuat batasan antara setiap klasifikasi menjadi semakin kabur. Nilai K yang bagus dapat dipilih dengan optimasi parameter, misalnya dengan menggunakan *cross-validation*. Kasus khusus dimana klasifikasi diprediksikan berdasarkan trainning data yang paling dekat (dengan kata lain, k=1) disebut metode nearest neighbor.

Ketepatan metode KNN sangan dipengaruhi oleh ada atau tidaknya fitur-fitur yang tidak relevan atau jika bobot fitur tersebut tidak setara dengan relevansinya

terhadap klasifikasi. Riset terhadap metode ini sebagian besar membahas bagaimana memilih dan memberi bobot terhadap fitur agar performa klasfikasi menjadi lebih baik.

Langkah-langkah untuk menghitung k-*nearest neighbor* :

1. Menentukan parameter K (jumlah tetangga yang paling dekat).
2. Menghitung kuadrat jarak euclid (*query inctance*) masing-masing obyek terhadap data sampel yang diberikan.
3. Kemudian mengurutkan objek-objek tersebut kedalam kelompok yang mempunyai jarak *euclidean* terkecil.
4. Mengumpulkan kategori Y (Klasifikasi nearest neighbor).
5. Dengan menggunakan kategori nearest neighbor yang paling mayoritas maka dapat diprediksikan nilai *query instance* yang telah dihitung.

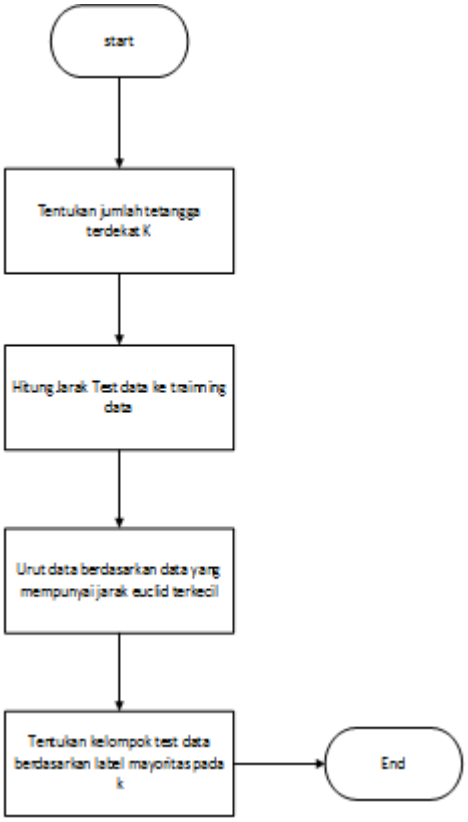
   Kelebihan dari metode k-nearest neighbor adalah sebagai berikut:

   a. Ketangguhan terhadap data yang memiliki banyak noise.
   b. Efektif terhadao data yan berukuran besar.

   Kekurangan dari metode k-nearest neighbor adalah sebagai berikut:

   a. Nilai k harus ditentukan secara manual.
   b. Trainning berdasarkan jarak harus menggunakan banyak sampel untuk mendapatkan hasil yang terbaik.
   c. Memerlukan komputasi tingkat tinggi karena perlunya menghitung satu persatu data testing terhadap semua data trainning.

Adapun metode dari KNN pada flowchart berikut :



Gambar 0.2 Flow chart metode KNN

2.8 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 meliputi rangkaian aktifitas analisis eroreintasi objek, perancangan beroreintasi objek, pemrograman beroreintasi objek, dan pengujian berorientasi objek[4].

Pada saat ini, metode beroreintasi objek banyak dipiih karena metodelogi lama banyak menimbulkan masalah seperti adanya kesulitan pada saat mentranformasi hasil dari satu tahap pengembangan ke tahap berikutnya, misalnya pada metde pendekatan terstruktur, jenis aplikasi yan dikembangkan saat ini berbeda dengan masa lalu. Aplikasi yang dikembangkan saat ini sangat beragam (aplikasi bisinis, *real-time*, *utillity* dan sebagainnya) dengan platform yang berbeda-beda, sehingga

menimbulkan tuntunan kebutuhan metodelogi pengembangan yang dapat mengakomodasikan ke semua jenis aplikasi.

Keuntungan menggunakan metodologi beroreintasi objek adalah sebagai berikut :

a. Meningkatkan Produktivitas

   Karena kelas dan objek yang ditemuka dalam suatu masalaha masih dapat dipakai ulang untuk masalah yang lain 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 yang sama pada saat analisis, perancangan maupun pengkodean.

e. Menigkatkan Kualitas Perangkat Lunak

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

2.9 Konsep Dasar PBO

Berikut merupakan konsep dasar pemrograman berorientasi objek:

a. Objek (*Object*)

   Objek adalah abstraksi dan sesuatu yang mewakili dunia nyata seperti benda, satuan organisasi, tempa, 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 berpengaruh pada status objeknya.

b. Kelas (*Class*)

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

c. Pembungkusan (*Encapsulation*)

Pembungkusan atribut data dan layanan yang mempunyai objek untuk menyembunyikan implementasi dan objek seshingga objek lain tidak mengetahui cara kerjanya

d. Pewarisan (*Inheritance*)

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

e. Metode

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

f. Polimorfisme

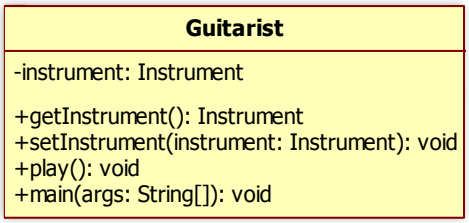
Kemampuan suatu objek untuk digunakan dibanyak tujuan berbeda dengan nama yang sama sehingga menghemat baris program.

2.10 UML

UML (*Unified Modelling Language*) adalah bahasa pemodelan untuk sistem atau perangkat lunak yang berparadigma ‘beoreintasi objek’. Pemodelan (*Modelling*) sesungguhnya digunakan untuk penyedarhaan permaslahan-permasalahan yang kompleks sedemikian rupa sehingga lebih mudaj dipelajari dan dipahami. Salah satu contohnya, alih-alih membawa palnet bumi ke meja kita, kita mungkin akan membawa peta-peta tentang bagian bumi tertentu ke meja kita untuk kita pelajari lebih jauh. Dalam hal ini, sasaran model sesungguhnya adalah abstraksi segala sessuatu yang ada di planet bumi menjadi gambaran-gambaran yang lebih mudah dipelajari dan dipahami. Adapun tujuan pemodelan (dalam kerangka pengembangan sistem/perangkat lunak aplikasi) adalah serupa dengan yang kita singgung diatas, yaitu sebagai sarana analisis, epeahaman,visualisasi, dan komunikasi antar anggota tim pengembangan (saat seorang

analisis/beberapa/banyak anggota), serta sebagai sarana dokumentasi (yang bermanfaat untuk menelaah perilaku perangkat lunak yang telah selesai dikembangkan).

Untuk secara efektif memodelkan sebuah sistem, diperlukan sebuah bahasa dengan model yang dapat dijelaskjan, dan disinilah UML diperlukan. Sebuah bahasa pemodelan dapat terdiri dai pseudo-code, kode aktual, gambar, diagram, atau bagian panjang deskripsi. Unsur-unsur yang membentuk bahasa pemodelan yang disebut notasi. Gambar 2.3 menunjukan contoh sepotong notasi UML.



Gambar 0.3 Class Diagram

Setiap pendekatan untuk pemodelan memiliki kelebihan dan kekurangan yang berbeda, namun UML memiliki enam keuntungan utama yaitu:

1. UML adalah bahasa formal

   Setiap elemen dari bahasa memiliki makna didefinisikan kuat, sehingga anda dapat yakin bahwa ketika anda memodelkan sebuah aspek tertentu dari sistem anda tidak akan salah paham.

2. UML singkat

   Seluruh bahasa terdiri dari notasi yang sederhana dan mudah.

3. Kompherensif

   Ini menggambarkan semua aspek penting dari sebuah sistem.

4. Terukur

   Dimana dibutuhkan, bahasa yang cukup untuk menangani proyek-proyek pemodelan sistem besar formal, tetapi juga skala ke proyek-proyek kecil, untuk menghindari berlebihan

5. Dibangun diatas pembelajaran

   UML adalah puncak dari praktek-praktek terbaik dalam komunitas berorientasi objek selama 15 tahun terakhir.

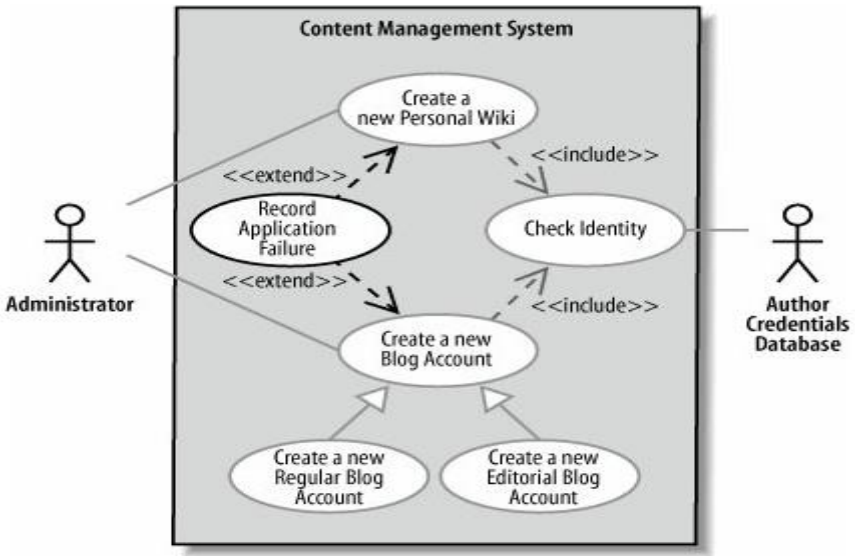
6. Standar

UML dikendalikan oleh kelompok standar terbuka dengan kontribusi aktif dari kelompok di seluruh dunia vendor dan akademisi, yang *fends off* "*vendor lock-in*." Standar memastikan UML transformability dan interoperabilitas, yang berarti Anda tidak terikat dengan suatu produk tertentu.

Di dalam UML terdapat beberapa diagram yang digunakan yaitu, *Use Case* diagram, *activity* diagram, *Class* diagram, *Statechart* diagram dan *sequence* diagram.

1. Use Case Diagram

Use case diagram digunakan untuk menggambarkan fungsionalitas dari sebuah sistem. *Use case* diagram yaitu mendeskripsikan apa yang sistem lakukan tanpa mendeskripsikan bagaimana sistem menyelesaikannya.

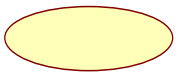


Gambar 0.4 Use case diagram

Berikut ini beberapa tabel simbol yang ada di dalam use case diagram:

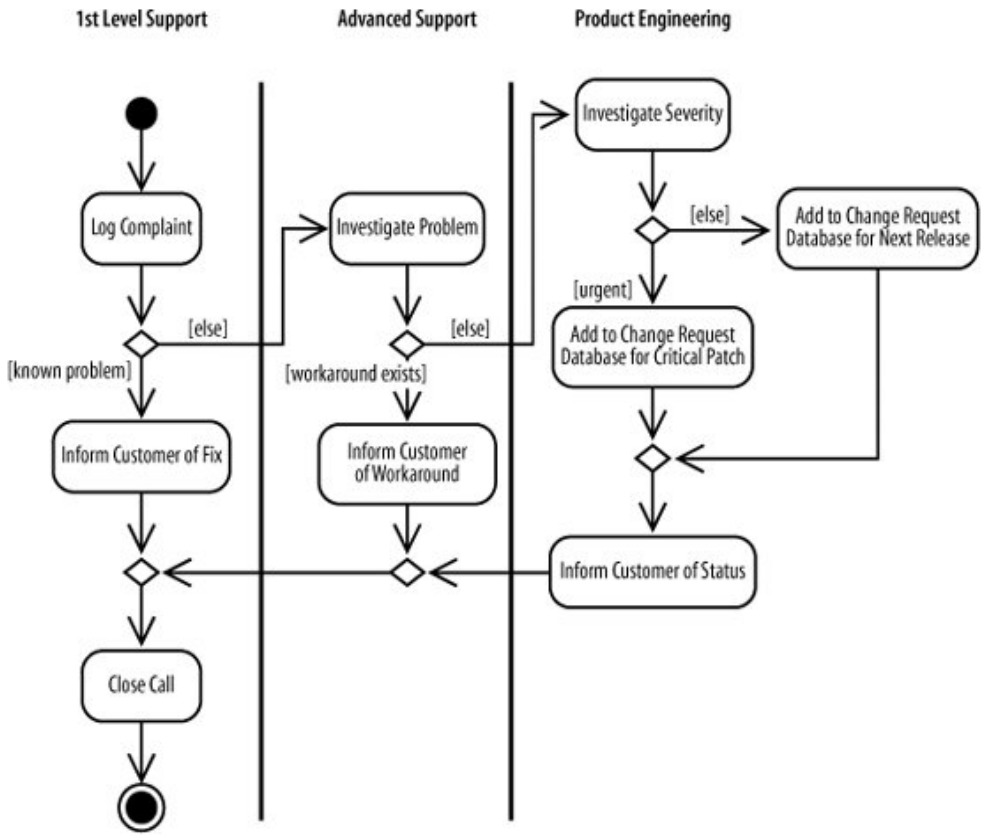
Tabel 0.1 Daftar Simbol

Simbol	Nama Simbol	Fungsi
	Aktor	Actor menggambarkan orang, system atau external entitas yang menyediakan atau menerima informasi dari system.

	*Use case*	Menggambarkan fungsionalitas dari sistem.
	Assosiation	Menggambarkan interaksi antara aktor dengan *use case*.
	Sistem *boundary*	Menggambarkan ruang lingkup atau batasan dari suatu sistem.

2. Activity Diagram

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity* diagram juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. Activity diagram merupakan state diagram khusus, di mana sebagian besar state adalah action dan sebagian besar transisi di-*trigger* oleh selesainya state sebelumnya. Oleh karena itu *activity* diagram tidak menggambarkan *behaviour* internal sebuah sistem secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum. Menggambarkan proses bisnis dan urutan aktivitas dalam sebuah proses. Dipakai pada *business modeling* untuk memperlihatkan urutan aktifitas proses bisnis. Struktur diagram ini mirip *flowchart* atau *Data Flow Diagram* pada perancangan terstruktur. Sangat bermanfaat apabila kita membuat diagram ini terlebih dahulu dalam memodelkan sebuah proses untuk membantu memahami proses secara keseluruhan. *Activity* diagram dibuat berdasarkan sebuah atau beberapa *use case* pada *use case* diagram.

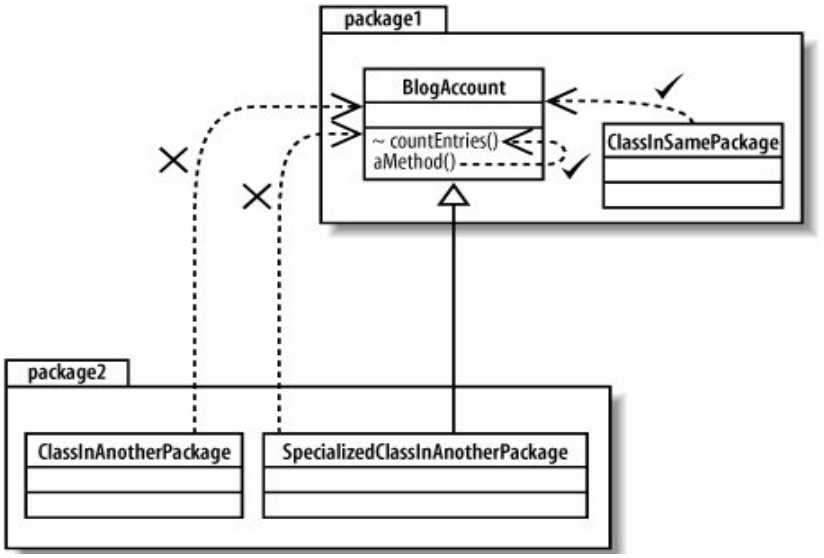


Gambar 0.5 Activity Diagram

3. Class Diagram

Class diagram merupakan salah satu diagram utama dari *Unified Modelling Language* (UML) untuk menggambarkan *class* atau *blueprint object* pada sebuah sistem. Pada *class* diagram juga digambarkan bagaimana interaksi hubungan antar *class* dalam sebuah konstruksi piranti lunak seperti hubungan asosiasi, agregasi, komposisi, dan inheritance. Standarisasi pemakaian *class* diagram yang ter *up to date* pada diagram UML 2.0.

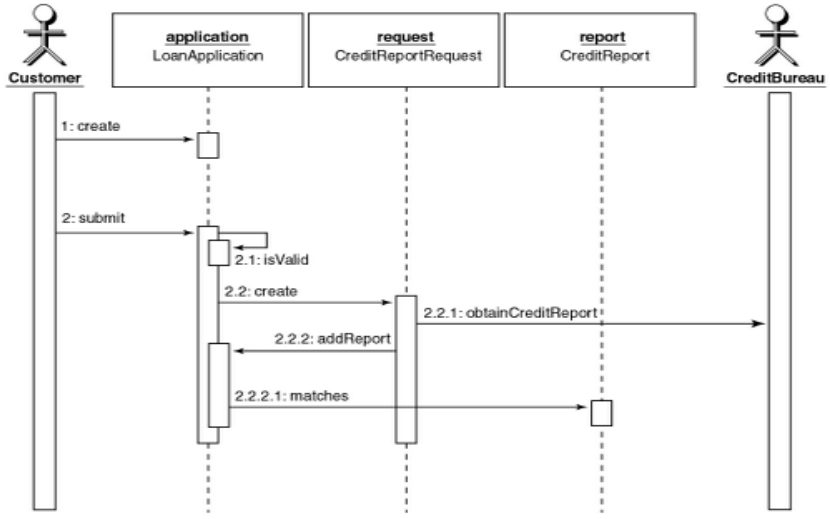
Sesuai dengan perkembangan *class* model, class dapat dikelompokkan menjadi *package*. Sehingga dapat membuat diagram yang terdiri atas *package*.



Gambar 0.6 Class diagram

4. Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem berupa *message* yang digambarkan terhadap waktu. *Sequence* diagram terdiri atas dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait.



Gambar 0.7 Sequence Diagram

2.11 Bahasa Pemrograman C#

C# (baca : C sharp atau see sharp) adalah sebuah bahasa pemrograman yang beroreintasi pada objek yang dikembangkan oleh .Net programming melalui visual studio. C# didasarkan pada C++, c# juga memiliki kemiripan dengan bahasa pemrograman seperti Visual Basic, Java, Delphi, dan tentu saja C++. C# memiliki kemudahan syntax (cara penulisan) seperti Visual Basic, dan memudahkan programmer dari berbagai latar belakang bhasa pemrograman tidak perlu waktu lama untuk menguasainya, karena C# lebih sederhana dibandingkan bahasa-bahasa pemrograman seperti C++ dan Java[11].

C# didesain oleh program deigner dari microsoft, Anders Hajlsberg. Sejak diluncurkan pada tahun 2000, C# dengan ceat merebut programmer C++ bahkan VB, dengan tata cara penulisan yang mirip C++dan interface mirio VB 6.0 menurut wikipedia, sebuah ensiklopedia gratis di internet pengguna C#.Net pada saat ini sudah melebihi pengguna VB.Net. sementara jumah penggguna bahasa pemrograman yang lain lebih rendah daripada pengguna VB.Net. masih menurut wikipeda, julah 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 pemrograman baru yang sedang berkembang dan dapat diteruma dengan baik oleh kebanyakan programmer 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(Hartanto,B, 2008).

Kelebihan C# yaitu :

1. Sederhana (simple) C# menghilangkan beberaoa hal yang bersifat kompleks yang terdapat dalam beberpaa macam bahasa pemrograman seperti Java dan C++, termasuk diantaranya menghilangkan macro, templates, multiple inheritance dan virtual base classes. Hal-hal tersebut yang dapat menyebabkan kebingunan pada saa mnggunakannyam dan juga berpotensial dapat menhadi masalah bagi para prograammer C++.

2. C# bersifat sederhana, karena bahasa ini didasarkan kepada bahasa C dan C++.
3. Modern apa karena adanya beberapa fitur seperti exception handling, garbage collection, extensible data types, dan code security (keamanan kode/bahasa pemrograman). Dengan adaya fitur-fitur tersebut, menjadikan bahasa C# sebagai bahasa pemrograman yang modern.
4. Powerfull dan fleksibel, c# bisa digunakan untuk membuat berbagai macam aplikasi, seperti aplikasi pengolahan kata, grafik, spreadsheets, atau bahkan membuat kompiler untuk sebuah bahasa pemorgraman.
5. Efisien, C# adalah bahasa pemorgraman yang menggunakana jumlah kata-kata yang tidak terlalu banyak. C# hanya berisi kata-kata yang biasa disebut dengan keywords. Keywords ini digunakan untuk menhelaskan berbgai macam informasi.[12].

Modular kode C# ditulis dengan pembagian masing-masing kelas-kelas (Classes) yang terri dari beberapa routines yang disebut sebaai member methods. Kelas-kelas dan metode-metode ini dapat digunakan kembali oleh program atau aplikasi lain. Hanya dengan memberikan informasi yang dibutuhkan oleh class dan metode yang dimaksud, maka kita dapat membuat suatu kode yang dapat digunakan oleh satu atau beberapa aplikasi dan program (*ruesable code*).

2.12 Visual Studio .Net

Visual Studio .Net merupakan editor yang paling cocok untuk membuat aplikasi yang berbasis Framework .Net, termasuk aplikasi dengan bahasa C#. dengan editor ini, maka akan bisa memanfaatkan kemampuan c#secara maksismal. Editor ini tidak hanya menyediakan berbagai macam tools dan wizard untuk membuat aplikasi C#, tapi juga termasuk fitur-fitur produktif seperti InteliSense dan bantuan yang dinamis[14].

Dengan InteliSense, jika mengetikan nama sebuah namespace atau nama class, maka anggota dari namespace atau class itu akan seccara otomatis dimunculkan sehingga kita tidak erlu mengingat anggota dari semua Iatau semua class yang kita gunakan. InteliSense, juga akan menampilkan semua argumen dan jenis typenya

ketika kita mengetikan nama dari sebuah metode. Visual Studio 6 jenis type dan class-class yang termasuk di dlam lingkungan Framework .Net.

Fitur bantuan yang dinamis (*dynamic help*) merupakan fitur yang batu yang ada dalam "Keluarga" editor visual studio. Ketika mengetikan sebuah code pada penjelasan yang berhubungan dengan kata-kata tempat dimana kita menempatkan cursor. Misalnya, ketika mengetikan kata namespace, maka jenedela yang terpisah tadi akan dimunculkan topik-topik yang berhubungan dengan kata kunci namespace. Hal ini tentu akan sangat membantu pada saat pembuatan program untuk dapat lebih memahami dan mengerti tentang kata-kata atau kode yang ditulis.