BAB 2

LANDASAN TEORI

2.1 Peringkasan Teks Otomatis

Peringkasan teks otomatis (*Automated Text Summarization*) merupakan pengertian dari sebuah aplikasi berbasis komputer yang memiliki fungsi utama untuk menghasilkan ringkasan dari sebuah artikel tanpa menghilangkan tujuan utama dari artikelnya. Tujuan utamanya adalah untuk menunjukkan beberapa kalimat utama yang digabung menjadi sebuah ringkasan, dengan harapan dapat mengurangi waktu untuk membaca dan memahami isi dari bacaan.

Sebuah sistem peringkasan teks akan diberikan masukkan berupa teks, kemudian melakukan peringkasan dan menghasilkan keluaran berupa teks yang lebih singkat dari teks aslinya. Hasil dari peringkasan tersebut mengandung poin – poin penting atau informasi utama yang ada dari teks asalnya.

Terdapat dua pendekatan pada teknik peringkasan teks, yaitu ekstraksi (*shallower approaches*) dimana pada teknik ini sistem menyalin unit – unit teks yang dianggap paling penting atau memiliki informasi – informasi utama dari teks sumber yang dijadikan bahan ringkasan dan satu lagi adalah teknik abstraksi (*deeper approaches*) dimana teknik ini mengambil intisari dari teks sumber, kemudian membuat sebuah ringkasan dengan menciptakan kalimat – kalimat baru yang merepresentasikan intisari teks sumber ke dalam bentuk berbeda dengan kalimat – kalimat pada teks sumber. Berdasarkan sumbernya, sebuah ringkasan dapat dihasilkan dari satu sumber (*single document*) atau berasal dari banyak sumber (*multy document*) [14].

Pada penelitian ini tipe peringkasan teks yang akan digunakan adalah ekstraktif dengan memanfaatkan aplikasi yang dijalankan pada komputer untuk menghasilkan sebuah ringkasan dari dokumen aslinya. Teks yang dijadikan sumber berasal dari satu sumber (*single document*).

2.2 Teks *Preprocessing*

Teks *preprocessing* adalah suatu tahapan yang dilakukan untuk mempersiapkan teks sumber menjadi data yang siap diolah pada tahapan selanjutnya. Jenis masukkan yang dibutuhkan pada proses ini adalah berupa teks dokumen yang nantinya teks tersebut akan melalui beberapa tahapan utama yaitu :

2.2.1 *Splitting* Kalimat

Memecah teks dokumen sumber menjadi bentuk kalimat – kalimat, ini merupakan tahapan awal *preprocessing*. *Splitting* kalimat adalah suatu proses untuk memecahkan string teks dokumen yang panjang menjad kalimat – kalimat. Delimiter yang digunnakan untuk memotong string teks dokumen menjadi kalimat adalah titik ".", tanda tanya "?" dan tanda seru "!" dengan menggunakan fungsi split().

2.2.2 *Tokenizing* Kata

Tokenizing kata adalah proses pemotongan string kalimat berdasarkan tiap kata yang menyusunnya. Pemecahan kalimat menjadi kata – kata dilakukan dengan cara mencari kalimat dengan delimiter *white space* (spasi, tab dan *new line*).

2.2.3 Case folding

Case folding adalah proses pemrosesan teks dimana semua teks diubah ke dalam case yang sama dan pada kasus ini teks diubah representasinya ke dalam hurus kecil semua (*lower case*).

2.2.4 *Filtering*

Data teks dalam dokumen yang sebelumnya sudah diubah ke dalam huruf kecil (*lower case*), selanjutnya dilakukan proses *filtering* teks. *Filtering* adalah tahapan pemrosesan teks dimana semua teks selain dari karakter "a-z", spasi, angka dan .% nantinya akan dihilangkan.

2.2.5 *Stopword Removal*

Stopword removal adalah sebuah proses penghilangan kata *stopword*. *Stopword* adalah kata – kata yang sering kali muncul dalam dokumen namun arti dari kata – kata tersebut tidak deskriptif dan tidak memiliki keterkaitan dengan tema tertentu [15].

2.3 Ekstraksi Fitur

Pada tahap ini dilakukan ekstraksi fitur terhadap setiap kalimat di dokumen lalu dilakukan pembobotan untuk setiap kalimat yang nantinya akan digunakan sebagai bobot fitur untuk perhitungan *Vikor*. Pada peringkasan teks otomatis terdapat berbagai fitur yang dapat digunakan. Dalam penelitian ini menggabungkan fitur yang dianggap penting dalam penelitian lainnya dan akurasinya dalam penggabungan fitur – fitur dianggap baik [6] [13]. Fitur-fitur tersebut terdiri dari :

1. Posisi Kalimat (F1)

Fitur ini merupakan fitur vital yang digunakan dalam menentukan bermaknanya suatu kalimat dalam dokumen. Dalam dokumen ilmiah, kalimat penting dalam suatu paragrafnya biasanya berada pada bagian awal dan akhirnya. Nilainya dihitung dengan rumus (1) sebagai berikut [9]:

$$Posisi\ Kalimat = Max\ \frac{1}{i}, \frac{1}{n-i+1} \qquad (1)$$

Dimana :

n = jumlah kalimat

i = urutan kalimat

2. Frekuensi Kata (F2)

Kebermaknaan suatu kata dapat ditentukan melalui frekuensi kemunculannya di suatu paragraph. Untuk itu digunakan sebuah metode untuk mengekstrak frekuensi kata dari paragraph yaitu tf-isf. TF-isf adalah sebuah metode yang digunakan untuk mengevaluasi kebermaknaan suatu kata di sebuah kalimat dokumen. Tf-isf dapat dihitung dengan rumus (2) dan (3) berikut [9]:

$$isf = log \frac{N}{n_l} \quad (2)$$

Dimana :

N = jumlah kalimat di dokumen

n_l = jumlah kalimat yang mengandung kata ke-i

$$Frekuensi\ Kata = Tf_i * isf \quad (3)$$

Dimana : Tf_i = jumlah kata pada kalimat ke-i

3. Kata kunci (F3)

Fitur ini akan menghitung nilai kata kunci, dimana kata kunci didapatkan dari 10 kata yang sering muncul di sebuah teks. Setelah kata kunci ditemukan, setiap kata yang sama dengan kata kunci akan diberi nilai 0,1 dan yang tidak sesuai akan diberi nilai 0 [10].

4. Kemiripan dengan judul (F4)

Judul mengandung kumpulan kata yang memberikan arah penting tentang konsep dari paragraf. Bila kata-kata dalam sebuah kalimat memiliki banyak kemiripan dengan kata pada judul maka nilai untuk kalimat tersebut dihitung sebagai berikut [11]:

$$Kemiripan\ dengan\ judul = \frac{|Kata\ di\ judul \cap kata\ di\ Kalimat(i)|}{|Kata\ di\ judul \cup kata\ di\ Kalimat(i)|} \quad (4)$$

5. Sentralitas (F5)

Sentralitas dari sebuah kalimat menyatakan kemiripannya ke kalimat lain. Bila sebuah kalimat memiliki sentralitas yang lebih tinggi, maka kalimat tersebut merupakan kandidat terbaik untuk di masukkan ke hasil ringkasan. Nilainya dihitung sebagai berikut [12]:

$$Sentralitas = \frac{|Kata\ di\ Kalimat(i) \cap kata\ di\ Kalimat(n-i)|}{|Kata\ di\ Kalimat(i) \cup kata\ di\ Kalimat(n-i)|} \quad (5)$$

6. Kalimat yang mengandung data numerik (F6)

Pada peringkasan teks haruslah mempertimbangkan data numerik, karena dalam kalimat yang berisi data numerik terdapat kaimat penting [11].

$$Numerik = \frac{\sum data\ angka\ dalam\ kalimat\ K}{length\ (K)} \quad (6)$$

7. Koneksi antar Kalimat dan Penjumlahan Bobot Koneksi antar Kalimat (F7)

Koneksi antar kalimat adalah banyaknya kalimat yang memiliki kata yang sama dengan kalimat lain dalam satu dokumen [13].

$$score\ f_i\ \ K\ \ =\ \ jumla\square koneksi\ antar\ -\ \ kalimat \quad (7)$$

Lalu menjumlahkan bobot dari koneksi antar kalimat.

$$Score\ f_i\ \ K\ \ =\ \ \sum Koneksi\ antar\ kalimat \quad (8)$$

2.4 Pembobotan Fitur

Pemberian bobot fitur pada metode VIKOR sangatlah berpengaruh. Karena bobot fitur yang diberikan akan digunakan sebagai nilai W pada perhitungan metode VIKOR. Pemberian bobot ini pada umumnya diberikan oleh pakar [4], dimana pakar ini dianggap orang yang berpengalaman dalam bidang terkait. Dalam kasus ini peringkasan teks otomatis ini bobot fitur diberikan oleh seorang pakar dari bidang sastra bahasa Indonesia yaitu Mahmud Fasya M.A. dari Universitas Pendidikan Indonesia (UPI).

2.5 Metode VIKOR

Metode VIKOR adalah sebuah *Multi Criteria Decision Making* (MCDM) yang algoritmanya berfungsi untuk menentukan rangking dari alternatif – alternatif yang ada dengan melihat hasil dari nilai – nilai *utilitas* dan *regrets* dari setiap alternative yang ada. Adapun langkah – langkah yang harus dilakukan untuk melakukan perhitungan pada metode VIKOR [11].

2.5.1 Normalisasi Fitur

Melakukan normalisasi matriks untuk menghasilkan sebuah matriks *decision* yang baru dari data kalimat dan fitur – fitur yang sudah didapatkan dari proses sebelumnya. Normalisasi fitur ini diterapkan dengan perhitungan (9) berikut :

$$f_{ij} = \frac{L_{ij}}{\sum_{i=1}^{m} L_{ij}^2}, i = 1, \dots, m; j = 1, \dots, n \qquad (9)$$

Dimana Lij adalah elemen matrik pada baris ke i dari kolom j dan m adalah jumlah fitur, pada kasus ini terdapat 7 fitur.

2.5.2 Menentukan Solusi Ideal

Menentukan solusi positif ideal dan solusi negative ideal. Solusi positif ideal adalah ketika suatu fitur berindeks besar dan dinyatakan itulah yang terbaik dan solusi negative ideal adalah ketika suatu fitur berindeks kecil dan dinyatakan itulah yang terbaik. Menentukan solusi ideal ini dapat dihitung dengan rumus (10) sebagai berikut :

$$A^* = \{\max f_{ij} | i = 1,2, \dots, m$$

$$= \{f_1^*, f_2^*, \dots, f_i^*, \dots, f_n^*\} \qquad (10)$$

$$A^- = \{\min f_{ij} | i = 1,2, \dots, m$$

$$= \{f_1^-, f_2^-, \dots, f_i^-, \dots, f_n^-\}$$

Dimana : A* = nilai maksimal dari fitur (f) yang telah dinormalisasi

A^- = nilai minimal dari fitur (f) yang telah dinormalisasi

2.5.3 Menghitung Ukuran *Utilitas* (S) dan Ukuran *Regret* (R)

Menghitung nilai atau ukuran dari *utilitas* (S) dan *regret* (R) dari setiap kriteria yang digunakan. Ukuran ukuran dari *utilitas* (S) dan *regret* (R) dari setiap kriteria dalam alternative dapat diperoleh dengan rumus (11) dan (12) sebagai berikut :

$$S_i = \sum_{j=1}^{n} W_j \frac{(f_j^* - f_{ij})}{(f_j^* - f_j^-)} \qquad (11)$$

$$R_i = max_j \; W_j \frac{f_j^* - f_{ij}}{f_j^* - f_j^-} \qquad (12)$$

Dimana : W = bobot fitur

f_{ij} = nilai fitur dari hasil normalisasi fitur

f^* = nilai ideal fitur maksimal

f^- = nilai ideal fitur minimal

S_i = nilai Utilitas kalimat ke -i

R_i = nilai terbesar dari dalam perhitungan utilitas ke -i

2.5.4 Menghitung Indeks Metode VIKOR

Indeks VIKOR digunakan untuk menampilkan kalimat terbaik yang akan dijadikan hasil ringkasan. Menghitung indeks VIKOR dengan cara mengsubstitusikan *Utilitas (Si)* dan *Regret (Ri)* ke dalam persamaan dengan indeks VIKOR pada alternatif ke-i seperti pada persamaan (13), dimana nilai v pada penelitian ini yang digunakan adalah 0,1 – 0,9.

$$Q_i = v \frac{S_i - S^-}{(S^* - S^-)} + (1 - v) \frac{R_i - R^-}{(R^* - R^-)} \quad (13)$$

Dimana v = nilai bobot indeks vikor yang telah ditentukan (0,1 – 0,9)

Ki = nilai utilitas kalimat ke- i

S^* = nilai maksimal *utilitas*

S^- = nilai minimal *utilitas*

Ri = nilai *regret* pada kalimat ke - i

R^* = nilai maksimal *regret*

R^- = nilai minimal *regret*

2.5.5 Merangking Berdasarkan Indeks VIKOR

Nilai indeks VIKOR yang diperoleh pada tahapan sebelumnya, selanjutnya akan dirangking untuk menentukan kalimat terbaik untuk hasil ringkasan. Perangkingan hasil ringkasan metode VIKOR ditentukan berdasarkan nilai indeks VIKOR terkecil [8].

2.6 Teknik Evaluasi Peringkasan Teks

Evaluasi peringkasan teks adalah cara untuk mengetahui hasil dari sebuah peringkasan itu baik oleh peringkasan secara manual ataupun memiliki tingkat akurasi yang baik atau tidak untuk peringkasan teks otomatis. Sebuah hasil ringkasan dapat dikatakan sudah tepat oleh sebagian orang, tetapi menurut sebagian orang lagi tidak demikian. Secara umum teradapat dua jenis metode evaluasi ringkasan, yaitu [1]:

1. Ekstrinsik yaitu kualitas sebuah ringkasan yang diukur berdasarkan bagaimana ringaksan tersebut dapat membantu penyelesaian tugas orang yang membaca ringkasan tersebut.

2. Intrinsik yaitu kualitas sebuah ringkasan yang diukur dari kualitas keluaran ringaksan yang dihasilkan.

Evaluasi peringkasan yang digunakan adalah evaluasi intrinsik. Salah satu metode untuk mengukur hasil ringkasannya adalah *ROUGE*. *ROUGE* mengukur kualitas hasil ringaksan dengan menghitung unit – unit yang *overlap* seperti urutan kata dan pasangan – pasangan kata antara ringkasan kandidat dan ringkasan hasil yang menjadi referensi *ROUGE* [16]. Evaluasi yang diukur adalah *overlap* dari isi atau disebut *recall* dan *precission* kalimat. Kombinasi antara nilai *recall* dan *precission* menhasilkan f-measure.

2.6.1 *Recall*

Recall merupakan istilah yang digunakan untuk kalimat terpanggil yang relevan dengan pernyataan atau vector *query*. Perbandingan jumlah informasi relevan yang didapatkan sistem dengan jumlah seluruh informasi relevan yang ada dalam koleksi informasi. *Recall* berhubungan dengan kemampuan dalam menemukan kalimat yang relevan [17]. Hal ini berarti *recall* adalah bagian dari proses evaluasi yang dapat digunakan sebagai alat ukur relevannya sebuah ringkasan.

$$\boldsymbol{recall} = \frac{hits}{hits+misses} \quad (14)$$

Dimana $\square its$ = Jumlah kata pada kalimat yang berhasil diekstrak sistem yang sesuai dengan hasil ekstrak manusia

$misses$ = Jumlah kata pada kalimat yang diekstrak manusia tetapi tidak terdapat dalam hasil ekstrak sistem

2.6.2 Precission

Precission adalah tingkat ketepatan hasil ringkasan. Perbandingan jumlah informasi relevan yang didapatkan sistem dengan jumlah seluruh informasi yang terambil oleh sistem baik yang relevan maupun tidak.

$$Precission = \frac{hits}{hits+noice} \quad (15)$$

Dimana $\square its$ = Jumlah kata pada kalimat yang berhasil diekstrak sistem yang sesuai dengan hasil ekstrak manusia
$noice$ = Jumlah kata pada kalimat yang diekstrak sistem tetapi tidak terdapat dalam kalimat yang diekstrak manusia

2.6.3 f-measure

f-measure adalah hubungan antara *recall* dan *precission* yang memperlihatkan hasil akurasi sistem. Nilai *recall* dan *precission* pada suatu keadaan dapat memiliki bobot yang berbed. Ukuran yang menampilkan timbal balik antara *recall* dan *precission* adalah *f-measure* yang merupakan bobot *harmonic mean* dari *recall* dan *precission* [18].

$$\textbf{\textit{f-measure}} = \frac{2 * precission * recall}{precission + recall} \quad (16)$$

Dimana *precission* = hasil dari perhitungan *precission*
Recall = hasil dari perhitungan *recall*

2.7 Pemrograman Berorientasi Objek

Pemrograman berorientasi objek (Inggris: *object-oriented programming* disingkat OOP) merupakan paradigma pemrograman yang berorientasikan kepada objek. Ini adalah jenis pemrograman di mana programmer mendefinisikan tidak hanya tipe data dari sebuah struktur data, tetapi juga jenis operasi (fungsi) yang dapat diterapkan pada struktur data. Dengan cara ini, struktur data menjadi objek yang meliputi data dan fungsi. Selain itu, pemrogram dapat membuat hubungan antara satu benda dan lainnya. Sebagai contoh, objek dapat mewarisi karakteristik dari objek lain.

Salah satu keuntungan utama dari teknik pemrograman berorientasi obyek atas teknik pemrograman prosedural adalah bahwa memungkinkan programmer untuk membuat modul yang tidak perlu diubah ketika sebuah jenis baru objek

ditambahkan. Seorang pemrogram hanya dapat membuat objek baru yang mewarisi banyak fitur dari objek yang sudah ada. Hal ini membuat program object-oriented lebih mudah untuk memodifikasi. Konsep dasar berorientasi objek diantaranya [19]:

1. Kelas (*Class*) adalah kumpulan objek-objek dengan karakteristik yang sama. Kelas merupakan definisi statik dan himpunan objek yang sama yang mungkin lahir atau diciptakan dari kelas tersebut. Sebuah kelas akan mempunyai sifat (atribut), kelakuan (operasi/metode), hubungan (relationship) dan arti. Suatu kelas dapat diturunkan dan kelas semula dapat diwariskan ke kelas yang baru.
2. Objek (*Object*) adalah abstraksi dan sesuatu yang mewakili dunia nyata seperti benda, manusia, satuan organisasi, tempat, kejadian, strutur, status, atau hal-hal lain yang bersifat abstrak. Objek merupakan suatu entitas yang mampu menyimpan informasi (status) dan mempunyai operasi (kelakuan) yang dapat diterapkan atau dapat berpengaruh pada status objeknya. Objek mempunyai siklus hidup yaitu diciptakan, dimanipulasi, dan dihancurkan.
3. Metode (*Method*) adalah operasi atau metode pada sebuah kelas hampir sama dengan fungsi atau prosedur pada terstruktur. Sebuah kelas boleh memiliki lebih dari satu metode atau operasi. Metode atau operasi yang berfungsi untuk memanipulasi objek itu sendiri.
4. Atribut (*Attribute*) dari sebuah kelas adalah variabel global yang dimiliki sebuah kelas. Atribut dapat berupa nilai atau elemen-elemen data yang dimiliki oleh objek dalam kelas objek. Atribut secara individual oleh sebuah objek, misalnya berat, jenis, nama, dan sebagainya.
5. Abstraksi (*Abstraction*) merupakan prinsip untuk merepresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan.
6. Enkapulasi (*Encapsulation*) adalah pembungkusan atribut data dan layanan (operasi-operasi) yang dipunyai objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerja.

7. Pewarisan (*Inheritance*) adalah mekanisme yang memungkinkan satu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dari dirinya.
8. Antarmuka (*Interface*) sangat mirip dengan kelas, tetapi tanpa atribut kelas dan tanpa memiliki metode yang dideklarasikan. Antarmuka biasanya digunakan agar kelas lain tidak langsung mengakses ke suatu kelas.
9. Generalisasi dan Spesialisasi menunjukkan 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 dan motor.
10. Komunikasi antar objek dilakukan lewat pesan (*message*) yang dikirim dan satu objek ke objek lainnya.
11. Polimorfisme (*Polymorphism*) adalah kemampuan suatu objek untuk digunakan dibanyak tujuan yang berbeda dengan nama yang sama sehingga menghemat baris program.
12. Package adalah sebuah kontainer atau kemasan yang daoat digunakan untuk mengelompokkan kelas-kelas sehingga memungkinkan beberapa kelas yang bernama sama disimpan dalam package yang berbeda [19].

2.8 UML

Pendekatan pembangunan perangkat lunak yang digunakan dalam penelitian ini adalah pendekatan beorientasi objek menggunakan unified modeling language. Unified Modeling Language (UML) adalah bahasa spesifikasi standar untuk mendokumentasikan, menspesifikasikan, dan membangun sistem perangkat lunak. Unified Modeling Language (UML) adalah himpunan struktur dan teknik untuk pemodelan desain program berorientasi objek (OOP) serta aplikasinya. UML adalah metodologi untuk mengembangkan sistem OOP dan sekelompok perangkat tool untuk mendukung pengembangan sistem tersebut. UML merupakan dasar bagi perangkat (tool) desain berorientasi objek dari IBM. UML dikembangkan sebagai suatu alat untuk analisis dan desain berorientasi objek oleh Grady Booch, Jim

Rumbaugh, dan Ivar Jacobson. Namun demikian UML dapat digunakan untuk memahami dan mendokumentasikan setiap sistem informasi. Penggunaan UML dalam industri terus meningkat. Ini merupakan standar terbuka yang menjadikannya sebagai bahasa pemodelan yang umum dalam industri peranti lunak dan pengembangan sistem,Dalam Penelitian ini UML yang digunakan sebagai berikut [19]:

2.8.1 *Use case Diagram*

Use case Diagram untuk memodelkan proses bisnis. Dalam konteks UML, tahap konseptualisasi dilakukan dengan pembuatan *use case diagram* yang sesungguhnya merupakan deskripsi peringkat tinggi bagaimana perangkat lunak (aplikasi) akan digunakan oleh penggunanya. Selanjutnya, *use case diagram* tidak hanya sangat penting pada analisis, tetapi juga sangat penting untuk perancangan (*design*), untuk mencari (mencoba menemukan) kelas-kelas yang terlibat dalam aplikasi, dan untuk melakukan pengujian (*testing*). Membuat *use case diagram* yang komprehensif merupakan hal yang sangat penting dilakukan pada tahap analisis. Dengan menggunakan *use case diagram*, akan mendapatkan banyak informasi yang sangat penting yang berkaitan dengan aturan –aturan bisnis. Dalam hal ini, setiap objek yang berinteraksi dengan sistem/perangka lunak (misalnya, orang, suatu perangkat keras, sistem lain, dan sebagainya) merupakan *actor* untuk sistem/perangkat lunak yang dibangun, sementara *use case* merupakan deskripsi lengkap tentang bagaimana sistem/perangkat lunak berprilaku untuk para *actor*-nya. Dengan demikian, *use case* program merupakan deskripsi lengkap tentang interaksi yang terjadi antara para *actor* dengan sistem/perangkat lunak yang sedang dibangun. Ketika mengembangkan *use case diagram*, hal yang pertama dilakukan adalah mengenali *actor* untuk sistem/aplikasi yang sedang dikembangkan. Dalam hal ini, ada beberapa karakteristik untuk para *actor*, yaitu *actor* ada diluar sistem yang sedang dikembangkan dan *actor* berinteraksi dengan sistem yang sedang dikembangkan.

2.8.2 *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 (internal processing). Oleh karena itu *activity diagram* tidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum. Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas. *Use case diagram* merupakan gambaran menyeluruh dan pada umumnya sangatlah tidak terperinci. Oleh karena itu, harus diperinci perilaku sistem untuk masing-masing *use case* yang ada. Penggunaan *activity diagram* dapat memberikan gambaran secara meyeluruh terhadap perangkat lunak yang dibangun.

2.8.3 Class *Diagram*

Class *diagram* merupakan *diagram* yang selalu ada di permodelan sistem berorientasi objek. Class *diagram* menunjukkan hubungan antar class dalam sistem yang sedang dibangun dan bagaimana mereka saling berkolaborasi untuk mencapai suatu tujuan. classs *diagram* menggambarkan interaksi dan relasi antar kelas yang ada di dalam suatu sistem. Kelas memiliki atribut dan metode. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas. Metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas. Atribut dan metode dapat memiliki salah satu sifat sebagai berikut :

1. Private, tidak dapat dipanggil dari luar kelas yang bersangkutan.
2. Protected, hanya dapat dipanggil oleh kelas yang bersangkutan dan anak -anak yang mewarisinya.
3. Public, dapat dipanggil oleh siapa saja.

Class *diagram* menggambarkan relasi atau hubungan antar kelas dari sebuah sistem. Berikut ini beberapa gambaran relasi yang ada dalam class *diagram*:

1. Assiciation

Hubungan antar class yang statis. Class yang mempunyai relasi asosiasi menggunakan class lain sebagai atribut pada dirinya.

2. Aggregation

Relasi yang membuat class yang saling terikat satu sama lain namun tidak terlalu berkegantungan.

3. Composition

Relasi agregasi dengan mengikat satu sama lain dengan ikatan yang sangat kuat dan saling berkegantungan.

4. Dependency

Hubungan antar class dimana class yang memiliki relasi dependency menggunakan class lain sebagai atribut pada method.

5. Realization

Hubungan antar class dimana sebuah class memiliki keharusan untuk mengikuti aturan yang ditetapkan class lainnya.

2.9 Perangkat Lunak Pendukung

Pada bagian ini akan dijelaskan tentang beberapa perangkat lunak dan bahasa pemrograman yang digunakan untuk mendukung dalam pengembangan aplikasi yang dibuat. Bahasa pemrograman yang digunakan diantaranya adalah java. Selain itu terdapat perangkat lunak yang digunakan yaitu Netbeans.

1. Pengertian Java

Java merupakan bahasa pemrograman berorientasi objek dan bebas platform yang awalnya dikembangkan oleh SUN Micro System dengan sejumlah keunggualn yang memungkinkan Java dijadikan sebagai bahasa pengembangan Enterprise. Beberapa keunggulan Java diantaranya Berbasis GUI, Berorientasi Objek, Dapat membangun Aplikasi Web, Multiplatform, Secured, Bisa digunakan pada Aplikasi Jaringan Terdistribusi, Mendukung Software Mission-Critical dan Multithread [6].

2. Pengertian Netbeans

NetBeans adalah IDE (Intergrated Development Environment), yang merupakan sebuah program/alat bantu yang terdiri atas editor, compiler, debuger dan *design* yang terintegrasi dalam satu aplikasi. NetBeans dibuat dan dikembangkan oleh Sun Microsystem dan umumnya digunakan untuk membuat dan mengembangkan aplikasi dengan bahasa pemrograman Java. Salah satu keunggulan dari NetBeans karena didukung dengan fasilitas drag and drop komponen, dan dukungan Rapid Application Development (pemrograman berbasis visual dan template) [6].