BAB 2
LANDASAN TEORI

2.1 Peringkasan Teks Otomatis

Peringkasan teks otomatis (*automatic text summarization*) adalah pembuatan versi yang lebih singkat dari sebuah teks dengan memanfaatkan aplikasi yang dijalankan pada komputer. Hasil peringkasan ini mengandung poin-poin penting dari teks asli. Ringkasan didefinisikan sebagai sebuah teks yang dihasilkan dari satu atau lebih teks, mengandung informasi dari teks asli dan panjangnya tidak lebih dari setengah teks asli [1].

Terdapat dua pendekatan pada peringkasan teks, yaitu ekstraksi (*shallower approaches*) dan abstraksi (*deeper approaches*). Pada teknik ekstraksi, sistem menyalin informasi yang dianggap paling penting dari teks asli menjadi ringkasan (sebagai contoh, klausa utama, kalimat utama, atau paragraf utama). Sedangkan teknik abstraksi melibatkan parafrase dari teks asli. Pada umumnya, abstraksi dapat meringkas teks lebih kuat daripada ekstraksi, tetapi sistemnya lebih sulit dikembangkan karena mengaplikasikan teknologi *natural language generation* yang merupakan bahasan yang dikembangkan tersendiri [1].

Berdasarkan jumlah sumbernya, ringkasan teks dapat dihasilkan dari satu sumber (*single-document*) atau dari banyak sumber (*multi-document*). Suatu ringkasan dapat bersifat *general*, yaitu ringkasan yang berupaya mengambil sebanyak mungkin informasi umum yang mampu menggambarkan keseluruhan isi teks. Selain itu dapat juga informasi yang diambil untuk ringkasan berdasar pada *query* yang didefinisikan *user*. 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*) [1].

2.2 Peringkasan Multi Dokumen

Peringkasan multi-dokumen merupakan ektraksi informasi dari beberapa dokumen yang mempunyai tema yang sama. Metode yang digunakan untuk meringkas multi-dokumen ini dengan membuat ringkasan single-dokumen yang kemudian digabungkan hasil ringkasannya menjadi satu.

2.3 *Text Pre-Processing*

Text Pre-Processing adalah suatu tahapan yang dilakukan untuk mempersiapkan teks sumber menjadi data yang akan diolah pada tahapan selanjutnya [5]. 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*

Memecah teks dokumen sumber menjadi bentuk kalimat–kalimat, ini merupakan tahapan awal *preprocessing*. Tokenizing kalimat adalah suatu proses untuk memecahkan string teks dokumen yang panjang menjadi kalimat–kalimat. Delimiter yang digunakan untuk memotong string teks dokumen menjadi kalimat adalah titik ".", tanda tanya "?" dan tanda seru "!" dengan menggunakan fungsi split() [6].

2.2.2 *Tokenizing* kata

Tokenizing adalah proses pemotongan string input berdasarkan tiap kata yang menyusunnya. Pemecahan kalimat menjadi kata-kata tunggal dilakukan dengan mencari kalimat dengan pemisah (delimiter) white space (spasi, tab, dan newline) [7].

2.2.3 Filtering

Data teks dalam dokumen yang sebelumnya sudah diubah ke dalam huruf kecil semua. Selanjutnya dilakukan proses filtering teks. Filtering adalah tahapan pemrosesan teks untuk menghilangkan tanda baca dan simbol [7].

2.2.4 *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. Misalnya "di", "oleh", "pada", "sebuah", "karena" dan lain sebagainya. Untuk itu digunakan sebuah kamus stopword yang didapat dari referensi *online* [8].

2.3 *TextRank*

TextRank merupakan salah satu model perankingan berbasis graf untuk diterapkan dalam graf tekstual. Algoritma perankingan berbasis graf adalah cara untuk menentukan seberapa penting sebuah vertex dalam graf, berdasarkan informasi yang diambil dari struktur graf keseluruhan secara rekursif. Dalam penerapannya pada peringkasan teks otomatis, TextRank mengekstraksi kalimat-kalimat beranking tinggi untuk dimasukkan ke dalam ringkasan.

2.3.1. Teks Sebagai Sebuah Graf

Pada peringkasan teks dengan metode pemeringkatan berbasis graf, vertex merepresentasikan kalimat-kalimat yang berada dalam dokumen sedangkan edge merepresentasikan similarity antara dua kalimat yang saling berelasi. Setiap kalimat tunggal yang diperoleh dari proses pemisahan kalimat dianggap sebagai vertex dalam graf. Pada algortime Textrank formula yang digunakan untuk menghitung similarity adalah Content Overlap. Formula Content Overlap dituliskan sebagai berikut [9]:

$$Similarity\ (\mathrm{Si}, \mathrm{Sj}) = \frac{|\{\mathrm{Wk} \mid \mathrm{Wk} \in \mathrm{Si}\ \&\ \mathrm{Wk} \in \mathrm{Sj}\}|}{\log(|Si|)+\log(|\mathrm{Sj}|)} \quad \textbf{(2.1)}$$

Dimana S_i dan S_j merupakan kalimat yang ingin dicari similaritasnya, W_k adalah jumlah kata yang sama antarkalimat S_i dengan kalimat S_j, S_i dan S_j adalah representasi panjang kalimat. Formula tersebut diberlakukan ke semua kalimat tunggal yang ada untuk mendapatkan bobot yang menghubungkan antarkalimat.

2.3.2. Proses Pemeringkatan

Proses pemeringkatan bertujuan mendapatkan bobot akhir dari suatu vertex. Semakin besar bobot akhir dari suatu vertex, maka semakin banyak informasi di dalamnya. Pada algoritma TextRank, proses pemeringkatan dilakukan dengan menggunakan formula untuk graf tidak berarah dan terboboti seperti yang dituliskan sebagai berikut [9]:

$$\mathrm{WS}(\mathrm{S_i}) = (1-\mathrm{d}) + \mathrm{d} * \sum_{\mathrm{Vj}\in\mathrm{Adj(Vi)}} \frac{\mathrm{W_{ji}}}{\sum_{\mathrm{Vk}\in\mathrm{Adj(Vj)}} wjk} \mathrm{WS}(\mathrm{S_j}) (\mathbf{2.2})$$

S_i = *vertex* yang akan dihitung skornya; S_j = vertex yang bertetanggan dengan S_i; S_k = vertex yang bertetanggan dengan S_j; W_{ji} = nilai similarity antara S_j dan Si; berdasarkan penelitian Ian Rogers pada tahun 2002, menganjurkan bahwa dumping factor (d) untuk diisi dengan 0.85 dan inisialisasi bobot kalimat $\mathrm{WS}(\mathrm{S_j})$ adalah 1 [10]. Setelah didapatkan bobot akhir setiap kalimat, urutkan mulai dari yang terbesar sampai yang terkecil.

2.3.3. Ekstraksi Kalimat

Proses ekstraksi kalimat dilakukan dengan mengambil 25% kalimat dari total kalimat yang ada di dokumen. Kalimat yang diekstraksi adalah kalimat yang memiliki peringkat paling tinggi. 25% hasil ekstraksi dari teks sumber memiliki 84% kalimat penting yang dipilih oleh para ahli untuk menjadi ringkasan. Hasil ekstraksi kalimat dijadikan sebagai hasil ringkasan [11].

2.3.4. Metode TF-IDF (Term Frequency – Inversed Document Frequency)

Metode Term Frequency-Inverse Document Frequency (tf-idf) adalah cara pemberian bobot hubungan suatu kata (term) terhadap dokumen. Untuk dokumen tunggal tiap kalimat dianggap sebagai dokumen. Metode ini menggabungkan dua konsep untuk perhitungan bobot, yaitu term frequency (tf) merupakan frekuensi

kemunculan term j pada dokumen i. Document frequency (df) adalah banyaknya kalimat dimana suatu term j muncul.

Dalam tf frekuensi term pilihan paling sederhana adalah dengan menggunakan frequensi baku dalam dokumen, yaitu berapa kali term (t) terjadi dalam dokumen (d) [12]. Nilai idf sebuah term (kata) dapat dihitung menggunakan dalam persamaan (2.3).

$$IDF = \log\left(\frac{N}{idf}\right) (\mathbf{2.3})$$

N adalah jumlah dokumen yang berisi term (t) dan idf adalah jumlah kemunculan (frekuensi) term terhadap N. Adapun algoritma yang digunakan untuk menghitung bobot (w) masing-masing dokumen terhadap kata kunci (query) menggunakan persamaan (2.4).

$$W_{d.t} = tf_{d.t} * IDF_t \ (\mathbf{2.4})$$

Dimana :

d : dokumen ke-n

t : term ke-t dari kata kunci

tf : term frekuensi/ frekuensi kata

W : bobot dokumen ke-d terhadap term ke-t

2.3.5. Cosine Similarity

Cosine similarity digunakan untuk menghitung pendekatan relevansi query terhadap dokumen. Semakin besar nilai kesamaan vektor query dengan vektor dokumen maka query tersebut dipandang semakin relevan dengan dokumen. Cosinus dari 00 adalah 1, dan kurang dari 1 untuk setiap sudut lainnya. Dengan demikian dua vektor dengan orientasi yang sama memiliki kesamaan cosinus dari 1, dua vektor pada 900 memiliki kesamaan 0. Cosine similarity terutama digunakan dalam ruang positif, dimana hasilnya dibatasi (0,1). Cosine similarity kemudian memberi tolak ukur seberapa mirip dua dokumen .

Cosinus dari dua vektor dapat diturunkan dengan menggunakan dot product Euclidean yaitu $S_i . S_j = \| S_i \| \| S_j \| \cos \Theta$ dimana vektor atribut, S_i dan S_j kesamaan dan besarnya ditukan dengan persamaan berikut [12].

$$\mathrm{CS}(S_i , S_j) = \frac{\Sigma_{t=1}^{n} W_{t.S_i} W_{t.S_j}}{\sqrt{\Sigma_{t=1}^{n} W_{t.S_i}^2} \sqrt{\Sigma_{t=1}^{n} W_{t.S_j}^2}} \quad (\mathbf{2.5})$$

Dimana :

t : term dalam Kalimat

$W_{t.S_i}$: bobot term t dalam blok S_i

$W_{t.S_j}$: bobot term t dalam blok S_j

2.4 Teknik Evaluasi Peringkasan Teks

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* [13]. Evaluasi yang diukur adalah *overlap* dari isi atau disebut *recall* dan *precission* kalimat. Kombinasi antara nilai *recall* dan *precission* menhasilkan f-measure.

2.4.1. 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. Hal ini berarti *recall* adalah bagian dari proses evaluasi yang dapat digunakan sebagai alat ukur relevannya sebuah ringkasan.

$$\boldsymbol{recall} = \frac{hits}{hits + misses} \quad (\mathbf{2.6})$$

Dimana $hits$ = 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

2.4.2. 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.

$$Precission = \frac{hits}{hits + noice} \quad (2.7)$$

Dimana $hits$ = 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

2.4.3. 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* [13].

$$\textbf{\textit{f-measure}} = \frac{2 * precission * recall}{precission + recall} \quad (2.8)$$

Dimana *precission* = hasil dari perhitungan *precission*

Recall = hasil dari perhitungan *recall*

2.4.4. Teknik Evaluasi Akurasi

Dalam penelitian ini, metode evaluasi yang digunakan adalah metode intrinsik penghitungan *Accuracy* yang menurut Zaman B dan E Winarko merupakan standar evaluasi dalam penghitungan information retrieval. Evaluasi perhitungan information retrieval dengan menggunakan *Accuracy* juga dapat digunakan dalam evaluasi perhitungan peringkas teks otomatis.

$$Accuracy = \frac{\sum Kalimat\ Sistem\ yang\ relevan\ dengan\ Hasil\ Manual}{\sum Kalimat\ Sistem\ yang\ muncul} \quad (\mathbf{2.9})$$

2.4.5. Teknik Evaluasi Kohesi

Dalam peringkasan teks multi dokumen, dilakukan dengan perhitungan kohesi untuk penyusunan ringkasan beberapa dokumen, berikut adalah rumus dari evaluasi untuk kohesi.

$$CoH = \frac{Log(C_s * 9+1)}{Log\ (M * 9+1)} \quad (\mathbf{2.10}) \qquad Ns = \frac{(o)*(o-1)}{2} \quad \mathbf{(2.11)}$$

$$C_s = \frac{\Sigma_{\forall Si,Sj \epsilon Summary}\ Similarity\ (S_i,S_j)}{N_s} \quad \mathbf{(2.12)}$$

$$M = maxSimilarity\ (S_i,S_j), S_i,S_j \leq N \quad \mathbf{(2.13)}$$

Dimana CoH merupakan kohesi dari ringkasan, Cs merupakan rata-rata similarity dari semua kalimat di hasil ringkasan S, $Similarity\ (S_i,S_j)$merupakan cosine similarity antara kalimat Si dan Sj, Ns adalah nomor tidak nol dari hubungan similarity pada ringkasan, O merupakan nomor kalimat pada ringkasan, M sesuai dengan nilai maksimum similarity pada kalimat di dokumen dan N merupakan nomor kalimat pada dokumen.

2.6 Pemrograman Berorientasi Objek

OOP (Object Oriented Programming) adalah suatu metode pemrograman yang berorientasi kepada objek. Semua data dan fungsi ini dibungkus dalam kelas-kelas

atau objek-objek. Bandingkan dengan logika pemrograman terstruktur. Setiap objek dapat menerima pesan, memproses data, dan mengirim pesan ke objek lainnya, Model data berorientasi objek dikatakan dapat memberi fleksibilitas yang lebih, kemudahan mengubah program, dan digunakan luas dalam teknik piranti lunak skala besar. Lebih jauh lagi, pendukung OOP mengklaim bahwa OOP lebih mudah dipelajari bagi pemula dibanding dengan pendekatan sebelumnya, dan pendekatan OOP lebih mudah dikembangkan dan dirawat.

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 [14]:

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 dapat digunakan untuk mengelompokkan kelas-kelas sehingga memungkinkan beberapa kelas yang bernama sama disimpan dalam package yang berbeda [14].

2.7 UML

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 mulai diperkenalkan oleh Object Management Group, sebuah organisasi yang telah mengembangkan model, teknologi, dan standar OOP sejak tahun 1980-an. Sekarang UML sudah mulai banyak digunakan oleh para praktisi OOP. UML merupakan dasar bagi perangkat (tool) desain berorientasi objek dari IBM.

UML adalah suatu bahasa yang digunakan untuk menentukan, memvisualisasikan, membangun, dan mendokumentasikan suatu sistem informasi. 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 [15].

2.6.1. Use Case Diagram

Use Case Diagram untuk memodelkan proses bisnis. Dalam konteks UML, tahap konseptualisasi dilakukan dengan pembuatan use cas 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 verinteraksi dengan sistem yang sedang dikembangkan [15].

2.6.2. 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 [15].

2.6.3. 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.

2.7 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 [15].

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) [15] .