BAB 2

LANDASAN TEORI

2.1 Plagiarisme

Pada bagian ini akan dibahas mengenai definisi mengenai plagiarisme itu sendiri beserta jenis-jenis teknik yang lazim digunakan dalam plagiarisme, batasan-batasan plagiarisme dan juga contoh-contoh plagiarisme yang pernah terjadi pada dunia nyata.

2.1.1 Definisi

Kata plagiarisme berasal dari bahasa Latin "*plagiare*" yang berarti mencuri. Plagiarisme adalah teknik penyalinan atau meniru karya orang lain yang diklaim menjadi hasil karya sendiri (tanpa seijin penulis aslinya).

Dalam Kamus Besar Bahasa Indonesia disebutkan bahwa *"Plagiat adalah pengambilan karangan orang lain dan menjadikannya seolah-olah karangan sendiri."* sedangkan *"Plagiarisme adalah penjiplakan yang melanggar hak cipta."*

Dalam *Oxford Dictionary* disebutkan bahwa *"Plagiarize is copy another person's work, is=deas, words, etc and pretend that they are your own."*

Menurut wikipedia**,** plagiarisme adalah penjiplakan atau pengambilan karangan, pendapat, dan sebagainya dari orang lain dan menjadikannya seolah karangan dan pendapat sendiri.

Dari definisi-definisi di atas, maka dapat disimpulkan bahwa plagiarisme adalah suatu tindakan penjiplakan karya orang lain dan membuat karya tersebut seolah-olah adalah hasil karya kita sendiri. Tindakan plagiarisme tersebut merupakan suatu bentuk pelanggaran hak cipta sehingga pelaku plagiarisme, yang biasa disebut dengan plagiator,

dapat dijatuhi hukuman karena tindakannya yang secara tidak langsung mencuri karya orang lain.

2.1.2 Jenis-jenis Plagiarisme

Beberapa jenis teknik plagiarisme yang dikenal selama ini, yaitu:

a. *Word-for-word plagiarism*: menyalin setiap kata secara langsung tanpa diubah sedikitpun

b. *Plagiarism of the form of a source*: menyalin dan menulis ulang kode-kode program tanpa mengubah struktur dan jalannya program

c. *Plagiarism of authorship*: mengakui hasil karya orang lain sebagai hasil karya sendiri dengan cara mencantumkan nama sendiri menggantikan nama pengarang sebenarnya.

Ada banyak cara yang dapat diterapkan untuk melakukan transformasi atas suatu kode program. Dua strategi yang paling umum untuk melakukan transformasi ini adalah:

a. Transformasi leksikal

Perubahan leksikal adalah perubahan pada kode program (*source code*). Untuk melakukan transformasi ini, tidak diperlukan pengetahuan akan bahasa pemrograman yang digunakan dalam kode program (*source code*) tersebut. Transformasi jenis ini meliputi:

1. Komentar diubah (ditambah, dikurangi, atau diganti)
2. Format penulisan atau identasi program diubah
3. Nama variable atau *identifier* diubah

b. Transformasi struktural

Perubahan struktural adalah perubahan pada struktur program. Untuk melakukan transformasi ini, diperlukan pengetahuan akan bahasa pemrograman yang digunakan dalam kode program (*source code*) tersebut. Metode ini sangat bergantung pada keahlian seseorang dalam menggunakan bahasa pemrograman tersebut. Transformasi ini meliputi:

1. pengubahan urutan metode yang tidak mengubah jalannya program
2. prosedur diubah menjadi fungsi dan sebaliknya
3. pemanggilan prosedur/fungsi diganti dengan isi prosedur/fungsi itu sendiri
4. pengubahan bentuk pengulangan yang digunakan (*repeat..until* menjadi *while..do*)
5. penggunaan *switch..case* untuk menangani *nested if*

2.1.3 Batasan Plagiarisme

Ada batasan-batasan tertentu yang digunakan untuk membedakan apakah sebuah penggunaan informasi termasuk dalam praktik plagiarisme atau tidak. Hal-hal yang dapat digolongkan sebagai plagiarisme adalah sebagai berikut:

- Mengakui tulisan orang lain sebagai tulisan sendiri
- Mengakui gagasan orang lain sebagai pemikiran sendiri
- Mengakui temuan orang lain sebagai kepunyaan sendiri

- Mengakui karya kelompok sebagai kepunyaan atau hasil sendiri, menyajikan tulisan yang sama dalam kesempatan yang berbeda tanpa menyebutkan asal-usulnya
- Meringkas dan memparafrasekan (mengutip tak langsung) tanpa menyebutkan sumbernya
- Meringkas dan memparafrasekan dengan menyebut sumbernya, tetapi rangkaian kalimat dan pilihan katanya masih terlalu sama dengan sumbernya.

Sementara itu, hal-hal yang tidak tergolong plagiarisme antara lain:

- menggunakan informasi yang berupa fakta umum.
- menuliskan kembali (dengan mengubah kalimat atau parafrase) opini orang lain dengan memberikan sumber jelas.
- mengutip secukupnya tulisan orang lain dengan memberikan tanda batas jelas bagian kutipan dan menuliskan sumbernya.

2.1.4 Faktor-faktor yang Menyebabkan Terjadinya Plagiarisme

Praktik plagiarisme yang terjadi di kalangan mahasiswa dapat terjadi karena berbagai macam faktor, seperti:

1. Kurangnya pengetahuan tentang aturan penulisan karya ilmiah

Dosen seringkali memberikan tugas kepada mahasiswanya untuk membuat tulisan karya ilmiah, seperti makalah. Dalam membuat karya tersebut tidak jarang mahasiswa memerlukan beberapa referensi untuk melengkapi tugas mereka. namun

dalam praktiknya, mereka tidak hanya sekedar menjadikannya referensi, tetapi mereka juga menjadikannya sebagai isi dari tugas mereka. Mereka tidak mencantumkan sumber dari referensi tersebut. Bahkan bukan tidak mungkin seorang mahasiswa melakukan *copy-paste* pekerjaan temannya atau seniornya.

Kurangnya pengetahuan mereka tentang tata cara penulisan karya ilimiah merupakan suatu penyebab terjadinya praktik plagiarisme. Referensi yang hanya sebagai penguat gagasannya hendaknya dia mencantumkan sumber referensi tersebut sebagai penghargaan terhadap orisinalitas sebuah karya tulis. Karena minimnya pengetahuan mahasiswa tentang cara penulisan karya ilmiah, dia tidak mencantumkan sumber dari referensi tersebut sehingga secara tidak sadar dia telah melakukan plagiarisme.

2. Penyalahgunaan teknologi

Kemajuan teknologi telah memperkenalkan internet kepada mahasiswa. Di dalam internet inilah mahasiswa mendapatkan kemudahan untuk memperoleh referensi. Seorang mahasiswa yang hendak mencari referensi tinggal mengetik ”kata kunci” dan beberapa saat kemudian referensi-referensi yang diinginkan muncul dalam layar monitor.

Kemudahan-kemudahan dalam mengakses internet inipun tidak jarang disalahgunakan oleh mahasiswa. Tanpa berpikir panjang, mahasiswa melakukan *copy - paste* tanpa mencantumkan sumber *copy*-an dari referensi tersebut. Bahkan tidak jarang mahasiswa mengumpulkan tugas dari hasil *copy-paste* tanpa adanya peng-*edit*-an terlebih dahulu.

3. Malas

Sifat malas merupakan sifat manusiawi, tak terkecuali bagi mahasiswa. Mahasiswa menjadi jenuh dan malas karena selalu dihadapkan dengan tugas yang menumpuk. Tugas dari berbagai mata kuliah tidak jarang mempunyai *deadline* yang hampir bersamaan. Hal ini tentu saja membuat mahasiswa kurang optimal mengerjakan tugasnya. Tidak jarang pula mahasiswa mengerjakan tugas dengan jalan pintas. Berdalihkan keterbatasan waktu, mahasiswa melakukan *copy-paste* atau plagiarisme dari pekerjaan teman ataupun dari hasil *browsing* di internet.

4. Tidak percaya diri

Ketidakpercayaan diri juga merupakan salah satu faktor penyebab terjadinya plagiarisme. Mahasiswa seringkali merasa takut untuk mengeluarkan ide-idenya karena kurangnya rasa percaya diri. Mereka beranggapan bahwa ide-ide mereka tidak layak ataupun tidak dapat diterima oleh publik. Mereka menilai hasil kutipan atau plagiat adalah karya yang sempurna. Padahal hal itu belum tentu benar. Tanpa menuangkan ide-ide mereka yang orisinil dan memperlihatkannya pada publik, mereka tidak akan pernah tahu apakah ide mereka itu bagus atau tidak.

5. Hanya menginginkan nilai bagus

Apakah tujuan dari mahasiswa mengerjakan tugas yang diberikan oleh dosen? Kebanyakan dari mereka pasti menginginkan nilai yang bagus. Penulis sendiri juga pernah berasumsi seperti itu. Terkadang dosen memberikan tugas karya ilmiah dengan

memberi batas minimal referensi yang harus digunakan. Hal ini tentu saja menimbulkan asumsi pada diri mahasiswa yaitu semakin banyak referensi maka mereka akan mendapat nilai semakin bagus. Mahasiswapun berlomba-lomba untuk memperoleh referensi yang bagus untuk dijadikan isi dari karya mereka. Demi memperoleh nilai bagus, mahasiswa melakukan plagiarisme. Mereka hanya mengutamakan nilai bagus tanpa berpikir dampaknya.

6. Sanksi belum ditegakkan secara tegas

Dalam dunia pendidikan di Indonesia, upaya perlindungan terhadap hak paten dari suatu karya ilmiah masih sedikit. Penegakan hukum terhadap tindakan plagiat suatu karya ilmiah juga masih lemah. Kalaupun ada yang sering terkena hukuman adalah mahasiswa yang ketahuan melakukan plagiarisme. Sedangkan bagi orang yang menawarkan jasa penyusunan dan penjualan skripsi serta tesis masih sering lepas dari jerat hukum. Sehingga penjiplakan suatu karya ilmiah masih sangat mudah untuk dilakukan. Bahkan untuk mengetahui keaslian dari karya tulis tersebut sangat sulit dilakukan. Oleh sebab itu plagiarisme sulit diberantas.

Melihat dari faktor-faktor yang menyebabkan plagiarisme tetap berlangsung di kalangan mahasiswa, ada beberapa hal yang harus diperhatikan untuk mengurangi plagiarisme tersebut, yaitu:

1. Mempelajari tata cara penulisan karya ilmiah

Mahasiswa dapat memanfaatkan teknologi untuk mempelajari tata cara penulisan suatu karya ilmiah yang benar. Mahasiswa dapat mengakses situs VAIL. Situs ini dapat

memberikan suatu bimbingan tentang bagaimana cara menulis suatu karya ilmiah yang benar tanpa adanya plagiarisme. Situs yang bernama VAIL ini dapat dijadikan bahan pembimbing menulis secara orisinal. Setelah membaca apa yang ada dalam situs VAIL tersebut, penulis sadar bahwa plagiarisme merupakan tindakan yang tidak baik. Karena itulah penulis berusaha untuk menghilangkan kebiasaan buruk tersebut.

2. Tindakan yang tegas bagi para pelaku plagiat

Tindakan plagiarisme yang menjamur di kalangan mahasiswa hendaknya menjadi cambuk bagi masyarakat pendidikan. Penegak hukum hendaknya memberikan sanksi yang tegas bagi para pelaku plagiarisme. Hal ini dilakukan agar para pelaku tersebut jera dengan perbuatan mereka. Penegak hukum dan masyarakat hendaknya menyadari bahwa plagiarisme hanya akan membawa keterpurukan bangsa.

3. Menanamkan moral pada masing - masing pribadi

Penanaman moral mempunyai peran penting dalam pemberantasan plagiarisme. Apabila dalam diri mahasiswa telah tertanam moral yang baik maka kenginginan untuk melakukan plagiarisme akan sirna. Seorang mahasiswa yang bermoral baik pasti akan menghargai hasil karya orang lain. Penghargaan terhadap orisinalitas sebuah karya juga merupakan suatu wujud dari kejujuran mahasiswa tersebut.

4. Meluruskan tujuan mengerjakan tugas

Mahasiswa hendaknya mempunyai pandangan ke depan. Mereka hendaknya meluruskan tujuan mereka dalam mengerjakan tugas. Mereka harus meyakini bahwa apa yang mereka kerjakan akan bermanfaat pada suatu saat. Dalam mengerjakan tugas pun

mereka harus menikmati prosesnya, sebab nilai yang bagus bukanlah tujuan akhir dari sebuah tugas yang diberikan oleh dosen.

2.1.5 Cara Menghindari Plagiarisme

Untuk menghindari tindakan plagiarisme tersebut di atas dapat dilakukan dengan beberapa cara, di antaranya:

a. Apabila kita ingin mengutip sebuah tulisan yang kita peroleh dari hasil *browsing* di internet, kita harus mencantumkan alamat *website* secara lengkap, dan juga mencantumkan tanggal, bulan, tahun serta waktu ketika kita mem-*browsing* situs tersebut.
b. Apabila kita ingin mengutip perkataan, ide, atau pendapat orang lain di dalam sebuah buku kita harus menuliskan nama penulis, judul buku, dan tahun pembuatan pada kalimat yang kita kutip tersebut. Selain itu, kita juga harus mencantumkan buku tersebut di dalam daftar pustaka.
c. Gunakanlah informasi yang berupa fakta umum sehingga tidak akan ada pihak yang merasa dirugikan.

Menurut www.umuc.edu/distance/odell/cip/vail/html menyatakan bahwa ada tiga cara mengutip suatu sumber:

1. *Quoting*/mengutip

   yaitu menuliskan kalimat yang akan dikutip didalam tanda petik dan menuliskan sumber bahan kutipan itu di belakang kalimat yang telah kita kutip.

Contoh:

"*This unique observation provides only one example of the strong maternal inclinations of female gorillas. An amazing aspect of the incident was that Effie, whose back was turned toward her infant, was aware of Poppy's silent plight even before the human onlooker facing both animals realized that something was amiss*" (Fossey, 1983, p. 89).

2. *Paraphrasing*/parafrase

   Yaitu dengan menambahkan kalimat orang lain didalam kalimat kita sendiri dengan tetap mencantumkan sumbernya secara lengkap.

   Contoh:

   "*In Gorillas in the Mist, Fossey (1983) gives an example to demonstrate that gorillas are very protective mothers. As a tiger approached her offspring Effie, a mother gorilla, sensed the danger even though she was not facing Effie. The researcher observing them did not notice the danger, but the mother gorilla's vigilance allowed her to take steps to save the baby gorilla* (p. 89).

3. *Summarizing*/meringkas

   Yaitu meringkas pendapat orang lain dengan tetap mencantumkan sumbernya. Kita bisa tetap merujuk pada hal yang kita kehendaki dengan bahasa kita sendiri.

   Contoh:

   In Gorillas in the Mist, Fossey (1983) illustrates that female gorillas instinctively protect their offspring as much or more so than female humans (p. 89).

Dari uraian tentang cara mengutip sumber secara benar yang telah di jelaskan diatas, maka sebaiknya kita bisa lebih teliti untuk menghindari plagiarisme.

2.1.6 Contoh-contoh Praktik Plagiarisme

Praktik plagiarisme literatur dapat terjadi dalam berbagai bidang, seperti dalam literatur akademis dan juga dalam literatur fiksi.

Contoh plagiarisme dalam bidang akademis menurut wikipedia adalah:

- James A. Mackay, seorang ahli sejarah Skotlandia, dipaksa menarik kembali semua buku biografi Alexander Graham Bell yang ditulisnya pada 1998 karena ia menyalin dari sebuah buku dari tahun 1973. Ia juga dituduh memplagiat biografi Mary Queen of Scots, Andrew Carnegie, dan Sir William Wallace. Pada 1999 ia harus menarik biografi John Paul Jones tulisannya dengan alasan yang sama.
- Ahli sejarah Stephen Ambrose dikritik karena mengambil banyak kalimat dari karya penulis-penulis lain. Ia pertama dituduh pada 2002 oleh dua penulis karena menyalin sebagian tulisan mengenai pilot-pilot pesawat pembom dalam Perang Dunia II dari buku karya Thomas Childers *The Wings of Morning* dalam bukunya *The Wild Blue*. Setelah ia mengakui plagiarisme ini, *New York Times* menemukan kasus-kasus plagiarisme lain.
- Penulis Doris Kearns Goodwin mewawancarai penulis Lynne McTaggart dalam bukunya dari tahun 1987, *The Fitzgeralds and the Kennedys*, dan ia menggunakan beberapa kalimat dari buku McTaggart mengenai Kathleen

Kennedy. Pada 2002, ketika kemiripan ini ditemukan, Goodwin mengatakan bahwa ia mengira bahwa rujukan tidak perlu kutipan, dan bahwa ia telah memberikan catatan kaki. Banyak orang meragukannya, dan ia dipaksa mengundurkan diri dari *Pulitzer Prize board*.

- Seorang ahli matematika dan komputer Dănuț Marcu mengaku telah menerbitkan lebih dari 378 tulisan dalam berbagai terbitan ilmiah. Sejumlah tulisannya ditemukan sebagai tiruan dari tulisan orang lain.
- Sebuah komite penyelidikan *University of Colorado* menemukan bahwa seorang profesor etnis bernama Ward Churchill bersalah melakukan sejumlah plagiarisme, penjiplakan, dan pemalsuan. Kanselir universitas tersebut mengusulkan Churchill dipecat dari *Board of Regents*.
- Mantan presiden AS Jimmy Carter dituduh oleh seorang mantan diplomat Timur Tengah Dennis Ross telah menerbitkan peta-peta Ross dalam buku Carter *Palestine: Peace, Not Apartheid* tanpa ijin atau memberi sumber.

Berikut ini adalah contoh-contoh plagiarisme dalam literature fiksi menurut wikipedia:

- Helen Keller dituduh pada 1892 menjiplak cerita pendek *The Frost King* dari karya Margaret T. Canby *The Frost Fairies*. Ia diadili di depan *Perkins Institute for the Blind*, dan dibebaskan dengan selisih satu suara. Ia menjadi paranoid akan plagiarisme sejak itu dan khawatir bahwa ia telah membaca *The Frost Fairies* namun kemudian melupakannya.
- Alex Haley dituntut oleh Harold Courlander karena sebagian novelnya *Roots* dituduh meniru novel Courlander *The African*.

- Dan Brown, penulis *The Da Vinci Code*, telah dituduh dan dituntut karena melakukan plagiarisme dua kali.
- Novel pertama Kaavya Viswanathan *How Opal Mehta Got Kissed, Got Wild and Got a Life*, dilaporkan mengandung jiplakan dari setidaknya 5 novel lain. Semua bukunya ditarik dari peredaran, kontraknya dengan *Little, Brown, and Co*. ditarik, dan sebuah kontrak film dengan *Dreamworks SKG* dibatalkan.

2.2 Metode *Edit Distance*

Metode *edit distance* adalah sebuah metode yang dibuat untuk digunakan dalam hal-hal yang berkaitan dengan perbandingan *string* atau pencocokan *string* (*string matching*). Penggunaan metode ini cukup luas, tidak hanya untuk mendeteksi plagiarisme, tetapi juga untuk hal-hal lainnya yang berkaitan dengan pencocokan *string*.

Dalam penulisan skripsi ini, metode *edit distance* digunakan untuk mendapat persentase kemiripan antar dua buah kode program (*source code*). Dari nilai persentase tersebut dapat diketahui besarnya tingkat kemiripan antar kedua kode program. Apakah masih dalam batas toleransi (80%) atau tidak. Jika ternyata persentasenya lebih besar dari 80%, dapat diambil kesimpulan bahwa ada indikasi telah terjadi sebuah praktik plagiarisme. Namun, hal di atas tidaklah mutlak karena pengguna dapat menentukan sendiri nilai batas toleransi.

2.2.1 Sejarah Metode *Edit Distance*

Metode *edit distance* bisa juga disebut *levensthein distance*. Metode ini ditemukan oleh Vladimir Levenshtein pada tahun 1965.

2.2.2 Penggunaan Metode *Edit Distance*

Metode *edit distance* dapat digunakan untuk bermacam-macam hal yang berhubungan dengan pencocokan *string* (*string matching*), yaitu:

- Revisi berkas (*file revision*)
- *Remote screen update problem*
- *Spelling correction*
- *Plagiarism detection*
- *Molecular biology*

  Metode ini dapat digunakan untuk mencocokan DNA atau asam amino yang terdapat dalam protein
- *Speech recognition*

2.2.3 Cara Kerja Metode *Edit Distance*

Pertama-tama dua buah kode program (*source code*) akan dijadikan menjadi dua buah *string* panjang, yang nantinya akan membentuk sebuah matriks dua dimensi. Panjang *string* yang lebih besar akan diletakkan sebagai kolom pada matriks dan panjang *string* yang lebih kecil akan diletakkan sebagai baris pada matriks.

Langkah-langkah metode *edit distance* adalah sebagai berikut:

a. Set n sebagai panjang *string1*

   Set m sebagai panjang *string2*

b. Jika n = 0, maka keluar dari program

   Jika m = 0, maka keluar dari program

c. Bentuk matriks dengan jumlah baris m dan jumlah kolom n

d. Inisialisasi baris pertama 0..n

   Inisialisasi kolom pertama 0..m

e. Cek setiap karakter pada *string1* (dengan i dari 1 sampai n)

   Cek setiap karakter pada *string2* (dengan j dari 1 sampai m)

f. Jika *string1*[i] = *string2*[j] maka *cost* = 0

g. Jika *string1*[i] $\neq$ *string2*[j] maka *cost* = 1

h. Elemen lainnya (matriks[i,j]) akan diisi dengan nilai minimum dari:

   1. sel di atasnya ditambah 1: matriks[i-1,j] + 1

   2. sel di sebelah kirinya ditambah 1: matriks[i,j-1] + 1

   3. sel pada diagonal atas dan kiri ditambah *cost*: matriks[i-1,j-1] + cost

i. Setelah semua sel terisi, maka perbedaan antara 2 *string* dapat dilihat pada matriks[n,m]

Berikut ini contoh dua buah *string* yang dibandingkan:

KORUPSI KOLUSI

Matriks yang terbentuk adalah sebagai berikut:

		K	O	L	U	S	I
	0	1	2	3	4	5	6
K	1						
O	2						
R	3						
U	4						
P	5						
S	6						
I	7						

Setelah dilakukan langkah-langkah metode di atas, matriks akan terisi menjadi sebagai berikut:

		K	O	L	U	S	I
	0	1	2	3	4	5	6
K	1	0					
O	2	1					
R	3	2					
U	4	3					
P	5	4					
S	6	5					
I	7	6					

		K	O	L	U	S	I
	0	1	2	3	4	5	6
K	1	0	1	2	3	4	5
O	2	1					
R	3	2					
U	4	3					
P	5	4					
S	6	5					
I	7	6					

		K	O	L	U	S	I
	0	1	2	3	4	5	6
K	1	0	1	2	3	4	5
O	2	1	0				
R	3	2	1				
U	4	3	2				
P	5	4	3				
S	6	5	4				
I	7	6	5				

		K	O	L	U	S	I
	0	1	2	3	4	5	6
K	1	0	1	2	3	4	5
O	2	1	0	1	2	3	4
R	3	2	1				
U	4	3	2				
P	5	4	3				
S	6	5	4				
I	7	6	5				

		K	O	L	U	S	I
	0	1	2	3	4	5	6
K	1	0	1	2	3	4	5
O	2	1	0	1	2	3	4
R	3	2	1	1			
U	4	3	2				
P	5	4	3				
S	6	5	4				
I	7	6	5				

		K	O	L	U	S	I
	0	1	2	3	4	5	6
K	1	0	1	2	3	4	5
O	2	1	0	1	2	3	4
R	3	2	1	1	2	3	4
U	4	3	2	2			
P	5	4	3	3			
S	6	5	4	4			
I	7	6	5	5			

		K	O	L	U	S	I
	0	1	2	3	4	5	6
K	1	0	1	2	3	4	5
O	2	1	0	1	2	3	4
R	3	2	1	1	2	3	4
U	4	3	2	2	1		
P	5	4	3	3			
S	6	5	4	4			
I	7	6	5	5			

K O L U S I

	0	1	2	3	4	5	6
K	1	0	1	2	3	4	5
O	2	1	0	1	2	3	4
R	3	2	1	1	2	3	4
U	4	3	2	2	1	2	
P	5	4	3	3	2	2	
S	6	5	4	4			
I	7	6	5	5			

		K	O	L	U	S	I
	0	1	2	3	4	5	6
K	1	0	1	2	3	4	5
O	2	1	0	1	2	3	4
R	3	2	1	1	2	3	4
U	4	3	2	2	1	2	
P	5	4	3	3	2	2	
S	6	5	4	4	3		
I	7	6	5	5	4		

		K	O	L	U	S	I
	0	1	2	3	4	5	6
K	1	0	1	2	3	4	5
O	2	1	0	1	2	3	4
R	3	2	1	1	2	3	4
U	4	3	2	2	1	2	3
P	5	4	3	3	2	2	
S	6	5	4	4	3		
I	7	6	5	5	4		

		K	O	L	U	S	I
	0	1	2	3	4	5	6
K	1	0	1	2	3	4	5
O	2	1	0	1	2	3	4
R	3	2	1	1	2	3	4
U	4	3	2	2	1	2	3
P	5	4	3	3	2	2	
S	6	5	4	4	3	2	
I	7	6	5	5	4	3	

K	O	L	U	S	I

	0	1	2	3	4	5	6
K	1	0	1	2	3	4	5
O	2	1	0	1	2	3	4
R	3	2	1	1	2	3	4
U	4	3	2	2	1	2	3
P	5	4	3	3	2	2	3
S	6	5	4	4	3	2	
I	7	6	5	5	4	3	

		K	O	L	U	S	I
	0	1	2	3	4	5	6
K	1	0	1	2	3	4	5
O	2	1	0	1	2	3	4
R	3	2	1	1	2	3	4
U	4	3	2	2	1	2	3
P	5	4	3	3	2	2	3
S	6	5	4	4	3	2	3
I	7	6	5	5	4	3	2

Dari matriks di atas terlihat bahwa jumlah perbedaan antara kedua *string* sama dengan nilai elemen matriks [7,6] yaitu bernilai 2 (dua).

Edit distance dari dua buah *string* yaitu *string1* dan *string2* didefinisikan sebagai jumlah minimum mutasi-mutasi titik yang dibutuhkan untuk mengubah *string1* menjadi *string2*. Sebuah mutasi titik dapat berupa pengubahan karakter, penambahan karakter, dan penghapusan karakter.

Fungsi penghitungan *edit distance* dari *string1* menjadi *string2* – e*ditDistance(string1, string2)* – dapat dirumuskan dalam persamaan rekursif sebagai berikut.

Perbedaan e*ditDistance(string1, string2)* dari dua buah teks tersebut dapat didefinisikan secara rekursif sebagai

$$d(\varepsilon,\varepsilon) = 0$$

$$d(s,\varepsilon) = d(\varepsilon,s) = |s|$$

$$d(s1+ch1, s2+ch2) = \min\begin{Bmatrix} d(s1,s2) + (ch1 == ch2\,?\,0:1) \\ d(s1+ch1, s2) + 1 \\ d(s1, s2+ch2) + 1 \end{Bmatrix}$$

Di mana,

ε = *empty string*

s1 = *string1*

s2 = *string2*

d = metode *edit distance*

Dari persamaan rekursif tersebut kita bisa mendapatkan sebuah fungsi rekursif untuk menghitung nilai *edit distance* yang memiliki kompleksitas $O(3^n)$, dengan $n = \max(|string1|, |string2|)$.

Persentase perbedaan antar dua buah *source code* dapat dihitung dengan rumus berikut:

$$P = \frac{D}{T}$$

Di mana,

P = persentase perbedaan

D = hasil keluaran metode *edit distance*

T = jumlah karakter terpanjang antara 2 masukan

Persentase kemiripan antar dua buah *source code* dapat dihitung dengan rumus berikut:

$$similarity = \left(1 - \frac{editDis\tan ce(string1, string2)}{\max(|string1|,|string2|)}\right) x100\%$$

Pada contoh di atas didapat jumlah perbedaan antara kedua *string* adalah 2. Dengan menggunakan rumus di atas, akan didapat persentase perbedaan:

$$P = \frac{D}{T} = \frac{2}{7} = 0.285714285 \approx 28.57\%$$

Sementara, persentase kemiripan antar dua buah *source code* adalah:

$$similarity = \left(1 - \frac{2}{7}\right) x100\% = 0.71428571 \approx 71.43\%$$

Dari nilai persentase dapat diketahui besarnya tingkat kemiripan antara kedua *source code*, apakah masih dalam batas toleransi (80%). Jika ternyata persentasenya lebih besar dari 80%, dapat diambil kesimpulan bahwa telah terjadi sebuah praktik plagiarisme.

Karena presentase yang didapat pada contoh di atas yaitu 71.43% lebih kecil dari 80% (batas toleransi), maka dapat disimpulkan bahwa tidak terjadi praktik plagiarisme. Perlu ditekankan, batas toleransi 80% tidaklah mutlak karena pengguna dapat menentukan sendiri nilai batas toleransi.

2.3 Dasar Perancangan Perangkat Lunak

Menurut Pressman (1997, p6), perangkat lunak adalah (1) instruksi (program komputer) yang ketika dieksekusi akan memberikan fungsi dan performa seperti yang diinginkan (2) struktur data yang memungkinkan program memanipulasi informasi secara proporsional, dan (3) dokumen yang menggambarkan operasi dan penggunaan program.

Menurut Sommerville (2001, p6), perancangan perangkat lunak adalah disiplin perancangan yang berhubungan dengan semua aspek dari produksi perangkat lunak dari tahap awal spesifikasi sistem sampai dengan pemeliharaan setelah sistem dalam tahap berjalan.

2.3.1 Daur Hidup Perangkat Lunak

Salah satu model perancangan perangkat lunak adalah dengan menggunakan model air terjun (*waterfall model*). Menurut Sommerville (2001, p45), tahap-tahap

utama dalam model air terjun yang menggambarkan aktivitas dasar pengembangan perangkat lunak adalah sebagai berikut:

- Analisis dan penentuan kebutuhan

  Tugas, kendala dan tujuan sistem ditentukan melalui konsultasi dengan pemakai sistem. Kemudian ditentukan cara yang dapat dipahami, baik oleh *user* maupun pengembang.

- Desain sistem dan perangkat lunak.

  Proses desain sistem terbagi dalam kebutuhan perangkat keras dan perangkat lunak. hal ini menentukan arsitektur perangkat lunak secara keseluruhan. Desain perangkat lunak mewakili fungsi sistem perangkat lunak dalam suatu bentuk yang dapat ditransformasikan ke dalam satu atau lebih program yang dapat dieksekusi.

- Implementasi dan pengujian unit.

  Dalam tahap ini, desain perangkat lunak direalisasikan dalam suatu himpunan program atau unit-unit program. Pengujian unit mencakup kegiatan verifikasi terhadap suatu unit sehingga memenuhi syarat spesifikasinya.

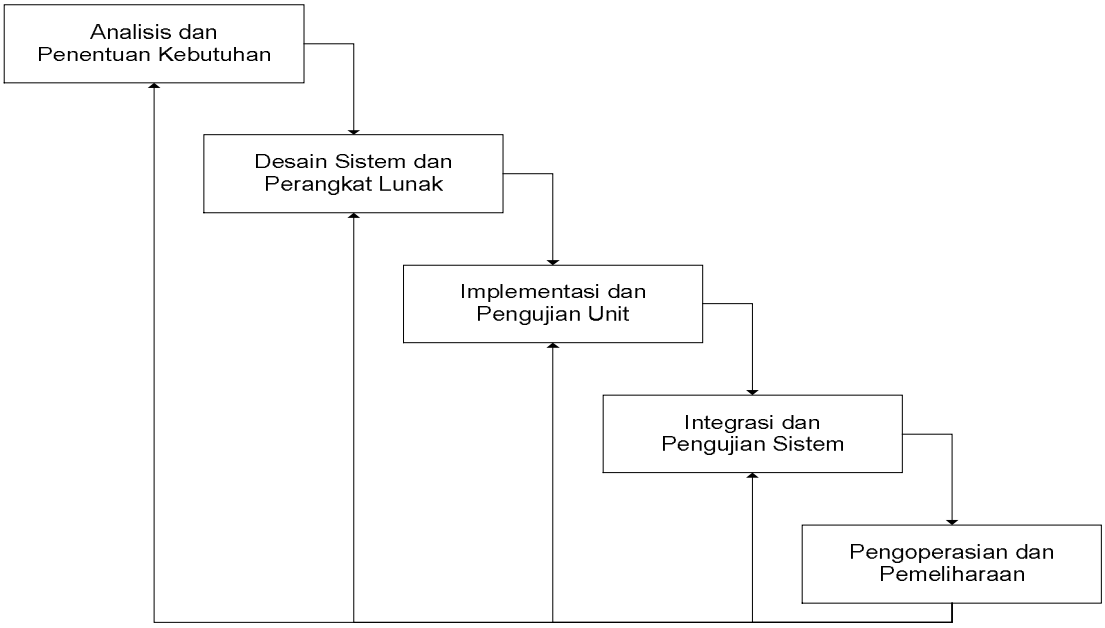
- Integrasi dan pengujian sistem.

  Unit program secara individual diintegrasikan dan diuji sebagai satu sistem yang lengkap untuk memastikan bahwa kebutuhan perangkat lunak telah terpenuhi. Setelah pengujian, sistem perangkat lunak disampaikan kepada *user*.

- Pengoperasian dan pemeliharaan.

  Secara normal, walaupun tidak selalu diperlukan, tahap ini merupakan bagian siklus hidup yang terpanjang. Sistem telah terpasang dan sedang dalam penggunaan. Pemeliharaan mencakup perbaikan kesalahan yang tidak ditemukan

dalam tahap-tahap sebelumnya, meningkatkan implementasi unit-unit sistem dan mempertinggi pelayanan sistem yang disebabkan oleh ditemukannya kebutuhan baru.



Gambar 2.1 Daur Hidup Perangkat Lunak

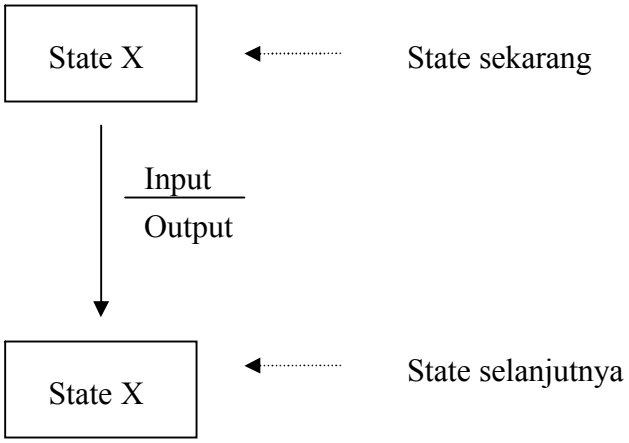
Sumber: Sommerville (2001).

2.4 Alat Bantu Perancangan

Alat Bantu perancangan yang digunakan dalam penulisan skripsi ini adalah *state transition diagram* dan *pseudocode.*

2.4.1 *State Transition Diagram*

State Transition Diagram adalah salah satu cara menggambarkan jalannya proses. Di dalamnya dapat dilihat *input* / kondisi, *state* proses, *output* / aksi yang terjadi dan perubahan *state*. Komponen dasar *state transition diagram* dapat dilihat pada gambar 2.3.



Gambar 2.2 Komponen Dasar *State Transition Diagram*

Sumber: Kowal (1988).

State, menunjukkan satu atau lebih kegiatan atau keadaan atau atribut yang menjelaskan bagian tertentu dari proses.

Anak panah berarah, menunjukkan perubahan *state* yang disebabkan oleh *input* tertentu (*state* X ke *state* Y).

Input / kondisi merupakan suatu kejadian pada lingkungan eksternal yang dapat dideteksi oleh sistem, misal sinyal, interupsi atau data. Hal ini menyebabkan perubahan dari satu *state* ke *state* lainnya atau dari satu aktivitas ke aktivitas lainnya.

Output / aksi merupakan hal yang dilakukan oleh sistem jika terjadi perubahan *state* atau merupakan reaksi terhadap kondisi. Aksi dapat menghasilkan *output*, tampilan pesan pada layar, kalkulasi atau kegiatan lainnya.

2.4.2 *Pseudocode*

Pseudocode adalah suatu bahasa umum yang menggunakan kosa kata dari satu bahasa (misal: bahasa Inggris) dan perintah (*syntax*) dari bahasa yang lain (misal: bahasa pemrograman terstruktur). (Pressman, 1997, p411)

Pseudocode adalah suatu bahasa pemrograman yang informal dan sangat fleksibel, yang tidak dimaksudkan untuk dieksekusi pada mesin, tetapi hanya digunakan untuk mengorganisir cara berpikir pemrogram sebelum melakukan *coding*. (Pege-Jones, 1980, p11)

Pseudocode dapat menjadi alternatif dalam perancangan perangkat lunak di samping alat bantu berupa diagram. Tidak ada standarisasi dalam hal penulisan *pseudocode*. Pemrogram dapat menulisnya dalam bahasa apa saja yang mereka sukai dan dipadukan dengan bahasa pemrograman tertentu. Pemrogram juga bebas menggunakan teknik dan aturannya sendiri.