BAB II
LANDASAN TEORI

2.1 Tinjauan Pustaka

Sebagai bahan pembanding dalam pengembangan aplikasi yang berjudul Analisis dan Perancangan Sistem Pengoreksian Ejaan Bahasa Indonesia dengan Menggunakan Metode *Levenshtein Distance*, berikut ini dipaparkan hasil penelitian yang telah dilakukan sebelumnya.

Penelitian yang dilakukan oleh Muliantara (2015), membahas tentang implementasi *Levenshtein Distance* untuk menampilkan saran perbaikan kesalahan pengetikan Bahasa Indonesia, bahwa Bahasa merupakan alat komunikasi lingual manusia baik secara lisan maupun tulisan. Peran Bahasa Indonesia yang baik dan benar sebagai bahasa resmi nasional memiliki arti yang sangat penting. Apalagi dalam penyusunan karya ilmiah. Hal ini disebabkan adanya data dan informasi yang ada didalamnya digunakan sebagai acuan untuk penelitian atau pengkajian selanjutnya bagi ilmuan lainnya, data acuan kosakata Bahasa Indonesia mengacu pada KBBI. Hal itu tentu saja menimbulkan beberapa masalah seperti kesalahan pengetikan. Kontrol pegecekan bahasa yang baik dan benar jika dilakukan secara manual pastinya akan menghabiskan banyak waktu, apalagi dijaman serba modern seperti saat ini.

Penelitian mengenai Santoso (2010), membahas tentang *crawling* website *e-government* pemerintah daerah Jawa Tengah, bahwa *crawling* merupakan sebuah metode yang digunakan oleh mesin pencari yang mengambil link halaman yang terdapat dalam web dari satu link ke link yang lain secara keseluruhan atau disebut *crawler* tidak terfokus, tentunya hasil yang didapat tidak

seluruhnya relevan dengan yang diharapkan. *Crawling* terfokus dapat digunakan untuk memenuhi kebutuhan organisasi atau individual dalam membuat topik - topik spesifik, misalnya untuk memelihara konten portal web, untuk mengumpulkan dokumen secara lokal atau memenuhi kebutuhan informasi yang kompleks. Dari kekurangan *crawler* tersebut peneliti mengajukan pembuatan *crawling* terfokus berdasarkan ontologi. Sehingga tujuan khusus yang ingin dicapai pada penelitian ini adalah terbentuknya model agregasi data atau informasi dengan teknik *crawling* terfokus berdasarkan ontologi yang dapat diimplementasikan dengan baik dan terukur pada situs-situs *e-Government* untuk pengelompokkan.

Penelitian yang dilakukan Lhoussain (2015) dinyatakan bahwa pengoreksian ejaan yang salah sangatlah penting. Dengan meggunakan metode *Levenshtein Distance* yang menggunakan operasi penambahan, permutasi dan penghapusan karakter untuk membandingkan dua kata yang berbeda dimana terdapat kata yang salah dan kata yang benar, perbandingan dua kata tersebut terjadi ketika melakukan perhitungan dari operasi metode *Levenshtein Distance* yang akan meghasilkan kata terdekat dari kata yang benar.

Penelitian Adiwidya (2010), mengenai algoritma *Levenshtein* dalam pendekatan *Approximate String Match*, menjelaskan bahwa algoritma *Levenshtein* merupakan algoritma yang digunakan untuk mencari jumlah operasi *string* yang paling sedikit untuk mentransformasikan suatu *string* menjadi *string* yang lain, dimana algoritma ini digunakan dalam pencarian *string* dengan pendekatan perkiraan. Ada tiga buah operasi *string* yang digunakan yaitu operasi penghapusan, penyisipan, dan penggantian.

Berdasarkan hasil penelitian diatas, dapat dijadikan pedoman atau acuan dalam pembuatan aplikasi Pengoreksian Ejaan Bahasa Indonesia dengan metode *Levenshtein Distance* yang berbasis web. Perancangan aplikasi menerapkan metode – metode yang telah dilakukan oleh peneliti sebelumnya, namun penelitian sebelumnya menggunakan aplikasi berbasis *desktop*. Sedangkan aplikasi yang akan dirancang berbasis *web*.

2.2 Landasan Teori

2.2.1 Pengertian *Levenshtein Distance*

Algoritma *Levenshtein Distance* ditemukan oleh Vladimir Levenshtein, seorang ilmuan asal Rusia pada tahun 1965, algoritma ini sering juga disebut dengan *Edit Distance*. Algoritma *Levenshtein Distance* bekerja dengan menghitung jumlah minimum pentranformasian suatu string menjadi string lain yang meliputi penghapusan, penyisipan, dan penggantian (Mulyanto,2010). Frederick J. Damerau telah mengembangkan algoritma *Levenshtein* bersama penambahan operasi untuk melakukan pencarian jarak antar huruf (Setiadi, 2013). Menurut Rabat, Prosedur kalkulasi *Levenshtein Distance* terjadi antara dua kata, dimana persamaan perhitungan adalah

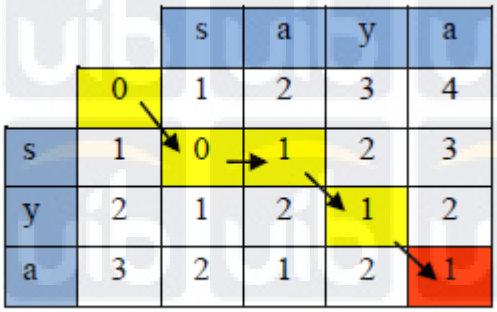
Gambar 2.1 Ilustrasi rumus *Levenshtein Distance*

$$D(i,j) = Minimum\{D(i-1,j)+1; D(i,j-1)+1; D(i-1,j-1)+\cos t\}$$

$$\text{With } \cos t = \begin{cases} 0 & if\ x_{i-1} = y_{j-1} \\ 1 & otherwise \end{cases}$$

Santiyasa (2015), telah memberikan contoh perhitungan *Levenshtein Distance* dengan menggunakan tabel matriks, berikut perhitungan *Levenshtein Distance* :

Gambar 2.2 Matriks perhitungan *Levenshtein Distance*



Contoh dari perhitungan *Levenshtein* menggunakan 2 string yang berbeda kemudian dihitung *Levenshtein distance*-nya pada gambar 2.2. Dapat dilihat hasil perhitungan *Levenshtein distance* antara 2 string "sya" dan "saya" adalah 1. Pengecekan dimulai dari iterasi awal dari kedua string kemudian dilakukan operasi penambahan, penyisipan dan penghapusan. Nilai *Levenshtein distance*-nya yaitu pada ujung kanan bawah matriks. Hanya ada satu proses penyisipan yang dilakukan yaitu penyisipan karakter "a" pada string "sya" sehingga menjadi "saya". Pada kasus pengecekan ejaan proses perhitungan ini dilakukan sejumlah kata yang ada pada basis data. Tentu saja untuk saran yang terbaik dibutuhkan daftar kata berhasa Indonesia yang lengkap. Sehingga kata yang disarankan bisa mendekati yang diharapkan oleh pengguna. Contoh perhitungan *Levenshtein Distance* dengan tabel matriks antara kata "sya" dengan "saya", dengan keterangan inisialisasi variable "i" sebagai penunjuk kata "sya" dan inisialisasi variabel "j" sebagai penunjuk kata "saya" sebagai berikut ini :

Tabel 2.1 Tahel matiks inisialisasi “i” dan “j”

		s	a	y	a
	0	1	2	3	4
s	1	[1 , 1]	[1 , 2]	[1 , 3]	[1 , 4]
y	2	[2 , 1]	[2 , 2]	[2 , 3]	[2 , 4]
a	3	[3 , 1]	[3 , 2]	[3 , 3]	[3 , 4]

Tabel 2.2 Tabel baris dan kolom pertama dan kedua pada tabel 2.1

		s	a	y	a
	0	1	2	3	4
s	1				
y	2				
a	3				

Pada tabel 2.2, adalah baris dan kolom pertama dan kedua dari tabel 2.1, hasil baris kedua didapatkan dari baris pertama (matriks [0,0] hingga [0,4]) di isi dengan nilai 0 hingga “j” (panjang kata pada baris pertama) dan hasil kolom kedua didapatkan dari kolom pertama (matriks [0,0] hingga [1,0]) di isi dengan nilai 0 hingga “i “(panjang kata pada kolom pertama), seperti yang terdapat pada tabel 2.3.

Tabel 2.3 Rincian baris pertama dan kedua untuk tabel 2.2

		s	a	y	a
	[0 , 0]	[0 , 1]	[0 , 2]	[0 , 3]	[0 , 4]
s	[1 , 0]				
y	[2 , 0]				
a	[3 , 0]				

Dimana tampilan rincian pengisian keseluruhan pada tabel matriks seperti yg terdapat pada tabel 2.4

Tabel 2.4 Rincian pengisian tabel matriks

		s	a	y	a
	[0 , 0]	[0 , 1]	[0 , 2]	[0 , 3]	[0 , 4]
s	[1 , 0]	[1 , 1]	[1 , 2]	[1 , 3]	[1 , 4]
y	[2 , 0]	[2 , 1]	[2 , 2]	[2 , 3]	[2 , 4]
a	[3 , 0]	[3 , 1]	[3 , 2]	[3 , 3]	[3 , 4]

Tabel 2.5 Perhitungan *Levenshtein Distance* matriks [1,1] yang terdapat pada tabel 2.4

		s	a	y	a
	0	1	2	3	4
s	1	0	[1 , 2]	[1 , 3]	[1 , 4]
y	2	[2 , 1]	[2 , 2]	[2 , 3]	[2 , 4]
a	3	[3 , 1]	[3 , 2]	[3 , 3]	[3 , 4]

[1,1]=min (lev(1−1,1)+1,lev(1,1−1)+1,lev(1−1,1−1)+0)

[1,1]=min (lev(0,1)+1,lev(1,0)+1,lev(0,0)+0)

[1,1]=min (1+1,1+1,0+0)

[1,1]=min (2,2,0) [1,1]=0

Perhitungan diteruskan hingga tabel terisi seluruhnya.

Tabel 2.6 Hasil akhir perhitungan *Levenshtein Distance*

		s	a	y	a
	0	1	2	3	4
s	1	0	1	2	3
y	2	1	2	1	2
a	3	2	3	2	1

Nilai yang terdapat pada kolom terakhir dari baris terakhir adalah nilai levenshtein distance antar kedua kata. Pada contoh di atas, nilai *Levenshtein Distance* antara kata "sya" dan "saya" adalah 1, semakin kecil angka nilai semakin mendekati kata saran yang diinginkan oleh pengguna.

2.2.2 *Crawling*

Menurut Benisius (2012), *Crawling* adalah proses menyediakan konten atau keseluruhan isi halaman yang terdapat pada *website* yang ingin di *crawl*, sistem akan menerima data – data sebagai berikut :

1. URL : merupakan alamat url dari konten yang ter-*crawling.*
2. Header : status HTTP yang diterima dari konten yang ter-*crawling.*
3. Title : judul dari setiap konten ter-*crawling* yang diambil dari kata – kata yang diapit oleh tag <title> … </title>.
4. Content : isi dari konten yang ter-*crawling* diambil dari kata – kata yang diapit oleh tag yang kita inginkan.

Hasil Proses *crawling* pada akhirnya akan menghasilkan suatu bank kata yang perbendaharaan katanya yang disebut dengan proses *indexing*. Kata – kata inilah yang kemudian akan membandingkan dengan kata kunci yang diinputkan oleh pengguna untuk dapat menghasilkan saran kata kunci bagi pengguna dalam melakukan proses pencarian selanjutnya secara lebih akurat. Setelah melakukan *indexing* data hasil *crawling* hasilnya akan disimpan didatabase dengan format *JSON*. Menurut Destian (2015), JSON (Java Script Object Notation) adalah format pertukaran data yang bersifat ringan, disusun oleh Douglas Crockford. Fokus JSON adalah pada representasi data di website. JSON dirancang untuk

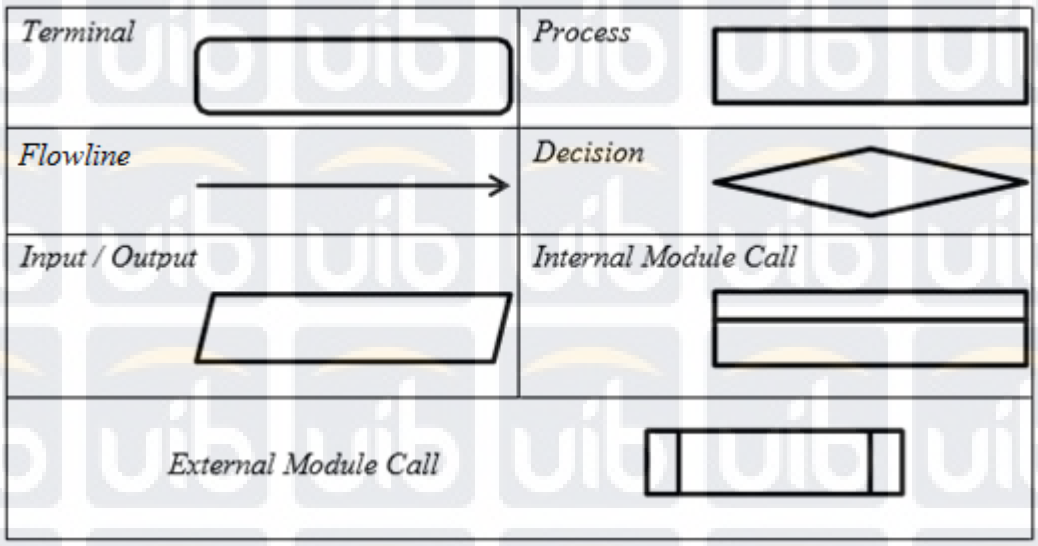
memudahkan pertukaran data pada situs dan merupakan perluasan dari fungsi-fungsi javascript.

Gambar 2.3 Ilustrasi proses crawling

2.2.3 *Flowchart*

Adelia (2011), *Flowchart* adalah penggambaran secara grafik dari langkah-langkah dan urut-urutan prosedur dari suatu program. *Flowchart* menolong *analyst* dan *programmer* untuk memecahkan masalah kedalam segmen- segmen yang lebih kecil dan menolong dalam menganalisis alternatif-alternatif lain dalam pengoperasian. *Flowchart* biasanya mempermudah penyelesaian suatu masalah khususnya masalah yang perlu dipelajari dan dievaluasi lebih lanjut. *Flowchart* adalah bentuk gambar/diagram yang mempunyai aliran satu atau dua arah secara sekuensial. *Flowchart* digunakan untuk merepresentasikan maupun mendesain program. Oleh karena itu *flowchart* harus bisa merepresentasikan komponen-komponen dalam bahasa pemrograman.

Tabel 2.7 Simbol – simbol *Flowchart*

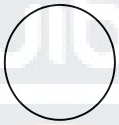


2.2.4 *Data Flow Diagram* (DFD)

Data flow diagram adalah suatu grafik yang menjelaskan sebuah sistem dengan menggunakan bentuk-bentuk dan simbol-simbol untuk menggambarkan aliran data dari proses-proses yang saling berhubungan. *Data flow diagram* ini adalah salah satu alat pembuatan model yang sering digunakan, khususnya bila fungsi-fungsi sistem merupakan bagian yang lebih penting dan kompleks dari pada data yang dimanipulasi oleh sistem.

Dengan kata lain, data flow diagram adalah alat pembuatan model yang memberikan penekanan hanya pada fungsi sistem. *Data flow diagram* ini merupakan alat perancangan sistem yang berorientasi pada alur data dengan konsep dekomposisi dapat digunakan untuk penggambaran analisa maupun rancangan sistem yang mudah dikomunikasikan oleh profesional sistem kepada pemakai pembuat program (Adelia, 2011).

Tabel 2.8 Simbol – simbol DFD

Simbol	Nama	Keterangan
	Entitas	Objek aktif yang mengendalikan aliran data dengan memproduksi serta menkomsumsi suatu data.
	Proses	Objek yang melakukan transformasi terhadap data.
	Aliran Data	Aliran data menghubungkan keluaran dari suatu objek atau proses yang terjadi pada suatu masukan.
	*Data Store*	Objek pasif dalam *DFD* yang menyimpan data untuk penggunaan lebih lanjut.

2.2.5 *Node JS*

Menurut Iqbal (2012), Node.js adalah sistem perangkat lunak yang didesain untuk pengembangan aplikasi web. Aplikasi ini ditulis dalam bahasa *JavaScript*, menggunakan basis *event* dan *asynchrounous I/O*. Tidak seperti kebanyakan bahasa *JavaScript* yang dijalankan pada peramban atau *browser*, *Node.js* dieksekusi sebagai aplikasi server.

Aplikasi ini terdiri dari *V8 JavaScript Engine* buatan *Google* dan beberapa modul bawaan yang terintegrasi. *Websocket-Node* yang merupakan implementasi *Websocket* pada *Node.js* dan *Express* yang merupakan kerangka kerja HTTP pada *Node.js*.

2.2.6 *Javascript*

Menurut Indra (2014), *JavaScript* adalah bahasa *scripting* kecil, ringan, berorientasi objek yang ditempelkan pada kode HTML dan di proses di sisi *client*. *JavaScript* digunakan dalam pembuatan website agar lebih interaktif dengan memberikan kemampuan tambahan terhadap HTML melalui eksekusi perintah di sisi *browser*. *JavaScript* dapat merespon perintah *user* dengan cepat dan menjadikan halaman *web* menjadi responsif.

JavaScript memiliki struktur sederhana, kodenya dapat disisipkan pada dokumen HTML atau berdiri sebagai satu kesatuan aplikasi.

2.2.7 Metodologi Pengembangan Sistem

Untuk mengembangkan suatu sistem informasi, kebanyakan perusahaan menggunakan suatu metodologi yang disebut metodologi pengembangan sistem. Yang dimaksud dengan metodologi ini adalah suatu proses standar yang diikuti oleh organisasi utuk melaksanakan seluruh langkah yang diperlukan untuk menganalis, meracang, mengimplementasikan, dan memelihara sistem informasi (Hoffer dkk, 1998).

Seperti yang berlaku pada kebanyakan proses, pengembangan sistem informasi juga memiliki daur hidup. Daur hidupnya disebut daur pengembangan

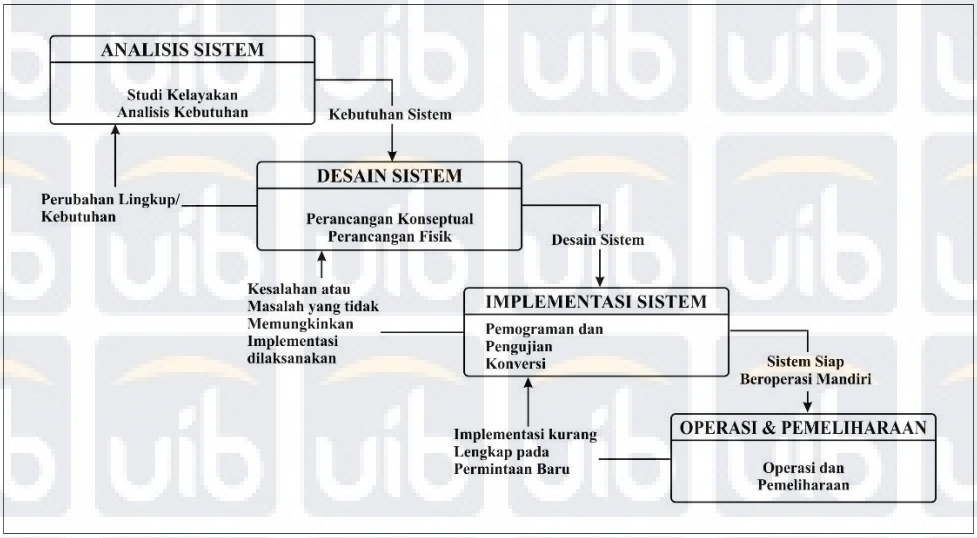
sistem informasi (O'Brien, 2001) atau secara lebih umum dinamakan SDLC *(System development life cycle)* atau daur hidup pengembangan sistem. SDLC merupakan metodologi klasik yang digunakan untuk pengembangan, memelihara, dan menggunakan sistem informasi. (Kadir, 2003: 398).

Jadi, SDLC *System development life cycle)* adalah suatu proses standar berbentuk siklus hidup yang diikuti oleh organisasi untuk pengembangan, memelihara, dan menggunakan sistem informasi. Dengan SDLC, proses pembangunan sistem dibagi menjadi beberapa langkah dan pada sistem yang besar, masing-masing langkah dikerjakan oleh tim yang berbeda. SDLC tidak hanya penting untuk proses produksi *software*, tetapi juga sangat penting untuk proses pemeliharaan perangkat lunak *(maintenance software)* itu sendiri.

Kadir (2003) secara umum menjabarkan siklus hidup pengembangan sistem SDLC *(system development life cycle)* dalam beberapa tahap berikut:

1. Analisis Sistem
2. Desain Sistem
3. Implementasi Sistem
4. Operasi dan Pemeliharaan

Gambar 2.4 Tahapan dalam Siklus Hidup Pengembangan Sistem



2.2.8 Analisis Sistem

Tahapan analisis sistem dimulai karena adanya permintaan terhadap sistem baru. Permintaan dapat datang dari seorang manajer diluar departemen sistem informasi atau dari pihak eksekutif yang melihat adanya masalah atau menemukan adanya peluang baru. Namun, ada kalanya inisiatif pengembangan sistem baru berasal dari bagian yang bertanggung jawab terhadap pengembangan sistem informasi, yang bermaksud mengembangkan sistem yang sudah ada atau mengatasi masalah-masalah yang belum ditangani.

Untuk melaksanakan hal tersebut, dibentuklah proyek baru yang ditangani dalam bentuk tim, yang melibatkan pemakai, analisis sistem, dan para spesialis sistem informasi yang lain. Tujuan utama anlisis sistem adalah untuk menentukan hal-hal detail tentang yang akan dikerjakan oleh sistem yang diusulkan.

a. Studi Kelayakan

Studi Kelayakan digunakan untuk menentukan kemungkinan keberhasilan solusi yang diusulkan. Tahap ini berguna untuk memastikan bahwa solusi

yang diusulkan tersebut benar-benar dapat dicapai dengan sumber daya dan dengan memperhatikan kendala yang terdapat pada perusahaan serta dampak terhadap lingkungan sekitarnya.

b. Analisis Kebutuhan

Analisis kebutuhan dilakukan untuk menghasilkan spesifikasi kebutuhan (disebut juga spesifikasi fungsional). Spesifikasi kebutuhan adalah spesifikasi yang rinci tentang hal-hal yang akan dilakukan sistem ketika diimplementasikan. Spesifikasi ini sekaligus dipakai untuk membuat kesepakatan antara pengembang sistem, pemakai yang kelak menggunakan sistem, manajemen, dan mitra kerja yang lain.

2.2.9 Desain Sistem

Desain sistem dapat dibagi menjadi dua tahapan, yakni perancangan konseptual dan perancangan fisik. Target akhir pada tahapan ini adalah menghasilkan rancangan yang memenuhi kebutuhan yang ditentukan selama tahapan analisis sistem. Hasil akhirnya berupa spesifikasi rancangan yang sangat rinci sehingga mudah diwujudkan pada saat pemograman.

Kedua tahapan dalam desain sistem untuk menghasilkan rancangan yang memenuhi kebutuhan antara lain:

a. Perancangan Konseptual

Perancangan konseptual sering kali disebut perancangan logis. Pada perancangan ini, kebutuhan pemakai dan pemecahan masalah yang teridentifikasi selama tahapan analisis sistem mulai dibuat untuk diimplementasikan. Hasil perancangan pada tahap konseptual ini akan digunakan pada tahap selanjutnya yaitu pada tahap perancangan fisik.

b. Perancangan Fisik

Pada perancangan fisik, rancangan yang bersifat konseptual diterjemahkan dalam bentuk fisik sehingga terbentuk spesifikasi yang lengkap tentang modul sistem dan antarmuka antarmodul, serta basis data secara fisik.

2.2.10 Implementasi Sistem

Pada tahap ini terdapat beberapa langkah yaitu: pemograman dan pengujian, instalasi perangkat keras dan perangkat lunak, serta pelatihan kepada pemakai. Sistem yang telah direncanakan, dianalisis dan didesain pada tahap-tahap sebelumnya akan dikonstruksi. Prosedur dalam tahap ini selalu sama, yakni menulis koding program, melakukan *testing*, dan mendokumentasikan hasil kerja. Pada akhir tahap ini, sistem sudah siap digunakan.

2.2.11 Pemeliharaan Sistem

Setelah masa sistem berjalan sepenuhnya menggantikan sistem lama, sistem memasuki pada tahapan operasi dan pemeliharaan. Selama sistem beroperasi, pemeliharaan sistem tetap diperlukan karena beberapa alasan. Pertama, mungkin sistem masih menyisakan masalah-masalah yang tidak terdeteksi selama masa pengujian sistem. Kedua, pemeliharaan diperlukan karena perubahan bisnis atau lingkungan. Dan ketiga, pemeliharaan juga bisa dipicu karena kinerja sistem menjadi menurun. (Kadir, 2003:400-415)