BAB 2
LANDASAN TEORI

2.1 Pengertian Sejarah dan Budaya

Sejarah merupakan peristiwa dan kisah – kisah yang terjadi pada masa lampau, sejarah sendiri berasal dari kata arab yaitu *syajarotun* yang artinya pohon yang kemudian artinya berkembang menjadi akar, keturunan, asal-usul dan silsilah. Bahkan menurut Ibnu Khaldun dalam bukunya yang berjudul *"Mukadimah"* mendifinisikan sejarah adalah "catatan tentang masyarakat umat manusia dan tentang perubahan-perubahan yang terjadi pada watak mayarakat itu" [1].

Budaya adalah suatu cara hidup yang berkembang dan dimiliki bersama oleh sebuah kelompok orang dan diwariskan dari generasi ke generasi. Budaya terbentuk dari banyak unsur yang rumit, termasuk sistem agama dan politik, adat istiadat, bahasa, perkakas, pakaian, bangunan, dan karya seni. Bahasa, sebagaimana juga budaya, merupakan bagian tak terpisahkan dari diri manusia sehingga banyak orang cenderung menganggapnya diwariskan secara genetis. Ketika seseorang berusaha berkomunikasi dengan orang-orang yang berbeda budaya dan menyesuaikan perbedaan-perbedaannya, membuktikan bahwa budaya itu dipelajari [3]. Dari definisi tersebut, dapat diperoleh pengertian mengenai kebudayaan adalah sesuatu yang akan memengaruhi tingkat pengetahuan dan meliputi sistem ide atau gagasan yang terdapat dalam pikiran manusia, sehingga dalam kehidupan sehari-hari, kebudayaan itu bersifat abstrak. Sedangkan perwujudan kebudayaan adalah benda-benda yang diciptakan oleh manusia

sebagai makhluk yang berbudaya, berupa perilaku dan benda-benda yang bersifat nyata, misalnya pola-pola perilaku, bahasa, peralatan hidup, organisasi sosial, religi, seni, dan lain-lain, yang kesemuanya ditujukan untuk membantu manusia dalam melangsungkan kehidupan bermasyarakat [3].

2.2 Permainan / game

Game berasal dari kata bahasa inggris yang berarti dasar permainan. Permainan dalam hal ini merujuk pada pengertian kelincahan intelektual (*Intellectual Playability Game*) yang juga bisa diartikan sebagai arena keputusan dan aksi pemainnya. Dalam *game*, ada target-target yang ingin dicapai pemainnya. Permainan adalah kegiatan yang kompleks yang didalamnya terdapat peraturan, *play* dan budaya. Sebuah permainan adalah sebuah sistem dimana pemain terlibat dalam konflik buatan, disini pemain berinteraksi dengan sistem dan konflik dalam permainan merupakan rekayasa atau buatan, dalam permainan terdapat peraturan yang bertujuan untuk membatasi perilaku pemain dan menentukan permainan.

Game bertujuan untuk menghibur, *game* banyak disukai oleh anak – anak hingga orang dewasa. *Game* sebenarnya penting untuk perkembangan otak, untuk meningkatkan konsentrasi dan melatih untuk memecahkan masalah dengan tepat dan cepat karena dalam *game* terdapat berbagai konflik atau masalah yang menuntut kita untuk menyelesaikannya dengan cepat dan tepat. Tetapi *game* juga bisa merugikan karena apabila kita sudah kecanduan *game* kita akan lupa waktu dan akan mengganggu kegiatan atau aktivitas yang sedang kita lakukan [4].

1. Pengertian Game Arcade

Sering disebut *ding - dong* di Indonesia, biasanya berada di daerah / tempat khusus dan memiliki *box* atau mesin yang memang khusus dirancang untuk jenis *video games* tertentu dan tidak jarang bahkan memiliki fitur yang dapat membuat pemainnya lebih menghayati permainan yang sedang di mainkannya, seperti pistol, kursi khusus, sensor gerakan, sensor injakkan dan stir mobil [4].

2. Pengertian Game Edukasi

Video Game jenis ini dibuat dengan tujuan spesifik sebagai alat pendidikan, entah untuk belajar mengenal warna untuk balita, mengenal hitung-hitungan, bahasa asing, sampai belajar sejarah. *Developer* yang membuatnya, harus memperhitungkan berbagai hal agar *game* ini benar-benar dapat mendidik, menambah pengetahuan dan meningkatkan keterampilan yang memainkannya. Target segmentasi pemain harus pula disesuaikan dengan tingkat kesulitan dan *design visual* ataupun animasinya [4].

2.3 Artifical Inteligent

AI atau *Artifical Inteligent* pertama kali di kemukakan pada tahun 1956 di konferensi Darthmouth. Sejak saat itu, AI terus di kembangkan sebab berbagai penelitian mengenai teori-teori dan prinsip-prisipnya juga terus berkembang.

Terdapat beberapa definisi AI, menurut Stuart Russel dan Peter Norvig mengelompokan definisi AI,kedalam empat kategori. Pertama *Thinking humanly* dan *Acting humanly* adalah dua definisi dalam arti yang sangat luas. Saat ini,

pemikiran manusia yang diluar rasio, yakni refleks dan inuitif(berhubungan dengan perasaan), belum dapat ditiruka sepenuhnya oleh komputer. Kedua *thinking rationally* dan *acting rationally* merupakan dasar pemikiran bahwa komputer juga bisa melakukan aksi secara rasional berdasarkan hasil penalaran tersebut[9].

2.3.1 **Generate And test**

Metode *Generate And Test*(GT) adalah metode yang paling sederhana dalam teknik pencarian heuristik. Jika pembangkitan sebuah solusi yang mungkin dikerjakan secara sistematis, maka prosedur ini menjamin akan menemukan solusinya, tetapi jika ruang masalahnya sangat luas, mungkin memerlukan waktu yang lama.

Algoritma GT menggunakan prosedur DFS karena sesuatu solusi harus dibangkitkan secara lengkap sebelum dilakukan test. Algoritma ini berbentuk sistematis pencarian sederhana yang mendalam dari suatu ruang permasalan. Terdapat 2 prosedur penting : pembangkit (membangkitkan sebuah solusi yang mungkin) dan Tes (menguji solusi yang di bangkitkan)[10]. Berikut algoritma *generate and test* :

1. Bangkitkan sebuah solusi yang mungkin (bisa berupa suatu keadaan/state).
2. Tes apakah solusi yang dibangkitkan tersebut adalah sebuah solusi yang bisa diterima sesuai dengan kriteria ang diberikan.
3. Jika solusi telah ditemukan, maka keluar. Jika belum kembali ke langkah 1

2.4 Objeck Oriented

Orientasi objek merupaka teknik dalam menyelesaikan masalah yang kerap muncul dalam pengembangan perangkat lunak, merupakan teknik kulminasi dalam menemukan cara yang efektif dalam membangun sistem dan menjadi metode yang paling banyak dipakai oleh para pengembang perangkat lunak saat ini. Orientasi objek merupakaan teknik pemodelan sistem riil yang berbasis objek. Inti dari konsep ini adalah objek yang merupakan model dari sistem nyata[7].

2.5 Action Script

ActionScript adalah bahasa pemrograman yang dibuat berdasarkan ECMAScript, yang digunakan dalam pengembangan situs web dan perangkat lunak menggunakan platform Adobe Flash Player. *ActionScript* juga dapat di gunakan untuk mengontrol objek, membuat navigasi, elemen interaktif, membuat movie Flash dan aplikasi Web yang interaktif. ActionScript memiliki sintaks yang hampir sama dengan JavaScript, yang telah hadir sejak Flash 4. Namun baru di versi 5 pemrograman dengan ActionScript terasa lebih kental [5].

ActionScript memiliki kesamaan dengan JavaScript, Contoh kemiripan yang signifikan misalnya pada even di Javascript jika akan memanggil fungsi atau kode skrip di atribut *onClick*, *onLoad*, *unUnload*, dsb di ActionScript pun sama jika terjadi sebuah even yang ditentukan, seperti tombol mouse ditekan/dilepas, kursor masuk ke daerah tertentu, dsb. Sebenarnya memang sintaks ActionScript dibuat mengikuti Javascript, dan pada dasarnya ActionScript masih lebih sederhana dari pada Javascript [6].

Meskipun memiliki kemiripan dengan Javascript, Action Script itu lebih mudah karena sifatnya yang interpreted dimana tidak perlu mengkompile begitu setelah mengcoding dan bisa langsung mencoba kode yang telah di tulis.

Action Script 3.0 pertamakali muncul pada adobe flash CS3, yang merupakan bahasa pemrograman yang di kembangkan dari Action Script 2.0. Dengan menggunakan sintak yang hampir mirip ternyata Action Script 3.0 ini sudah berorientasi objek, di mana semua ke unggulan pemrograman objek ada pada Action Script 3.0 ini [5].

2.6 Unified Modelling Language (UML)

Unintified Modelling Language (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah system [7].

Dari berbagai penjelasan rumit yang terdapat di dokumen dan buku-buku UML. Konsepsi dasar UML bisa kita rangkumkan dalam gambar 2.1 berikut [7].

Major Area	*View*	*Diagrams*	*Main Concepts*
structural	static view	class diagram	class, association, generalization, dependency, realization, interface
	use case view	use case diagram	use case, actor, association, extend, include, use case generalization
	implementation view	component diagram	component, interface, dependency, realization
	deployment view	deployment diagram	node, component, dependency, location
dynamic	state machine view	statechart diagram	state, event, transition, action
	activity view	activity diagram	state, activity, completion transition, fork, join
	interaction view	sequence diagram	interaction, object, message, activation
		collaboration diagram	collaboration, interaction, collaboration role, message
model management	model management view	class diagram	package, subsystem, model
extensibility	all	all	constraint, stereotype, tagged values

Gambar 1.1 Konsep UML [7].

Seperti juga tercantum pada gambar sebelumnya UML mendefinisikan diagram-diagram sebagai berikut:

1. *Use case diagram*

Use Case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Sebuah *Use Case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use Case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-create sebuah daftar belanja, dan sebagainya. Sebuah *Use Case* dapat meng-include fungsionalitas *Use Case*

lain sebagai bagian dari proses dalam dirinya. Sebuah *Use Case* dapat di-include oleh lebih dari satu *Use Case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang common. Sebuah *Use Case* juga dapat meng-extend *Use Case* lain dengan behaviour-nya sendiri. Sementara hubungan generalisasi antar *Use Case* menunjukkan bahwa *Use Case* yang satu merupakan spesialisasi dari yang lain.

2. *Class diagram*

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). *Class* diagram menggambarkan struktur dan deskripsi *Class*, package dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. *Class* memiliki tiga area pokok yaitu : Nama (dan *stereotype*), Atribut dan Metoda.

3. *Statechart diagram*

Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari stimuli yang diterima. Pada umumnya *statechart diagram* menggambarkan *class* tertentu (satu class dapat memiliki lebih dari satu *statechart diagram*).

4. *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*). Sebuah aktivitas dapat direalisasikan oleh satu *Use Case* atau lebih. *Activity diagram* menggambarkan proses yang berjalan, sementara *Use Case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas. *Activity diagram* dapat dibagi menjadi beberapa object *swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.

5. *Sequence diagram*

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, display, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah event untuk menghasilkan output tertentu. Diawali dari apa yang men-trigger aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan.

6. *Collaboration diagram*

Collaboration diagram juga menggambarkan interaksi antar objek seperti *sequence diagram*, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian message. Setiap message memiliki *sequence number*, di mana *message* dari level tertinggi memiliki nomor 1. *Messages* dari *level* yang sama memiliki *prefiks* yang sama.

7. *Component diagram*

Component diagram menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (*dependency*) di antaranya. Komponen piranti lunak adalah modul berisi *code*, baik berisi *source code* maupun *binary code*, baik *library* maupun *executable*, baik yang muncul pada *compile time*, *link time*, maupun *run time*. Umumnya komponen terbentuk dari beberapa *class* dan/atau *package*, tapi dapat juga dari komponen-komponen yang lebih kecil. Komponen dapat juga berupa *interface*, yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain.

2.7 Pengujian

Pengujian adalah proses menjalankan program dengan maksud untuk mencari kesalahan (error), Kasus uji yang baik adalah kasus yang memiliki peluang untuk mendapatkan kesalahan yang belum diketahui.

2.7.1 **Blackbox Testing**

Test case ini bertujuan untuk menunjukkan fungsi Perangkat Lunak tentang cara beroperasinya, apakah pemasukan data keluaran telah berjalan sebagaimana yang diharapkan dan apakah informasi yang disimpan secara eksternal selalu dijaga kemutakhirannya

2.7.2 **Whitebox Testing**

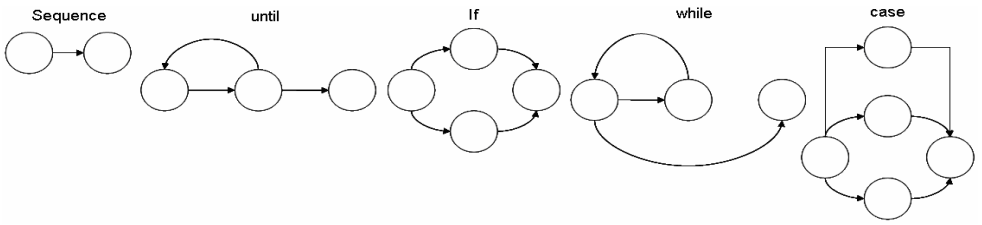
White Box Testing atau pengujian *glass box* adalah metode desain *test case* menggunakan struktur kontrol desain prosedural untuk mendapatkan *test case.* Dengan menggunakan metode *White Box* analisis sistem akan memperoleh Test Case sebagai berikut:

- Menjamin seluruh independent path di dalam modul yang dikerjakan sekurang-kurangnya sekali
- Mengerjakan seluruh keputusan logikal
- Mengerjakan seluruh loop yang sesuai dengan batasannya
- Mengerjakan seluruh struktur data internal yang menjamin validitas.

1. Uji Coba Basis Path

Merupakan teknik uji coba *white box* yang diusulkan Tom McCabe. Metode ini memungkinkan perancang test case mendapatkan ukuran kekompleksan logical dari perancangan prosedural dan menggunakan ukuran ini sebagai etunjuk

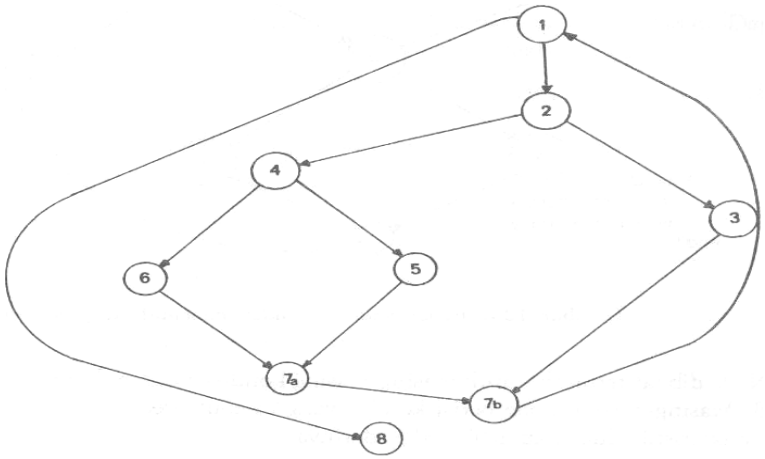
untuk mendefinisikan basis set dari jalur pengerjaan. Test case yang didapat digunakan untuk mengerjakan basis set yang menjamin pengerjaan setiap perintah minimal satu kali selama uji coba.



Gambar 2.2 Notasi Diagram Alir

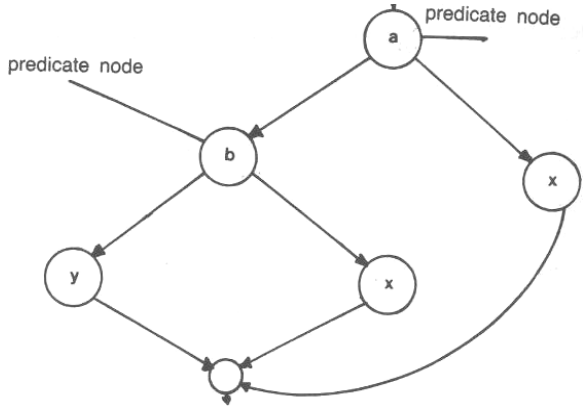
Lingkaran (node), menggambarkan satu/lebih perintah prosedural. Urutan proses dan keputusan dapat dipetakan dalam satu node. Tanda panah (edge), menggambarkan aliran kontrol. Setiap node harus mempunyai tujuan node. Region adalah daerah yang dibatasi oleh edge dan node. Termasuk daerah diluar grafik alir. Contoh menterjemahkan pseudo code ke grafik alir akan ditunjukan pada gambar 2.3

```
1 : do while record masih ada
baca record
2 : if record ke 1 = 0
3 : then proses record
simpan di buffer
naikan counter
4 : else if record ke 2 = 0
5 : then riser counter
6 : proses record
simpan pada file
7a: endif
endif
7b: end do
 8 : end
```



Gambar 2.3 Diagram Alir

Nomor pada pseudo code berhubungan dengan nomor node. Apabila ditemukan kondisi majemuk (compound condition) pada pseudo code pembuatan grafik alir menjadi rumit. Kondisi majemuk mungkin terjadi pada operator Boolean (AND, OR, NAND, NOR) yang dipakai pada perintah if. Berikut contoh gambar akan ditunjukan padagambar 2.4 berikut



Gambar 2.4 Logika gabungan

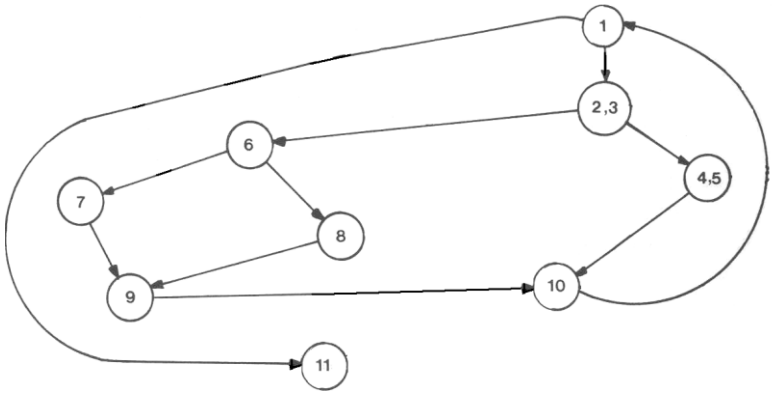
```
if A or B
then procedure x
else procedure y
endif
```

Node dibuat terpisah untuk masing-masing kondisi A dan B dari pernyataan IF A OR B. Masing-masing node berisi kondisi yang disebut pridicate node dan mempunyai karakteristik dua atau lebih edge darinya.

2. Cyclomatic Complexity

Cyclomatic complexity adalah metrik software yang menyediakan ukuran kuantitatif dari kekompleksan logikal program. Apabila digunakan dalam konteks metode uji coba basis path, nilai yang dihitung untuk cyclomatic complexity menentukan jumlah jalur independen dalam basis set suatu program dan memberi batas atas untuk jumlah uji coba yang harus dikerjakan untuk menjamin bahwa seluruh perintah sekurang-kurangnya telah dikerjakan sekali.

Jalur independent adalah jalur yang melintasi atau melalui program dimana sekurang-kurangnya terdapat proses perintah yang baru atau kondisi yang baru.



Gambar 2.5 Contoh *Flowgraph*

Path 1 = 1 - 11

Path 2 = 1 - 2 - 3 - 4 - 5 - 10 - 1 - 11

Path 3 = 1 - 2 - 3 - 6 - 8 - 9 ...: 10 - 1 - 11

Path 4 = 1 - 2 - 3 - 6 - 7 - 9 - 10 - 1 - 11

Path 1,2,3,4 yang telah didefinisikan diatas merupakan basis set untuk diagram alir.

Cyclomatic complexity digunakan untuk mencari jumlah path dalam satu flowgraph. Dapat dipergunakan rumusan sebagai berikut：

1. Jumlah region grafik alir sesuai dengan cyclomatic complexity.

2. Cyclomatix complexity V(G) untuk grafik alir dihitung dengan rumus:

V(G) = E - N + 2

Dimana:

E = jumlah edge pada grafik alir

N = jumlah node pada grafik alir

3. Cyclomatix complexity V(G) juga dapat dihitung dengan rumus:

V(G) = P + 1

Dimana P = jumlah predicate node pada grafik alir

Pada Gambar 2.5 dapat dihitung cyclomatic complexity:

1. Flowgraph mempunyai 4 region

2. V(G) = 11 edge - 9 node + 2 = 4

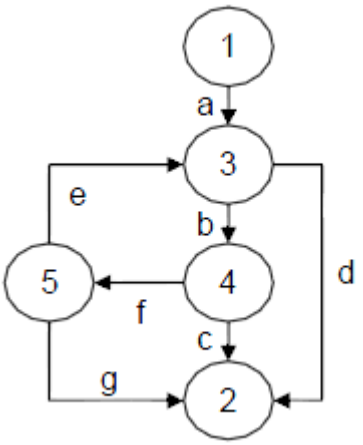
3. V(G) = 3 predicate node + 1 = 4

Jadi cyclomatic complexity untuk flowgraph adalah 4.

3. Graph Metrik

Graph metrik merupakan software yang dikembangkan untuk membantu uji coba basis path atau struktur data. Graph metrik adalah matrik empat persegi yang mempunyai ukuran yang sama dengan jumlah node pada *flowgraph*. Masing-masing baris dan kolom mempunyai hubungan dengan node yang telah ditentukan dan pemasukan data matrik berhubungan dengan hubungan (*edge*) antar *node*.

Berikut ini adalah contoh sederhana pemakaian graph metrik dapat digambarkan pada gambar 2.6.



	1	2	3	4	5
1			a		
2					
3				b	
4		c			f
5		g	e		

Gambar 2.6 Penggunaan Graph Metrik

Hubungan bobot menyediakan tambahan informasi tentang aliran kontrol. Secara simpel hubungan bobot dapat diberi nilai 1 jika ada hubungan antara node atau nilai 0 jika tidak ada hubungan. Dapat juga hubungan bobot diberi tanda dengan:

- Kemungkinan link (edge) dikerjakan
- Waktu yang digunakan untuk proses selama traversal dari link
- Memori yang diperlukan selama traversal link
- Sumber daya yang diperlukan selama traversal link

2.8 Tinjauan Perangkat Lunak

1. Adobe Flash

Adobe Flash (dahulu bernama *Macromedia Flash* dikarenakan *Macromedia* yang merupakan produsen pembuat *flash profesional* kini telah merjer dengan *adobe corp,* perubahan terjadi pada *macromedia flash* series 9 menjadi Adobe Flash CS3 pada April 16, 2007) merupakan tools yang dikembangkan untuk membuat berbagai aplikasi berbasis *internet*. Pada awalnya, *Flash* yang dilengkapi bahasa pemrograman *ActionScript* digunakan oleh *developer web* untuk mendesain web menjadi lebih interaktif dengan berbagai macam animasi. Namun, kemudian Flash banyak digunakan untuk membuat aplikasi multimedia interaktif. Seperti iklan *banner*, *intro film*, CD interactive, hingga pembuatan dan animasi[8].

Dalam *Flash*, terdapat teknik-teknik membuat animasi, fasilitas *action script*, *filter*, *custom easing* dan dapat memasukkan *video* lengkap dengan fasilitas *playback* FLV. Keunggulan yang dimiliki oleh *Flash* ini adalah ia mampu diberikan sedikit *code* pemograman baik yang berjalan sendiri untuk mengatur animasi yang ada didalamnya atau digunakan untuk berkomunikasi dengan program lain seperti HTML, PHP, dan *Database* dengan pendekatan XML, dapat dikolaborasikan dengan web, karena mempunyai keunggulan kecil dalam ukuran file outputnya. Aplikasi *Flash* merupakan sebuah standar aplikasi industri perancangan animasi web dengan peningkatan dalam pengaturan dan perluasan kemampuan integrasi yang lebih baik [8].

2. Adobe Photoshop

Photoshop adalah sebuah software pengolah gambar yang dibuat oleh *Adobe* yang sudah diakui kehandalannya, karena fasilitas yang diberikan didalamnya tergolong lengkap dari mulai fitur pengolahan gambar hingga efek – efek cantik yang beragam[8].

3. Adobe Illustrator

Adobe Illustrator merupakan aplikasi pengolah vektor yang handal dalam menciptakan bentuk-bentuk geometri dasar, seperti titik, garis, kurva dan juga polygon dapat di kreasikan secara menarik. Fungsi utama dari aplikasi ini adalah sebagai pengolah *vektor* namun tidak hanya itu, aplikasi ini juga dapat menggambar kartun dan juga membuat efek-efek untuk sebuah gambar [8].