BAB II

LANDASAN TEORI

2.1 Konsep Dasar *Dashboard*

2.1.1 Definisi *Dashboard*

Few (2006) menggunakan istilah *information dashboard*, yang didefinisikan sebagai tampilan visual dari informasi penting, yang diperlukan untuk mencapai satu atau beberapa tujuan, dengan memadukan dan mengatur informasi dalam satu layar (*single screen*), sehingga kinerja organisasi dapat dipantau secara sekilas. Tampilan visual mengandung pengertian bahwa penyajian informasi harus dirancang sebaik mungkin sehingga mata manusia dapat menangkap informasi secara cepat dan otak manusia dapat memahami maknanya secara benar.

2.1.2 Tujuan Penggunaan *Dashboard*

Eckerson (2006) menyatakan bahwa *dashboard* memiliki beberapa manfaat, yaitu

1. Mengomunikasikan strategi.

Dashboard digunakan untuk mengomunikasikan strategi dan tujuan yang dibuat oleh eksekutif, kepada semua pihak yang berkepentingan, sesuai dengan peran dan levelnya dalam organisasi.

2. Memantau dan menyesuaikan pelaksanaan strategi.

Dashboard digunakan untuk memantau pelaksanaan dari rencana dan strategi yang telah dibuat. *Dashboard* memungkinkan pihak eksekutif untuk mengidentifikasi permasalahan kritis dan membuat strategi untuk mengatasinya.

3. Menyampaikan wawasan dan informasi ke semua pihak.

Dashboard menyajikan informasi secara sekilas menggunakan grafik, simbol, bagan dan warna-warna yang memudahkan pengguna dalam memahami dan mempersepsi informasi secara benar.

2.1.3 Karakteristik *Dashboard*

Malik (2005) menyatakan karakteristik *dashboard* dalam akronim S-M-A-R-T (*Synergetic*, *Monitor*, *Accurate*, *Responsive*, *Timely*) dan I-M-P-A-C-T (*Interactive*, *More data history*, *Personalized*, *Analytical*, *Collaborative*, *Trackability*). Penjelasan mengenai karakteristik dapat dilihat pada Tabel 2.1.

Tabel 2.1. Karakteristik *Dashboard*

Karakteristik	Penjelasan
Synergetic	Ergonomis dan memiliki tampilan visual yang mudah dipahami oleh pengguna. *Dashboard* mensinergikan informasi dari berbagai aspek yang berbeda dalam satu layar.
Monitor	Menampilkan KPI yang diperlukan dalam pembuatan keputusan dalam domain tertentu, sesuai dengan tujuan pembangunan *dashboard* tersebut.
Accurate	Informasi yang disajikan harus akurat, dengan tujuan untuk mendapatkan kepercayaan dari penggunanya.
Responsive	Merespon *threshold* yang telah didefinisikan, dengan memberikan *alert* (seperti bunyi alarm, blinker, email) untuk mendapatkan perhatian pengguna terhadap hal-hal yang kritis.
Timely	Menampilkan informasi terkini yang diperlukan untuk pengambilan keputusan.
Interactive	Pengguna dapat melakukan *drill down* dan mendapatkan informasi lebih detail, analisis sebab akibat dan sebagainya.
More data History	Pengguna dapat melihat tren sejarah dari KPI, misalkan melihat perbandingan *market share* periode saat ini dengan beberapa tahun yang lalu, untuk mengetahui apakah kondisi sekarang lebih baik atau tidak.
Personalized	Penyajian informasi harus spesifik untuk setiap jenis pengguna sesuai dengan domain tanggung jawab, hak akses, dan batasan akses data.
Analytical	Memberikan fasilitas bagi pengguna untuk melakukan analisis, seperti analisis sebab akibat.

Tabel 2.1. Lanjutan Karakteristik *Dashboard*

Karakteristik	Penjelasan
Collaborative	Memberikan fasilitas pertukaran catatan (laporan) antar pengguna mengenai hasil pengamatan *dashboard* masing-masing, sebagai sarana untuk komunikasi dalam rangka melakukan fungsi manajemen dan kontrol.
Trackability	Memungkinkan setiap pengguna untuk mengkustomisasi metrik yang akan dilacaknya.

Sumber : Malik (2005)

2.1.4 Ciri-ciri *Dashboard* yang baik

Few (2006) mengungkapkan *Dashboard* yang didesain dengan baik akan menampilkan informasi sebagai berikut :

a. Terorganisasi dengan baik.
b. Singkat, terutama dalam bentuk ringkasan dan pengecualian.
c. Spesifik dan telah disesuaikan untuk pengguna dan tujuan *Dashboard*.
d. Ditampilkan secara ringkas dan terkadang ditampilkan dalam media kecil (tabel, grafik) sebagai sarana untuk mengomunikasikan data dan pesan dengan jelas dan langsung pada intinya.

2.1.5 Key Performance Indicator

Key Performance Indicator (KPI) merupakan sekumpulan ukuran mengenai aspek kinerja yang paling penting untuk menentukan kesuksesan organisasi pada masa sekarang dan masa yang akan datang (Parmenter, 2007).

Parmenter (2007) mempunyai tujuh karakteristik umum KPI sebagai berikut :

a. Ukuran non-finansial (tidak dinyatakan dalam bentuk nilai mata uang).
b. Diukur secara sering (dalam harian atau 24 jam / 7 hari).

c. Ditindaklanjuti oleh CEO dan tim manajemen senior.

d. Semua staf harus memahami pengukuran dan tindakan koreksi.

e. Baik individu maupun tim ikut bertanggung jawab.

f. Berpengaruh signifikan (misalnya: berpengaruh hampir pada inti semua faktor kunci keberhasilan CSF (*Critical Success Factor*) dan lebih dari satu perspektif BSC (*Balance Score Card*).

g. Berpengaruh positif (misalnya: memengaruhi ukuran kinerja yang lain secara positif).

Berikut ini adalah beberapa contoh KPI yang dibahas pada tugas akhir ini yaitu sebagai berikut :

Tabel 2.2. Contoh KPI

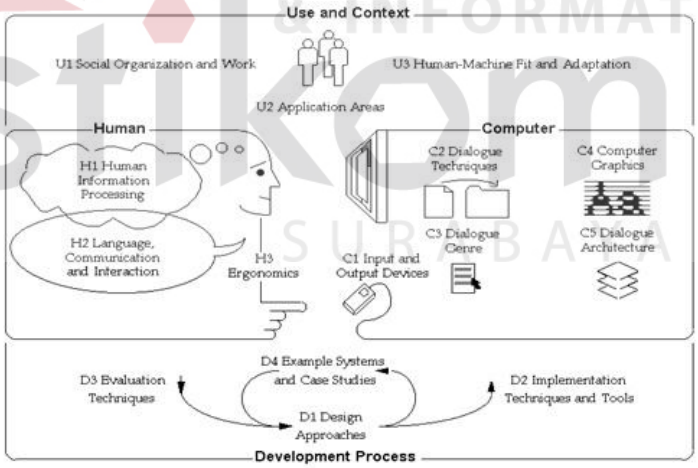
No	Kegiatan	Parameter	Skor
1	Indeks kualitas Pembelajaran	1.00 - 1.99	1
		2.00 - 2.49	2
		2.50 - 2.99	3
		3.00 - 3.49	4
		3.50 - 4.00	5
2	Kehadiran Dosen	0 – 5.99	1
		6 – 8.99	2
		9 – 11.99	3
		12 – 13.99	4
		14	5
3	Ketepatan waktu penyerahan soal ujian	Terlambat > 6 hari	1
		Terlambat 5 - 6 hari	2
		Terlambat 3 - 4 hari	3
		Terlambat 1 - 2 hari	4
		Tepat Waktu	5
4	Ketepatan waktu penyerahan nilai ujian	Terlambat > 6 hari	1
		Terlambat 5 - 6 hari	2
		Terlambat 3 - 4 hari	3
		Terlambat 1 - 2 hari	4
		Tepat Waktu	5
5	Jumlah SKS	0 – 7.99	1
		8 – 9.99	2
		10 – 11.99	3

Tabel 2.2. Lanjutan contoh KPI

No	Kegiatan	Parameter	Skor
		12 – 13.99	4
		14 – 16	5

2.2 Interaksi Manusia dan Komputer

Interaksi Manusia dan Komputer atau yang biasa dikenal sebagai *Human Computer Interaction* (HCI) merupakan satu disiplin ilmu yang mengkaji komunikasi atau interaksi di antara pengguna dengan sistem (Sudarmawan dan Ariyus, 2007). Model interaksi antara manusia dengan sistem melibatkan tiga komponen, yaitu pengguna, interaksi dan sistem itu sendiri, seperti ditunjukkan pada Gambar 2.1. Kunci utama HCI adalah daya guna (*usability*), yang berarti bahwa suatu sistem harus mudah digunakan, memberi keamanan kepada pengguna, mudah dipelajari, dan sebagainya.



Gambar 2.1 Interaksi Manusia dan Komputer
(Sudarmawan dan Ariyus, 2007)

Definisi dari interaksi manusia-komputer adalah sebagai berikut :

a. Sekumpulan proses, dialog dan kegiatan di mana melaluinya pengguna memanfaatkan dan berinteraksi dengan komputer.

b. Suatu disiplin ilmu yang menekankan pada aspek desain, evaluasi dan implementasi dari sistem komputer interaktif untuk kegunaan manusia dengan mempertimbangkan fenomena-fenomena di sekitar manusia itu sendiri.

c. Suatu studi ilmiah tentang masyarakat di dalam lingkungan kerjanya.

Menurut Rizky (2006), terdapat beberapa tips desain yang harus diperhatikan sebelum memulai sebuah proses desain *interface* sebagai berikut :

1. Memenuhi kaidah estetika

   Sebuah desain dapat disebut baik secara estetika apabila :

   a) Didalamnya terdapat perbedaan yang jelas dan kontras antar elemen dalam sebuah tampilan, misalnya tampilan tombol yang berbeda warna dengan tampilan textbox.

   b) Terdiri dari beberapa kelompok yang jelas antar inputan dan tombol proses.

   c) Antar elemen dan kelompok tampilan dipisah dengan *alignment* yang rapi.

   d) Sederhana dan tidak terlalu banyak aksesoris (gambar, animasi, icon) yang terkesan sia-sia.

2. Dapat dimengerti

   Sebuah desain harus dapat dimengerti dengan cepat dari segi tampilan secara visual, fungsi yang akan ditonjolkan, penggunaan kata-kata yang singkat dan jelas baik dalam tampilan maupun dalam perintah. Penggunaan metafora atau pemisalan yang berlebihan dalam sebuah fungsi harus dihindari.

3. Kompabilitas

   Sebuah desain *interface* harus dapat memenuhi kompabilitas dari berbagai segi antara lain :

   a) Kompatibilitas pengguna yaitu dapat digunakan oleh kalangan yang lebih luas, baik berdasarkan strata pendidikan maupun berdasarkan usia.
   b) Kompatibilitas penggunaan yaitu dapat memenuhi fungsi dan tujuan yang ingin dicapai dari perancangan sebuah perangkat lunak dan perangkat keras yang digunakan.
   c) Kompabilitas produk yaitu agar perangkat lunak dapat berjalan dengan baik di berbagai perangkat keras yang ada dan sistem operasi yang menjadi target aplikasi.

4. Komprehensif

   Sebuah sistem yang baik akan membimbing penggunanya agar dapat dan lebih mudah memahami apa yang harus diperhatikan, bagaimana cara melakukan sesuatu, kapan dan di mana melakukan sesuatu, dan mengapa harus melakukan sesuatu.

5. Konfigurabilitas

   Sebuah sistem juga harus dapat dikonfigurasi ulang jika pengguna mengignkan sesuatu berdasarkan fungsi tertentu.

6. Konsistensi

   Memiliki konsistensi dalam penempatan dan pemilihan gaya komponen visual misalnya tombol atau icon yang seragam.

7. Kontrol pengguna

   Pengguna dapat melakukan kontrol jika suatu saat terjadi kesalahan dalam proses serta pemilihan fungsi tambahan dari suatu sistem. Hindari desain yang nantinya akan membatasi pengguna dalam memilih tampilan tertentu.

8. Efisien

   Desain dibuat seefisien mungkin, terutama dalam penempatan komponen, misalnya penempatan tombol dalam sebuah panel yang dapat menarik perhatian pengguna

9. Mudah dikenali

   Gunakan anatar muka yang sudah dikenal oleh pengguna, misalnya penempatan *icon Cut, Copy, Paste* secara standar dalam sebuah toolbar.

10. Toleransi

    Tidak ada sebuah sistem yang sempurna, karenanya terdapat beberapa toleransi untuk kesalahan yang mungkin terjadi. Usakahan agara terjadi sebuah pesan yang dapat membimbing pengguna untuk keluar dari kesalahan yang terjadi.

11. Sederhana

    Terdapat lima cara membuat desain sederhana dan tetap seau keinginan pengguna, yaitu :

    a) Sembunyikan komponen visual jika tidak diperlukan.

    b) Sediakan pilihan standar atau *default*.

    c) Minimalkan penggunaan berbagai macam alignment.

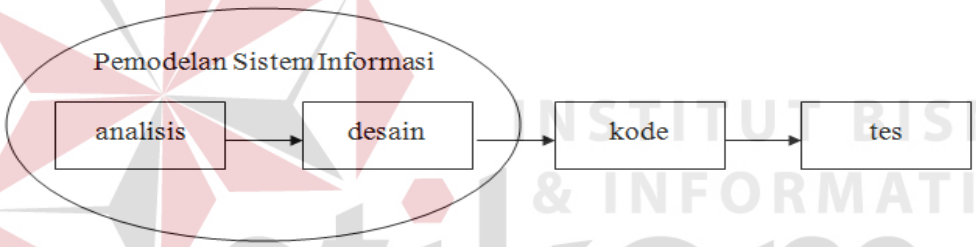
    d) Usahakan agara fungsi yang sering digunakan terlihat.

e) Perhatikan konsep konsistensi.

2.3 *System Development Life Cycle* (SDLC)

System Development Life Cycle (SDLC) adalah pendekatan melalui beberapa tahap untuk menganalisis dan merancang sistem yang dimana sistem tersebut telah dikembangkan dengan sangat baik melalui penggunaan siklus kegiatan penganalisis dan pemakai secara spesifik (Kendall dan Kendall, 2003).

Dalam penerapan SDLC terdapat beberapa model pengembangan salah satunya model *Sekuensiel Linier* yang sering juga disebut siklus hidup klasik (*classic life cycle*) atau model air terjun (*waterfall*).



Gambar 2.2 Model *Sekuensiel Linier* (Pressman, 2002)

Model *Sekuensial linier* mengusulkan sebuah pendekatan terhadap pengembangan perangkat lunak yang sistematis dan sekuensial yang mulai pada tingkat dan kemajuan sistem pada seluruh analisis, desain kode, pengujian dan pemeliharaan (Pressman, 2002). Dimodelkan setelah siklus rekayasa konvensional, model sekuensial linier melingkupi aktivitas-aktivitas sebagai berikut:

1 Analisis kebutuhan perangkat lunak. Proses pengumpulan kebutuhan diintensifikasikan dan difokuskan, khususnya pada perangkat lunak. Untuk memahami sifat program yang dibangun, perekayasa perangkat lunak (analisis)

harus memahami domain informasi, tingkah laku, unjuk kerja dan antarmuka (interface) yang diperlukan. Kebutuhan baik untuk sistem maupun perangkat lunak didokumentasikan dan dilihat lagi dengan pelanggan.

2 Desain. Desain perangkat lunak sebenarnya adalah proses multi langkah yang berfokus pada empat etribut sebuah program yang berbeda, struktur data, arsitektur perangkat lunak, representasi interface dan detail (algoritma) prosedural. Proses desain menerjemahkan syarat/kebutuhan ke dalam sebuah representasi perangkat lunak yang dapa diperkirakan demi kualitas sebelum dimulai pemunculan kode. Sebagaimana persyaratan. Desaian didokumentasikan dan menadi bagian dari konfigurasikan dan menjadi bagian dari konfigurasi perangkat lunak.

3 *Coding*. Desain harus diterjemahkan ke dalam bentuk mesin yang bisa dibaca. Langkah pembuatan kode melakukan tugas ini. Jika desain dilakukan dengan cara yang lengkap, pembuatan kode dapat diselesaikan secara mekanis.

4 Pengujian. Sekali kode dibuat, pengujian program dimulai. Proses pengujian berfokus pada logika internal perangkat lunak memastikan bahwa semua pernyataan sudah diuji, dan pada seksternal fungsional. Yaitu mengarahkan pengujian untuk menemukan kesalahan-kesalahan dan memastikan bahwa input yang dibatasi akan memberikan hasil aktual yang sesuai dengan hasil yang dibutuhkan.

2.4 Web

Aplikasi web adalah sebuah sistem informasi yang mendukung interaksi pengguna melalui antarmuka berbasis web. Fitur-fitur aplikasi web biasanya berupa *data persistence*, mendukung transaksi dan komposisi halaman web

dinamis yang dapat dipertimbangkan sebagai hibridisasi, antara hipermedia dan sistem informasi (Simamarta, 2010). Sedangkan Hanson (2000) menyebutkan web merupakan sistem yang menyebabkan pertukaran data di internet menjadi mudah dan efisien yang terdiri atas 2 komponen dasar yaitu :

a) Web *Server* adalah sebuah komputer dan software yang menyimpan dan mendistribusikan data ke komputer lainnya melalui internet.
b) Web *Browser* adalah perangkat lunak yang dijalankan pada komputer pengguna atau *client* yang meminta informasi dari *server* web dan menampilkannya sesuai dengan *file* data itu sendiri.

Dalam membangun web dibutuhkan bahasa pemrograman yang sesuai dengan kebutuhan pengguna. Menurut Indrajani (2007), bahasa pemrograman adalah perangkat lunak atau *software* yang dapat digunakan dalam proses pembuatan program yang melalui beberapa tahapan-tahapan penyelesaian masalah.

Proses pemrograman komputer bukan saja sekedar menulis suatu urutan instruksi yang harus dikerjakan oleh komputer akan tetapi bertujuan untuk memecahkan suatu masalah serta membuat mudah pekerjaan pengguna komputer (*user*). Didalam membuat sebuah program komputer, tentu tidak terlepas dari sifat individu pemrogram (*Programmer*). Karakteristik seorang pemrogram yang mutlak dimiliki yaitu:

a. Memiliki pola pikir yang logis
b. Memiliki ketekunan dan ketelitian yang tinggi
c. Memiliki penguasaan bahasa pemrograman yang baik
d. Memiliki pengetahuan teknik pemrograman yang baik

Untuk membuat suatu Program yang kompleks tahap-tahap yang harus dilakukan *programmer* adalah :

1. Definisi Masalah

Programmer harus memahami permasalahan yang timbul kemudian mengidentifikasikan permasalah yang ada sehingga dapat menentukan batasan masalah.

2. Analisa Kebutuhan

Programmer harus menentukan kebutuhan data untuk masukan dan keluaran yang diminta, bahasa pemrograman yang digunakan serta tipe komputer apa sebagai pendukungnya.

3. Desain Algoritma

Algoritma yang didesain harus memiliki kebenaran secara logika sebelum siap diimplementasikan ke dalam bentuk program.

4. Bahasa Pemrograman

Bahasa Pemrograman adalah media untuk membuat Program.

5. *Testing* dan *Debugging*

Testing untuk menguji program sampai terbebas dari kesalahan.

Adapun jenis-jenis bahasa pemrograman web yang berjalan pada Sistem Operasi Windows sebagai berikut :

a) Bahasa Pemrograman HTML

Menurut Firdaus (2007), *Hypertext Markup Language* (HTML) merupakan salah satu pemrograman web yang bersifat statis. HTML ini lebih berfungsi untuk mengatur struktur tampilan web, membuat suatu "*Link*" atau sambungan ke lokasi di internet yang lain dan aplikasi agar bisa berjalan di

halaman web *browser*. HTML saat ini merupakan standar Internet yang didefinisikan dan dikendalikan penggunanya oleh *World Wide Web Consortium* (W3C).

b) Bahasa Pemrograman CSS

Menurut Diar Puji Oktavian (2010), *Cascading Style Sheet* (CSS) berfungsi untuk mengatur tampilan dengan kemampuan jauh lebih baik dari *tag* maupun atribut standar HTML. CSS sebenarnya adalah suatu kumpulan atribut untuk fungsi format tampilan dan dapat digunakan untuk mengontrol tampilan banyak dokumen secara bersamaan. Tampilan CSS terkadang dapat berbeda tergantung pada tipe *browser* yang dipakai**.**

c) Bahasa Pemrograman PHP

Menurut Firdaus (2007), *Hypertext Prepocessor* (PHP) merupakan sebuah bahasa *scripting* berbasis *server side scripting* yang terpasang pada HTML dan berada di *server* dan digunakan untuk membuat halaman *web* yang dinamis. Saat ini PHP adalah bahasa pemrograman script yang paling banyak dipakai untuk membuat situs web yang dinamis, walaupun tidak tertutup kemungkinan digunakan untuk pemakaian lain.

d) Bahasa Pemrograman ASP

Menurut Kurniawan (2000), *Active Server Page* (ASP) adalah halaman web yang memuat *script* atau program yang akan diakses oleh web server sebelum digunakan oleh user. ASP bersifat *server side* sehingga dapat diakses melalui browser apa pun. ASP juga merupakan salah satu produk teknologi yang disediakan oleh Microsoft.

e) Bahasa Pemrograman JavaScript

Menurut Hakim (2010), Java Script merupakan bahasa *scripting* yang dapat bekerja di sebagian besar *web browser*. *Java Script* dapat disisipkan didalam *web* menggunakan *tag scrip* dan berjalan pada sisi client. Agar dapat menjalankan *script* yang ditulis menggunakan JavaScript, dibutuhkan JavaScript-*enabled browser* yaitu *browser* yang mampu menjalankan JavaScript.

f) Bahasa Pemrograman AJAX

Menurut Yoevestian (2008), AJAX adalah sebuah fitur yang bisa digunakan untuk membantu pembangunan web. Tujuannya adalah untuk membangun sebuah web yang lebih cepat tanggap, dengan melakukan pertukaran data dalam ukuran lebih kecil dengan server secara diam-diam. AJAX merupakan bahasa pemrograman yang dapat membuat web lebih interaktif.

Berikut ini adalah perbandingan antar bahasa pemrograman yang disebutkan di atas adalah sebagai berikut :

Tabel 2.3. Perbandingan bahasa pemrograman web

Kategori	PHP	ASP	HTML	CSS	JavaScript	AJAX
Koneksi Database	Banyak	Banyak	-	-	-	-
Jenis Pemrograman	Server Side	Server Side	Client Side	Client Side	Client Side	Client Side
Sistem Operasi	Banyak	Windows	Banyak	Banyak	Banyak	Banyak
Development	Cepat	Kurang Cepat	Cepat	Cepat	Cepat	Cepat

Berdasarkan dari perbandingan dari beberapa kategori diatas yang dapat digunakan pada tugas akhir ini adalah PHP dan ASP. Hal ini dikarenakan kedua bahasa pemrograman tersebut sama-sama berjenis *Server Side* dan mampu

terkoneksi kedalam database. Berikut ini adalah perbandingan *performance* dari PHP dan ASP yang sudah diuji oleh Wrensoft adalah sebagai berikut :

1) Pengujian pada website skala kecil yang terdiri dari 400 halaman dan 296.601 kata

Tabel 2.4. Pengujian pada website skala kecil (waktu dalam detik)

	1 *words*			2 *words*	3 *words*
Platform	***Test* 1**	***Test* 2**	***Test* 3**	***Test* 4**	***Test* 5**
PHP	0.0397	0.0320	0.0317	0.0323	0.0327
ASP	0.0767	0.0600	0.0637	0.0633	0.0647

Sumber : Wrensoft (2014)

2) Pengujian pada website skala sedang yang terdiri dari 60.000 halaman dan 5.638.632 kata

Tabel 2.5. Pengujian pada website skala sedang (waktu dalam detik)

	1 *words*			2 *word*	3 *words*
Platform	***Test* 6**	***Test* 7**	***Test* 8**	***Test* 9**	***Test* 10**
PHP	0.8643	0.8413	0.8247	0.9120	0.9437
ASP	2.3673	2.2343	2.4000	2.4603	2.5027

Sumber : Wrensoft (2014)

Pengujian diatas menggunakan spesifikasi server sebagai berikut :

- Intel Core i7-4790 @ 3.60 GHz
- Windows 8.1 Pro (64-bit)
- Plextor M6e PCI-E SSD 256 GB
- 16 GB of RAM

Dari 2 (dua) pengujian diatas dapat disimpulkan bahwa bahasa pemrograman yang sesuai dengan kebutuhan untuk tugas akhir ini adalah PHP. Beberapa alasannya adalah sebagai berikut :

- PHP memiliki *performance* lebih cepat dari ASP.
- PHP mendukung banyak database terutama Oracle yang digunakan pada server Stikom Surabaya.
- PHP dari pengembangan sangat mudah karena didukung banyak referensi dan komunitas. Hal tersebut dapat memudahkan dalam pengembangan web Sicyca.
- PHP dapat berjalan di *multi platform*. Hal ini dapat membantu apabila Stikom Surabaya melakukan migrasi ke server berbasis Linux.

2.5 Black Box Testing

Menurut Al Fatta (2007), *black box testing* dilakukan tanpa pengetahuan detil struktur internal dari sistem atau komponen yang dites. Biasanya disebut juga sebagai *behavioral testing, specification-based testing, input/output testing atau functional testing*. *Black box testing* berfokus pada kebutuhan fungsional pada software, berdasarkan pada spesifikasi kebutuhan dari software.

Dengan adanya *black box testing*, perekayasa software dapat menggunakan sekumpulan kondisi masukan yang dapat secara penuh memeriksa keseluruhan kebutuhan fungsional pada suatu program. *Black box testing* bukan teknik alternatif daripada *white box testing*. Lebih daripada itu, *black box testing* merupakan pendekatan pelengkap dalam mencakup error dengan kelas yang berbeda dari metode *white box testing.*