BAB II

LANDASAN TEORI

2.1. Sistem Informasi

Sistem adalah kumpulan dari komponen-komponen peralatan model *requirements*, *fuctions*, dan *interface* sedangkan informasi adalah suatu hasil olahan data dalam bentuk agregat untuk menghasilkan pengetahuan atau dengan kata lain suatu yang dapat menambah pengetahuan penerimanya jika dibangun dari data yang tepat dan sesuai dengan permasalahan yang dihadapi. Jadi, sistem informasi adalah sekumpulan elemen yang bekerja secara bersama-sama baik secara secara manual maupun berbasis komputer dalam melaksanakan pengolaan data yang berupa pengumpulan, penyimpanan, pemprosesan data untuk menghasilkan informasi yang bermakna dan berguna bagi proses pengambil keputusan (Simarmata, 2009).

2.2. Perjalanan Dinas

Definisi perjalanan dinas :

1. Perjalanan Dinas adalah aktifitas yang dilakukan di luar Wilayah Kerja karyawan yang mengharuskannya melakukan perjalanan dari tempat kedudukan kerja / tempat tinggalnya menuju lokasi kerja yang dimaksud baik menginap ataupun tidak sekurang-kurangnya selama 1 (satu) hari atau maksimum selama 14 (empat belas) hari kalender, termasuk dalam hal ini perjalanan dari tempat kedudukan kerja / tempat tinggal ke lokasi embarkasi untuk meninggalkan Indonesia dan / atau dari lokasi debarkasi ke tempat kedudukan kerja / tempat tinggal di dalam negeri untuk kepentingan perusahaan atas perintah pejabat berwenang.
2. Wilayah Kerja adalah cakupan area kerja karyawan yang bisa berupa area satu kota / kabupaten atau beberapa kota / kabupaten tertentu yang ditetapkan oleh Manajemen bagi

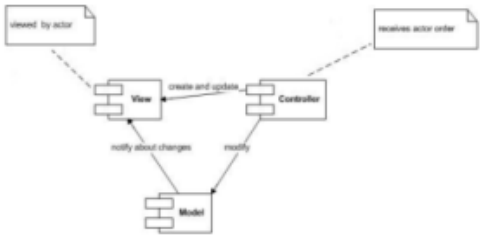
seorang karyawan. Ketetapan Wilayah Kerja bagi masing-masing Karyawan ditetapkan oleh Direksi dan dipublikasikan oleh bagian HRD.

3. Perusahaan adalah Infoglobal Group
4. Karyawan adalah karyawan tetap atau tenaga kerja waktu tertentu di Infoglobal Group
5. Kategori Jabatan adalah level jabatan structural/fungsional Karyawan. Ketetapan kategori Jabatan bagi masing-masing Karyawan ditetapkan oleh Direksi dan dipublikasikan oleh bagian HRD.
6. Reimbuse adalah penggantian pengeluaran biaya sesuai nilai pengeluaran yang sebenarnya, berdasarkan bukti pengeluaran yang sah (at cost)
7. Lumpsum adalah sejumlah tertentu uang yang dibayarkan sekalligus tanpa memperhitungkan nilai pengeluaran yang sebenarnya
8. Surat Perintah Perjalanan Dinas (SPPD) adalah surat penugasan kepada karyawan untuk melaksanakan Perjalanan Dinas

2.3. Pola Arsitektural Model View Control

Menurut Lethbridge dan Laganiere (Lethbridge & Laganiere, 2002, hal. 337-339), MVC merupakan suatu pola arsitektural yang digunakan untuk membantu memisahkan lapisan antarmuka pengguna dari bagian-bagian lain sistem. MVC tidak hanya membantu menjalankan keterpaduan lapisan dari lapisan-lapisan antarmuka pengguna, tetapi juga membantu mengurangi tercampurnya lapisan tersebut dengan sistem yang lainnya, demikian juga antara aspek-aspek berbeda dari *user interface* itu sendiri.

Pola MVC memisahkan lapisan fungsionalitas dari sistem (model) dari dua aspek antarmuka pengguna, yaitu *view* dan *controller*.



Gambar Error! No text of specified style in document**..1**
Pola Arsitektur MVC Antarmuka Pengguna

Walaupun ketiga komponen umumnya merupakan instansi dari satu kelas, kita menggunakan diagram komponen untuk menekankan fakta bahwa komponen-komponen juga dapat dipisahkan atau diproses.

Model terdiri dari kelas-kelas pokok yang mana instansi-instansinya akan dilihat dan dimanipulasi. *View* terdiri dari objek-objek yang digunakan untuk membuat tampilan data dari model di antarmuka pengguna. *View* juga menampilkan berbagai macam kontrol dengan mana pengguna dapat berinteraksi. *Controller* terdiri dari objek-objek yang mengontrol dan menangani interaksi pengguna dengan *view* dan model, memiliki logika yang merespon pada saat pengguna mengetikkan sesuatu pada suatu field atau mengklik kontrol mouse.

2.4. Framework CodeIgneter

CodeIgneter adalah sebuah framework menggunakan bahasa pemrograman PHP yang dibuat oleh Qiang Xue tanggal 1 Januari 2008. Sebelumnya Qiang juga mengembangkan framework PRADO selama bertahun-tahun. CI secara resmi dirilis untuk memenuhi kebutuhan para pengembang perangkat lunak berbasis web (Badiyanto, 2013, p. 7).

Framewok CI adalah framework PHP berbasis komponen untuk pengembangan aplikasi web berskala besar. Framework CI menyediakan reusability maksimum dalam pemrograman web dan mampu meningkatkan kecepatan dalam membuat aplikasi web. Nama Yii diambil dari singkatan “Yes It Is!” (Badiyanto, 2013, p. 7).

Membangun sebuah aplikasi dengan framework CI, pengembang cukup menggunakan kelas yang tersedia di PHP sendiri. Semua yang dibangun menggunakan CI menggunakan arsitektur Model-View-Controller (MVC). (Badiyanto, 2013, p. 8)

2.5. PHP

PHP: Hypertext Preprocessor adalah bahasa skrip yang dapat ditanamkan atau disisipkan kedalam HTML/PHP banyak dipakai untuk membuat situs web dinamis. PHP dapat juga digunakan untuk membangun sebuah CMS. Sebagian besar sintaks mirip dengan bahasa C, Java, dan Perl, ditambah beberaoa fungsi PHP yang spesifik. Tujuan utama penggunaan bahasa ini adalah untuk memungkinkan perancang dan menulis halaman web menjadi dinamis dengan cepat (Badiyanto, 2013, p. 32).

Pada saat ini PHP cukup populer sebagai peranti pemrograman web, terutama di lingkungan Linux. Walaupun demikian, PHP sebenarnya juga dapat berfungsi pada *server-server* yang berbasis UNIX, Windows, dan Macintosh (Kadir, 2002).

2.6. MySQL

Database MySQL merupakan sebuah database server SQL multiuser dan multithreaded (Badiyanto, 2013, p. 57). Sebagai *database server* dengan konsep *database* modern, MySQL memiliki keistimewaan. Beberapa keistimewaan dimiliki MySQL sebagai berikut:

a. Portability

Database MySQL berfungsi dengan stabil tanpa kendala, berarti berlaku pada berbagai sistem operasi seperti Windows, Linux, FreeBSD, Mac OS X *Server*, Solaris, Amiga, HP-Unix, dan lain-lain.

b. Open Source

MySQL merupakan *database open source* (gratis), di bawah lisensi GPL

c. *Multiuser*

MySQL merupakan *database* yang dapat digunakan untuk menangani beberapa *user* dalam waktu bersamaan tanpa mengalami masalah. Dan memungkinkan sebuah *database server* MySQL dapat diakses *client* secara bersamaan pula.

d. *Performace Tuning*

MySQL mempunyai kecepatan yang cukup baik dalam menangani *query-query* sederhana, serta mampu memproses lebih banyak SQL per satuan waktu.

e. *Column Type*

Database MySQL didukung dengan tipe data yang sangat kompleks, seperti *signed/unsigned integer, float, double, char, varchar, text, blob, data, time, datetime, timestamp, year, set* serta *enum*.

f. *Command* dan *Functions*

MySQL *server* memiliki operator dan fungsi secara penuh yang mendukung perintah *SELECT* dan *WHERE* dalam *query*.

g. *Security*

Sistem *security* pada MySQL mempunyai beberapa lapisan sekuritas seperti tingkatan *subnetmask, hostname*, dan izin akses *user* dengan sistem perizinan yang mendetil serta *password* terenkripsi.

h. *Scalability* dan *Limits*

MySQL mempunyai kemampuan menangani *database* dalam skala cukup besar, dengan jumlah *record* lebih dari 50 juta dan 60 ribu tabel serta 5 miliar baris. Selain itu dapat menampung indeks sampai 32 indeks pada tiap tabelnya.

i. *Connectivity*

Adanya kemampuan MySQL melakukan koneksi dengan *client* menggunakan protokol TCP/IP, Unix socket (Unix), atau Named Pipes (NT).

j. *Localization*

Adanya kemampuan dalam mendeteksi kesalahan (*error code*) pada *client* menggunakan lebih dari dua puluh bahasa.

k. *Interface*

MySQL memiliki *Interface* terhadap berbagai aplikasi dan bahasa pemograman menggunakan fungsi API (*Application Programming Interface*).

l. *Clients* dan *Tools*

Database MySQL dilengkapi berbagai *tools* yang dapat digunakan untuk administrasi *database*.

m. Struktur Tabel

MySQL memiliki struktur tabel cukup baik serta cukup fleksibel (Syaukani, 2005, hal. 2).

2.7. Prototype

Prototype merupakan suatu metode dalam pengembangan sistem yang menggunakan pendekatan untuk membuat sesuatu program dengan cepat dan bertahap sehingga segera dapat dievaluasi oleh pemakai. Hal ini berbeda dengan pendekatan SDLC tradisional (konvensional) yang lebih banyak menghabiskan waktu untuk menghasilkan spesifikasi yang sangat rinci sebelum pemakai dapat mengevaluasi sistem. Mengingat kebanyakan pemakai mengalami kesulitan dalam memahami spesifikasi sistem berakibat bahwa pemakai tidak begitu paham sampai pengujian dilakukan. Selain itu, *prototype* membuat proses pengembangan sistem

informasi menjadi lebih cepat dan lebih mudah, terutama pada keadaan kebutuhan pemakai sulit untuk diidentifikasi (Kadir, 2002, hal. 416).

Pendekatan model *prototype* digunakan jika pemakai hanya mendefinisikan objektif umum dari perangkat lunak tanpa merinci kebutuhan input, pemrosesan dan outputnya, sementara pengembang tidak begitu yakin akan efisiensi algoritma, adaptasi sistem operasi, atau bentuk interaksi manusia-mesin yang harus diambil. (Nugroho, Ratnasari, Ramadhani, & Putro, 2009, hal. 18)

Adakalanya dijumpai pendapat bahwa *prototype* itu sama dengan metode RAD atau *Rapid Application Development* (O' Brien, 2002), tetapi ada juga yang menyatakan bahwa *prototype* itu menyerupai RAD (Turban, McClean, & Wetherbe, Information Technology for Management Making Connections for Strategis Advantage, 1999). Secara garis besar sasaran *prototype* adalah sebagai berikut (Lucas, 2000) :

a. Mengurangi waktu sebelum pemakai melihat sesuatu yang konkret dari usaha pengembangan sistem.
b. Menyediakan umpan balik yang cepat dari pemakai kepada pengembang.
c. Membantu menggambarkan kebutuhan pemakai dengan kesalahan yang lebih sedikit.
d. Meningkatkan pemahaman pengembang dan pemakai terhadap sasaran yang seharusnya dicapai oleh sistem.
e. Menjadikan keterlibatan pemakai sangat berarti dalam analisis dan desain sistem.

Perlu diketahui, *prototype* dapat berdiri sebagai metode pengembangan tersendiri, tetapi juga dapat menjadi bagian dari SDLC. Beberapa versi SDLC yang lebih baru sering kali menyertakan *prototype* sebagai alternatif atau suplemen dalam tahapan analisis dan desain sistem (Turban, McClean, & Wetherbe, Information Technology for Management Making Connections

for Strategis Advantage, 1999). Dalam banyak kasus, *prototype* lebih digunakan untuk mendukung SDLC daripada untuk menggantikannya (Roomney, Steinbart, & Cushing, 1997).

Tabel Error! No text of specified style in document.**.1**
Kelebihan dan Kekurangan *Prototype*
(Nugroho, Ratnasari, Ramadhani, & Putro, 2009, hal. 19)

Kelebihan *Prototype*	Kekurangan *Prototype*
Mendefinisikan objektif secara keseluruhan dan mengidentifikasi kebutuhan yang sudah diketahui.	Walaupun pemakai melihat berbagai perbaikan dari setiap versi *prototype*, tetapi pemakai mungkin tidak menyadari bahwa verdi tersebut dibuat tanpa memperhatikan kualitas dan pemeliharaan jangka panjang.
Melakukan perancangan secara cepat sebagai dasar untuk membuat *prototype*.	Pengembang kadang-kadang membuat kompromi implementasi dengan menggunakan sistem operasi yang tidak relevan dan algoritma yang tidak efisien.
Menguji coba dan mengevaluasi *prototype* dan kemudian melakukan penambahan dan perbaikan-perbaikan terhadap *prototype* yang sudah dibuat.	-

2.8. Use Case Diagram

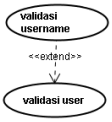
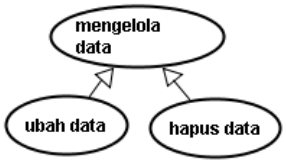
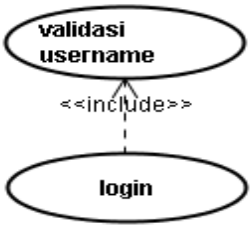
Fungsionalitas *use case* direpresentasikan dengan alira-aliran peristiwa. *Use case* menggambarkan bagaimana realisasi dari setiap *use case* yang ada pada model (Yulianto, Gartina, Sari, Astuti, & Witanti, 2009, hal. 92).

Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut aktor dan *use case*.

a. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah orang, tapi aktor belum tentu merupakan orang.
b. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antarunit atau aktor.

Berikut adalah simbol-simbol yang ada pada diagram *use case* :

Tabel Error! No text of specified style in document.**.2**
Simbol-simbol *Use Case Diagram*

Simbol	Deskripsi
Use case 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.
Aktor 	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri.
Asosiasi	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor.
Extend 	*Case* tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan. Arah panah mengarah pada *use case* yang ditambahkan.
Generalisasi 	Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *use case* dimana fungsi yang satu lebih umum dari lainnya. Arah panah mengarah pada *use case* yang menjadi generalisasinya (umum).
Include / *Use*s 	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini. Ada dua sudut pandang mengenai penggunaan inckude di *use case* : a. Include berarti *use case* yang ditambahkan akan selalu dipanggil saat *use case* tambahan dijalankan. b. Include berarti *use case* yang ditambahkan akan selalu melakukan pengecekan apakah *use case* yang ditambahkan telah dijalankan.

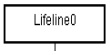
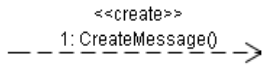
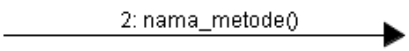
Arah panah relasi pada use *case* mengarah pada use *case* yang lebih besar kontrolnya atau yang dipakai (Ariani & Shalahuddin, 2011, hal. 130-133).

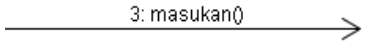
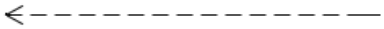
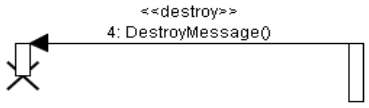
2.9. Sequence Diagaram

Sequence diagram menggambarkan interaksi antara sejumlah objek dalam urutan waktu. Kegunaannya untuk menunjukkan rangkaian pesan yang dikirim antara objek juga interaksi antar objek yang terjadi pada titik tertentu dalam ekesekusi sistem (Yulianto, Gartina, Sari, Astuti, & Witanti, 2009, hal. 93).

Banyaknya *sequence diagram* yang harus digambar adalah sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada *sequence diagram* sehingga semakin banyak *use case* yang didefinisikan maka *sequence diagram* yang harus dibuat juga semakin banyak. Berikut adalah simbol-simbol yang ada pada *sequence diagram* :

Tabel Error! No text of specified style in document.**.3**
Simbol-Simbol *Sequence Diagram*

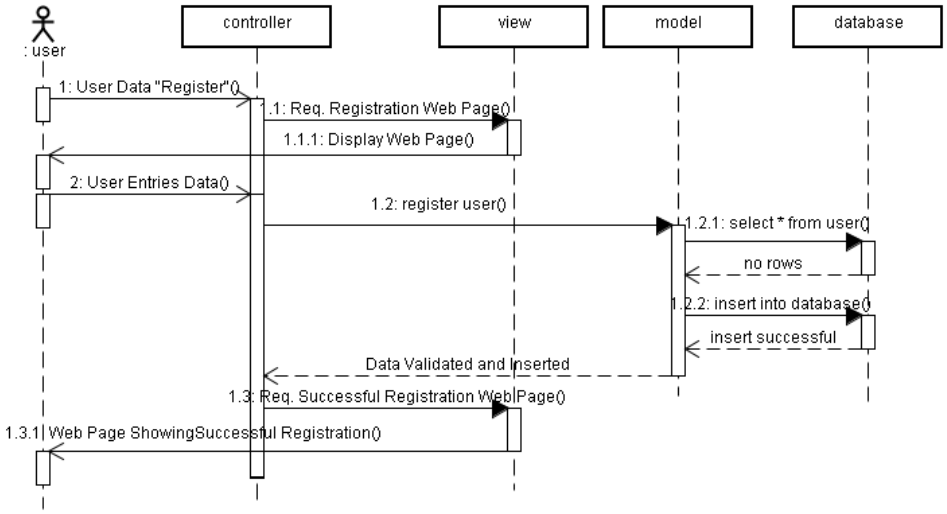
Simbol	Deskripsi
Aktor  atau 	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri.
Garis hidup	Menyatakan kehidupan suatu objek.
Objek	Menyatakan objek yang berinteraksi pesan.
Waktu aktif 	Menyatakan objek dalam keadaan aktif dan berinteraksi pesan.
Pesan tipe *create* 	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat.
Pesan tipe *call* 	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri, arah panah mengarah pada objek yang memiliki operasi/metode.
Pesan tipe *send*	Menyatakan bahwa suatu objek mengirimkan data/masukan/informasi ke objek

Simbol	Deskripsi
	lainnya, arah panah mengarah pada objek yang dikirimi.
Pesan tipe *return* 	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian.
Pesan tipe *destroy* 	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada *create* maka ada *destroy*.

Penggambaran letak pesan harus berurutan, pesan yang lebih atas dari lainnya adalah pesan yang berjalan terlebih dahulu (Ariani & Shalahuddin, 2011, hal. 137-140).

2.9.1. Sequence Diagram Model View Controller

Contoh *sequence diagram* pada MVC :



Gambar Error! No text of specified style in document.**.2**
Contoh Sequence Diagram pada MVC

Pada diagram di atas dijelaskan bahwa ketika seorang *user* memilih menu *register*, maka *controller* akan mengirimkan permintaan menampilkan halaman registration *web page* kepada *view*. Setelah halaman registration ditampilkan ke *user*, maka *user* menginputkan data, data akan

dikirimkan ke *database* untuk di cek keberadaannya. Jika data belum terekam, maka data yang tadi akan diinputkan ke dalam *database*. Kondisi data yang valid akan dikirimkan ke *controller*, sehingga *controller* akan mengirimkan permintaan *registration success* ke *view*. *View* akan menampilkan halaman *registration success* ke *user* (Idaho, 2002).

2.10. Activity Diagram

Diagram aktivitas atau *activity diagram* memodelkan *workflow* (aliran kerja) atau urutan aktivitas dalam sebuah proses (Yulianto, Gartina, Sari, Astuti, & Witanti, 2009, hal. 106). Diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan oleh aktor, jadi aktivitas yang dapat dilakukan oleh sistem. Diagram aktivitas juga banyak digunakan untuk mendefiniskan hal-hal berikut :

a. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
b. Urutan atau pengelompokan tampilan dari sistem/*user interface* di mana setiap ativitas dianggap memiliki sebuah rancangan antarmuka tampilan.
c. Rancangan pengujian di mana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya. (Ariani & Shalahuddin, 2011, hal. 134)

Berikut adalah simbol-simbol yang ada pada diagram aktivitas : (Ericsson, 2011)

Tabel Error! No text of specified style in document.**.4**
Simbol-Simbol *Activity Diagram*

Simbol	Keterangan
Status Awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki status awal.
Aktivitas 	Aktivitas yang dilakukan sistem, biasanya diawali dengan kata kerja

Percabangan 	Asosiasi percabangan terjadi dimana jika ada pilihan aktivitas lebih dari satu
Penggabungan	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
Status Akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.



Gambar Error! No text of specified style in document.**.3**
Activity Diagram

Terdapat dua poin pada diagram, yaitu (Larman, 2005, hal. 478) :

a. Setelah sebuah aksi berakhir, maka akan menghasilkan transisi keluar secara otomatis.

b. Diagram menampilkan aliran kontrol dan aliran data.

2.11. Blackbox Texting

Pengujian ini fokus kepada persyaratan fungsional perangkat lunak. Pengujian ini memungkinkan pelaku RPL mendapatkan serangkaian kondisi input yang memenuhi persyaratan fungsional suatu program.

Pengujian ini berusaha menemukan kesalahan dengan kategori sebagai berikut:

a. Fungsi-fungsi yang salah atau hilang

b. Kesalahan antarmuka

c. Kesalahan struktur data atau akses basisdata eksternal

d. Kesalahan kinerja

e. Kesalahan inisialisasi atau terminasi (Nugroho, Ratnasari, Ramadhani, & Putro, 2009, hal. 121)

Pengujian dirancang untuk menjawab beberapa pertanyaan sebagai berikut :

a. Bagaimana validitas fungsionalnya diuji ?

b. Bagaimana perilaku sistem dan performansi diuji ?

c. Jenis *input* seperti apa yang akan menghasilkan kasus uji yang baik ?

d. Apakah sistem secara khusus sensitif terhadap nilai *input* tertentu ?

e. Bagaimana batasan-batasan kelasi data disolasi ?

f. Berapa *ratio* data dan jumlah data yang dapat ditoleransi oleh sistem ?

g. Apa akibat yang timbul dari kombinasi spesifik data pada operasi sistem ?

Meskipun dalam pelaksanaan testing kita dapat menguji keseluruhan fungsionalitas perangkat lunak namun formal *black box testing* yang sebenarnya kita dapat memilih *subset test* yang secara efektif dan efisien dapat menemukan cacat. Dengan cara ini *black box testing* dapat membantu memaksimalkan *testing investment* (Hendraputra, et al., 2009).