BAB II
LANDASAN TEORI

1.7. Tinjauan Pustaka

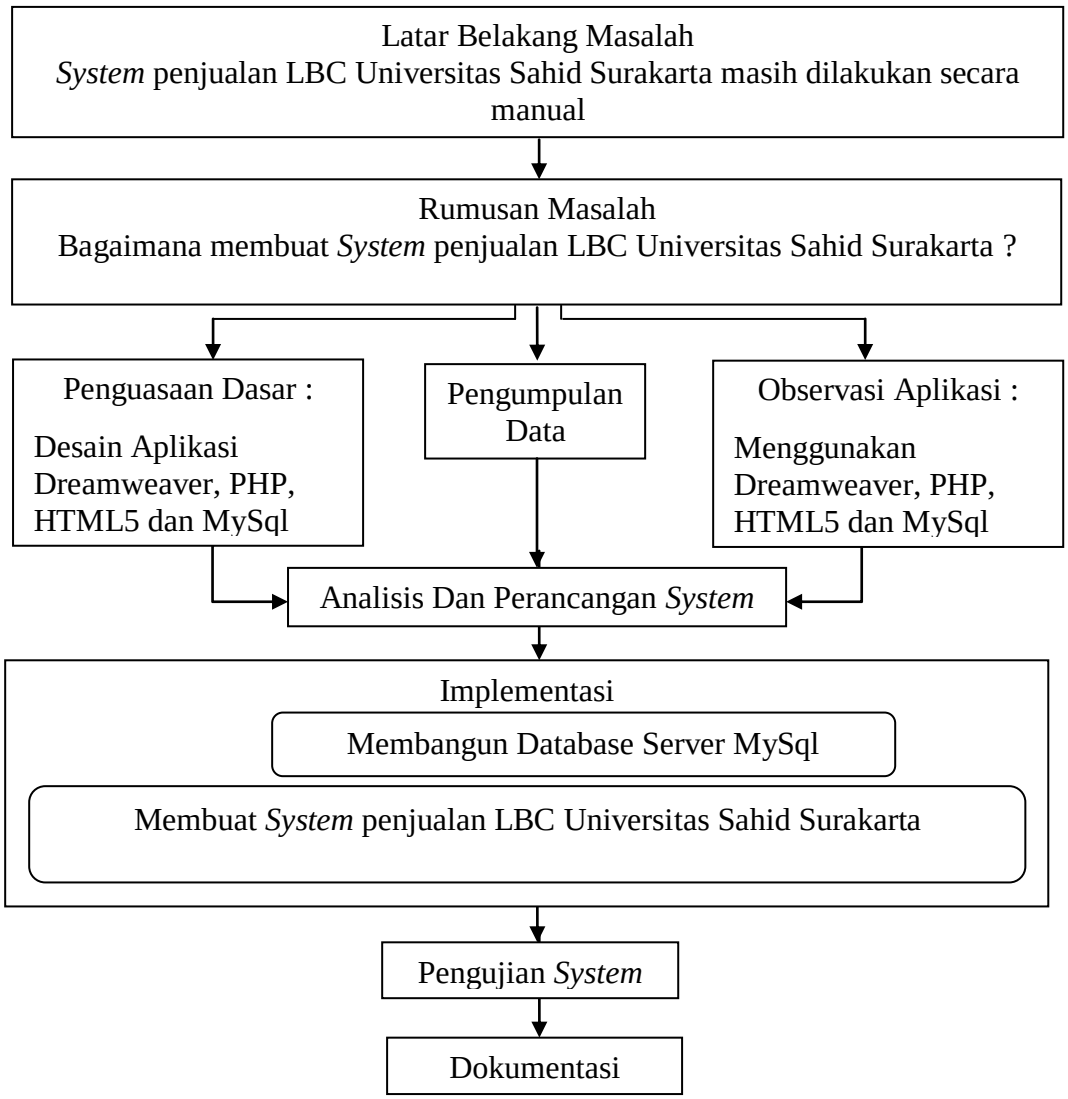
2.1.1. Kajian Pustaka

Andrian (2014), dalam penelitian yang berjudul "*System* Informasi Penjualan Dan Monitoring Di Toko Royal Motor Bandung" menjelaskan bahwa *System* informasi penjualan yang terintegrasi dengan data barang dapat memudahkan dalam transaksi pejualan. Monitoring persedian stok barang, agar dapat melakukan pembelian kembali dapat menggunakan metode pengisian kembali persediaan yaitu *Re Order Point*. Dengan menggunakan metode ini, maka pembelian ulang akan dilakukan apabila jumlah stok barang berada dibawah titik pembelian ulang. Titik pembelian ulang adalah suatu kondisi dimana jumlah stok persediaan sama dengan permintaan selama masa waktu pesanan *(lead time)*, tetapi karena adanya permintaan yang bersifat fluktuatif, maka persediaan pengaman (*safety stock*) perlu ditambahkan ke dalam perhitungan titik pengisian ulang tersebut.

Pada penelitian yang dilakukan oleh Arvyaningrum (2012), yang berjudul "*System* Informasi Penjualan Buku Pada Toko Buku Pustaka Gemilang Utama", diuraikan bahwa penjualan buku pada pustaka gemilang utama masih dilakukan secara manual sehingga kinerjanya belum efektif. Hal itu tercermin pada sering terjadinya keterlambatan penyusunan laporan penjualan, dan kesalahan pencatatan serta persediaan perhitungan.

Jamal dan Yulianto (2013), berkaitan dengan proses pembuatan nota penjualan dan perhitungan jumlah harga penjualan di Toko dan Jasa Widodo Computer Ngadirojo sering terjadi permasalahan berupa ketidakcocokan data ini disebabkan akibat kesalahan manusia (*humam error*) dimana terjadi kelalaian yang dilakukan oleh petugas kasir, kelalaian yang dilakukan terjadi ketika kasir melakukan pelayanan saat banyak konsumen yang melakukan transaksi pembayaran, sehingga kasir sering melakukan kesalahan karena kurang teliti dalam pembuatan nota, perhitungan jumlah total dan pencatatan data ke buku arsip penjualan.

1.8. Kerangka Pemikiran



Gambar 2.1. Diagram Kerangka Pemikiran

Penjelasan kerangka pemikiran (Gambar 2.1) :

1. Latar Belakang Masalah

   Latar belakang Tugas Akhir ini adalah *System* penjualan LBC Universitas Sahid Surakarta masih dilakukan secara manual.

2. Rumusan Masalah

   Bagaimana membuat *System* penjualan LBC Unversitas Sahid Surakarta ?

3. Pengumpulan Data Tertulis dan Tidak Tertulis

   Mengumpulkan semua data yang diperlukan dalam penelitian, baik melalui *interview*, observasi maupun dokumentasi.

4. Penguasaan Dasar

   Penulis mulai membuat tampilan atau tema dan pemrograman dasar PHP serta HTML5 menggunakan dreamweaver dan pembuatan database MySql dengan tujuan supaya lebih menguasai tentang bahasa pemrograman tersebut.

5. Observasi Aplikasi

   Mencari beberapa aplikasi atau tinjauan pustaka yang berkaitan dengan administrasi surat baik melalui internet, karya ilmiah, buku yang dapat dijadikan referensi dalam membuat *System* penjualan LBC Universitas Sahid Surakarta.

6. Analisis dan Perancangan *System*

   Menganalisis *System* yang berjalan saat ini, serta merancang *System* yang akan di bangun seperti apa perancangan tampilan dan menu apa saja yang akan ada dalam *System* tersebut.

7. Implementasi

   Membangun database *server* MySql sesuai dengan data-data yang didapatkan dengan kebutuhan *System* dan membuat *System* penjualan LBC Universitas Sahid Surakarta.

8. Pengujian *System*

   Pada tahap akhir *System* telah siap digunakan di LBC Universitas Sahid Surakarta dan membuat dokumentasi dari keseluruhan penelitian tugas akhir ini.

1.9. Landasan Teori

2.3.1. *System*

Al Fatta (2007 : 3) memberikan beberapa definisi *System* secara umum :

1. Kumpulan dari bagian-bagian yang bekerja sama untuk mencapai tujuan yang sama.
2. Sekumpulan objek-objek yang saling berelasi dan berinteraksi serta hubungan antar objek bisa di lihat sebagai satu kesatuan yang dirancang untuk mencapai satu tujuan.

Secara sederhana *System* dapat diartikan sebagai suatu kumpulan atau himpunan dari unsur atau variabel-variabel yang saling terorganisasi, saling berinteraksi, dan saling bergantung sama lain (Al Fatta, 2007 : 3).

2.3.2. Penjualan

"Penjualan artinya penjualan barang dagangan sebagai usaha pokok perusahaan yang biasanya dilakukan secara teratur" (Marom, 2002:28).

2.3.3. *Website*

Website merupakan kumpulan dari halaman *web* yang sudah dipublikasikan di jaringan internet dan memiliki domain/URL yang dapat diakses semua pengguna internet dengan cara mengetikkan alamatnya (Rudyanto Arief, 2011:8).

1.10. *Software* Pendukung

2.4.1. Dreamweaver

Dreamweaver merupakan salah satu program aplikasi yang digunakan untuk membangun sebuah *website*, baik secara grafis maupun dengan menulis kode sumber secara langsung. "Adobe dreamweaver merupakan program untuk membuat atau mengedit *web* yang dikeluarkan oleh Adobe System yang juga dikenal sebagai Macromedia Dreamweaver. *Software* ini digunakan karena memiliki fitur-fitur yang menarik dan cenderung mudah dalam penggunaanya (Wahana Komputer, 2011).

2.4.2. PHP

PHP (PHP: *Hypertext Preprocessor*) adalah bahasa *server-side scripting* yang menyatu dengan HTML untuk membuat halaman *web* yang dinamis. Karena PHP merupakan *server-side scripting* maka *sintaks* dan perintah-perintah PHP akan

dieksekusi di server kemudian hasilnya dikirimkan ke *browser* dalam *form*at HTML. Dengan demikian kode program yang ditulis dalam PHP tidak akan terlihat oleh *user* sehingga keamanan *web* lebih terjamin. PHP dirancang untuk membentuk suatu tampilan berdasarkan permintaan terkini, seperti menampilkan isi basis data ke halaman *web* (Rudyanto Arief, 2011 : 43).

2.4.3. XAMPP *Control Panel*

XAMPP adalah sebuah *software* yang berfungsi untuk menjalankan *website* berbasis PHP dan menggunakan pengolah data MYSQL di komputer lokal". XAMPP berperan sebagai *server web* pada komputer lokal. XAMPP juga dapat disebut sebuah *Cpanel server virtual*, yang dapat membantu melakukan *preview* sehingga dapat dimodifikasi *website* tanpa harus *online* atau terakses dengan internet (Wicaksono, 2008 : 7).

2.4.4. MySql

MySQL adalah salah satu jenis *database server* yang sangat terkenal dan banyak digunakan untuk membangun aplikasi *web* yang menggunakan *database* sebagai sumber dan pengelolaan datanya. Kepopuleran MySQL antara lain karena MySQL menggunakan SQL sebagai bahasa dasar untuk mengakses *database*-nya sehingga mudah untuk digunakan, kinerja *query* cepat, dan mencukupi untuk kebutuhan *database* perusahaan-perusahaan skala menengah-kecil. MySQL juga bersifat *open source* dan *free* (Anda tidak perlu membayar untuk menggunakannya) pada berbagai *platform* (kecuali pada Windows, yang bersifat *shareware*). MySQL didistribusikan dengan lisensi *open source* GPL (*General Public Licence*) mulai versi 3.23, pada bulan juni 2000. *Software* MySQL bisa diunduh di http://www.mysql.org atau http://www.mysql.com (Rudyanto Arief, 2011 : 151).

2.4.5. Adobe Photoshop

Adobe Photoshop adalah perangkat lunak editor citra buatan Adobe System yang dikhususkan untuk pengeditan foto atau gambar dan pembuatan efek. Perangkat ini banyak digunakan oleh photografer digital dan perusahaan iklan sehingga dianggap sebagai pemimpin pasar (*Market Leader*) untuk perangkat

lunak pengolahan gambar atau foto dan bersama Adobe Acrobat dianggap sebagai produk terbaik yang pernah diproduksi oleh Adobe System (Simartata, 2010).

1.11. Analisis *System*

Analsis *System* adalah sebuah istilah yang secara kolektif mendeskripsikan fase-fase awal pengembangan *System*. Analisis *System* adalah teknik pemecahan masalah yang menguraikan bagian-bagian komponen dengan mempelajari seberapa bagus bagian-bagian komponen tersebut bekerja dan berinteraksi untuk mencapai tujuan mereka. Analisis *System* merpakan tahapan paling awal dari pengembangan *System* menjadi fondasi menentukan keberhasilan *System* in*form*asi yang dihasilkan nantinya. Tahapan ini sangat penting karena menentukan bentuk *System* yang harus dibangun. Tahapan ini bisa merupakan tahap yang mudah jika klien sangat paham dengan masalah yang dihadapi dalam organisasinya dan tahu betul fungsionalitas dari *System* informasi yang akan dibuat. Tetapi tahap ini bisa menjadi tahap yang paling sulit jika klien tidak bisa mengidentifikasi kebutuhannya atatu tertutup terhadap pihak luar yang ingin mengetahui detail proses-prose bisnisnya (Al Fatta, 2007 : 44).

1.12. Perancangan *System*

Perancangan *System* adalah merancang atau mendesain suatu *System* yang baik, yang isinya adalah langkah-langkah operasi dalam proses pengolahan data dan prosedur untuk mendukung operasi *System* (Bin Ladjamuddin, 2005).

1.13. Metode Perancangan *System* Berorientasi Objek

Metode Berorientasi Objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya (Nugroho, 2005).

Sebuah *System* yang dibangun dengan berdasarkan metode berorientasi objek adalah sebuah *System* yang komponennya dibungkus (*dienkapsulasi*) menjadi kelompok data dan fungsi. Setiap komponen dalam *System* tersebut dapat mewarisi atribut dan sifat dan komponen lainnya serta dapat berinteraksi satu sama lainnya (Rachman, dkk, 2012 : 2).

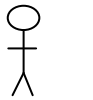
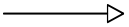
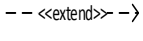
Metode perancangan metodologi berorientasi objek menggunakan diagram UML (*Unified Modeling Languange*) adalah sebuah “bahasa” yang telah menjadi

standart dalam industri untuk visualisasi, merancang dan mendokumentasikan *System* perangkat lunak (Nugroho, 2002).

2.7.1. *Use Case Diagram*

Use case diagram mengambarkan sejumlah *external actors* dan hubungannya ke *use case* yang diberikan oleh *System*. Use case adalah deskripsi fungsi yang disediakan oleh *System* dalam bentuk teks sebagai dokumentasi dari *use case symbol* namun dapat juga dilakukan dalam *activity diagrams*. Use case digambarkan hanya dilihat dari luar oleh *actor* (keadaan lingkungan *System* yang dilihat user) dan bukan bagaimana fungsi yang ada dalam *System* (Kusumo, 2004:3). Simbol-Simbol yang digunakan pada *Use Case Diagram* ditunjukkan pada Tabel 2.1.

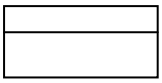
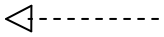
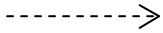
Tabel 2.1. Tabel Simbol *Use Case Diagram*

NO	GAMBAR	NAMA	KETERANGAN
1.		*Aktor*	Idealization orang eksternal, proses, atau hal yang berinteraksi dengan *System*,sub*System* atau kelas.
2.		*Use case*	Sebuah *use case* menggambar-kan interaksi dengan*actor* sebagai urutan pes an antara*System* dan aktor satu atau lebih.
3.		*System Boundary*	Menspesifikasikan paket yang menampilkan *System* secara terbatas.
4.		*Generalizatio n*	Hubungan antara *use case* umum dan *use case* yang lebih spesifik yang mewarisi dan menambahkan fitur.
5.		*Comunicatio n Association*	Jalur komunikasi antara *actor* dan *use case* yang berpartisipasi didalam.
6.		*Extend*	Penyisipan perilaku tambahan kedalam basis *use case* yang tidak tahu tentang hal itu.
7.		*include*	Penyisipan perilaku tambahan kedalam basis *use case* yang secara eksplisit menggambarkan penyisipan.

2.7.2. *Class Diagram*

Class diagram menggambarkan struktur statis *class* di dalam *System*. *Class* merepresentasikan sesuatu yang ditangani oleh *System*. *Class* dapat berhubungan dengan yang lainmelalui berbagai cara: *associated* (terhubung satu sama lain), *dependent* (satu *class* tergantung/menggunakan *class* yang lain), *specialed* (satu *class* merupakan spesialisasi dari *class* lainya), atau *package* (grup bersama sebagai satu unit). Sebuah *System* biasanya mempunyai beberapa *class diagram* (Kusumo, 2004:3). Simbol-Simbol yang digunakan pada *Class Diagram* ditunjukkan pada Tabel 2.2.

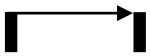
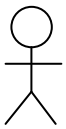
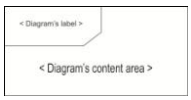
Tabel 2.2. Tabel Simbol *Class Diagram*

NO	GAMBAR	NAMA	KETERANGAN
1.		*Generalization*	Hubungan dimana objek anak (*descendent*) berbagai perilaku dan struktur data dari objek yang ada diatas objek induk (*ancestor*).
2.		*Nary Association*	Upaya untuk menghindari asosiasi dengan lebih dari dua objek.
3.		*Class*	Himpunan dari objek–objek yang terbagi atribut serta operasi yang sama.
4.		*Collaboration*	Deskripsi dari urutan aksi-aksi yang ditampilkan *System* yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
5.		*Realization*	Operasi yang benar-benar dilakukan oleh suatu objek.
6.		*Dependency*	Hubungan yang terjadi pada suatu elemen mandiri (*independent*) akan mempengaruhi elemen yang bergantung pada elemen yan tidak mandiri.
7.		*Association*	Untuk menghubungkan objek satu dengan objek yang lainnya.

2.7.3. *Sequence Diagram*

Sequence diagram menggambarkan kolaborasi dinamis antara sejumlah objek. Kegunaan untuk menunjukan rangkaian pesan yang dikirim Antara objek juga interaksi antara objek, sesuatu yang terjadi pada titik tertentu dalam eksekusi *System* (Kusumo, 2004:4). Simbol-Simbol yang digunakan pada *Sequence Diagram* ditunjukkan pada Tabel 2.3.

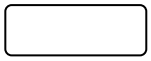
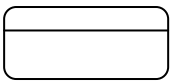
Tabel 2.3. Tabel Simbol *Sequence Diagram*

NO	GAMBAR	NAMA	KETERANGAN
1.		*LifeLine*	Objek *entity*, antarmuka yang saling berinteraksi.
2.		*Message*	*Message* ditampilkan sebagai anak panah dari lifeline dari satu objek ke objek yang lain.
3.		*Actor*	Pengguna di luar system.
4.		*Fragment*	Menggambarkan batas grafis suatu diagram.

2.7.4. *Statechart Diagram*

Statechart diagram menggambarkan semua *state* (kondisi) yang dimiliki oleh suatu *object* dari suatu *class* dan keadaan yang menyebabkan *state* berubah. Kejadian dapat berupa *object* lain yang mengirim pesan. *State class* tidak digambarkan untuk semua *class*, hanya yang mempunyai sejumlah *state* yang terdefini dengan baik dan kondisi *class* berubah oleh *state* yang berbeda (Kusumo, 2004:4). Simbol-Simbol yang digunakan pada *Statechart Diagram* ditunjukkan pada Tabel 2.4.

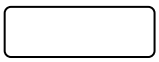
Tabel 2.4. Tabel Simbol *Statechart Diagram*

NO	GAMBAR	NAMA	KETERANGAN
1.		*State*	Meliputi seluruh pesan dari *object* yang dapat mengirim dan menerima.
2.		*Start state*	Masing-masing diagram harus mempunyai satu dan hanya satu *start state*
3.		*Stop State*	Sebuah *object* boleh mempunyai banyak *stop state*
5.	*class name.sent event*	*State Transition*	Sebuah kejadian yang memicu sebuah *state object* dengan cara memperbarui satu atau lebih nilai atributnya.
6.		*State Detail*	*Action-action* yang mengiringi seluruh *state transition* ke sebuah *state*.

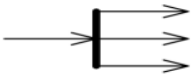
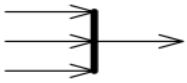
2.7.5. *Activity Diagram*

Activity diagram menggambarkan rangkaian aliran aktifitas, digunakan untuk mendeskripsikan aktifitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktifitas lainnya seperti *use case* atau interaksi (Kusumo, 2004:4). Simbol-Simbol yang digunakan pada *Activity Diagram* ditunjukkan pada Tabel 2.5.

Tabel 2.5. Tabel Simbol *Activity Diagram*

NO	GAMBAR	NAMA	KETERANGAN
1.		*Activity State*	Aktivitas yang mewakili pelaksanaan dalam pernyataan dalam prosedur atau pelaksanaan kegiatan dalam alur kerja.
2.		*Branch/Merge*	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.

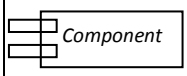
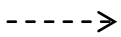
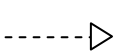
Lanjutan Tabel 2.5. Tabel Simbol *Activity Diagram*

3.		*Initial State*	Status awal aktivitas *System*, sebuah diagram aktivitas memiliki sebuah status awal.
4.		*Final State*	Status akhir yang dilakukan *System*.
5.		*Fork Node*	Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran.
6.		*Join Node*	Beberapa aliran masukan tertentu berubah menjadi satu aliran.

2.7.6. *Component Diagram*

Component diagram menggambarkan struktur fisik kode dari komponen. Komponen dapat berupa *sourcecode*, komponent biner, atau *executable component*. Sebuah komponen berisi in*form*asi tentang *logic clas*s atau *class* yang diimplementasikan sehingga membuat pemetaan dari *logical view* ke *component view* (Kusumo, 2004:4). Simbol-Simbol yang digunakan pada *Component Diagram* ditunjukkan pada Tabel 2.6.

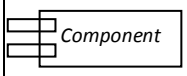
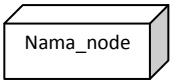
Tabel 2.6. Tabel Simbol *Component Diagram*

NO	GAMBAR	NAMA	KETERANGAN
1.		*Component*	Sebuah komponen melambangkan sebuah entitas *software* dalam sebuah *System*. Sebuah komponen dinotasikan sebagai sebuah kotak segiempat dengan dua kotak kecil tambahan yang menempel disebelah kirinya.
2.		*Interface*	In*form*asi yang menjadi ciri dari perilaku.
3.		*Usage*	Situasi di mana satu elemen membutuhkan lain untuk fungsi yang benar.
4.		*Realization*	Hubungan antara spesifikasi dan implementasinya.

2.7.7. *Deployment Diagram*

Deployment Diagram menggambarkan arsitektur fisik dari perangkat keras dan perangkat lunak *System*, menunjukkan hubungan komputer dengan perangkat (*nodes*) satu sama lain dan jenis hubungannya. Di dalam *nodes*, *executeable component* dan *object* yang dialokasikan untuk memperlihatkan unit perangkat lunak yang dieksekusi oleh *node* tertentu dan ketergantungan komponen (Kusumo, 2004:4). Simbol-Simbol yang digunakan pada *Deployment Diagram* ditunjukkan pada Tabel 2.7.

Tabel 2.7. Tabel Simbol *Deployment Diagram*

NO	GAMBAR	NAMA	KETERANGAN
1.		*Component*	Komponen-komponen yang ada diletak-kan didalam *node*.
2.		*Node*	*Node* menggambarkam bagian-bagian *hardware* dalam sebuah *System*.
3.		*Association*	Sebuah *association* digambarkan sebagai sebuah garis yang menghubungkan dua *node* yang mengindikasikan jalur komunikasi antara element-elemen *hardware*.

1.14. Metode Pengujian *System*

Beberapa *test-case* harus dilaksanakan dengan beberapa perbedaan strategi transaksi, *query*, atau jalur navigasi yang mewakili penggunaan *System* yang tipikal, kritis, atau *abnormal*. Isu kunci dalam pengembangan *System* adalah pemilihan sekelompok *test-case* yang cocok, sekecil, dan secepat mungkin, untuk meyakinkan perilaku *System* secara detail. Pengujian harus menyangkut unit *testing*, yang mengecek *validasi* dari prosedur dan fungsi-fungsi secara *independent* dari komponen *System* yang lain.

Kemudian modul *testing* harus menyusul dilakukan untuk mengetahui apakah penggabungan beberapa unit dalam satu modul sudah berjalan dengan baik, termasuk eksekusi modul yang saling berelasi, apakah sudah berjalan sesuai karakteristik *System* yang diinginkan (Al Fatta, 2007 : 170-171).

Berikut ringkasan beberapa kategori *test* yang bisa dilakukan menurut (Al Fatta, 2007 : 170-174).

2.8.1. *Sub Testing*

Sub testing adalah pengujian yang difokuskan pada pengujian struktur kendali sebelum semua modul dituliskan. *System* perangkat lunak secara umum terdiri dari modul yang berelasi, baik secara *hierarki* maupun relasional.

2.8.2. Unit *Testing*

Jika struktur antar modul sudah terbukti bagus, maka pengujian yang tak kalah pentingnya adalah pengujian unit. Pengujian unit digunakan untuk menguji setiap modul untuk menjamin setiap modul menjalankan fungsinya dengan baik.

Metode untuk melakukan unit *testing*, yaitu :

2.8.1.1. *Black Box Testing*

Terfokus pada apakah unit program memenuhi kebutuhan (*requirement*) yang disebutkan dalam spesifikasi. Pada *black box testing*, cara pengujian hanya dilakukan dengan menjalankan atau mengeksekusi unit atau modul, kemudian diamati apakah hasil dari unit itu sesuai dengan proses bisnis yang diinginkan.

2.8.3. Pengujian *System*

1. Melakukan proses evaluasi terhadap *system* yang sudah ada apakah *system* sudah sesuai yang diharapkan user.
2. Menilai dan mengevaluasi terhadap *output*.
3. Menguji terhadap *input*, pengelolaan (proses) dan *output system*.

Melakukan penilaian dan evaluasi terhadap komponen *system* prosedur pelaksanaan kegiatan dan mutu atau kualitas hasil *system.*