BAB II

LANDASAN TEORI

2.1 Java

Menurut Rickyanto (2003). Java adalah suatu teknologi di dunia *software* komputer. Selain merupakan suatu bahasa pemrograman, Java juga merupakan suatu *platform*. Java merupakan teknologi dimana teknologi tersebut mencakup java sebagai bahasa pemrograman yang memiliki sintaks dan aturan pemrograman sendiri. Juga mencakup java sebagai *platform* dimana teknologi ini memiliki *virtual machine* dan *library* yang diperlukan untuk menulis dan menjalankan program yang ditulis dengan bahasa pemrograman Java.

Java merupakan suatu teknologi yang unik dan revolusioner dan merupakan teknologi pertama di dunia *software* yang memiliki semboyan "*write once, run anywhere*". Semboyan tersebut telah terbukti karena banyak program java dapat dijalankan diberbagai *platform* sistem operasi seperti Linux, Windows maupun Unix.

Adapun karakteristik Java menurut Rickyanto (2003) adalah:

a. Sederhana: Java tidak memiliki sintaks yang aneh tetapi banyak menggunakan sintaks c++ yang sudah banyak dikenal, sehingga java tidak menyulitkan bagi para programmer. Bahkan java memberikan banyak keunggulan dan kemudahan dibanding C++
b. Berorientasi objek: Java merupakan pemrograman berorientasi objek yang murni. Dalam pemrograman Java semua adalah objek, terkecuali tipe primitif

c. Dapat didistribusikan Dengan mudah: Sifat distribusi dari Java sangat tampak sebagai *applet* dan *library* yang mampu bekerja dalam jaringan dan bekerja dengan objek terdistribusi (RMI) dengan sangat baik.

d. Aman: Program Java memiliki *library security* serta *policy* yang membatasi akses *applet* di komputer *client*.

e. Diinterpretasi oleh interpreter: Java memerlukan *virtual machine* yang bertindak sebagai interpreter yang menterjemahkan bytecode (file class) menjadi bahasa mesin yang dimengerti oleh komputer *host.*

f. Portabel: Java dapat dijalankan diberbagai *platform* tanpa perubahan kode.

g. Multi *threading*: Java memiliki banyak kemampuan untuk menangani dan menjalankan banyak *thread* sekaligus.

h. Dinamis: Java merupakan teknologi yang terus berkembang dan hal ini tampak nyata sekali dengan *library* yang terus ditingkatkan kemampuan dan kelengkapannya.

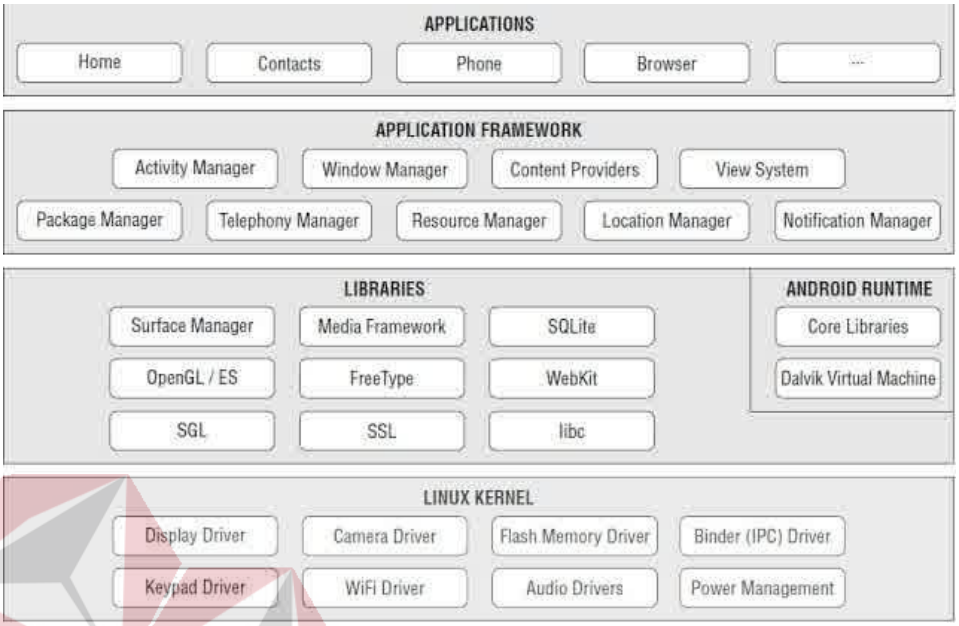
i. Netral terhadap arsitektur *hardware*: Java dapat dijalankan dengan baik pada komputer yang memiliki arsitektur berbeda-beda.

j. Robust: Java merupakan teknologi yang mampu menolong programmer untuk menghasilkan program secara cepat dan handal karena Java mencegah adanya memori *leaking*, meniadakan *pointer* serta mencegah berbagai eror yang mungkin terjadi dengan adanya proses pengecekan awal pada kompilasi.

2.2 Android

Menurut Suprianto (2012), Android adalah sistem operasi bergerak (*mobile operating system*) yang mengadopsi sistem operasi Linux, namun telah

dimodifikasi. Android diambil alih oleh Google pada tahun 2005 dari Android, Inc.



Gambar 2.1 Arsitektur Sistem Operasi Android

Secara garis besar sistem operasi Android terbagi menjadi lima tingkatan:

a. Linux kernel: Adalah kernel dasar Android. Tingkat ini berisi semua *driver* perangkat tingkat rendah untuk komponen *hardware* perangkat Android

b. *Libraries*: Berisi semua kode program yang menyediakan layanan-layanan utama sistem operasi Android.

c. Android *Runtime*: Kedudukannya setingkat dengan *libraries.* Android Runtime menyediakan kumpulan pustaka inti yang dapat diaktifkan oleh pengembang untuk menulis kode aplikasi Android dengan bahasa pemrograman Java.

d. *Application Framework*: Adalah semacam kumpulan *class built-in* yang tertanam dalam sistem operasi Android, sehingga pengembang dapat memanfaatkannya untuk aplikasi yang sedang dibangun.

e. *Applications*: Pada tingkat inilah kita akan bekerja. Seperti aplikasi Android pada umumnya yang dapat di-*download* dan di-*instal* dari *market* Android.

2.3 Aplikasi Mobile

Menurut Buyens (2001) aplikasi mobile berasal dari kata *application* dan *mobile*. *Application* yang artinya penerapan, lamaran, penggunaan. Secara istilah aplikasi adalah program siap pakai yang direka untuk melaksanakan suatu fungsi bagi pengguna atau aplikasi yang lain dan dapat digunakan oleh sasaran yang dituju, sedangkan *mobile* dapat diartikan sebagai perpindahan dari suatu tempat ke tempat yang lain.

Maka aplikasi *mobile* dapat diartikan sebuah program aplikasi yang dapat dijalankan atau digunakan walaupun pengguna berpindah-pindah dari satu tempat ke tempat yang lain serta mempunyai ukuran yang kecil. Aplikasi *mobile* ini dapat diakses melalui perangkat nirkabel, pager, PDA, telepon seluler, *smartphone*, dan perangkat sejenisnya. Perangkat *mobile* memiliki banyak jenis dalam hal ukuran, desain dan layout, tetapi memiliki karakteristik yang sangat berbeda dari sistem *desktop*. Berikut karakteristik perangkat *mobile*, diantaranya:

a. Ukuran yang kecil: Perangkat *mobile* memiliki ukuran yang kecil. Konsumen menginginkan perangkat yang terkecil untuk kenyamanan dan mobilitas mereka.
b. Memory yang terbatas: Perangkat *mobile* juga memiliki memori yang kecil, yaitu *primary (RAM)* dan *secondary (disk)*.
c. Daya proses yang terbatas: Sistem *mobile* tidaklah setangguh rekan mereka yaitu *desktop.*

d. Mengkonsumsi daya yang rendah: Perangkat *mobile* menghabiskan sedikit daya dibandingkan dengan mesin *desktop*
e. Kuat dan dapat diandalkan: Karena perangkat *mobile* selalu dibawa kemana saja, mereka harus cukup kuat untuk menghadapi benturan-benturan, gerakan, dan sesekali tetesan-tetesan air.
f. Konektivitas yang terbatas: Perangkat *mobile* memiliki *bandwith* rendah, beberapa dari mereka bahkan tidak tersambung.

2.4 Smartphone

Menurut Williams & Sawyer (2007), *Smartphone* adalah telepon selular dengan mikro prosesor, memori, layar dan modem bawaan. *Smartphone* merupakan ponsel multimedia yang menggabungkan fungsionalitas PC dan *handset* sehingga menghasilkan *gadget* yang mewah, dimana terdapat pesan teks, kamera, pemutar musik, video, game, akses email, TV digital, *search engine*, pengelola informasi pribadi, fitur GPS, jasa telepon internet dan bahkan terdapat telepon yang juga berfungsi sebagai kartu kredit.

2.5 Tablet PC

Komputer tablet adalah komputer yang punya layar minimal 4,8 inchi dan memiliki konektifitas Wi-Fi. Selain itu karena tab-tab punya layanan 3G.maka juga bisa dianggap sebagai sebuah *smartphone*. Komputer tablet android biasanya memiliki *hard drive* asli untuk data *storage*. Beberapa merek awal yang muncul contohnya adalah Dell Streak dan Galaxy Tab dengan ukuran 7 inchi dan 10 inchi (Winarno, 2012).

2.6 Android SDK (Software Development Kit)

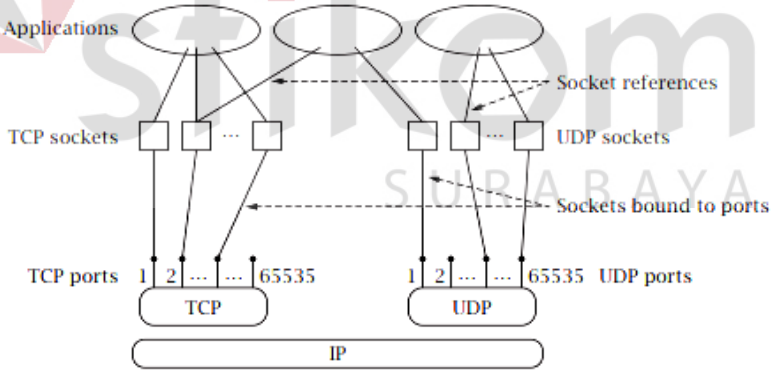
Menurut Suprianto (2012) SDK Android berisi *debugger*, *library*, *emulator*, dokumentasi, contoh kode program dan tutorial. SDK Android adalah mesin utama untuk mengembangkan aplikasi Android. Sedangkan menurut Safaat (2012) Android SDK adalah tools API (*Application Programming Interface*) yang diperlukan untuk mulai mengembangkan aplikasi pada *platform* Android menggunakan bahasa pemrograman Java. Sebagai *platform* aplikasi netral, Android memberi kesempatan untuk membuat aplikasi yang dibutuhkan yang bukan merupakan aplikasi bawaan *handphone/ smartphone*.

Cara kerjanya adalah Android SDK memberi *librar*y yang nantinya dapat digunakan oleh pembuat aplikasi untuk membuat applikasi Android. Aplikasi tersebut dapat di-*debug* dengan *debugger* yang sudah disediakan oleh SDK. Kemudian aplikasi tersebut dijalankan dengan menggunakan *emulator* yang juga sudah disediakan oleh Android SDK

2.7 Socket

Menurut Kenneth (2002) Socket adalah suatu abtraksi yang mana aplikasi dapat mengirim dan menerima data seperti sama halnya dengan membuka suatu *file* untuk dibaca dan ditulis pada tempat penyimpanan *file*. Socket memungkinkan aplikasi untuk masuk kedalam jaringan dan berkomunikasi dengan aplikasi lain yang juga masuk kedalam jaringan yang sama. Informasi yang ditulis kedalam socket pada suatu aplikasi pada suatu mesin dapat dibaca oleh aplikasi lain pada mesin yang berbeda dan sebaliknya.

Berbagai jenis socket berhubungan dengan berbagai rangkaian protokol yang mendasari dan berbagai susunan protokol dalam sebuah rangkaian. Salah satunya adalah rangkaian protokol TCP/IP. Jenis utama dari socket dalam TCP/IP adalah stream socket dan datagram socket. Stream socket menggunakan TCP sebagai protokol ujung ke ujung (dengan menggunakan IP sebagai dasarnya) dan memberikan jasa byte-stream yang reliable (terpercaya). Datagram socket menggunakan UDP (sekali lagi dengan IP sebagai dasarnya) dan memberikan datagram service paling mudah yang dapat digunakan oleh aplikasi untuk mengirim pesan individu sampa dengan panjang 65.500 byte. Sebuah TCP/IP socket diidentifikasi secara unik dengan *Internet Address* (alamat internet / IP), protokol ujung ke ujung (TCP/UDP) dan sebuah nomor port. Gambar 2.2 mengambarkan hubungan logika antara aplikasi, abstraksi socket protokol dan nomor port dalam sebuah host



Gambar 2.2 Socket, Protocol dan Port

2.8 Sistem Pelayanan Restoran

Menurut Marsum (2005), Restoran adalah suatu tempat atau bangunan yang diorganisasi secara komersial yang menyelenggarakan pelayanan yang baik kepada semua tamunya baik berupa makan maupun minum.

Tujuan operasi restoran adalah untuk mencari untung. Selain bertujuan bisnis atau mencari untung, membuat puas para tamu pun merupakan tujuan operasi restoran yang utama. Urutan-urutan kerja di restoran waktu buka/ operasi adalah:

a. Menyambut dan mengucapkan salam: Memberi ucapan salam kepada tamu atau lebih lazim disebut *greeting's* kepada tamu waktu masuk restoran.
b. Mendudukkan tamu: Tamu kita antarkan ke tempat duduk. Penerima tamu harus tahu pasti dimana atau meja-meja nomor berapa yang masih kosong atau belum dipesan tamu dan berapa kapasitas masing-masing.
c. Memberikan daftar minuman (waktu makan siang atau malam): Yang dimaksud daftar minuman disini ialah minuman yang diminum tamu sebelum memulai makan dengan tujuan untuk membangkitkan nafsu makan.
d. Memberi daftar makanan: Penerima tamu juga dituntut untuk memahami benar-benar makanan/produk yang dijual sehingga dapat menyarankan kepada tamu dengan baik dan penjualan dapat meningkat.
e. Menuangkan air es: Untuk memberikan kesempatan kepada tamu untuk mempelajari daftar makanan atau menu yang telah diberikan, *Waiter* lalu menuangkan air es ke gelas tamu.
f. Mengambil pesanan tamu: Dalam restoran kecil, pesanan tamu diambil-dicatat dengan atau dalam memo, kemudian dipindahkan ke sebuah *check/bill.* Di dalam restoran yang lebih besar, pesanan diambil dan ditulis di atas sebuah *Kitchen Order Tiket* (KOT), dibuat rangkap sejumlah *system control* restoran itu.

g. Menuliskan pesanan tamu: Kadang-kadang ada restoran yang memakai sistem lain dalam menuliskan *order*/pesanan tamu. Ada juga yang pesanan tamu langsung disalin dari memo kedalam cek/ *bill* oleh *captain.* Cek sudah *carbonized*, dibuat rangkap tiga atau empat.

h. Periksalah kebersihan dan kondisi piring: Yang cacat, retak atau gempil harus disingkirkan. Sedangkan yang kurang bersih/flek dikembalikan lagi tempat pencucian piring.

i. Periksa makanan pesanan tamu: Setiap pesanan tamu yang kita terima dari dapur, sebelum masuk restoran perlu diperiksa terlebih dahulu.

j. Periksalah makanan penghias (*Garnir*): Setiap makanan yang dipesan tamu pasti diberi makanan penghias, pelengkap dan penyerta/ *garner.*

k. Menghidangkan makanan pada tamu: Untuk menghidangkan makanan pada tamu lazimdipergunakan bagi persegi panjang.

l. Memberikan *bill*/cek: *Bill* atau cek dapat diberikan kepada tamu beberapa saat setelah tamu selesai makan makanan penutup serta meminum teh/kopinya. Atau kadang-kadang kita tunggu sesaat sampai tamu meminta *bill*nya.

2.9 Sistem Basis Data

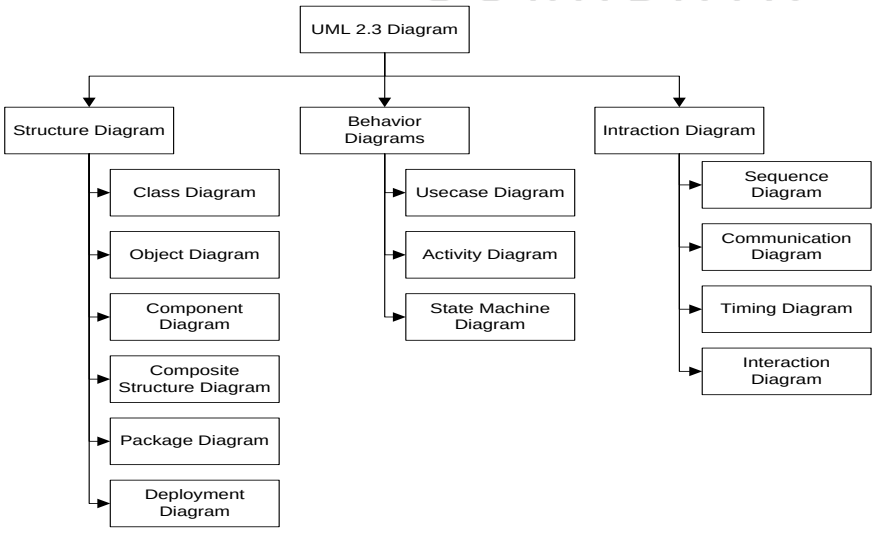
Menurut Marlinda (2004:1), sistem basis data adalah suatu sistem menyusun dan mengelola *record-record* menggunakan komputer untuk menyimpan atau merekam serta memelihara operasional lengkap sebuah organisasi/perusahaan sehingga mampu menyediakan informasi optimal yang diperlukan pemakai untuk proses mengambil keputusan.

2.10 UML (Unified Modeling Language)

Menurut Fowler (2005). Unified Modeling Language (UML) adalah keluarga notasi grafis yang didukung oleh meta-model tunggal, yang membantu pendeskripsian dan desain sistem perangkat lunak, khususnya sistem yang dibangun pemrograman berorientasi objek (OO)

Menurut Rosa (2011) pada perkembangan teknik pemrograman berorientasi objek, muncullah sebuah standarisasi bahasa pemodelan untuk membangun perangkat lunak yang dibangun dengan menggunakan teknik pemrograman berorientasi objek yaitu *Unified Modeling Language* (UML). UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun dan dokumentasi dari sistem perangkat lunak. UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung.

Pada gambar UML terdiri dari 13 macam diagram dikelompokkan dalam 3 kategori. Pembagian kategori dan macam-macam diagram tersebut dapat dilihat pada gambar 2.3.



Gambar 2.3 Diagram UML

Berikut ini penjelasan singkat dari pembagian kategori tersebut.

a. *Structur* diagram yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan

b. *Behavior* Diagram yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sistem

c. *Interaction* diagram yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antar subsistem pada suatu sistem

Dalam pembuatan Tugas Akhir ini, menggunakan beberapa jenis diagram yaitu: *use Case Diagram*, *Activity Diagram*, *Sequance Diagram*, *Class Diagram*, *Componen Diagram*, *Deployment Diagram* dan *Statechard Diagram*

2.10.1 Usecase Diagram

Usecase diagram menyajikan interaksi antar *usecase* dan *actor*. Dimana *actor* dapat berupa orang, peralatan atau sistem lain yang berinteraksi dengan sistem yang sedang dibagun. *Usecase* menggambarkan fungsionalitas sistem atau persyaratan-persyaratan yang harus dipenuhi sistem dari pandangan pemakai. Sholiq (2006),

2.10.2 Activity Diagram

Menggambarkan aliran fungsionalitas sistem. Pada tahap pemodelan bisnis,diagram aktivitas dapat digunakan untuk menunjukkan aliran kerja bisnis

(*business work flow*). Dapat juga digunakan untuk menggambarkan aliran kejadian (*flow of event*) dalam *use case*. Sholiq (2006).

2.10.3. Sequance Diagram

Digunakan untuk menunjukkan aliran fungsionalitas dalam *use case* yang disusun berdasarkan urutan waktu. Alur pembacaan diagram ini adalah dari atas ke bawah.Diagram ini dapat dibaca dengan memperhatikan objek-objek dan pesan-pesan yang ada pada diagram. Sholiq (2006).

2.10.4. Class Diagram

Menunjukkan interaksi anatar kelas dalam sistem. Kelas mengandung informasi dan tingkah laku (*behavior)* yang berkaitan dengan informasi tersebut. Sebuah kelas pada diagram kelas dibuat untuk setiap tipe obyek pada diagram sequensial atau diagram kolaborasi. Sholiq (2006).

2.10.5. Componen Diagram

Menunjukkan model secara fisik komponen perangkat lunak pada sistem dan hubungannya antar mereka. Diagram komponen digunakan oleh siapapun yang bertanggung jawab untuk melakukankompilasi sistem. Diagram ini juga menunjukkan komponen apa yang dibutuhkan saat proses kompilasi dan menampilkan komponen *run-time* apa saja yang dibuat sebagai hasil dari proses kompilasi. Sholiq (2006).

2.10.6. Deployment Diagram

Menampilkan rancangan fisik jaringan dimana berbagai komponen akan terdapat disana. Diagram *deployment* digunakan oleh manajer proyek, arsitek sistem dan karyawan distribusi untuk memahami rancangan fisik sistem dan dimana saja terdapat subsistem yang akan dibuat. Diagram ini membantu manajer proyek mengkomunikasikan tentang apa yang sistem inginkan terhadap pemakai, juga membantu bagian pengembangan untuk merencanakan distribusi yang akan ditawarkan. Sholiq (2006).

2.10.7. Statechart Diagram

Diagram Statechart diagram atau Statechart diagram menyediakan sebuah cara untuk memodelkan bermacam-macam keadaan yang mungkin dialami oleh sebuah objek. Jika dalam diagram kelas menunjukkan gambaran statis kelas-kelas dan relasinya,diagram statechart digunakan untuk memodelkan tingkah laku dinamik sistem. Sholiq (2006).

2.11 Sytem Development Life Cycle (SDLC)

System Development Life Cycle (SDLC) adalah pendekatan bertahap untuk melakukan analisa dan membangun rancangan sistem dengan menggunakan siklus yang spesifik terhadap kegiatan pengguna. Pengembangan sistem informasi dapat dilakukan dengan berbagai metode. Proses pengembangan sistem yang dimulai dari tahap perencanaan sampai implementasinya disebut dengan istilah *System Development Life Cycle* (SDLC). Tahapan-tahapan SDLC menurut Kendall (2003) adalah sebagai berikut:

a. Mengidentifikasi masalah, peluang dan tujuan.

Menentukan permasalahan-permasalahan apa yang terjadi dan apa yang menyebabkan sasaran pada sistem lama belum tercapai. Kemudian mengidentifikasi peluang pengembangan sistem termasuk fisibilitas secara teknis, ekonomis dan operasional bahwa peningkatan dapat dilakukan melalui penggunaan sistem informasi terkomputerisasi, selanjutnya pada tahap ini juga dilakukan identifikasi tujuan dari pengembangan sistem informasi.

b. Menentukan kebutuhan informasi

Memahami informasi apa yang dibutuhkan pemakai agar bisa ditampilkan dalam pekerjaan. Serta mengetahui detil fungsi-fungsi dalam sistem termasuk mengetahui siapa saja yang terlibat, kegiatan apa saja yang ada, lingkungan kerja yang mana, waktu yang diperlukan serta bagaimana mekanisme atau prosedur yang berlaku.

c. Menganalisis kebutuhan sistem.

Dilakukan penguraian suatu sistem informasi yang utuh ke dalam komponen-komponennya untuk mengidentifikasi dan mengevaluasi permasalahan-permasalahan, peluang-peluang, maupun hambatan-hambatan yang terjadi dan kebutuhan yang diharapkan sehingga dapat diusulkan perbaikan-perbaikannya.

d. Merancang sistem yang direkomendasikan.

Memberikan gambaran yang jelas dari rancang bangun yang lengkap. Terdapat dua bagian dalam perancangan sistem, yaitu rancangan sistem secara umum atau desain makro dan rancangan sistem secara terinci atau rancangan fisik. Kegiatan yang dilakukan dalam tahap ini meliputi:

1. Desain model dari sistem informasi yang akan dikembangkan.
2. Desain *output* adalah keluaran dari sistem informasi yang dapat dilihat, dapat berupa tampilan dilayar, kertas laporan dan sebagainya.
3. Desain *input* yang perlu didesain secara rinci dari *input* adalah bentuk dari dokumen dasar yang digunakan dan bentuk tampilan dari *input* di alat *input*.
4. Desain basis data ini adalah mengintegrasikan kumpulan dari data yang saling berhubungan antara satu dengan lainnya dan membuatnya tersedia untuk aplikasi yang bermacam-macam di dalam suatu organisasi, yang terdiri dari beberapa file yang diperlukan dalam suatu proses pengolahan data.
5. Desain tehnologi digunakan untuk menerima *input*, menjalankan model, menyimpan dan mengakses data, menghasilkan dan mengirimkan keluaran dan membantu pengendalian sistem secara keseluruhan. Tehnologi ini perlu dirancang untuk menyesuaikan dengan sistem informasi yang akan digunakan dengan memperhatikan tiga hal pokok, yaitu perangkat keras, perangkat lunak dan teknisi.

e. Mengembangkan dan mendokumentasikan perangkat lunak.

   Tahap ini dilakukan untuk mengembangkan suatu perangkat lunak yang diperlukan, dalam kegiatannya diperlukan kerjasama antara penganalisis dan pemrograman.

f. Menguji dan mempertahankan sistem.

   Sebelum sistem informasi dapat digunakan, maka harus dilakukan pengujian terlebih dahulu.

g. Mengimplementasikan dan mengevaluasi sistem.

Implementasi sistem dilakukan setelah rancangan selesai dan melakukan evaluasi untuk revisi dengan segera terhadap sistem untuk memastikan kesesuaian dengan kebutuhan. Evaluasi diharapkan bahwa sistem baru lebih efisien operasionalnya dan efektif dalam pencapaian tujuan dan lebih mudah digunakan.

2.12 Testing dan Implementasi Sistem

Menurut Romeo (2003:3), *Testing software* adalah proses mengoperasikan *software* dalam suatu kondisi yang dikendalikan untuk:

a. Verifikasi. Apakah telah berlaku sebagaimana yang ditetapkan (menurut spesifikasi)?
b. Mendeteksi error.
c. Validasi. Apakah spesifikasi yang ditetapkan telah memenuhi keinginan atau kebutuhan pengguna yang sebenarnya?

Menurut Romeo (2003:33), *Test Case* merupakan tes yang dilakukan berdasarkan pada suatu inisialisasi, masukan, kondisi ataupun hasil yang telah ditentukan sebelumnya.

a. White Box Testing

White box testing atau *glass box testing* atau *clear box testing* adalah suatu metode disain *test case* yang menggunakan struktur kendali dari disain prosedural. Metode disain *test case* ini dapat menjamin:

1. Semua jalur (*path*) yang independen/terpisah dapat dites setidaknya sekali tes.

2. Semua logika keputusan dapat dites dengan jalur yang salah atau jalur yang benar.
3. Semua *loop* dapat dites terhadap batasannya dan ikatan operasionalnya.
4. Semua struktur internal data dapat dites untuk memastikan validasinya.

b. Black Box Testing

Black box testing atau *behavioral testing* atau *specification-based testing*, *input/output testing* atau *functional testing* dilakukan tanpa sepengetahuan detil struktur internal dari sistem atau komponen yang dites. *Black box testing* berfokus pada kebutuhan fungsional pada *software*, berdasarkan spesifikasi kebutuhan dari *software*.

Menggunakan *black box testing*, perekayasa *software* dapat menggunakan sekumpulan kondisi masukan yang dapat secara penuh memeriksa keseluruhan kebutuhan funsional pada suatu program. Kategori *error* dapat diketahui melalui *black box testing*, antara lain:

1. Fungsi yang hilang atau tidak benar.
2. *Error* dari antar-muka.
3. *Error* dari struktur data atau akses eksternal *database*.
4. *Error* dari kinerja atau tingkah laku.
5. *Error* dari inisialisasi dan terminasi.