

BAB II

LANDASAN TEORI

2.1 Sistem Informasi

2.1.1 Pengertian Sistem

Sistem adalah suatu kumpulan atau himpunan dari unsur, komponen atau variabel-variabel yang terorganisasi, saling berinteraksi, saling tergantung satu sama lain dan terpadu.

Menurut Jogiyanto (2005) mengemukakan bahwa sistem adalah kumpulan dari elemen-elemen yang berinteraksi untuk mencapai suatu tujuan tertentu. sistem ini menggambarkan suatu kejadian-kejadian dan kesatuan yang nyata adalah suatu objek nyata, seperti tempat, benda, dan orang-orang yang betul-betul ada dan terjadi.

2.1.2 Pengertian Informasi

Menurut Jogiyanto (2005) “Informasi adalah data yang diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi penggunanya”. Informasi juga dapat diartikan sebagai data yang telah diolah menjadi bentuk yang mempunyai arti dan manfaat bagi manusia. Sedangkan data adalah aliran fakta mentah yang menunjukkan peristiwa yang telah terjadi dalam organisasi dan lingkungan fisik sebelum diorganisir dan ditata menjadi suatu bentuk yang bisa dipahami dan digunakan.

Informasi merupakan hal yang sangat penting bagi manajemen didalam pengambilan keputusan. Informasi dapat diperoleh dari sistem informasi (*information System*) atau yang disebut juga dengan *processing system* atau *information processing system*.

2.1.3 Pengertian Sistem Informasi

Leitch Rosses (Jogiyanto, 2005) mengemukakan sistem informasi adalah suatu sistem didalam organisasi yang mempertemukan kebutuhan pengelola transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari

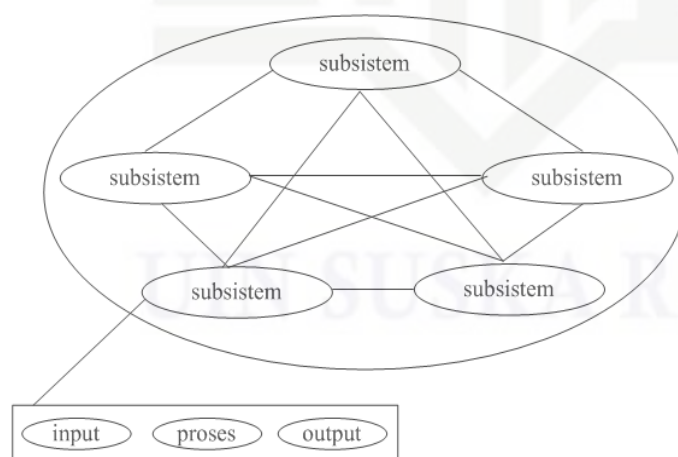
suatu organisasi dan menyediakan pihak luar tertentu dengan laporan-laporan yang diperlukan.

Menurut Jogiyanto (2005) Sistem Informasi didefinisikan oleh Rober Leitch dan K. Roscoe Davis: adalah suatu sistem dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat material dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan-laporan yang digunakan.

Melihat definisi sistem dan informasi serta definisi dari sistem informasi sendiri dapat dikatakan bahwa sistem informasi adalah suatu sistem dimana komponen-komponennya saling berhubungan dan bekerjasama untuk mengubah data menjadi informasi yang bermanfaat dan berguna bagi pemakai melalui serangkaian prosedur baik menggunakan perangkat keras maupun perangkat lunak.

2.1.4 Karakteristik Sistem

Suatu sistem mempunyai beberapa karakteristik yaitu, komponen atau elemen (*component*), batas sistem (*boundary*), lingkungan luar sistem (*environment*), penghubung (*interface*), masukan (*input*), pengolah (*process*), keluaran (*output*), sasaran (*objective*) atau tujuan (*goal*). Karakteristik sebuah sistem dapat dilihat pada Gambar 2.1.



Gambar 2.1 Karakteristik Sistem

(Sumber: Mulyanto, 2009)

Hak Cipta Dilindungi Undang-Undang

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber:

- a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penyusunan laporan, penulisan kritik atau tinjauan suatu masalah.
- b. Pengutipan tidak merugikan kepentingan yang wajar UIN Suska Riau.

2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin UIN Suska Riau.

1. Komponen Sistem

Suatu sistem tidak berada dalam lingkungan yang kosong, tapi sebuah sistem berada dan berfungsi di dalam lingkungan yang berisi sistem lainnya. Suatu sistem terdiri dari jumlah komponen yang saling berinteraksi, bekerjasama membentuk satu kesatuan. Apabila suatu sistem merupakan salah satu dari komponen sistem lain yang lebih besar, maka akan disebut dengan sub sistem, sedangkan sistem yang lebih besar tersebut adalah lingkungannya.

2. Batas Sistem (*Boundary*)

Merupakan pembatas atau pemisah antara suatu system dengan system yang lainnya atau dengan lingkungan luarnya. Batas sistem menentukan konfigurasi, ruang lingkup, atau kemampuan sistem . Batas sistem ini memungkinkan suatu sistem dipandang sebagai sebagai suatu kesatuan. Batas suatu sistem juga menunjukkan ruang lingkup (*scope*) dari sistem tersebut.

3. Lingkungan Luar Sistem (*Environment*)

Lingkungan luar adalah apa pun di luar batas dari sistem yang dapat memengaruhi operasi sistem, baik pengaruh yang menguntungkan ataupun yang merugikan. Pengaruh yang menguntungkan ini tentunya harus dijaga sehingga akan mendukung kelangsungan operasi sebuah sistem. Sedangkan lingkungan yang merugikan harus ditahan dan dikendalikan agar tidak mengganggu kelangsungan sebuah sistem.

4. Penghubung Sistem (*Interface*)

Penghubung merupakan hal yang sangat penting, sebab tanpa adanya penghubung, sistem akan berisi kumpulan subsistem yang berdiri sendiri dan tidak saling berkaitan.

Penghubung sistem merupakan media penghubung antara satu subsistem dengan subsistem yang lainnya. Penghubung inilah yang akan menjadi media yang digunakan data dari masukan sehingga keluaran. Dengan adanya penghubung, suatu sub sistem dapat berinteraksi dan berintegrasi

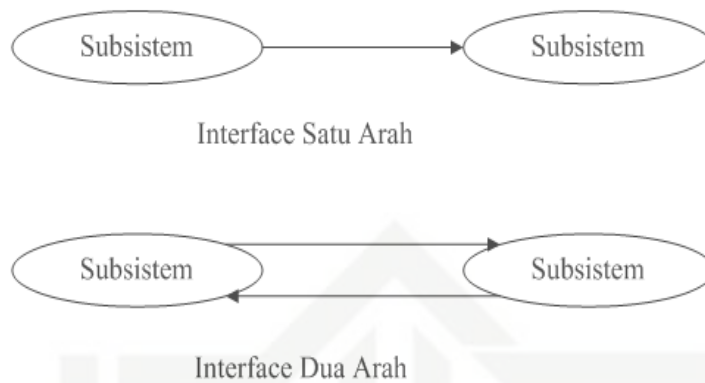
Hak Cipta Dilindungi Undang-Undang

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber:

- Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penyusunan laporan, penulisan kritik atau tinjauan suatu masalah.
- Pengutipan tidak merugikan kepentingan yang wajar UIN Suska Riau.

2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin UIN Suska Riau.

dengan sub sistem yang lain membentuk satu kesatuan. Model penghubung sistem dapat dilihat pada Gambar 2.2.



Gambar 2.2 Model *Interface*

(Sumber: Mulyanto, 2009)

5. Masukan Sistem

Masukan atau input merupakan energi yang dimasukkan kedalam system. Masukan dapat berupa masukan perawatan (*maintenance input*) dan masukan sinyal. *Maintenance input* adalah bahan yang dimasukkan agar sistem tersebut dapat beroperasi. *Signal input* adalah masukan yang diproses untuk mendapatkan keluaran.

6. Keluaran Sistem

Keluaran merupakan hasil dari pemrosesan. Keluaran dapat berupa informasi sebagai masukan pada sistem lain atau hanya sebagai sisa pembuangan.

7. Pengolah Sistem

Pengolahan sistem merupakan bagian yang melakukan perubahan dari masukan untuk menjadi keluaran yang diinginkan. Dalam sistem informasi, pengolahan dapat berupa operasi penjumlahan, pengurangan, perkalian, pembagian, pengurutan, atau operasi lainnya yang nantinya akan mengubah masukan berupa data informasi yang berguna.

8. Sasaran Sistem

Suatu sistem pasti memiliki sasaran (*objective*) atau tujuan (*goal*). Apabila sistem tidak mempunyai sasaran, maka operasi sistem tidak akan ada gunanya. Tujuan inilah yang mengarahkan suatu sistem. Tanpa adanya tujuan, sistem menjadi tidak terarah dan terkendali. Tujuan sistem informasi tergantung pada kegiatan yang ditangani.

Secara umum suatu sistem memiliki tiga tujuan utama, yaitu:

- a. Mendukung fungsi kepengurusan manajemen.
- b. Mendukung pengambilan keputusan manajemen.
- c. Mendukung kegiatan operasi perusahaan.

2.2 Analisis sistem

Analisis sistem adalah teknik pemecahan masalah yang menguraikan bagian-bagian komponen dengan mempelajari seberapa bagus bagian-bagian komponen tersebut bekerja dan berinteraksi untuk mencapai tujuan mereka.

Tujuan utama analisis sistem adalah untuk menentukan hal-hal detail yang akan dikerjakan oleh sistem yang diusulkan, analisis sistem mencakup spesifikasi perangkat lunak yang diinginkan dengan harapan agar terjadi komunikasi antar pembuat perangkat dengan pemakai.

Tahap analisis sistem dilakukan setelah tahap perencanaan sistem (*System Planning*). Tahap analisis merupakan tahap yang kritis dan penting, karena kesalahan pada tahap ini akan menyebabkan juga kesalahan selanjutnya.

Tujuan utama dalam tahap analisis untuk melihat secara keseluruhan permasalahan yang dihadapi oleh PT. Acc Pratama Rokan Hulu. Hal ini bertujuan menemukan kelemahan dalam sistem lama, sehingga dapat diusulkan perbaikannya. Dalam tahap analisis terdapat langkah-langkah dasar yang harus dilakukan yaitu sebagai berikut:

1. *Identify*, yaitu mengidentifikasi masalah.
2. *Understand*, yaitu memahami kerja dari sistem yang ada.
3. *Analyze*, yaitu menganalisis sistem.
4. *Report*, yaitu membuat laporan hasil analisis.

2.3 Rekayasa Perangkat Lunak

Bagian ini, akan dijelaskan definisi perangkat lunak (*software*), definisi rekayasa perangkat lunak, proses rekayasa perangkat lunak, *Object Oriented Analysis and Design* (OOAD) dan *Unified Modelling Language* (UML).

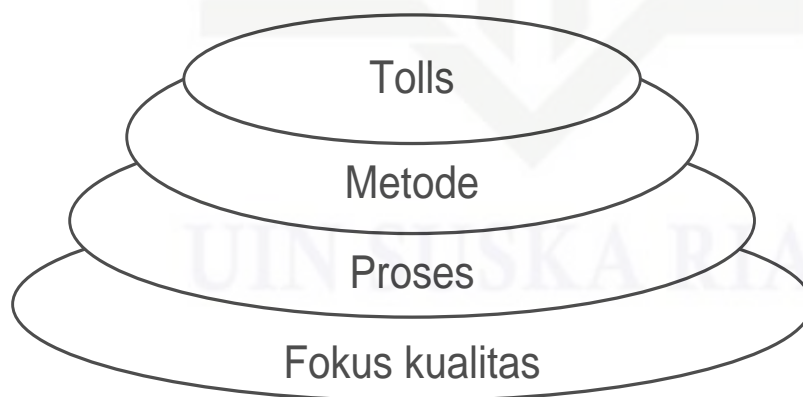
2.3.1 Pengertian Perangkat Lunak

Komputer merupakan mesin yang memproses fakta atau data menjadi informasi. Komputer digunakan orang untuk meningkatkan hasil kerja dan memecahkan berbagai masalah. Yang menjadi pemroses data atau pemecah masalah itu adalah perangkat lunak.

Perangkat lunak adalah perintah atau program komputer yang jika dijalankan akan menampilkan hasil sesuai dengan yang diinginkan. Struktur data yang memungkinkan sebuah program untuk mengubah suatu informasi. Informasi deskriptif dalam bentuk *hardcopy* atau *softcopy* yang menjelaskan cara kerja dan manfaat sebuah program (Pressman, 2010).

2.3.2 Pengertian Rekayasa Perangkat Lunak

Rekayasa perangkat lunak adalah pembuatan perangkat lunak dengan menggunakan prinsip rekayasa yang kuat untuk menghasilkan perangkat lunak yang ekonomis, handal, dan bekerja secara efisien. Rekayasa perangkat lunak merupakan teknologi yang bertingkat atau berlapis. Lapisan-lapisan teknologi rekayasa perangkat lunak dapat dilihat pada Gambar 2.3.



Gambar 2.3 Lapisan Teknologi Rekayasa Perangkat Lunak

(Sumber: Pressman, 2010)

Hak Cipta Dilindungi Undang-Undang

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber:

- a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penyusunan laporan, penulisan kritik atau tinjauan suatu masalah.
- b. Pengutipan tidak merugikan kepentingan yang wajar UIN Suska Riau.

2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin UIN Suska Riau.

1. Berpusat pada kualitas (*a quality focus*)

Semua proses perancangan (termasuk rekayasa perangkat lunak) sangat dipengaruhi oleh komitmen organisasi terhadap kualitas. Filosofi-filosofi tentang kualitas akan terus memperbaiki proses perancangan dan akhirnya akan berpengaruh terhadap pendekatan rekayasa perangkat lunak yang lebih efektif.

2. Proses (*process*)

Lapisan ini merupakan lapisan yang menghubungkan teknologi-teknologi yang digunakan dalam perancangan program dan memungkinkan pembuatan program diselesaikan dengan tepat waktu. Proses ini mendefinisikan *framework* yang harus dibuat agar teknologi yang digunakan dalam pembuatan program dapat dimanfaatkan dengan efektif.

3. Metode (*Methods*)

Lapisan metode dari rekayasa perangkat lunak menjelaskan secara teknis bagaimana cara membangun perangkat lunak. Lapisan ini terdiri dari tugas-tugas yang mencakup tentang analisis kebutuhan (*requirement analysis*), model desain (*design modelling*), pembuatan program (*program construction*), pengujian (*testing*) dan pendukung (*support*).

4. Alat (*Tools*)

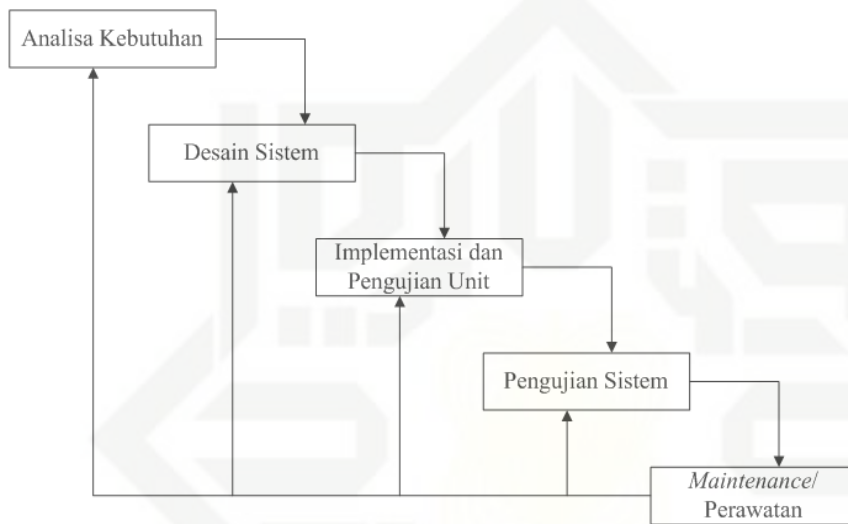
Lapisan ini menyediakan bantuan secara otomatis dan semi-otomatis untuk Lapisan proses dan metode. Ketika alat sudah terintegrasi sehingga informasi yang dihasilkan oleh suatu alat dapat digunakan oleh yang lain, maka terbentuklah sebuah sistem untuk membantu proses perekayasaan perangkat lunak yang disebut *Computer Aided Software Engineering*.

2.3.3 Model Proses Software

Siklus hidup sistem (*system life cycle*), atau yang disingkat SLC adalah proses evolusi yang diikuti dalam menetapkan sistem dan sub sistem informasi berbasis komputer. SLC terdiri dari serangkaian tugas yang erat mengikuti langkah-langkah pendekatan sistem, karena tugas-tugas tersebut mengikuti sebuah pola yang teratur dan dilakukan secara *top-down*, SLC sering disebut sebagai

pendekatan air terjun (*waterfall approach*) bagi pengembangan dan penggunaan sistem (Jogiyanto, 2005).

Model *waterfall* adalah struktur pengembangan sistem dimana setiap tahap harus diselesaikan terlebih dahulu secara penuh sebelum diteruskan ke tahap berikutnya untuk menghindari terjadinya pengulangan tahapan. Struktur pengembangan sistem *waterfall* dapat dilihat pada Gambar 2.4.



Gambar 2.4 Kerangka Kerja Model *Waterfall*

(Sumber: Agus Mulyanto, 2009)

Penulis menggunakan struktur pengembangan sistem tersebut karena penulis dalam menerapkan tahapan pengembangan sistem diselesaikan setiap tahap agar tidak terjadi pengulangan tahapan diantaranya:

1. Analisis Kebutuhan

Pada *fase* analisis kebutuhan iniseorang analisis sistem mengumpulkan kebutuhan secara lengkap, kemudian dianalisis dan didefinisikan kebutuhan-kebutuhan yang harus dipenuhi oleh sistem. Pada *fase* ini harus dikerjakan secara lengkap supaya menghasilkan desain yang lengkap. Biasanya kualitas informasi yang didapat dari *fase* analisis kebutuhan atau analisis sistem sangat mempengaruhi kualitas sistem.

Hak Cipta Dilindungi Undang-Undang

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber:

- a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penyusunan laporan, penulisan kritik atau tinjauan suatu masalah.
- b. Pengutipan tidak merugikan kepentingan yang wajar UIN Suska Riau.

2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin UIN Suska Riau.

2. Desain Sistem

Setelah kebutuhan dikumpulkan secara lengkap, maka mulai merancang sistem dengan digambarkan dengan model sistem menggunakan *Unified Modelling Language* (UML).

3. Implementasi dan Pengujian Unit

Kemudian pada *fase* implementasi, desain sistem diterjemahkan kedalam kode-kode dengan menggunakan bahasa pemrograman yang sudah ditentukan. Kemudian dilakukan pengujian terhadap unit-unit yang dihasilkan.

4. Pengujian Sistem

Pada *fase* pengujian sistem, unit-unit tersebut disatukan dan dilakukan pengujian secara keseluruhan.

5. Maintenance/Perawatan

Tahapan selanjutnya yaitu untuk memelihara sistem yang telah dibuat, supaya terjaga dengan baik.

Beberapa kelebihan dan kekurangan membangun sistem menggunakan metode *waterfall*:

1. Kelebihan Metode *Waterfall*

- a. Kualitas dari sistem yang dihasilkan akan baik. Ini dikarenakan oleh pelaksanaannya secara bertahap. Sehingga tidak terfokus pada tahapan tertentu.
- b. Dokumen pengembangan sistem sangat terorganisir, karena setiap *fase* harus terselesaikan dengan lengkap sebelum melangkah ke *fase* berikutnya. Jadi setiap *fase* atau tahapan akan mempunyai dokumen tertentu.
- c. Metode ini masih lebih baik digunakan walaupun sudah tergolong kuno, daripada menggunakan pendekatan asal-asalan. Selain itu, metode ini juga masih masuk akal jika kebutuhan sudah diketahui dengan baik.

2. Kelemahan *Waterfall*

- Diperlukan majemen yang baik, karena proses pengembangan tidak dapat dilakukan secara berulang sebelum terjadinya suatu produk.
- Kesalahan kecil akan menjadi masalah besar jika tidak diketahui sejak awal pengembangan yang berakibat pada tahapan selanjutnya.
- Pelanggan sulit menyatakan kebutuhan secara eksplisit sehingga tidak dapat mengakomodasi ketidak pastian pada saat awal pengembangan.
- Pelanggan harus sabar, karena pembuatan perangkat lunak akan dimulai ketika tahap desain sudah selesai. Sedangkan pada tahap sebelum desain bisa memakan waktu yang lama.
- Pada kenyataannya, jarang mengikuti urutan sekuensial seperti pada teori. Iterasi sering terjadi menyebabkan masalah baru.

2.3.4 Pengertian OOAD

OOAD adalah metode analisis yang memeriksa *requirements* dari sudut pandang kelas kelas dan objek yang ditemui dalam ruang lingkup permasalahan yang mengarahkan arsitektur *software* yang didasarkan pada manipulasi objek-objek sistem atau sub sistem. OOAD merupakan cara baru dalam memikirkan suatu masalah dengan menggunakan model yang dibuat menurut konsep sekitar dunia nyata. Dasar pembuatan adalah objek,yang merupakan kombinasi antara struktur data dan perilaku dalam satu entitas.

2.3.5 Konsep OOAD

OOAD mencakup analisis dan desain sebuah sistem dengan pendekatan objek, yaiut analisis berorientasi objek (OOA) dan desain berorientasi objek (OOD). OOA adalah metode analisis yang memeriksa *requirement* (syarat/keperluan) yang harus dipenuhi sebuah sistem dari sudut pandang kelas-kelas dan objek-objek yang ditemui dalam ruang lingkup perusahaan. Sedangkan OOD adalah metode untuk mengarahkan arsitektur *software* yang didasarkan pada manipulasi objek-objek sistem atau sub sistem (Sutopo, 2002).

Hak Cipta Dilindungi Undang-Undang

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber:

- a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penyusunan laporan, penulisan kritik atau tinjauan suatu masalah.
- b. Pengutipan tidak merugikan kepentingan yang wajar UIN Suska Riau.

2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin UIN Suska Riau.

Terdapat beberapa konsep dalam OOAD, yaitu (Sutopo, 2002):

1. Objek (*Object*)

- a. *Objek* adalah benda secara fisik dan konseptual yang ada di sekitar kita. Sebuah objek memiliki keadaan sesaat yang disebut *state*.
- b. *State* dari sebuah objek adalah kondisi dari objek atau himpunan keadaan yang menggambarkan objek tersebut. *State* dinyatakan dengan nilai dari atribut objeknya.
- c. *Atribut* adalah nilai internal suatu objek yang mencerminkan karakteristik objek, kondisi sesaat, koneksi dengan objek lain dan identitas.
- d. Perilaku objek (*Behaviour*) mendefinisikan bagaimana sebuah objek bertindak dan memberi reaksi. *Behaviour* ditentukan oleh himpunan semua atau beberapa operasi yang dapat dilakukan oleh objek tersebut, yang dicerminkan oleh *interface*, *service*, dan *method* dari objek tersebut.
- e. *Interface* adalah pintu untuk mengakses *service* dari objek
- f. *Service* adalah fungsi yang dapat dikerjakan oleh sebuah objek
- g. *Method* adalah mekanisme internal objek yang mencerminkan perilaku objek tersebut.

2. Kelas (*Class*)

Class adalah himpunan objek yang sejenis yaitu mempunyai sifat (atribut), perilaku umum (operasi), relasi umum dengan objek lain dan semantik umum. *Class* adalah abstraksi dari objek dalam dunia nyata. *Class* menetapkan spesifikasi perilaku dan atribut dari objek tersebut.

3. Kotak Hitam (*Black Box*)

Sebuah objek adalah kotak hitam. Konsep ini menjadi dasar implementasi objek. Dalam operasi *object oriented* hanya *developer* yang dapat memahami detail proses yang ada didalam kotak tersebut, sedangkan *user* tidak perlu mengetahui apa yang dilakukan yang penting mereka dapat

2.3.6 Teknik Pemodelan OOAD

Berikut adalah teknik pemodelan dalam *object oriented analyst design* (Sutopo, 2002):

1. Model Objek:
 - a. Model objek Menggambarkan struktur statis dari suatu objek dalam sistem dan relasinya
 - b. Model objek berisi diagram objek. Diagram objek adalah graph dimana nodenya adalah kelas yang mempunyai relasi antar kelas.
2. Model Dinamik:
 - a. Model dinamik menggambarkan aspek dari sistem yang berubah setiap saat.

menggunakan objek untuk memproses kebutuhan mereka. Kotak hitam berisi kode dan data.

- a. *Encapsulation*, yaitu proses menyembunyikan detail implementasi sebuah objek. Untuk mengakses data objek tersebut adalah melalui *interface*. Untuk berkomunikasi dengan objek digunakan *message*.
- b. *Message* adalah permintaan agar objek menerima untuk membawa metode yang ditunjukkan oleh perilaku dan mengembalikan result dari aksi tersebut kepada objek pengirim (sender).

4. Asosiasi dan Agregasi

- a. Asosiasi adalah hubungan yang mempunyai makna antara sejumlah objek. Asosiasi digambarkan dengan sebuah garis penghubung diantara objeknya. Contoh: asosiasi karyawan dengan unit kerja. Setiap karyawan bekerja di satu unit kerja, sedangkan unit kerja dapat memiliki beberapa karyawan.
- b. *Agregasi* adalah bentuk khusus sebuah asosiasi yang menggambarkan seluruh bagian pada suatu objek merupakan bagian dari objek yang lain. Contoh: kopling dan piston adalah bagian dari mesin, sedangkan mesin, roda, *body* merupakan bagian dari sebuah mobil.

Hak Cipta Dilindungi Undang-Undang

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber.

a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penyusunan laporan, penulisan kritik atau tinjauan suatu masalah.

b. Pengutipan tidak merugikan kepentingan yang wajar UIN Suska Riau.

2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin UIN Suska Riau.

Hak Cipta Dilindungi Undang-Undang

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber:

- Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penyusunan laporan, penulisan kritik atau tinjauan suatu masalah.
- Pengutipan tidak merugikan kepentingan yang wajar UIN Suska Riau.

2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin UIN Suska Riau.

- Model dinamik dipergunakan untuk menyatakan aspek kontrol dari sistem.
 - Model dinamik berisi *state diagram*. *State diagram* adalah *graph* dimana *node*-nya adalah *state* dan *arc* adalah transisi antara *state* yang disebabkan oleh *event*.
3. Model Fungsional:
- Model fungsional menggambarakan transformasi nilai data di dalam sistem.
 - Model fungsional berisi data *flow diagram*. DFD adalah suatu *graph* dimana *node*-nya menyatakan proses dan *arc*-nya adalah aliran data.

2.3.7 Karakteristik Objek

Berikut karakteristik dari objek (Sutopo, 2002) :

- Objek
 - Identitas berarti bahwa data diukur mempunyai nilai tertentu yang membedakan entitas disebut Objek.
 - Objek dapat kongkrit, seperti halnya arsip dalam sistem, atau konseptual seperti kebijakan penjadwalan dalam *multiprocessing* pada sistem operasi.
 - Setiap objek mempunyai sifat yang melekat pada identitasnya.
 - Dua objek dapat berbeda walaupun bila semua nilai atributnya identik
- Macam-Macam Objek dapat dilihat pada Gambar 2.5.



Mobil



Singa

| NoPeg | Nama |
|-------|-------|
| 96001 | Susan |
| 96002 | David |
| 97001 | Shila |

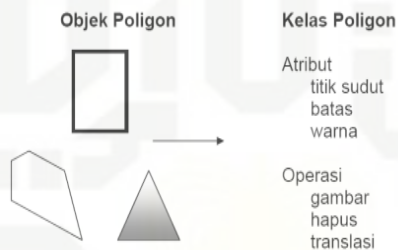
Tabel

Gambar 2.5 Macam-Macam objek

2. Kelas dan Objek

- Kelas merupakan gambaran sekumpulan Objek yang terbagi dalam atribut, operasi, metode, hubungan, dan makna yang sama.
- Suatu kegiatan mengumpulkan data (*atribut*) dan perilaku (operasi) yang mempunyai struktur data sama ke dalam satu grup.
- Kelas Objek merupakan wadah bagi Objek. Dapat digunakan untuk menciptakan Objek.
- Objek mewakili fakta/keterangan dari sebuah kelas.

Penjelasan secara umum kelas dan objek dapat dilihat pada Gambar 2.6.



Gambar 2.6 Kelas dan Objek

2.3.8 Karakteristik Metodologi Berorientasi Objek

Metodologi pengembangan sistem berorientasi objek mempunyai tiga karakteristik utama (Nugroho, 2005):

1. *Encapsulation* (Peng kapsulan)

Encapsulation merupakan dasar untuk pembatasan ruang lingkup program terhadap data yang diproses. Data dan prosedur atau fungsi dikemas bersama-sama dalam suatu objek, sehingga prosedur atau fungsi lain dari luar tidak dapat mengaksesnya. Data terlindung dari prosedur atau objek lain, kecuali prosedur yang berada dalam objek itu sendiri.

2. *Inheritance* (Pewarisan)

Inheritance adalah teknik yang menyatakan bahwa anak dari objek akan mewarisi data/atribut dan metode dari induknya langsung. Atribut dan metode dari objek dari objek induk diturunkan kepada anak objek, demikian seterusnya. *Inheritance* mempunyai arti bahwa atribut dan operasi yang dimiliki bersama di antara kelas yang mempunyai hubungan

Hak Cipta Dilindungi Undang-Undang

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber:

- a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penyusunan laporan, penulisan kritik atau tinjauan suatu masalah.
- b. Pengutipan tidak merugikan kepentingan yang wajar UIN Suska Riau.

2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin UIN Suska Riau.

secara hirarki. Suatu kelas dapat ditentukan secara umum, kemudian ditentukan spesifik menjadi sub kelas. Setiap sub kelas mempunyai hubungan atau mewarisi semua sifat yang dimiliki oleh kelas induknya, dan ditambah dengan sifat unik yang dimilikinya. Kelas Objek dapat didefinisikan atribut dan *service* dari kelas Objek lainnya. *Inheritance* menggambarkan generalisasi sebuah kelas.

3. *Polymorphism* (Perbedaan Bentuk)

Polimorfisme yaitu konsep yang menyatakan bahwa sesuatu yang sama dapat mempunyai bentuk dan perilaku berbeda. Polimorfisme mempunyai arti bahwa operasi yang sama mungkin mempunyai perbedaan dalam kelas yang berbeda. Kemampuan objek-objek yang berbeda untuk melakukan metode yang pantas dalam merespon message yang sama. Seleksi dari metode yang sesuai bergantung pada kelas yang seharusnya menciptakan Objek.

2.4 Model Umum Perancangan Analisis dan Perancangan Sistem

Adapun model perancangan analisis dan perancangan sistem yang akan digunakan adalah:

2.4.1 *Unified Modeling Language* (UML)

Pada perkembangan perangkat lunak, diperlukan bahasa yang digunakan untuk memodelkan perangkat lunak yang akan dibuat dan perlu adanya standarisasi agar orang diberbagai negara dapat mengerti pemodelan perangkat lunak. Banyak orang yang telah membuat bahasa pemodelan pembangunan perangkat lunak sesuai dengan teknologi pemrograman yang berkembang pada saat itu, misalnya sempat berkembang dan digunakan oleh banyak pihak adalah *Data Flow Diagram* (DFD) untuk memodelkan perangkat lunak yang menggunakan pemrograman prosedural atau struktural, kemudian juga ada *State Transition Diagram* (STD) yang digunakan untuk memodelkan sistem *real time* (waktu nyata).

Perkembangan teknik pemrograman berorientasi objek, munculah sebuah standarisasi bahasa pemodelan untuk pembangunan 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 (S. Rosa A dan M.Salahuddin, 2011).

1. Sejarah UML

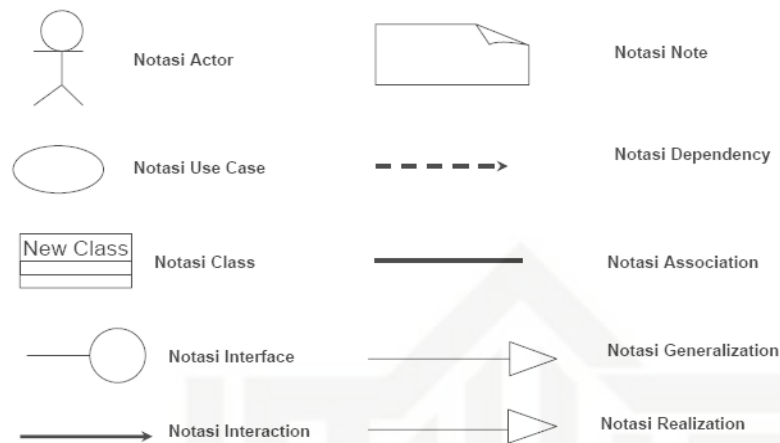
Bahasa pemrograman berorientasi objek yang pertama dikembangkan dikenal dengan Simula-67 yang dikembangkan pada tahun 1967. Bahasa pemrograman ini kurang berkembang dan dikembangkan lebih lanjut, namun dengan kemunculannya telah memberikan sumbangan yang besar pada *develover* pengembang bahasa pemrograman berorientasi objek selanjutnya (S. Rosa A dan M.Salahuddin, 2011).

2. Tujuan UML

Tujuan utama *Unified Modeling Language* (UML) diantaranya adalah sebagai berikut:

- Menyediakan bahasa pemodelan visual yang *expresip* dan siap pakai untuk mengembangkan dan pertukaran model-model yang berarti.
- Menyediakan mekanisme perluasan dan spesialisasi untuk memperluas konsep-konsep inti.
- Mendukung spesifikasi independen bahasa pemrograman dan proses pengembangan tertentu.
- Menyediakan basis formal untuk pemahaman bahasa pemodelan.
- Mendukung konsep-konsep pengembangan level lebih tinggi seperti komponen, kolaborasi, *framework* dan *pattern*.

Dalam diagram UML terdapat beberapa notasi yang digunakan, notasi tersebut akan dijelaskan pada Gambar 2.7.



Gambar 2.7 Notasi Dalam UML

(Sumber: S. Rosa A dan M.Salahuddin, 2011)

3. Diagram-Diagram UML

UML mempunyai sejumlah elemen grafis yang bisa dikombinasikan menjadi diagram. Karena ini merupakan sebuah bahasa, UML mempunyai aturan untuk menggabungkan dan mengkombinasikan elemen-elemen tersebut.

Dalam membangun suatu model perangkat lunak dengan UML, digunakan bentuk-bentuk diagram atau simbol untuk merepresentasikan elemen-elemen dalam sistem. Bentuk diagram yang digunakan untuk merepresentasikannya adalah sebagai berikut (Nugroho, 2005):

1. *Use case Diagram*
2. *Activity Diagram*
3. *Sequence diagram*
4. *Class Diagram*

Bentuk diagram UML di atas akan dijelaskan pada Tabel 2.1.

Tabel 2.1 Tipe Diagram UML

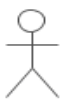
| Diagram | Tujuan |
|-----------------|--|
| <i>Use Case</i> | Menunjukkan sekumpulan kasus fungsional dan aktor dan hubungannya. |
| <i>Activity</i> | Pandangan operasi, bagaimana objek-objek bekerja, aksi-aksi yang mempengaruhi obyek, pandangan <i>use case workflow</i> . |
| <i>Sequence</i> | Berfungsi untuk <i>overview</i> perilaku sistem, menunjukkan objek-objek yang diperlukan, mendokumentasikan skenario dari suatu diagram <i>use case</i> , memeriksa jalur-jalur pengaksesan. |
| <i>Class</i> | Memodelkan kosakata di sistem, distribusi dan tanggung jawab, tipe primitif, kolaborasi, skema <i>database</i> logik. |

(Sumber: Nugroho, 2005)









2.4.2 Use Case Diagram

Use case atau diagram *use case* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu. Terdapat simbol-simbol pada *use case diagram* yang akan dijelaskan pada Tabel 2.2.

Tabel 2.2 Simbol-Simbol *Use Case Diagram*

| No | Gambar | Nama | Keterangan |
|----|---|--------------|---|
| 1 |  | <i>Actor</i> | Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan <i>use case</i> . |

Tabel 2.2 Simbol-Simbol *Use Case Diagram* (Lanjutan)

| No | Gambar | Nama | Keterangan |
|----|---|-----------------------|--|
| 2 |  | <i>Dependency</i> | Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (<i>independent</i>) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (<i>independent</i>). |
| 3 |  | <i>Generalization</i> | Hubungan dimana objek anak (<i>descendent</i>) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (<i>ancestor</i>). |
| 4 |  | <i>Include</i> | Menspesifikasikan bahwa <i>use case</i> sumber secara <i>eksplisit</i> . |
| 5 |  | <i>Extend</i> | Menspesifikasikan bahwa <i>use case</i> target memperluas perilaku dari <i>use case</i> sumber pada suatu titik yang diberikan. |
| 6 |  | <i>Association</i> | Apa yang menghubungkan antara objek satu dengan objek lainnya. |
| 7 |  | <i>System</i> | Menspesifikasikan paket yang menampilkan sistem secara terbatas. |
| 8 |  | <i>Use Case</i> | Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu actor |
| 9 |  | <i>Collaboration</i> | Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi). |

Hak Cipta Dilindungi Undang-Undang


1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber:

a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penyusunan laporan, penulisan kritik atau tinjauan suatu masalah.

b. Pengutipan tidak merugikan kepentingan yang wajar UIN Suska Riau.

2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin UIN Suska Riau.

Tabel 2.2 Simbol-Simbol *Use Case Diagram* (Lanjutan)

| No | Gambar | Nama | Keterangan |
|----|---|-------------|---|
| 10 |  | <i>Note</i> | Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi |

(Sumber: Nugroho, Adi. 2005)

2.4.3 *Class Diagram*



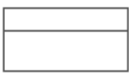
Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi.

1. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas.
2. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.


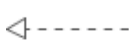
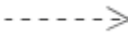
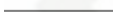
Terdapat simbol-simbol pada *class diagram* yang akan dijelaskan pada

Tabel 2.3.

Tabel 2.3 Simbol-Simbol *Class Diagram*

| No | Gambar | Nama | Keterangan |
|----|---|-------------------------|---|
| 1 |  | <i>Generalization</i> | Hubungan dimana objek anak (<i>descendent</i>) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (<i>ancestor</i>). |
| 2 |  | <i>Nary Association</i> | Upaya untuk menghindari asosiasi dengan lebih dari 2 objek. |
| 3 |  | <i>Class</i> | Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama. |

Tabel 2.3 Simbol-Simbol *Class Diagram* (Lanjutan)

| No | Gambar | Nama | Keterangan |
|----|---|----------------------|--|
| 4 |  | <i>Collaboration</i> | Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu actor. |
| 5 |  | <i>Realization</i> | Operasi yang benar-benar dilakukan oleh suatu objek. |
| 6 |  | <i>Dependency</i> | Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (<i>independent</i>) akan memengaruhi elemen yang bergantung padanya elemen yang tidak mandiri. |
| 7 |  | <i>Association</i> | Apa yang menghubungkan antara objek satu dengan objek lainnya. |

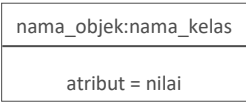

(Sumber: Nugroho, Adi. 2005)

2.4.4 *Object Diagram*

Object Diagram menggambarkan struktur sistem dari segi penamaan objek dan jalannya objek dalam sistem. Pada diagram objek harus dipastikan semua kelas yang sudah didefinisikan pada diagram kelas harus dipakai objeknya, karena jika tidak, pendefinisian kelas itu tidak dapat dipertanggungjawabkan.

Untuk apa mendefinisikan sebuah kelas sedangkan pada jalannya sistem, objeknya tidak pernah dipakai. Hubungan *link* pada diagram objek merupakan hubungan memakai dan dipakai dimana dua buah objek akan dihubungkan oleh link jika ada objek yang dipakai oleh objek lainnya. Terdapat simbol-simbol pada *object diagram* yang akan dijelaskan pada Tabel 2.4.

Tabel 2.4 Simbol-Simbol *Object Diagram*




| No | Gambar | Nama | Keterangan |
|----|---|---------------|---|
| 1 |  | <i>Object</i> | Objek dari kelas saat sistem dijalankan |
| 2 |  | <i>Link</i> | Relasi antar objek |

(Sumber: Nugroho, Adi. 2005)

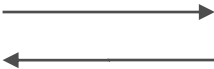
2.4.5 Collaboration Diagram

Collaboration diagram merupakan prinsip yang sama dengan *sequence diagram*, sama-sama memodelkan interaksi antar objek-objek, yang membedakan hanya cara penggambarannya saja. Pada *collaboration diagram* ini, objek-objek dan *message* (pesan) yang digambarkan mirip seperti *flowchart*, hanya saja untuk menjaga urutan pesan yang diterima oleh masing-masing objek, pesan-pesan tersebut diberi nomor urutan pesan. Terdapat simbol-simbol pada *collaboration diagram* yang akan dijelaskan pada Tabel 2.5.

Tabel 2.5 Simbol-Simbol *Collaboration Diagram*

| Simbol | Keterangan |
|---|--|
|  | Actor |
|  | <i>Object instance:</i> Obyek yang dibuat, melakukan tindakan, dan / atau dimusnahkan selama lifeline |
|  | Interaksi <i>link</i> : Merupakan indikasi bahwa obyek kejadian dan berkolaborasi aktor dan pertukaran pesan. |

Tabel 2.5 Simbol-Simbol *Collaboration Diagram* (Lanjutan)

| Simbol | Keterangan |
|---|--|
|  | Sinkronis pesan: Seketika sebuah komunikasi antara objek-objek yang menyampaikan informasi, dengan harapan bahwa tindakan akan dimulai sebagai hasil. |

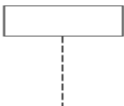

(Sumber: Nugroho, Adi. 2005)

2.4.6 Sequence Diagram


Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambar *sequence diagram* maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

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. Terdapat simbol-simbol pada *Sequence Diagram* yang akan dijelaskan pada Tabel 2.6.

Tabel 2.6 Simbol-Simbol *Sequence Diagram*

| No | Gambar | Nama | Keterangan |
|----|---|-----------------|--|
| 1 |  | <i>LifeLine</i> | Objek <i>entity</i> , antarmuka yang saling berinteraksi. |
| 2 |  | <i>Message</i> | Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi |

Tabel 2.6 Simbol-Simbol *Sequence Diagram* (Lanjutan)

| No | Gambar | Nama | Keterangan |
|----|---|----------------|--|
| 3 |  | <i>Message</i> | Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi |

(Sumber: Nugroho, Adi. 2005)

2.4.7 Activity Diagram



Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem.

Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut:




1. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
2. Urutan atau pengelompokan tampilan dari sistem / *user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
3. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.

Terdapat simbol-simbol pada *Activity Diagram* yang akan dijelaskan pada Tabel 2.7.

Tabel 2.7 Simbol-Simbol *Activity Diagram*

| No | Gambar | Nama | Keterangan |
|----|---|-----------------|---|
| 1 |  | <i>Activity</i> | Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain |
| 2 |  | <i>Action</i> | State dari sistem yang mencerminkan eksekusi dari suatu aksi |

Tabel 2.7 Simbol-Simbol *Activity Diagram* (Lanjutan)

| No | Gambar | Nama | Keterangan |
|----|---|------------------------------------|--|
| 3 |  | <i>Initial Node</i> | Bagaimana objek dibentuk atau diawali. |
| 4 |  | <i>Actify</i> <i>Final Node</i> | Bagaimana objek dibentuk dan dihancurkan |
| 5 |  | <i>Fork Node</i> | Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran |

(Sumber: Nugroho, Adi. 2005)

2.5 Basisdata (*Database*)

Basis data merupakan kumpulan dari data yang saling berhubungan dengan yang lainnya, tersimpan diperangkat keras komputer dan digunakan perangkat lunak untuk memanipulasinya. *Database* merupakan salah satu komponen yang penting dalam sistem informasi, karena merupakan basis dalam menyediakan informasi bagi para pemakai. Penerapan *database* dalam sistem informasi disebut dengan *database system*.

Sistem basis data (*database system*) adalah suatu sistem informasi yang mengintegrasikan kumpulan dari data yang saling berhubungan satu dengan yang lainnya dan membuatnya tersedia untuk beberapa aplikasi yang bermacam-macam di dalam suatu organisasi. Dengan sistem basis data ini tiap-tiap orang atau bagian dapat memandang *database* dari beberapa sudut pandang yang berbeda. Bagian kredit dapat memandangnya sebagai data piutang, bagian penjualan dapat memandangnya sebagai data penjualan, bagian personalia dapat memandangnya sebagai data karyawan, bagian gudang dapat memandangnya sebagai data persediaan. Semuanya terintegrasi dalam sebuah data yang umum. Berbeda dengan sistem pengolahan data tradisional, sumber data ditangani sendiri-sendiri untuk tiap aplikasinya.

2.6 Hierarki Data Dalam Database

Data dalam sebuah *database* disusun berdasarkan sistem hirarki yang unik, yaitu:

- a. *Database* merupakan kumpulan file yang saling terkait satu sama lain.
- b. *File* yaitu kumpulan dari *record* yang saling terkait dan memiliki format *field* yang sama dan sejenis.
- c. *Record* yaitu atribut dari *record* yang menunjukkan suatu unit data dari individu tertentu.
- d. *Field* yaitu atribut dari *record* yang menunjukkan suatu item dari data
- e. *Byte* yaitu atribut dari *field* yang berupa huruf yang membentuk nilai dari sebuah *field*.
- f. *Byte* yaitu bagian terkecil dari data secara keseluruhan, yaitu berupa karakter ASCII nol atau satu yang merupakan komponen pembentuk *byte*.

2.7 Internet

Internet merupakan salah satu infrastruktur utama e-bisnis. Pada mulanya, jaringan *internet* dikembangkan sebagai saluran khusus untuk aktivitas riset dan keperluan para akademisi. Dalam perkembangannya, *internet* dieksploitasi untuk berbagai keperluan lainnya, termasuk untuk keperluan bisnis. *Internet* itu sendiri sebenarnya adalah singkatan dari *Interconnection Networking*. Menurut Randall dan Latulipe Secara sederhana, *internet* bisa diartikan sebagai “a global network of computer networks”. Dengan demikian, pada dasarnya *internet* merupakan jaringan komputer yang sangat besar terbentuk dari jaringan–jaringan kecil yang saling terhubung satu sama lain. Jaringan *internet* sukses dikembangkan dan diuji coba pertama kali pada tahun 1969 oleh U.S. Department of Defense dalam proyek ARPANet (*Advanced Research ProjectsNetwork*).

2.8 Software Yang Digunakan

Berikut adalah beberapa software yang digunakan dalam pembuatan sistem informasi penjualan berbasis *web*:

2.8.1 PHP (*Hypertext Preprocessor*)

PHP singkatan dari *Hypertext Preprocessor* yang digunakan sebagai bahasa *script server* dalam pengembangan *web* yang disisipkan pada dokumen HTML.

1. Sejarah Singkat PHP

PHP diciptakan pertama kali oleh Rasmus Lerdorf pada tahun 1994. Awalnya, PHP digunakan untuk mencatat jumlah serta untuk mengetahui siapa saja pengunjung pada *homepage*-nya. Rasmus Lerdorf adalah salah seorang pendukung *open source*. Oleh karena itu, ia mengeluarkan *Personal Home Page Tools* versi 1.0 secara gratis, kemudian menambah kemampuan PHP 1.0 dan meluncurkan PHP 2.0.

Pada tahun 1996, PHP telah banyak digunakan dalam website di dunia. Sebuah kelompok pengembang *software* yang terdiri dari Rasmus, Zeew Suraski, Andi Gutman, Stig Bakken, Shane Caraveo, dan Jim Winstead bekerja sama untuk menyempurnakan PHP 2.0. akhirnya, pada tahun 1998, PHP 3.0 diluncurkan. Penyempurnaan terus dilakukan sehingga pada tahun 2000 dikeluarkan PHP 4.0. Tidak berhenti sampai disitu, kemampuan PHP terus ditambah, dan saat buku ini disusun, versi terbaru telah dikeluarkan adalah PHP 5.0 X (Kasiman:2006).

2. Pengertian PHP

PHP merupakan bahasa *server-side* yang cukup handal, yang akan disatukan dengan HTML dan berada di *server*. Artinya, sintaks dan perintah yang diberikan akan sepenuhnya dijalankan di *server* sebelum dikirim ke komputer klien. Pada awal tahun 1995, Rasmus Lerdorf membuat produk bernama PHP/FI (*Personel Home Page/Form Interpreter*). Produk yang merupakan cikal bakal PHP ini ditulis menggunakan bahasa C, dan memiliki kemampuan untuk berkomunikasi dengan database serta membuat halaman dinamis.

Seluruh aplikasi yang berbasis *web* dapat dibuat menggunakan PHP. Salah satu kelebihan PHP adalah kemampuan untuk dapat melakukan koneksi dengan berbagai *database*, seperti MySQL, PostgreSQL, dan Access. Selain itu PHP juga bersifat *open source*, untuk dapat menggunakannya kita tidak perlu membayar.

Variabel PHP digunakan untuk menyimpan data yang nilainya dapat berubah-ubah. Dalam bahasa PHP, variabel dimulai dengan tanda "\$". Aturan penulisan variabel antara lain sebagai berikut:

- a. Hanya ada 3 karakter yang dapat digunakan untuk nama variabel yaitu huruf, angka dan garis bawah.
 - b. Karakter pertama setelah tanda "\$" harus huruf atau garis bawah.
 - c. Jika nama variabel lebih dari satu kata. Tidak boleh ada tanda spasi di antara keduanya.
3. Keunggulan PHP

PHP (*Hypertext Preprocessor*) adalah skrip yang berjalan dalam *server side* yang ditambahkan dalam HTML (*Hyper Text Markup Languages*). Skrip ini akan membuat suatu aplikasi dapat diintegrasikan ke dalam HTML sehingga suatu halaman HTML tidak lagi bersifat statis, namun menjadi dinamis. Sifat *server side* ini membuat pengerjaan skrip tersebut dikerjakan di *server* sedangkan yang dikirimkan kepada *browser* adalah hasil proses dari skrip tersebut yang sudah berbentuk HTML.

Keunggulan dari PHP adalah sebagai berikut:

- a. Bahasa pemrograman PHP adalah sebuah bahasa script yang tidak melakukan sebuah kompilasi dalam penggunaannya.
- b. *Web Server* yang mendukung PHP dapat ditemukan dimana-mana dari mulai apache, IIS, Lighttpd, hingga Xitami dengan konfigurasi yang relatif mudah.
- c. Dalam sisi pengembangan lebih mudah, karena banyaknya forum dan developer yang siap membantu dalam pengembangan.
- d. Dalam sisi pemahaman, PHP adalah bahasa *scripting* yang paling mudah karena memiliki referensi yang banyak.

Hak Cipta Dilindungi Undang-Undang

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber:

- a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penyusunan laporan, penulisan kritik atau tinjauan suatu masalah.
- b. Pengutipan tidak merugikan kepentingan yang wajar UIN Suska Riau.

2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin UIN Suska Riau.

- e. Skrip asli tidak dapat dilihat, sehingga keamanan lebih terjamin.
- f. PHP adalah bahasa *open source* yang dapat digunakan di berbagai mesin (*Linux, Unix, Macintosh, Windows*) dan dapat dijalankan secara *runtime* melalui *console* serta juga dapat menjalankan perintah-perintah sistem.

PHP dapat digunakan pada semua sistem operasi, antara lain *Linux, Unix* (termasuk variannya *HP-UX, Solaris dan OpenBSD*), *Microsoft Windows, Mac OS X*, dan masih banyak lagi lainnya, bahkan PHP dapat bekerja sebagai suatu *CGI processor*.

PHP tidak terbatas pada hasil keluaran HTML. PHP juga memiliki kemampuan untuk mengolah keluaran gambar, *file PDF*, dan movies Flash. PHP juga dapat menghasilkan teks seperti XHTML dan *file XML* lainnya.

Salah satu fitur yang dapat diandalkan oleh PHP adalah dukungannya terhadap banyak *database*. Berikut *database* di dukung oleh PHP (Kasiman, 2006):

- 1) *Adabas D*
- 2) *Dbase*
- 3) *Direct MS-SQL*
- 4) *Empress*
- 5) *Interbase*
- 6) *MSQL*
- 7) *MySQL*
- 8) *Oracle*
- 9) *Unix DBM*
- 10) *Velocis*

2.8.2 Adobe Photoshop

Adobe Photoshop adalah *software* pengolah gambar yang sangat *powerfull* dengan segala fasilitasnya. Dapat digunakan sebagai sebuah *image editor*, atau program penyunting gambar yang berfungsi untuk membuat, menyunting, dan memodifikasi gambar-gambar digital yang terdapat di dalam komputer. Hasil gambar olah dengan *Adobe Photoshop* ini banyak dilihat diberbagai *website*, brosur, koran, majalah, dan media lainnya. Kemampuan serta fasilitasnya yang lengkap membuatnya diminati oleh para seniman, professional, maupun pemula yang membutuhkan sebuah program gambar yang lengkap namun mudah dalam penggunaannya.

2.8.3 Macromedia Dreamweaver

Macromedia Dreamweaver merupakan *software web design* yang paling banyak digunakan di dunia. Dengan *Macromedia Dreamweaver* ini kita akan banyak belajar mengenai penggunaan *Spry Framework* untuk keperluan mendesain *web* yang profesional, di samping itu banyak pula diperoleh informasi mengenai bagaimana meng-*edit image*, membuat template, menggunakan *cascading style sheet (CSS)* dan membuat *web* foto album.

Sebagai contoh adalah *Dreamweaver 8*. *Dreamweaver* menjadi pilihan terunggul, baik oleh pereka *web* yang profesional, maupun mereka yang baru mengerti mengenai *internet*.

Ciri-ciri *Dreamweaver* yang terbaik termasuk kode HTML yang dikemas secara sederhana serta dihadapan pada pilihan-pilihan HTML yang terbaru seperti HTML Dinamik dan gaya sunting melata CSS. *Dreamweaver* juga mempunyai fitur cara penyunting teks yang terpadu serta didukung dengan bahasa pemrograman atau *JavaSkrip*. *Macromedia Dreamweaver* memadukan BBEdit (aturan cara HTML yang terpopuler bagi Macintosh) dan *Homesite* (untuk Windows) dengan bentuk WYSIWYG yang mudah digunakan. Menggunakan program *Dreamweaver* para pengguna dapat menikmati manfaat bagaimana menyusun kode HTML yang baik.

Beraneka ragam bentuk grafik *Dreamweaver* menggunakan palet dan template yang sudah tersedia untuk memudahkan pengguna-pengguna yang baru mulai belajar menciptakan *Web* yang didalamnya memuat berbagai ciri-ciri seperti animasi, borang interaktif dan penyelesaian *e-commerce*, walaupun mereka tidak memahami HTML.

Dreamweaver menyediakan pengguna memilih berbagai macam bentuk template untuk membangun web yang baik. *Macromedia Dreamweaver* juga terdapat *tools* yang memudahkan pengeditan untuk pengguna baru, seperti mencari dan mengganti garis-garis teks ataupun kode dengan apa saja parameter yang ditentukan. Panel perjalanan (*behaviors panel*) juga memudahkan penciptaan Java Script yang menarik tanpa pengetahuan pengekodean HTML.

2.8.4 MySQL

Berikut ini akan dijelaskan pengertian beserta keunggulan dari MySQL:

1. Pengertian MySQL

MySQL adalah sebuah aplikasi *Relational Database Management Server* (RDBMS) yang sangat cepat dan kokoh serta bersifat *open source*. MySQL merupakan salah satu jenis *database server* yang banyak digunakan di dunia maya, yang menggunakan *SQL* sebagai bahasa dasar untuk mengakses database. Database adalah sekumpulan tabel yang saling berhubungan satu sama lain, yang tujuannya adalah memelihara informasi dan membuat informasi tersedia saat dibutuhkan. Untuk menambah, mengakses, dan memproses data yang disimpan di komputer, diperlukan sistem manajemen database seperti *MySQL*.

2. Kelebihan MySQL

MySQL dapat digunakan pada berbagai *platform* sistem operasi. Keunggulan MySQL dalam mengolah database adalah (Saputro. 2005) :

a. Kecepatan.

Berdasarkan hasil pengujian, MySQL memiliki kecepatan yang paling baik dibandingkan RDBMS lainnya. Contohnya MySQL 4.0 kinerja *query* naik sebesar 200% dari kinerja biasa.

Hak Cipta Dilindungi Undang-Undang

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber:

- a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penyusunan laporan, penulisan kritik atau tinjauan suatu masalah.
- b. Pengutipan tidak merugikan kepentingan yang wajar UIN Suska Riau.

2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin UIN Suska Riau.

- b. Mudah digunakan.
Perintah dalam MySQL dan aturan-aturannya relatif mudah diingat dan diimplementasikan, karena MySQL menggunakan *Structured Query Language* (SQL) sebagai bahasa standar *database*.
- c. *Open Source*
MySQL sudah menggunakan konsep *Open Source*, siapapun dapat mengembangkan MySQL dan hasil pengembangannya dipublikasikan kepada para pemakai.
- d. Kapabilitas
MYSQL mampu memproses data yang tersimpan dalam database dengan jumlah 50 juta *record*, 60.000 tabel dan 5.000.000.000 jumlah baris, serta mampu memproses sebanyak 32 *indeks* per-tabel.
- e. Biaya murah
Pemakai dapat menggunakan MySQL tanpa harus mengeluarkan biaya yang cukup mahal selama mengikuti konsep *open source/GNU Public License*.
- f. Keamanan
MySQL menerapkan system keamanan dan hak akses secara bertingkat, termasuk dukungan dengan keamanan data secara pengacakan lapisan data. Adanya tingkatan *user* dan jenis akses yang beragam dan sistem pengacakan *password* (*encrypted password*)
- g. *Lintas Platform*
MySQL dapat dijalankan pada beberapa sistem operasi, diantaranya yaitu Linux, Windows, FreeBSD, Novell Netware, Sun Solaris, Sco OpenUnix dan IBM's AIX.
- h. Minim "bug"
Khususnya pada MySQL dengan keterangan "*recommended*".

Hak Cipta Dilindungi Undang-Undang

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber:
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penyusunan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar UIN Suska Riau.
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin UIN Suska Riau.

3. Fungsi MySQL

Ada beberapa fungsi yang digunakan dalam MySQL. Fungsi tersebut sangat erat kaitannya dengan *query* SQL. Akan tetapi *user* tidak dapat langsung menggunakan perintah SQL pada *script* PHP. Disini fungsi MySQL-lah yang digunakan sebagai penghubung antar SQL sehingga *query* tersebut dapat dijalankan pada *server* dan dapat dilihat hasilnya oleh *client* (Nugroho, 2005).

Fungsi MySQL dapat juga dikatakan sebagai interpreter *query* karena setiap *user* menggunakan *query* SQL, maka fungsi ini harus diletakkan. Dengan kata lain *query* SQL tidak dapat dijalankan tanpa adanya fungsi MySQL.