BAB II
LANDASAN TEORI

2.1 Konsep Dasar Sistem Informasi

Suatu sistem sangat dibutuhkan dalam suatu perusahaan atau instansi pemerintahan, karena sistem sangatlah menunjang terhadap kinerja perusahaan atau instansi pemerintah, baik yang berskala kecil maupun besar supaya dapat berjalan dengan baik diperlukan kerjasama diantara unsur-unsur yang terkait dalam sistem tersebut.

2.1.1 Pengertian Sistem

Secara Sederhana, suatu sistem dapat diartikan sebagai suatu kumpulan atau himpunan dari unsur, komponen, atau variabel yang terorganisir, saling berinteraksi, saling tergantung satu sama lain, dan terpadu. (Tata Sutarbi 2012).

2.1.2 Karakteristik Sistem

Model umum sebuah sistem adalah *input,* proses dan *output*. Hal ini merupakan konsep sebuah sistem yang sangat sederhana sebab sebuah sistem dapat mempunyai beberapa masukan dan keluaran. Selain itu, sebuah sistem mempunyai karakteristik atau sifat-sifat tertentu yang mencirikan bahwa hal tersebut bisa dikatakan sebagai suatu sistem. Adapun karakteristik yang dimaksud adalah sebagai berikut :

a. Komponen Sistem (*System Components)*

Suatu sistem yang terdiri dari sejumlah komponen yang salling berinteraksi, artinya saling bekerja sama membentuk satu kesatuan. Komponen-komponen sistem tersebut dapat berupa suatu fungsi tertentu dan mempengaruhi proses sistem secara keseluruhan. Suatu sistem dapat mempunyai sistem yang lebih besar atau sering disebut “supra sistem”.

b. Batasan Sistem (*System Boundary*)

Ruang lingkup sistem merupakan daerah yang membatasi antara sistem dengan sistem yang lain atau sistem sedengan lingkungan luarnya.

Batasan sistem ini memungkinkan suatu sistem dipandang sebagai suatu kesatuan yang tidak dapat dipisahkan.

c. Lingkungan Luar Sistem (System Environment)

Bentuk apapun yang ada diluar lingkup atau batasan sistem yang mempengaruhi operasi sistem tersebut disebut lingkungan luar sistem. Lingkungan luar sistem ini dapat bersifat menguntungkan dan dapat juga bersifat merugikan sistem tersebut. Dengan demikian, lingkungan luar tersebut harus tetap dijaga dan dipelihara. Lingkungan luar yang merugikan harus dikendalikan. Kalau tidak, maka akan mengganggu kelangsungan hidup sistem tersebut.

d. Penghubung Sistem (Inteface)

Media yang menghubungkan sistem dengan subsistem lain disebut penghubung sistem atau *interface*. Penghubung ini memungkinkan sumber-sumber daya mengalir dari satu subsistem ke subsistem lain. Bentuk keluaran dari satu subsistem akan menjadi masukan untuk subsistemlain melalui penghubung tersebut. Dengan demikian, dapat terjadi suatu integrasi sistem yang membentuk satu kesatuan.

e. Masukan Sistem (Input)

Energi yang dimasukkan ke dalam sistem disebut masukkan sistem, yang dapat berupa pemeliharaan (*maintenance input)* dan sinyal (*signal input*). Contoh, di dalam suatu unit sistem komputer, *"program*" adalah *maintenance input* yang digunakan untuk mengoperasikan komputernya dan *"data"* adalah signal *input* untuk diolah menjadi informasi.

f. Pengolahan Sistem (Process)

Suatu sistem dapat mempunyai suatu proses yang akan mengubah masukan menjadi keluaran, contohnya adalah sistem akuntantsi. Sistem ini akan mengolah data transaksi menjadi laporan-laporan yang dibutuhkan oleh pihak manajemen.

g. Keluaran Sistem (Output)

Hasil energi yang diolah dan diklasifikasi menjadi keluaran yang berguna. Keluaran ini merupakan masukan bagi subsistem yang lain seperti sistem informasi. Keluaran yang dihasilkan adalah informasi. Informasi ini

dapat digunakan sebagai masukkan untuk pengambilan keputusan atau hal-hal lain sebagai *input* bagi subsistem lain.

h. Sasaran Sistem (Objective)

Suatu sistem memiliki tujuan dan sasaran yang pasti dan bersifat deterministic. Kalau suatu sistem tidak memiliki sasaran maka operasi sistem tidak ada gunanya. Suatu sistem dikatakan berhasil bila mengenai sasaran atau tujuan yang telah direncanakan..

2.1.3 Klasifikasi Sistem

Menurut Tata Sutarbi (2012), Sistem merupakan suatu bentuk integrasi antara lain satu komponen dengan komponen lain karena sistem meiliki sasaran yang berbeda untuk setiap kasus yang terjadi yang ada di dalam sistem tersebut. Oleh karena itu, sistem dapat diklasifikasikan dari beberapa sudut pandang di antaranya :

a) Sistem abstrak dan sistem fisik.

Sistem Abstrak adalah sistem yang berupa pemikira atau ide-ide yang tidak tampak secara fisik, misalnya sistem teologia, yaitu sistem yang berupa pemikiran hubungan antara manusia dengan Tuhan. Sedangkan sistem fisik adalah sistem yang ada secara fisik. Misalnya sistem komputer, sistem produksi, sistem penjualan, sistem administrasi personalia dan lain sebagainya.

b) Sistem alamiah dan sistem buatan manusia.

Sistem alamiah adalah sistem yang terjadi melalui proses alam, tidak dibuat oleh manusia. Misalnya sistem perputaran bumi, terjadinya siang dan malam, pergantian musim. Sedangkan sistem buatan manusia merupakan sistem yang melibatkan interaksi manusia dan mesin yang disebut *human machine system*. Sistem informasi berbasis komputer merupakan contoh *human machine* karena menyangkut penggunaan komputer yang berinteraksi dengan manusia.

c) Sistem determinasi dan sistem probabilistik.

Sistem yang beroperasi dengan tingkah laku yang dapat diprediksi disebut sistem *deterministic*. Sistem komputer adalah contoh sistem yang tingkah lakunya dapat dipastikan berdasarkan program-program komputer yang dijalankan. Sedangkan sistem yang bersifat probabilistik adalah sistem yang kondisi masa depannya tidak dapat diprediksi karena mengandung unsur *probabilistic*.

d) Sistem terbuka dan sistem tertutup.

Sistem terbuka adalah sistem yang berhubungan dan dipenguruhi oleh lingkungan luarnya. Sistem ini menerima masukan dan menghasilkan keluaran untuk subsistem lainnya. Sedangkan sistem tertutup adalah sistem yang tidak berhubungan dan tidak terpengaruh oleh lingkungan luarnya. Sistem ini bekerja otomatis tanpa camput tangan pihak luar.

2.2 Konsep Dasar Informasi

2.2.1 Pengertian Informasi

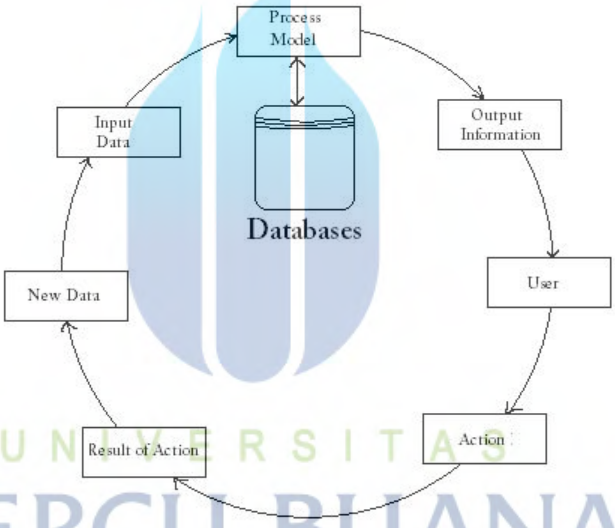
Menurut Tata Sutarbi (2012), Informasi adalah sebuah istilah yang tepat dalam pemakaian umum. Informasi dapat mengenai data mentah, data tersusun, kapasitas sebuah saluran komunikasi, dan lain sebagainya.Informasi ibarat darah yang mengalir didalam tubuh suatu organisasi, sehingga informasi ini sangat penting di dalam suatu organisasi, sehingga informasi ini sangat penting di dalam suatu informasi. Suatu sistem yang kurang mendapat informasi akan menjadi luruh, kerdil dan akhirnya berakhir.

Teori informasi lebih tepat disebut sebagai teori matematis dan komunikasi. Sumber informasi adalah data. Data merupakan kenyataan yang menggambarkan suatu kejadian serta merupakan suatu bentuk yang masih mentah yang belum dapat bercerita banyak sehingga perlu diolah lebih lanjut melalui suatu model untuk menghasilkan informasi. Jelaslah kiranya bahwa data adalah sumber informasi. Pengolah informasi adalah salah satu elemen kunci dalam sistem konseptual. Pengolahan informasi dapat meliputi elemen-elemen komputer, elemen-elemen non komputer atau kombinasinya.

2.2.2 Siklus Informasi

Menurut Tata Sutarbi (2012), Data merupakan bentuk yang masih mentah yang belum dapat berceritera banyak, sehingga perlu diolah lebih lanjut. Data diolah melalui suatu model untuk dihasilkan informasi.

Data yang diolah melalui suatu model menjadi informasi, penerima kemudian menerima informasi tersebut, membuat suatu keputusan dan melakukan tindakan, yang berarti menghasilkan suatu yang lain yang akan membuat sejumlah data kembali. Data tersebut akan ditangkap sebagai input, diproses kembali lewat suatu model dan seterusnya membentuk siklus. Siklus ini oleh Jhon Burch disebut dengan siklus informasi (information cycle).Siklus ini disebut juga dengan siklus pengolahan data (data *processing cycle*).



Gambar 1.1 Siklus Sistem informasi

2.2.3 Kualitas Informasi

Dengan informasi yang berkualitas kita bisa mudah mengambil suatu keputusan. Menurut Al-Bahra bin Ladjamudin (Sutabri, 2012) kualitas informasi (*quality of information*) dipengaruhi oleh beberapa faktor. Faktor-faktor yang dapat mepengaruhi kualitas informasi diantaranya adalah sebagai berikut:

1. **Akurat (*accurate*)**

   Informasi harus bebas dari kesalahan-kesalahan dan tidak bias atau menyesatkan. Akurat juga berarti informasi harus jelas mencerminkan maksudnya.

2. **Tepat waktu (timeline)**

   Informasi yang datang pada si penerima tidak boleh terlambat. Informasi yang sudah usang tidak akan mempunyai nilai lagi karena informasi merupakan landasan dalam pengambilan keputusan.

3. **Relevan (relevance)**

   Informasi tersebut mempunyai manfaat untuk pemakainya. Relevansi informasi untuk orang satu dengan yang lain berbeda.

4. **Ekonomis (economy)**

   Informasi yang dihasilkan mempunyai daya jual yang tinggi serta biaya operasinal untuk menghasilkan informasi tersebut minimal, informasi tersebut juga mampu memberikan dampak yang luas terhadap laju pertumbuhan ekonomi dan teknologi informasi.

5. **Efisien (efficiency)**

   Informasi yang berkualitas memiliki sintaks ataupun kalimat yang sederhana (tidak berbelit-belit), namun mampu memberikan makna dan hasil.

6. **Dapat dipercaya (reliability)**

   Informasi tersebut berasal dari sumber yang dapat dipercaya.Sumber tersebut juga telah teruji tingkat kebenarannya. Misalkan output suatu program komputer, bisa dikategorikan sebagai reliability karena program komputer akan memberikan output sesuai dengan input yang diberikan.

2.3 Pemodelan Sistem

Prototyping paradigmadimulai dengan pengumpulan kebutuhan.Secara ideal prototipe berfungsi sebagai sebuah mekanisme untuk mengidentifikasi kebutuhan perangkat lunak. Bila prototipe sedang bekerja atau dibangun,

pengembang harus mempergunakan fragmen-fragmen yang ada atau mengaplikasikan alat-alat bantu. (Verdi Yasin 19)

Gambar 2.2 Model PrototypeParadigma (Roger S. Pressman, 2002)

Proses pada *prototyping* antara lain sebagai berikut:

a. Proses pengumpulan kebutuhan

Pengembang dan pengguna aplikasi bertemu dan menentukan tujuan umum, kebutuhan yang diketahui dan gambaran bagian-bagian yang akan dibutuhkan berikutnya.

b. Proses perancangan

Pada tahap ini, penulis melakukan perancangan dilakukan cepat dan rancangan mewakili aspek software yang diketahui.Dengan rancangan ini menjadi dasar pembuatan *prototype*.

c. Tahap evaluasi *prototype*

Pengguna aplikasi mengevaluasi *prototype* yang dibuat dan dipergunakan untuk menjelaskan kebutuhan software.

Secara ideal prototype berfungsi sebagai sebuah mekanisme untuk mengidentifikasi kebutuhan perangkat lunak. Bila prototype yang sedang bekerja dibangun, pengembang harus mempergunakan fragmen-fragmen program yang ada atau mengaplikasikan alat-alat bantu (contohnya report generator, window manager, dll) yang memungkinkan program yang bekerja untuk memunculkan secara cepat.

2.4 Teori Pengujian

Pengujian sebaiknya dilakukan secara bertahap, tidak dilakukan secara satuu tahap langsung di akhir untuk menghindari kesulitan penelusuran jika terjadi kesalahan (*error*). Pengujian yang dilakukan antara lain sebagai berikut :

2.4.1 Black Box Testing

Black Box adalah menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian dimaksudkan untuk mengetahui apakah fungsi –fungsi, masukan dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan.

2.4.2 White Box Testing

White Box adalah menguji perangkat lunak dari segi desain dan kode program apakah mampu menghasikan fungsi-fungsi, masukan, dan keluaran yang sesuai dengan spesifikasi kebutuhan.

2.5 Konsep Dasar Basis Data

2.5.1 Pengertian Basis Data

Menurut Fathansyah (2012:2) basis data dapat didefinisikan dalam sejumlah sudut pandang seperti:

a. Himpunan kelompok data (arsip) yang saling berhubungan yang diorganisasi sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah.
b. Kumpulan data yang saling berhubungan yang disimpan secara bersama sedemikian rupa dan tanpa pengulangan (redudansi) yang tidak perlu, untuk memenuhi berbagai kebutuhan.
c. Kumpulan file/table/arsip yang saling berhubungan yang disimpan dalam media penyimpanan elektronis.

Basis data merupakan sekumpulan data yang disusun secara logis dan dikendalikan secara sentral. Basis data memiliki bagian-bagian yang penting, yaitu:

1. **Karakter-karakter**

   Karakter merupakan bagian data yang terkecil, dapat berupa karakter *numeric*, huruf ataupun karakter-karakter khusus yang membentuk suatu item data.

2. ***Field***

   Suatu *field* menggambarkan suatu atribut dari *record* yang menunjukkan suatu item dari data, seperti misalnya nama alamat dan lain sebagianya. Kumpulan dari *field* membentuk suatu *record*.

3. ***Record***

   Kumpulan dari *field* membentuk suatu *record*.*Record* menggambarkan suatu unit data individu tertentu. Kumpulan dari *record* membentuk suatu *file*. Misalnya file personalia, tiap-tiap *record* dapat mewakili data tiap-tiap karyawan.

4. ***File***

   File terdiri dari *record-record* yang menggambarkan satu kesatuan data yang sejenis. Misalnya file matakuliah berisi data tentang semua mata kuliah yang ada.

5. ***Database***

   Kumpulan dari file membentuk suatu *database*.

2.5.2 Sistem Basis Data

Sistem basis data dapat terbagi dalam beberapa komponen penting, yakni:

1. **Data**

   Merupakan informasi yang disimpan dalam suatu struktur tertentu yang terintegrasi.

2. ***Hardware***

   Perangkat keras yang biasanya digunakan untuk pengelolaan system basis data yaitu:

   a. Komputer (untuk system yang *stand-alone* atau sistem jaringan)
   b. Memori sekunder yang *on-line* (*harddisk*).

c. Memori sekunder yang *off-line* (*Tape* atau *Removable Disk*) untuk keperluan *backup*data.

d. Media/perangkat komunikasi (untuk system jaringan)

3. Sistem Operasi (*Software*)

Program yang mengaktifkan/memfungsikan sistem komputer, mengendalikan seluruh sumber daya dalam komputer dan melakukan operasi-operasi dasar dalam komputer yang meliputi operasi *input-output* (IO), pengelolaan file dan sebagainya.

4. Basis Data

Basis data sebagai inti dari sistem basis data.Basis data menyimpan data serta struktur sistem basis data baik untuk entitas maupun objek-objeknya secara detail.

5. *Database Management System* (DBMS)

Pengelolaan basis data secara fisik tidak dilakukan oleh pemakai secara langsung, tetapi ditangani oleh sebuah perangkat lunak (*software*) khusus yang disebut DBMS (*Database Management System*). DBMS akan menentukan bagaimana data diorganisasi, disimpan, diubah, dan diambil kembali. Selain itu DBMS juga menerapkan mekanisme pengamanan data, pemakaian data secara bersamaan, pemaksaan keakuratan/konsistensi data, dan sebagainya.

6. Pemakai (*User*)

a. *Database Administrator* (DBA) adalah orang atau tim yang bertugas mengelola system basis data secara keseluruhan.

b. *Programmer*adalah orang atau tim yang bertugas membuat program aplikasi, misalnya untuk bidang perbankan, administrasi, akuntansi, dan lain-lain.

c. *End User* adalah orang yang mengakses basis data dengan mengggunakan *query language*atau program aplikasi yang dibuat oleh *programmer.*

7. Perangkat Lunak Lain

Perangkat lunak lain ini berfungsi untuk melakukan pengisian, pengubahan, dan pengambilan data. Program ini ada yang sudah disediakan

bersamaan dengan DBMS-nya, ada juga yang harus dibuat sendiri dengan menggunakan *development tools*.

2.5.3 Perancangan Basis Data

Perancangan basis data merupakan tahap merancang spesifikasi basis data yang akan diterapkan oleh sistem. Aktifitas perancangan basis data akan mentrasformasikan spesifikasi kebutuhan untuk tempat penyimpanan data yang akan dikembangkan selama analisis basis data ke dalam spesifikasi terstuktur untuk memandu implementasi langsung basis data. Ada tiga tahapan atau bentuk spesifikasi perancangan basis data, yakni :

1. Perancangan basis data konseptual

   Merupakan upaya untuk membuat model yang masih bersifat konsep. Perancangan basis data secara konseptual terdiri dari tiga langkah, yaitu:

   a. Penentuan entitas pada basis data
   b. Pendefinisian hubungan antar entitas
   c. Penerjemahan hubungan ke dalam entitas

2. Perancangan basis data secara logika

   Sasaran perancangan basis data secara logika untuk menterjemahkan perancangan konseptual (mencerminkan kebutuhan data pada organisasi yang kita analisis di tahap-tahap sebelumnya) ke perancangan basis data logika yang dapat langung di implementasikan pada sistem basis data yang dipilih.

3. Perancangan basis data secara fisik

   Merupakan tahapan untuk menuangkan basis data yang bersifat logis menjadi basis data fisik yang tersimpan pada media penyimpanan eksternal.Sasaran utama perancangan basis data secara fisik adalah meningkatkan efisiensi dalam pemrosesan basis data. Perancangan basis data secara fisik membutuhkan beberapa pilihan kritis yang akan berimbas pada integritas dan kinerja dari aplikasi termasuk diantaranya pemilihan DBMS yang spesifik. Kunci-kunci untuk melakukan pilihan-pilihan mencangkup :

4. Pemilihan format penyimpanan (dinamakan tipe data) untuk tiap atribut dari model data logika. Format dipilih untuk meminimalisasi dan

mengoptimalkan penggunaan ruang fisik dan untuk memaksimalkan integritas data.

5. Pengelompokan atribut-atribut dari model data logika ke-rekaman fisik.
6. Perancangan rekaman-rekaman pada memori sekunder (terutama hardisk) sehingga rekaman-rekaman secara individual maupun kelompok rekaman (kita namakan pengorganisasian berkas) dapat disimpan, dipanggil kembali serta diperbaharui dengan cepat.
7. Memilih struktur (dinamakan indeks dan arsitektur basis data) untuk menyimpan dan menghubungkan berkas-berkas (file) sehingga pemanggilan data berlangsung dengan cara yang efisien.
8. Menyiapkan strategi-strategi untuk menangani *query* pada basis data dengan tujuan mengoptimalisasikan kinerja basis data dalam menangani *query*.

Menurut Fathansyah (2012 : 15)DBMS merupakan perantara bagi pemakai dengan basis data dalam *disk*.Cara berinterkasi antara pemakai dengan basis data tersebut diatur dalam suatu bahasa khusus yang ditetapkan oleh perusahaan pembuat DBMS.Bahasa itu dapat kita sebut sebagai Bahasa Basis Data yang terdiri atas sejumlah perintah (*statement*) yang diformulasikan dan dapat diberikan user dan dikenali/diproses oleh DBMS untuk melakukan suatu aksi tertentu. Contoh-contoh bahasa basis data adalah SQL, dBase, QUEL, dan sebagainya.

Sebuah Bahasa Basis Data biasanya dapat dipilah ke dalam 2 bentuk yaitu:

1. *Data Definitian Language* (DDL)

   Struktur basis data yang menggambarkan skema basis data secara keseluruhan dan didesain dalam bahasa khusus yang disebut *Data Definitian Language* (DDL).Dengan bahasa istilah inilah kita dapat membuat tabel baru, membuat indeks, mengubah tabel, menentukan struktur penyimpanan tabel, dan sebagainya. Hasil dari kompilasi perintah DDL adalah kumpulan tabel yang disimpan dalam *file* khusus yang disebut Kamus Data (*Data Disctionary*). Kamus Data merupakan suatu metadata (*super-data*) yaitu data yang mendeskripsikan data

sesuangguhnya. Kamus data ini akan selalu diakses dalam suatu operasi basis data sebelum suatu *file* data yang sesungguhnya diakses.

2. *Data Manipulation Language* (DML)

   Merupakan bentuk Bahasa Basis Data yang berguna untuk melakukan manipulasi dan pengambilan data pada suatu basis data. Manipulasi data dapat berupa:

   - Penambahan data baru ke suatu basis data
   - Penghapusan data dari suatu basis data
   - Pengubahan data di suatu basis data

Data Manipulation Language (DML) merupakan bahasa yang bertujuan memudahkan pemakai untuk mengakses data sebagaimana direpresentasikan oleh Model Data. Ada 2 jenis DML, yaitu:

a. Prosedural, yang mensyaratkan agar pemakai menentuka, data apa yang diinginkan serta bagaimana cara mendapatkannya.
b. Non Prosedural, yang membuat pemakai dapat menentukan data apa yang diinginkan tanpa menyebutkan bagaimana cara mendapatkannya.

2.6 Analisis Dan Desain Berorientasi Objek

Menurut Prabowo Pudjo Widodo & Herlawati (2011:2), Pemrograman berorientasi objek bekerja dengan baik ketika dibarengi dengan *Object-oriented Analysis and Design Process (OOAD*).(Wampler, 2001:2) mengatakan jika kita membuat program berorientasi objek tanpa OOAD, ibarat membangun rumah tanpa terlebih dahulu menganalisa apa saja yang dibutuhkan oleh rumah itu, tanpa perencanaan, tanpa blueprint, tanpa menganalisis ruangan apa saja yang diperlukan, berapa besar rumah yang akan dibangun dan sebagainya

2.6.1 Objek (Object)

Orientasi objek merupakan teknik dalam menyelesaikan masalah yang kerap muncul dalam pengembangan perangkat lunak. Teknik ini merupakan titik kulminasi dalam menemukan cara yang efektif dalam membangun sistem dan menjadi metode yang paling banyak dipakai oleh para pengembang perangkat lunak

saat ini. Orientasi objek merupakan teknik permodelan sistem riil yang berbasis objek.Inti dari konsep ini adalah objek yang merupakan model dari sistem nyata.

Menurut (Douglas, 2004: bab 2.1) objek adalah entitas yang memiliki atribut, karakter (behavour) dan kadangkala disertai kondisi (state). Objek merepresentasikan sesuatu sistem real seperti siswa, sistem control permukaan sayap pesawat, sensor atau mesin. Objek juga merepresentasikan sesuatu dalam bentuk konsep seperti nasabah bank, merek dagang, pernikahan atau sekedar *listing*. Bahkan bisa juga menyatakan visualisasi seperti, bentuk huruf (*font*), histogram, poligon, garis atau lingkaran.Semuanya memiliki fitur atribut (untuk data), behavior (*operation* atau *method*), keadaan (memori), identitas dan tanggung jawab. Proses menjabarkan sistem nyata menjadi objek dinamakan abstraksi (*abstraction*). Abstraksi mengeliminir aspek yang tidak di perlu dalam suatu objek.

2.6.2 Kelas (Class)

Menurut (Prabowo Pudjo Widodo & Herlawati, 2011:3), Kelas adalah penggambaran satu set objek yang memiliki atribut dan *behavior* yang sama. Kelas mirip tipe data pemograman non objek, tapi lebih komprehensif karena terdapat struktur sekaligus karakteristiknya.Kita dapat membentuk kelas baru yang lebih spesifik dari kelas *general*-nya.Kelas dan objek merupakan jantung dari pemrograman berorientasi objek. Untuk menghasilkan program jenis ini sangat penting untuk selalu berfikir dalam konsep objek.

2.6.3 Pembungkusan (*Encapsulation*)

(Nugroho, 2005:6) mengartikan pembungkusan sebagai penggabungan potongan-potongan informasi dan perilaku-perilaku spesifikasi spesifik yang bekerja pada informasi tersebut, kemudian mengemasnya menjadi apa yang disebut sebagai objek. Dalam perbankan kita mengenal objek rekening yang memiliki perilaku-perilaku misalnya buka, tutup, penarikan, penyimpanan, ubah nama, ubah alamat dan sebagainya. Akibatnya, perubahan – perubahan pada sistem perbankan yang berkaitan dengan rekening-rekening dapat secara sederhana diimplementasikan satu kali saja pada objek rekening.Keuntungan lainnya adalah membatasi efek-efek perubahan pada sistem. Misalnya, saat manajemen bank menentukan jika seseorang memiliki rekening pinjaman di bank yang

bersangkutan, rekening pinjaman itu harus dapat juga digunakan sebagai sarana bagi penarikan rekening.

2.6.4 Pewarisan (*Inheritance*) dan Generalisasi/Spesialiasi

Menurut (Prabowo Pudjo Widodo & Herlawati, 2011:5), konsep di mana metode dan atau atribut yangditentukan di dalam sebuah objek kelas dapat diwariskan atau digunakan lagi oleh objek kelas lainnya.Sedangkan generalisasi/spesialisasi merupakan teknik dimana atribut dan perilaku yang umum pada beberapa tipe kelas objek, dikelompokan (atau abstraksi) ke dalam kelasnya sendiri (dinamakan supertype).Atribut dan metode kelas objek supertype kemudian diwariskan oleh kelas objek tersebut (dinamakan subtype).

2.6.5 Polimorfisme

Menurut (Prabowo Pudjo Widodo & Herlawati, 2011:5), polimorfisme berarti suatu fungsionalitas yang diimplementasikan dengan berbagai cara yang berbeda. Pada terminologi berorientasi objek, ini berarti kita dapat memiliki berbagai implementasi untuk sebagai fungsionalitas tertentu. Sebagai contoh, misalkan kita akan mengembangkan sistem berbasis grafis. Saat pengguna mau menggambar sesuatu, misalnya garis atau lingkaran, sistem akan memunculkan perintah gambar. Sistem akan mengenali berbagai bentuk bentuk gambar, masing-masing dengan prilakunya sendiri-sendiri. Manfaat dari polimorfisme adalah kemudahan pemeliharaannya.Jika perlu menambahkan gambar baru (misalnya segitiga), maka cukup menambahkan fungsi baru (fungsi menggambar segitiga) sedangkan fungsi umumnya (fungsi gambar) tidak mengalami perubahan (Nugroho, 2005:10).

2.7 Unified Modeling Language (UML)

2.7.1 Sejarah UML

Pendekatan berorientasi objek *Unified Modeling Languange* (UML.2.0) berawal dari *Object Management Group* (OMG) dan ditemukan oleh Grady Booch, James Rumbaugh, dan Ivar Jacobson. Pendekatan *model-driven*, analisis dimulai dengan menggunakan kasus dan scenario kemudian mendifinisikan kelas domain

masalah yang terlibat pekerjaan pengguna. UML termasuk model persyaratan dengan diagram *use case*, deskripsi *use case, diagram activity* dan *diagram sequence*.

Pada 1996, *Object Management Group* (OMG) mengajukan proposal agar adanya standarisasi pemodelan berorientasi objek dan pada bulan September 1997 UML diakomodasi oleh OMG sehingga sampai saat ini UML telah memberikan kontribusinya yang cukup besar di dalam metodologi berorientasi objek dan hal-hal yang terkait di dalamnya.Secara fisik, UML adalah sekumpulan spesifikasi yang dikeluarkan oleh OMG. UML terbaru adalah UML 2.3 yang terdiri dari 4 macam spesifikasi, yaitu *Diagram Interchange Specification,* UML *Infrastructure*, UML Superstructure, dan *Object Constraint Language* (OCL). (Rosa A. S & M. Shalahuddin, 2013:139).

2.7.2 Pengenalan UML

UML tipe 2.0 adalah suatu metode terbuka yang digunakan untuk menspesifikasikan, menvisualisasikan, membangun dan mendokumentasikan artifak – artifak dari suatu pengembangan sistem piranti lunak yang berbasis pada objek.UML mendefinisikan notasi dan syntax seperti bahasa – bahasa lainnya. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak dan UML 2.0 terdiri dari 13 jenis diagram resmi seperti yang tertulis berikut ini : (Willey, 2010)

2.7.3 Diagram UML

Diagram UML telah dibagi menjadi dua kelompok : satu untuk memodelkan struktur sistem dan dua untuk modeling *behavior*. Diagram *structure* digunakan untuk mewakili data dan hubungan statis yang berada dalam suatu sistem informasi. Diagram *behavior* menyediakan analis dengan cara untuk menggambarkan hubungan dinamis antara objek yang mewakili sistem informasi bisnis. Diagram ini memungkinkan pemodelan perilaku dinamis dari objek individu.Diagram *behavior* mendukung analis dalam pemodelan persyaratan fungsional dari suatu sistem informasi yang berkembang. (Dennis, 2010).

Tabel 1.1 UML 2.0 Diagram Summary (Dennis, 2010)

Diagram Name	Kegunaan	Fase utama
Diagram Structure		
Class	Menggambarkan hubungan antara kelas dimodelkan dalam sistem.	Analis, Design
Object	Menggambarkan hubungan antara objek dimodelkan dalam sistem. Berfungsi ketika kasus aktual dari kelas akan berkomunikasi lebih baik terhadap model.	Analis, Design
Package	Kelompok elemen UML lain bersama-sama untuk membentuk konstruksi tingkat yang lebih tinggi.	Analis, Design, Implementasi
Deployment	menampilkan arsitektur fisik sistem. Bisa juga digunakan untuk menunjukkan perangkat lunak, komponen yang dikerahkan ke arsitektur fisik.	Design Fisik, Implementasi
Component	Menggambarkan hubungan fisik di antara komponen-komponen perangkat lunak.	Design Fisik, Implementasi
Composite Structure	Menggambarkan struktur internal dari kelas-yaitu, hubungan antara bagian-bagian dari sebuah kelas.	Analis, Design
Diagram Behavior		
Activity	Menggambarkan pekerjaan bisnis yang mengalir bebas dalam kelas, aliran kegiatan dalam kasus penggunaan, atau metode desain yang rinci.	Analis, Design

Sequence	Model perilaku objek dalam kasus yang digunakan berfokus pada urutan berdasarkan waktu dari suatu kegiatan.	Analis, Design
Communication	Model perilaku objek dalam kasus digunakan berfokus pada komunikasi antara satu set berkolaborasi objek dari suatu kegiatan.	Analis, Design
Interaction Overview	Menggambarkan gambaran tentang aliran kontrol dari suatu proses.	Analis, Design
Timing	Menggambarkan interaksi yang terjadi di antara satu set objek dan state berubah-ubah dan masuk melalui sebuah time axis.	Analis, Design
Behavioral StateMachine	Mengkaji perilaku dalam salah satu kelas.	Analis, Design
Protocol State Machine	Menggambarkan dependensi antara antarmuka yang berbeda dari sebuah kelas.	Analis, Design
Use Case	Menangkap kebutuhan bisnis untuk sistem dan untuk menggambarkan interaksi antara sistem dan lingkungannya.	Analis

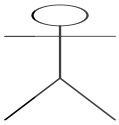
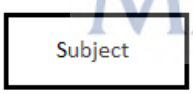
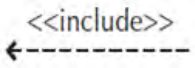
2.7.4 Use Case Diagram

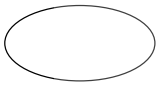
Use case atau diagram atau diagram *use case* merupakan pemodelan untuk sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Syarat penamaan *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut aktor dan use case. (Wiley, 2010)

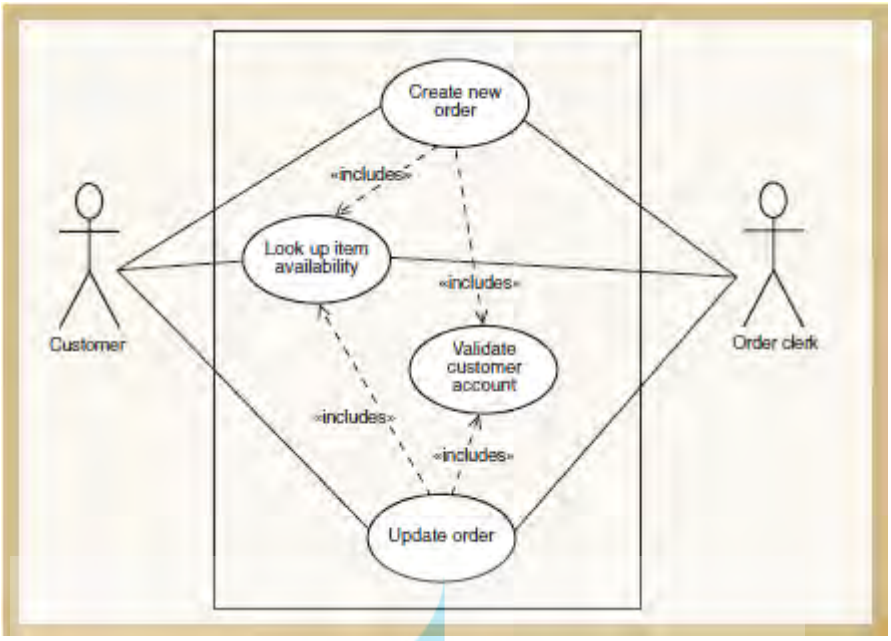
1. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
2. *Use case* merupakan fungsionalitas yang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

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

Tabel 2.2 Simbol – simbol Pemodelan *Use Case* Diagram (Willey, 2010)

GAMBAR	NAMA	KETERANGAN
	*Actor*	Aktor adalah orang atau sistem yang berasal dari manfaat dan eksternal untuk subjek. 1. Digambarkan sebagai salah satu tongkat (default) atau jika aktor non-manusia yang terlibat, sebagai persegi panjang dengan aktor <<>> di dalamnya (alternatif). 2. label dengan perannya. 3. Dapat dikaitkan dengan aktor-aktor lain menggunakan spesialisasi / asosiasi superclass, dilambangkan dengan panah dengan panah berongga. Ditempatkan di luar batas subjek.
	*Subject Boundary*	Termasuk nama subjek di dalam atau di atas. Merupakan lingkup subjek, misalnya, sistem atau individu proses bisnis.
	*Generalization*	Merupakan kasus penggunaan khusus ke yang lebih umum. Memiliki panah yang diambil dari kasus penggunaan khusus untuk kasus penggunaan dasar.
	*Include*	Merupakan dimasukkannya fungsi satu kasus penggunaan dalam yang lain.

		Memiliki panah yang diambil dari kasus penggunaan dasar untuk kasus penggunaan digunakan.
<<extend>> ----------→	*Extend*	Merupakan perpanjangan dari kasus penggunaan untuk memasukkan perilaku opsional. Memiliki panah yang diambil dari kasus penggunaan ekstensi untuk kasus penggunaan dasar.
_____	*Association*	*Link* aktor dengan *use case* (s) dengan yang berinteraksi.
	*Use case*	*Use Case*: 1. Merupakan bagian utama dari fungsi sistem. 2. Dapat memperpanjang kasus penggunaan lain. 3. Dapat mencakup use case lain. 4. Apakah ditempatkan di dalam batas sistem. Apakah label dengan frase kata kerja-kata benda deskriptif.



Gambar 2.3 Contoh Use Case Diagram (Satzinger, 2010)

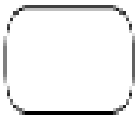
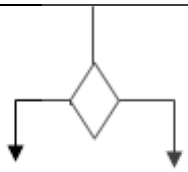
2.7.5 Activity Diagram

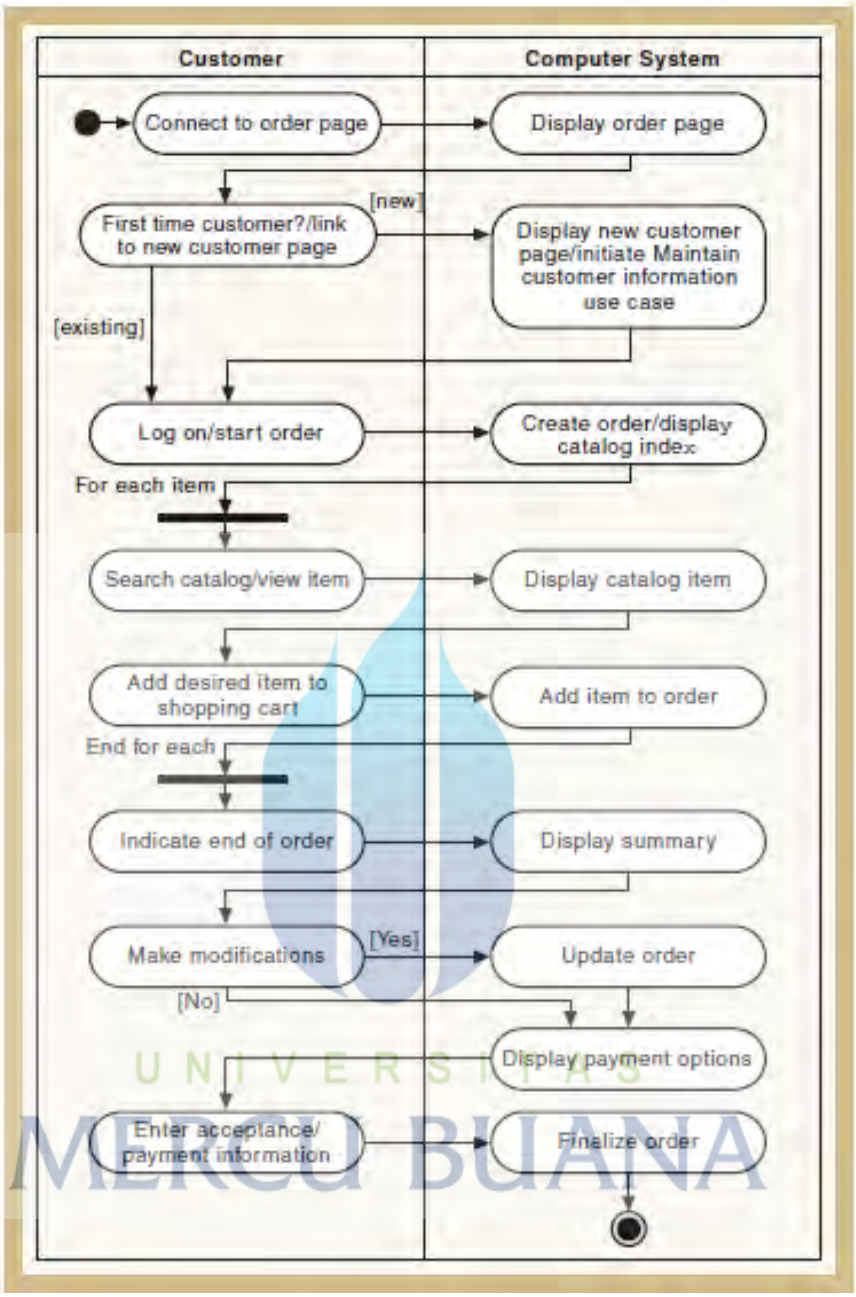
Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, 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.
4. Rancangan menu yang akan ditampilkan pada perangkat lunak. (Wiley, 2010).

Tabel 3.3 Simbol – simbol Activity Diagram (Willey, 2010)

SIMBOL	NAMA	KETERANGAN

	Activity	merupakan sebuah gambaran aktivitas yang terjadi.
	Swimlane	memisahkan organisasi bisnis yang bertanggung jawab terhadap aktifitas yang terjadi.
	Initial Node	merupakan tanda awal dari sebuah aktivitas.
	Actifity Final Node	merupakan tanda berakhirnya sebuah aktivitas.
	*Decision Node*	Pilihan untuk pengambilan keputusan
	Fork	Digunakan untuk menunjukkan kegiatan – kegiatan yang dilakukan secara paralel atau untuk menggabungkan dua kegiatan paralel menjadi satu
	*Control Flow*	Menunjukkan urutan eksekusi
	Object Flow	Menunjukkan aliran objek dari satu kegiatan (atau tindakan) untuk kegiatan lain (atau tindakan)..



Gambar 2.4 Contoh Activity Diagram (Satzinger, 2010)

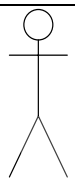
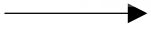
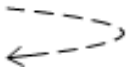
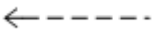
2.7.6 Sequence Diagram

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

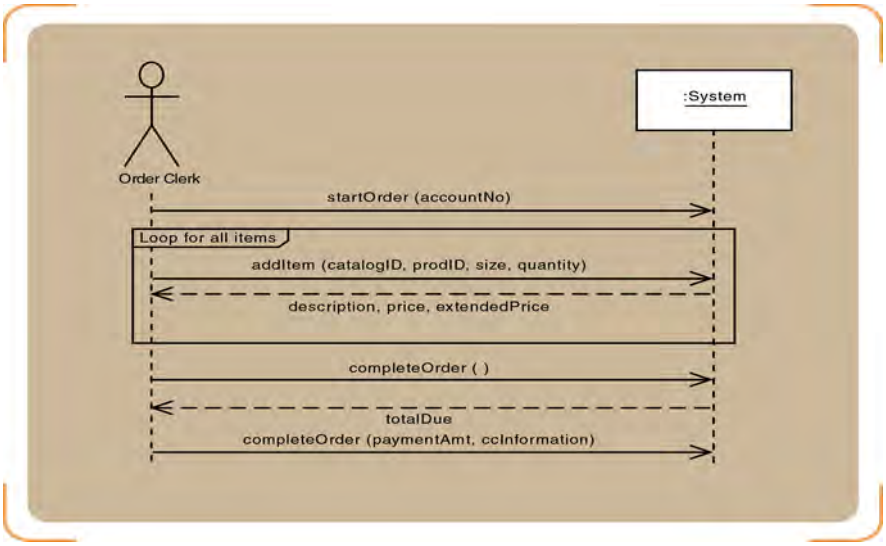
yang dimiliki kelas yang diinstansiasi menjadi objek itu. Banyaknya diagram sekuen yang harus digambar adalah minimal sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak. (Wiley, 2010).

Berikut simbol yang ada pada diagram sequence:

Tabel 4.4 Simbol – simbol Sequence Diagram (Willey, 2010)

SIMBOL	NAMA	KETERANGAN
	Actor	orang atau sistem yang berasal dari manfaat dan eksternal ke sistem yang berpartisipasi secara berurutan dengan mengirim dan / atau menerima pesan
	LifeLine	Menyatakan kehidupan suatu objek.
	*Execcution Occurrence*	Menyatakan objek dalam keadaan aktif dan berinteraksi pesan.
	Message	Pesan yang mengambarkan komunikasi yang terjadi antar objek.
	*Message (return)*	Pesan yang dikirim untuk diri sendiri secara langsung.
	*Message (return)*	Pesan yang dikirim untuk diri sendiri.

Sistem *Sequence Diagram* digunakan dalam hubungan dengan perincian *descriptions* atau dengan diagram akitivitas untuk menunjukan langkah proses dan interaksi antara sistem dengan actor.



Gambar 2.5 Contoh Sistem Sequence Diagram (Satzinger, 2010)

2.7.7 Class Diagram

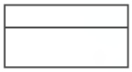
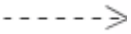
Diagram kelas merupakan kumpulan kelas-kelas objek. Oleh karena itu pengertian kelas sangat penting sebelum merancang diagram kelas. (Whitten, 2004:410) mengartikan kelas sebagai satu set objek yang dimiliki atribut dan perilaku yang sama. Kelas kadang-kadang disebut kelas objek (*object class*). Secara alami, objek yang berupa buku analisis disain dan buku pemrograman terstruktur kita kelompokan dalam satu kelas, yaitu kelas buku.Kedua objek memiliki atribut dan perilaku yang serupa.Contohnya, kedua objek mungkin memiliki atribut yang serupa seperti nomor ISBN, Judul, tanggal penerbitan, edisi, dan sebagainya. Demikian juga, kedua objek memiliki perilaku yang sama misalnya membuka dan menutup.

Secara teknis, (Pender, 2003: bab 5) mengartikan sebuah kelas sebagai suatu definisi sumber daya yang termasuk di dalamnya informasi-informasi yang menggambarkan fitur suatu entitas dan bagaimana penggunaannya. Sedangkan objek adalah entitas yang bersifat unik yang mengikuti aturan-aturan yang sudah didefinisikan dalam kelasnya.

Kelas menggambarkan suatu group yang memiliki kesamaan keadaan dan perilaku.Kelas merupakan cetak biru suatu objek dalam sistem orientasi objek.Dapat dikatakan kelas adalah sejenis alat pengklasifikasi. Sebagai contoh

Volkswagen, Toyota dan ford merupakan kumpulan mobil sehingga kita dapat mengkelompokan dalam kelas yang diberi nama mobil. Seuatu kelas bisa menyatakan konsep yang dapat dilihat maupun abstrak (Pilone, 2005:bab 2).

Tabel 5.5 Fungsi Simbol Class Diagram

No	Simbol	Keterangan
1		***Generalization*** Target dari hubungannya ke kelas yang lebih general (umum).
2		***Agregation***versi kuat dari asosiasi, agregrasi mengimplikasikan kepemilikian suatu kelas.
3		***Composition*** Hubungan yang paling kuat antar kelas. Komposisi digunakan untuk mengambil seluruh bagian hubungan.
4		***Class***menggambarkan suatu group yang memiliki kesamaan keadaan dan perilaku.
5		***Dependency***Hubungan ketergantungan (*dependency relationship*). Ketergantungan antar kelas bermakna suatu kelas menggunakan atau memiliki pengetahuan terhadap kelas lainnya.
6		***Association***Hubungan pada asosiasi lebih kuat dari hubungan ketergantungan dalam arti suatu kelas tetap berhubungan dengan kelas lain seterusnya.

2.7.8 User Interface

user interface melibatkan *input* dan *output* yang melibatkan pengguna sistem. Sebuah *user interface* memungkinkan pengguna untuk berinteraksi dengan komputer untuk merekam transaksi. (Satzinger, 2010).

a. *Physiscal aspect of the user interface*

Aspek fisik dari interaksi manusia dan komputer meliputi peralatan yang benar-benar bersentuhan dengan pengguna, termasuk *keyboard, mouse,touch screen,* atau *keypad.*

b. *Perceptual aspect of teh user interface*

Aspek *perceptual* dari interaksi manusia dan komputer termasuk semua yang pengguna akhir lihat, dengar, atau sentuh (melampaui aspek fisik).

c. *Conceptual aspect of the user interface*

Aspek konseptual dari interaksi manusia dan komputer meliputi semua yang pengguna ketahui mengenai penggunaan sitem, termasuk semua masalah utama yang dimanipulasi oleh user, operasi-operasi yang dapat ditampilkan, dan prosedur penggunaanya.

2.8 SQL

Didalam sebuah sistem terdapat basis data yang sangat banyak maka diperlukan suatu cara atau teknik untuk komunikasi dengan basis data tersebut. SQL (yang biasa dibaca dengan Sequel) adalah singkatan dari "*Structured Query Language*" yaitu suatu bahasa standar yang digunakan untuk memanipulasi dan memperoleh data dari sebuah basis data yang saling berelasi.SQL merupakan bahasa yang kuat dan dengan kepandaian menggunakan unsur – unsur bahasa tersebut, seorang database administrator dapat melakukan pengoperasian basis data yang kompleks dan sulit. Secara umum SQL dapat dibagi menjadi 2, yaitu Data Definition Language (DDL) dan Data Manipulation Language (DML).

Data Definition Language (DDL) adalah suatu sub bahasa SQL yang digunakan untuk membangun kerangka sebuah basis data atau bisa juga merupakan suatu fungsi yang digunakan untuk mendefinisikan atribut – atribut basis data, table, atribut kolom, dan batasan – batasan terhadap atribut serta hubungan antar tabel. Perintah SQL yang termasuk dalam DDL diantaranya :

1. ***CREATE***

   Perintah ini digunakan untuk membuat sebuah database baru, tabel baru, dan kolom.

   Contoh : *CREATE DATABASE* Penjualan

2. ***ALTER***

   Perintah ini digunakan untuk mengubah struktur terhadap atribut – atribut yang terdapat didalam suatu database, tabel, maupun kolom yang sudah ada pada basis data tersebut.

   Contoh : *ALTER TABLE* Transaksi *ADD* Tanggal *DATE*

3. ***DROP***

   Perintah ini digunakan melakukan penghapusan terhadap database ataupun tabel yang telah ada pada database tersebut.

   Contoh : *DROP DATABASE* Penjualan

Data Manipulation Language (DML) adalah suatu sub bahasa SQL yang digunakan untuk melakukan manipulasi terhadap data yang terdapat dalam basis data tersebut.Perintah SQL yang termasuk dalam DML diantaranya :

1. ***SELECT***

   Perintah ini digunakan untuk melakukan pengambilan atau menampilkan data yang diinginkan pada database tersebut.

   Contoh : *SELECT* Regional *FROM* Transaksi

2. ***INSERT***

   Perintah ini digunakan untuk melakukan penyisipan atau masukan data ke dalam tabel sebuah database.

   Contoh : *INSERT INTO* Transaksi *VALUES* (‘Sumatera Utara’,’Medan’)

3. ***UPDATE***

   Perintah ini digunakan untuk melakukan pengupdatean terhadap data yang terdapat di dalam tabel tersebut.

   Contoh : *UPDATE* Transaksi *SET* kota = ‘Aceh’ *WHERE* regional = ‘Sumatera Utara’

4. ***DELETE***

   Perintah ini digunakan untuk melakukan penghapusan terhadap data yang terdapat di dalam tabel tersebut.

   Contoh : *DELETE* kota *FROM* Transaksi

2.9 Perangkat Lunak Pendukung

2.9.1 HTML (*hyper text markup language*)

HTML (*hyper text markup language*) adalah bahasa standar pemograman untuk membuat halaman web yang terdiri dari kode-kode tag tertentu, kemudian kode-kode tersebut diterjemahkan oleh web browser untuk menampilkan halaman web yang terdiri dari berbagai macam format tampilan seperti teks, grafik, animasi link, maupun audio-video.

2.9.2 CSS (*Cascading Style Sheets*)

CSS adalah kepanjangan dari *Cascading Style Sheets*. CCS skrip yang berisi rangkaian instruksi yang menentukan suatu teks akan tertampil di halaman *web browser*. Perancangan tampilan web dapat dilakukan dengan mendefinisikan *font* (huruf), *color* (warna), margin (ukuran), latar belakang (*background*), dan ukuran *font (font size)* dan lain-lain. Elemen-elemen seperti *color* (warna), *font* (huruf), *sizes*(ukuran), dan *spacing* (jarak) disebut juga "styles".

2.9.3 Java Script

Bahasa pemograman java, perintah-perintahnya ditulis dengan kode yang disebut skrip. Java adalah bahasa pemograman berorientasi objek, sedangkan *script* adalah serangkaian instruksi program. Ada beberapa hal yang harus diperhatikan dalam pengelolaan pemograman JavaScript, diantaranya JavaScript mengenal kode secara *case* sensitive, yang artinya JavaScript membedakan huruf besar dan huruf kecil, jika Anda pernah belajar bahasa pemograman seperti Turbo C atau C++, php.

2.9.4 ASP.NET

ASP.NET Merupakan teknologi dari Microsoft yang dikhususkan untuk pengembangan aplikasi berbasis web dinamis berbasis pada platform .Net Framework. ASP.NET didesain untuk memberikan kemudahan pada pengembang web untuk membuat aplikasi berbasis web dengan cepat, mudah, dan efisien karena meminimalkan penulisan kode program dengan bantuan komponen – komponen yang sudah disediakan sehingga dapat meningkatkan produktifitas.

2.9.5 DEVELOPMENT EXPRESS (DEV EXPRESS)

DevExpress atau *Development Express* merupakan sebuah komponen *control* tambahan pada ASP.NET yang digunakan untuk menambahkan fitur, meringankan tugas, dan menghemat waktu pemakai/programmer dalam membuat suatu program.

DevExpress menawarkan pula seperangkat komponen presentasi dan pelaporan serta menambahkan beberapa fungsi, karena itulah setiap situs *web* yang menggunakan ASP.NET banyak menggunakan aplikasi tambahan ini karena kemudahannya untuk memperindah tampilan tanpa memasukan *coding* yang sangat banyak.

2.9.6 .NET Framework

ASP.NET di atas *platform* .NET Framework oleh karena itu kita harus tahu sedikit mengenai *platform* ini. .NET Framework sebenernya adalah satu set kumpulan teknologi dari Microsoft yang ditujukan untuk membantu pengembang untuk mengembangkan aplikasi secara aman, mudah, efisien, dan produktif.

.NET Framework mendukung beberapa bahasa pemrograman, ada pun bahasa pemrograman yang di-support secara resmi oleh Micorosft adalah C# (Csharp), VB, dan C++ tetapi sekarang banyak bahasa lain yang juga dikembangkan untuk men-s*upport platform* .NET di antaranya Ruby (IronRuby), Phyton (IronPhyton), dll. Untuk mengembangkan aplikasi berbasis .NET sebenarnya dapat digunakan lebih dari satu bahasa pemrograman (Language Interoperability) misal sebagian program menggunakan C# dan sebagian lagi menggunakan VB, tetapi disarankan untuk memilih hanya satu bahasa pemrograman saja agar aplikasi yang dibuat lebih mudah untuk di-*maintain*. Bahasa paling banyak digunakan di *platform* .NET saat ini adalah C# dan VB.

.NET Framework sebenarnya terdiri dari dua komponen utama, yaitu CLR (*Common Language Runtime*) dan BCL (*Base Class Library*).

2.9.7 C# (C-Sharp)

Bahasa pemrograman C# (baca: C-Sharp) dirancang oleh Microsoft Corp. sebagai bahasa pemrograman yang sangat berdaya-guna, aman (*secure*), serta mudah digunakan. Sebagai bagian dari platform .NET, bahasa pemrograman C# dirancang sedemikian rupa untuk bekerja dengan sangat baik diataas framework .NET yang mampu digunakan untuk menulis perangkat lunak handal demi layanan yang cepat. Bahasa pemrograman C# juga dapat digunakan untuk mengembangkan aplikasi sarana bergerak (mobile application), aplikasi berbasis Web (*Web-based applications*), serta aplikasi – aplikasi berkala besar (*enterprise*).

2.9.8 Microsoft Visual Studio

Untuk membuat aplikasi ASP.NET sebenernya Anda bisa menggunakan sembarang editor seperti Notepad, tetapi untuk mempercepat dan mempermudah pengembangan program maka Micorsoft menyediakan IDE (Integrated Development Environment) yang sangat handal, yaitu Visual Studio.

Visual Studio mempunyai banyak sekali fitur yang memanjakan programmer seperti drag and drop designer, Intellisense, syntax checking, integrated compilation, debugging dan banyak fitur lain yang akan meningkatkan produktifitas programmer dalam mengembangkan aplikasi. Versi gratis dari Visual Studio adalah Visual Studio Express.

2.9.9 Web Server

Untuk menjalankan aplikasi ASP.NET 4.5 dibutuhkan program aplikasi yang diberi nama web server. Web Server yang biasa digunakan merupakan bawaan dari windows (biasanya ada pada Windows XP Profesional Edition, Windows 2003 Server, Windwos 2008 Server, Windows Vista, Windows 7 Professional, atau Windows 8 Profesional), yaitu IIS (Internet Information Services).

Untuk mempermudah instalasi IISm . NET Framework, dan modul – modul yang lain pada komputer server Anda dapat menggunakan Web Platform Installer (Web PI) yang disediakan oleh Microsoft. Dengan Web PI Anda dapat memilih aplikasi dan modul yang akan digunakan

2.9.10 SQL Server

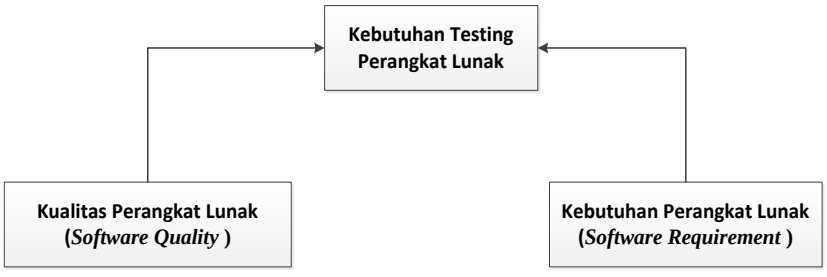
SQL Server merupakan versi gratis dari SQL Server, Anda dapat menggunakan dan mendistribusikan database ini secara gratis. Biar pun gratis, namun databse ini sangat handal dan memiliki beberapa keunggulan :

- Mudah untuk di-install;
- Menyediakan tool yang sederhana untuk manajemen database;
- Mendukung windows authentication;
- "Secure by default" setting;
- Dapat didistribusikan secara gratis;
- Memiliki fungsionalitas yang lengkap meliputi trigger, stored procedure, function, extended indexes, transact SQL;
- Dukungan terhadap XML;
- Sangat terintegrasi dengan Visual Studio.

2.10 Metode Pengujian

Pengujian perangkat lunak adalah sebuah proses yang diejawantahkan sebagai siklus hidup dan merupakan bagian dari proses rekayasa perangkat lunak secara terintegrasi demi memastikan kualitas dari perangkat lunak serta memenuhi kebutuhan teknis yang telah disepakati dari awal(Rizky, 2011:237). Alasan utama yang mengapa pengujian (*testing*) perangkat lunak harus dilakukan adalah berdasarkan atas dua hal yaitu :

1. *Software Requirement* atau kebutuhan data yang harus dipenuhi perangkat lunak
2. *SoftwareQuality* atau kualitas perangkat lunak



Gambar 3.6 Kualitas perangkat lunak

2.10.1 Metode *White Box*

White Box testing secara umum merupakan jenis *testing* yang lebih berkonsentrasi terhadap 'isi' dari perangkat lunak itu sendiri. Jenis ini lebih banyak berkonsentrasi kepada *source code* dari perangkat lunak yang dibuat sehingga membutuhkan proses testing yang jauh lebih lama dan lebih 'mahal' dikarenakan membutuhkan ketelitian dari para *tester* serta kemampuan teknis pemrograman bagi para *tester*-nya (Rizky, 2011:261).

Prinsip dari keluaran tipe testing ini adalah:

1. Menjamin bahwa semua alur program yang independen (dalam bentuk modul, *form*, *procedure*, *class*, dan lainnya) telah dites minimal satu kali.
2. Telah melakukan testing terhadap semua kondisi percabangan dengan nilai *true* dan *false.*
3. Telah melakukan testing terhadap semua jenis perulangan dengan kondisi normal dan kondisi yang dianggap melampaui batas perulangan (umumnya kondisi yang melampaui batas harus diatasi oleh prosedur tertentu)
4. Telah melakukan testing terhadap struktur data internal (seperti variabel) agar terjaga validasinya.

Beberapa teknik yang terdapat dalam jenis *white box* testing antara lain:

1. *Decision(branch) Coverage*

   Teknik testing ini focus terhadap hasil dari setiap skenario yang dijalankan terhadap bagian perangkat lunak yang mengandung percabangan (*if ... then ... else*).
2. *ConditionCoverage*

   Teknik ini hampir mirip dengan teknik *Decision(branch) Coverage,* tetapi dijalankan terhadap percabangan yang dianggap kompleks atau percabangan majemuk.
3. *PathAnalysis*

   Teknik testing ini berusaha menjalankan kondisi yang ada dalam perangkat lunak serta berusaha mengoreksi apakah kondisi yang dijalankan telah sesuai dengan alur diagram yang terdapat dalam proses perancangan.
4. *Executiontime*

Pada teknik ini, perangkat lunak berusaha dijalankan atau dieksekusi kemudian dilakukan pengukuran waktu pada saat inpu dimasukkan hingga input dikeluarkan. Waktu eksekusi yang dihasilkan kemudian dijadikan bahan evaluasi dan dianalisa lebih lanjut untuk melihat apakah perangkat lunak telah berjalan sesuai dengan kondisi yang dimaksud oleh tester.

5. *AlgorithmAnalysis*

   Teknik ini pada umumnya jarang dilakukan jika perangkat lunak yang dibuat berjenis sistem informasi. Sebab tekni ini membutuhkan kemampuan matematis yang cukup tinggi dari para tester karena didalamnya berusaha melakukan analisa terhadap algoritma yang diimplementasikan pada perangkat lunak.

2.10.2 Metode *Black Box*

Black Boxtesting adalah tipe testing yang memperlakukan perangkat lunak yang tidak diketahui kinerja internalnya. Sehingga para *tester* memandang perangkat lunak seperti layaknya sebuah 'kotak hitam' yang tidak penting dilihat isinya, tapi cukup dikenai proses testing di bagian luarnya. Jenis testing ini hanya memandang perangkat lunak dari sisi spesifikasi dan kebutuhan yang telah didefinisikan pada saat awal perancangan (Rizky, 2011:264).

Beberapa keuntungan yang diperoleh dari jenis testing ini antara lain:

1. Anggota tim *tester* tidak harus dari seseorang yang memiliki kemampuan teknis di bidang pemrograman
2. Kesalahan dari perangkat lunak ataupun '*bug'* seringkali ditemukan oleh komponen *tester* yang berasal dari pengguna.
3. Hasil dari *black box testing* dapat memperjelas kontradiksi ataupun kerancuan yang mungkin timbul dari eksekusi sebuah perangkat lunak.
4. Proses testing dapat dilakukan lebih cepat disbanding dengan *white box testing.*

Beberapa teknik testing yang tergolong dalam tipe ini antara lain :

1. *EquivalencePartitioning*

Pada teknik ini, setiap inputan data dikelompokkan dalam grup tertentu, yang kemudian dibandingkan outputnya.

2. *BoundaryValue Analysis*

   Pada teknik ini, dilakukan inputan yang melebihi dari batasan sebuah data.

3. *CauseEffect Graph*

   Dalam teknik ini, dilakukan proses testing yang menghubungkan sebab dari sebuah inputan dan akibatnya pada output yang dihasilkan.

4. *RandomData Selection*

   Teknik ini berusaha melakukan proses inputan data dengan menggunakan nilai acak. Dari hasil inputan tersebut kemudian dibuat sebuah tabel yang menyatakan validitas dari output yang dihasilkan.

5. *FeatureTest*

   Pada teknik ini, dilakukan proses testing terhadap terhadap spesifikasi dari perangkat lunak yang telah selesai dikerjakan.