BAB II

TINJAUAN PUSTAKA

II.1. Sistem

Sistem adalah kumpulan dari elemen-elemen yang berinteraksi untuk mencapai suatu tujuan tertentu (Jogiyanto HM ; 2005 : 2)

Sistem adalah sebagai suatu kumpulan atau himpunan dari unsur, komponen, atau variabel yang terorganisir, saling berinteraksi, saling bergantung satu sama lain, dan terpadu (Tata Sutabri ; 2005 : 2)

II.1.1. 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 memiliki karakteristik atau sifat-sifat tertentu, yang mencirikan bahwa hal tersebut bisa dikatakan sebagai suatu sistem. Adapun karakteristik yang dimaksud adalah sebagai berikut (Tata Sutabri ; 2005 :11) :

1. **Komponen Sistem (*Component*)**

   Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, artinya saling berkerja sama membentuk satu kesatuan. Komponen-komponen sistem tersebut dapat berupa suatu bentuk subsistem. Setiap subsistem memiliki sifat dari sistem yang menjalankan suatu fungsi

tertentu dan mempengaruhi proses sistem secara keseluruhan. Suatu sistem dapat mempunyai sistem yang lebih besar, yang disebut “ supra sistem ”.

2. **Batasan Sistem *(Boundary).***

Ruang lingkup sistem merupakan daerah yang membatasi antara sistem dengan sistem yang lain atau sistem dengan lingkungan luarnya. Batasan sistem ini memungkinkan suatu sistem dipandang sebagai satu kesatuan yang tidak dapat dipisah-pisahkan.

3. **Lingkungan Luar Sistem *(Environment)*.**

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

4. **Penghubung Sistem *(Interface)***

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

5. **Masukan Sistem *(Input)***

Energi yang dimasukan ke dalam sistem disebut masukan sistem, yang dapat berupa pemeliharaan (*maintenance input*) dan sinyal (*signal input*). Contohnya di dalam suatu sistem unit komputer. “ Program ” adalah *maintenance* input yang digunakan untuk mengoperasikan komputernya dan “ data ” adalah *signal input* untuk diolah menjadi informasi.

6. **Keluaran Sistem (*Output*)**

Hasil dari energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna. Keluaran ini merupakan masukan bagi subsistem yang lain. Contoh, sistem informasi. Keluaran yang dihasilkan adalah informasi. Informasi ini dapat digunakan sebagai masukan untuk pengambil keputusan atau hal-hal lain yang menjadi *input* bagi subsistem lain.

7. **Pengolah Sistem (*Proses*).**

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

8. **Sasaran Sistem (*Objective*)**

Suatu sistem memiliki tujuan dan sasaran yang pasti dan bersifat *deterministik*. Kalau suatu sistem tidak memiliki sasaran, maka operasi sistem tidak ada gunanya. Suatu sistem dikatakan berhasil bila mengenai sasaran atau tujuan yang telah direncanakan (Tata Sutabri ; 2005 : 11-12).

II.1.2. Klasifikasi Sistem

Sistem merupakan suatu bentuk integrasi antara satu komponen dengan komponen lain karena sistem memiliki sasaran yang berbeda setiap kasus yang terjadi yang ada di dalam sistem tersebut. Oleh karena itu sistem dapat diklasifikasikan dari beberapa sudut pandangannya antara lain (Tata Sutabri ; 2005 : 13) :

1. Sistem abstrak dan sistem fisik

Sistem abstrak adalah sistem yang berupa pemikiran atau ide-ide yang tidak tampak secara fisik, misalnya sistem teologia, yaitu sistem yang berupa pemikiran hubungan antara manusia dengan tuhan, sedangkan sistem fisik merupakan sistem yang ada secara fisik, misalnya sistem komputer, sistem produksi, sistem penjualan, sistem administrasi personalia, dan lain sebagainya.

2. 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 malam, pergantian musim. Sedangkan sistem buatan manusia dengan mesin, merupakan melibatkan interaksi manusia dengan mesin, yang disebut " *human machine system* ". Sistem informasi berbasis komputer merupakan contoh *human machine system* karena menyangkut penggunaan komputer yang berinteraksi dengan manusia.

3. Sistem deterministik dan sistem probabilistik.

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

4. Sistem terbuka dan sistem tertutup.

Sistem tertutup merupakan sistem yang tidak berhubungan dan tidak terpengaruh oleh lingkungan luarnya. Sistem ini bekerja secara otomatis tanpa campur tangan pihak luar. Sedangkan sistem terbuka adalah sistem yang berhubungan dan dipengaruhi oleh lingkungan luarnya. Sistem ini menerima masukan dan menghasilkan keluaran untuk subsistem lainnya (Tata Sutabri ; 2005 : 13).

Dari pendapat diatas sistem adalah sekumpulan elemen yang saling terkait atau terpadu yang dimaksudkan untuk mencapai suatu tujuan.

II.1.3. Daur Hidup Sistem

Siklus hidup sistem (*system life cycle*) adalah merupakan proses evolusioner yang diikuti dalam menerapkan sistem atau subsistem informasi komputer. Siklus hidup sistem terdiri dari serangkaian tugas yang erat mengikuti langkah-langkah pendekatan sistem sistem karena tugas-tugas tersebut mengkuti pola yang teratur dan dilakukan secara *top down*. Siklus hidup sistem sering

disebut sebagai pendekatan air terjun (*waterfall approach*) bagi pembangunan dan pengembangan sistem.

Pembangunan sistem hanyalah salah satu dari rangkaian daur hidup suatu sistem. Meskipun demikian, proses ini merupakan aspek yang sangat penting. Kita akan melihat beberapa fase/tahapan dari daur hidup suatu sistem (Tata Sutabri ; 2005 : 14).

1. Mengenali adanya kebutuhan.

Sebelum segala sesuatunya terjadi, timbul suatu kebutuhan atau problema yang harus dapat dikenali sebagaimana adanya. Kebutuhan dapat terjadi sebagai hasil perkembangan dari organisasi dan volume yang meningkat melebihi kapasitas dari sistem yang ada. Semua kebutuhan ini harus dapat didefenisikan dengan jelas. Tanpa adanya kejelasan dari kebutuhan yang ada, pembangunan sistem akan kehilangan arah dan efektifitasnya.

2. Pembangunan sistem

Suatu proses atau seperangkat prosedur yang harus diikuti untuk menganalisis kebutuhan yang timbul dan membangun suatu sistem untuk dapat memenuhi kebutuhan tersebut.

3. Pemasangan sistem

Setelah tahap pembangunan sistem selesai, sistem kemudian akan dioperasikan. Pemasangan sistem merupakan tahap yang penting pula dalam daur hidup sistem. Peralihan dari tahap pembangunan menuju tahap operasional terjadi pemasangan sistem yang sebenarnya, yang akan merupakan langkah akhir dari suatu pembangunan sistem.

4. Pengoperasian sistem

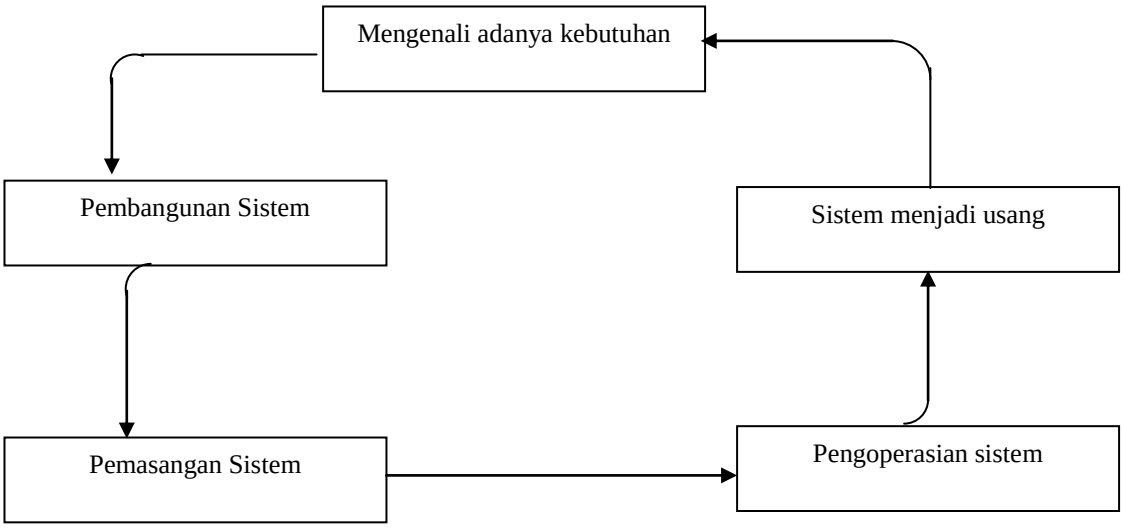
Program-program komputer dan prosedur-prosedur pengoperasian yang membentuk suatu sistem informasi semuanya bersifat statis. Sedangkan organisasi ditunjang oleh sistem informasi tadi. Program komputerdan prosedur perancangan tersebut selalu mengalami perubahan-perubahan, itu karena pertumbuhan kegiatan bisnis, perubahan pengaturan, dan kebijaksanaan ataupun kemajuan teknologi. Untuk mengatasi perubahan-perubahan tersebut, sistem harus diperbaiki atau diperbaharui.

5. Sistem menjadi usang

Kadang perubahan yang terjadi begitu drastis, sehingga tidak dapat diatasi hanya dengan melakukan perbaikan-perbaikan pada sistem yang berjalan. Tibalah saatnya secara ekonomis dan teknis sistem yang ada sudah tidak layak lagi untuk dioperasikan dan sistem yang baru perlu dibangun untuk menggantikannya (Tata Sutabri ; 2005 : 14 - 15).

Sistem informasi kemudian akan melanjutnya daur hidupnya. Sistem dibangun untuk memenuhi kebutuhan yang muncul. Sistem beradaptasi terhadap perubahan-perubahan lingkungannya dinamis. Sampailah pada kondisi dimana sistem tersebut tidak dapat lagi beradaptasi dengan perubahan-perubahan yang ada atau secara ekonomis tidak layak lagi untuk dioperasikan. Sistem yang baru kemudian dibangun untuk menggantikanya.

Untuk dapat menggambarkan daur hidup sistem ini, lihat pada gambar II.1. sebagai berikut (Tata Sutabri ; 2005 : 15).



Gambar II.1. Daur Hidup Sistem

Sumber : Tata Sutabri (2005 : 15)

II.1.3.1. Informasi

Informasi adalah data yang diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi yang menerimanya (Jogiyanto, HM ; 2005 : 8).

Informasi adalah data yang telah diklasifikasikan diolah atau diinterpretasi untuk digunakan dalam proses pengambil keputusan. Sistem pengolahan informasi megolah data menjadi informasi atau tepatnya mengolah dari bentuk tak berguna menjadi berguna bagi penerimanya (Tata Sutabri ; 2005 : 23).

Dari pendapat diatas informasi adalah data yang diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi yang menerimanya.

II.1.3.2. Sistem Informasi

Sistem informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari suatu organisasi dan menyediakan

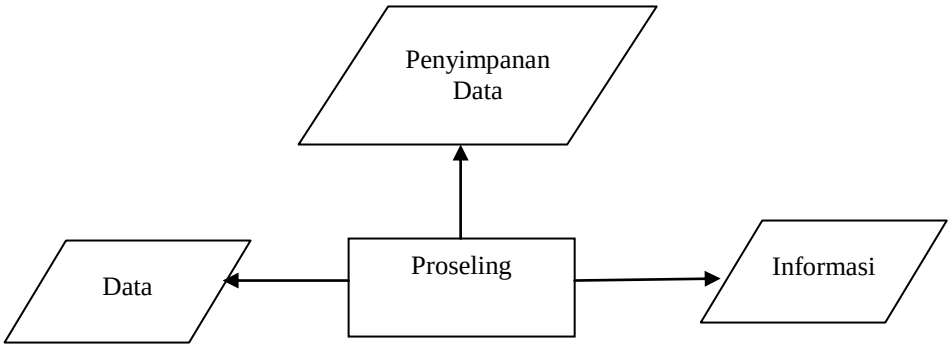
pihak luar tertentu dengan laporan-laporan yang diperlukan (Jogiyanto, HM ; 2005 : 11).

Sistem informasi adalah berupa suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan data transaksi harian yang mendukung operasi yang bersifat manajerial dengan kegiatan strategi suatu organisasi untuk dapat menyediakan kepada pihak luar tertentu dengan laporan-laporan yang diperlukan (Tata Sutabri ; 2005 :42).

Dari pendapat diatas sistem informasi adalah sebuah rangkaian prosedur formal di mana data atau dikelompokan, diproses menjadi informasi, dan didistribusikan kepada pemakai.

II.1.3.3. Data

Mengenai pengertian data, lebih jelas apa yang didefenisikan oleh Drs. Jhon J. Longkutoy dalam bukunya " Pengenalan Komputer " sebagai berikut : isitilah data adalah suatu istilah majemuk yang berarti fakta atau bagian dari fakta yang mengandung arti yang dihubungkan dengan kenyataan simbol-simbol, gambar-gambar, angka-angka, huruf-huruf, atau simbol-simbol yang menunjukan suatu ide, objek, kondisi, atau situasi dan lain-lain (Tata Sutabri ; 2005 : 16).



Gambar II.2. Pemrosesan Data

Sumber : Tata Sutabri (2005 : 16)

Dari pendapat diatas data adalah deskripsi tentang benda, kejadian aktivitas, dan transaksi, yang tidak mempunyai makna atau tidak berpengaruh secara langsung kepada pemakai.

II.1.3.4. Sistem Informasi Akuntansi

Akuntansi merupakan bahasa dari bisnis. Setiap perusahaan menerapkannya sebagai alat komunikasi. Secara klasik akuntansi merupakan proses pencatatan (*recording*), pengkelompokkan (*classifiying*), perangkuman (*summarizing*) dan pelaporan (*reporting*) dari kegiatan transaksi perusahaan. Tujuan akhir dari kegiatan akuntansi adalah penerbitan laporan-laporan keuangan. Laporan-laporan keuangan adalah merupakan suatu informasi. Sistem informasi yang berbasis pada komputer sekarang dikenal dengan istilah sistem informasi akuntansi atau SIA (*accounting information system* atau AIS). Sistem informasi akuntansi SIA didefenisikan oleh Stephen A. Moscove dan Mark G. Simkin sebagai berikut ini. SIA adalah merupakan komponen operasi yang mengumpulkan, mengklasifikasikan, memproses, menganalisis, mengkomunikasikan informasi pengambilan keputusan dengan orentasi financial yang relevan bagi pihak-pihak

dalam perusahaan (secara prinsip adalah manajemen). Menurut Robert G. Murdick, Thomas C. Fuller dan Joel E. Ross : SIA adalah kumpulan kegiatan - kegiatan dari organisasi yang bertanggung jawab untuk menyediakan informasi keuangan dan informasi yang didapatkan dari transaksi data untuk tujuan pelaporan internal kepada manajer untuk digunakan dalam pengendalian dan perencanaan sekarang dan operasi masa depan serta pelaporan eksternal kepada pemegang saham, pemerintah dan pihak - pihak luar lainnya (Jogiyanto, HM ; 2005 : 17)

II.2. Laporan Arus Kas Masuk Dan Kas Keluar

Laporan keuangan dasar yang telah kami sajikan sejauh ini hanya menyediakan informasi mengenai arus kas perusahaan (penerimaan kas dan pembayaran kas). Sebagai contoh neraca komperatif menunjukkan kenaikan asset tetap selama setahun tersebut, tetapi tidak menunjukkan laba bersih, tetapi tidak mengindikasikan jumlah kas yang dihasilkan dari aktivitas operasi. Laporan saldo laba menunjukkan deviden kas yang diumumkan tetapi tidak dividen kas yang dibayarkan pada tahun tersebut. Tidak ada satu pun dari laporan ini menyajikan ringkasan mengenai dari mana datangnya kas dan bagaimana penggunaannya.

II.2.1. Kegunaan Laporan Arus Kas

Laporan arus kas (*Statement Of Cash Flow*) melaporkan penerimaan kas, pembayaran kas dan perubahan bersih pada kas yang dihasilkan dari aktivitas operasi, aktivitas investasi, dan pendanaan selama satu periode. Informasi yang

terdapat pada laporan arus kas harus dapat membantu para investor, kreditor, dan lainnya untuk menilai :

1. Kemampuan entitas dalam memperoleh arus kas di masa depan. Dengan memeriksa hubungan antar pos pada laporan arus kas, para investor dan pihak lainnya dapat membuat prediksi mengenai jumlah, waktu, dan ketidakpastian mengenai arus kas di masa depan dengan lebih baik dibandingkan jika mereka menggunakan data akrual.
2. Kemampuan entitas untuk membayar deviden dan memenuhi kewajiban. Jika sebuah perusahaan tidak memiliki cukup kas, mereka tidak dapat membayar karyawan melunasi utang, atau membayar deviden. Para karyawan, kreditor, dan pemegang saham umumnya tertarik terutama pada laporan laporan ini, karena laporan ini, karena laporan ini sendiri menunjukkan arus kas dalam kegiatan bisnis.
3. Alasan atas perbedaan antara angka laba bersih dan kas bersih yang dihasilkan (digunakan) oleh aktivitas operasi. Laba bersih menyediakan informasi mengenai keberhasilan atau kegagalan sebuah perusahaan bisnis. Meski demikian beberapa pihak mengkritik laba bersih berbasis akrual, karena membutuhkan banyak perkiraan. Hasilnya, keandalan dari angka tersebut sering dipertanyakan. Hal tersebut tidak terjadi pada kas. Banyak pembaca dari laporan arus kas ingin mengetahui alasan adanya perbedaan antara laba bersih dan kas bersih yang dihasilkan oleh aktivitas operasi. Kemudian mereka dapat menilai sendiri keandalan jumlah laba tersebut.

4. Transaksi- transaksi investasi dan pendanaan kas selama periode tersebut. Dengan memeriksa transaksi-transaksi investasi dan pendanaan sebuah perusahaan, pembaca laporan keuangan dapat mengerti dengan lebih baik mengapa asset dan kewajiban berubah selama periode tersebut (Jerry J. Weygandt, dkk ; 2008 : 323-324).

II.2.2. Menyusun Laporan Arus Kas

Laporan arus kas disusun secara berbeda dari tiga jenis laporan keuangan dasar lainnya. Pertama, laporan ini tidak disusun dari neraca saldo yang disesuaikan. Laporan ini memerlukan informasi terinci mengenai perubahan pada saldo akun yang terjadi di dua titik waktu. Kedua, laporan arus kas berhubungan dengan penerimaan dan pembayaran kas. Hasilnya, pengaruh dari penggunaan akuntansi akrual harus disesuaikan untuk menentukan arus kas.

Informasi yang disiapkan untuk laporan ini umumnya datang dari tiga sumber :

1. Neraca komparatif. Informasi pada neraca komparatif menunjukkan jumlah perubahan pada asset, kewajiban, dan ekuitas pemegang saham dari awal hingga akhir periode.
2. Laporan laba rugi saat ini. Informasi pada ini membantu menentukan jumlah kas yang dihasilkan atau digunakan oleh operasi selama periode tersebut.
3. Informasi tambahan. Informasi ini termasuk data transaksi yang dibutuhkan untuk menentukan bagaimana kas dihasilkan atau digunakan selama periode tersebut (Jerry J. Weygandt, dkk ; 2008 : 328).

II.3. Basis Data (*Database*)

Basis data menurut Stephen dan Plew adalah (2000) adalah mekanisme yang digunakan untuk menyimpan informasi atau data (Janner Simarmata ; 2006 : 1).

Ramakrishnan dan Gehkre (2003) menyatakan basis data sebagai kumpulan data, yang umumnya mendeskripsikan aktivitas satu organisasi atau lebih yang berhubungan (Janner Simarmata ; 2006 : 1).

1. Keuntungan DBMS (*Database Management System*)

   DBMS memungkinkan perusahaan maupun pengguna individu untuk :

   a. Mengurangi perulangan data

      Apabila dibandingkan dengan *file-file* komputer yang disimpan terpisah di setiap lokasi komputer. DBMS mengurangi jumlah total *file* dengan menghapus data yang terduplikasi di berbagai *file.* Data terduplikasi selebihnya dapat ditempatkan dalam satu *file.*

   b. Mencapai independensi data

      Spesifikasi data disimpan dalam skema pada tiap program aplikasi. Perubahan dapat dibuat pada struktur data tanpa mempengaruhi program yang mengakses data.

   c. Mengintergrasikan data beberapa *file*

      Saat *file* dibentuk sehingga menyediakan kegiatan logis, maka organisasi fisik bukan merupakan kendala. Organisasi logis, pandangan pengguna, dan program aplikasi tidak harus tercemin pada media.

d. Mengambil data dan informasi dengan cepat.

Hubungan-hubungan logis, bahasa manipulasi data, serta bahasa *query* memungkinkan pengguna mengambil data dalam hitungan detik atau menit.

e. Meningkatkan keamanan

DBMS *mainframe* maupun komputer mikro dapat menyertakan beberapa lapis keamanan seperti kata sandi (*password*), direktori pemakai, dan bahasa sandi (*encryption*) sehingga data yang dikelola akan lebih aman.

2. Kerugian DBMS

Keputusan menggunakan DBMS mengikat perusahaan atau pengguna untuk :

a. Memperoleh perangkat lunak

DBMS *mainframe* masih sangat mahal. Walaupun harga DBMS berbasis komputer mikro lebih murah, tetapi tetap merupakan pengeluaran besar bagi suatu oraganisasi kecil.

b. Memperoleh konfigurasi perangkat keras yang besar

DBMS sering memerlukan kapasitas penyimpanan dan memori lebih besar dari pada program aplikasi lain.

c. Mempekerjakan dan mempertahankan staf DBA

DBMS memerlukan pengetahuan khusus agar dapat memanfaatkan kemampuannya secara penuh. Pengetahuan khusus ini disediakan paling baik oleh para pengguna basis data (DBA).

Baik basis data terkomputerisasi maupun DBMS bukanlah prasyarat untuk memecahkan masalah. Namun, keduanya memberikan dasar-dasar mengunakan komputer sebagai suatu sistem informasi bagi para spesialis informasi dan pengguna (Janner Simarmata ; 2006 : 8-9)

II.3.1. Kamus Data

Kamus data (KD) atau data dictionary (DD) yang disebut juga dengan system dictionary data adalah katalog fakta tentang data dan kebutuhan-kebutuhan informasi dari suatu sistem informasi. Dengan demikian KD, analisis sistem dapat mendefenisikan data yang mengalir di sistem dengan lengkap. KD dibuat pada pada tahap analisis sistem dan digunakan baik pada tahap analisis maupun pada tahap perancangan sistem. Pada tahap analisis, KD dapat digunakan sebagai alat komunikasi antara analisis sistem dengan pemakai sistem tentang data yang mengalir di sistem. Pada tahap perancangan sistem KD digunakan untuk merancang input, merancang laporan-laporan dan database. KD dibuat berdasarkan arus data yang ada di DAD. Arus data di DAD sifatnya global, hanya ditunjukkan nama arus datanya saja (Jogiyanto, HM ; 2005 : 725).

II.3.2. Isi Kamus Data

KD harus dapat mencerminkan keterangan yang jelas tentang data yang dicatatnya. Untuk maksud keperluan ini maka KD harus memuat hal-hal berikut ini :

1. **Nama Arus Data**

   Karena KD dibuat berdasarkan arus data yang mengalir di DAD, maka nama dari arus data juga harus dicatat di KD, sehingga mereka yang membaca DAD

dan memerlukan penjelasan lebih lanjut tentang suatu arus data tertentu di DAD dapat langsung mencarinya dengan mudah di KD.

2. Alias

Alias atau nama lain dari data dapat dituliskan bila nama lain ini ada. Alias perlu ditulis karena data yang mempunyai nama yang berbeda untuk orang atau departemen satu dengan yang lainnya. Misalnya bagian pembuat faktur dan langganan menyebut bukti penjualan sebagai faktur, sedang bagian gudang menyebutnya sebagai tembusan permintaan persediaan barang. Baik faktur dan tembusan permintaan persediaan ini mempunyai struktur data yang sama tetapi mempunyai struktur yang berbeda.

3. Bentuk data

Telah diketahui bahwa arus data mengalir :

a. Dari kesatuan luar ke suatu proses, data yang mengalir ini biasanya tercatat di suatu dokumen atau formulir.
b. Hasil dari suatu proses ke kesatuan luar, data yang mengalir biasanya terdapat di media laporan atau *query* tampilan layar atau dokumen hasil cetakan komputer.
c. Hasil dari suatu proses yang lain, data yang mengalir biasanya dalam bentuk variabel atau parameter yang dibutuhkan oleh proses penerimanya.
d. Hasil dari suatu proses yang direkamkan ke simpanan data, data yang mengalir ini biasanya berbentuk suatu variabel.
e. Dari simpanan data dibaca oleh suatu proses, data yang mengalir ini biasanya berupa suatu *field* (item data).

Dengan demikian bentuk dari data yang mengalir dapat berupa :

1. Dokumen dasar atau formulir
2. Dokumen hasi cetakan komputer
3. Laporan tercetak
4. Tampilan dilayar monitor
5. Variabel
6. Parameter
7. *Field*

Bentuk dari data ini perlu dicatat di KD, karena dapat digunakan untuk mengkelompokan KD ke dalam kegunaanya, sewaktu perancangan sistem KD yang mencatat data yang mengalir dalam bentuk dokumen dasar atau formulir yang digunakan untuk merancang bentuk input sistem. KD yang mencatat data yang mengalir dalam bentuk laporan tercetak dan dokumen hasil cetakan komputer akan digunakan untuk merancang *output* yang akan dihasilkan oleh sistem. KD yang mencatat data yang mengalir dalam bentuk tampilan layar monitor akan digunakan juga untuk merancang tampilan layar yang akan dihasilkan oleh sistem. KD yang mencatat data yang mengalir ke dalam bentuk parameter dan variabel akan digunakan untuk merancang proses dari program. KD yang mencatat data yang mengalir ke dalam bentuk dokumen, formulir, laporan, dokumen cetakan komputer, tampilan di layar monitor, variabel, dan *field* akan digunakan untuk merancang database.

4. Arus data

Arus data menunjukkan dari mana data yang mengalir dan kemana data akan menuju. Keterangan arus data ini perlu dicatat di KD supaya memudahkan mencari arus data di DAD (Jogiyanto, HM ; 2005 : 726-727).

II.3.3. *Entity Relationship Diagaram* (ERD)

Merupakan suatu model untuk menjelaskan hubungan antar dua dalam basis data berdasarkan suatu persepsi bahwa *real word* terdiri dari *object-object* dasar yang memmpunyai hubungan atau antar *object-object* tersebut. Relasi antar *object* dengan menggunakan simbol-simbol grafis tertentu.

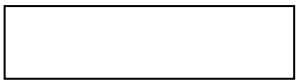
Model *entity relationship* adalah suatu penyajian dengan menggunakan *entity* dan *relationship*. Diperkenalkan pada tahun 1976 oleh P.P. Chen.

II.3.4. Komponen-komponen yang terdapat didalam *Entity Relationship* Model.

1. *Entity*
   a. Adalah sesuatu yang dapat dibedakan dalam dunia nyata dimana informasi yang berkaitan dengannya dikumpulkan.
   b. *Entity* set adalah kumpulan *entity* yang sejenis.
   c. Symbol yang digunakan untuk *entity* adalah persegi panjang.
   d. *Entity* set dapat berupa :
      1. *Entity* yang bersifat fisik, yaitu *entity* yang dapat dilihat. Contohnya : rumah, kendaraan, mahasiswa, dosen, dan lain-lain.

2. *Entity* yang bersifat konsep atau *logic*, yaitu *entity* yang tidak dapat dilihat. Contohnya : pekerjaan, perusahaan, rencana, mata kuliah, dan lain-lain.

e. Simbol yang digunakan untuk *entity* adalah persegi panjang.



Gambar II.3. *Entity*

Sumber : Linda Marlinda, S. Kom (2004:17)

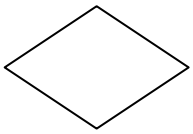
2. *Relationship*

a. Adalah hubungan yang terjadi antara satu atau lebih *entity*.

b. *Relationship* tidak mempunyai keberadaan fisik, kecuali yang mewarisi hubungan antara *entity* tersebut.

c. *Relationship* set adalah kumpulan relationship yang sejenis.

d. Simbol yang digunakan adalah bentuk belah ketupat, *diamond* atau *rectangle.*



Gambar II.4. *Relationship*

Sumber : Linda Marlinda, S. Kom (2004:18)

3. *Attribute*

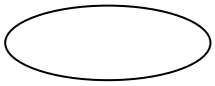
a. Adalah karakteristik dari *entity* atau *relationship* yang menyediakan penjelasan detail tentang atau *relationship* tersebut.

b. *Attribute value* (nilai atribut) adalah suatu data aktual atau informasi yang disimpan di suatu *attribute* di dalam suatu *entity* atau *relationship*.

c. Terdapat dua jenis atribute, yaitu :

1. *Indetifer (key)*, untuk menentukan suatu *entity* secara unik.
2. *Descriptor (nonkey attribute)*, untuk menentukan karakteristik dari suatu *entity* yang tidak unik.

d. Simbol yang digunakan adalah bentuk oval



Gambar II.5. *Attribute*

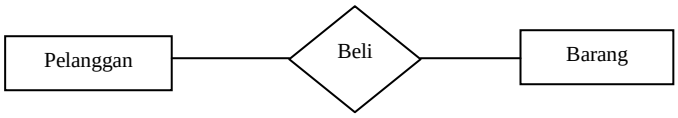
Sumber : Linda Marlinda, S. Kom (2004:18)

4. Indicator Tipe

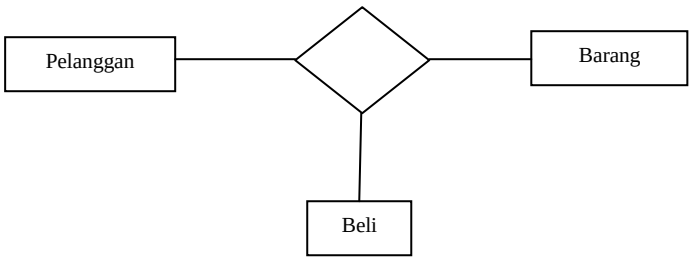
a. *Indicator tipe associative object*

Berfungsi sebagai suatu objek dan suatu *relationship*

Contoh :



Menjadi :



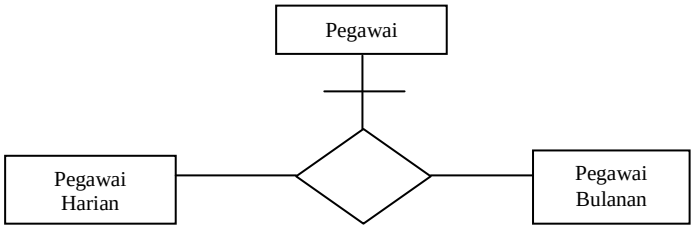
Gambar II.6. Indicator Tipe

Sumber : Linda Marlinda, S. Kom (2004:19)

b. *Indicator tipe supertipe*

Terdiri dari suatu object dan sub kategori atau lebih yang dihubungkan dengan dihubungkan dengan relationship yang tidak bernama (Linda Marlinda, S. Kom 2004 : 17 - 19).

Contoh :



Gambar II.7. Indicator Tipe SuperTipe
Sumber : Linda Marlinda, S. Kom (2004:19)

II.3.5. Normalisasi

Normalisasi adalah proses pengkelompokkan *attribute* - *attribute* dan suatu relasi sehingga membentuk *Well Structure Relation.* Normalisasi merupakan proses pengkelompokkan elemen data menjadi suatu tabel-tabel menunjukan entity dan relasinya. Normalisasi ditemukan pada tahun 1970 oleh E. F. CODD.

II.3.6. Well-Structure Relation

Well- Structure Relation adalah sebuah *relation* dengan jumlah kerangkapan datanya sedikit (*Minimum amount of redundancy*), serta memberikan kemungkinan bagi user untuk melakukan *Insert, Delete,* dan *Modify* terhadap baris-baris data pada *relation* tersebut, yang tidak berakibat terjadinya *Error* atau INKONSENTENSI DATA, yang disebabkan oleh operasi-operasi tersebut.

Contoh :

Terdapat sebuah *Relation Course* dengan ketentuan sebagai berikut :

a. Setiap mahasiswa hanya boleh mengambil satu mata kuliah saja.

b. Setiap mata kuliah mempunyai uang kuliah yang standar (tidak tergantung pada mahasiswa yang mengambil mata kuliah tersebut).

Tabel II.1. *Relation Course*

STUDENT-ID	KODE-MTK	BIAYA
92130	CS-200	75
92200	CS-300	100
99250	CS-300	75
92425	CS-400	150
92500	CS-300	100
92575	CS-500	50

Sumber : Linda Marlinda, S. Kom (2004 :115)

Relation Course diatas merupakan sebuah *relation* yang sederhana dan terdiri dari 3 kolom/*attribute* (Linda Marlinda ; 2004 : 115).

a. Bentuk tidak normal (*Unnormalized Form*)

Bentuk ini merupakan kumpulan data yang akan direkam, tidak ada keharusan mengikuti suatu format tertentu. Dapat saja data tidak lengkap atau terduplikasi. Data dikumpulkan apa adanya dengan saat menginput.

Contoh data :

Tabel II.2. Bentuk tidak normal (*Unnormalized Form*)

No_Siswa	Nama	Pa	Kelas 1	Kelas 2	Kelas 3
22890100	Shandy	Linda	1234	1543	1543
22890101	Susi	Riska	1234	1775	

Sumber :Linda Marlinda, S. Kom (2004 : 122)

Siswa yang punya nomor siswa, nama, dan pa mengikuti 3 mata pelajaran/kelas. Di sini ada perulangan kelas 3 kali ini bukan bentuk tidak 1 NF.

b. Bentuk normal ke satu (1 NF/ *Fisrt Normal Form*)

Suatu relasi 1 NF dan hanya jika sifat dan setiap relasi attributenya bersifat *atomic*. Atom adalah zat terkecil yang masih memiliki sifat induknya. Bila dipecah lagi maka ia tidak memiliki sifat induknya.

Ciri-ciri 1 NF :

1. Setiap data dibentuk kedalam *flat file* data terbentuk per satu *record* nilai dan field berupa " *atomic value* ".
2. Tidak ada *set attribute* yang berulang atau bernilai ganda,
3. Tiap *field*
4. Hanya satu pengertian.

Tabel II.3. Bentuk Normal Ke Satu (1 NF/ *Fisrt Normal Form*)

No_Siswa	Nama	Pa	Kelas 1
22890100	Shandy	Linda	1234
22890100	Shandy	Linda	1543
22890101	Susi	Riska	1234
22890101	Susi	Riska	1775
22890101	Susi	Riska	1543

Sumber :Linda Marlinda, S. Kom (2004 : 122)

c. Bentuk normal ke dua (2 NF/ *Second Normal Form*)

Bentuk normal kedua mempunyai syarat yaitu bentuk data telah memenuhi kriteria bentuk normal kesatu. *Attribute* bukan kunci haruslah bergantung secara fungsi pada *primary key*. Jadi, untuk membentuk normal kedua haruslah sudah ditentukan kunci-kunci *field.* Kunci *field*

haruslah unik dan dapat mewakili *attribute* lain yang menjadi anggotanya. Misal : dari contoh relasi siswa pada 1 NF terlihat bahwa *primary key* adalah nomor siswa. Nama siswa dan pa bergantung fungsi pada no_siswa, tetapi kode_kelas bukanlah fungsi dan siswa dipecah menjadi 2 relasi :

Relasi siswa :

Tabel II.4. Bentuk Normal Kedua Relasi Siswa (2 NF/ *Second Normal Form*)

No_Siswa	Nama	Pa
22890100	Shandy	Linda
22890101	Susi	Riska

Sumber :Linda Marlinda, S. Kom (2004 : 123)

Relasi ambil_Kelas

Tabel II.5. Bentuk Normal Kedua Relasi ambil_Kelas (2 NF/ *Second Normal Form*)

No_Siswa	Kode Kelas
22890100	1234
22890100	1543
22890101	1234
22890101	1775
22890101	1543

Sumber : Linda Marlinda, S. Kom (2004 : 123)

d. Bentuk normal ketiga (3 NF/*Third Normal Form*)

Untuk menjadi bentuk normal ketiga maka relasi dalam bentuk normal kedua dan semua *attribute* bukan primer tidak punya hubungan yang transistif. Dengan kata lain, setiap *attribute* bukan kunci haruslah bergantung hanya pada *primary key* dan *primary key* secara menyeluruh.

Contoh pada bentuk normal kedua diatas termasuk juga bentuk normal ketiga karena seluruh attribute yang ada bergantung penuh pada kunci primernya.

e. *Boyee-Cood Normal Form* (BCNF)

BNCF mempunyai paksaan lebih kuat dan bentuk normal ketiga untuk menjadi BCNF, relasi harus dalam bentuk normal kesatu dan setiap *attribute* harus bergantung fungsi pada *attribute superkey.*

Pada contoh di bawah ini terdapat relasi seminar dengan ketentuan sebagai berikut :

1. Kunci primer adalah no_siswa +seminar
2. Siswa bopleh mengambil satu atau dua seminar.
3. Setiap siswa dibimbing oleh salah satu di antara 2 instruktur seminar tersebut.
4. Setiap instruktur boleh hanya mengambil satu seminar saja.

   Pada contoh ini no_siswa dan seminar menunjuk seorang instruktur.

Relasi seminar

Tabel II.6. *Boyee-Cood Normal Form* (BCNF)

No_Siswa	Seminar	Instruktur
22890100	2281	Si doel
22890101	2281	Pak tile
22890102	2291	Mandra
22890101	2291	Basuki
22890109	2291	Basuki

Sumber :Linda Marlinda, S. Kom (2004 : 124)

Bentuk relasi seminar adalah bentuk normal ketiga, tetapi tidak BCNF karena nomor seminar masih tergantung fungsi pada instruktur. Jika setiap instruktur dapat mengajar hanya pada satu seminar saja, maka seminar bergantung fungsi pada satu *attribute* bukan *superkey* seperti disyaratkan oleh BCNF. Maka relasi seminar haruslah dipecah menjadi dua yaitu :

Tabel II.7. *Boyee-Cood Normal Form* (BCNF)

Relasi Pengajar

Instruktur	Seminar
Si doel	2281
Pak tile	2281
Mandra	2291
Basuki	2291

No_Siswa	Instruktur
22890100	Si doel
22890101	Pak tile
22890102	Mandra
22890101	Basuki
22890109	Basuki

Sumber :Linda Marlinda, S. Kom (2004 : 124)

f. Bentuk normal keempat (4NF)

Relasi R adalah bentuk 4 NF jika dan hanya jika relasi tersebut juga termasuk BCNF dan semua ketergantungan *multivaluei* adalah juga ketergantungan fungsional.

g. Bentuk normal kelima (5 NF)

Disebut juga PINF (*Projection Join Normal Form*) dan 4 NF dilakukan dengan menghilangakan ketergantungan *join* yang merupakan kunci kandidat (Linda Marlinda, S. Kom ; 2004 : 122-125).

II.4. *Unified Modeling Language* (UML)

UML singkatan dari *Unified Modelling Langguage* yang berarti bahasa pemodelan standart. (Chonoles ; 2003 : 6) mengatakan sebagai bahasa, berarti

UML memiliki sintaks dan *semantic*. Ketika kita membuat model menggunakan konsep *UML* ada aturan –aturan yang harus diikuti. Bagaimana elemen pada model-model yang kita buat harus berhubungan satu dengan lainnya harus mengikuti standart yang ada. *UML* bukan hanya sekedar diagram, tetapi juga menceritakan konteksnya. Ketika pelanggan memesan sesuatu dari sistem, bagaimana transaksinya? Bagaimana sistem mengatasi error yang terjadi? Bagaimana keamanan terhadap sistem yang ada kita buat? Dan sebagainya dapat dijawab dengan *UML*.

UML diaplikasikan untuk maksud tertentu, biasanya antara lain untuk :

1. Merancang perangkat lunak.
2. Sarana komunikasi antara perangkat lunak dengan bisnis.
3. Menjabarkan sistem secara rinci untuk analisa dan mencari apa yang diperlukan sistem.
4. Mendokumentasikan sistem yang ada, proses-proses dan organisasinya.

UML telah diaplikasikan dalam investasi perbankan,lembaga kesehatan, departemen pertahanan, sistem terdistribusi, sistem pendukung alat kerja, retail, sales, dan supplier.

Blok pembangunan utama *UML* adalah diagram. Beberapa diagram ada yang rinci (jenis *timing diagram*) dan lainya ada yang bersifat umum (misalnya diagram kelas). Para pengembang sistem berorientasikan objek menggunakan bahasa model untuk menggambarkan, membangun dan mendokumentasikan sistem yang mereka rancang. *UML* memungkinkan para anggota team untuk bekerja sama dalam mengaplikasikan beragam sistem. Intinya, *UML* merupakan alat

komunikasi yang konsisten dalam mensuport para pengembang sistem saat ini. Sebagai perancang sistem mau tidak mau pasti menjumpai *UML*, baik kita sendiri yang membuat sekedar membaca diagram *UML* buatan orang lain (Prabowo Pudji Widodo, Herlawati ; 2011 : 6 - 7).

II.4.1. Diagram-*Diagram UML*

Beberapa literatur menyebutkan bahwa *UML* menyediakan Sembilan jenis diagram, yang lain menyebutkan delapan karena ada beberapa yang digabung, misalnya diagram komunikasi, diagram urutan, dan diagram pewaktuan digabung menjadi diagram interaksi. Namun demikian model-model itu dapat dikelompokan berdasarkan sifatnya yaitu statis atau dinamis. Jenis diagram itu antara lain :

1. Diagram Kelas. Bersifat statis. Diagram ini memperlihatkan himpunan kelas-kelas, antarmuka-antarmuka, kolaborasi, serta relasi-relasi diagram. Diagram ini umu dijumpai pada pemodelan sistem berorentasi objek. Meskipun bersifat statis, sering pula diagram kelas memuat kelas-kelas.
2. Diagram Paket (*Package Diagram*) bersifat statis. Diagram ini memperlihatkan kumpulan kelas-kelas merupakan bagian dari diagram komponen.
3. Diagram *Use Case* bersifat statis. Diagram ini memperlihatkan himpunan *use-case* dan aktor-aktor (suatu jenis khusus dari kelas). Diagram ini terutama sangat penting untuk mengorganisasi dan memodelkan perilaku suatu sistem yang dibutuhkan serta diharapkan pengguna.

4. Diagram interaksi dan *Sequence* (urutan). Bersifat dinamis. Diagram urutan adalah diagram interaksi yang menekankan pada pengiriman pesan dalam waktu tertentu.
5. Diagram komunikasi (*Communication Diagram*) bersifat dinamis. Diagram sebagai pengganti diagram kolaborasi *UML* yang menekankan organisasi *structural* dari objek-objek yang menerima serta mengirim pesan.
6. Diagram *Statechart* (*Statechart Diagram)* bersifat dinamis. Diagram status memperlihatkan keadaan-keadaan pada sistem, memuat status (*State)*, transisi kejadian serta aktifitas. Diagram ini terutama penting untuk memperlihatkan sifat dinamis dari antarmuka (*interface*), kelas, kolaborasi dan terutam penting pada pemodelan sistem-sistem yang reaktif.
7. Diagram aktivitas (*Activity Diagram*) bersifat dinamis. Diagram aktivitas adalah tipe khusus dari diagram status yang memperlihatkan aliran dari suatu sistem. Diagram ini terutama penting dalam pemodelan fungsi-fungsi suatu sistem dan member tekanan pada aliran kendali antar objek.
8. Diagram komponen (*Component Diagram*) bersifat statis. Diagram komponen ini memperlihatkan organisasi serta kebergantungan sistem/perangkat lunak pada komponen-komponen yang telah ada sebelumnya. Diagram ini berhubungan diagram kelas dimana komponen dipetakan kedalam satu atau lebih kelas-kelas. Antarmuka-antarmuka serta kolaborasi-kolaborasi.
9. Diagram *Deployment* (*Deployment Diagram)* bersifat statis. Diagram ini memperlihatkan konfigurasi saat aplikasi dijalankan (*run time)*. Memuat simpul-simpul berserta komponen-komponen yang ada di dalamnya. Diagram

Deployment berhubungan erat dengan diagram komponen dimana diagram ini memuat satu atau lebih komponen-komponen. Diagram ini sangat berguna saat aplikasi kita berlaku sebagai aplikasi yang dijalankan pada banyak mesin (*distributed computing).*

Kesembilan diagram ini tidak mutlak harus digunakan dalam pengembangan perangkat lunak, semuanya dibuat sesuai dengan kebutuhan. Pada *UML* dimungkinkan kita menggunakan diagram-diagram lainya misalnya *Data Flow Diagram, Entity Relationship Diagram* dan sebagainya (Prabowo Pudji Widodo, Herlawati ; 2011 : 10-12).

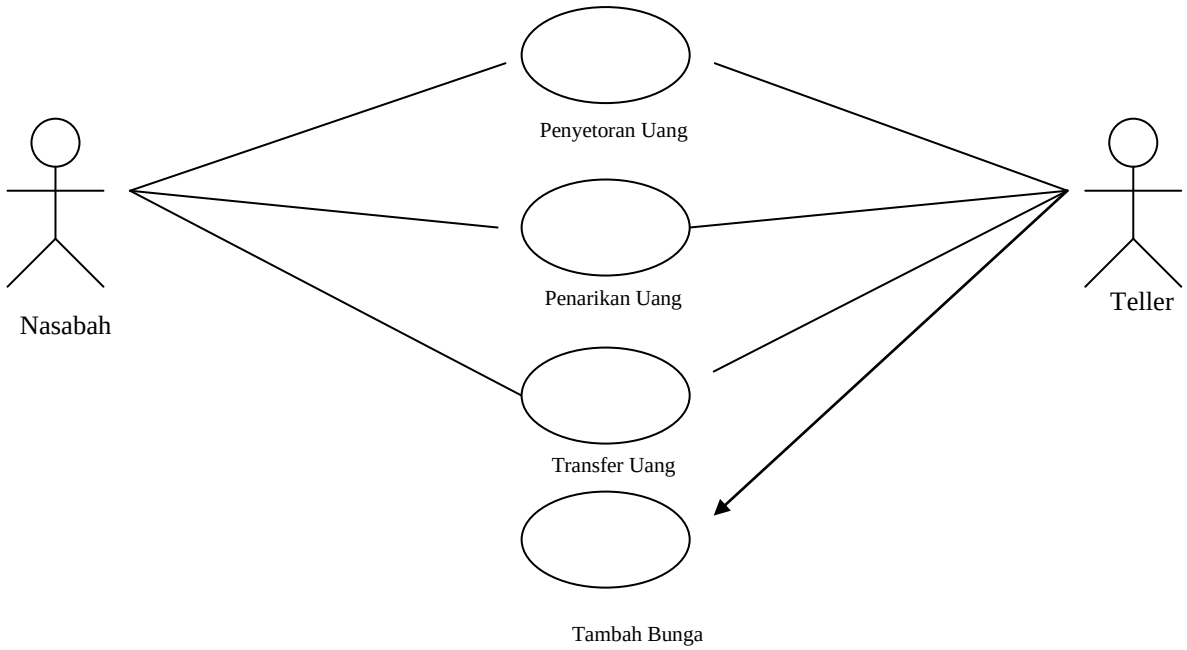
1. *Diagram Use Case (Use Case Diagram)*

   Use Case menggambarkan *external view* dari sistem yang akan kita buat modelnya. Menurut Pooley (2005 : 15) mengatakan bahwa model *use case* dapat dijabarkan dalam diagram, tetapi yang perlu diingat, diagram tidak identik dengan model karena model lebih luas dari diagram.

   komponen pembentuk diagram *use case* adalah :

   a. Aktor (*actor),* menggambarkan pihak-pihak yang berperan dalam sistem.
   b. *Use Case*, aktivitas/ sarana yang disiapkan oleh bisnis/sistem.
   c. Hubungan (*Link),* aktor mana saja yang terlibat dalam *use case* ini.

   Gambar di bawah ini merupakan salah satu contoh bentuk diagram *use case (*Prabowo Pudji Widodo, Herlawati ; 2011 : 15 - 16).

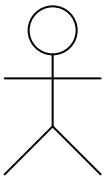


Gambar II.8. Diagram *Use Case*

Sumber : Prabowo Pudjo Widodo, Herlawati (2011:17)

2. Aktor

Menurut Chonoles (2003 : 17) menyarankan sebelum mebuat use case dan menentukan aktornya, agar mengidentifikasi siapa saja pihak yang terlibat dalam sistem kita. Pihak yang terlibat biasanya dinamakan *stakeholder*.



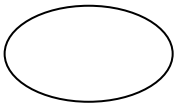
Gambar II.9. Aktor

Sumber : Prabowo Pudjo Widodo, Herlawati (2011:17)

3. *Use Case*

Menurut Pilone (2005 : 21) *use case* menggambarkan fungsi tertentu dalam suatu sistem berupa komponen kejadian atau kelas. Sedangkan menurut

Whitten (2004 : 258) mengartikan *use case* sebagai urutan langkah-langkah yang secara tindakan saling terkait (skenario) baik terotomatisasi maupun secara manual, untuk tujuan melengkapi satu tugas bisnis tunggal. *Use case* digambarkan dalam bentuk *ellips*/oval



Gambar II.10. Simbol *Use Case*
Sumber : Prabowo Pudjo Widodo, Herlawati (2011:22)

Use case sangat menentukan karakteristik sistem yang kita buat, oleh karena itu Chonoles (2003:22-23) menawarkan cara untuk menghasilkan *use case* yang baik yakni :

a. Pilihlah nama yang baik

Use case adalah sebuah *behaviour* (prilaku), jadi seharusnya dalam frase kata kerja. Untuk membuat namanya lebih detil tambahkan kata benda mengindikasikan dampak aksinya terhadap suatu kelas objek. Oleh karena itu diagram *use case* seharusnya berhubungan dengan diagram kelas.

b. Ilustrasikan perilaku dengan lengkap.

Use case dimulai dari inisiasi oleh aktor primer dan berakhir pada aktor dan menghasilkan tujuan. Jangan membuat *use case* kecuali anda mengetahui tujuannya. Sebagai contoh memilih tempat tidur (*King Size, Queen Size,* atau *double*) saat tamu memesan tidak dapat dijadikan *use case* karena merupakan bagian dari *use case* pemesanan kamar dan tidak

dapat berdiri sendiri (tidak mungkin tamu memesan kamar tidur jenis king tapi tidak memesan kamar hotel).

c. Identifikasi perilaku dengan lengkap.

Untuk mencapai tujuan dan menghasilkan nilai tertentu dari aktor, use case harus lengkap. Ketika memberi nama pada *use case*, pilihlah frasa kata kerja yang implikasinya hingga selesai. Misalnya gunakan frasa *reserve a room* (pemesanan kamar) dan jangan *reserving a room* (memesan kamar) karena memesan menggambarkan perilaku yang belum selesai.

d. Menyediakan use case lawan (*inverse)*

Kita biasanya membutukan *use case* yang membatalkan tujuan, misalnya pada us*e case* pemesanan kamar, dibutuhkan pula *use case* pembatalan pesanan kamar.

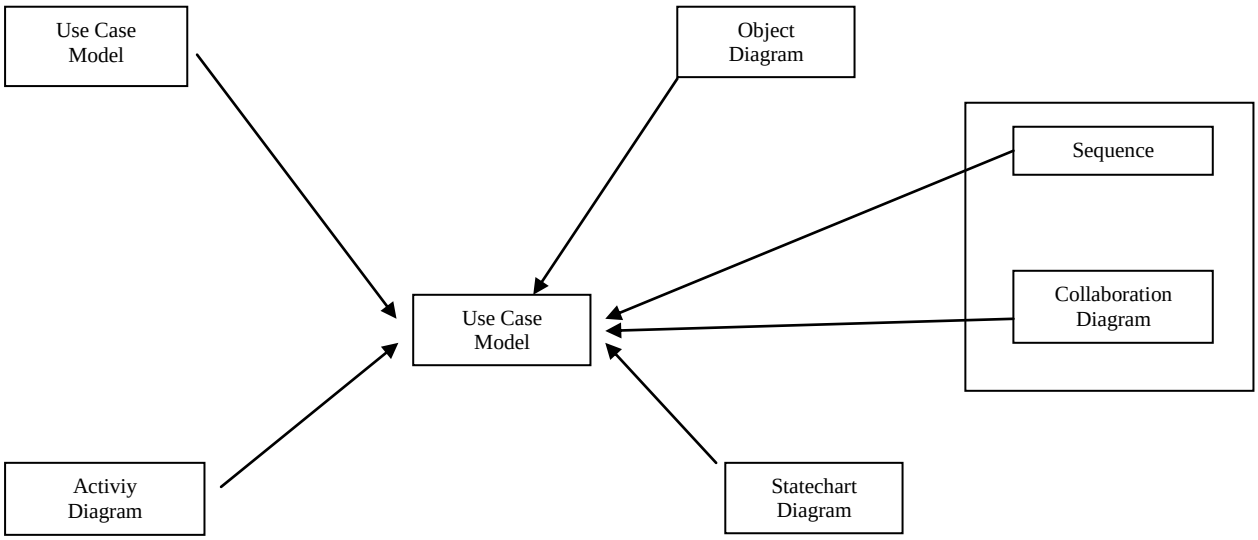
e. Batasi use case hingga satu perilaku saja.

Kadang kita cenderung membuat *use case* yang lebih dari satu tujuan aktivitas. Guna menghindari kerancuan, jagalah use case kita hanya fokus pada satu hal. Misalnya, penggunaan *use case check in* dan *check out* dalam satu *use case* menghasilkan ketidakfokusan, karena memiliki dua perilaku yang berbeda.

4. Diagram Kelas (*Class Diagram*)

Diagram kelas adalah inti dari proses pemodelan objek. Baik *forward engineering* maupun *reverse engineering* memanfaatkan diagram ini *forward engineering* adalah proses perubahan model menjadi kode program

sedangkan *reverse engineering* sebaliknya merubah kode program menjadi model (Prabowo Pudji Widodo, Herlawati ; 2011 : 37).



Gambar II.11. Hubungan Diagram Kelas Dengan Diagram *UML* lainya
Sumber : Prabowo Pudjo Widodo, Herlawati (2011 : 38)

5. Diagram Aktivitas (*Activity Diagram*)

Diagram aktivitas lebih memfokuskan diri pada eksekusi dan alur sistem dari pada bagaimana sistem dirakit. Diagram ini tidak hanya memodelkan software melainkan memodelkan bisnis juga. Diagram aktivitas menunjukan aktivitas sistem dalam kumpulan aksi - aksi. Ketika digunakan dalam pemodelan *software*, diagram aktivitas merepresentasikan pemanggilan suatu fungsi tertentu misalnya *call.* Sedangkan bila digunakan dalam pemodelan bisnis, diagram ini menggambarkan aktivitas yang dipicu oleh kejadian-kejadian diluar seperti pemesanan atau kejadian-kejadian internal misalnya

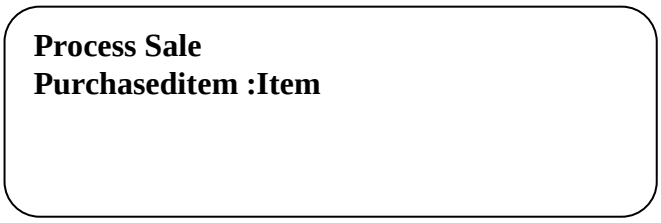
penggajian tiap jumat sore (Prabowo Pudji Widodo, Herlawati ; 2011 : 143-145).

Aktivitas merupakan kumpulan aksi - aksi. Aksi - aksi melakukan langkah sekali saja tidak boleh dipecah menjadi beberapa langkah-langkah lagi. Contoh aksinya yaitu :

a. Fungsi Matematika
b. Pemanggilan Perilaku
c. Pemrosesan Data

Ketika kita menggunakan diagram aktivitas untuk memodelkan perilaku suatu *classfier* dikatakan kontek dari aktivitas. Aktivitas dapat mengakses atribut dan operasi *classfier*, tiap objek yang terhubung dan parameter - parameter jika aktivitas memiliki hubungan dengan perilaku. Ketika digunakan dengan model proses bisnis, informasi itu biasanya disebut *process-relevant data*. Aktivitas diharapkan dapat digunakan ulang dalam suatu aplikasi, sedangkan aksi biasanya *specific* dan digunakan hanya untuk aktivitas tertentu.

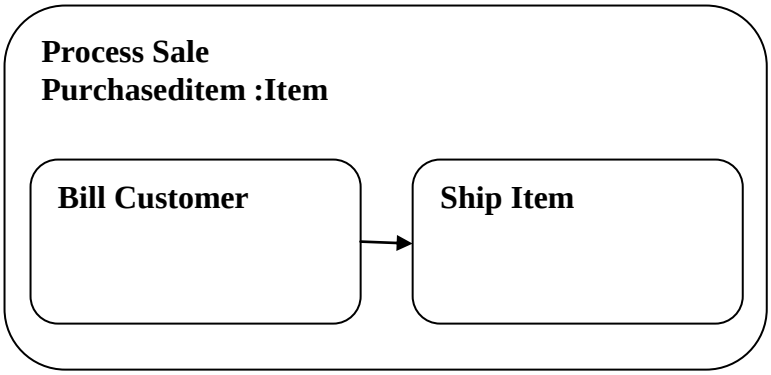
Aktivitas digambarkan dengan persegi panjang tumpul. Namanya ditulis di kiri atas. Parameter yang terlibat dalam aktivitas ditulis dibawahnya.



Gambar II.12. Aktivitas serderhana tanpa rincian

Sumber : Prabowo Pudjo Widodo, Herlawati (2011 : 145)

Detail aktivitas dapat dimasukan di dalam kotak. Aksi diperlihatkan dengan symbol yang sama dengan aktivitas dan namanya diletakkan didalam persegi panjang.



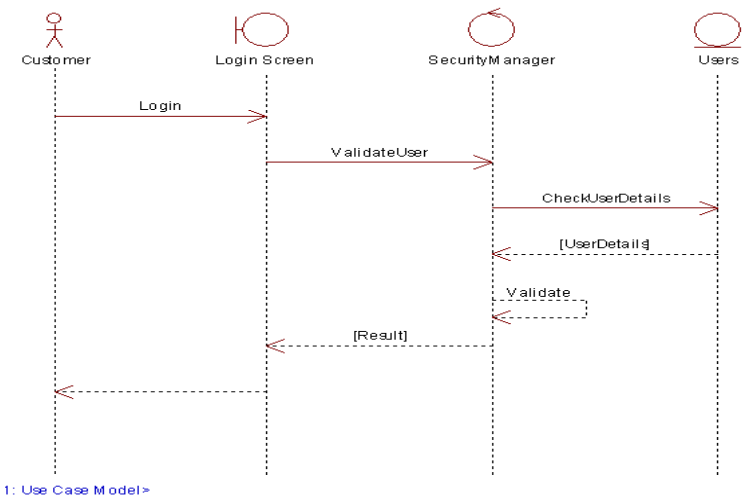
Gambar II.13. Aktivitas dengan detail rincian

Sumber : Prabowo Pudjo Widodo, Herlawati (2011:145)

6. *Squence Diagram*

Menurut Douglas (2004 : 174) menyebutkan ada tiga diagram primer UML dalam memodelkan skenario interaksi, yaitu diagram urutan (*sequence diagram),* diagram waktu (*timing diagram)* dan diagram komunikasi (*communication diagram).*

Menurut Pilone (2005 : 174) menyatakan bahwa diagram yang paling banyak dipakai adalah diagram urutan. Gambar II.9. memperlihatkan contoh diagram urutan dengan notasi - notasinya yang akan dijelaskan nantinya (Prabowo Pudji Widodo, Herlawati ; 2011 : 174-175).



Gambar II.14. Diagram Urutan

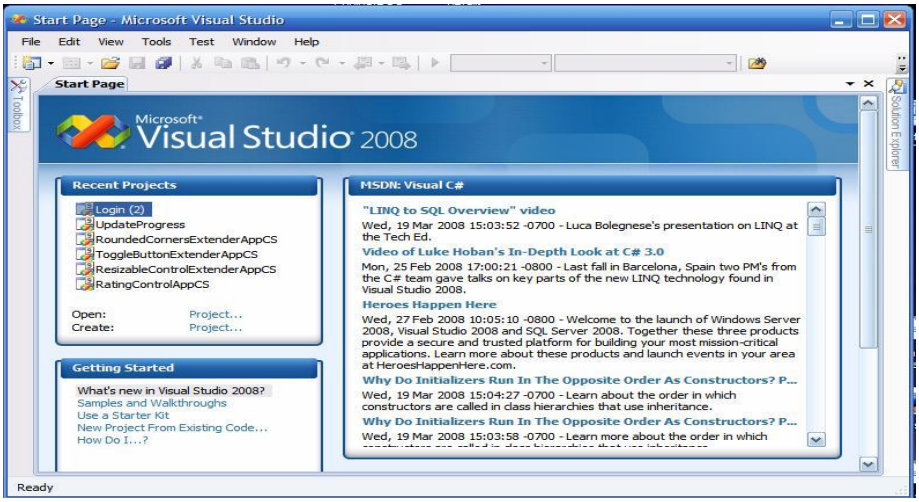
Sumber : Prabowo Pudjo Widodo, Herlawati (2011:175)

II.5. Bahasa Pemograman *Microsoft Visual Studio* 2008

Microsoft Visual Studio 2008 merupakan kelanjutan dari *Microsoft Visual Studio* sebelumnya, yaitu *Visual Studio. Net* 2003 yang diproduksi oleh Microsoft. Pada bulan Februari 2002 *Microsoft* memproduksi teknologi. *Net Framework* versi 1.0, teknologi. *Net* ini didasarkan atas susunan berupa *Net Framework*, sehingga setiap produk baru yang terkait dengan teknologi. Net akan selalu berkembang mengikuti perkembangan. *Net Frameworknya*. Pada perkembangannya nantinya mungkin untuk membuat program dengan teknologi. *Net* memungkinkan para pengembang perangkat lunak akan dapat menggunakan lintas sistem operasi, yaitu dapat dikembangkan di sistem operasi windows juga dapat dijalankan pada sistem operasi lain, misalkan pada sistem operasi *Linux*, seperti yang telah dilakukan pada pemograman *Java* oleh *Sun Microsystem*. Pada saat ini perusahaan - perusahaan sudah banyak meng*update* aplikasi lama yang dibuat *Microsoft Visual Basic* 6.0 ke teknologi. *Net* karena kelebihan-kelebihan

yang ditawarkan, terutama memungkinkan pengembang perngkat lunak secara cepat mampu membuat program *robust*, serta berbasiskan integrasi ke internet yang dikenal dengan *XML Web Service* (Ketut Darmayuda ; 2008 : 1)

Untuk melihat tampilan visual studio 2008 dapat dilihat pada gambar II.16. sebagai berikut :



Gambar II.15. Tampilan Utama Visual Studio 2008

Sumber : Ketut Darmayuda (2008 : 12)

II.6. *Microsoft SQL Server*

SQL Server Management Studio membantu anda mengatur database dengan mudah. Anda dapat melakukan pengaturan atas beberapa pada sebuah komputer saja atau melakukan pengaturan *server* secara *remote*. Anda dapat juga membuat *database*, *table*, *index*, dan melakukan manipulasi data terhadap *database* dan tabel-tabelnya.

SQL Server Management Studio memiliki beberapa komponen penting yang mewakili kegunaanya dalam perancangan *database*, dan melakukan pengaturan sistem secara keseluruhan. Komponen-komponen tersebut itu adalah :

a. *Registered Server*

b. *Object Explorer*

c. *Query editor*

Adapun tampilan Microsoft SQL Server 2008 dapat dilihat pada gambar.17. sebagai berikut (Wahana Komputer ; 2008 : 40)

Gambar II.16. *SQL Server 2008*

Sumber : Andi (2008 : 20)

II.6.1. *Interface SQL Server 2008*

Ada 3 *interface* utama saat bekerja dengan *SQL Server* 2008 adalah sebagai berikut :

1. *Registered Server*

   Bila pada tampilan pertama anda tidak melihat panel ini maka anda dapat menampilkannya pada menu *View Regristed Servers* atau menekan kombinasi tombol CTRL + ALT + G. Panel ini memungkinkan anda menjaga koneksi-koneksi dengan server - server yang pernah digunakan. Koneksi-koneksi ini dapat digunakan untuk memeriksa dari server tersebut (*online* atau *offline*) atau melakukan pada obyek-obyeknya (menggunakan pada panel *object explorer*). Setiap user memiliki daftar tersendiri dari *regristered server* yang

disimpan pada mesin lokal. Anda dapat melakukan penambahan atau pengurangan koneksi ke *server*. Anda dapat mengkelompokan koneksi – koneksi ke server tersebut berdasarkan tipe servernya yaitu *Database Engine, Analysis Service, Reporting Service, Intergration Service*.

2. *Object Explorer*

   Anda dapat melihat berbagai obyek yang ada pada sebuah server pada panel ini. Bila panel ini tidak terlihat maka anda dapat menampilkannya dengan menu *View Object Explorer*. Apabila anda melakukan ekpansi dari sebuah cabang maka sebuah struktur logika dari sebuah obyek akan muncul. Anda dapat mengklik tanda + pada sebelah kiri untuk melakukan ekspansi cabang. Untuk melakukan koneksi pada sebuah server klik kanan pada nama servernya kemudian pilih *Connect*, untuk melakukan *Disconnect* , klik kanan dan pilih *disconnect*.

3. *Query Editor*

   Jendela ini digunakan untuk membuat, melakukan editing perintah perintah T-SQL dan mengeksekusi perintah tersebut. Jendela ini akan muncul otomatis setiap kali anda melakukan kegiatan yang berhubungan dengan query. Apabila anda berminat membuat *query* baru atau melakukan *editing file query* yang telah ada, maka jendela ini otomatis akan muncul. Anda dapat membuat *File Query With Current Connection* atau *Database Engine Query atau SQL Server Commpact Query* atau memanfaatkan tombol *New Query* pada *toolbar* (Wahana Komputer ; 2008 : 45 - 49).

II.6.2. Komponen –Komponen *SQL Server* 2008

Ada 5 komponen-komponen SQL Server 2008 adalah sebagai berikut :

1. *Literal Value*

   Yang termasuk dengan *literal value* adalah huruf (a – z), *numerik* (0 - 9), dan hexadesimal (0x). *Literal value* juga dikenal dengan konstanta. Sebuah *konstanta string* terdiri atas satu atau beberapa karakter yang diapit tanda kutip tunggal *(apostroph)* atau tanda petik ganda. Defenisi *konstanta string* sebaiknya digunakan dengan tanda petik tunggal karena tanda petik ganda dalam *query* memiliki fungsi lain.

2. Delimeter

   Bahwa sebaiknya menggunakan tanda petik tunggal untuk mengawali dan mengakhiri sebuah konstanta *string* dari pada tanda petik ganda. Hal ini karena tanda petik ganda juga digunakan sebagai delimeter atau pemisah. Tanda kutip ganda tidak dapat digunakan sebagai pembuka dan penutup dari sebuah konstanta string perintah berikut ini diberikan yaitu *Set Queted Identifier On* Standar perinth tersebut adalah *ON*. Perintah tersebut mengakibatkan tanda petik ganda diatur menjadi *Delimeted Identifier*. *Delimeted Identifier* adalah *identifier* khusus yang memungkinkan reserver word digunakan menjadi *identifier* dan juga membolehkan adanya spasi pada nama *database.*

3. Komentar

   Komentar dalam pemograman ataupun *scripting* diperlukan untuk memberikan keterangan singkat tentang kode-kode yang ada dibawahnya.

Sehingga sewaktu ada kerusakan, kesalahan, *programmer* dapat dengan mudah menngerti apa kegunaan dari kode tersebut pada *SQL Server* terdapat dua macam komentar yaitu :

a. Komentar yang menggunakan tanda / * * /. Dengan tanda ini komentar yang anda berikan dapat terdiri atas beberapa baris di awali dengan / * dan di akhiri dengan */.
b. Komentar yang menggunakan tanda – untuk memberi komentar pada baris yang dimaskud saja. Biasanya digunakan untuk menerangkan *identifier* atau *reserved word.*

4. *Identifier*

   Dalam pemograman bahasa *T-SQL* untuk *query*, *identifier* digunakan untuk melakukan identifikasi *database* dan obyek-obyeknya seperti tabel dan *index*. Diidentifikasi dengan string karakter dengan panjang maksimal 128 karakter. *Identifier* ini dapat terdiri atas berbagai macam huruf, angka dan karakter khusus (@, _#, $). Setiap identifier harus dimulai dengan huruf, atau karakter khusus tidak boleh dengan angka.

5. *Reserved Word.*

   Reserved word atau kata kunci adalah kata yang memilikiarti khusus dan harus dituliskan dengan aturan tertentu. Dalam bahasa *T-SQL reserved word* dan juga memiliki banyak fungsi. Reserved word tidak dapat digunakan sebagai nama sebuah obyek kecuali obyek tersebut didefenisikan sebagai *delimeted identifier*. (Wahana Komputer ; 2008 : 84 - 86).