BAB II

TINJAUAN PUSTAKA

II.1. Sistem Informasi Akuntansi

Sistem informasi akuntansi adalah sistem yang bertujuan untuk mengumpulkan dan memproses data serta melaporkan informasi yang berkaitan dengan saksi keuangan. Misalnya salah satu input dari sistem informasi akuntansi pada sebuah toko baju. Kita memproses transaksi dengan mencatat penjualan tersebut ke dalam jurnal penjualan, mengklasifikasikan transaksi dengan menggunakan kode rekening, dengan memposting transaksi ke dalam jurnal. Kemudian, secara periodik sistem informasi auntansi akanmenghasilkan output berupa laporan keuangan yang terdiri dari neraca dan laporan laba/rugi.

Sistem informasi yang kadang kala di sebut sebagai pemrosesan data, merupakan sistem buatan manusia yang biasanya terdiri dari sekumpulan komponen baik manual ataupun berbasis komputer, yang terintegrasi untuk mengumpulkan, menyimpan, dan mengelola data serta menyediakan informasi kepada pihak-pihak yang berkepentingan sebagai pemakai informasi tersebut.

Lingkup sistem informasi akuntansi dapat dijelaskan dari manfaat yang didapat dari informasi akuntansi. Manfaat atau tujuan sistem informasi akuntansi tersebut adalah sebagai berikut :

1. Mengamankan harta / kekayaan perusahaan. Harta / kekayaan di sini meliputi kas perusahaan, persediaan barang dagangan, termasuk aset tetap perusahaan.

2. Menghasilkan beragam informasi untuk pengambilan keputusan. misal, pengelola toko swalayan memerlukan informasi mengenai barang apa saja yang diminati oleh konsumen. Membeli barang yang kurang laku berarti kas akan terjebak dalam persediaan dan berarti kehilangan kesempatan untuk membeli barang dagangan yang laku.
3. Menghasilkan informasi untuk pihak eksternal. Setiap pengelola usaha memiliki kewajiban untuk membayar pajak. Besarnya pajak yang dibayar tergantung pada omset penjualan (jika pengelola memilih menggunakan norma dalam perhitungan pajaknya) atau tergantung pada laba rugi usaha (jika pengelola memilih untuk tidak menggunakan norma dalam perhitungan pajaknya).
4. Menghasilkan informasi untuk penilaian kinerja karyawan atau divisi. Sistem informasi dapat juga dimanfaatkan untuk penilaian kinerja karyawan atau divisi.
5. Menyediakan data masa lalu untuk kepentingan audit (pemeriksaan). Data yang tersimpan dengan baik sangat memudahkan proses audit (pemeriksaan).
6. Menghasilkan informasi untuk penyusunan dan evaluasi anggaran perusahaan. Anggaran merupakan alat yang sering digunakan perusahaan untuk mengendalikan pengeluaran kas.
7. Menghasilkan informasi yang diperlukan dalam kegiatan perencanaan dan pengendalian. Selain berguna untuk membandingkan informasi yang

berkaitan dengan anggaran dan biaya standar dengan kenyataan seperti yang telah dikemukakan (Anastasia Diana, 2011 : 5-7).

II.2. Pendapatan Jasa

Pendapatan adalah arus masuk bruto dari manfaat ekonomi yang timbul akibat aktivitas normal perusahaan selama satu periode, arus masuk itu mengakibatkan kenaikan modal (ekuitas) dan tidak berasal dari kontribusi penanaman modal. Arus masuk dimaksud adalah hasil dari penjualan produk perusahaan.

Jasa adalah hasil penggunaan produk atau fasilitas perusahaan perusahaan berupa produk tidak berwujud. Untuk memproduksi jasa, perusahaan juga menggunakan bahan baku berupa tenaga kerja dan masukan modal. Biaya untuk karyawan, asuransi, kegiatan administrasi adalah contoh jasa antara lain penyewaan properti, mobil atau aset lain perusahaan.

Arus masuk bruto atau pendapatan adalah hasil dari penjualan produk yang hanya diterima dan dapat diterima oleh perusahaan. Jumlah yang ditagih atas nama pihak ketiga, seperti pajak pertambahan nilai, bukan merupakan manfaat ekonomi yang mengalir ke perushaaan dan tidak mengakibatkan kenaikan modal, dan karena itu harus dikeluarkan dari pendapatan.

Pengertian penjualan jasa bisa menyangkut pelaksanaan tugas yang secara kontraktual telah disepakati untuk dilakukan selama suatu periode waktu. Jasa tersebut dapat diserahkan selama satu periode atau lebih. Penghasilan bersih dari

penjualan jasa adalah penghasilan setelah dikurangi semua komisi enjualan, retur,

Harga penjualan kotor 100 unit @ Rp. 10.000 =		Rp. 1.000.000
Komisi penjualan 5 %	Rp. 50.000	
Diskon 10%	Rp. 100.000	
Retur 4 unit	Rp. 40.000	Rp. 190.000

rabat, diskon dan sebagainya (Ir. Kuswadi ; 2010 : 58).

II.3. Metode *Single Step*

Laporan rugi/laba bentuk tunggal adalah laporan rugi laba yang menggabungkan penghasilan penghasilan menjadi satu kelompok dan menggabungkan biaya-biaya pada kelompok lain. Sehingga untuk meghitung rugi/laba bersih hanya memerlukan satu langkah tunggal yaitu total penghasilan dikurangi total biaya.

Dalam laporan rugi/laba bentuk ini, hanya dikenal satu jenis laba, yaitu dari dua bagian yaitu sebagai berikut:

1) Penghasilan-penghasilan:
    a) Penjualan barang/jasa bersih
    b) Penghasilan lain-lain
2) Biaya-biaya:
    a) Harga pokok penjualan barang/jasa
    b) Biaya usaha
    c) Biaya di luar usaha (Prof. Drs.H. Lili M. Sadeli, M.pd., 2015 : 24)

contoh:

bentuk tunggal *(Singgle Step)*

TOKO "PASUNDAN"
LAPORAN RUGI/LABA
Untuk Bulan yang Berakhir 31 Maret 2000
(dalam ribuan)

Penghasilan- penghasilan:

Hasil penjualan bersih……..	Rp	24.566.100,00
Pendapatan lain-lain……	Rp	180.000,00
Jumlah penghasilan……	Rp	24.746.100,00

Biaya-biaya:

Harga pokok penjualan……	Rp	15.195.750,00
Biaaya penjualan…….	Rp	3.788.100,00
Biaya umum dan administrasi…….	Rp	1.252.800,00
Biaya lain-lain…….	Rp	87.750,00
Jumlah biaya…….	Rp	20.324.400,00
Laba bersih……	Rp	4.421.700,00

Gambar II.1 Contoh Laporan *Single Step*
(Sumber : Prof. Drs.H. Lili M. Sadeli, M.pd., 2015 : 24)

II.4. Microsoft Visual Studio

Microsoft Visual Studio merupakan sebuah perangkat lunak lengkap (*suite*) yang dapat digunakan untuk melakukan pengembangan aplikasi, baik itu aplikasi bisnis, aplikasi personal, ataupun komponen aplikasinya, dalam bentuk

aplikasi console, aplikasi Windows, ataupun aplisai Web. Visual Studio mencakup *compiler*, SDK, *Integrated Development Environment* (IDE), dan dokumentasi (umumnya berupa MSDN *Library*). *Compiler* yang dimasukkan ke dalam paket Visual Studio antara lain Visual C++, Visual C#, Visual Basic, Visual Basic.Net, Visual InterDev, Visual J++, Visual J#, Visual FoxPro, dan Visual SourceSafe.

Microsoft Visual Studio dapat digunakan untuk mengembangkan aplikasi dalam *native code* (dalam bentuk bahasa mesin yang berjalan diatas Windows) ataupun *managed code* (dalam bentuk *Microsoft Intermediate Language* diatas .Net *Framework*). Selain itu, Visual Studi juga dapat digunakan untuk mengembangkan aplikasi *Silverlight*, aplikasi *Windows Mobile* (yang berjalan diatas .Net *Compact Framework*).

II.5. Database

Database merupakan kumpulan data yang saling berhubungan, hubungan antar data dapat ditunjukkan dengan adanya *field* kunci dari setiap tabel yang beda. Dalam satu *file* atau tabel terdapat *record-record* yang sejenis, sama besar, sama bentuk, yang merupakan satu kumpulan entitas yang seragam. Satu *record* terdiri dari *field* yang saling berhubungan menunjukkan bahwa *fiel* tersebut satu pengertian yang lengkap dan disimpan dalam satu *record*. Basis data mempunyai beberapa kriteria penting yaitu :

1. Bersifat data oriented dan bukan program oriented.

2. Dapat digunakan oleh beberapa program aplikasi tanpa perlu mengubah basis datanya.
3. Dapat dikembangkan dengan mudah, baik *volume* maupun strukturnya.
4. Dapat memenuhi kebutuhan sistem-sistem baru secara mudah.
5. Dapat digunakan dengan cara-cara yang berbeda.

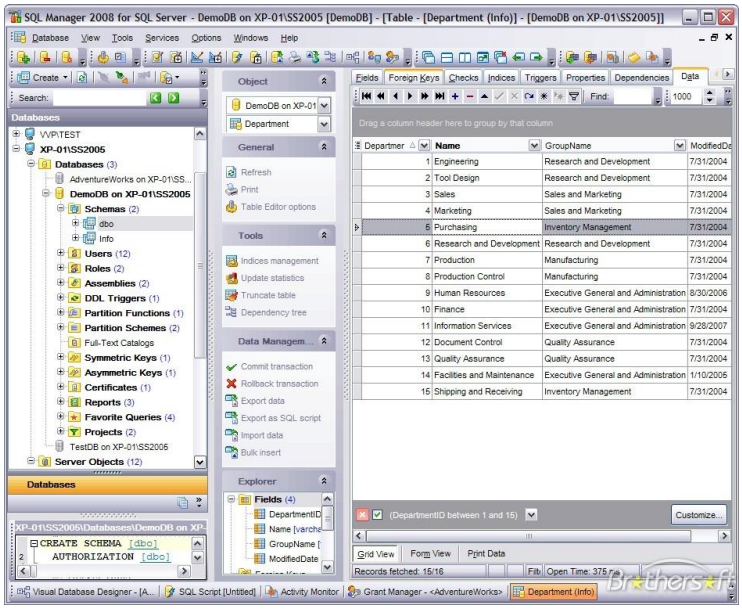
Prinsip utama *database* adalah pengaturan data dengan tujuan utama fleksibel dan kecepatan pada saat pengambilan data kembali. Adapun ciri-ciri basis data di antaranya adalah sebagai berikut :

1. Efesiensi meliputi kecepatan, ukuran dan ketepatan.
2. Data dalam jumlah besar.
3. Berbagi pakai (dipakai bersama-sama atau *sharebility*).

Mengurangi bahkan menghilangkan terjadinya duplikasi dan data yang tidak konsisten (Windu Gata ; 2013 : 19).

II.6. *SQL Server*

SQL Server 2008 adalah sebuah RDBMS (*Relational Database Management System*) yang di-develop oleh *Microsoft*, yang digunakan untuk menyimpan dan mengolah data. Pada *SQL Server* 2008, kita bisa melakukan pengambilan dan modifikasi data yang ada dengan cepat dan efisien. Pada *SQL Server* 2008, kita bisa membuat object-object yang sering digunakan pada aplikasi bisnis, seperti membuat *database, table, funcation, stored procedure, trigger* dan *view*. Selain *object*, kita juga menjalankan perintah SQL (*Structured Query Language*) untuk mengambil data. (Cybertron Solution; 2010 : 101-102).



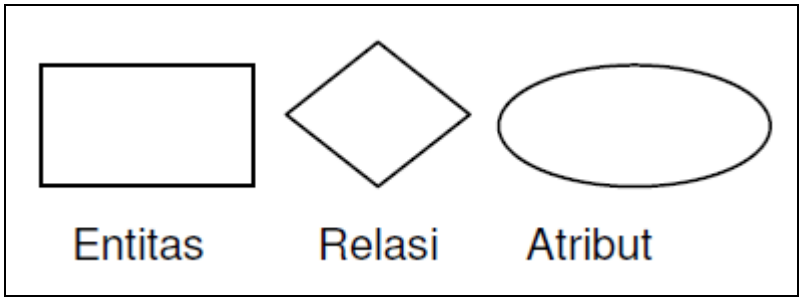
Gambar II.2. Tampilan SQL Server
(*Sumber : Cybertron Solution; 2010 : 101-102*)

II.7. *Entity Relationship Diagram* (ERD)

Model *entity-relationship* pertama kali diperkenalkan oleh Peter Chen pada tahun 1976. Dalam pemodelan ini dilakukan dengan tahapan sebagai berikut:

a. Memilih entitas-entitas yang akan disusun dalam basis data dan menentukan hubungan antar entitas yang telah dipilih.
b. Melengkapi atribut-atribut yang sesuai pada entitas dan hubungan sehingga diperoleh bentuk tabel normal penuh (ternormalisasi).

Elemen-elemen dalam model ER dapat digambarkan pada gambar diagram di bawah ini :



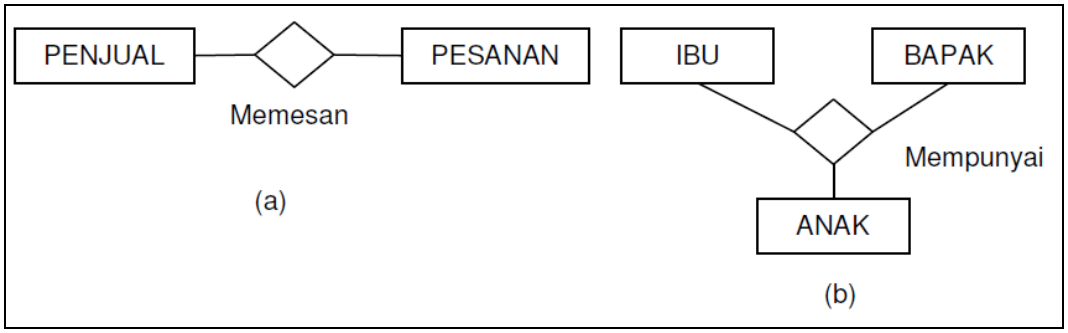
Gambar II.3. Elemen ERD
(*Sumber : Haidar Dzacko, 2007 : 21*).

Entitas merupakan sesuatu yang dapat diidentifikasikan dalam lingkungan kerja pengguna. Entitas yang diberikan tipe dikelompokkan ke kelas entitas. Perbedaan antara kelas entitas dan instansi entitas adalah sebagai berikut:

a. Kelas entitas adalah kumpulan entitas dan dijelaskan oleh struktur atau format entitas di dalam kelas.
b. Instansi kelas merupakan bentuk penyajian dari fakta entitas.

Umumnya terdapat banyak instansi entitas di dalam setiap entitas kelas. Setiap entitas kelas memiliki atribut yang menjelaskan karakteristik dari entitas tersebut, sedangkan setiap instansi entitas mempunyai identifikasi yang dapat bernilai unik (mempunyai nilai yang berbeda untuk setiap identifikasinya) atau non-unik (dapat bernilai sama untuk setiap identifikasinya).

Antara entitas diasosiakan dalam suatu hubungan (*relationship*). Suatu relasi dapat memiliki beberapa atribut. Jumlah kelas entitas dalam suatu relasi disebut derajat relasi. Gambar di bawah ini merupakan contoh dari relasi berderajat dua dan relasi berderajat tiga (Haidar Dzacko ; 2007 : 21).



Gambar II.4. (a) Relasi Berderajat Dua (b) Relasi Berderajat Tiga
(*Sumber : Haidar Dzacko, 2007 : 21*)

II.8. Kamus Data

Kamus data (data *dictionary*) dipergunakan untuk memperjelas aliran data yang di gambarkan pada DFD. Kamus data adalah kumpulan daftar elemen data yang mengalir pada system perangkat lunak sehingga masukan *(input)* dan eluaran *(output)* dapat di pahami secara umum (memiliki standar cara penulisan). Kamus data data implementasi program dapat menjadi parameter masukan atau keluaran dari sebuah fungsi atau prosedur (Rosa A.S, 2015:73)

II.9. Teknik Normalisasi

Normalisasi adalah teknik perancangan yang banyak digunakan sebagai pemandu dalam merancang basis data relasional. Pada dasarnya, normalisasi adalah proses dua langkah yang meletakkan data dalam bentuk tabulasi dengan menghilangkan kelompok berulang lalu menghilangkan data yang terduplikasi dari tabel rasional.

Teori normalisasi didasarkan pada konsep bentuk normal. Sebuah tabel relasional dikatakan berada pada bentuk normal tertentu jika tabel memenuhi himpunan batasan tertentu. Ada lima bentuk normal yang tekah ditemukan.

II.9. Bentuk-bentuk Normalisasi

1. **Bentuk normal tahap pertama (1" Normal Form)**

   Contoh yang kita gunakan di sini adalah sebuah perusahaan yang mendapatkan barang dari sejumlah pemasok. Masing-masing pemasok berada pada satu kota. Sebuah kota dapat mempunyai lebih dari satu pemasok dan masing-masing kota mempunyai kode status tersendiri.

2. **Bentuk normal tahap kedua (2^{nd} normal form)**

   Definisi bentuk normal kedua menyatakan bahwa tabel dengan kunci utama gabungan hanya dapat berada pada 1NF, tetapi tidak pada 2NF. Sebuah tabel relasional berada pada bentuk normal kedua jika dia berada pada bentuk normal kedua jika dia berada pada 1NF dan setiap kolom bukan kunci yang sepenuhnya tergantung pada seluruh kolom yang membentuk kunci utama.

3. **Bentuk normal tahap ketiga (3^{rd} normal form)**

   Bentuk normal ketiga mengharuskan semua kolom pada tabel relasional tergantung hanya pada kunci utama. Secara definisi, sebuah tabel berada pada bentuk normal ketiga (3NF) jika tabel sudah berada pada 2NF dan setiap kolom yang bukan kunci tidak tergantung secara transitif pada kunci utamanya.

4. **Boyce Code Normal Form (BCNF)**

   Setelah 3NF, semua masalah normalisasi hanya melibatkan tabel yang mempunyai tiga kolom atau lebih dan semua kolom adalah kunci. Banyak praktisi berpendapat bahwa menempatkan entitas pada 3NF

sudah cukup karena sangat jarang entitas yang berada pada 3NF bukan merupakan 4NF dan 5NF.

5. Bentuk Normal Tahap Keempat dan Kelima

Sebuah tabel relasional berada pada bentuk normal keempat (4NF) jika dia dalam BCNF dan semua ketergantungan multivalue merupakan ketergantungan fungsional. Bentuk normal keempat (4NF) didasarkan pada konsep ketergantungan multivalue (MVD).

Sebuah tabel berada pada bentuk normal kelima (5NF) jika ia tidak dapat mempunyai dekomposisi lossless menjadi sejumlah tabel lebih kecil. Empat bentuk normal pertama berdasarkan pada konsep ketergantungan fungsional, sedangkan bentuk normal kelima berdasarkan pada konsep ketergantungan gabungan (*join dependence*) (Janner Simarmata, 2010 : 76 - 86).

II.10. UML (*Unified Modeling Language*)

Menurut Windu Gata (2013 : 4) Hasil pemodelan pada OOAD terdokumentasikan dalam bentuk *Unified Modeling Language* (UML). UML adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak.

UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar

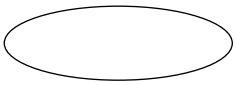
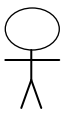
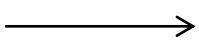
bahasa pemodelan umum dalam industri perangkat lunak dan pengembangan sistem.

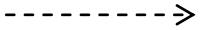
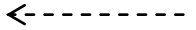
Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut :

1. *Use case* Diagram

   Use case diagram merupakan pemodelan untuk kelakukan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dapat dikatakan *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Simbol-simbol yang digunakan dalam *use case* diagram, yaitu :

Tabel II.1. Simbol *Use Case*

Gambar	Keterangan
	*Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukan pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama *use case*.
	Aktor adalah *abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan aktor, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa muncul dalam beberapa peran. Perlu dicatat bahwa aktor berinteraksi dengan *use case*, tetapi tidak memiliki control terhadap *use case*.
	Asosiasi antara aktor dan *use case*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengidikasikan aliran data.
	Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengidinkasikan bila aktor berinteraksi secara pasif dengan sistem.

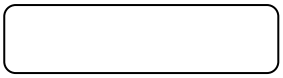
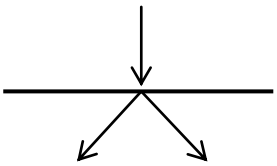
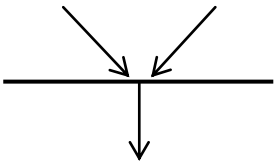
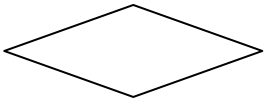
	*Include*, merupakan di dalam *use case* lain (*required*) atau pemanggilan *use case* oleh *use case* lain, contohnya adalah pemanggilan sebuah fungsi program.
	*Extend,* merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.

(Sumber : Windu Gata ; 2013 : 4)

2. Diagram Aktivitas (*Activity Diagram*)

Activity Diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Simbol-simbol yang digunakan dalam *activity diagram*, yaitu :

Tabel II.2. Simbol *Activity Diagram*

Gambar	Keterangan
	*Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	*End point*, akhir aktifitas.
	*Activites*, menggambarkan suatu proses/kegiatan bisnis.
	*Fork* (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan pararel menjadi satu.
	*Join* (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi.
	*Decision Points*, menggambarkan pilihan untuk pengambilan keputusan, *true, false.*

New Swimline	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa.

(Sumber : Windu Gata ; 2013 : 6)

3. *Class Diagram* (Diagram Kelas)

Merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggng jawab entitas yang menentukan perilaku sistem.

Class diagram juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan. *Class diagram* secara khas meliputi: Kelas (*Class*), Relasi, *Associations*, *Generalization* dan *Aggregation*, Atribut (*Attributes*), Operasi (*Operations/Method*), *Visibility*, tingkat akses objek eksternal kepada suatu operasi atau atribut. Hubungan antar kelas mempunyai keterangan yang disebut dengan *multiplicity* atau kardinaliti.

Tabel II.3. *Multiplicity Class Diagram*

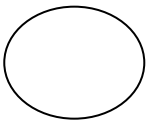
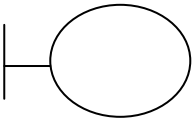
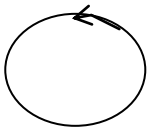
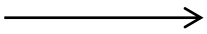
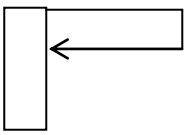
Multiplicity	**Penjelasan**
1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1 atau lebih
1..*	1 atau lebih
0..1	Boleh tidak ada, maksimal 1
n..n	Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimum 4

(Sumber : Windu Gata ; 2013 : 9)

4. Diagram Urutan (*Sequence Diagram*)

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Simbol-simbol yang digunakan dalam *sequence diagram*, yaitu :

Tabel II.4. Simbol *Sequence Diagram*

Gambar	Keterangan
	*Entity Class,* merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data.
	*Boundary Class,* berisi kumpulan kelas yang menjadi *interface* atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan formentry dan *form* cetak.
	*Control class,* suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek.
	*Message*, simbol mengirim pesan antar *class*.
	*Recursive,* menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri.
	*Activation, activation* mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivitas sebuah operasi.
	Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation*.

(Sumber : Windu Gata ; 2013 : 7)