BAB II

TINJAUAN PUSTAKA

II.1. Pengertian Sistem

Sistem merupakan kumpulan dari unsur atau elemen-elemen yang saling berkaitan / berinteraksi dan saling mempengaruhi dalam melakukan kegiatan bersama untuk mencapai suatu tujuan tertentu. Contoh : Sistem komputer, terdiri dari software, hardware, dan brainware (Asbon Hendra, S.Kom.;2012;157).

II.1.1. Karakteristik Sistem

a. Komponen(*Component*)

Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, bekerja sama membentuk suatu kesatuan. Komponen-komponen sistem dapat berupa suatu subsistem atau bagian-bagian dari sistem. Setiap sistem,tidak peduli betapa pun kecilnya, selalu mengandung komponen-komponen atau subsistem-subsistem. Setiap subsistem mempunyai sifat-sifat dari sistem untuk menjalankan suatu fungsi tertentu dan mempengaruhi fungsi sistem secara keseluruhan. Suatu sistem dapat mempunyai suatu sistem yang lebih besar yang disebut supra sistem. Sebagai contoh,suatu perusahaan dapat disebut suatu sistem dan industri merupakan suatu sistem yang lebih besar dapat disebut supra sistem. Kalau dipandang industri sebagai suatu sistem,maka perusahaan dapat disebut sebagai subsistem. Demikian juga apabila perusahaan dipandang sebagai sistem, maka sistem akuntansi adalah subsistemnya.

b. Batas Sistem (*Boundary*)

Batas sistem merupakan daerah yang membatasi antara suatu sistem dengan sistem lainnya atau dengan lingkungan luarnya. Batas sistem ini memungkinkan suatu sistem dipandang sebagai suatu kesatuan, karena dengan batas sistem ini fungsi dan tugas dari subsistem yang satu dengan yang lainnya berbeda tapi tetap saling berinteraksi. Batas suatu sistem menunjukkan ruang lingkup(*scope*) dari sistem tersebut.

c. Lingkungan Luas Sistem (*Environment*)

Environment merupakan segala sesuatu dari luar batas sistem yang mempengaruhi operasi dari suatu sistem. Lingkungan luar sistem ini dapat bersifat menguntungkan atau merugikan. Lingkungan luar yang menguntungkan harus dipelihara dan dijaga agar tidak hilang pengaruhnya, sedangkan lingkungan luar yang merugikan harus dimusnahkan atau dikendalikan agar tidak mengganggu operasi sistem.

d. Penghubung Sistem (*Interface*)

Merupakan media penghubung antara satu subsitem dengan subsistem lainnya untuk membentuk satu kesatuan sehingga sumber-sumber daya mengalir dari subsistem yang satu ke subsistem yang lainnya. Dengan kata lain, *outpput* dari suatu subsistem akan menjadi *input* dari subsistem lainnya.

e. Masukan Sistem (*Input*)

Merupakan energi yang dimasukkan ke dalam sisten. Masukan dapat berupa masukan perawatan (*maintenance input*) adalah energi yang

dimasukkan supaya sistem tersebut dapat beroperasi. Masukan sinyal (*signal output*) adalah energi yang diproses untuk didapatkan keluaran. Sebagai contoh , didalam sistem komputer, program adalah *maintenance input* yang digunakan untuk mengoperasikan komputernya dan data adalah *signal output* untuk diolah menjadi informasi.

f. Keluaran Sistem (*Output*)

Merupakan hasil dari energi yang diolah sistem, meliputi *output* yang berguna, contohnya informasi yang dikeluarkan oleh komputer. Dan *output* yang tidak berguna dikenal sebagai sisa pembuangan, contohnya panas yang dikeluarkan oleh komputer.

g. Pengolah Sistem (*Process*)

Merupakan bagian yang memproses masukan untuk menjadi keluaran yang diinginkan. Contoh CPU pada komputer, bagian produksi yang mengubah bahan baku menjadi barang jadi, serta bagian akuntansi yang mengolah data transaksi menjadi laporan keuangan.

h. Tujuan Sistem (*Goal*)

Setiap sistem pasti mempunyai tujuan apapun sasaran yang mempengaruhi input yang dibutuhkan dan output yang dihasilkan. Dengan kata lain, suatu sistem akan dikatakan berhasil kalau pengoperasian sistem itu mengenai sasaran atau tujuannya. Jika sistem tidak mempunyai sasaran, maka operasi sistem tidak akan ada gunanya (Asbon Hendra, S.Kom. ; 2012 ; 156-160).

II.1.2. Siklus Hidup Pengembangan Sistem

Siklus hidup pengembangan sistem terdiri dari 5 (lima) tahapan, yaitu :

1. Sistem Investigasi (*Investigation System*)

Tahap ini meliputi pertimbangan dari usulan yang dihasilkan oleh proses perencanaan IT / bisnis. Tahap ini juga meliputi pembelajaran awal dari solusi sistem informasi yang diusulkan untuk menemukan prioritas dan kesempatan bisnis sebuah perusahaan yang diidentifikasi dalam proses perencanaan.

Dalam tahap ini, pengembang sistem melakukan perencanaan mengenai sistem informasi yang akan dibuat. Seberapa besar perubahan yang harus dibuat dari sistem awal, infrastruktur apa saja yang dibutuhkan, berapa besar *cost* pengembangan dan *benefit* yang nantinya akan dihasilkan. Hasil akhir dari tahap perencanaan ini adalah proposal proyek atau dokumen perencanaan proyek.

2. Sistem Analisis (*Analysis System*)

Sistem analisis menggambarkan apa yang harus dilakukan sistem untuk menemukan informasi yang dibutuhkan oleh pemakai. Pembelajaran sistem analisis pada umumnya meliputi :

1. Informasi yang dibutuhkan oleh perusahaan dan pemakai akhir.
2. Aktifitas, sumber daya, dan produk dari satu ataunlebih sistem informasi yang digunakan saat ini.

3. Kemampuan sistem informasi yang dibutuhkan untuk menemukan informasi yang diperlukan, dan pemegang saham bisnis lainnya yang menggunakan sistem.

Dalam tahap ini, pengembang sistem melakukan analsis mengenai data-data apa saja yang harus dikelola, informasi apa saja yang harus dihasilkan, apa saja *Entitas* dan bagaimana *Relationship*nya. Hasil dari tahap ini adalah ER-Diagram. Selain itu, analisis mengenai pengendalian internal (*internal control*) juga perlu dilakukan. Sistem informasi sangat terkait dengan SPI (*Struktur Pengendalian Internal*), karena informasi yang dihasilkan dari sistem informasi harus memenuhi karakteristik kualitatif informasi. Untuk dapat memenuhi karakteristik kualitatif informasi tersebut, sistem informasi harus digunakan juga sebagai bagian dari SPI. Adapun komponen dari SPI adalah Lingkungan Pengendalian, Penilaian Risiko, Aktivitas Pengendalian, Informasi dan Komunikasi, Pengawasan (*Monitoring*). Dalam tahap Aktivitas Pengendalian, terdapat Pengendalian Umum (*General Control*) dan Pengendalian Aplikasi (*Application Control*).

3. Sistem Perancangan (*Design System*)

Sistem perancangan menjelaskan bagaimana sistem akan menyelesaikan tujuan ini. Sistem perancangan terdiri dari aktivitas perancanangan (*hardware, software, people, network,* dan *resources*) yang menghasilkan spesifikasi sistem yang memenuhi kebutuhan fungsional yang dikembangkan dalam proses sistem analisis.

Dalam tahap ini, pengembang sistem merancang sistem informasi dalam DBMS (*Database Management System*). ER-Diagram dan pengendalian atas risiko yang mungkin muncul, diterapkan dalam rancangan aplikasi menggunakan DBMS, sehingga akan menghasilkan aplikasi sistem informasi. Bila lebih mutakhir, aplikasi sistem informasi dapat dibuat terintegrasi antar siklus.

4. Sistem Implementasi (*Implementation System*)

Ketika sistem informasi yang baru telah selesai dirancang maka harus diterapkan dan dipelihara agar dapat beroperasi dengan baik. Tahap ini meliputi pengujian sistem, pelatihan *user* untuk mengoperasikan sistem baru, mengubah sistem lama ke sistem bisnis yang baru, dan mengatur akibat dari perubahan sistem pada pemakai akhir. Implementasi adalah tahap penting dalam pengembangan teknologi informasi untuk mendukung karyawan, pelanggan, dan pemegang saham perusahaan bisnis lainnya. Implementasi merupakan proses yang sulit dan memakan waktu. Bagaimanapun tahap ini penting dalam memastikan kesuksesan dari pengembangan sistem yang baru, bahkan sistem yang dirancang dengan baik sekalipun dapat gagal jika diterapkan dengan tidak baik.

Dalam tahap ini, pengembang sistem mengimplementasikan sistem informasi dalam organisasi. Permasalahan yang biasa terjadi adalah penolakan karyawan atas sistem baru (*user resistance*). Ada beberapa metoda yang dapat digunakan untuk mengatasi permasalahan ini seperti *phased in*, *parallel*, *direct*, *big-bang*, dan lain sebagainya.

5. Sistem Pemeliharaan (*Maintance System*)

Sistem pemeliharaan meliputi pengawasan, evaluasi, dan modifikasi sistem operasional bisnis untuk membuat peningkatan sesuai dengan yang dibutuhkan. Pemeliharaan juga penting bagi masalah lain yang timbul selama pengoperasian sistem. Aktivitas pemeliharaan meliputi proses peninjauan sesudah tahap implementasi untuk memastikan bahwa sistem baru yang diimplementasikan memenuhi tujuan bisnis yang dibangun. Pemeliharaan juga meliputi pembuatan modifikasi untuk membangun sistem selama perubahan dalam lingkungan bisnis.

Dalam tahap ini, sistem yang sudah diterapkan diperiksa secara berkala. *Bugs-bugs* yang muncul dibenahi, pemutakhiran *field* dalam *table* dilakukan jika terdapat transaksi atau data baru, atau pengelolaan konsistensi data. O'Brien (2006) (O'Brien. ; 2006 ; 56-6s3).

II.2. Pengertian Informasi

Informasi merupakan data yang telah di proses menjadi bentuk yang memiliki arti bagi penerima dan dapat berupa fakta, suatu nilai yang bermanfaat. Jadi, ada suatu proses transformasi data menjadi suatu informasi = input-proses-output.

Data merupakan *raw material* untuk suatu informasi. Perbedaan informasi dan data sangat relatif, tergantung pada nilai gunanya bagi manajemen yang memerlukan. Suatu informasi bagi level manajemen tertentu bisa menjadi data bagi manajemen level di atasnya, atau sebaliknya.

Kuantitas informasi merupakan satuan ukuran informasi, tergantung representasi. Untuk representasi biner, satuannya bit, byte, word, dan lain-lain.

Kualitas informasi adalah bias terhadap eror karena kesalahan cara pengukuran dan pengumpulan, kegagalan mengikuti prosedur pemrosesan, kehilangan atau data tidak terproses, kesalahan prosedur perekaman atau koreksi data, kesalahan file histori/ master, kesalahan prosedur pemrosesan, serta ketidakberfungsian sistem.

Umur informasi, kapan atau sampai kapan sebuah informasi memiliki nilai/ arti bagi penggunanya. Ada *condition information* (mengacu pada titik waktu tertentu) dan *operating information* (menyatakan suatu perubahan pada suatu *range* tertentu).

Nilai informasi, ditentukan dari dua hal, yaitu manfaat dan biaya mendapatkannya. Suatu informasi dikatakan bernilai bila manfaatnya lebih efektif dibandingkan dengan biaya mendapatkannya. Pengukuran nilai informasi biasanya dihubungkan dengan analisis *cost effectiveness* atau *cost benefit*.
(Asbon Hendra, S.Kom. ; 2012 ; 156-160).

II.3. Pengertian Sistem Informasi

Sistem informasi adalah suatu sistem terintegrasi yang mampu menyediakan informasi yang bermanfaat bagi penggunanya. Sebuah sistem terintegrasi atau sistem manusia-mesin, untuk menyediakan informasi untuk mendukung operasi, manajemen dalam suatu organisasi. Ada empat operasi dasar dari sistem informasi, yaitu mengumpulkan, mengolah, menyimpan, dan

menyebarkan informasi. Informasi mungkin dikumpulkan dari lingkungan dalam atau luar dan memungkinkan didistribusikan ke dalam atau luar organisasi. Contoh sebuah sistem informasi penjualan : pengumpulan data transaksi dan faktur penjualan. Sistem ini memanfaatkan perangkat keras dan perangkat lunak komputer, prosedur manual, model manajemen, dan basis data (Asbon Hendra, S.Kom;2012;168-169).

II.4. Pengertian Penjualan

Penjualan merupakan tujuan utama dilakukannya kegiatan perusahaan. Perusahaan dalam menghasilkan barang/ jasa mempunyai tujuan akhir, yaitu untuk menjual barang/ jasa tersebut. Oleh karena itu, penjualan memegang peranan penting bagi perusahaan agar produk yang dihasilkan oleh perusahaan dapat terjual dan memberikan penghasilan bagi perusahaan. Penjualan yang dilakukan perusahaan bertujuan untuk menjual barang/ jasa yang diperlukan sebagai sumber pendapatan untuk menutup semua ongkos untuk memperoleh data.

Penjualan adalah pemindahan hak milik atas barang atau pemberian jasa yang dilakukan penjualan kepada pembeli dengan harga yang disepakati bersama dengan jumlah yang dibebankan kepada pelanggan dalam penjualan barang/jasa dalam suatu periode akuntansi.

Penjualan merupakan pengalihan hak milik atas barang dengan imbalan uang sebagai gantinya dengan persetujuan untuk menyerahkan barang kepada pihak lain dengan menerima pembayaran. (Freddy Rangkuti ; 2010 ; 57).

II.5. Entity Relationship Diagram (ERD)

Entity Relationship Diagram atau ERD adalah sebuah diagram yang secara konseptua; memetakan hubungan antara penyimpanan pada diagram DFD di atas. ERD ini digunakan untuk melakukan permodelan terhadap struktur data dan hubungannya. Penggunaan ERD ini dilakukan untuk mengurangi tingkat kerumitan penyusunan sebuah database yang baik. (Wahana Komputer ; 2010 ; 30).

II.6. Kamus Data

Kamus data (*data dictionary*) mencakup definisi-definisi dari data yang disimpan dalam basis data dan dikendalikan oleh sistem manajemen basis data. Struktur basis data yang dimuat dalam kamus data adalah kumpulan dari seluruh definisi field, definisi tabel, relasi tabel dan hal-hal lainnya. Nama field data, jenis data (seperti teks dan angka atau tanggal), nilai-nilai yang valid untuk data dan karakteristik-karakteristik lainnya akan disimpan dalam kamus data. Perubahan-perubahan pada struktur data hanya dilakukan satu kali di dalam kamus data; program program aplikasi yang mempergunakan data tidak akan ikut terpengaruh (Raymond McLeod Jr dan George P Schell ; 2007 ; 171).

II.7. Normalisasi

Menurut Kusrini (2007 : 39 – 43), salah satu topik yang cukup kompleks dalam dunia manajemen *database* adalah proses untuk menormalisasi tabel-tabel dalam *database relasional.*

Dengan normalisasi kita ingin mendesain *database relasional* yang terdiri dari tabel-tabel berikut :

1. Berisi data yang diperlukan.
2. Memiliki sesedikit mungkin redundansi.
3. Mengakomodasi banyak nilai untuk tipe data yang diperlukan.
4. Mengefisienkan update.
5. Menghindari kemungkinan kehilangan data secara tidak disengaja/tidak diketahui.

Alasan utama dari normalisasi *database* minimal sampai dengan bentuk normal ketiga adalah menghilangkan kemungkinan adanya "*insertion anomalies"*, "*deletion anomalies",* dan "*update anomalies".* Tipe-tipe kesalahan tersebut sangat mungkin terjadi pada *database* yang tidak normal.

II.7.1. Bentuk-bentuk Normalisasi

a. Bentuk tidak normal

Bentuk ini merupakan kumpulan data yang akan direkam, tidak ada keharusan mengikuti format tertentu, dapat saja tidak lengkap dan terduplikasi. Data dikumpulkan apa adanya sesuai keadaanya.

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

Definisi :

Sebuah table disebut 1NF jika :

- Tidak ada baris yang duplikat dalam tabel tersebut.
- Masing-masing cell bernilai tunggal

Catatan: Permintaan yang menyatakan tidak ada baris yang duplikat dalam sebuah tabel berarti tabel tersebut memiliki sebuah kunci, meskipun kunci tersebut dibuat dari kombinasi lebih dari satu kolom atau bahkan kunci tersebut merupakan kombinasi dari semua kolom.

c. Bentuk normal tahap kedua (2nd normal form)

Bentuk normal kedua (2NF) terpenuhi jika pada sebuah tabel semua atribut yang tidak termasuk dalam primary key memiliki ketergantungan fungsional pada primary key secara utuh.

d. Bentuk normal tahap ketiga (3rd normal form)

Sebuah tabel dikatakan memenuhi bentuk normal ketiga (3NF), jika untuk setiap ketergantungan fungsional dengan notasi X -> A, dimana A mewakili semua atribut tunggal di dalam tabel yang tidak ada di dalam X, maka :

- X haruslah superkey pada tabel tersebut.
- Atau A merupakan bagian dari primary key pada tabel tersebut.

e. Bentuk Normal Tahap Keempat dan Kelima

Penerapan aturan normalisasi sampai bentuk normal ketiga sudah memadai untuk menghasilkan tabel berkualitas baik. Namun demikian, terdapat pula bentuk normal keempat (4NF) dan kelima (5NF). Bentuk Normal keempat berkaitan dengan sifat ketergantungan banyak nilai (*multivalued dependency*) pada suatu tabel yang merupakan pengembangan dari ketergantungan fungsional. Adapun bentuk normal

tahap kelima merupakan nama lain dari *Project Join Normal Form* (PJNF).

f. Boyce Code Normal Form (BCNF)

- Memenuhi 1st NF
- Relasi harus bergantung fungsi pada atribut superkey.

II.8. Pengertian Database

Database atau basis *data* adalah sekumpulan *data* yang memiliki hubungan secara logika dan diatur dengan susunan tertentu serta disimpan dalam media penyimpanan komputer. Data itu sendiri adalah representasi dari semua fakta yang ada pada dunia nyata. *Database* sering digunakan untuk melakukan proses terhadap data-data tersebut untuk menghasilkan informasi. Dalam *database* ada sebutan-sebutan untuk satuan data yaitu :

1. Karakter, ini adalah satuan data terkecil. *Data* terdiri atas susunan karakter yang pada akhirnya mewakili data yang memiliki arti dari sebuah fakta.
2. *Field*, adalah kumpulan dari karakter yang memiliki fakta tertentu, misalnya seperti nama siswa, tanggal lahir, dan lain-lain.
3. *Record*, adalah kumpulan dari *field*. Pada *record* anda dapat menemukan banyak sekali informasi penting dengan cara mengombinasikan *field-field* yang ada.
4. Tabel, adalah sekumpulan dari *record-record* yang memiliki kesamaan entity dalam dunia nyata. Kumpulan tabel adalah *database* (Wahana Komputer ; 2010 ; 24).

II.9. Sekilas Tentang Visual Basic

Visual basic dibuat oleh microsoft, merupakan salah satu bahasa pemrograman berorientasi objek yang mudah dipelajari. Selain menawarkan kemudahan, Visual Basic juga cukup handal untuk digunakan dalam pembuatan berbagai aplikasi, terutama aplikasi database. Visual basic merupakan bahasa pemrograman event drive, dimana program akn menunggu sampai ada respons dari user/ pemakai program aplikasi yang dapat berupa kejadian atau event, misalnya ketika user mengklik tombol atau menekan enter. Jika kita membuat aplikasi dengan visual basic maka kita akan mendapatkan file yang menyusun aplikasi tersebut, yaitu :

1. File Project(*.vbp)

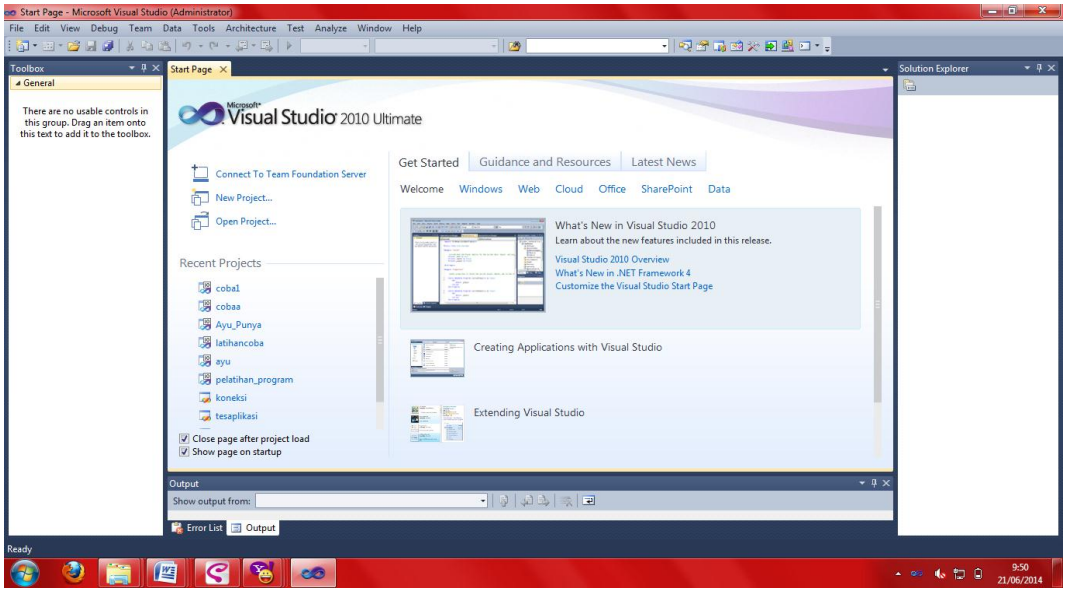
   File ini merupakan kumpulan dari aplikasi yang kita buat. File project bisa berupa file *.frm, *.dsr atau file lainnya.

2. File Form(*.frm)

   File ini merupakan file yang berfungsi untuk menyimpan informasi tentang bentuk form maupun interface yang kita buat, (Edy Winarto;2010:1).

Visual Basic merupakan salah satu bahasa pemrograman yang handal dan banyak digunakan oleh pengembang untuk membangun berbagai macam aplikasi windows. Visual basic 2008 atau visual basic 9 adalah versi terbaru yang telah diluncurkan oleh Microsoft bersama C#, Visual C++, dan visual web developer dalam satu paket visual studio 2008. Visual basic 2008 merupakan aplikasi pemrograman yang menggunakan tekhnologi .Net

Framework. Tekhnologi .Net Framework merupakan komponen windows yang terintegrasi serta mendukung pembuatan, penggunaan aplikasi, dan halaman web. Tekhnologi .Net Framework mempunyai 2 komponen utama yaitu, CLR (Common Language Runtime) dan class library. CLR digunakan untuk menjalankan aplikasi yang berbasis .Net, sedangkan library adalah kelas pustaka atau perintah yang digunakan untuk membangun aplikasi. (Wahana Komputer;2010:4).



Gambar II.1. Tampilan Visual Basic 2010

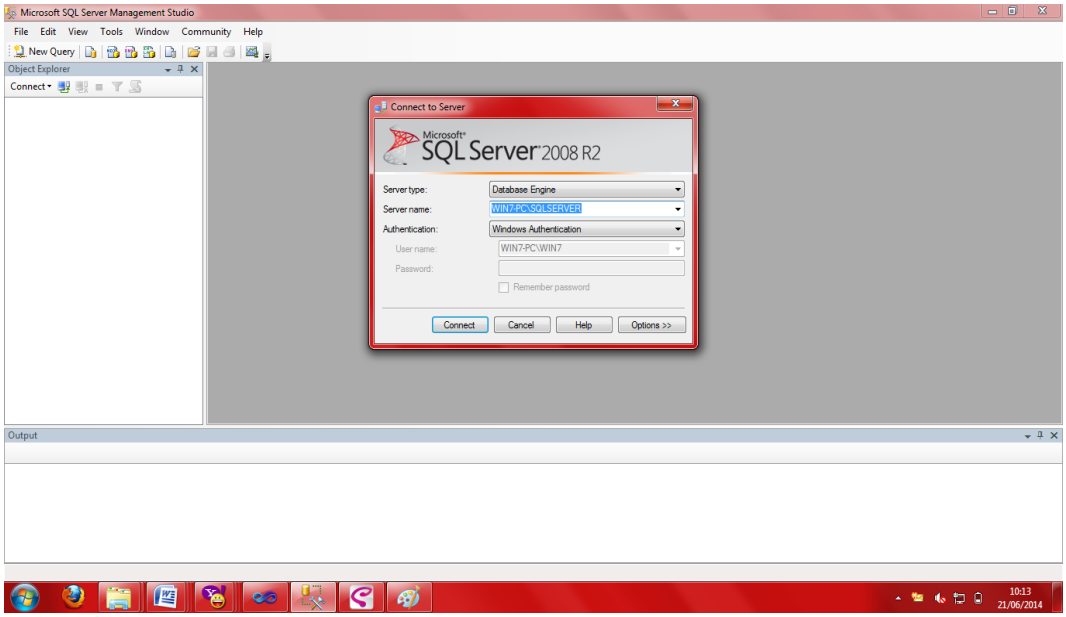
Sumber : Edy Winarto (2010:2)

II.10. SQL Server Management Studio 2008

SQL Server Management Studio 2008 adalah program yang dibuat oleh *microsoft* untuk membantu *user* ataupun admin melakukan tugas-tugas yang berhubungan dengan *server* database . *SQL Server* memiliki beberapa komponen penting yang memiliki kegunaannya dalam perancangan *database* , dan

melakukan pengaturan sistem secara keseluruhan . Komponen-komponen tersebut adalah :

- *Registered Server*
- *Object Explorer*
- *Query Editor* (Wahana Komputer;2010:40).



Gambar II.2. Tampilan *Microsoft SQL Server 2008* R2

Sumber : Wahana Komputer (2010 : 40)

II.11. *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 berorentasikan 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 Pudjo Widodo Herlawati ; 2011 ; 6).

II.11.1. *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 adal;ah 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.

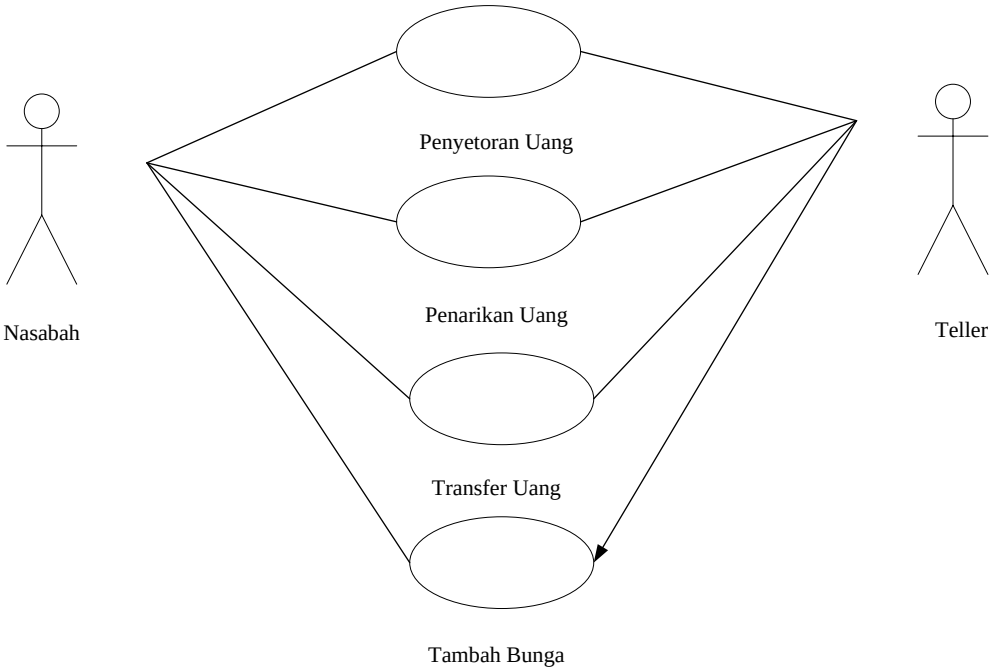
Diagram Use Case (use case diagram)

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.*

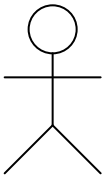


Gambar II.3. Diagram *Use Case*

Sumber : Probowo Pudjo Widodo(2011: 17)

1. 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*.

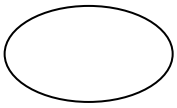


Gambar II.4. Aktor

Sumber : Probowo Pudjo Widodo (2011:17)

2. *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.5. Simbol *Use Case*

Sumber : Probowo Pudjo Widodo (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 dobel) 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 u*se case* pemesanan kamar, dibutuhkan pula *use case* pembatalan pesanan kamar.

e. Batasi use case hingga satu perilaku saja.

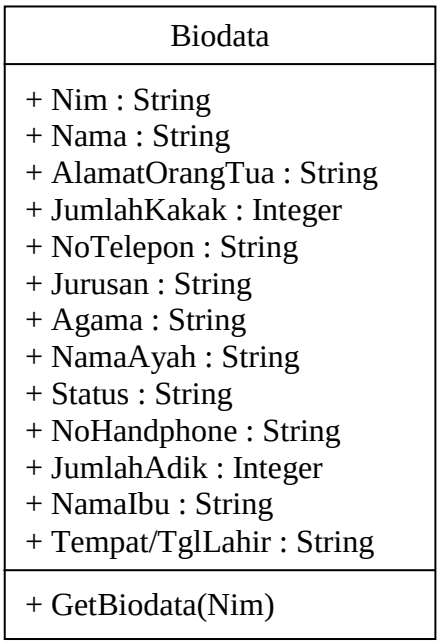
Kadang kita cenderung membuat *use case* yang lebih dari satu tujuan aktivitas. Guna menghindari kerancuan, jagalah use case kita hanya fokus pada satu hal. Misalnya, penggunaan *use case check in* dan *check out* dalam satu *use case* menghasilkan ketidakfokusan, karena memiliki dua perilaku yang berbeda.

3. Diagram Kelas (*Class Diagram*)

Diagram kelas mempunyai dua jenis yaitu *domain class diagram* dan *design class diagram*. Fokus *domain class diagram* adalah pada sesuatu dalam lingkungan kerja pengguna, bukan pada *class* perangkat lunak yang nantinya akan anda rancang. Sedangkan *design class diagram* tujuannya adalah untuk mendokumentasikan dan menggambarkan kelas-kelas dalam pemrograman yang nantinya akan dibangun.

Gambar II.6. Notasi *Domain Diagram Class*

Sumber : E. Triandini dan G. Suardika (2012 : 49 -50)



Gambar II.6. Notasi *Design Diagram Class*

Sumber : E. Triandini dan G. Suardika (2012 : 49 -50)

4. Diagram Aktivitas (*Activity Diagram*)

Diagram aktivitas lebih memfokuskan diri pada eksekusi dan alur sistemdari 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 (Probowo Pudji Widodo ;2011 : 143-145).

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

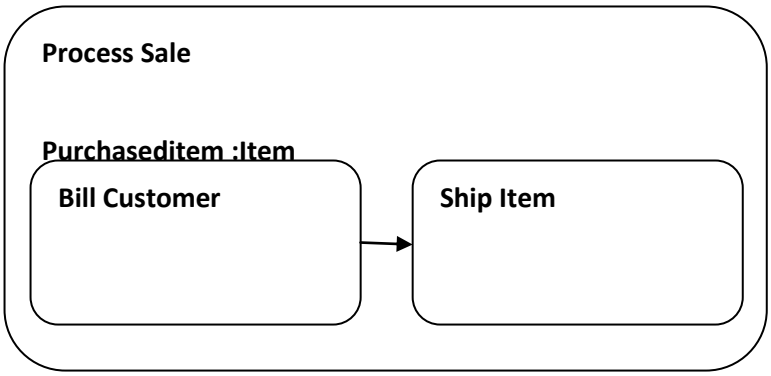
a. Fungsi Matematika
b. Pemanggilan Perilaku
c. Pemrosesan Data

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.

Gambar II.7. Aktivitas serderhana tanpa rincian

Sumber : Probowo Pudjo Widodo (2011:145)

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

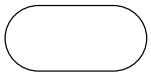


Gambar II.8. Aktivitas dengan detail rincian

Sumber : Probowo Pudjo Widodo (2011:145)

Berikut komponen-komponen dalam *activity diagram*, yaitu :

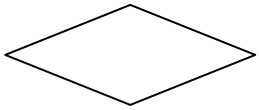
a. *Action State*, adalah langkah-langkah dalam sebuah *activity*. *Action* bisa terjadi saat memasuki *activity*, meninggalkan *activity*, atau pada *event* yang spesifik.



Gambar II.9. *Action State*

Sumber : Probowo Pudjo Widodo (2011:145)

b. *Decision*, menunjukkan dimana sebuah keputusan perlu dibuat dalam aliran kerja.



Gambar II.10. *Decision*

Sumber : Probowo Pudjo Widodo (2011:145)

c. *Transition*, menunjukkan bagaimana aliran kerja itu berjalan dari satu aktivitas ke aktivitas lainnya.

Gambar II.11 *Transition*

Sumber : Probowo Pudjo Widodo (2011:145)

d. *Syncronization*, menunjukkan dua atau lebih langkah dalam aliran kerja berjalan secara serentak *a Fork and Join.*

Gambar II.12. *Syncronization*

Sumber : Probowo Pudjo Widodo (2011:145)

e. *Swimlane*, menunjukkan siapa yang bertanggung jawab melakukan aktivitas dalam suatu diagram.

Gambar II.13. *Swimlane*

Sumber : Probowo Pudjo Widodo (2011:145)

f. *Star State*, memperlihatkan dimana aliran kerja berawal.

Gambar II.14. *Start State*

,Sumber : Adi Nugroho(2005:66)

g. End State, memperlihatkan dimana aliran kerja berakhir.

Gambar II.15. *End State*

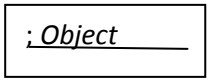
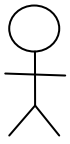
Sumber : Adi Nugroho(2005:66)

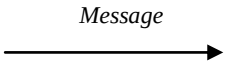
5. *Sequence Diagram*

Menurut John Satzinger, 2010, dalam buku *System Analysis and Design in a Changing World*, "System Sequence Diagram (SSD) adalah diagram yang digunakan untuk mendefinisikan input dan output serta urutan interaksi antara pengguna dan sistem untuk sebuah use case (E. Triandini dan G. Suardika ; 2012 : 71).

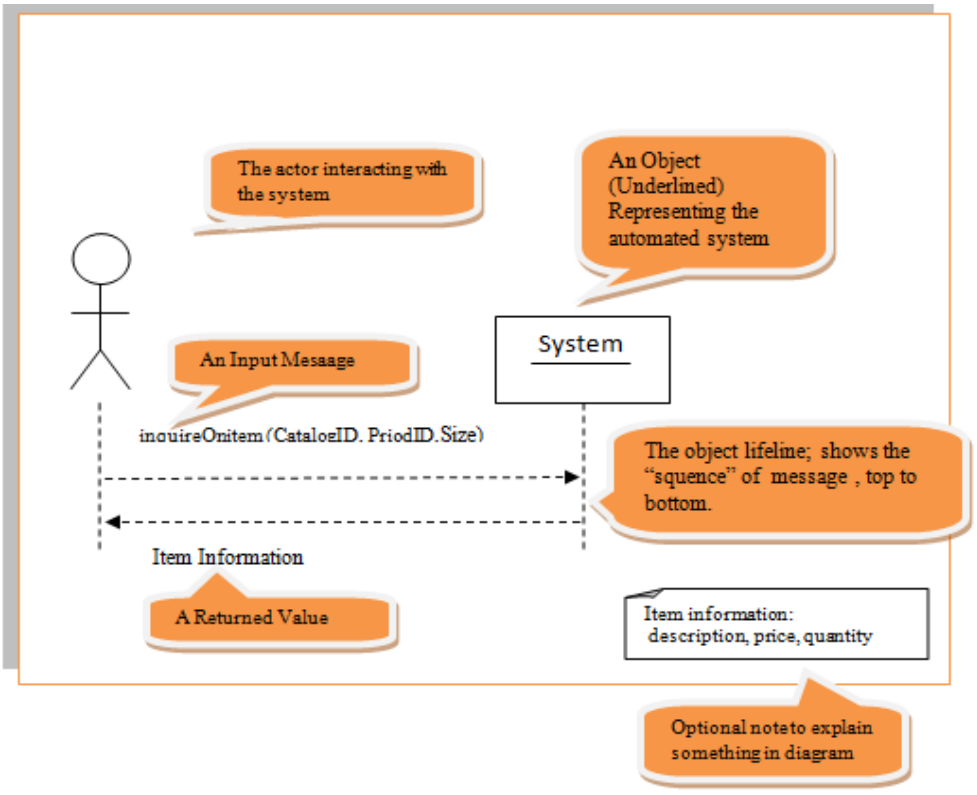
Berikut adalah notasi-notasinya :

Tabel II.1. Notasi Sequence Diagram

Object	*Object* merupakan instance dari sebuah *class* dan dituliskan tersusun secara horizontal. Digambarkan sebagia sebuah *class* (kotak) dengan nama obyek di dalamnya yan g diawali dengan sebuah titik koma.	
Actor	*Actor* juga dapat berkomunikasi dengan objek, maka *actor* juga dapat diurutkan sebagai kolom. Simbol *Actor* sama dengan simbol pada *Actor Use Case Diagram.*	

Lifeline	*Lifeline* mengindikasikan keberadaan sebuah object dalam basis waktu. Notasi untuk *Lifeline* adalah garis putus-putus vertikal yang ditarik dari sebuah obyek.	
Activation	*Activation* dinotasikan sebagai kotak segi empat yang digambar pada sebuah *Lifeline*. *Activation* mengindikasikan sebuah obyek yang akan melakukan sebuah aksi.	
Message	*Message* digambarkan dengan anak panah horizontal antara *activation*. *Mesaage* mengindikasikan komunikasi antara *object-object.*	

Sumber : Adi Nugroho (2005 : 92)



Gambar II.16. Notasi Sequence Diagram

Sumber : Evi Triandini dan Gede Suardika (2012 : 71)