BAB II

TINJAUAN PUSTAKA

II.1.1. Sistem

Sistem dapat diartikan *sebagai suatu kumpulan atau himpunan dari unsur, komponen, atau variabel yang terorganisasi, saling berinteraksi, saling tergantung satu sama lain dan terpadu*.

1. Teori sistem secara umum pertama kali diuraikan oleh Kenneth Bulding, terutama menekankan pentingnya perhatian terhadap setiap bagian yang membentuk sebuah sistem. Teori sistem mengatakan bahwa setiap unsur pembentuk organisasi itu penting dan harus mendapat perhatian yang utuh supaya manajer dapat bertindak lebih efektif.
2. Teori sistem melahirkan konsep-konsep futuristik, antara lain yang terkenal adalah konsep sibernetika (*cybernetics)*. Konsep atau bidang kajian ilmiah ini terutama berkaitan dengan upaya-upaya untuk menerapkan berbagai disiplin ilmu, yaitu ilmu perilaku, fisika, biologi, dan teknik.

Konsep lain yang terkandung di dalam definisi sistem adalah konsep sinergi. Konsep ini mengandaikan bahwa di dalam suatu sistem, output dari suatu organisasi diharapkan lebih besar daripada output individual atau output masing-masing bagian. Kegiatan bersama yang dilakukan oleh bagian yang terpisah tetap saling berhubungan akan menghasilkan efek total yang lebih besar daripada jumlah bagian secara individu terpisah. (Tata Sutabri ; 2012 : 3) .

II.1.2. Karakteristik Sistem

Sistem mempunyai beberapa karakteristik atau sifat-sifat tertentu, antara lain :

1. Komponen Sistem (*Component*)

   Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, yang bekerja sama membentuk satu kesatuan. Komponen-komponen sistem tersebut dapat berupa suatu bentuk subsistem.

2. Batasan Sistem (*Boundary*)

   Merupakan daerah yang membatasi antara sistem dengan sistem yang lainnya atau sistem dengan lingkungan luarnya.

3. Lingkungan Luar Sistem (*Environment*)

   Bentuk apapun yang ada diluar ruang lingkup atau batasan sistem yang mempengaruhi operasi sistem tersebut disebut dengan lingkungan luar sistem.

4. Penghubung Sistem (*Interface*)

   Media yang menghubungkan sistem dengan subsistem yang lain disebut dengan penghubung sistem atau interface. Penghubung ini menungkinkan sumber-sumber daya mengalir dari satu subsistem ke subsistem yang lain.

5. Masukan Sistem (*Input*)

   Energi yang dimasukkan ke dalam sistem disebut masukan sistem, yang dapat berupa pemeliharaan.

6. Keluaran Sistem (*Output*)

   Hasil energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna. Keluaran ini merupakan masukan bagin subsistem yang lain.

7. Pengolahan Sistem (*Process*)

   Suatu sistem dapat mempunyai suatu proses yang akan mengubah masukan menjadi keluaran.

8. Sasaran Sistem (*Object*)

   Suatu sistem memiliki tujuan dan sasaran yang pasti dan bersifat deterministik. Kalau suatu sistem tidak memiliki sasaran, maka operasi sistem tidak ada gunanya. (Tata Sutabri ; 2012 : 13-14).

II.1.3. Klasifikasi Sistem

Suatu sistem dapat diklasifikasikan menjadi seperti berikut :

1. Sistem abstrak dan fisik

   Sistem abstrak adalah suatu sistem yang berupa pemikiran atau ide-ide yang tidak tampak secara fisik, sedangkan sistem fisik adalah sistem yang ada secara fisik

2. Sistem alamiah dan sistem buatan manusia

   Sistem alamiah adalah sistem yang terjadi melalui proses alam, sedangkan sistem buatan manusia adalah sistem yang dirancang oleh manusia.

3. Sistem deterministik dan sistem probalistik

   Sistem deterministik yang beroperasi dengan tingkah laku yang dapat diprediksi disebut sistem deterministik. Sedangkan sistem yang bersifat probabilistik adalah sistem yang kondisi masa depannya tidak dapat diprediksi, karena mengandung unsur probilitas.

4. Sistem terbuka dan sistem tertutup

   Sistem tertutup adalah sistem yang tidak berhubungan dan tidak terpengaruh oleh lingkungan luarnya. sedangkan sistem terbuka adalah sistem yang berhubungan dan dipengaruhi oleh lingkungan luarnya, yang menerima masukan dan menghasilkan keluaran untuk subsistem lainnya. (Tata Sutabri ; 2012 : 15).

II.1.4. Pengertian Informasi

Informasi adalah data yang sudah diolah menjadi suatu bentuk sebuah bentuk yang berarti bagi penerimanya, dan bermanfaat dalam pengambilan keputusan pada saat sekarang atau yang akan datang. Informasi juga merupakan fakta-fakta atau data yang telah diproses sedemikian rupa atau mengalami proses transformasi data sehingga berubah bentuk menjadi informasi (Jurnal SAINTIKOM Vol 9, No. 2 Agustus 2010).

II.1.5. Kualitas Informasi

Informasi yang berkualitas memiliki 3 karakteristi, yaitu :

1. Akurat (*Accurate*)

   Informasi harus bebas dari kesalahan, tidak bias ataupun menyesatkan. Akurat juga berarti bahwa informasi itu harus dapat dengan jelas mencerminkan maksudnya.

2. Tepat pada waktunya (*timeliness*)

   Informasi yang sampai kepada si penerima tidak boleh terlambat, didalam Informasi yang sudah usang tidak akan mempunyai nilai lagi, karena informasi merupakan landasan di dalam pengambilan keputusan.

3. Relevan (*relevance*)

   Informasi tersebut mempunyai manfaat untuk pemakainya. Relevansi informasi untuk setiap orang berbeda. Menyampaikan informasi tentang penyebab kerusakan mesin produksi kepada akuntan perusahaan tentunya kurang relevan. (Tata Sutabri ; 2012 : 33-34).

II.1.6. Sistem Informasi Akuntansi

Sistem merupakan serangkaian bagian yang saling tergantung dan bekerja sama untuk mencapai tujuan tertentu. Suatu sistem pasti tersusun dari sub-sub sistem yang lebih kecil yang saling tergantung dan bekerja sama untuk mencapai tujuan (Anastasia Diana & Lilis Setiawati ; 2011: 3).

II.1.6.1.Sistem Informasi Akuntansi

Definisi sistem informasi akuntansi dalam buku yang berjudul *Analisis dan Desain Sistem Informasi* adalah sebagai berikut:

"Sistem Informasi Akuntansi adalah kumpulan kegiatan-kegiatan dari organisasi yang bertanggupng jawab untuk menyediakan informasi keuangan dan informasi yang didapat 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. (Hartono : 2005 : 17)

II.2. Pengertian Persediaan

Bagaimana perusahaan mengklasifikasikan persediaanya tergantung pada apakah perusahaan adalah pedagang (perusahaan dagang) atau pembuat (perusahaan manufaktur). Untuk perusahaan dagang, persediaanya dinamakan

persediaan barang dagangan (hanya ada satu klasifikasi), dimana barang dagangan ini dimiliki oleh perusahaan dan sudah langsung dalam bentuk siap untuk dijual dalam kegiatan bisnis normal sehari-hari. Sedangkan untuk perusahaan manufaktur, mula-mula persediaanya belum siap untuk dijual sehingga perlu diolah terlebih dahulu. Persediaanya diklasifikasikan menjadi tiga, yaitu bahan mentah, barang setengah jadi (barang dalam proses), dan barang jadi (produk akhir). Jadi, dalam perusahaan manufaktur, perusahaan jenis ini terlebih dahulu akan mengubah (merakit) input atau bahan mentah (*raw material*) menjadi output atau barang jadi (*finished goods/ final goods*), baru kemudian dijual kepada para pelanggan (distributor) mengenai kepemilikan barang, barang yang masih dalam perjalanan (good in transit) seharusnya masuk atau diperhitungkan sebagai bagian persediaan dari pihak yang memang secara hukum memiliki hak yang sah atas barang tersebut. Untuk tujuan akuntansi, hak kepemilikan barang biasanya ditentukan di awal transaksi jual-beli, yaitu berdasarkan pada perjanjian atau syarat-syarat penjualan yang disepakati. (Hery ; 2011 : 70).

II.3. Metode FIFO (*First In First Out*)

FIFO (*First In First Out*) yaitu metode penetapan harga pokok persedian yang didasarkan atas anggapan bahwa barang-barang terdahulu dibeli akan merupakan barang yang dijual pertama kali. Dalam metode ini persedian akhir (S.R.Soemarso; 2009:394). Jika perusahaan menggunakan metode FIFO, persedian akan dinilai dengan harga pembelian paling akhir. Apabila kuantitas

pada pembelian ini tidak cukup diterapkan pada persedian akhir, maka akan diambilkan dari pembelian terakhir berikutnya. (S.R.Soermarso; 2009: 388).

Dalam metode fisik dan perpetual, FIFO (*First In First Out*) merupakan barang yang masuk (dibeli atau diproduksi) lebih dahulu akan dikeluarkan (dijual) lebih dahulu. Sehingga yang tersisa pada akhir periode adalah barang yang berasal dari pembelian atau produksi terakhir (Rudianto; 2009: 239).

II.4. Visual Basic . Net

II.4.1. Sejarah Visual Basic . Net

Pada zaman dahulu ada sebuah bahasa pemrograman yang diberi nama Basic (*Beginner's All- purpose Symbolic Intruction Code*). Sesuai dengan namanya, Basic ditujukan sebagai bahasa yang paling sederhana bagi mereka yang tidak terlalu familiar dengan dunia pemrograman.

Pada tahun 1991 Microsoft mengeluarkan Visual Basic, pengembangan dari Basic yang berubah dari sisi pembuatan antarmukanya. Visual Basic sampai sekarang masih menjadi salah satu bahasa pemrograman terpopuler di dunia.

Pada akhir tahun 1999, Teknologi.NET di umumkan. Microsoft memosisikan teknologi tersebut sebagai platform untuk membangun XML *Web Service*. XML *Web Service* memungkinkan aplikasi tipe apa pun dapat berjalan pada system computer dengan type manapun dan dapat mengambil data yang tersimpan pada server dengan tipe apa pun melalui Internet.

Visual Basic.Net adalah Visual Basic yang direkayasa kembali untuk digunakan pada *platform*.Net sehingga aplikasi yang dibuat menggunakan Visual

Basic.Net dapat berjalan pada sistem computer apapun, dan dapat mengambil data dari server dengan tipe apapun asalkan terinstal .Net Framework.

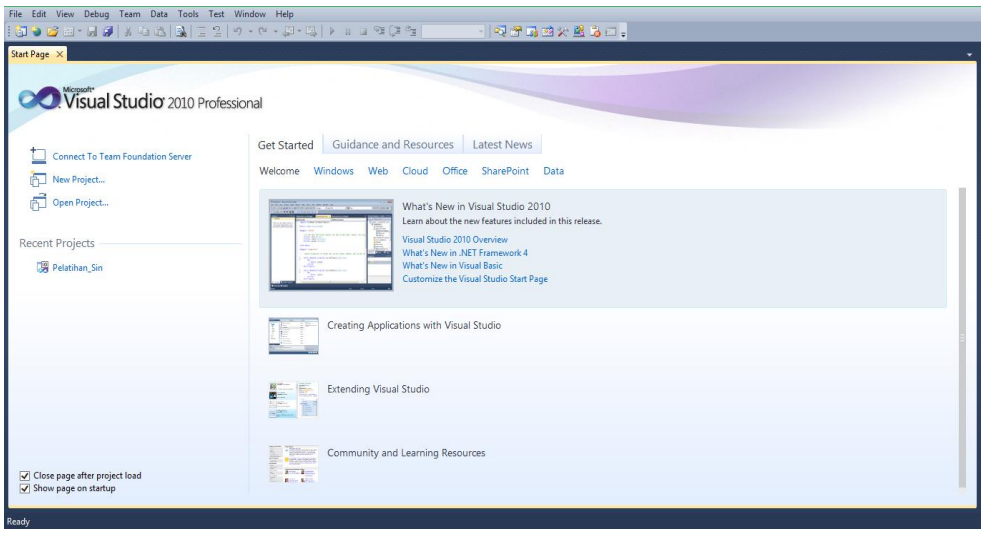
Berikut ini perkembangan Visual Basic .Net :

a. Visual Basic . Net 2002 (VB 7.0)
b. Visual Basic . Net 2003 (VB 7.1)
c. Visual Basic 2005 (VB 8.0)
d. Visual Basic 2008 (VB 9.0)
e. Visual Basic 2010 (VB 10.0)
f. Visual Basic 2012 (VB 11.0)
g. Visual Basic 2013

II.4.2. Berkenalan dengan Visual Basic . Net

II.4.2.1. IDE (Integrated Development Environment)

Pada waktu Visual Basic 2010 di jalankan, akan tampil sebuah Start Page seperti terlihat pada Gambar II.1.

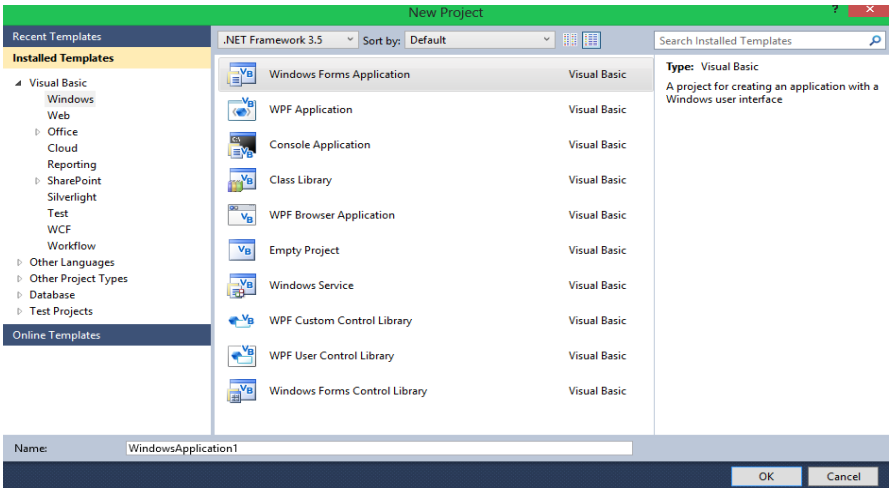


Gambar II.1. Start Page dari Visual studio 2010

Sumber : Priyanto Hidayatullah (2014 : 23)

Untuk membuka proyek yang ada gunakan tombol **Open Project** atau langsung mengklik pada daftar proyek yang ditampilkan sedangkan untuk membuaka sebuah proyek baru, klik tombol **New Project**.

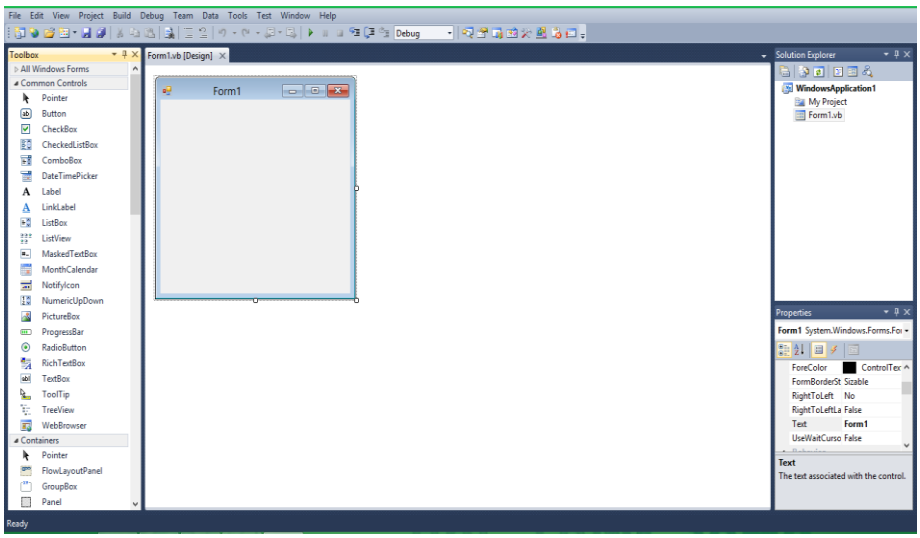
Setelah itu akan muncul kotak dialog **New Project**. Pada kotak pilih **Other Languages** > **Visual Basic** > **Windows** > **Windows Forms Application**. Untuk memberi nama proyek dapat dilakukan pada bagian **Name**, dan tekan OK (Gambar II.2.)



Gambar II.2. Kotak Dialog New Project

Sumber : Priyanto Hidayatullah (2014 : 24)

Pada **IDE Visual Basic 2010 (Gambar II.3.)** untuk Windows Application secara default telah terdapat sebuah form. Form tersebut bernama Form1. Pada form inilah tempat meletakkan kontrol- kontrol atau komponen- komponen untuk membuat sebuah aplikasi Windows Form dan kontrol- kontrol dari aplikasi inilah yang biasanya disebut dengan GUI (*Graphical User Interface*). Jadi user akan berinteraksi dengan sebuah program aplikasi melalui GUI. Pada IDE Visual Studio 2010 terdapat menu bar, toolbar, toolbox, solution explorer, dan properties windows.

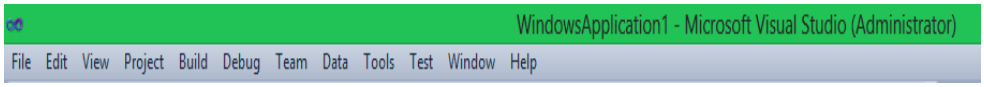


Gambar II.3. IDE Visual Studio 2010

Sumber : Priyanto Hidayatullah (2014 : 25)

II.4.2.2. Menu Bar

Menu bar adalah bagian dari IDE yang terdiri atas perintah- perintah untuk mengatur IDE, mengedit kode, dan mengeksekusi program. Di dalam menu bar, perintah- perintah dikelompokkan ke dalam beberapa bagian sesuai jenis perintah tersebut. Menu bar pada Visual Studio 2010 dilihat pada Gambar II.4.



Gambar II.4. Menu Bar

Sumber : Priyanto Hidayatullah (2014 : 25)

Untuk menggunakan menu atau pilihan pada menu bar, kita tinggal mengeklik pada menu atau pilihan yang akan dijalannkan. Sebagai contoh, untuk membuat sebuah proyek yang baru, pilih menu **File** > **New** > **Project**.

Pada IDE Visual Studio 2010, terdapat 12 menu utama. Menu – menu tersebut antara lain sebagai berikut :

1. Menu **File** berisi perintah- perintah untuk membuat proyek, membuka proyek, menutup proyek, mencetak data dari proyek, dan lain- lain.
2. Menu **Edit** berisi perintah- perintah untuk undo, cut, paste, dan lain- lain.
3. Menu **View** berisi perintah- perintah untuk menampilkan windows-windows dari IDE dan toolbar.
4. Menu **Project** berisi perintah- perintah untuk mengatur proyek dan file-file.
5. Menu **Build** berisi perintah- perintah untuk meng-compile program.

6. Menu **Debug** berisi perintah- perintah untuk men-debug dan menjalankan program.
7. Menu **Team** berisi tentang connect to team fundation server.
8. Menu **Data** berisi perintah- perintah untuk berhubungan dengan basis data.
9. Menu **Tools** berisi perintah- perintah untuk mengakses komponen IDE tambahan dan mengubah IDE.
10. Menu **Test** mengelolah pengujian yang akan dilakukan pada proyek.
11. Menu **Window** berisi perintah- perintah untuk mengatur dan menampilkan Windows.
12. Menu **Help** berisi perintah- perintah untuk mengakses fasilitas bantuan.

II.4.2.3. Toolbar

Toolbar fungsinya sama seperti menu. Bedanya pada toolbar pilihan-pilihan berbentuk icon. Untuk memilih suatu proses yang akan dilakukan, kita tinggal menekan icon yang sesuai dengan proses yang kita inginkan. Bagian toolbar terlihat seperti pada Gambar II.5.

Gambar II.5. Toolbar

Sumber : Priyanto Hidayatullah (2014 : 27)

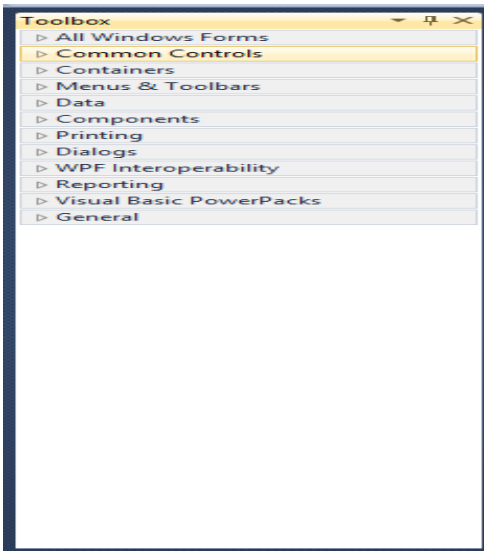
II.4.2.4. Toolbox

Toolbox adalah tempat di mana kontrol- kontrol dan komponen-komponen diletakkan. Kontrol dan komponen disimpan pada Toolbox dengan berbagai kategori :

1. Common Controls, berisi kontrol- kontrol umum yang sering digunakan seperti button, label, textbox, dsb.
2. Containers, berisi kontrol penyimpan kontrol lainnya seperti panel, group box, tabcontrol, dsb.
3. Menu & Toolbar, berisi kontrol dan komponen menu, context menu dan toolbar.
4. Data, berisi kontrol dan komponen pengolahan data.
5. Components, berisi komponen- komponen seperti timer, imagelist, dsb.
6. Printing, berisi komponen untuk pencetakan dokumen.
7. Dialog, berisi komponen untuk berinteraksi dengan pengguna dalam hal membuka file, menyimpan file, membuka folder dll.
8. WPF interoperability, berisi komponen untuk Windows Presentation Foundation.
9. Reporting, berisi kontrol untuk membuat laporan pada Visual Studio 2010. Untuk studi kasus pada buku ini, kita akan menggunakan fasilitas *reporting* dari Crystal Report yang jauh lebih lengkap dan powerful namun gratis.
10. Visual Basic PowerPacks, berisi beberapa kontrol tambahan untuk menggambar (contoh : oval, garis, persegi) dan fungsi tambahan lainnya.

11. General, jika ada kontrol atau komponen yang mau ditambahkan dan belum memiliki kategori yang jelas, maka bisa dimasukkan ke kategori ini.

Selain kategori di atas, ada All Windows Form yang berisi semua kontrol dan komponen yang ada (lihat Gambar II.6.).

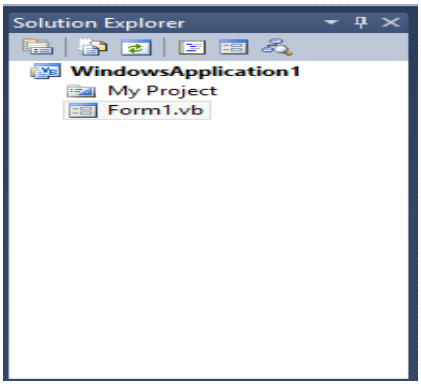


Gambar II.6. Toolbox

Sumber : Priyanto Hidayatullah (2014 : 28)

II.4.2.5. Solution Explorer

Solution explorer memberikan tampilan daftar file- file dari proyek yang sedang dibuat. Pada jendela solution explorer terdapat beberapa tombol dan tree yang berisi daftar dari file- file yang digunakan dalam proyek (lihat Gambar II.7.).

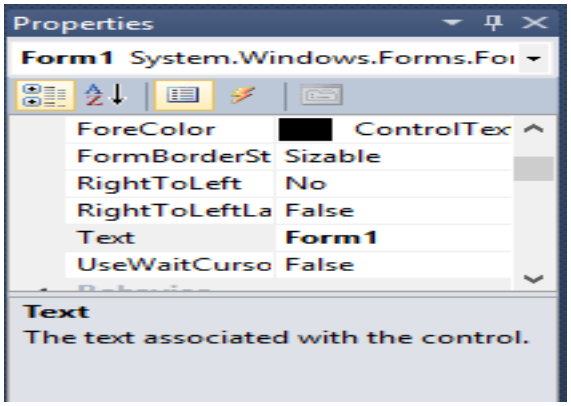


Gambar II.7. Solution Explorer

Sumber : Priyanto Hidayatullah (2014 : 29)

II.4.2.6. Properties Window

Properties Window adalah tempat menyyimpan property dari setiap objek kontrol dan komponen. Properties Window juga dipakai untuk mengatur property dari objek kontrol dan komponen yang dipakai. Dengan propertie window, kita dapat mengubah property yang nantinya akan dipakai sebagai default dari objek kontrol dan komponen pada waktu pertama kali program dieksekusi.



Gambar II.8. Properties Window

Sumber : Priyanto Hidayatullah (2014 : 30)

II.5. SQL Server

Database Management System (DBMS) adalah aplikasi untuk mengelolah basis data. DBMS biasanya menawarkan beberapa kemampuan yang terintegrasi seperti:

1. Membuat, menghapus, menambah, dan memodifikasi basis data.
2. Pada beberapa DBMS pengelolaannya berbasis windows (berbentuk jendela- jendela) sehingga lebih mudah digunakan.
3. Tidak semua orang bisa mengakses basis data yang ada sehingga memberikan keamanan bagi data.
4. Kemampuan berkomunikasi dengan program aplikasi yang lain. Misalnya dimungkinkan untuk mengakses basis data SQL Server menggunakan aplikasi yang dibuat menggunakan VB.NET.
5. Kemampuan pengaksesan melalui komunikasi antarkomputer (client server).

Microsoft SQL Server adalah salah satu aplikasi DBMS yang sudah sangat banyak digunakan oleh para pemrograman aplikasi basis data. Contoh DBMS lainnya adalah : MySQL (freeware), PostgreSQL (freeware), MS Access dari Microsoft, DB2 dari IBM, Oracle dan Oracle Corp, Dbase, FoxPro, dsb.

II.5.1. Tool pada MS SQL Server 2008 R2

II.5.1.1. Microsoft SQL Server Management Studio

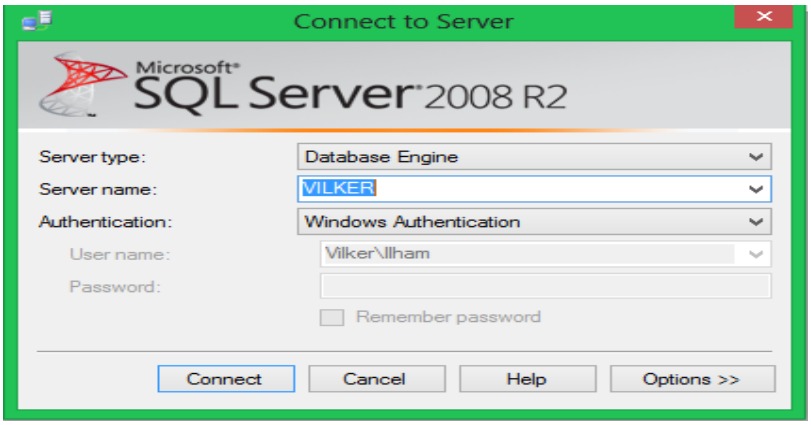
Tool ini digunakan untuk :

1. Membuat Database

Misalnya kita ingin membuat basis data dengan nama Inventori.

Langkah kerjanya adalah :

a. Bukalah Microsoft SQL Server Management Studio. Isikan **Database Engine** pada tipe *server*, nama *database server* Anda dan *windows Authentification* untuk tipe otentifikasi agar lebih mudah, tekan **Connect.**



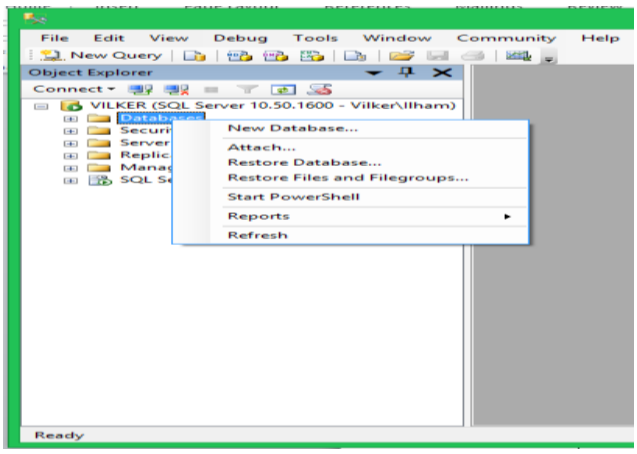
Gambar II.9. Login ke Microsoft SQL Server Management Studio

Sumber : Priyanto Hidayatullah (2014 : 187)

b. Pastikan Object Explorer terbuka, jika belum, buka dengan memilih menu **View** > **Object Explorer**.

c. Pada Object Explorer, klik kanan pada **Database**.

d. Pilih **New Database…**



Gambar II.10. Membuat database baru

Sumber : Priyanto Hidayatullah (2014 : 188)

e. Tulis nama basis data pada baris *Name*, misalnya **inventori**,

   Catatan : Dalam satu SQL Server, nama basis data harus berbeda. Gunakan nama yang menggambarkan isi datanya. Jangan gunakan nama basis data yang 'tidak nyambung' dengan isi datanya. Misalnya nama basis datanya swalayan tapi basis data tersebut berisi data rumah sakit.

f. **Initial Size** adalah ukuran awal basis data yang kita buat. Ubahlah ukurannya dari 3 MB menjadi 5 MB.

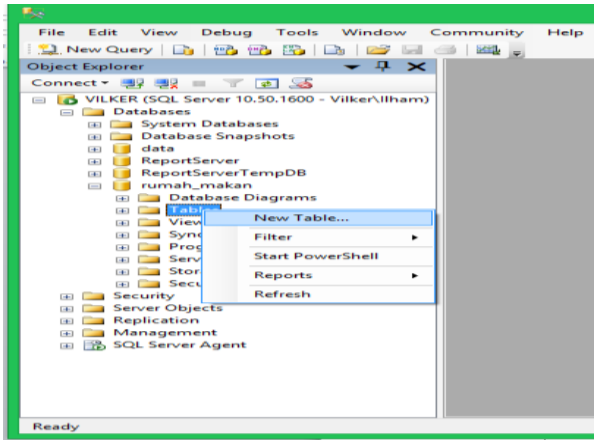
g. Tekan **OK**.

2. Merancang Tabel

Pada dasarnya, data- data di dalam basis data disimpan di dalam tabel. Untuk merancang tabel, langkah kerjanya adalah :

a. Pilih basis data tempat kita membuat tabelnya.

b. Klik kanan pada bagian **Tables** kemudian pilih **New Table**…



Gambar II.11. Membuat tabel baru

Sumber : Priyanto Hidayatullah (2014 : 191)

c. Masukkan informasi- informasi tentang tabel yang ingin kita buat.

   Column Name : nama kolom dalam tabel kita.

   Data type : tipe data kolom yang bersangkutan.

   Length : panjang data kolom yang bersangkutan.

   Allow Nulls : Ceklis jika kolom tersebut diperbolehkan tidak diisi.

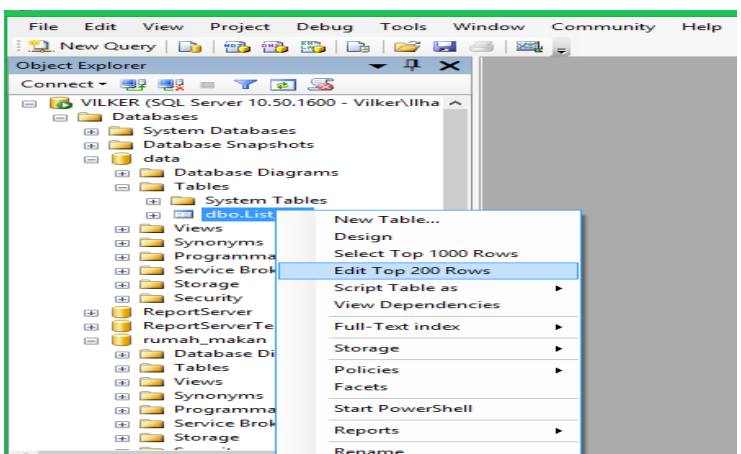
d. Untuk membuat setelan *primary key* pada table, dapat dilakukan dengan jalan klik kanan pada nama kolom yang akan dijadikan *primary key* kemudian pilih **Set Primary Key**.

e. Setelah selesai, tutup jendelanya kemudian isi nama tabel yang baru saja kita buat. Tekan **OK**.

3. Mengisi Tabel

Langkah- langkah untuk mengisi tabel adalah :

1. Buka basis data yang mengandung tabel yang kita cari.
2. Klik kanan tabel tersebut.
3. Kemudian pilihlah **Edit Top 200 Rows**.
4. Kemudian Isilah data- data yang kita inginkan.



Gambar II.12. Mengisikan data ke dalam tabel

Sumber : Priyanto Hidayatullah (2014 : 194)

II.6. Database

Istilah database banyak memiliki definisi. Untuk sebagian kalangan sederhana *database* diartikan sebagai kumpulan data (buku, nomor telepon, daftar pegawai, dan lain sebagainya). Ada juga yang menyebut *database* dengan definisi lain yang lebih formal dan tegas. *Database* didefenisikan sebagai kumpulan data yang terintegrasi dan diatur sedemikian rupa sehingga data tersebut dapat dimanipulasi, diambil dan dicari secara cepat.

Selain berisi data, *database* juga berisi *metadata*. Metadata adalah data yang menjelaskan tentang struktur dari data itu sendiri. Sebagai contoh, Anda dapat memperoleh informasi tentang nama-nama kolom dan tipe yang ditampilkan tersebut disebut *metadata* **(**Budi Raharjo ; 2011 : 3).

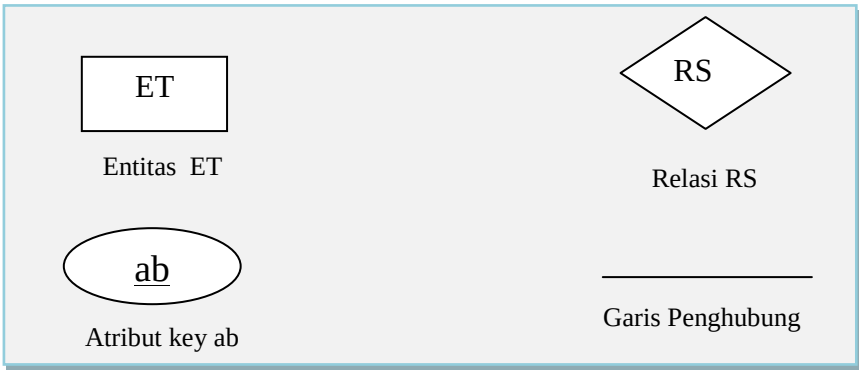
II.6.1. Pemodelan Data

Terdapat beberapa penjelasanmengenai pemodelan basis data. Suatu basis data dapatdigunakan secara bebas untuk menggambarkan dan memberikan deskripsi mengenai kumpulan informasi uang tersimpan dalam *data storage* komputer. Secara sederhana, defenisi untuk model basis data adalah sekumpulan

notasi atau simbol untuk menggambarkan data dan relasinya, berdasarkan suatu konsep dan aturan tertentu suatu pemodelan **(**Yudi Priyadi ; 2014 : 10).

II.6.2. Notasi Diagram E-R

Pemodelan basis data dengan menggunakan diagram relasi antar entitas, dapat dilakukan dengan menggunakan suatu pemodelan basis datayang bernama Diagram *Entity-Relational* (selanjutnya disingkat Diagram E-R). Pada Gambar II.13, terdapat suatu simbol/notasi dasar yang digunakan pada Diagram E-R,yaitu entitas,relasi, atribut, dan garis penghubung **(**Yudi Priyadi ; 2014 : 20).



Gambar II.13 : Notasi Dasar Diagram E-R

(Sumber : Yudi Priyadi ; 2014 : 20)

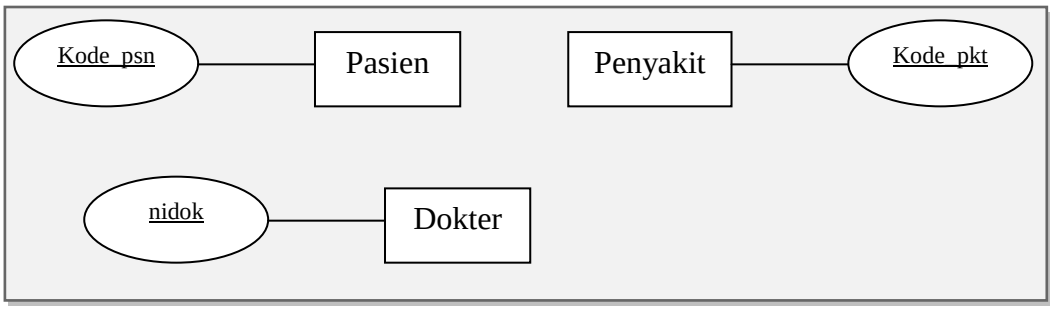
1. Entitas

   Merupakan notasi untuk mewakili suatu objek dengan karakteristik sama, yang dilengkapi oleh atribut, sehingga pada suatu lingkungan nyata setiap

objek akan berbeda dengan objek lainya. Pada umumnya, objek dapat berupa benda, pekerjaan, tempat dan orang.

2. Atribut

Merupakan notasi yang menjelaskan karakteristik suatu entitas dan juga relasinya. Atribut dapat sebagai key yang bersifat unik, yaitu *Primary Key* atau *Foreign Key*. Selain itu, atribut juga dapat sebagai atributdeskriptif saja, yaitu sebagai pelengkap deskripsi suatu entitas dan relasi.

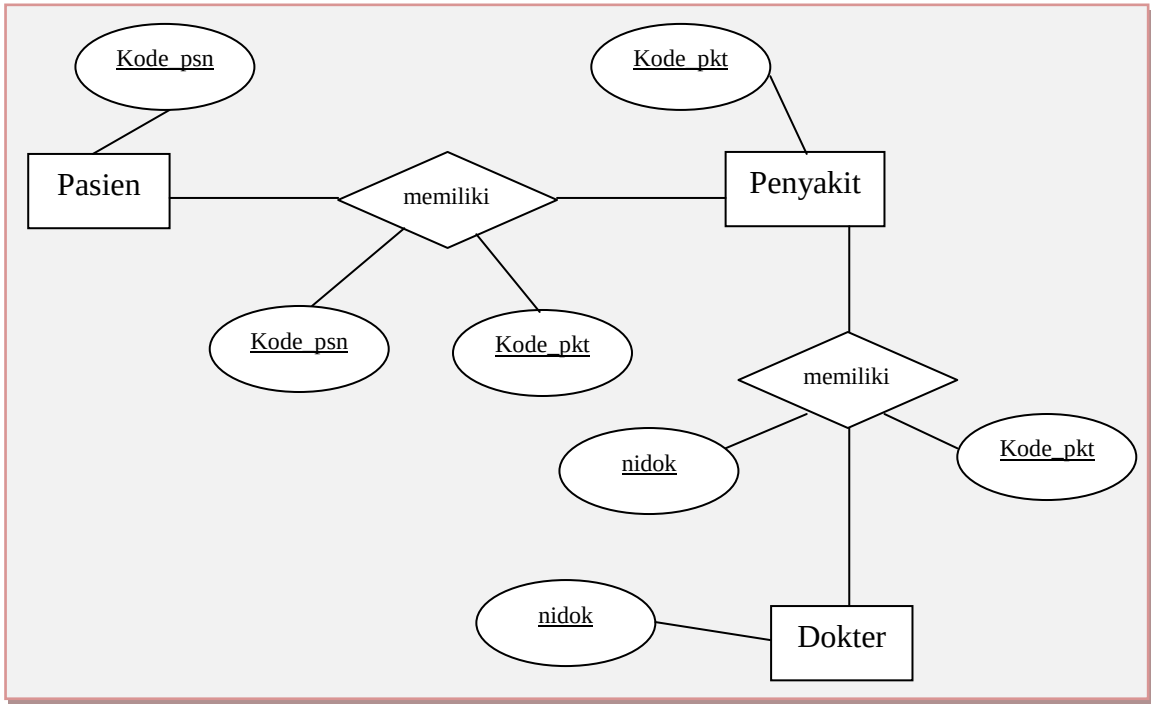


Gambar II.14 : Atribut Key pada Entitas

(Sumber : Yudi Priyadi ; 2014 : 23)

3. Relasi

Merupakan notasi yang digunakan untuk menghubungkan beberapa entitas berdasarkan fakta pada suatu lingkungan.



Gambar II.15 : Pemilihan Relasi untuk Entitas

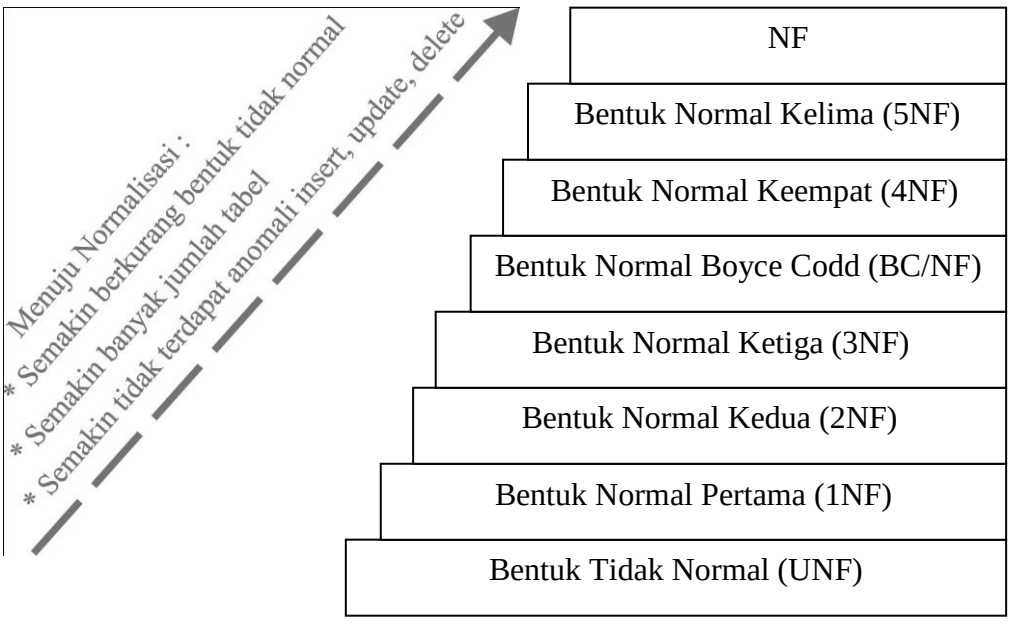
(Sumber : Yudi Priyadi ; 2014 : 25)

4. Garis penghubung

   Merupakan notasi untuk merangkaikan keterkaitan antar notasi yang digunakan dalam Diagram E-R, yaitu entitas, relasi dan atribut.

II.6.3. Normalisasi

Normalisasi merupakan proses sistematis yang dilakukan pada struktur tabel basis data menjadi struktur tabel yang memiliki integritas data, sehingga tidak memiliki data anomali pada saat melakukan *insert, delete,* dan *update.* Pada Gambar II.16, tahapan proses sistematis yang dilakukan mulai dari bentuk tidak normal menjadi bentuk normal memiliki suatu syarat yang harus dipenuhi pada saat menuju suatu bentuk yang lebih baik (*well structured relation*).



Gambar II.16 : Tahapan Proses Bentuk Normalisasi

(Sumber : Yudi Priyadi ; 2014 : 67)

Setiap syarat dalam tahapan suatu bentuk normal memiliki keterkaitan, hal ini disebabkan karena pada setiap bentuk normal mengalami penyempurnaan untuk bentuk normal selanjutnya. Bentuk tidak normal akan semakin berkurang, setelah melalui tahapan berubahan bentuk normalisasi, sehingga berdampak pada jumlah tabel yang semakin banyak, tetapi menuju perbaikan ke dalam bentuk *well structured relation.* Hal ini terjadi akibat dari pengelompokan data suatu tabel agar memiliki ketergantungan secara fungsional **(**Yudi Priyadi ; 2014 : 67).

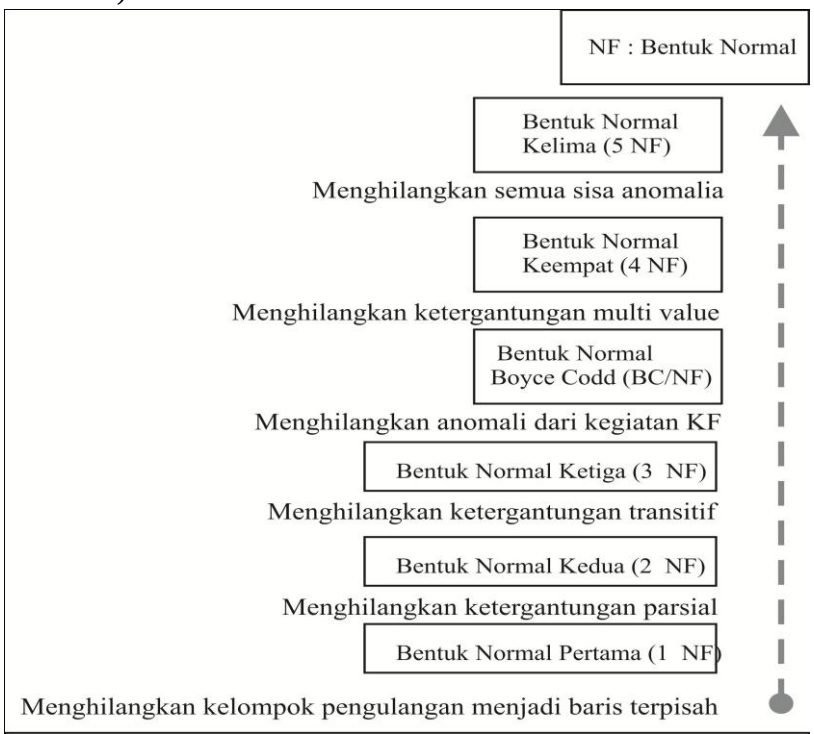
II.6.4. Aturan Proses Normalisasi

Secara sederhana, kegiatan normalisasi adalah melakukan dekomposisi atau penguraian tabel beserta datanya, menjadi tabel yang normal menurut konsep RDBMS. Merujuk pada gambar II.17, dekomposisi diawali dengan melakukan analisis pada suatu tabel atau beberapa contoh formulir yang sudah memiliki data lengkap dalam basis data, tetapi masih dalam bentuk yang tidak normal (UNF). Oleh karena itu agar dapat memenuhi syarat bentuk normal pertama (1NF), pada setiap barisnya diisikan suatu *value* dengan kelompok data yang sama, berdasarkan suatu atribut key. Dengan demikian, kelompok pengulangan dalam

suatu baris dapat dihilangkan, karena sudah tidak terdapat *value* yang kosong untuk setiap *field* dan *record*nya

Setelah memenuhi syarat bentuk normal pertama (1NF), proses berikutnyaadalah menghilangkan ketergantungan secara parsial, yaitu dengan cara melakukan dekomposisi tabel menjadi beberapa kelompok tabel berdasarkan field yang memiliki status sebagai key. Hal ini dapat dilakukan oleh salah satu field saja, dengan tetap tidak mengubah arti relasi dan ketergantungannya. Oleh sebab itu, disebut ketergantungan fungsional sebagian (*partiallly functional*), sehingga syarat bentuk normal kedua (2NF) sudah tercapai.

Bentuk normal kedua (2NF) merupakan syarat yang harus dimiliki untuk menuju bentuk normal ketiga (3NF). Pada proses ini, dilakukan dengan menghilangkan ketergantungan secara transitif, yaitu suatu konsep untuk tabel dari hasil relasi yang didalamnya terdapat ketergantungan secara tidak langsung pada beberapa atributnya. Pada umumnya proses normalisasi sudah dapat tercapai pada bentuk normal ketiga (3NF), yaitu dengan menghasilkan tabel yang tidak mengalami anomali basis data pada saat proses *insert, delete,* dan *update* **(**Yudi Priyadi ; 2014 : 68).



Gambar II.17 : Tahapan Aturan Proses Normalisasi

(Sumber : Yudi Priyadi ; 2014 : 69)

II.7. *Unified Modeling Language (UML)*

Pada perkembangan teknik pemrograman berorientasi objek, muncullah sebuah standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakanteknik pemrograman berorientasi objek, yaitu *Unified Modeling Language* (UML). UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun dan dokumentasi dari sistem perangkat lunak. UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung (Rosa A.S & M. Shalahuddin ; 2011 : 118).

UML hanya berfungsi untuk melakukan pemodelan, jadi penggunaan UML tidak terbatas pada metodologi tertentu, meskipun pada kenyataannya UML paling banyak digunakan pada metode berorientasi objek.

UML diaplikasikan untuk maksud tertentu, biasanya antara lain :

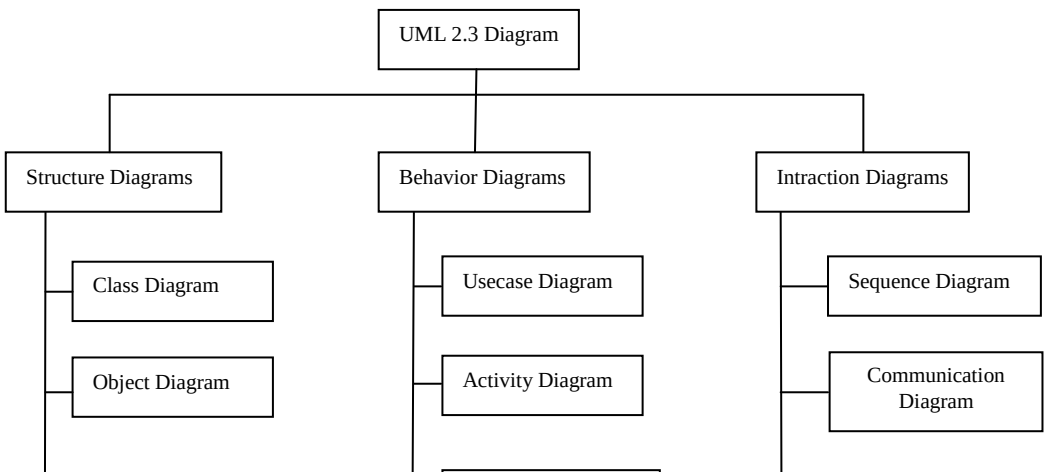
1. Merancang perangkat Lunak.
2. Sarana Komunikasi antara perangkat lunak dengan proses bisnis.

3. Menjabarkan sistem secara rinci untuk analisa dan mencari apa yang diperlukan sistem.
4. Mendokumentasikan sistem yang ada, proses-proses dan organisasinya.

Blok pembangunan utama UML adalah diagram. Beberapa diagram ada yang rinci (jenis *timming diagram*) dan lainya ada yang bersifat umum (misalnya diagram kelas). Para pengembang sistem berorientasi objek menggunakan bahasa model untuk menggambarkan, membangun dan mendokumentasikan sistem yang mereka rancang. UML memungkinkan para anggota team untuk bekerja sama dengan bahasa model yang sama dengan mengaplikasikan beragam sistem. Intinya UML merupakan alat komunikasi yang konsisten dalam mendukung para pengembang sistem saat ini (Prabowo Pudjo Widodo & Herlawati ; 2011 : 6).

II.7.1 Diagram-Diagram UML

Menurut (Rosa A.S & M. Shalahuddin ; 2011 : 120) Pada UML 2.3 terdiri dari 13 macam diagram yang dikelompokan dalam 3 kategori. Pembagian kategori dan macam-macam diagram tersebut dapat dilihat pada gambar II.18 di bawah ini



Gambar II.18 : Diagram UML

(Sumber : Rosa A.S & M. Shalahuddin ; 2011 : 121)

Berikut ini penjelasan singkat dari pembagian kategori tersebut

1. *StructureDiagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
2. *Behavior Diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.

3. *Interaction Diagrams* yaitu kumpulan diagram yang digunakan untukmenggambarkan interaksi sistem dengan sistem lain maupun interaksi antar subsistem pada suatu sistem.

1. ***Class Diagram***

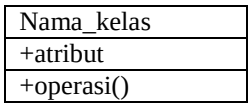
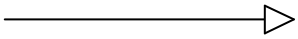
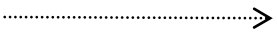
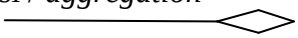
Diagram kelas atau *Class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem.

Kelas memiliki apa yang disebut atribut dan metode atau operasi.

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

Berikut Tabel II.1 menerangkan simbol-simbol pada diagram kelas :

Tabel II.1. Diagram Kelas

Simbol	Deskripsi
Kelas 	Kelas pada struktur sistem
Antarmuka / *interface*  Nama_*interface*	Sama dengan konsep *interface* dalam pemrograman berorientasi objek
Asosiasi / *association* Asosiasi berarah/ *directed association* 	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity* Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*
Generalisasi 	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum khusus)
Kebergantungan 	Relasi antar kelas dengan makna kebergantungan antar kelas
Agregasi / *aggregation* 	Semua bagian (*whole part*)

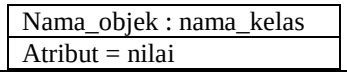
(Sumber : Rosa A.S & M. Shalahuddin ; 2011 : 124)

2. ***Object Diagram***

Diagram objek menggambarkan struktur sistem dari segi penamaan objek dan jalannya objek dalam sistem. Pada diagram objek harus dipastikan semua kelas yang sudah didefenisikan pada diagram kelas harus dipakai objeknya, karena jika tidak, pendefenisian kelas itu tidak dapat dipertanggungjawabkan. Untuk apa mendefenisikan sebuah kelas sedangkan pada jalannya sistem, objeknya tidak pernah dipakai.

Berikut adalah Tabel II.2 menerangkan simbol-simbol diagram objek

Tabel II.2. Diagram Paket

Simbol	Deskripsi
Objek 	Objek dari kelas yang berjalansaat sistem dijalankan
Link	Relasi antar objek

(Sumber : Rosa A.S & M. Shalahuddin ; 2011 : 124)

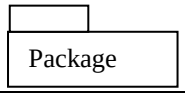
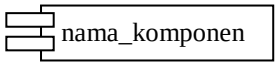
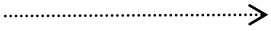
3. ***Component Diagram***

Diagram komponen atau component diagram dibuatuntuk menunjukkan organisasi dan ketergantungan di antara kumpulan komponen dalam sebuah sistem. Diagram komponen fokus pada komponen sistem yang dibutuhkan dan ada didalam sistem. Komponen dasar yang biasanya ada dalam suatu sistem adalah sebagai berikut :

1) Komponen *user interface* yang menangani tampilan
2) Komponen *bussiness procesiing* yang menangani fungsi-fungsi proses bisnis
3) Komponen data yang menangani manipulasi data
4) Komponen *security* yang menangani keamanan sistem

Komponen lebih terfokus pada penggolongan secara umum fungsi-fungsi yang diperlukan, berikut Tabel II.3 yang menerangkan simbol-simbol yang ada pada diagram komponen

Tabel II.3. Diagram Komponen

Simbol	Deskripsi
Package 	*Package* merupakan sebuah bungkusan dari satu atau lebih komponen
Komponen 	Komponen Sistem
Kebergantungan / *dependency* 	Kebergantungan antar komponen, arah panah mengarah pada komponen yang dipakai
Antar muka / *interface*  nama_interface	Sama dengan konsep *interface*pada pemrograman berorientasi objek, yaitu sebagai antarmuka komponen agar tidak mengakses langsung komponen
Link	Relasi antar komponen

(Sumber : Rosa A.S & M. Shalahuddin ; 2011 : 126)

4. ***Use Case Diagram***

Use case atau diagram *use case* merupakan pemodelan untuk kelakuan (*behaviour)*sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar,*use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu. Syarat penamaan pada *use case* adalah nama didefenisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefenisian apa yang disebut aktor dan *use case.*

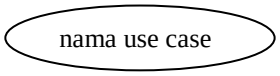
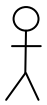
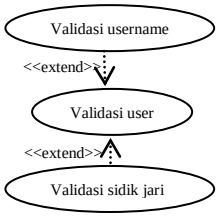
1) Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang

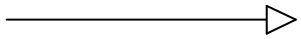
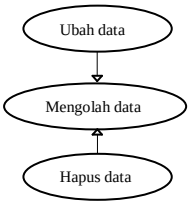
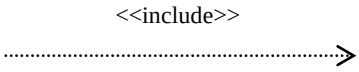
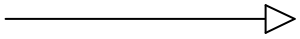
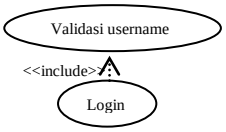
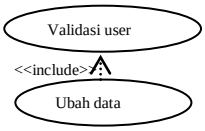
akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.

2) *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

Berikut Tabel II.4 menerangkan simbol-simbol pada diagram *use case*

Tabel II.4. Diagram Use case

Simbol	Deskripsi
Use case 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama *use case*
Aktor / *actor*  nama aktor	Orang, proses, atau sistem yang lain berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan di buat itu sendiri
Asosiasi / *association*	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case*, atau *usecase* memiliki interasi dengan aktor
Ekstensi / *extend* <<extend>>	Relasi usecase tambahan ke subuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanppa *use case* tambahan itu, mirip dengan prinsip inheritance pada pemrograman berorientasi objek, biasanya *use case* tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan misal  arah panah mengarah pada *use case* yang ditambahkan
Generalisasi / *generalization*	Hubungan generalisasi dan spesialisasi (umum – khusus) antara dua buah use

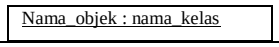
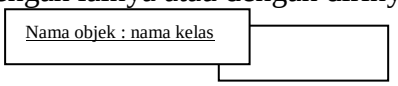
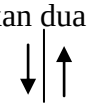
	case dimana fungsi yang satu adalah fungsi yang lebih umum dari lainya misalnya :  Arah panah mengarah pada use case yang menjadi generalisasinya (umum)
Menggunakan / *include* / *uses*  	Relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan memerlukan use case ini untuk menjalankan fungsinya atau sebagai syarat dijalankan use case ini Ada 2 sudut pandang yang cukup besar mengenai include di usecase 1. include berarti use case yang ditambahkan akan selalu dipanggil saat use case dijalankan misal pada kasus berikut :  2. include berarti use case yang tambahan akan selalu melakukan pengecekan apakah use case yang di tambahkan telah di jalankan sebelum use case tambahan di jalankan, misal pada kasus berikut :  Kedua interpretasi di atas dapat dianut salah satu atau keduanya tergantung pada pertimbangan dan interpretasi yang dibutuhkan.

(Sumber : Rosa A.S & M. Shalahuddin ; 2011 : 131)

5. **Communication Diagram**

Diagram komunikasi mengelompokkan message pada kumpulan diagram sekuen menjadi sebuah diagram. Dalam diagram komunikasi yang dituliskan adalah operasi / metode yang di jalankan antara objek yang satu dengan objek lainnya secara keseluruhan, oleh karna itu dapat di ambil dari jalanya interaksi pada semua diagram sekuen. Berikut adalah Tabel II.5 yang menerangkan simbol-simbol yang ada pada diagram komunikasi :

Tabel II.5 Diagram Komunikasi

Simbol	Deskripsi
Objek 	Objek yang melakukan interaksi pesan
Link	Relasi antar objek yang menghubungkan objek satu dengan lainya atau dengan dirinya sendiri 
Arah pesan / stimulus	Arah pesan yang terjadi, jika pada suatu link ada dua arah pesan yang berbeda, maka arah juga deigambarkan dua arah pada dua sisi link 

(Sumber : Rosa A.S & M. Shalahuddin ; 2011 : 140)

6. ***Activity Diagram***

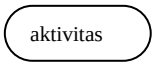
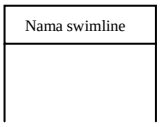
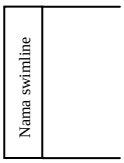
Diagram aktivitas atau activity diagram menggambarkan workflow (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Diagram aktivitas juga banyak digunakan untuk mendefenisikan hal-hal berikut :

1) Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sitemyang didefenisikan

2) Urutan atau pengelompokan tampilan dari sistem/user interface dimana setiap aktivitasdianggap memiliki sebuah rancangan antarmuka tampilan
3) Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefenisikan kasus ujinya.

Berikut adalah Tabel II.6 yang menggambarkan simbol-simbol yang ada pada diagram aktivitas :

Tabel II.6 Diagram Aktivitas

Simbol	Deskripsi
Status awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki status awal
Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja
Percabangan / decesion 	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu
Penggabungan / join	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu
Status akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir
Swimlane  atau 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi

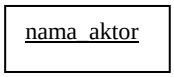
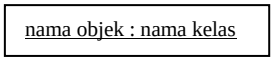
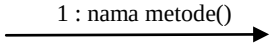
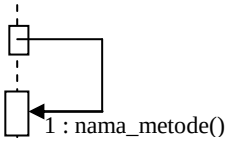
(Sumber : Rosa A.S & M. Shalahuddin ; 2011 : 134)

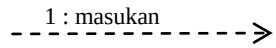
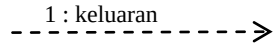
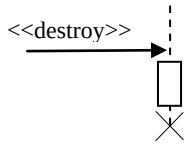
7. Sequence Diagram

Diagram sekuen menggambarkan kelakuan objek pada use case dengan mendeskripsikan waktu hidup objek dan messasge yang dikirimkan dan diterima antar objek. Banyaknya diagram objek yang digambarkan adalah

sebanyak pendefinisian use case yang memiliki proses sendiri atau yang penting semua use case yang telah didefenisikan interaksi jalanya pesan sudah dicakup dapa diagram sekuen sehingga semakin banyak use case yang didefenisikan maka diagram sekuen yang harus dibuat juga semakin banyak. Berikut adalah Tabel II.7 yang menerangkan simbol-sombol yang ada pada diagram sekuen :

Tabel II.7 Diagram Squence

Simbol	Deskripsi
Aktor  atau  tampa waktu aktif	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang, biasanya di nyatakan menggunakan kata benda di awali frase nama aktor
Garis hidup / lifeline	Menyatakan kehidupan suatu objek
Objek 	Menyatakan objek yang berintaraksi pesan
Waktu aktif 	Menyatakan objek dalam keadaan aktif dan berinteraksi pesan
Pesan tipe create 	Objek yang lain, arah panah mengarah pada objek yang dibuat
Pesan tope call 	Menyatakan suatu objek memanggil operasi / metode yang ada pada objek lain atau dirinya sendiri  Arah panah mengarah pada objek yang memiliki operasi / metode, karena ini memanggil operasi / metode maka operasi / metode yang di panggil harus ada pada diagram kelas sesuai dengan kelas objek yang berinteraksi

Pesan tipe send 	Menyatakan bahwa suatu objek mengirimkan data / masukan / informasi ke objek lainya, arah panah mengarah pada objek yang dikirimi
Pesan tipe return 	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasiatau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian
Pesan tipe destroy 	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada create maka ada destroy

(Sumber : Rosa A.S & M. Shalahuddin ; 2011 : 138)