BAB II

LANDASAN TEORI

II.1. Teori Sistem

Pengertian Sistem menurut Mulyadi dalam Menurut Rizan Machmud (2013:441), adalah sebagai berikut : “Sekelompok dua atau lebih komponen-komponen yang saling berkaitan (subsistem-subsistem yang bersatu untuk mencapai tujuan yang sama)”.

Menurut Kusrini (2010:5), Kata Sistem mempunyai beberapa pengertian, tergantung dari sudut mana kata tersebut didefenisikan. Secara garis besar ada dua pendekatan yang dilakukan yaitu :

1. Pendekatan sistem yang lebih menekankan pada elemen-elemen atau kelompoknya, yang didalam hal ini sistem ini didefenisikan sebagai suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu aturan tertentu.
2. Pendekatan sistem sebagai jaringan kerja dari prosedur, yang lebih menekankan urutan operasi didalam sistem. Prosedur didefenisikan sebagai urutan operasi kerja (tulis-menulis), yang biasanya melibatkan beberapa orang didalam satu atau lebih departemen yang diterapkan untuk menjamin penanganan yang seragam dari transaksi bisnis yang terjadi.

Pendekatan sistem yang lebih menekankan pada elemen-elemen atau komponennya mendefinisikan sistem sebagai sekumpulan elemen-elemen yang saling terkait atau terpadu yang dimaksudkan untuk mencapai suatu tujuan.

Dengan demikian didalam suatu sistem, komponen-komponen ini tidak dapat berdiri sendiri, tetapi sebaliknya, saling berhubungan hingga berbentuk suatu kesatuan hingga tujuan sistem dapat tercapai.

II.1.1. Karakteristik sistem

Menurut Kusrini (2010:6), Sistem mempunyai beberapa karakteristik atau sifat-sifat tertentu antara lain :

1. Komponen sistem (*Components*)

   Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, artinya saling bekerja sama untuk membentuk suatu kesatuan. Komponen-komponen sistem atau elemen-elemen sistem dapat berupaya suatu subsistem atau bagian dari sistem. Setiap sistem tidak peduli berapapun kecilnya, selalu mengandung komponen-komponen atau subsistem.

2. Batasan sistem (*Boundary*)

   Batas sistem merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lainnya atau dengan lingkungan luarnya.

3. Lingkungan luar sistem (*Environtment*)

   Lingkungan luar dari suatu sistem adalah apapun diluar batas dari sistem yang mempengaruhi operasi sistem. Lingkungan luar sistem dapat bersifat menguntungkan dan dapat juga bersifat merugikan sistem tersebut

4. Penghubung sistem (*Interface*)

   Merupakan media penghubung antara satu subsistem dengan subsistem yang lainnya. Melalui perhubungan ini memungkinkan sumber-sumber daya mengalir dari subsistem yang lainnya.

5. Masukan sistem (*Input*)

   Merupakan energi yang dimasukkan ke dalam sistem. Masukan dapat berupa masukan perawatan (*maintenance input*) dan masukan sinyal (*signal input*).

6. Keluaran sistem (*Output*)

   Keluaran adalah hasil dari energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna dan sisa pembuangan.

7. Pengolah sistem (*Process*)

   Suatu sistem yang dapat mempunyai suatu bagian pengolah yang akan merubah masukan menjadi keluaran.

8. Sasaran sistem (*Objective*)

   Suatu sistem pasti mempunyai tujuan atau sasaran. Kalau suatu sistem tidak mempunyai sasaran, maka operasi sistem tidak ada gunanya. Suatu sistem dikatakan berhasil bila mengenai sasaran atau tujuannya.

II.1.2. Klasifikasi sistem

Menurut Kusrini (2010:7), Sistem dapat diklasifikasikan dari beberapa sudut pandangan, diantaranya sebagai berikut :

1. Sistem abstrak dan sistem fisik.

   Sistem abstrak adalah sistem yang berisi gagasan atau konsep. Misalnya, sistem teologi yang berisi gagasan tentang hubungan manusia dan Tuhan. Sistem fisik merupakan sistem yang secara fisik dapat dilihat. Misalnya sistem komputer, sistem sekolah, sistem akuntansi, dan sistem transportasi.

2. Sistem alamiah dan sistem buatan manusia.

   Sistem alamiah adalah sistem yang terjadi karena alam (tidak dibuat manusia). Misalnya, sistem tata surya. Sistem buatan manusia adalah sistem yang dibuat oleh manusia. Misalnya, sistem komputer dan sistem mobil.

3. Sistem tertentu dan sistem tak tentu.

   Sistem tertentu adalah sistem yang operasinya dapat diprediksi secara tepat. Misalnya, sistem komputer. Sistem tak tentu adalah sistem yang tak dapat diramal dengan pasti karena mengandung unsur probabilitas. Misalnya, sistem arisan dan sistem sediaan.

4. Sistem tertutup dan sistem terbuka

   Sistem tertutup adalah sistem yang tidak bertukar materi, informasi, atau energi dengan lingkungan. Misalnya, reaksi kimia dalam tabung terisolasi. Sistem terbuka adalah sistem yang berhubungan dengan lingkungan dan dipengaruhi oleh lingkungan.

II.2. Informasi

Menurut Kusrini (2010:7), Informasi adalah data yang diolah menjadi sebuah bentuk yang berarti bagi pengguna, yang bermanfaat dalam pengambilan keputusan saat ini atau mendukung sumber informasi. Data belum memiliki nilai sedangkan informasi sudah memiliki nilai. Informasi dikatakan bernilai bila manfaatnya lebih besar bila dibandingkan biaya untuk mendapatkannya.

II.2.1. Kualitas Informasi

Menurut Kusrini (2010:8), Informasi yang berkualitas memiliki 3 kriteria 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 datang pada penerima tidak boleh terlambat. Didalam pengambilan keputusan, informasi yang sudah usang tidak lagi bernilai. Bila informasi datang terlambat sehingga pengambilan keputusan terlambat dilakukan, hal ini dapat berakibat batal pada perusahaan.

3. Relevan (*Relevance)*

   Informasi yang disampaikan harus mempunyai keterkaitan dengan masalah yang akan dibahas dengan informasi tersebut. Informasi harus bermanfaat bagi pemakainya. Disamping karakteristik, nilai informasi juga ikut menentukan kualitasnya. Nilai informasi (*Value of Informatioan)* ditentukan oleh 2 hal, yaitu manfaat dan biaya untuk mendapatkannya. Suatu informasi dikatakan bernilai bila manfaat lebih besar disamping biaya untuk mendapatkannya.

II.3. Sistem Informasi

Menurut Kusrini (2010:8-9), Sistem informasi adalah suatu sistem dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi bersifat menajerial dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dan laporan-laporan yang diperlukan.

Defenisi umum sistem informasi adalah suatu system yang terdiri atas rangkaian sub system informasi terhadap pengolahan data untuk menghasilkan informasi yang berguna dalam pengambilan keputusan.

II.3.1. Komponen Sistem Informasi

Menurut Kusrini (2010:9), Dalam suatu sistem informasi terdapat komponen-komponen sebagai berikut :

1. Perangkat Keras (*hardware),* mencakup sebagai piranti fisik seperti computer dan printer
2. Perangkat lunak (*Sofrware)* atau program, sekumpulan intruksi yang memunggkinkan
3. Prosedur, yaitu sekumpulan aturan yang dipakai untuk mewujudkan pemrosesan data dalam pembangkitan keluaran yang dikehendaki
4. Orang atau semua pihak yang bertanggung jawab dalam pengembangan system informasi, pemrosesan dan penggunaan keluaran system informasi.
5. Basis data *(Database)* yaitu sekumpulan table, hubungan dan lain-lain yang berkaitan dengan penyimpanan data.
6. Jaringan komputer dan komunikasi data, yaitu system penghubung yang memungkinkan sumber (*Resources)* dipakai secara bersama-sama atau diakses oleh sejumlah pemakai.

II.4 Sistem Informasi Akuntansi

Menurut Kusrini (2010:9), Sistem Informasi Akuntansi merupakan sistem informasi yang mengubah data transaksi bisnis yang menjadi informasi keuangan yang berguna bagi pemakainya.

Tujuan dari Sistem Informasi Akuntansi adalah :

1. Mendukung operasi sehari-hari
2. Mendukung pengambilan keputusan manajemen
3. Memenuhi kewajiban yang berhubungan dengan pertanggungjawaban

Komponen-komponen yang terdapat dalam Sistem Informasi Akuntansi adalah sebagai berikut :

a. Orang-orang yang mengoperasikan sistem tersebut
b. Prosedur-prosedur, baik manual maupun terotomatisasi, yang dilibatkan dalam pengumpulan, pemrosesan dan penyimpanan data aktivitas-aktivitas organisasi.
c. Data tentang proses-proses bisnis
d. Software yang dipakai untuk memproses data organisasi
e. Infrastruktut teknologi informasi

Didalam organisasi Sistem Informasi Akuntansi berfungsi untuk :

a. Mengumpulkan dan menyimpan aktivitas yang dilaksanakan disuatu organisasi, sumber daya yang dipengaruhi oleh aktivitas-aktivitas tersebut dan para pelaku aktivitas tersebut
b. Mengubah data dan informasi yang berguna bagi manajemen
c. Menyediakan pengendalian yang memadai.

Sistem Informasi Akuntansi merupakan pendukung aktivitas organisasi. Yang termasuk pendukung aktivitas organisasi adalah :

a. Infrastruktur perusahaan, akuntansi, hukum dan administrasi umum
b. Sumber daya manusia : perekrutan, pengontrolan, pelatihan dan kompensasi kepada pegawai.

c. Teknologi : Peningkatan produk dan jasa (Penelitian)

d. Pembelian

Sementara itu aktivitas utamanya adalah :

a. *Inbount Logistics*, penerimaan, penyimpanan dan distribusi bahan-bahan masukan.

b. Operasi : aktivitas untuk mengubah masukan menjadi barang atau jasa

c. *Outbount Logistics :* distribusi produk kepelanggan

d. Pemasaran dan Penjualan

e. Pelayanan : Dukungan purna jual maintenance

II.4.1. Siklus Sistem Informasi Akuntansi

Menurut Kusrini (2010:11), Sistem Informasi Akuntansi memiliki beberapa sistem bagian (*sub system)* yang berupa siklus akuntansi. Siklus akuntansi menunjukkan prosedur akuntansi, mulai dari sumber data sampai ke proses pencatatan/pengolahan akuntansinya. Berikut ini adalah pembagian dari siklus akuntasi.

1. Siklus pendapatan
2. Siklus pendapatan merupakan prosesur pendapatan yang dimulai dari bagian penjualan otorisasi kredit, pengambilan barang, penerimaan barang, penagihan sampai dengan penerimaan kas
3. Siklus pengeluaran kas
4. Siklus pengeluaran kas merupakan prosedur pengeluaran kas yang dimulai dari proses pembelian sampai proses pembayaran.
5. Sikluas konversi

6. Siklus konversi merupakan siklus produksi, dimulai dari bahan mentah sampai barang jadi.
7. Siklus Manajemen Sumber Daya Manusia (SDM)
8. Siklus Manajemen Sumber Daya Manusia (SDM) merupakan siklus yang melibatkan proses penggajian pada karyawan
9. Siklus buku besar dan laporan keuangan

   Siklus ini berupa prosedur pencatatan dan perekaman ke jurnal dan buku besar dan pencetakan laporan keuangan yang datanya diambil dari buku besarnya.

Didalam sebuah Sistem Informasi Akuntansi, tidak semua siklus harus diimplementasikan. Yang wajib ada dalam system tersebut adalah siklus buku besar dan laporan keuangan. Transaksi-transaksi yang termasuk dalam siklus tetapi tidak diimplementasikan, misalnya penggajian, dapat dimasukkan dalam siklus buku besar.

II.5. Account Payable Procedure (Hutang Dagang)

Menurut Supryanto (2014:3*), Account Payable Procedure*, catatan hutang adalah berupa kartu hutang yang diselenggarakan untuk setiap kreditur, yang memperlihatkan catatan mengenai nomor faktur dari pemasok. Jumlah yang terutang, jumlah pembayaran, dan saldo hutang.

Hutang didefinisikan sebagai pengorbanan manfaat ekonomi di masa datang yang bersifat *probable* yang timbul dari kewajiban sekarang dari suatu entitas untuk menyerahkan harta atau menyediakan jasa ke entitas lain di kemudian hari sebagai akibat dari transaksi atau kejadian masa lalu.

Ada beberapa jenis hutang jangka pendek, yaitu:

1. Hutang dagang Account Payable adalah jumlah uang yang masih harus dibayarkan kepada pemasok, karena perusahaan melakukan pembelian barang.
2. Hutang Wesel atau Promes adalah kewajiban yang dibuktikan dengan janji tertulis tanpa syarat untuk membayar sejumlah uang tertentu pada tanggal yang telah ditentukan di kemudian hari.
3. Beban-beban yang masih harus dibayar atau accrual liabilities adalah kewajiban terhadap beban-beban yang telah terjadi, tapi belum dibayar karena belum jatuh tempo pada akhir periode yang bersangkutan.
4. Hutang Deviden adalah deviden yang dapat dibayar sebagaimana diumumkan oleh dewan komisaris perusahaan tapi pada akhir periode beum dibayar dan dicatat sebagai hutang deviden.
5. Pendapatan yang diterima di muka.
6. Bagian dari hutang Jangka Panjang yang jatuh tempo.
7. Penyajian hutang lancar dalam Neraca.

Elemen pengendalian internal terhadap hutang jangka pendek adalah:

1. Diselenggarakannya catatan hutang (dengan kartu atau arsip *voucher* yang belum dibayar) yang secara periodic direkonsiliasi dengan rekening kontrol hutang yang bersangkutan di dalam buku besar.
2. Diadakan pengecekan informasi di dalam buku catatan hutang dagang dengan menggunakan pernyataan piutang diterima dari kreditur.

3. Adanya prosedur yang memberitahu bagian hutang mengenai adanya barang yang dikembalikan kepada penjual.
4. Faktur dari penjual harus dicek pertama kali oleh bagian pembelian sebelum dibayar.
5. *Voucher* dan dokumen pendukungnya dicap “lunas” atau diberi tanda setelah dilaksanakan pembayarannya, untuk mengindari pembayaran kedua kalinya.
6. Adanya otorisasi dari pihak berwenang untuk setiap pembayaran hutang jangka pendek.
7. Semua penarikan hutang harus diotorisasi oleh yang berwenang..

II.6. *Unified Modeling Language* (*UML)*

Yuni Sugiarti (2013:34), *Unified Modeling Language* (*UML)* adalah sebuah “bahasa” yang telah menjadi standar industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, system operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa-bahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi procedural dalam VB atau C.

II.6.1. Pengenalan UML

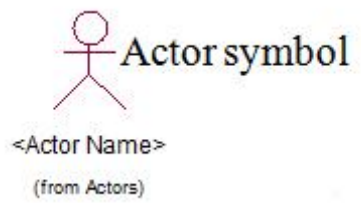
Yuni Sugiarti (2013:36), UML itu adalah salah bentuk language atau bahasa. UML didefinisikan sebagai bahsa visual untuk menjelaskan, memberikan spesifikasi, merancang, membuat model, dan mendokumentasikan aspek-aspek dari sebuah sistem. Karena tergolong bahasa visual, UML lebih mengedepankan penggunaan diagram untuk menggambarkan aspek dari sistem yang sedang dimodelkan.

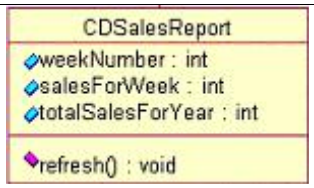
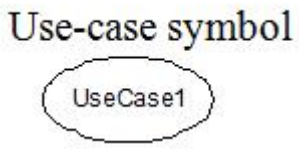
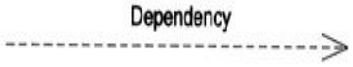
UML adalah salah satu bentuk notasi atau bahsa yang sama digunakan oleh professional dibidang *software* untuk mengambarkan atau memodelkan sebuah sistem. Sebelumnya ada banyak notasi atau bahasa lain untuk mecapai keperluan misalnya DFD (*Data Flow Diagram*) dan Booch Diagram. UML telah menjadi *de facto standard language.*

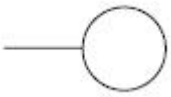
UML berfungsi sebagai jembatan dalam mengkomunikasikan beberapa aspek dalam system melaui sejumlah elemen grafis yang bias dikombinasikan menjadi diagram. UML mempunyai banyak diagram yang dapat mengakomodasikan berbagai sudut pandang dari suatu perangkat lunak yang akan dibangun.

Simbol-simbol UML dapat dilihat pada tabel di bawah ini.

Table II.1. Simbol-simbol UML

No	Simbol	Keterangan
1		*Actor*; menentukan peran yang dimainkan oleh *user* atau sistem lain yang berinteraksi dengan subjek. *Actor* adalah segala sesuatu yang berinteraksi langsung dengan sistem aplikasi komputer, seperti orang, benda atau lainnya. Tugas *actor* adalah memberikan informasi kepada sistem dan dapat memerintahkan sistem

		untuk melakukan sesuatu tugas.
2		*Class diagram*; Notasi utama dan yang paling mendasar pada diagram UML adalah notasi untuk mempresentasikan suatu *class* beserta dengan atribut dan operasinya. *Class* adalah pembentuk utama dari sistem berorientasi objek.
3		*Use case* merupakan awal yang sangat baik untuk setiap fase pengembangan berbasis objek, design, testing, dan dokumentasi yang menggambarkan kebutuhan sistem dari sudut pandang di luar sistem.
		*Realization*; *Realization* menunjukkan hubungan bahwa elemen yang ada di bagian tanpa panah akan merealisasikan apa yang dinyatakan oleh elemen yang ada di bagian dengan panah.
		*Interaction*; *Interaction* digunakan untuk menunjukkan baik aliran pesan atau informasi antar obyek maupun hubungan antar obyek.
		*Dependency*; *Dependency* merupakan relasi yang menunjukan bahwa perubahan pada salah satu elemen memberi pengaruh pada elemen lain. Terdapat 2 *stereotype* dari *dependency*, yaitu *include* dan *extend*. Include menunjukkan bahwa suatu bagian dari elemen (yang ada digaris tanpa panah) memicu eksekusi bagian dari elemen lain (yang ada di garis dengan panah). *Extend* menunjukkan bahwa suatu bagian dari elemen di garis tanpa panah bisa disisipkan ke dalam elemen yang ada di garis dengan panah.
		*Note*; *Note* digunakan untuk memberikan keterangan atau komentar tambahan dari suatu elemen sehingga bisa langsung terlampir dalam model. *Note* ini bisa disertakan ke semua elemen notasi yang lain.
		*Association*; *Association* menggambarkan navigasi antar *class* (*navigation*), berapa banyak obyek lain yang bisa berhubungan dengan satu obyek (*multiplicity* antar

		class) dan apakah suatu *class* menjadi bagian dari *class* lainnya (*aggregation*).
		*Generalization*; *Generalization* menunjukkan hubungan antara elemen yang lebih umum ke elemen yang lebih spesifik.
		*Package; package* adalah mekanisme pengelompokkan yang dgunakan untuk menandakan pengelompokkan elemen-elemen model.
		*Interface*; *Interface* merupakan kumpulan operasi berupa implementasi dari suatu *class*. Atau dengan kata lain implementasi operasi dalam *interface* dijabarkan oleh operasi di dalam *class*.

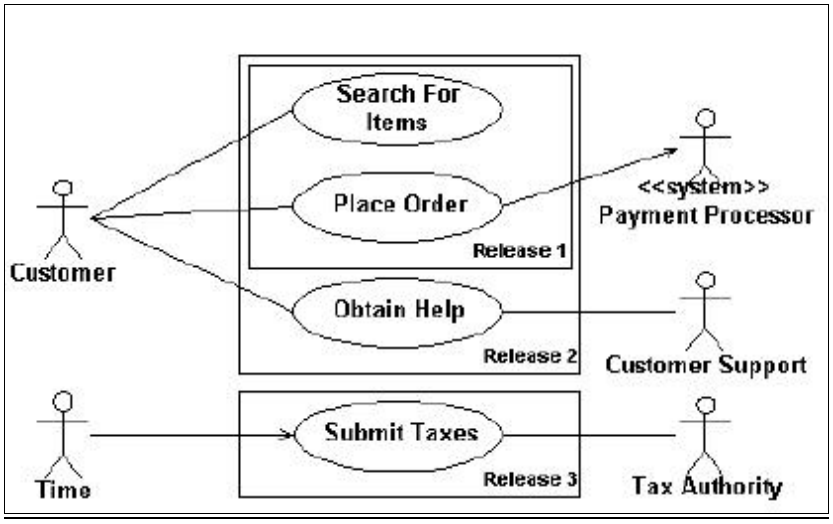
(Sumber : Yuni Sugiarti 2013 : 57)

Menurut Rossa A.S dan M. Shalahuddin (2013:155), Adapun jenis-jenis dari tipe diagram *UML* adalah sebagai berikut :

1. *Use Case Diagram*

   Use case atau diagram use case merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. Use case mendeskripsikan sebuah interaksi antara satu atau lebih actor dengan sistem informasi yang akan dibuat. Dengan kata lainnya, use case digunakan untuk mengetahui fungsi apa saja yang berhak menggunkan fungsi-fungsi itu.

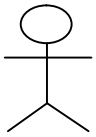
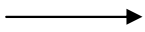
   Use *Case* Diagram *Use Case* menunjukkan 3 aspek dari sistem yaitu : *actor, use case dan system/sub system boundary*. *Actor* mewakili peran orang, *system* yang lain atau alat ketika berkomunikasi dengan *use case*. Berikut gambar notasi *use case :*



Gambar II.1. Contoh *Use Case* Diagram
(Sumber : Haviluddin 2011 : 4)

Berikut ini Tabel simbol-simbol dari *Use Case* Diagram adalah:

Tabel II.2. Simbol-simbol *Use Case Diagram*

Nama Komponen	**Deskripsi**	**Gambar**
Use Case	Menunjukkan proses yang terjadi pada system	
Actor	Menunjukkan user yang akan menggunakan sistem.	
Association	Sebuah garis yang berfungsi menghubungkan *Actor* dengan *Use Case*.	
Extend	Perluasan dari *Use Case* lain jika kondisi atau syarat terpenuhi.	<<extends>> 
Include	Menjelaskan bahwa *Use Case* termasuk dalam *Use Case* lain.	<<include>> 

(Sumber : Rosa A.S , M Shalahuddin 2013 : 156)

2. *Class Diagram*

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.

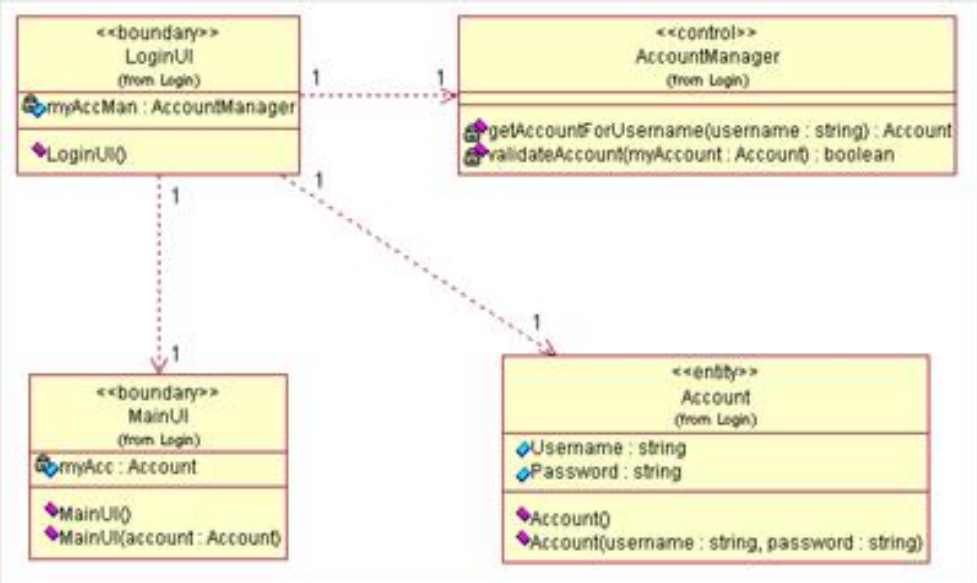
a. Atribut merupakan variable yang dimiliki oleh suatu kelas.
b. Atribut mendeskripsikan property dengan sebaris teks didalam kotak kelas tersebut.
c. Operasi atau metode adalalh fungsi-fungsi yang dimiliki oleh suatu kelas tersebut.

Diagram kelas mendeskripsikan jenis-jenis objek dalam sistem dan berbagai hubungan statis yang terdapat diantara mereka. Diagram kelas juga menunjukan properti dan operasi sebuah kelas dan batasan-batasan yang terdapat dalam hubungan-hubungan objek tersebut.

Diagram kelas menggambarkan struktur dan deskripsi class, package dan objek beserta hubungan satu sama lain seperti containment, pewarisan, asosiasi, dan lain-lain.

Kelas memiliki tiga area pokok :

b. Nama (dan *stereotype*)
c. Atribut
d. Method



Gambar II.2 Contoh *Class* diagram.
(Sumber : Haviluddin2011 : 3)

3. *Activity Diagram*

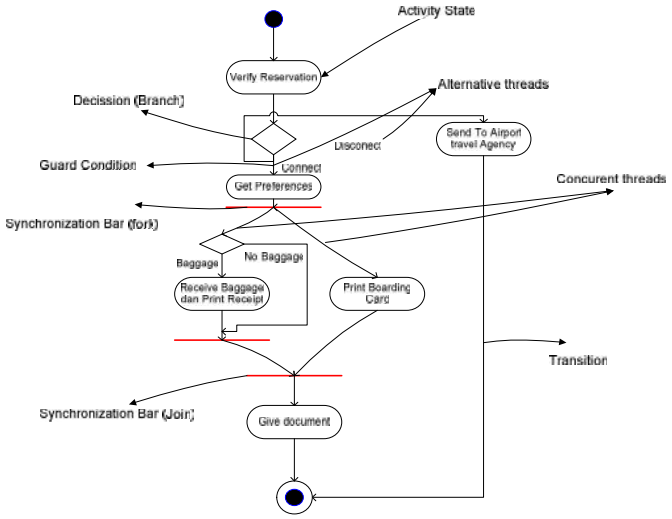
Activity Diagram atau diagram aktivitas menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Yang perlu diperhatikan disini adalah bahwa *Activity Diagram* menggambarkan aktivitas system bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem.

Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut:

a. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
b. Urutan atau pengelompokkan tampilan dari sistem/*user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.

c. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujiannya.

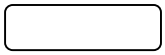
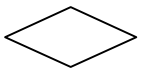
Activity Diagram menggambarkan berbagai aliraktivitas dalam system yang sedang dirancang, bagaimana masing-masing berawal, decision yang mungkin terjadi, dan bagaimana proses parallel yang mungkin terjada pada beberapa eksekusi.

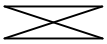
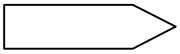
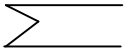


Gambar II.3 Contoh *Activity* diagram.
(Sumber : Haviluddin 2011 : 4)

Berikut ini Tabel simbol-simbol dari *Use Case* Diagram adalah:

Tabel II.3. Simbol Yang Ada Pada *Activity Diagram*

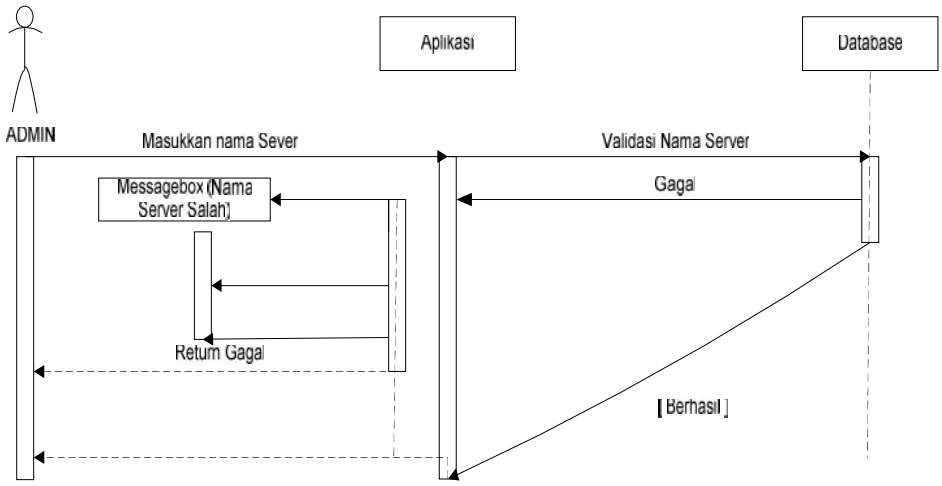
SIMBOL	KETERANGAN
	Titik Awal
	Titik Akhir
	*Activity*
	Pilihan untuk pengambilan keputusan.

	*Fork* : digunakan untuk menunjukkan kegiatan yang dilakukan secara 31cenario atau untuk menggabungkan dua kegiatan 31cenario menjadi satu.
	*Rake* ; menunjukkan adanya dekomposisi
	Tanda waktu
	Tanda pengiriman
	Tanda penerimaan
	Aliran Akhir (*Flow Final*)

(Sumber : Yuni Sugiarti 2013 : 59)

4. *Sequence Diagram*

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan message yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu. Membuat diagram sekuen juga dibutuhkan untuk melihat scenario yang ada pada *use case*.

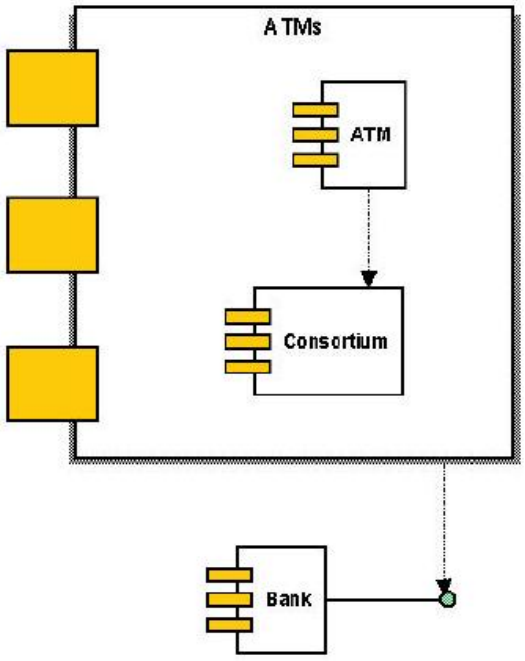


Gambar II.4 Contoh *Sequence* diagram.
(Sumber : Haviluddin 2011 : 5)

5. *Component Diagram*

Component Diagram mengandung *component, interface* dan *relationship.*

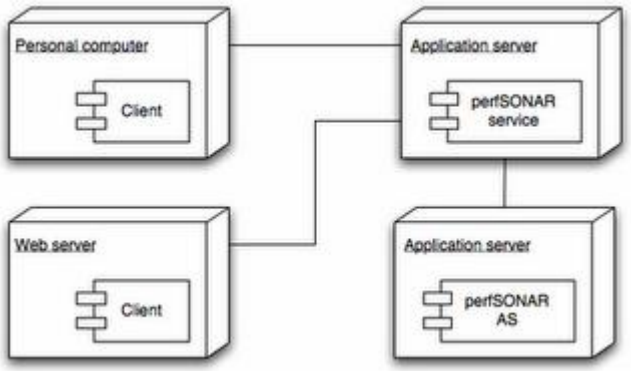
Notasi *component Diagram* dapat dilihat seperti gambar di bawah ini :



Gambar II.5. Notasi *Component Diagram*
(Sumber : Haviluddin 2011 : 3)

6. *Deployment Diagram*

Deployment Diagram menunjukkan tata letak sebuah sistem secara fisik, menampakkan bagian-bagian *software* yang berjalan pada bagian-bagian *hardware*. Sistem terdiri dari node-node dimana setiap node diwakili untuk sebuah kubus. Garis yang menghubungkan antara 2 kubus menunjukkan hubungan di antara kedua node tersebut. Tipe node bisa berupa *device* yang berwujud *hardware* dan bisa juga *processor*.



Gambar II.6. Contoh *Deployment* diagram.
(Sumber : Haviluddin 2011 : 4)

II.7. Pengertian Basis Data (*Database*)

Istilah basis data tersusun atas dua suku kata, yaitu basis dan data (basis data = basis + data). Dalam sistem bilangan biner, kita dapat menuliskan beberapa contoh bilangan sebagai berikut (Edhy Sutanta:2011:25).

0 → sama dengan 0 dalam sistem bilangan desimal.

1 → sama dengan 1 dalam sistem bilangan desimal.

10 → sama dengan 2 dalam sistem bilangan desimal.

11 → sama dengan 3 dalam sistem bilangan desimal.

100→ sama dengan 4 dalam sistem bilangan desimal.

Menurut Edhy Sutanta (2011:29) istilah basis data dapat dipahami sebagai suatu kumpulan data terhubung (*interrelated data*) yang disimpan secara bersama-sama pada suatu media, tanpa mengatap satu sama lain atau tidak perlu suatu kerangkapan data (kalau ada kerangkapan data tersubut harus seminimal mungkin dan terkontrol.

II.7.1. Tujuan Basis Data

Menurut Edhy Sutanta (2011:53), Basis data bertujuan untuk mengatur data sehingga diperoleh kemudahan, ketepatan dan kecepatan dalam pengambilan informasi. Untuk mencapai tujuan, ada lima aspek penting dalam basis data yaitu :

a. Kerangkapan data (*data redundancy*)

   Kerangkapan data (*data redundancy*) adalah munculnya data –data yang sama secara melimpah (berulang kali) pada file basis data yang semestinya tidak diperlukan. Misalnya ada data mahasiswa yang memuat nim, nama, alamat dan atribut lainya, sementara kita punya data lain tentang data kartu hasil studi mahasiswa yang isinya terdapat nim, nama, mata kuliah dan nilai. Pada kedua data tersebut kita temukan atribut nama.

b. Inkonsisistensi data (*data inconsistency*)

   Inkonsisistensi data (*data inconsistency*) atau data tidak konsisten adalah munculnya data yang tidak konsisten pada medan/kolom yang sama dalam satu atau beberapa file data yang dihubungkan/ direlasikan.

c. Data Terisolasi (*data isolation*)

   Data terisolasi disebabkan oleh pemakaian beberapa file basis data dimana program aplikasi tidak dapat mengakses data-data dari file tertentu, kecuali

bila program aplikasi diubah/ditambah sehingga seolah-olah ada file yang terpisah/terisolasi terhadap file lain dalam basis data.

d. Keamanan data (*data security*)

Keamanan data (*data security*) adalah merupakan aspek kritis dalam basis data. Prinsip dasar dari keamanan data dalam basis data adlah bahwa data-data dalam basis data merupakan sumber informasi yang bersifat sangat penting dan rahasia.

e. Integritas data (*data integrity*)

Integritas data (*data integrity*) berhubungan dengan kinerja sistem agar dapat melakukan kendali/kontrol pada semua bagian sistem. Integritas dimaksudkan sebagai suatu sarana untuk meyakinkan bahwa data-data yang tersimpan dalam basis data selau berada dalam kondisi yang benar.

II.7.2. Keuntungan Pengembangan Basis Data

Menurut Edhy Sutanta (2011, 49), Penyusunan suatu basis data dimaksudkan untuk mengatasi permasalahan-permasalahan pada saat pengolahan data. Basis data yang dikembangkan dengan benar, sesuai dengai batasan/kriteria pengolahan data secara basis data akan memberikan beberapa keuntungan, yaitu :

1. Kerangkapan data dapat diminimalkan

Jika file-file dalam basis data dalam program aplikasi diciptakan oleh perancang yang berbeda pada waktu yang berselang cukup lama maka beberapa bagian data akan mengalami kerangkapan. Pengembangan basis data yang sesuai dengan definisi basis data seperti di muka dapat menghindarkan terjadinya kerangkapan data tesebut.

2. Inkonsistensi data dapat dihindari

   Basis data yang terbebas dari kerangkapan data akan terhindar dari munculnya data-data yang tidak konsisten.

3. Data dalam basis data dapat digunakan secara bersama (multiuser)

   Dalam rangka meningkatkan kinerja sistem dan untuk memperoleh respon waktu yang cepat. Beberapa sistem mengizinkan banyak pemakai untuk dapat meng-update data secara simultan.

4. Standarisasi data dapat dilakukan

   Definisi file basis data di dalam kamus data memungkinkan dilakukannya penerapan standarisasi data dalam basis data. Sekalipun demikian, pada implementasinya sering kali standarisasi data tidak cukup dilakukan pada definisi file basis data. Masih diperlukan proses standarisasi lebih lanjut atas nilai-nilai rinci data melalui form input data.

5. Pembatasan untuk keamanan data dapat diterapkan

   Data-data dalam basis data dapat di atur sehingga hanya pemakai tertentu yang mempunyai wewenang saja yang dapat mengaksesnya.

6. Integritas data dapat dipelihara

   Intergritas berhubungan dengan kinerja sistem agar dapat melakukan kendali/kontrol pada semua bagian sistem sehingga sistem selalu beroperasi dalam pengendalian penuh.

7. Perbedaan kebutuhan data dapat diseimbangkan

   Setiap pemakai dalam sistem memiliki kebutuhan yang berbeda-beda. Pengembangan basis data yang benar mampu menyeimbangkan

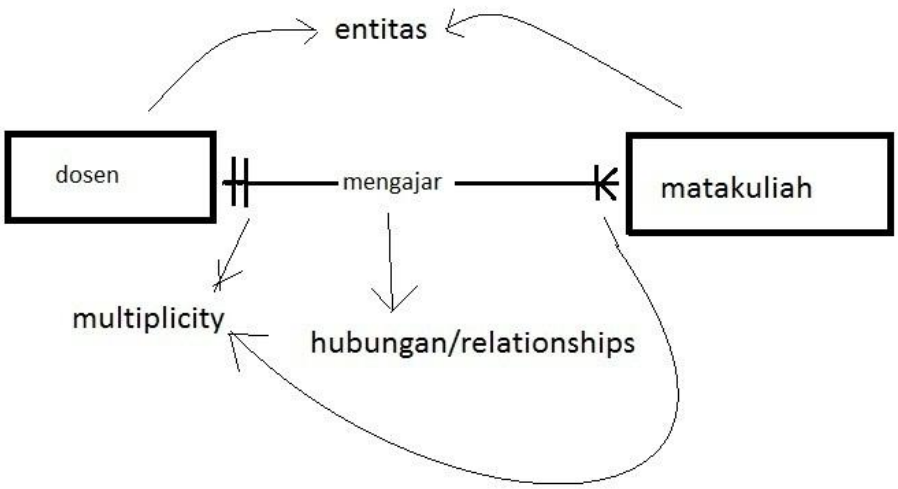
perbedaan-perbedaan kebutuhan tersebut karena secara konseptual akan menggunakan basis data yang sama.

II.7.3. Pemodelan Basis Data

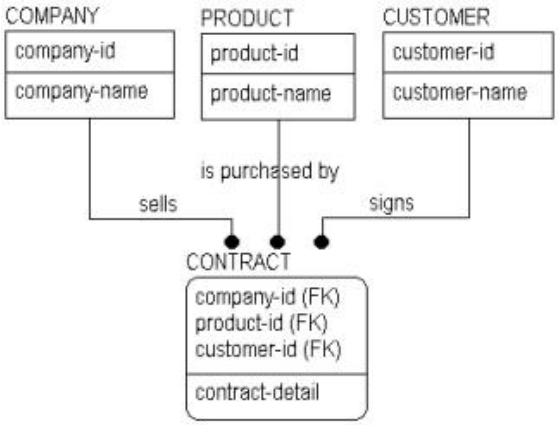
Menurut Edhy Sutanta (2011:88),*Semantic Model* merupakan suatu model data yang dikembangkan berdassarkan obyek. *Semantic Model* digunakan untuk menjelaskan hubungan antar basis data kepada pengguna secara logik. *Semantic Model* hampir sama dengan *Entity Relationship Model.* Perbedaanya adalah kerelasian antar obyek dasar tidak dinyatakan dengan symbol, teapi secara semantic, yaitu menggunakan kata-kata (*semantic*). Untuk merancang suatu aplikasi basis data alat yang biasa digunakan adalah *Entity Relationship Diagram (ERD).* ERD didasarkan dari artikel yang dipublikasikan oleh Peter Phin Shan Chen.

Entity Relationship Model adalah abstraksi konseptual yang mewakili struktur dari suatu basis data.

Dalam perkembangannya banyak diciptakan notasi ERD yang berbeda-beda seperti telihat gambar dibawah ini.



Gambar II.7. Diagram Dengan Notasi Crows Foot



Gambar II.8. Diagram Dengan Notasi Relational
(Sumber : Edhy Sutanta 2011 : 90)

II.7.4. Normalisasi

Menurut Samiaji Sarosa (2010:6-9), Normalisasi adalah teknik yang dirancang untuk merancang tabel basis data relasional untuk meminimalkan duplikasi data dan menghindarkan basis data tersebut anomali. Suatu basis data dikatakan tidak normal jika terjadi 3 (tiga) anomali berikut :

1. *Insertion Anomaly*

   Anomali yang terjadi jika ada data yang tidak bisa disisipkan kedalam table.

Contoh tabel bentuk tidak normal dapat dilihat pada tabel di bawah ini.

Tabel II.4. Bentuk tidak normal

Kode Dokter	Nama Dokter	Spesialisasi	Nama Pasien
D1	Bashri	Kandungan	Rini Santi
D2	Andarini	Kulit	Shanti Anwar
D3	Irawan	Mata	Wijaya

(Sumber : Samiaji Sarosa 2010 : 6)

2. *Update/Modification anomaly*

Anomali yang terjadi jika ada perubahan pada suatu item data maka harus mengubah lebih dari satu baris data.

Langkah-langkah normalisasi sampai pada bentuk 3NF adalah sebagai berikut :

a. First Nornal Form (1NF)

Untuk menjadi 1NF suatu table harus memenuhi dua syarat. Syarat pertama tidak ada kelompok data atau *field* yang berulang. Syarat kedua harus ada *primary key (PK)* atau kunci unik, atau kunci yang membedakan satu bari dengan baris yang lain dalam satu table. Pada dasarnya sebuah table selamat tidak ada kolom yang sama merupakan bentuk table dengan 1NF.

Tabel II.5. Bentuk *First Nornal Form (1NF)*

Kode Dokter	Nama Dokter	Spesialisasi	Nama Pasien
D1	Bashri	Kandungan	Rini
D1	Bashri	Kandungan	Santi
D2	Andarini	Kulit	Shanti
D2	Andarini	Kulit	Anwar
D3	Irawan	Mata	Wijaya

(Sumber : Samiaji Sarosa 2010 : 7)

b. Second Nornal Form (2NF)

Untuk menjadi 2NF suatu table harus berada dalam kondisi 1NF dan tidak memilik *partial dependencies. Partial dependencies* adalah

suatu kondisi jika atribut non kunci (Non PK) tergantung sebagian tetapi bukan seluruhnya pada PK.

Tabel II.6. Bentuk *Second Nornal Form (2NF)*

Kode Dokter	Nama Dokter	Spesialisasi
D1	Bashri	Kandungan
D2	Andarini	Kulit
D3	Irawan	Mata

Kode Dokter	Nama Pasien
D1	Rini
D1	Santi
D2	Shanti
D2	Anwar
D3	Wijaya

(Sumber : Samiaji Sarosa 2010 : 9)

c. Third Nornal Form (3NF)

Untuk menjadi 3NF suatu table harus berada dalam kondisi 2NF dan tidak memilik *transitive dependencies. Transitive dependencies* adalah suatu kondisi dengan adanya ketergantunga fungsional antara 2 atau lebih atribut non kunci (Non PK).

Tabel II.7. Bentuk *Third Nornal Form (3NF)*

Kodedkt	NamaDkt	KodePsn	NamaPsn	No	KodeObt	NamaObt
D1	Bashri	P1	Shanti	1	OB01	Refagan
D1	Bashri	P1	Shanti	2	OB02	Panadol
D1	Bashri	P1	Shanti	3	OB03	Balfirik
D2	Andarini	P2	Anwar	1	OB02	Panadol
D2	Andarini	P2	Anwar	2	OB03	Balfirik
D3	Irawan	P3	Wijaya	1	OB02	Panadol
D3	Irawan	P3	Wijaya	2	OB03	Balfirik

KodeObt	NamaObt
OB01	Refagan
OB02	Panadol
OB03	Balfirik

(Sumber : Samiaji Sarosa 2010 : 9)

II.7.5. SQL Server 2008

Menurut Ema dan Anggit Dwi Hartanto (2012:63), Bahasa *query* merupakan bahasa khusus yang digunakan untuk melakukan manipulasi dan menanyakan pertanyaan (query) yang berhubungan dengan data dalam basis data. Bahasa query tidak sama dengan bahasa pemrograman, dimana bahasa query tidak memiliki kemampuan untuk menyelesaikan banyak masalah seperti bahasa pemrograman pada umumnya.

Dalam pemrograman basis data, salah satu bahasa yang harus kita kuasai adalah SQL. SQL merupakan bahasa komputer standar yang digunakan untuk

berkomunikasi dengan sistem manajemen basis data relasional (RDBMS). SQL sering disebutkan sebagai singkatan dari *Stuctrured Query Language*.

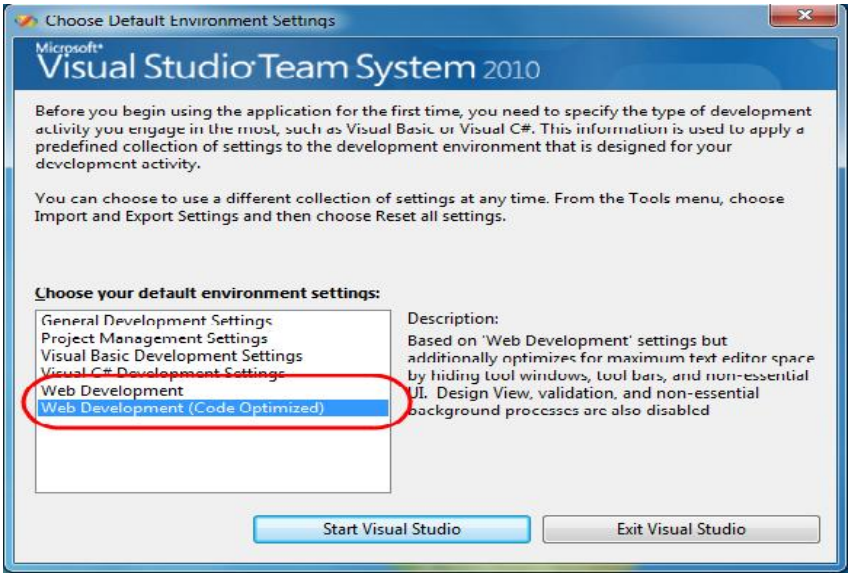
II.8. Visual Basic 2010

Menurut Wahana Komputer (2010:2), Visual Basic 2010 merupakan salah satu bagian dari produk pemrograman terbaru yang dikeluarkan oleh Microsoft, yaitu Micosoft Visual Studio 2010. Sebagai produk lingkungan pemngembangan terintegrasi atau IDE andalan yang dikeluarkan oleh Microsoft, Visual Studio 2010 menambahkan perbaikan-perbaikan fitur dan fitur baru yang lebih lengkap dibandingkan versi Visual Studio pendahulunya, yaitu Microsoft Visual Studio 2008.

Visual Studio meruapakan produk pemrograman andalan Microsoft Corporation, yang di dalamnya berisi beberapa jenis IDE pemrograman seperti *Visual Basic, Visual C++, Visual Web Developer. Visual C#, dan Visual F#.* Semua IDE pemrograman tersebut sudah mendukung penuh implementasi .*Net Framework* terbaru, yaitu .*Net Framework* 4.0 yang merupakan pengembangan dari .*Net Freamework* 3.5. Adapun *database* standar yang disertakan adalah *Microsoft SQL Server 2008 Express.*

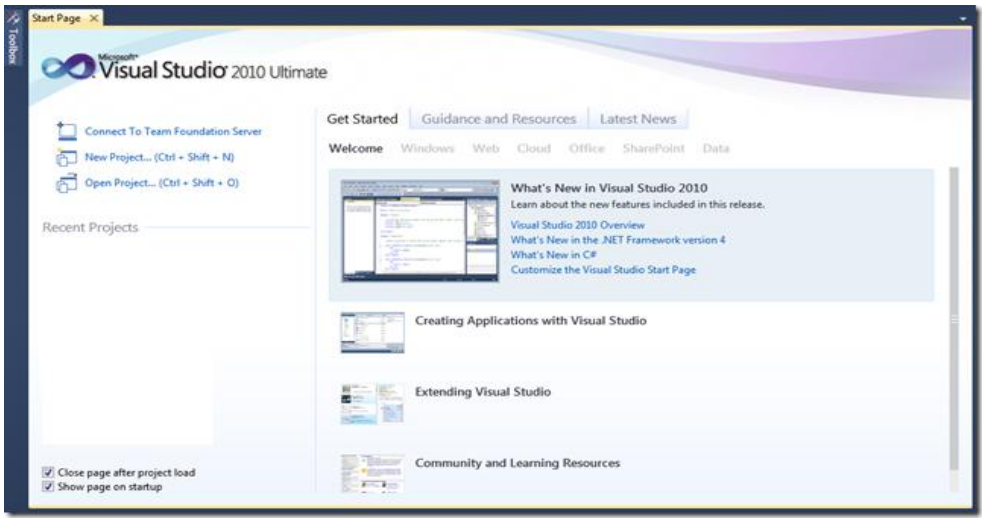
II.8.1. Antar Muka Visual Basic 2010

Saat menjalankan Visual Basic 2010 pertama kali muncul jendela *chose default environtment settings.* Disini bisa memilih apakah ingin memilih antar muka di Visual Studio. Untuk *programmer* Visual Basic lebih baik memilih *Visual Basic Development Centre.*



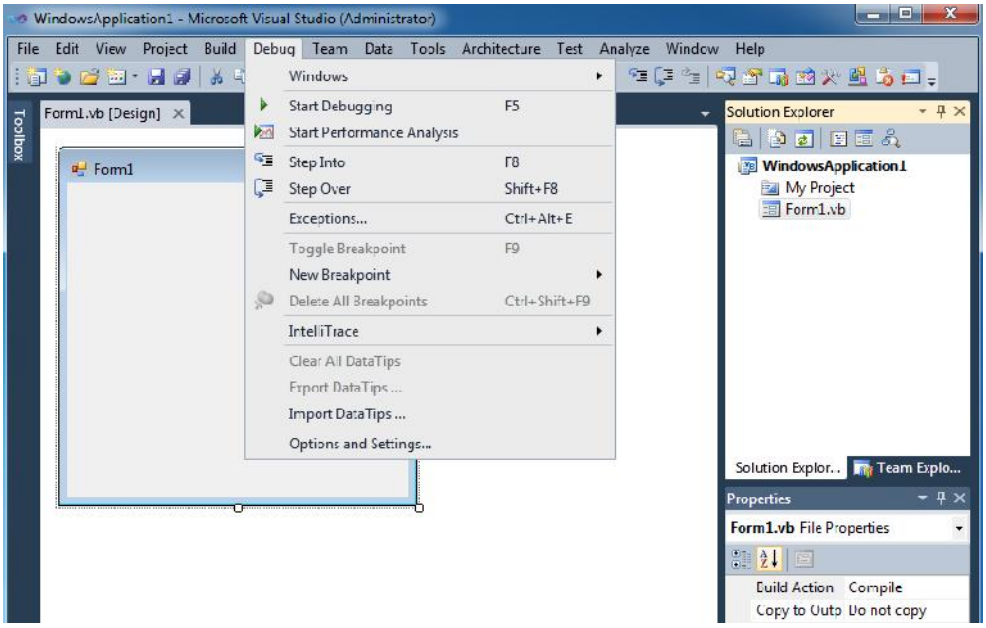
Gambar II.9. Form Chose Default Environtment Settings
(Sumber : Wahana Komputer 2010 : 11)

Dibagian awal Visual Basic, bisa memilih *Start Page*. *Start Page* adalah halaman yang mencantumkan informasi-informasi seputar program dan juga informasi RSS dari sumber tertentu. Jika tidak ingin menampilkan hal ini hilangkan tanda centang pada *Show Page On Startup.*



Gambar II.10. Start Page Visual Basic 2010
(Sumber : Wahana Komputer 2010 : 12)

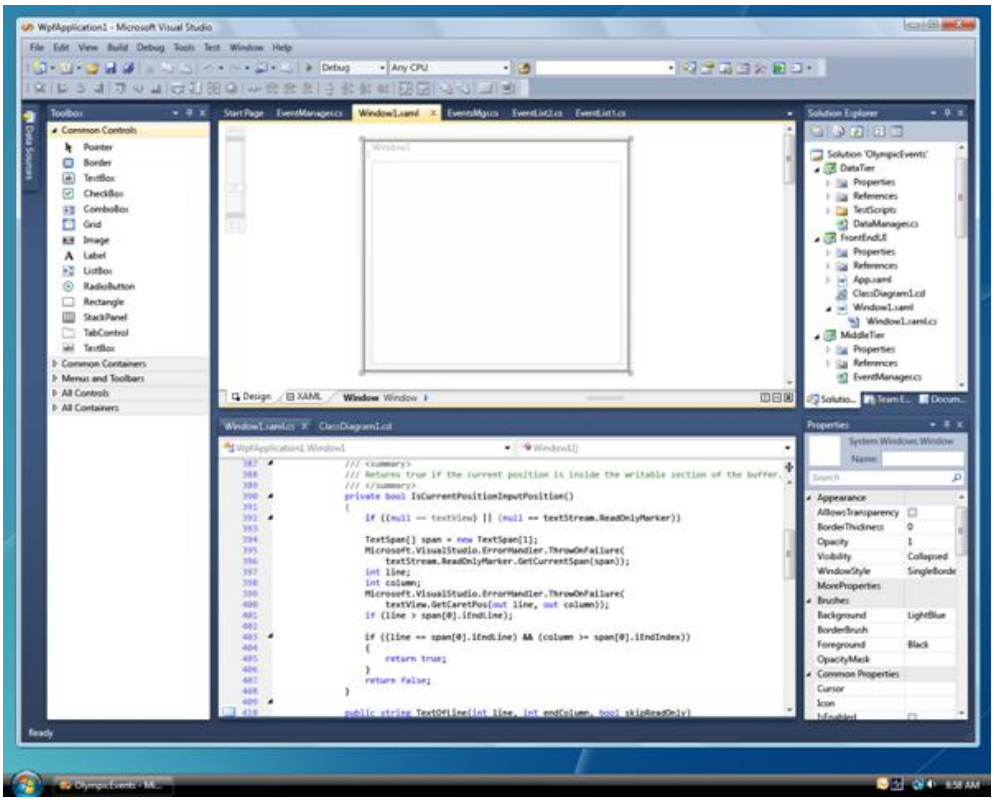
Secara umum, setiap anda menjalankan visual basic pertama kali, Anda akan dihadapkan oleh Start Page yang berisi berbagai macam informasi yang akan mengantarkan kepada anda semua hal mengenai Visual Basic 2010 secara detail. Apabila Anda terkoneksi dengan internet, maka bagian Start Page juga akan menampilkan informasi terkini mengenai Visual Basic 2010 yang berasal dari RSS Feed yang disediakan, sehingga Anda tidak akan ketinggalan informasi terbaru mengenai Visual Basic 2010.



Gambar II.11. Tampilan IDE (*Integrated Development Environtment)* setelah *Start Page* ditutup
(Sumber : Wahana Komputer 2010 : 13)

Menu aplikasi berisi daftar menu lengkap terletak disebelah atas aplikasi yang bisa diakses oleh pengguna Visual Basic 2010 mulai dari membuat aplikasi baru, menjalankan aplikasi, melakukan debugging aplikasi, mengakses bantuan, dan sebagainya. Untuk dapat mengkases menu aplikasi, lakukan klik pada salah satu tulisan menu yang terdapat disebelah atas aplikasi Visual Basic 2010.

Jika ada sebuah form yang terlihat, tampilan lengkap IDE seperti gambar berikut ini:



Gambar II.12. Tampilan lengkap IDE
(Sumber : Wahana Komputer 2010 : 15)

Komponen-komponen dari IDE adalah :

1. Solution Explorer merupakan fitur yang terletak di sebelah kanan atas di bawah menu aplikasi yang digunakan untuk menampilkan daftar desain form dengan struktur tree dan project yang sedang dibuka.
2. Properties merupakan fitur dari Visual Basic 2010 yang digunakan untuk melakukan pengaturan properti dari objek-objek yang digunakan dalam desain form yang sedang anda buat.

3. Komponen Toolbox merupakan fitur Visual Basic 2010 yang digunakan untuk menampilkan daftar dari komponen baik visual maupun nonvisual yang bisa ditambahkan ke dalam desain form yang sedang anda buat.
4. Form designer merupakan fitur dari Visual Basic 2010 yang digunakan untuk membuat desain antarmuka atau interface dari aplikasi yang anda kembangkan. (Wahana Komputer, 2010:13-15)