BAB II

LANDASAN TEORI

II.1. Sistem Informasi

Sistem Informasi adalah sistem pemrosesan data , sistem yang biasanya terdiri dari sekumpulan komponen baik manual ataupun berbasis komputer yang terintegrasi untuk mengumpulkan, menyimpan, dan mengelola data serta menyediakan informasi kepada pihak-pihak yang berkepentingan sebagai pengguna informasi tersebut. (Anastasia Diana dan Lilis Setiawati ; 2011 ; 4)

II.2. Sistem Informasi Akuntansi

Sistem Informasi Akuntansi adalah sistem yang bertujuan untuk mengumpulkan dan memproses data serta melaporkan informasi yang berkaitan dengan transaksi keuangan. (Anastasia Diana dan Lilis Setiawati ; 2011 ; 4)

II.2.1. Tujuan Sistem Informasi Akuntansi

Adapun tujuan dari sistem informasi sebagai berikut :

1. Mengamankan harta / kekayaan perusahaan meliputi kas perusahaan dan persediaan barang perusahaan.
2. Menghasilkan beragam informasi untuk pengambilan keputusan.
3. Menghasilkan informasi untuk pihak eksternal.
4. Menghasilkan informasi untuk penilaian kinerja karyawan atau divisi.
5. Menyediakan data masa lalu untuk kepentingan audit.

6. Menghasilkan informasi untuk penyusunan dan evaluasi anggaran perusahaan.
7. Menghasilkan informasi yang diperlukan dalam kegiatan perencanaan dan pengendalian.

II.3. Metode Garis Lurus & Metode Jam Jasa

Metode garis lurus digunakan berdasarkan aktiva tetap tersebut setiap periode sama, Apabila metode garis lurus ini menghasilkan beban penyusutan yang jumlahnya sama setiap periode, maka akan terjadi pembandingan yang tepat antara pendapatan dengan biaya.

Dengan menggunakan metode garis lurus, besarnya beban penyusutan periodik dapat dihitung sebagai berikut :

$$\text{Rumus} = \frac{\text{Harga Perolehan} - \text{Estimasi Nilai Residu}}{\text{Estimasi Masa Manfaat}}$$

(Sumber : Hery, 2014 : 143)

Metode Jam Jasa ini berdasarkan aktiva tetap dihitung dengan dasar satuan jam jasa. Besarnya beban penyusutan ini akan berfluktuasi setiap periodenya tergantung pada jumlah konstribusi jam jasa yang diberikan oleh aset bersangkutan.

Dalam menghitung besarnya beban penyusutan, metode ini membutuhkan estimasi umur aset berupa jumlah jam jasa yang dapat diberikan oleh aset bersangkutan.

Rumus = Harga Perolehan – Estimasi Nilai Residu

Estimasi Total Jam Jasa

(Sumber : Hery, 2014 : 150)

II.4. Basis Data (Database)

Istilah basis data/*database* 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.

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

II.4.1. Tujuan Basis Data

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.(Edhy Sutanta ; 2011 ; 53)

II.4.2. Manfaat/Keuntungan Basis Data

Ada banyak manfaat yang diperoleh dengan menggunakan basis data, Manfaat/keuntungan basis data diantaranya adalah :

1. Kerangkapan data dapat diminimalkan

   Jika file-file basis data dalam program aplikasi disiptakan oleh perancang yang berbeda pada waktu yang berselang cukup lama maka beberapa bagian data akan mengalami kerangkapan.

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 meningkatkan kinerja sistem dan untuk memperoleh respons waktu yang cepat, beberapa sistem mengizinkan banyak pemakai untuk dapat meng-*update* data secara bersamaan

4. Standarisasi data dapat dilakukan

   Definisi file basis data di dalam kamus data memungkinkan dilakukannya penerapan standarisasi data dalam basis data.

5. Pembatasan untuk keamanan untuk keamanan data dapat diterapkan

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

6. Integritas data dapat dipelihara

   Integritas 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 menggunkan basis data yang sama. (Edhy Sutanta;2011;49-50)

II.5. Database SQL Server

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*. (Ema dan Anggit Dwi Hartanto;2012:62)

II.6. Visual Basic 2010

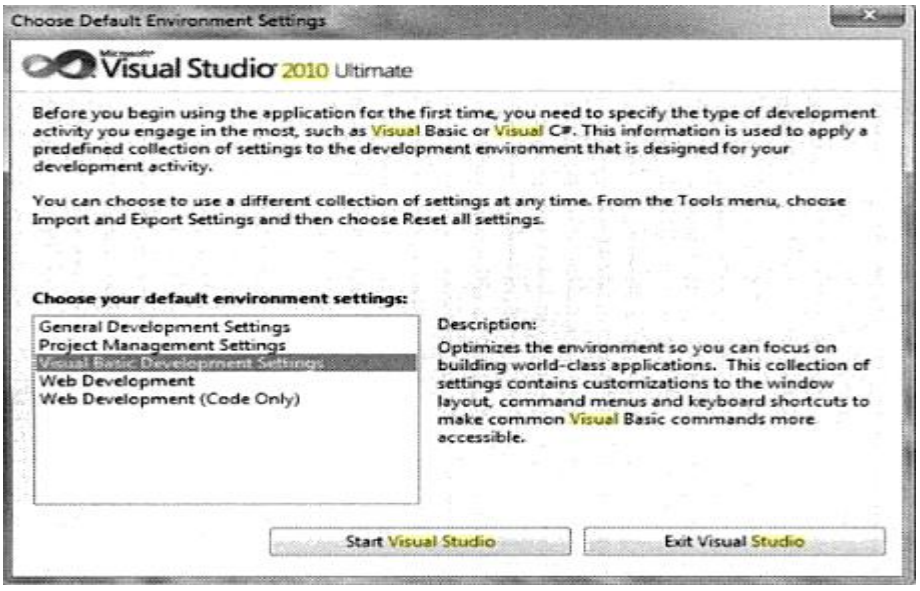
Visual Basic 2010 merupakan salah satu bagian dari produk pemrograman terbaru yang dikeluarkan oleh *Microsoft*, yaitu *Microsoft Visual Studio* 2010.

Sebagai produk lingkungan penembangan terintegrasi atau IDE andalan yang dikeluarkan oleh *Microsoft*.

Visual Studio merupakan bahasa 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 Expres.* (Wahana Komputer ; 2010 : 2)

II.6.1. Antar Muka Visual Basic 2010

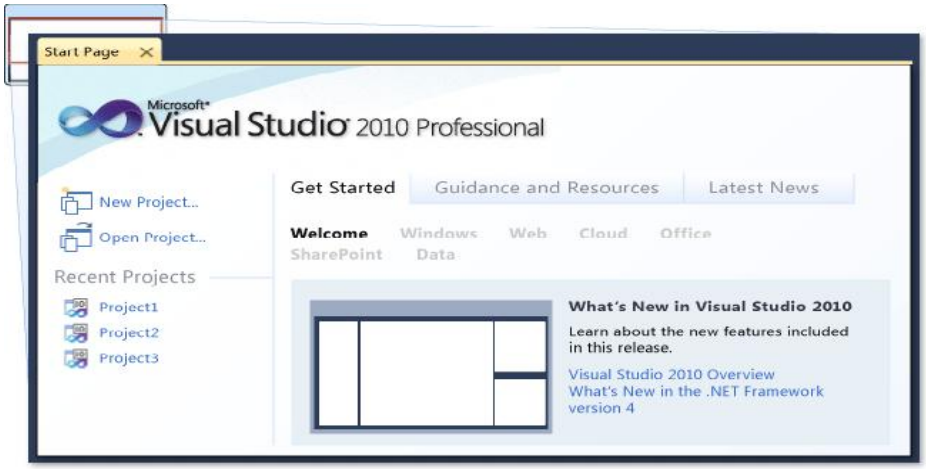
Saat menjalankan *Visual Basic* 2010 pertama kali muncul jendela *choose default environtment settings.* Disini bisa memilih apakah ingin memilih antar muka di *Visual Studio*. Untuk *programmer Visual Basic* lebih baik memilih *Visual Basic Development Centre.*



Gambar II.1 Form Chose Default Environtment Settings

(Sumber : Wahana Komputer ; 2010 ; 11)

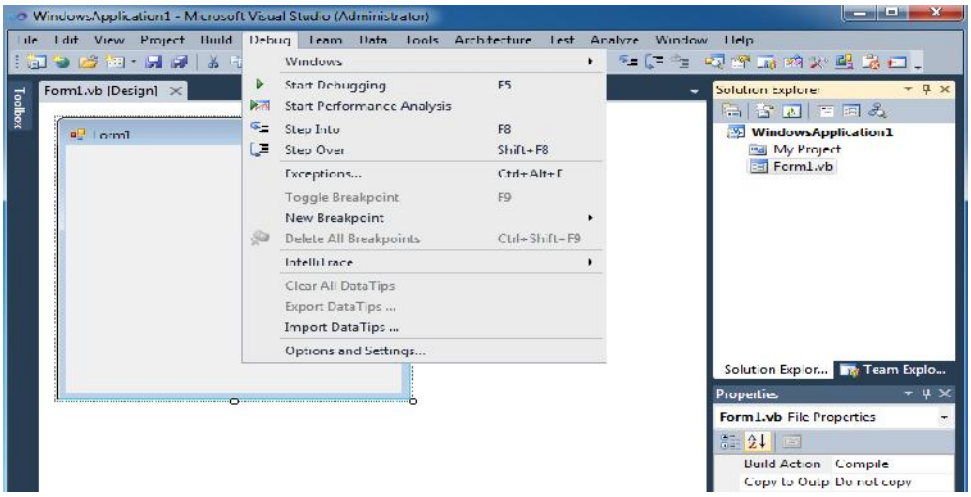
Dibagian awal *visual basic*, bisa memilih *Start Page*. *Start Page* adalah halaman yang mencantumkan informasi-informasi seputar program dan juga informasi RSS dari sumber tertentu. Jika tidak ingin menampilkan hal ini hilangkan tanda centang pada *Show Page On Startup.*



Gambar II.2 Start Page Visual Basic 2010

(Sumber : Wahana Komputer ; 2010 ; 12)

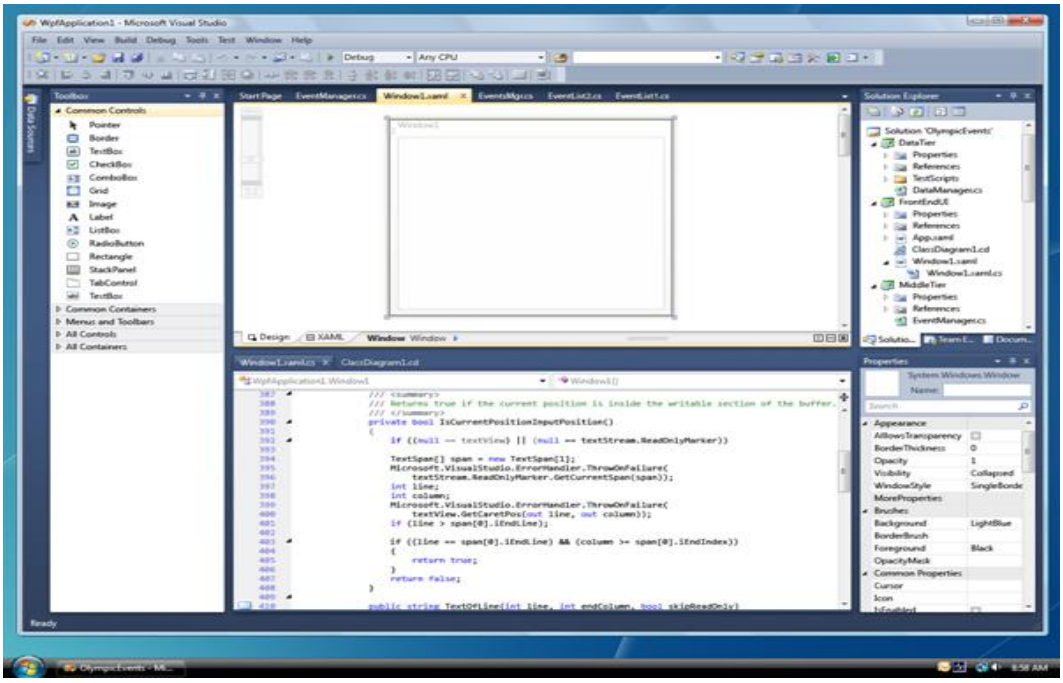
Jika *start page* ditutup terlihat tampilan sebagai berikut :



Gambar II.3 Tampilan IDE (*Integrated Development Environtment)* setelah *Start Page* ditutup

(Sumber : Wahana Komputer ; 2010 ; 13)

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



Gambar II.4 Tampilan lengkap IDE

(Sumber : Wahana Komputer ; 2010 ; 14)

Komponen-komponen dari IDE adalah :

1. Dibagian kiri terdapat toolbox yang menampilkan semua objek tool yang bisa dimasukkan kedalam form untuk membuat program.
2. Dibagian tengah terdapat tempat meletakkan form dan kode, baik disaat desain ataupun pada saat program dijalankan.

3. Dibagian kanan terdapat solution explorer yang merupakan explorer untuk melihat file-file disebuah objek.
4. Dikanan bawah terdapat propertis untuk melihat properti dari nilai-nilai pada objek yang dipilih dibagian tengah.

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

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, sistem 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. (Yuni Sugiarti ; 2013 : 34)

II.7.1.Pengenalan UML

UML itu adalah salah bentuk language atau bahasa. UML didefinisikan sebagai bahasa 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 bahasa 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 sistem 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. (Yuni Sugiarti ; 2013 : 36)

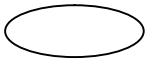
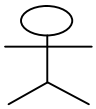
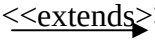
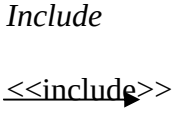
Adapun jenis-jenis dari tipe diagram *UML* adalah sebagai berikut :

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 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*. (Rossa A.S, M.Shalaluddin ; 2013 ; 155)

Tabel II.1. Simbol-simbol *Use Case Diagram*

Simbol	Deskripsi
Use Case 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan menggunakan kata kerja diawal frase nama *use case* .
Actor 	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 actor adalah gambar orang, tapi biasanya dinyatakan menggunakan kata benda diawal frase nama aktor.
Association	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor .
Extend 	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu mirip dengan prinsip inheritance pada pemrograman berorientasi objek.
Include 	Menjelaskan bahwa *use case* termasuk dalam *use case* lain.

Sumber : Rosa.A.S, M,Shalaluddin ; 2013 ; 156-157

2. *ClassDiagram*

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 properti dengan sebaris teks didalam kotak kelas tersebut.
c. Operasi atau metode adalah 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.

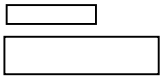
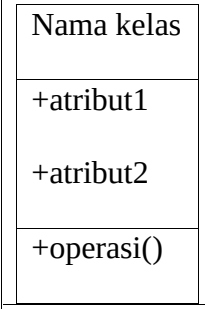
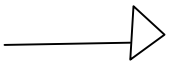
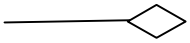
Kelas memiliki tiga area pokok :

a. Nama (dan *stereotype*)
b. Atribut
c. Metoda

Contoh kelas Manusia :

- Atribut : nama ,usia, tanggal lahir.
- Method/operasi :berjalan, makan, minum. (Yuni Sugiarti ; 2013 ; 58)

Tabel II.2. Simbol-simbol *Class Diagram*

Simbol	Deskripsi
Package 	Package merupakan sebuah bungkusan dari satu atau lebih kelas
Operasi 	Kelas pada struktur sistem.
Interface/Antarmuka 	Sama dengan konsepinterface dalam pemrograman berorientasi objek.
Asosiasi	Relasi antar kelas dengan makna umum asosiasi biasanya juga disertai dengan *multiplicity*
Generalisasi 	Relasi antar kelas dengan makna generalisasi-spesialisasi(umum khusus)
Agregasi 	Relasi antar kelas dengan makna semua-bagian (*whole-part*)

Sumber : Yuni Sugiarti ; 2013 ; 59

3. *Statechart Diagram*

Statechart diagram atau State machine diagram menggambarkan perubahan status atau transisi status dari sebuah mesin atau sistem atau objek. *State machine diagram* merupakan pengembangan dari diagram *Finite State Automata* dengan penambahan beberapa fitur dan konsep baru. *Statechart*

diagram cocok digunakan untuk menggambarkan alur interaksi pengguna dengan sistem. (Rosa A.S, M.Shalaluddin ; 2013 ; 163)

4. *Activity Diagram*

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

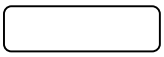
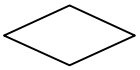
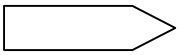
Diagram aktifitas juga banyak digunakan untuk mendefinisikan hal-hal berikut:

- Rancangan proses bisnis dimana setiap urutan aktifitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
- Urutan atau pengelompokkan tampilan dari sistem/*user interface* dimana setiap aktifitas dianggap memiliki sebuah rancangan antarmuka tampilan.
- Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujiannya.

Activity Diagram menggambarkan berbagai alir aktifitas dalam sistem yang sedang dirancang, bagaimana masing-masing berawal, decision yang mungkin terjadi, dan bagaimana proses paralel yang mungkin terjadi pada beberapa eksekusi. (Yuni Sugiarti ; 2013 ; 80)

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

Simbol	Deskripsi
	Status awal aktivitas sistem,sebuah diagram aktivitas memiliki sebuah status awal.

	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah satus akhir.
	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
	Percabangan (*decision*) dimana jika ada pilihan aktivitas lebih dari satu.
	*Join* : kegiatan penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
	*Rake* ; menunjukkan adanya dekomposisi
	Tanda waktu
	Tanda pengiriman

Sumber : Rosa A.S, M.Shalaluddin ; 2013 ; 162

5. *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 di instansiasi menjadi objek itu. Membuat diagram sekuen juga dibutuhkan untuk melihat skenario yang ada pada *use case*.

6. *Collaboration Diagram*

Collaboration diagram juga menggambarkan interaksi antar objek seperti *sequence diagram* , tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian *message.*

7. *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*. (Rosa A.S, M.Shalaluddin ; 2013 ; 165-169)