BAB II

TINJAUAN PUSTAKA

II.1. Pengertian Perancangan

Perancangan adalah disiplin manajerial dan teknis yang berkaitan dengan pembuatan dan pemeliharaan produk perangkat lunak secara sistematis, termasuk pengembangan dan modifikasinya, yang dilakukan pada waktu yang tepat dan dengan mempertimbangkan faktor biaya. (Fauzi, H.D, 2007)

II.2. Pengertian Perangkat Lunak

Perangkat lunak adalah istilah umum untuk data yang di format dan disimpan secara *digital*, termasuk program komputer, dokumentasinya, dan berbagai informasi yang bisa dibaca dan ditulis oleh *computer*. Dengan kata lain, bagian sistem *computer* yang tidak berwujud. Istilah ini menonjolkan perbedaan dengan perangkat keras *computer*. (Fauzi, H.D, 2007)

II.3. Pengertian *Input*

Input adalah semua data dan perintah yang dimasukkan ke dalam *memory computer*. Sebuah perangkat *input* adalah komponen piranti keras yang memungkinkan pengguna memasukkan data ke dalam *computer* atau bisa juga disebut unit luar yang digunakan untuk memasukkan data dari luar ke dalam *mikroprosesor*. Misalnya data yang berasal dari *keyboard* dan *mouse*.

(Arief, Julianto , 2008)

II.4. Pengertian Data

Menurut Arief, Julianto (2008) data adalah bentuk jamak dari *datum*, berasal dari bahasa Latin yang berarti "sesuatu yang diberikan". Dalam penggunaan sehari-hari data berarti suatu pernyataan yang diterima secara apa adanya. Pernyataan ini adalah hasil pengukuran atau pengamatan suatu variabel yang bentuknya dapat berupa angka, kata-kata, atau citra.

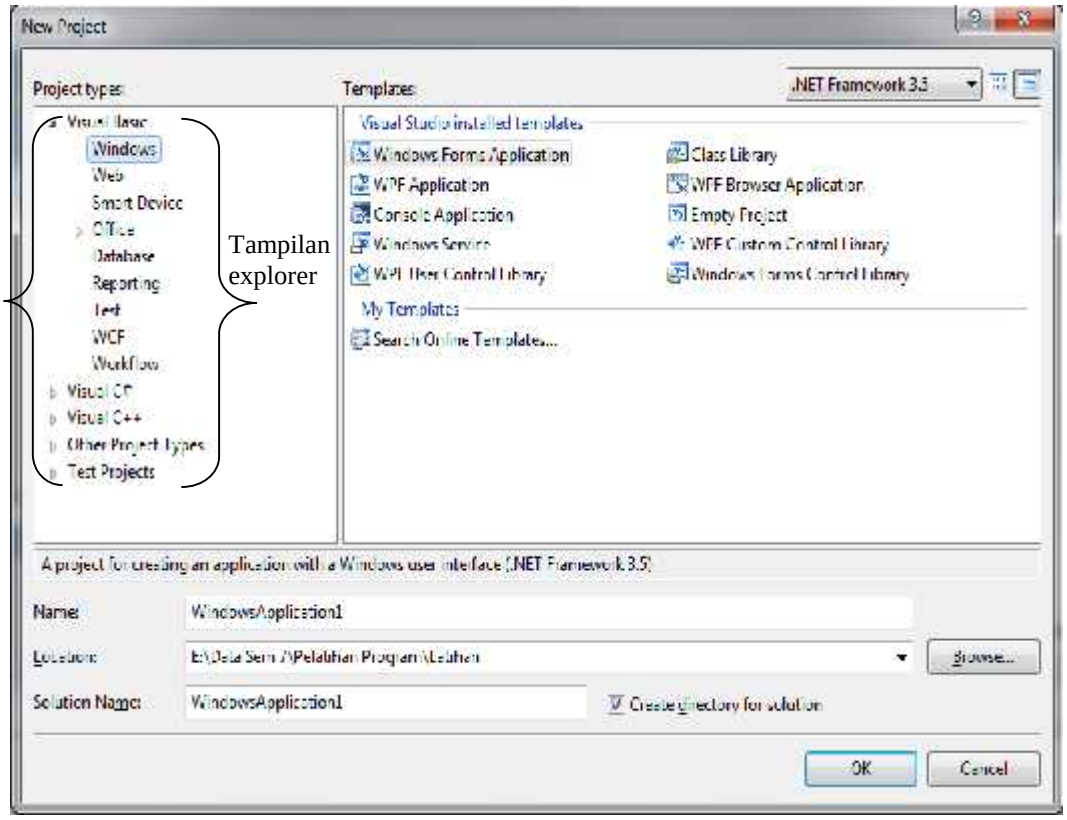
Data juga merupakan informasi yang disimpan yang dapat sewaktu - waktu di gunakan oleh penggunannya. Sumber dari informasi adalah data. Data merupakan bentuk jamak dari bentuk tunggal data-item. Data merupakan bentuk yang belum dapat memberikan manfaat yang besar bagi penerimanya, sehingga perlu suatu model yang nantinya akan dikelompokkan dan diproses untuk menghasilkan informasi.

Dengan demikian yang dimaksud dengan data adalah segala sesuatu yang menggambarkan suatu fakta dari kejadian yang merupakan bahan yang dapat diolah untuk menjadi suatu informasi. Dalam konteks penyimpanan data sistem dapat diartikan sebagai suatu kumpulan dari data-data yang ada sebagai elemen satu dan segala operasi yang terkait dengan data tersebut sebagai elemen yang lain, untuk mencapai suatu tujuan yaitu bagaimana data dapat disimpan dengan baik dan bagaimana data dapat diakses dengan baik.

II.5. Pengenalan *Visual Basic. NET* 2008

Visual Basic.Net dilihat dari sejarahnya merupakan pengembangan dari bahasa *Basic*, sehingga aturan penulisan bahasanya pun sama dengan bahasa *Basic*. Akan tetapi, oleh karena adanya tuntutan perkembangan teknologi maka

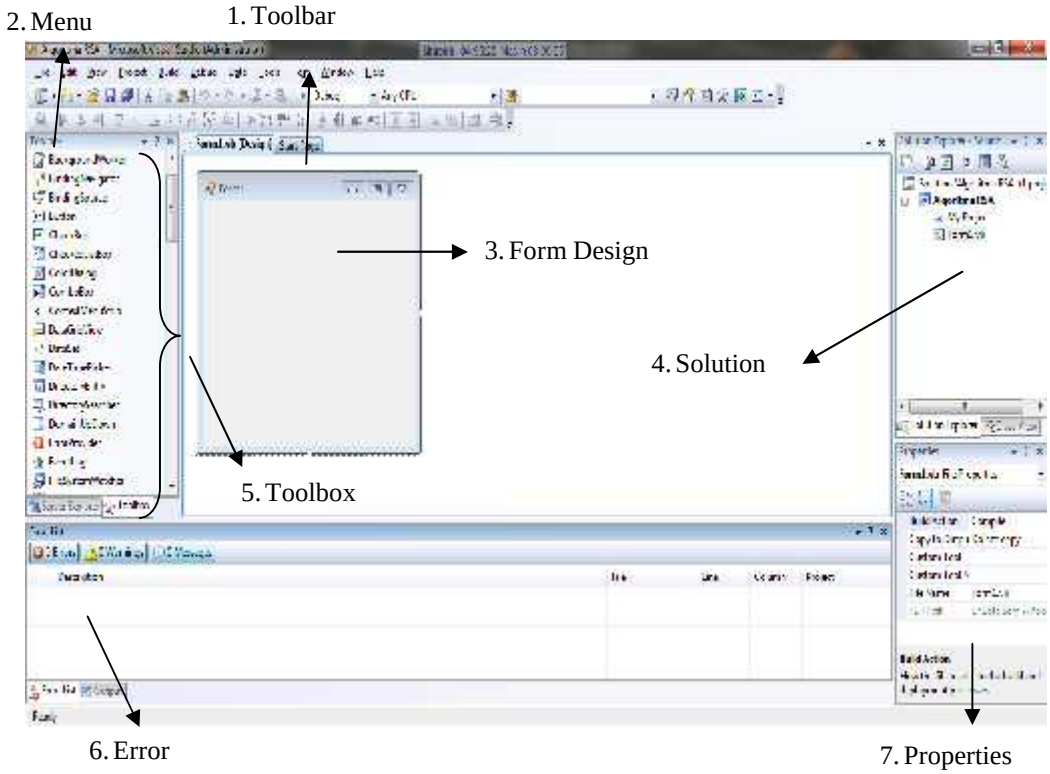
bahasa *Visual Basic.Net* memiliki beberapa tambahan yang tidak ada di bahasa *Basic* aslinya (Wahana Komputer, 2006 ; hal 1).



Gambar II.1. Jendela *New Project VB 2008*
Sumber (Muhammad Sadeli ; 2009 ; 5)

1. ***Windows*** adalah *project* yang sering digunakan untuk membangun aplikasi-aplikasi *dekstop* seperti (membuat aplikasi *dekstop*, komponen *ActiveX*, *file* DLL, dan lain sebagainya). Karena menggunakan *interface windows* baik *command line* ataupun *windows form* yang memiliki *form* dan kontrol, yang terbaru di *VB 2008* adalah WPF (*Windows Presentation Foundatio*n (*Windows, Web*) XAML) yang memugkinkan suatu pekerjaan GUI (*Grafik User Interface*) dan kode untuk program dibuat secara terpisah.

2. ***Web*** adalah *project* yang dapat digunakan untuk membuat aplikasi berbasis *web* menggunakan *ASP.Net 3.5*.
3. ***Smart Device*** digunakan untuk membuat perangkat aplikasi pemograman yang dapat diimplimentasikan pada *Handphone* tertentu seperti PDA (*Personal Digital Assistant*). Pemograman *Smart Device* pada *VB 2008* menggunakan *.NET Compact Framework* 3.5 dan berjalan diatas sistem *Windows CE* (*Compact Edition*).
4. ***Office*** adalah suatu *project* yang dapat menyediakan atau menjalankan atau memanggil aplikasi yang terdapat pada program *Office* (2003, 2007) seperti *Excel, Word, Power Point,* dan lain sebagainya.



Gambar II.2. Halaman Kerja *Visual Basic* 2008
Sumber (Muhammad Sadeli ; 2009 ; 7)

1. ***Menu Bar*** adalah suatu menu yang terdiri dari 11 menu utama, masing-masing memiliki *sub menu* dan perintah lengkap dengan *shortcut key*.
2. ***Toolbar Standart*** adalah suatu baris menu yang mempunyai fungsi yang sama pada setiap *Tool Standart* pada umumnya, seperti fungsi untuk menyimpan, meng-*copy*, menambah *project* baru, mengatur tampilan program dan masih banyak lagi.
3. ***Form Design*** adalah suatu lembar *form* yang berfungsi untuk merancang tampilan aplikasi secara visual dengan menempatkan kontrol-kontrol yang diperlukan.
4. ***Toolbox*** adalah suatu jendela yang berfungsi untuk menampung komponen-komponen *standard*.
5. ***Solution Explorer*** adalah suatu jendela yang berfungsi untuk menampilkan *object* yang digunakan untuk membuat aplikasi seperti : *form*, *class*, dan *object* lainnya.
6. ***Properties Windows*** adalah suatu jendela yang berfungsi untuk mengatur nilai *properties* dari masing-masing komponen yang akan digunakan.
7. ***Error List*** adalah suatu jendela yang berfungsi untuk menampilkan setiap kesalahan dari pembuatan kode program suatu aplikasi. (Muhammad Sadeli ; 2009 ; 7 - 8)

II.6. SQL Server

SQL Server adalah satu jenis database relasional yang mendukung aplikasi dengan arsitektur *client/server (two tier)* yang mana sebagian proses dilakukan oleh *server* dan sebagian lagi dilakukan oleh aplikasi sehingga dapat mengurangi

lalu lintas jaringan, karena *SQL Server* hanya memberikan data yang diperlakukan oleh *client* saja. (Chan Syahrial, 2005).

II.7. SQL Server 2005

SQL Server 2005 merupakan penyempurnaan dari *SQL Server 2000* dan ditambah fitur-fitur baru. *SQL Server 2005* memperluas kinerja, keandalan, ketersediaan, program mobilitas dan mudah dalam penggunaanya. *SQL Server 2005* meliputi beberapa fitur baru yang membuatnya menjadi suatu *flatrom database* yang sempurna untuk memproses transaksi *database* berskala besar dan aplikasi *e-commerce.* (Chan Syahrial, 2005).

II.8. Jaringan Client Server

Pengertian *Client Server* menurut E. Setiawan (2003) mengemukakan bahwa ”Client Server adalah arsitektur jaringan yang memisahkan client(biasanya aplikasi yang menggunakan *GUI*) dengan *server*”.

Sedangkan menurut Reddick dan King (2001) ”*Client Server* adalah sistem terdistribusi dengan beberapa karakteristik yaitu *servis* (layanan), *Sharing resources* (sumber daya), *Asymmetrical protokol* (protokol yang tidak simetris), *Transparansi (*kejelasan*), Mix and Match* (Pencampuran dan perbandingan)”

Sistem Pengecekan adalah sistem berbasis komputer yang menggunakan pengetahuan, fakta, dan teknik penalaran dalam mencegah masalah yang biasanya hanya dapat dilakukan oleh satu komputer dalam pengecekan hasil kelulusan tersebut (Martin dan Oxman, 1988).

Sedangkan menurut Frans Newman *Client-Server* adalah salah satu model komunikasi 2 komputer atau lebih yang berfungsi melakukan pembagian tugas,

Client bertugas untuk melakukan *input, update*, penghapusan, dan menampilkan data sebuah *database*. Sementara *Server* bertugas menyediakan pelayanan untuk melakukan manajemen, yaitu menyimpan dan mengolah *database*. (Newman Frans, 2001).

II.9. UML (*Unifield Modelling Language*)

UML (*Unified Modelling Language*) adalah salah satu alat bantu yang sangat handal di dunia perkembangan sistem yang berorientasi objek. Hal ini disebabkan karena UML menyediakan bahasa pemodelan visual yang memungkinkan bagi perkembangan sistem untuk membuat cetak biru atas visi mereka dalam bentuk yang baku, mudah dimengerti serta dilengkapi dengan mekanisme yang efektif untuk berbagi (*sharing*) dan mengkomunikasikan rancangan dengan baik (Munawar ; 2005 : 17).

UML merupakan kesatuan bahasa pemodelan yang dikembangkan oleh *Booch, Object Modeling Technique* (OMT) dan *Object Oriented Engineering* (OOSE). Metode *Booch* dari *Grady Booch* sangat terkenal dengan nama metode *Design Object Oriented*. Metode ini menjadikan proses analisis dan design ke dalam empat tahapan interatif, yaitu: identifikasi kelas-kelas dan objek-objek, identifikasi semantik dari hubungan objek dan kelas tersebut, perincian *interface* dan implementasi. Keunggulan metode *Booch* adalah pada detail dan kayanya dengan notasi dan elemen. Pemodelan OMT yang dikembangkan oleh Rumbaugh didasarkan pada analisis terstuktur dan pemodelan *entity-relationship*. Tahapan utama dalam metodologi ini adalah analisis, disain sistem, desain objek dan implementasi. Keunggulan metode ini adalah dalam penotasian yang mendukung

semua konsep OO. Metode OOSE dari Jacobson lebih memberi penekanan dan *use case*. OOSE memiliki tiga tahapan yaitu membuat model *requirement* dan analisis, desain dan implementasi dan model pengujian (test Model). Keunggulan metode ini adalah mudah dipelajari karena memiliki notaasi sederhana namun mencakup seluruh tahapan dalam rekayasa perangkat lunak.

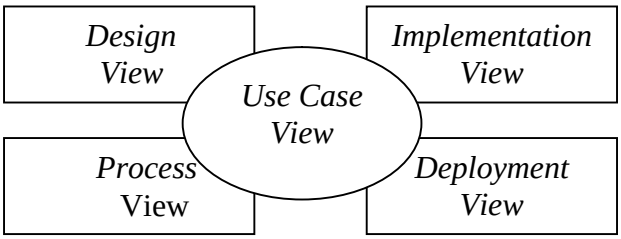
Dengan UML, metode *Booch*, OMT dan OOSE digabungkan dengan elemen-elemen dari metode lain yang lebih efektif dan elemen-elemen baru yang belum ada pada metode terdahulu sehingga UML lebih ekspresif dan seragam dari pada metode lainnya. Unsur-unsur yang membentuk UML ditunjukkan dalam Gambar II.3.

Gambar II.3. Unsur-unsur yang membentuk UML

Sumber : Pemodelan Visual dengan UML, Munawar, 2005 : 18

UML adalah hasil kerja dari konsorsium berbagai organisasi yang berhasil Pdijadikan sebagai standar baku dalam OOAD (*Object Oriented Analysis* dan *Design*). UML tidak hanya domain dalam penotasian dilingkungan OO tetapi juga populer di luar lingkungan OO. Ada tiga karakter penting yang melekat di UML yaitu sketsa, cetak biru dan bahasa *pemrograman*. Sebagai sebuah sketsa UML bisa berfungsi sebagai sebuah cetak biru karena sangat lengkap dan detil. Dengan cetak biru ini maka akan bisa diketahui informasi detil tentang coding program (*Forward engineering*) atau bahkan membaca program dan mengimplementasikannya kembali ke dalam *diagram* (*reverse engineering*). *Reverse engineering* sangat berguna pada situasi dimana kode program yang tidak terdokkumentasi asli hilang atau bahkan belum perna dibuat sama sekali. Sebagai bahasa pemrograman, UML dapat diterjemahkan diagram yang ada di UML menjadi kode program siap untuk dijalankan.

UML dibangun atas model 4+1 *view*. Model ini didasarkan pada fakta bahhwa struktur sebuah sistem dideskripsikan dalam *view* dimana salah satu diantaranya *use case view*. *Use case view* ini memegang peran khusus untuk mengintegrasikan *content* ke *view* yang lain. Model 4+1 *view* ditunjukkan pada gambar II.4.



Gambar II.4. Model 4+1 *View*

Sumber : Pemodelan Visual dengan UML, Munawar, 2005 : 20

Kelima *view* tersebut tidak berhubungan dengan *diagram* yang dideskripsikan di UML. Setiap *view* berhubungan dengan perspektif tertentu dimana sistem akan diuji. *View* yang berbeda akan menekankan pada aspek yang berbeda dari sistem yang mewakili tentang sistem bisa dibentuk dengan menggabungkan informasi-informasi yang ada pada kelima *view* tersebut.

Use case view mendefinisikan perilaku eksternal sistem. Hal ini menjadi daya tarik bagi *end user*, analis dan *tester*. Pandangan ini mendefenisikan kebutuhan sistem karena mengandung semua *view* yang lain yang mendeskripsikan aspek-aspek tertentu dari peran dan sering dikatakan yang mendrive proses perkembangan perangkat lunak.

Design view mendeskripsikan struktur logika yang mendukung fungsi-fungsi yang dibutuhkan di *use case*. *Design view* ini berisi definisi komponen program, *class-class* utama bersama-sama dengan spesifikasi *data*, perilaku dan interaksinya. Informasi yang terkandung di *view* menjadi pergantian para progremer karena menjelaskan secara detil bagaimana fungsionalitas *sistem* akan diimplementasikan.

Implementasi *view* menjelaskan komponen-komponen *visi* yang akan dibangun. Hal ini berbeda dengan komponen *logic* yang dideskripsikan pada *design view*. Termasuk disini diantaranya *file exe*, *library* dan *database*. Informasi yang ada di *view* dan integrasi sistem.

Proses *view* berhubungan dengan hal-hal yang berkaitan dengan *concurrency do* dalam sistem. Sedangkan *deployment view* menjelaskan bagaimana komponen-komponen fisik didistribusikan ke lingkungan fisik seperti

jaringan komputer dimana sistem akan dijalankan. Kedua *view* ini menunjukan kebutuhan *non fungsional* dari sistem seperti toleransi kesalahan dan hal-hal yang berhubungan dengan kinerja. (Munawar ; 2005 : 17-21).

II.9.1. *Use Case Diagram*

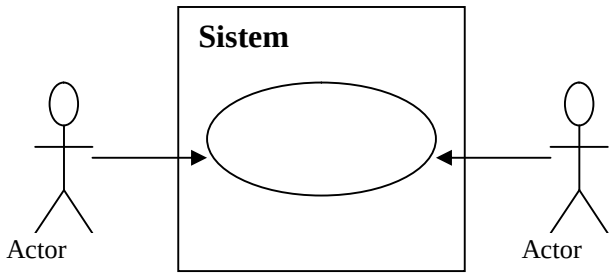
Use case adalah deskripsi fungsi dari sebuah sistem dari perspektif pengguna. *Use case* bekerja dengan cara deskripsikan tipikal interaksi antara *user* (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Urutan langkah-langkah yang menerangkan antara pengguna dan sistem yang disebut *scenario*. Setiap *scenario* mendeskripsikan urutan kejadian. Setiap urutan diinisialisasi oleh orang, sistem yang lain, perangkat keras dan urutan waktu. Dengan demikian secara singkat bisa dikatakan *use case* adalah serangkaian *scenario* yang digabungkan bersama-sama oleh pengguna tujuan umum pengguna.

Dalam pembicaraan tentang *use case*, pengguna biasanya disebut dengan *actor*. *Actor* adalah sebuah peran yang bisa dimainkan oleh pengguna dalam interaksinya dengan sistem.

Model *use case* adalah bagai dari model *requirement*. Termasuk disini adalah problem domain *object* dan penjelasan tentang *user interface*. *Use case* memberikan spesifikasi fungsi-fungsi yang ditawarkan oleh sistem dari *persfectif user*.

Notasi *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*. Ilustrasi *actor, use case* dan *system* ditunjukkan pada gambar II.5.



Gambar II.5. *Use Case* Diagram

Sumber : Pemodelan Visual dengan UML, Munawar, 2005 : 64

Untuk mengidentifikasi *actor*, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks targer sistem. *Actor* adalah *abstraction* dari orang dan sistem yang lain mengaktifkan fungsi dari target sistem. Orang atau sistem bila muncul dalam beberapa peran. Perlu dicatat bahwa *actor* berinteraksi dengan *use case*, tetapi tidak memiliki kontrol atas *use case*.

Use case adalah abstraksi dari interaksi antara sistem dan *actor*. Oleh kerena itu sangat penting untuk memilih abstraksi yang cocok. Use case dibuat berdasarkan keperluan *actor*. *Use case* harus merupakan ’apa’ yang dikerjakan *software* aplikasi, bukan ’bagaimana’ *software* aplikasi mengerjakannya. Setiap *use case* harus diberi nama yang menyatakan apa hal yang dicapai dari hasil interaksinya dengan *actor*. Namun *use case* boleh terdiri dari beberapa kata dan tidak boleh ada dua *use case* yang memiliki nama yang sama (Munawar ; 2005 : 63-66).

II.9.2. *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

Susunan kelas suatu sistem yang baik pada diagram kelas sebaiknya memiliki jenis-jenis kelas berikut:

1. Kelas main, kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.
2. Kelas yang menangani tampilan sistem, kelas yang mendefinisikan dan mengatur tampilan ke pemakai.
3. Kelas yang diambil dari pendefinisian *use case*, kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisian *use case*.
4. Kelas yang diambil dari pendefinisian data, kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data.

Jenis-jenis kelas diatas juga dapat digabungkan satu sama lain sesuai dengan pertimbangan yang dianggap baik asalkan fungsi-fungsi yang sebaiknya ada pada struktur kelas tetap ada. Susunan kelas juga dapat ditambahkan kelas utilitas seperti koneksi ke *basis data*, membaca *file* teks, dan lain sebagainya sesuai kebutuhan.

Dalam mendefinisikan metode yang ada di dalam kelas perlu memperhatikan apa yang disebut dengan *cohension* dan *coupling*. *Cohension* adalah ukuran seberapa dekat keterkaitan instruksi di dalam sebuah metode terkait satu sama lain sedangkan *coupling* adalah ukuran seberapa dekat keterkaitan instruksi antara metode yang satu dengan yang lain dalam sebuah kelas. Sebagai aturan secara umum maka sebuah metode yang dibuat harus memiliki kadar *cohesion* yang kuat dan kadar *coupling* yang lemah (Rosa A.S dan M. Shalahuddin ; 2011 : 122-123).

II.9.3 *Activity Diagram*

Activity diagram adalah teknik untuk mendeskripsikan logika prosedural, proses bisnis dan aliran kerja dalam banyak kasus. *Activity diagram* mempunyai peran seperti halnya *flowchart*, akan tetapi perbedaanya dengan *flowchart* adalah *activity diagram* bisa mendukung perilaku paralel sedangkan *flowchart* tidak bisa (Munawar ; 2005 : 87).