

## **BAB II**

### **LANDASAN TEORI**

Untuk mendukung sebuah pelaksanaan penelitian landasan teori merupakan dasar teori yang akan membuktikan bahwa penelitian tersebut berkualitas, memiliki dasar pengetahuan yang dapat dipertanggung jawabkan untuk melanjutkan penelitian sebelumnya, memperbaiki atau dapat juga mematahkan teori-teori sebelumnya.

#### **2.1 Definisi Website**

*Website* atau situs dapat diartikan sebagai kumpulan halaman yang menampilkan informasi data teks, data gambar diam atau bergerak, data animasi, suara, *video* dan atau gabungan dari semuanya, baik yang bersifat statis maupun dinamis yang membentuk satu rangkaian bangunan yang saling terkait dimana masing – masing dihubungkan dengan jaring – jaring halaman (*hyperlink*). Bersifat statis apabila isi informasi *website* tetap, jarang berubah, dan isi informasinya searah hanya dari pemilik *website*. Bersifat dinamis apabila isi informasi *website* selalu berubah – ubah, dan isi informasinya interaktif dua arah berasal dari pemilik serta pengguna *website* (*Jurnal Algoritma, Vol 9, No 40, Bekasi, 2012*).

Aplikasi web adalah sebuah sistem informasi yang mendukung interaksi pengguna melalui antarmuka berbasis web. Fitur-fitur aplikasi web biasanya berupa data persistence, mendukung transaksi dan komposisi halaman web dinamis yang dapat dipertimbangkan hibridisasi, antara hipermedia dan sistem informasi. Menurut Chan aplikasi web adalah aplikasi yang mempunyai arsitektur *three-tier* yang dijalankan pada *browser* sebagai *front end* (*Jurnal Algoritma, Vol 9, No 2, Jakarta, 2016*).

## 2.2 Definisi Informasi

Informasi adalah data yang diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi penerimanya (Hutahaean, 2014).

## 2.3 Definisi Sistem Informasi

Dalam praktik, istilah sistem informasi lebih sering dipakai tanpa embel-embel berbasis komputer walaupun dalam kenyataannya komputer merupakan bagian yang penting (Abdul Kadir, 2014).

## 2.4 Definisi Basis Data

Basis Data adalah sistem terkomputerisasi yang tujuan utamanya adalah memelihara data yang sudah diolah atau informasi dan membuat informasi tersedia saat dibutuhkan. Pada intinya basis data adalah media untuk menyimpan data agar dapat diakses dengan mudah dan cepat. Sistem informasi tidak dapat dipisahkan dengan kebutuhan akan basis data apapun bentuknya, entah *file text* ataupun *Database Management System (DBMS)* Kebutuhan basis data dalam sistem informasi adalah untuk memasukkan, menyimpan dan mengambil data, selain itu untuk membuat laporan berdasarkan data yang telah disimpan (Rosa A.S M. Shalahudin, 2014).








pemrosesan basis data terdistribusi adalah pemrosesan basis data dimana pelaksanaan transaksi, pengambilan dan pembaharuan data yang terjadi melewati dua atau lebih komputer yang biasanya terpisah secara geografis dan tidak saling terkait (Kroenke, 1995).

Basis data adalah suatu pengorganisasian sekumpulan data yang saling terkait sehingga memudahkan aktivitas untuk memperoleh informasi. Basis data dimaksudkan untuk mengatasi problem pada sistem yang memakai pendekatan berbasis berkas (Abdul Kadir, 2014).

## 2.5 Definisi *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. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas, sedangkan operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas (Rosa A.S M. Shalahudin, 2014).

**Tabel 2.1 Simbol *Class Diagram*.**

NO	GAMBAR	NAMA	KETERANGAN
1		<i>Generalization</i>	Hubungan dimana objek anak ( <i>descendent</i> ) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk ( <i>ancestor</i> ).
2		<i>N-Ary Association</i>	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
3		<i>Class</i>	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
4		<i>Collaboration</i>	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu actor.
5		<i>Realization</i>	Operasi yang benar-benar dilakukan oleh suatu objek.
6		<i>Dependency</i>	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri ( <i>independent</i> ) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri.
7		<i>Association</i>	Apa yang menghubungkan antara objek satu dengan objek lainnya.

## 2.6 Definisi *Object Oriented Programming* (OOP)

*Object Oriented Programming* (OOP) merupakan model pemrograman yang berbasis pada konsep obyek, diantaranya berisi data, sering dikenal sebagai atribut dan kode, dalam bentuk prosedur, sering dikenal sebagai metode. Sebuah fitur dari obyek adalah bahwa prosedur obyek dapat mengakses dan sering memodifikasi data dari obyek yang saling berhubungan. Dalam OOP, program dirancang dengan membuat obyek yang dapat berinteraksi satu sama lain (Kindler, E.; Krivy, I. (2011)).










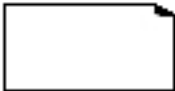
Pemrograman berorientasi obyek didefinisikan menggunakan objek, tetapi tidak semua teknik terkait dan struktur dalam bahasa pemrograman mendukung OOP (Michael Lee Scott, *Programming language pragmatics, Edition 2, Morgan Kaufmann* (2006)).

Kadang-kadang objek mewakili entitas yang lebih abstrak, seperti objek yang mewakili file terbuka, atau benda yang menyediakan layanan menerjemahkan pengukuran. Bahasa yang mendukung *class* hampir selalu mendukung *Inheritance*. Hal ini memungkinkan *Class* yang akan diatur dalam hirarki yang mewakili hubungan. Misalnya, *Class* karyawan mungkin mewarisi dari *Class* Person. Semua data dan metode yang tersedia untuk *Class* induk juga muncul di *Class* anak dengan nama yang sama. Misalnya, *Class* Orang mungkin mendefinisikan variabel *first\_name* dan *last\_name* dengan metode *make\_full\_name* (). Ini juga akan tersedia dalam *Class* karyawan, yang bisa menambahkan variabel posisi dan gaji. Teknik ini memungkinkan mudah digunakan kembali prosedur dan data yang sama definisi, selain berpotensi *mirroring* hubungan dunia nyata dengan cara yang intuitif. *Class-class* dan *subclass* sesuai dengan *set* dan *subset* di logika matematika. Daripada menggunakan tabel *database* dan pemrograman subrutin, pengembang memanfaatkan objek, pengguna mungkin lebih akrab dengan Objek dari domain aplikasi merek (Jacobsen, Ivar; Magnus Christerson; Patrik Jonsson; Gunnar Overgaard (1992)).

## **2.7 Definisi Use Case Diagram**

*Use case* atau *use case* diagram merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut aktor dan *use case*. 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. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor (Rosa dan Shalahuddin, 2013).

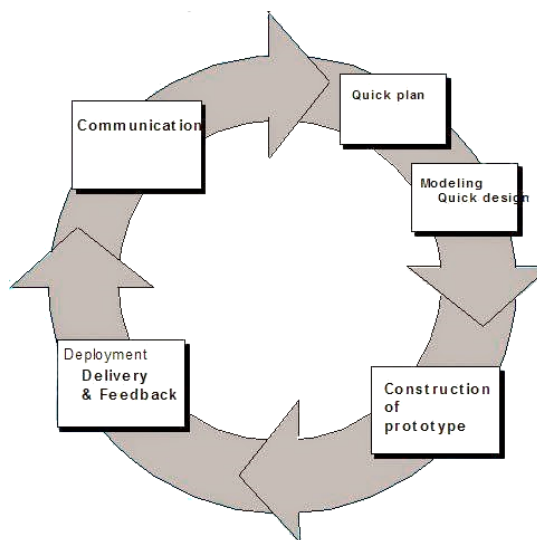
Tabel 2.2 Simbol-Simbol *UseCase*.

No	Gambar	Nama	Keterangan
1		<i>Actor</i>	Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan <i>use case</i> .
2		<i>Dependency</i>	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri ( <i>independent</i> ) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri ( <i>independent</i> ).
3		<i>Generalization</i>	Hubungan dimana objek anak ( <i>descendent</i> ) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk ( <i>ancestor</i> ).
4		<i>Includes</i>	Menspesifikasikan bahwa <i>use case</i> sumber secara eksplisit.
5		<i>Extend</i>	Menspesifikasikan bahwa <i>use case</i> target memperluas perilaku dari <i>use case</i> sumber pada suatu titik yang diberikan.
6		<i>Association</i>	Apa yang menghubungkan antara objek satu dengan objek lainnya.
7		<i>System</i>	Menspesifikasikan paket yang menampilkan system secara terbatas.
8		<i>Use Case</i>	Deskripsi dari urutan aksi-aksi yang ditampilkan suatu hasil yang terukur bagi suatu <i>actor</i> .
9		<i>Collaboration</i>	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemennya ( <i>sinergi</i> ).
10		<i>Note</i>	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.

## 2.8 Definisi Model Prototipe (*Prototyping*)

Sebuah prototipe (*Prototyping*) adalah versi awal dari sistem perangkat lunak yang digunakan untuk mendemonstrasikan konsep-konsep, percobaan rancangan, dan menemukan lebih banyak masalah dan solusi yang memungkinkan (*Sommerville, 2011*).

Sering kali pelanggan mendefinisikan sejumlah sasaran perangkat lunak secara umum, tetapi tidak bias mengidentifikasi spesifikasi kebutuhan yang rinci untuk fungsi – fungsi dan fitur – fitur yang nantinya akan dimiliki perangkat lunak yang akan dikembangkan. Dalam kasus lain, pengembang perangkat lunak mungkin merasa tidak pasti tentang efisiensi suatu algoritma yang akan digunakan dalam pengembangan perangkat lunak, atau juga merasa tidak pasti akan kemampuan perangkat lunak untuk beradaptasi dengan sistem operasi yang akan digunakan. Dalam kasus – kasus seperti ini dan dalam banyak situasi lain, pradigma pembuatan prototipe (*Prototyping*) mungkin menawarkan pendekatan yang paling baik. Meskipun pembuatan prototipe dapat digunakan sebagai model proses yang berdiri sendiri, pembuatan prototipe lebih umum digunakan sebagai teknik yang dapat diimplementasikan di dalam konteks setiap model proses perangkat lunak.



**Gambar 2.1 Metode Prototipe**

Sumber : (*Roger S. Pressman, 2012*)

Pembuatan prototipe (*Prototyping*) dimulai dengan dilakukan komunikasi antara tim pengembang perangkat lunak dengan pelanggan (*stakeholder*) untuk mengidentifikasi spesifikasi kebutuhan perangkat lunak. Rancangan cepat (*quick design*) akan memulai konstruksi pembuatan prototipe (*Prototyping*) meliputi rancangan sistem dan rancangan antar muka. Prototipe (*Prototyping*) kemudian akan diserahkan kepada para *stakeholder* dan kemudian mereka akan melakukan evaluasi – evaluasi tertentu terhadap prototipe yang dibuat. Pembentukan prototipe (koding) akan dilakukan berdasarkan data – data yang diperoleh tim pengembang. Setelah selesai tim pengembang akan menyerahkan system/perangkat lunak kepada *stakeholder*/pelanggan yang kemudian akan memberikan umpan balik yang akan digunakan untuk memperluas spesifikasi kebutuhan. Iterasi akan terjadi saat prototipe diperbaiki untuk memenuhi kebutuhan dari para *stakeholder*, pada saat yang sama memungkinkan kita untuk lebih memahami kebutuhan apa yang akan dikerjakan pada iterasi selanjutnya (Roger S. Pressman, 2012).

## **2.9 Definisi Text Editor**

*Text Editor* adalah suatu *software* aplikasi atau suatu *program* komputer yang memungkinkan anda sebagai penggunanya untuk membuat, mengubah atau mengedit *file* teks yang ada berupa *plain text*. *Text editor* ini sebenarnya bisa digunakan untuk membuat *program-program* komputer dan mengedit *source code* dari bahasa pemrograman. Selain itu, *text editor* juga bisa dimanfaatkan untuk membuat halaman *web* atau *template web design* dan juga membuat aplikasi tertentu. *Software* aplikasi satu ini memang secara umum ditujukan untuk mempermudah aktivitas pemrograman.

*Text Editor* memiliki fitur-fitur yang sangat kecil dan sederhana. Namun ada juga beberapa *text editor* kini sudah menawarkan fungsi yang luas dan kompleks. Seperti *Unix* dan *Linux* adalah contohnya. Dalam sistem operasinya sudah tersedia Editor VI (atau varian), tapi banyak juga yang mencakup editor *Emacs*. Sementara sistem operasi dari *Windows* itu sendiri menyediakan *Notepad* standar. Walaupun



sudah tersedia secara bawaan, banyak programmer lebih menyukai *text editor* lainnya yang memang fiturnya lebih banyak atau lengkap (Yasha, 2018, <https://www.dewaweb.com/blog/panduan-text-editor-atom-sublime-notepad/>. Jam 13.20 dan tanggal akses 07 July 2019).

### **2.10 Definisi Bootstrap**

*Bootstrap* merupakan *framework* untuk membangun desain *web* secara responsif dan cepat. Artinya, tampilan *web* yang dibuat oleh *bootstrap* akan menyesuaikan ukuran layar dari *browser* yang kita gunakan baik di *desktop*, *tablet* ataupun *mobile device*. Sehingga, *user* akan mendapatkan pengalaman yang lebih baik dalam berselancar tanpa mempertimbangkan perangkat apa yang harus digunakan.

Sejatinya, apabila kita menggunakan *bootstrap*, kita tinggal menggunakan nama *class* (untuk *css*) dan *library* (*javascript*) yang sudah ditentukan oleh *bootstrap* tanpa perlu menulis kode dari 0 (awal) sehingga, bisa menghemat waktu dalam pengembangan *website* untuk urusan UI (*User Interface*). *Bootstrap* memiliki tampilan yang indah dan dapat di *customisasi*.

Dengan *bootstrap* kita juga bisa membangun *web* dinamis ataupun statis, tentunya harus didukung dengan teknologi lain dalam pengembangannya. (Fadlullah Fadul, <https://www.apacara.com/tutorial/bootstrap/belajar-bootstrap-untuk-pemula.html>. Jam 13.40 dan tanggal akses 07 July 2019).

### **2.11 Definisi XAMPP**

Xampp adalah perangkat yang menggabungkan tiga aplikasi kedalam satu paket yaitu Apache, MySQL, dan PHPMyAdmin. Dengan Xampp pekerjaan anda sangat dimudahkan karena dapat menginstalasi dan mengkonfigurasi ketiga aplikasi tersebut dengan sekaligus dan otomatis.

*Apache* adalah sebuah *web server open source*, jadi semua orang dapat menggunakannya secara gratis, bahkan anda bisa mengedit kode programnya. Fungsi utama dari *Apache* yakni menghasilkan halaman *web* yang benar sesuai dengan yang dibuat oleh seorang *web programmer*, dengan menggunakan kode *PHP*.

*PHP* adalah bahasa pemrograman untuk membuat *web*, dengan *PHP* anda dapat membuat halaman *web* yang dinamis. Selain mendukung di sistem operasi *Windows*, *PHP* juga dapat digunakan pada *mac OS*, *Linux*, dan sistem operasi yang lainnya.

*MySQL* adalah sistem manajemen *database* yang sering digunakan bersama *PHP*. *PHP* juga mendukung pada *Microsoft Access*, *Database Oracle*, *d-Base*, dan sistem manajemen *database* lainnya. *SQL (Structured Query Language)* adalah bahasa terstruktur yang digunakan secara khusus untuk mengolah *database*. Dan *MySQL* merupakan sebuah sistem manajemen *database*.

Dengan aplikasi yang juga *open source* ini, anda dapat membuat dan mengolah *database* beserta isinya, menambahkan, mengubah, dan menghapus data yang berada dalam *database*. Diperlukan *MySQL*, dan *PHPMysqlAdmin* adalah salah satu aplikasi yang anda bisa gunakan. Dengan *PHPMysqlAdmin* anda dapat membuat tabel, mengisi data, dan pekerjaan lainnya dengan mudah, tanpa harus mengafal perintahnya (Affandi, Khaerul. <http://khaerulaffandi.weebly.com/mengenal-apa-itu-xamppapachephp-dan-mysql.html>. Jam 14.01 dan tanggal akses 07 July 2019).

## 2.12 Definisi UML (*Unified Modeling Language*)

“*Unified Modeling Language* (UML) adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak. UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem” (Windu dan Grace, 2013). *Unified Modeling Language* (UML) adalah sebuah bahasa yang berdasarkan grafik atau gambar untuk memvisualisasi, menspesifikasikan, membangun, dan pendokumentasian dari sebuah sistem pengembangan software berbasis OO (*Object-Oriented*). UML sendiri juga memberikan standar penulisan sebuah sistem *blue print*, yang meliputi konsep bisnis proses, penulisan kelas-kelas dalam bahasa program yang spesifik, skema database, dan komponen - komponen yang diperlukan dalam sistem *software* (<http://www.omg.org>). Diagram *Unified Modelling Language* (UML) (Siti Fatima, 2015) antara lain sebagai berikut:

1. *Use Case* Diagram, *Use case* menggambarkan *external view* dari sistem yang akan kita buat modelnya (Prabowo Pudjo Widodo, 2011) Model *use case* dapat dijabarkan dalam diagram *use case*, tetapi perlu diingat, diagram tidak indetik dengan model karena model lebih luas dari diagram. (Pooley, 2003:15). *Use case* harus mampu menggambarkan urutan aktor yang menghasilkan nilai terukur (Prabowo Pudjo Widodo, 2011).
2. *Class* Diagram, Kelas sebagai suatu set objek yang memiliki atribut dan perilaku yang sama, kelas kadang disebut kelas objek (Whitten, 2004:410). Class memiliki tiga area pokok yaitu : 1) Nama, kelas harus mempunyai sebuah nama. 2) Atribut, adalah kelengkapan yang melekat pada kelas. Nilai dari suatu kelas hanya bisa diproses sebatas atribut yang dimiliki. 3) Operasi, adalah proses yang dapat dilakukan oleh sebuah kelas, baik pada kelas itu sendiri ataupun kepada kelas lainnya.

Tabel 2.3 Simbol *Class Diagram*

No	Simbol	Keterangan
1		Simbol ini adalah simbol untuk sebuah kelas pada struktur sistem. penulisan disana tidak diperbolehkan menggunakan spasi. simbol ini memiliki 3 susunan, yaitu kotak pertama adalah nama kelas, kedua atribut dan terakhir operasi.
2		Lingkaran ini adalah simbol untuk interface atau dalam bahasa indonesianya antar muka. konsep yang digunakan pun sama dengan pemrograman berorientasi object (OOP).
3		Simbol ini sering disebut dengan simbol Association atau dalam bahasa indonesianya yaitu asosiasi. Garis ini adalah garis yang digunakan untuk menghubungkan atau merelasikan kelas satu dengan kelas yang lainnya dengan makna umum.
4		Nama dari simbol ini adalah indirected association atau dalam bahasa indonesianya adalah asosiasi berarah. Simbol ini merupakan simbol relasi antar kelas seperti yang diatas, namun yang membedakan pada relasi ini adalah cara penggunaannya. Simbol ini digunakan jika kelas yang satu digunakan oleh kelas yang lainnya.
5		Simbol ini bernama Generalisasi. Generalisasi digunakan untuk menghubungkan antar kelas dengan arti umum-khusus. Jadi jika ada kelas bermakna umum dan kelas bermakna khusus dapat menggunakan simbol ini.
6		Nama dari simbol ini adalah Aggregation atau dalam bahasa indonesia nya Agregasi. Simbol ini adalah simbol yang menghubungkan antar kelas dengan makna untuk semua bagian. Jadi relasi ini digunakan jika kelas yang satu adalah semua bagian dari kelas yang lainnya.
6		Nama dari simbol ini adalah Dependency atau dalam bahasa indonesia nya ketergantungan. Kadangkala sebuah class menggunakan class yang lain. Umumnya penggunaan dependency digunakan untuk menunjukkan operasi pada suatu class yang menggunakan class yang lain. Sebuah dependency dilambangkan sebagai sebuah panah bertitik-titik.

3. *Activity Diagram*, Diagram aktifitas menunjukkan aktifitas sistem dalam bentuk kumpulan aksi-aksi, bagaimana masing-masing aksi tersebut dimulai, keputusan yang mungkin terjadi hingga berakhirnya aksi. *Activity diagram* juga dapat menggambarkan proses lebih dari satu aksi dalam waktu bersamaan. “Diagram *activity* adalah aktifitas-aktifitas, objek, state, transisi state dan event. Dengan kata lain kegiatan diagram alur kerja menggambarkan perilaku sistem untuk aktifitas” (Haviluddin, 2011).

4. *Sequence* Diagram “Secara mudahnya *sequence* diagram adalah gambaran tahap demi tahap, termasuk kronologi (urutan) perubahan secara logis yang seharusnya dilakukan untuk menghasilkan sesuatu sesuai dengan *use case* diagram.” (Haviluddin, 2011).

**Tabel 2.4 Simbol *Sequence* Diagram**

<b>aktor</b>  atau <span style="border: 1px solid black; padding: 2px;">nama_aktor</span>	<ul style="list-style-type: none"> <li>● orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi dan mendapat manfaat dari system.</li> <li>● Berpartisipasi secara berurutan dengan mengirimkan dan / atau menerima pesan.</li> <li>● Ditempatkan di bagian atas diagram.</li> </ul>
<b>objek</b> <span style="border: 1px solid black; padding: 2px;">objek:kelas</span>	<p><b>Sebuah objek:</b></p> <ul style="list-style-type: none"> <li>● Berpartisipasi secara berurutan dengan mengirimkan dan / atau menerima pesan.</li> <li>● Ditempatkan di bagian atas diagram.</li> </ul>
<b>Garis hidup objek</b> 	<ul style="list-style-type: none"> <li>● Menandakan kehidupan obyek selama urutan.</li> <li>● diakhiri tanda X pada titik di mana kelas tidak lagi berinteraksi.</li> </ul>
<b>Objek sedang aktif berinteraksi</b> 	<p><b>Fokus kontrol:</b></p> <ul style="list-style-type: none"> <li>● Adalah persegi panjang yang sempit panjang ditempatkan di atas sebuah garis hidup.</li> <li>● Menandakan ketika suatu objek mengirim atau menerima pesan.</li> </ul>
<b>pesan</b> 	<p>objek mengirim satu pesan ke objek lainya</p>
	<p>menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat</p>
	<p>menyatakan bahwa suatu objek mengirimkan masukan ke objek lainnya arah panah mengarah pada objek yang dikirim</p>
	<p>objek/metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian</p>
	<p>menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada create maka ada destroy</p>

### 2.13 Definisi Kamus Data

Kamus data (*data dictionary*) di pergunakan untuk memperjelas aliran data yang digambarkan pada DFD. Kamus data adalah kumpulan daftar elemen data yang mengalir pada sistem perangkat lunak sehingga masukan (*input*) dan keluaran (*output*) dapat dipahami secara umum. Kamus data biasanya berisi :

1. Nama, nama dari data.
2. Digunakan pada, merupakan proses-proses yang terkait data.
3. Deskripsi, merupakan deskripsi data.
4. Informasi tambahan, seperti tipe data, nial data, batas nilai data, dan komponen yang membentuk data.
5. Kamus data memiliki beberapa simbol untuk menjelaskan informasi tambahan sebagai berikut :

Contoh Kamus Data :

No	Nama Data	Tipe Data	Range	Deskripsi
1	Username	Varchar	10	Username
2	Password	Varchar	20	Password
3	Nm_Pegawai	Varchar	20	Nama Pegawai
4	Level	Char	1	Level

**Gambar 2.2 Contoh Kamus Data**

**Table 2.5 Simbol-Simbol Kamus Data.**

Simbol	Keterangan
=	Disusun atau terdiri dari
+	Dan
[]	Baik...atau.....
{ }n	N kali diulang atau bernilai banyak
()	Data optional
*...*	Batas komentar

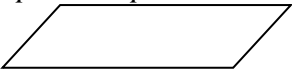

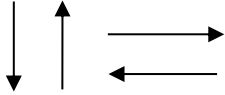

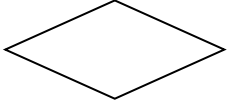
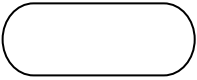

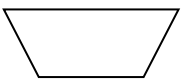
Kamus data pada DFD nanti harus dapat dipetakan dengan hasil perancangan basis data yang dilakukan sebelumnya. Jika ada kamus data yang tidak dapat dipetakan pada tabel hasil perancangan basis data dengan perancangan dengan DFD masih belum sesuai, sehingga harus ada yang diperbaiki baik perancangan basis datanya, perancangan DFD nya atau keduanya. (Rosa A.S M. Shalahudin, 2014)

#### 2.14 **Definisi *Flowchart***

*Flowchart* adalah adalah suatu bagan dengan simbol-simbol tertentu yang menggambarkan urutan proses secara mendetail dan hubungan antara suatu proses (instruksi) dengan proses lainnya dalam suatu program (Cybernur *Wordpress*, 2010).

Flowchart menurut pandangan Indrajani merupakan gambaran secara grafik dari langkah-langkah dan urutan prosedur suatu program. (Indrajani, 2011).

Tabel 2.6 Simbol *Flowchart*

Simbol	Keterangan
<p data-bbox="240 450 427 488"><i>Input / Output</i></p> 	<p data-bbox="627 465 1273 539">Simbol <i>input/output</i> digunakan untuk mewakili data <i>input/output</i>.</p>
<p data-bbox="240 589 523 627">Proses Komputerisasi</p> 	<p data-bbox="627 595 1273 701">Simbol proses (<i>processing symbol</i>) atau simbol pengolah yang digunakan suatu proses dengan program terkomputerisasi.</p>
<p data-bbox="240 734 363 772">Garis alir</p> 	<p data-bbox="627 768 1273 842">Simbol garis alir (<i>flow lines symbol</i>), digunakan untuk menunjukkan arus dari proses.</p>
<p data-bbox="240 889 403 927">Penghubung</p> 	<p data-bbox="627 898 1273 1037">Simbol penghubung (<i>connector symbol</i>), digunakan untuk menunjukkan sambungan dari bagan alir yang terputus dihalaman yang sama / dihalaman yang lain.</p>
<p data-bbox="240 1061 379 1099">Keputusan</p> 	<p data-bbox="627 1077 1273 1193">Simbol keputusan (<i>decision symbol</i>), digunakan untuk suatu penyelesaian kondisi didalam program.</p>
<p data-bbox="240 1238 360 1276">Terminal</p> 	<p data-bbox="627 1261 1273 1335">Simbol terminal digunakan untuk menunjukkan awal dan akhir dari suatu program.</p>
<p data-bbox="240 1397 368 1435">Dokumen</p> 	<p data-bbox="627 1413 1273 1529">Menunjukkan dokumen yang digunakan untuk input dan output baik secara manual maupun komputerisasi.</p>
<p data-bbox="240 1574 427 1612">Proses manual</p> 	<p data-bbox="627 1597 1273 1671">Menunjukkan pekerjaan yang dilakukan secara manual.</p>