BAB II

LANDASAN TEORI

2.1 Tinjauan Pustaka

2.1.1 Konsep Dasar Sistem Informasi

A. Pengertian Sistem Informasi

Tidak semua data dapat diolah dan digunakan sebagai bahan pertimbangan keputusan dalam perusahaan. Oleh karena itu dibutuhkan suatu sistem yang dapat mengolah data. Sistem itu harus dirancang sedemikian rupa agar dapat menntukan validitas data yang berasal dari berbagai sumber.

Sistem informasi dapat didefinisikan sebagai kumpulan elemen yang saling berhubungan satu sama lain yang membentuk satu kesatuan untuk mengintegrasikan data memproses dan menyimpan serta mendistribusikan informasi. Dengan kata lain Sistem Informasi merupakan kesatuan elemen-elemen yang saling berinteraksi secara sistematis dan teratur untuk menciptakan dan membentuk aliran informasi yang akan mendukung pembuatan keputusan dan melakukan kontrol terhadap jalannya perusahaan (Watung, dkk,2014:1).

Menurut *McLeod* dalam Watung dkk (2014:2) mendefinisikan bahwa “informasi adalah data yang diolah menjadi bentuk lebih berguna dan lebih berarti bagi yang menerimanya”.

2.1.2 Konsep Dasar *Website*

A. *Internet*

Menurut Hidayatullah dan Kawistara (2014:1-2) "*Internet* adalah jaringan *global* yang menghubungkan komputer-komputer di seluruh dunia. Dengan *internet*, sebuah toko *online* bisa tetap terbuka selama 24 jam sehari dan 7 hari seminggu tanpa henti. Dengan *internet*, kejadian penting yang terjadi di suatu negara bisa segera diketahui oleh orang lain di negara yang berbeda".

B. *Website*

Menurut Hidayat (2010:2) menyimpulkan bahwa :
Website atau situs dapat diartikan sebagai kumpulan halaman-halaman yang digunakan untuk menampilkan informasi teks, gambar diam atau gerak, animasi, suara, dan atau gabungan dari semuanya, baik yang bersifat statis maupun dinamis yang membentuk suatu rangkaian bangunan yang saling terkait, yang masing-masing dihubungkan dengan jaringan-jaringan halaman. Hubungan antara satu halaman web dengan halaman web yang lainnnya disebut *hyperlink*, sedangkan teks yang dijadikan media penghubung disebut *hypertext*.

2.1.3 Bahasa Pemrograman

A. *PHP*

Menurut Ardhana (2013:4) menyatakan bahwa "*PHP* atau *PHP Hypertext Prepocesssor* merupakan bahasa pemrograman yang berjalan disisi server. Ketika *user* melakukan akses ke sebuah alamat situs dengan mengetikkan alamat *URL*nya, *browser* akan mengirimkan *request*/permintaan ke *webserver*".

B. *Javascript*

Menurut Menurut Andi (2012:iii) "*Javascript* merupakan salah satu bahasa *script* yang dijalankan pada sisi *client*, yaitu *browser".*

C. *Jquery*

Menurut Hidayatullah dan Kawistara (2014:426) "*Jquery* adalah kumpulan fungsi-fungsi *javascript* yang sudah dibentuk sebagai suatu objek*".* Sehingga penggunaan *jquery* ini bisa dikategorikan sebagai suatu *library* yang nantinya kita hanya perlu menggunakan fungsi-fungsi di dalam *library* tersebut. *Jquery* pertama kali dirilis oleh *john resig* pada tahun 2006. Pada perkembangannya *jquery* tidak sekadar sebagai *library javascript,* namun memiliki kehandalan dan kelebihan yang cukup banyak. Hal tersebut menyebabkan banyak *developer* web menggunakannya. *Jquery* dikenal dengan slogan "*Write less, do more"* artinya penulisan kode yang sedikit tetapi memiliki beberapa aksi (*action*). *Jquery* juga merupakan *library open source* dengan lisensi *GNU General Public License* dan *MIT License.* Ukuran file *jquery* tidak lebih dari 200kb. *Jquery* juga sudah *support plugin-plugin* tambahan untuk fungsi-fungsi pada masalah yang lebih spesifik.

D. *CSS*

Menurut Winarno dan Zaki (2015:69) "*CSS* merupakan singkatan dari *casscading style sheets*. *CSS* berfungsi mendefinisikan bagaimana elemen *HTML* ditampilkan". *Style* sendiri mulai diperkenalkan sejak versi *HTML* 4.0 untuk menentukan *style* dokumen. *Style sheet* eksternal bisa menghemat banyak waktu. *Style sheet eksternal* diletakan di file tersendiri yang memiliki ekstensi *.css.* Kenapa *CSS* sangat penting? Ini karena pada awalnya *HTML* tidak pernah diniatkan untuk berisi *tag* untuk pemformatan dokumen. *HTML* hanya diniatkan untuk mendefinisikan konten dokumen saja.

E. *Database* / Basis Data

Menurut Watung (2014:2) menyatakan bahwa "Basis data merupakan komponen terpenting dalam pembangunan Sistem Informasi, karena menjadi tempat untuk menampung dan mengorganisasikan seluruh data yang ada dalam sistem, sehingga dapat dieksplorasi untuk menyusun informasi-informasi dalam berbagai bentuk". Basis data merupakan himpunan kelompok data yang saling berkaitan. Dengan basis data, pengguna dapat menyimpan data secara terorganisasi. Setelah data disimpan, informasi harus mudah diambil. Prinsip utama basis data adalah pengaturan data dengan tujuan utama fleksibilitas dan kecepatan dalam pengambilan data kembali.

F. *MySQL*

Menurut Watung (2014:2) mendefinisikan bahwa "*MySQL* adalah sebuah perangkat lunak sistem manajemen basis data *SQL* (bahasa Inggris: *database management system*) atau *DBMS* yang *multithread, multiuser,* dengan sekitar 6 juta instalasi di seluruh dunia*"*.

H. *Framework CodeIgniter*

Menurut Ardhana (2013:22) menyatakan bahwa "*Framework* secara sederhana dapat diartikan sebagai kumpulan dari fungsi atau prosedur dan class untuk tujuan tertentu yang sudah siap untuk digunakan sehingga mempermudah dan mempercepat programmer dalam membuat program tanpa harus membuat fungsi atau class dari awal". *CodeIgniter* adalah salah satu sekian banyak dari *framework PHP* yang ada. *CodeIgniter* dikembangkan oleh oleh Rick Ellis pendiri CEO *EllisLab.com*

2.1.4 *UML (Unified Modeling Language)*

Menurut Widodo dan Herlawati (2011:6) "*UML* singkatan dari *Unified Modeling Language* yang berarti bahasa pemodelan standar".

> Sedangkan menurut *Chonoles* tahun 2003 dalam Widodo dan Herlawati (2011:6-7) mengatakan :
> Sebagai bahasa, berarti *UML* memiliki sintaks dan semantik. Ketika kita membuat *model* menggunakan konsep *UML* ada aturan-aturan yang harus diikuti. Bagaimana elemen pada *model-model* yang kita buat berhubungan satu dengan lainnya harus mengikuti standar yang ada. *UML* bukan hanya sekedar *diagram*, tetapi juga menceritakan konteksnya. Ketika pelanggan memesan sesuatu dari sistem, bagaimana transaksinya? Bagaimana *sistem* mengatasi *error* yang terjadi? Bagaimana keamanan terhadap sistem yang kita buat? Dan sebagainya dapat dijawab dengan *UML*.

UML diaplikasikan untuk maksud tertentu, biasanya antara lain untuk (Widodo dan Herlawati, 2011:6):

1. Merancang perangkat lunak.
2. Sarana komunikasi antara perangkat lunak dengan proses bisnis.
3. Menjabarkan sistem secara rinci untuk analisa dan mencari apa yang diperlukan sistem.
4. Mendokumentasikan sistem yang ada, proses-proses dan organisasinya.

UML telah diaplikasikan dalam bidang investasi perbankan, lembaga kesehatan, departemen pertahanan, sistem terdistribusi, sistem pendukung alat kerja, *retail*, *sales* dan *supplier*.

A. *Use Case Diagram*

Menurut Widodo dan Herlawati (2011:6) "salah satu kontributor terhadap diagram *use case* dalam *UML* adalah *Ivar Jacobsen. Use case* menggambarkan *external view* dari sistem yang akan kita buat modelnya".

Sedangkan, menurut *Pooley* tahun 2003 dalam Widodo dan Herlawati (2011:16) mengatakan bahwa "model *use case* dapat dijabarkan dalam diagram *use*

case, tetapi yang perlu diingat, diagram tidak identik dengan model karena model lebih luas dari diagram".

Komponen pembentuk diagram *use case* adalah (Widodo dan Herlawati, 2011:16) :

1. Aktor (*actor*), menggambarkan pihak-pihak yang berperan dalam sistem.
2. *Use Case*, aktivitas/sarana yang disiapkan oleh bisnis/sistem.
3. Hubungan (*link*), aktor mana saja yang terlibat dalam *use case* ini.

B. *Activity Diagram*

Menurut Widodo dan Herlawati (2011:143-144) "Diagram aktivitas lebih memfokuskan diri pada eksekusi dan alur sistem dari pada bagaimana sistem itu dirakit". Diagram ini tidak hanya memodelkan *software* melainkan memodelkan model bisnis juga. Diagram aktivitas menunjukan aktivitas sistem dalam bentuk kumpulan aksi-aksi. Ketika digunakan dalam pemodelan *software*, diagram aktivitas mempresentasikan pemanggilan suatu fungsi tertentu misalnya *call*. Sedangkan bila digunakan dalam pemodelan bisnis, diagram ini menggambarkan aktivitas yang dipicu oleh kejadian-kejadian internal misalnya proses penggajian tiap jumat sore.

C. *Component Diagram*

Menurut *Chonoles* tahun 2003 dalam Widodo dan Herlawati (2011:93) mengatakan "Manfaat diagram komponen adalah bila ada salah satu komponen yang rusak atau tidak sesuai dengan tujuan sistem, kita tinggal mengganti komponen itu dengan komponen yang lain".

Agar komponen dapat dibongkar pasang, harus memenuhi kriteria sebagai berikut (Widodo dan Herlawati, 2011:93-95) :

1. Memiliki kerja internal yang tersembunyi. Objek yang berada didalamnya harus bebas akses oleh objek diluar komponen. Jadi jika kita ingin benar-benar objek dalam komponen kita bisa di-*replaace* kata harus membuat objek tersebut tidak memiliki *dependency* terhadap objek pada komponen lain.
2. Memiliki antar muka(*interface*). *Interface* mendeskripsikan operasi apa yang harus diambil terhadap suatu komponen dan bukan bagaimana operasi itu dikerjakan. Menyediakan *interface* adalah suatu cara untuk menyembunyikan kerja *internal* suatu komponen dari objek diluar komponen.
3. Komponen didalam harus *independent.* Kita harus yakin bahwa objek didalam komponen kita tidak tahu menahu dengan objek lain diluar komponen sebab jika tidak saat kita mengganti objek tersebut, sistem akan terganggu.
4. Antar muka terhadap komponen lain harus tersedia. Manfaatnya agar objek *internal* berhubungan dengan objek di komponen lain lewat antar muka. Jadi saat objek *internal* ingin berhubungan dengan objek lain diluar komponen alurnya adalah lewat antar muka komponennya kemudian menuju antar muka komponen objek sasaran yang diteruskan ke objek itu.

D. *Deployment Diagram*

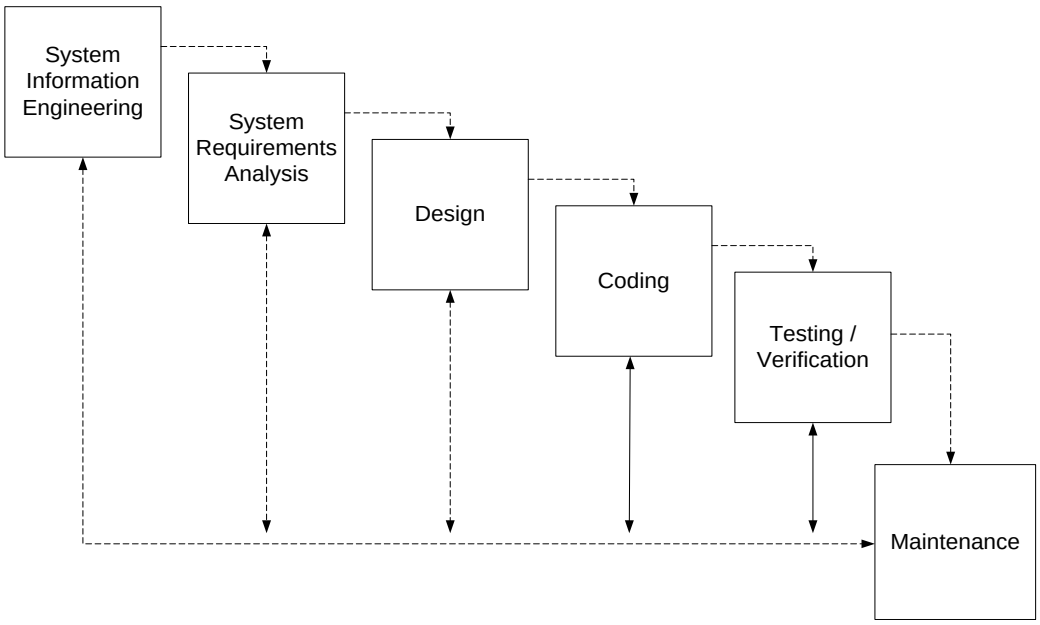
Menurut Widodo dan Herlawati (2011:109-110) “Model diagram *deployment* bagian-bagian perangkat lunak suatu sistem ke perangkat keras yang akan mengeksekusinya”.

Sedangkan, menurut *Pender* tahun 2003 dalam Widodo dan Herlawati (2011:110) mengatakan bahwa “Walaupun diagram komponen dan *deployment*

merupakan bagian dari spesifikasi *UML*,bagi sebagian besar perancang merupakan hal baru".

2.1.5 Model Pengembangan Sistem

Menurut *Pressman* dalam Watung (2014:3-4) menyatakan bahwa "Salah satu model perancangan yang dapat digunakan adalah metode *waterfall*. Nama model ini sebenarnya adalah *linier sequential model*".



Sumber : Pressman dalam Watung (2014:3)

Gambar II.1
Model *Waterfall*

Gambar II.1 diatas adalah tahapan dari model *waterfall*. *Pressman* memecah model ini menjadi 6 tahapan meskipun secara garis besar sama dengan tahapan-tahapan model *waterfall* pada umumnya. Berikut adalah penjelasan dari tahap-tahap yang dilakukan di dalam model ini menurut *Pressman :*

1. *System / Information Engineering and Modeling.*

   Pemodelan ini diawali dengan mencari kebutuhan dari keseluruhan sistem yang akan diaplikasikan ke dalam bentuk *software*. Hal ini sangat penting,

mengingat *software* harus dapat berinteraksi dengan elemen-elemen yang lain seperti *hardware, database,* dsb. Tahap ini sering disebut dengan *project definition.*

2. *Software Requirements Analysis.*

   Proses pencarian kebutuhan diintensifkan dan difokuskan pada *software*. Untuk mengetahui sifat dari program yang akan dibuat, maka para *software engineer* harus mengerti tentang domain informasi dari *software,* misalnya fungsi yang dibutuhkan, *user interface,* dsb.

3. *Design*

   Proses ini digunakan untuk mengubah kebutuhan-kebutuhan diatas menjai representasi kedalam bentuk "*blueprint" software* sebelum *coding* dimulai. Desain harus dapat mengimplementasikan kebutuhan yang telah disebutkan pada tahap sebelumnya.

4. *Coding*

   Untuk dapat dimengerti oleh mesin, dalam hal ini adalah komputer, maka desain tadi harus diubah menjadi bentuk yang dapat dimengerti oleh mesin, yaitu ke dalam bahasa pemrograman melalui proses *coding.* Tahap ini merupakan implementasi dari tahap *design* yang secara teknis nantinya dikerjakan oleh *programmer.*

5. *Testing / Verification*

   Sesuatu yang dibuat haruslah diujicobakan. Demikian juga dengan *software.* Semua fungsi-fungsi *software* harus diujicobakan, agar *software* bebas dari *error,* dan hasilnya harus benar-benar sesuai dengan kebutuhan yang sudah didefinisikan sebelumnya.

6. *Maintenance*

Pemeliharaan suatu *software* diperlukan, termasuk di dalamnya adalah pengembangan, karena *software* yang dibuat tidak selamanya hanya seperti itu. Ketika dijalankan mungkin saja masih ada *error* kecil yang tidak ditemukan sebelumnya, atau ada penambahan fitur-fitur yang belum ada pada *software* tersebt. Pengembangan diperlukan ketika adanya perubahan dari eksternal seperti ketika ada pergantian sistem operasi, atau perangkat lainnya.

2.1.6 Diagram Relasi Entitas (*Entity Relationship Diagram*)

Menurut Simarmata dan paryudi dalam Octavian (2011:150) menyatakan bahwa "*Entity relationship (ER)* data model didasarkan pada persepsi terhadap dunia nyata yang tersusun atas kumpulan objek-objek dasar yang disebut entitas dan hubungan antarobjek".

Pada *model entity-relationship,* semesta data yang ada di 'dunia nyata' diterjemahkan/ditransformasikan dengan memanfaatkan sejumlah perangkat konseptual menjadi sebuah diagram data, yang umum disebut sebagai *diagram entity-relationship* (*diagram E-R*). Sebelum kita membahas lebih jauh tentang bagaimana *diagram E-R* itu dapat kita gambarkan, maka yang harus lebih dulu diketahui adalah komponen-komponen pembentuk *model entity-relationship* (Fathansyah, 2007:72-75) :

A. **Komponen *ERD***

Komponen *Entity Relationship Diagram* adalah sebagai berikut :

1. Entitas *(Entity)*

Entitas merupakan individu yang mewakili sesuatu yang nyata (eksistensinya) dan dapat dibedakan dari sesuatu yang lain.

2. Atribut *(attributes/properties)*

   Setiap entitas pasti memiliki atribut yang mendeskripsikan karakteristik *(property)* dari entitas tersebut. Penentuan/pemilihan atribut-atribut yang relevan bagi sebuah entitas merupakan hal penting lainnya dalam pembentukan model data.

3. Relasi*(Relationship)*

   Relasi menunjukkan adanya hubungan diantara sejumlah entitas yang berasal dari himpunan entitas yang berbeda.

B. **Derajat *Relationship***

Menurut Fathansyah (2007:77-79) mengungkapkan bahwa "Kardinalitas Relasi menunjukkan jumlah maksimum entitas yang dapat berelasi dengan entitas pada himpunan entitas yang lain". Berikut adalah macam-macam kardinalitas, yaitu (Fathansyah, 2007:77-79):

1. Satu ke Satu *(One to One)*

   Setiap entitas pada himpunan entitas A berhubungan dengan paling banyak dengan satu entitas pada himpunan entitas B, dan begitu juga sebaliknya setiap entitas pada himpunan entitas B berhubungan dengan paling banyak dengan satu entitas pada himpunan entitas A.

2. Satu ke Banyak *(One to Many)*

   Setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B, tetapi tidak sebaliknya, dimana setiap entitas pada himpunan entitas B berhubungan dengan paling banyak dengan satu entitas pada himpunan entitas A.

3. Banyak ke Satu *(Many to One)*

   Setiap entitas pada himpunan entitas A berhubungan dengan paling banyak dengan satu entitas pada himpunan entitas B, tetapi tidak sebaliknya, dimana setiap entitas pada himpunan entitas A berhubungan dengan paling banyak satu entitas padahimpunan entitas B.

4. Banyak ke Banyak *(Many to Many)*

   Setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B, dan demikian juga sebaliknya, dimana setiap entitas pada himpunan entitas B dapat berhubungan dengan banyak entitas pada himpunan entitas A.

2.1.7 *Logical Record Structure* (*LRS*)

Menurut Fathansyah (2007:114-124) menyatakan bahwa "Cara transformasi Model Data yang dinyatakan kedalam Diagram E-R ke dalam basis data fisik". Berikut akan ditunjukkan cara transformasi yang sifatnya umum/*standard* yaitu (Fathansyah, 2007:114-124):

1. Transformasi Umum/Dasar

   Aturan umum dalam pemetaan Model Data (Level Konseptual dalam Abstraksi Data) yang kita gambarkan dengan diagram E-R menjadi Basis Data Fisik (Level Fisik Dalam Abstraksi Data) adalah :

   a. Setiap himpunan entitas akan diimplementasikan sebagai sebuah tabel (*file data*)
   b. Relasi dengan Derajat Relasi 1-1 (satu-ke-satu) yang menghubungkan 2 buah himpunan entitas akan direpresentasikan dalam bentuk

penambahan/penyertaan atribut-atribut relasi ke tabel yang mewakili salah satu dari kedua himpunan entitas.

c. Relasi dengan Derajat Relasi 1-N (satu-ke-banyak) yang menghubungkan dua buah himpunan entitas, juga akan direpresentasikan dalam bentuk pemberian/pencantuman atribut *key* dari himpunan entitas pertama (yang berderajat satu) ke tabel yang mewakili himpunan entitas kedua (yang berderajat N).

d. Relasi dengan Derajat Relasi N-N (banyak-ke-banyak) yang menghubungkan dua buah himpunan entitas, akan diwujudkan dalam bentuk tabel (*file data*) khusus yang memiliki *field* (tempat *Foreign Key*) yang berasal dari *key-key* dari himpunan entitas yang dihubungkannya.

2. Implementasi Himpunan Entitas Lemah dan Sub Entitas

Penggunaan himpunan entitas lemah (*Weak Entity Sets*) dan Sub entitas dalam Diagram E-R di implementasikan dalam bentuk tabel sebagaimana himpunan entitas kuat *(Strong Entity Sets).*

3. Implementasi Relasi Tunggal*(UnaryRelation)*

Implementasi relasi tunggal *(Unary Relation)* dari/ke himpunan entitas yang sama dalam Diagram E-R tergantung pada derajat relasinya.

4. Implementasi Relasi Multi Entitas*(N-ary Relation)*

Secara umum, relasi multi entitas yang menghubungkan lebih dari dua himpunan entitas (N himpunan entitas, dimana N>2) akan diimplemtasikan sebagai sebuah tabel khusus (tentu saja, setiap himpunan entitas yang terlibat dalam relasi juga akan direpresentasikan dalam tabel-tabel terpisah).

5. Implementasi Relasi Ganda *(Redudant Relation)*

   Tidak ada yang istimewa dalam mengimplementasikan relasi ganda di antara dua himpunan entitas. Implementasinya ditinjau pada masing-masing relasi tanpa terikat satu sama selain berdasarkan Derajat Relasi di masing-masing relasi terebut.

6. Implementasi Spesialisasi dan Generalisasi

   Spesialisasi terhadap sebuah himpunan entitas akan menghasilkan sejumlah himpunan entitas baru: satu himpunan entitas kuat/bebas yang akan menjadi acuan bagi himpunan entitas lainnya dan sisanya merupakan sub entitas.

7. Implementasi Agregasi

   Sesungguhnya Agregasi dapat dipandang sebagaimana relasi pada umumnya (yang menghubungkan dua himpunan entitas). Karena relasi ini dibentuk dari relasi lain (relasi prasyarat) yang secara kronologis lebih dulu terbentuk,maka pengimplementasiannya juga harus dilakukan setelah relasi prasyarat tersebut terimplementasikan. Selanjutnya tinggal meninjau derajat relasi dari relasi agregasinya.

2.1.8 *Object Oriented Prgramming*

Menurut Ardhana (2013:5-6) menyatakan bahwa "*Object Oriented Programming* atau biasa disebut dengan pemrograman berorientasi obyek memberikan gambaran dimana sebuah perangkat lunak atau *software* dijadikan sebagai kumpulan obyek-obyek yang saling berinteraksi dalam suatu sistem". Berikut adalah istilah penting dalam dalam *Object Oriented Programming* Ardhana (2013:5-6):

1. *Abstraction*

   Abstraction atau lebih dikenal dengan abstraksi adalah teknik untuk menentukan ciri, sifat atau informasi penting dari suatu obyek yang akan ditampilkan dan mana yang tidak ditampilkan atau disembunyikan.

2. *Object*

   Elemen dasar dari konsep *Object Oriented Programming* adalah obyek itu sendiri. *Object* atau obyek merupakan abstraksi dari suatu dalam dunia nyata. Kecenderungan pada obyek selalu terkandung *attribute* dan *method* didalamnya. *Attibute* adalah data yang terdapat pada obyek sedangkan *method* adalah operasi-operasi yang disediakan oleh boyek untuk mengakses atau melakukan modifikasi atau perubahan terhadap *attribute* yang dimilikinya.

3. *Class*

   Class merupakan sekumpulan obyek yang memiliki kesamaan keadaan dan perilaku. Class berperan sebagai sarana pengkapsulan kumpulan data dan kumpulan *method*. Kumpulan *method* berfungsi untuk meakukan operasi data pada *class* tersebut.

4. *Inheritance*

   Inheritance atau lebih dikenal dengan pewarisan merupakan *class* yang dibuat berdasarkan *class* yang sudah ada sebelumnya. Dalam hal ini dapat dikatakan bahwa *inheritance* merupakan *class* baru yang dibuat dengan mewarisi sifat-sifat dari *class* sebelumnya yang disebut sebagai *subclass*. *Class* yang mewariskan sifat-sifatnya disebut *superclass.*

5. *Polymorphism*

Polymorphism adalah suatu object yang dapat memiliki berbagai bentuk yaitu sebagai obyek dari *class*-nya sendiri atau sebagai obyek dari *sepperclass*-nya

2.1.9 Pengujian

Menurut Al Fatta (2007:172) menyimpulkan bahwa: *black box testing* adalah terfokus pada apakah unit program memenuhi kebutuhan (*requirement*) yang disebutkan dalam spesifikasi. Pada *black box testing*, cara pengujian hanya akan dilakukan dengan menjalankan atau mengeksekusi unit atau modul, kemudian diamati apakah hasil dari unit itu sesuai dengan proses bisnis yang diinginkan.

2.2 Penelitian Terkait

Dalam pembuatan sistem informasi ikhtisar kas madjid ini, penulis menggunakan beberapa acuan yang bersumber dari beberapa jurnal ilmiah.

> Menurut Wardana dan Aribowo (2013:119) menyimpulkan bahwa :
> Suatu informasi pada kenyataan akan lebih efisien dan efektif dengan diterapkanya komputerisasi, karena segala sesuatu dituntut serba cepat dan akurat. Namun demikian harus diakui saat ini Masjid Jogokariyan Yogyakarta, dalam pengelolaan administrasi sekertariat dan keuangan masih mengunakan cara manual, dalam pembuatan laporan penghimpunan data jamaah, kegiatan, mubaligh, dan pembuatan laporan penghimpunan dana yang diterima tidak memiliki keterangan dari mana diperoleh, tidak adanya perincian jumlah yang diberikan oleh donatur, serta tidak ada keterangan mengenai jenis dana yang diterima. Tujuan penelitian ini adalah membuat aplikasi sistem informasi administrasi masjid Jogokariyan, guna mendukung kinerja dan tugas pengelolaan administrasi masjid. Hasil penelitian ini menunjukan bahwa analisis kebutuhan sistem yang berhubungan dengan informasi administrasi masjid. Dari penelitian tersebut dihasilkan sebuah aplikasi sistem informasi administrasi masjid Jogokariyan Yogyakarta yang dapat digunakan untuk membantu kinerja petugas dalam mengelola administrasi sekertariatan masjid dan pengelolaan keuangan masjid.

> Menurut Rochman (2014:42) menyimpulkan bahwa :
> Tujuan penelitian ini adalah merancang sistem informasi pengendalian donatur jamaah masjid dan mengimplementasikan sistem informasi

pengendalian donatur, agar semua jamaah dapat lebih percaya terhadap amal jariyah yang mereka sumbangkan ke masjid. Pembuatan sistem informasi pengendalian donatur jamaah masjid menggunakan bahasa pemrograman foxpro9 dan pembangunan database menggunakan software bawaan Visual FoxPro yang inklud dalam software Visual FoxPro Pembuatan program aplikasi disusun berdasarkan hirarchies menu yang meliputi Menu Utama, Sub Menu Donatur, No Akun, Jurnal, Laporan Donatur, Laporan Pembangunan, dan Laporan Yayasan, Sedang pembangunan database terdiri dari tabel tabel antara lain Tabel Donatur dan Tabel No Akun.