**BAB 2**

**LANDASAN TEORI**

* 1. **Teori Umum**
     1. **Pengertian *System Development Life Cycle***

*System Development Life Cycle (SDLC)* adalah serangkaian proses atau tahapan yang bertujuan untuk menghasilkan sebuah sistem informasi yang mampu mendukung atau sesuai dengan kebutuhan bisnis dari organisasi. SDLC terdapat empat tahapan atau proses dasar yaitu *Planning, Analysis, Design dan Implementation (Dennis, 2015).*

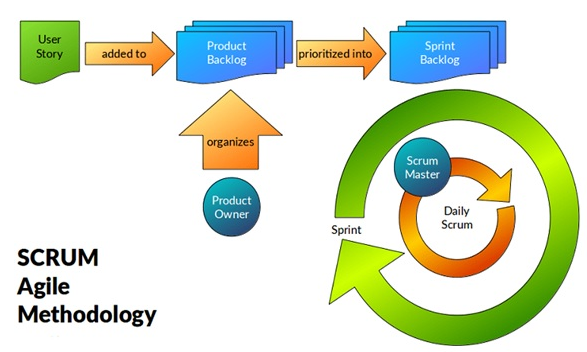
Empat tahapan atau proses *SDLC* yaitu sebagai berikut :

1. *Planning* yaitu mengidentifikasi nilai bisnis, analisis kelayakan dan perencanaan proyek.
2. *Analysis* yaitu menyelidiki sistem saat ini atau yang sedang berjalan dan yang akan dibuat atau yang baru, pengumpulan kebutuhan atau persyaratan dan mempresentasikan proposal sistem.
3. *Design* yaitu menentukan siapa yang akan bertanggung jawab untuk mengembangkan sistem dan desain arsistektu antar muka, basis data dan program.
4. *Implementation* yaitu melakukan pengujian sistem, instalasi dan perencanaan dukungan.

Terdapat beberapa pendekatan metodologi untuk menjalankan SDLC diantaranya adalah :

1. *Structured Design* dengan contoh modelnya seperti *Waterfall Development* dan *Parallel Development.*
2. Pendekatan *Rapid Application Development* dengan contoh modelnya seperti *Phased Development, Prototyping dan Throwaway Prototyping.*
3. Pendekatan *Agile Development* dengan contoh modelnya seperti *Extreme Programming* dan *Scrum.*
   * 1. **Pengertian Scrum Model**

*Scrum* adalah sebuah kerangka kerja untuk mengatur pengembangan proyek pada pendekatan *Agile Development* (Dennis, 2015). *Scrum* dapat mengurangi ketidak efektifan dari manajemen produk dan teknik kerja sehingga dapat meningkatkan kinerja produk, tim dan lingkungan kerja.



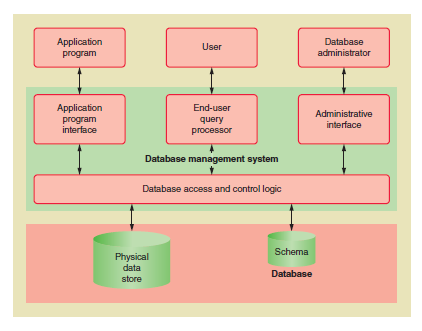
Gambar 2. 1 Scrum Agile Methodology

* *Scrum Master* adalah orang yang secara efektif mengoperasikan kelompok *scrum* agar selalu berjalan dengan baik dan sebagai mediator untuk memastikan solusi dalam menyelesaikan masalah-masalah yang ada.
* *Product Owner* adalah orang yang bertanggung jawab dan menyediakan tim *scrum* menghasilkan produk yang dapat diserahkan kepada klien.
* *Product Backlog* adalah daftar keinginan pengguna dalam bentuk yang mereka harapkan ketika proyek ini selesai, dari daftar ini dapat menentukan proyek yang telah dikerjakan memenuhi keinginan pengguna atau belum.
* *Sprint Backlog* adalah hasil dari sebagian dari *Product Backlog* yang harus dikembangkan, menentukan daftar mana yang harus dikerjakan terlebih dahulu dan cara melakukan untuk dapat memenuhi harapan para pengguna yang ada pada *Product Backlog.*
* *User Story* adalah hasil dari wawancara pengguna dan didokumentasikan dalam perasyaratan perangkat lunak yang digunakan dalam agile.

* + 1. **Pengertian *Database Management System (DBMS)***

*Database* adalah kumpulan data terintegrasi yang diatur dan dikelola secara terpusat. Basis data berisi informasi tentang lusinan atau ratusan kelas yang diatur dan dikelola oleh sistem manajemen basis data *"DBMS"*. *DBMS* adalah komponen perangkat lunak sistem yang dibeli dan diinstal secara terpisah dari komponen perangkat lunak sistem lainnya.

Menurut Satzinger, Jackson, & Burd (2012), DBMS memiliki 4 (empat) komponen kunci, yaitu *application program interface* (API), *query interface, administrative interface,* dan set data pokok yang dapat mengakses program dan subrutin.



Gambar 2. 2 Komponen dan Interaksi DBMS

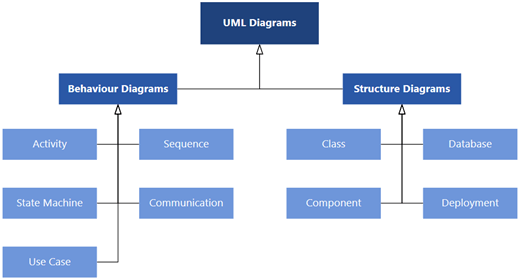
Keunggulan adanya DBMS sebagai media pengelola data adalah praktis, cepat dan *up-to-date*, selain itu penggunaan DBMS juga sangat luas. DBMS menyediakan beberapa akses data penting dan kapabilitas manajemen, termasuk:

1. Akses bersama oleh banyak pengguna dan program aplikasi.
2. Akses ke data tanpa menulis pada program aplikasi (contohnya, melalui bahasa pemrograman).
3. Keseragaman aplikasi dan akses yang konsisten, serta kontrol konten.
4. Integrasi antar data tersimpan pada banyak *server* yang didistribusikan pada banyak lokasi.

Pada dasarnya DBMS berguna sebagai media penyimpanan data yang tersusun dengan rapi sehingga aplikasi dapat mengakses dan bertransaksi data dengan aman dan cepat, tanpa khawatir adanya kesalahan atau bahkan kerusakan data dan informasi yang berguna bagi pengguna.

Contoh DBMS diantaranya Microsoft SQL Server, Oracle, dan MySQL. MySQL merupakan salah satu platform DBMS yang paling sering digunakan oleh para *programmer* di dunia. Hampir semua aplikasi berbasis web ataupun aplikasi berbasis *desktop* menggunakan platform *database* gratis ini. DBMS ini mendukung banyak aplikasi lain dan menjadi ketergantungan dalam sistem. Contohnya Apache dan PHP sebagai *web service*, Java di aplikasi Android, atau bahasa pemrograman lain di berbagai platform teknologi saat ini. Kemudahan mengintegrasikan MySQL dengan platform lain menjadikan MySQL salah satu pilihan terbaik bagi programmer dalam mendukung pengembangan sebuah sistem.

* + 1. ***Unified Modeling Language* *(UML)***



Gambar 2. 3 UML Diagram

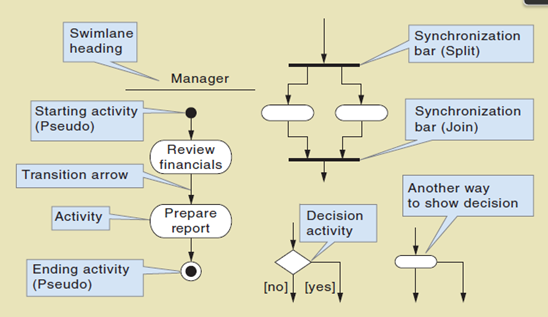
Undefied Modelling Language (UML) adalah sekelompok notasi grafis yang didukung oleh mata model tunggal, yang membantu pendeskripsian dan desain system perangkat lunak, khususnya system yang dibangun menggunakan pemrograan berorientasi objek (OO) (Flower, 2004).

UML adalah Bahasa standar yang digunakan untuk menjelaskan dan menvisualisasikan artifak dari proses analysis dan desain berorientasi obyek. Diagram use case atau use case diagram menyajikan interaksi antara use case dan actor. Dimana, aktor dapat berupa orang, peralatanm atau system lain yang berinteraksi dengan system lain yang berinteraksi dengan system yang sedang dibangun. Use case menggambarkan fungsional sistem atau persyaratan yang harus dipenuhi sistem dari pandangan pemakai. Diagram aktvitas atau activity diagram yang menggambarkan aliran fungsional sistem. Pada tahap pemodelan, activity diagram dapat digunakan untuk memajukan aliran kerja bisnis (*business workflow*). Diagram sekuensial digunakan untuk menunjukkan aliran fungsionalitas dan use case (Kendall, 2010).

Unified Modelling Language menampilkan suatu rabgkaian perangkat standar untuk mendikumentasikan analisis dan perancangan berorientasi obyek dari suatu sistem perangkat lunak.

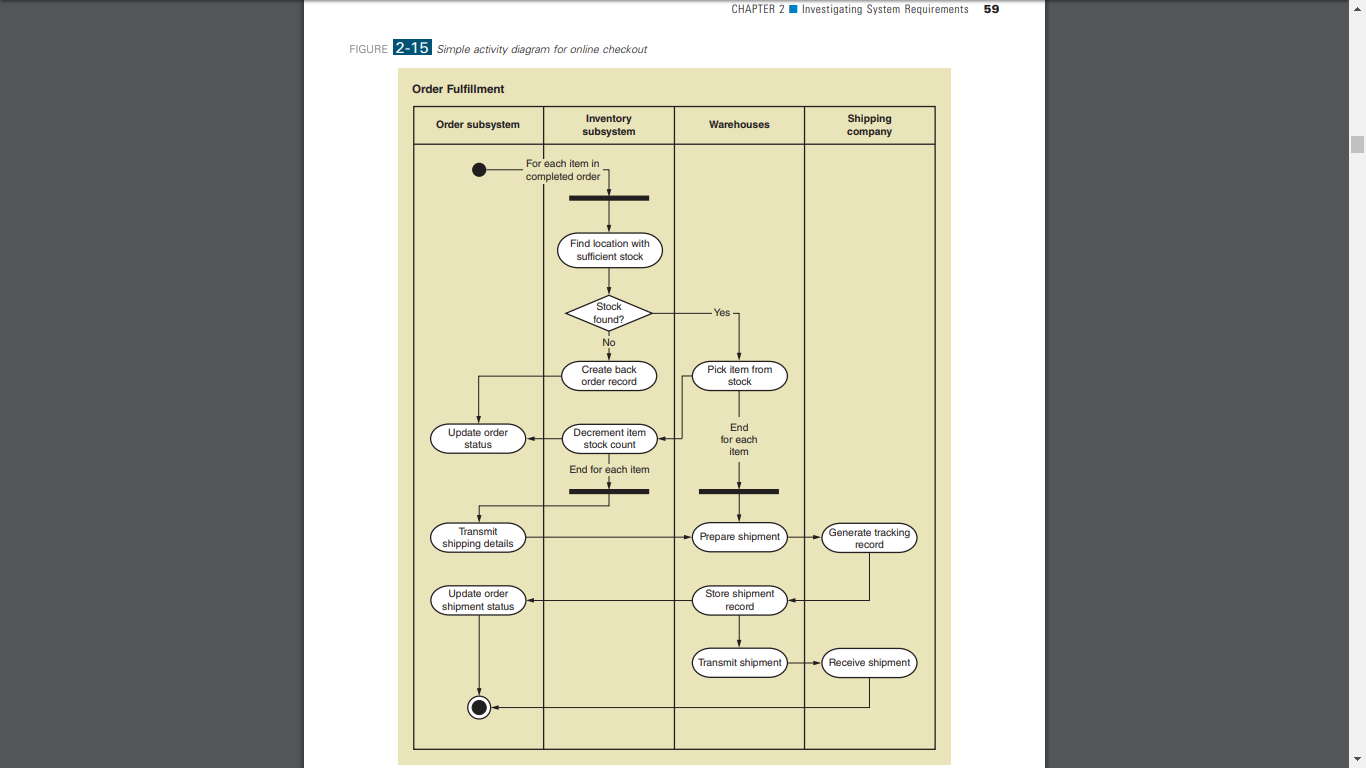
* + - 1. ***Activity Diagram***

Menurut Satzinger, Jackson, & Burd (2012), *activity diagram* menggambarkan berbagai macam aktivitas *user* (atau sistem), *user*/aktor yang melakukan tiap aktivitas, dan urutan aliran dari aktivitas-aktivitas tersebut. Terdapat sejumlah simbol yang digunakan dalam *activity diagram* yang ditunjukan pada gambar 2.4.



Gambar 2. 4 Notasi Activity Diagram

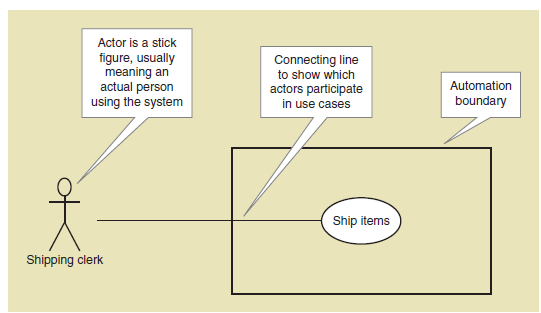
Pada *activity diagram* di atas, terdapat beberapa notasi di antaranya sebagai berikut:

* 1. *Swimlane Heading* mewakili siapakah *user/*aktor yang melakukan aktivitas.
  2. *Starting Activity* (Pseudo) menjelaskan awal dari *activity diagram*.
  3. *Transition Arrow* berbentuk panah, menjelaskan aliran dari aktivitas.
  4. *Activity* mewakili aktivitas yang dilakukan oleh setiap aktor.
  5. *Ending Activity* (Pseudo) menjelaskan akhir dari *activity diagram.*
  6. *Synchronization Bar Split* menjelaskan adanya pemisahan aktivitas menjadi beberapa aktivitas yang paralel.
  7. *Synchronization Bar Join* menjelaskan adanya mempertemukan beberapa aktivitas yang berjalan paralel menjadi aktivitas yang awalnya terpisah.
  8. *Decision Activity* menjelaskan aktivitas yang akan dilakukan selanjutnya berdasarkan kondisi tertentu, biasanya diberikan pilihan *Yes* atau *No.*
  9. *For each loop* menjelaskan aktivitas yang sama dapat dilakukan berulang kali dimulai dari *for each item* sampai dengan *end for each item*.

Gambar 2. 5 Contoh Activity Diagram

* + - 1. ***Use Case Diagram***

*Use case diagram* adalah model UML yang digunakan untuk menunjukan aktivitas dan hubungannya terhadap *user*/aktor*.* Menurut Satzinger, Jackson, & Burd (2012), gambar 2.5 adalah contoh sederhana dari sebuah *Use case.*



Gambar 2. 6 Notasi Use Case Diagram

*Use case* memiliki notasi dasar sederhana, yakni sebagai berikut:

Aktor

Aktor direpresentasikan dengan figur *stick*, yang kemudian diberi nama sesuai dengan aktivitas yang dilakukannya (contohnya “shipping clerk”, “customer”, dan sebagainya). Aktor akan selalu berada diluar *automation boundary*. Terkadang, aktor dalam *use case* tidak selalu berupa manusia, namun bisa menjadi sistem lain atau alat yang menerima layanan dari sistem.

*Use case*

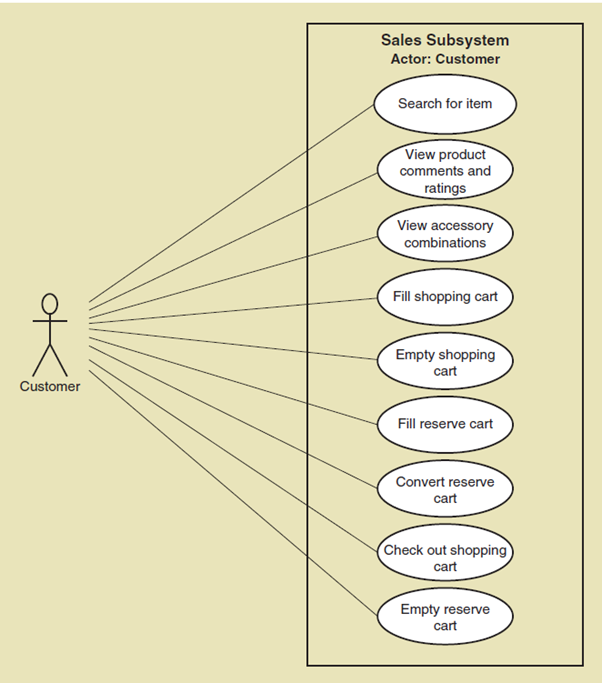
*Use case* direpresentasikan dalam bentuk oval dengan nama aktivitas di dalamnya. Sebagai contoh “ship items”, “request data”, “create account” , dan lain-lain.

*Connecting Line*

Garis penghubung atau *connecting line* berfungsi sebagai penanda adanya hubungan antara aktor dengan *Use case* tersebut.

*Automation Boundary*

*Automation boundary* mendefinisikan adanya batas antara aktor dengan sistem, yang direpresentasikan dalam bentuk persegi panjang dan di dalamnya terdapat *Use case*, sehingga interaksi aktor dengan *Use case* melewati *automation boundary.* Gambar 2.7 merupakan contoh *Use case* diagram dari subsistem penjualan.



Gambar 2. 7 Contoh Use Case Diagram

Pada penelitian ini, kami menggunakan relasi *includes* pada *Use case* *diagram*. Relasi *includes* biasa ditulis “<<*includes*>>”, yang menurut Satzinger, Jackson, & Burd (2012), “<<*includes*>>” digunakan sebagai relasi *Use case* tambahan ke sebuah *Use case* dimana *Use case* yang ditambahkan memerlukan *Use case* ini untuk menjalankan fungsinya.

Cara membaca relasi *includes* dalam *Use case* ini ialah *fill shopping cart* (mengisi keranjang belanja) termasuk (*includes*) mencari produk (*search for item*), melihat rating dan komentar produk (*view product comments and ratings*) serta melihat aksesori kombinasi (*view accessory combinations*).

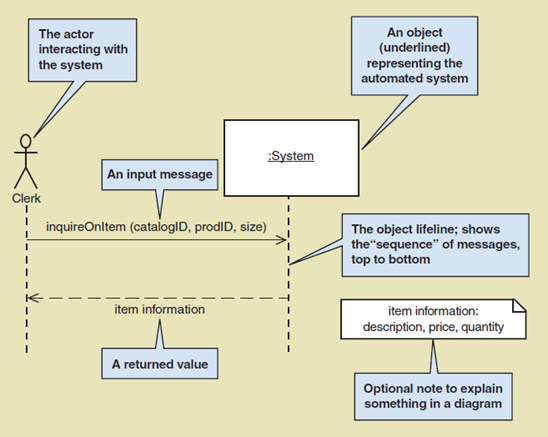
Selain relasi <<*includes*>>, penelitian ini juga menggunakan relasi “<<*extend*>>”. Relasi “<<*extend*>>” digunakan ketika *Use case* yang ditambahkan dapat berdiri sendiri tanpa harus melalui proses yang lain.

Bergantung pada kebutuhan analis, *Use case* dapat dideskripsikan atau bisa disebut dengan *Use case* *description*. Menurut Satzinger, Jackson & Burd (2012), *Use case* *description* merupakan penjelasan terperinci mengenai proses dari suatu *Use case* atau bisa disebut juga sebagai daftar kasus penggunaan diagram *Use case* yang memberikan gambaran dari semua penggunaan kasus untuk sistem. *Use case* *description* dibedakan menjadi 2 (dua), yaitu:

1. *Brief Use case Description*, yaitu digunakan untuk *Use case* yang sederhana, khususnya sistem yang dikembangkan untuk aplikasi kecil yang mudah dimengerti.
2. *Fully Developed Description*, adalah metode paling formal untuk mendokumentasikan *Use case* karena menjelaskan secara rinci setiap proses dalam *Use case* diagram. Gambar 2.8 adalah contoh *fully developed use case description* dari membuat akun pelanggan.
   * + 1. ***System Sequence Diagram***

Dalam pendekatan *object-oriented*, alur informasi dilakukan melalui pengiriman pesan baik ke dan dari aktor atau bolak-balik antara objek internal. *System Sequence Diagram* digunakan untuk mendeskripsikan alur informasi ini baik ke dalam maupun keluar sistem. Oleh karena itu, SSD adalah tipe diagram interaksi dimana SSD mendokumentasikan *input* dan *output* serta mengidentifikasi interaksi antara aktor dan sistem.

Terdapat sejumlah simbol yang digunakan dalam *system sequence diagram*.



Gambar 2. 8 Notasi System Sequence Diagram

Pada gambar 2.8, terdapat beberapa notasi diantaranya sebagai berikut:

1. Aktor (*Actor*)

Pada gambar 2.8, aktor bertugas sebagai *Clerk* yang memiliki interaksi dengan sistem.

1. Sistem (*System*) adalah objek yang mewakili sistem otomatis.
2. Garis putus-putus vertikal (*Lifeline/Object Lifeline*)

Garis putus-putus vertikal berfungsi sebagai pembatas antara aktor dan sistem yang ada.

1. Panah utuh (*Solid Arrow)*

Panah utuh yang berada diantara dua garis *lifeline* berfungsi sebagai penjelas mengenai pesan-pesan atau fungsi yang dijalankan oleh aktor ke sistem. Pesan ini dinamakan *input message*, dimana mewakilkan pesan yang dikirim oleh aktor ke sistem.

1. Panah putus-putus (*Dashed Arrow*)

Pada gambar 2.8, terdapat panah putus-putus dari sistem yang mengarah ke aktor. *Dashed arrow* ini berfungsi sebagai penjelas dari tanggapan sistem ke actor.

1. Pesan (*Message*)

*Message* atau pesan berfungsi sebagai penjelas tujuan dari data yang dikirim.

1. *Input Message*

*Input message* ialah pesan dari aktor ke sistem. Contoh dari *input message* (lihat gambar 2.8) adalah “inquireOnItem (catalogID, prodID, size)”. *Input message* harus memiliki nama aktivitas serta *class* yang memiliki hubungan pada aktivitas tersebut.

1. *Return Message*

*Return Message* ialah pesan timbal balik ketika sistem merespon balik *input message* dari aktor. Pada gambar 2.9, contoh *return message* ialah “item information”.

1. *Optional note*

*Optional note* ialah keterangan yang bersifat tidak wajib/opsional, dimana memiliki fungsi untuk menjelaskan lebih rinci tentang hal-hal terkait pesan/*message* pada diagram.

* + - 1. ***Class Diagram***

*Class* adalah kategori atau klasifikasi yang digunakan untuk menggambarkan kumpulan object. Setiap *object* memiliki *class*. Contohnya adalah Apel, Pisang, dan Jeruk adalah object yang termasuk dalam *class* Buah. *Class* yang menggambarkan *object* dalam *domain* masalah disebut *domain classes. Domain classes* memiliki atribut dan asosiasi.

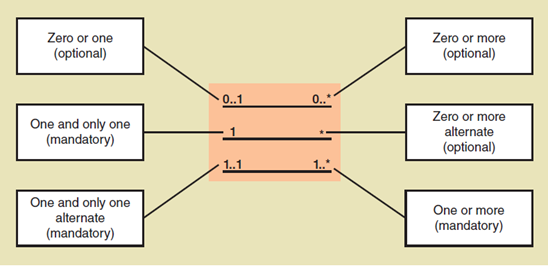
*Class diagram* digunakan untuk menunjukkan *object* sebuah *class* pada sistem. Salah satu tipe pada UML *class diagram* yang menunjukkan *object* pada *domain* masalah *user* disebut *Domain Model Class Diagram*. Jenis lain dari *Class Diagram* adalah *Design Class Diagram* yang digunakan dalam merancang *class* pada perangkat lunak.

Pada *Class Diagram,* persegi panjang mewakili *class* dan garis-garis yang menghubungkan antar persegi panjang menunjukkan asosiasi atau hubungan antar *class*. Gambar di bawah ini menunjukkan simbol untuk *Domain Model Class* tunggal, yaitu *Customer*. Simbol *Domain Model Class* adalah persegi panjang dengan dua bagian. Bagian atas berisi *class name* (nama kelas), dan bagian bawah daftar *class* *attributes* (atribut kelas). Nama kelas dan nama attribut ditulis menggunakan notasi *camelback* atau *camelcase*, dimana antar kata ditulis tanpa spasi atau garis bawah. Nama kelas dimulai dengan huruf kapital, nama attribut dimulai dengan huruf kecil.

Sedangkan pada *Design Model Class* juga menggunakan simbol -simbol yang dimiliki *Domain Model Class* dengan tambahan bagian ketiga, yaitu daftar *class methods*. Daftar *class methods* tidak ada pada *Domain Model Class*.

*Domain* *Model Class Diagram* juga memiliki notasi *multiplicity* untuk menghubungkan antar *class*, yaitu:

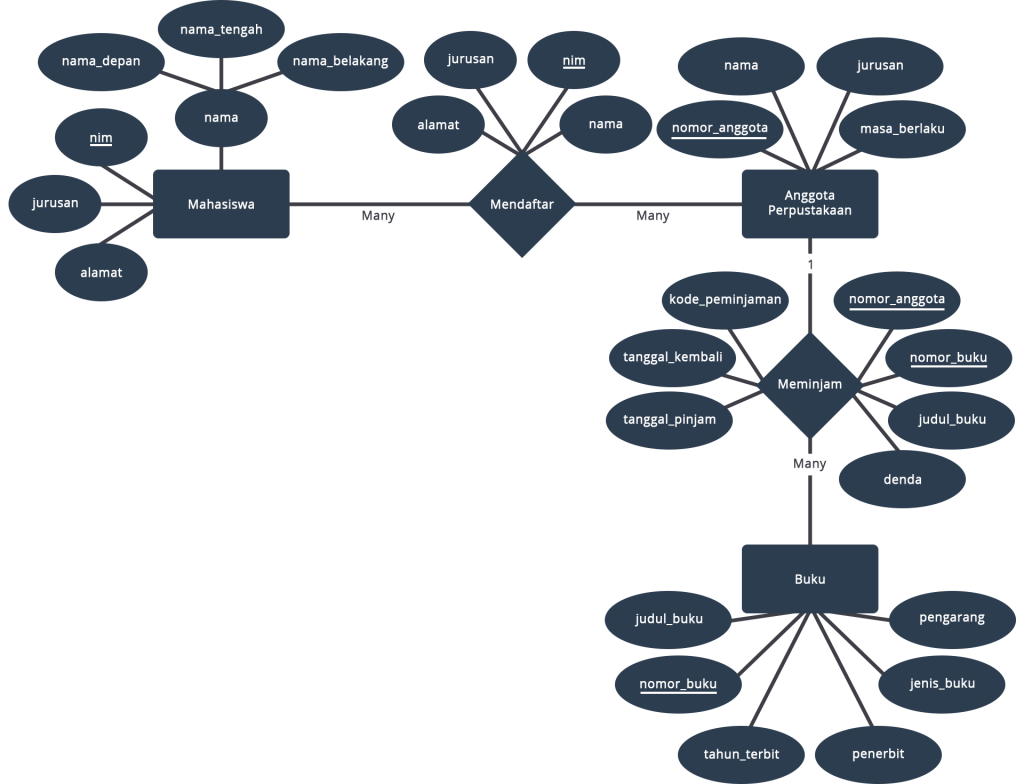
1. *Zero or one* (tidak wajib): boleh nol atau satu.
2. *One and only one* (wajib): wajib satu dan hanya satu.
3. *One and only one alternate* (wajib): wajib satu dan hanya satu alternatifnya.
4. *Zero or more* (tidak wajib): boleh nol atau lebih.
5. *Zero or more alternate* (tidak wajib): boleh nol atau lebih alternatifnya.



Gambar 2. 9 Notasi Multiplicy

* + 1. ***Entity Relationship Diagram***

ERD (Entity-Relationship Diagram) atau Entity-Relationship Diagram adalah diagram yang digunakan untuk mendesain database dan merinci hubungan atau hubungan antara objek atau entitas dan atributnya. Dengan menggunakan ERD, Anda dapat menggambarkan sistem database yang dihasilkan dengan cara yang lebih terstruktur dan bersih.



Gambar 2. 10 Contoh ERD

ERD sendiri digunakan dalam desain database dan sering digunakan untuk debugging database ketika ada masalah dengan database. Men-debug database tidak mudah. Terutama jika database bermasalah berisi banyak tabel dan Anda perlu menulis SQL yang kompleks. Menulis skema database menggunakan ERD memudahkan untuk menemukan dan memecahkan masalah dengan database Anda.

* + 1. **Pengujian Sistem**

Pada penelitian ini, kami menggunakan pengujian sistem dengan metode *black-box testing*. Menurut Pressman (2010), *Black-box Testing* adalah pengujian yang fokus pada kebutuhan fungsional pada sebuah perangkat lunak. Dengan melakukan pengujian *black-box* dapat diperoleh kumpulan input yang akan menguji seluruh kebutuhan fungsional sebuah aplikasi. Pengujian *black-box* bukan alternatif untuk pengujian *white-box*, namun bersifat sebagai pendekatan lain yang digunakan untuk melihat permasalahan yang berbeda dengan metode pengujian *white-box*. Jadi ketika pembuatan sistem telah selesai, maka akan dilakukan pengujian terkait setiap fungsi yang ada pada sistem, apakah fungsi yang dibuat sesuai dengan validitas yang ditentukan atau terdapat *error*, sehingga validitas sistem tidak dijalankan.

* + 1. ***Software Tools***
       1. ***PHP***

PHP atau *Hypertext Preprocessor* adalah bahasa pemrograman yang berjalan pada *server side scripting* dan bersifat *open source* (sumber terbuka). PHP ini sering digunakan dalam pembuatan aplikasi berbasis website yang berjalan secara dinamis, sehingga dapat terintegrasi dengan basis data (*database*). Karena PHP tergolong dalam bahasa pemrograman berbasis *server-side,* maka script yang digunakan akan diproses oleh server. Jenis server yang sering dipakai adalah Apache, Nginx, dan LiteSpeed.

PHP diciptakan oleh seorang programmer bahasa C bernama Rasmus Lerdorf dengan tujuan untuk mencatat jumlah pengunjung yang masuk pada halaman beranda (*homepage*) Web pribadinya. Pada tahun 1995, PHP/FI (*Form Interpreter*) dirilis dengan kemampuan untuk membangun tampilan aplikasi web (*web app*), melakukan pemrosesan form, serta mendukung (*supporting*) data pada MySQL. Dengan antusias dan permintaan dari pengguna internet, akhirnya pada pertengahan tahun 1997, Rasmus Lerdorf menyerahkan pengembangannya kepada tim developer yang terdiri dari Zeev Suraski dan Andi Gutmans. Sehingga, lahirlah versi PHP 3.0 (PHP: Hypertext Preprocessor 3) yang memiliki dukungan luas terhadap *database* (Oracle dan MySQL).

Seiring berkembangnya teknologi informasi dan penggunaan website yang telah mendukung HTML5, maka PHP terus melakukan inovasi dan pengembangan . PHP mengalami beberapa peningkatan versi, dimulai dari PHP versi 4.0 hingga yang terbaru PHP versi 8.0 yang memiliki fitur dan fungsionalitas yang lebih optimal. Dengan fitur yang lebih *advanced*, sehingga bahasa pemrograman ini mampu dan dapat diandalkan dalam pembuatan website dengan skala dan trafik yang besar dan kompleks.

Sintaks Dasar PHP

Dalam penulisan script PHP diawali dengan **<?php** dan ditutup dengan **?>,** sebagai contoh :

<?php

echo “Hello World !”;

?>

Setiap akhir sintaks PHP diakhiri dengan tanda ; (titik koma). Dan untuk menyimpan file PHP maka file akan disimpan dengan ekstensi .php .

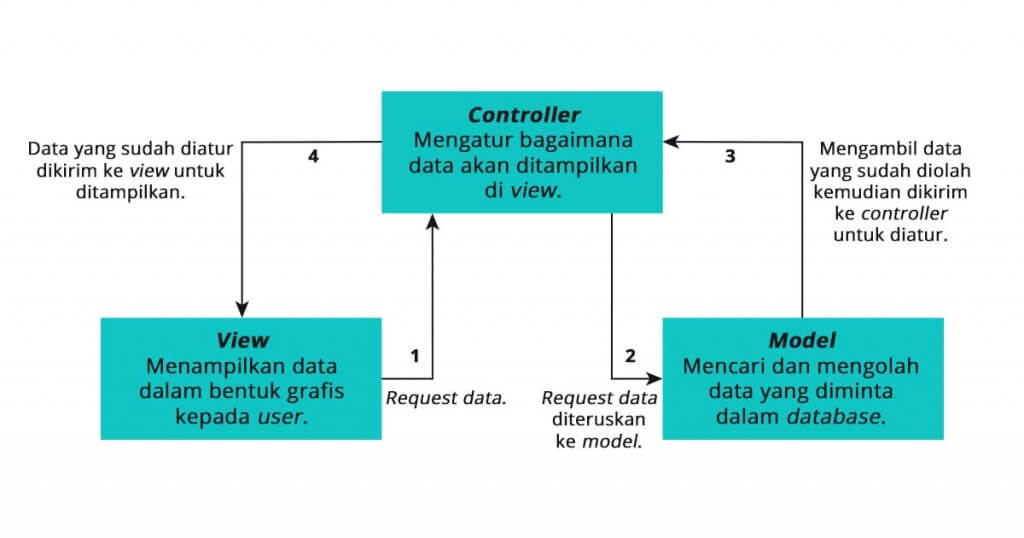
* + - 1. ***Code Igniter***

Framework atau kerangka kerja adalah kumpulan dari fungsi, prosedur, class dan juga instruksi yang tergabung dalam wadah khusus, guna memudahkan developer melakukan pemrograman. Umumnya, framework ini akan membantu para developer untuk melakukan pemrograman secara efektif dan lebih mudah, dari segi waktu dan juga pekerjaan yang dilakukan.

CodeIgniter adalah salah satu framework terbaik yang biasa digunakan oleh developer PHP. Jenis framework ini banyak digunakan karena sudah terbukti membantu perkembangan segala jenis website berbasis PHP dengan lebih mudah dan juga efektif. CodeIgniter juga memiliki library yang lebih luas dibandingkan framework lainnya. Selain itu, framework ini juga terkenal ringan sehingga tidak memakan banyak resources website.

Framework CodeIgniter pertama kali diperkenalkan pada tahun 2006 oleh Rick Ellis. Sejak diluncurkan, framework ini terus mengalami perkembangan dari segi fitur dan juga fungsi. Sampai saat ini, framework CodeIgniter sudah memasuki versi v4.1.3 yang dibuat untuk digunakan pada PHP 7.3 ke atas (termasuk PHP 8.0). framework CodeIgniter bersifat open-source. Ini bisa dengan mudah mengunduh dan menggunakan framework ini secara gratis.

Framework ini menggunakan model MVC (model, view, dan controller) yang sangat cocok digunakan untuk membangun website dinamis dengan lebih cepat dan efektif.



Gambar 2. 11 Konsep MVC

Berikut ini adalah penjelasan singkat akan ketiga komponen tersebut:

1. **Model** — komponen ini berhubungan langsung dengan database, umumnya digunakan untuk memasukan, memanipulasi data (insert, update, delete, search), dan menangani validasi controller. Komponen ini tidak berhubungan langsung dengan bagian view.
2. **View** — komponen yang berhubungan dengan semua data terkait dengan file template HTML yang diatur oleh controller. View juga berfungsi untuk merepresentasikan data kepada user. Komponen ini tidak berhubungan langsung dengan view.
3. **Controller** — komponen yang menghubungkan komponen model dan view, komponen ini juga berfungsi untuk menerima request dan data dari user.

CodeIgniter juga memiliki fitur-fitur berguna yang membantu developer membuat sebuah website. Berikut adalah beberapa fitur utama yang ada pada framework:

1. Kompatibel dengan banyak jenis database
2. Menyediakan query builder support.
3. CodeIgniter bersifat Independent.
4. Mengamankan website Anda dari cross site scripting.
5. Menyediakan validasi form/data dan juga session management.
   * + 1. ***MySQL***

Menurut Adi Nugroho (2011) MySQL (My Structured Query Language) adalah: “ Suatu sistem basis data relation atau Relational Database management

System (RDBMS) yang mampu bekerja secara cepat dan mudah digunakan MySQL juga merupakan program pengakses database yang bersifat jaringan, sehingga dapat digunakan untuk aplikasi multi user (banyak pengguna). MySQL didistribusikan gratis dibawah lisensi GPL (General Public License). Dimana setiap program bebas menggunakan MySQL namun tidak bisa dijadikan produk turunan yang dijadikan closed source atau komersial”.

MySQL salah satu software yang tergolong ke dalam database server dan bersifat open source. Open source menyatakan bahwa software ini di lengkapi dengan source code (kode yang di pakai untuk membuat MySQL) yang kode nya dijalankan secara langsung dalam sistem operasi dan bisa di peroleh dengan cara mengunduh di internet secara gratis. Selain itu MySQL juga bersifat multiplatform yaitu dapat di jalankan pada berbagai sistem operasi.

Dalam bahasa SQL pada umumnya informasi tersimpan dalam table-tabel yang secara logik merupakan struktur dua dimensi terdiri dari baris (record) dan kolom (field). Sedangkan dalam sebuah database dapat terdiri dari beberapa table. Pengaksesan data dalam database dapat di lakukan dengan mudah melalui Structure Query Languange. Data dalam database dapat di akses melalui aplikasi non web (misalnya dengan Visual Basic) maupun aplikasi web (misalnya dengan PHP).

* + - 1. ***HTML***

HTML (*Hypertext Markup Language*) adalah sekumpulan simbol-simbol atau tag-tag yang dituliskan dalam sebuah *file* yang digunakan untuk menampilkan halaman pada *web browser*. HTML merupakan bahasa pemrograman *web* yang memiliki aturan tertentu dalam menuliskan kode-kodenya sehingga *browser* dapat menampilkan informasi.

Dokumen HTML mirip dengan dokumen tulisan biasa. Namun dalam dokumen HTML, sebuah tulisan dapat memuat instruksi yang ditandai dengan kode atau lebih dikenal dengan tag tertentu. Misalnya, jika ingin membuat tulisan ditampilkan menjadi tebal, penulisan dalam dokumen HTML dilakukan dengan cara mengaktifkan instruksi cetak tebal, kemudian diikuti tulisan yang ingin ditebalkan. Setelah itu, diakhiri dengan tanda untuk menonaktifkan cetak tebal tersebut.

Contoh skrip HTML

<html>

<head>

<title>Judul</title>

</head>

<body>

<p><b>Ini halaman web saya</b></p>

</body>

</html>

* + - 1. ***CSS***

CSS (*Cascading Style Sheet*) adalah sekumpulan kode-kode yang dgunakan untuk menghias dan mengatur gaya tampilan halaman web supaya lebih elegan dan menarik. CSS digunakan oleh *programmer* untuk menentukan warna, tata letak *font*, dan semua aspek lain dari tampilan dokumen di halaman web mereka.

CSS biasanya dikaitkan dengan HTML, karena keduanya saling melengkapi. HTML digunakan untuk membuat struktur, atau konten dari halaman web. Sedangkan, CSS digunakan untuk mengatur tampilan dari halaman web tersebut.

Contoh skrip CSS

<style type="text/css">

.jenisfont

{ font-family: "Aharoni";}

</style>

* 1. **Teori Khusus**

### 2.2.1 Pengertian Dosen

Menurut UU No. 14 Tahun 2005 Pasal (1): Dosen adalah pendidik profesional dan ilmuwan dengan tugas utama mentransformasikan, mengembangkan, dan menyebarluaskan ilmu pengetahuan, teknologi, dan seni melalui pendidikan, penelitian, dan pengabdian kepada masyarakat.

Sementara itu, menurut Permenristekdikti No.2 Tahun 2016:

* + - 1. Dosen adalah pendidik profesional dan ilmuwan dengan tugas utama mentransformasikan, mengembangkan, dan menyebarluaskan Ilmu Pengetahuan dan Teknologi melalui Pendidikan, Penelitian, dan Pengabdian kepada Masyarakat.
      2. Dosen Tetap adalah dosen yang bekerja penuh waktu yang berstatus sebagai tenaga pendidik tetap pada satuan pendidikan tinggi tertentu.
      3. Dosen Tidak Tetap adalah dosen yang bekerja paruh waktu yang berstatus sebagai tenaga pendidik tidak tetap pada satuan pendidikan tinggi tertentu.
      4. Dosen dengan perjanjian kerja adalah dosen yang direkrut dengan perjanjian kerja minimal 2 (dua) tahun dan dapat diperpanjang sesuai kebutuhan.

### 2.2.2 Pengertian Kedudukan Dosen

Menurut UU No. 14 Tahun 2005 Pasal (3): Dosen mempunyai kedudukan sebagai tenaga profesional pada jenjang pendidikan tinggi yang diangkat sesuai dengan peraturan perundang-undangan. Pengakuan kedudukan dosen sebagai tenaga profesional sebagaimana dimaksud pada ayat (1) dibuktikan dengan sertifikat pendidik.

### 2.2.3 Tugas dan Kewajiban Dosen

Menurut UU No. 14 Tahun 2005 Pasal (60) Dalam melaksanakan tugas keprofesionalan, dosen berkewajiban:

melaksanakan pendidikan, penelitian, dan pengabdian kepada Masyarakat

merencanakan, melaksanakan proses pembelajaran, serta menilai dan mengevaluasi hasil pembelajaran

meningkatkan dan mengembangkan kualifikasi akademik dan kompetensi secara berkelanjutan sejalan dengan perkembangan ilmu pengetahuan, teknologi, dan seni.

* 1. ***Literature Review***

**Supanji Setyawan, Nuwun Priyono, Chaidir Iswanaji, “Development Model of E-Budgeting and E-Reporting System on The Management of Village Fund Finance”, Jurnal Dinamika Akuntansi, 2017**

Pengelolaan anggaran desa masih belum tertata dengan baik dikarenakan ditemukan banyak penyalahgunaan dana. Hal ini banyak terjadi di lapangan, sehingga menyebabkan beberapa kendala seperti tidak terwujudnya transparansi pengelolaan dana dan akuntabilitas desa. enyusunan e-budgeting dalam pengelolaan anggaran oleh Pemerintah Kabupaten Magelang membantu meningkatkan efisiensi dalam merealisasikan anggaran desa dengan cara membuat proses kegiatan anggaran, yang dimulai dari perencanaan, budgeting, hingga pengendalian anggaran, sehingga pengedalian anggaran menjadi lebih cepat dan dapat mengurangi biaya yang dikeluarkan oleh perangkat Desa Balesari di dalam mencapai realisasi anggaran. Aplikasi Siskeudes akan membantu aparatur desa untuk mengelola anggaran desa dalam bentuk e-budgeting dan e-reporting. Siskeudes akan membantu aparatur desa dalam mengelola anggaran desa dengan melakukan pencatatan dan pelaporan, sehingga akan meningkatkan kinerja dan akuntabilitas anggaran desa. Sumber Daya Manusia yang masih rendah menjadi kendala bagi Desa Balesari. Karena banyak aparatur desa yang tidak memiliki latar belakang akuntansi, sehingga hal ini menimbulkan tingginya tingkat kesalahan di dalam berbagai proses pengelolaan dana desa.

**Galih Setyo Pambudi, Sriyanto, Ary Arvianto, “Rancang Bangun Sistem Informasi Manajemen Aset Berbasis Web Untuk Optimalisasi Penelusuran Aset Di Teknik Industri Undip”, Jurnal Teknik Industri, Universitas Diponegoro, 2016**

Kelengkapan data sebuah aset di sebuah instansi sangatlah penting, karena akan digunakan sebagai gambaran di masa mendatang. Ketidaktertiban pencatatan asset menjadi kendala di PSTI dan sebagian jurusan lain di Fakultas Teknik UNDIP.

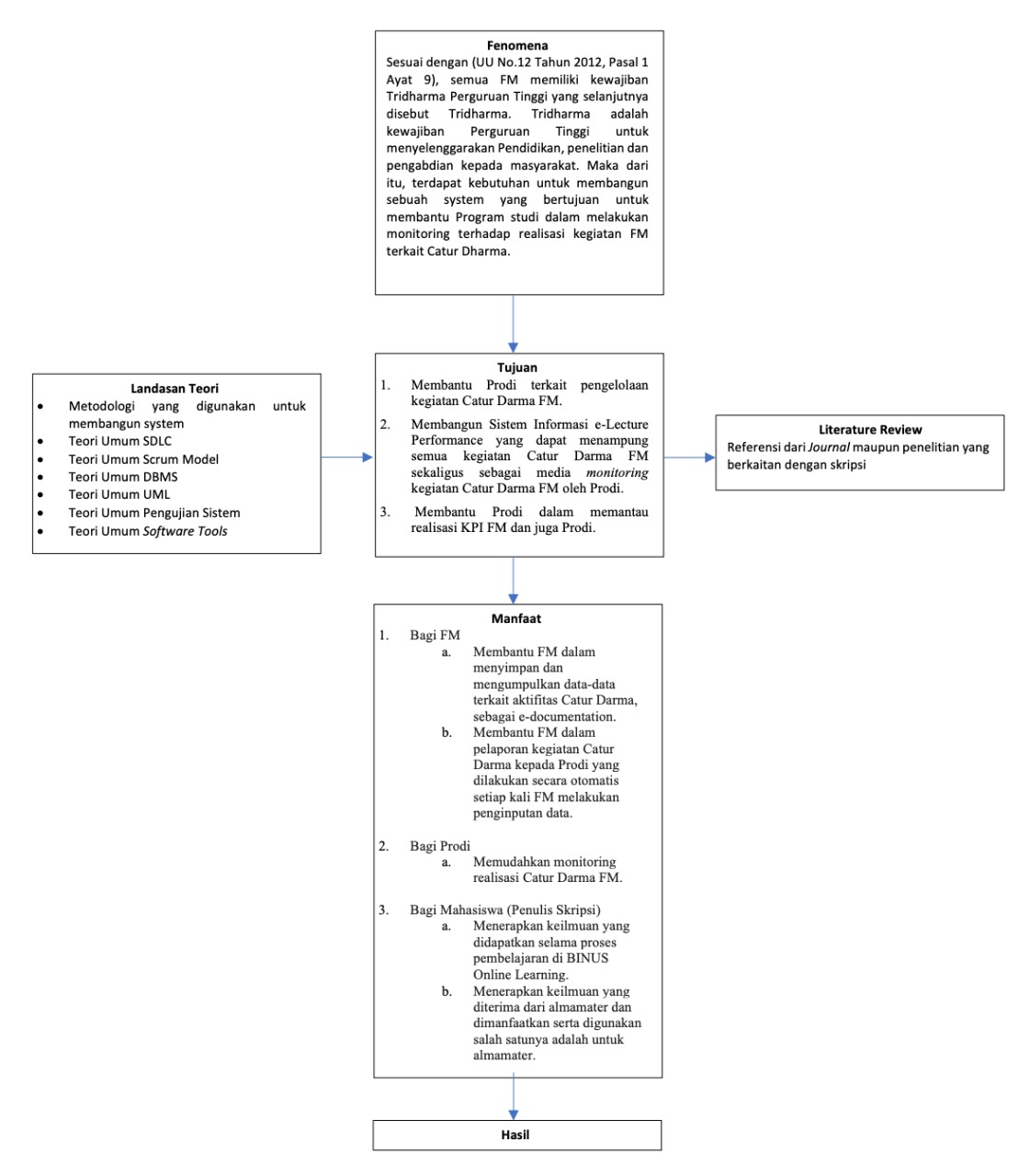
Sistem informasi manajemen aset yang dirancang digunakan untuk mengoptimalisasikan penelusuran aset pada PSTI. Rancangan sistem ini diharapkan dapat melengkapi berbagai kekurangan pada sistem yang lama dan dapat menciptakan kondisi tertib dokumen dan tertib administrasi pengelolaan aset.

Web SIMA berfungsi untuk membantu PSTI dalam melakukan identifikasi proses manajemen aset yang saat ini sedang berjalan serta merancang proses bisnis sistem manajemen aset baru yang lebih efisien.

Pendataan terkait barang habis pakai sehingga pegawai selalu menganggap sisa habis pada akhir bulan, padahal belum pasti pada akhir bulan semua stok barang habis pakai itu habis.

* 1. ***Kerangka Pikir***

Kerangka pikir pada penelitian ini berisi alur penggambaran proses analisis dan pengembangan sistem dari awal sampai akhir secara keseluruhan. Pada gambar 2.12, terdapat kerangka pikir penelitian kami.



Gambar 2. 12 Kerangka Pikir