# BAB 2 LANDASAN KEPUSTAKAAN

## 2.1 Kajian Pustaka

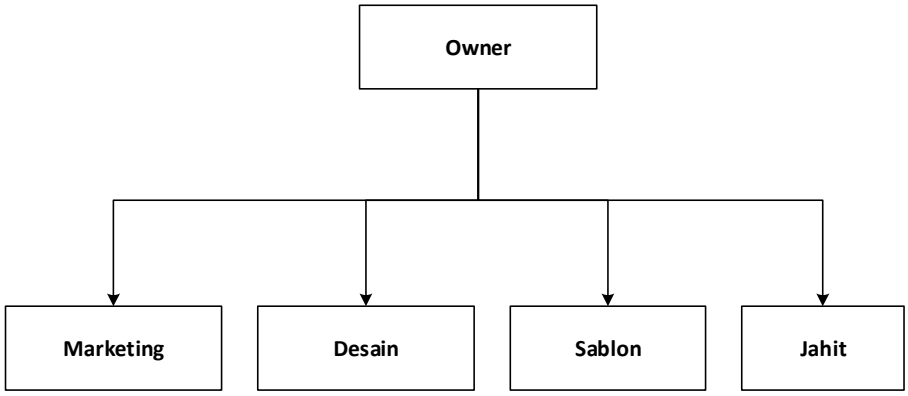
Penelitian mengenai perancangan sistem informasi persediaan barang pada Universitas Stikubank Semarang yang dilakukan oleh Siti Munawaroh. Menurut Siti, banyak permasalahan yang terdapat pada sistem persediaan alat tulis kantor Universitas Stikubank Semarang yaitu masih menggunakan program aplikasi excel, sehingga waktu yang dibutuhkan (akreditasi, ujian formatif ataupun sumatif) masih kesulitan memperoleh laporan mengenai persediaan alat tulis kantor, sistem informasi persediaan barang yang dibuat oleh Siti akan menghasilkan data barang, data suplier, data kegiatan, barang masuk, pengeluaran ATK, laporan persediaan barang ATK, laporan data suplier, daftar jenis kegiatan, dan laporan barang masuk (Munawaroh, 2006). Perbedaan penelitian yang dilakukan oleh Siti dengan penulis adalah penelitian yang dilakukan oleh Siti masih dalam tahap perancangan dan tidak melakukan pengelompokan setiap periode tertentu, sedangkan pada penelitian yang dilakukan penulis melakukan pengembangan sistem informasi yang terdapat pengelompokan setiap periode tertentu.

Perbandingan antara 3 *platform* model pengembangan perangkat lunak telah dilakukan oleh Adel Alshamrani dan Abdullah Bahattab dengan judul "*A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model*". Telah didapatkan bahwa *Waterfall Model* memiliki beberapa keuntungan yang diantaranya (Alshamrani & Bahattab, 2015) (1) mudah dimengerti dan diimplementasikan; (2) banyak digunakan dan diketahui; (3) desain sistem jelas sebelum dibuat perangkat lunak; (4) susunan yang sistematis sehingga sangat mudah untuk diimplementasikan; (5) meminimalkan *overhead* perencanaan; (6) dan fase diproses dan diselesaikan satu per satu. Dan menurut (Alshamrani & Bahattab, 2015) *Waterfall Model* baik digunakan ketika kebutuhan sudah sangat diketahui, jelas, dan tetap. Pada penelitian yang dilakukan penulis telah didapatkan bahwa kebutuhan dari pengguna sudah sangat jelas sehingga penulis akan menggunakan model *Waterfall* untuk pengembangan sistem informasi.

Terdapat penelitian yang berkaitan dengan sistem informasi penjualan berbasis web pada Toko Ali Computer yang dilakukan oleh Ikhtiar Rizki. Menurut Ikhtiar, Proses pemasaran masih kurang maksimal karena hanya melakukan penjualan langsung kepada konsumen yang datang ke toko sehingga konsumen yang berada di luar kota atau jauh dari lokasi tempat penjualan akan merasa kesulitan untuk memesan atau mengetahui produk yang dijual. Penelitian yang dilakukan mengunakan model pengembangan perangkat lunak *Waterfall* dikarenakan model tersebut bersifat sekuensial dan sistematis. Penelitian yang dilakukan penulis menggunakan model *Waterfall* untuk membangun sistem informasi persediaan barang, harga pokok produksi, dan transaksi penjualan pada Son Screen Printing Sidoarjo.

## 2.2 Son Screen Printing Sidoarjo

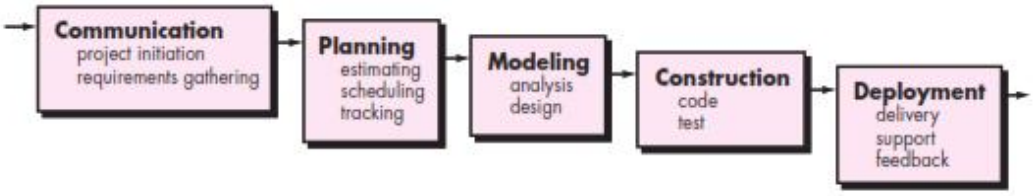
Son Screen Printing merupakan sebuah peruahaan yang bergerak dibidang industri yang memproduksi pakaian sesuai dengan permintaan pelanggan, perusahaan ini telah memiliki banyak suplier dan pelanggan. Banyak toko-toko pakaian yang besar maupun kecil membeli pakaian pada Son Screen Printing Sidoarjo untuk dijual kembali di toko mereka. Perusahaan ini telah berdiri selama 5 tahun yaitu pada tahun 2012. Berdasarkan hasil wawancara kepada pemilik perusahaan (Owner) bernama Muhammad Mukhlason, S.Kom. memiliki jumlah pegawai bagian marketing yaitu 2 orang, desain 2 orang, sablon 8 orang, dan jahit 8 orang. Jadi, total pegawai yang dimiliki perusahaan adalah 20 orang. Berikut ini merupakan struktur organisasi dari Son Screen Printing Sidoarjo yang ditunjukkan pada Gambar 2.1.




**Gambar 2.1 Struktur Organisasi**

## *2.3* Pengembangan Perangkat Lunak dengan Model *Waterfall*

*Waterfall* merupakan model klasik atau lama yang bersifat sistematis atau berurutan dalam membangun suatu perangkat lunak (Pressman, 2010). Berikut ini merupakan fase-fase dalam model *Waterfall* menurut Presman ditunjukkan pada Gambar 2.2.




**Gambar 2.2 Model *Waterfall***

Sumber : (Pressman, 2010)

1. Komunikasi (*Communication*)

Pada langkah ini merupakan analisa terhadap kebutuhan perangkat lunak dan tahap untuk mengadakan pengumpulan data dengan melakukan pertemuan dengan klien, maupun pengumpulan data-data tambahan lainnya, baik yang terdapat pada artikel, internet, maupun dari jurnal (Mariz, et al., 2017).

2. Perencanaan (*Planning*)

Proses perencanaan merupakan langkah lanjutan dari proses komunikasi (analisis kebutuhan). Tahapan ini akan menghasilkan dokumen identifikasi pengguna atau bisa dikatakan sebagai data yang berhubungan secara langsung dengan keinginan user dalam pembuatan perangkat lunak, termasuk rencana yang akan dilakukan (Mariz, et al., 2017).

3. Perancangan (*Modeling*)

Proses ini akan menerjemahkan syarat kebutuhan ke sebuah perancangan perangkat lunak yang dapat diperkirakan sebelum dilakukan pengkodean (*coding*). Perancangan berfokus pada rancangan struktur data, arsitektur software, representasi antarmuka dan detail (algoritma) prosedural. Tahapan ini akan menghasilkan dokumen yang disebut *Specification Requirement Software* (Mariz, et al., 2017).

4. Pengkodean (*Construction*)

*Construction* merupakan proses membuat kode program. Pengkodean (*coding*) merupakan penerjemah desain dalam bahasa yang dikenali oleh komputer (bahasa pemrograman). Programmer akan menerjemahkan kebutuhan yang diminta oleh user. Tahapan inilah yang merupakan tahapan implementasi dalam mengerjakan suatu perangkat lunak, artinya penggunaan komputer akan digunakan secara maksimal pada tahapan ini. Setelah pengkodean dilakukan maka akan dilakukan pengujian terhadap sistem yang telah dibuat sebelumnya. Tujuan pengujian adalah menemukan kesalahan yang terdapat pada sistem untuk diperbaiki agar sesuai dengan kebutuhan pengguna (Mariz, et al., 2017).

5. Pemeliharaan (*Deployment*)

Tahapan ini bisa dikatakan akhir dalam pembuatan sebuah perangkat lunak atau sistem. Setelah melakukan analisa, desain, dan pengkodean maka sistem yang sudah jadi akan digunakan oleh pengguna. Kemudian perangkat lunak yang telah dibuat harus dilakukan pemeliharaan secara berkala agar kinerja sistem tetap baik (Mariz, et al., 2017).

## 2.4 Sistem Informasi

Sistem Informasi merupakan gabungan dari manusia, alat teknologi, media, prosedur, dan pengendalian yang bertujuan untuk menata jaringan komunikasi yang penting. Proses atas transaksi-transaksi tertentu dan rutin, membantu manajemen dalam menyediakan dasar pengambilan keputusan yang tepat (Rizki, et al., 2014).

Sistem informasi merupakan suatu kesatuan elemen yang saling berinteraksi atau berhubungan secara berurutan dan teratur untuk menghasilkan bentuk aliran informasi yang dapat mendukung pembuatan keputusan dan melakukan manajemen (Rahadi, et al., 2014).

Sistem informasi adalah kombinasi dari *hardware*, *software*, dan jaringan telekomunikasi yang dibangun dan digunakan orang untuk mengumpulkan, membuat, dan mendistribusikan data yang bermanfaat. Namun, cara terbaik untuk menggunakan sistem informasi adalah mendukung strategi organisasi dengan cara yang memungkinkan perusahaan untuk memperoleh atau mempertahankan keunggulan kompetitif dibandingkan pesaingnya. (Valacich & Schneider, 2012).

Tujuan sistem informasi adalah untuk mendapatkan informasi yang benar kepada orang yang tepat pada waktu yang tepat dalam jumlah yang tepat dan dalam format yang tepat. Karena sistem informasi dimaksudkan untuk menyediakan informasi yang berguna, yang dimulai dengan mendefinisikan ***information*** dan dua istilah yaitu ***data items*** dan ***knowledge*** yang memiliki keterkaitan yang kuat (Jr. & Cegielski, 2007).

***Data items*** mengacu pada penjelasan dasar mengenai suatu hal, kejadian, aktivitas, dan transaksi yang dicatat, diklasifikasikan, dan disimpan namun tidak diatur untuk menyampaikan arti tertentu. *Data items* bisa berupa angka, huruf, suara, atau gambar. Contoh *data item* adalah nilai siswa di kelas dan jumlah jam kerja seorang karyawan pada minggu tertentu. ***Information*** mengacu pada data yang telah diatur sehingga memiliki arti yang lebih dan bernilai bagi penerimanya. Misalkan, nilai rata-rata IPK adalah data, namun nama siswa ditambah dengan IPKnya adalah informasi. ***Knowledge*** terdiri dari data dan / atau informasi yang telah disusun dan diproses untuk menyampaikan pemahaman, pengalaman, akumulasi pembelajaran, dan keahlian karena bertujuan untuk masalah bisnis yang sedang dijalankan. Misalkan, rekrutmen perusahaan di universitas telah menemukan bahwa mahasiswa dengan nilai rata-rata di atas 3,0 memiliki keberhasilan dalam pelajaran manajemen. Berdasarkan pengalamannya, perusahaan tersebut perusahaan dapat memutuskan untuk mewawancarai mahasiswa dengan IPK di atas 3,0 (Jr. & Cegielski, 2007).

Sehingga dapat disimpulkan bahwa sistem informasi adalah sebuah sistem yang dapat mengelola data baik yang sederhana maupun yang kompleks untuk dijadikan sebuah informasi yang dapat dimanfaatkan untuk pengambilan keputusan.

## 2.5 Persediaan Barang

Persediaan adalah penyimpanan bahan, baik bahan pembantu, bahan setengah jadi, bahan setengah jadi, bahan baku, maupun bahan lainnya, yang digunakan untuk kebutuhan yang akan datang. Penyimpanan ini dilakukan karena perusahaan terkadang membutuhkan bahan-bahan tersebut secara tiba-tiba, sehingga perusahaan tidak akan kesusahan dalam mendapatkan bahan yang dibutuhkan. Tidak hanya itu penyimpanan dapat terjadi juga, ketika terdapat sisa bahan ketika produksi selesai dibuat (Rahadi, et al., 2014).

Jika perusahaan termasuk dalam kelompok perusahaan manufaktur atau berskala besar berarti persediaan yang akan dikelola meliputi persediaan barang

produk yang jadi, persediaan produk yang sedang dalam proses pembuatan, persediaan bahan baku / bahan mentah, persediaan bahan penolong dan persediaan lainnya. Sedangkan jika perusahaan termasuk dalam kelompok perusahaan dagang, maka persediaan yang dikelola hanya satu macam saja yaitu persediaan barang dagangan yang dibeli dan kemudian dijual kembali (Munawaroh, 2006).

Dari pengertian tersebut, maka dapat disimpulkan bahwa pengelolaan persediaan barang tergantung dari jenis perusahaan yang sedang berjalan. Salah satu perushaan yang tidak memerlukan persediaan barang adalah lembaga pendidikan, termasuk universitas yang merupakan organisasi / perusahaan yang tidak menggunakan persediaan untuk dijual kembali ataupun diolah dan kemudian dijual kembali. Sehingga pengelolaan persediaan yang dimiliki dapat dikatakan hanya sebatas membeli dan kemudian digunakan untuk kegiatan sehari-hari (Munawaroh, 2006).

## 2.6 Harga Pokok Produksi

Harga pokok produksi berpengaruh secara langsung terhadap perhitungan harga pokok penjualan, demikian halnya dengan nilai persediaan barang baik yang jadi, maupun proses yang tercantum dalam neraca perusahaan. Jika, penjualan produk terlalu mahal dapat menyebabkan pelanggan tidak ingin memesan produk diperusahaan, namun jika penjualan produk lebih terjangkau daripada perusahaan lainnya dan tetap mendapatkan keuntungan maka pelanggan akan sering memesan produk diperusahaan (Afifah, 2012). Penentuan harga pokok produksi terbagi menjadi 2 yaitu *full costing* dan *variable costing*.

Menurut (Jatmiko, et al., 2014) *full costing* atau sering adalah metode dalam menentukan harga pokok produksi, yang membedakan seluruh biaya produksi, baik yang berperilaku tetap maupun variabel terhadap produk. Dengan demikian harga pokok produksi menurut metode *full costing* terdiri dari unsur biaya produksi sebagai berikut:

HPP = BBB + BTKL + BOV + BOT

Keterangan:

HPP : Harga Pokok Produksi

BBB : Biaya Bahan Baku

BTKL : Biaya tenaga Kerja Langsung

BOV : Biaya Overhead Variabel

BOT : Biaya Overhead Tetap

Sistem harga pokok produksi memiliki kinerja yaitu menerima data master dari setiap kebutuhan produksi dari bahan baku tenaga kerja serta faktor overhead dari perusahaan yang berasal dari masukan pengguna sistem informasi. Dari data yang diterima dihitung masukan yang akan menentukan harga pokok produksi dari setiap biaya bahan baku, biaya tenaga kerja, biaya

overhead variabel dan tetap. Setelah mengetahui rincian dari biaya bahan baku, tenaga kerja, overhead variabel, dan overhead tetap, maka sistem akan menjumlahkan dari ketiga komponen tersebut menjadi informasi harga pokok produksi.

Selain *full* costing, menurut (Komara & Sudarma, 2016), terdapat metode penentuan harga pokok produksi lainnya yaitu *variable costing* merupakan suatu metode yang digunakan untuk menentukan harga pokok produksi namun hanya memperhitungkan biaya produksi yang berperilaku sebagai variabel ke dalam harga pokok produksi. Yang digunakan untuk menentukan harga pokok produksi terdiri dari biaya bahan baku, biaya tenaga kerja langsung, dan biaya overhead variabel. Unsur harga pokok produksi menurut metode *variabel costing*. Dengan demikian harga pokok produksi menurut metode *variable costing* terdiri dari unsur biaya produksi sebagai berikut:

HPP = BBB + BTKL + BOV

Keterangan:

HPP : Harga Pokok Produksi

BBB : Biaya Bahan Baku

BTKL : Biaya tenaga Kerja Langsung

BOV : Biaya Overhead Variabel

## 2.7 Transaksi Penjualan

Penjualan merupakan suatu usaha untuk mengembangkan rencana strategis yang diarahkan pada usaha pemuasan kebutuhan dan keinginan pelanggan sehingga mendapatkan penjualan yang menghasilkan keuntungan. Penjualan merupakan sumber penghasilan suatu perusahaan, karena dari penjualan dapat diperoleh keuntungan. Suatu usaha menarik pelanggan bertujuan untuk mengetahui daya tarik mereka sehingga dapat mengetahui hasil produk yang dihasilkan sesuai dengan keinginan pelanggan. Penjualan adalah penerimaan yang diperoleh dari pengiriman barang dagangan atau dari penyerahan pelayanan dalam bursa sebagai barang pertimbangan. Pertimbangan ini dapat dalam bentuk tunai peralatan kas atau harta lainnya dalam artian yang dapat diperjualbelikan. Pendapatan perusahaan dapat diperoleh pada saat penjualan, karena terjadi pertukaran yang tentunya saling menguntungkan (Nurcahyono, 2012).

## *2.8 Bussines Processs Model Notation* (BPMN)

BPMN adalah untuk menstandardisasi model proses bisnis dan notasi dalam menghadapi berbagai notasi pemodelan dan sudut pandang yang berbeda. Dengan demikian, BPMN akan menyediakan sarana komunikasi yang sederhana untuk mengkomunikasikan informasi proses kepada pengguna bisnis lainnya, pelaksana proses, pelanggan, dan pemasok. BPMN dibatasi hanya untuk mendukung konsep pemodelan yang berlaku untuk Proses Bisnis (Object

Management Group, 2011). BPMN memiliki empat kategori elemen sebagai penunjang dalam pembuatan BPMN yang dijelaskan sebagai berikut (Object Management Group, 2011) :

### *2.8.1 Flow Object*

*Flow Object* dibagi menjadi 3, yaitu *event, activity,* dan *gateway*. Berikut penjelasannya :

1. *Event*

   *Event* merupakan sesuatu yang terjadi dan memiliki dampak dalam proses bisnis. Suatu *event* dapat berasal dari internal dan eksternal suatu proses. *Event* dibagi menjadi tiga yaitu *start event, intermediate event,* dan *end event*. Setiap proses selalu memiliki sebuah *start event* untuk menunjukan awal dari proses bisnis. Tabel 2.1 merupakan penjelasan dari simbol *event*.

**Tabel 2.1 Simbol-Simbol *Event* BPMN**

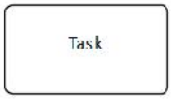
| No | Nama | Gambar | Deskripsi |
|---|---|---|---|
| 1. | *Start* |  | Mendeskripsikan dimana suatu proses dimulai |
| 2. | *Intermediate* |  | Mendeskripsikan dimana suatu terjadi diantara awal dan akhir proses. Akan mempengaruhi alur dari proses, tapi tidak akan memulai atau memberhentikan proses. |
| 3. | *End* |  | Mendeskripsikan suatu proses berakhir. |

Sumber : (Object Management Group, 2011)

2. *Activity*

   *Activity* merupakan tugas yang dilakukan dalam sebuah proses bisnis. *Activity* ditunjukan dengan kotak dengan ujung bulat dengan nama yang menjelaskan aktivitas yang dilakukan. Terdapat dua macam *activity* yaitu *task* dan *sub process*. Tabel 2.2 merupakan penjelasan dari simbol *activity.*

**Tabel 2.2 Simbol-Simbol *Activity* BPMN**

| No | Nama | Gambar | Deskripsi |
|---|---|---|---|
| 1. | *Task* |  | Merupakan aktivitas yang dilakukan pada alur proses |
| 2. | *Sub Process* |  | Merupakan sebuah aktivitas majemuk yang dimasukkan dalam proses. Aktivitas majemuk tersebut dapat dijelaskan dengan lebih detail. |

Sumber : (Object Management Group, 2011)

3. *Gateway*

   *Gateway* bertanggung jawab mengontrol bagaimana alur dari sebuah proses bisnis. Tabel 2.3 merupakan penjelasan dari simbol *gateway.*

**Tabel 2.3 Simbol-Simbol *Gateway* BPMN**

| No | Nama | Gambar | Deskripsi |
|---|---|---|---|
| 1. | *Exclusive* |  | Sebagai *divergence* : digunakan untuk membuat jalur alternatif dalam sebuah proses, tapi hanya satu yang dipilih.<br><br>Sebagai *convergence* : digunakan untuk mengabungkan jalur alternatif. |
| 2. | *Parallel* |  | Mendeskripsikan proses yang dijalankan secara bersamaan |
| 3. | *Inclusive* |  | Mendeskripsikan sebuah proses yang dipecah menjadi beberapa jalur. |
| 4. | *Complex* |  | Mendeksripsikan alur yang kompleks pada sebuah proses bisnis |

Sumber : (Object Management Group, 2011)

## 2.8.2 Connections

*Connections* adalah elemen yang menghubungkan *flow objects*. Tabel 2.4 merupakan penjelasan dari simbol *connections*.

**Tabel 2.4 Simbol-Simbol *Connections* BPMN**

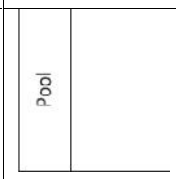
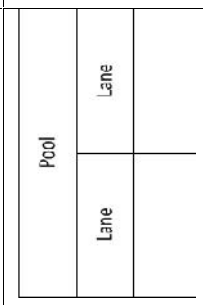
| No | Nama | Gambar | Deskripsi |
|---|---|---|---|
| 1. | *Sequence Flow* |  | Menunjukan urutan aktivitas yang dilakukan pada sebuah proses |
| 2. | *Association* |  | Menunjukan hubungan antara data, teks, artifak lain, dan *flow object* pada sebuah proses |
| 3. | *Message Flow* |  | Menunjukan alur pesan antara dua partisipan yang mampu mengirim dan menerima pesan |

Sumber : (Object Management Group, 2011)

## 2.8.3 Swimlanes

*Swimlanes* merupakan wadah grafis yang membagi suatu set aktivitas dengan aktivitas lain. Tabel 2.5 meupakan penjelasan simbol *swimlanes*.

**Tabel 2.5 Simbol-Simbol *Swimlanes* BPMN**

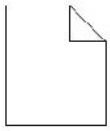
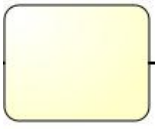
| No | Nama | Gambar | Deskripsi |
|---|---|---|---|
| 1. | *Pool* |  | Merupakan wadah yang berisi satu proses dan *sequence flow* yang menghubngkan aktivitas |
| 2. | *Lane* |  | Digunakan untuk mempresentasikan tanggungjawab aktivitas pada sebuah proses. |

Sumber : (Object Management Group, 2011)

### 2.8.4 Artifacts

*Artifacts* mempresentasikan sebuah objek diluar sebuah proses. *Artifact* dapat mempresentasikan data atau catatan yang menjelaskan sebuah proses atau dapat digunakan untuk mengelola tugas atau proses. Tabel 2.6 meupakan penjelasan simbol *artifacts*.

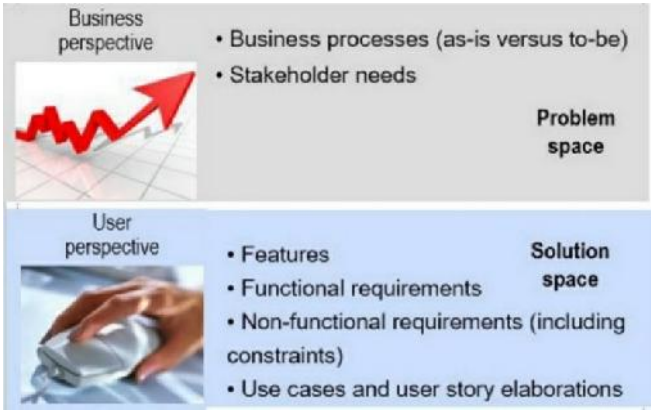
**Tabel 2.6 Simbol-Simbol *Artifacts* BPMN**

| No | Nama | Gambar | Deskripsi |
|---|---|---|---|
| 1. | *Data Object* |  | Mempresentasikan informasi yang mengalir pada sebuah proses seperti dokumen bisnis, surat, *email* dan lain-lain. |
| 2. | *Data Store* |  | Tempat dimana proses dapat membaca atau menulis data |
| 3. | *Annotation* |  | Menunjukan informasi tambahan kepada pembaca sebuah diagram BPMN. |
| 4. | *Group* |  | Memungkinkan untuk mengelompokkan elemen secara informal |

Sumber : **(Object Management Group, 2011)**

## 2.9 Levels Of Requirement

Terdapat dua tingkatan sudut pandang dalam melakukan analisis kebutuhan yang dapat dilihat pada Gambar 2.3 sebagai berikut (Mohlin, 2011):




**Gambar 2.3 *Levels of Requirements***

Sumber : (Mohlin, 2011)

### *2.9.1 Business Perspective*

*Business Perspective* adalah sudut pandang yang melihat kebutuhan pengembangan aplikasi dari sisi fungsi dan proses bisnis untuk memenuhi tujuan bisnis yang ingin dicapai. Tujuannya adalah untuk memberikan pemahaman terhadap *stakeholder* bagaimana alur proses bisnis yang ada pada perusahaan tersebut yang bertujuan untuk mendapatkan gambaran secara jelas bagaimana proses bisnis yang sudah berjalan dan bagian mana dari proses bisnis tersebut yang akan ditingkatkan kinerjanya dengan menggunakan aplikasi tersebut sehingga mendapatkan kebutuhan yang mendukung tujuan bisnis yang ingin dicapai. *Business Perspective* dapat disebut juga sebagai domain masalah.

#### *2.9.1.1 Business Process (As-Is vs To-Be)*

*Business Process as-is* menjelaskan secara keseluruhan proses bisnis yang berjalan saat ini di sebuah perusahaan atau organisasi dengan menggunakan pernyataan dan gambar. Tujuan dari analisa *Business Process as-is* untuk memperjelas dan memahami seperti apa kondisi saat ini dari proses bisnis yang ada dan bagaimana alur bisnis tersebut berjalan pada perusahaan atau organisasi (Brandenburg, 2017).

Proses analisa yang dilakukan untuk mengetahui permasalahan dan kelemahan yang ada pada proses bisnis tersebut. Permasalahan tersebut dipetakan dalam tabel analisis permasalahan, sehingga bisa diketahui bagaimana solusi dan perbaikan untuk proses bisnis tersebut. Pada tabel 2.7 merupakan cara dalam melakukan analisis permasalahan (Bittner & Spence, 2002).

**Tabel 2.7 Analisis Permasalahan**

| | |
|---|---|
| Masalah | [Mendeskripsikan masalah] |
| Mempengaruhi | [Pemangku kepentingan yang terpengaruh oleh masalah] |
| Dampak | [Dampak dari masalah] |
| Solusi | [Solusi beserta manfaatnya] |

Sumber : (Bittner & Spence, 2002)

*Business Process to-be* menjelaskan bagaimana peningkatan kinerja proses bisnis kedepannya kepada *stakeholder* perusahaan atau organisasi. Usulan yang diajukan berdasarkan apa yang sudah dilakukan pada analisa *Business Process as-is* untuk memenuhi seluruh permasalahan yang ada pada analisa *Business Process as-is.* Tujuan dari analisa *Business Process to-be* adalah untuk memberikan penjelasan dan gambaran kepada *stakeholder* perusahaan atau organisasi bagaimana usula-usulan tersebut disatukan dengan proses bisnis yang ada, apakah sudah memenuhi keinginan dari *stakeholder* yang terlibat dalam proses bisnis tersebut (Brandenburg, 2017).

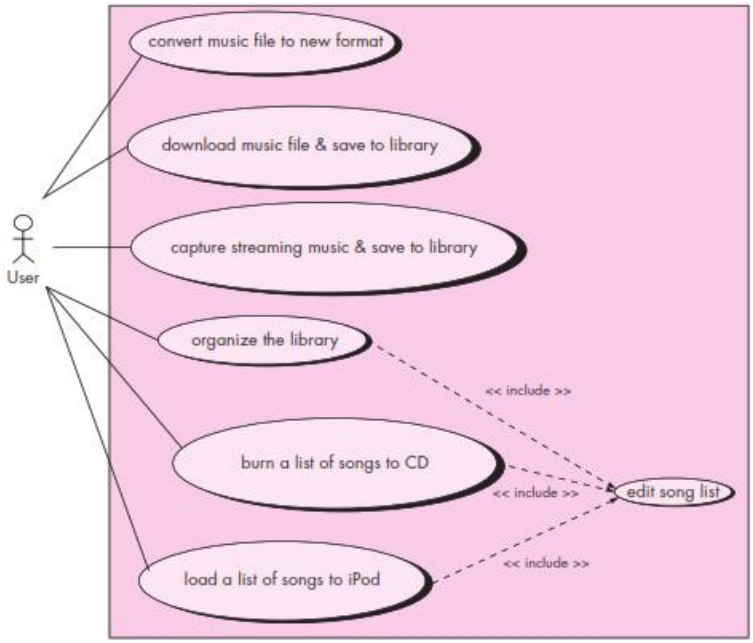
## 2.10 *Unified Modelling Language* (UML)

*Unified Modeling Language* adalah bahasa untuk menulis rancangan sebuah perangkat lunak. UML dapat digunakan untuk membangun dan mendokumentasikan kerangka perangkat lunak. Di sisi lain, dengan membangun rancangan arsitektur untuk digunakan para konstruksi perusahaan, pembangun perangkat lunak membuat UML diagram untuk membantu pengembang perangkat lunak untuk membangun perangkat lunak. Sehingga lebih mudah untuk dipahami, mudah untuk menentukan sistem dan mudah untuk menjelaskan desain sistem (Pressman, 2010).

### 2.10.1 *Use Case* Diagram

Model *use case* diagram membantu untuk menentukan fungsi dan fitur perangkat lunak dari perspektif pengguna. *Use case* menjelaskan bagaimana pengguna berinteraksi dengan sistem dengan cara mendefinisikan langkah langkah yang diperlukan untuk mencapai tujuan tertentu. *Use case* diagram merupakan gambaran dari semua *use case* dan hubungan antar setiap *use case*. Dalam use case diagram, *use case* ditampilkan dalam bentuk oval. Para aktor terhubung dengan setiap *use case* menggunakan garis. Karena use case diagram menampilkan semua use case, hal ini sangat membantu untuk memastikan bahwa semua fungsi dari sistem sudah tercakup (Pressman, 2010). Contoh use case diagram dapat dilihat pada Gambar 2.4.

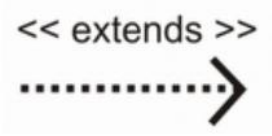
Untuk menggunakan *use case* diagram harus memperhatikan simbol-simbol yang digunakan pada Tabel 2.8 ditunjukkan seluruh simbol yang dapat digunakan ketika membuat *use case* diagram.



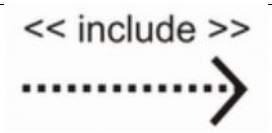

**Gambar 2.4 Contoh *Use Case Diagram***

Sumber : (Pressman, 2010)

**Tabel 2.8 Simbol-Simbol *Use Case* Diagram**

| No | Nama | Gambar | Fungsi |
|---|---|---|---|
| 1. | *Use Case* |  | Komunikasi antara aktor dan use case yang berpartisipasi pada use case atau use case memiliki interaksi dengan aktor. |
| 2. | Aktor |  | Relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan dapat berdiri sendiri walau tanpa use case tambahan itu; mirip dengan |
| 3. | *Association* |  | prinsip inheritance pada pemrograman berorientasi objek, biasanya use case tambahan memiliki nama depan yang sama dengan use case yang ditambahkan. |
| 4. | *Extend* |  | Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah use case dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya |

**Tabel 2.8 Simbol-Simbol *Use Case* Diagram (lanjutan)**

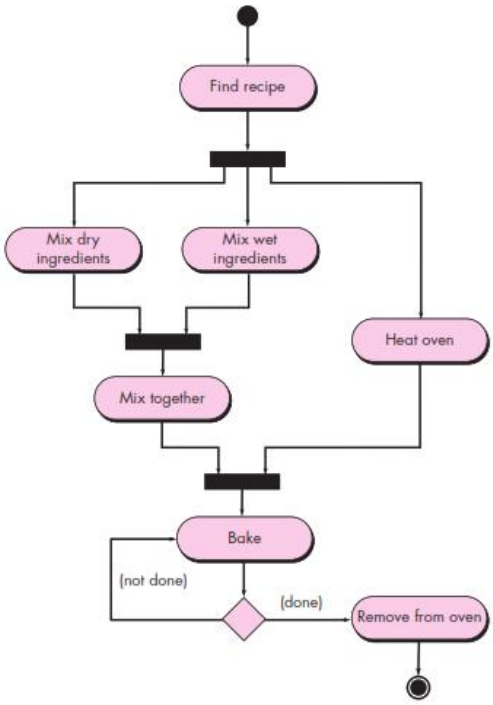
| No | Nama | Gambar | Fungsi |
|---|---|---|---|
| 5. | *Generalization* |  | Relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan memerlukan use case ini untuk menjalankan fungsinya atau sebagai syarat dijalankan use case ini |
| 6. | *Include* |  | Komunikasi antara aktor dan use case yang berpartisipasi pada use case atau use case memiliki interaksi dengan aktor. |

Sumber : (A.S & Shalahuddin, 2015)

### 2.10.2 Activity Diagram

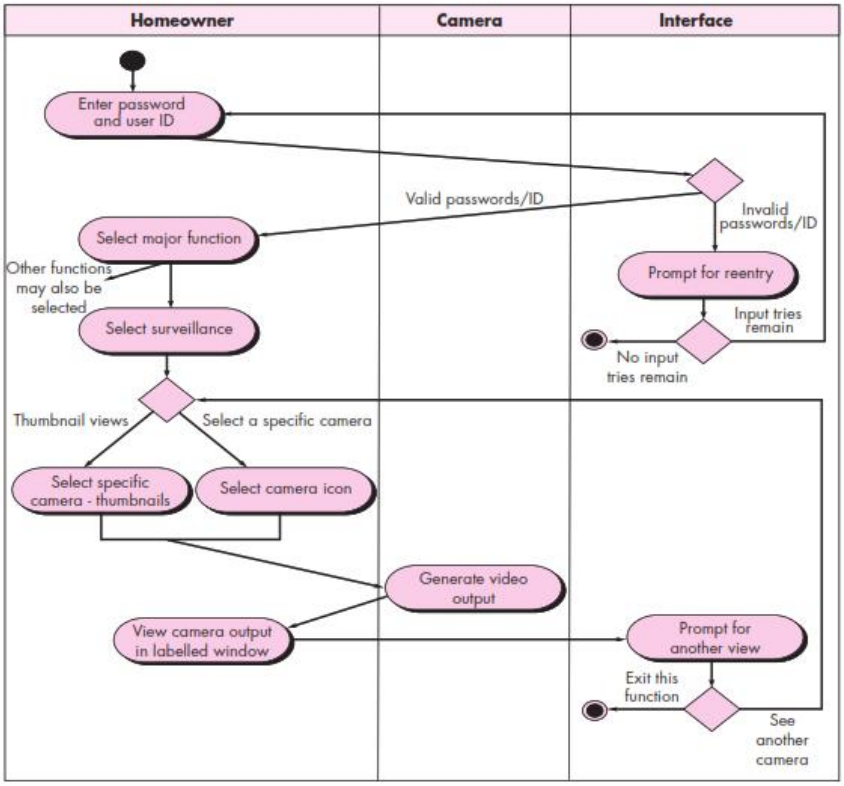
UML model *Activity diagram* menjelaskan perilaku dinamis dari system atau bagian bagian sistem melalui aliran proses yang dilakukan sistem. Hal ini sama dengan model *flow chart* tetapi sedikit berbeda dikarenakan *activity diagram* dapat menampilkan aliran proses sistem secara bersamaan. Komponen utama dari *activity diagram* adalah *action node*, diwakili oleh bulat persegi panjang, yang sesuai dengan tugas yang dilakukan oleh *perangkat lunak* panah dari satu *node* ke *node* lain menjelaskan aliran kontrol. Hal ini berarti bahwa setelah *action* pertama selesai, *action* kedua baru dijalankan. Sebuah lingkaran hitam pekat menjelaskan awal proses *activity* dimulai. Sebuah titik hitam yang yang dikelilingi lingkaran hitam menjelaskan akhir dari proses *activity.* Garis horizontal berwarna hitam merupakan pemisah dua *action* atau lebih secara bersamaan (Pressman, 2010). *Activity diagram* terdapat 2 bentuk pembuatan yang pertama akan ditunjukkan pada Gambar 2.5 dan yang kedua merupakan variasi penggunaan dari *activity diagram* akan ditunjukkan pada Gambar 2.6 yaitu model *Swimlane Diagram* yang fungsinya sama dengan *activity diagram* namun tingkah laku aktor lebih jelas pada *Swimlane Diagram*.

Untuk membuat struktur *activity diagram* dengan baik dan benar harus memperhatikan dan mengerti ketika menggunakan simbol dikarenakan jika terdapat kesalahpahaman karena simbol yang salah maka tahap perancangan akan susah untuk mendefinisikannya dan menyebabkan kesalahan pada waktu implementasi. Pada Tabel 2.9 ditunjukkan simbol-simbol yang dapat digunakan untuk membuat *activity diagram*.




**Gambar 2.5 Contoh *Activity Diagram***

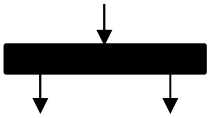
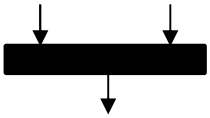
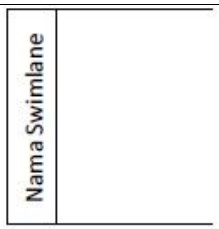
Sumber : (Pressman, 2010)




**Gambar 2.6 Contoh *Swimlane Diagram***

Sumber : (Pressman, 2010)

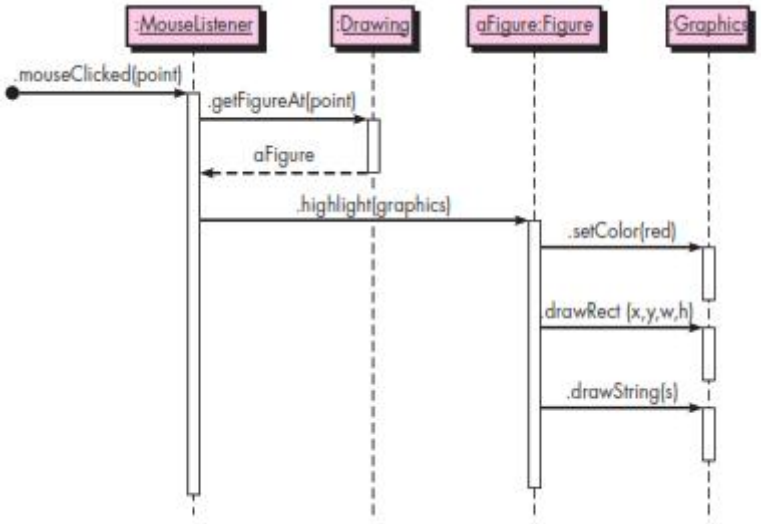
**Tabel 2.9 Simbol-Simbol *Activity Diagram***

| No | Nama | Gambar | Fungsi |
|---|---|---|---|
| 1. | Status Awal |  | Status awal aktivitas sistem, sebuah diagram aktivitas memiliki status awal |
| 2. | Aktifitas |  | Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja |
| 3. | *Decision* |  | Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu |
| 4. | *Fork* (Percabangan) |  | Asosiasi *fork* dimana pemisah beberapa aliran konkuren dari satu aliran tunggal |
| 5. | *Join* (Penggabungan) |  | Asosiasi *join* dimana lebih dari satu aktivitas digabungkan menjadi satu |
| 6. | Status Akhir |  | Status Akhir yang dilakukan sistem, sebuah diagram memiliki sebuah status akhir |
| 7. | *Swimlane* |  | Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi |

Sumber : (A.S & Shalahuddin, 2015)

### 2.10.3 Sequence Diagram

Berbeda dengan *class diagram* yang menampilkan struktur statis dari komponen sistem *perangkat lunak, sequence diagram* digunakan untuk menampilkan struktur yang dinamis antara obyek selama fungsi dijalankan. *Sequence diagram* menampilkan proses pengiriman pesan antar obyek untuk menyelesaikan fungsi tertentu. Alasan lain dengan menggunakan *sequence diagram* adalah untuk menampilkan interaksi suatu *use case* atau satu skenario sistem *perangkat lunak. Sequence diagram* menampilkan metode panggilan menggunakan panah horizontal dari pelaku ke target pelaku, diberi label dengan nama metode dan termasuk parameter, jenis dan jenis timbal balik. Untuk kasus *looping, conditional,* dan struktur kontrol lainnya dalam *sequence diagram*, dapat menggunakan *frame* interaksi seperti persegi panjang yang mengelilingi bagian dari *diagram* (Pressman, 2010). Contoh *sequence diagram* dapat dilihat pada Gambar 2.7.



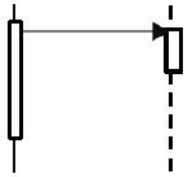

**Gambar 2.7 Contoh *Sequence Diagram***
Sumber : (Pressman, 2010)

Dalam menggunakan atau membuat struktur *sequence diagram* dengan baik dan benar harus memperhatikan dan mengerti ketika menggunakan simbol dikarenakan jika terdapat kesalahpahaman karena simbol yang salah maka tahap implementasi akan susah untuk dilakukan. Pada Tabel 2.10 ditunjukkan simbol-simbol yang dapat digunakan untuk membuat *sequence diagram*.

**Tabel 2.10 Simbol-Simbol *Sequence Diagram***

| No | Nama | Gambar | Fungsi |
|---|---|---|---|
| 1. | *Actor* |  | Menggambarkan orang yang berinteraksi dengan sistem |
| 2. | *Boundary* |  | Menggambarkan antarmuka dan interaksi satu atau lebih aktor dengan sistem |
| 3. | *Controller* |  | Menggambarkan perilaku mengatur, mengkoordinasikan perilaku sistem dan dinamika dari suatu sistem, menangani tugas utama dan mengontrol alur kerja suatu sistem |
| 4. | *Entity* |  | Menggambarkan informasi atau data yang harus disimpan oleh sistem |

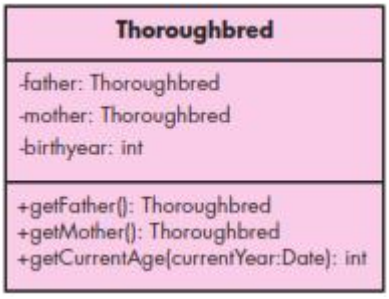
**Tabel 2.10 Simbol-Simbol *Sequence Diagram* (lanjutan)**

| No | Nama | Gambar | Fungsi |
|---|---|---|---|
| 5. | *Lifeline* |  | Mengindikasikan keberadaan sebuah obyek dalam basis waktu |
| 6. | *Message* |  | Menggambarkan pesan / hubungan antar objek, yang menunjukkan urutan kejadian yang terjadi |

Sumber : (A.S & Shalahuddin, 2015)

### 2.10.4 Class Diagram

Untuk kelas model, termasuk atribut kelas, pengoperasian, dan hubungan antar kelas dengan kelas lain, UML menyediakan model class diagram. Class diagram menyediakan view statis atau struktural dari sistem. Class diagram tidak menunjukkan sifat dinamis dari komunikasi antar objek setiap kelas dalam diagram. Unsur utama dalam class diagram adalah kotak, yang merupakan ikon yang digunakan untuk mewakili kelas kelas. Setiap kotak dibagi dengan garis horizontal. Bagian atas berisi nama class. Bagian tengah berisi atribut tiap kelas. Setiap atribut mengacu pada sesuatu yang merupakan objek pada kelas. Setiap atribut dapat memiliki nama, tipe dan simbol. Simbol ditunjukkan dengan -, #, ~ atau +, yang masing masing menunjukan atribut tersebut private, protected, package atau public (Pressman, 2010). Contoh *class diagram* dapat dilihat pada Gambar 2.8.




**Gambar 2.8 Contoh *Class Diagram***

Sumber : (Pressman, 2010)

Dalam menggunakan atau membuat struktur *class diagram* dengan baik dan benar harus memperhatikan dan mengerti ketika menggunakan simbol dikarenakan jika terdapat kesalahpahaman karena simbol yang salah maka tahap

implementasi akan susah untuk dilakukan. Pada Tabel 2.11 ditunjukkan simbol-simbol yang dapat digunakan untuk membuat *class diagram*.

**Tabel 2.11 Simbol-Simbol *Class Diagram***

| No | Nama | Gambar | Fungsi |
|---|---|---|---|
| 1. | *Class* |   | *Class* merupakan sebuah *template* dimana objek dibuat. *Class* mendefnisikan atribut, operasi dan *instance*. |
| 2. | *Association* |  | Relasi yang menunjukan hubungan antara dua *class*. |
| 3. | *Generalization* |  | Relasi yang menunjukan pewarisan dari *class* utama (*parent*) ke *class* anak (*child)* |
| 4. | *Dependencies* |  | Relasi yang menunjukan suatu *class* bergantung pada *class* yang lain |
| 5. | *Agregasi / Aggregation* |  | Relasi yang menunjukan sebuah elemen yang terdiri dari beberapa komponen kecil |

Sumber : (Sukamto & Shalahudin, 2014)

## 2.11 Framework

Framework dapat diartikan sebagai koleksi atau kumpulan potongan-potongan program yang disusun atau diorganisasikan sedemikian rupa, sehingga dapat digunakan untuk membantu membuat aplikasi untuh tanpa harus membuat semua kodenya dari awal. Saat ini ada banyak framework PHP, diantaranya: Zend, Cake PHP, Trax, Symfony, Codeigniter dan sebagainya. Tentu saja, setiap framework memiliki kelebihan dan kekurangannya masing-masing. Keuntungan yang dapat diperoleh dari penggunaan framemork adalah (Octafian, 2015) :

1. Waktu pembuatan aplikasi situs web jauh lebih singkat.
2. Kode aplikasi situs web menjadi lebih mudah dibaca, karena sedikit dan sifatnya pokok, detailnya adalah kode dari framework.
3. Situs web menjadi lebih mudah diperbaiki, karena tidak perlu fokus ke semua komponen kode situs web, terutama kode sistem framework.
4. Tidak perlu lagi membuat kode penunjang aplikasi situs web seperti koneksi database, validasi form, GUI, dan keamanan.

5. Pikiran pengembang menjadi lebih terfokus ke kode alur permasalahan situs web, apa yang ditampilkan dan layanan apa saja yang diberikan dari aplikasi situs web tersebut.
6. Jika dikerjakan team work, maka akan lebih terarah karena sistem framework, mengharuskan adanya keteraturan peletakan kode. Seperti bagian pengambilan database terpisah dengan bagian pengaturan tampilan untuk pengunjang.

## 2.12 CodeIgniter

Codeigniter adalah sebuah framework PHP yang dapat membantu mempercepat developer dalam pengembangan aplikasi situs web berbasis PHP dibandingkan jika menulis semua kode program dari awal. Codeigniter menyediakan banyak library untuk mengerjakan tugas-tugas yang umumnya ada pada sebuah aplikasi berbasis web. Selain itu, struktur dan susunan logis dari Codeigniter membuat aplikasi yang dibuat menjadi semakin teratur dan rapi. Dengan demikian developer dapat fokus pada fitur-fitur apa yang dibutuhkan oleh aplikasi dengan membuat kode program seminimal mungkin. Codeigniter pertama kali dibuat oleh Rick Ellis, CEO Ellislab, Inc. (http://ellislab.com), sebuah perusahaan yang memproduksi sebuah CMS (Content Management System) yang cukup handal, yaitu ExpressionEngine (http://www.expressionengine.com). Saat ini, Codeigniter dikembangkan dan dimaintain oleh ExpressionEngine Development Team. Beberapa keuntungan menggunakan Codeigniter, diantaranya (Octafian, 2015):

1. Gratis

   Codeigniter berlisensi dibawah Apache/ BSD opensource, sehingga penggunaannya secarabebas.

2. Ditulis menggunakan PHP 4

   Meskipun Codeigniter dapat berjalan pada PHP 5, namun sampai saat ini kode program Codeigniter masih dibuat dengan menggunakan PHP 4. Hal ini dilakukan agar Codeigniter dapat tersebar lebih luas di komunitas PHP. Karena hingga saat ini, sebagian besar web hosting masih menggunakan PHP 4. Jika Codeigniter dibuat dengan PHP 5, tentu saja hasilnya juga akan jauh lebih canggih, karena bisa memanfaatkan teknologi PHP 5 yang saat ini masih belum dapat dilakukan oleh PHP 4, misalnya untuk menerapkan konsep OOP Multiple Inheritance.

3. Berukuran kecil

   Ukuran Codeigniter yang kecil merupakan keunggulan tersendiri. Dibandingkan framework lain yang berukuran besar, serta membutuhkan resource yang besar pula untuk berjalan. Pada Codeigniter, bisa diatur agar sistem meload library yang dibutuhkan saja, sehingga sistem dapat berjalan ringan dan cepat.

4. Menggunakan konsep MVC

   Codeigniter menggunakan konsep MVC (Model View Controller) yang memungkinkan pemisahan antara layer application-logic dan presentation.

5. URL yang sederhana

   Secara default, URL yang dihasilkan Codeigniter sangat bersih (clean) dan Search Engine Friendly (SEF).

6. Memiliki paket library yang lengkap

   Codeigniter memiliki library yang lengkap untuk mengerjakan operasi-operasi yang umum dibutuhkan oleh sebuah aplikasi berbasis web, misalnya mengakses database, mengirim email, memvalidasi form, menangani session dan sebagainya.

7. Extebsible

   Sistem dapat dikembangkan dengan mudah dengan menggunakan plugin dan helper, atau dengan menggunakan hooks.

8. Tidak memerlukan template engine

   Meskipun Codeigniter dilengkapi dengan templateparses sederhana yang dapat digunakan, tetapi hal ini tidak mengharuskan untuk menggunakannya. Penggunaan template enggine dapat mengurangi performance dari sistem.

9. Dokumentasi lengkap dan jelas

   Dari sekian banyak framework, Codeigniter adalah satu-satunya framework dengan dokumentasi yang lengkap dan jelas. Tim pengembang Codeigniter berkomitmen bahwa dokumentasi juga sama pentingnya dengan kode program Codeigniter itu sendiri. Source code Codeigniter juga dilengkapi komentar didalamnya, sehingga memperjelas fungsi sebuah kode program.

10. Komunitas

    Komunitas penggunaa Codeigniter saat ini berkembang pesat, dan dapat berpartispasi di http://codeigniter.com /forums/.

Beberapa hal yang harus diketahui dari Codeigniter (Octafian, 2015) :

a. Kebutuhan Sistem

   Untuk menjalankan Codeigniter diperlukan *server* yang menjalankan PHP versi 4.3.2 atau yang lebih tinggi. Codeigniter dapat mendukung RDBMS MySQL (4.1+), MySQLi, Ms.SQL Server, Postgres, Oracle, SQLite, dan ODBC.

b. Fitur Codeigniter

   Berikut ini adalah fitur-fitur yang didukung oleh Codeigniter:

*Model View Controller based, PHP 4 compatible, Extremely light weight., Full featured database classes with support for serveral platforms, Active record database support, Form and data validation, Securityand XSS filtering, Session management, Email sending class, Support attachment, HTML/ Text email, multiple protocols (sendmail, SMTP,danmail ), Image manipulation library (cropping, resizing, rotating), Support GD, ImageMagick dan NetPBM, File uploading class, FTP class, Localization, Pagination, Data encryption, Benchmarking, Full page caching, Error logging, Application profiling, Scaffolding, Calendaringclass, User agent class, Zip encoding class, Template engine class, Trackback class, XML-RPC library, Unit testing class, Search engine friendly URLs, Flexible URI routing, Support for hooks, class extensions, dan plugins, Large library of helper functions.*

## 2.13 Pengujian Perangkat Lunak

Pengujian perangkat lunak sangat diperlukan untuk memastikan perangkat lunak atau aplikasi yang sudah atau sedang dibuat dapat berjalan sesuai dengan fungsionalitas yang diharapkan. Pengembang atau penguji perangkat lunak harus menyiapkan sesi khusus untuk menguji program yang sudah dibuat agar kesalahan ataupun kekurangan dapat dideteksi sejak awal dan dikoreksi secepatnya. Pengujian atau testing sendiri merupakan elemen kritis dari jaminan kualitas perangkat lunak dan merupakan bagian yang tidak terpisah dari siklus hidup pengembangan perangkat lunak seperti halnya analisis, desain, dan pengkodean. Langkah-langkah pengujian software ada 4 yaitu (Mustaqbal, et al., 2015):

1. Unit testing-testing per unit yaitu mencoba alur yang spesifik pada struktur modul kontrol untuk memastikan pelengkapan secara penuh dan pendeteksian error secara maksimum
2. Integration testing – testing per penggabungan unit yaitu pengalamatan dari isu-isu yang diasosiasikan dengan masalah ganda pada verifikasi dan konstruksi program
3. High-order test yaitu terjadi ketika software telah selesai diintegrasikan atau dibangun menjadi satu –tidak terpisah-pisah
4. Validation test yaitu menyediakan jaminan akhir bahwa software memenuhi semua kebutuhan fungsional, kepribadian dan performa.

### *2.13.2 White Box Testing*

White Box Testing adalah salah satu cara untuk menguji suatu aplikasi atau software dengan cara melihat modul untuk dapat meneliti dan menganalisa kode dari program yang di buat ada yang salah atau tidak. Kasus yang sering menggunakan white box testing akan di uji dengan beberapa tahapan yaitu (Mustaqbal, et al., 2015):

1. Pengujian seluruh keputusan yang menggunakan logikal.

2. Pengujian keseluruh loop yang ada sesuai batasan-batasannya.
3. Pengujian pada struktur data yang sifatnya internal dan yang terjamin validitasnya.

#### *2.13.2.2 Basis Path Testing*

*Basis Path Testing* adalah teknik *white-box testing* yang pertama kali diajukan oleh Tom McCabe, memungkinkan perancangan *Test case* mengukur kompleksitas logis dari desain prosedural dan menggunakannya sebagai pedoman untuk menetapkan basis set dari jalur eksekusi. *Test case* yang dilakukan menggunakan basis set dijamin untuk menggunakan setiap statemen didalam program paling tidak sekali selama pengujian (Pressman, 2010, p. 485).

Perhitungan *Cyclomatic Complexity* menggunakan 3 cara berikut (Pressman, 2010):

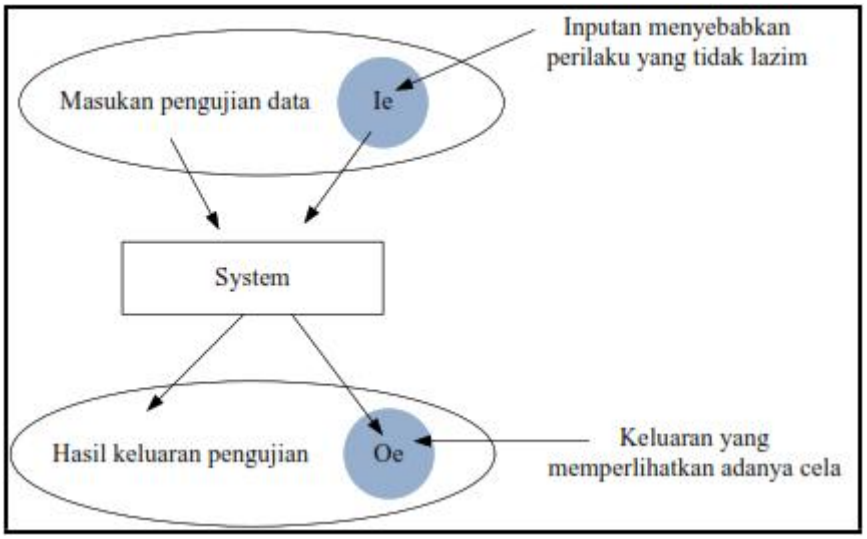
1. Jumlah *region flow graph* mempunyai hubungan dengan *Cyclomatic Complexity* V(G).
   *V(G) = R* (2.1)
   Dimana :
   V(G) = *Cyclomatic Complexity*
   R = Jumlah *region*
2. *Cyclomatic Complexity* V(G) untuk *flow graph* didefinisikan sebagai:
   *V(G) = E – N + 2* (2.2)
   Dimana :
   V(G) = *Cyclomatic Complexity*
   E = Jumlah *edge* pada *flow graph*
   N = Jumlah *node* pada *flow graph*
3. *Cyclomatic Complexity* V(G) untuk *flow graph* juga didefinisikan sebagai:
   *V(G) = P + 1* (2.3)
   Dimana :
   V(G) = *Cyclomatic Complexity*
   P = Jumlah *predicate node* pada *flow graph*

Berdasarkan pengujian *white-box* dengan menggunakan teknik *basis path,* algoritma yang diuji benar ketika jumlah hasil dari persamaan 2.1, persamaan 2.2 dan persamaan 2.3 sama.

### 2.13.3 *Black Box Testing*

Proses black-box testing adalah pengujian yang berasal dari spesifikasi sistem. Sistem diperlakukan seperti sebuah kotak hitam (black-box) yang mana perilaku hanya bisa ditentukan dengan mempelajari inputan dan keluaran yang berhubungan. Nama lainnya adalah pengujian fungsional karena penguji hanya dikonsentrasikan terhadap fungsionalitas dan tidak pengimplementasi dari

perangkat lunak. Gambar 2.9 mengilustrasikan model dari sebuah sistem yang diasumsikan dalam pengujian black-box. Penguji memberikan masukan-masukan untuk komponen atau sistem dan memeriksa keluaran yang cocok. Jika keluaran tidak terprediksi, pengujian mendeteksi sebuah masalah dengan perangkat lunak tersebut (Octafian, 2015).




**Gambar 2.9 Pengujian Black-box**

Sumber : (Octafian, 2015)

### 2.13.4 Compatibility Testing

Komputer yang berbeda, perangkat display, sistem operasi, browser, dan kecepatan koneksi jaringan dapat memiliki pengaruh signifikan pada operasi WebApp. Setiap konfigurasi komputasi dapat mengakibatkan perbedaan kecepatan pemrosesan sisi klien, resolusi layar, dan kecepatan koneksi. Sistem operasi liku-liku dapat menyebabkan masalah pemrosesan WebApp. Browser yang berbeda terkadang menghasilkan hasil yang sedikit berbeda, terlepas dari tingkat standarisasi HTML dalam WebApp. Plug-in yang diperlukan mungkin atau mungkin tidak tersedia untuk konfigurasi tertentu.

Dalam beberapa kasus, masalah kompatibilitas kecil tidak menimbulkan masalah signifikan, namun di pihak lain, kesalahan serius dapat ditemui. Misalnya, kecepatan download mungkin tidak dapat diterima, kurangnya plug-in yang diperlukan dapat membuat konten tidak tersedia, perbedaan browser dapat mengubah tata letak halaman secara dramatis, gaya font dapat diubah.

Dan menjadi tidak terbaca, atau formulir mungkin tidak berjalan dengan benar. Pengujian kompatibilitas berusaha untuk mengungkap masalah ini sebelum WebApp online.

Langkah pertama dalam pengujian kompatibilitas adalah menentukan seperangkat konfigurasi komputasi sisi klien "yang biasa dihadapi" dan variannya. Intinya, struktur pohon dibuat, mengidentifikasi setiap platform komputasi, perangkat display biasa, sistem operasi yang didukung pada platform,

browser yang tersedia, kemungkinan kecepatan koneksi Internet, dan informasi serupa. Selanjutnya, serangkaian tes validasi kompatibilitas diturunkan, sering disesuaikan dari tes antarmuka yang ada, tes navigasi, tes kinerja, dan tes keamanan. Maksud dari tes ini adalah untuk mengungkap kesalahan atau masalah eksekusi yang dapat ditelusuri ke perbedaan konfigurasi (Pressman, 2010).

### *2.13.5 User Acceptance Testing*

Sebelum perangkat lunak diberikan kepada pengguna, perangkat lunak diuji untuk memastikan bahwa semuanya memuaskan sesuai dengan persyaratan pengguna yang disebutkan dalam *Specification Requirement Software*. Pengujian yang dimaksud adalah *User Acceptance Testing*. Pengujian UAT dilakukan saat perangkat lunak sudah siap digunakan dan sebelum perangkat lunak digunakan oleh pengguna. Tujuan utama pengujian UAT adalah untuk mendapatkan kepercayaan pengguna bahwa perangkat lunak itu bebas dari kesalahan dan memenuhi semua persyaratan pengguna yang ditentukan diawal (Goel & Gupta, 2014).

UAT adalah sarana dimana perusahaan memastikan bahwa perangkat lunak yang dibuat benar-benar akan memenuhi persyaratan pengguna. Pengujian ini adalah pengujian akhir yang dilakukan setelah pengujian fungsional, sistem, dan regresi selesai. Tujuan utama UAT adalah memeriksa perangkat lunak terhadap kebutuhan bisnis. Hal ini dilakukan oleh pengguna akhir yang sudah tidak asing lagi dengan kebutuhan bisnis. Pengujian ini hampir sama dengan pengujian *black-box testing* di mana dua atau lebih pengguna akhir terlibat dalam pengujian. Pengguna mengerti mengenai UAT dan harapan dari perangkat lunak yang dibuat. UAT dilakukan oleh pengguna yang berhubungan langsung dengan perangkat lunak yang dibuat sebelum perangkat lunak digunakan oleh perusahaan atau pengguna yang sebenarnya, pengujian inilah pilihan terakhir bagi pelanggan untuk menguji perangkat lunak (Goel & Gupta, 2014).

Winarno Surachmad dalam “Dasar dan Teknik Research Pengantar Metodologi Ilmiah”, memberikan pedoman sebagai berikut : “Apabila populasi cukup homogen (serba sama), terhadap populasi di bawah 100 dapat dipergunakan sampel sebesar 50%, di atas 1.000 sebesar 15% (Surachmad, 1982).