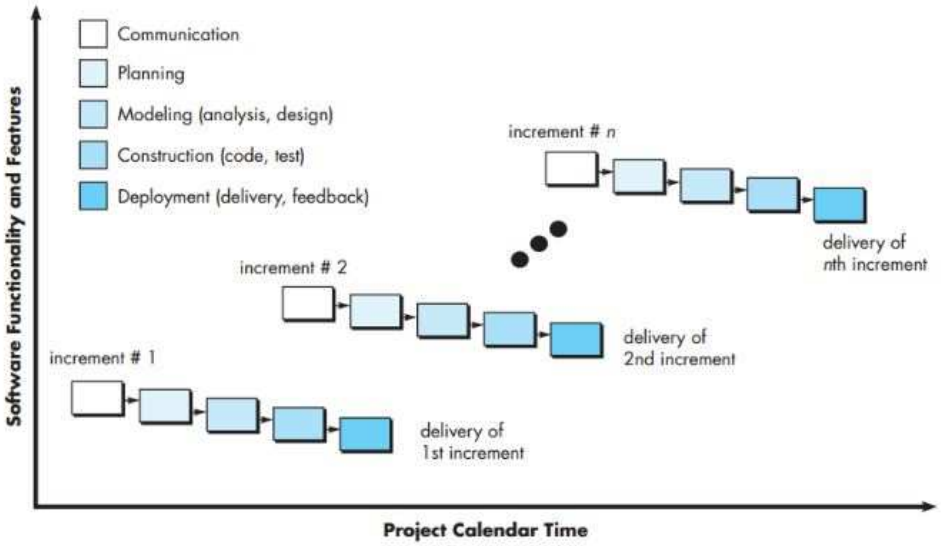
BAB 2
TINJAUAN REFERENSI

2.1 Model Incremental

Model Incremendal adalah gabungan dari proses linear dan parallel. Model ini memiliki tahapan komunikasi, perencanaan, pemodelan (Analisa dan desain), konstruksi (kode dan tes), dan terakhir adalah tahap pengembangan. Saat model ini digunakan, increment pertama merupakan suatu produk utama. Selanjutnya, increment akan meningkat apabila increment pertama sudah selesai. Model ini akan selesai apabila semua increment telah selesai dilakukan. (Pressman & Maxim, 2015)



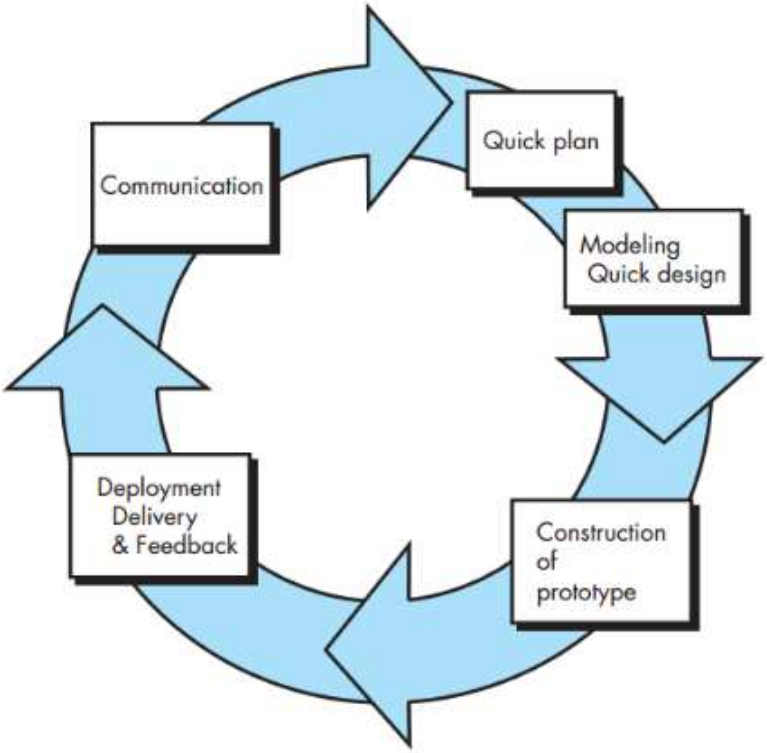
Gambar 2.1~~2.1~~ Contoh Proses Model Incremental

2.2 Model Prototype

Model Prototype adalah suatu cara untuk programmer mendesain suatu kerangka berdasarkan dengan keinginan dari target konsumen.

Proses pertama dari prototyping ini adalah komunikasi. Komunikasi ini melibatkan kita dan konsumen untuk saling berdiskusi mengenai perangkat lunak apa yang akan di buat dan beberapa garis besar serta *requirement* yang dibutuhkan. Setelah itu, ada tahap *Quick Design*. Tahap ini merupakan tahapan kita memberikan gambaran secara garis besar terhadap perangkat lunak yang akan dibuat dari sisi tampilan layar yang akan dilihat oleh konsumen. Tahapan selanjutnya adalah

konstruksi prototipe dan akan dievaluasi oleh target konsumen hingga kriteria yang dibutuhkan terpenuhi. (Pressman & Maxim, 2015).



Gambar 2.2~~2.2~~ Proses Model Prototype

2.3 UML

UML adalah suatu metode pemodelan secara visual yang digunakan sebagai sarana perancangan sistem berorientasi objek. UML merupakan hasil evolusi dari pemodelan bahasa object-oriented. Dalam UML terdapat beberapa jenis diagram, yaitu

2.3.1 Use Case Diagram

Menurut (Gomaa, 2011), Use case merupakan suatu diagram interaksi antara satu atau lebih aktor dari suatu sistem. Use case memberikan gambaran terhadap tahapan interaksi dari satu aktor atau lebih terhadap suatu sistem. Model use case mendeskripsikan fungsional dari sebuah sistem yang didalamnya ada aktor dan use cases.

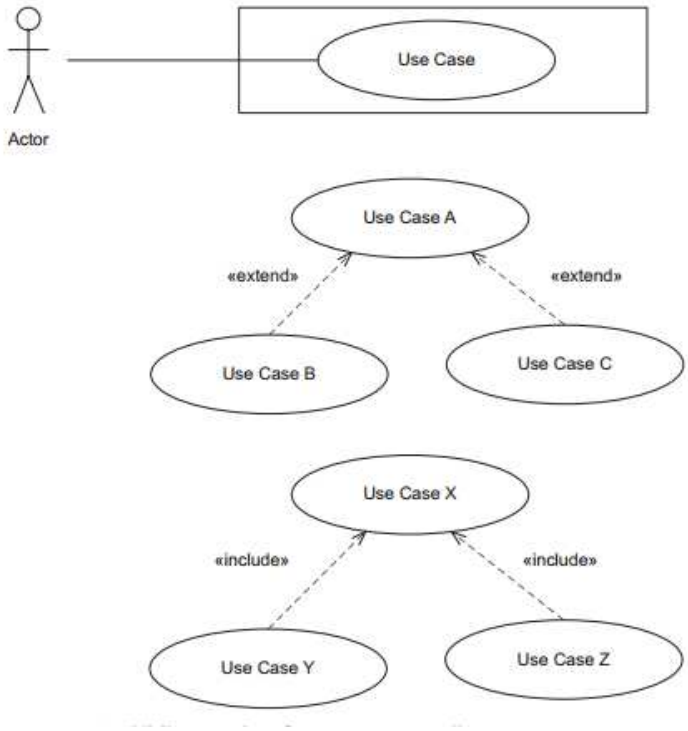
Use case akan selalu dimulai dengan inputan dari aktor dan akan mengandung beberapa tahapan lain antara aktor dan sistem. Sebuah use case tidak memberitahu isi dari sebuah sistem. Sebuah use case bisa terdiri dari

satu aktor dan satu sistem, namun use case yang lebih kompleks akan memiliki lebih dari satu aktor.

Untuk setiap use case terdapat dokumentasi yang diperlukan dan berbentuk deskriptif. Berikut adalah komponen-komponen yang digunakan dalam dokumentasi tersebut:

1. *Use case name:* nama use case
2. *Summary:* deskripsi singkat terkait dengan use case
3. *Dependency:* ketergantungan dengan use case lain
4. *Actors*: aktor yang terkait
5. *Preconditions:* kondisi yang harus benar sebelum use case jalan
6. *Description of main sequence*: deskripsi tahapan aktor dan sistem
7. *Description of alternative sequence:* deskripsi alternative dari *main sequence*
8. *Nonfunctional requirements:* deskripsi persyaratan tidak fungsional
9. *Postcondition:* kondisi yang akan terjadi saat use case selesai

   Outstanding questions: pertanyaan selama proses pengembangan



Gambar 2.3 Contoh *Use Case Diagram*

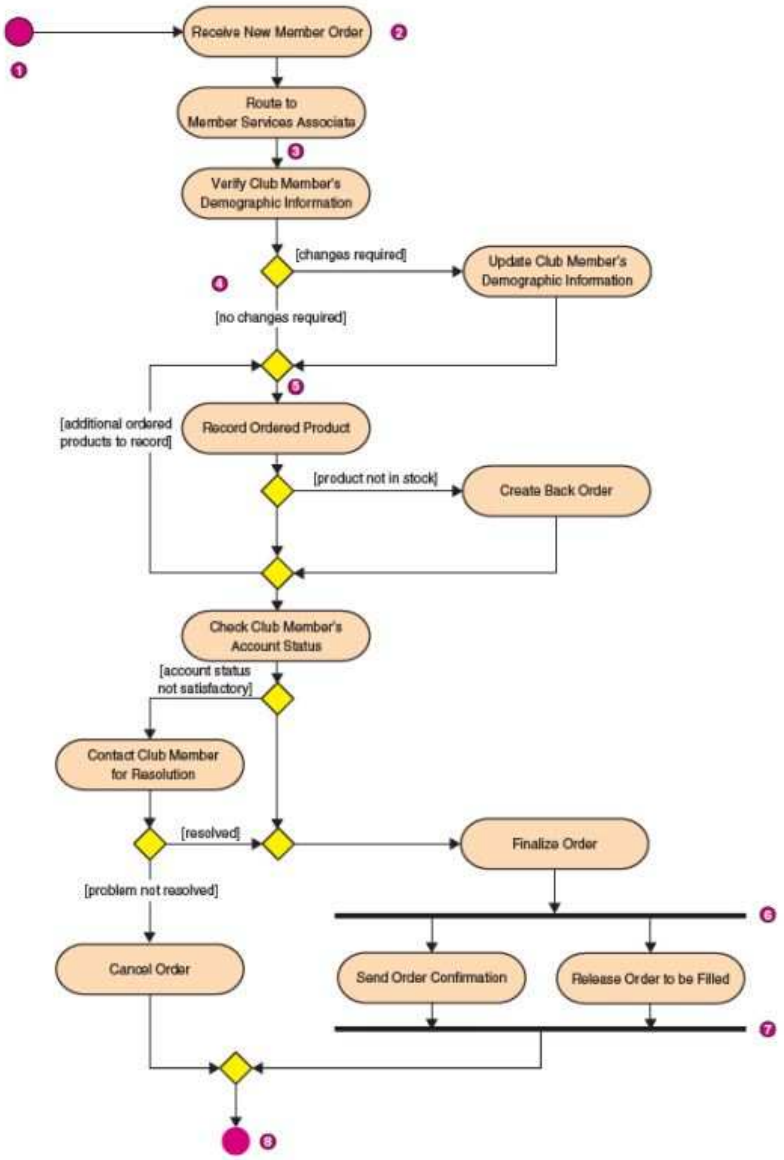
2.3.2 *Activity Diagram*

Menurut, (Whitten & Bentley, 2007), *Activity Diagram* merupakan representasi tahapan dari sebuah use case. *Activity Diagram* ini menunjukkan alur yang lebih jelas dari suatu use case. Selain itu, *Activity Diagram* juga dapat berfungsi untuk mengambarkan tahapan antara use case yang ada. Berikut contoh *Activity Diagram*.

Activity diagram memiliki beberapa komponen yaitu:

1. *Initial node*. Merupakan lingkaran padat yang menggambarkan awal dari sebuah proses
2. *Actions.* Sebuah kotak yang merupakan gambaran langkah-langkah.
3. *Flow.* Merupakan panah yang mengindikasikan tahapan berikutnya dari sebuah aksi.
4. *Decision*. Merupakan bentuk wajik dengan satu arus yang masuk dan dua atau lebih arus yang keluar. Hal ini merupakan kondisi yang dapat terjadi.
5. *Merge.* Merupakan bentuk wajik dengan dua atau lebih arus yang masuk dan satu yang keluar.
6. *Activity final.* Merupakan lingkaran padat didalam lingakaran kosong yang menggambarkan bahwa proses sudah selesai.

Pada *activity diagram*, terdapat istilah *swimlane. Swimlane* adalah sebuah partisi yang dibuat untuk melakukan spesfikasi terhadap aktor yang dapat melakukan sesuatu.



Gambar 2.4~~2.3~~ Example of *Activity Diagram*

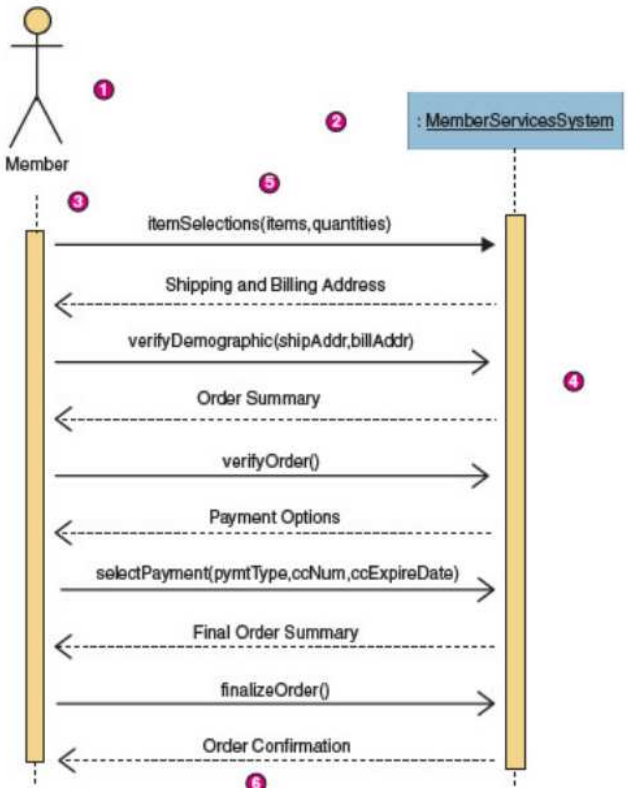
2.3.3 *Sequence diagram*

Menurut (Whitten & Bentley, 2007), *Sequence diagram* merupakan diagram yang menggambarkan komunikasi antara objek. Selain itu, diagram ini membantu untuk mengidentifikasi *high-level message* yang terjadi antara objek dan sistem.

Untuk bagian *Sequence Diagram*

1. *Actor*. Aktor yang digunakan merupakan simbol aktor yang digunakan pada *use case*.

2. *System*. Kotak yang menggambarkan sistem sebagai ” *black box”* atau *whole*. Tanda titik dua merupakan standar dari notasi *sequence diagram* yang mengindikasikan objek yang sedang berjalan dalam sebuah sistem.
3. *Lifelines.* Merupakan garis vertical yang mengarah kebawah dari aktor dan simbol sistem.
4. *Activation bars.* Merupakan bar yang mengindikasikan jangka waktu saat yang berpartisipasi sedang aktif dalam interaksi tersebut.
5. *Input messages*. Merupakan panah horizontal dari aktor ke sistem yang berkaitan dan berisi *input* yang ingin disampaikan.
6. *Output messages*. Merupakan panah horizontal dari sistem ke aktor dan berisi respon dari sistem.
7. *Frame*. Merupakan kotak yang dapat mengindikasikan *loops, alternate fragments,* atau *optional steps.*



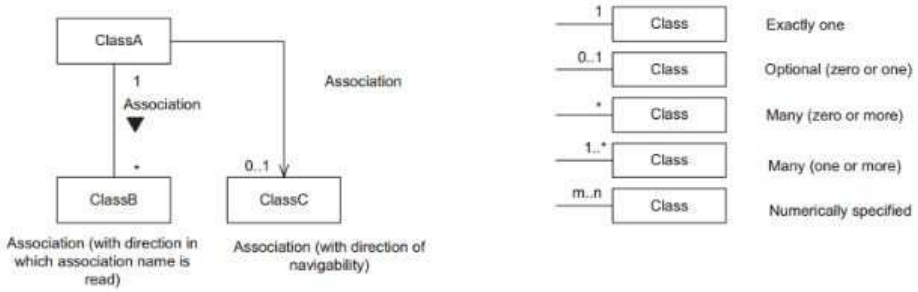
Gambar 2.5~~2.4~~ Example of *Sequence diagram*

2.3.4 Class Diagram

Menurut (Gomaa, 2011), Class diagram ini berfungsi untuk menggambarkan hubungan objek yang akan digunakan dalam suatu sistem. Class diagram digambarkan dengan kotak-kotak dan relasi statis antara mereka digambarkan dengan garis yang menghubungkan antar kotak

a. Associations

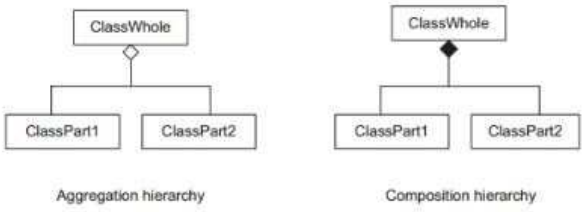
Associations merupakan hubungan statis dan structural antara dua atau lebih kelas. Hubungan antara dua kelas disebut dengan binary association. Dalam hubungan antara dua kelas tersebu terdapat suatu penjelasan terkait dengan banyaknya suatu instansi dari satu kelas yang akan terkait dengan kelas lain. Hal ini disebut dengan istilah multiplicity.



Gambar 2.6~~2.5~~ Associations: Example of Multiplicity

b. Aggregation and Composition Hierarchies

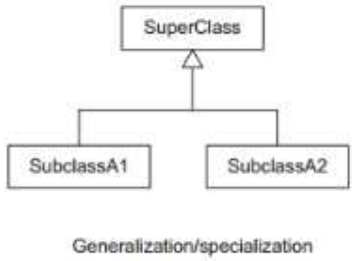
Relasi composition merupakan relasi yang lebih kuat dan ditunjukkan dengan simbol belah ketupat hitam dan relasi aggregation merupakan relasi yang tidak lebih kuat dari relasi composition ditunjukkan dengan simbol belah ketupat kosong.



Gambar 2.7~~2.6~~ Aggregation and Composition

c. Generalization/Specialization Hierarchy

Generalisasi merupakan suatu cara untuk menggambarkan hubungan antara kelas induk dengan kelas turunan (inherited).



Gambar 2.8~~2.7~~ Generalization/ Specialization Hierarchy

2.4 Object Oriented Programming

Menurut (Clark, 2011), OOP atau Object Oriented Programming merupakan suatu pendekatan untuk pengembangan suatu software yang berbasis object. Object memiliki action atau method yang akan saling berhubungan satu sama lain terkait dengan object tersebut. Ada beberapa kelebihan dalam menggunakan metode ini, yaitu:

1. Kemampuan untuk mengatur dan mengimplementasikan perubahan pada program secara efisien.
2. Kemampuan dalam pembagian tugas dengan tim untuk mengerjakan bagian-bagian dari sistem yang ada.
3. Kemampuan untuk menggunakan kembali komponen-komponen yang ada di program lain.

2.5 Kohana

Kohana merupakan suatu *framework* yang *open source*, HMVC dan dibuat dengan dasar PHP. Kohana memiliki kelebihan dalam desain *filesystem*, sedikit konfigurasi, pencarian letak error yang mudah dan kode-kode untuk keamanan telah di *review* dan dimasukan dalam *library* Kohana. (Team, 2008)

2.6 HMVC

Menurut (McArthur, 2008) MVC merupakan Model View Controller. MVC membuat desain dari suatu program menjadi lebih sederhana. Model merupakan bagian yang akan memproses bagian logika dari suatu aplikasi yang sedang dibuat. View merupakan bagian yang akan berinteraksi dengan pengguna dari program

tersebut dan berkaitan dengan HTML, CSS dan Javascript. Terakhir, Controller merupakan bagian yang menyatukan Model dan View agar dapat saling terhubung dan berjalan dengan baik.

Menurut (Straughan, 2011), HMVC merupakan suatu cara untuk melakukan satu atau lebih permintaan terhadap *server* tanpa harus menampilkan suatu halaman secara berkali-kali. Pada model HMVC, controller utama akan di eksekusi terlebih dahulu sebelum semua controller lain.

2.7 PHP

PHP (*Hypertext Preprocessor)* merupakan Bahasa yang digunakan untuk bagian server. PHP cocok digunakan untuk membuat halaman web yang dinamis. Selain itu, PHP memiliki integrase yang baik terhadap aplikasi basis data seperti MySQL.

Saat halaman web di buka, HTML akan diproses oleh browser yang ada namun untuk PHP, harus menggunakan server. Semua kode PHP akan diproses didalam server sebelum semuanya dikirimkan ke browser pengguna. (Hansen & Lengstorf, 2014)

2.8 Database

Menurut (Coronel & Morris, 2015), *Database* atau basis data merupakan sebuah pengaturan data secara efisien. Database merupakan kumpulan data yang terstruktur dalam suatu tempat penyimpanan data. Pada umumnya, Database akan diatur dalam suatu program yang dapat mengatur struktur dari database dan kontrol data pada database tersebut. Program ini biasa disebut DBMS atau Database *Manage*menet System.

2.9 ERD (*Entity Relationship Diagram)*

Menurut (Coronel & Morris, 2015), ERD merupakan bagian dari ERM atau *Entity Relationship Model* yang pada umumnya digunakan untuk representasi secara grafik terhadap model komponen pada *database*. Ada beberapa komponen untuk model ER ini.

1. *Entity* merupakan sebuah kotak yang akan merepresentasikan data yang akan dikumpulkan dan disimpan.
2. *Attributes* merupakan kumpulan data yang aka nada dalam kotak *entity*. Kumpulan data tersebut mendeskripsikan karakteristik dari sebuah *entity*.

Relationships merupakan sebuah deskripsi asosiasi antara data. Ada tiga jenis relasi data yaitu: *one-to-many (1:M), many-to-many (M:N), one-to-one (1:1).* Berikut merupakan gambar penjelasan simbol yang akan digunakan.

CROW'S FOOT SYMBOLS		
SYMBOL	CARDINALITY	COMMENT
	(0,N)	Zero or many; the "many" side is optional.
	(1,N)	One or many; the "many" side is mandatory.
	(1,1)	One and only one; the "1" side is mandatory.
	(0,1)	Zero or one; the "1" side is optional.

Gambar 2.9 *Crow's Foot Symbols*

2.10 MySql

Menurut (Bhatti & Rad, 2017), MySQL merupakan *framework open-source* untuk administrasi database. MySQL dapat dijalankan diberbagai server.

2.11 Eight Golden Rules of interface design

Dalam merancang tampilan suatu aplikasi, ada delapan prinsip yang harus diperhatikan. Berikut adalah delapan prinsip tersebut (Shneiderman & Plaisant, 2010):

1. *Strive for consistency* (Mempertahankan konsistensi tampilan)

   Di situasi yang sama, tindakan yang dilakukan harus konsisten atau sama juga. Konsistensi ini sangat dibutuhkan dalam perancangan antarmuka terutama, pemilihan warna, *layout, capitalization, font* dan sebagainya.

2. *Cater to universal usability* (Memenuhi kebutuhan secara universal)

   Perlu diketahui terdapat berbagai variasi pengguna yang memiliki kebutuhan yang bervariasi juga. Baik itu dari segi perbedaan keahlian, rentang usia, kebutuhan khusus, ataupun perbedaan tingkatan teknologi masing-masing. Dengan menambahkan penjelasaan yang lebih rinci untuk pengguna baru ataupun menambahkan *shortcuts* untuk pengguna yang sudah terbiasa menggunakan aplikasi tersebut akan meningkatkan kualitas tampilan antar muka aplikasi tersebut.

3. *Offer informative feedback* (Memberikan umpan balik yang informatif)

Untuk semua tindakan yang dilakukan oleh pengguna, harus diberikan umpan balik oleh system. Untuk tindakan yang kecil, umpan balik yang diberikan bisa dengan sederhana, namun untuk tindakan yang lebih besar/penting dibutuhkan umpan balik yang lebih substansial.

4. *Design dialogs to yield closure* (Merancang dialog yang menandakan penutupan)

Urutan tindakan harus dikelompokkan menjadi pembukaan, isi, dan penutupan. Umpan balik dari system ketika pengguna menyelesaikan suatu tindakan akan memberikan rasa puas kepada pengguna. Umpan balik tersebut juga berfungsi sebagai petunjuk untuk mempersiapkan tindakan selanjutnya

5. *Prevent errors* (Pencegahan terjadinya kesalahan)

Desainlah sistem yang mampu mencegah pengguna melakukan kesalahan fatal. Jika pengguna melakukan kesalahan, tampilan antar muka harus mendeteksi kesalahan dan memberikan umpan balik yang sederhana, konstruktif, petunjuk spesik untuk pemulihan.

6. *Permit easy reversal of actions* (Memungkinkan pembalikan tindakan yang mudah)

Desainlah sistem yang mampu mengembalikan efek yang telah dilakukan oleh pengguna. Fitur ini akan mengurangi kecemasan pengguna karena dia mengetahui bahwa tindakannya bisa dikembalikan seperti semula.

7. *Support internal locus of control* (Mendukung lokus kontrol internal)

Biasanya, pengguna yang sudah berpengalaman ingin merasa bahwa mereka yang berkuasa terhadap tampilan antarmuka mereka. Mereka tidak ingin mengubah kebiasaan mereka, dan akan terganggu jika mereka mendapatkan kesulitan dalam menghasilkan hasil yang mereka inginkan.

8. *Reduce short-term memory load* (Mengurangi beban memori jangka pendek)

   Manusia memiliki kapasitas yang terbatas untuk mengolah informasi dalam memori jangka pendek sehingga *designer* harus menghindari informasi yang berlebihan dalam satu layar tampilan antar muka.

2.12 Lima Faktor Manusia Terukur

Menurut (Ben Shneiderman, 2010) , terdapat lima faktor manusia terukur yang dapat dijadikan sebagai pusat evaluasi, yaitu:

a. *Time to learn* (Waktu belajar)

   Waktu yang dibutuhkan oleh pengguna untuk mempelajari cara menggunakan *software*-nya.

b. *Speed of performance* (Kecepatan kinerja)

   Waktu yang dibutuhkan oleh perangkat lunak untuk mengerjakan tugas yang diberikan pengguna.

c. *Rate of errors by ~~user~~users* (Tingkat kesalahan oleh pengguna)

   Berapa banyak kesalahan yang dilakukan oleh pengguna dalam melakukan tugas tersebut.

d. *Retention over time* (Daya ingat)

   Kemampuan pengguna mempertahankan pengatahuannya dari waktu ke waktu. Daya ingat berhubungan dengan waktu untuk belajar dan frekuensi penggunaan.

e. *Subjective satisfaction* (Kepuasan subjektif)

   Kepuasan penguna terhadap berbagai aspek pada saat menggunakan perangkat lunak tersebut.

2.13 Bootstrap

Menurut (Spurlock, 2013)**,** Bootstrap adalah suatu produk *open source* dari Mark Otto dan Jacab Thornton, dimana ketika diluncurkan keduanya bekerja di perusahaan Twitter. Sejak diluncurkan, Bootstrap telah berevolusi dari proyek yang sepenuhnya berbasis CSS hingga menyertakan sejumlah plugin JavaScript dan icon yang berjalan seiring dengan *form* dan *buttons.* Keunggulan Bootstrap adalah, kita dapat membuat sendiri *website* yang sesuai dengan keperluan kita.

2.14 Hyper Text Markup Language 5 (HTML5)

Menurut (MacDonald, 2011) , Hyper Text Markup Language(HTML) merupakan bahasa yang umum digunakan dalam pembuatan *web.* Secara garis besar,

ide dari HTML adalah penggunaan elemen untuk menyusun konten dari sebuah *web*. Ide ini tidak berubah hingga saat skripsi ini ditulis, bahkan *web pages* yang paling tua masih bekerja dengan baik di versi *browser* yang paling terbaru di masa sekarang (termasuk Firefox dan Chrome yang bahkan belum ada/eksis disaat *web pages* tersebut pertama kali dibuat).

HTML5 merupakan versi kelima dari HTML. Menurut (MacDonald, 2011), HTML5 memiliki sudut pandang yang berbeda. Semua hal yang telah berfungsi dengan baik di versi HTML sebelumnya akan tetap berfungsi dengan baik di HTML5. HTML5 berawal dari sebuah standar dan terpecah menjadi bagian-bagian kecil yang lebih mudah digunakan.

2.15 Cascadubg Style Sheets (CSS)

Menurut (Ian Pouncey, 2011), *Cascading Style Sheets* (CSS) adalah bahasa yang dirancang untuk menggambarkan tampilan dokumen yang ditulis dalam bahasa *markup* seperti HTML. Dengan CSS, kita dapat mengontrol warna teks, gaya tulisan, spasi antar paragraf, dan lainnya. Manfaat utama dari CSS adalah CSS dapat digunakan oleh lebih dari satu halaman, yang berarti gaya keseluruhan *website* dapat disesuaikan tanpa harus mengubah setiap halaman secara terpisah.

Adapun keuntungan dalam menggunakan CSS menurut (Ian Pouncey, 2011), yaitu:

1. Presentasi keseluruhan situs *web* dapat menggunakan satu atau beberapa dokumen.
2. Browser mulai mendukung beberapa *style sheet* alternatif, yang memiliki fitur yang memungkinkan lebih dari satu desain untuk mempresentasikan sebuah *website* dalam waktu yang bersamaan.
3. *Style sheet* memungkinkan konten untuk dioptimalkan lebih dari satu jenis perangkat
4. *Download style sheets* lebih cepat karena dokumen *web* menggunakan CSS biasa sehingga mengurangi *bandwidth browser.*
5. Pengguna situs *web* dapat membuat *style sheet* mereka sendiri, fitur tabg dapat membuat banyak situs *web* dapat diakses.

Ada empat cara mengaplikasikan CSS ke HTML:

1. CSS dapat disertakan dalam dokumen denngan menggunakan *style sheet* yang disertakan antara tag <style> dan </style> secara langsung dalam dokumen. Tag ini harus muncul diantara tag <head> dan </head>
2. CSS dapat disertakan dalam dokumennya sendiri dan dihubungkan ke dokumen HTML dengan menggunakan elemen <link>. Perlu menyertakan rel =" stylesheet" sehingga *browser* mengetahui apa yang kita inginkan.
3. CSS dapat diimpor melalui cara *embedded* ataupun *linked style sheet* menggunakan @import.
4. Deklarasi CSS dapat diterapkan langsung ke sebuah elemen di dalam dokumen HTML menggunakan *inline style* / style dalam tag dengan atribut *style.*

Berikut contoh CSS:

```
body {
    width: 650px;
    margin: 0 auto;
    background: #000;
    color: #FFF;
    font: 12px sans-serif;
}

h1 {
    font-size: 24px;
}
```

Gambar 2.10~~2.9~~ Contoh CSS

2.16 Gulp

Menurut (Maynard, 2015, hal. 6), Gulp adalah sistem pembuatan JavaScript *streaming* yang dibangun dengan node.js. Gulp memanfaatkan *power of streams* dan *code over configuration* untuk mengotomatisasi, mengatur, dan menjalankan pembangunan tugas sehingga cepat dan efisien. Dengan hanya membuat instruksi kecil, Gulp menggunakan *single purpose plugins* untuk memodifikasi dan memproses *file* proyek.

2.17 Javascript

Menurut (Suehring & Valade, 2013), JavaScript digunakan untuk pemrograman *web* untuk meningkatkan atau menambah pengalaman penggunas saat menggunakan halaman *web.* Sekarang ini JavaScript merupakan bagian penting dari halaman web, seperti Google Map, fungsi *scroll* dengan cara klik dan menggeser *map* bisa terjadi dengan bantuan JavaScript. Program JavaScript bekerja di *web browser* pengguna. Keuntungan dari hal ini adalah tidak diperlukannya server untuk menjalankan program tersebut. Namun dikarenakan program JavaScript bergantung dengan *browser* pengguna, hal ini memungkinkan program tersebut menghasilkan hasil yang beragam sesuai dengan versi dan jenis *browser* yang digunakan.

2.18 jQuery

Menurut (Bibeault, Katz, & Rosa, 2015), jQuery adalah JavaScript library populer dan kaya akan fitur yang dibuat oleh John Resig pada 2006 untuk memudahkan *client-side scripting*. jQuery mampu memanipulasi, membaca dokumen HTML, *event handling*, animasi, dan Ajax yang memiliki API yang mudah digunakan untuk berbagai jenis *browser*.

```
var checkedValue;
var elements = document.getElementsByTagName('input');
for (var i = 0; i < elements.length; i++) {
  if (elements[i].type === 'radio' &&
      elements[i].name === 'some-radio-group' &&
      elements[i].checked) {
    checkedValue = elements[i].value;
    break;
  }
}
```

Contrast that with how it can be done using jQuery:

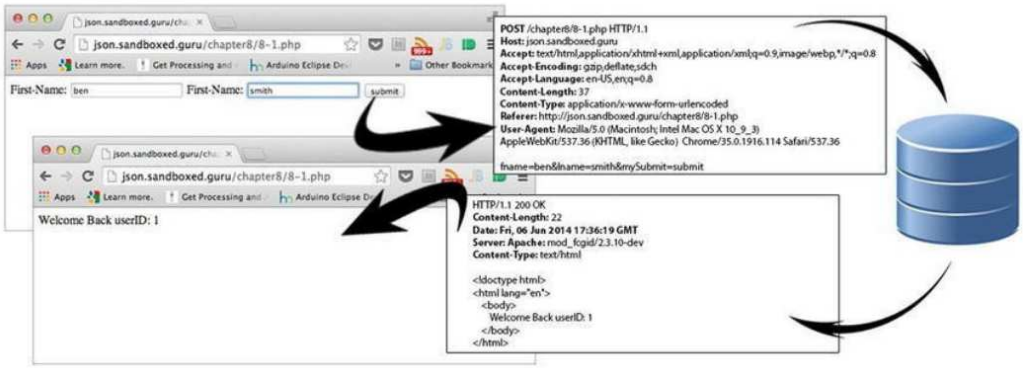
```
var checkedValue =
  jQuery('input:radio[name="some-radio-group"]:checked').val();
```

Gambar 2.11~~2.10~~ Contoh jQuery

jQuery memiliki motto yaitu "*Write less, do more.*". Figur 1 menunjukan jQuery memudahkan *client-side scripting* dan menghasilkan *lines of code* (LoC) yang jauh lebih sedikit dengan tujuan yang sama.

2.19 Ajax

Menurut (Smith, 2015), Ajax adalah *Asynchronous JavaScript and XML*. Ajax memiliki kelebihan untuk melakukan HTTP-Request seperti GET dan POST tanpa harus melakukan navigasi ke halaman lain dari halaman yang sedang dibuka.



Gambar 2.12~~2.11~~ Contoh Proses Ajax

Ajax merupakan tekonlogi untuk melakukan pertukaran data pada halaman yang sedang diakses dan server.

2.20 Software testing

Menurut (Kumar & Syed, 2011), Software testing merupakan proses verifikasi yang digunakan untuk melakukan cek terhadap error dan kekurangna yang ada dalam setiap pengembangan dan desain yang ada dalam software tersebut.

Menurut (J.Myers, Badgett, & Sandler, 2012), Untuk melakukan testing tersebut, ada dua hal relevan yang dapat dilakukan, yaitu:

1. Black-box testing, untuk pengecekan menggunakan metode ini dan mencari seluruh error yang ada maka pengujian yang dilakukan harus dengan penginputan yang lengkap dan mencoba segala kemungkinan yang ada tanpa melihat struktur dalam dari software tersebut.
2. White-box testing, untuk pengecekan menggunakan metode ini, lain hal dengan Black-box testing, White-box testing melakukan cek terhadap struktur program, logika program maupun kode yang ada.