BAB 2
TINJAUAN PUSTAKA

2.2 Teori yang berkaitan dengan *Software Engineering*

Berikut ini adalah teori-teori yang berkaitan dengan *Software Engineering* yang bersangkutan dalam penulisan skripsi ini:

2.2.1 Pengertian *Software*

Software adalah sebuah program komputer yang apabila dijalankan akan menyediakan fitur dan fungsi, atau bisa diartikan juga sebuah struktur data yang menjalankan program untuk dapat memanipulasi informasi. (Pressman, 2010, pp. 4).

Software memiliki beberapa ciri-ciri. Berikut ini beberapa ciri-ciri *software*:

1. *Software is developed or engineered; it is not manufactured in the classical sense.*

   Sebuah *Software* dirancang dengan pengembangan, tidak dirancang dengan hanya sekali jadi seperti dalam pembuatan suatu produk. Yang artinya sebuah perancangan *software* membutuhkan pengembangan-pengembangan lebih lanjut.

2. *Software doesn't wear out*

   Sebuah *software* membutuhkan pengembangan lebih lanjut yang bertujuan untuk terus menyesuaikan keadaan teknologi yang terus berkembang.

3. *Although the industry is moving toward component-based construction, most software continues to be custom built*

   Komponen pada *software* harus dirancang untuk dapat digunakan kembali baik untuk program yang sama maupun program yang berbeda. Tidak seperti pada proses industrial yang membuat komponen untuk dapat memiliki desain yang lebih menarik dari produk sebelumnya dengan menggunakan komponen yang sama.

Software memiliki beberapa kategori. Berikut ini beberapa kategori *software*:

1. *System Software*

   Kumpulan program yang digunakan untuk menjalankan program lainnya.

   Contoh: *compilers, editor,* dan *file management*
2. *Application Software*

   Sebuah program yagn dibuat untuk memenuhi suatu kebutuhan.

   Contoh: aplikasi yang memiliki nilai jual
3. *Engineering/Scientific software*

   Program yang dirancang untuk memenuhi kebutuhan perancangan atau penelitian.

   Contoh: aplikasi yang digunakan untuk kepentingan ilmiah (astronomi, biologi, dan lain-lain)
4. *Embedded software*

   Program yang dibuat dan tertanam pada suatu *hardware* yang digunakan untuk *end-user* atau sistem itu sendiri.

   Contoh: *software* pada mesin cuci, *software* pada mobil untuk keadaan bensin
5. *Product-line software*

   Program yang dibuat untuk tujuan tertentu dengan digunakan oleh banyak *customer*.

   Contoh: Microsoft Word, Microsoft Excel
6. *Web applications*

   Program yang berjalan dengan menggunakan *web browser*.

   Contoh: *web game*, *website*
7. *Artificial intelligence software*

   Program yang berupa algortima *non-numerical* untuk menangani masalah yang rumit.

   Contoh: robotik, pengenalan suara dan gambar

2.2.2 Pengertian *Software Engineering*

Software Engineering atau rekayasa perangkat lunak adalah sebuah proses memecahkan masalah *customer* dengan pengembangan yang terus dikembangkan sehingga menjadikan produk *software* yang dapat mengikuti perkembangan teknologi. (Timothy C. & Robert L., 2005, pp. 6).

2.2.3 Pengertian Sistem

Sistem adalah kumpulan dari komponen-komponen yang saling berhubungan dan berinteraksi untuk melakukan suatu tugas agar mencapai suatu tujuan. (Williams & Sawyer, 2007, pp. 510).

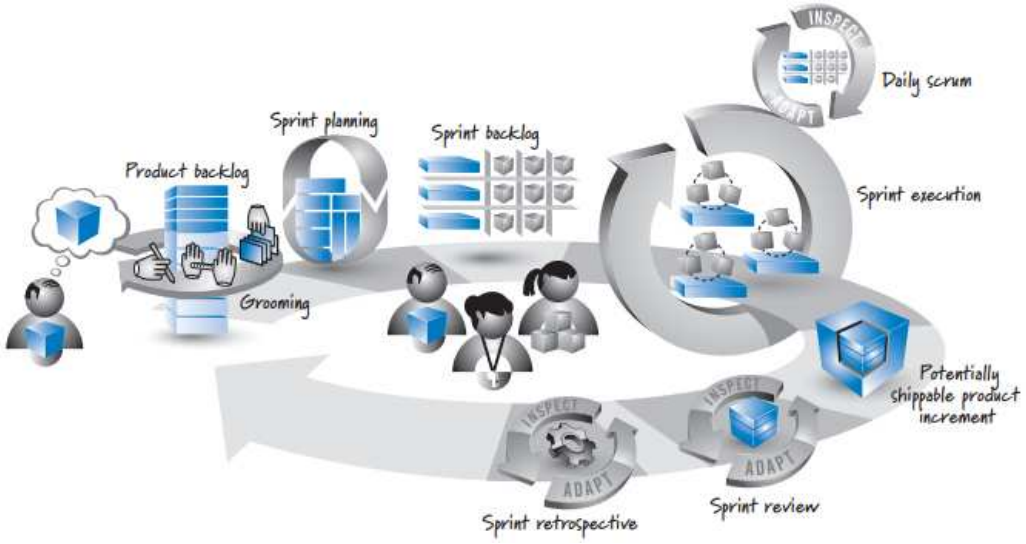
2.2.4 Pengertian Sistem Informasi Manajemen

Menurut Bodnar & Hopwood (2012), Sistem Informasi Manajemen adalah suatu kumpulan perangkat keras atau perangkat lunak yang dirancang untuk dapat mentransformasikan data didalamnya menjadi bentuk informasi yang berguna.

2.2.5 *Agile Software Development*

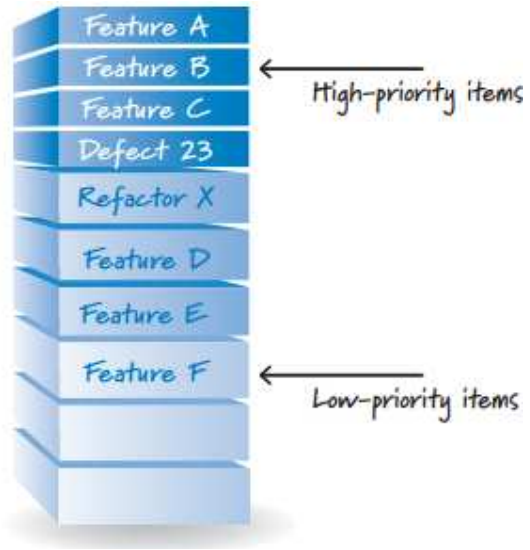
Agile Software Development adalah sebuah pengembangan *Software* yang mementingkan kepentingan *customer* atau *user* dengan mengimplementasikan *software* yang diinginkan segara mungkin dan menambahkan fitur-fitur *software* secara bertahap. Memiliki jumlah anggota yang sedikit yang mementingkan hasil *software* dari pada analisis dan desain *software*. (Pressman, 2010, pp. 65).

2.2.5.1 *Scrum*



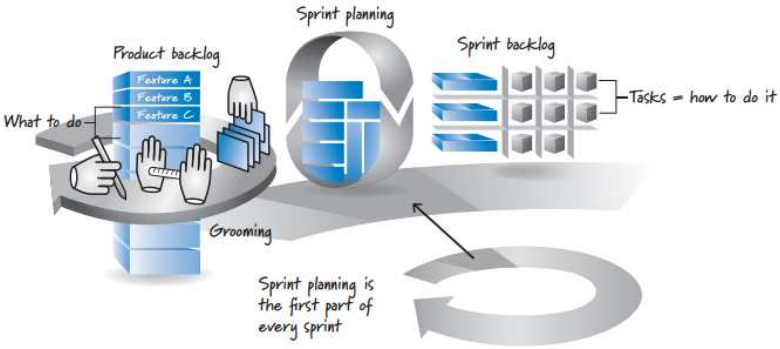
Scrum menurut Kenneth S. Rubin (2013), merupakan bagian dari *agile software development* untuk membangun sebuah produk *software* yang inovatif. Dengan menggunakan pendekatan *agile*, program dirancang dengan menggunakan *scrum framework* yaitu sebagai berikut:

1. *Product backlog*



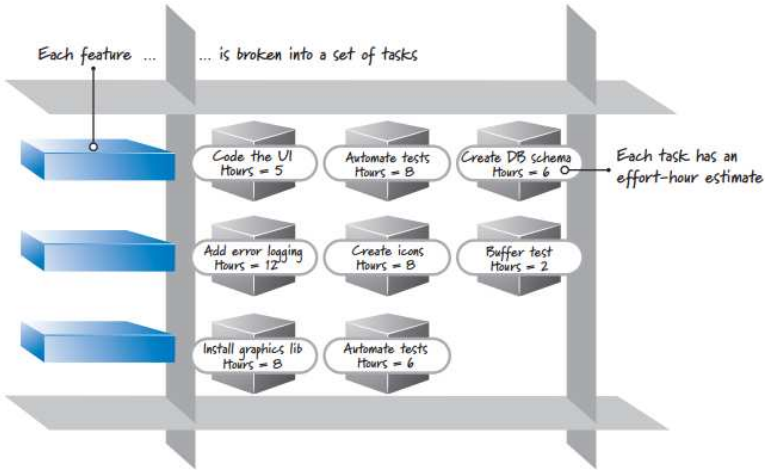
Product backlog merupakan fitur-fitur sistem yang diperlukan dalam pengembangkan aplikasi sesuai dengan *user requirements* yang diurutkan berdasarkan prioritas fitur.

2. *Sprint Planning*



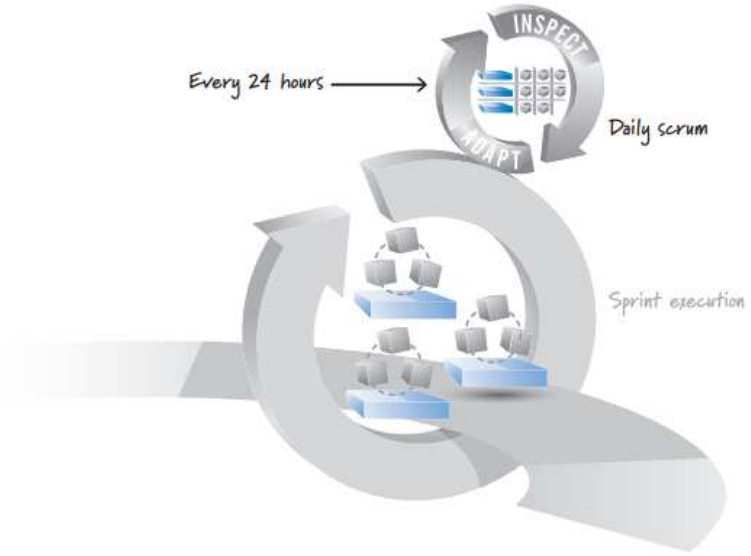
Sprint Planning merupakan penjadwalan pengerjaan *sprint backlog* dalam bentuk tanggal mulai pengerjaan *product backlog* hingga tanggal akhir pengerjaan *product backlog*.

3. *Sprint Backlog*



Sprint backlog merupakan pembagian sebuah fitur utama menjadi fitur-fitur kecil dengan menggunakan durasi waktu pengerjaan fitur tersebut berdasarkan jam perancangan fitur.

4. *Daily Scrum*



Daily Scrum merupakan sebuah diskusi singkat yang membahas mengenai *sprint* yang sedang dilaksanakan. Umumnya *daily scrum* memiliki beberapa pertanyaan yang diajukan dalam diskusi tersebut, speerti:

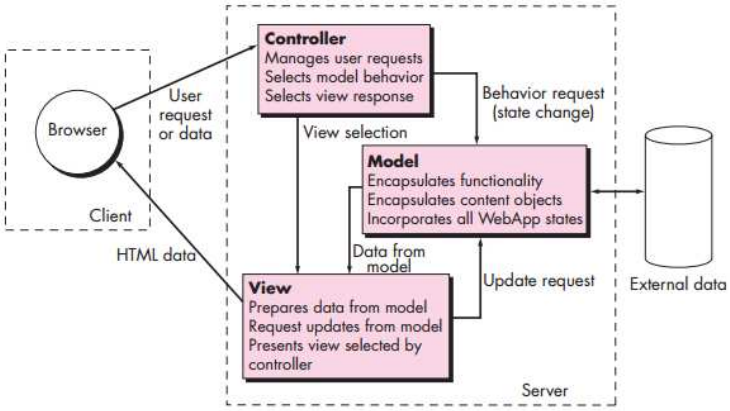
- Apa yang sudah dikerjakan pada *daily scrum* sebelumnya?
- Apa yang harus dikerjakan pada *daily scrum* berikutnya?
- Apa masalah yang menghambat pengerjaan yang sedang berlangsung?

Daily scrum ini, dapat dimanfaatkan untuk mengsosialisasikan perkembangan *sprint*, mencari solusi masalah, dan sinkronisasi tujuan dari *sprint* tersebut.

2.2.6 *Architecture Design*

Architecture Design adalah sebuah desain bagian-bagian web yang akan ditampilkan kepada *user*, dan bagan alur halaman yang sudah ditentukan. Berikut rancangan yang digunakan dalam penulisan skripsi:

2.2.6.1 *Model View Controller* (MVC)



Gambar 0.2 ***Model View Controller***

(Sumber: Software Engineering A Practitioner's Approch
Seventh Edition - Roger S. Pressman, 2010)

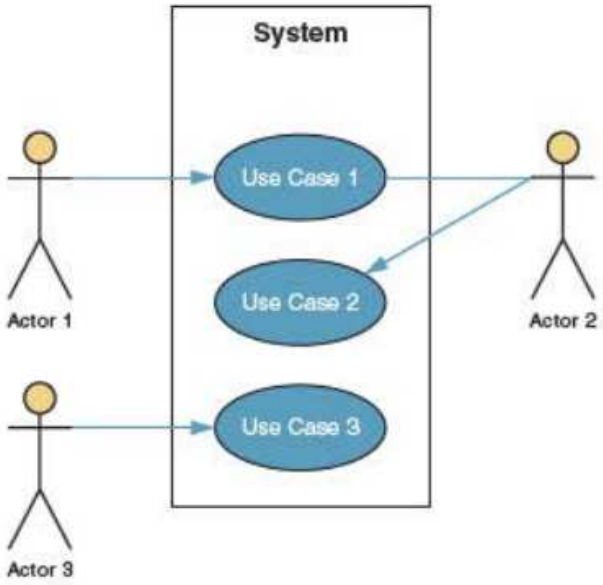
Dalam sebuah proses pertukaran data antar *browser* dan server dibutuhkan sebuah model untuk menjelaskan aliran data yang berlangsung, yang disebut *Model View Controller* (MVC). MVC adalah salah satu bagian dari *WebApp Infrastructure models* yang menjelaskan tentang tampilan fungsional dan informasi dari konten sebuah *Web Application*. MVC memiliki tiga bagian utama yaitu (Pressman, 2010, pp. 387):

1. *Model*, mengatur data-data yang masuk dari *View* berupa *update* data sebelumnya dan *model* berupa data *behavior* yang akan digunakan untuk mengakses *external* data.
2. *View*, menampilkan seluruh data dari *model* yang dijalankan melalui *controller*.
3. *Controller*, mengelola data yang masuk dari *user* yang dikembalikan ke *user* melalui *view*, dan memberikan *behavior* kepada *model*.

2.2.7 *Unified Model Language* (UML)

UML adalah sebuah penyampaian standar untuk membuat rancangan *software*. UML salah satu tujuan penggunaan UML adalah untuk visualisasi, spesifikasi, konstruksi dan dokumentasi sebuah sistem *software*. (Pressman, 2010, pp. 841).

2.2.7.1 *Use case Diagram*



Gambar 2.3 Contoh *Use Case Diagram*

(Sumber: System Analysis and Design Methods – Whitten & Bentley, 2007)

Use case merupakan pemodelan untuk menjelaskan fungsionalitas dan fitur *software* berdasarkan cara lihat atau perspektif *user* terhadap *software* tersebut. Berikut ini adalah simbol-simbol yang digunakan pada *Use case*:

1. *Actor*

   Actor adalah pengguna yang akan berinteraksi dengan sistem untuk mendapatkan hasil yang dibutuhkan

2. *Relationship*

   Relationship memiliki kegunaan untuk menjelaskan hubungan antar dua simbol yang saling berhubungan. Terdapat 5 macam *relationship*, yaitu:

a. *Associations*

Gambar 0.5 ***Associations Relationship***

(Sumber: System Analysis and Design Methods – Whitten & Bentley, 2007)

Sebuah hubungan yang menggambarkan interaksi antara *actor* dan *use case*. Terdapat dua macam relasi *association*, yaitu:

- Terdapat arah panah yang mengarah dari *user* ke *usecase*, mengartikan *use case* tersebut digunakan oleh *actor* yang bersangkutan.
- Hanya berupa garis antar *actor* dengan *use case*, menandakan hubungan *use case* dengan server luar atau *actor* yang menerima data dari *use case.*

b. *Extends*

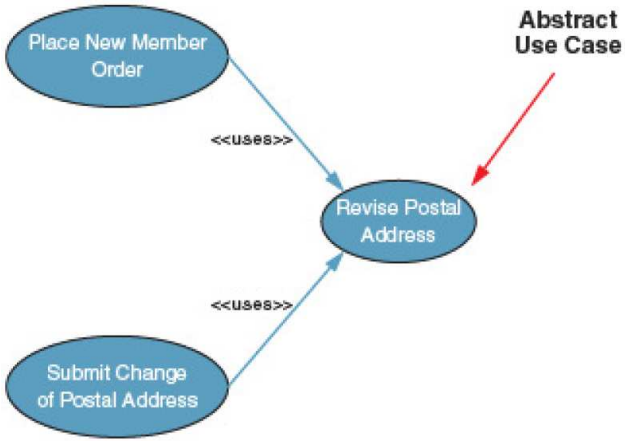
Gambar 0.6 ***Extends Relationship***

(Sum*ber: System Analysis and Design* Methods – Whitten & Bentley, 2007)

Hubungan antar *use case* yang kompleks menjadi *use case* baru, dengan tujuan untuk mudah dimengerti. *Extends* dilambangkan dengan **<<extends>>** antar *use*

case. *Extends* digunakan pada fase analisis bukan pada fase pengumpulan permintaan (*requirments phase*).

c. *Includes atau uses*

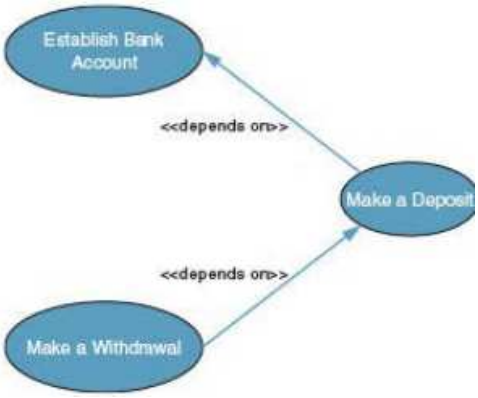


Gambar 0.7 ***Uses Relationship***

(Sumber: System Analysis and Design Methods – Whitten & Bentley, 2007)

Include menjelaskan hubugan antar *use case* yang memiliki fungsi identik menjadi *use case* baru dengan tujuan mengurangi redundansi. Sedangkan *uses* menjelaskan hubugan *abstract use case* dengan *use case* dan digunakan pada fase analisis. Dilambangkan dengan <<uses>> atau <<includes>>.

d. *Depends on*

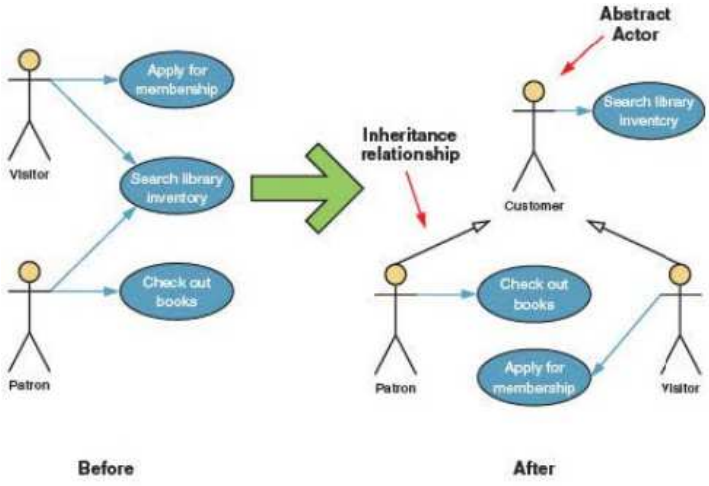


Gambar 0.8 ***Depends on Relationship***

(Sumber: System Analysis and Design Methods – Whitten & Bentley, 2007)

Menjelaskan *use case* yang harus diselesaikan terlebih dahulu sebelum *use case* berikutnya dijalankan. Dilambangkan dengan <<depends on>>.

e. *Inheritance*



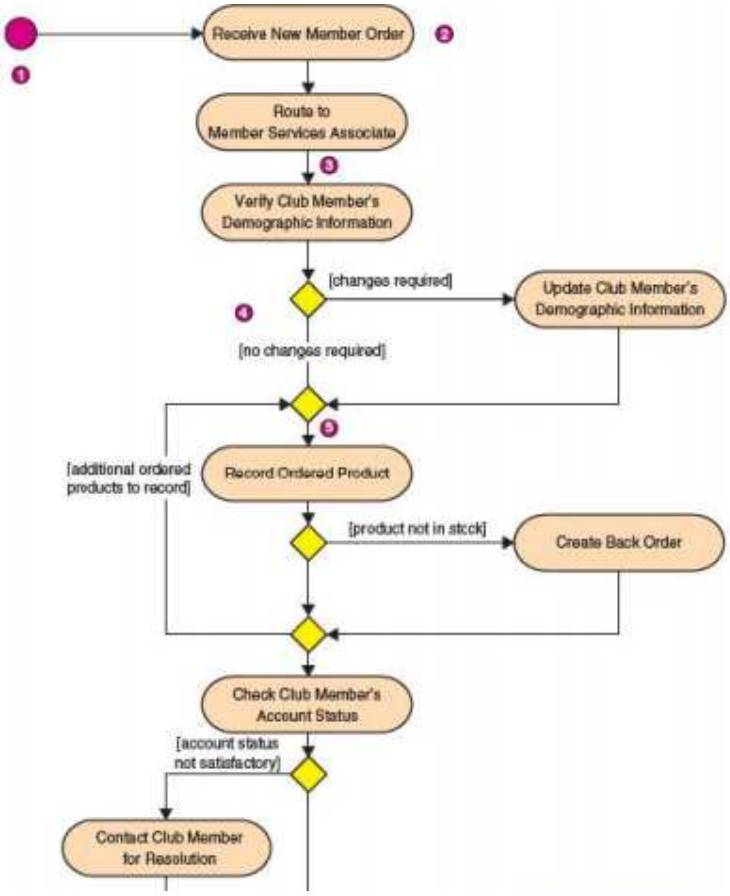
Gambar 0.9 ***Inheritance Relationship***

(Sumber: System Analysis and Design Methods – Whitten & Bentley, 2007)

Menjelaskan *actor* yang mengerjakan *use case* yang sama dalam satu *use case*. Digambarkan dengan garis yang memiliki panah.

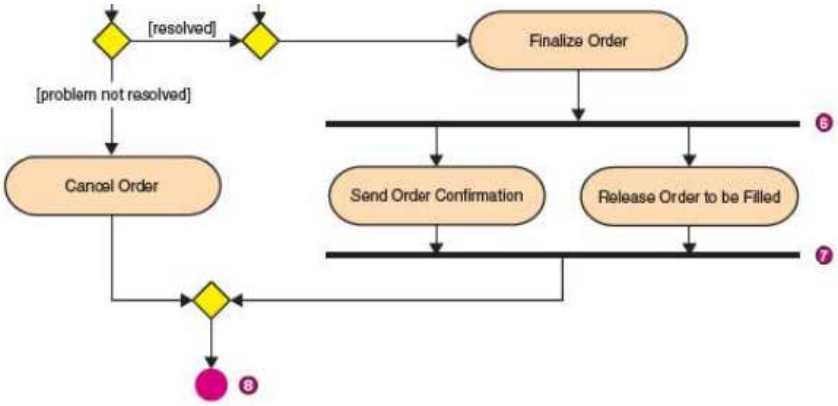
2.2.7.2 *Activity Diagram*

Activity diagram adalah diagram yang menjelaskan *behavior* sebuah sistem yang berjalan mengikuti alur *action* yang sistem tersebut jalankan. (Pressman, 2010, pp. 853).



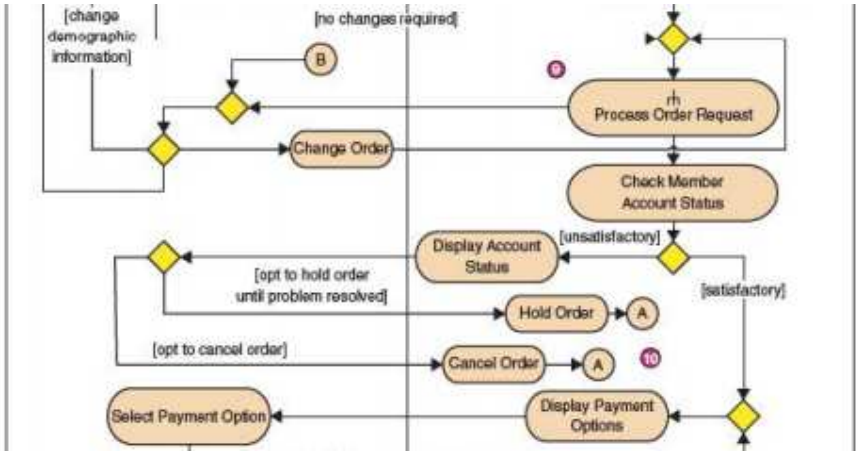
Gambar 0.10 ***Acitivity Diagram***

(Sumber: System Analysis and Design Methods – Whitten & Bentley, 2007)



Gambar 0.11 ***Activity Diagram***

(Sumber: System Analysis and Design Methods – Whitten & Bentley, 2007)



Gambar 0.12 ***Activity Diagram***

(Sumber: System Analysis and Design Methods – Whitten & Bentley, 2007)

Berikut ini adalah simbol-simbol yang digunakan pada *Activity Diagram*:

1. *Initial Node*

   Menandakan awal dari suatu proses. Dilambangkan dengan lingkaran bulat penuh.

2. *Actions*

   Menunjukan tindakan yang dilakukan. Dilambangkan dengan persegi sisi bulat.

3. *Flow*

   Arah yang menandakan alur proses yang berjalan. Dilambangkan dengan arah panah antar simbol.

4. *Decision*

   Pembagian sebuah *flow* menjadi dua atau tiga *flow*. Dilambangkan dengan belah ketupat.

5. *Merge*

   Penggabungan dua atau tiga *flow* menjadi sebuah *flow*. Dilambangkan dengan belah ketupat.

6. *Fork*

   Menjelaskan proses atau *flow* yang berjalan bersamaan. Dilambangkan dengan garis hitam horisontal yang dilalui oleh dua buah *flow*.

7. *Join*

   Akhir dari sebuah *fork* dengan menggabungkan *flow*. Dilambangkan dengan garis hitam horisontal yang dilalui oleh dua buah *flow*.

8. *Activity Final*

   Akhir dari sebuah proses yang berjalan. Dilambangkan dengan lingkaran bulat penuh.

9. *Subactivity indicator*

   Menunjukan proses yang terbagi di lain *activity diagram*. Dilambangkan dengan garpu terbalik.

10. *Connector*

    Simbol tambahan untuk mengatur kompleksitas diagram. Dilambangkan dengan lingkaran dengan huruf di dalamnya sebagai variabel.

2.2.7.3 *Class Diagram*

Menurut Bentley (2007), *Class Diagram* adalah diagram yang menjelaskan *statistic or structural view, object class,* dan hubungan antar *class* pada sebuah sistem. *Class diagram* digunakan untuk mengatur objek yang berasal dari *use case* dan digambarkan untuk mendokumentasikan hubungan antar objek. Sebuah *class* digambarkan dalam menggunakan kotak dan untuk menghubungkan antar *class* digunakan garis. Dalam *class* akan menampung data berupa nama *class*, *attribute*, dan *operation*.

Class diagram memiliki beberapa macam komponen yaitu:

1. *Visibility*

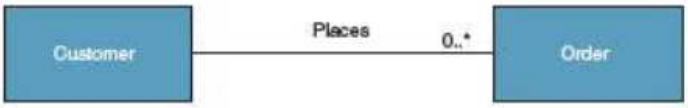
   Visibility adalah sebuah simbol yang menggambarkan sebuah *attribute* pada suatu *class* akan memiliki peran terhadap *class* lain. Memiliki 3 macam jenis *visibility* menurut Bentley (2007), yaitu:

Tabel 0.1 *Visibility*

Visibility	Nama	Deskripsi
-	*Private*	*Attribute* dapat diakses dan *method* hanya dapat dipanggil dari dalam *class* itu sendiri
+	*Public*	*Attribute* dapat diakses dan *method* dapat dipanggil oleh *class* lain
#	*Protected*	*Attribute* dapat diakses dan *method* dapat dipanggil oleh *method* lain dengan *attribute* atau *method* yang ada pada *subclass* atau *class* itu sendiri

2. *Instance level relationship*

a. *Association*

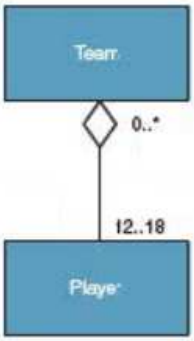


Gambar 0.13 *Association*

(Sumber: System Analysis and Design Methods – Whitten & Bentley, 2007)

Association adalah hubungan antar dua *class* yang digambarkan dengan garis yang menghubungkan antar *class* dan tidak memiliki relasi tertentu antar *class*.

b. *Aggregation*



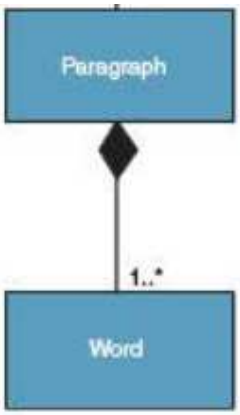
Gambar 0.14 *Aggregation*

(Sum*ber: System Analysis and*

Design Methods –Whitten & Bentley, 2007)

Aggregation adalah hubungan antar *class* yang menjelaskan suatu *class* merupakan bagian dari *class* yang dituju.

c. *Composition*



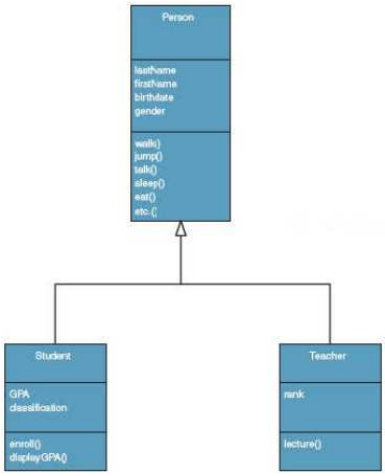
Gambar 0.15 *Composition*

(Sum*ber: System Analysis and*
Design Methods – Whitten & Bentley, 2007)

Composition merupakan bagian dari relasi *Aggregation* yang menjelaskan *class* yang menuju suatu *class* menjadi tanggung jawab pembuatan maupun dihancurkan *class* yang dituju.

3. *Class level relationship*

a. *Generalization*



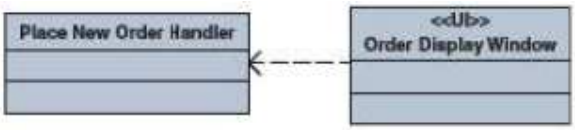
Gambar 0.16 *Generalization*

(Sumber: System Analysis and

Design Methods – Whitten & Bentley, 2007)

Generalization adalah hubungan *inheritance* atau pewarisan dari *class* induk (*superclass*) menjadi *class* turunan (*subclass*) sehingga *class* turunan memiliki *attribute* yang dimiliki oleh *class* induk.

4. *Dependency*



Gambar 0.17 *Dependency*

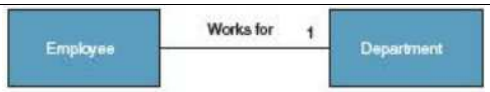
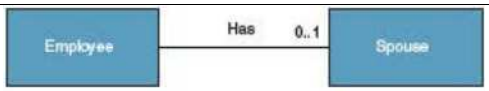
(Sum*ber: System Analysis and Design* Methods – Whitten & Bentley,2007)

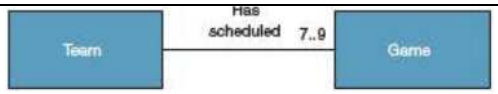
Dependency adalah sebuah relasi antar *class* yang apabila terdapat perubahan pada salah satu *class* maka *class* lain akan ikut berubah.

5. *Multiplicity*

Multiplicity adalah nilai minimum-maksimum pada sebuah objek *class* yangh berhubungan dengan *class* yang bersangkutan

Tabel 0.2 *Multiplicity*

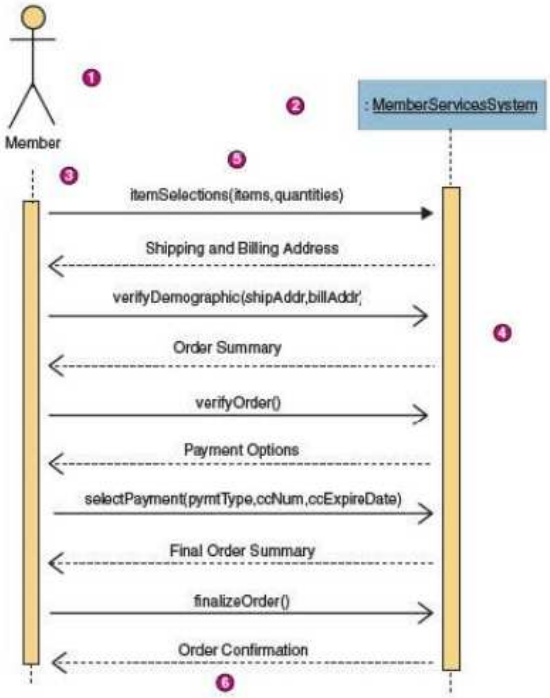
Multiplicity		Deskripsi
1		1 Pekerja bekerja pada 1 departemen
0..1		Salah seorang pekerja memiliki istri dan ada yang tidak
0..*		Seorang *customer* dapat banyak melakukan pembayaran ada yang tidak sama sekali
1..*		Dalam perkuliahan menyediakan 1 mata kuliah sampai sekian

		banyak mata kuliah
m..n		Sebuah tim mendapat 7 sampai 9 kali permainan

2.2.7.4 *Sequence Diagram*

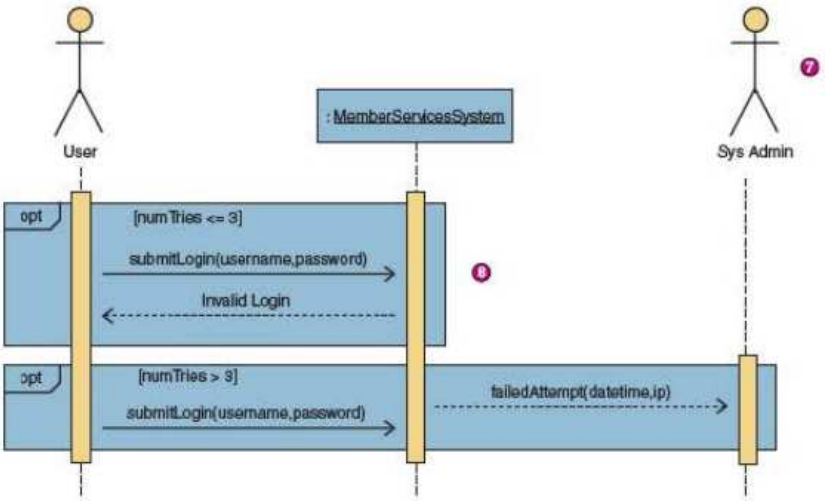
Merupakan interaksi antara *actor* atau *use case* dengan sistem yang berjalan berdasarkan alur waktu. (Bentley, 2007, pp.394).

Berikut ini adalah notasi-notasi yang digunakan pada *Sequence Diagram*:



Gambar 0.18 *Sequence Diagram*

(Sumber: *System Analysis and Design* Methods – Whitten & Bentley,2007)



Gambar 0.19 *Sequnce Diagram*

(Sum*ber: System Analysis and Design* Methods – Whitten & Bentley,2007)

1. *Actor*

   Seorang *user* yang akan berinteraksi dengan sistem, *actor* pada *sequence diagram* juga dapat menggunakan *actor* pada *use case diagaram*.

2. *System*

   Sistem yang digunakan untuk memperoleh data yang akan diproses.

3. *Lifelines*

   Alur jalan yang menjelaskan lamanya suatu *use case* atau sistem berjalan. Di gambarkan garis vertikal putus-putus di bawah *actor* dan sistem.

4. *Activation bars*

   Sebuah *bar* pada *lifelines* yang menjelaskan *use case* atau sistem tersebut sedang berinterkasi dengan sistem lain.

5. *Input messages*

   Sebuah garis yang membawa *input* baik dari *actor* maupun sistem. Di gambarkan dengan garis panah horisontal, dan memiliki format penulisan diawali dengan huruf kecil pada awal kata, huruf kapital pada inisial nama, tanpa spasi, jika terdapat *parameters* dipisahkan menggunakan koma (,).

Contoh: itemSelections

selectPayment(pymType,ccNum,ccExpDate)

6. *Output messages*

   Sebuah garis yang membawa hasil dari sistem. Di gambarkan dengan garis panah horisontal putus-putus.

7. *Receiver Actor*

   Actor lain atau sistem eksternal yang menerima *input* dari sistem yang bersangkutan.

8. *Frame*

   Sebuah penanda apabila terdapat *loop*, *fragment* tambahan, dan *optional steps*. Di gambarkan dengan sebuah kotak yang melingkupi satu bagian proses atau lebih.

2.2.8 *Black-box Testing*

Black-box testing menurut Pressman (2010) disebut juga dengan *behavioral testing* yang merupakan *testing* dengan memasukkan data yang menguji fungsi-fungsi yang ada dalam sebuah sistem. *Black-box testing* ini memiliki tujuan untuk mencari *errors* dengan beberapa kategori, yaitu:

- *Incorect or missing functions*

  Testing yang menjalankan fungsi apakah terdapat kesalahan fungsi atau kurangnya fungsi pada sebuah sistem.

- *Interface error*

  Testing yang menguji coba tampilan apakah sesuai dengan tujuan fungsi suatu sistem.

- *Error in data structure or external database access*

  Testing yang mencoba sturuktur data atau *database* eksternal sudah benar, saling terhubung, dan tersusun rapi dalam sistem.

- *Behavior or performance error*

  Testing yang menguji performa sistem apakah sistem dapat berjalan dengan efisien, dan cepat menyediakan informasi kepada *user*.

- *Initialization and termination error*

  Testing yang menguji apakah terdapat kesalahan pada initialisasi fungsi sehingga menimbulkan *error* pada sistem.

2.2.9 *Apache Bench*

Apache Bench adalah sebuah *tool* yang dimiliki oleh Apache yang digunanakan untuk menguji Apache yang digunakan dalam sistem dengan mengirimkan *request* kedalam sistem dan mengetahui seberapa cepat Apache dapat menyediakan informasi.

2.2.10 *PageSpeed Insight*

PageSpeed Insight merupakan *tool* yang digunakan untuk mengukur performa sebuah halaman *website*. *PageSpeed* menggunakan penilaian antara 0 hingga 100 poin untuk mengindikasikan seberapa baik sebuah halaman *website* dapat menyediakan informasi baik secara menyeluruh pada sebuah *page* maupun pada sebagian halaman yang sudah pernah di-*load*.

2.2.11 *Data Modeling*

Data modeling adalah teknik membuat model *requirement data* dan desain *database* sistem suatu program. (Bentley, 2007, pp. 97).

2.2.11.1 *Entity Relationship Diagram (ERD)*

Menurut Bentley (2007), ERD diajukan oleh seseorang yang bernama Peter Chen, yang bertujuan untuk mendesain *relational database system*. ERD memiliki komponen-komponen utama yaitu:

- *Data object*
- *Attributes*
- *Relationships*
- *Various type indicator*

2.3 Teori yang berkaitan dengan penelitian

2.3.1 *Performance Management*

Performance management menurut Aguinis (2009) adalah proses dimana *manager* dan pekerja bekerja bersama untuk membuat perencanaan, melakukan *monitoring* dan melakukan *review* terhadap hasil kerja pekerja dan kontribusi pekerja terhadap organisasi. Bukan sekedar laporan tahunan mengenai *performance* karyawan, *performance management* juga mengenai proses yang terus menerus dilakukan untuk membuat objektif sebuah pekerjaan, menilai kemajuan, memberikan pelatihan selama proses dan

memberikan *feedback* untuk memastikan para pekerja memenuhi objektif dari pekerjaan mereka serta tujuan karir mereka.

2.3.2 *Uniform Resource Locator* (URL)

URL adalah jenis URI (*Uniform Resource Identifier*) yang digunakan untuk mengakses lokasi jaringan sumber daya seperti halaman *web*, *file* grafis, atau file MP3. URL terdiri dari protokol, nama *domain*, dan lokasi hirarkis *file* pada *server web*. (FELKE-MORRIS, 2013, pp. 13). Format penulisan UML:

scheme://host.domain:port/path/filename

2.3.3 *8 Golden Rules*

Menurut Ben Shneiderman pada bukunya *Designing the User Interface* (2013), dalam halaman sebuah *website* harus memiliki beberapa aturan untuk dapat dikategori halaman tersebut mudah digunakan oleh *user* yang mengakses, yaitu:

1. Konsistensi
2. Melayani kebutuhan secara *universal*
3. Memiliki Timbal balik atau *feedback* yang informatif
4. Desain dialog untuk hasil akhir
5. Penanganan masalah yang sederhana
6. Mudah mengembalikan tindakan sebelumnya
7. Mendukung pengendalian internal
8. Mengurangi beban memori jangka pendek

2.3.4 *World Wide Web*

Pada tahun 1989, Tim Berners-Lee bersama timnya pada CERN mengembangkan sebuah sistem protokol baru untuk internet sehingga dapat *sharing* dokumen yang kemudian dinamakan *World Wide Web*. Sistem baru ini dikembangkan dengan tujuan untuk dapat digunakan banyak orang diseluruh dunia yang menggunakan internet dapat mengakses dokumen-dokumen dari banyak *database* yang saling terhubung. Pada akhir tahun 1990, teknologi tersebut telah dikembangkan dan digunakan diseluruh NeXT Computer bertempat di CERN. Dan pada tahun 1991, sistem tersebut

di implementasi di seluruh bagian komputer d.n diperkenalkan diseluruh dunia.

World Wide Web atau yang bisa disebut web merupakan kumpulan dari dokumen yang terhubung dengan *link*. Dokumen-dokumen ini diakses menggunakan *web browser*. (Sebesta, 2013, pp. 6).

2.3.5 *Web Application*

Web Application merupakan *hypertext files* yang saling terhubung yang menyediakan informasi berupa teks maupun grafis. *Web application* terus berkembang tidak hanya berdiri sebuah data sendiri, fungsi komputasi, dan konten *end user*, tetapi sudah menggunakan fitur *database* dan aplikasi bisnis. (Pressman, 2010, pp. 8).

2.3.6 HTML 5

Hypertext Markup Language atau HTML adalah Bahasa pemograman yang digunakan untuk mendesain tampilan sebuah web.

HTML5 merupakan pengembangan dari HTML yang memiliki beberapa fitur yang hanya dimiliki oleh HTML5 seperti: *semantic elements, web form widgets, audio and video support,* dan *canvas drawing menggunakan javascript.* (Matthew, 2013, pp. 6).

2.3.7 CSS 3

Cascading Style Sheet merupakan *style sheet language* yang digunakan *programmer* atau *user* untuk dapat mendesain halaman seperti HTML. (Bruce Lawson, 2012, pp. 10).

2.3.8 JavaScript

JavaScript digunakan pada halaman website seperti untuk *Self-completing text box, pop-up menus, slideshow,real-time mapping,* dan *webmail* yang mendukung halaman web HTML. (Matthew, 2013, pp. 451).

2.3.9 JQuery

JQuery merupakan bagian dari JavaScript. *JQuery* digunakan dalam HTML seperti untuk *event handling, animation,* dan *document manipulation.*

2.3.10 PHP

PHP dikembangkan oleh Rasmus Lerdorf pada tahun 1994, yang pada saat itu dikembangkan dengan tujuan untuk dapat menghitung jumlah pengunjung pada halaman web Lerdorf. Saat ini, PHP telah banyak digunakan dalam *website*, *web server*, dan menjadi produk *open source*. (Sebesta, 2014, pp. 366).

Menurut Welling (2009) PHP termasuk Bahasa pemograman yang berbasis *server-side scripting* yang dirancang untuk pengembangan web. PHP dapat dikombinasikan dengan bahasa pemograman HTML sehingga dapat membentuk *website* yang lebih dinamis.

2.3.11 MySQL

Merupakan *open source SQL database management system*, menggunakan Bahasa *Structure Query Language* (SQL) untuk mengakses *database*.

2.3.12 Framework CodeIgniter

Merupakan PHP *Framework* dan *toolkit* untuk membuat web yang mendukung fitur-fitur tambahan.

2.3.13 Bootstrap

Bootstrap merupakan *framework* pada bahasa pemograman HTML, CSS, dan JavaScript untuk membuat sebuah halaman web yang responsif.

2.4 Computer Application

Berikut ini aplikasi komputer yang digunakan dalam pengerjaan skripsi:

2.4.1 XAMPP

XAMPP merupakan sebuah aplikasi *freeware* yang digunakan untuk merancang aplikasi berbasis Apache seperti: MariaDB, PHP, dan Perl.

2.4.2 ATOM EDITOR

Atom merupakan *text editor* yang disediakan dengan gratis atau *freeware* yang bertujuan sebagai media untuk membuat sebuah code atau program dengan berbagai macam bahasa pemograman, seperti: HTML, PHP, dan C++. Atom memiliki beberapa fitur-fitur yang disediakan

terhadap *user* untuk dapat memudahkan dalam membangun sebuah sistem, seperti *auto complete text*, *themes*, *cross platform*, dan *multiple panes*.

2.4.3 APACHE

Apache merupakan sebuah *web server software* yang paling banyak digunakan dalam dunia komputer dan menjadi peran yang paling penting dalam perkembangan *World Wide Web*. Apache ini dibangun dengan menggunakan bahasa C++/C dan disedikan di berbagai *platform* seperti Windows, Linux, OS X, dan *platform* lainnya.

2.4.4 PHPMYADMIN

Merupakan *freeware* yang dibangun menggunakan bahasa pemograman PHP yang bertujuan untuk mengatur jalannya MySQL pada sebuah web. Umumnya, PHPMyAdmin berhubungan dengan *database* berupa tabel, kolom, relasi, indeks, dan *user* yang ditampilkan melalui *User Interface* dan dapat Menjalankan SQL.