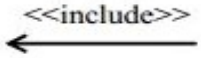
II.11.2 *Usecase* Diagram

Usecase merupakan sebuah teknik yang digunakan dalam pengembangan sebuah software atau sistem informasi untuk menangkap kebutuhan fungsional dari sistem yang bersangkutan, *Usecase* menjelaskan interaksi yang terjadi antara 'aktor' inisiator dari interaksi sistem itu sendiri dengan sistem yang ada, sebuah *Usecase* direpresentasikan dengan urutan langkah yang sederhana.

Berikut pengertian *usecase* diagram menurut Satzinger (2011:20) "*Usecase* Diagram merupakan rangkaian tindakan yang dilakukan oleh sistem, aktor mewakili *user* atau sistem lain yang berinteraksi dengan sistem yang dimodelkan".

Adapun beberapa simbol yang digunakan pada use case diagram beserta keterangannya dapat dilihat pada dibawah.

Tabel II.1 Simbol Pada *Use Case Diagram*

Simbol	Keterangan
	Aktor : Mewakili peran orang, sistem yang lain, atau alat ketika berkomunikasi dengan *use case*
	*Use case* : Abstraksi dan interaksi antara sistem dan aktor
	Association : Abstraksi dari penghubung antara aktor dengan use case
	*Generalisasi* : Menunjukkan spesialisasi aktor untuk dapat berpartisipasi dengan use case
	Menunjukkan bahwa suatu use case seluruhnya merupakan fungsionalitas dari use case lainnya
	Menunjukkan bahwa suatu use case merupakan tambahan fungsional dari use case lainnya jika suatu kondisi terpenuhi

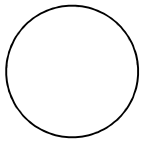
II.11.3 DFD (*Data Flow Diagram*)

Data Flow Diagram (DFD) adalah alat pembuatan model yang memungkinkan profesional sistem untuk menggambarkan sistem sebagai suatu jaringan proses fungsional yang dihubungkan satu sama lain dengan alur data, baik secara manual maupun komputerisasi. DFD ini sering disebut juga dengan nama *Bubble chart*, *Bubble* diagram, model proses, diagram alur kerja, atau model fungsi (Rosa A.S dan M.Shalahudin, 2017).

DFD ini adalah salah satu alat pembuatan model yang sering digunakan, khususnya bila fungsi-fungsi sistem merupakan bagian yang lebih penting dan kompleks dari

pada data yang dimanipulasi oleh sistem. Dengan kata lain, DFD adalah alat pembuatan model yang memberikan penekanan hanya pada fungsi sistem. Berikut merupakan tabel komponen-komponen DFD menurut Yourdon dan DeMarco.

Tabel II.2 Komponen DFD Menurut Yourdon dan DeMarco

Komponen	Simbol	Keterangan
Proses	 (Lingkaran)	Menyatakan sebuah proses dari suatu data yang masuk. Sebagaimana suatu proses, jika ada *input* maka harus ada *output*
Data Flow	(Anak Panah)	Menerangkan perpindahan data atau paket data atau informasi dari satu bagian sistem ke bagian lainnya.
Terminator	 (Segi Empat)	Mewakili entitas eksternal yang berkomunikasi dengan sistem yang sedang dikembangkan.
Data Store	 (Dua Garis Sejajar)	Menerangkan suatu tempat penyimpanan data. *Data store* hanya berkaitan dengan komponen proses.

(Sumber : Rosa A.S dan M.Shalahudin, 2017)

II.11.4 ERD (Entity Relationship Diagram)

Model ERD berfungsi untuk menggambarkan relasi dari dua file atau dua tabel yang dapat digolongkan dalam tiga macam bentuk relasi, yaitu *one-to-one*, *one-to-many*, dan *many-to-many*. Penggambaran ini akan membantu analis sistem dalam melakukan perancangan proses yang kelak akan dituangkan dalam bentuk baris-baris program. Berikut merupakan komponen-komponen ERD :

a. Entitas dan Atribut

   Entitas adalah suatu objek dan memiliki nama. Secara sederhana dapat dikatakan bahwa jika objek ini tidak ada di suatu *enterprise* (lingkungan tertentu), maka enterprise tersebut tidak dapat berjalan normal.

b. Relasi

   Relasi adalah penghubung antara satu entitas (*master file*) dengan entitas lain di dalam sebuah sistem komputer. Pada akhirnya, relasi akan menjadi file transaksi (*transaction file*) di komputer.

c. Derajat Kardinalitas (*Cardinality Degree*)

   Hubungan antar entitas ditandai pula oleh derajat kardinalitas. Fungsi dari derajat kardinalitas ini adalah untuk menentukan entitas kuat dan entitas lemah. Tiga jenis derajat kardinalitas adalah :

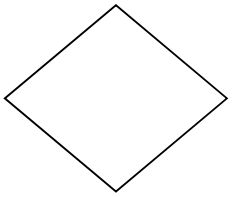
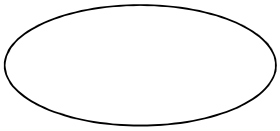
   1. One-to-one, Dilambangkan dengan 1 : 1
   2. One-to-Many, Dilambangankan dengan 1 : M dan sebaliknya
   3. Many-to-Many, Dilambangkan dengan M : M

   Entitas dengan derajat kardinalitas 1 adalah entitas lemah sehingga entitas tersebut boleh digabung saja dengan entitas yang kuat atau bisa disebut dengan derajat kardinalitas M.

d. Primary Key

   Di setiap entitas di dalam ERD harus ada atribut (*field)* yang dipilih untuk dijadikan kunci utama atribut *(primarykey/ key field)*, yaitu atribut yang dijadikan identitas yang menjamin keunikan (tidak ada yang sama) dari isi datanya.

Tabel II.3 Simbol ERD

Notasi	Arti	Penjelasan
	Entity	Entitas adalah suatu benda atau obyek di dalam dunia nyata yang dapat dikenali secara berbeda dari obyek yang lain
	Relationship	Relasi adalah hubungan di antara beberapa entitas
	Attribute	Atribut adalah sifat-sifat deskriptif yang dimiliki oleh setiap anggota dari suatu himpunan entitas.

(Sumber : Rosa A.S dan M.Shalahudin, 2017)

II.11.5 *Sequence* Diagram

Diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan message yang dikirimkan dan diterima antar objek (Shalahuddin & Sukamto, 2016). Untuk menggambarkan diagram sekuen, perancang suatu perangkat lunak harus mengetahui terlebih dahulu objek - objek yang terlibat dalam sebuah use case beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek tersebut. Selain itu, dalam pembuatan diagram sekuen juga dibutuhkan skenario yang ada pada use case.

Dalam pembuatan diagram sekuen, banyaknya diagram sekuen yang harus digambar adalam minimal sebanyak pendefinisian use case yang memiliki proses sendiri atau yang terpenting ketika semua use case yang telah didefinisikan sudah tercakup pada diagram sekuen. Sehingga, semakin banyak use case yang

didefinisikan. Maka, diagram sekuen yang harus digambarkan juga akan semakin banyak

II.12 Pengujian Perangkat Lunak

Pengujian perangkat lunak adalah elemen kritis dari jaminan kualitas perangkat lunak dan merepresentasikan kajian pokok dari spesifikasi, desain, dan pengkodean. Sejumlah aturan yang berfungsi sebagai sasaran pengujian pada perangkat lunak adalah (Sukamto, 2009) :

1. Pengujian adalah proses eksekusi suatu program dengan maksud menemukan kesalahan.
2. *Test case* yang baik adalah *test case* yang memiliki probabilitas tinggi untuk menemukan kesalahan yang belum pernah ditemukan sebelumnya.
3. Pengujian yang sukses adalah pengujian yang mengungkap semua kesalahan yang belum pernah ditemukan sebelumnya.

Karakteristik umum pengujian perangkat lunak adalah sebagai berikut (Sukamto,2009):

1. Pengujian dimulai pada level modul dan bekerja keluar kearah integrase pada sistem berbasiskan komputer.
2. Teknik pengujian yang berbeda sesuai dengan poin-poin yang berbeda pada waktunya.
3. Pengujian diadakan oleh *software developer* dan untuk proyek besar oleh *groip testing* yang *independent.*
4. *Testing* dan *debugging* adalah aktivitas yang berbeda tetapi *debugging* harus diakomodasikan pada setiap strategi *testing.*

Metode Pengujian perangkat lunak ada 3 jenis, yaitu (Sukamto,2009):

1. *White box/ Glass box* – pengujian operasi.
2. *Black box* – pengujian sistem.
3. *Use case* – membuat input dalam perancangan *blackbox* dan pengujian *statebased.*