2.1.4 Analisa dan Perancangan Sistem

2.1.4.1 Object Oriented Analysis and Design

Pendekatan *Object Oriented Analysis and Design* (OOAD) merupakan pendekatan analisa dan perancangan yang digunakan oleh penulis. Berikut ini adalah beberapa teori yang terkait dengan pendekatan ini.

2.1.4.1.1 Pengertian Object Oriented Analysis and Design

Satzinger, Jackson dan Burd (2012: 241) mendefinisikan *Object Oriented Approach* atau pendekatan berorientasi objek sebagai "pengembangan sistem berdasarkan cara pandang bahwa suatu sistem adalah sekumpulan objek yang bekerja bersama."

Satzinger, Jackson dan Burd (2012: 241) mendefinisikan suatu *Object* sebagai "sesuatu di dalam sistem informasi yang merespon suatu pesan dengan mengeksekusikan sebuah *function* atau *method."*

Satzinger, Jackson dan Burd (2012: 241) mendefinisikan *Object Oriented Analysis (OOA)* sebagai "suatu proses mengidentifikasikan dan mendefinisikan *use case* dan kumpulan *Object* atau *Class* pada sistem."

Satzinger, Jackson dan Burd (2012: 241) mendefinisikan *Object Oriented Design (OOD)* sebagai "suatu proses mendefinisikan semua tipe *Object* yang diperlukan untuk berkomunikasi dengan orang-orang dan peralatan dalam sistem, menunjukan bagaimana suatu *Object* berinteraksi untuk menyelesaikan tugas."

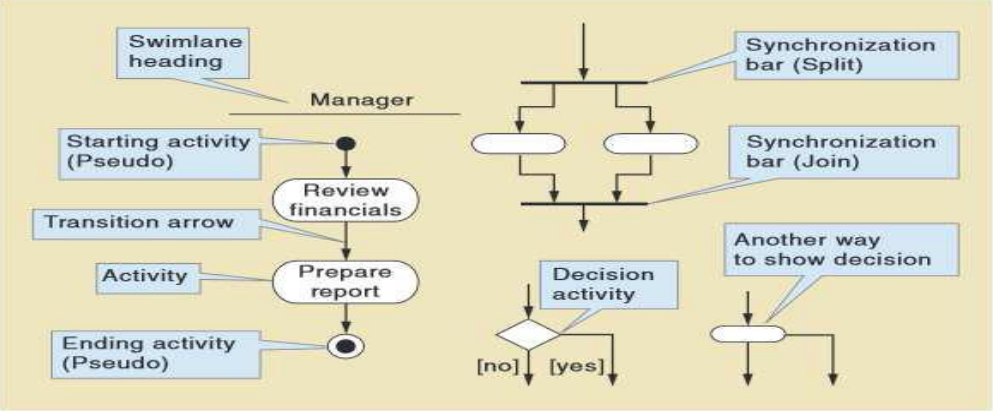
2.1.4.2 UML (*Unified Modeling Language*)

Menurut Satzinger, Jackson dan Burd (2012: 46), *Unified Modeling Language* adalah "suatu set standar konstruksi model dan notasi yang didefinisikan oleh *Object Management Group.* Dengan menggunakan UML, analis dan pengguna dapat mengerti

berbagai variasi diagram yang digunakan dalam projek pengembangan sistem." Sebelum adanya UML, tidak ada standar, sehingga diagram dapat menjadi membingungkan, karena pembuatannya berbeda dari perusahaan ke perusahaan lain, (atau dari buku ke buku). Selanjutnya penulis akan membahas mengenai teori model yang akan digunakan dalam penulisan, yaitu meliputi *Activity Diagram, Domain Model Class Diagram, Multilayer Sequence Diagram, Use Case Diagram,* dan *Use Case Description.*

2.1.4.3 *Activity Diagram*

Menurut Satzinger, Jackson dan Burd (2012: 57), "sebuah *Activity Diagram* mendeskripsikan aktivitas dari user (atau sistem); siapa yang mengerjakan setiap aktivitas, dan alur berurutan akan aktivitas tersebut." Berikut adalah simbol-simbol yang digunakan dalam *Activity Diagram:*

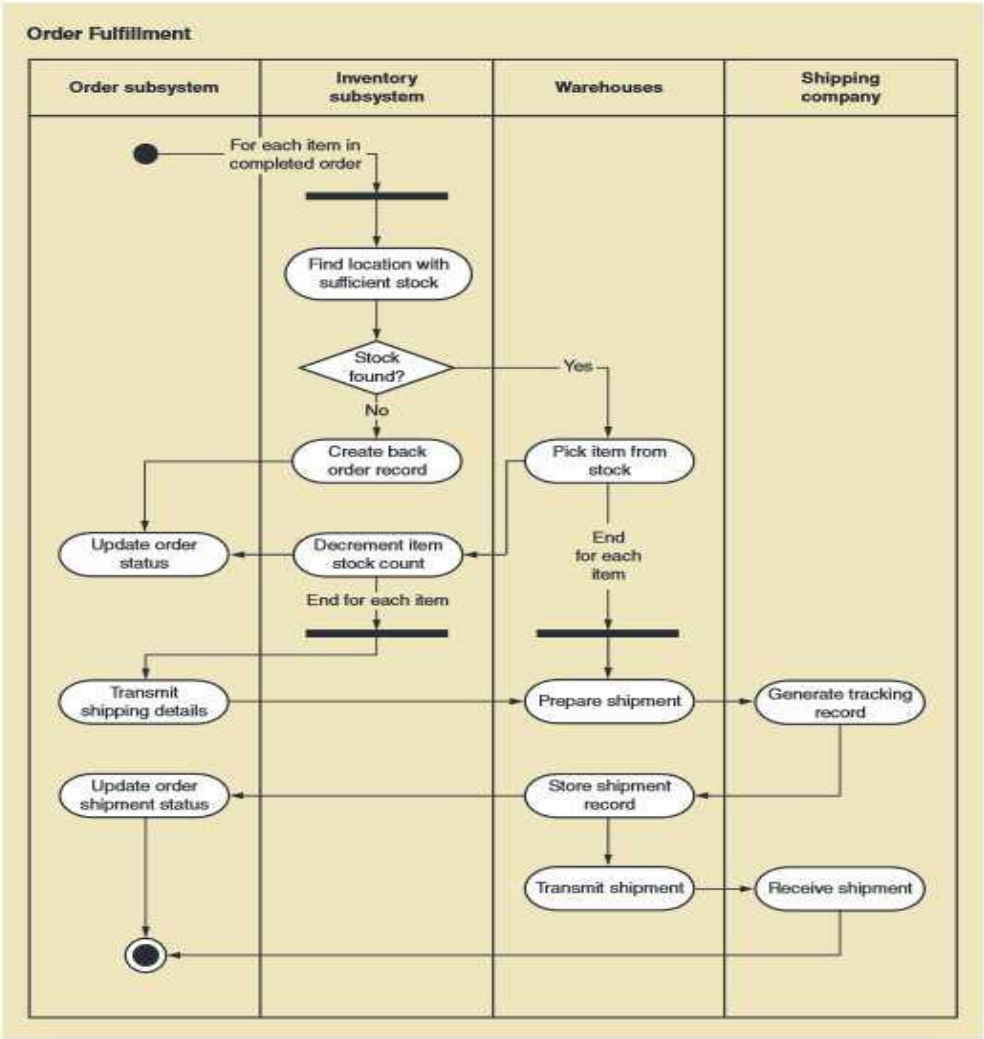


Gambar 2-1 : Simbol Activity Diagram

Sumber: *Satzinger, Jackson dan Burd (2012: 57)*

Gambar diatas menunjukan simbol-simbol dasar yang digunakan pada *Activity Diagram.* Lingkaran oval menggambarkan suatu aktivitas. Panah menggambarkan alur berurutan dari setiap aktivitas. Lingkaran hitam menggambarkan awal dan akhir dari workflow. Bentuk diamond menggambarkan decision point dimana alur akan mengikuti

satu arah atau yang lainya. Swimlane menunjukan siapa yang mengerjakan aktivitas tersebut. Garis hitam tebal menggambarkan *synchronization bar* yang berfungsi menunjukan aktivitas yang dilakukan secara bersamaan. Berikut adalah contoh *Activity Diagram* yang menggambarkan proses *online checkout* :



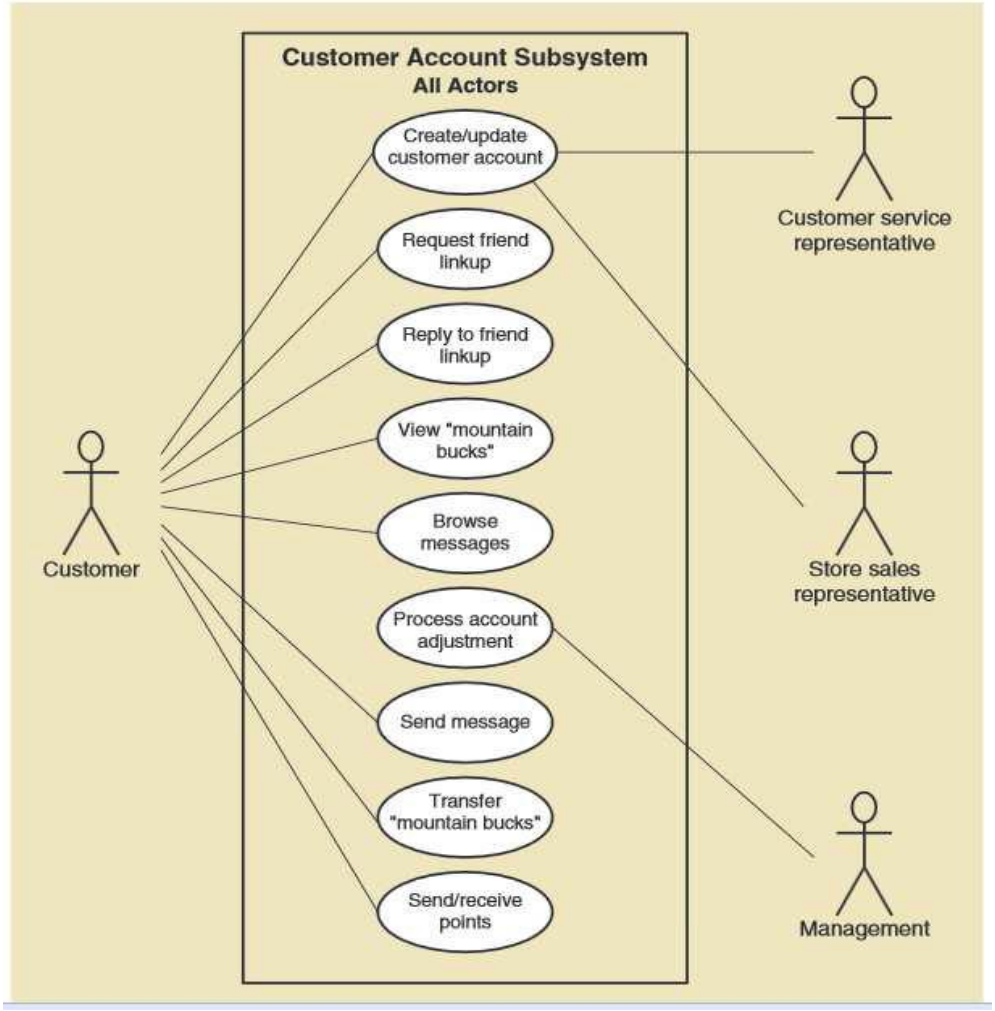
Gambar 2-2* : *Activity Diagram; Online Checkout

Sumber: Satzinger, Jackson dan Burd (2012: 59)

2.1.4.4 *Use Case Diagram*

Menurut Satzinger, Jackson dan Burd (2012: 69), sebuah *Use case* adalah "sebuah aktivitas yang dilakukan oleh sistem, sebagai respon akan *request* yang dilakukan oleh user."

Menurut Satzinger, Jackson dan Burd (2012: 78), *Use Case Diagram* adalah "model UML yang digunakan untuk menggambarkan hubungan antar *Use case* dengan *Actor* yang mengerjakanya."



***Gambar 2-3* : *Use Case Diagram* 'Customer Account Subsystem'**

Sumber: Satzinger, Jackson dan Burd (2012: 82)

Gambar diatas menunjukan contoh *Use Case Diagram*. *Actor* digambarkan dengan *stick figure*, dan garis lurus digunakan untuk menggambarkan relasi antar *Actor* dan *use case*. Dalam relasi antar *use case*, terdapat 2 jenis hubungan khusus, yaitu:

a. <<includes>>; yaitu hubungan antar *use case* yang menunjukan bahwa satu *use case* termasuk dalam *use case* lain.

b. <<extends>>; yaitu hubungan antar *use case* yang menunjukan bahwa satu *use case* memiliki kemungkinan untuk dilanjutkan ke *use case* lain.

2.1.4.5 *Fully Developed Use Case Description*

Satzinger, Jackson dan Burd (2012: 121) mengemukakan bahwa *Use Case Description* "adalah suatu model *textual* yang mendaftarkan dan mendeskripsikan proses apa saja yang dilakukan dalam sebuah *use case*."

Menurut Satzinger, Jackson dan Burd (2012: 122) *Fully Developed Use Case Description* adalah "metode yang paling formal dalam mendokumentasikan sebuah *use case*."

Dalam sebuah *Fully Developed Use Case Description,* baris pertama digunakan untuk menjelaskan nama *use case*. Baris kedua digunakan untuk menjelaskan *scenario* yang ada pada *use case* tersebut. Baris ketiga digunakan untuk menjelaskan *event* apa yang memicu terjadinya *use case*. Baris keempat digunakan untuk menjelaskan *brief description* atau deskripsi singkat akan *use case* tersebut. Baris kelima digunakan untuk menjelaskan *actor* yang terlibat dalam *use case*. Baris keenam digunakan untuk menjelaskan *use case* lain yang terlibat dengan *use case*. Baris ketujuh digunakan untuk menjelaskan *stakeholder* yang terlibat dalam *use case*. Baris kedelapan - *preconditions,* menjelaskan keadaan apa yang harus dipenuhi sebelum *use case* dilakukan. Baris kesembilan - *postconditions,* menjelaskan keadaan apa yang harus dipenuhi setelah *use case* dijalankan. Baris kesepuluh digunakan untuk menjelaskan alur aktivitas antara *actor* dan sistem. Baris kesebelas digunakan untuk menjelaskan *exception conditions,* yaitu keadaan alternatif yang mungkin terjadi. Berikut adalah contoh sebuah *Fully Developed Use Case Description* yang menggambarkan *use case create customer account :*

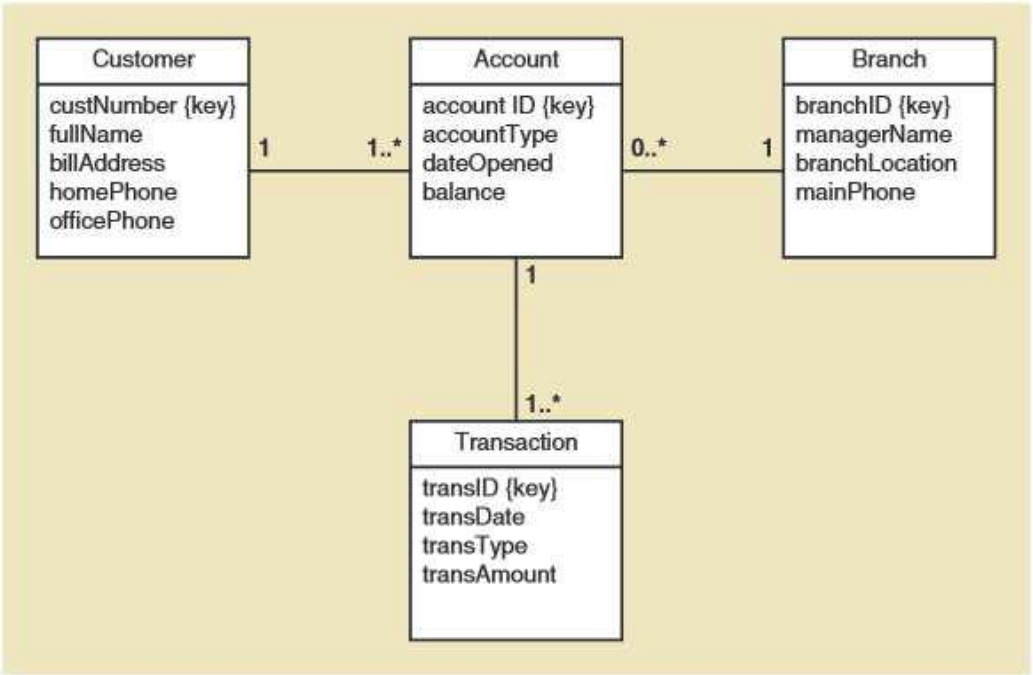
Use case name:	*Create customer account.*	
Scenario:	Create online customer account.	
Triggering event:	New customer wants to set up account online.	
Brief description:	Online customer creates customer account by entering basic information and then following up with one or more addresses and a credit or debit card.	
Actors:	Customer.	
Related use cases:	Might be invoked by the *Check out shopping cart* use case.	
Stakeholders:	Accounting, Marketing, Sales.	
Preconditions:	Customer account subsystem must be available. Credit/debit authorization services must be available.	
Postconditions:	Customer must be created and saved. One or more Addresses must be created and saved. Credit/debit card information must be validated. Account must be created and saved. Address and Account must be associated with Customer.	
Flow of activities:	**Actor**	**System**
	1. Customer indicates desire to create customer account and enters basic customer information.	1.1 System creates a new customer. 1.2 System prompts for customer addresses.
	2. Customer enters one or more addresses.	2.1 System creates addresses. 2.2 System prompts for credit/debit card.
	3. Customer enters credit/debit card information.	3.1 System creates account. 3.2 System verifies authorization for credit/debit card. 3.3 System associates customer, address, and account. 3.4 System returns valid customer account details.
Exception conditions:	1.1 Basic customer data are incomplete. 2.1 The address isn't valid. 3.2 Credit/debit information isn't valid.	

Gambar 2-4 : Fully Developed Use Case Description 'Create Customer'

Sumber: *Satzinger, Jackson dan Burd (2012: 123)*

2.1.4.6 *Domain Model Class Diagram*

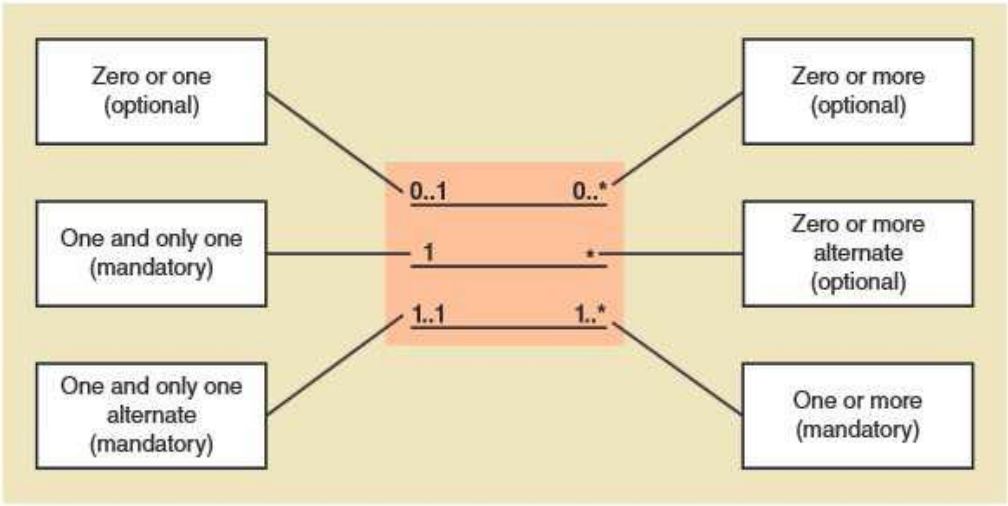
Menurut Satzinger, Jackson dan Burd (2012: 101) sebuah *class* adalah "sebuah kategori atau klasifikasi dari sebuah benda atau objek." Sedangkan *domain class* adalah "sebuah *class* yang mendeskripsikan objek dari *problem domain."Domain Model Class Diagram* adalah sebuah *class diagram* yang hanya memasukan *class* dari *problem domain.* Berikut adalah contoh gambar yang menggambarkan sebuah *Domain Model Class Diagram* untuk sebuah bank :



***Gambar 2-5* : *Domain Model Class Diagram* untuk sebuah bank**

Sumber: Satzinger, Jackson dan Burd (2012: 103)

Dan berikut adalah gambar yang menjelaskan tentang *multiplicity* pada *class diagram :*

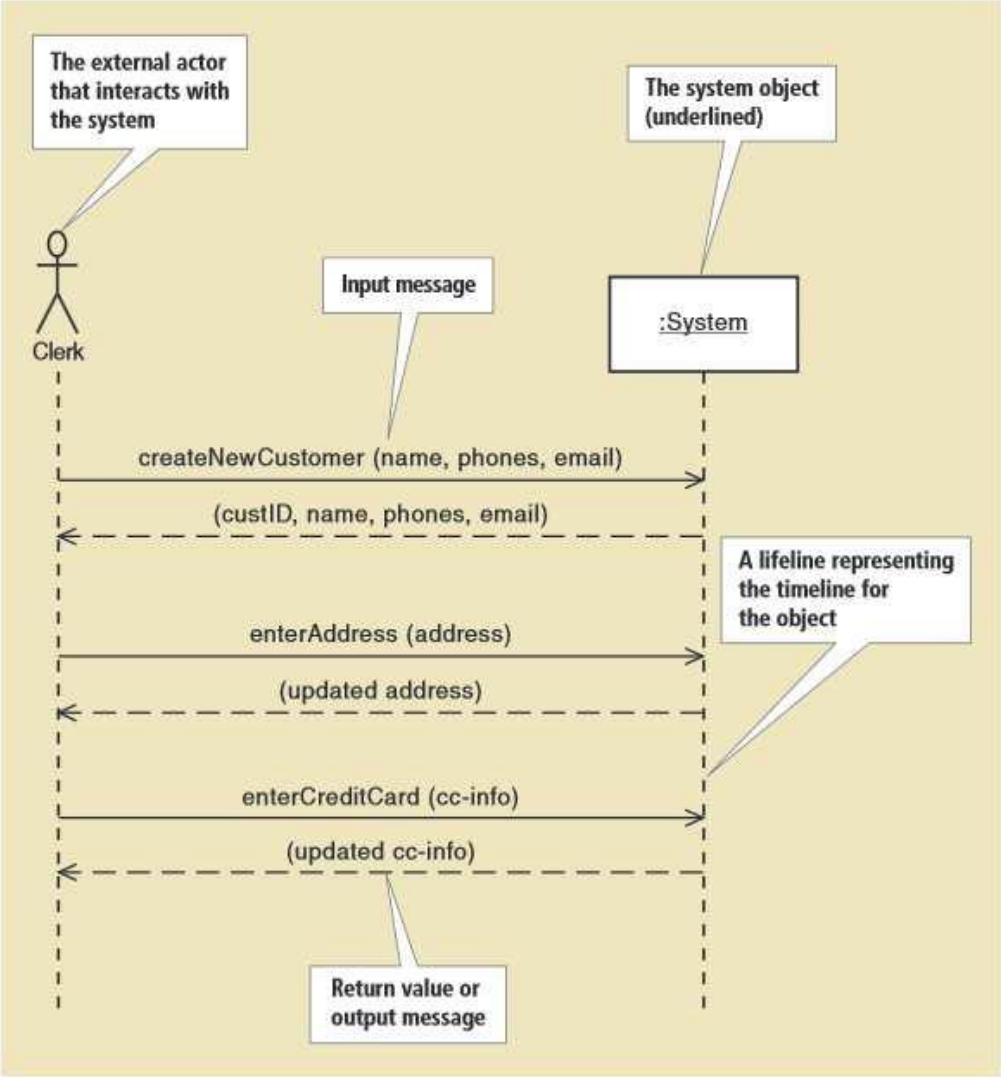


Gambar 2-6 : Multiplicity pada Class Diagram.

Sumber: *Satzinger, Jackson dan Burd (2012: 102)*

2.1.4.7 *Multilayer Sequence Diagram*

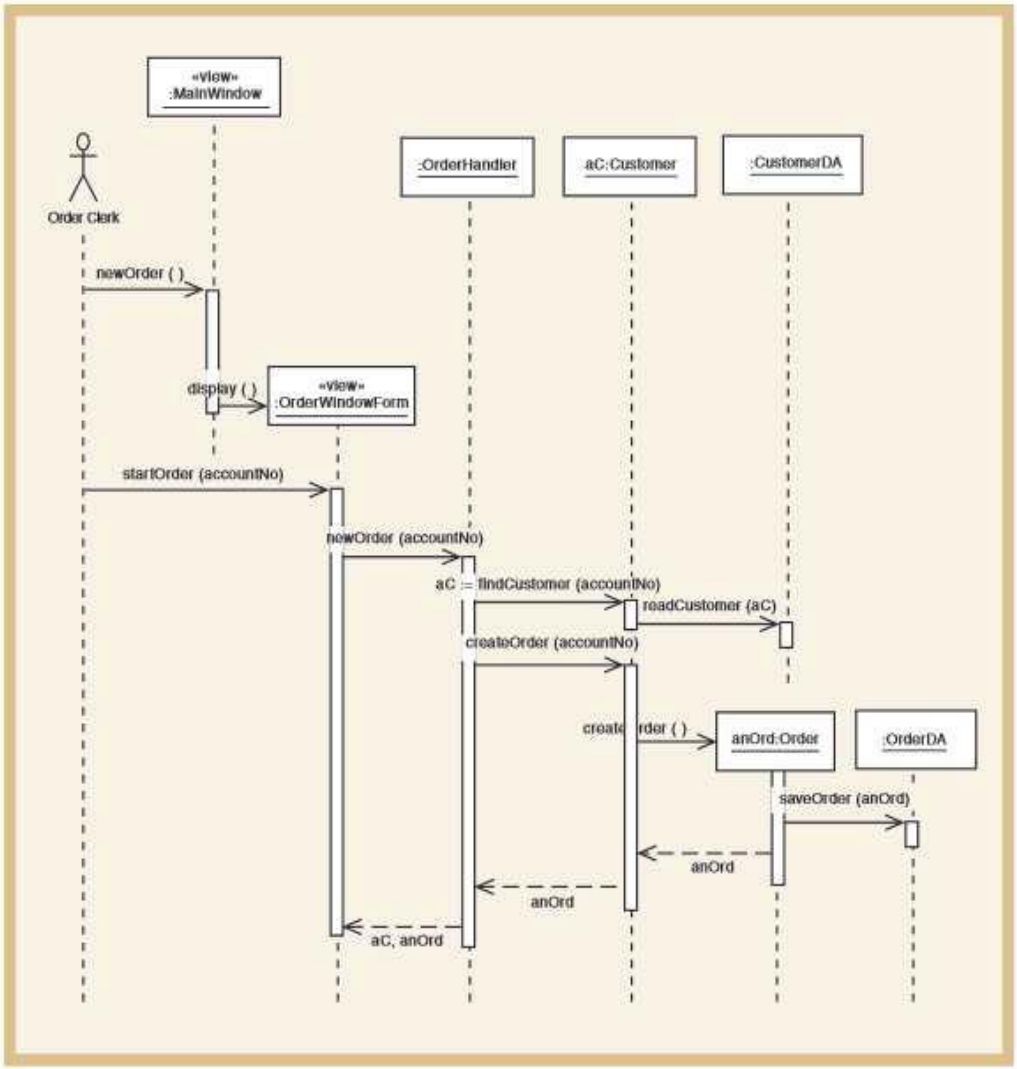
Menurut Satzinger, Jackson dan Burd (2012: 332) *Sequence Diagram* adalah "sebuah tipe diagram interaksi yang menekankan urutan dari pesan yang dikirimkan antara objek untuk sebuah *use case* tertentu." Berikut adalah contoh gambar yang menggambarkan sebuah *System Sequence Diagram :*



Gambar 2-7 : SSD untuk use case 'Create Customer Account'

Sumber: Satzinger, Jackson dan Burd (2012: 333)

Multilayer Sequence Diagram adalah "*Sequence Diagram* yang menggambarkan interaksi antar objek dalam sistem dengan menggunakan *View Layer, Domain Layer,* dan *Data Layer*." Berikut adalah contoh gambar *Multilayer Sequence Diagram* :



Gambar 2-8 **: Contoh** ***Multilayer Sequence Diagram*** **menggunakan** ***View Layer, Domain Layer,*** **dan** ***Data Layer.***

Sumber: Satzinger, Jackson dan Burd (2010: 454)