Diagram: Business Domain Model

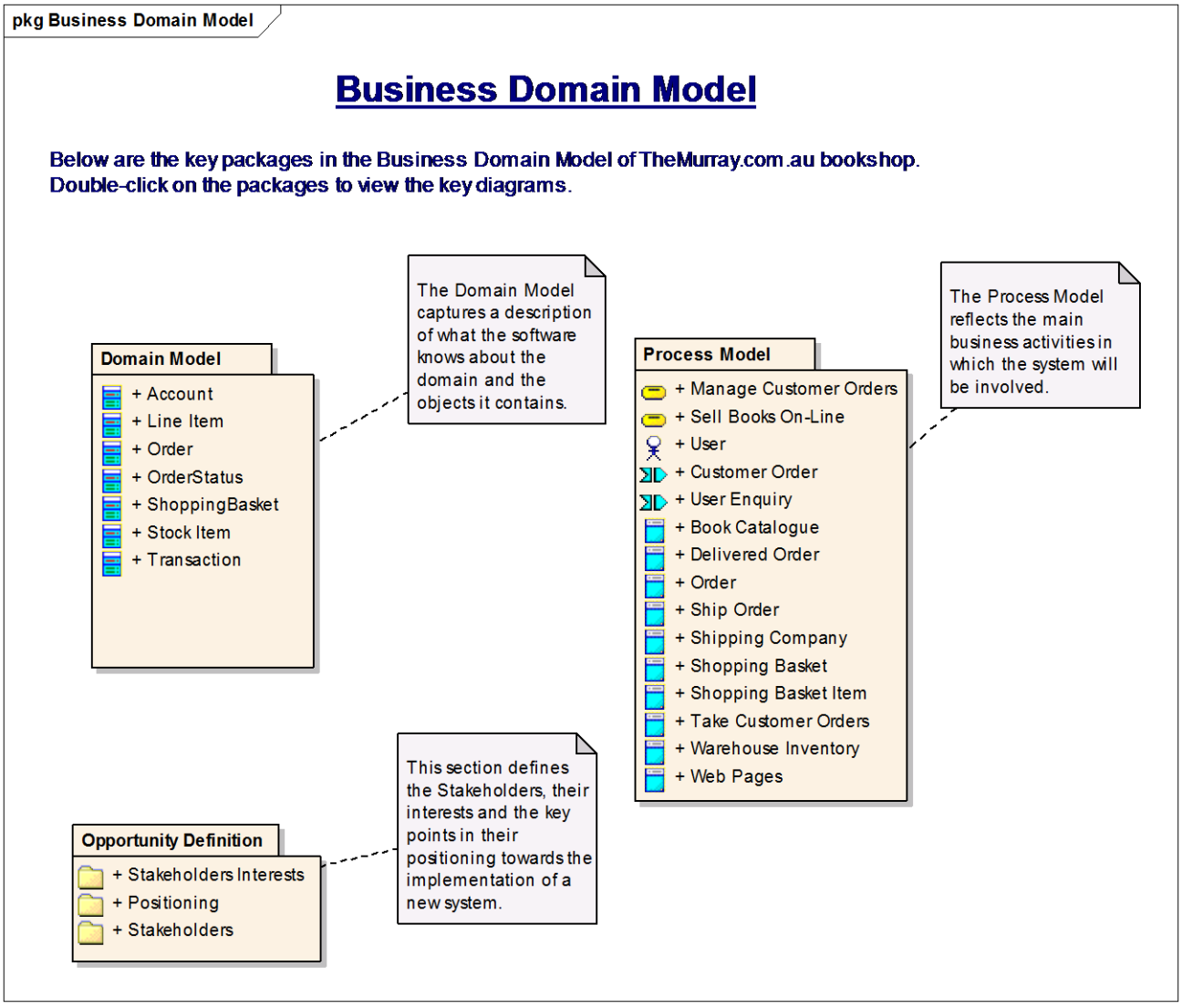


Diagram: Domain Model

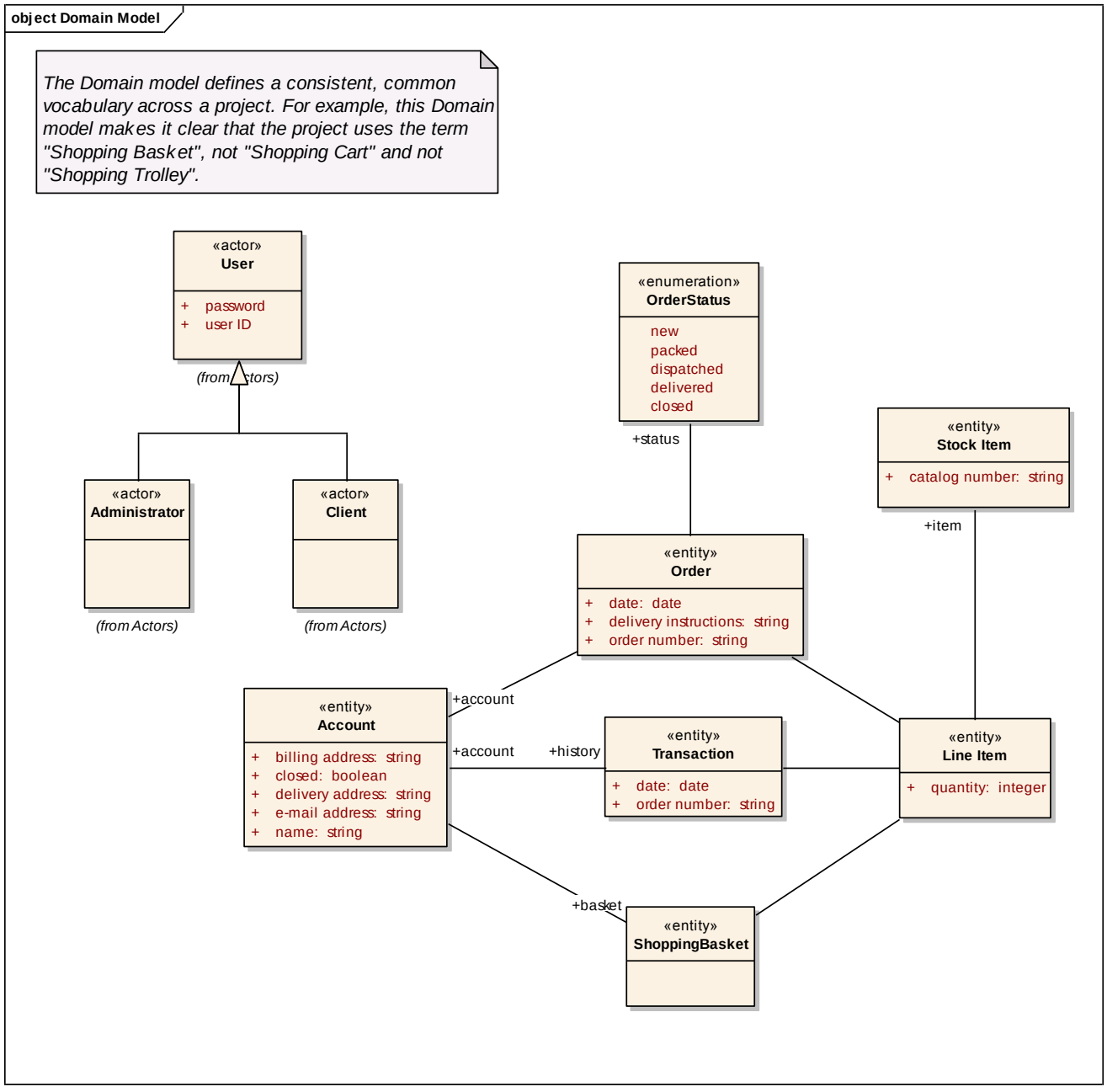


Diagram: Opportunity Definition

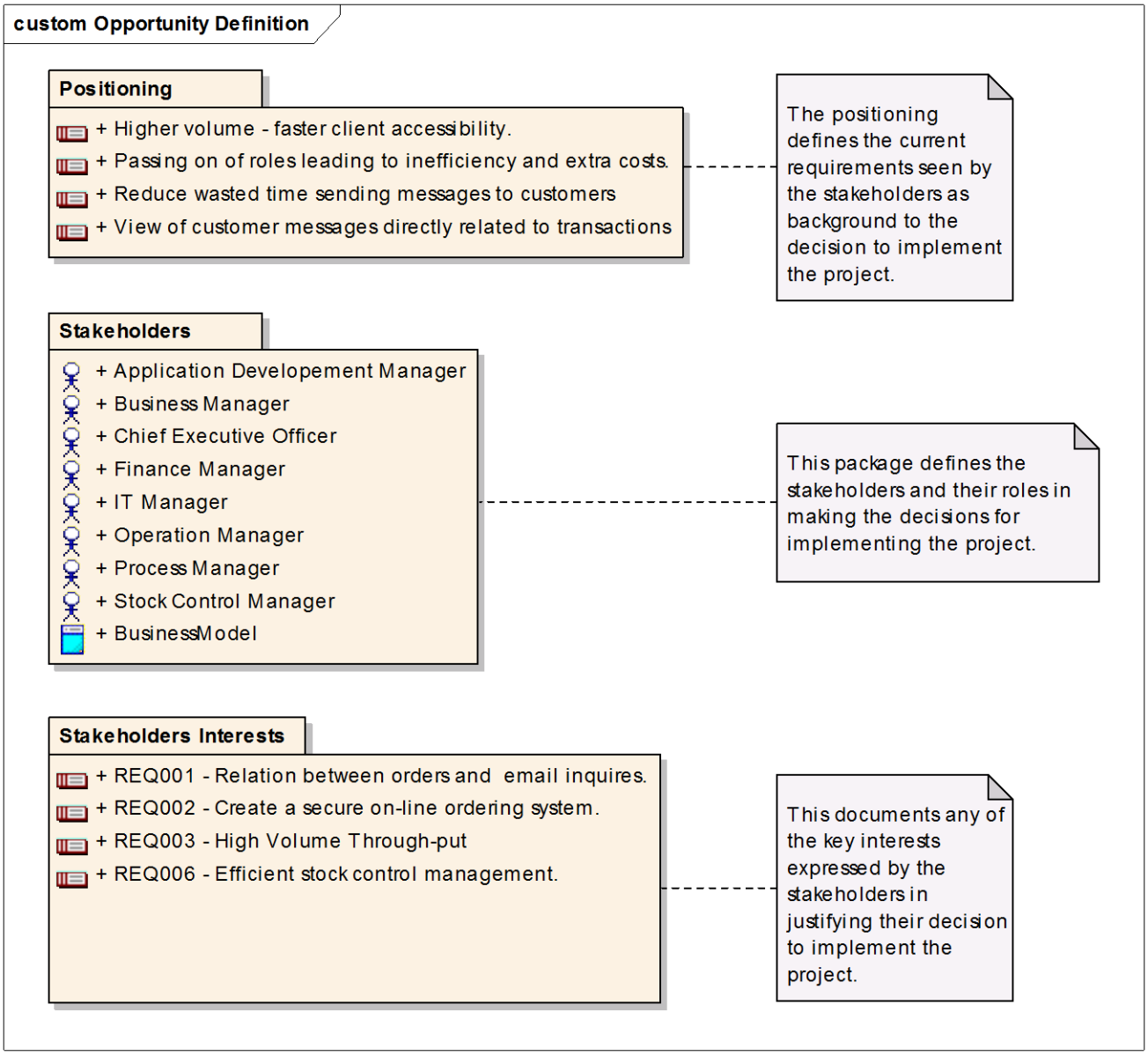


Diagram: Stakeholders

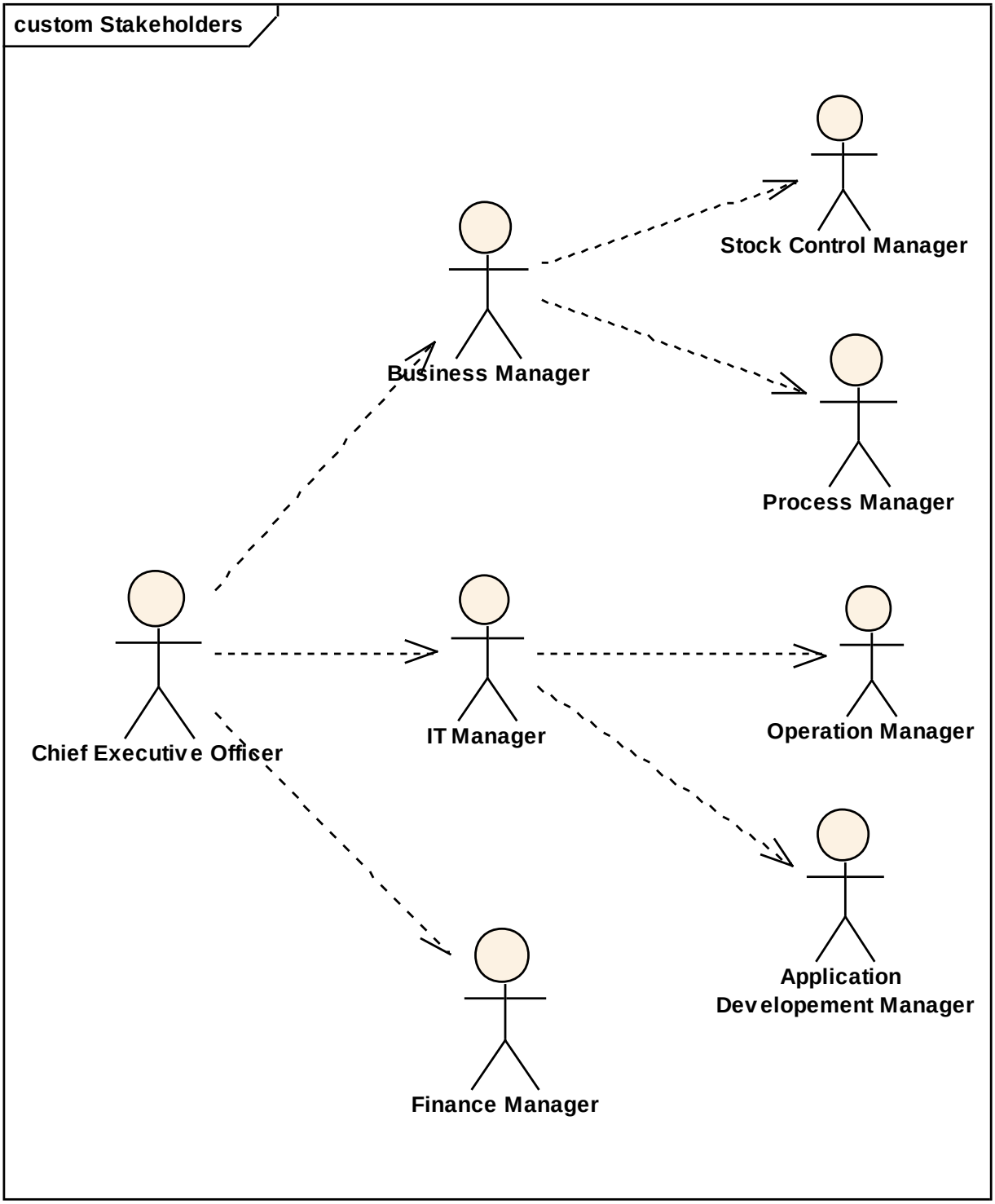


Diagram: Stakeholders Images

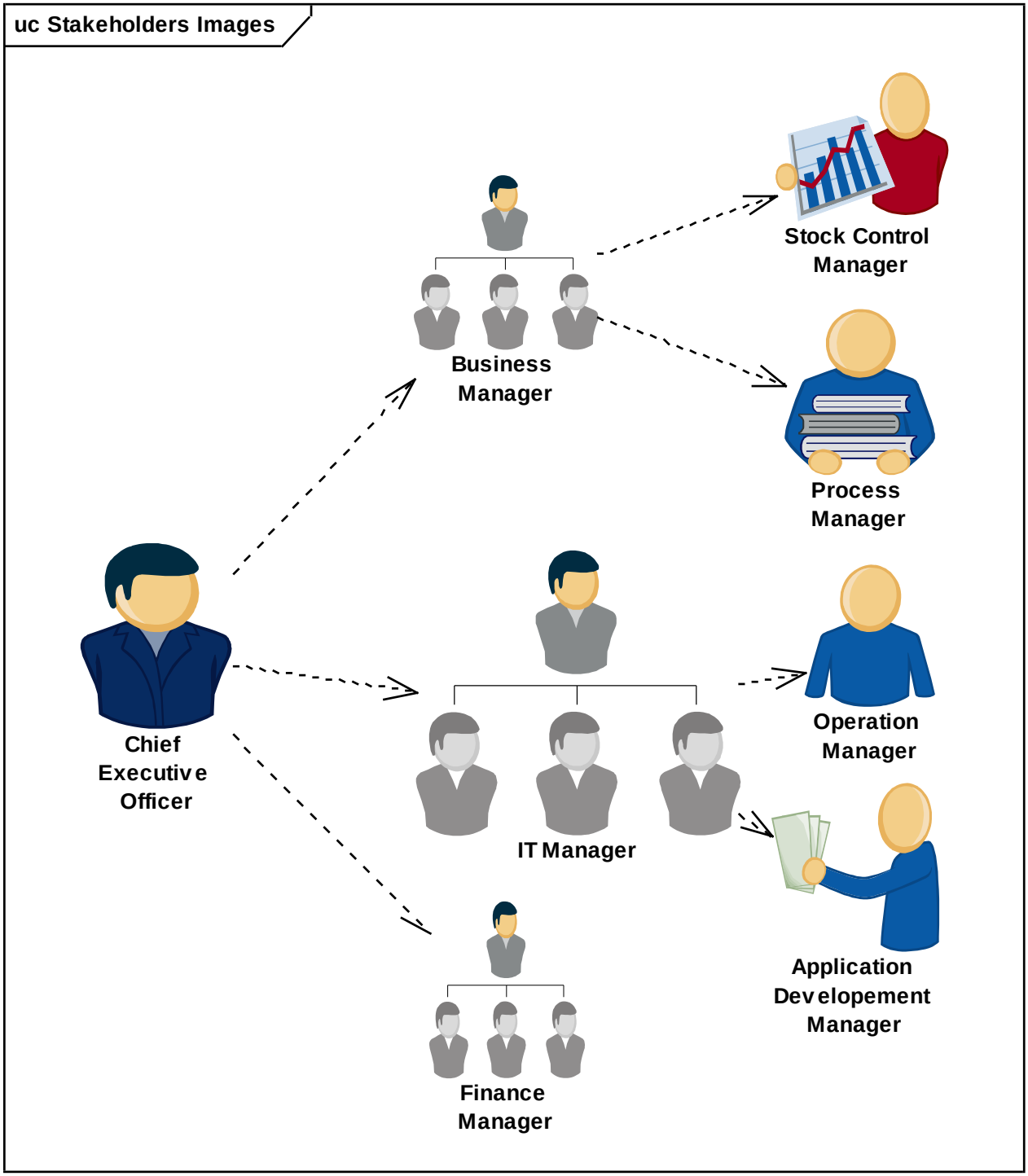


Diagram: Stakeholders Interests

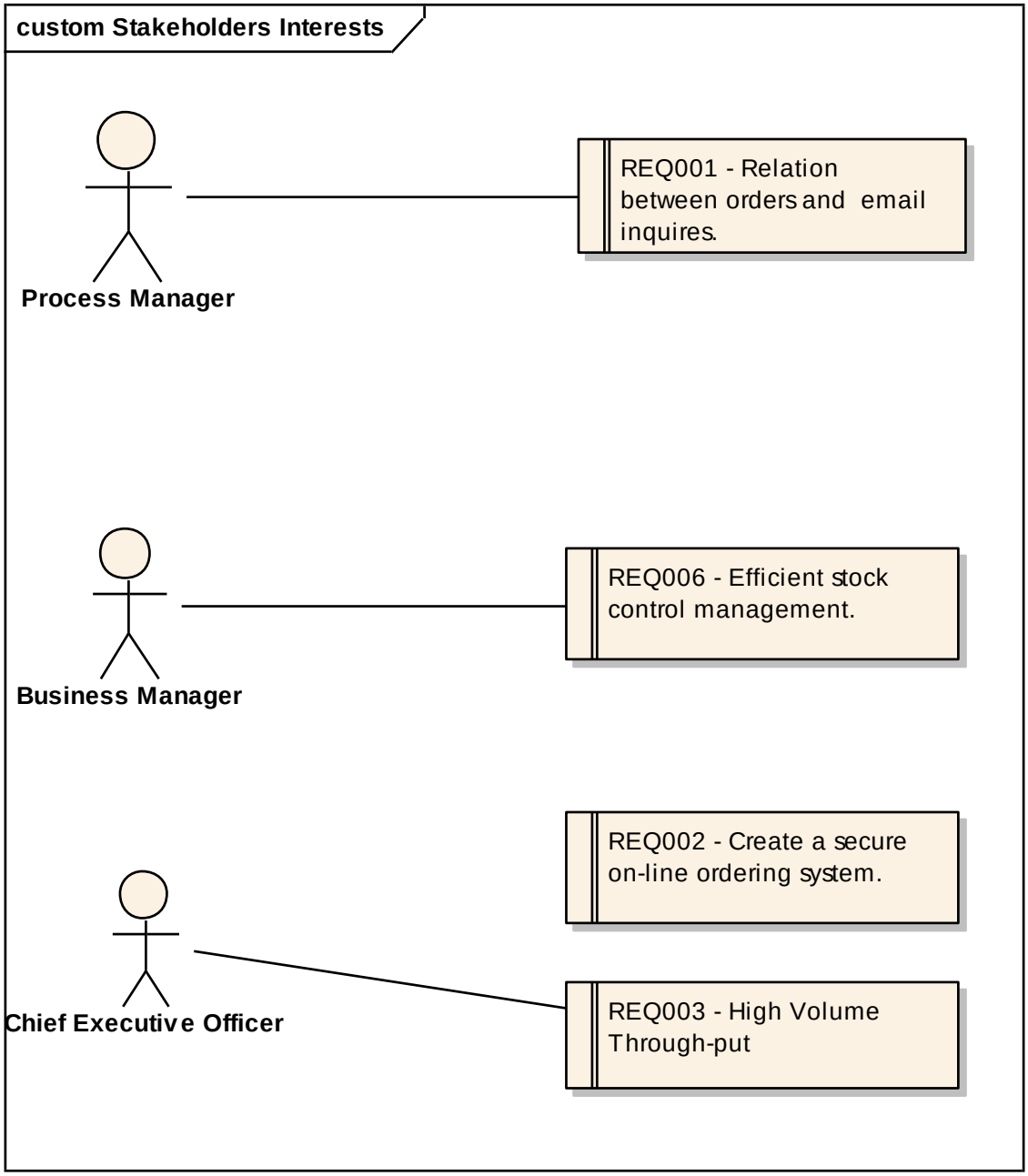


Diagram: Positioning

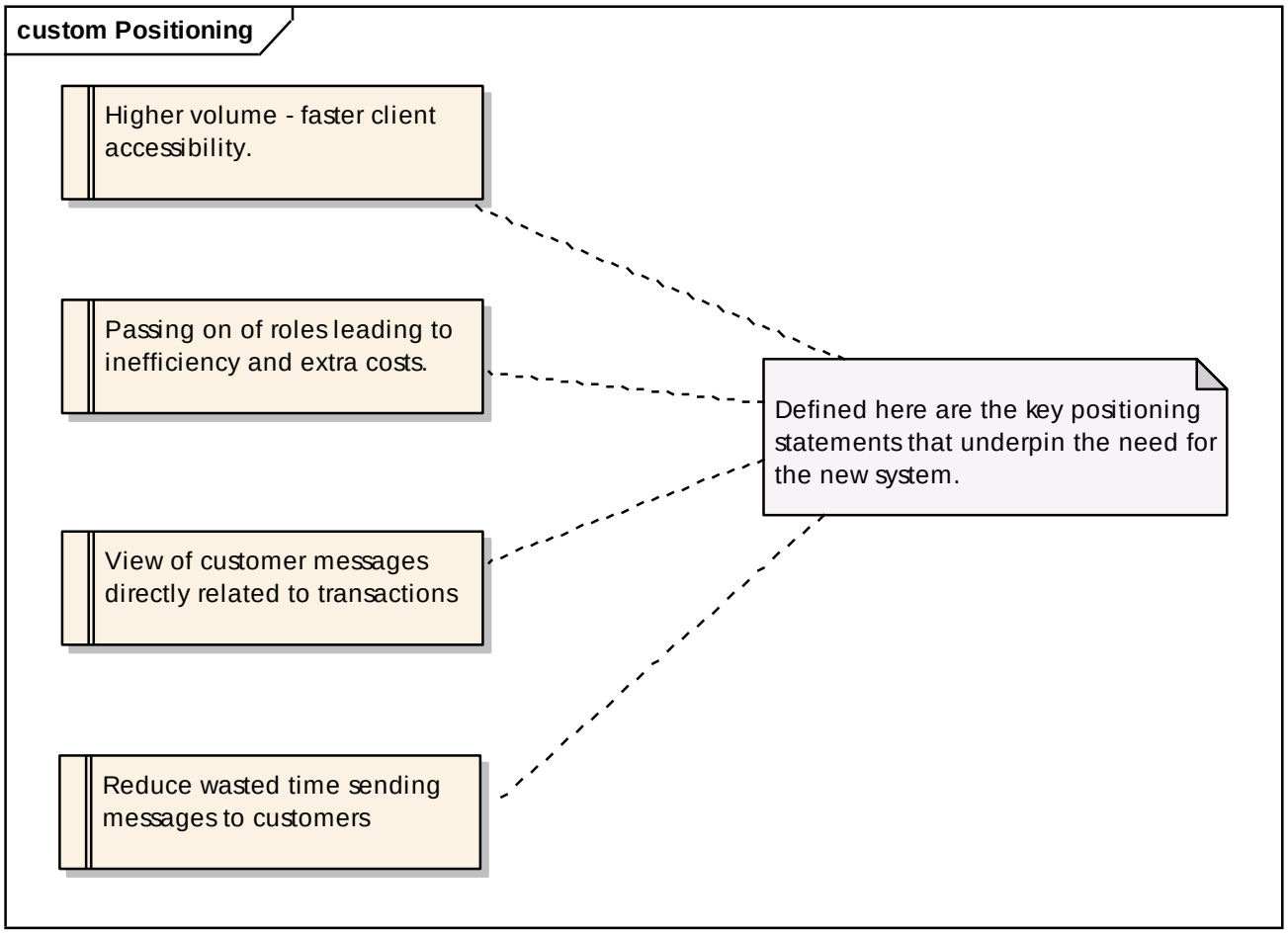


Diagram: Process Model

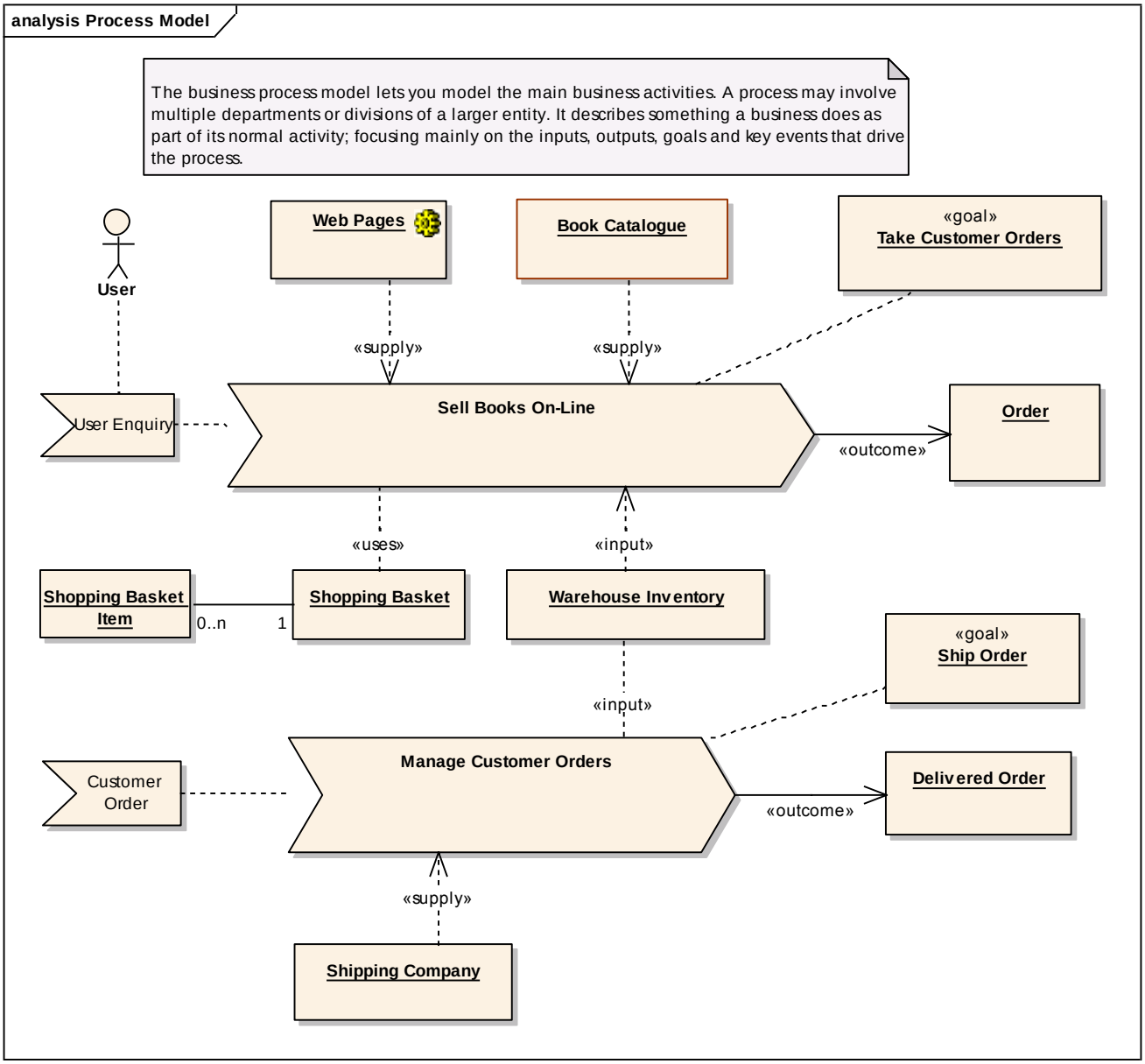


Diagram: Requirements Model

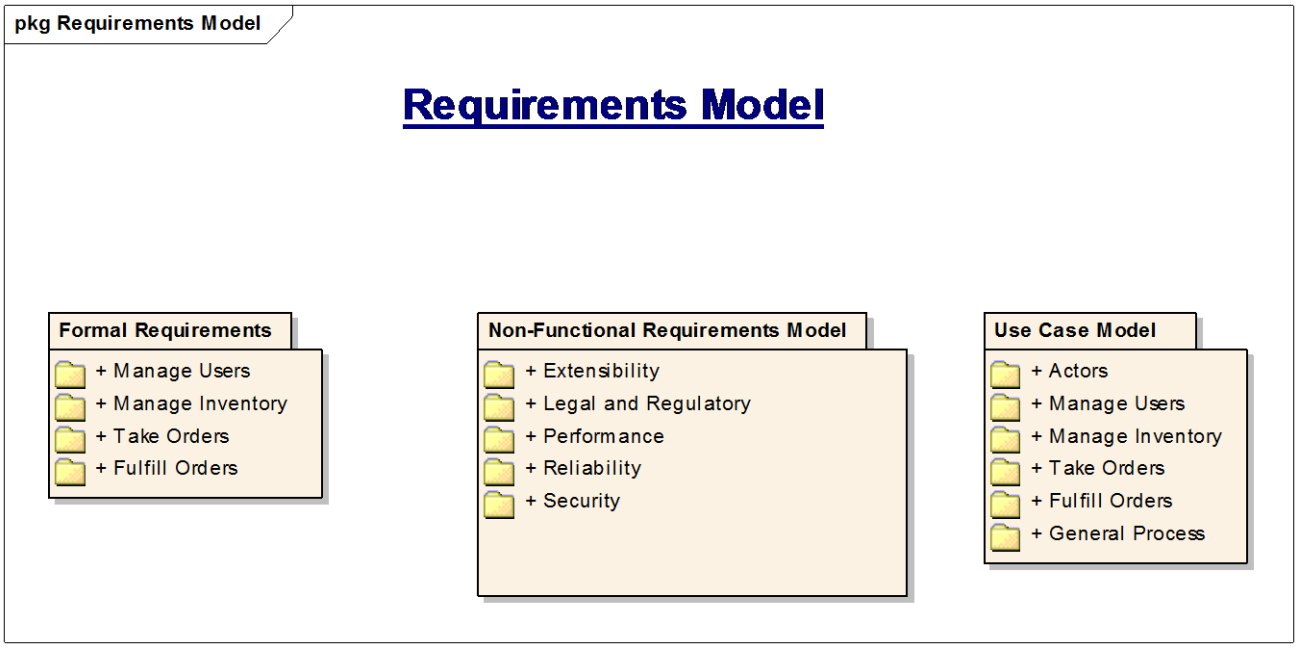


Diagram: Formal Requirements

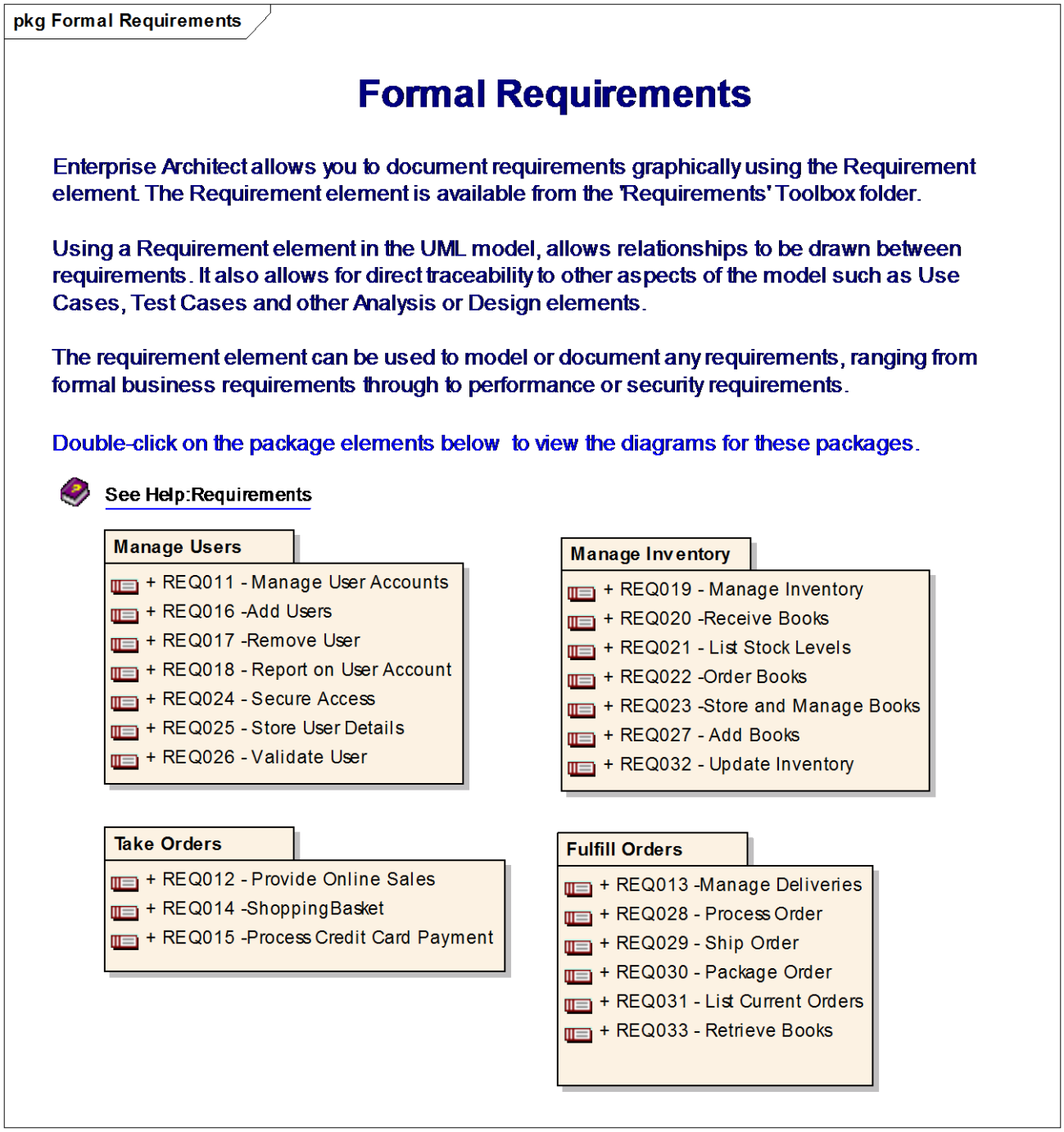
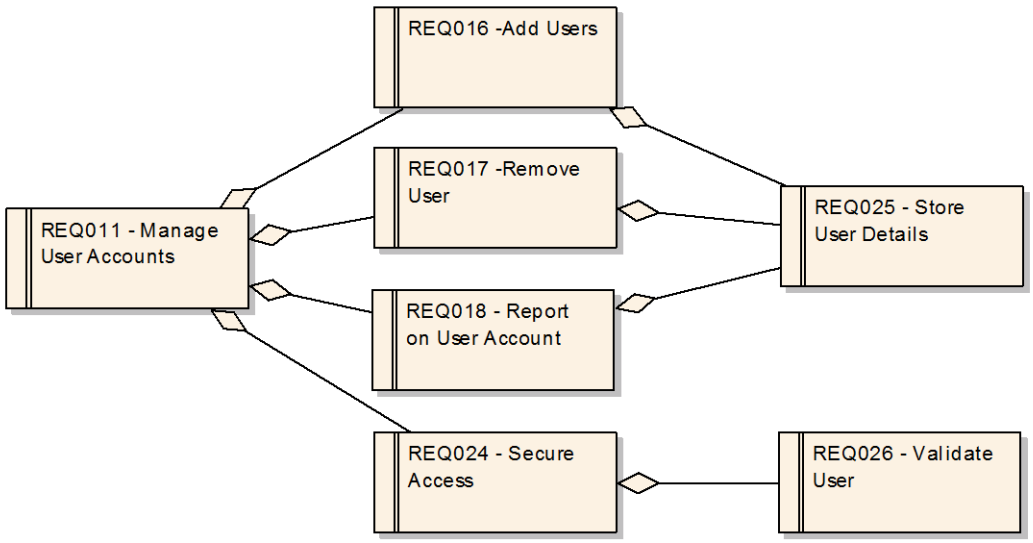


Diagram: Manage Users

custom Manage Users

Requirements are defined using the Custom Element of type Requirement. To view the detailed description of a Requirement, double-click on the element to view the properties.



Requirements can have relationships with other elements such as other Requirements, Use Cases etc. To view the Traceability of a requirement use the Hierarchy window. The Hierarchy window can be accessed from the main menu: View | Hierarchy or (Ctrl-Shift-4).

For more information on Requirements and the Hierarchy View see the following help topics:

Requirements Management

Hierarchy View

Diagram: Manage Inventory

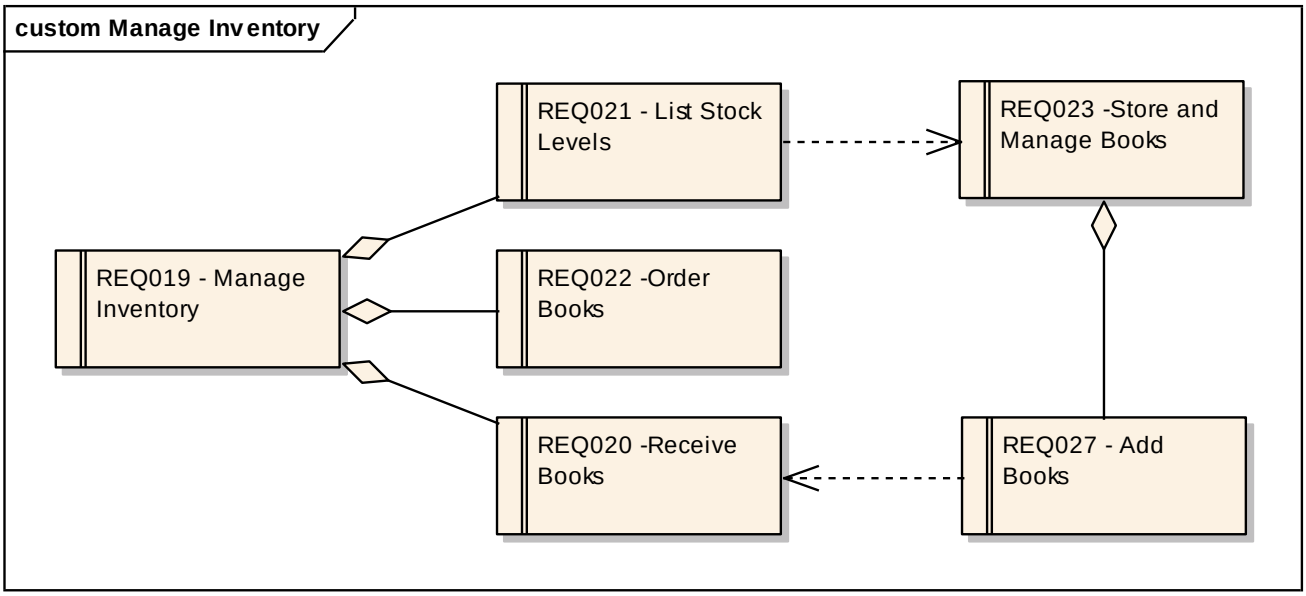


Diagram: Take Orders

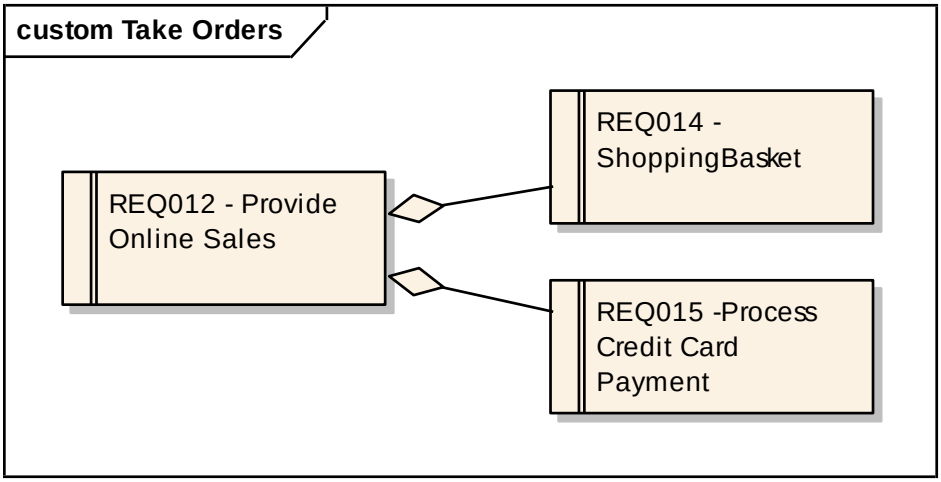


Diagram: Fulfill orders

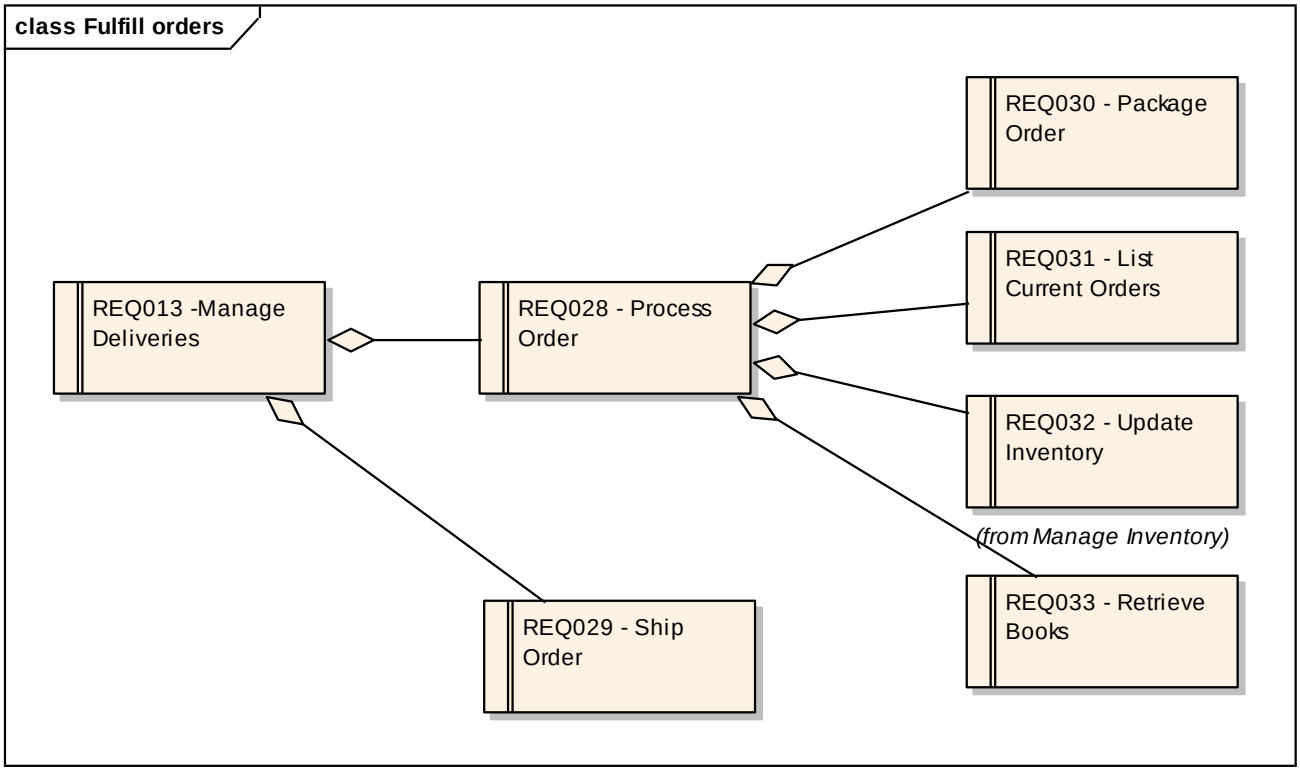


Diagram: Use Case Model

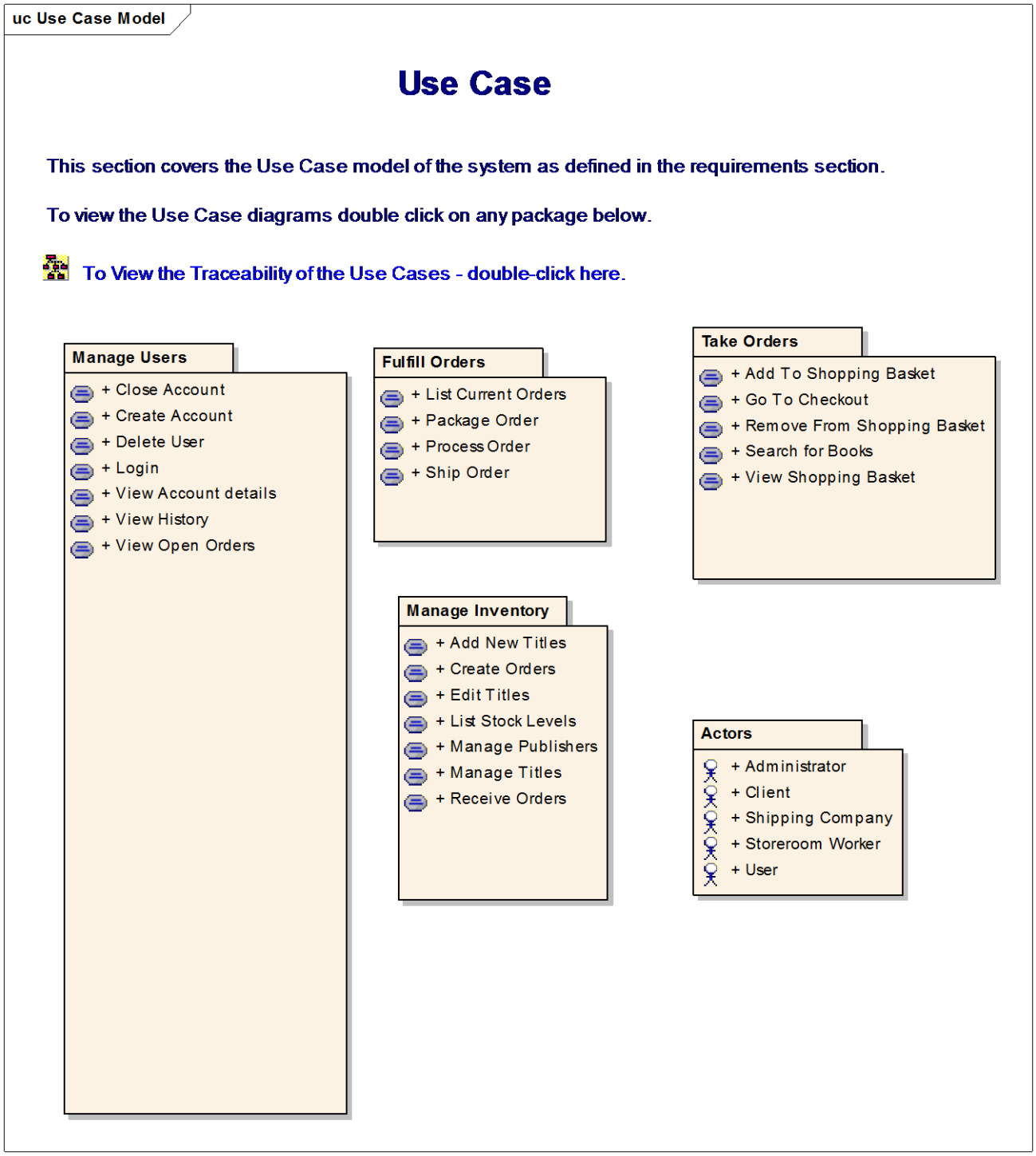


Diagram: Actors

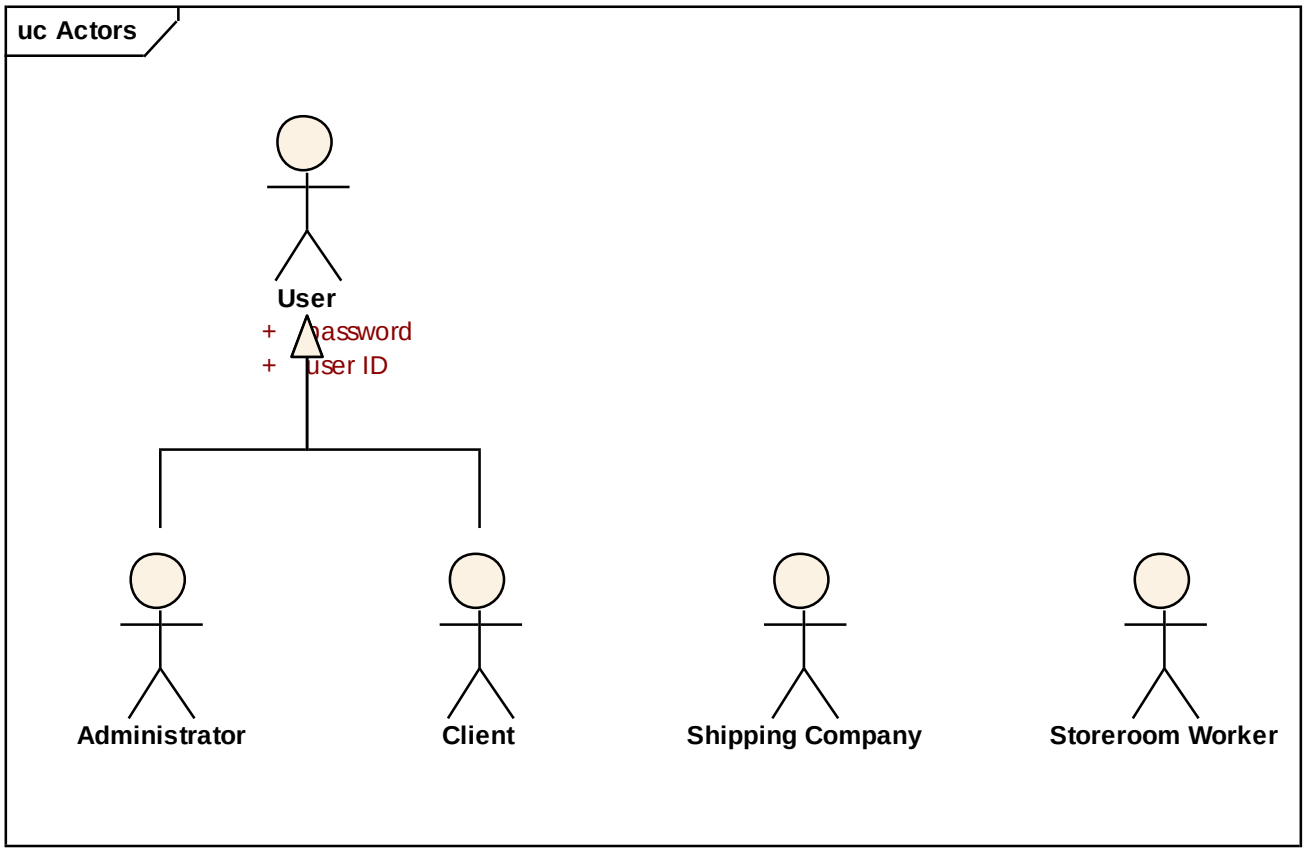


Diagram: Actors - Images

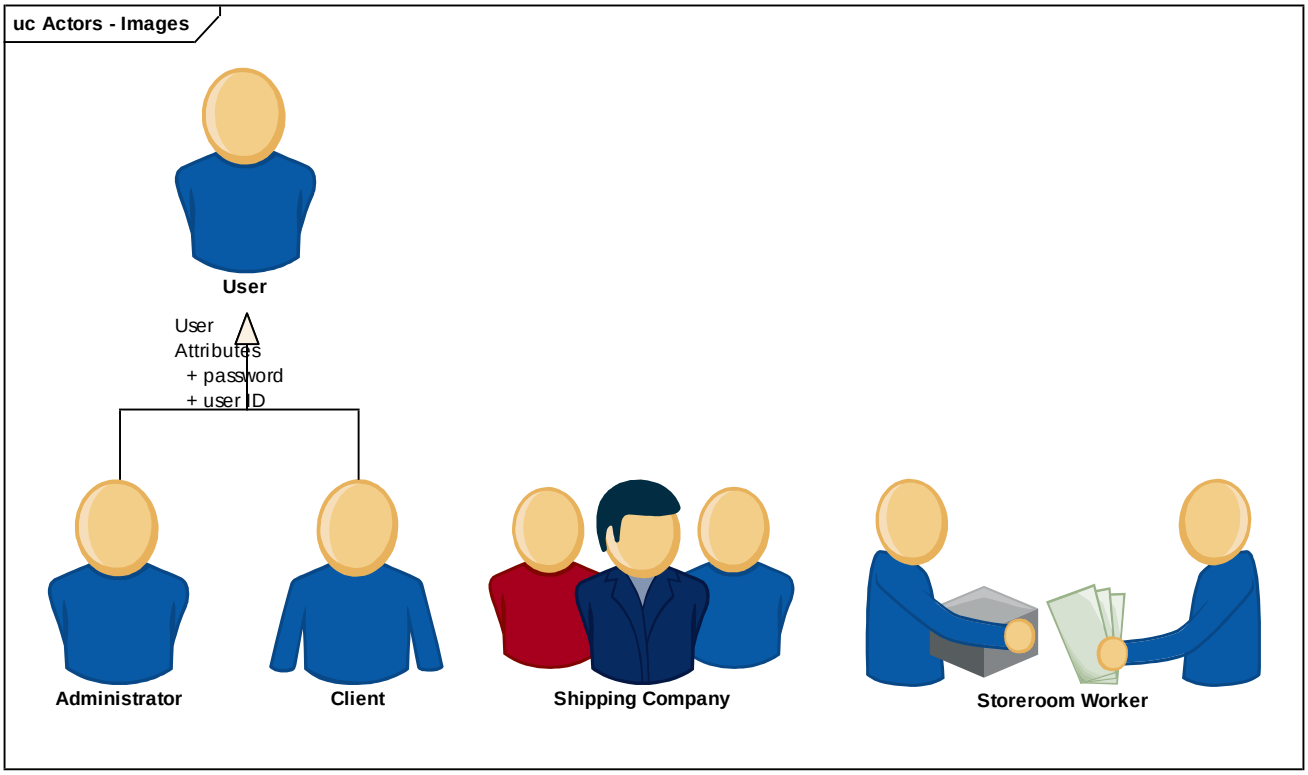


Diagram: Close Account

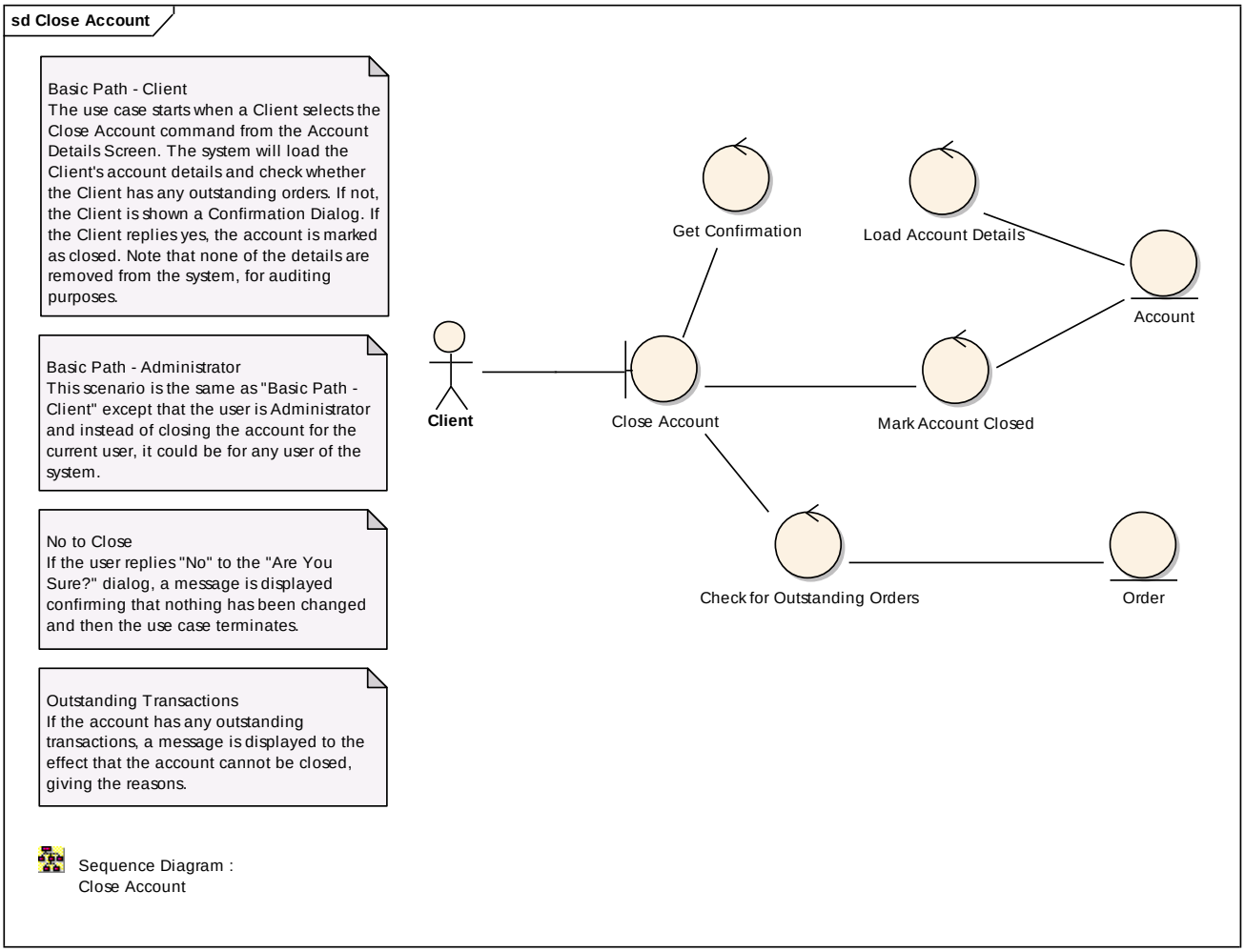


Diagram: Close Account

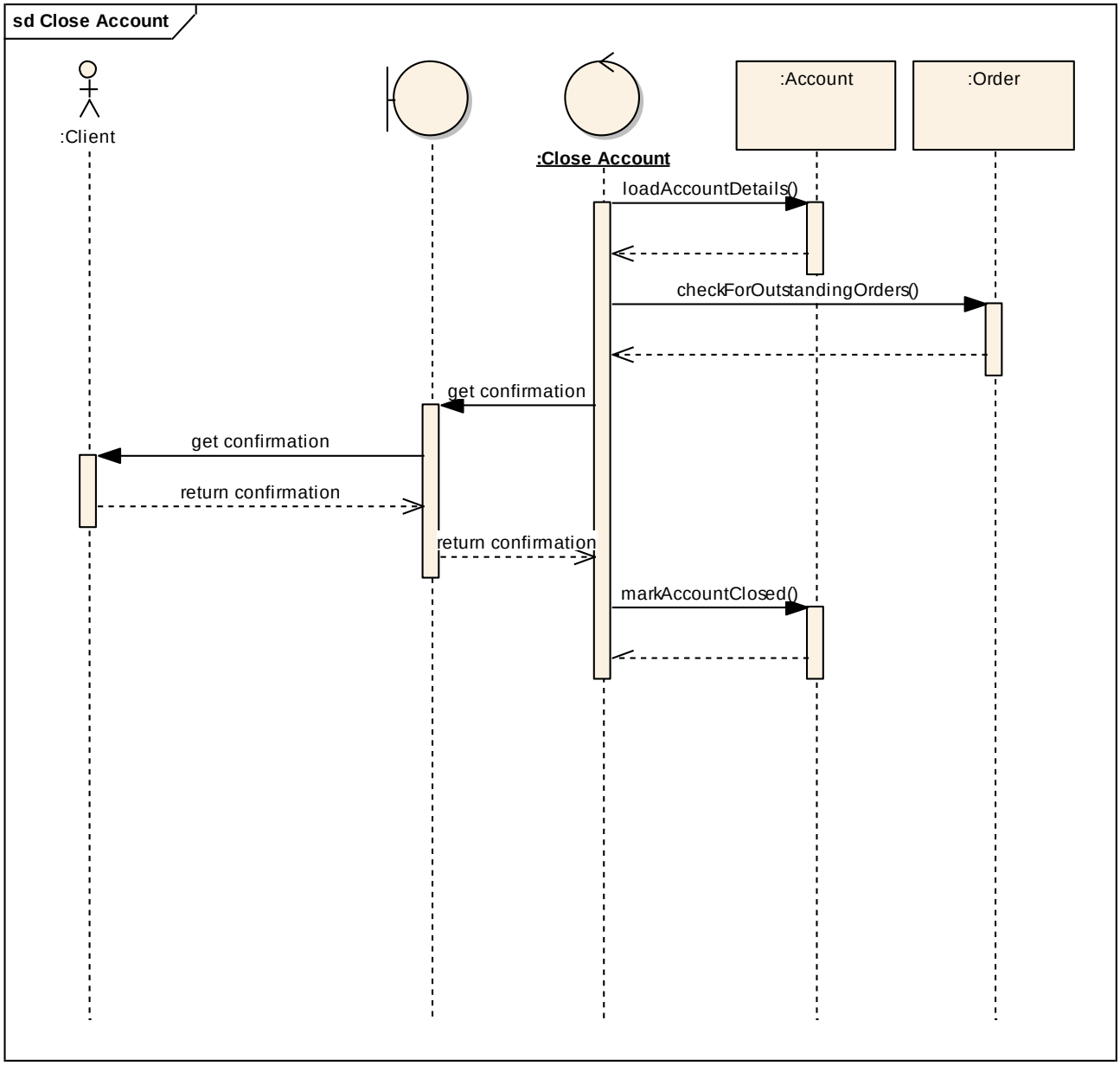


Diagram: Create Account

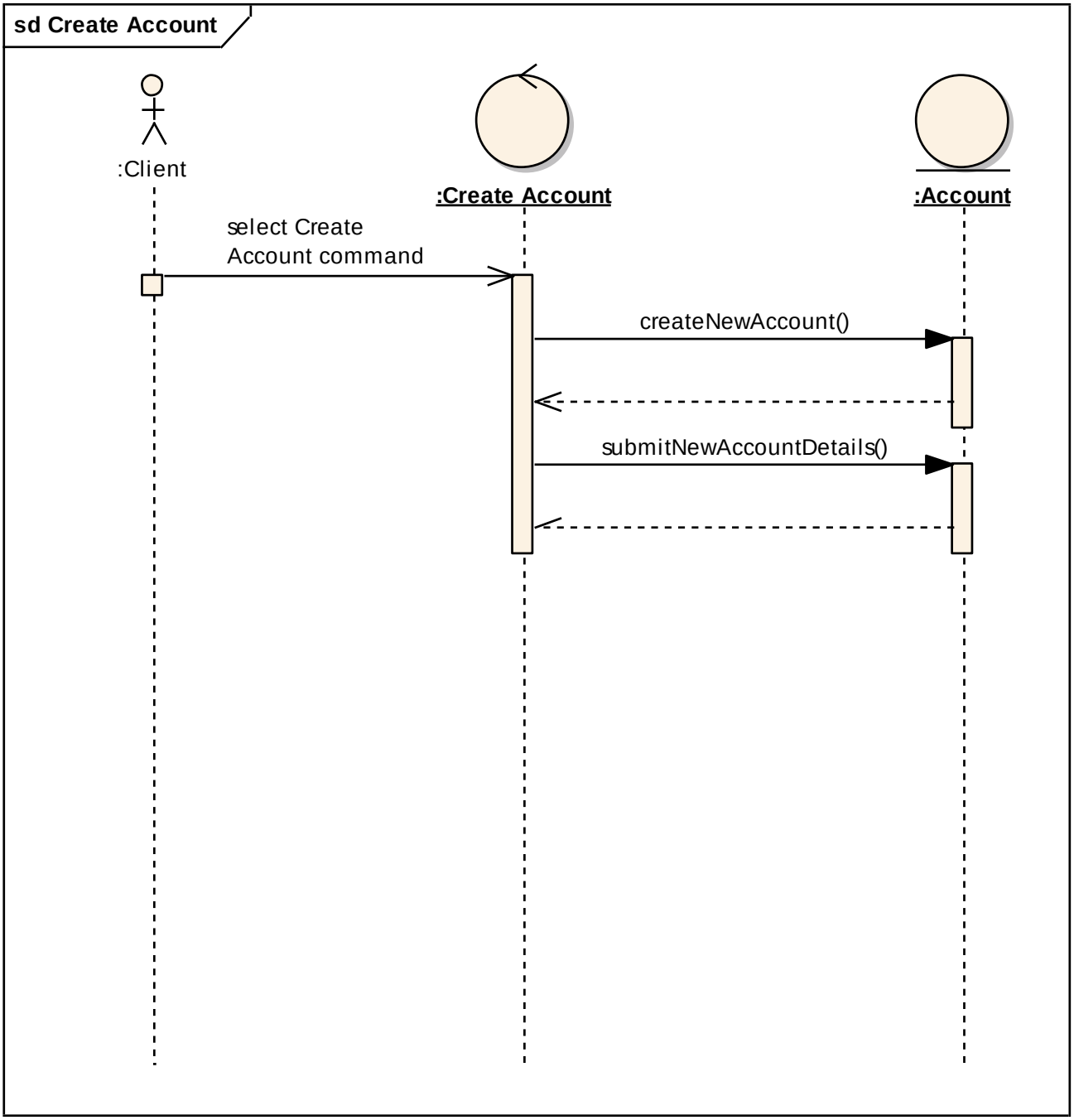


Diagram: Create Account

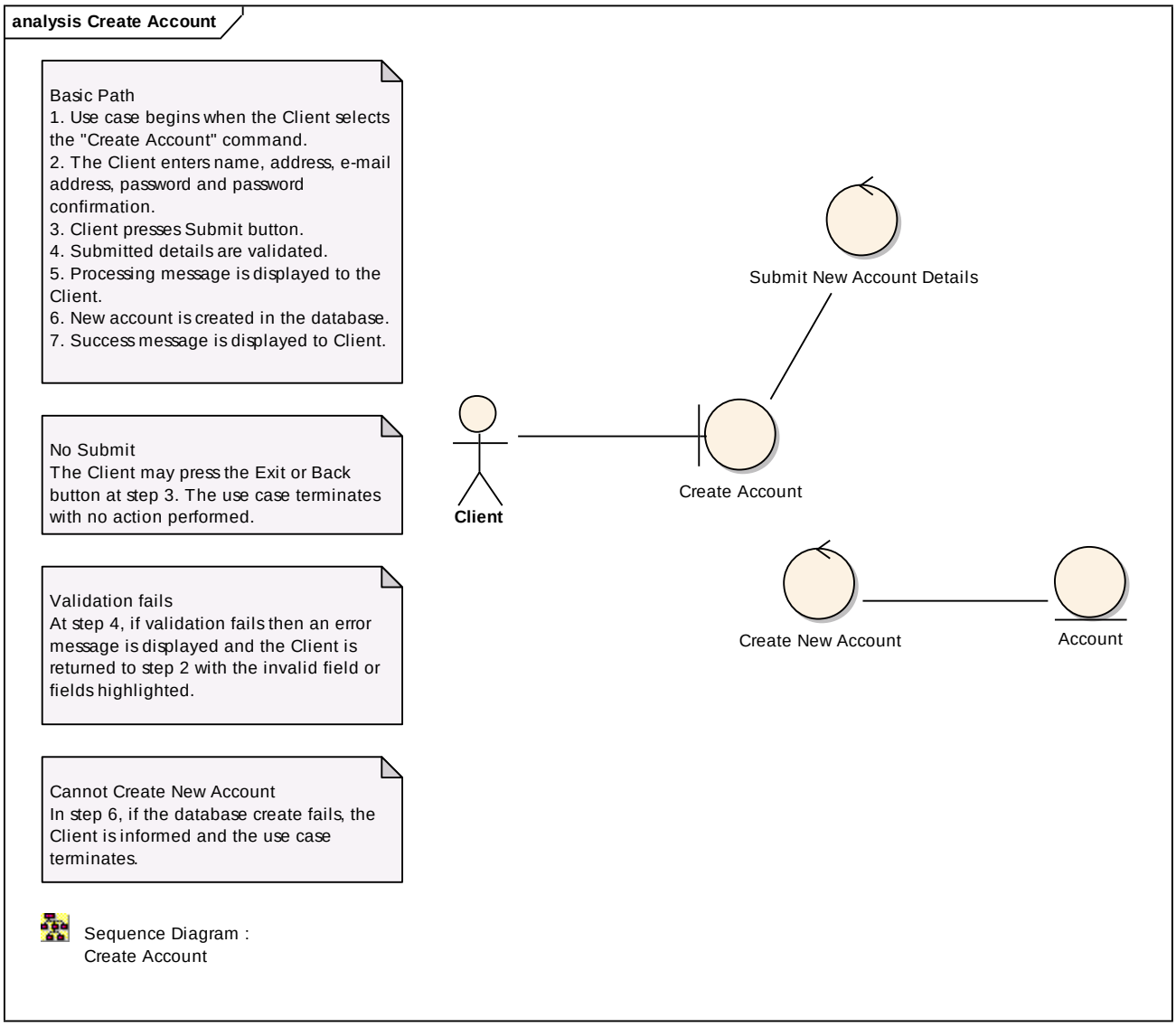


Diagram: Delete User

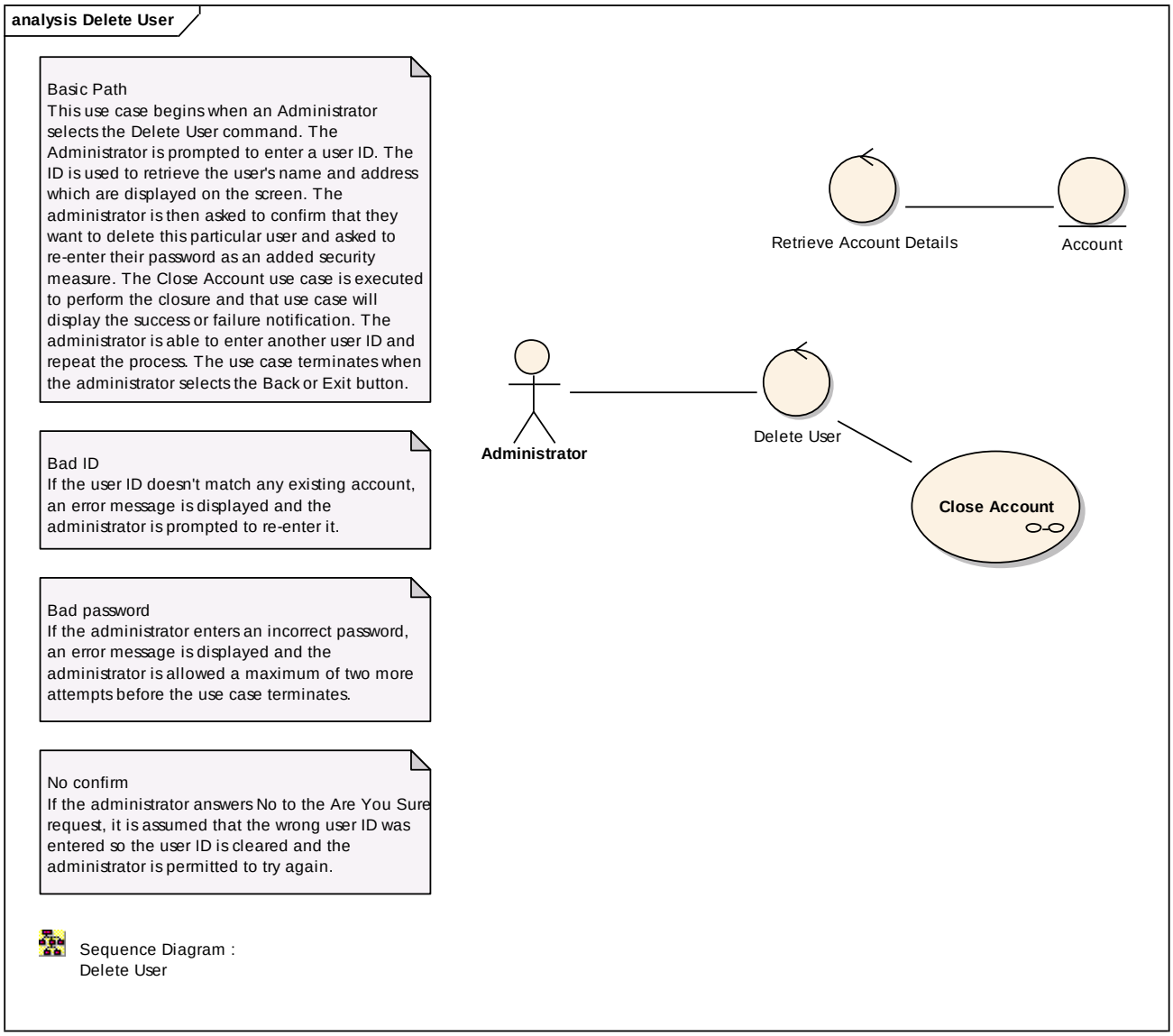


Diagram: Delete User

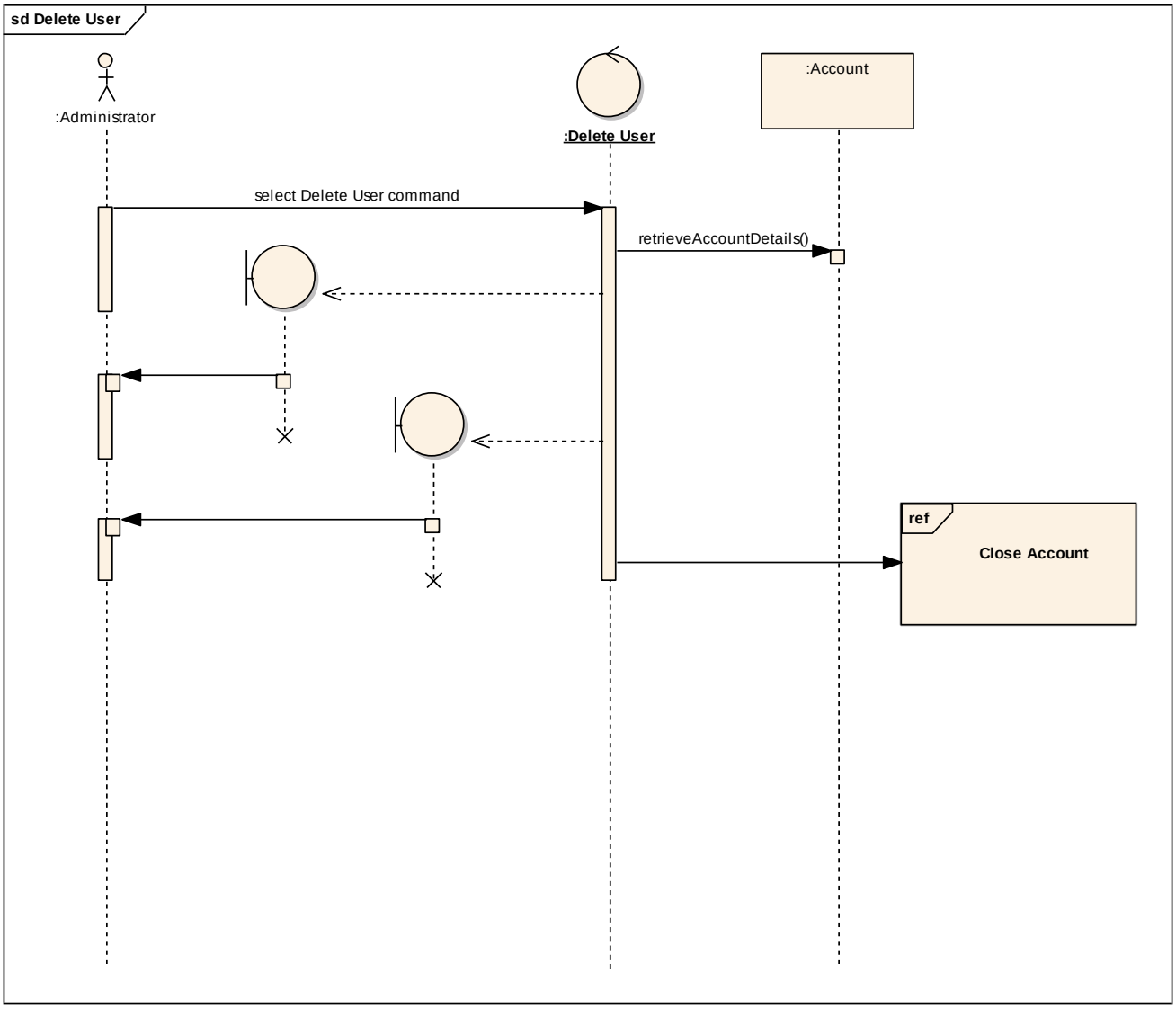


Diagram: Login

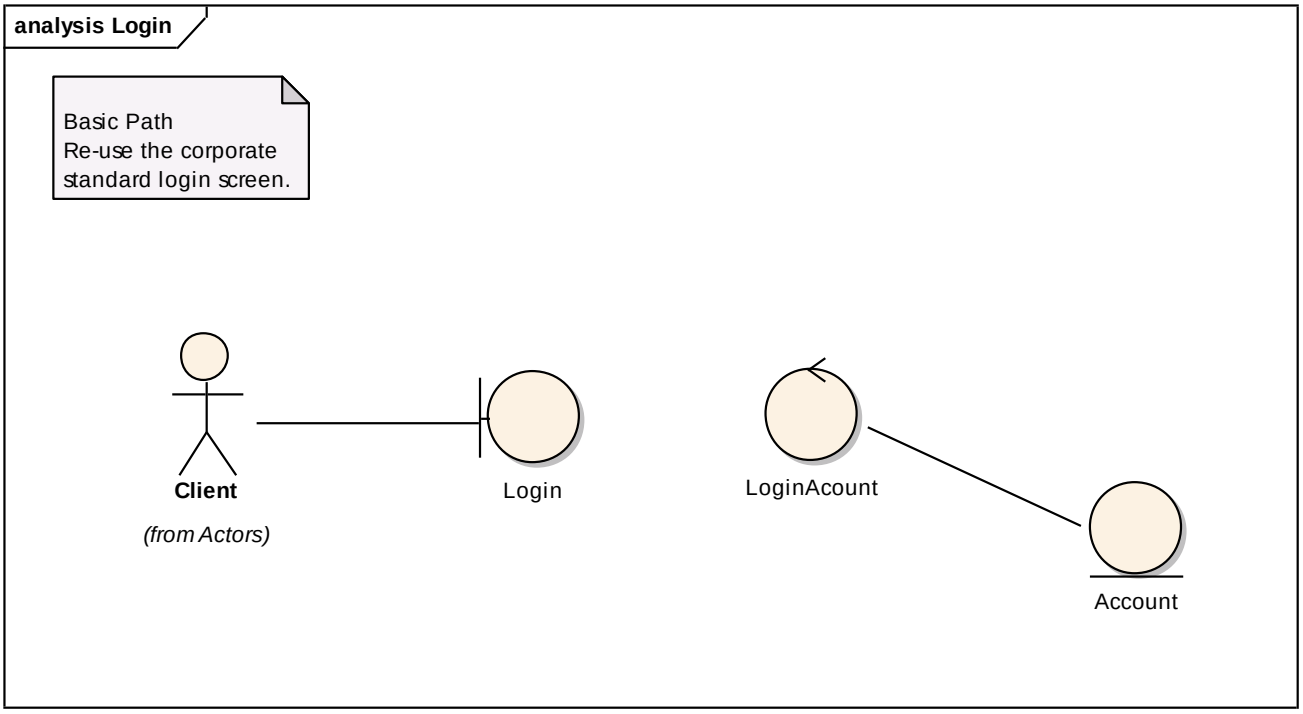


Diagram: Login

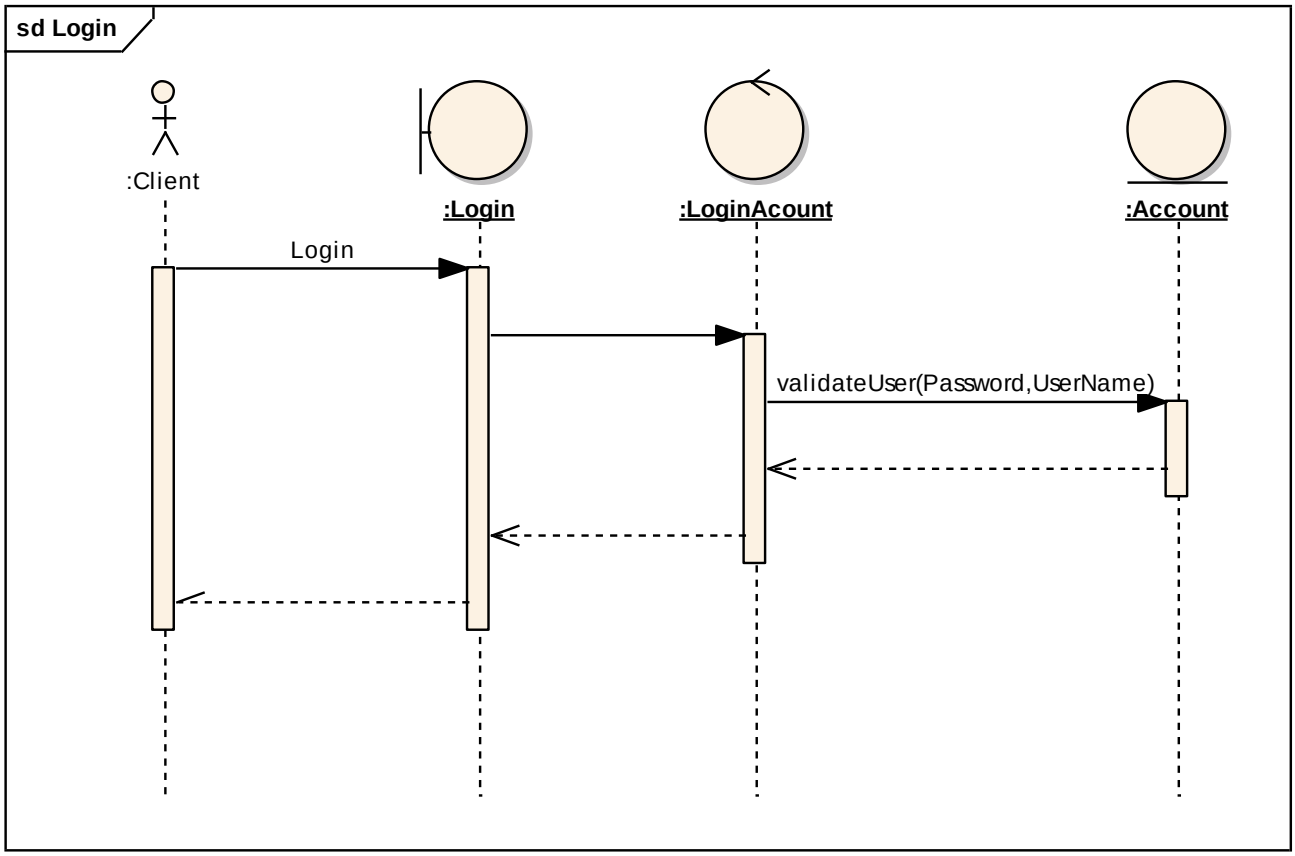


Diagram: Manage Users

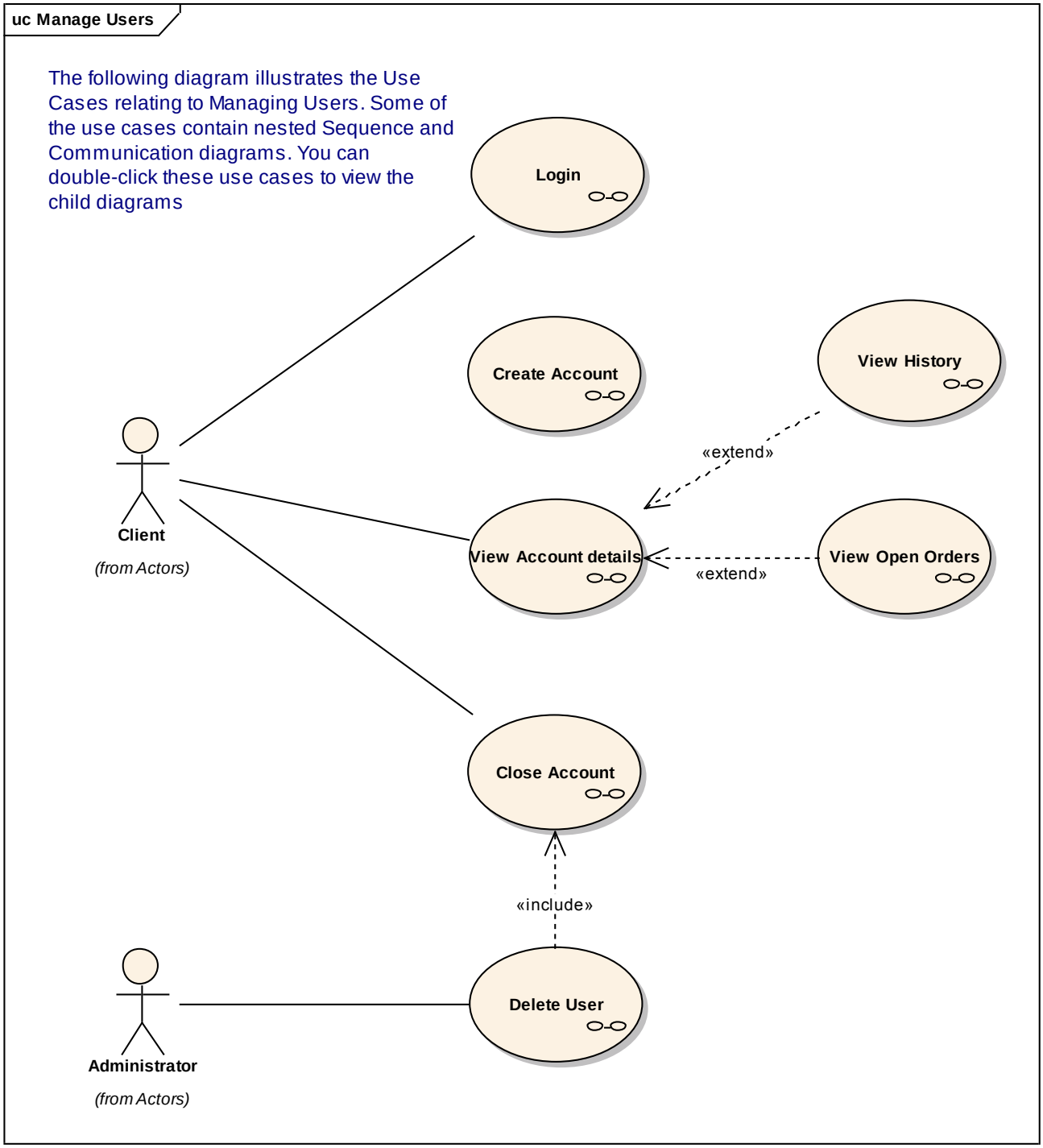


Diagram: View Account details

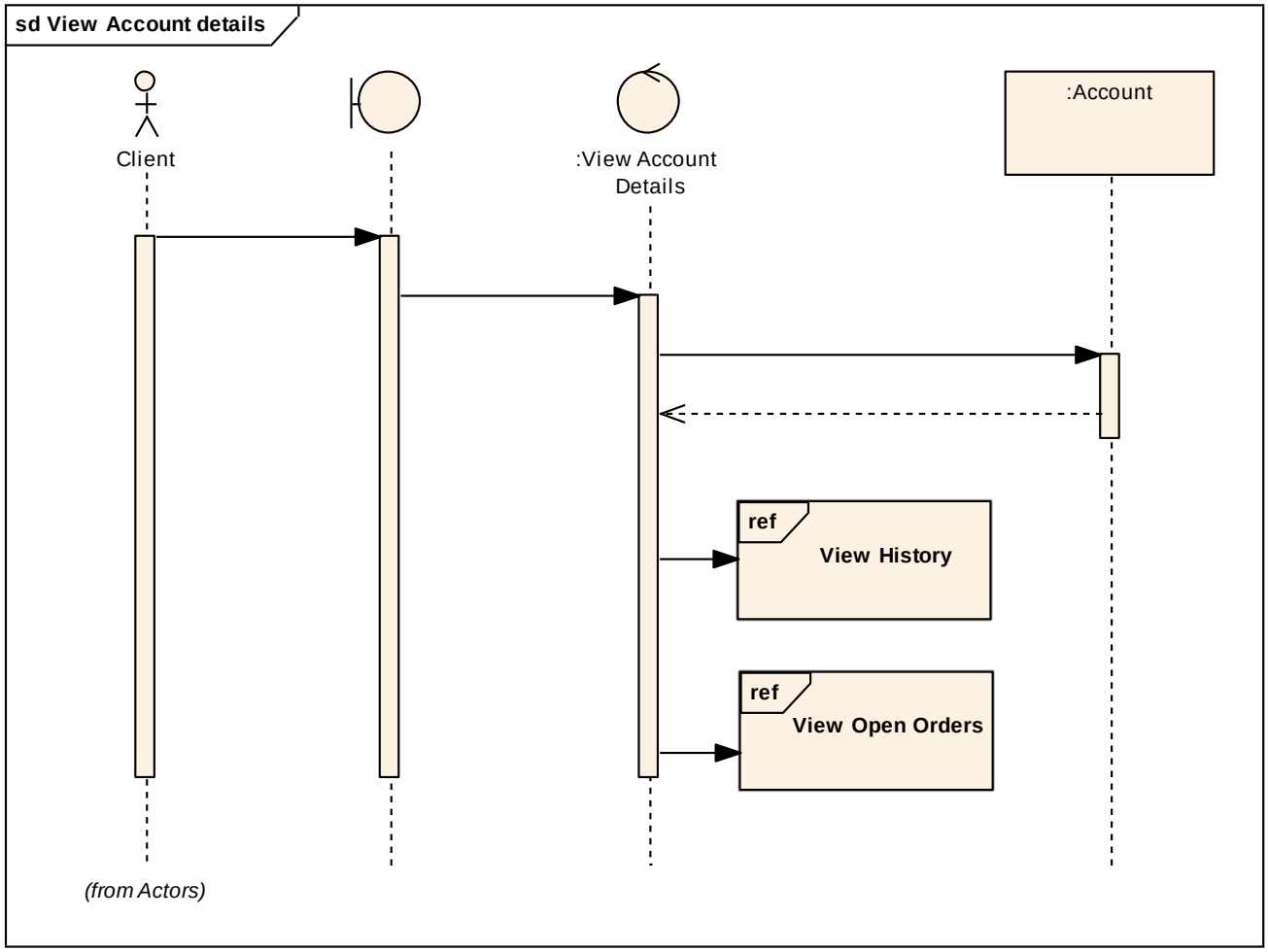


Diagram: View Account details

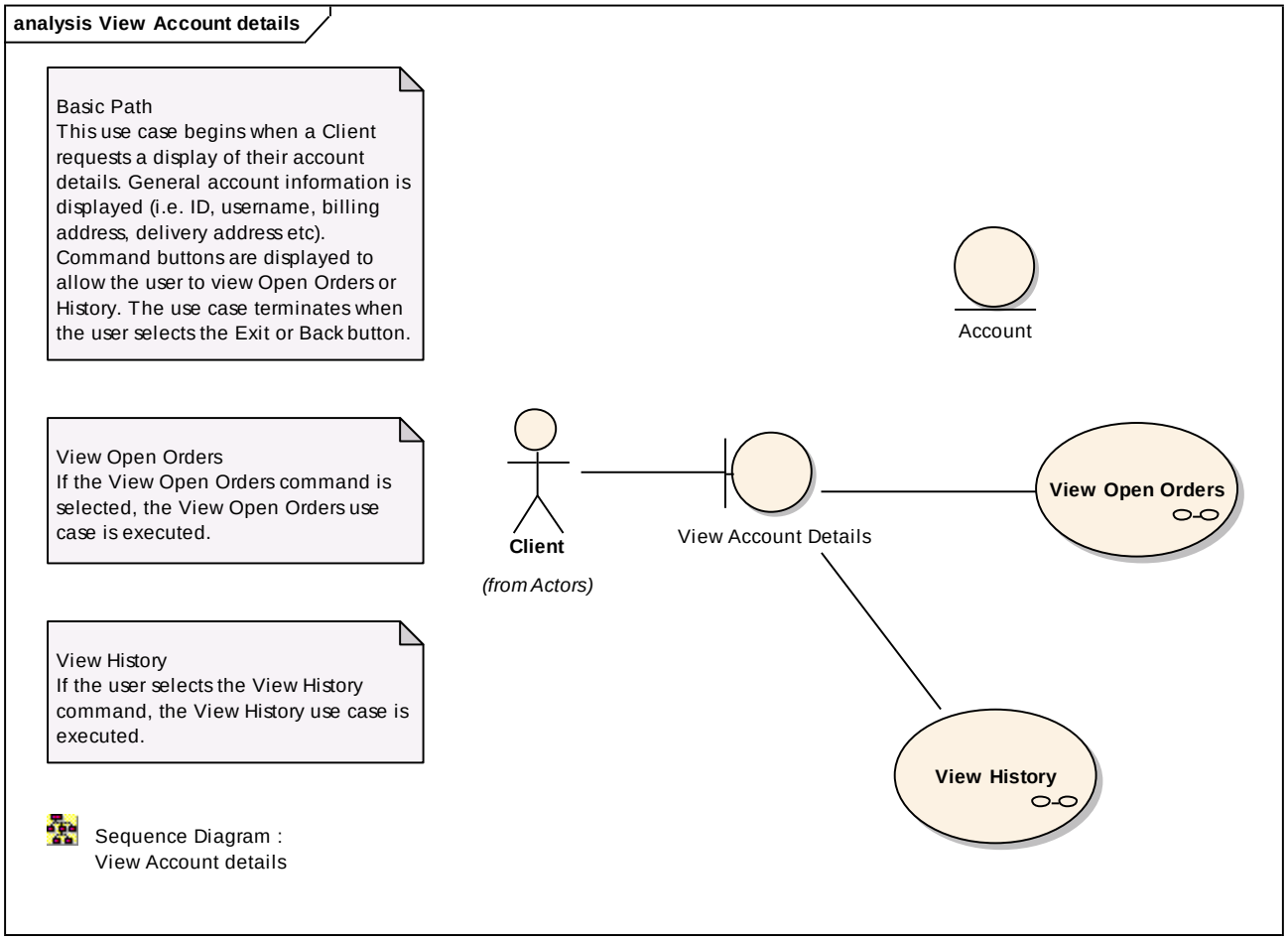


Diagram: View History

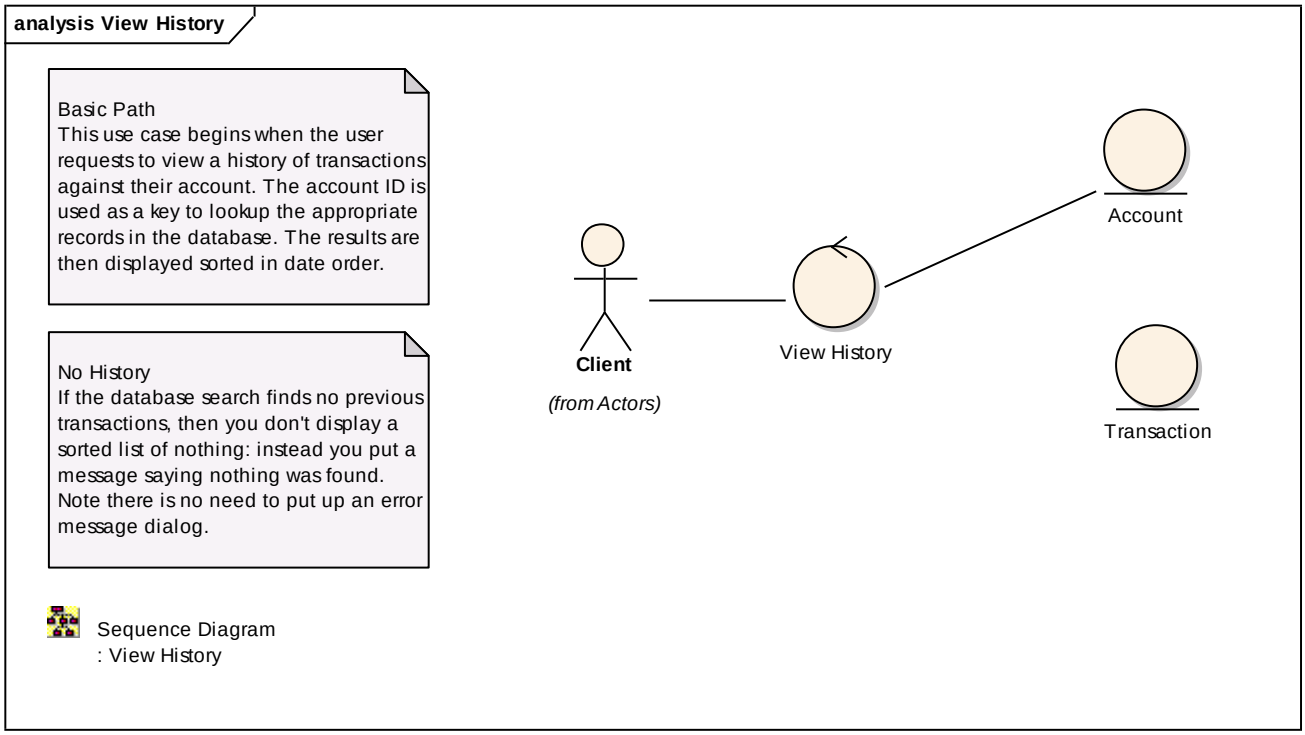


Diagram: View History

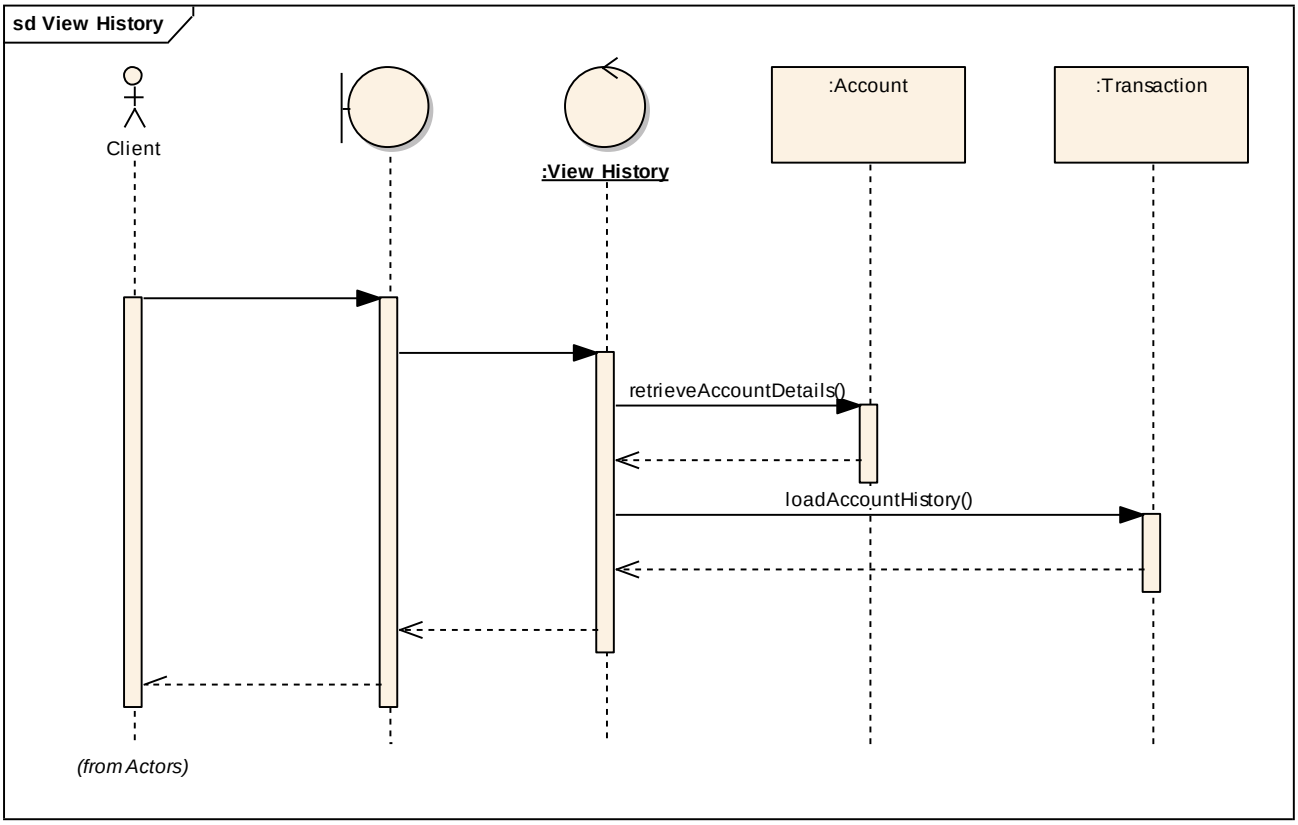


Diagram: View Open Orders

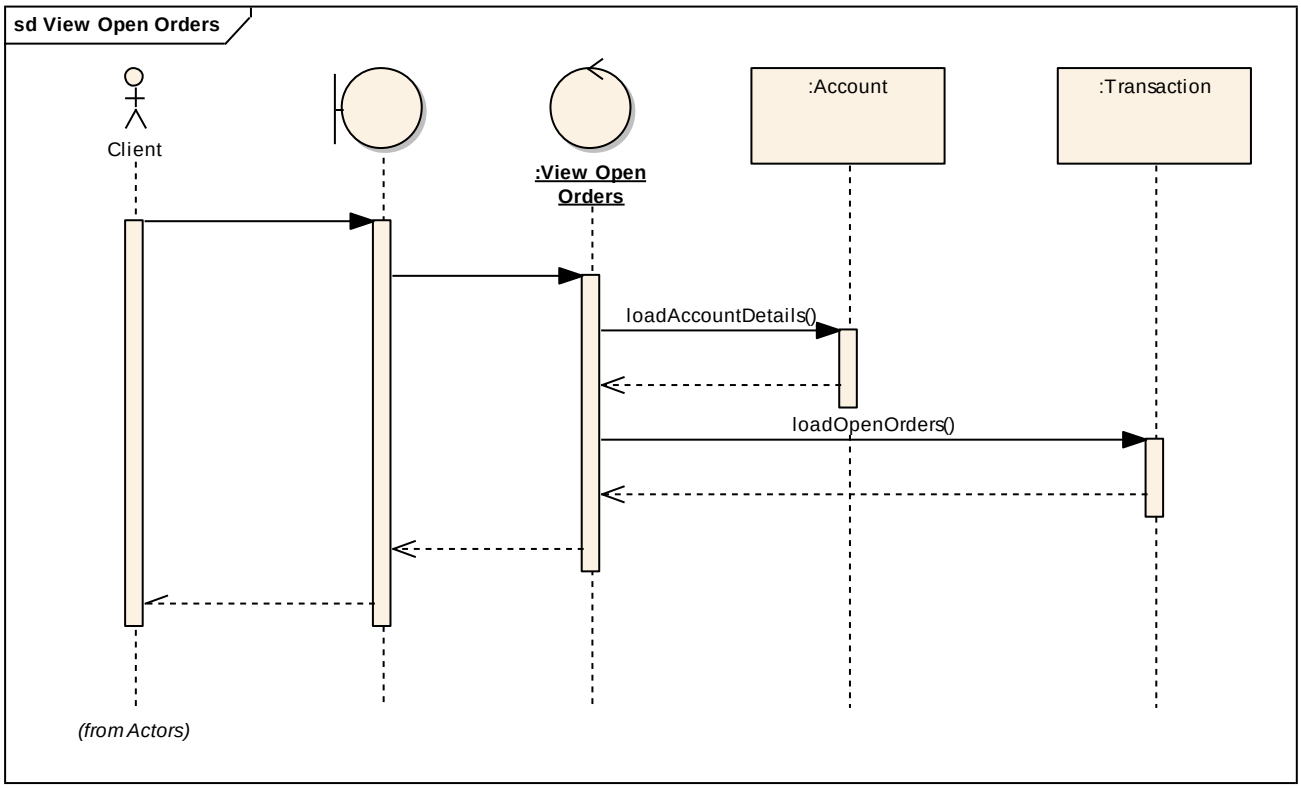


Diagram: View Open Orders

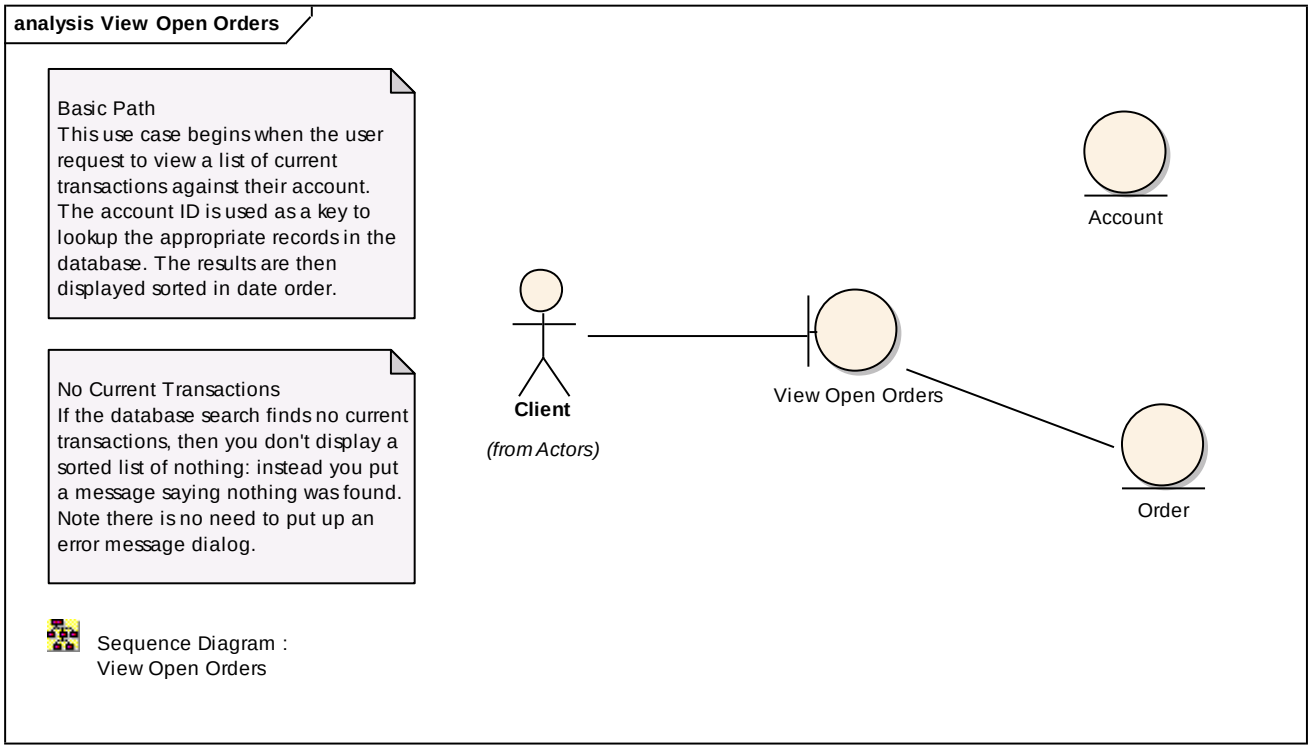


Diagram: Create Orders

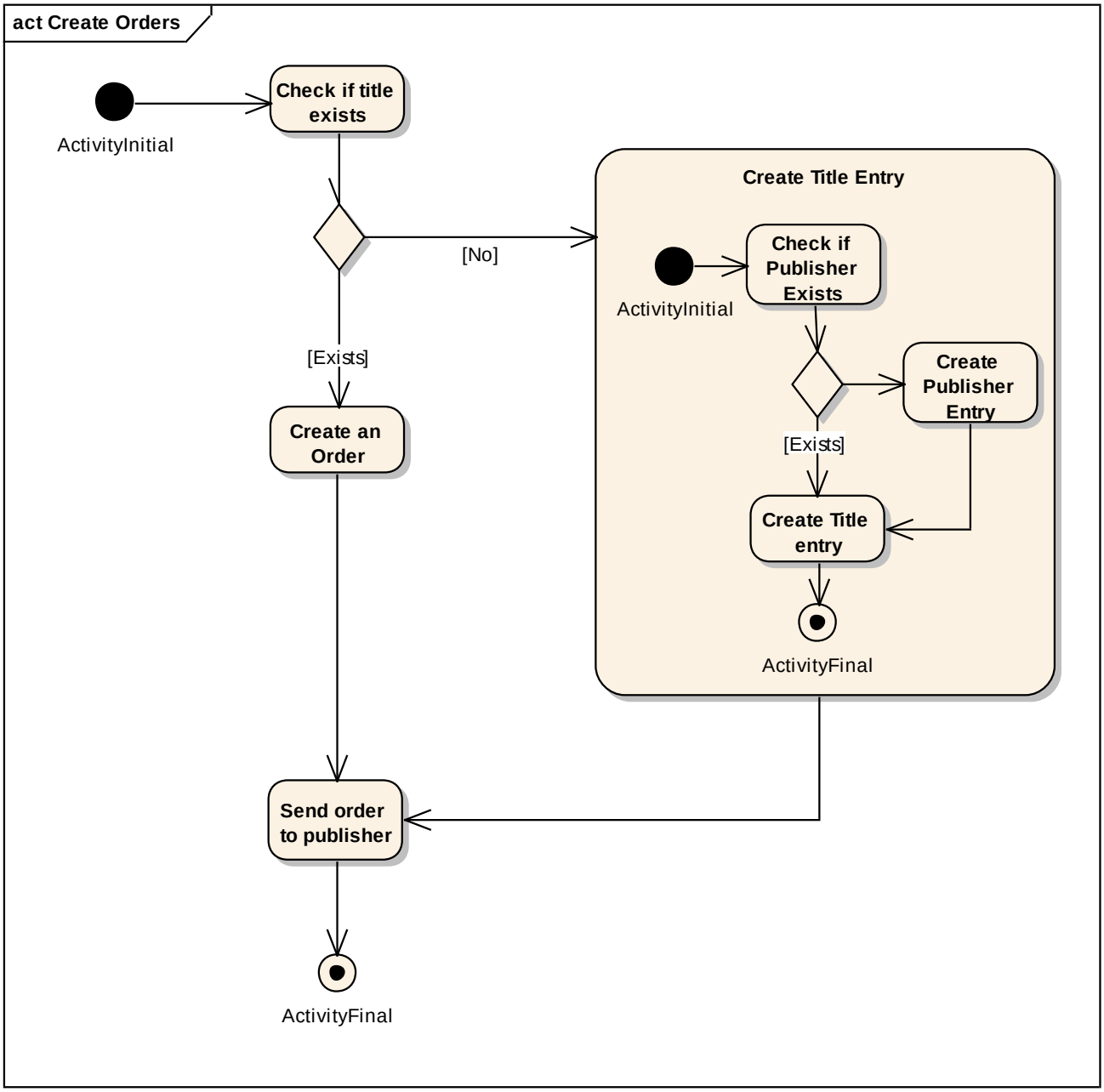


Diagram: Manage Inventory

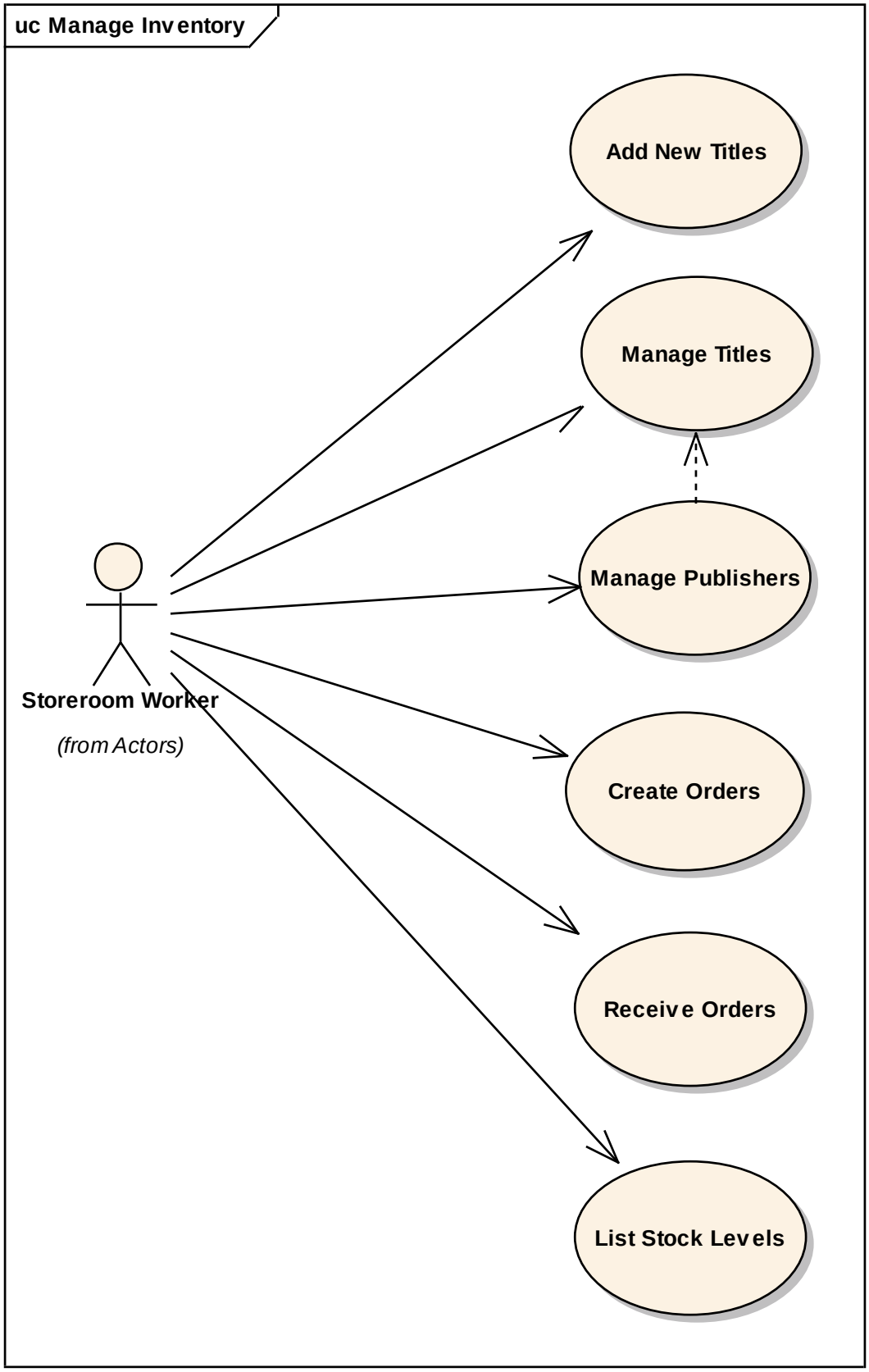

uc Manage Inventory
Add New Titles
Manage Titles
Manage Publishers
Create Orders
Receive Orders
List Stock Levels
Storeroom Worker
(from Actors)

Diagram: Manage Titles State

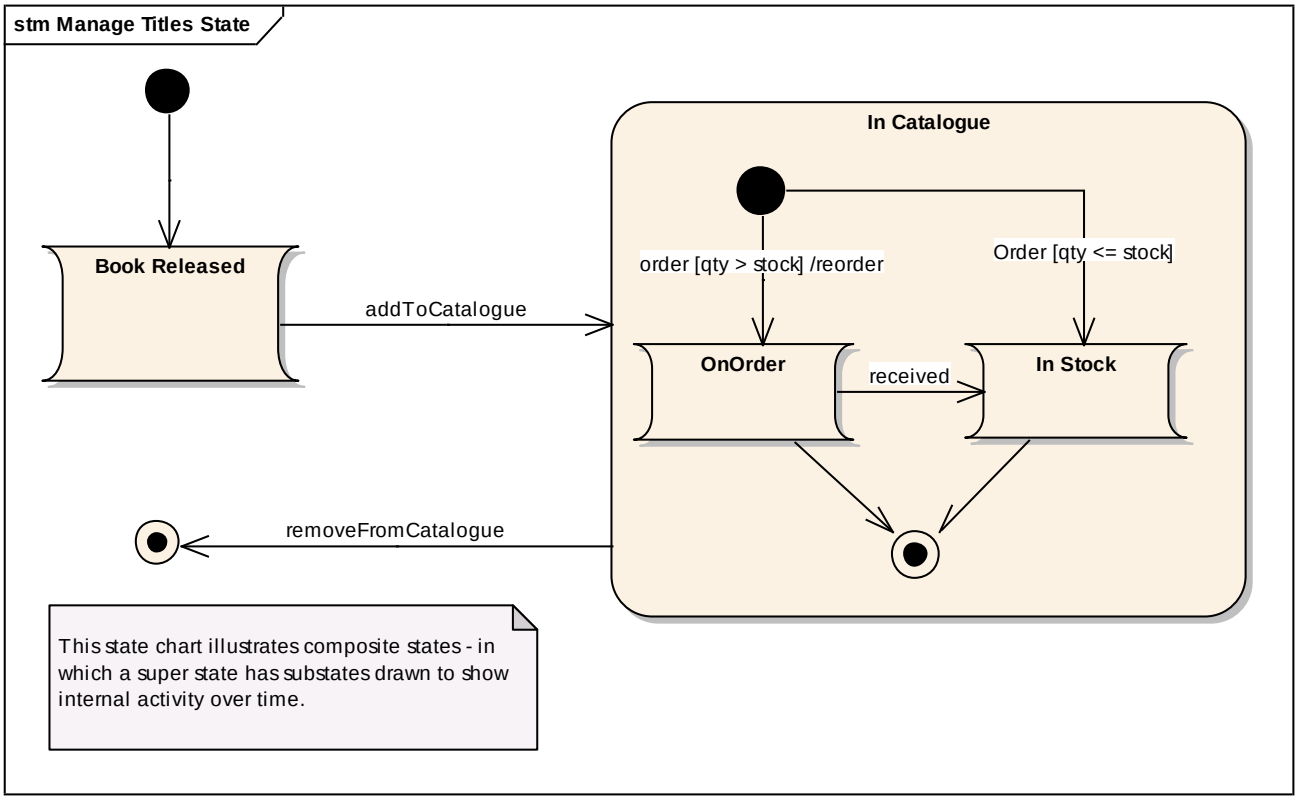


Diagram: Add To Shopping Cart

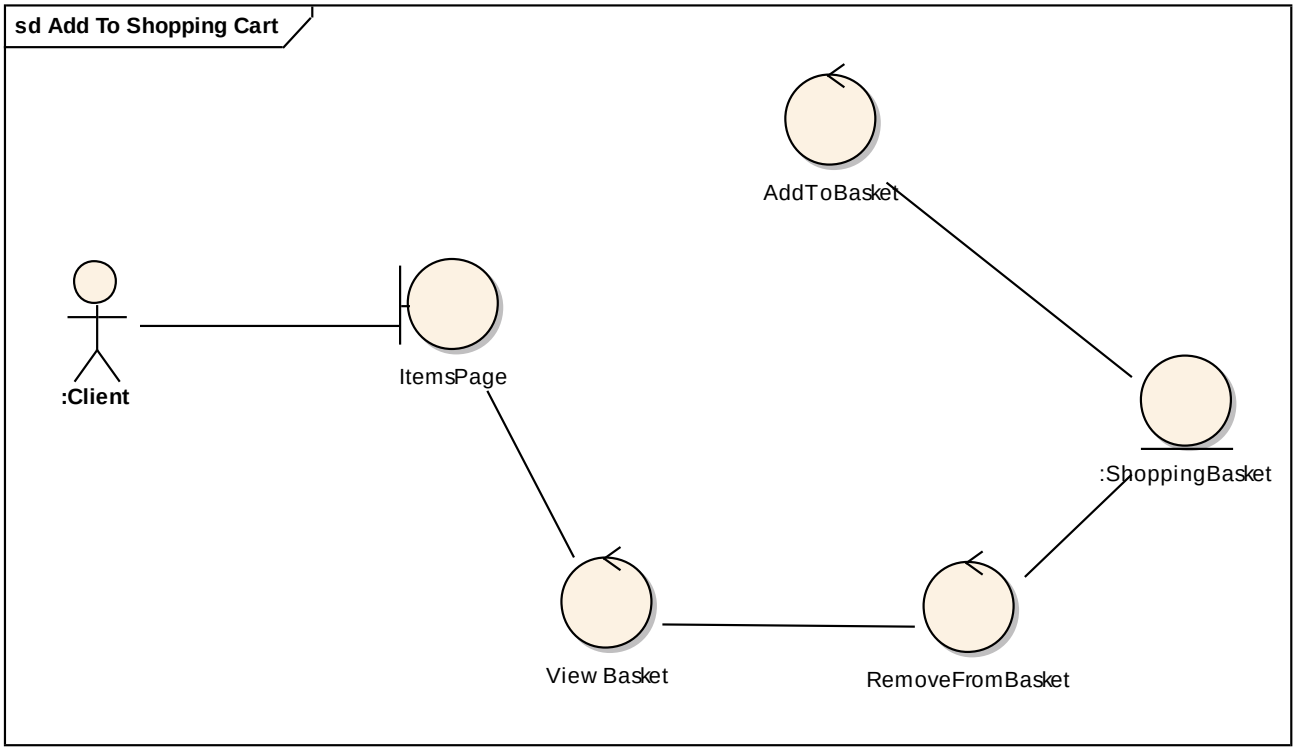


Diagram: Add To Shopping Cart

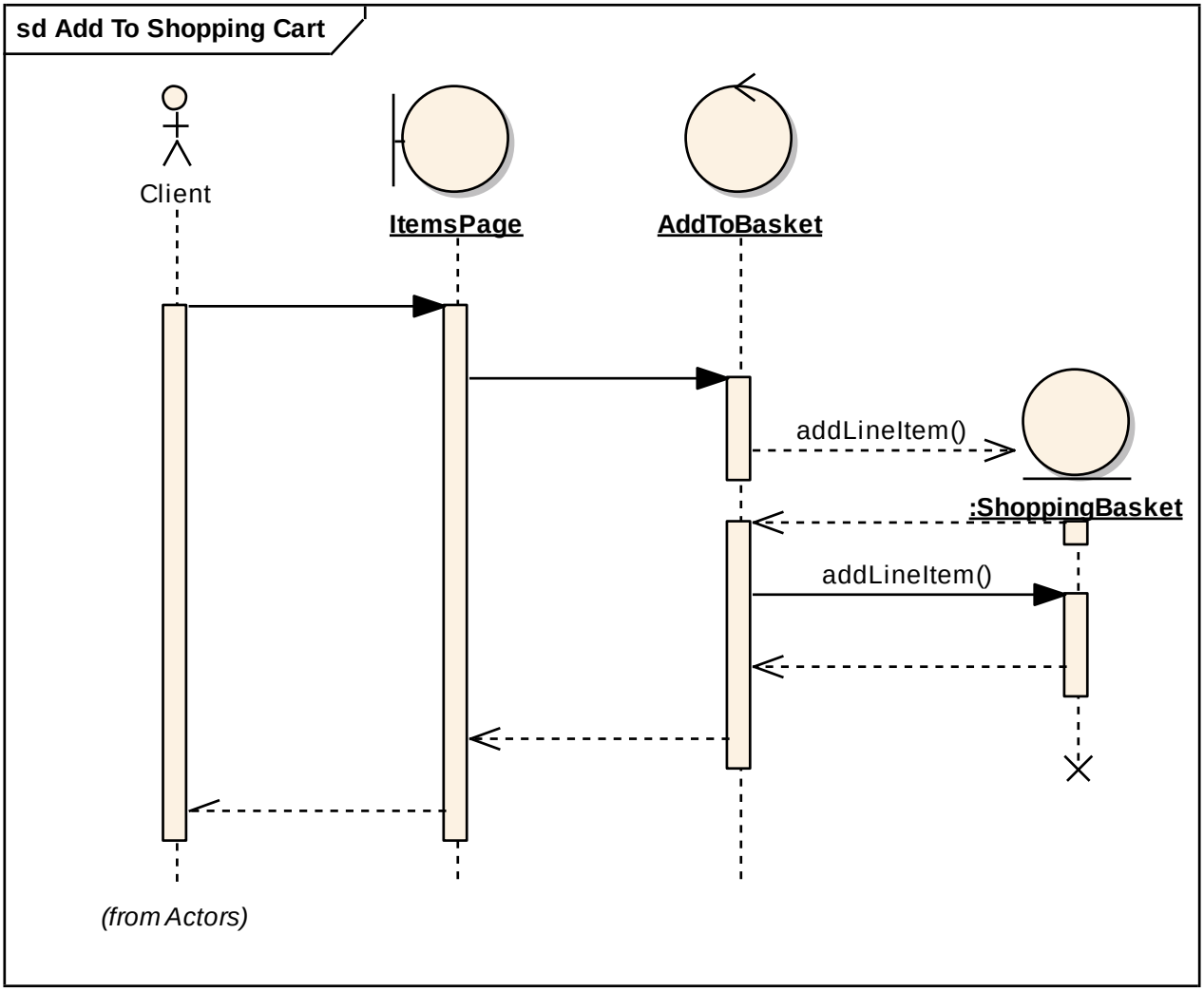


Diagram: Sell Books

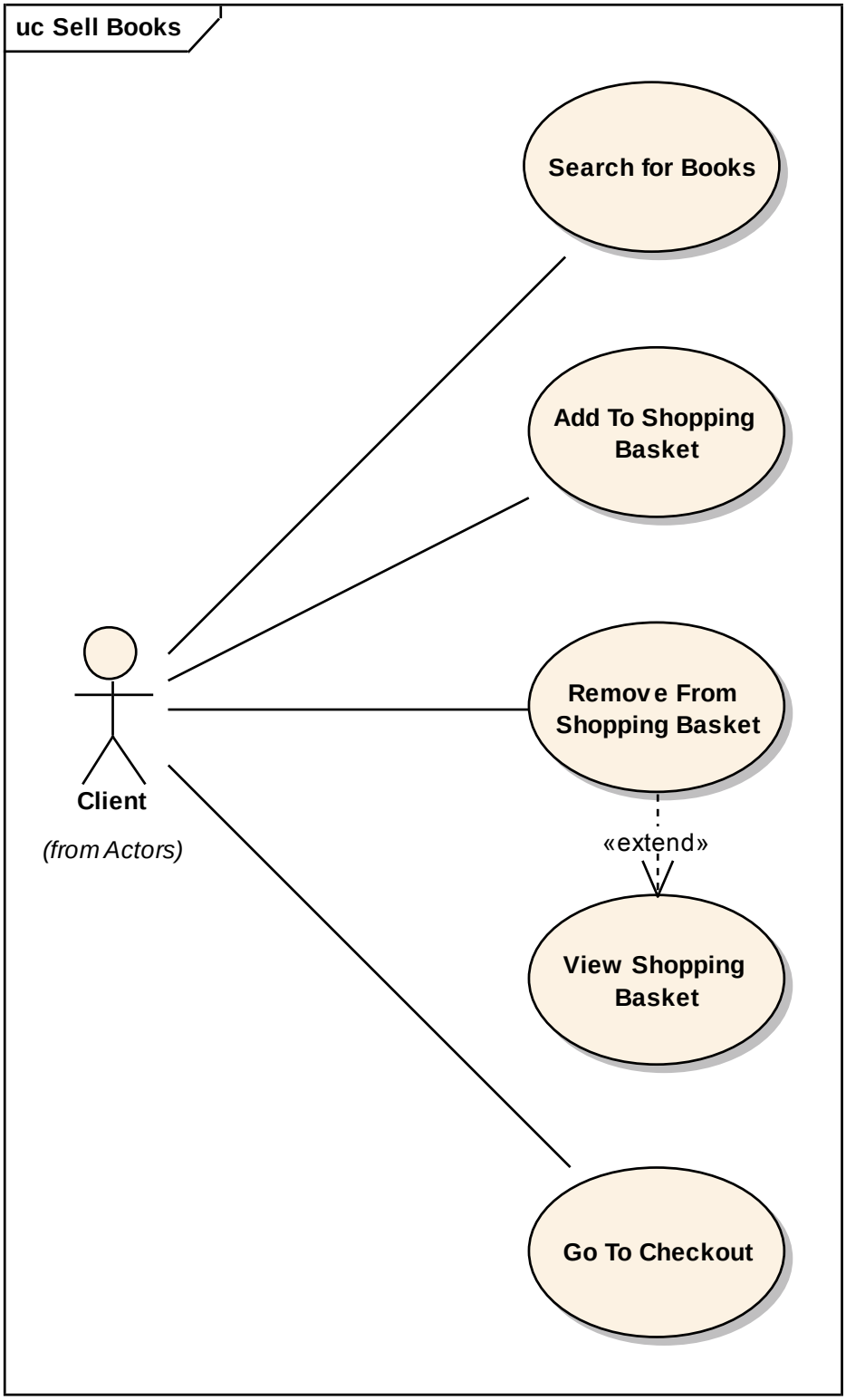


Diagram: Credit Card Problem

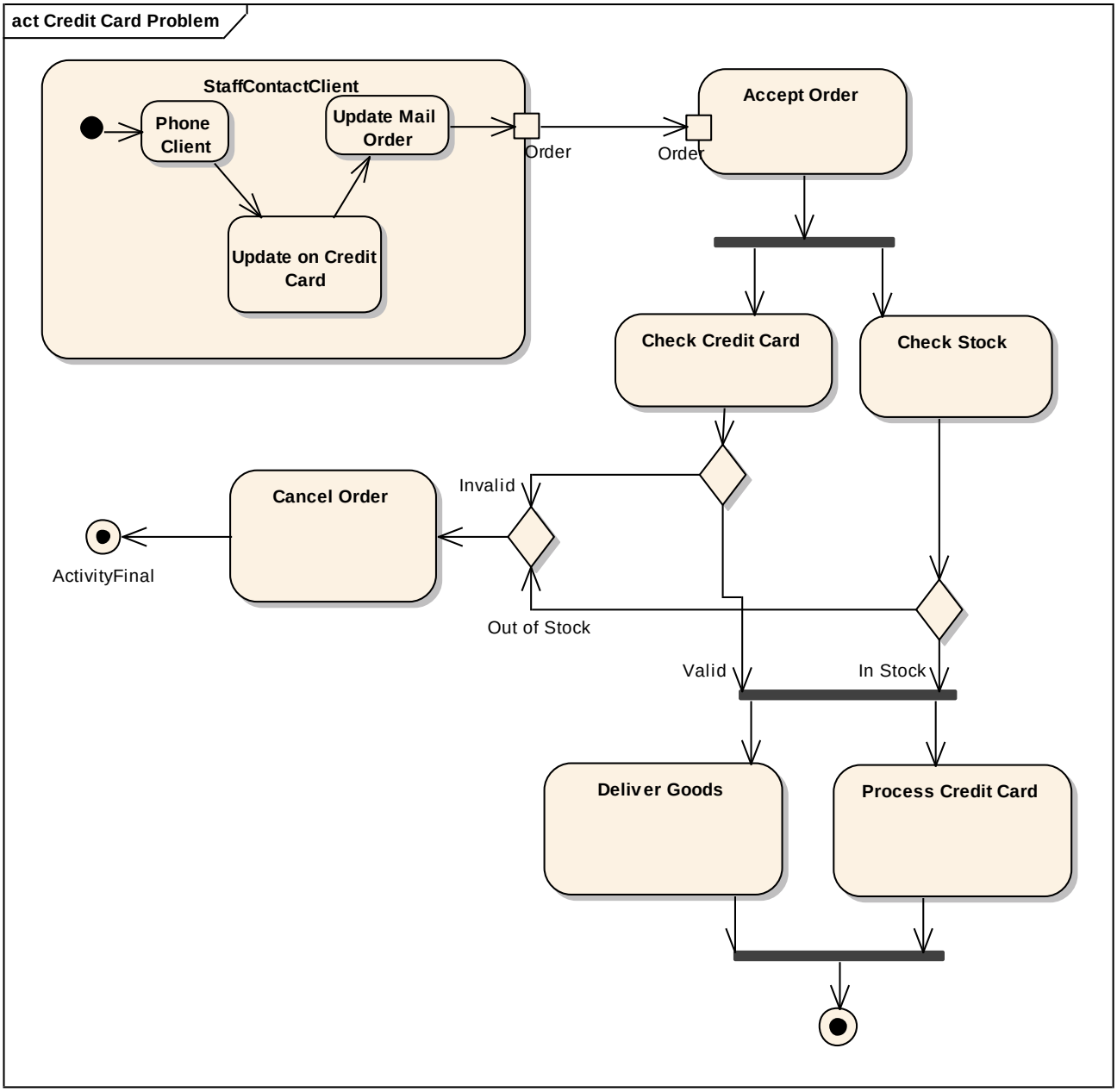


Diagram: Deliver Books

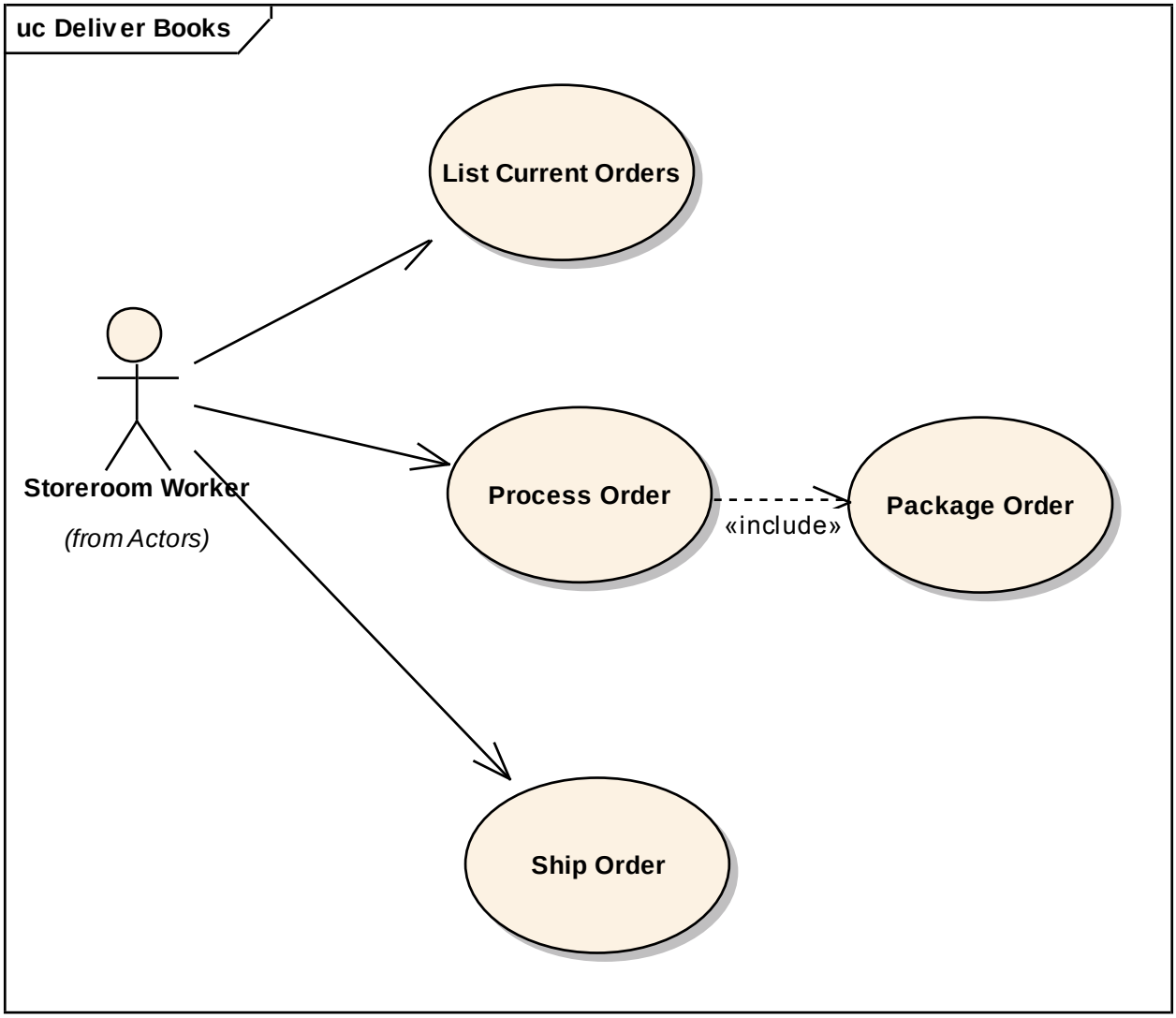


Diagram: Invoice Payment

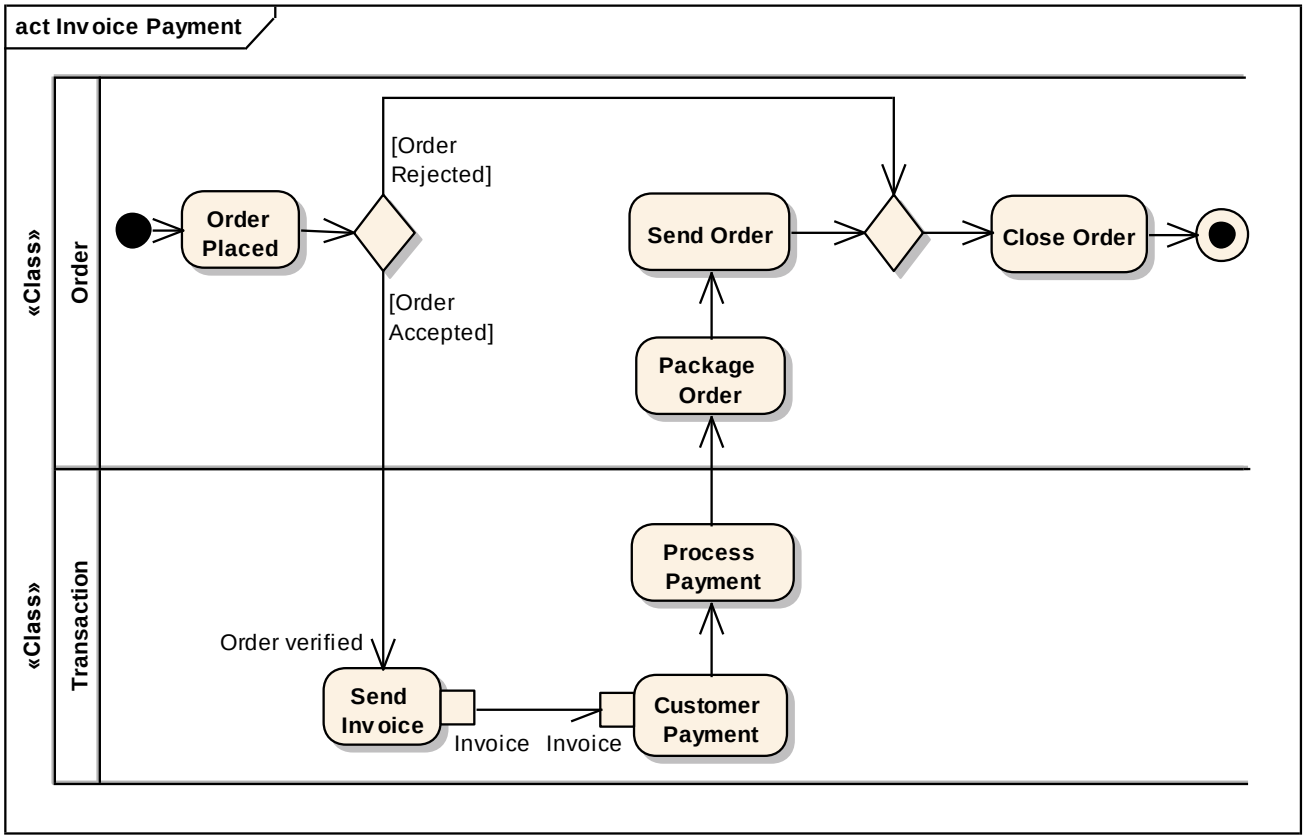


Diagram: Customer Process

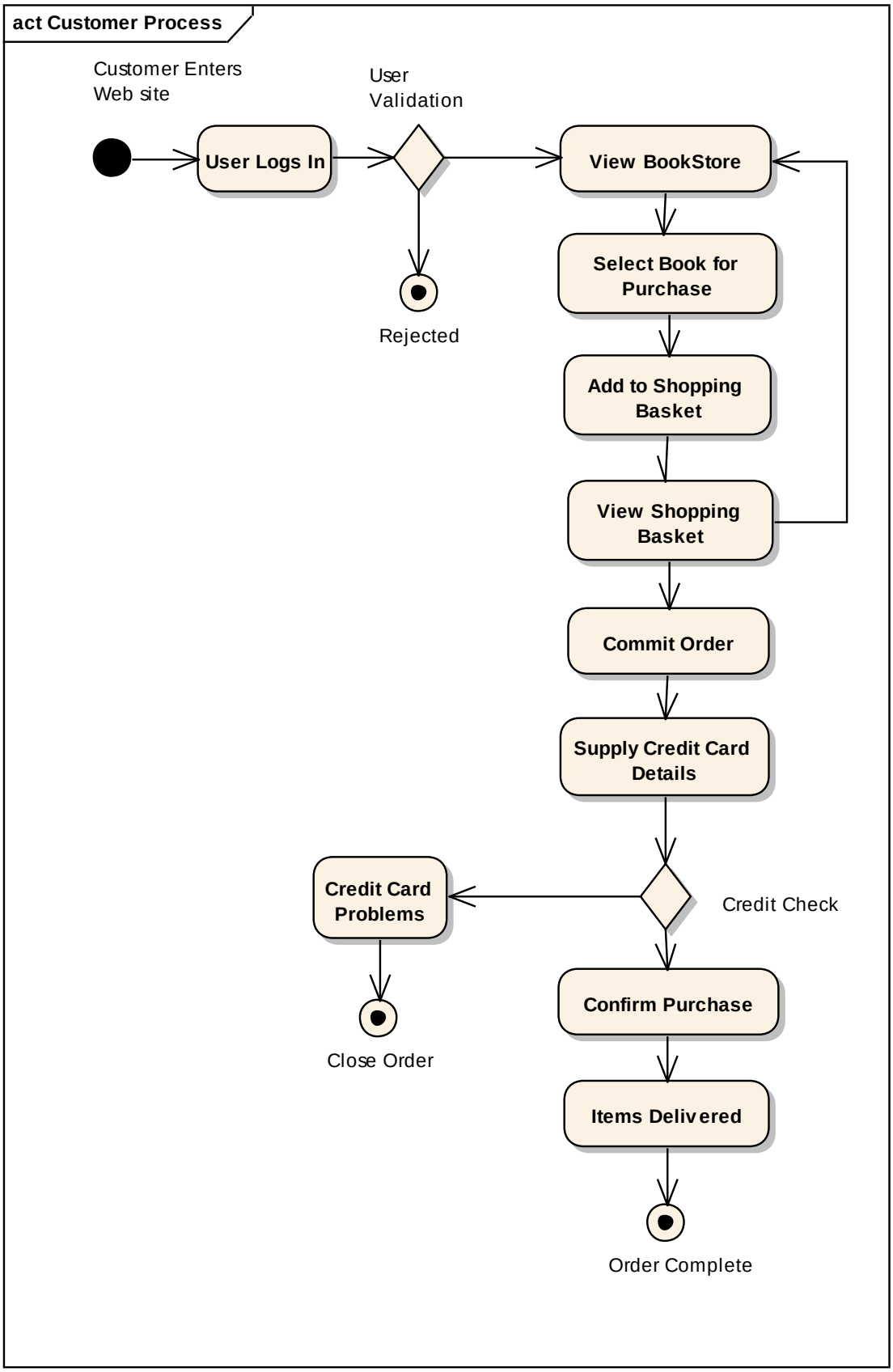

act Customer Process
Customer Enters Web site
User Validation
User Logs In
View BookStore
Select Book for Purchase
Add to Shopping Basket
View Shopping Basket
Commit Order
Supply Credit Card Details
Credit Check
Credit Card Problems
Confirm Purchase
Items Delivered
Rejected
Close Order
Order Complete

Diagram: InteruptibleActivityRegion

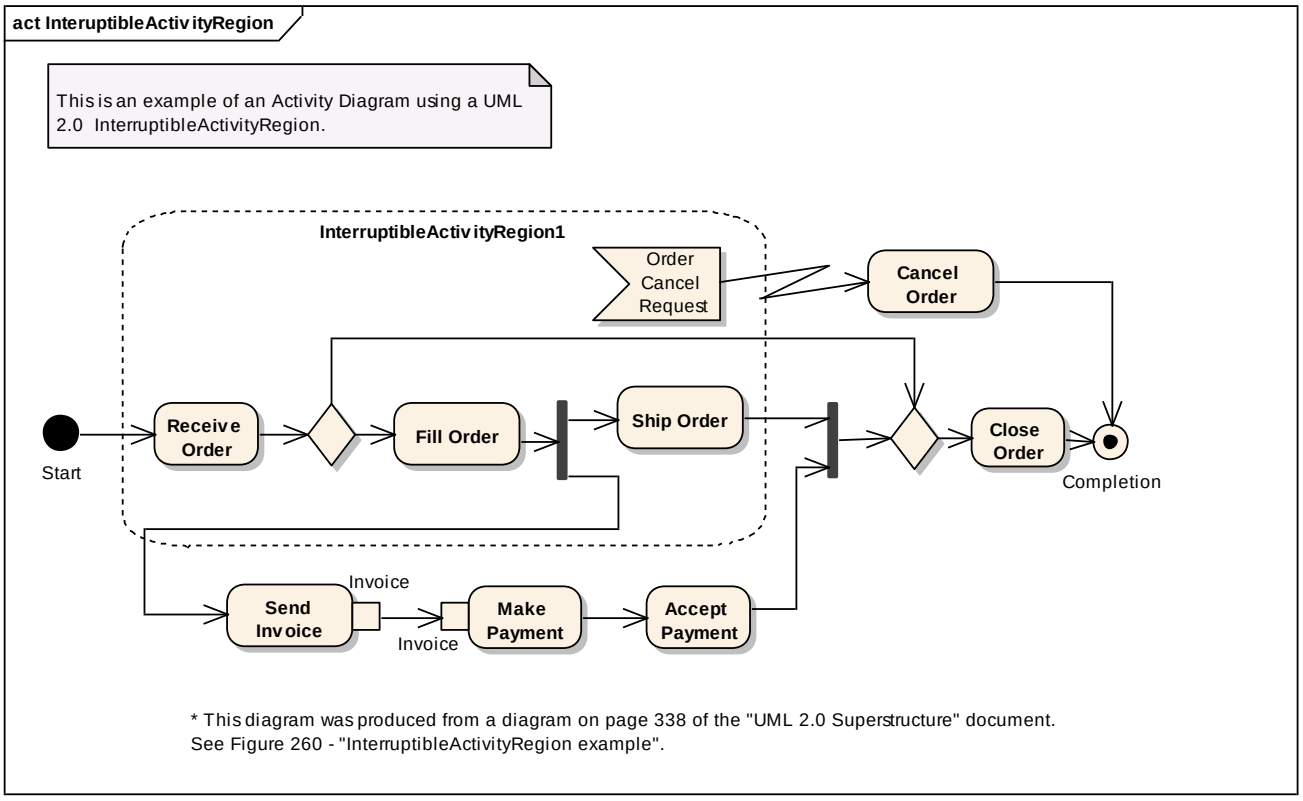


Diagram: Manage Inventory

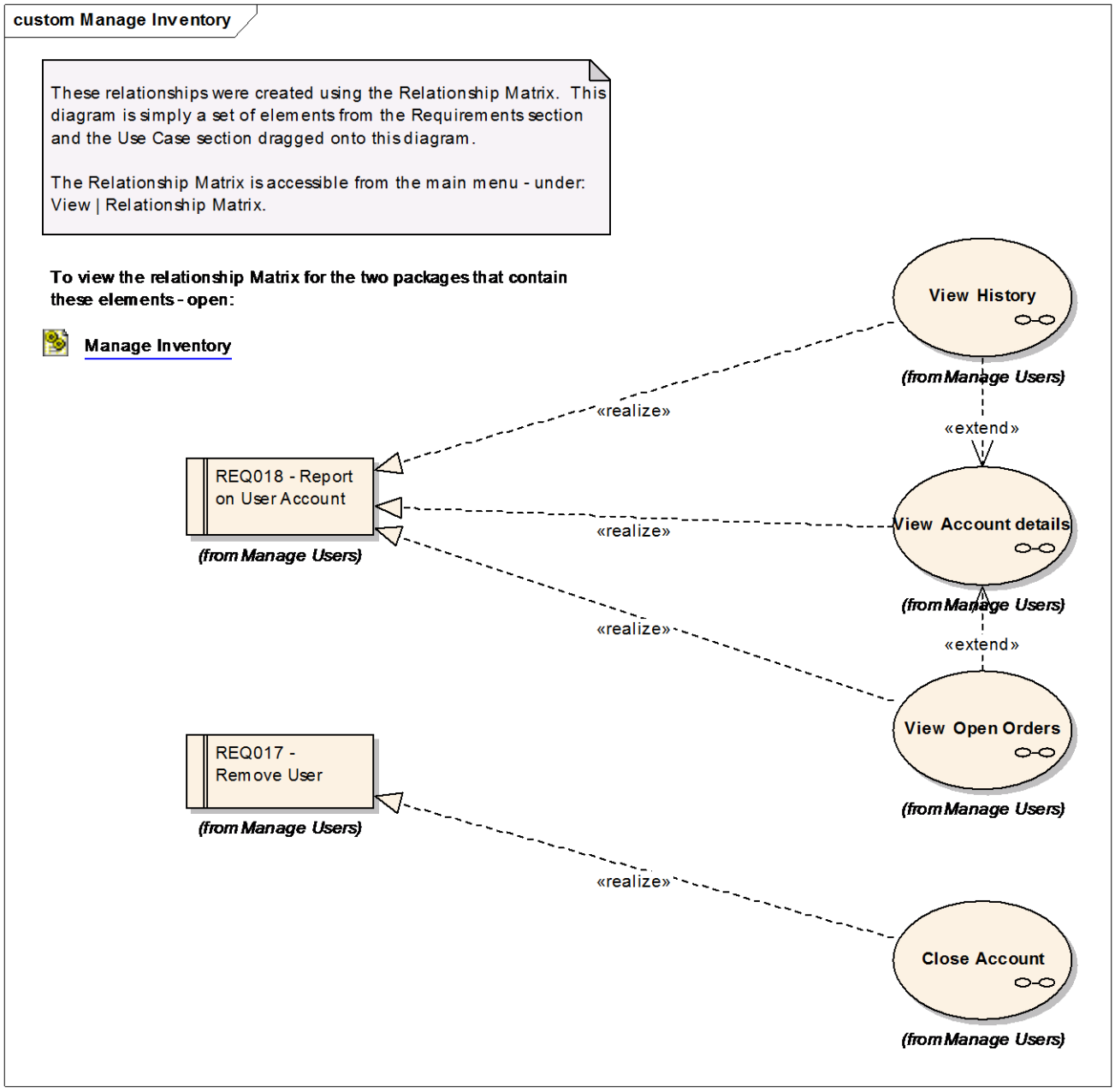


Diagram: ManageUsers

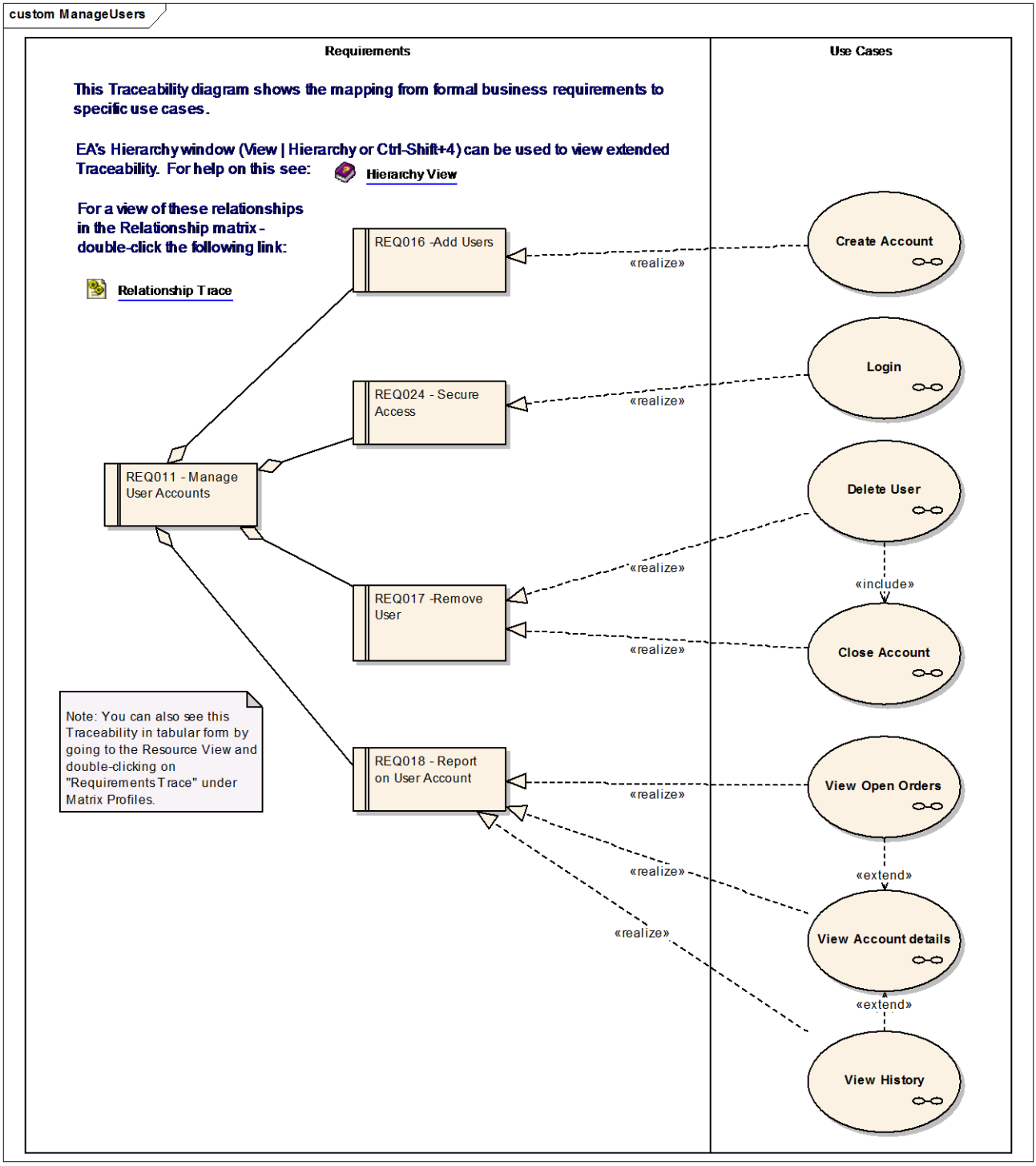


Diagram: Non-Functional Requirements Model

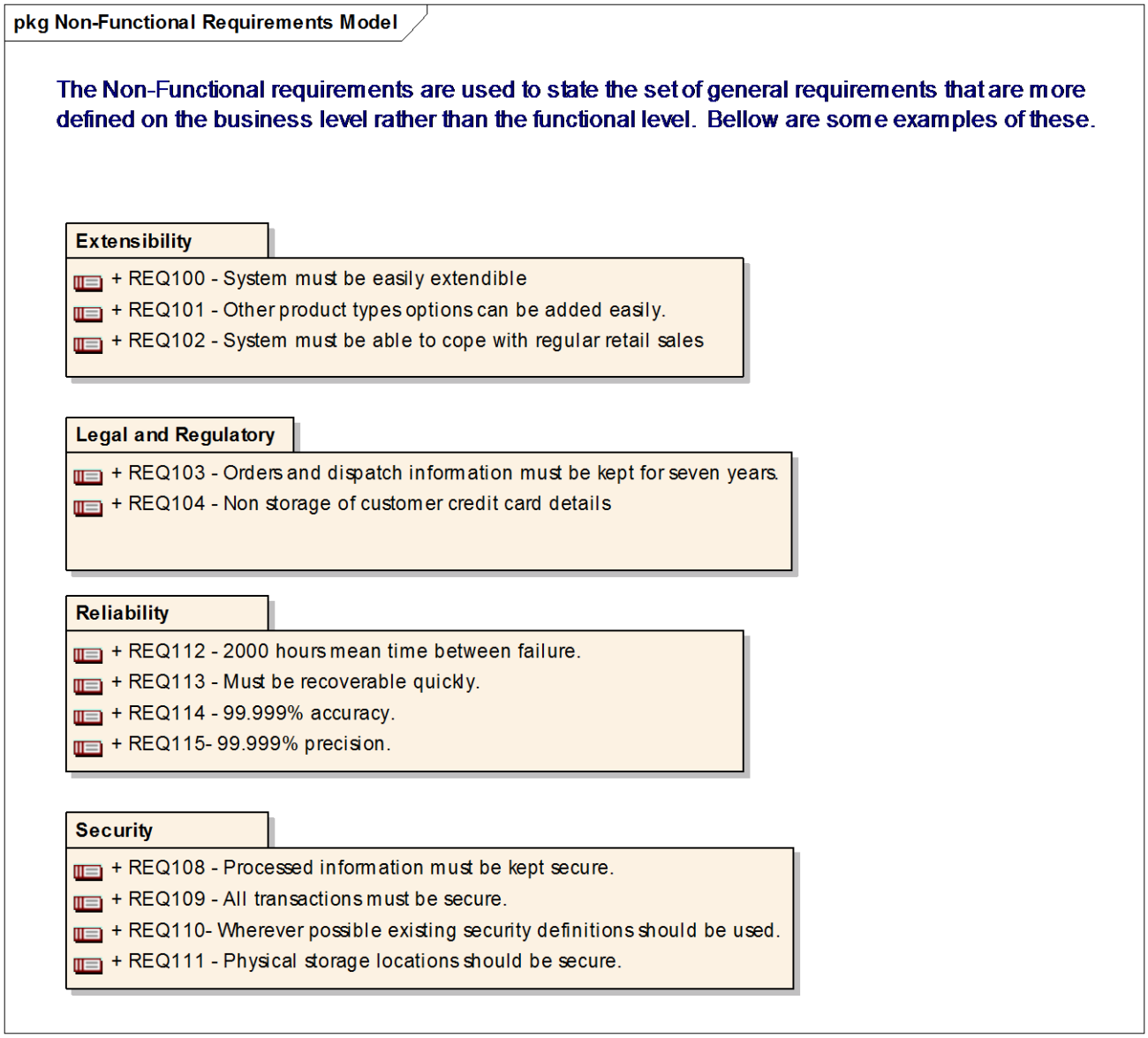


Diagram: Extensibility Main

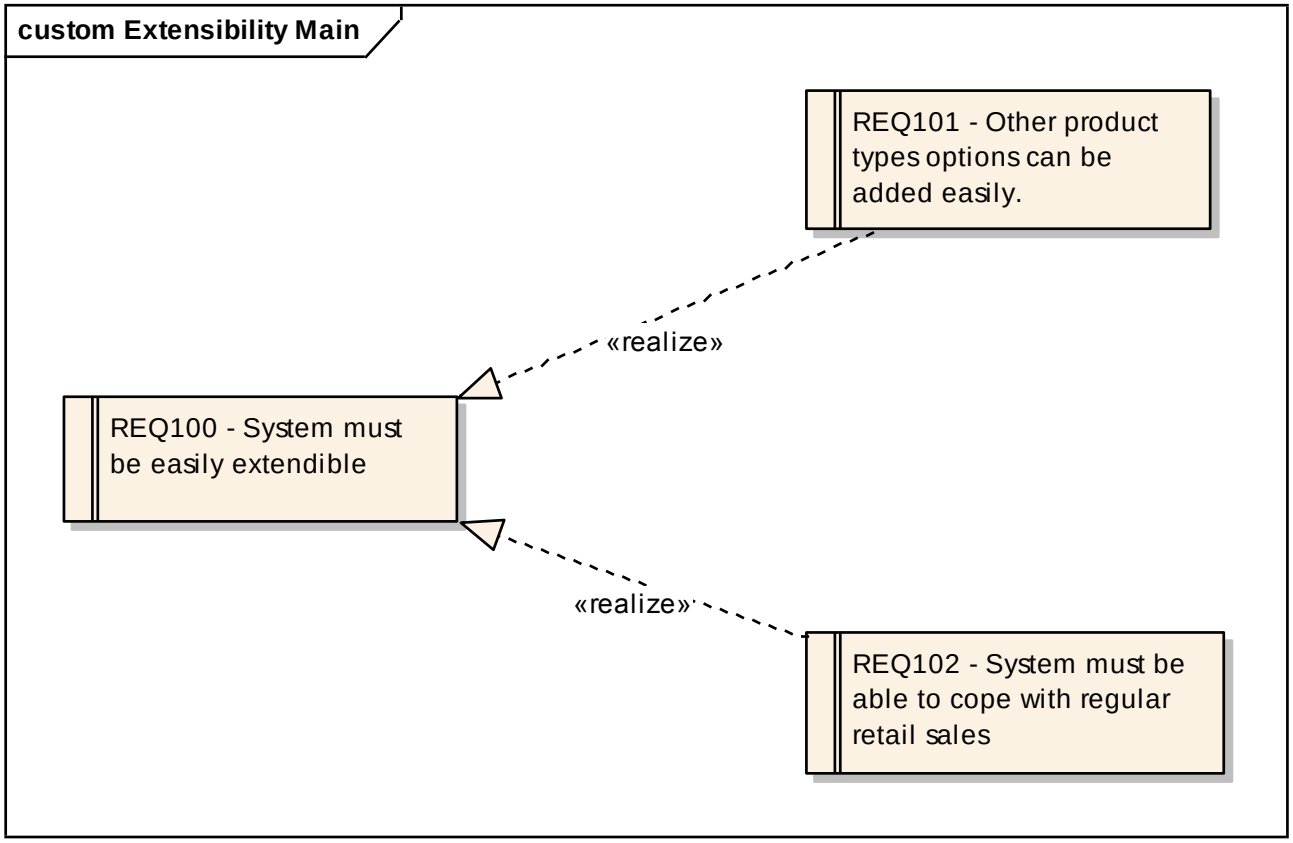


Diagram: Legal and Regulatory

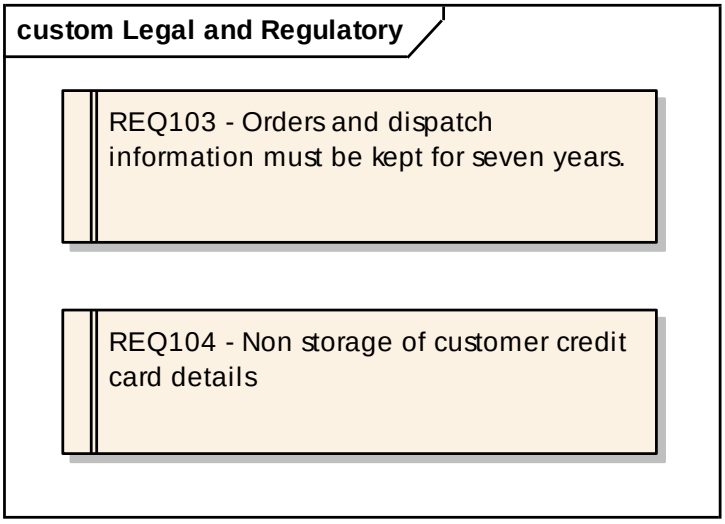


Diagram: Performance Main

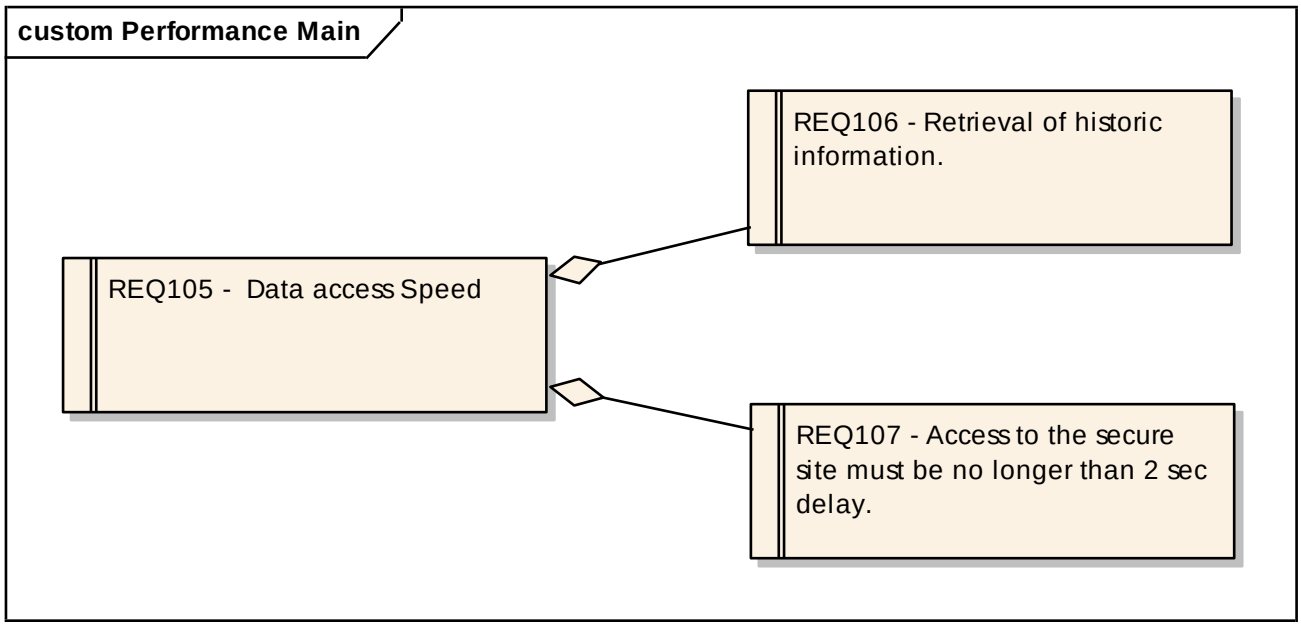


Diagram: Reliability Main

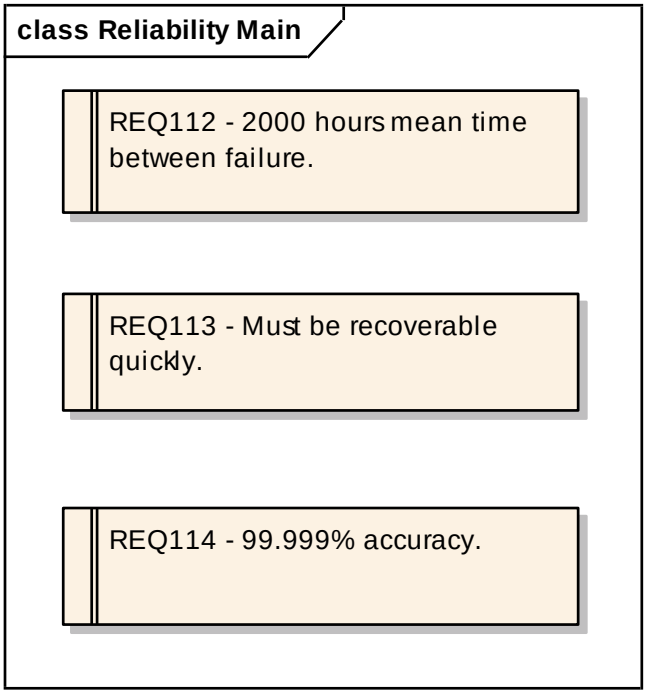


Diagram: Security Main

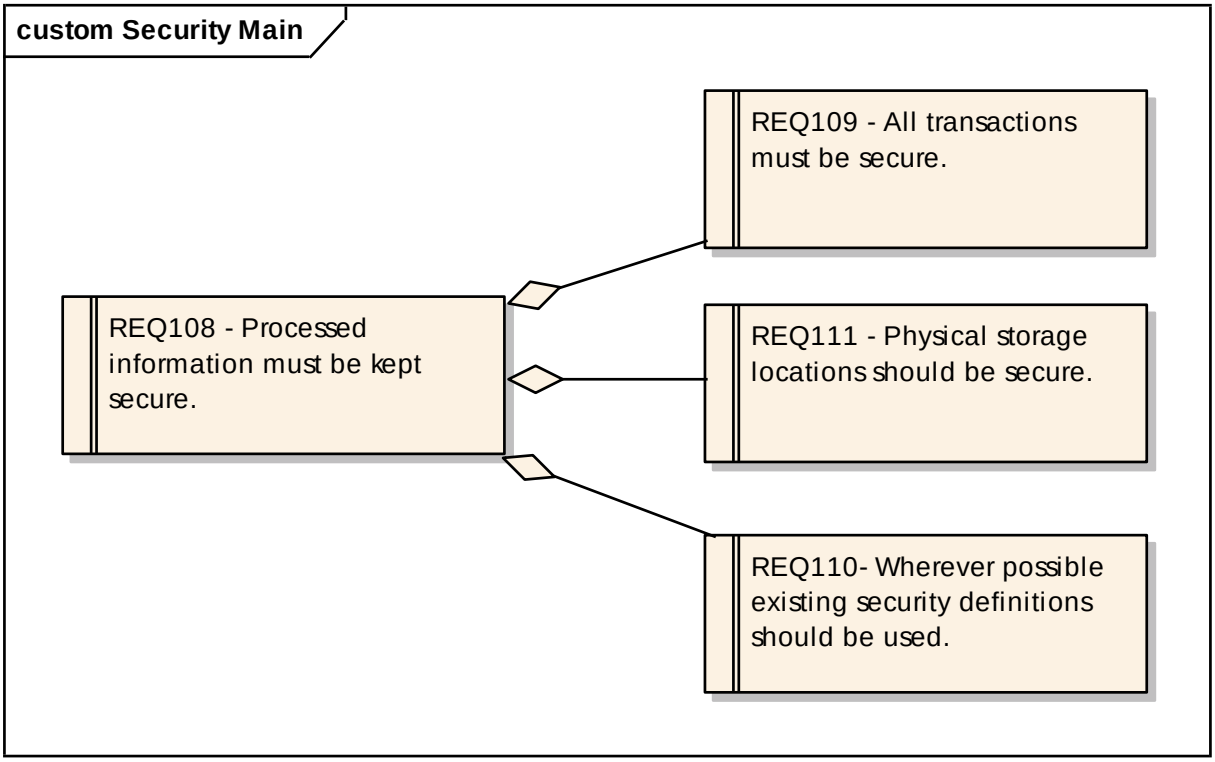


Diagram: System Model

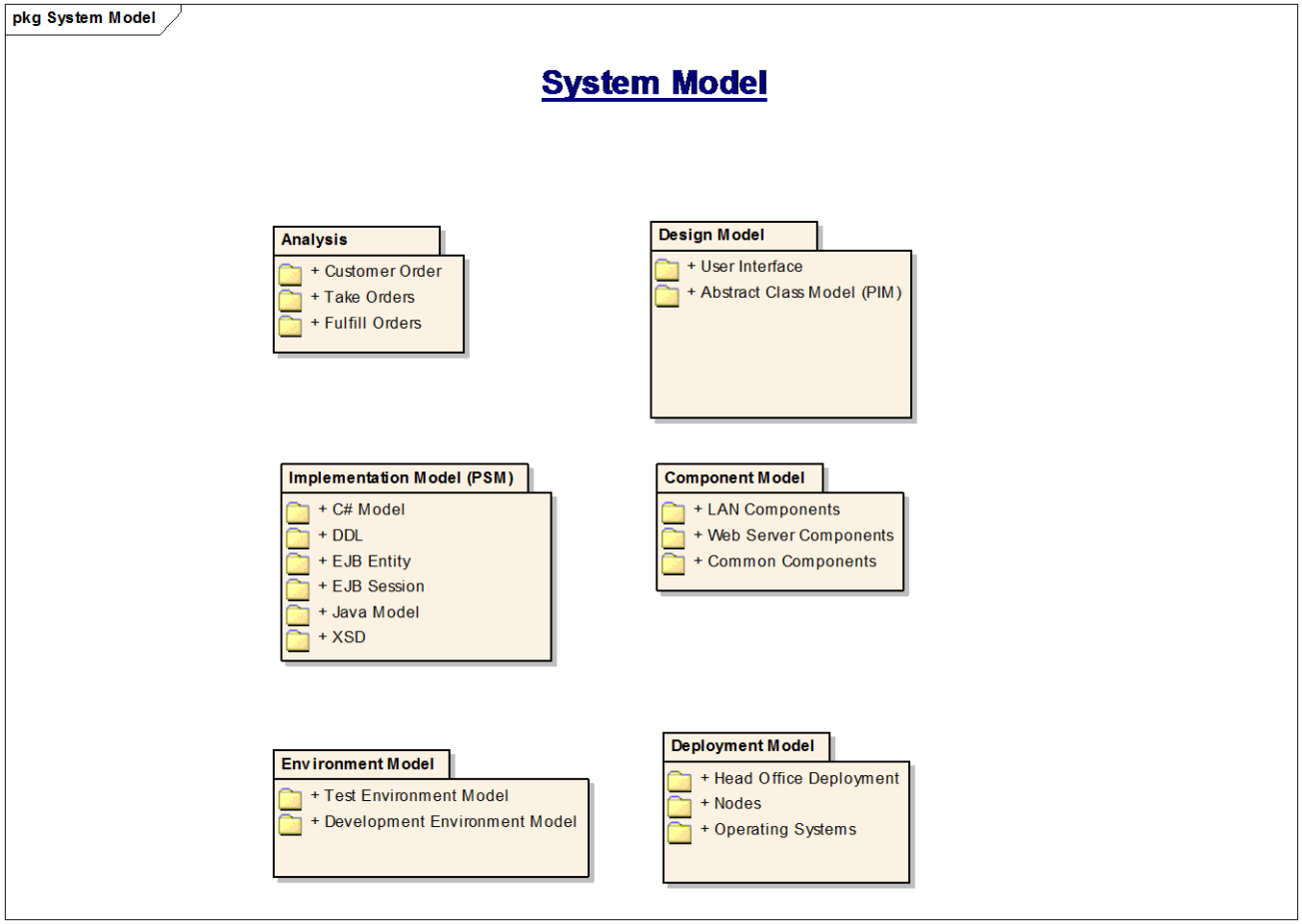


Diagram: Design Model

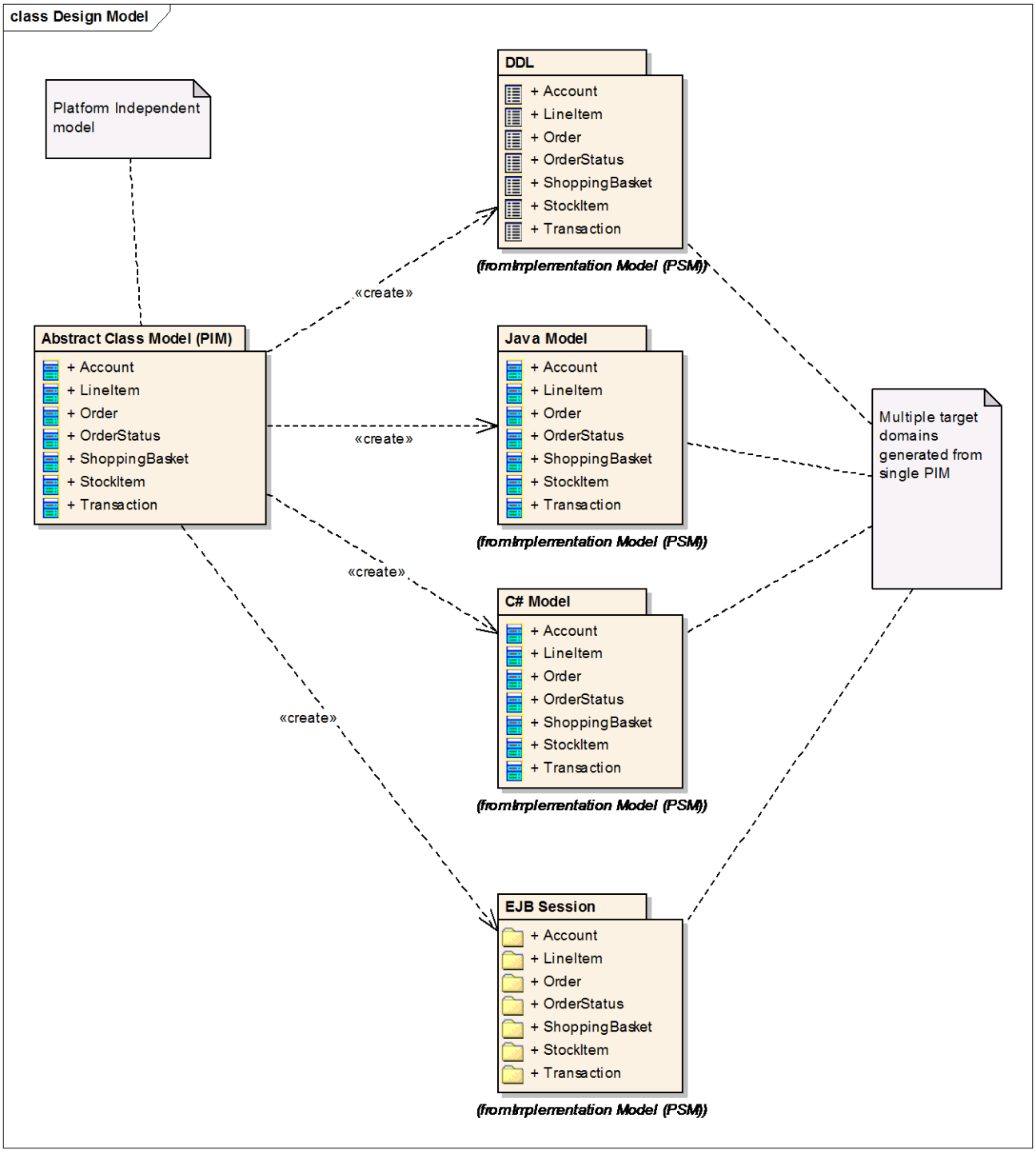


Diagram: Class Model

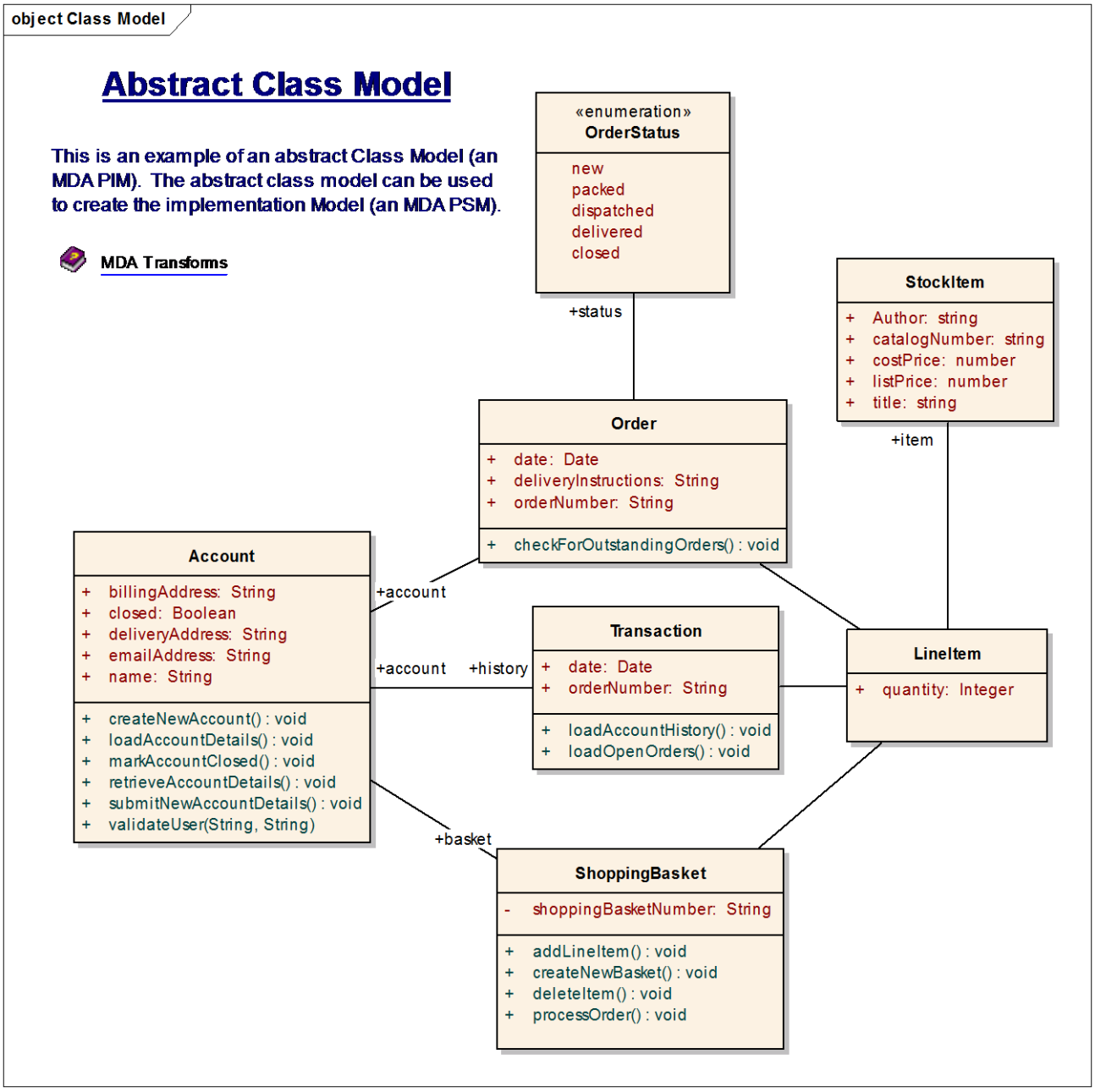


Diagram: User Interface

pkg User Interface

User Interface

EA supports the creation of User Interface screens. Below are links to the diagrams defining the User Interface design for the system under development.

Browse Catalogue

Register User : Register User

PlaceOrder : Place Order

View Basket

Login Screen : Login Screen

The Elements used to create these are available from : Toolbox | Custom. The custom elements used are the 'Screen' and 'UI Control' Elements.

Changing the Stereotype on a UI element allows different types of User Interface elements to be selected.

For more information on using these see: User Interface Group UI Elements

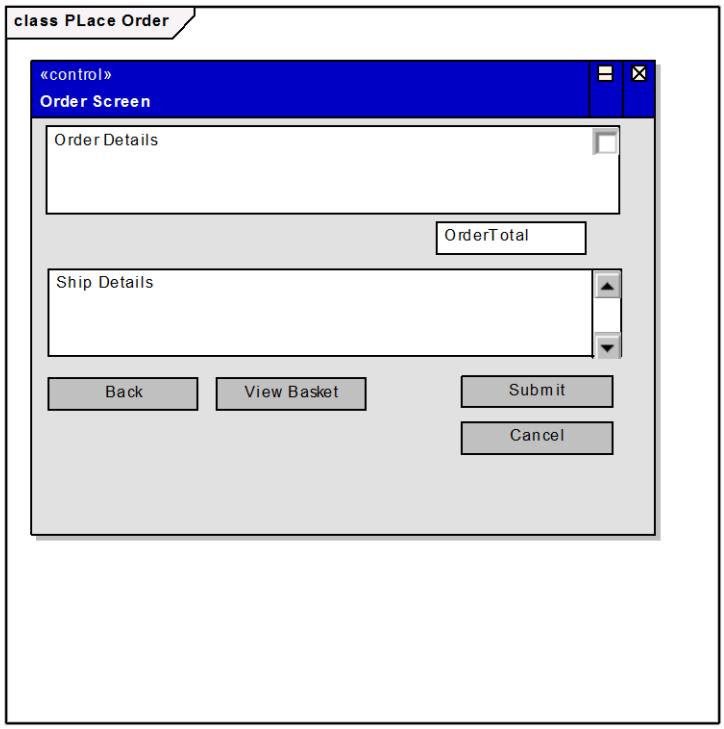


Diagram: User Management

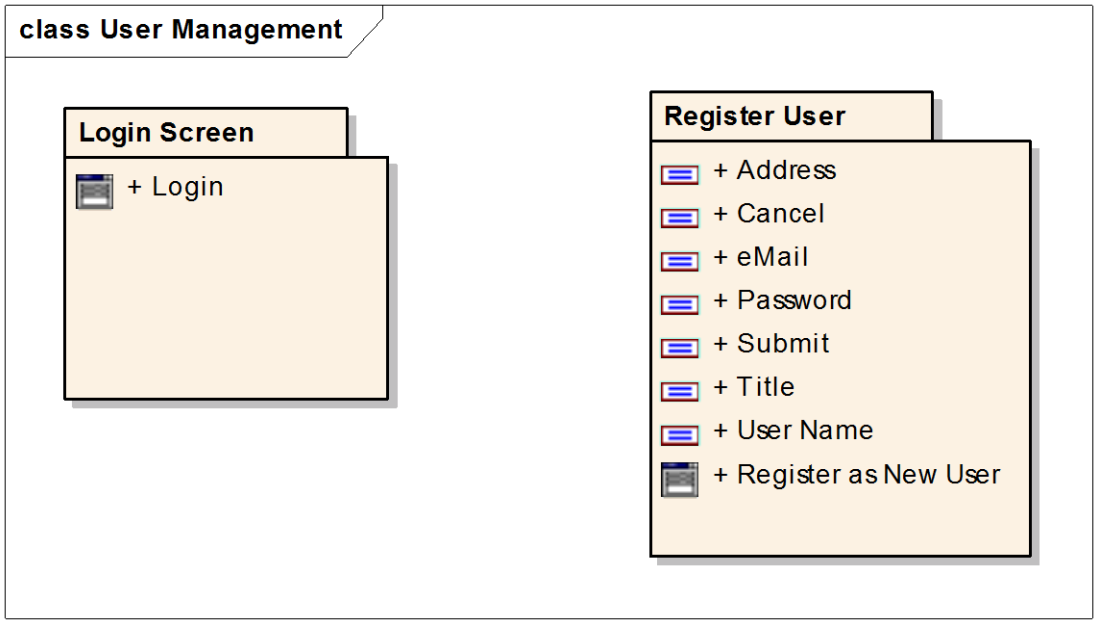


Diagram: Implementation

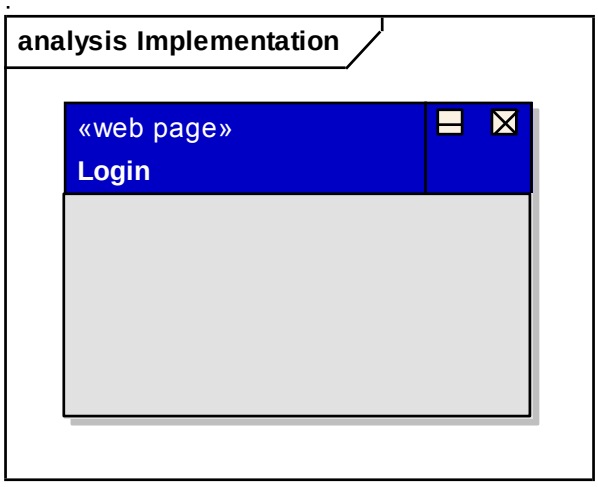


Diagram: Login Screen

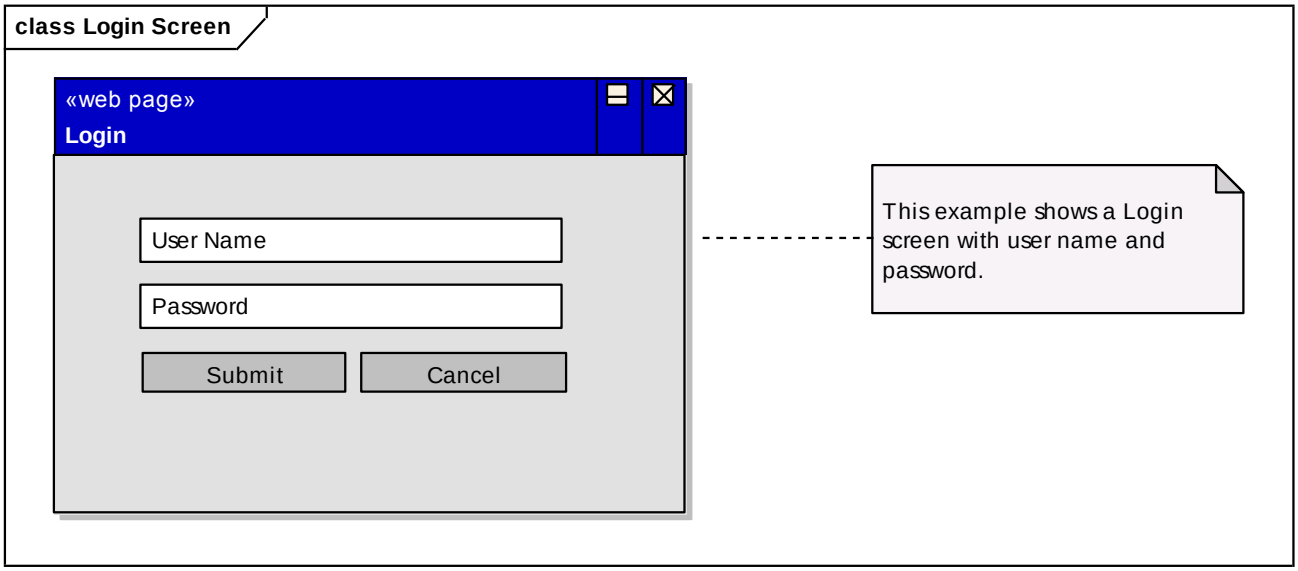


Diagram: Register User

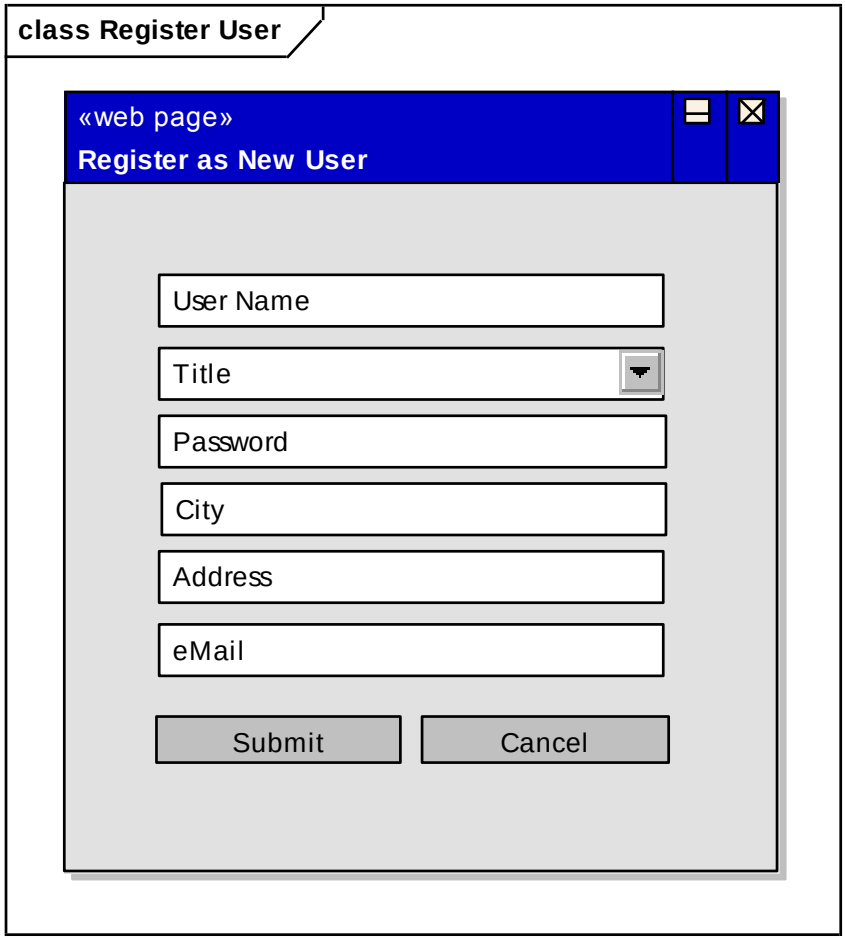


Diagram: Inventory

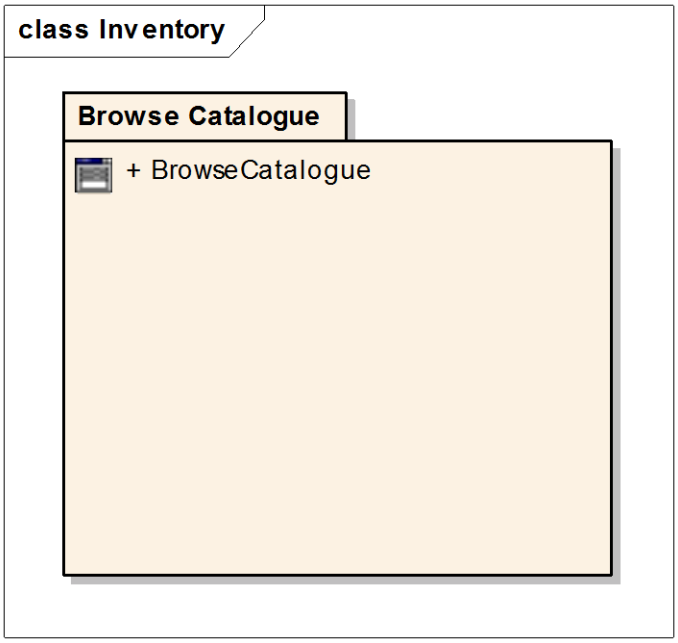


Diagram: Browse

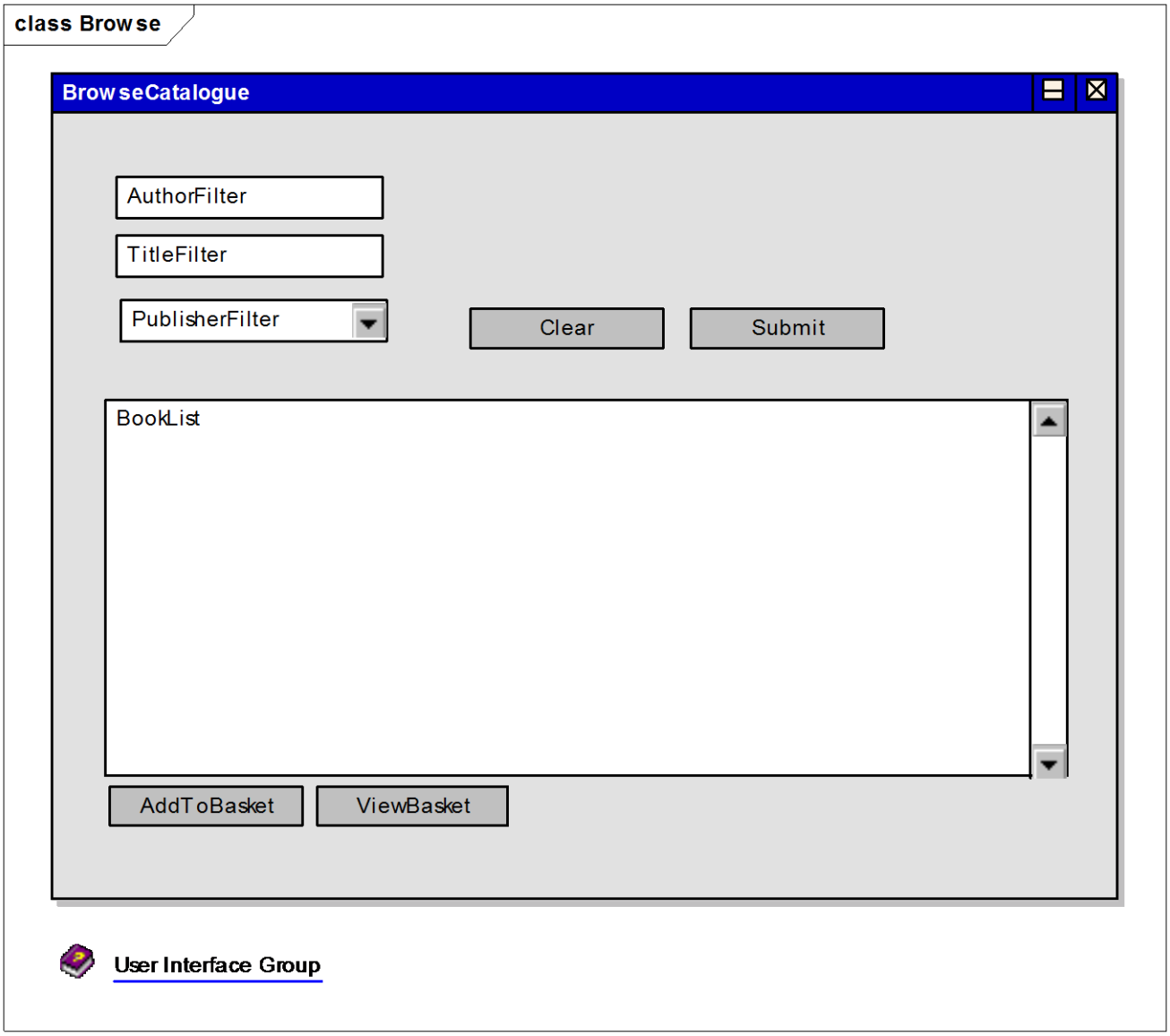


Diagram: Orders

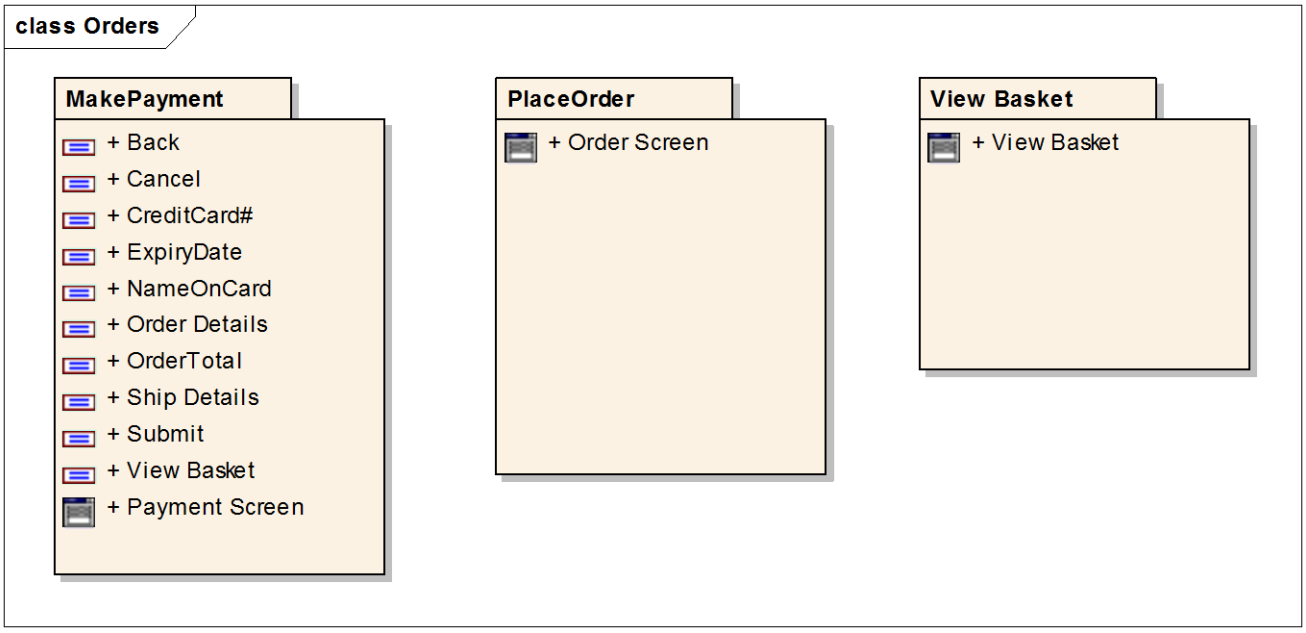


Diagram: Make Payment

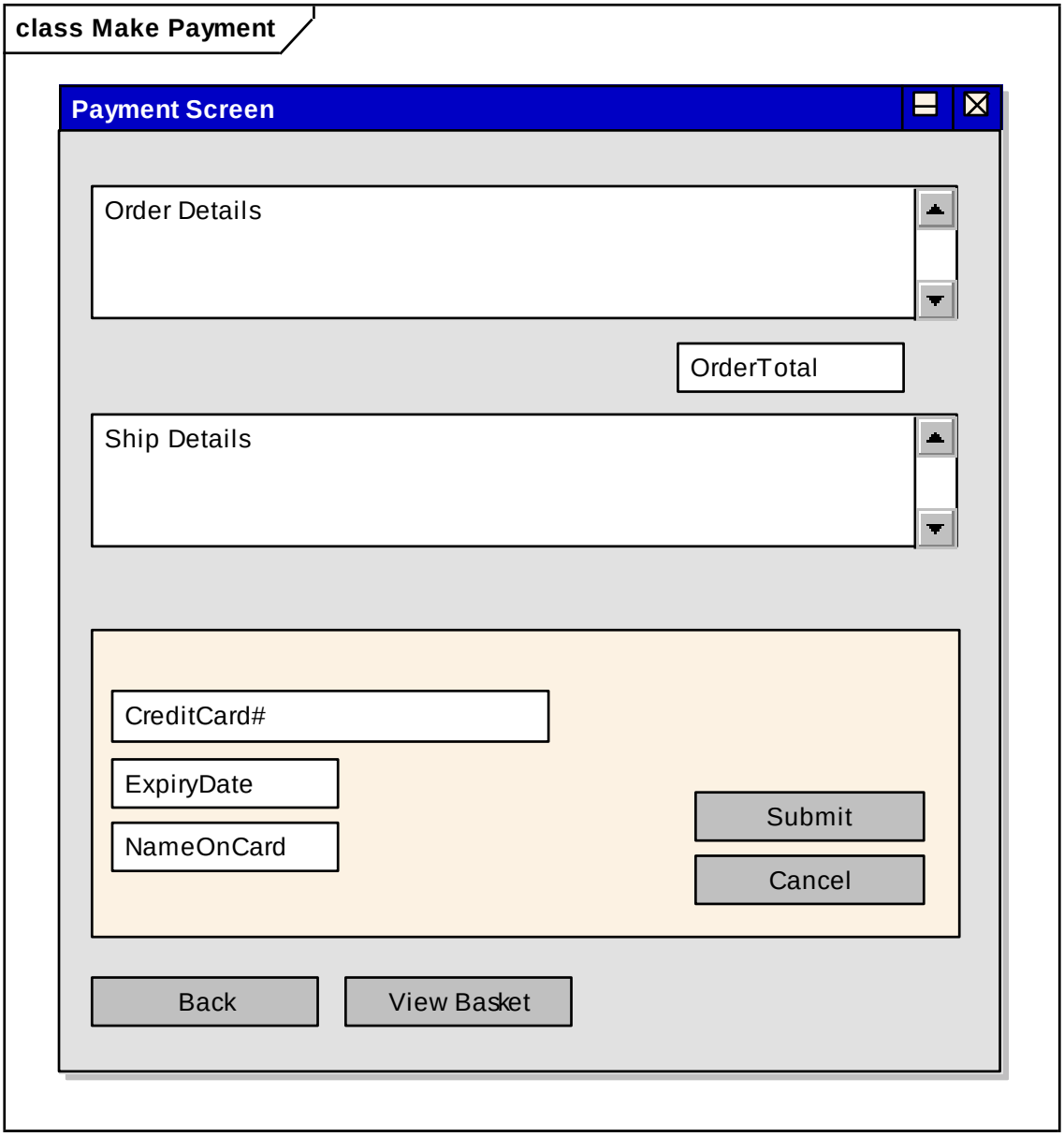


Diagram: PLace Order

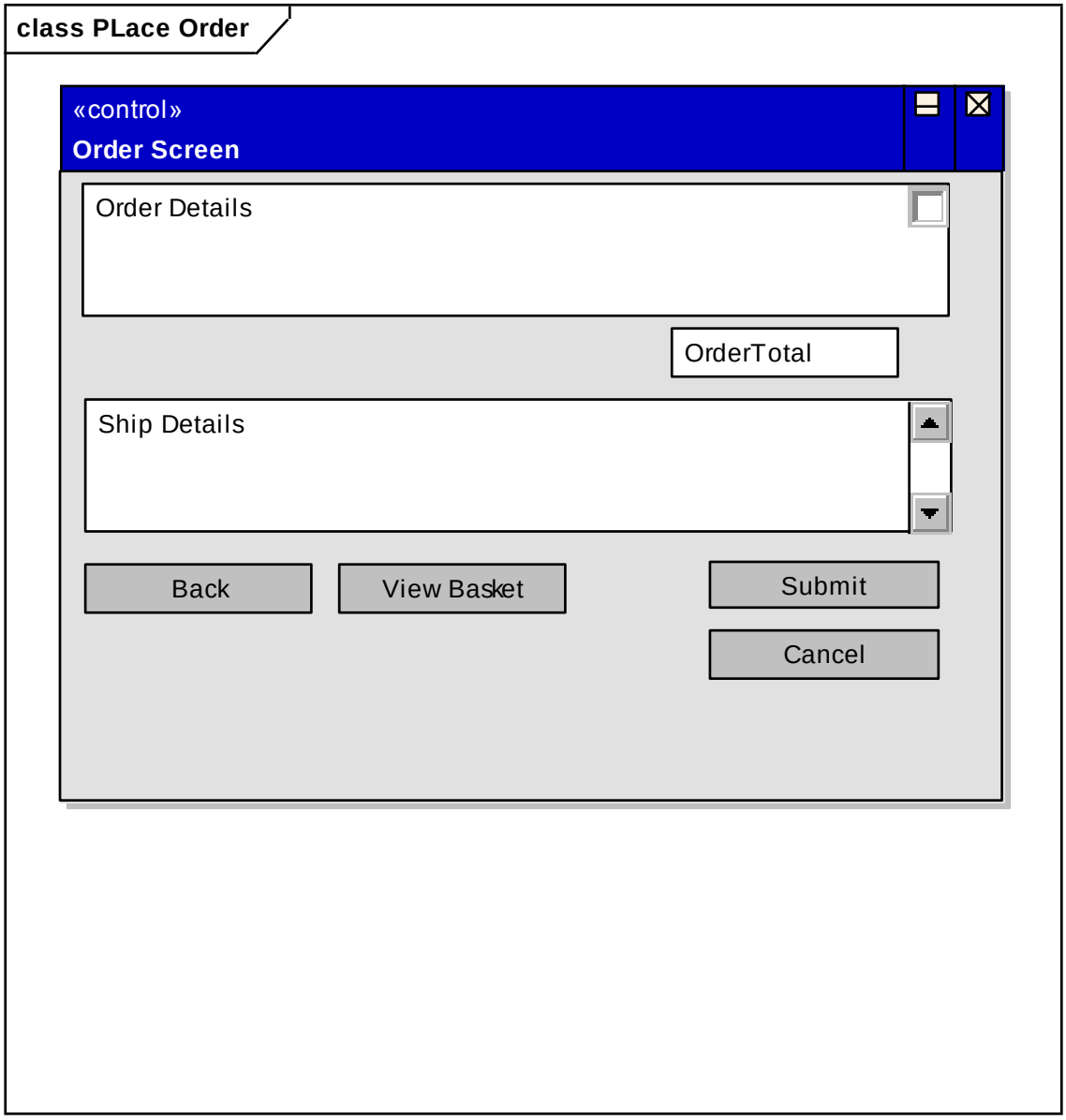


Diagram: View Cart

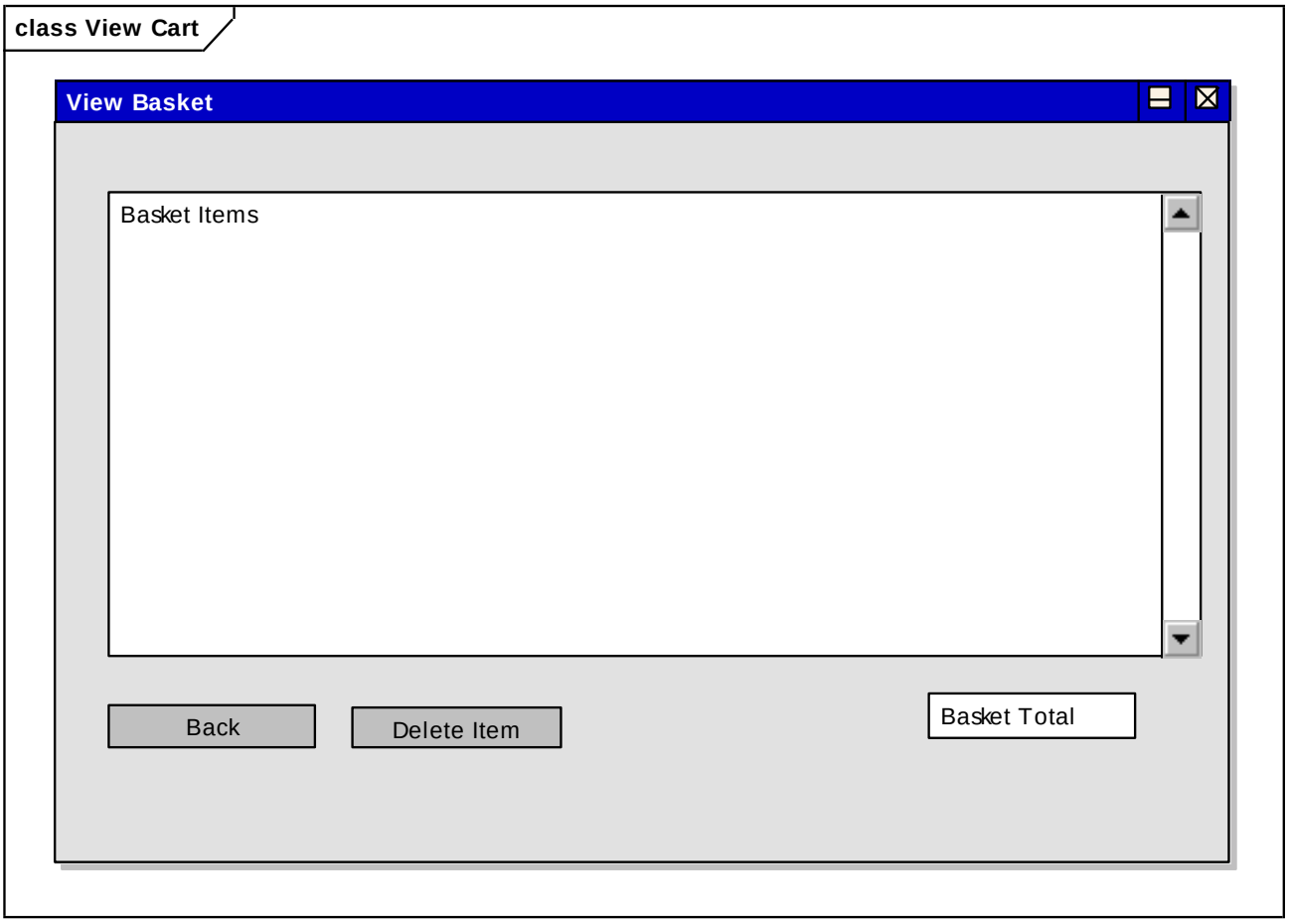


Diagram: Traceability

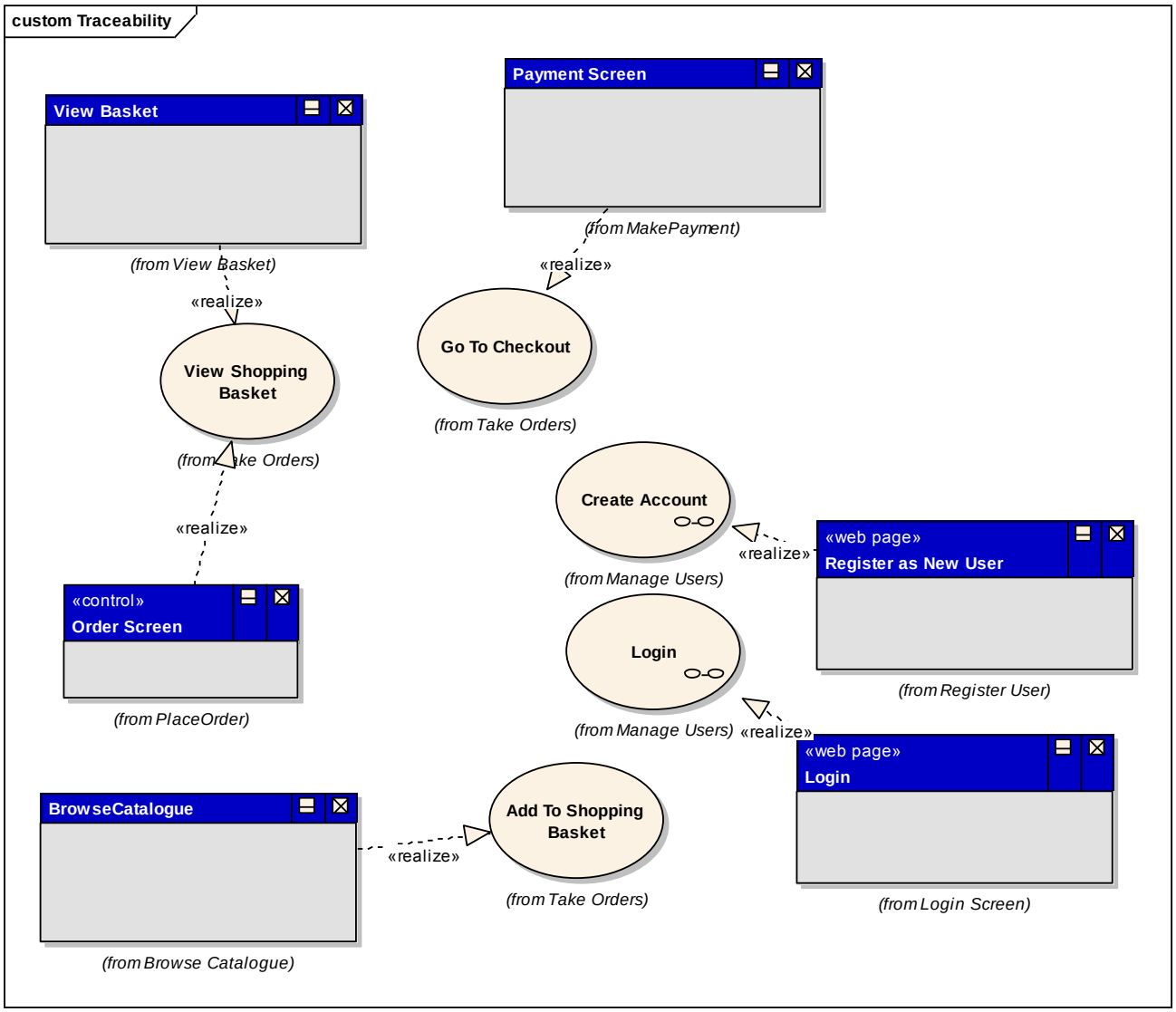


Diagram: Component Model

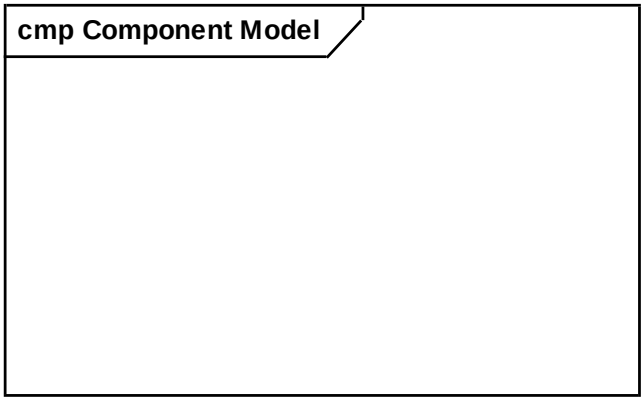


Diagram: SQL Server 2003

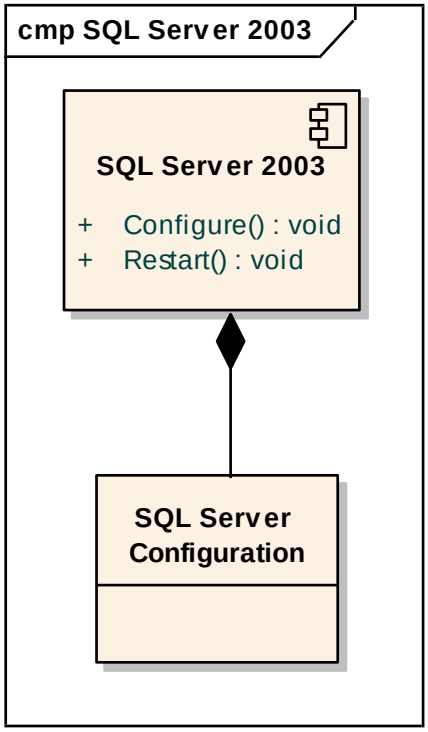


Diagram: Implementation

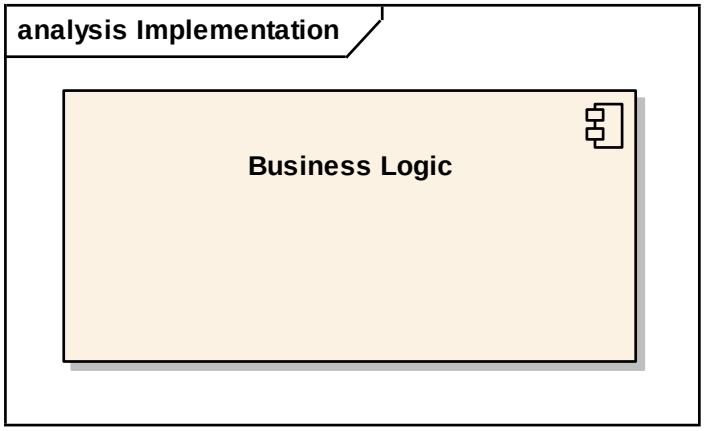


Diagram: Server Components

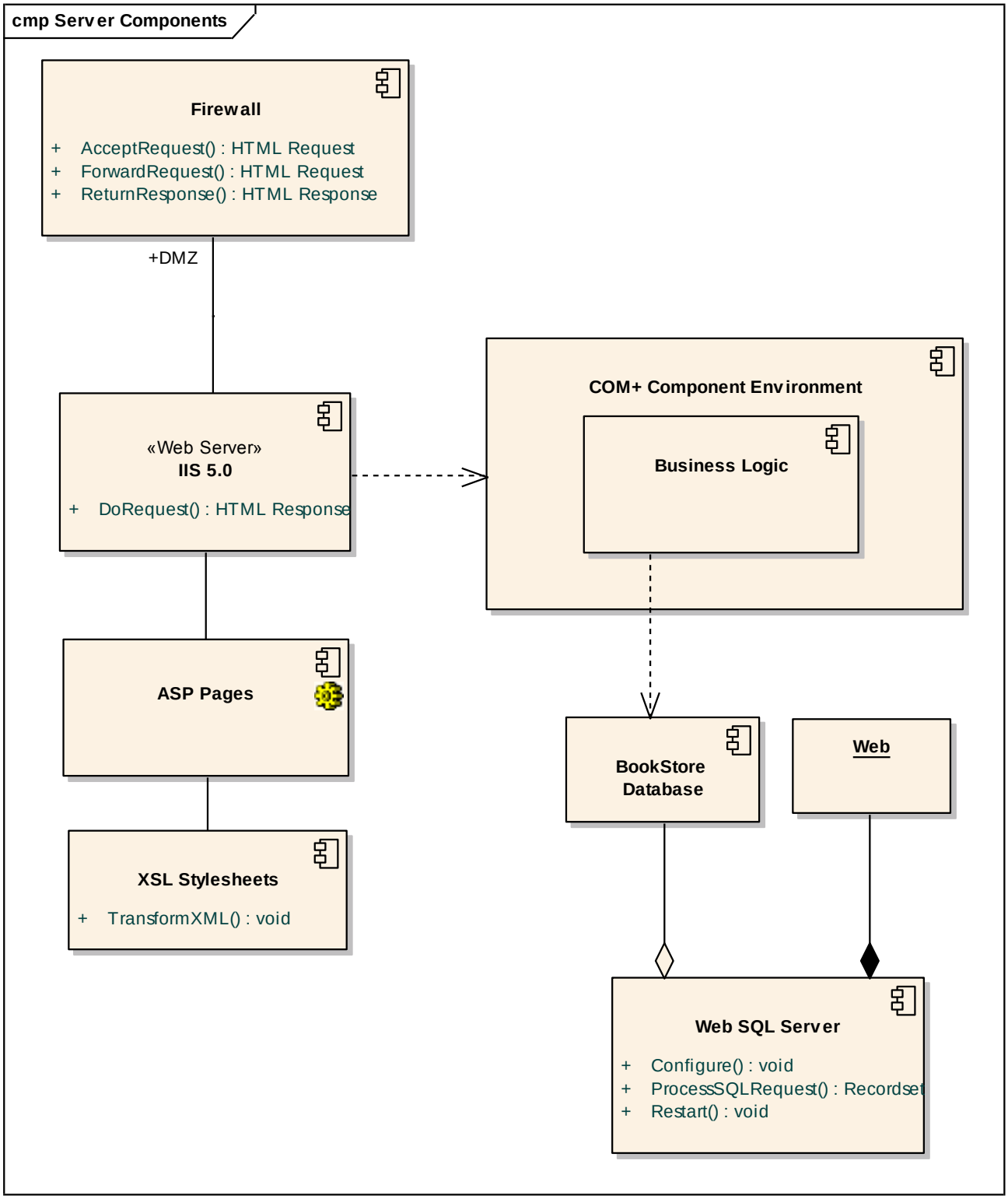


Diagram: LAN Components

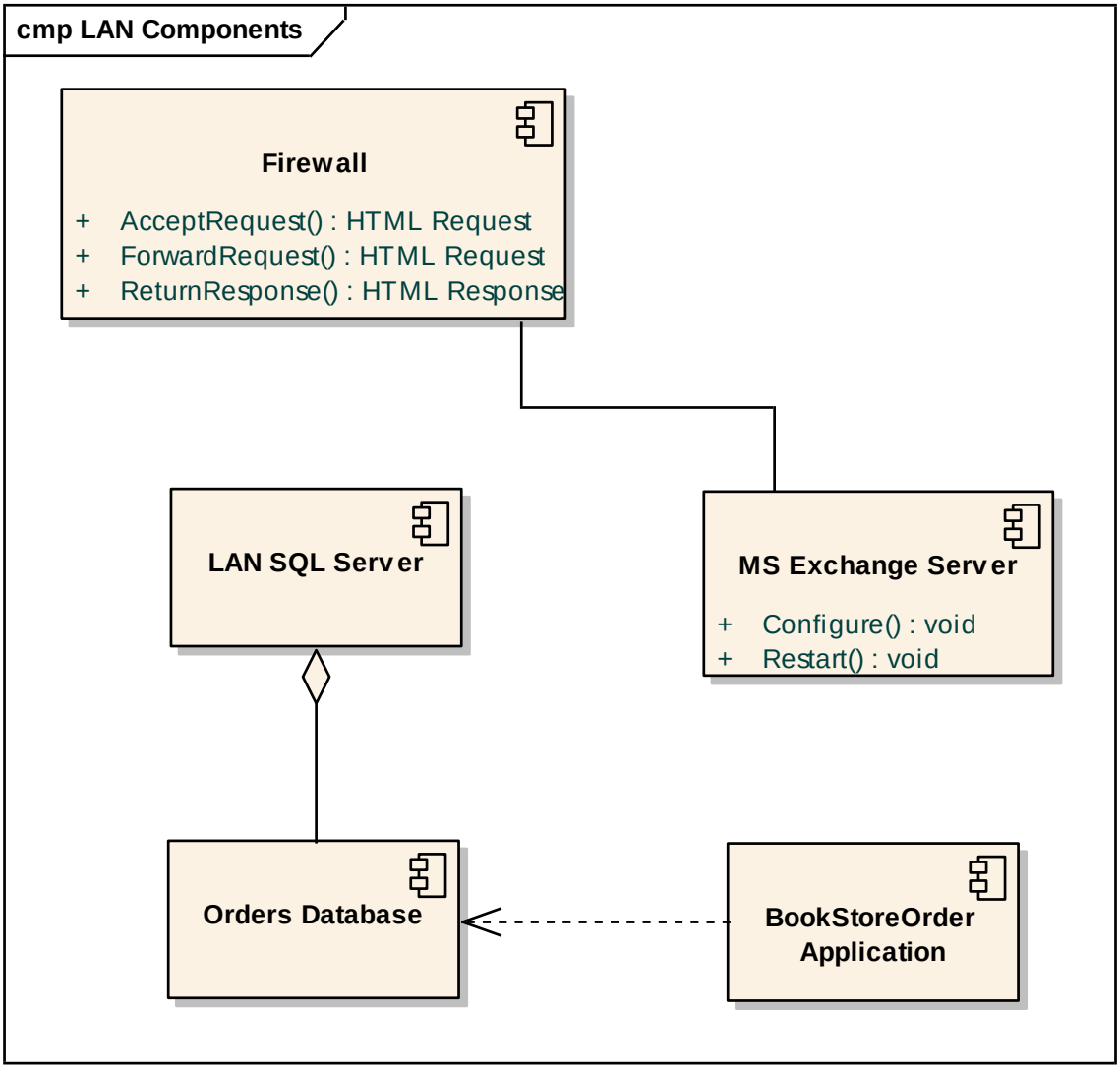


Diagram: Deployment Model

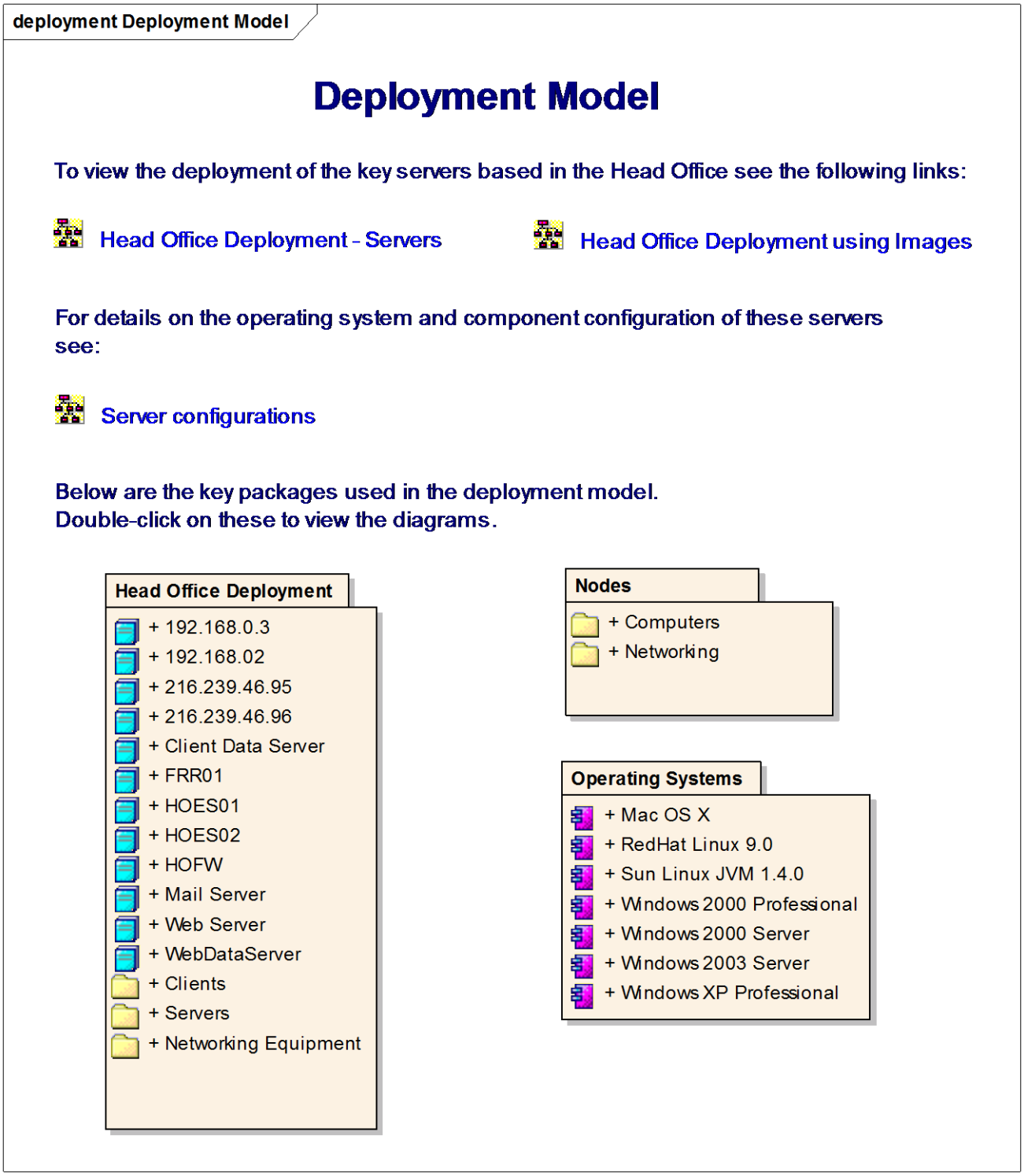


Diagram: HO Server Images

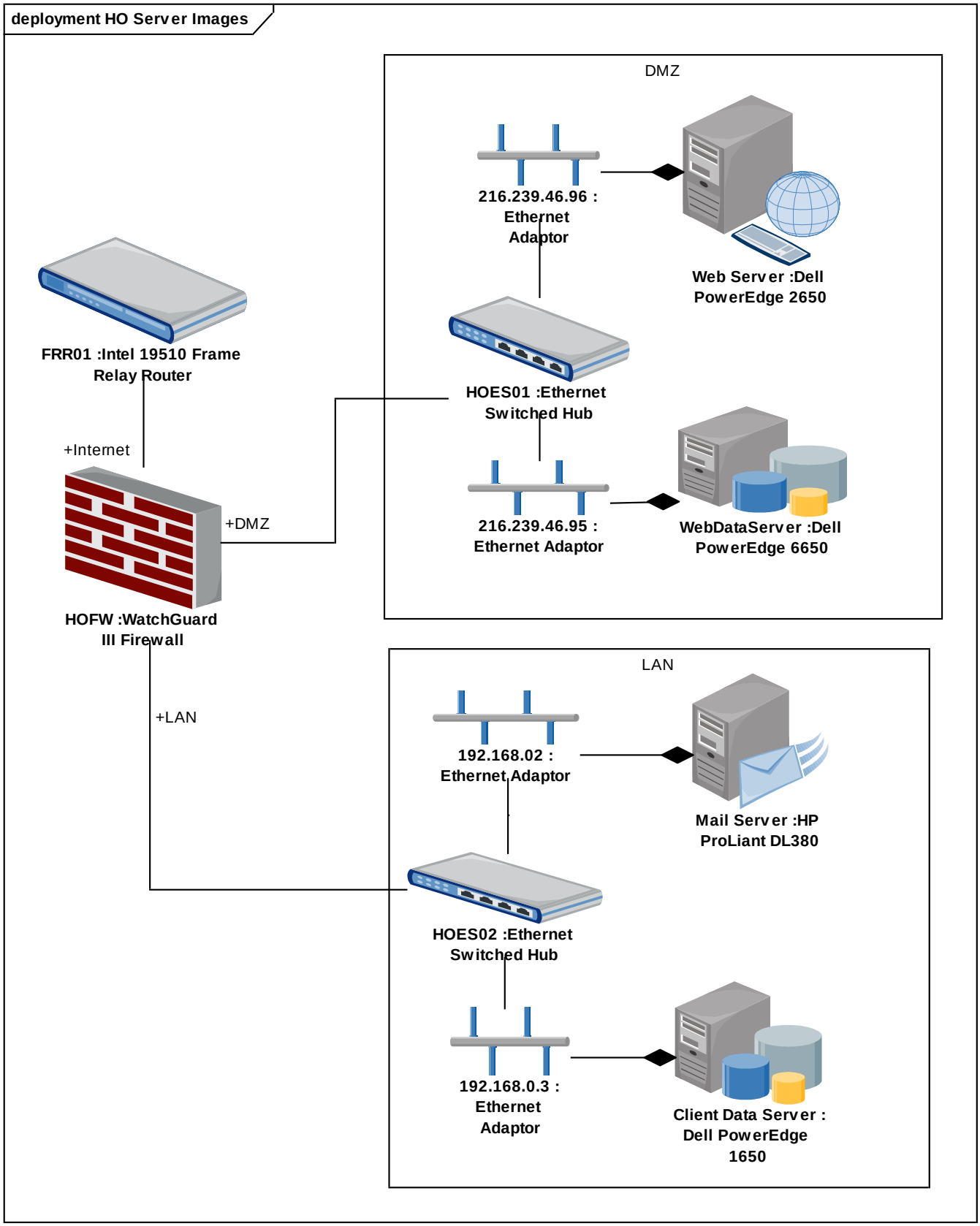

deployment HO Server Images
DMZ
216.239.46.96 : Ethernet Adaptor
Web Server :Dell PowerEdge 2650
FRR01 :Intel 19510 Frame Relay Router
HOES01 :Ethernet Switched Hub
+Internet
216.239.46.95 : Ethernet Adaptor
WebDataServer :Dell PowerEdge 6650
+DMZ
HOFW :WatchGuard III Firewall
LAN
+LAN
192.168.02 : Ethernet Adaptor
Mail Server :HP ProLiant DL380
HOES02 :Ethernet Switched Hub
192.168.0.3 : Ethernet Adaptor
Client Data Server : Dell PowerEdge 1650

Diagram: HO Servers

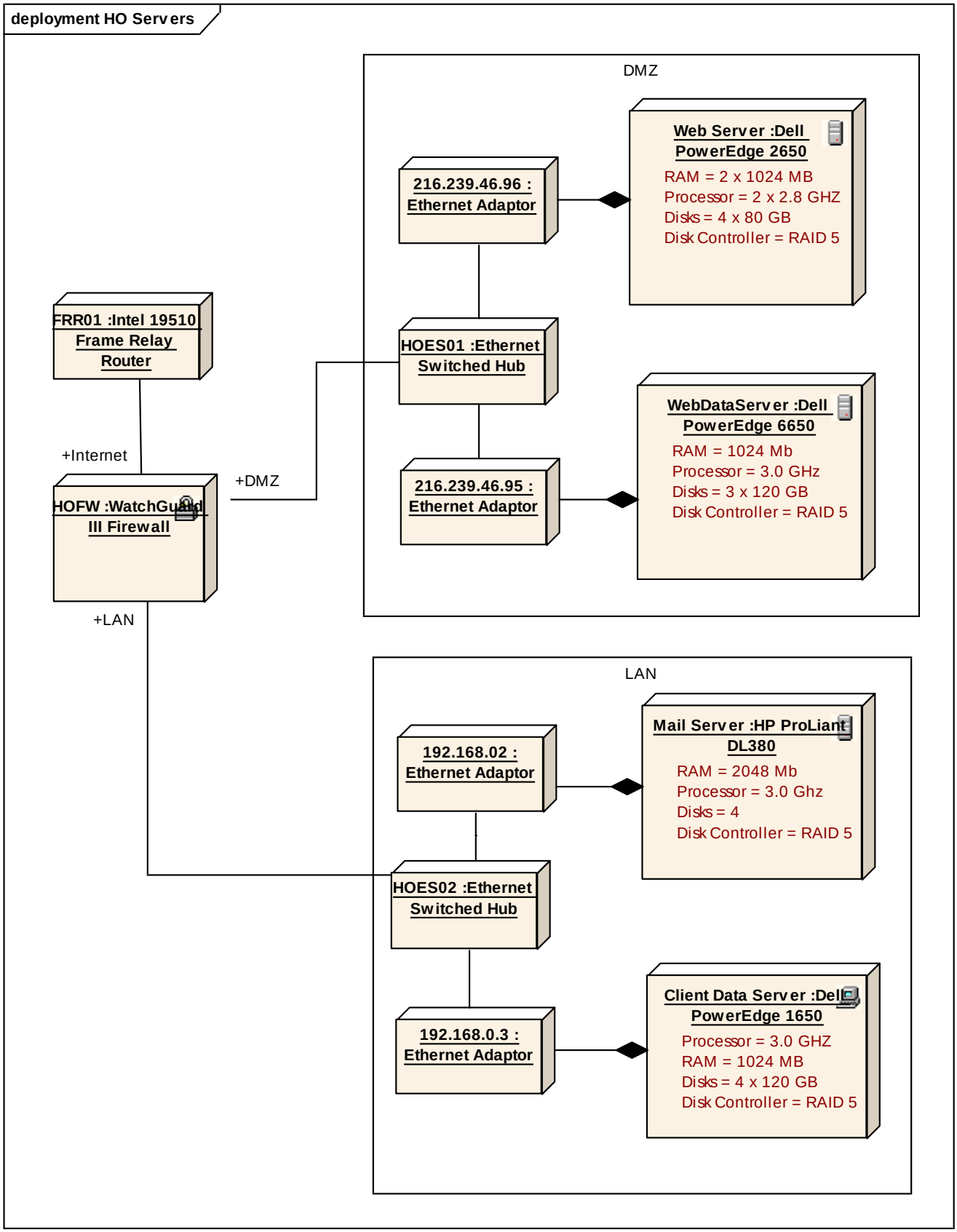

deployment HO Servers
DMZ
Web Server :Dell PowerEdge 2650
RAM = 2 x 1024 MB
Processor = 2 x 2.8 GHZ
Disks = 4 x 80 GB
Disk Controller = RAID 5
216.239.46.96 : Ethernet Adaptor
FRR01 :Intel 19510 Frame Relay Router
HOES01 :Ethernet Switched Hub
WebDataServer :Dell PowerEdge 6650
RAM = 1024 Mb
Processor = 3.0 GHz
Disks = 3 x 120 GB
Disk Controller = RAID 5
216.239.46.95 : Ethernet Adaptor
+Internet
+DMZ
HOFW :WatchGuard III Firewall
+LAN
LAN
Mail Server :HP ProLiant DL380
RAM = 2048 Mb
Processor = 3.0 Ghz
Disks = 4
Disk Controller = RAID 5
192.168.02 : Ethernet Adaptor
HOES02 :Ethernet Switched Hub
Client Data Server :Dell PowerEdge 1650
Processor = 3.0 GHZ
RAM = 1024 MB
Disks = 4 x 120 GB
Disk Controller = RAID 5
192.168.0.3 : Ethernet Adaptor

Diagram: Office Client 1

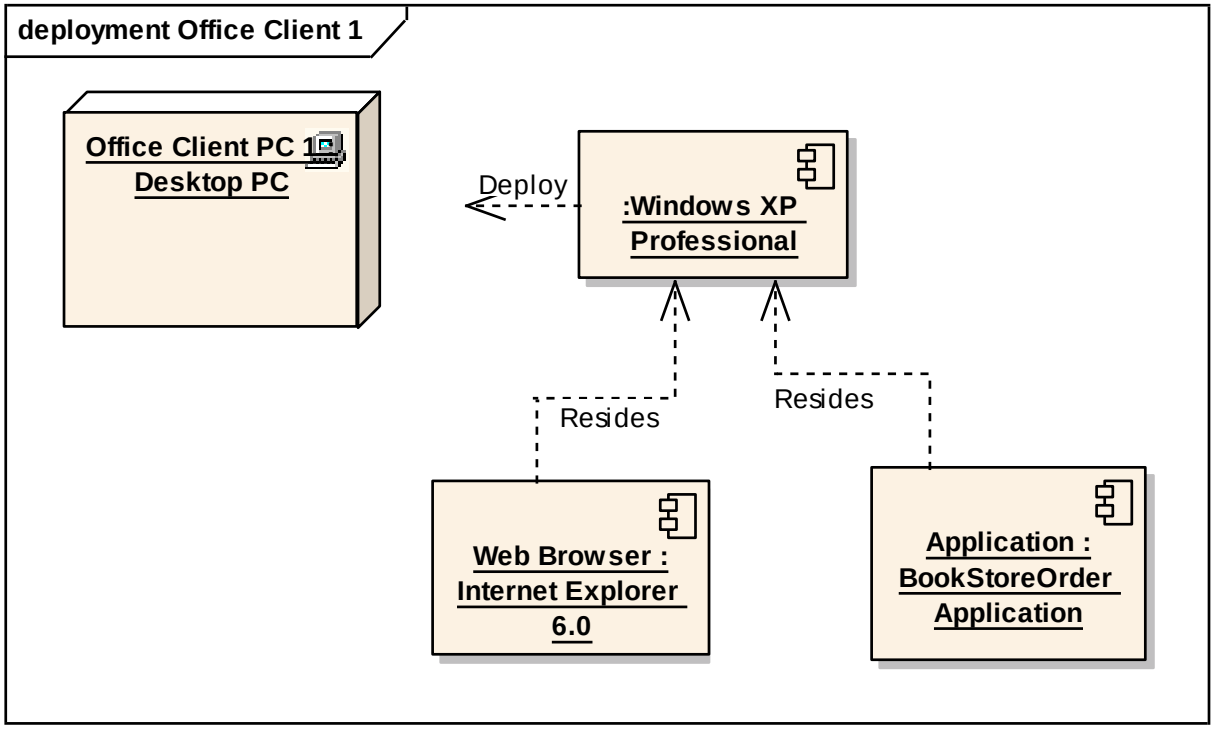


Diagram: Office Client 2

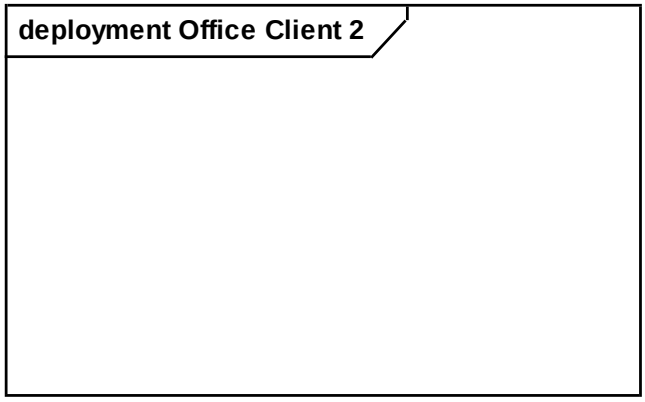


Diagram: Servers

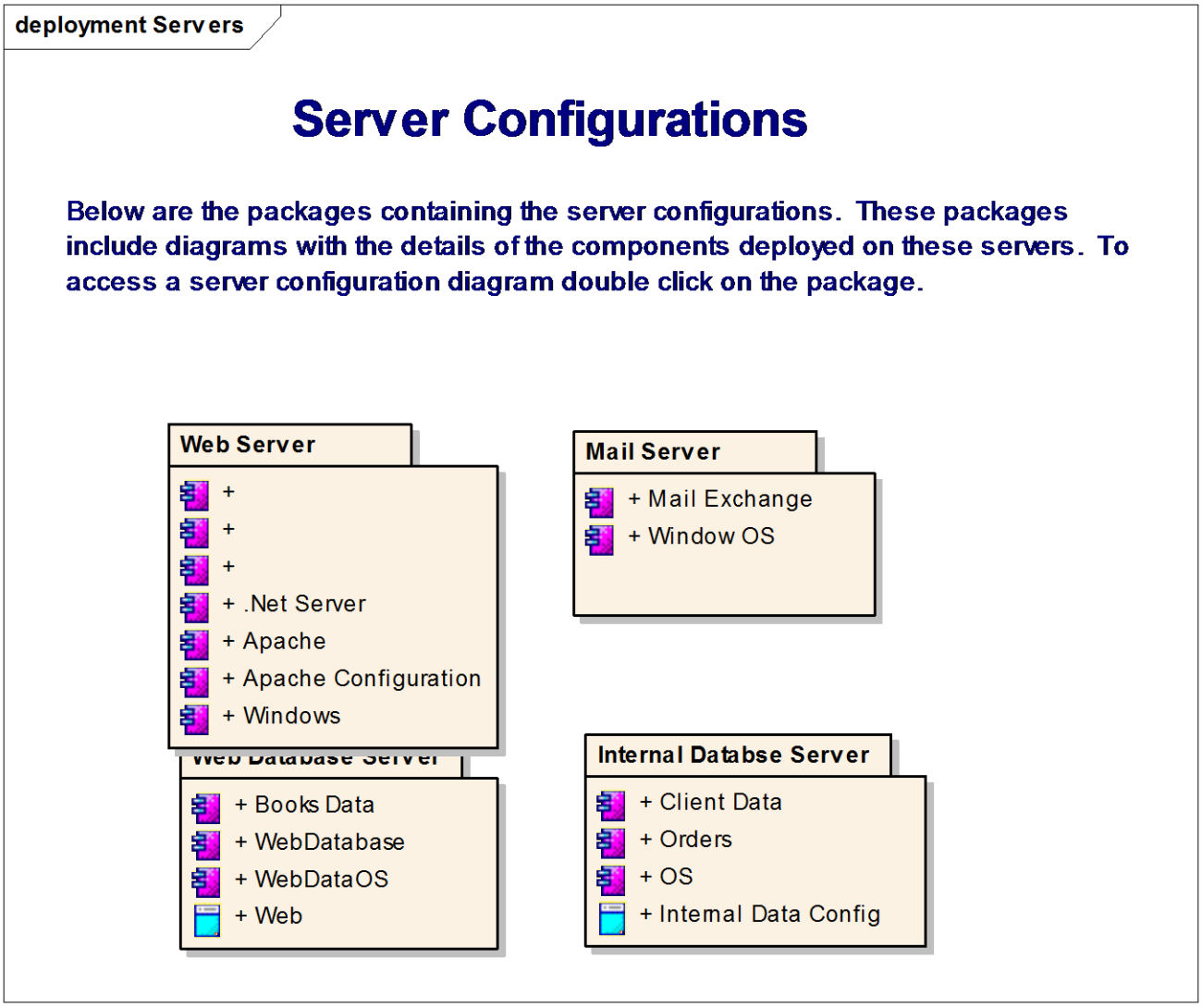


Diagram: Web Server

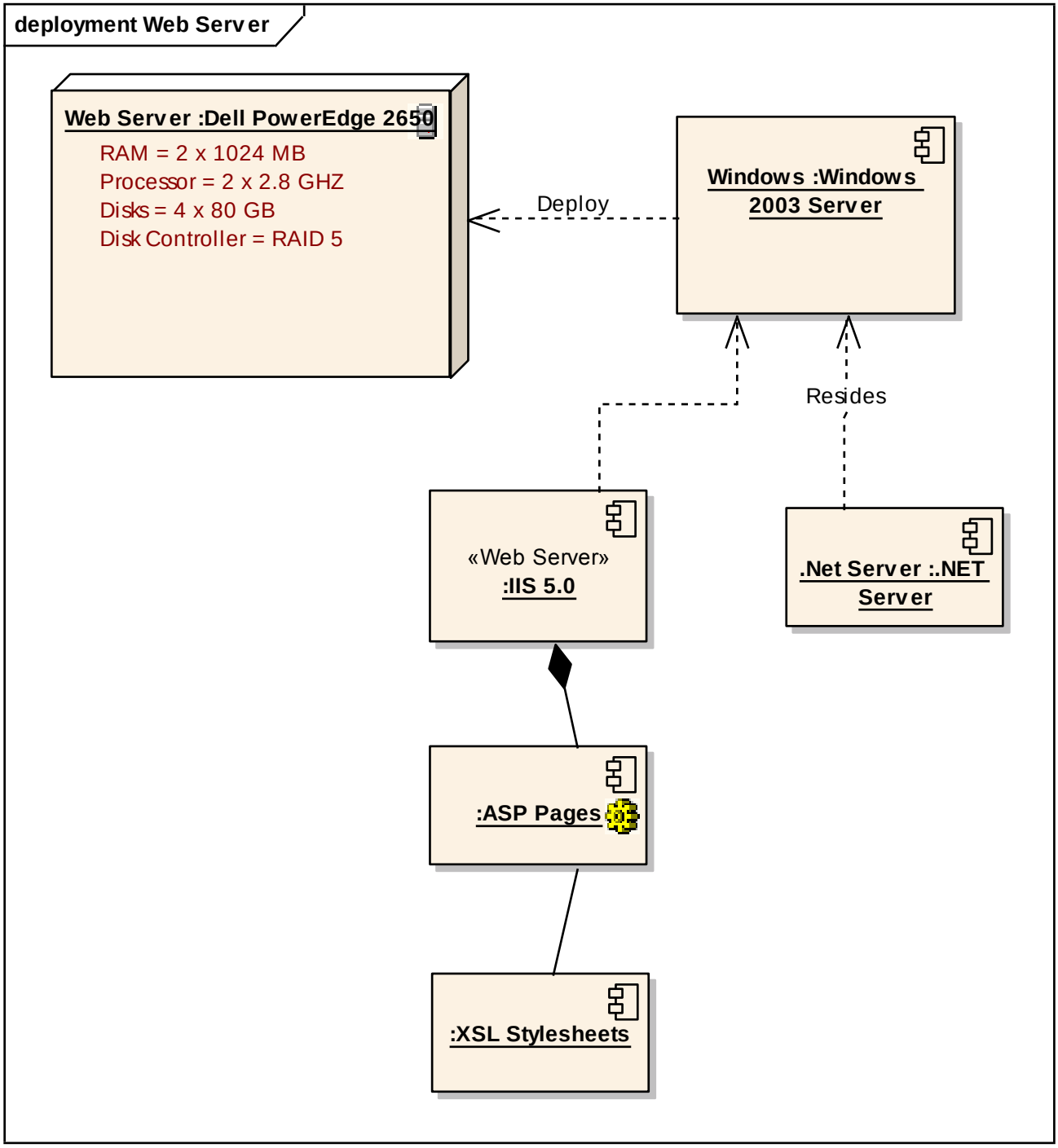


Diagram: Mail Server

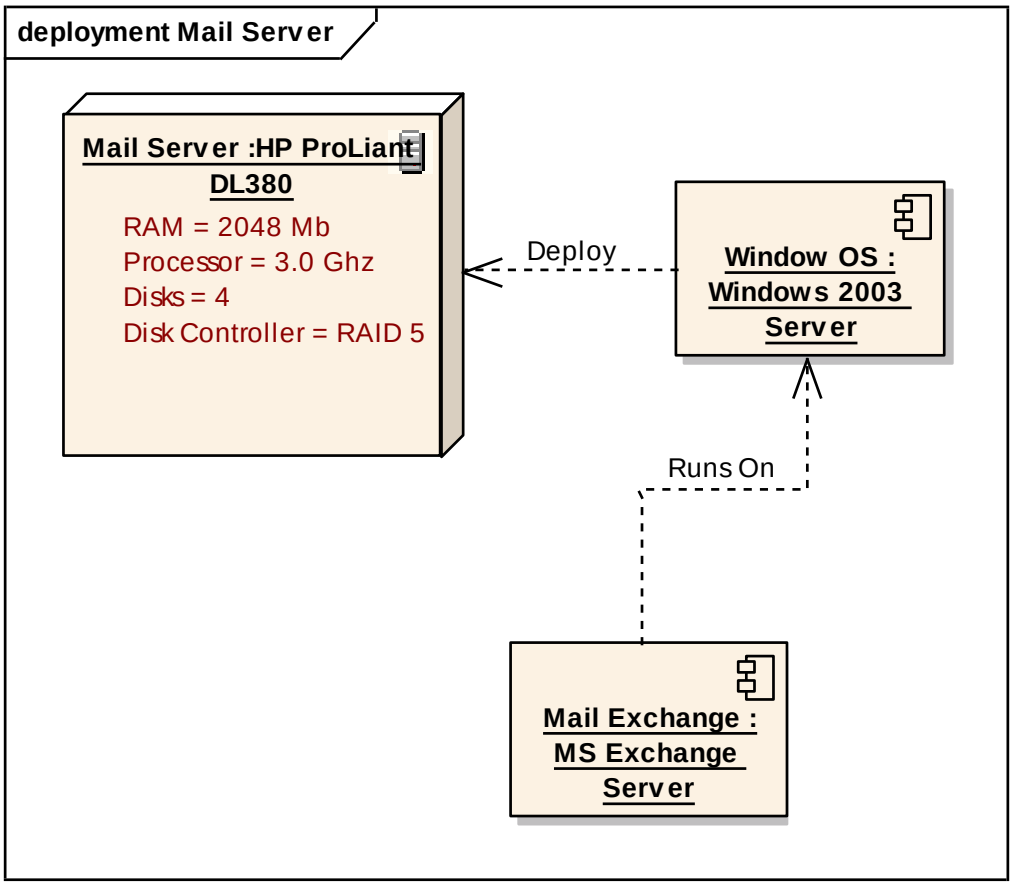


Diagram: Database Server

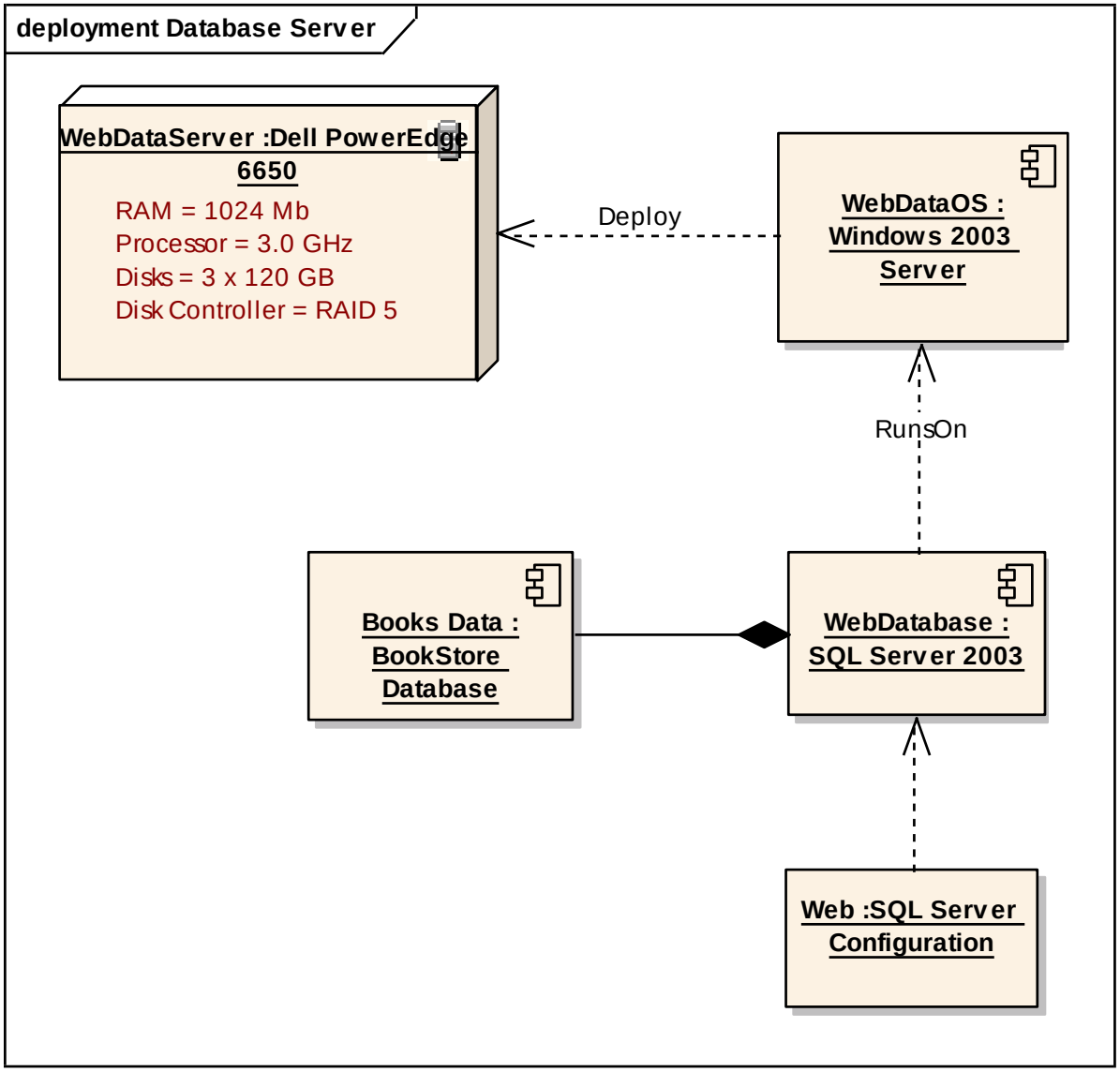


Diagram: Internal Database Server

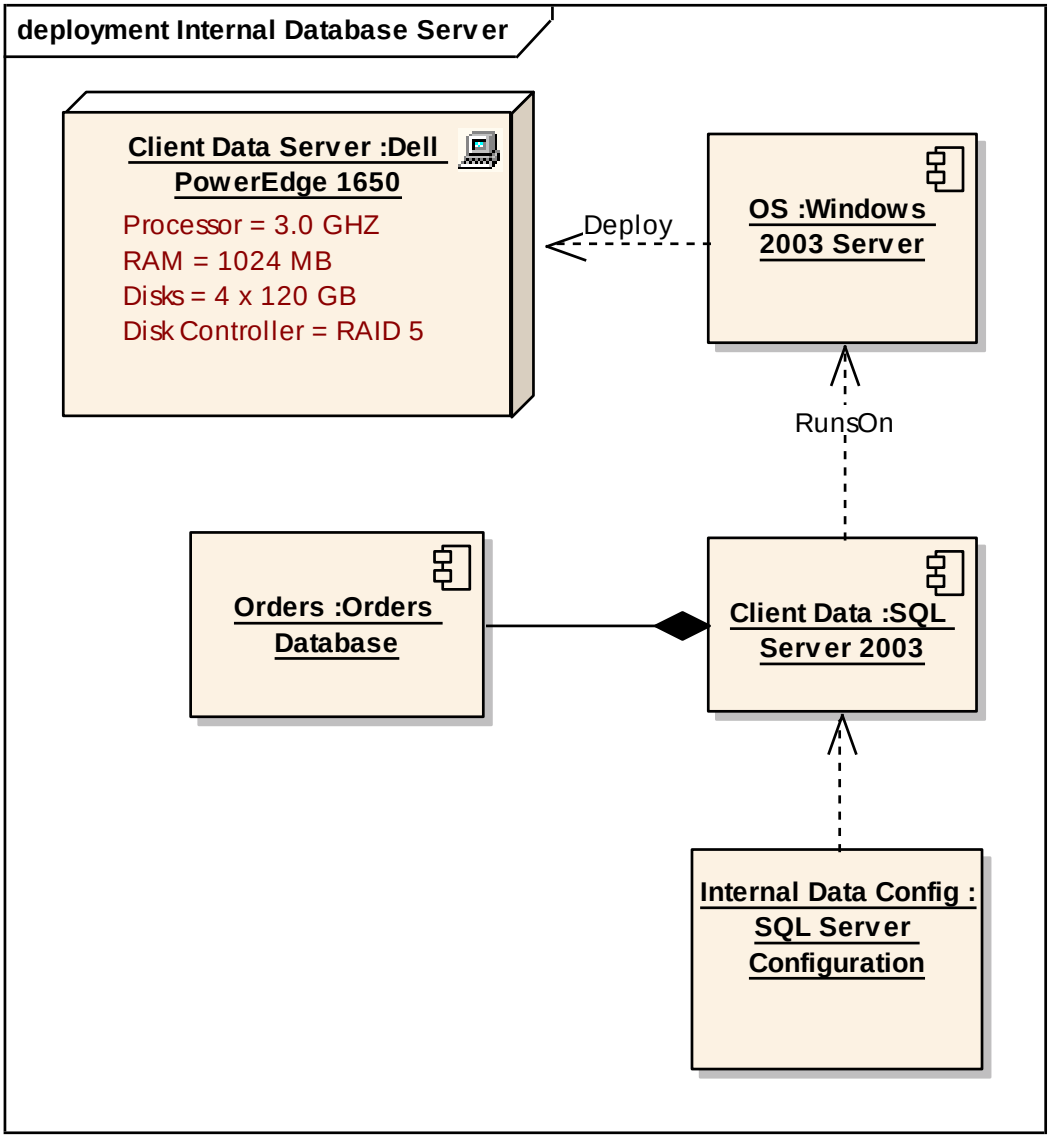


Diagram: Implementation Model (PSM)

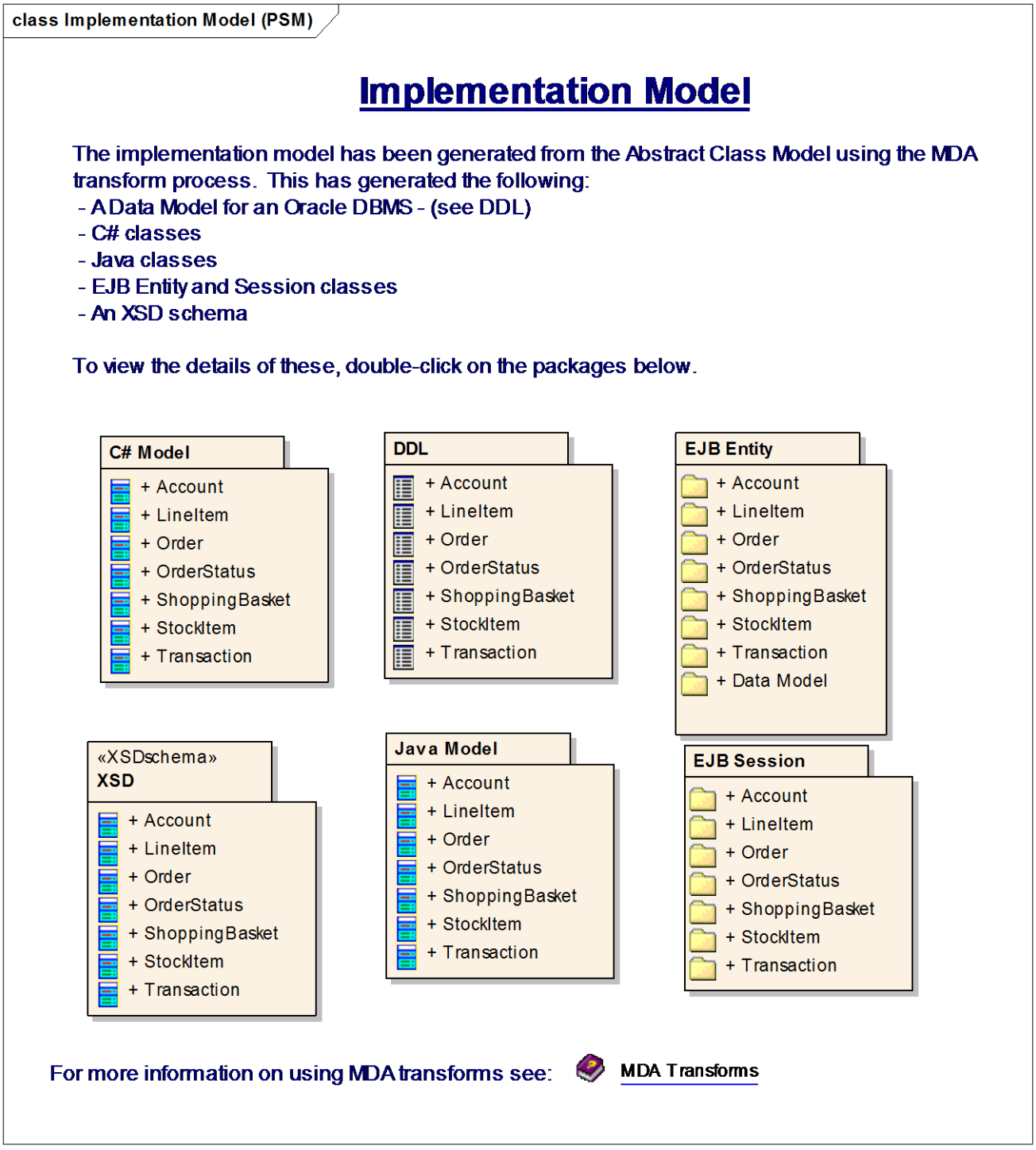


Diagram: C# Model

class C# Model

Class Model

This Class diagram represents the MDA transform generated from the Abstract Class model PIM. For more information on MDA transforms see:

MDA Transforms

This Class model can be forward engineered to the C# code. For more information on processes of code generation, reverse engineering of source code and synchronization between the source code and model - see:

Code Engineering

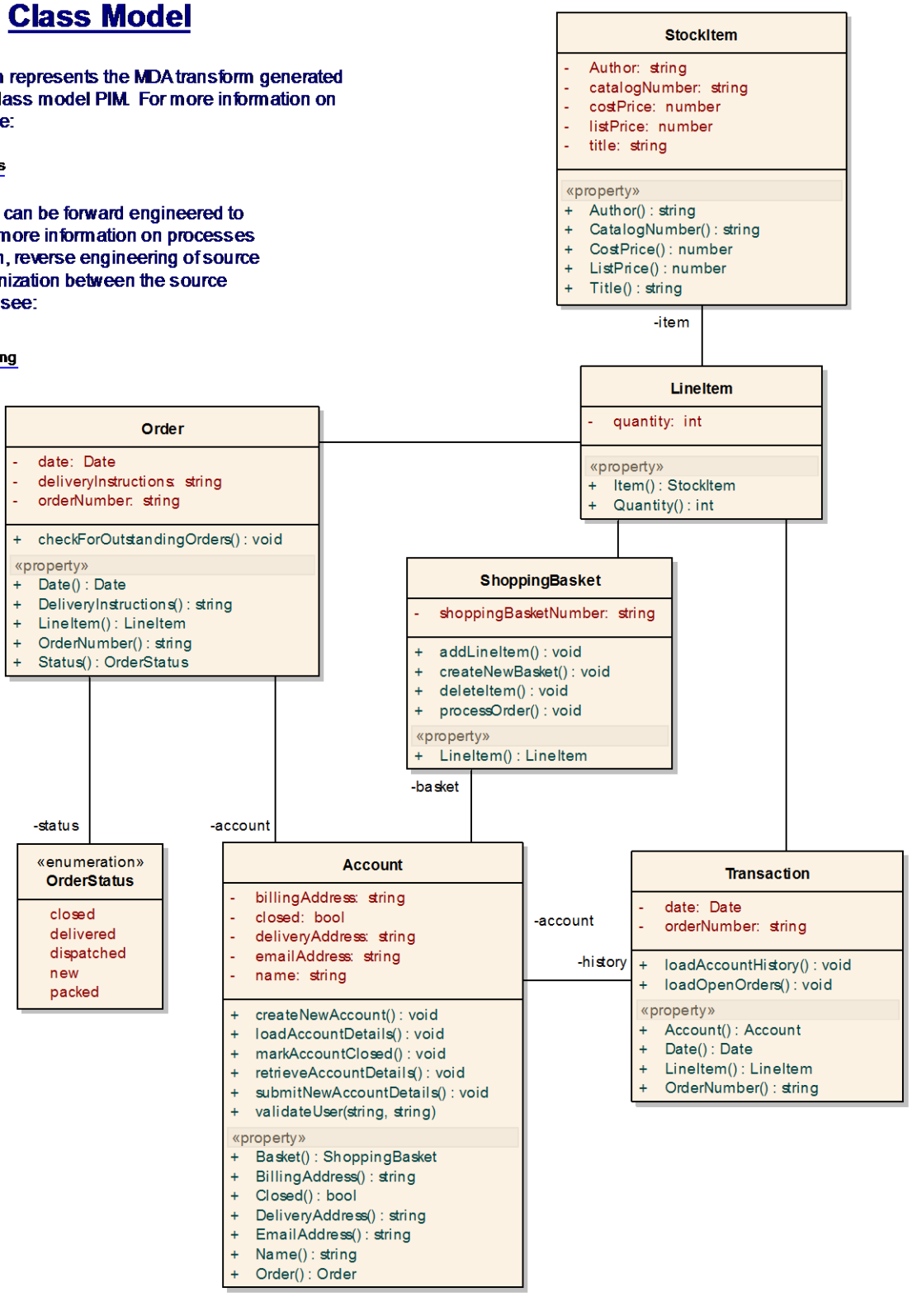


Diagram: C# Model - No Attributes

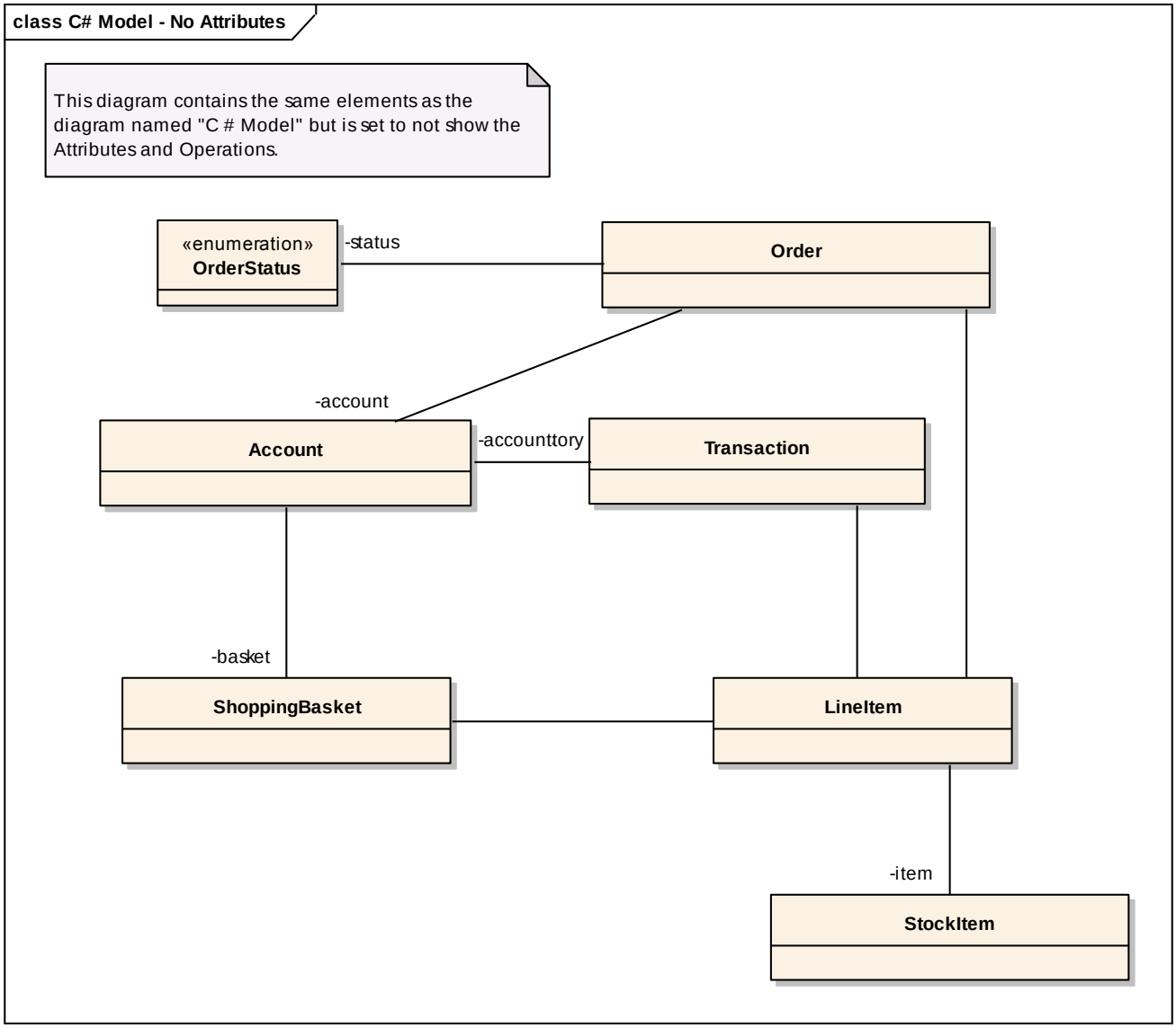


Diagram: DDL

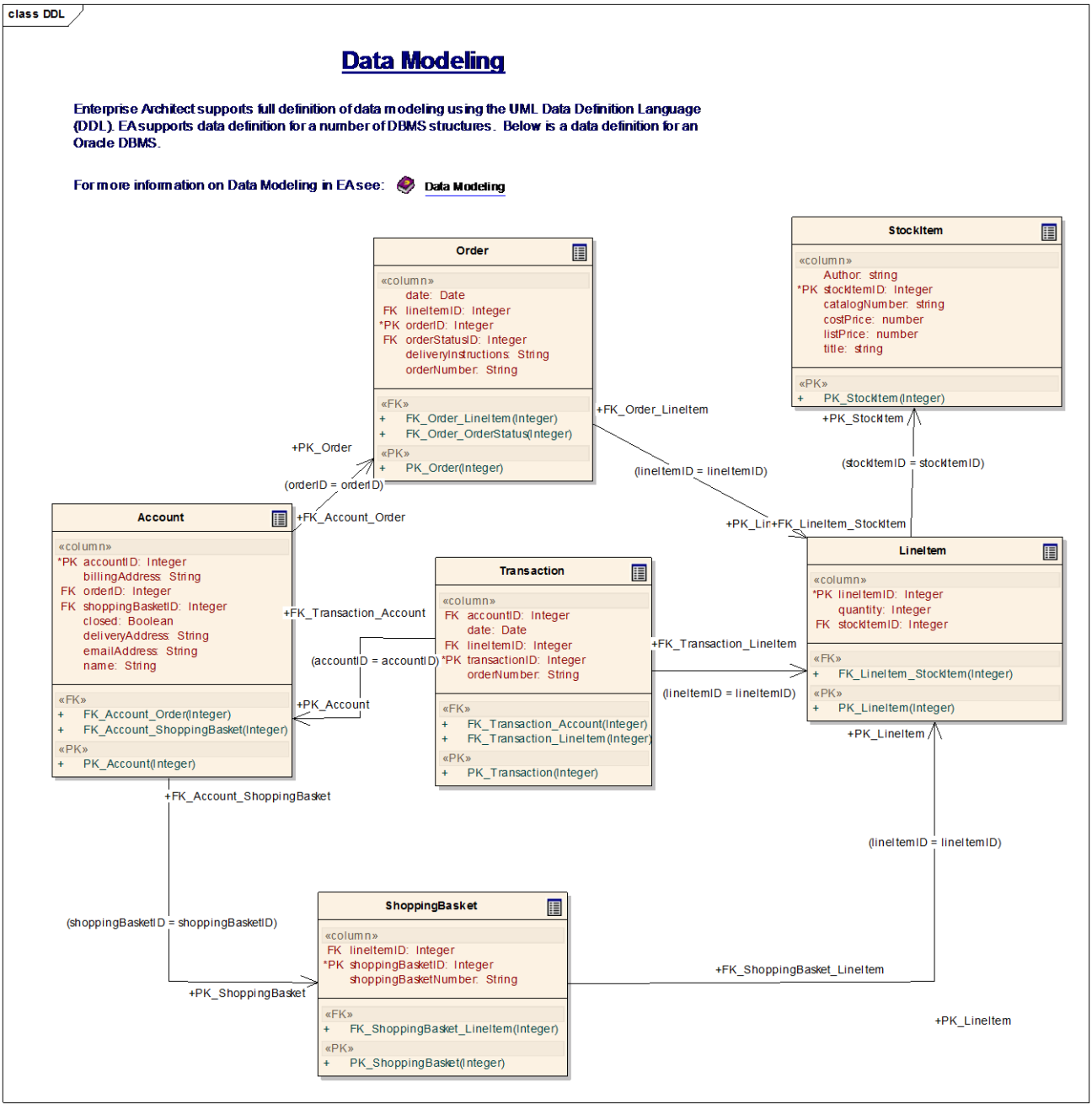

class DDL
Data Modeling
Enterprise Architect supports full definition of data modeling using the UML Data Definition Language (DDL). EA supports data definition for a number of DBMS structures. Below is a data definition for an Oracle DBMS.
For more information on Data Modeling in EA see:
Data Modeling
Order
«column»
date: Date
FK lineItemID: Integer
*PK orderID: Integer
FK orderStatusID: Integer
deliveryInstructions: String
orderNumber: String
«FK»
+ FK_Order_LineItem(Integer)
+ FK_Order_OrderStatus(Integer)
«PK»
+ PK_Order(Integer)
StockItem
«column»
Author: string
*PK stockItemID: Integer
catalogNumber: string
costPrice: number
listPrice: number
title: string
«PK»
+ PK_StockItem(Integer)
Account
«column»
*PK accountID: Integer
billingAddress: String
FK orderID: Integer
FK shoppingBasketID: Integer
closed: Boolean
deliveryAddress: String
emailAddress: String
name: String
«FK»
+ FK_Account_Order(Integer)
+ FK_Account_ShoppingBasket(Integer)
«PK»
+ PK_Account(Integer)
Transaction
«column»
FK accountID: Integer
date: Date
FK lineItemID: Integer
*PK transactionID: Integer
orderNumber: String
«FK»
+ FK_Transaction_Account(Integer)
+ FK_Transaction_LineItem(Integer)
«PK»
+ PK_Transaction(Integer)
LineItem
«column»
*PK lineItemID: Integer
quantity: Integer
FK stockItemID: Integer
«FK»
+ FK_LineItem_StockItem(Integer)
«PK»
+ PK_LineItem(Integer)
ShoppingBasket
«column»
FK lineItemID: Integer
*PK shoppingBasketID: Integer
shoppingBasketNumber: String
«FK»
+ FK_ShoppingBasket_LineItem(Integer)
«PK»
+ PK_ShoppingBasket(Integer)
+FK_Order_LineItem
+PK_Order
(orderID = orderID)
+FK_Account_Order
(lineItemID = lineItemID)
+PK_Lir+FK_LineItem_StockItem
+PK_StockItem
(stockItemID = stockItemID)
+FK_Transaction_Account
(accountID = accountID)
+PK_Account
+FK_Transaction_LineItem
(lineItemID = lineItemID)
+PK_LineItem
+FK_Account_ShoppingBasket
(lineItemID = lineItemID)
(shoppingBasketID = shoppingBasketID)
+PK_ShoppingBasket
+FK_ShoppingBasket_LineItem
+PK_LineItem

Diagram: EJB Entity

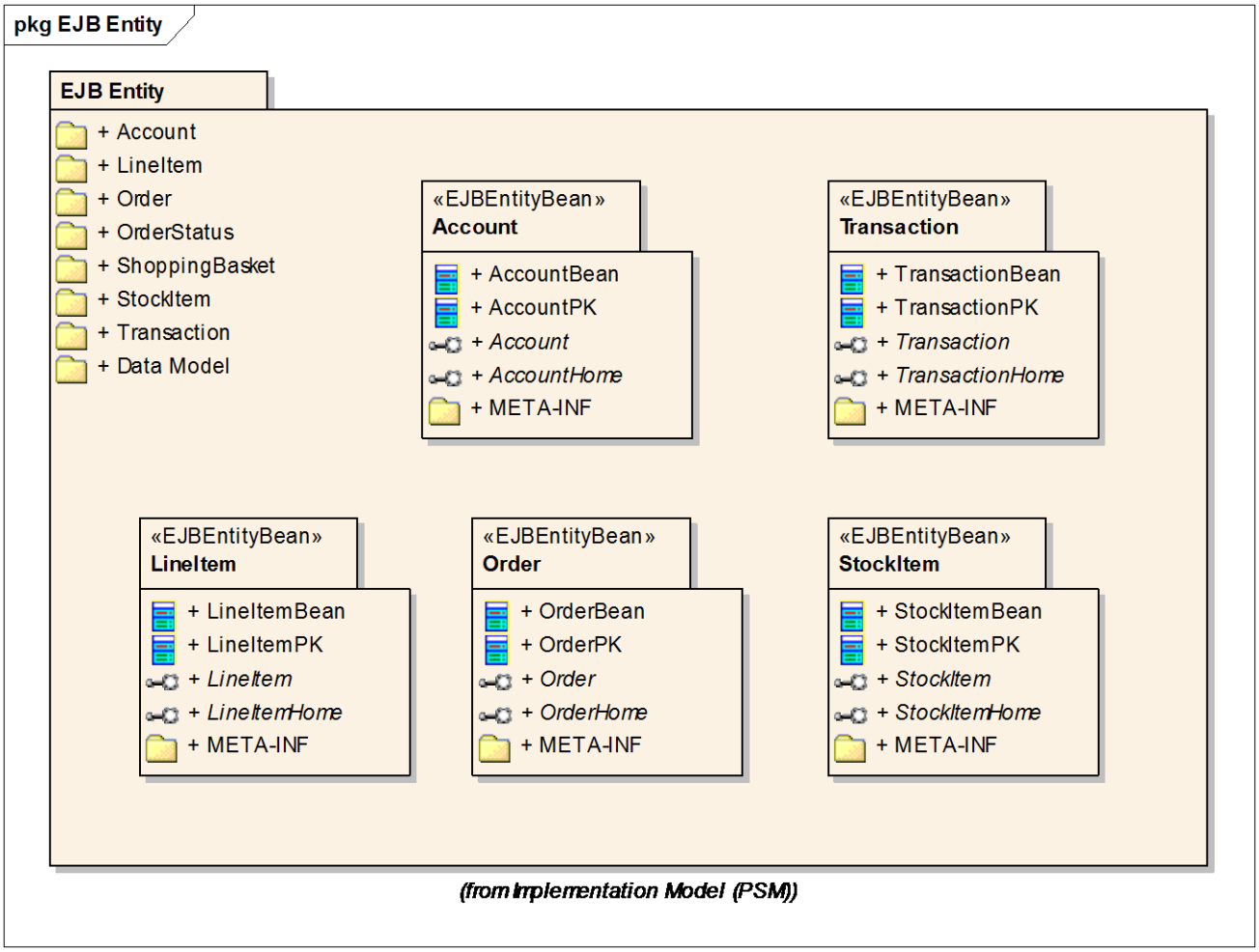


Diagram: Data Model

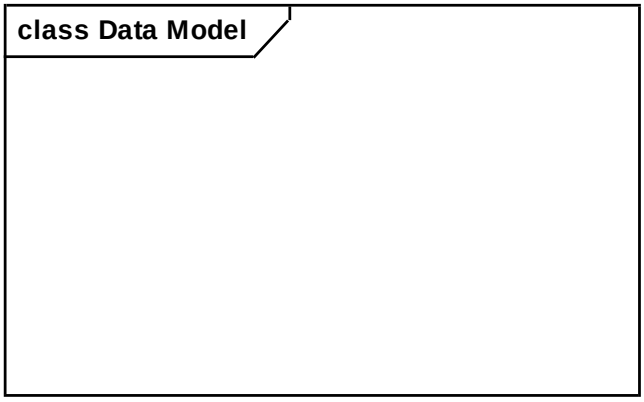


Diagram: Account

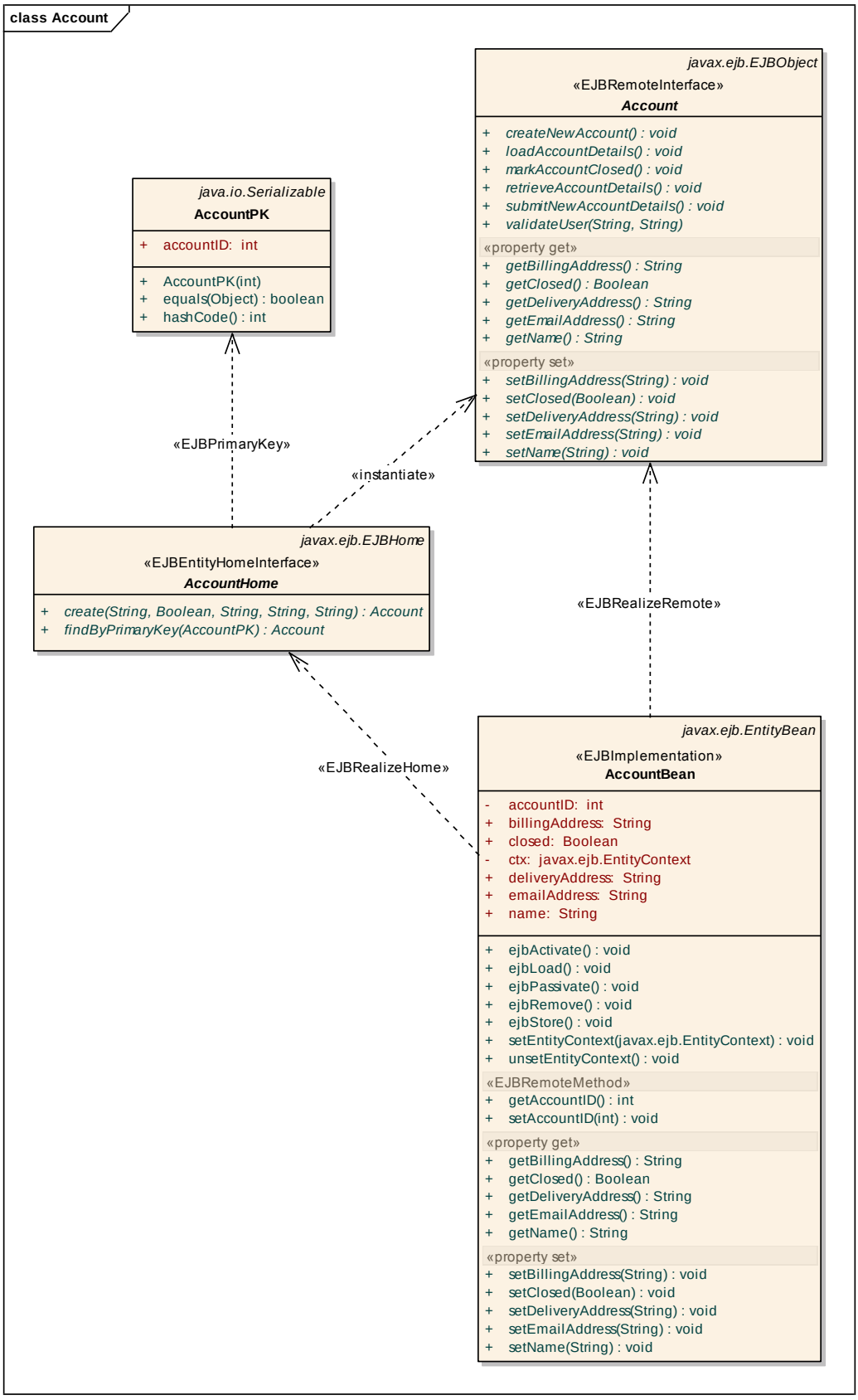

class Account
javax.ejb.EJBObject
«EJBRemoteInterface»
Account
+ createNewAccount() : void
+ loadAccountDetails() : void
+ markAccountClosed() : void
+ retrieveAccountDetails() : void
+ submitNewAccountDetails() : void
+ validateUser(String, String)
«property get»
+ getBillingAddress() : String
+ getClosed() : Boolean
+ getDeliveryAddress() : String
+ getEmailAddress() : String
+ getName() : String
«property set»
+ setBillingAddress(String) : void
+ setClosed(Boolean) : void
+ setDeliveryAddress(String) : void
+ setEmailAddress(String) : void
+ setName(String) : void
java.io.Serializable
AccountPK
+ accountID: int
+ AccountPK(int)
+ equals(Object) : boolean
+ hashCode() : int
«EJBPrimaryKey»
«instantiate»
javax.ejb.EJBHome
«EJBEntityHomeInterface»
AccountHome
+ create(String, Boolean, String, String, String) : Account
+ findByPrimaryKey(AccountPK) : Account
«EJBRealizeRemote»
«EJBRealizeHome»
javax.ejb.EntityBean
«EJBImplementation»
AccountBean
- accountID: int
+ billingAddress: String
+ closed: Boolean
- ctx: javax.ejb.EntityContext
+ deliveryAddress: String
+ emailAddress: String
+ name: String
+ ejbActivate() : void
+ ejbLoad() : void
+ ejbPassivate() : void
+ ejbRemove() : void
+ ejbStore() : void
+ setEntityContext(javax.ejb.EntityContext) : void
+ unsetEntityContext() : void
«EJBRemoteMethod»
+ getAccountID() : int
+ setAccountID(int) : void
«property get»
+ getBillingAddress() : String
+ getClosed() : Boolean
+ getDeliveryAddress() : String
+ getEmailAddress() : String
+ getName() : String
«property set»
+ setBillingAddress(String) : void
+ setClosed(Boolean) : void
+ setDeliveryAddress(String) : void
+ setEmailAddress(String) : void
+ setName(String) : void

Diagram: META-INF

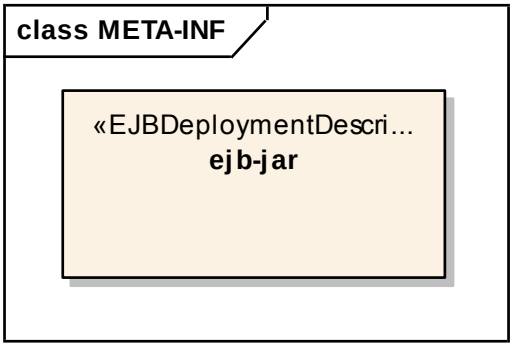


Diagram: LineItem

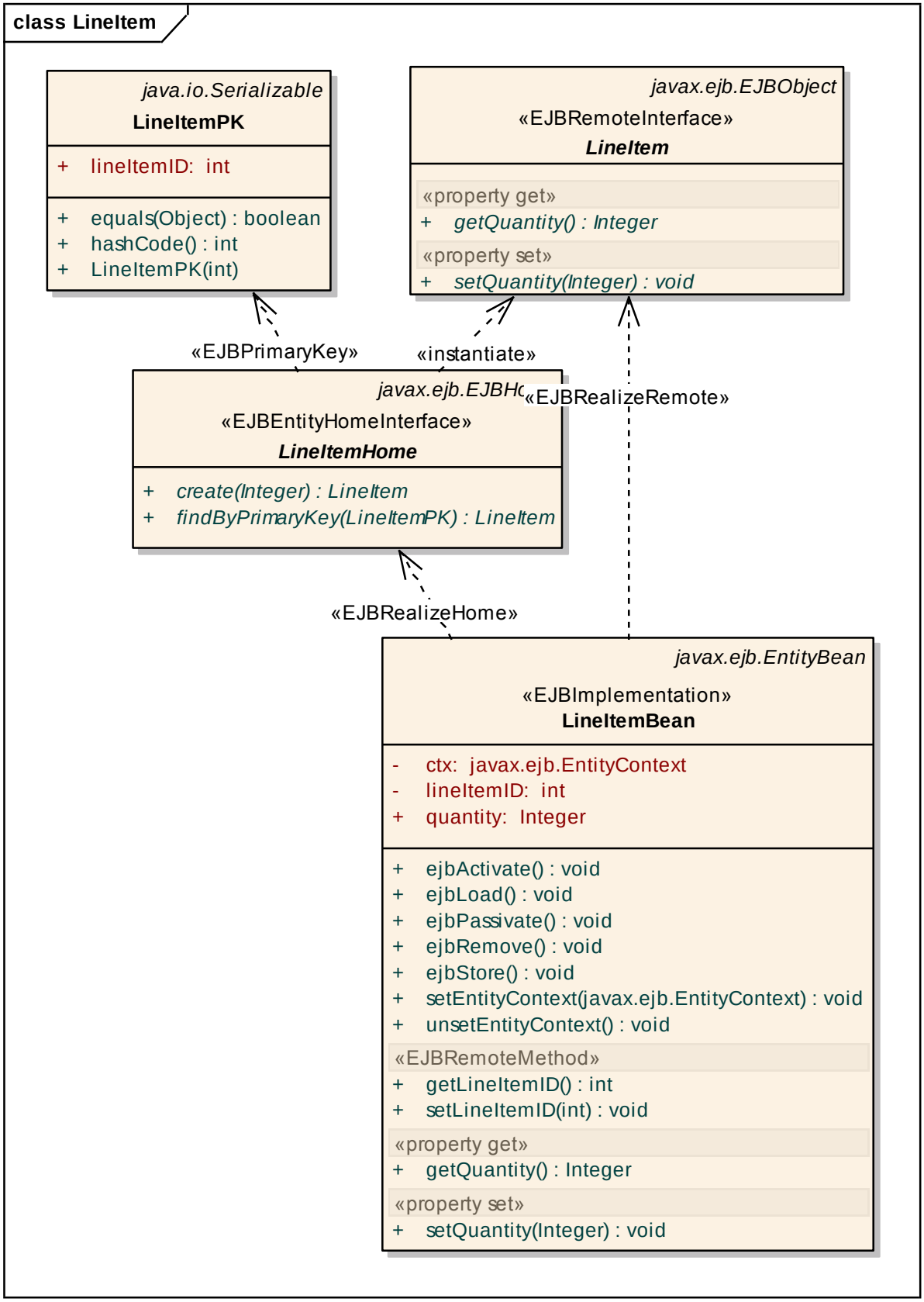

class LineItem
java.io.Serializable
LineItemPK
+ lineItemID: int
+ equals(Object) : boolean
+ hashCode() : int
+ LineItemPK(int)
javax.ejb.EJBObject
«EJBRemoteInterface»
LineItem
«property get»
+ getQuantity() : Integer
«property set»
+ setQuantity(Integer) : void
«EJBPrimaryKey»
«instantiate»
«EJBRealizeRemote»
javax.ejb.EJBHo
«EJBEntityHomeInterface»
LineItemHome
+ create(Integer) : LineItem
+ findByPrimaryKey(LineItemPK) : LineItem
«EJBRealizeHome»
javax.ejb.EntityBean
«EJBImplementation»
LineItemBean
- ctx: javax.ejb.EntityContext
- lineItemID: int
+ quantity: Integer
+ ejbActivate() : void
+ ejbLoad() : void
+ ejbPassivate() : void
+ ejbRemove() : void
+ ejbStore() : void
+ setEntityContext(javax.ejb.EntityContext) : void
+ unsetEntityContext() : void
«EJBRemoteMethod»
+ getLineItemID() : int
+ setLineItemID(int) : void
«property get»
+ getQuantity() : Integer
«property set»
+ setQuantity(Integer) : void

Diagram: META-INF

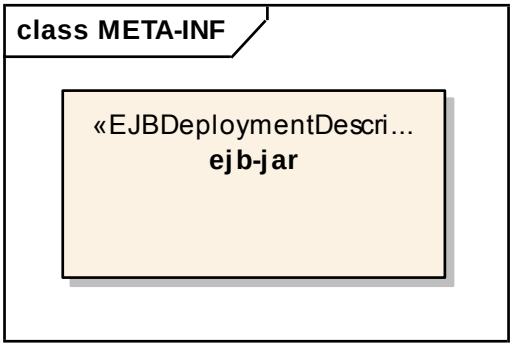


Diagram: Order

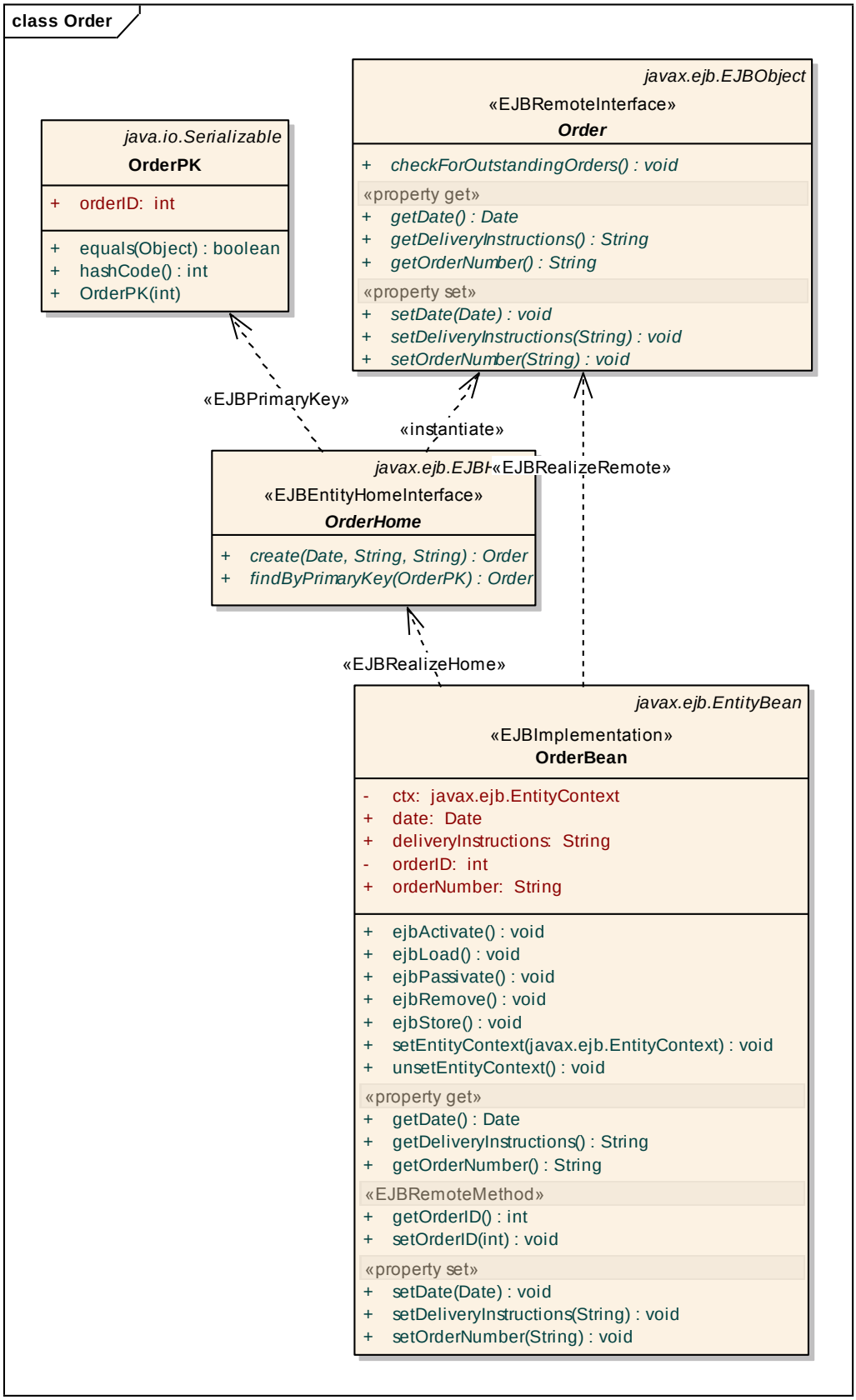

class Order
javax.ejb.EJBObject
«EJBRemoteInterface»
Order
+ checkForOutstandingOrders() : void
«property get»
+ getDate() : Date
+ getDeliveryInstructions() : String
+ getOrderNumber() : String
«property set»
+ setDate(Date) : void
+ setDeliveryInstructions(String) : void
+ setOrderNumber(String) : void
java.io.Serializable
OrderPK
+ orderID: int
+ equals(Object) : boolean
+ hashCode() : int
+ OrderPK(int)
«EJBPrimaryKey»
«instantiate»
«EJBRealizeRemote»
javax.ejb.EJBH
«EJBEntityHomeInterface»
OrderHome
+ create(Date, String, String) : Order
+ findByPrimaryKey(OrderPK) : Order
«EJBRealizeHome»
javax.ejb.EntityBean
«EJBImplementation»
OrderBean
- ctx: javax.ejb.EntityContext
+ date: Date
+ deliveryInstructions: String
- orderID: int
+ orderNumber: String
+ ejbActivate() : void
+ ejbLoad() : void
+ ejbPassivate() : void
+ ejbRemove() : void
+ ejbStore() : void
+ setEntityContext(javax.ejb.EntityContext) : void
+ unsetEntityContext() : void
«property get»
+ getDate() : Date
+ getDeliveryInstructions() : String
+ getOrderNumber() : String
«EJBRemoteMethod»
+ getOrderID() : int
+ setOrderID(int) : void
«property set»
+ setDate(Date) : void
+ setDeliveryInstructions(String) : void
+ setOrderNumber(String) : void

Diagram: META-INF

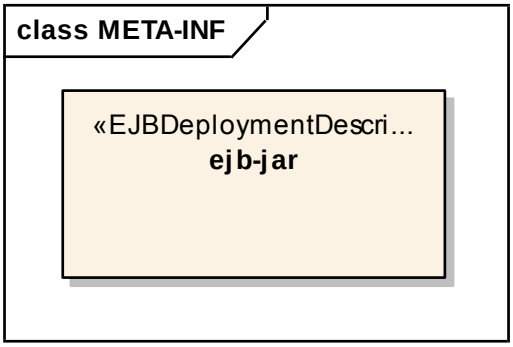


Diagram: StockItem

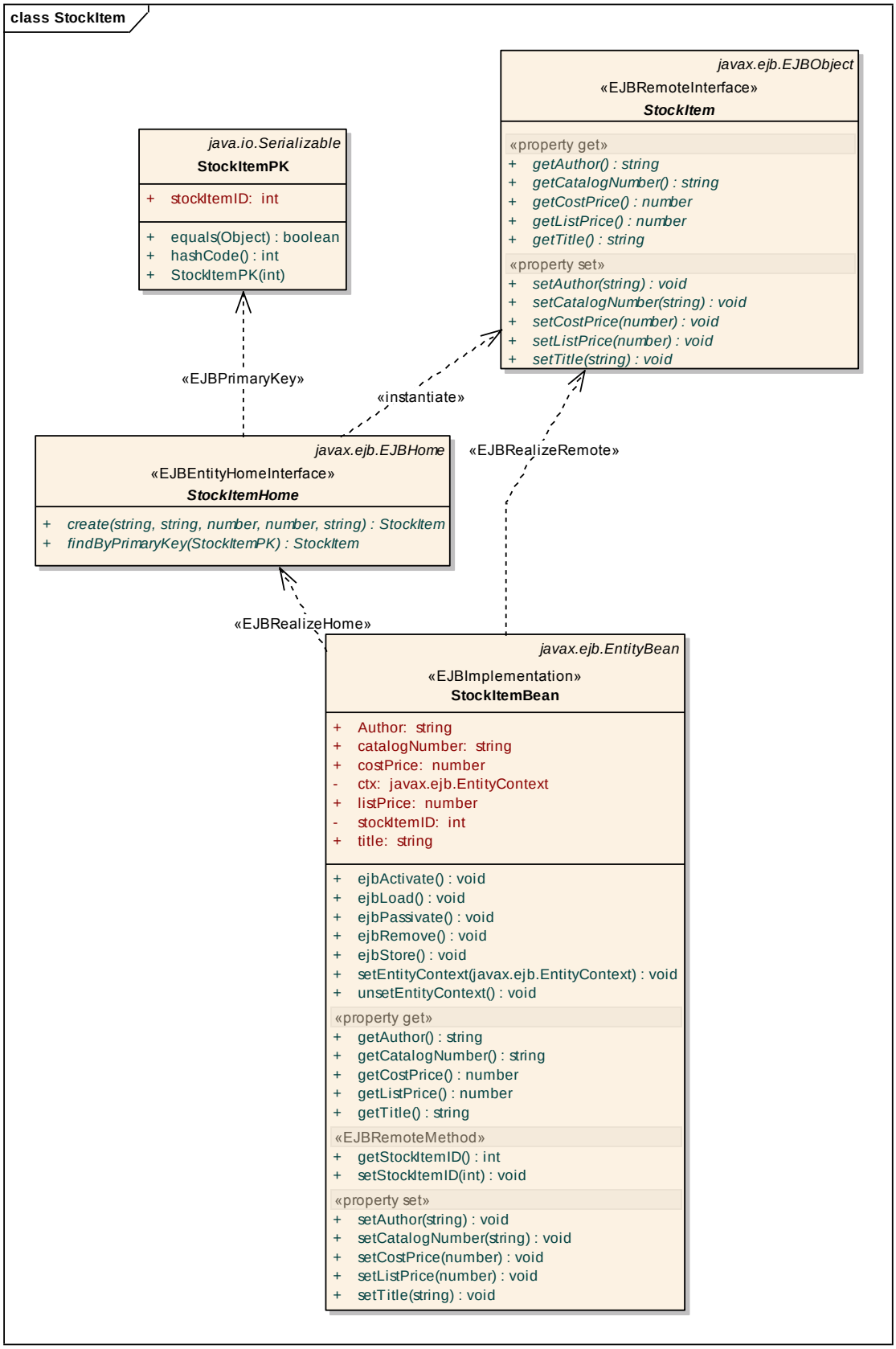

class StockItem
javax.ejb.EJBObject
«EJBRemoteInterface»
StockItem
«property get»
+ getAuthor() : string
+ getCatalogNumber() : string
+ getCostPrice() : number
+ getListPrice() : number
+ getTitle() : string
«property set»
+ setAuthor(string) : void
+ setCatalogNumber(string) : void
+ setCostPrice(number) : void
+ setListPrice(number) : void
+ setTitle(string) : void
java.io.Serializable
StockItemPK
+ stockItemID: int
+ equals(Object) : boolean
+ hashCode() : int
+ StockItemPK(int)
«EJBPrimaryKey»
«instantiate»
«EJBRealizeRemote»
javax.ejb.EJBHome
«EJBEntityHomeInterface»
StockItemHome
+ create(string, string, number, number, string) : StockItem
+ findByPrimaryKey(StockItemPK) : StockItem
«EJBRealizeHome»
javax.ejb.EntityBean
«EJBImplementation»
StockItemBean
+ Author: string
+ catalogNumber: string
+ costPrice: number
- ctx: javax.ejb.EntityContext
+ listPrice: number
- stockItemID: int
+ title: string
+ ejbActivate() : void
+ ejbLoad() : void
+ ejbPassivate() : void
+ ejbRemove() : void
+ ejbStore() : void
+ setEntityContext(javax.ejb.EntityContext) : void
+ unsetEntityContext() : void
«property get»
+ getAuthor() : string
+ getCatalogNumber() : string
+ getCostPrice() : number
+ getListPrice() : number
+ getTitle() : string
«EJBRemoteMethod»
+ getStockItemID() : int
+ setStockItemID(int) : void
«property set»
+ setAuthor(string) : void
+ setCatalogNumber(string) : void
+ setCostPrice(number) : void
+ setListPrice(number) : void
+ setTitle(string) : void

Diagram: META-INF

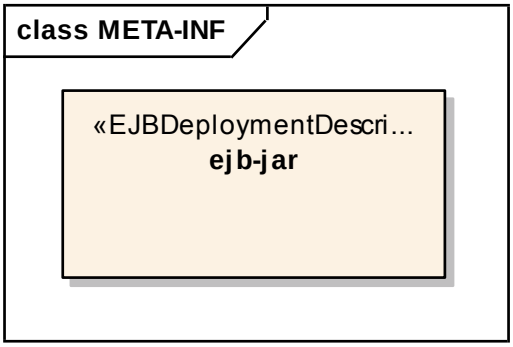


Diagram: Transaction

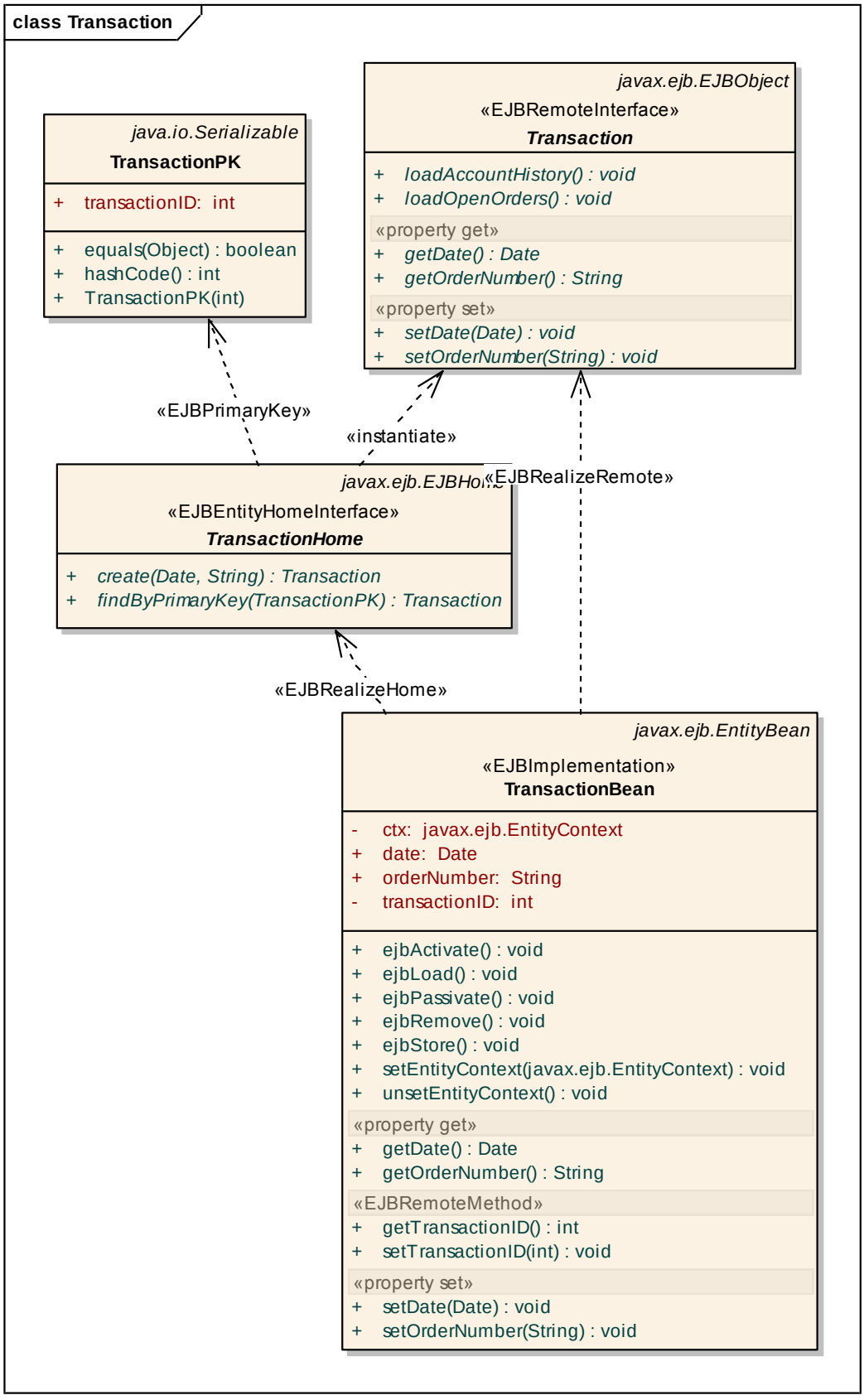

class Transaction
java.io.Serializable
TransactionPK
+ transactionID: int
+ equals(Object) : boolean
+ hashCode() : int
+ TransactionPK(int)
javax.ejb.EJBObject
«EJBRemoteInterface»
Transaction
+ loadAccountHistory() : void
+ loadOpenOrders() : void
«property get»
+ getDate() : Date
+ getOrderNumber() : String
«property set»
+ setDate(Date) : void
+ setOrderNumber(String) : void
«EJBPrimaryKey»
«instantiate»
«EJBRealizeRemote»
javax.ejb.EJBHome
«EJBEntityHomeInterface»
TransactionHome
+ create(Date, String) : Transaction
+ findByPrimaryKey(TransactionPK) : Transaction
«EJBRealizeHome»
javax.ejb.EntityBean
«EJBImplementation»
TransactionBean
- ctx: javax.ejb.EntityContext
+ date: Date
+ orderNumber: String
- transactionID: int
+ ejbActivate() : void
+ ejbLoad() : void
+ ejbPassivate() : void
+ ejbRemove() : void
+ ejbStore() : void
+ setEntityContext(javax.ejb.EntityContext) : void
+ unsetEntityContext() : void
«property get»
+ getDate() : Date
+ getOrderNumber() : String
«EJBRemoteMethod»
+ getTransactionID() : int
+ setTransactionID(int) : void
«property set»
+ setDate(Date) : void
+ setOrderNumber(String) : void

Diagram: META-INF

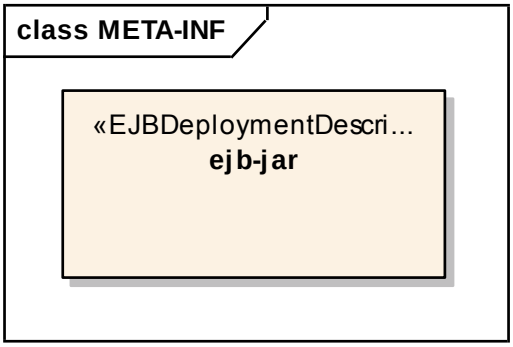


Diagram: OrderStatus

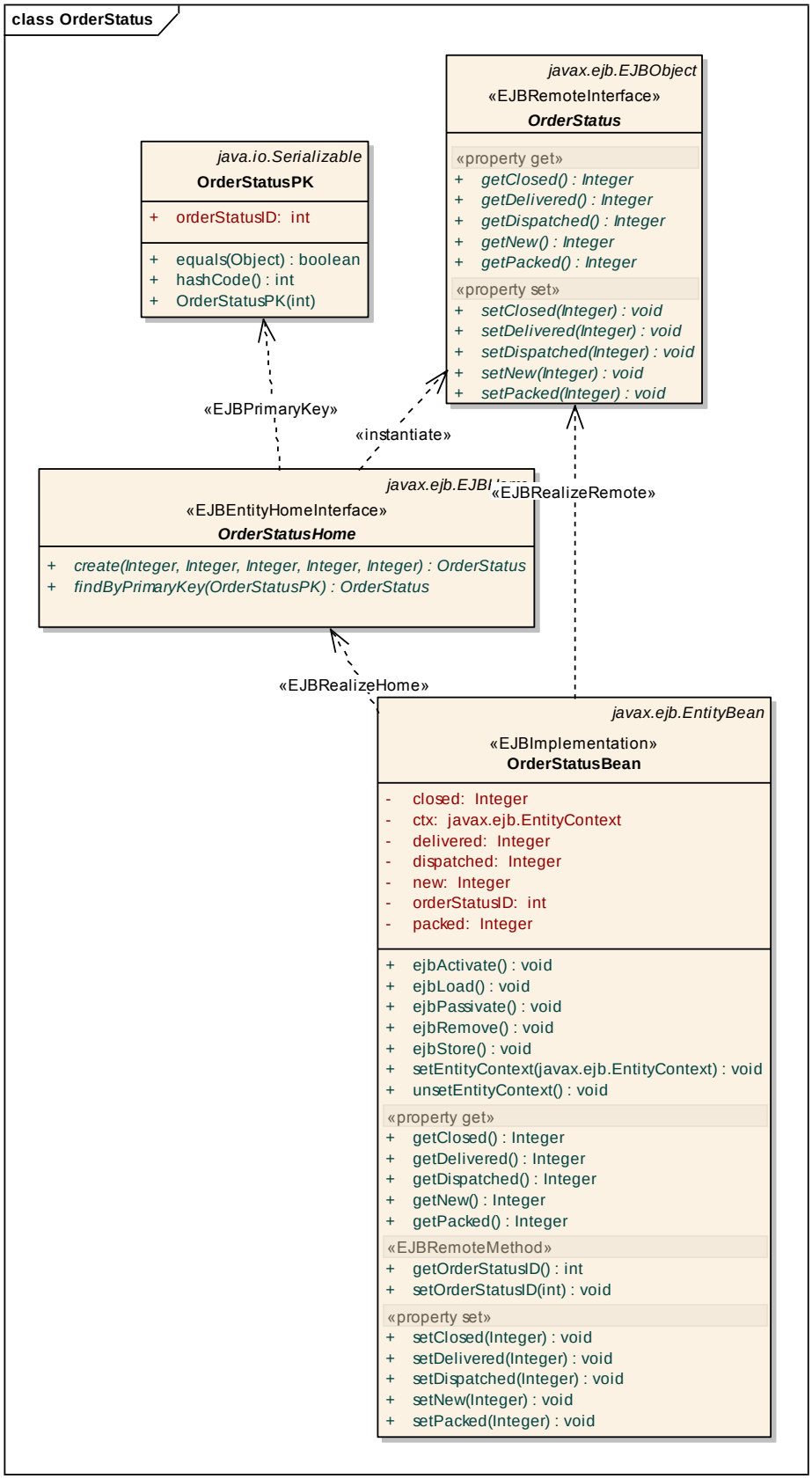

class OrderStatus
javax.ejb.EJBObject
«EJBRemoteInterface»
OrderStatus
«property get»
+ getClosed() : Integer
+ getDelivered() : Integer
+ getDispatched() : Integer
+ getNew() : Integer
+ getPacked() : Integer
«property set»
+ setClosed(Integer) : void
+ setDelivered(Integer) : void
+ setDispatched(Integer) : void
+ setNew(Integer) : void
+ setPacked(Integer) : void
java.io.Serializable
OrderStatusPK
+ orderStatusID: int
+ equals(Object) : boolean
+ hashCode() : int
+ OrderStatusPK(int)
«EJBPrimaryKey»
«instantiate»
«EJBRealizeRemote»
«EJBEntityHomeInterface»
OrderStatusHome
+ create(Integer, Integer, Integer, Integer, Integer) : OrderStatus
+ findByPrimaryKey(OrderStatusPK) : OrderStatus
«EJBRealizeHome»
javax.ejb.EntityBean
«EJBImplementation»
OrderStatusBean
- closed: Integer
- ctx: javax.ejb.EntityContext
- delivered: Integer
- dispatched: Integer
- new: Integer
- orderStatusID: int
- packed: Integer
+ ejbActivate() : void
+ ejbLoad() : void
+ ejbPassivate() : void
+ ejbRemove() : void
+ ejbStore() : void
+ setEntityContext(javax.ejb.EntityContext) : void
+ unsetEntityContext() : void
«property get»
+ getClosed() : Integer
+ getDelivered() : Integer
+ getDispatched() : Integer
+ getNew() : Integer
+ getPacked() : Integer
«EJBRemoteMethod»
+ getOrderStatusID() : int
+ setOrderStatusID(int) : void
«property set»
+ setClosed(Integer) : void
+ setDelivered(Integer) : void
+ setDispatched(Integer) : void
+ setNew(Integer) : void
+ setPacked(Integer) : void

Diagram: META-INF

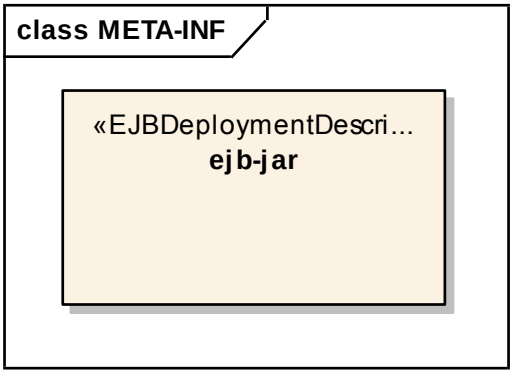


Diagram: ShoppingBasket

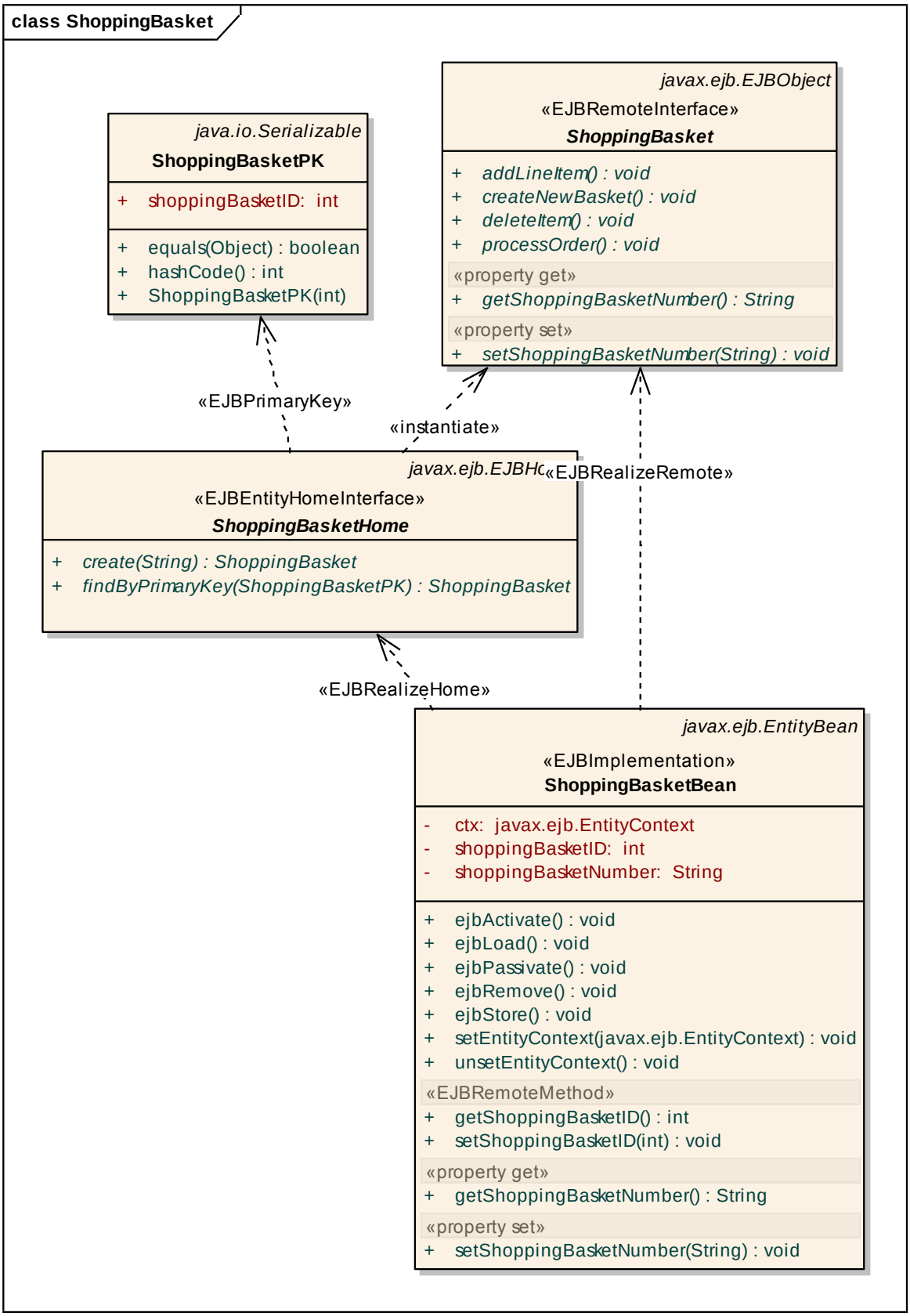

class ShoppingBasket
java.io.Serializable
ShoppingBasketPK
+ shoppingBasketID: int
+ equals(Object) : boolean
+ hashCode() : int
+ ShoppingBasketPK(int)
javax.ejb.EJBObject
«EJBRemoteInterface»
ShoppingBasket
+ addLineItem() : void
+ createNewBasket() : void
+ deleteItem() : void
+ processOrder() : void
«property get»
+ getShoppingBasketNumber() : String
«property set»
+ setShoppingBasketNumber(String) : void
«EJBPrimaryKey»
«instantiate»
«EJBRealizeRemote»
javax.ejb.EJBHo
«EJBEntityHomeInterface»
ShoppingBasketHome
+ create(String) : ShoppingBasket
+ findByPrimaryKey(ShoppingBasketPK) : ShoppingBasket
«EJBRealizeHome»
javax.ejb.EntityBean
«EJBImplementation»
ShoppingBasketBean
- ctx: javax.ejb.EntityContext
- shoppingBasketID: int
- shoppingBasketNumber: String
+ ejbActivate() : void
+ ejbLoad() : void
+ ejbPassivate() : void
+ ejbRemove() : void
+ ejbStore() : void
+ setEntityContext(javax.ejb.EntityContext) : void
+ unsetEntityContext() : void
«EJBRemoteMethod»
+ getShoppingBasketID() : int
+ setShoppingBasketID(int) : void
«property get»
+ getShoppingBasketNumber() : String
«property set»
+ setShoppingBasketNumber(String) : void

Diagram: META-INF

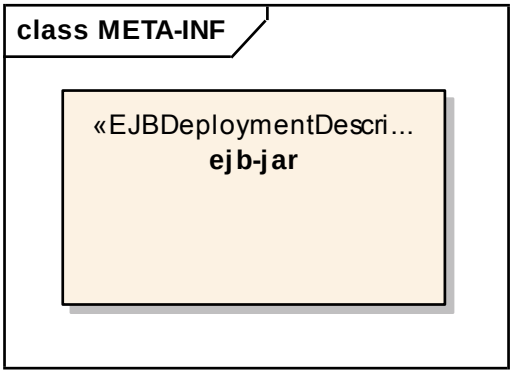


Diagram: EJB Session

Diagram: Account

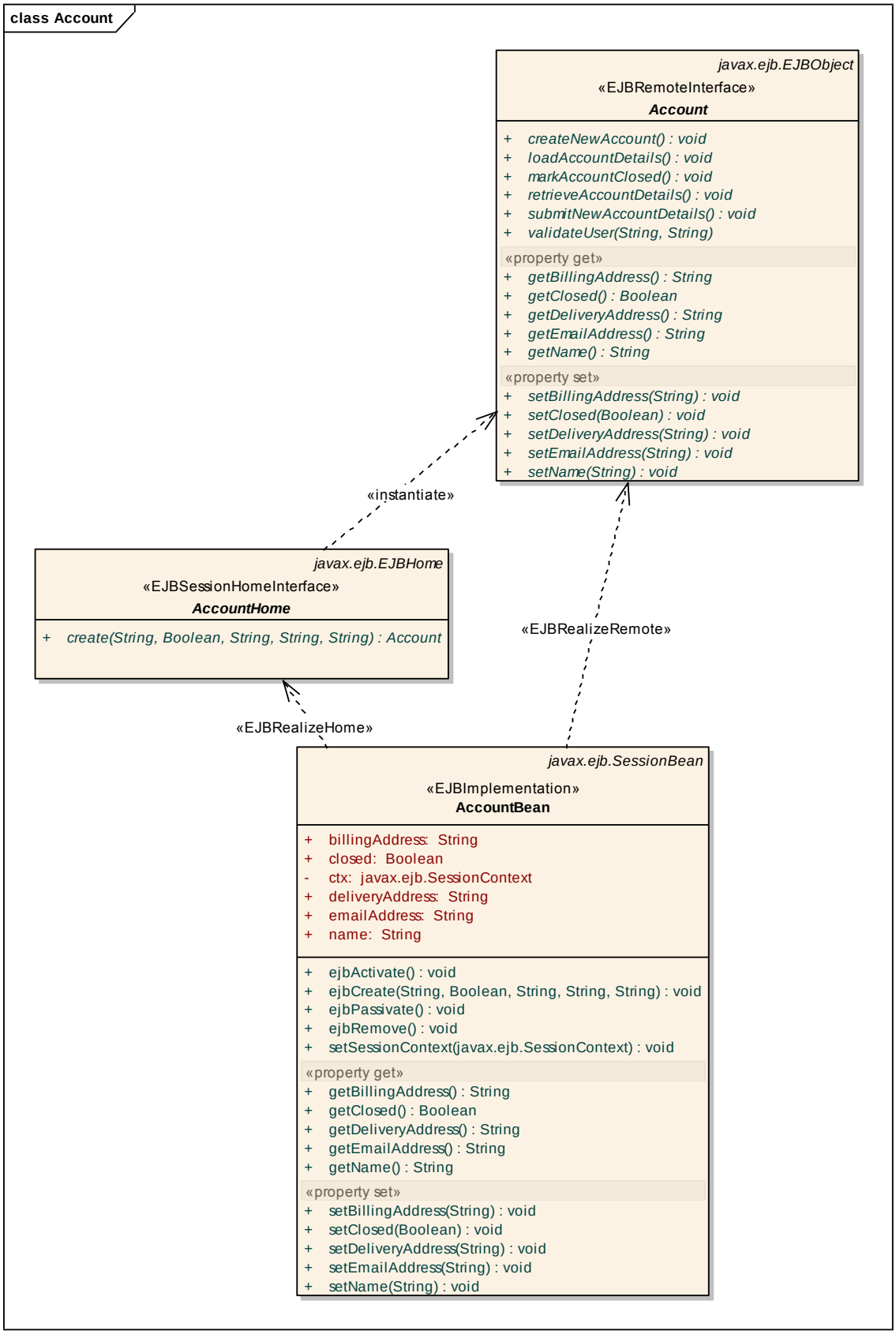

class Account
javax.ejb.EJBObject
«EJBRemoteInterface»
Account
+ createNewAccount() : void
+ loadAccountDetails() : void
+ markAccountClosed() : void
+ retrieveAccountDetails() : void
+ submitNewAccountDetails() : void
+ validateUser(String, String)
«property get»
+ getBillingAddress() : String
+ getClosed() : Boolean
+ getDeliveryAddress() : String
+ getEmailAddress() : String
+ getName() : String
«property set»
+ setBillingAddress(String) : void
+ setClosed(Boolean) : void
+ setDeliveryAddress(String) : void
+ setEmailAddress(String) : void
+ setName(String) : void
«instantiate»
javax.ejb.EJBHome
«EJBSessionHomeInterface»
AccountHome
+ create(String, Boolean, String, String, String) : Account
«EJBRealizeRemote»
«EJBRealizeHome»
javax.ejb.SessionBean
«EJBImplementation»
AccountBean
+ billingAddress: String
+ closed: Boolean
- ctx: javax.ejb.SessionContext
+ deliveryAddress: String
+ emailAddress: String
+ name: String
+ ejbActivate() : void
+ ejbCreate(String, Boolean, String, String, String) : void
+ ejbPassivate() : void
+ ejbRemove() : void
+ setSessionContext(javax.ejb.SessionContext) : void
«property get»
+ getBillingAddress() : String
+ getClosed() : Boolean
+ getDeliveryAddress() : String
+ getEmailAddress() : String
+ getName() : String
«property set»
+ setBillingAddress(String) : void
+ setClosed(Boolean) : void
+ setDeliveryAddress(String) : void
+ setEmailAddress(String) : void
+ setName(String) : void

Diagram: META-INF

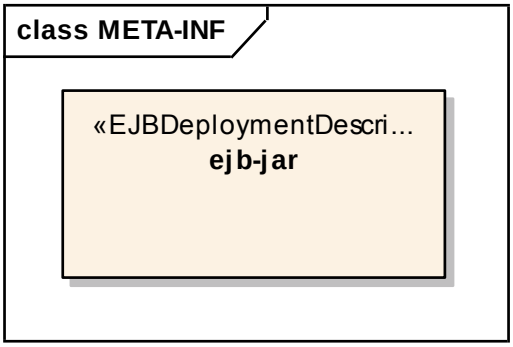


Diagram: LineItem

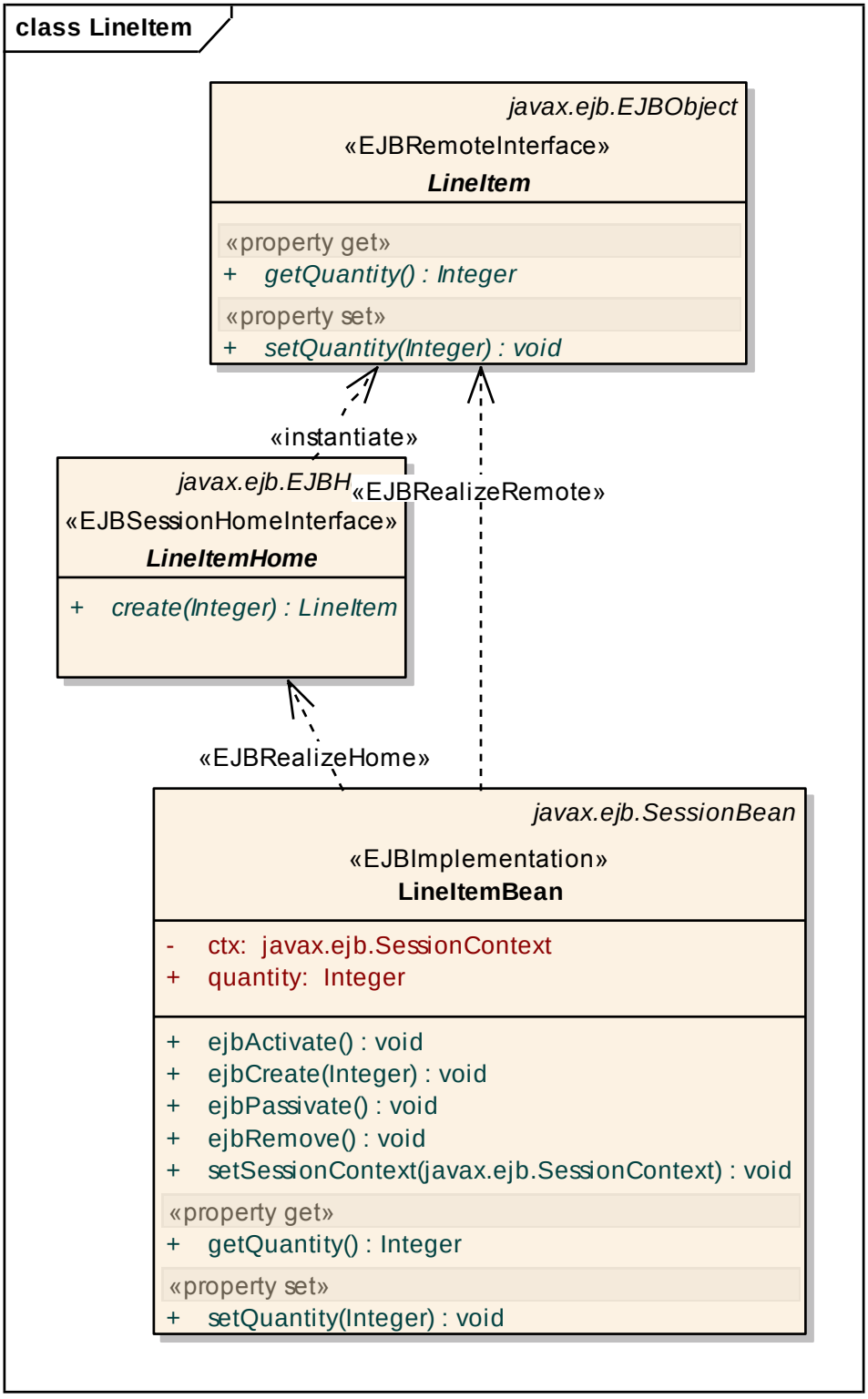


Diagram: META-INF

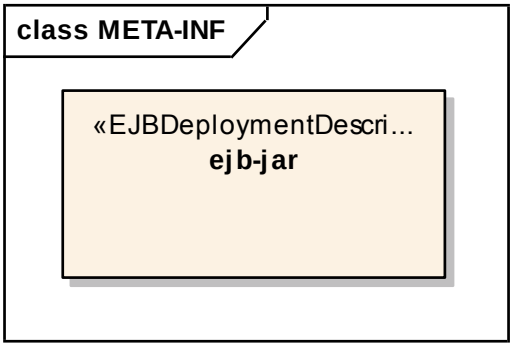


Diagram: Order

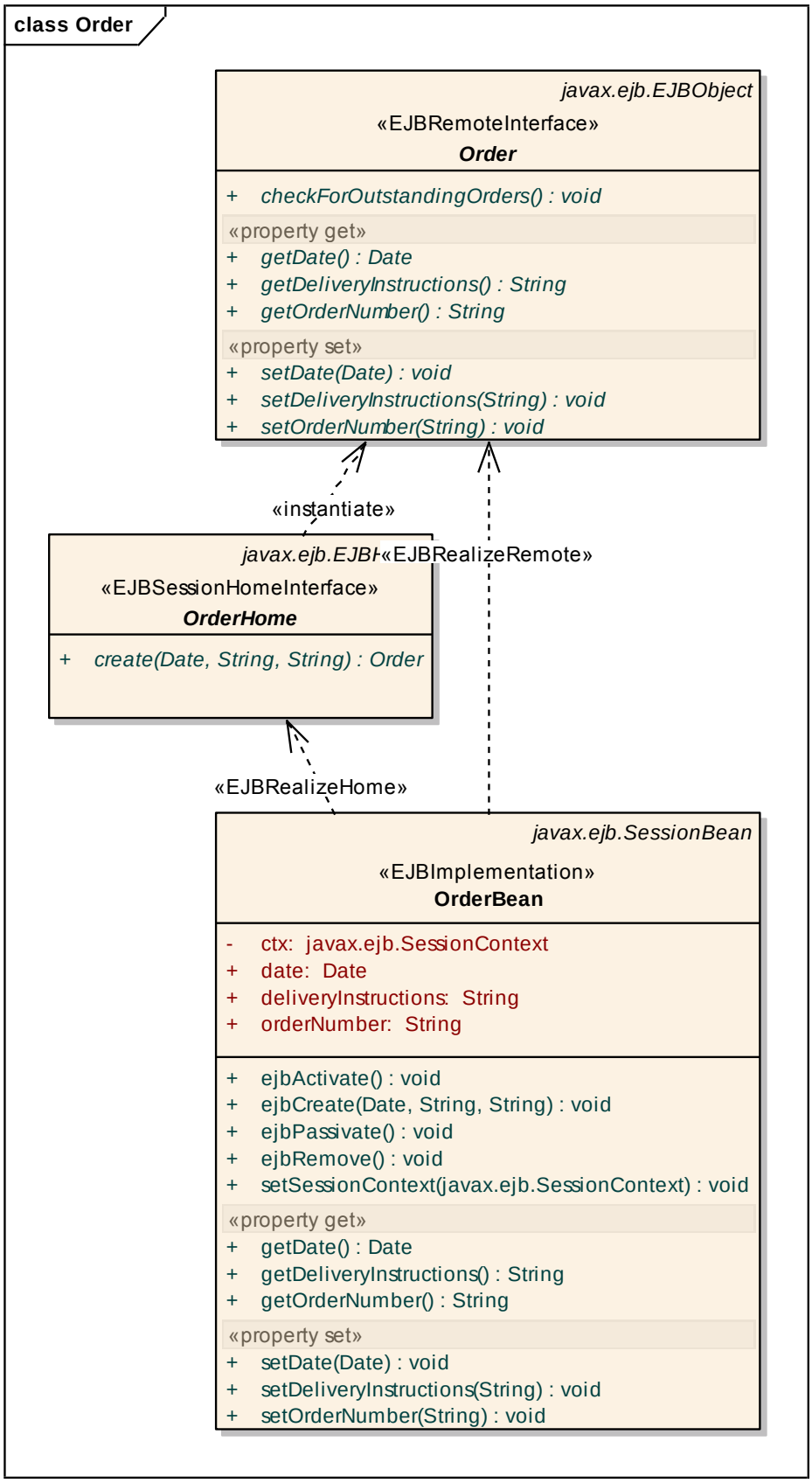

class Order
javax.ejb.EJBObject
«EJBRemoteInterface»
Order
+ checkForOutstandingOrders() : void
«property get»
+ getDate() : Date
+ getDeliveryInstructions() : String
+ getOrderNumber() : String
«property set»
+ setDate(Date) : void
+ setDeliveryInstructions(String) : void
+ setOrderNumber(String) : void
«instantiate»
javax.ejb.EJBH
«EJBRealizeRemote»
«EJBSessionHomeInterface»
OrderHome
+ create(Date, String, String) : Order
«EJBRealizeHome»
javax.ejb.SessionBean
«EJBImplementation»
OrderBean
- ctx: javax.ejb.SessionContext
+ date: Date
+ deliveryInstructions: String
+ orderNumber: String
+ ejbActivate() : void
+ ejbCreate(Date, String, String) : void
+ ejbPassivate() : void
+ ejbRemove() : void
+ setSessionContext(javax.ejb.SessionContext) : void
«property get»
+ getDate() : Date
+ getDeliveryInstructions() : String
+ getOrderNumber() : String
«property set»
+ setDate(Date) : void
+ setDeliveryInstructions(String) : void
+ setOrderNumber(String) : void

Diagram: META-INF

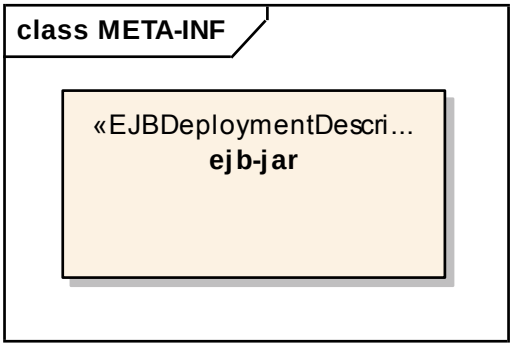


Diagram: ShoppingBasket

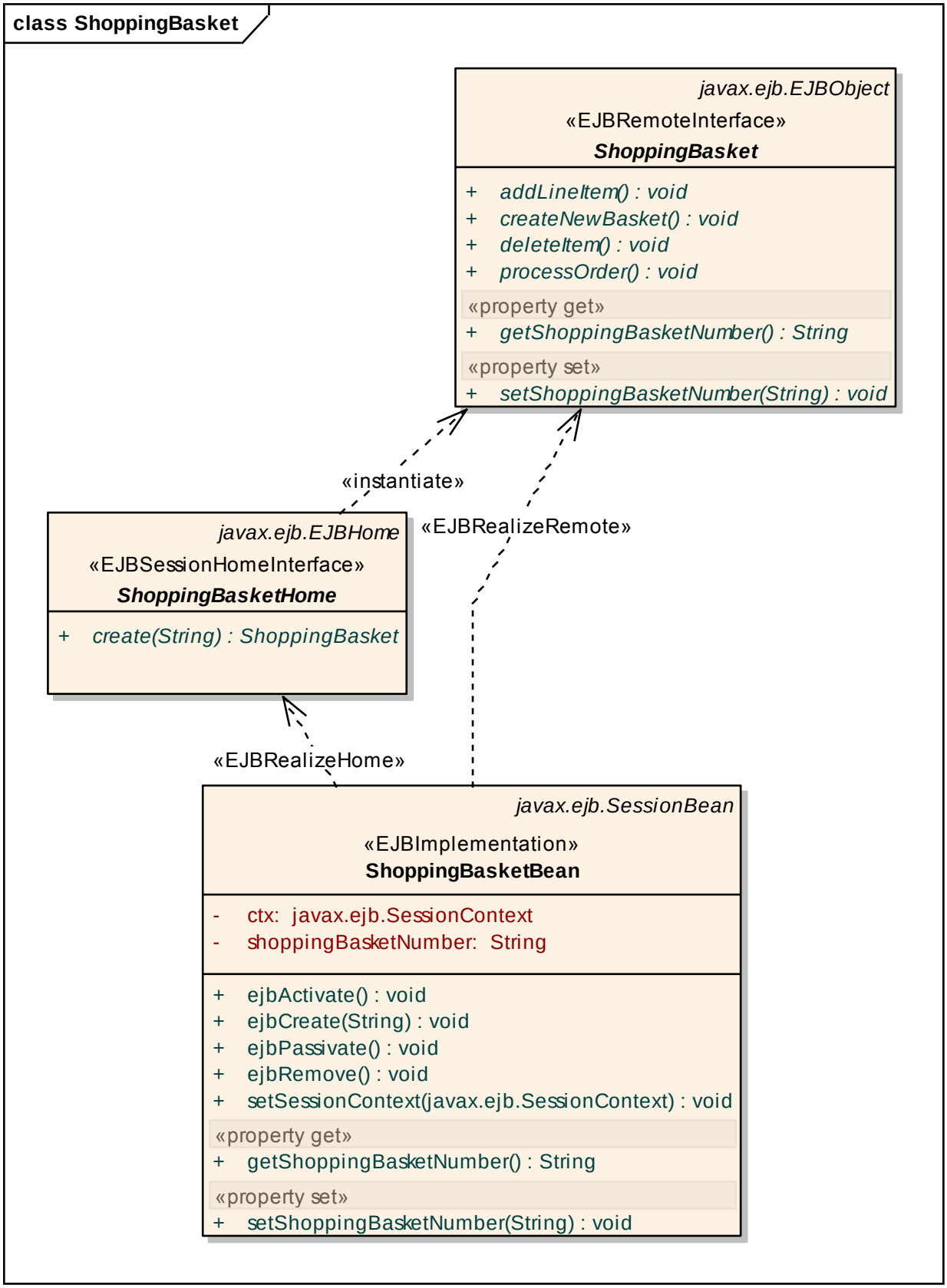

class ShoppingBasket
javax.ejb.EJBObject
«EJBRemoteInterface»
ShoppingBasket
+ addLineItem() : void
+ createNewBasket() : void
+ deleteItem() : void
+ processOrder() : void
«property get»
+ getShoppingBasketNumber() : String
«property set»
+ setShoppingBasketNumber(String) : void
«instantiate»
«EJBRealizeRemote»
javax.ejb.EJBHome
«EJBSessionHomeInterface»
ShoppingBasketHome
+ create(String) : ShoppingBasket
«EJBRealizeHome»
javax.ejb.SessionBean
«EJBImplementation»
ShoppingBasketBean
- ctx: javax.ejb.SessionContext
- shoppingBasketNumber: String
+ ejbActivate() : void
+ ejbCreate(String) : void
+ ejbPassivate() : void
+ ejbRemove() : void
+ setSessionContext(javax.ejb.SessionContext) : void
«property get»
+ getShoppingBasketNumber() : String
«property set»
+ setShoppingBasketNumber(String) : void

Diagram: META-INF

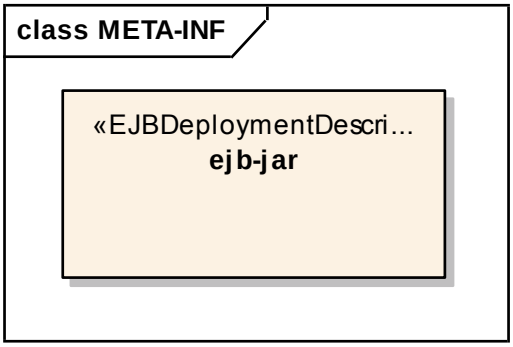


Diagram: StockItem

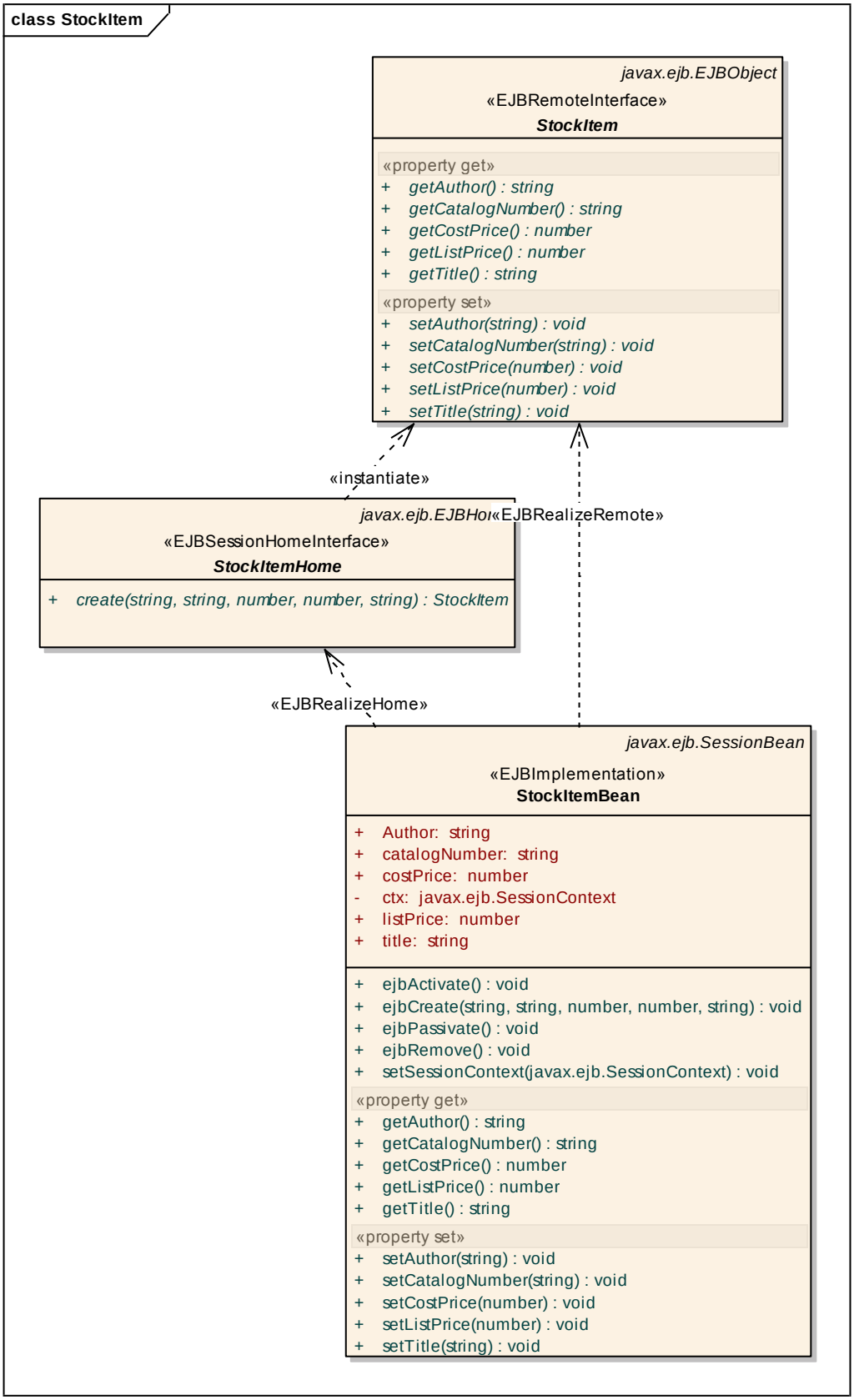

class StockItem
javax.ejb.EJBObject
«EJBRemoteInterface»
StockItem
«property get»
+ getAuthor() : string
+ getCatalogNumber() : string
+ getCostPrice() : number
+ getListPrice() : number
+ getTitle() : string
«property set»
+ setAuthor(string) : void
+ setCatalogNumber(string) : void
+ setCostPrice(number) : void
+ setListPrice(number) : void
+ setTitle(string) : void
«instantiate»
«EJBRealizeRemote»
javax.ejb.EJBHo
«EJBSessionHomeInterface»
StockItemHome
+ create(string, string, number, number, string) : StockItem
«EJBRealizeHome»
javax.ejb.SessionBean
«EJBImplementation»
StockItemBean
+ Author: string
+ catalogNumber: string
+ costPrice: number
- ctx: javax.ejb.SessionContext
+ listPrice: number
+ title: string
+ ejbActivate() : void
+ ejbCreate(string, string, number, number, string) : void
+ ejbPassivate() : void
+ ejbRemove() : void
+ setSessionContext(javax.ejb.SessionContext) : void
«property get»
+ getAuthor() : string
+ getCatalogNumber() : string
+ getCostPrice() : number
+ getListPrice() : number
+ getTitle() : string
«property set»
+ setAuthor(string) : void
+ setCatalogNumber(string) : void
+ setCostPrice(number) : void
+ setListPrice(number) : void
+ setTitle(string) : void

Diagram: META-INF

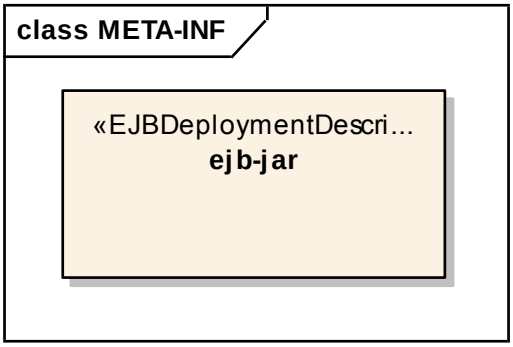


Diagram: Transaction

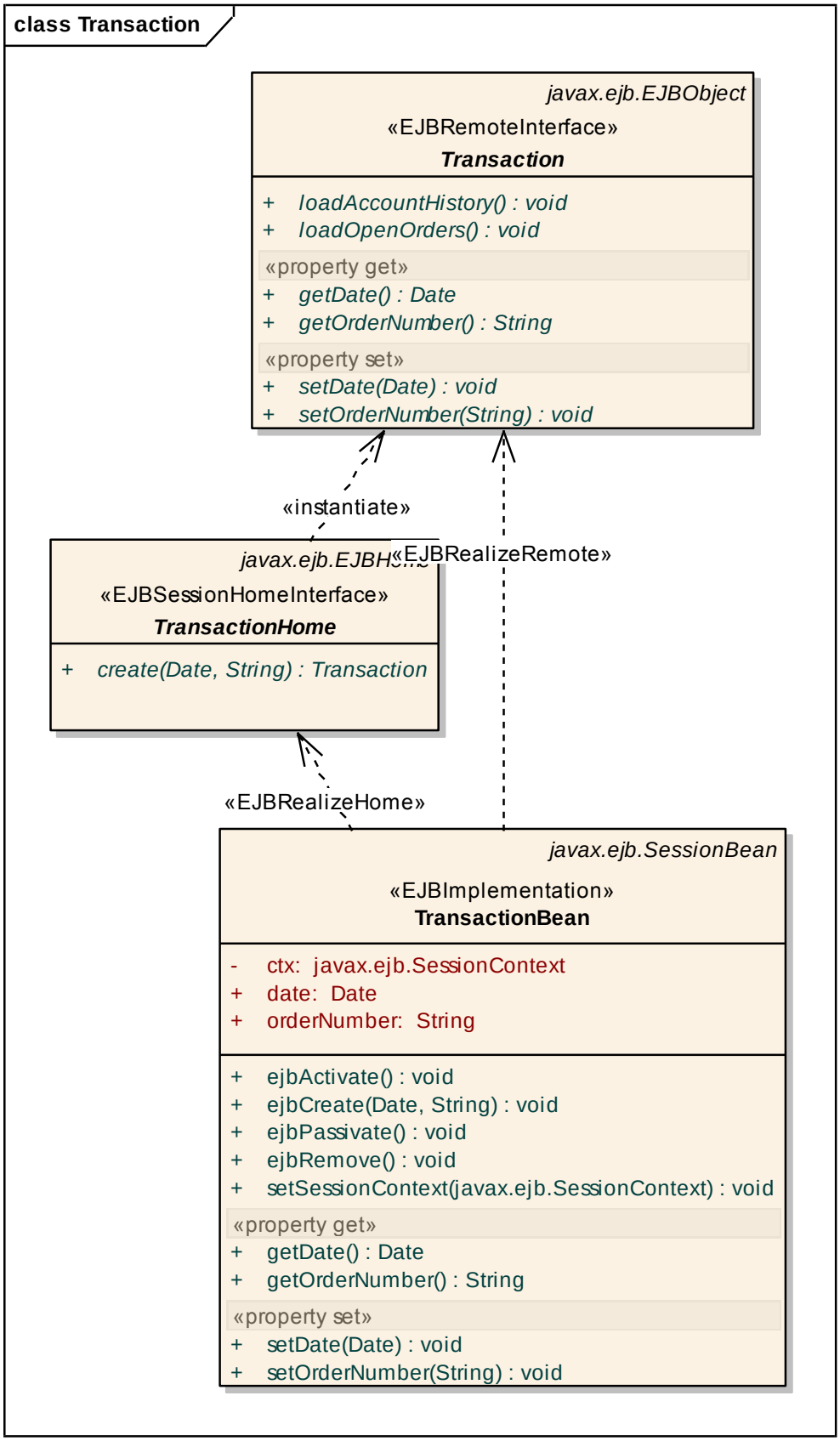

class Transaction
javax.ejb.EJBObject
«EJBRemoteInterface»
Transaction
+ loadAccountHistory() : void
+ loadOpenOrders() : void
«property get»
+ getDate() : Date
+ getOrderNumber() : String
«property set»
+ setDate(Date) : void
+ setOrderNumber(String) : void
«instantiate»
«EJBRealizeRemote»
javax.ejb.EJBHome
«EJBSessionHomeInterface»
TransactionHome
+ create(Date, String) : Transaction
«EJBRealizeHome»
javax.ejb.SessionBean
«EJBImplementation»
TransactionBean
- ctx: javax.ejb.SessionContext
+ date: Date
+ orderNumber: String
+ ejbActivate() : void
+ ejbCreate(Date, String) : void
+ ejbPassivate() : void
+ ejbRemove() : void
+ setSessionContext(javax.ejb.SessionContext) : void
«property get»
+ getDate() : Date
+ getOrderNumber() : String
«property set»
+ setDate(Date) : void
+ setOrderNumber(String) : void

Diagram: META-INF

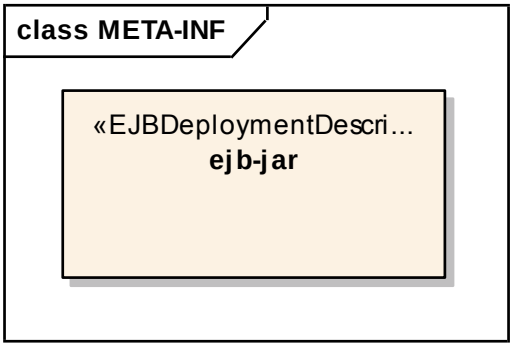


Diagram: OrderStatus

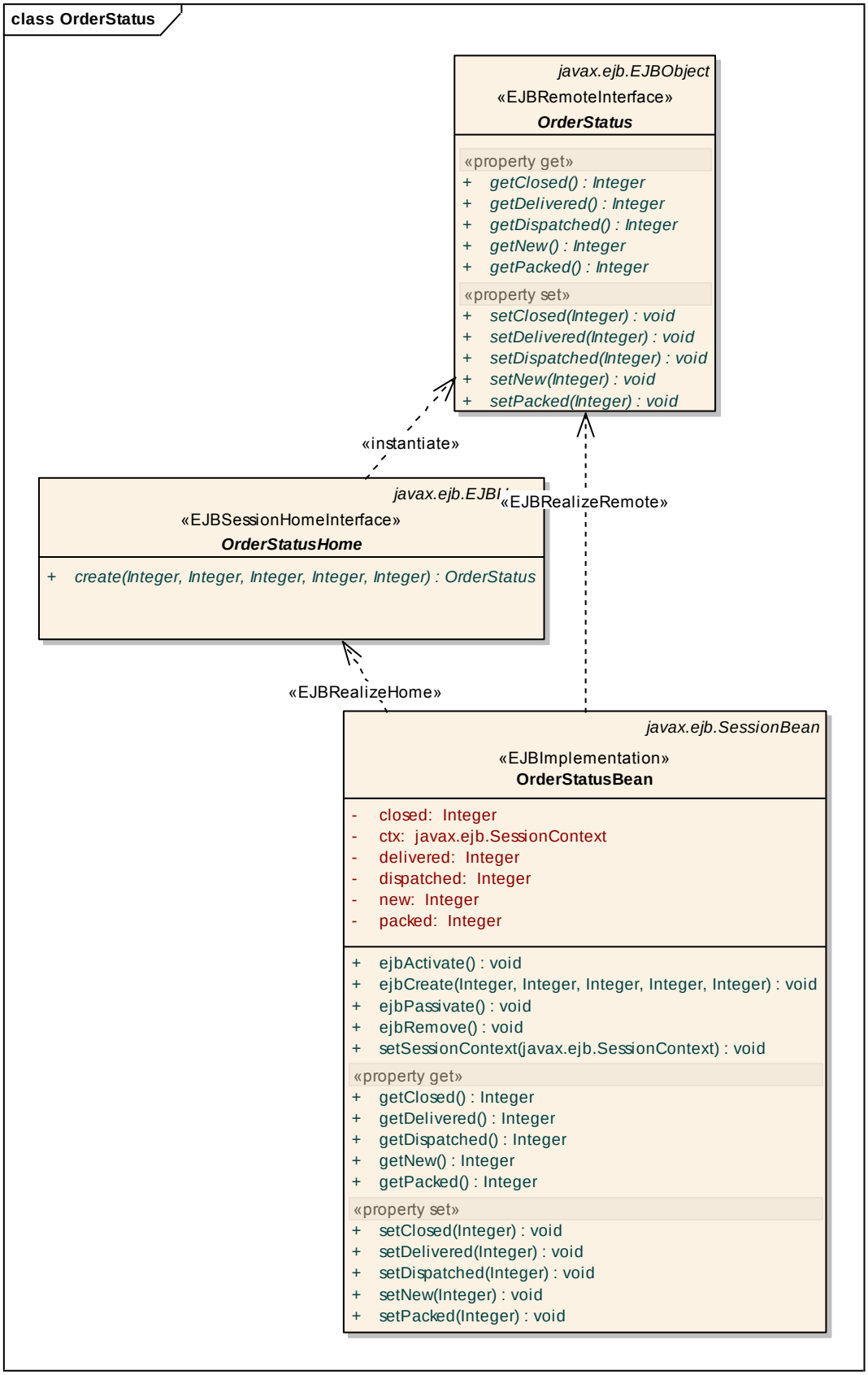

class OrderStatus
javax.ejb.EJBObject
«EJBRemoteInterface»
OrderStatus
«property get»
+ getClosed() : Integer
+ getDelivered() : Integer
+ getDispatched() : Integer
+ getNew() : Integer
+ getPacked() : Integer
«property set»
+ setClosed(Integer) : void
+ setDelivered(Integer) : void
+ setDispatched(Integer) : void
+ setNew(Integer) : void
+ setPacked(Integer) : void
«instantiate»
«EJBRealizeRemote»
javax.ejb.EJBI
«EJBSessionHomeInterface»
OrderStatusHome
+ create(Integer, Integer, Integer, Integer, Integer) : OrderStatus
«EJBRealizeHome»
javax.ejb.SessionBean
«EJBImplementation»
OrderStatusBean
- closed: Integer
- ctx: javax.ejb.SessionContext
- delivered: Integer
- dispatched: Integer
- new: Integer
- packed: Integer
+ ejbActivate() : void
+ ejbCreate(Integer, Integer, Integer, Integer, Integer) : void
+ ejbPassivate() : void
+ ejbRemove() : void
+ setSessionContext(javax.ejb.SessionContext) : void
«property get»
+ getClosed() : Integer
+ getDelivered() : Integer
+ getDispatched() : Integer
+ getNew() : Integer
+ getPacked() : Integer
«property set»
+ setClosed(Integer) : void
+ setDelivered(Integer) : void
+ setDispatched(Integer) : void
+ setNew(Integer) : void
+ setPacked(Integer) : void

Diagram: META-INF

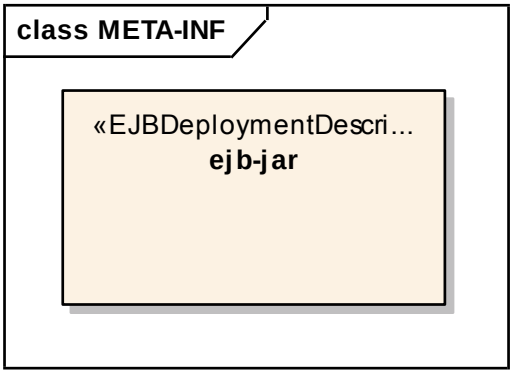


Diagram: XSD

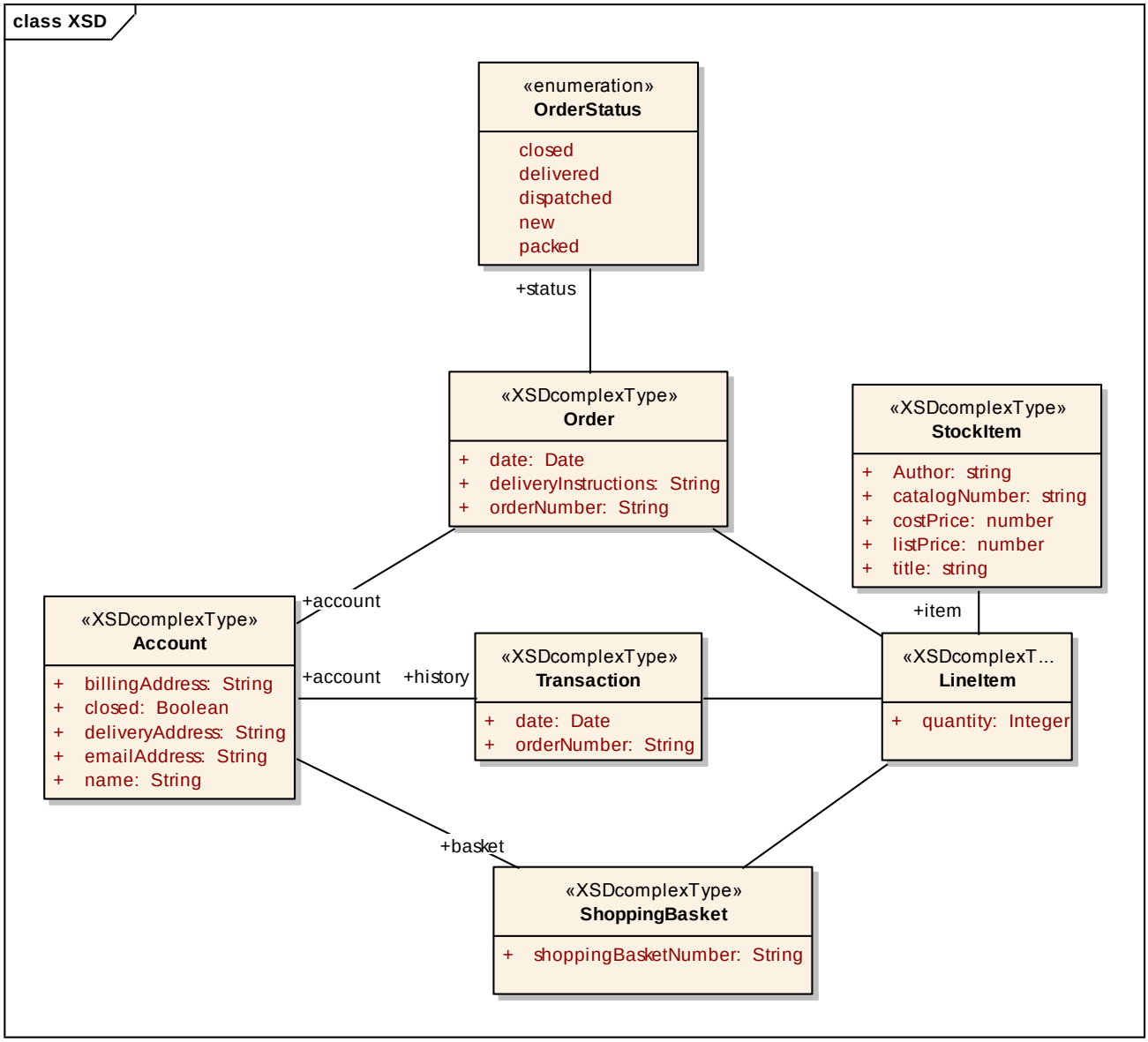

class XSD
«enumeration»
OrderStatus
closed
delivered
dispatched
new
packed
+status
«XSDcomplexType»
Order
+ date: Date
+ deliveryInstructions: String
+ orderNumber: String
«XSDcomplexType»
StockItem
+ Author: string
+ catalogNumber: string
+ costPrice: number
+ listPrice: number
+ title: string
+item
«XSDcomplexType»
Account
+ billingAddress: String
+ closed: Boolean
+ deliveryAddress: String
+ emailAddress: String
+ name: String
+account
+account
+history
«XSDcomplexType»
Transaction
+ date: Date
+ orderNumber: String
«XSDcomplexT...
LineItem
+ quantity: Integer
+basket
«XSDcomplexType»
ShoppingBasket
+ shoppingBasketNumber: String

Diagram: Java Model

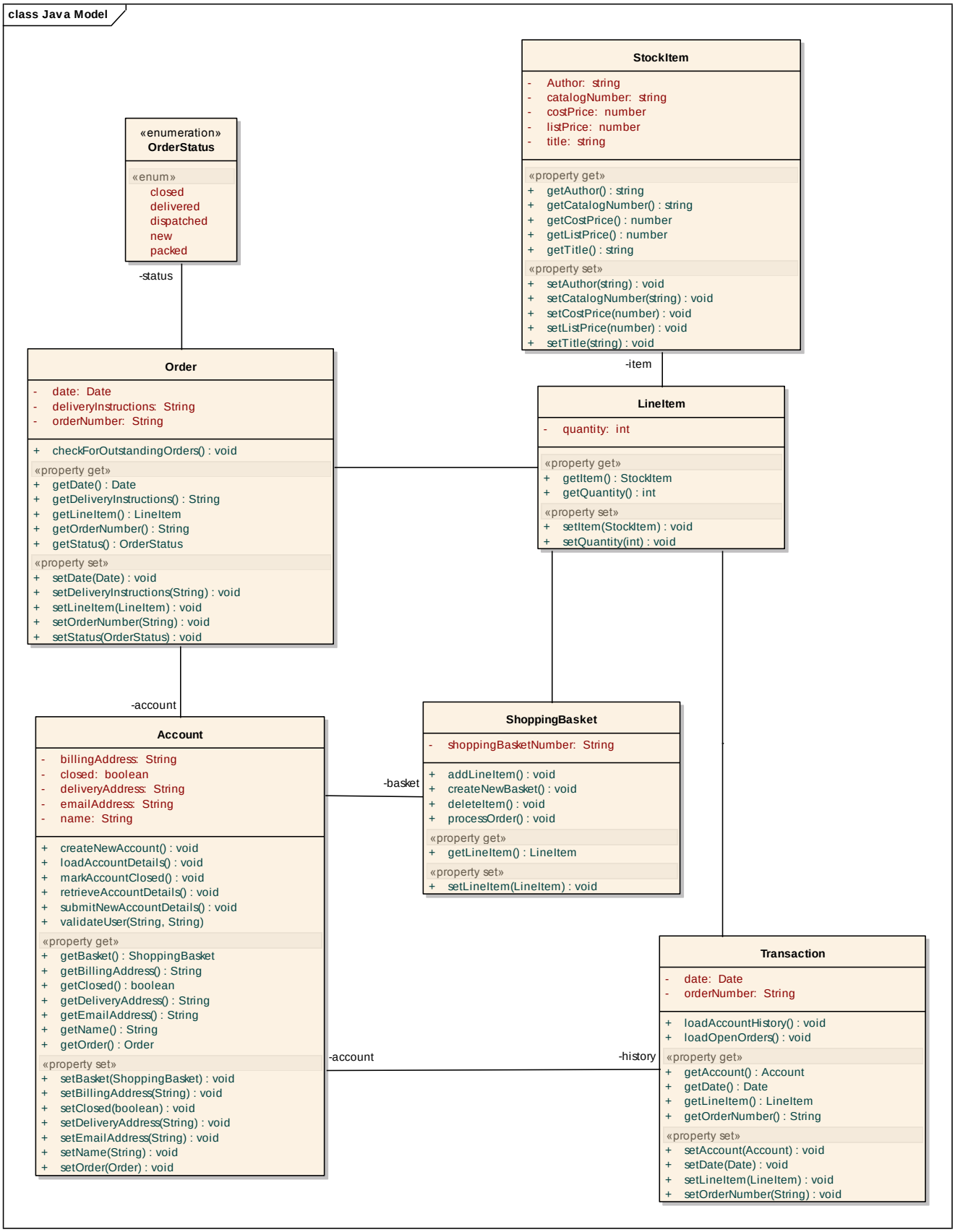

class Java Model
«enumeration»
OrderStatus
«enum»
closed
delivered
dispatched
new
packed
-status
StockItem
- Author: string
- catalogNumber: string
- costPrice: number
- listPrice: number
- title: string
«property get»
+ getAuthor() : string
+ getCatalogNumber() : string
+ getCostPrice() : number
+ getListPrice() : number
+ getTitle() : string
«property set»
+ setAuthor(string) : void
+ setCatalogNumber(string) : void
+ setCostPrice(number) : void
+ setListPrice(number) : void
+ setTitle(string) : void
-item
Order
- date: Date
- deliveryInstructions: String
- orderNumber: String
+ checkForOutstandingOrders() : void
«property get»
+ getDate() : Date
+ getDeliveryInstructions() : String
+ getLineItem() : LineItem
+ getOrderNumber() : String
+ getStatus() : OrderStatus
«property set»
+ setDate(Date) : void
+ setDeliveryInstructions(String) : void
+ setLineItem(LineItem) : void
+ setOrderNumber(String) : void
+ setStatus(OrderStatus) : void
LineItem
- quantity: int
«property get»
+ getItem() : StockItem
+ getQuantity() : int
«property set»
+ setItem(StockItem) : void
+ setQuantity(int) : void
-account
Account
- billingAddress: String
- closed: boolean
- deliveryAddress: String
- emailAddress: String
- name: String
+ createNewAccount() : void
+ loadAccountDetails() : void
+ markAccountClosed() : void
+ retrieveAccountDetails() : void
+ submitNewAccountDetails() : void
+ validateUser(String, String)
«property get»
+ getBasket() : ShoppingBasket
+ getBillingAddress() : String
+ getClosed() : boolean
+ getDeliveryAddress() : String
+ getEmailAddress() : String
+ getName() : String
+ getOrder() : Order
«property set»
+ setBasket(ShoppingBasket) : void
+ setBillingAddress(String) : void
+ setClosed(boolean) : void
+ setDeliveryAddress(String) : void
+ setEmailAddress(String) : void
+ setName(String) : void
+ setOrder(Order) : void
-basket
ShoppingBasket
- shoppingBasketNumber: String
+ addLineItem() : void
+ createNewBasket() : void
+ deleteItem() : void
+ processOrder() : void
«property get»
+ getLineItem() : LineItem
«property set»
+ setLineItem(LineItem) : void
-account
-history
Transaction
- date: Date
- orderNumber: String
+ loadAccountHistory() : void
+ loadOpenOrders() : void
«property get»
+ getAccount() : Account
+ getDate() : Date
+ getLineItem() : LineItem
+ getOrderNumber() : String
«property set»
+ setAccount(Account) : void
+ setDate(Date) : void
+ setLineItem(LineItem) : void
+ setOrderNumber(String) : void

Diagram: Test Environment Model

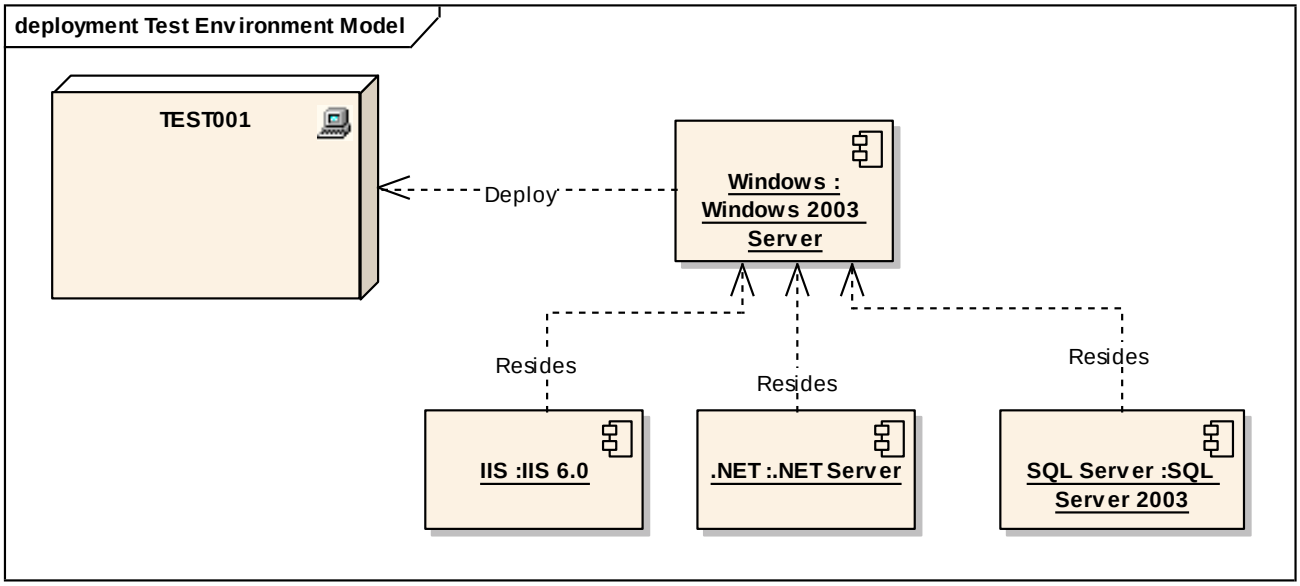


Diagram: Development Environment Model

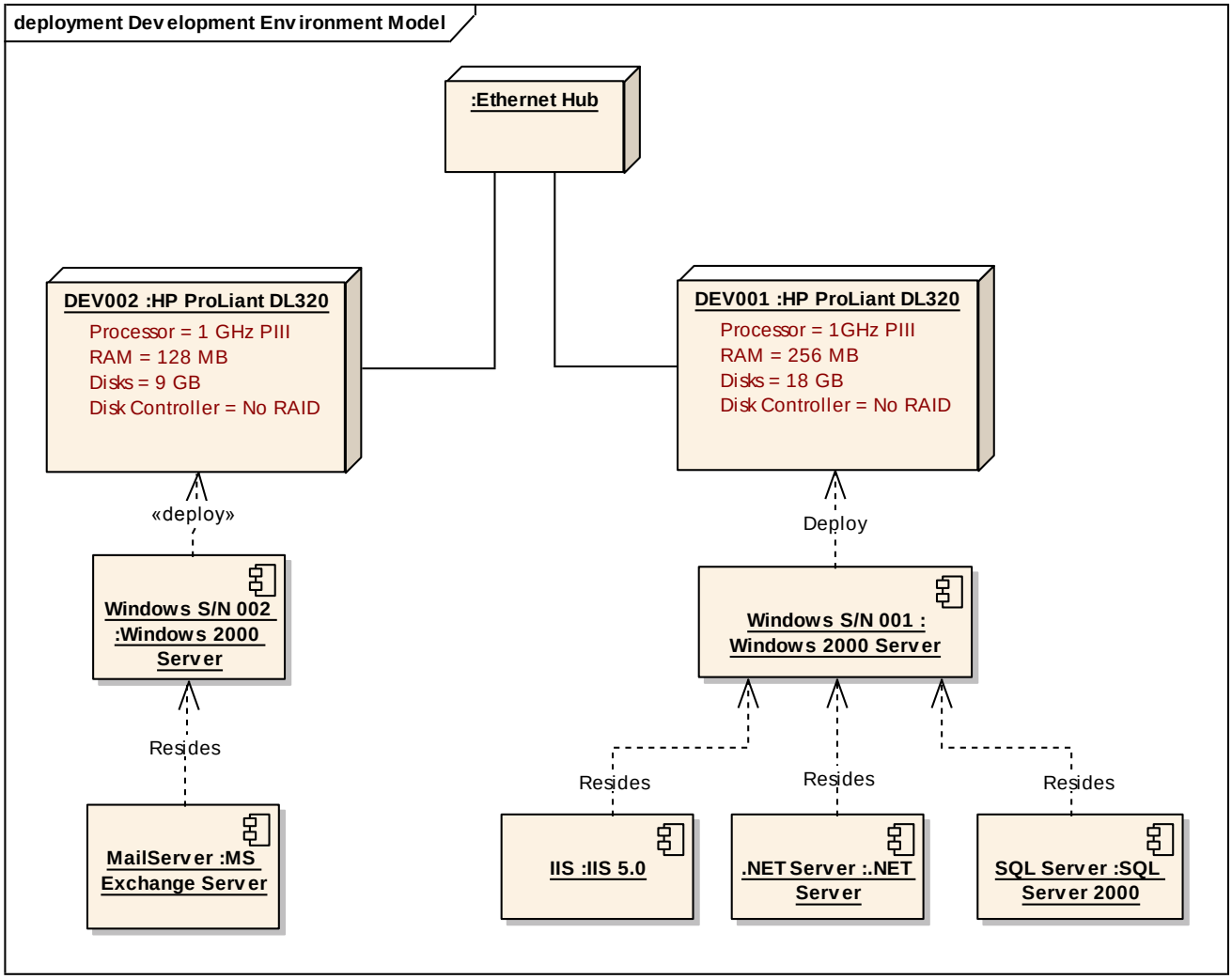


Diagram: View Orders

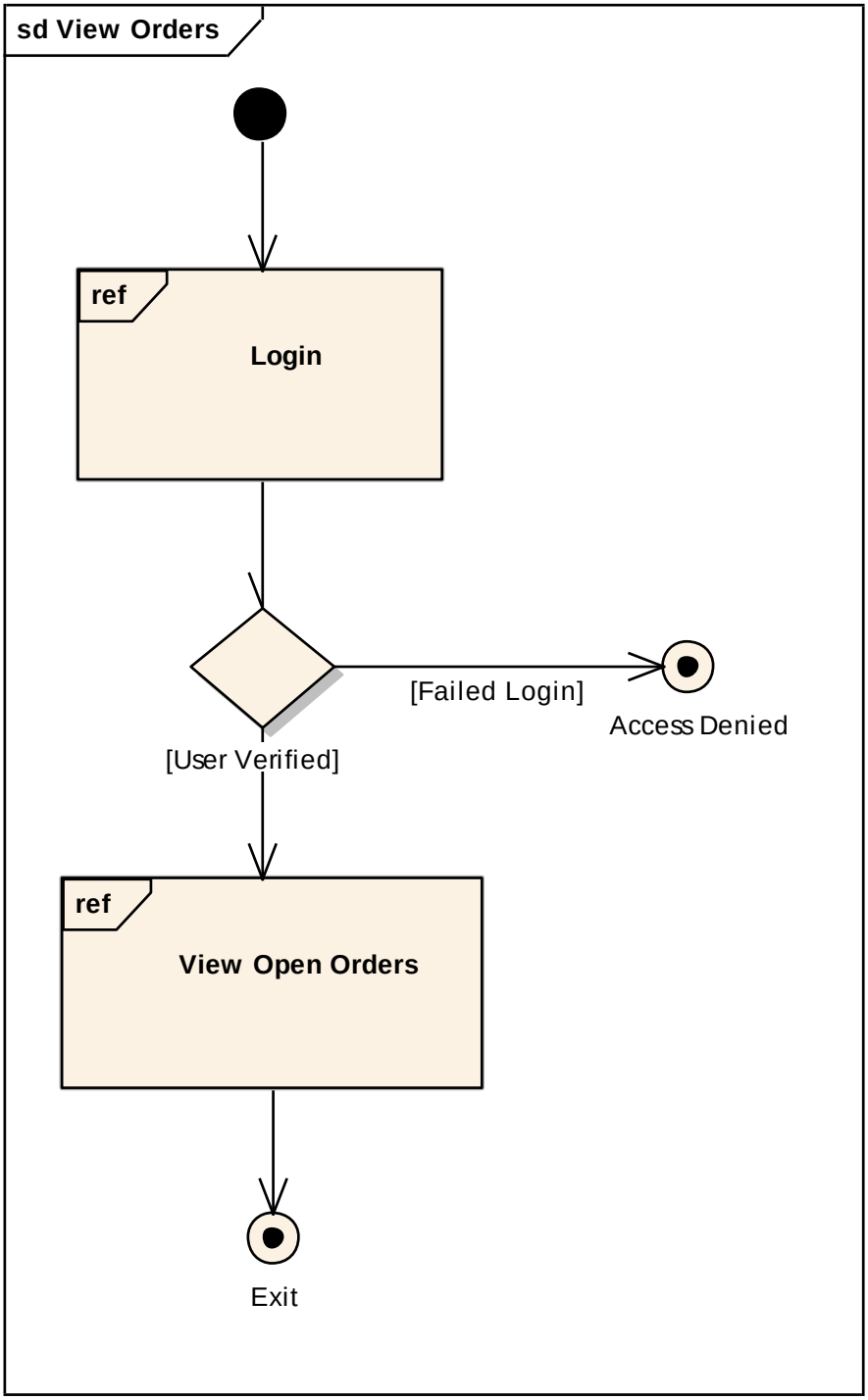


Diagram: Login

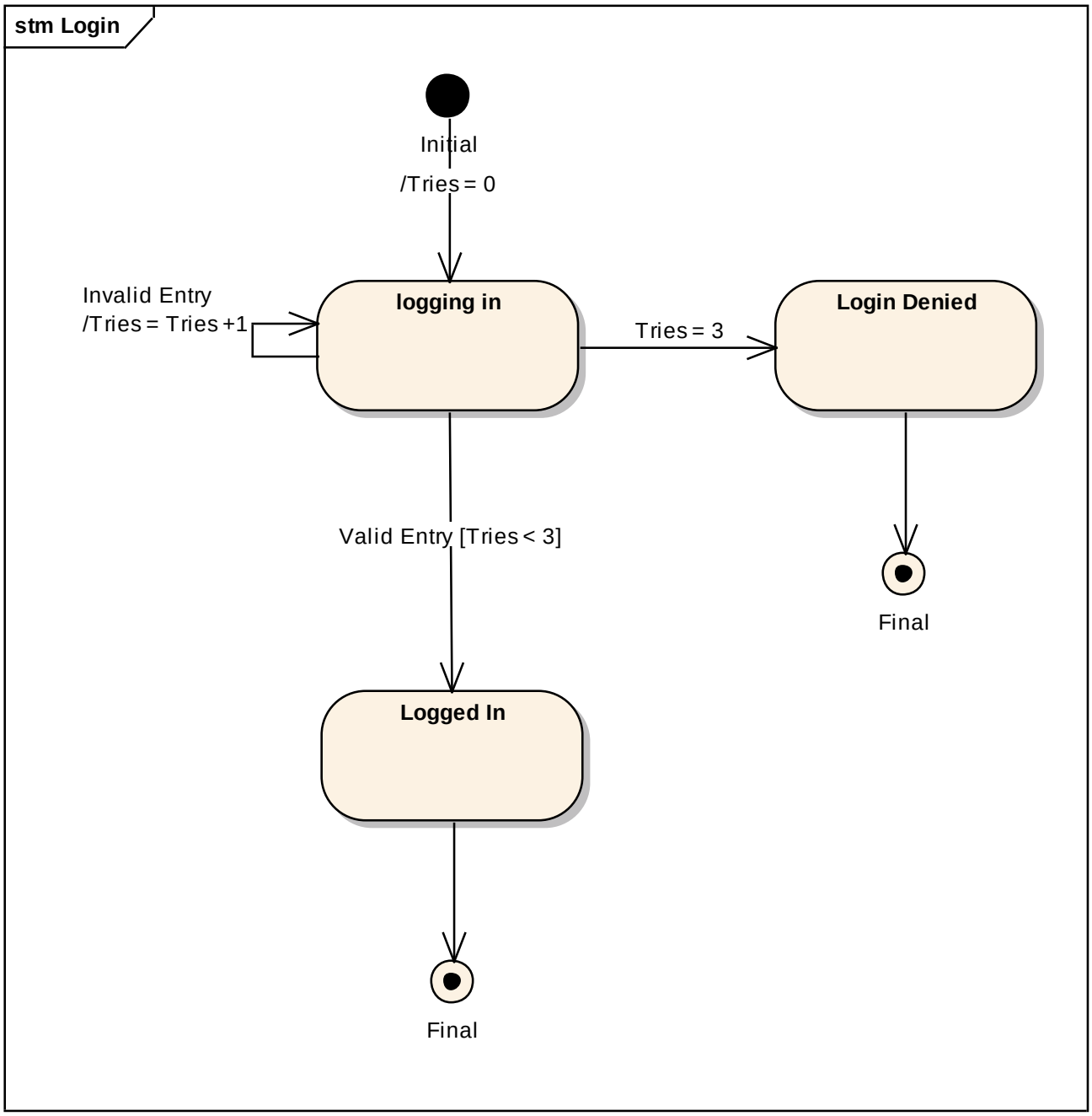


Diagram: Search

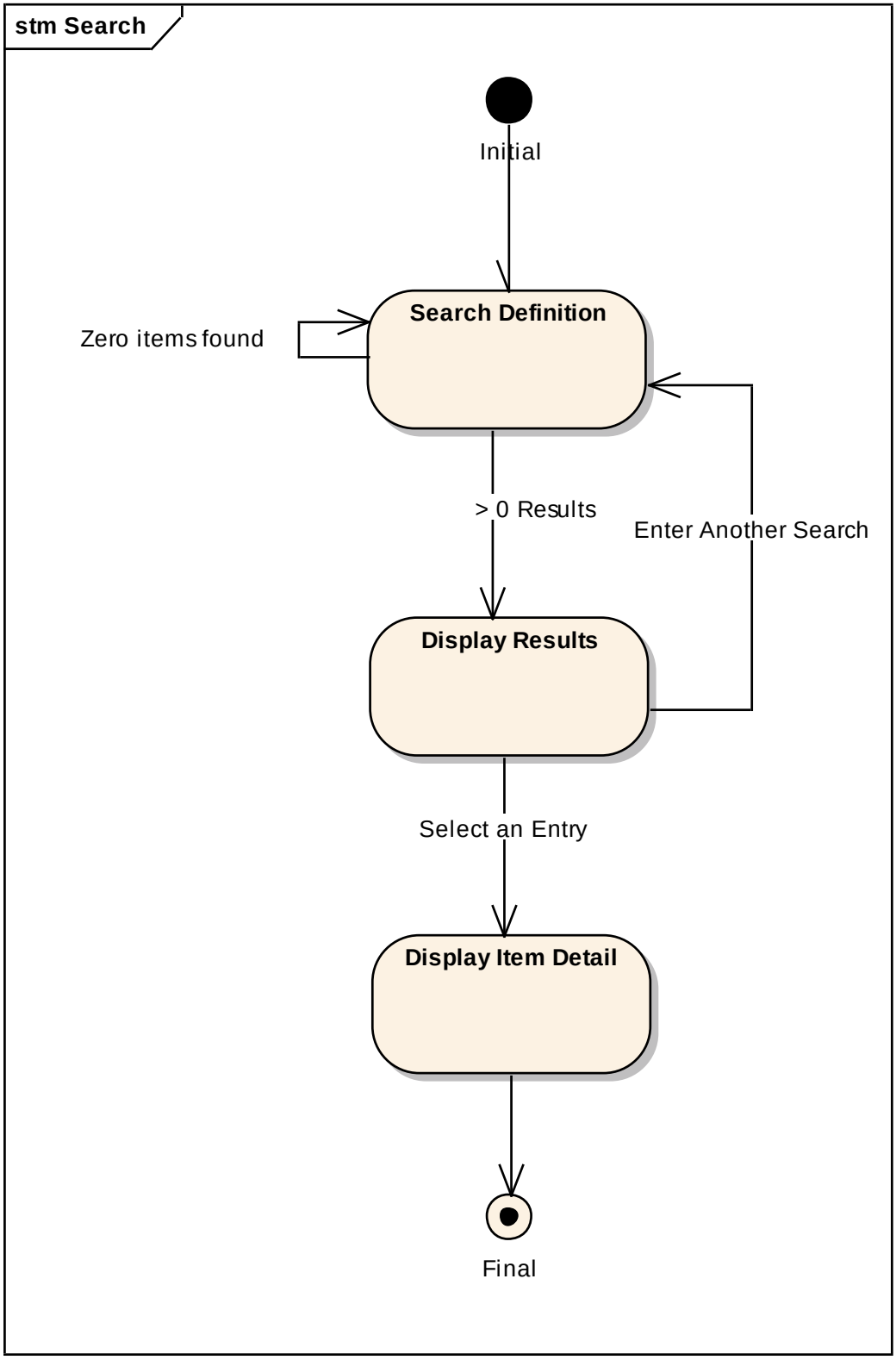


Diagram: Customer Order

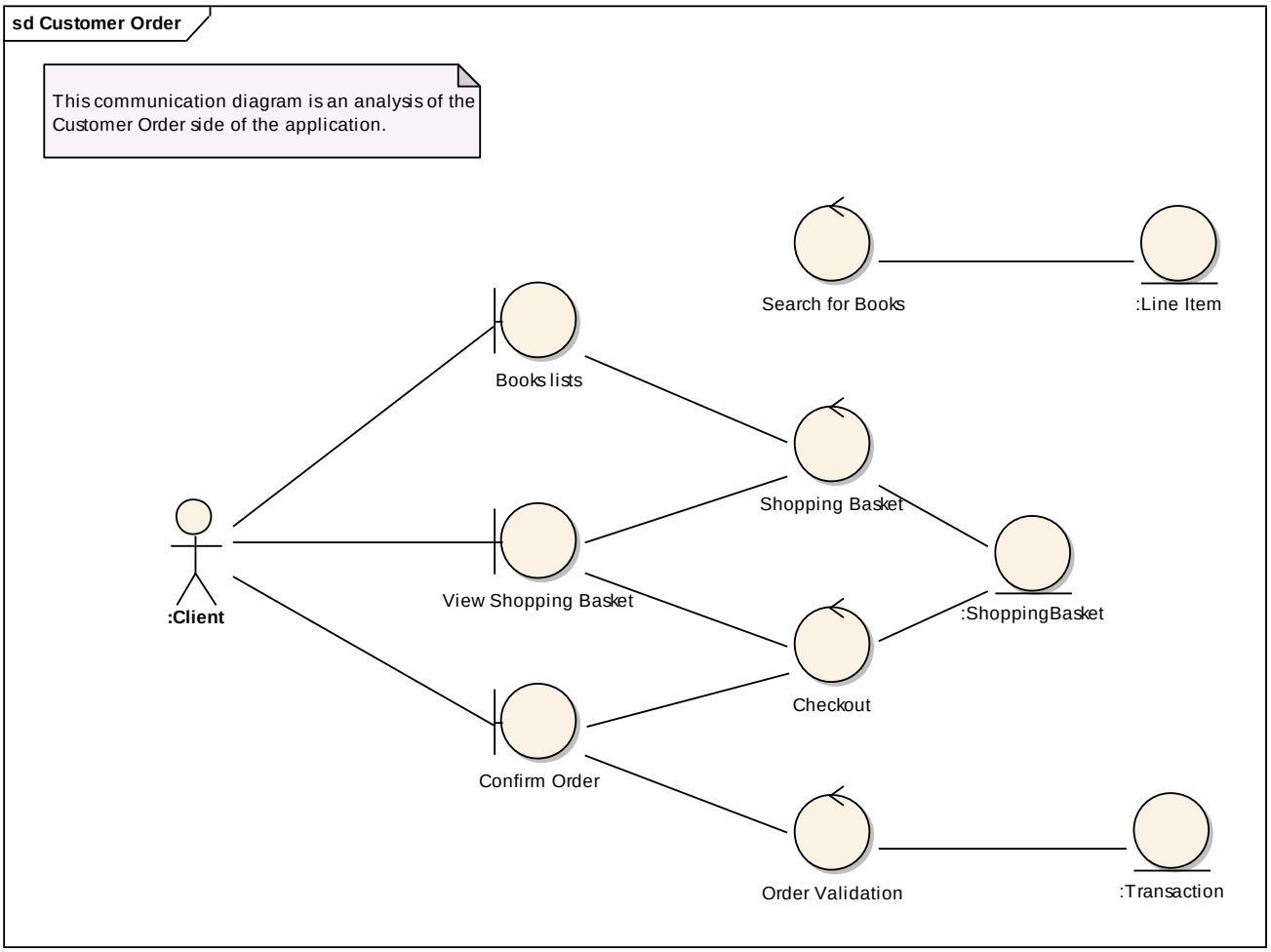


Diagram: QA Model

class QA Model

Quality Assurance

There are a number methods for setting up testing definitions and tracking test results in Enterprise Architect. Two key formats that can be used for this are:

1. Internal test definitions within an element.
2.. Attaching a custom element of type "Test Case" to existing elements.

Below are links with examples of using these two methods.

1. Display internal testing in an Element (i.e. Class)

2. Use a Test Case Element

Enterprise Architect also supports custom elements for Issues and Changes. These are useful for logging maintenance updates. These can be also be viewed with the test definitions displayed.

For examples of this see:

Changes Example

Issues Example

For a document of the Test Plan see the internal RTF document covering the test strategy:

Testing::Test Plan

Diagram: Take Orders

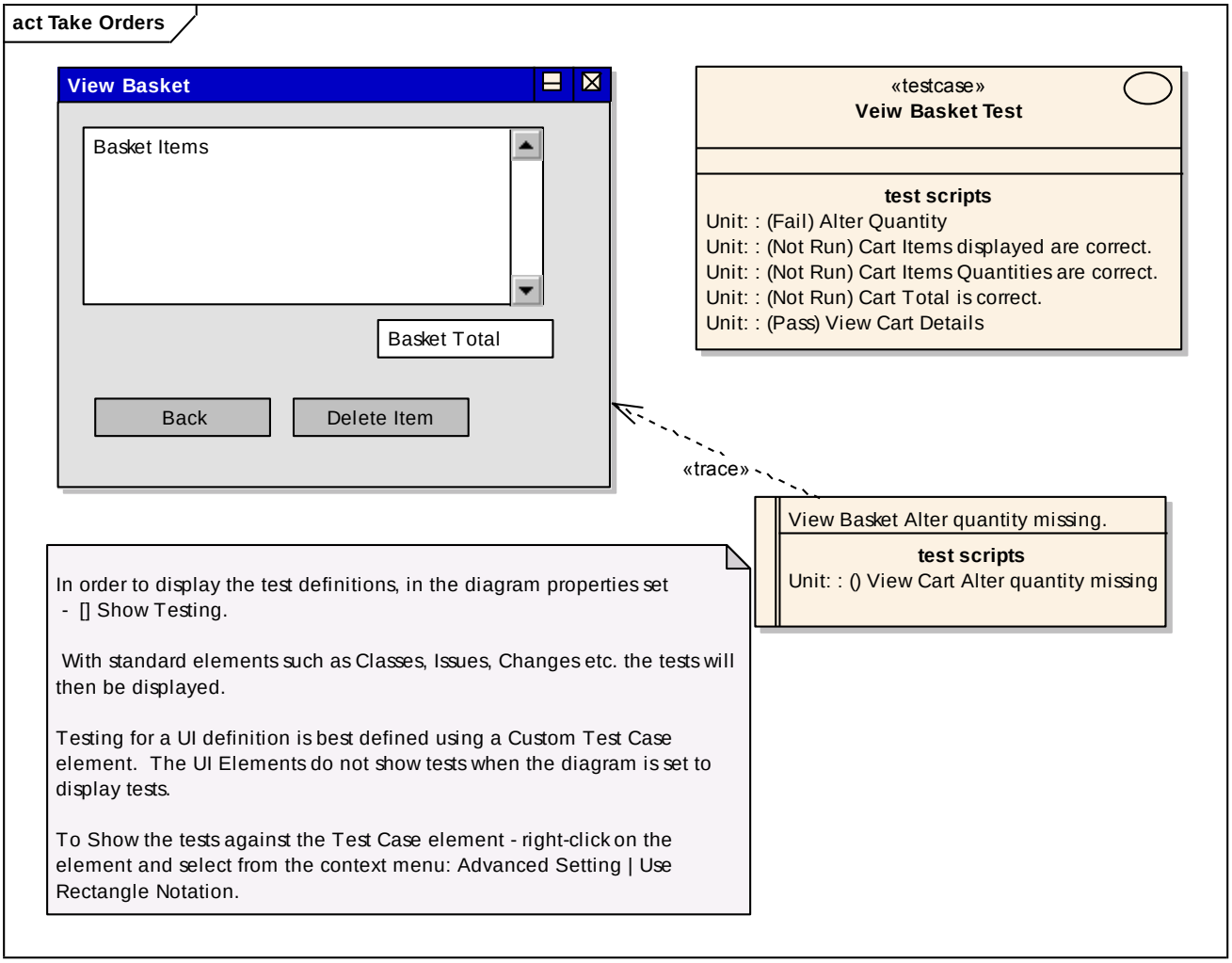


Diagram: Place Order

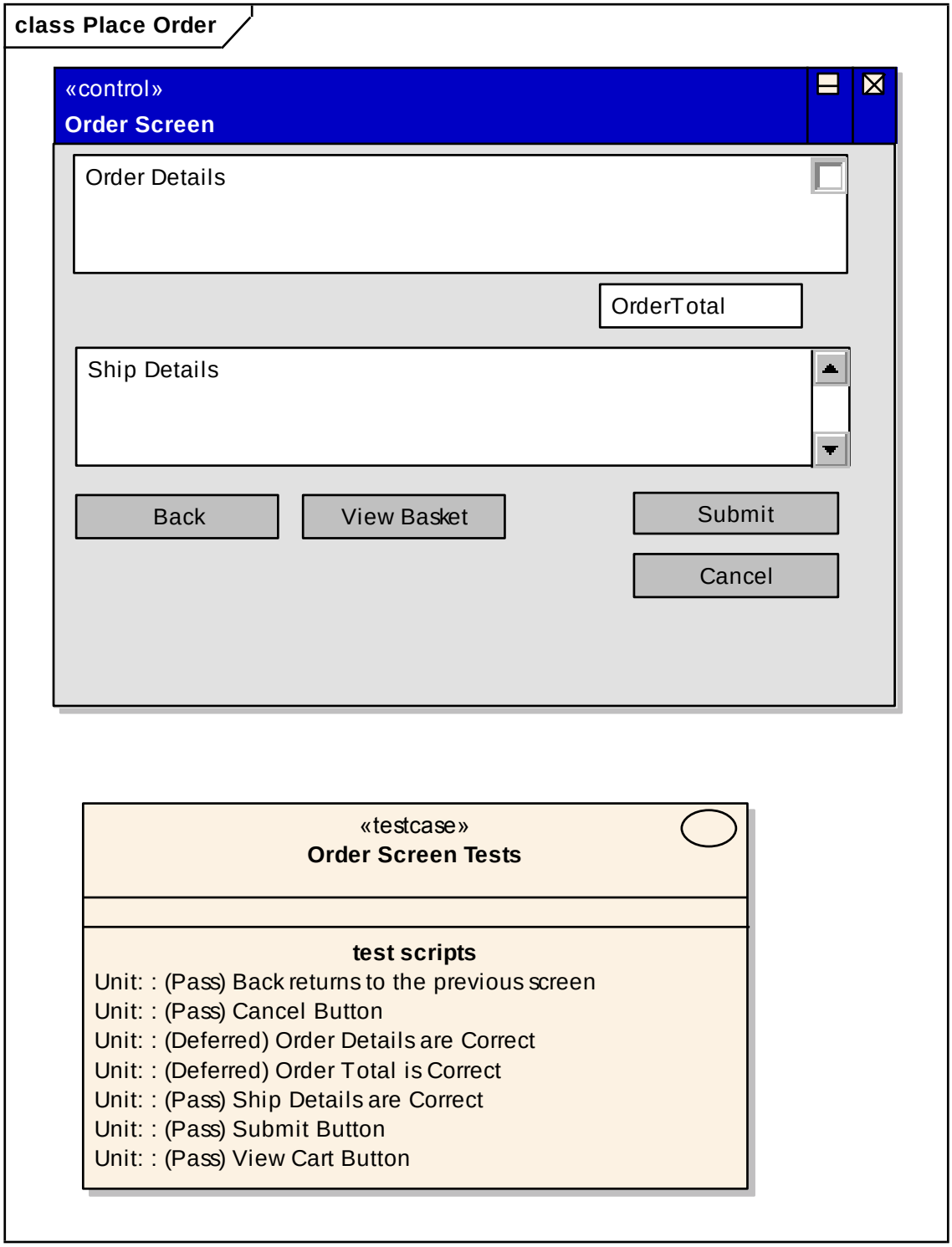


Diagram: Account

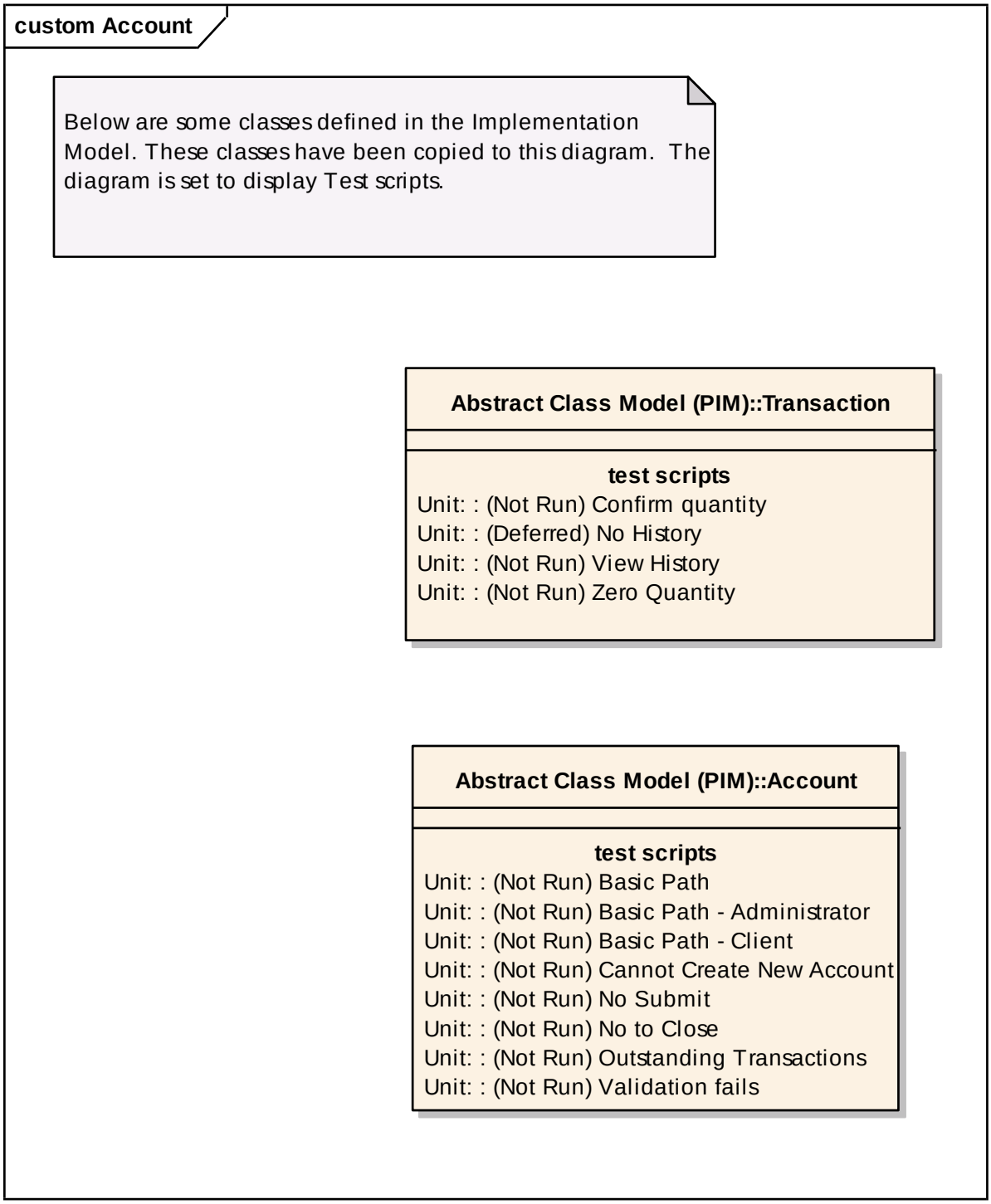


Diagram: Orders

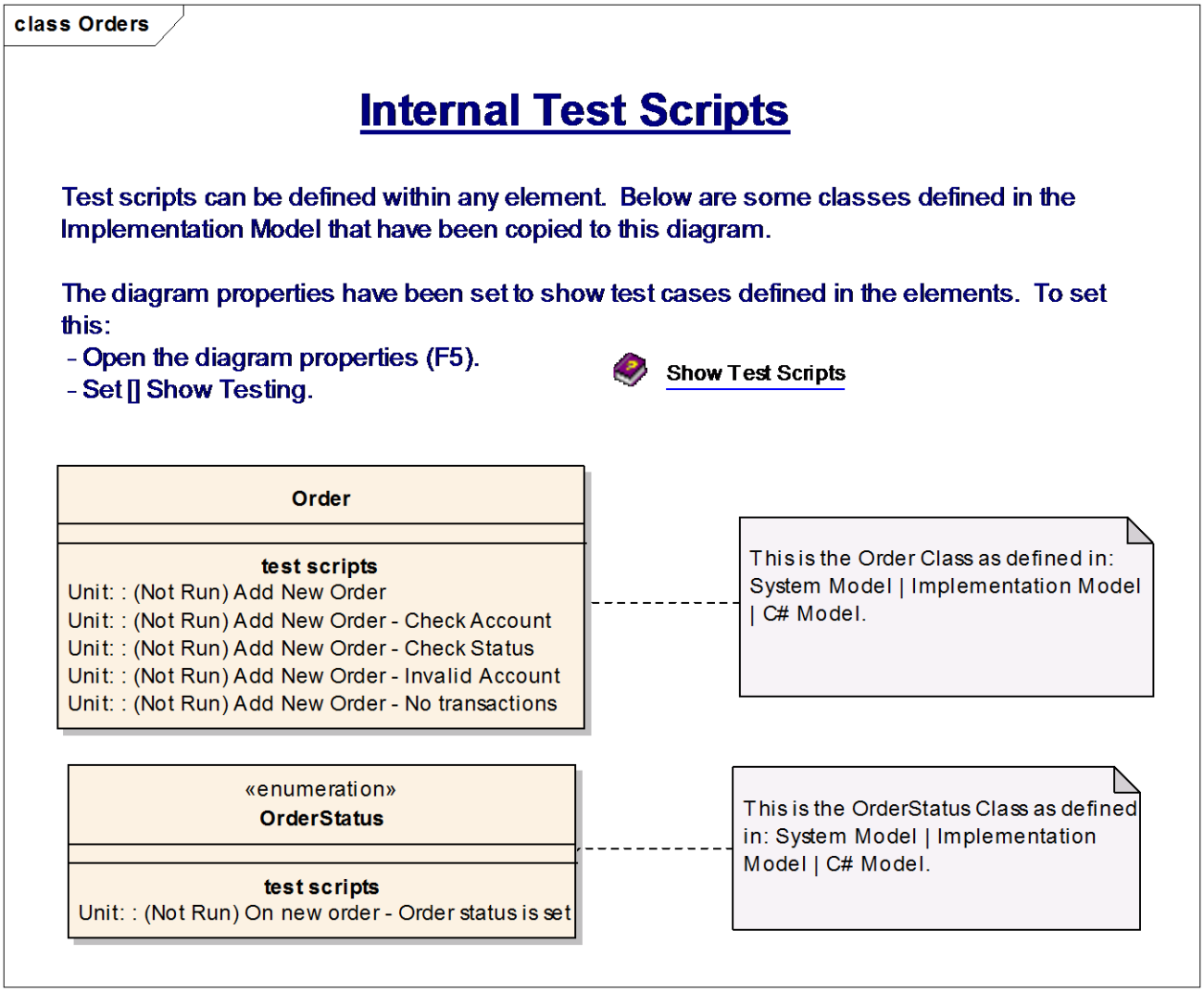


Diagram: Shopping Cart

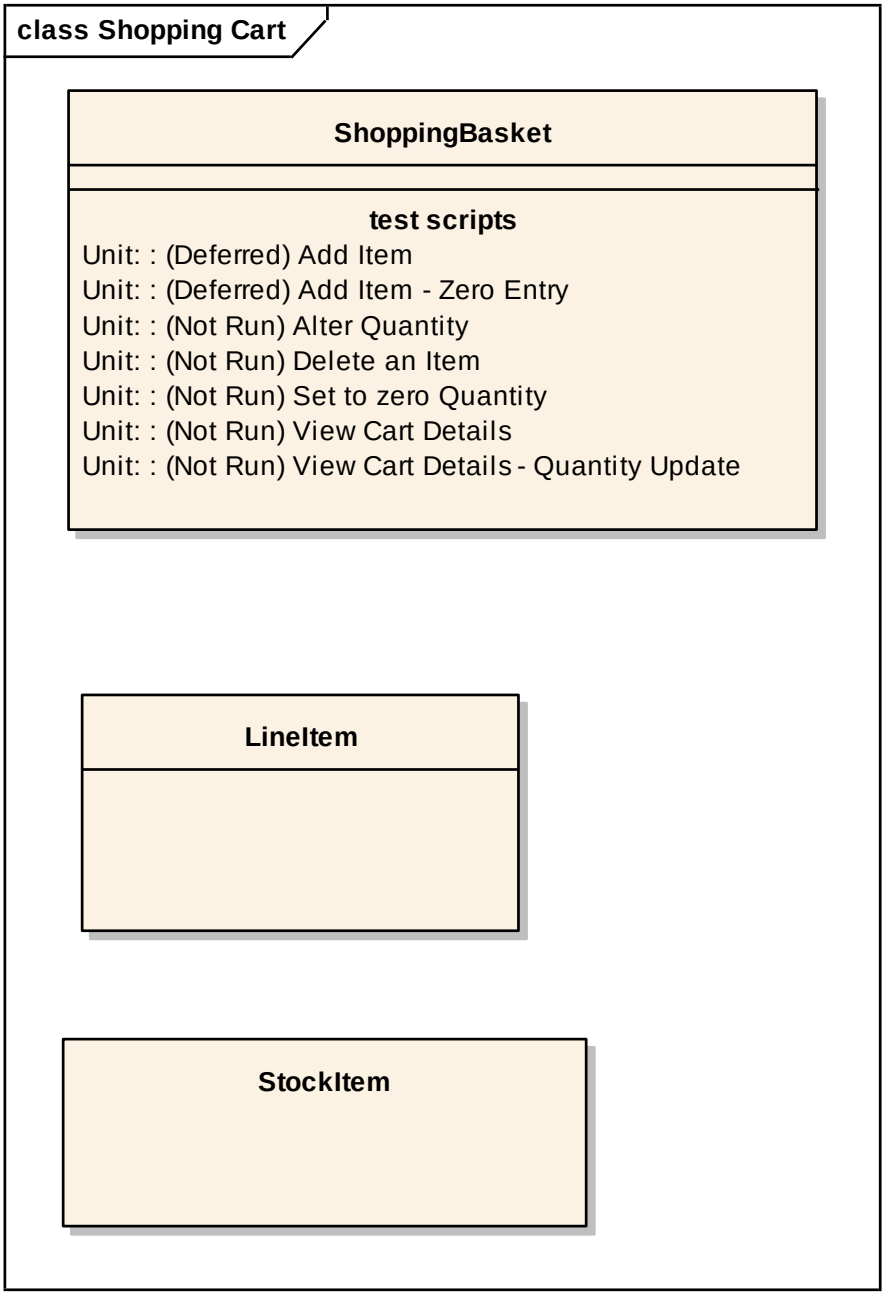


Diagram: User Interface

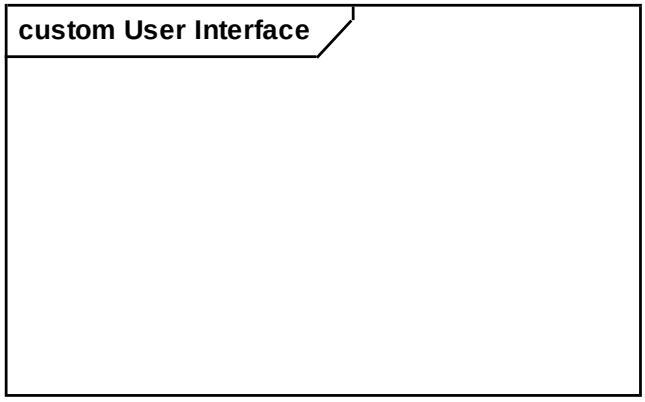


Diagram: Back End Services

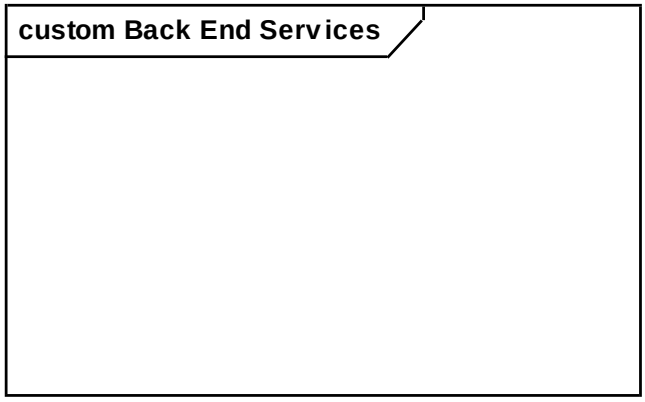


Diagram: Delivery Process

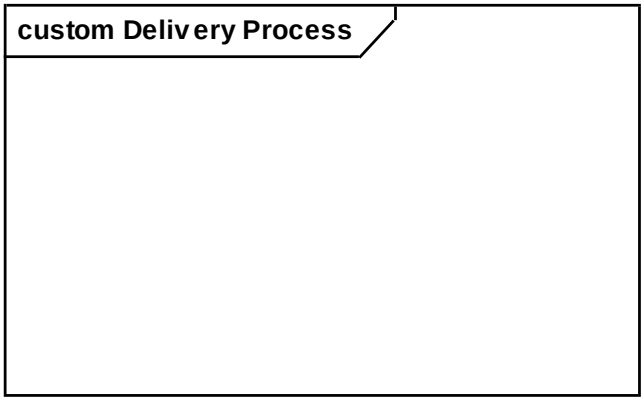


Diagram: Issues

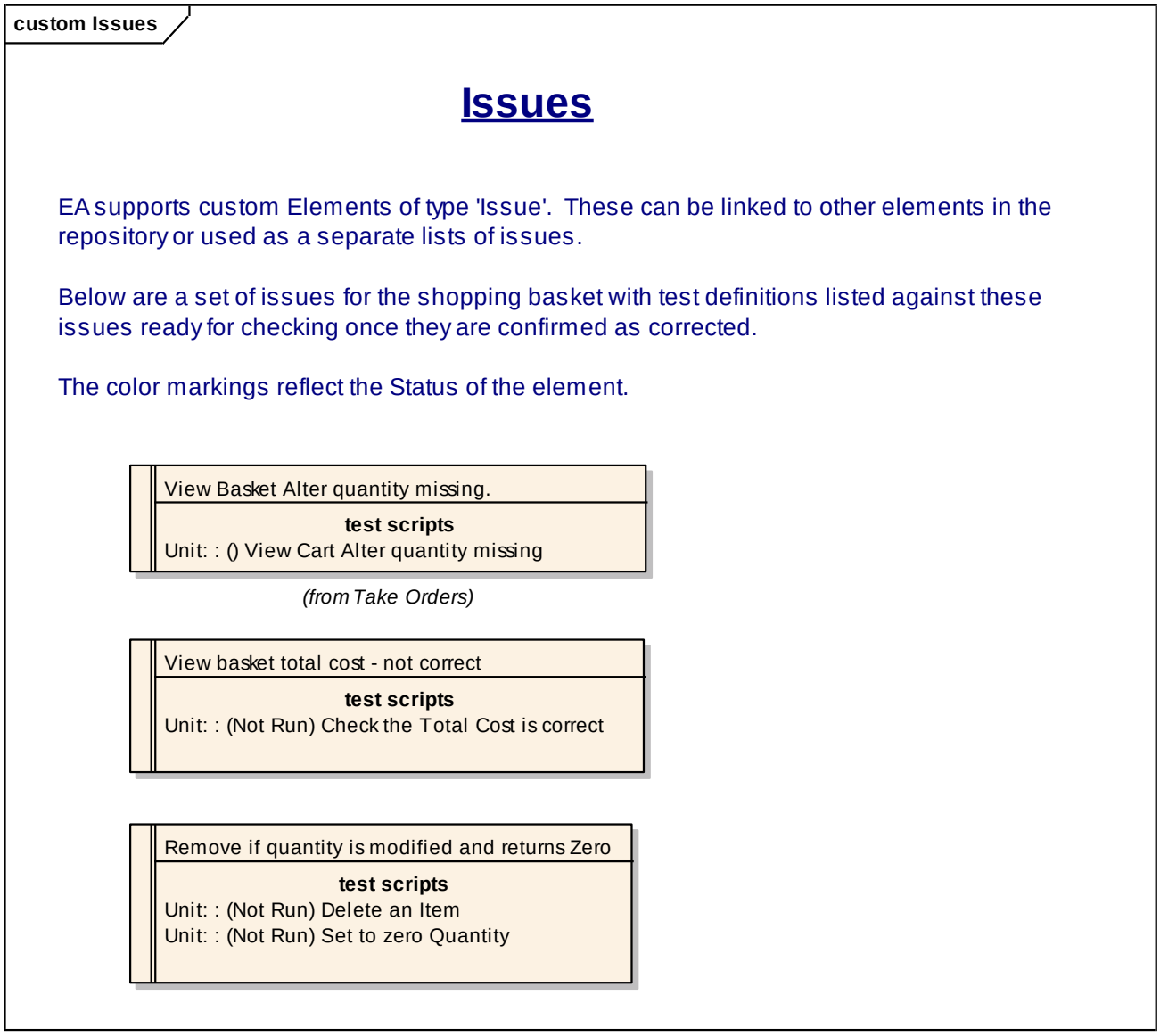


Diagram: Changes

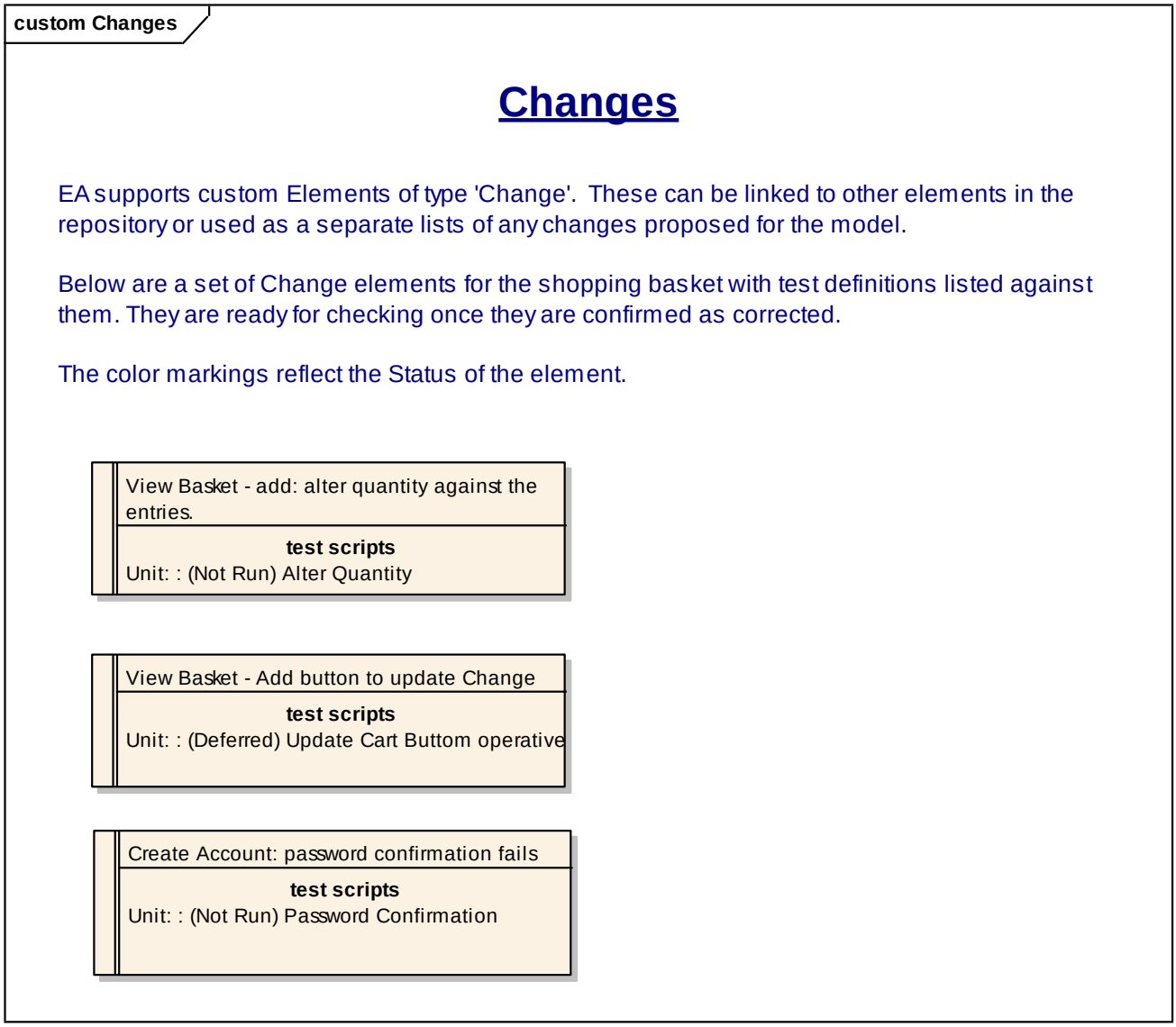


Diagram: Issues

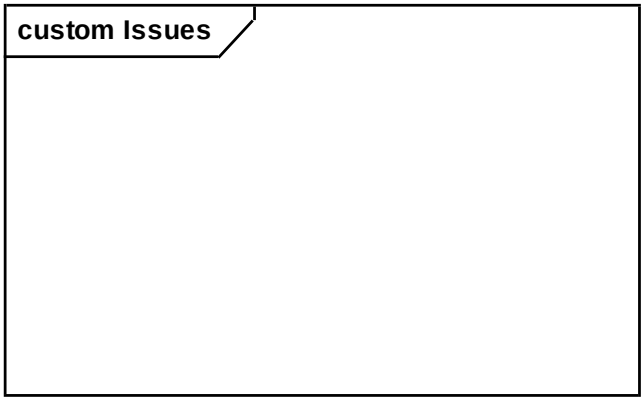


Diagram: Changes

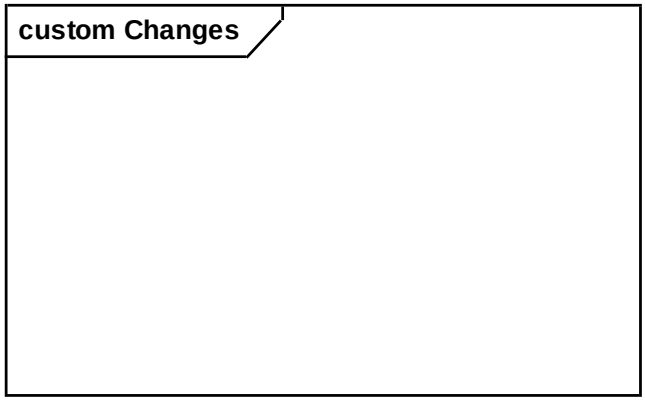


Diagram: Resources

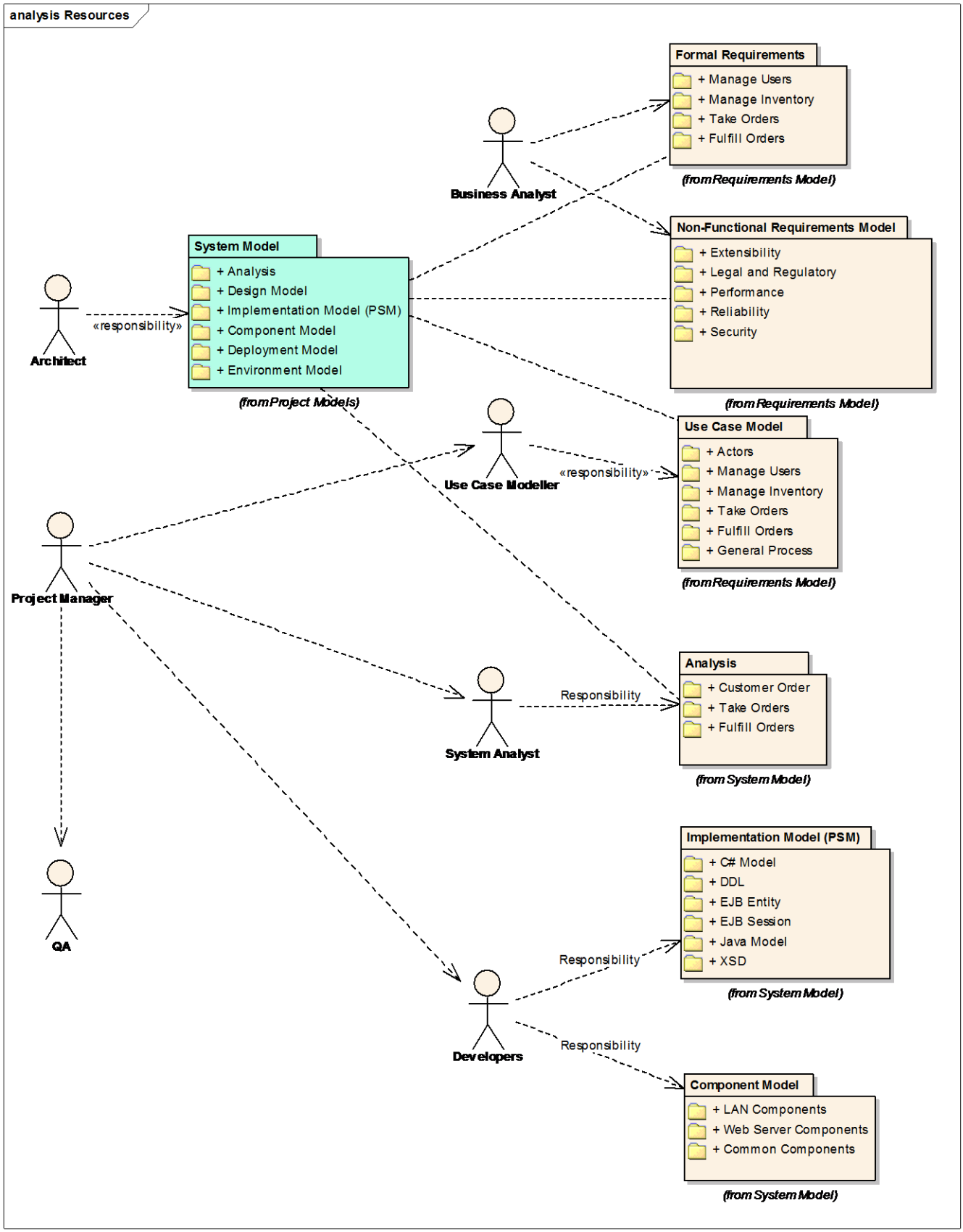

analysis Resources
Formal Requirements
+ Manage Users
+ Manage Inventory
+ Take Orders
+ Fulfill Orders
(from Requirements Model)
Business Analyst
Non-Functional Requirements Model
+ Extensibility
+ Legal and Regulatory
+ Performance
+ Reliability
+ Security
(from Requirements Model)
System Model
+ Analysis
+ Design Model
+ Implementation Model (PSM)
+ Component Model
+ Deployment Model
+ Environment Model
(from Project Models)
«responsibility»
Architect
Use Case Model
+ Actors
+ Manage Users
+ Manage Inventory
+ Take Orders
+ Fulfill Orders
+ General Process
(from Requirements Model)
«responsibility»
Use Case Modeller
Project Manager
Analysis
+ Customer Order
+ Take Orders
+ Fulfill Orders
(from System Model)
Responsibility
System Analyst
QA
Implementation Model (PSM)
+ C# Model
+ DDL
+ EJB Entity
+ EJB Session
+ Java Model
+ XSD
(from System Model)
Responsibility
Developers
Responsibility
Component Model
+ LAN Components
+ Web Server Components
+ Common Components
(from System Model)

Diagram: Resources Overview

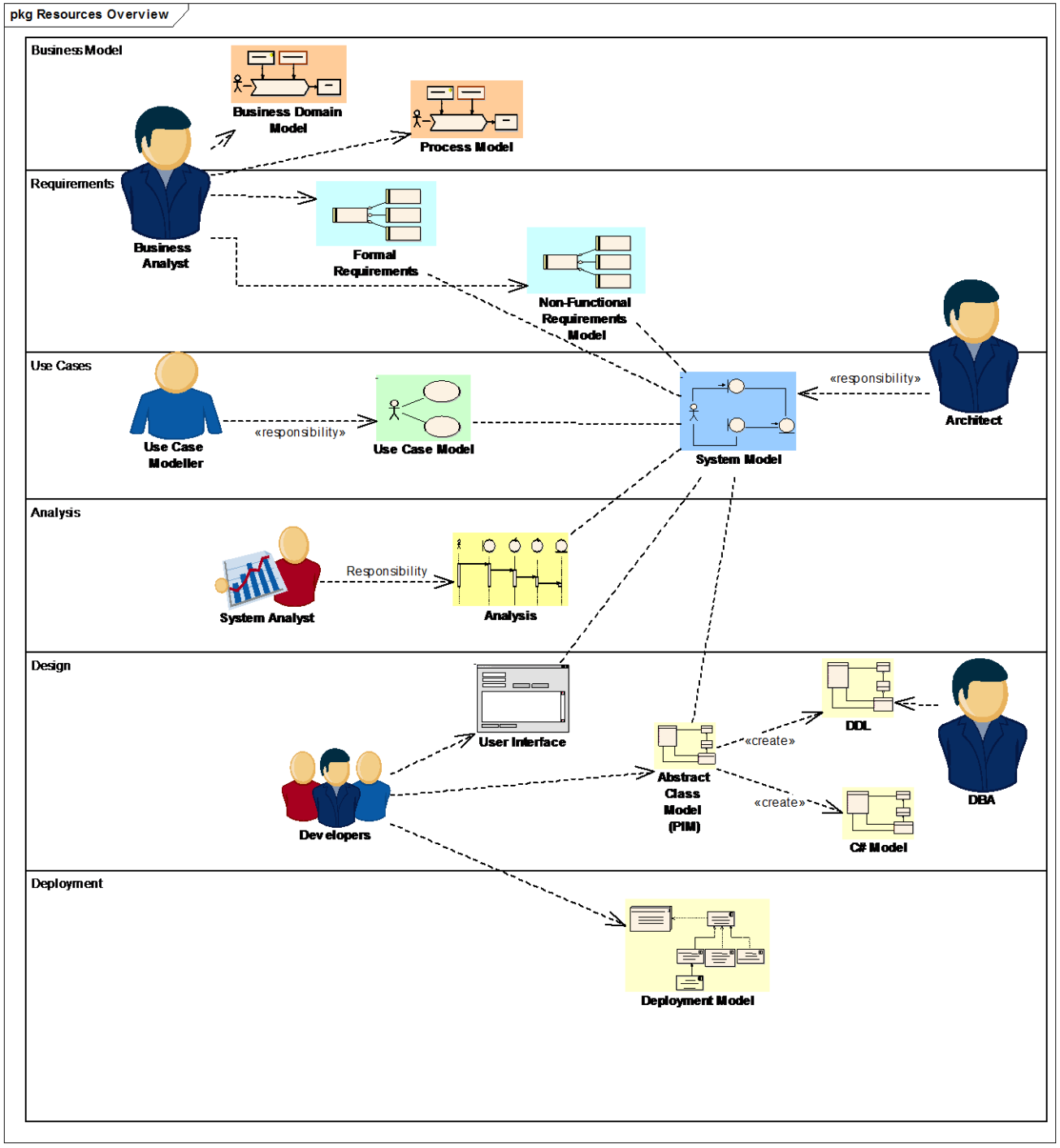


Diagram: Resource Allocation

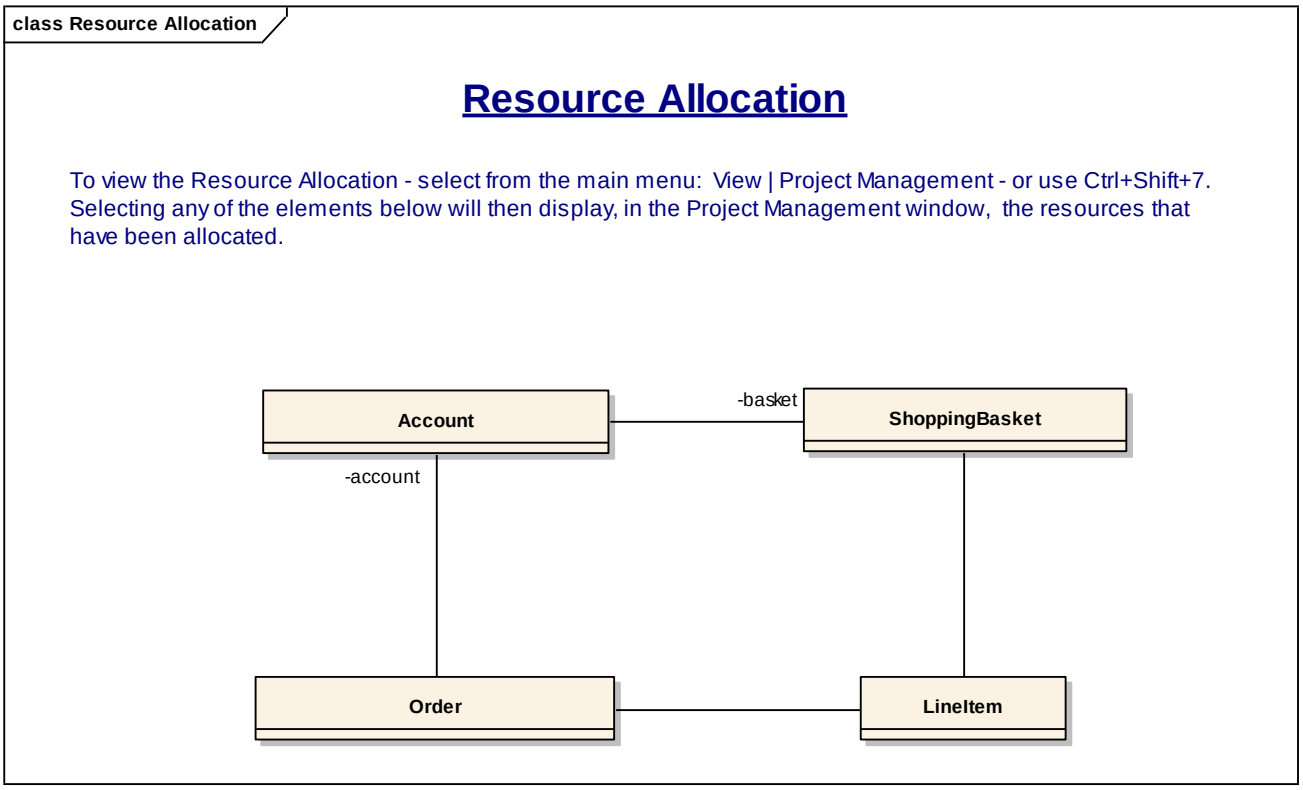


Diagram: Metrics and Estimation

class Metrics and Estimation

Metrics and Estimation

EA supports the definition of Use Case Metrics for elements defined in the Use Case model. For a set of Use Cases with Metrics set - see the following diagram:

 Manage Users

To set up the Use Case Metrics and Estimation open the properties for a Use Case element and set value for: Complexity [ ].

For defining the technical and environmental factors -TCF and ECF open from the main menu: Configuration | Metrics and Estimation Types.

For more details on creating Use Case Metrics see:

 http://www.sparxsystems.com.au/UCMetrics.htm

For reports on estimating the project size use the option on the main menu:
- Project | Use Case Metrics.

EA also supports the definition of Effort and Metrics for elements defined in the model. To access this use the main menu option: View | Project Management (or Ctrl+Shift+7).

Diagram: Risk

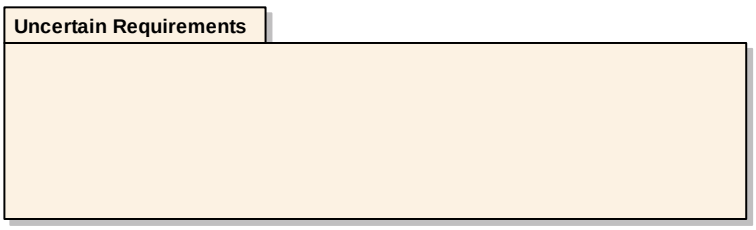
object Risk

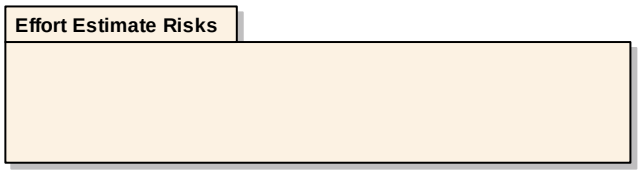
Risk

Below are the risks defined in the model. These are grouped in the packages related to the typical risk areas. The elements contained in these are linked elements defined in the source packages.

To view the risk definitions- from the main Menu select:
- View | Project Management (Ctrl+Shift+7).
- then select the Risk tab on the base of the window.

- Open a package below (double-click on it) then select an Element from the diagram to view its risk definition.





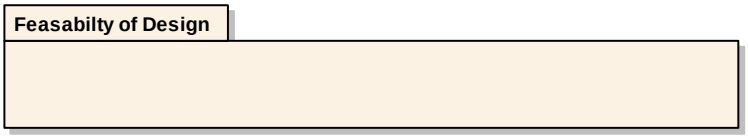


Diagram: Uncertain Requirements

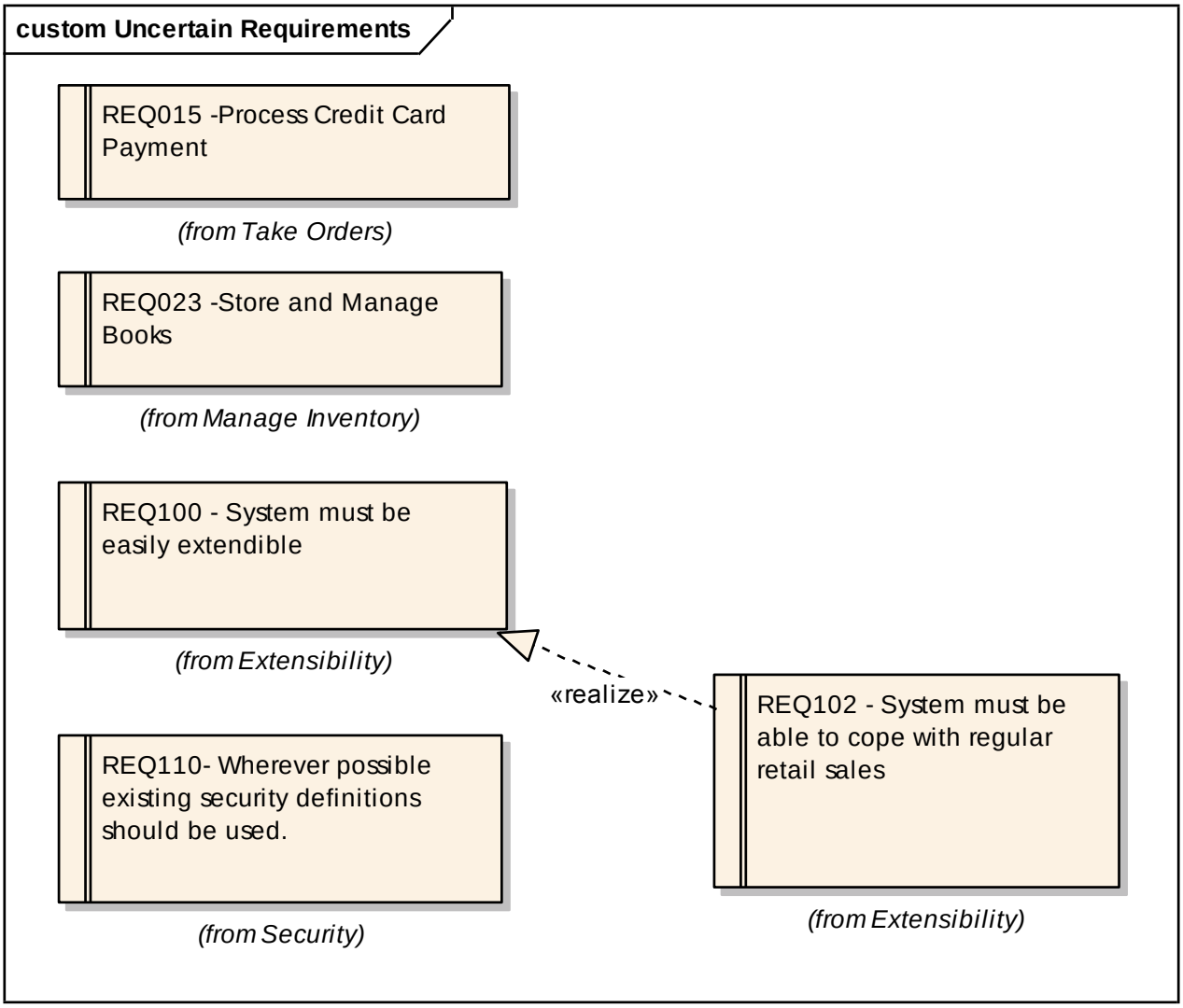


Diagram: Effort Estimate Risks

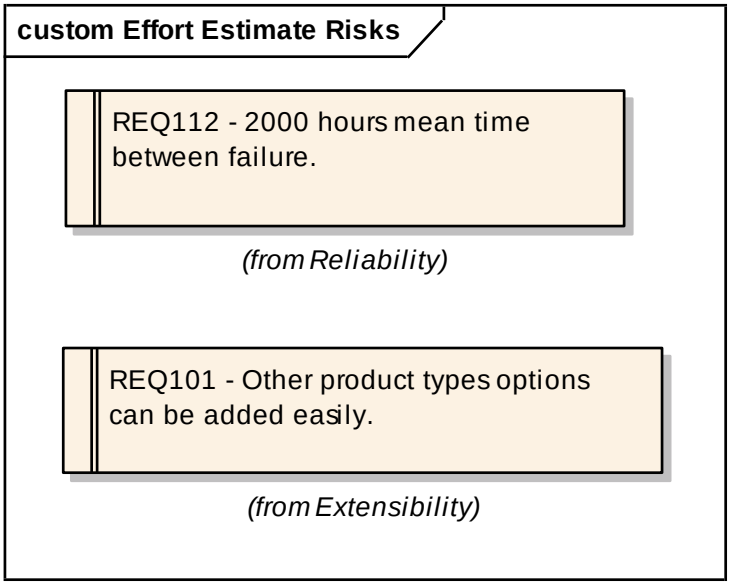


Diagram: Feasabilty of Design

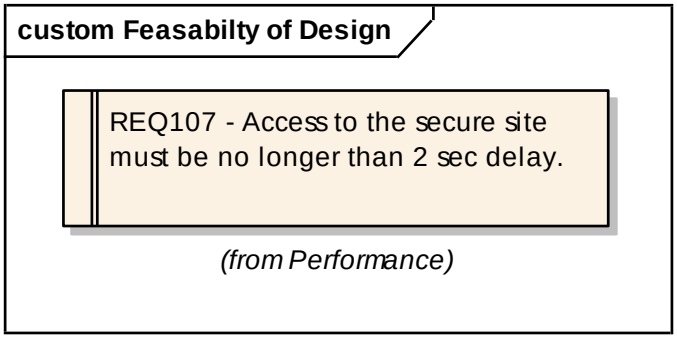


Diagram: Custom Diagrams

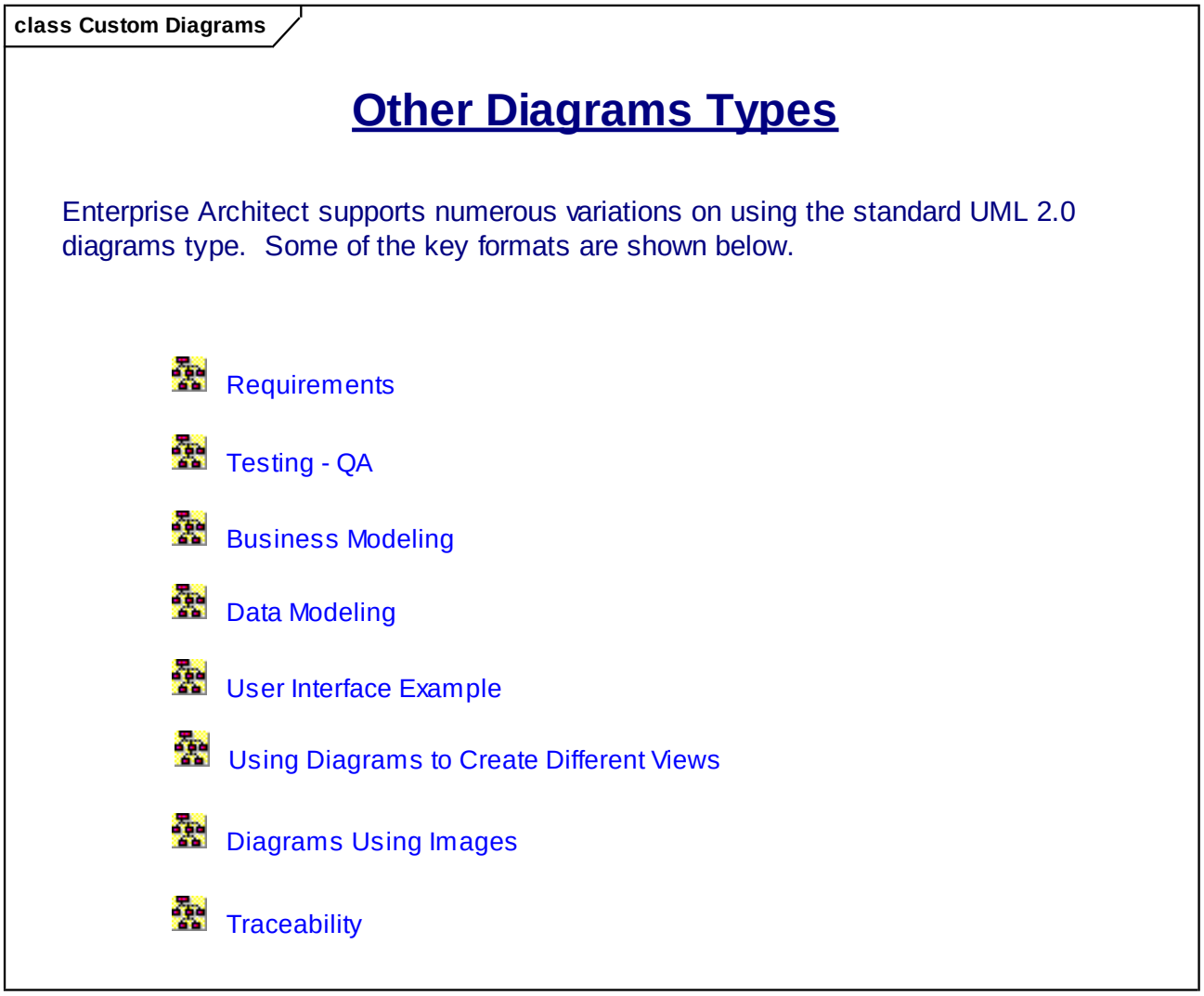


Diagram: Diagram using Images

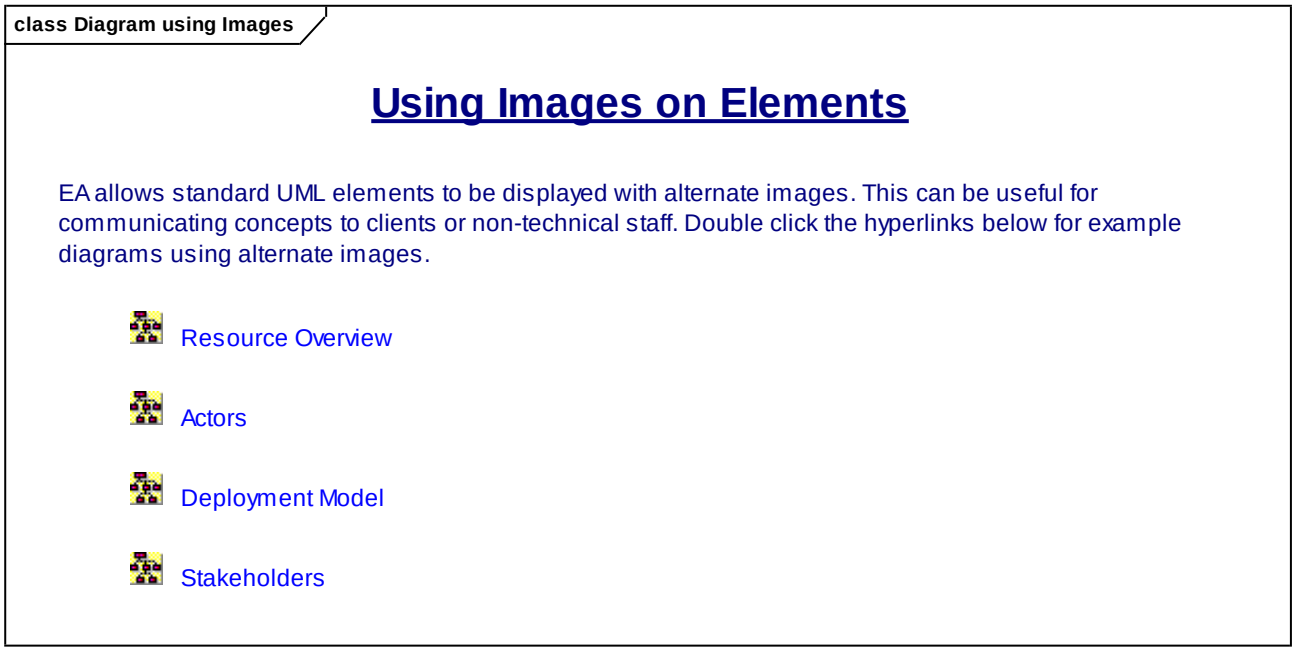


Diagram: Model Overview

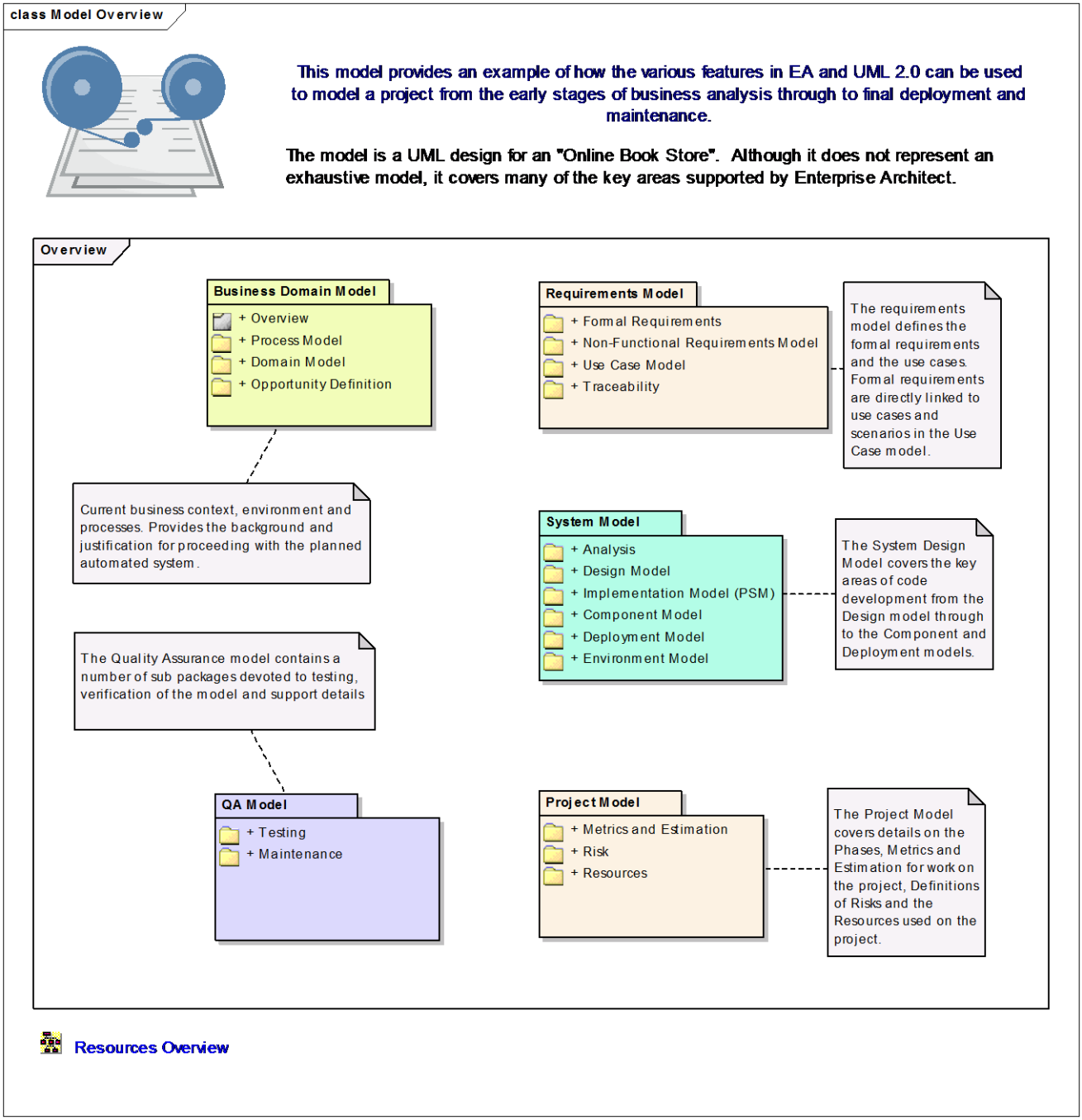


Diagram: UML 2.0 Diagrams

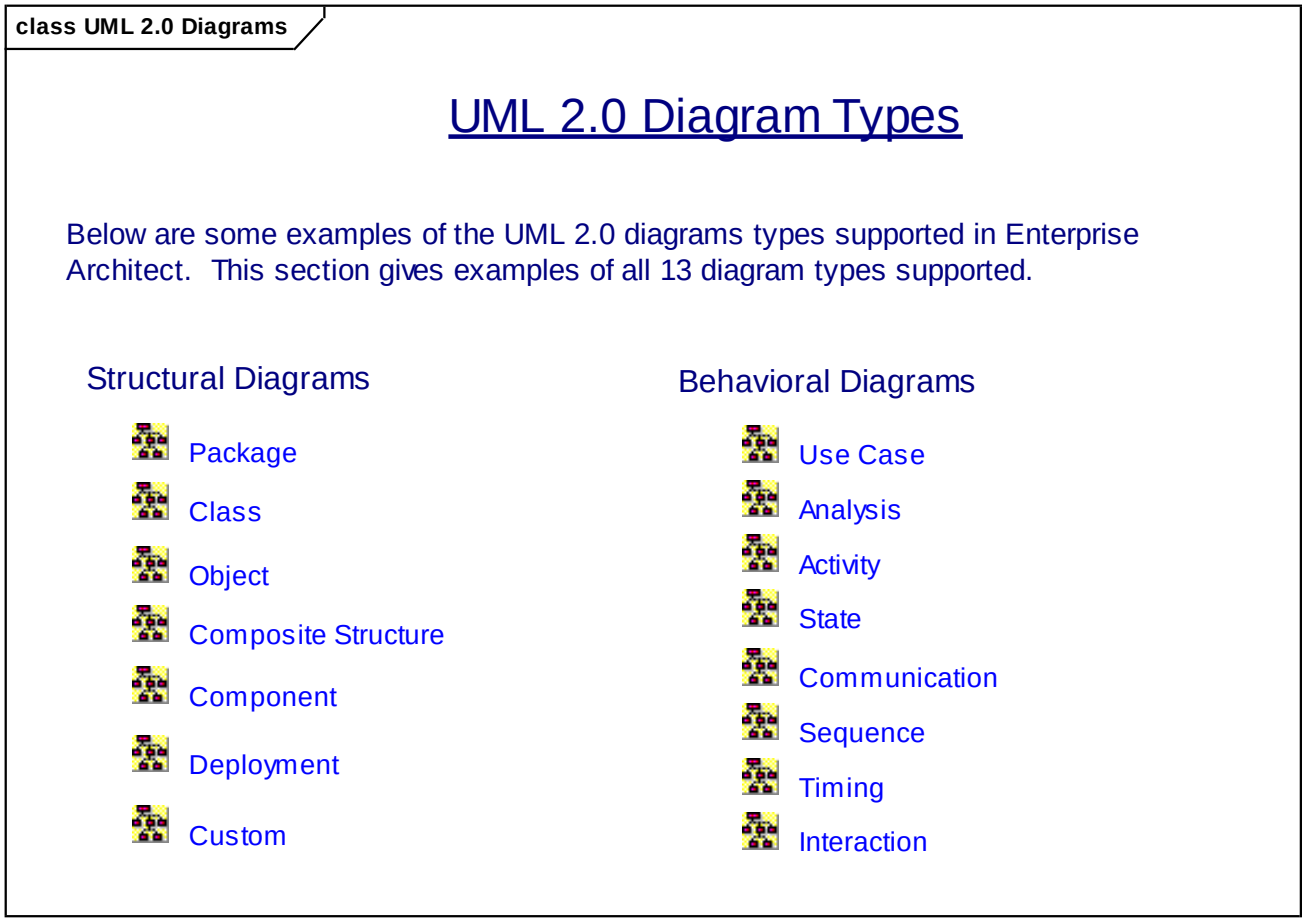


Diagram: Welcome

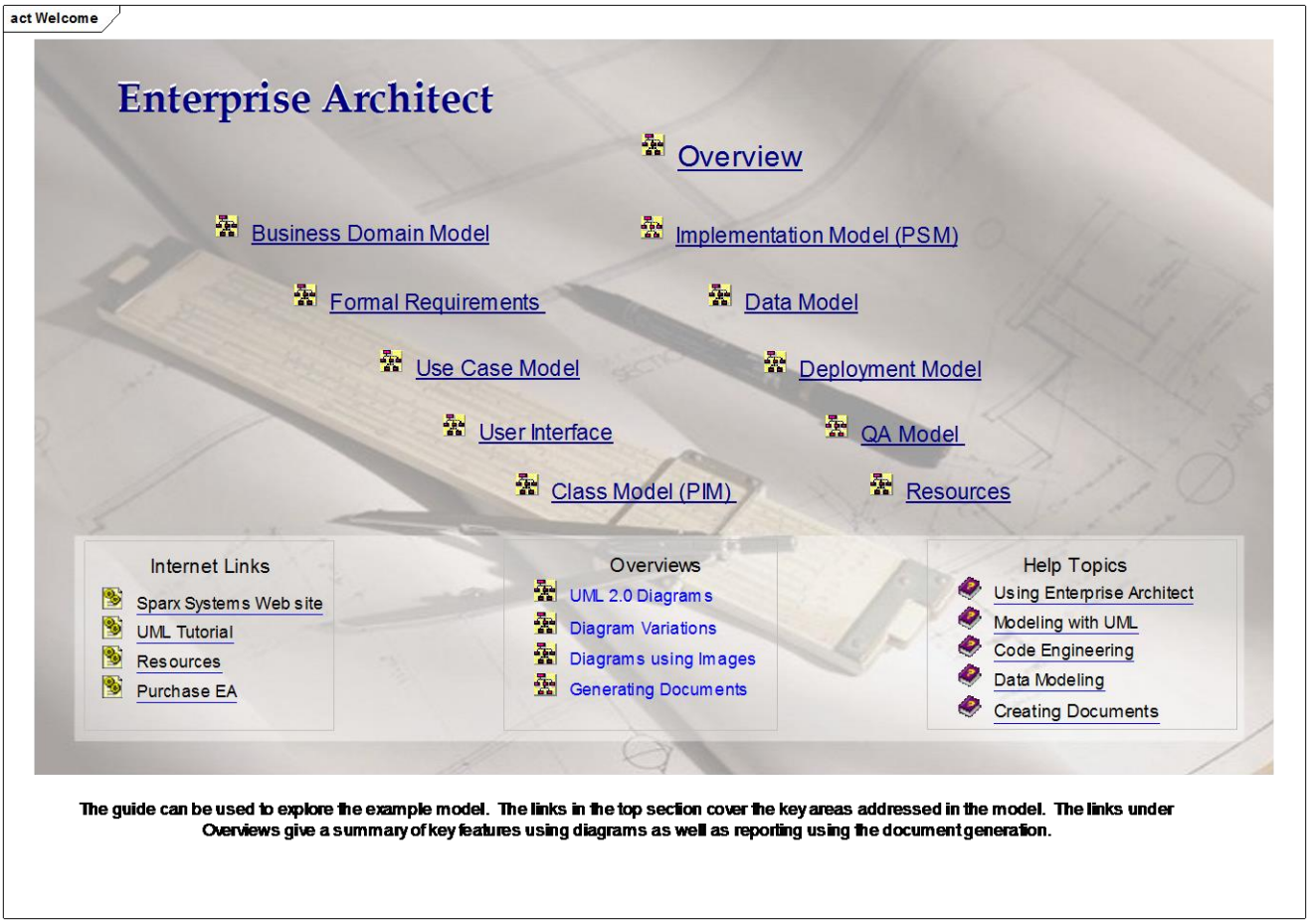


Diagram: Activity Diagrams

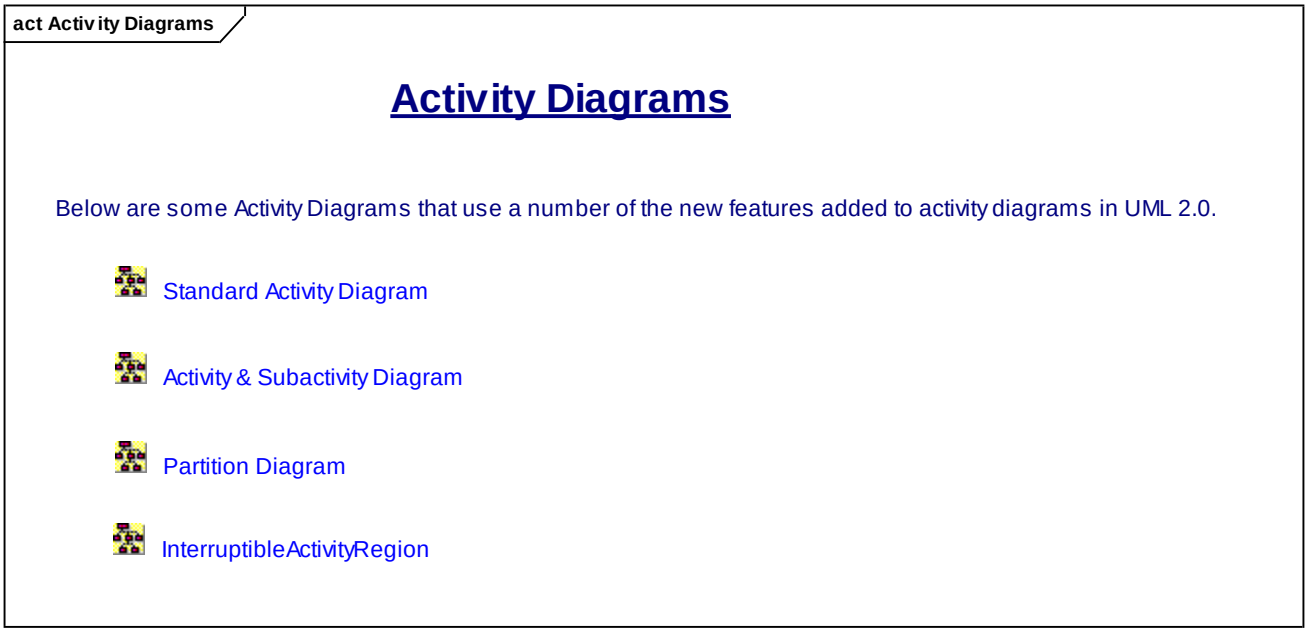


Diagram: Class diagrams

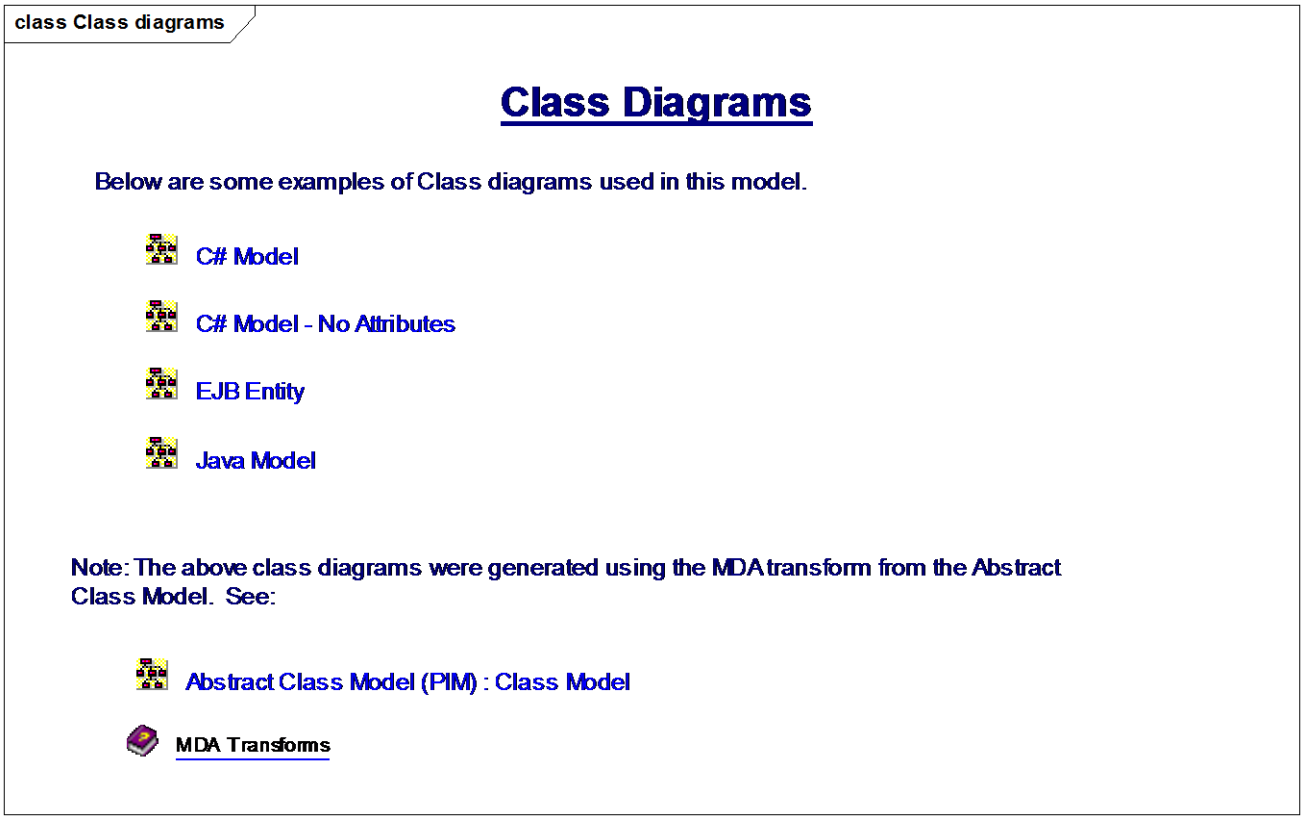


Diagram: Communication diagrams

sd Communication diagrams

Communication Diagrams

Below are some examples of Communication Diagrams as used in the Use Case model.

 Create Account

 Delete User

 View Account details

 View History

 Login

 Close Account

Diagram: Component Diagrams

cmp Component Diagrams

Component Diagrams

Below are some examples of Component Diagrams used in this model.

 LAN Components

 Web Server Components

Diagram: Custom Diagrams

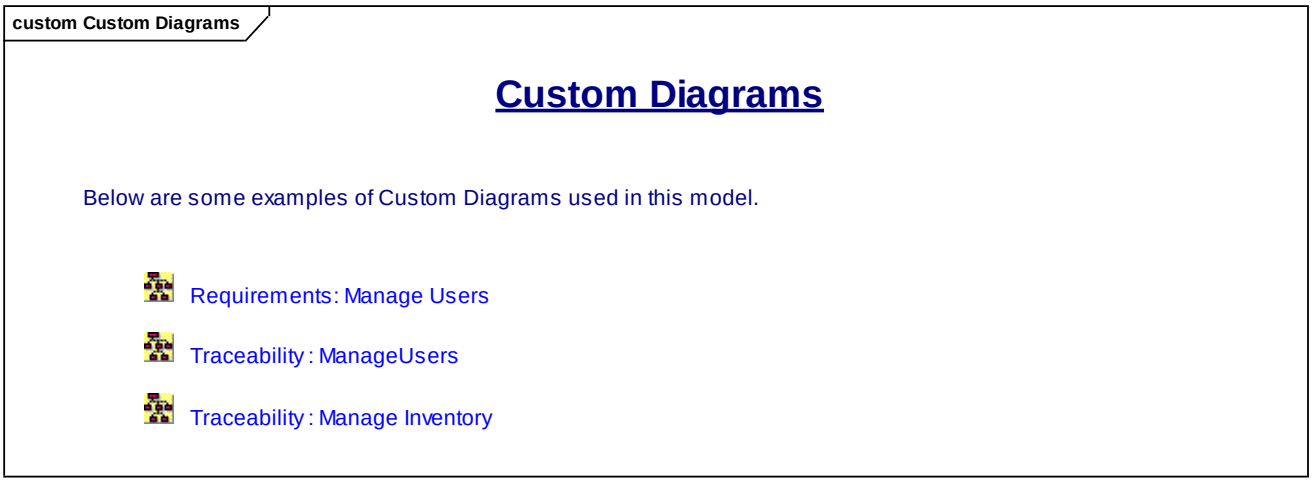


Diagram: Diagram Settings

pkg Diagram Settings

Diagram Settings

Diagrams can be created with a number of different settings for displaying the same information. Below are some examples of this.

Class Diagram with Attributes and Operations Viewable

Class diagram with the Attributes and operations not shown.

Class Diagram only showing the Testing .

To alter these settings bring up the diagram properties using the option from the main menu:
- Diagram | Properties
- or right-click on the diagram and from the context menu select: Properties.

The Diagram Properties window includes - Appearance Options (see below). This displays options for showing/hiding the properties viewed on the current diagram.

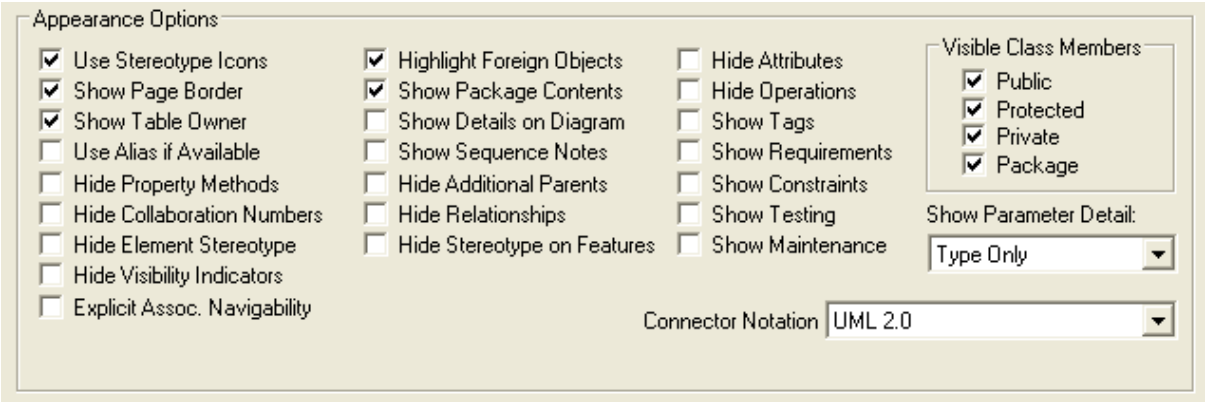


Diagram: Interaction diagrams

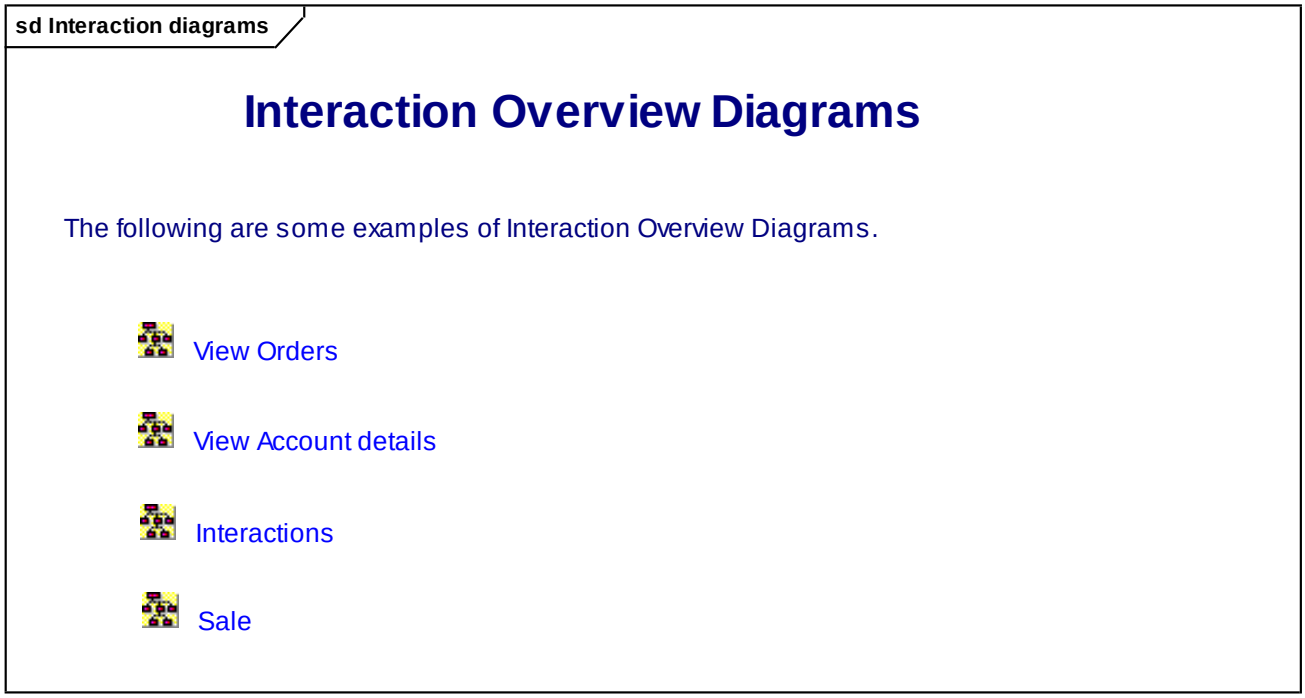


Diagram: Package Diagrams

pkg Package Diagrams

Package Diagrams

Below are some examples of Package diagrams used in this model:

 Business Domain Model

 Formal Requirements

 EJB Entity

Diagram: Sequence Diagrams

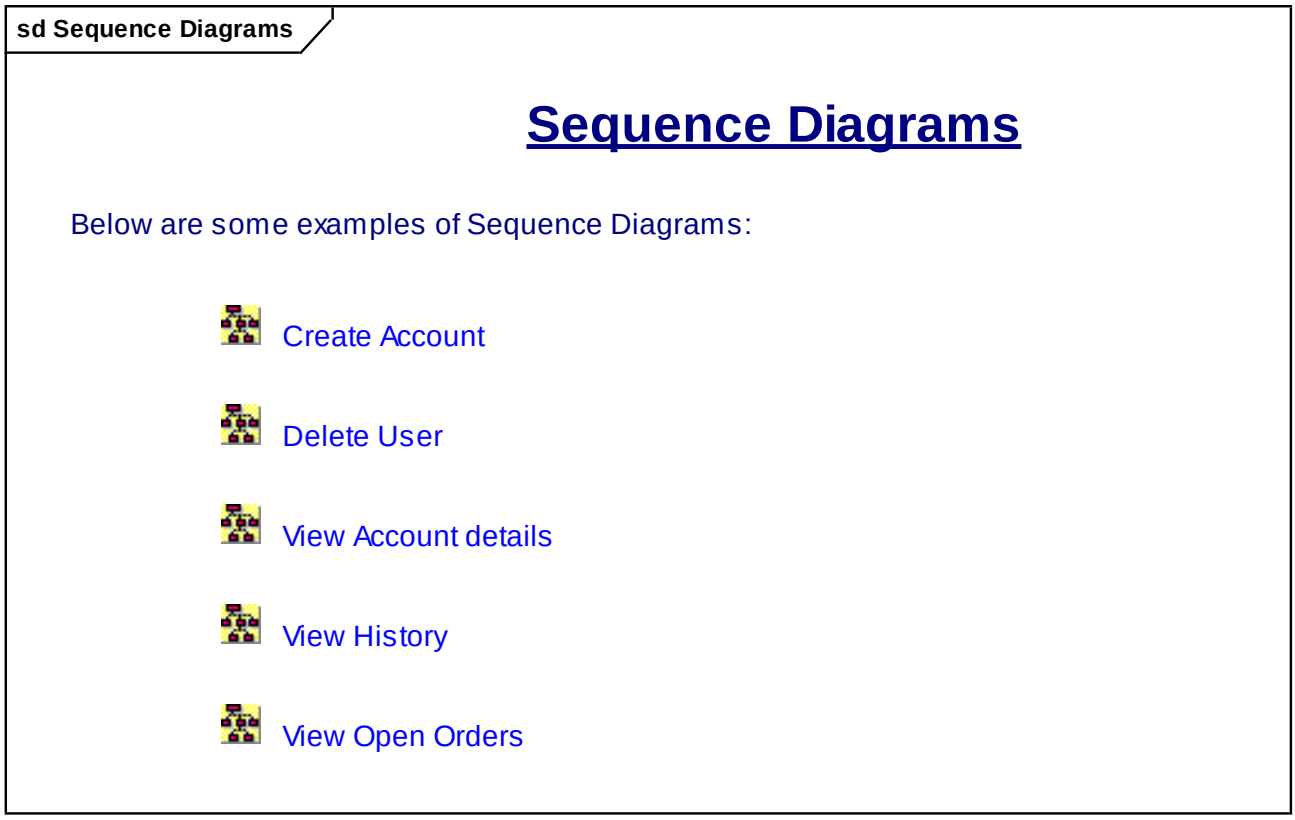


Diagram: State Diagrams

pkg State Diagrams

State Chart Diagrams

Below are some examples of Statechart Diagrams used in the model:

 Login

 Manage Titles State

 Search

Diagram: Timing

sd Timing

Timing Diagrams

Enterprise Architect supports the UML 2.0 Timing Diagram. There are two lifelines defined in UML 2.0 - the State Lifeline and the Value Lifeline. The following are examples of these:

 State Lifeline

 Value Lifeline

These elements can be combined within a single diagram:

 State and Value combined

Diagram: Use Case diagrams

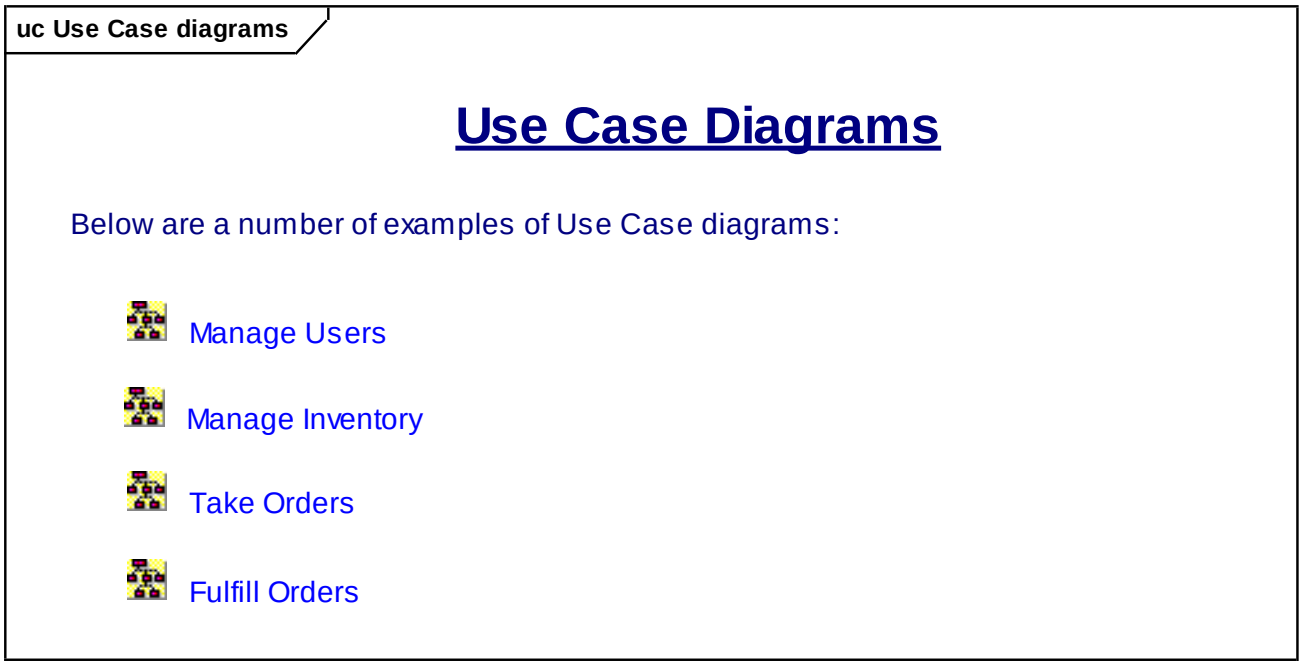


Diagram: State Lifeline

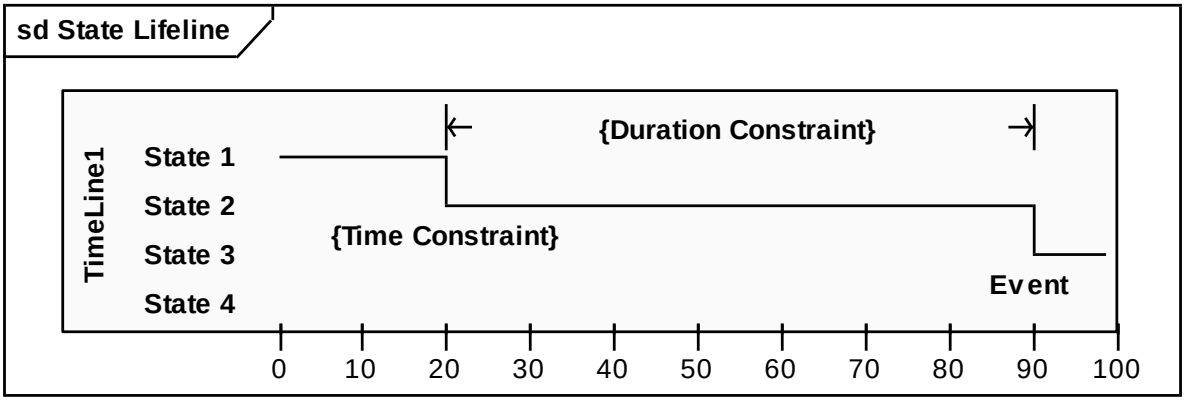


Diagram: Timing Diagram

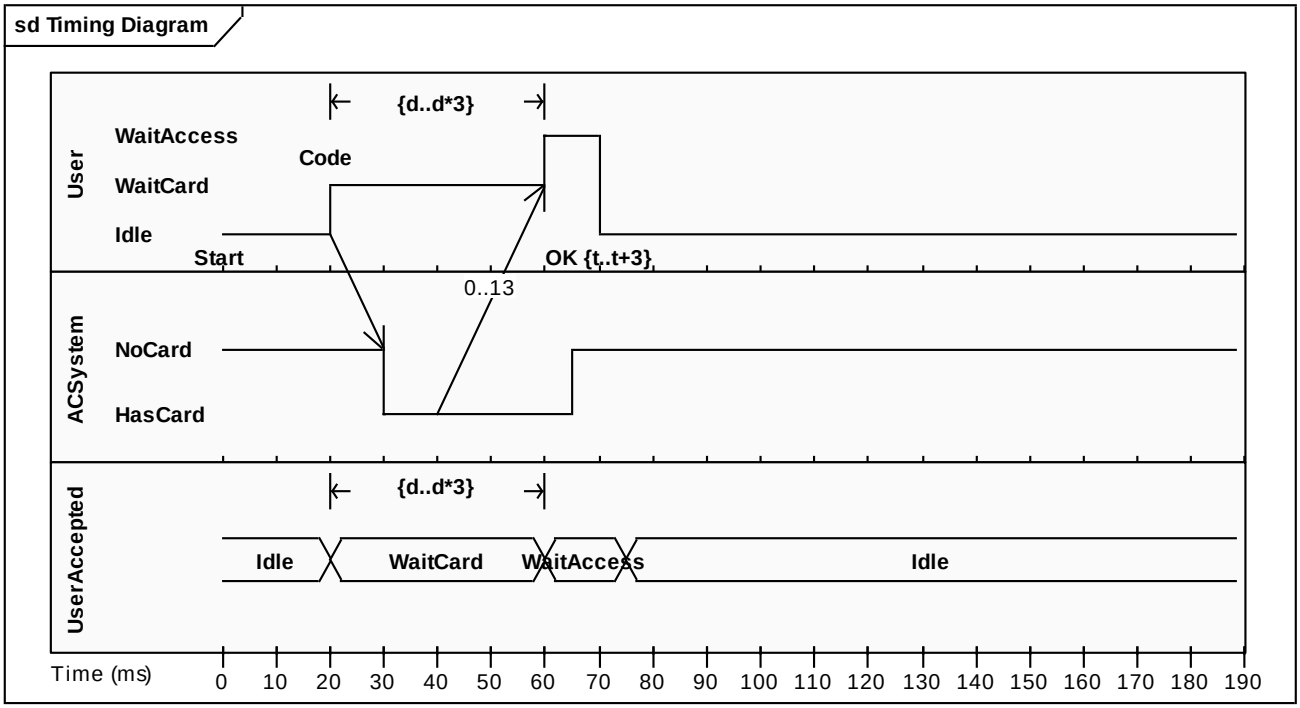


Diagram: Value Lifeline

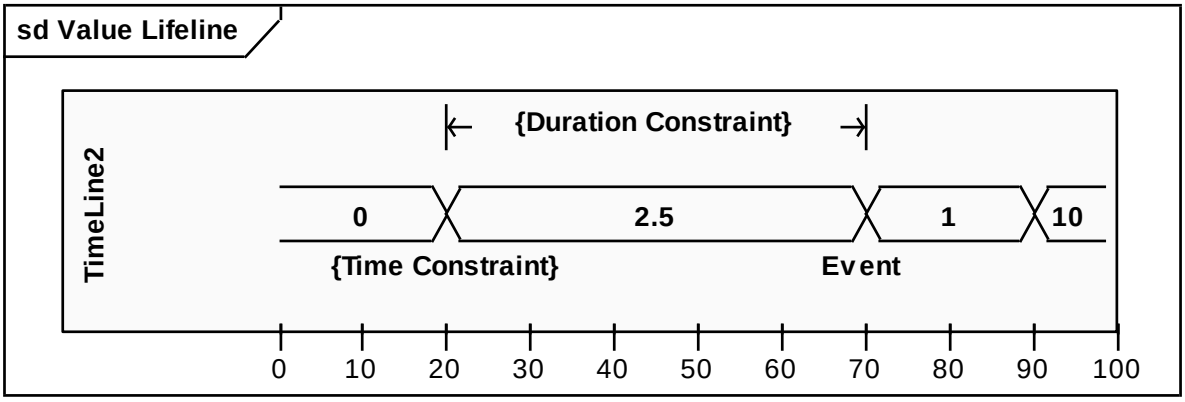


Diagram: CancelSale

Diagram: Checkout

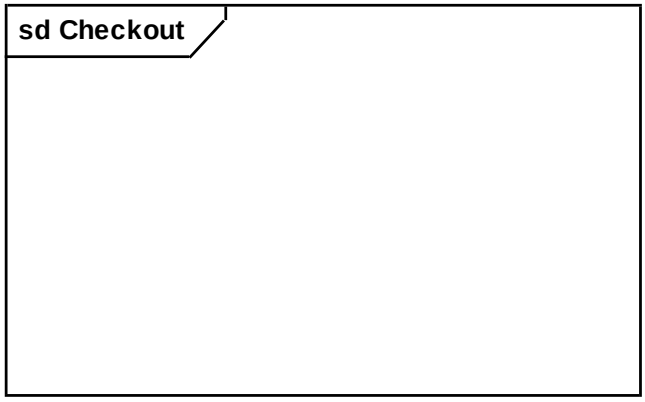


Diagram: CreateRecord

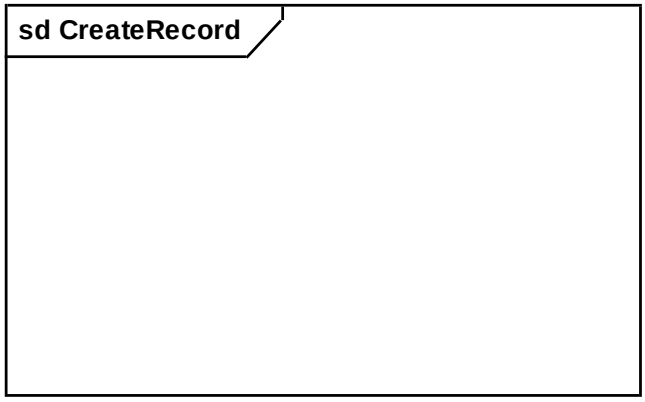


Diagram: Sale

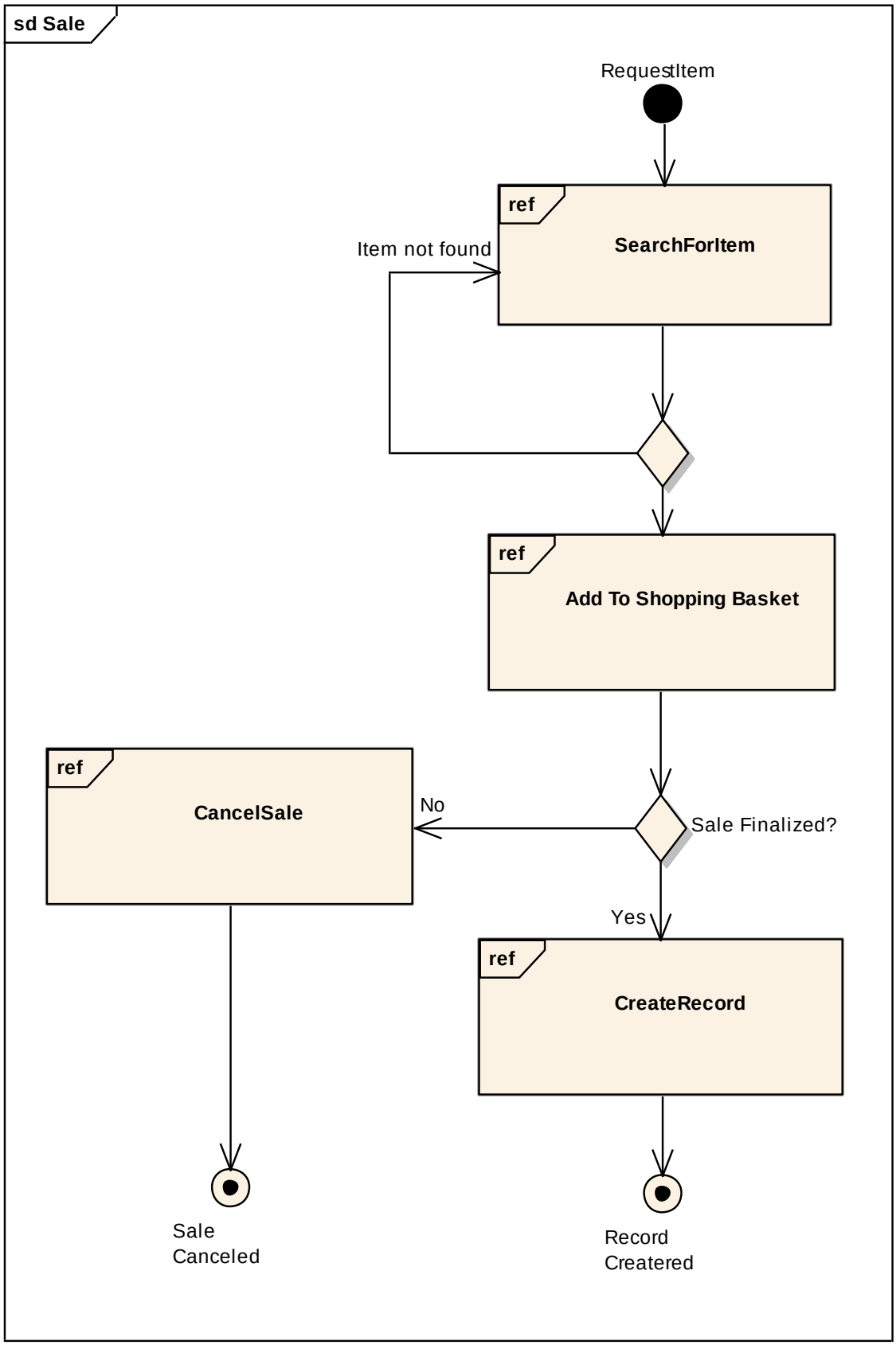

sd Sale
RequestItem
ref
SearchForItem
Item not found
ref
Add To Shopping Basket
ref
CancelSale
No
Sale Finalized?
Yes
ref
CreateRecord
Sale
Canceled
Record
Createred

Diagram: Fragment

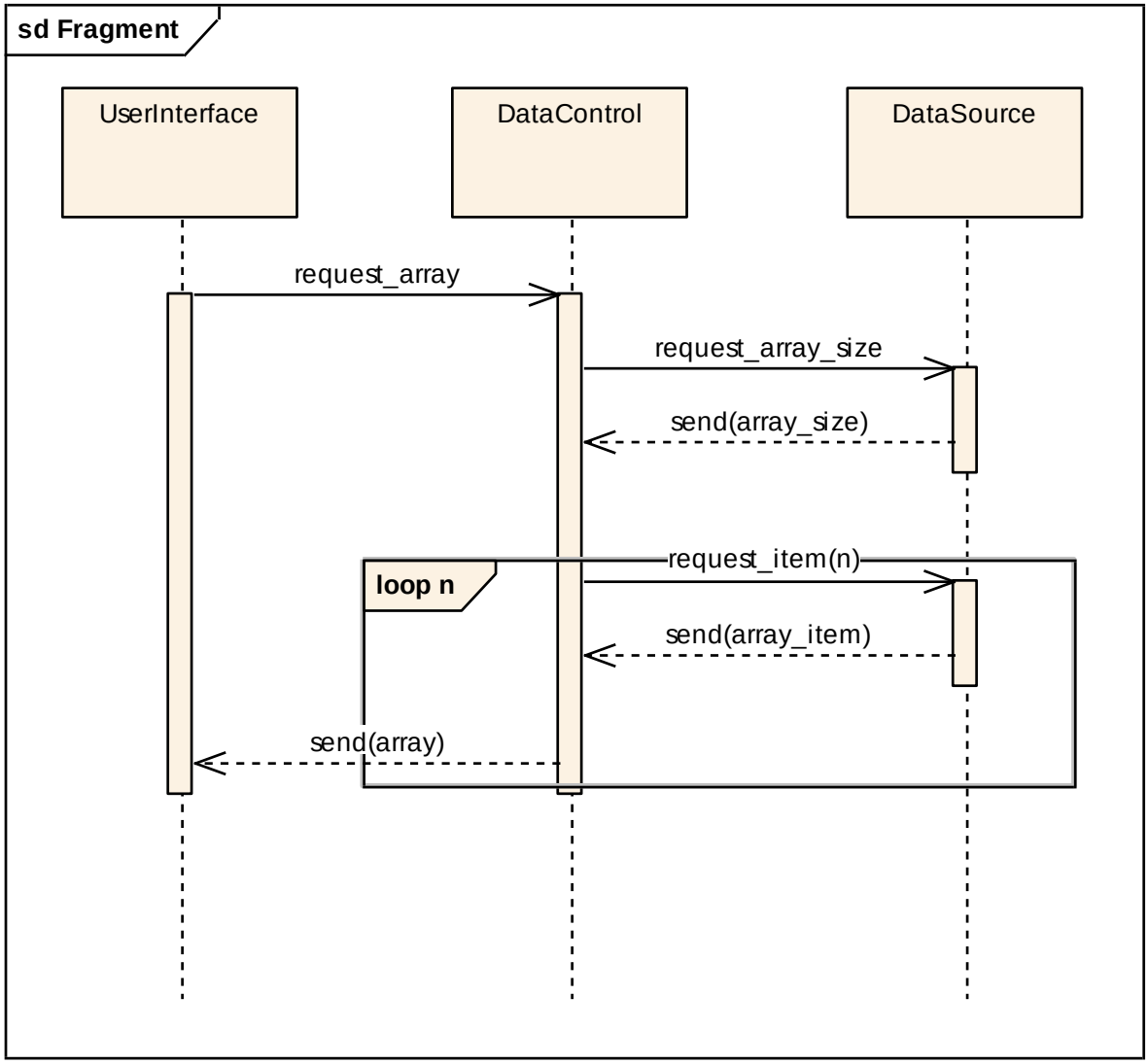


Diagram: Composite Structure

composite structure Composite Structure

Composite Structure Diagrams

Below are two examples of composite structure diagrams, one giving an example of the CollaborationOcurrence element, the other giving an example of Property.

 Composite Structure - CollaborationOccurrence

 Composite Structure - Properties

Diagram: Composite Structure-Properties

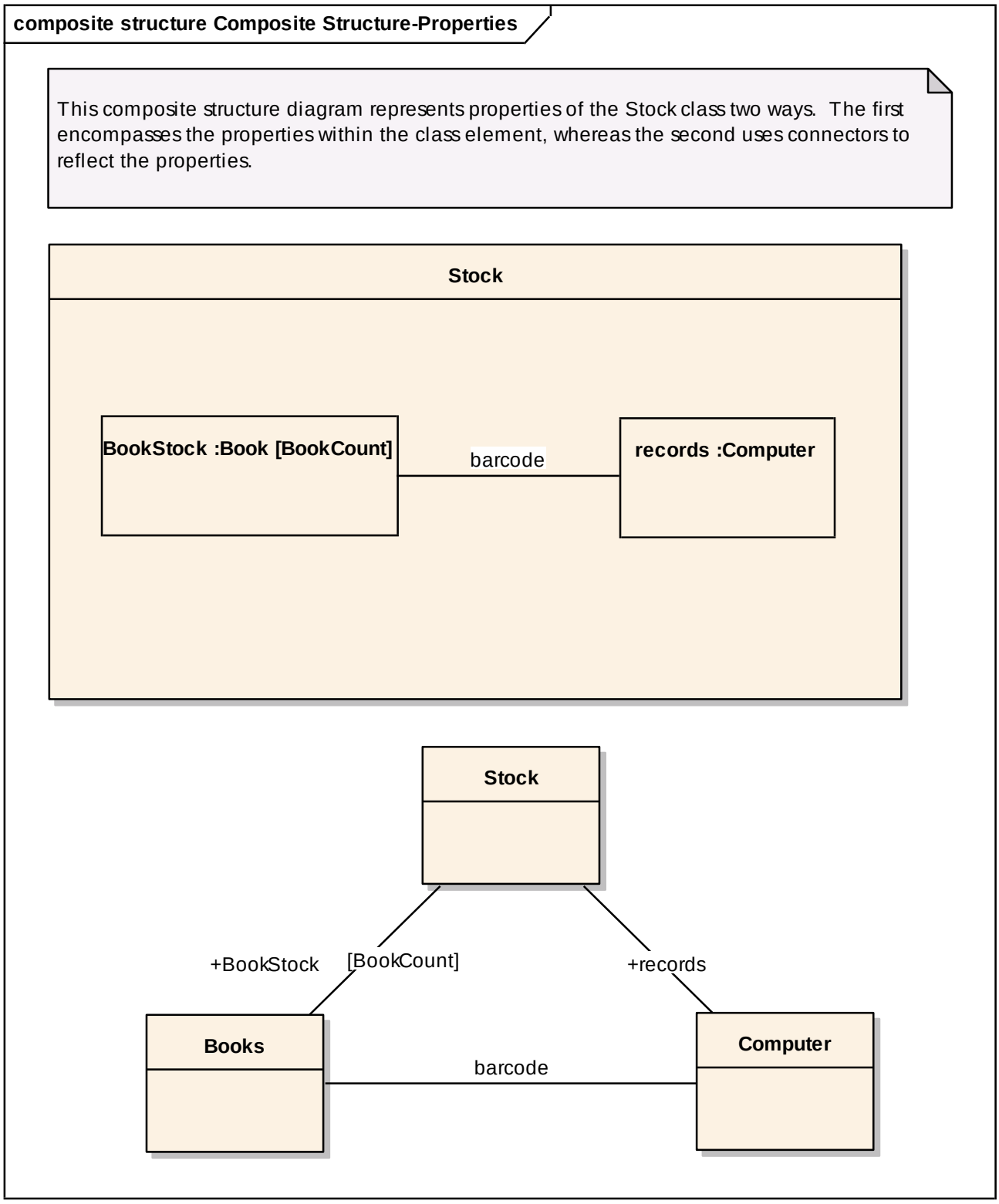


Diagram: Sale

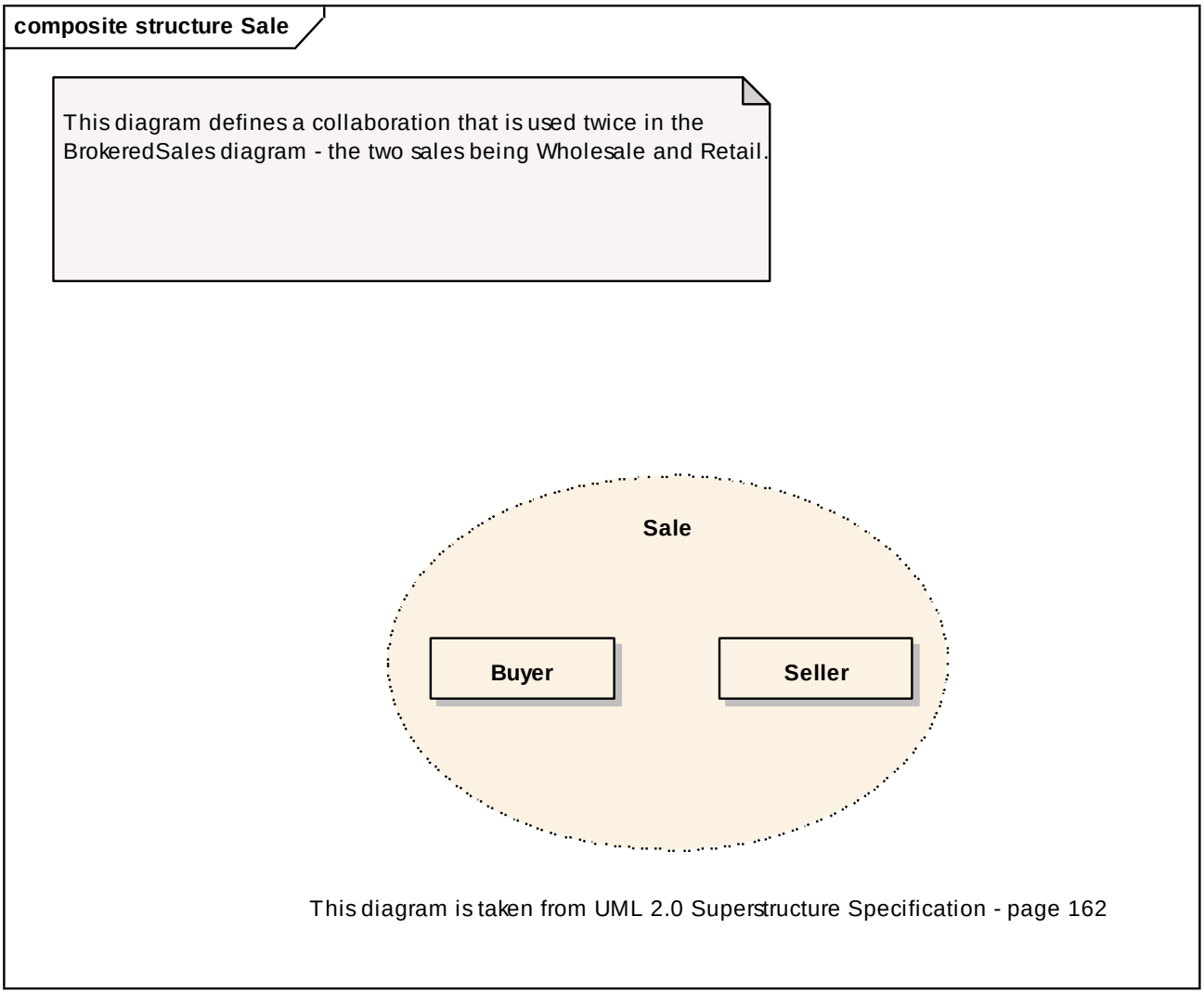


Diagram: brokered

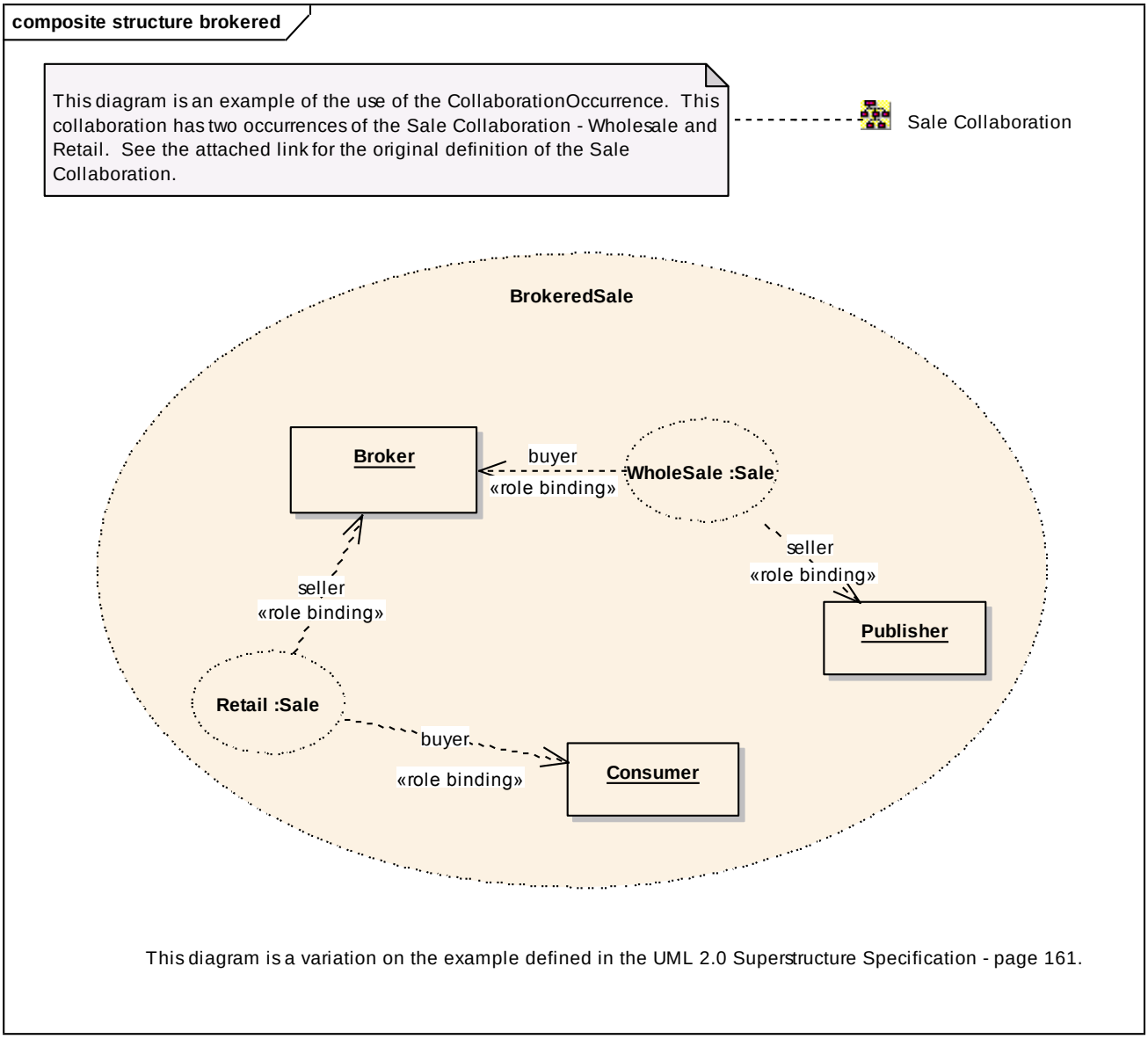


Diagram: Object

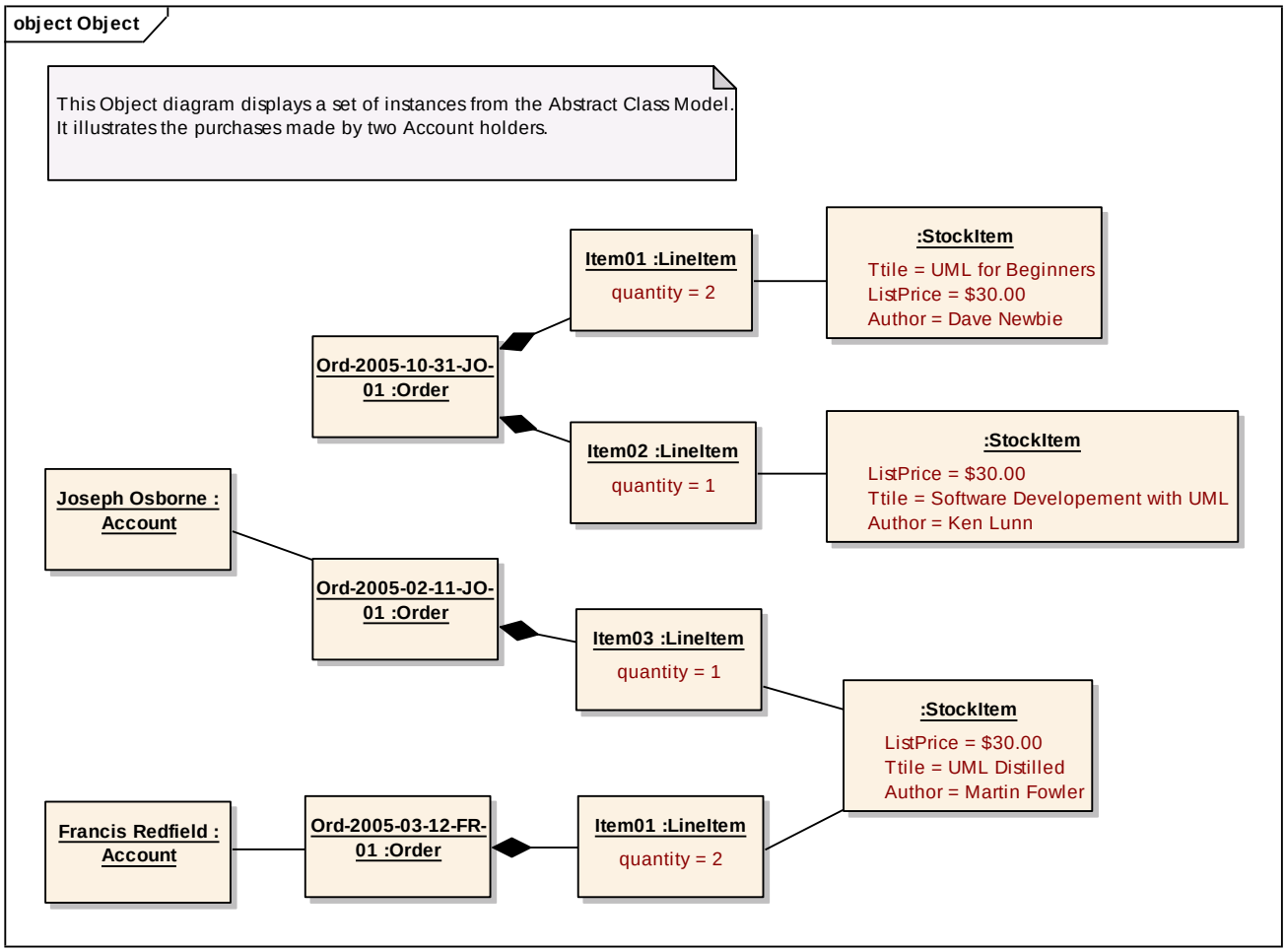


Diagram: Documentation

custom Documentation

Guide to Using EA's Documentation

Enterprise Architect supports the creation of professional and completely customized documentation directly from the application using both RTF report generation and HTML generation.

There are a number of different methods for running reports. Details for these are as follows:

Standard RTF reports:

Standard RTF reports are user definable and generate a report on a package tree:

Details and Examples of Standard RTF Documents

HTML Reports

Enterprise Architect supports the generation of HTML reports based on a package tree:

Details on HTML Reports

Virtual Documents

Virtual Documents allow packages, unrelated in the project tree, to be grouped into a single RTF report.

Details and Examples of Virtual Documents

Word Master Documents

EA supports book-marking Elements and Packages in reports. These can used with Word to group parts of different reports into a master document.

Details on Using Word Master Documents

RTF Reports Using Search Results

The results of a search can be used to create an RTF document.

Details on Creating Reports from Search Results

Diagram: HTML reports

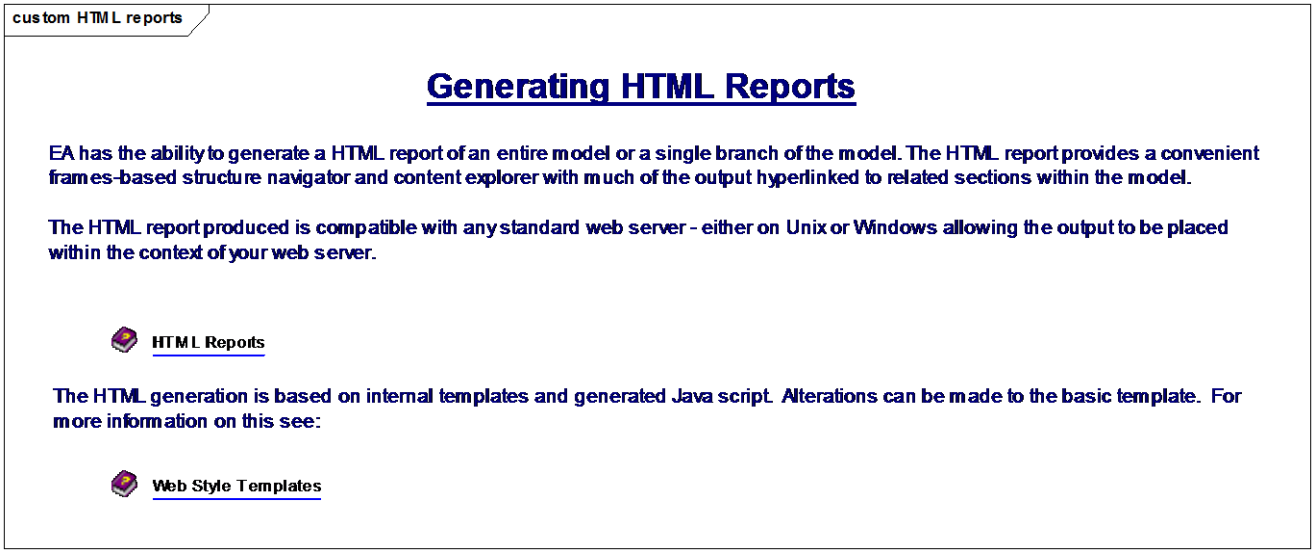


Diagram: Reports from Search Results

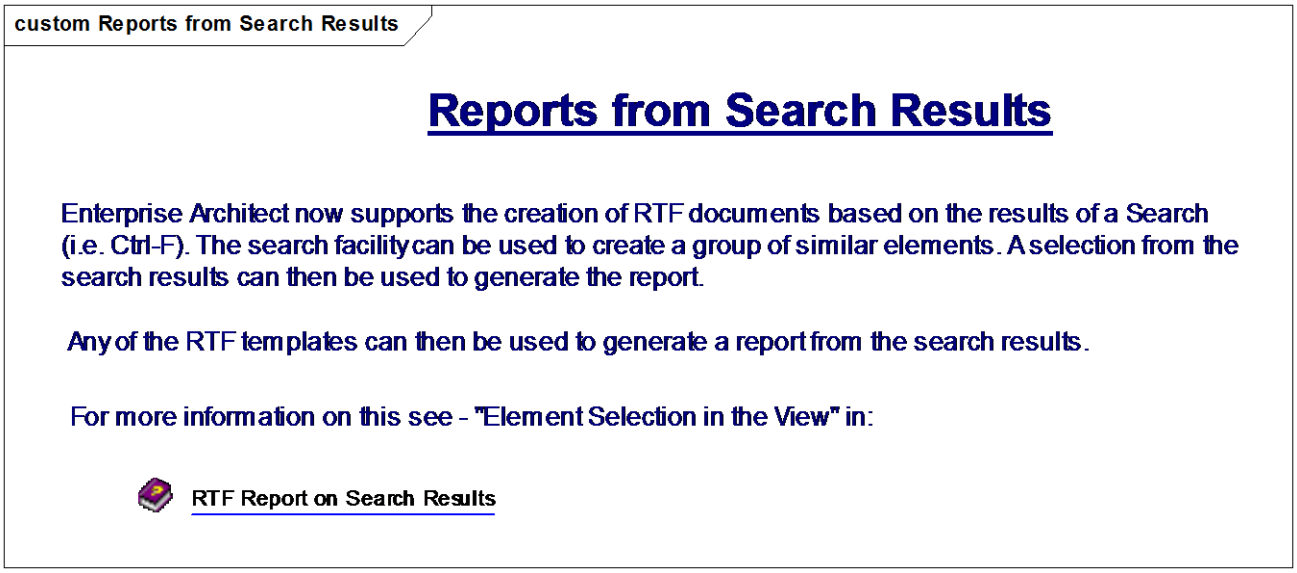


Diagram: Standard RTF

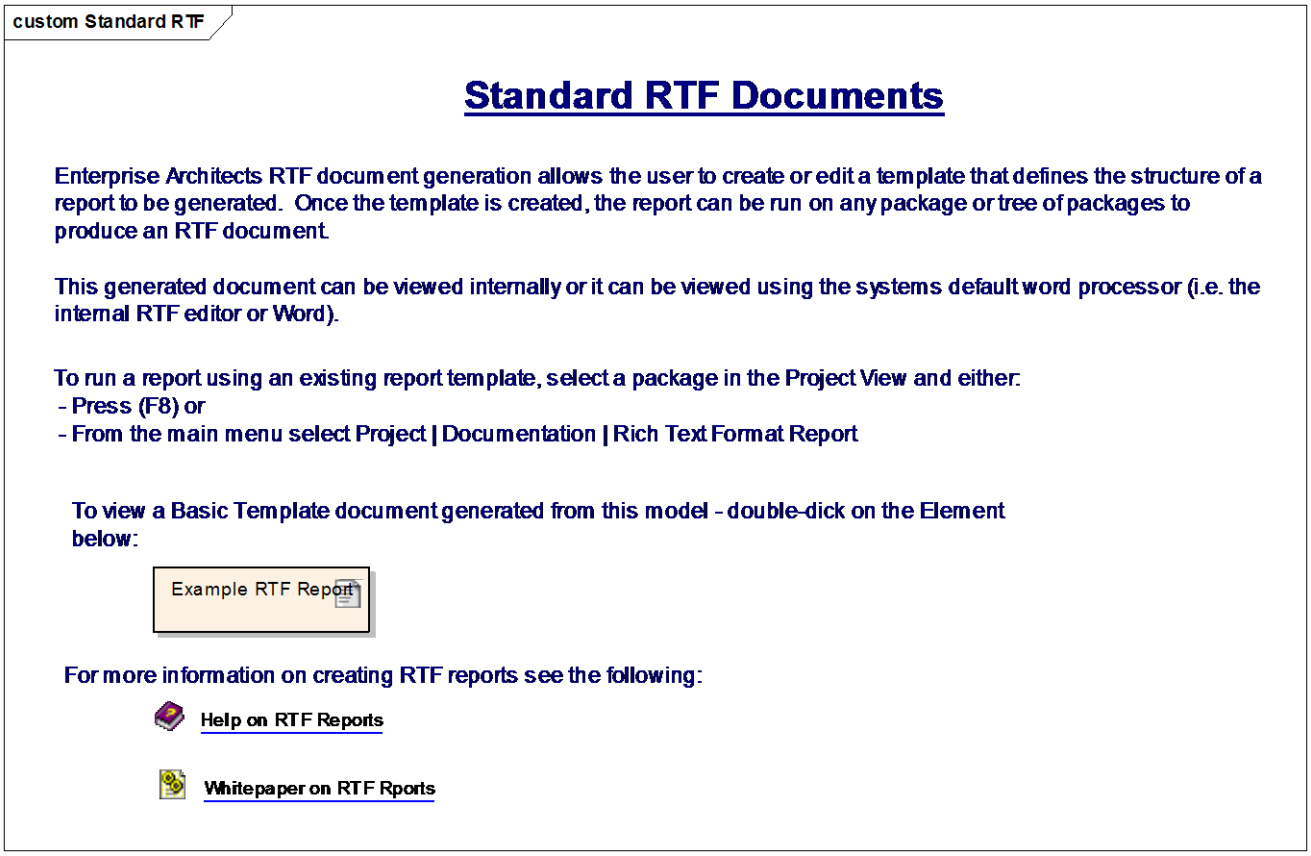


Diagram: Virtual Documents

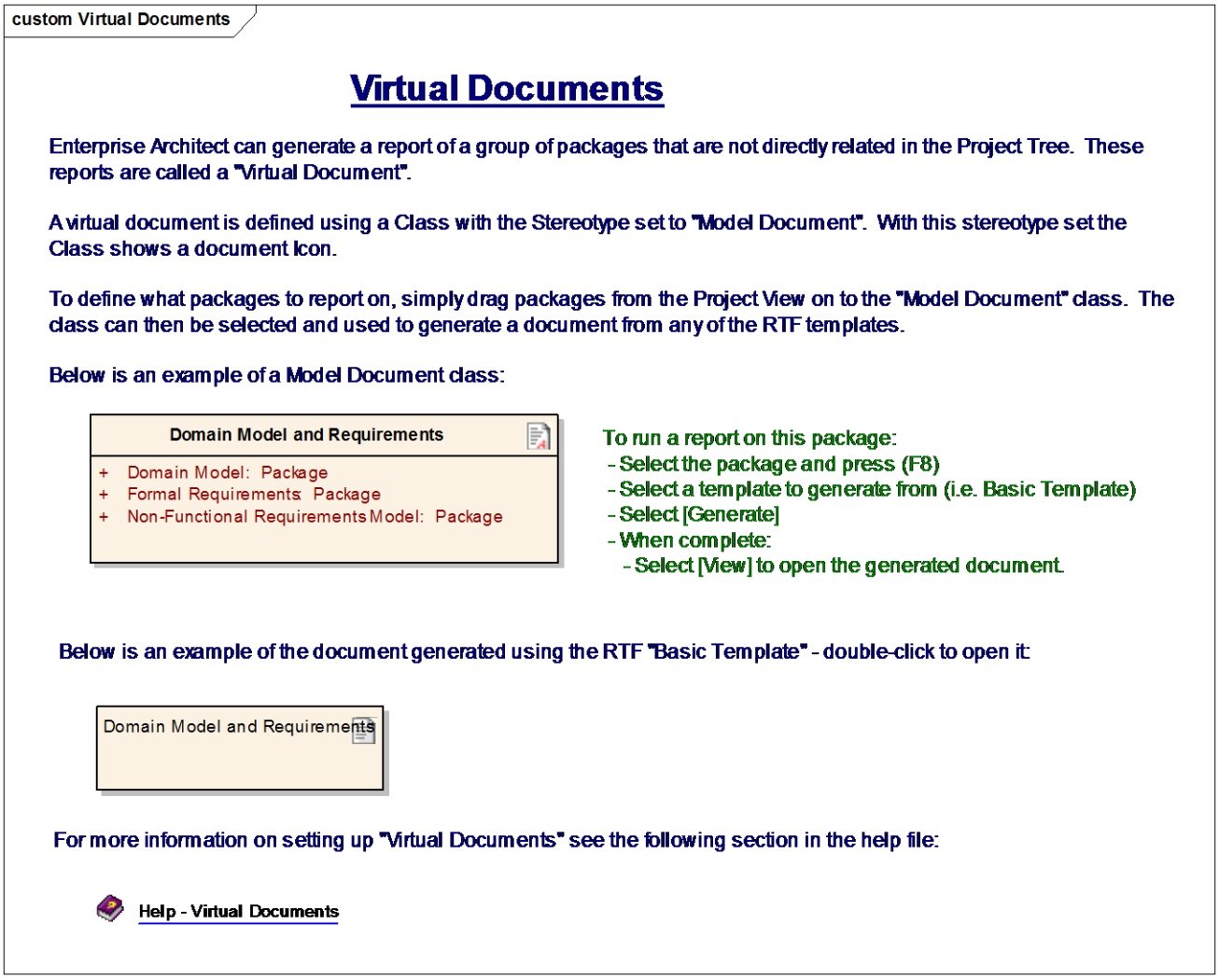


Diagram: Word Master Documents