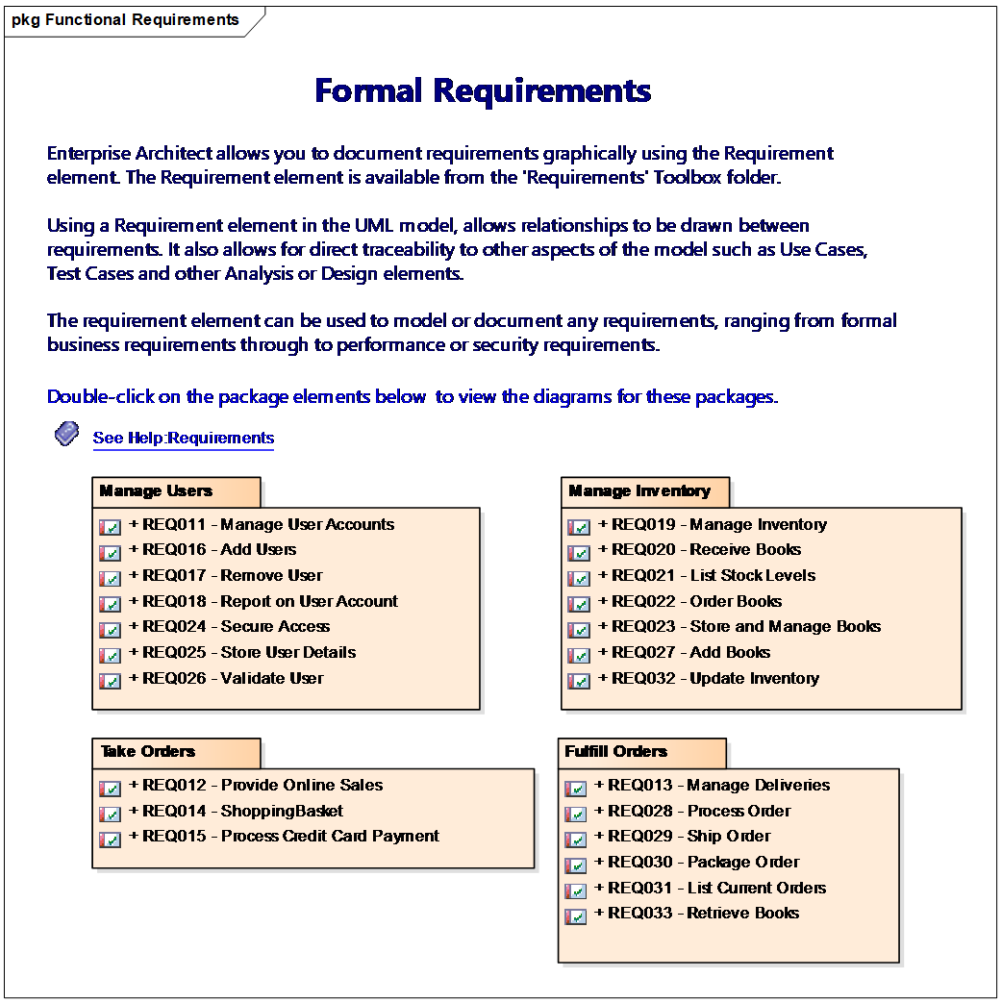
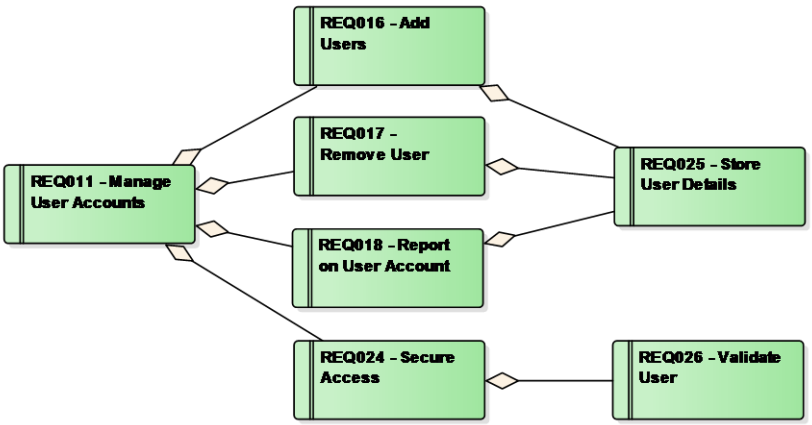
Requirements Model

Formal Requirements



req 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.



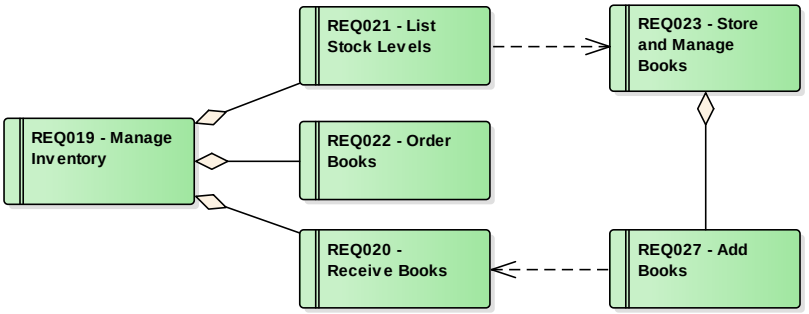
Requirements can have relationships with other elements such as other Requirements, Use Cases etc. To view the Traceability of a requirement use the Hierarchy window. The Hierarchy window can be accessed from the main menu: View | Hierarchy or (Ctrl-Shift-4).

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

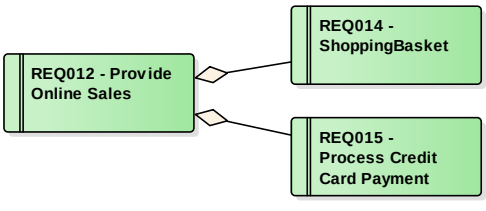
RequirementsManagement

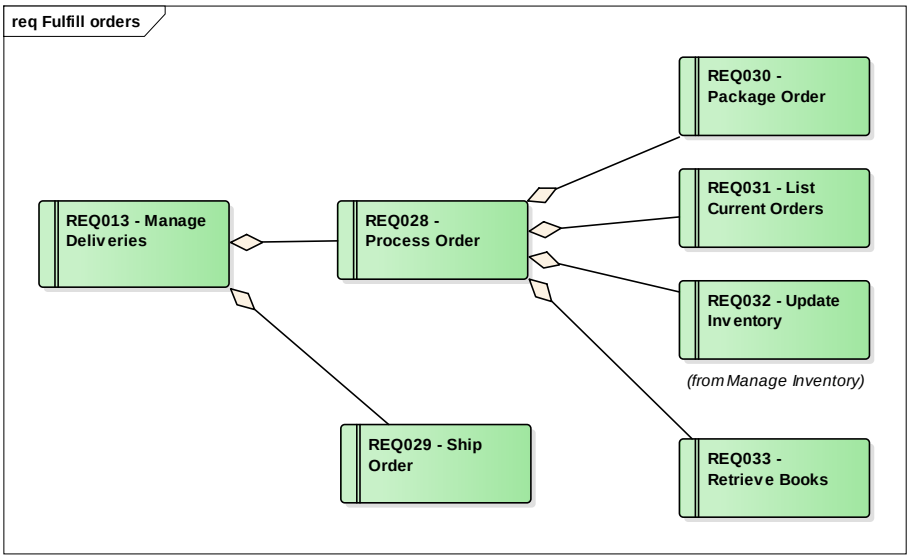
Traceability Window

req Manage Inventory

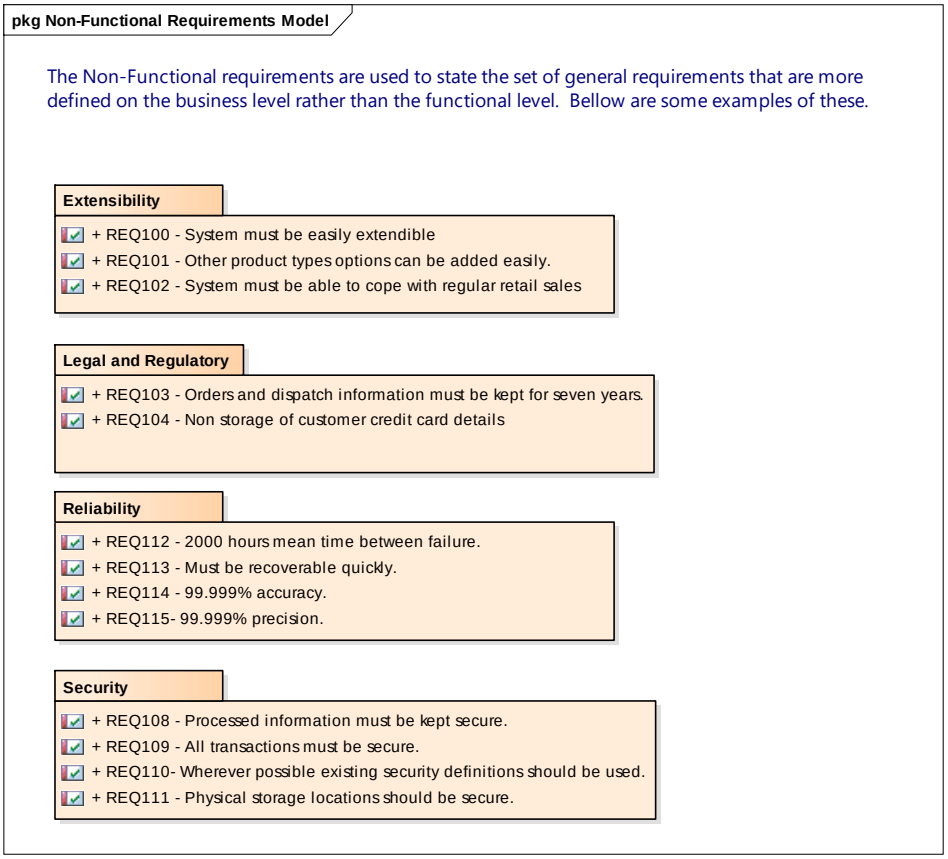


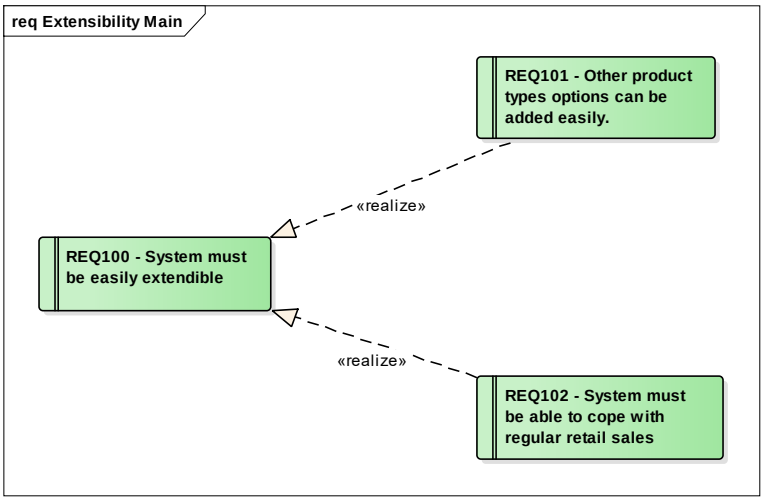
req Take Orders

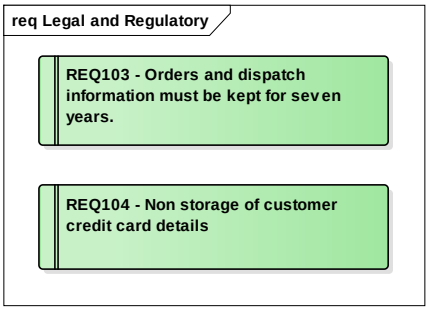


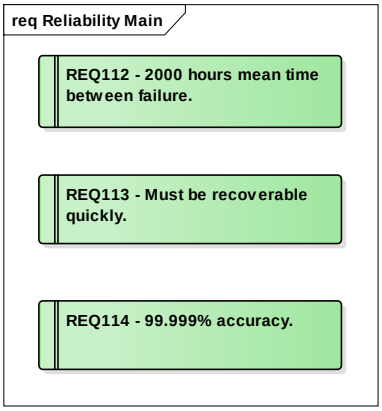


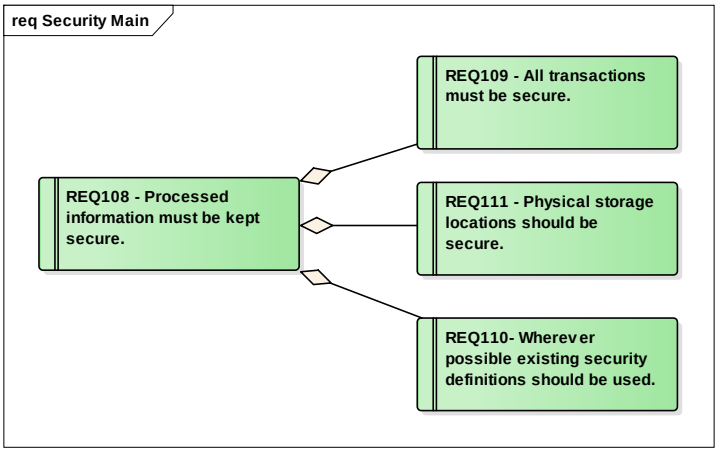
Non-Functional Requirements Model



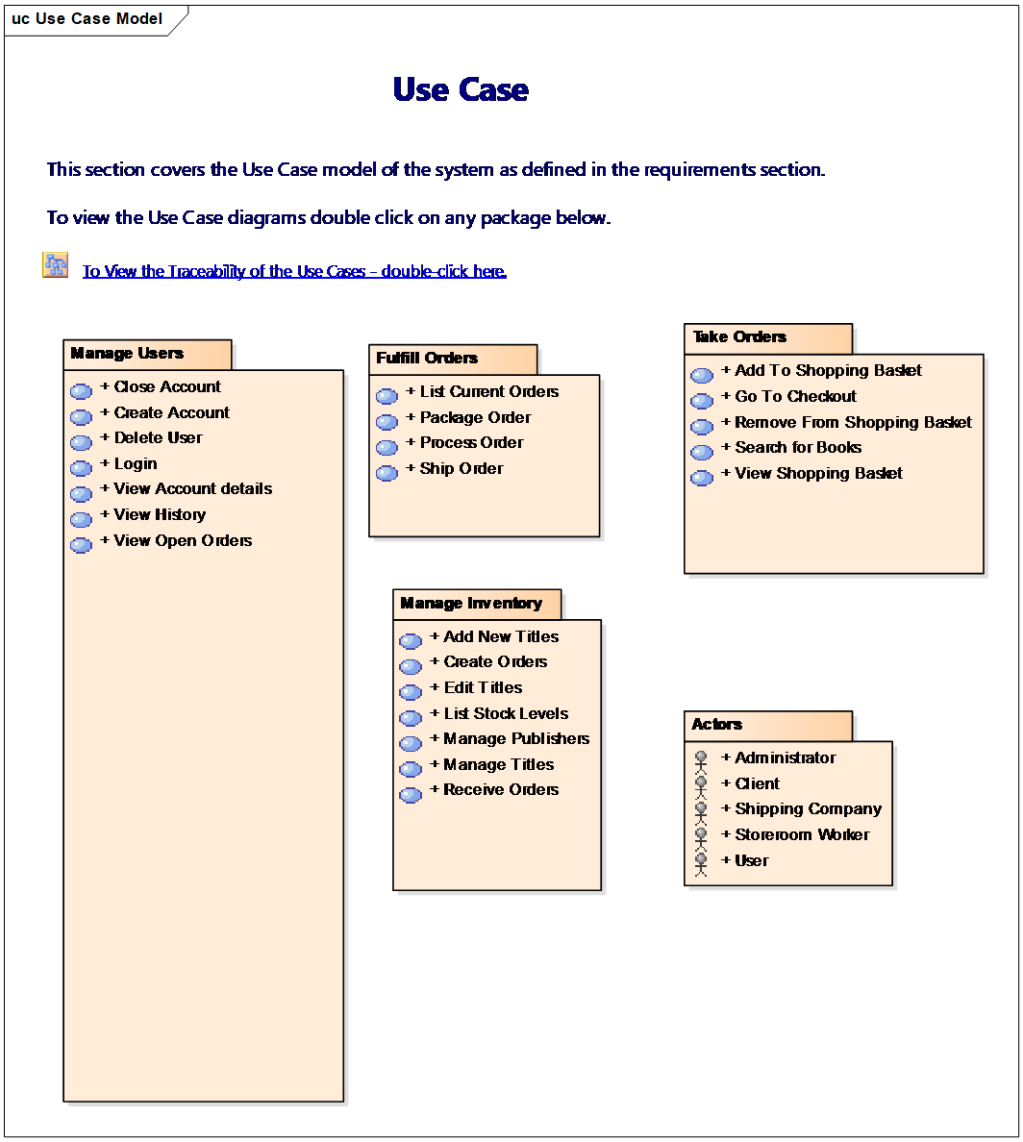

req Extensibility Main
REQ101 - Other product types options can be added easily.
«realize»
REQ100 - System must be easily extendible
«realize»
REQ102 - System must be able to cope with regular retail sales

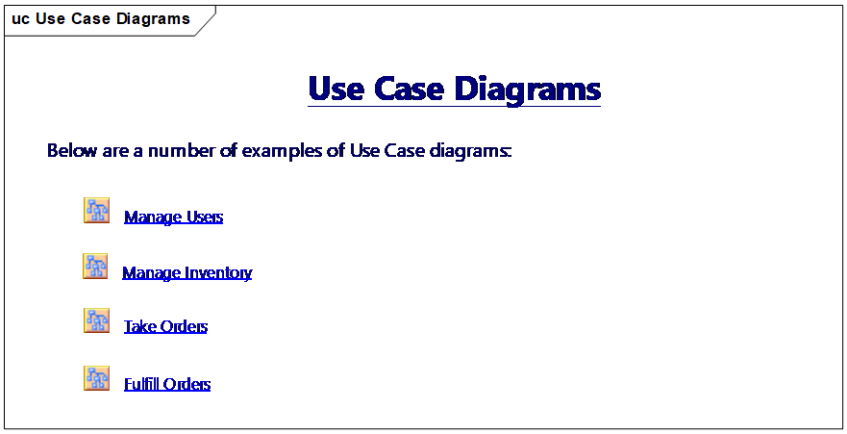

req Legal and Regulatory
REQ103 - Orders and dispatch information must be kept for seven years.
REQ104 - Non storage of customer credit card details

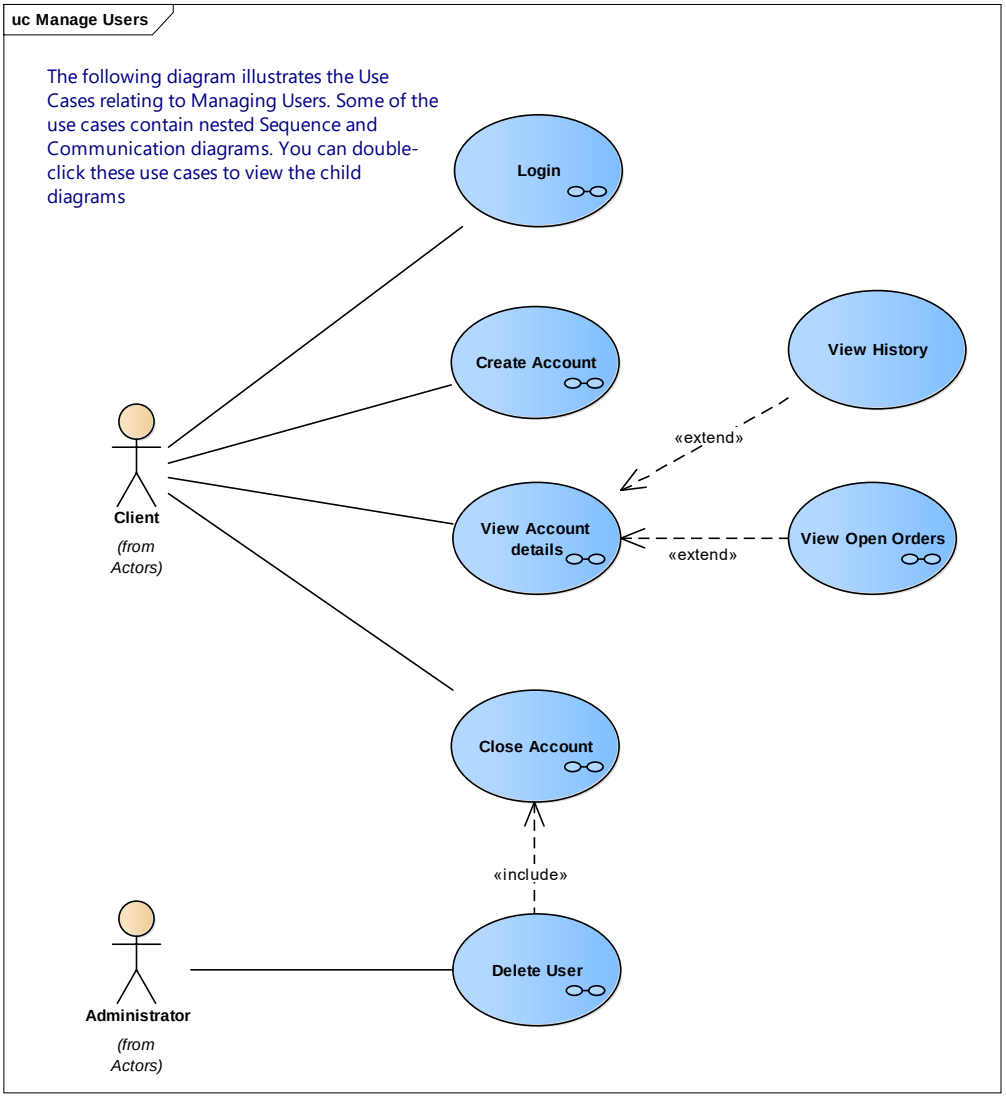

req Reliability Main
REQ112 - 2000 hours mean time between failure.
REQ113 - Must be recoverable quickly.
REQ114 - 99.999% accuracy.

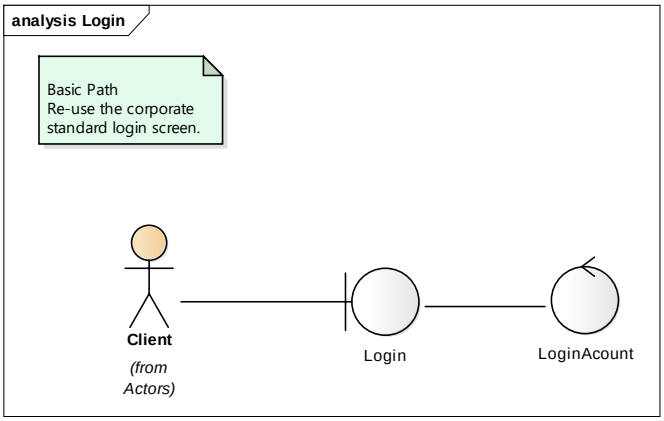

req Security Main
REQ109 - All transactions must be secure.
REQ108 - Processed information must be kept secure.
REQ111 - Physical storage locations should be secure.
REQ110- Wherever possible existing security definitions should be used.

Use Case Model






uc Manage Users
The following diagram illustrates the Use Cases relating to Managing Users. Some of the use cases contain nested Sequence and Communication diagrams. You can double-click these use cases to view the child diagrams
Login
Create Account
View History
«extend»
Client
(from Actors)
View Account details
View Open Orders
«extend»
Close Account
«include»
Administrator
(from Actors)
Delete User


analysis Login
Basic Path
Re-use the corporate standard login screen.
Client
(from Actors)
Login
LoginAcount

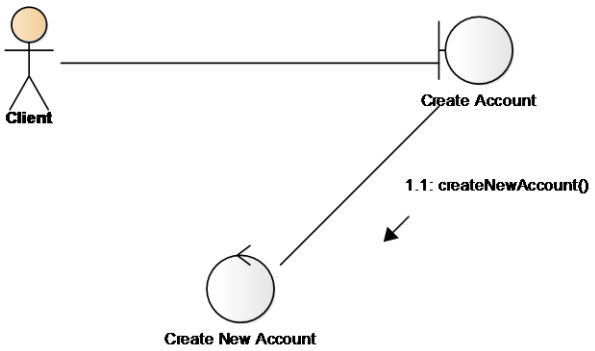
sd Create Account

Basic Path
1. Use case begins when the Client selects the "Create Account" command.
2. The Client enters name, address, e-mail address, password and password confirmation.
3. Client presses Submit button.
4. Submitted details are validated.
5. Processing message is displayed to the Client.
6. New account is created in the database.
7. Success message is displayed to Client.

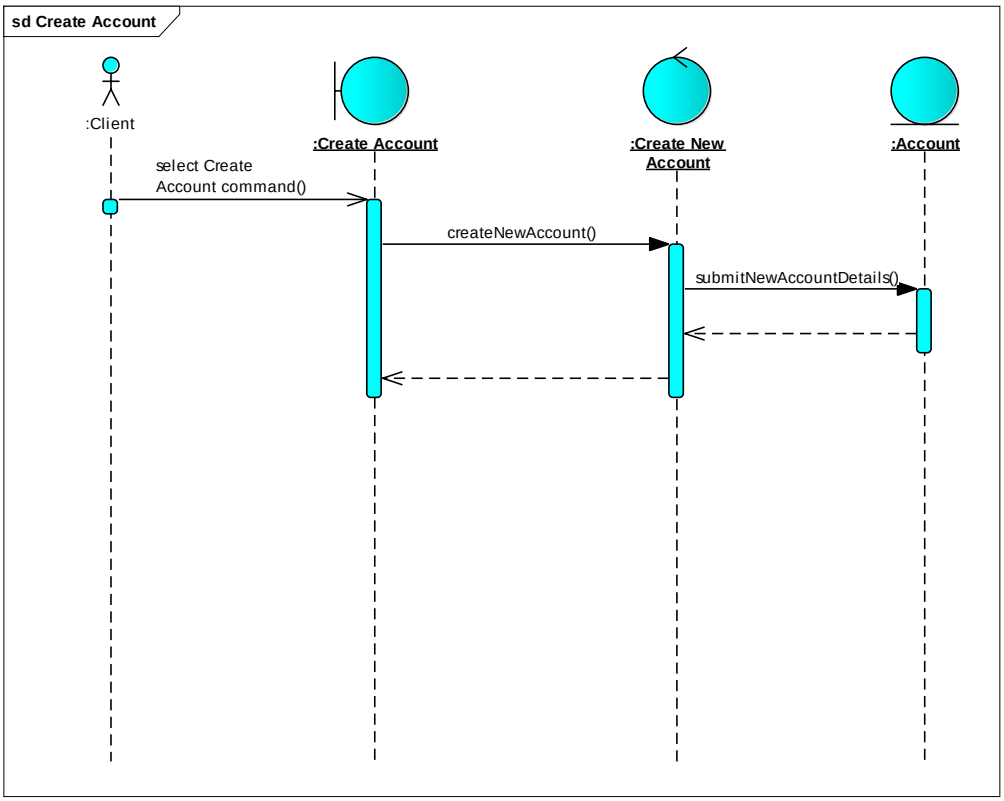
No Submit
The Client may press the Exit or Back button at step 3. The use case terminates with no action performed.

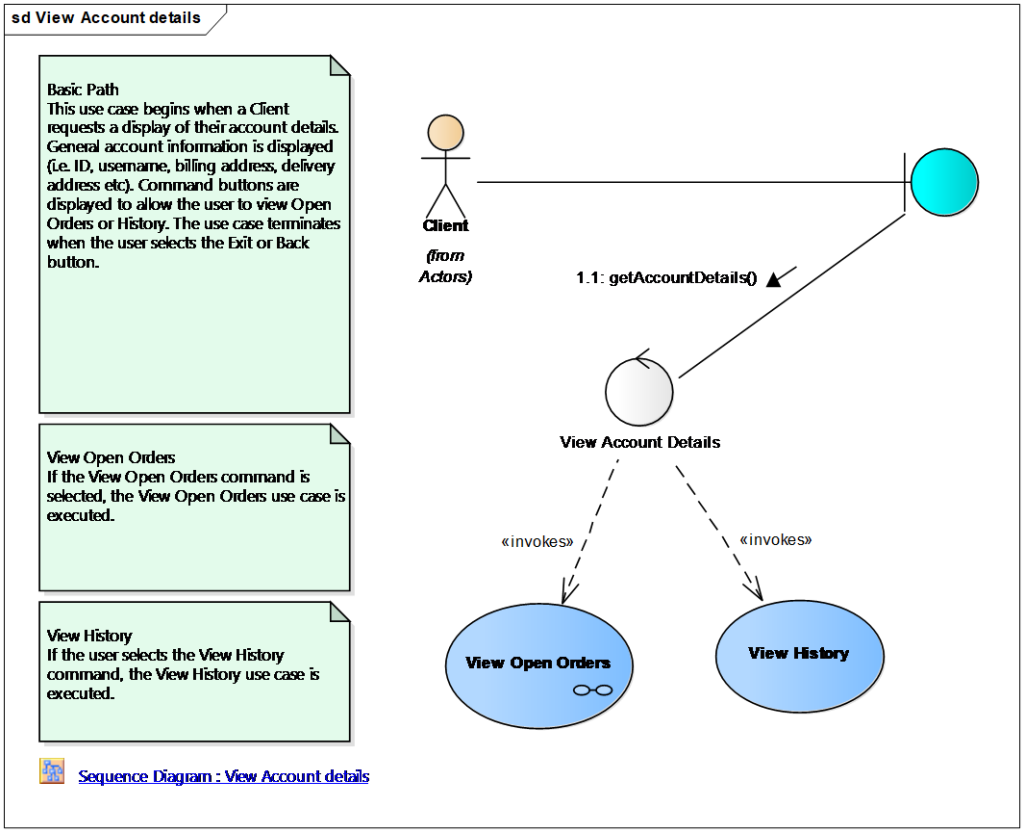
Validation fails
At step 4, if validation fails then an error message is displayed and the Client is returned to step 2 with the invalid field or fields highlighted.

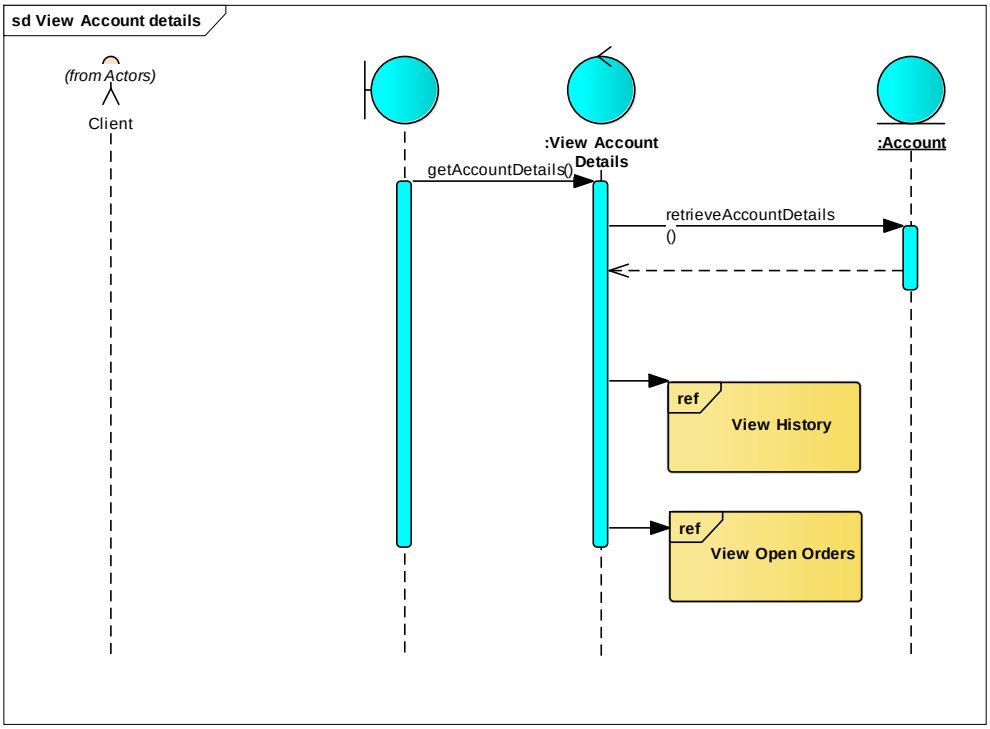
Cannot Create New Account
In step 6, if the database create fails, the Client is informed and the use case terminates.

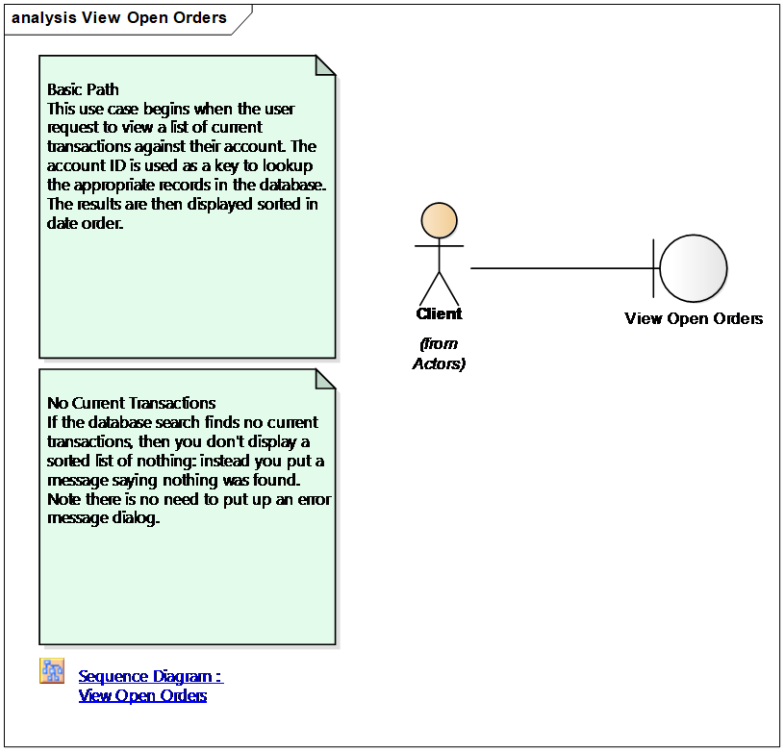


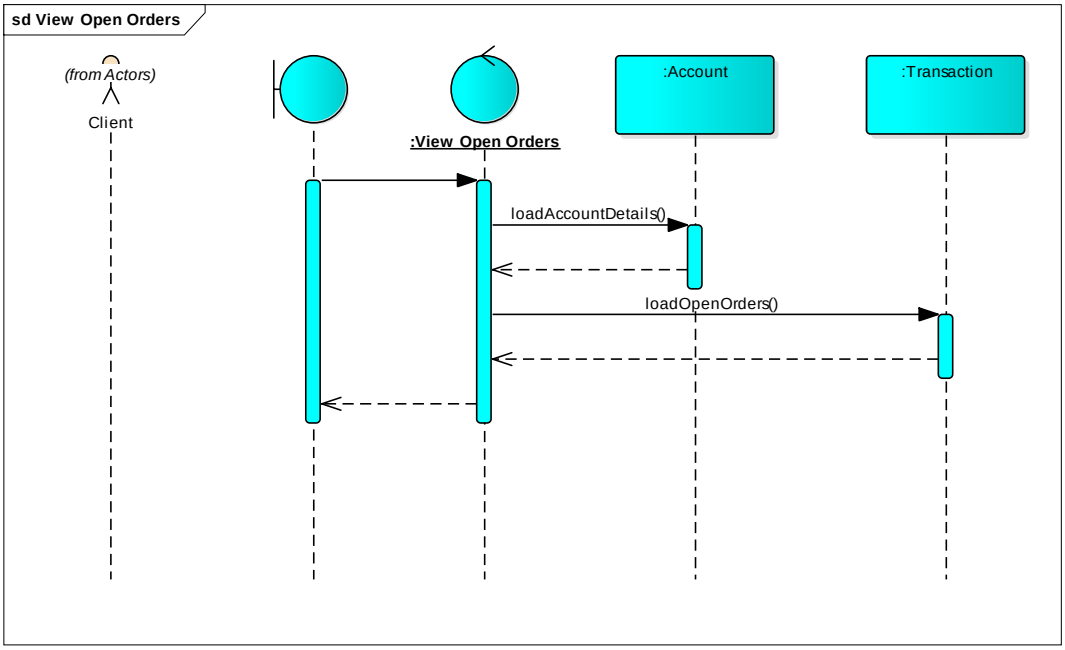
Sequence Diagram : Create Account

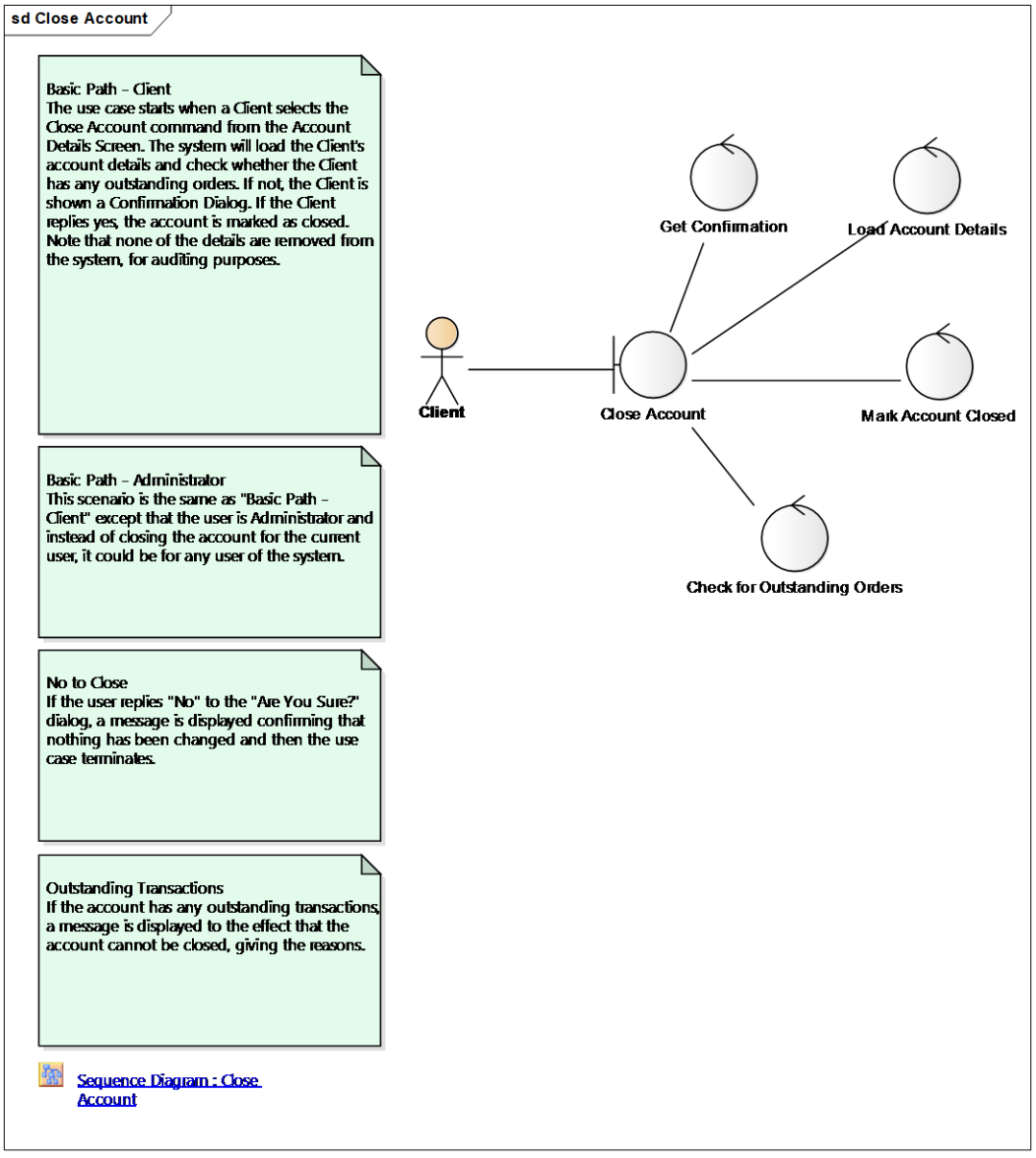

sd Create Account
:Client
:Create Account
:Create New Account
:Account
select Create Account command()
createNewAccount()
submitNewAccountDetails()

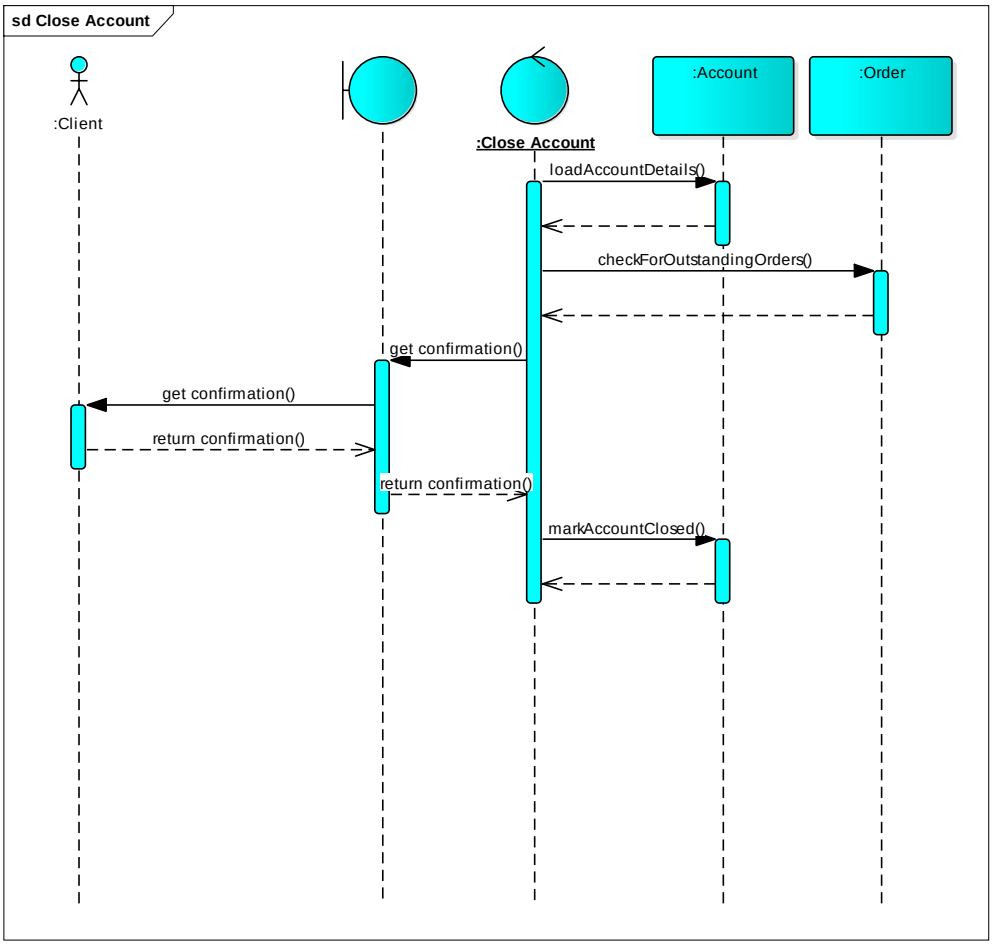

sd View Account details
Basic Path
This use case begins when a Client requests a display of their account details. General account information is displayed (i.e. ID, username, billing address, delivery address etc). Command buttons are displayed to allow the user to view Open Orders or History. The use case terminates when the user selects the Exit or Back button.
View Open Orders
If the View Open Orders command is selected, the View Open Orders use case is executed.
View History
If the user selects the View History command, the View History use case is executed.
Sequence Diagram : View Account details
Client
(from Actors)
1.1: getAccountDetails()
View Account Details
«invokes»
«invokes»
View Open Orders
View History


sd View Account details
(from Actors)
Client
:View Account Details
:Account
getAccountDetails()
retrieveAccountDetails
()
ref
View History
ref
View Open Orders


analysis View Open Orders
Basic Path
This use case begins when the user request to view a list of current transactions against their account. The account ID is used as a key to lookup the appropriate records in the database. The results are then displayed sorted in date order.
No Current Transactions
If the database search finds no current transactions, then you don't display a sorted list of nothing: instead you put a message saying nothing was found. Note there is no need to put up an error message dialog.
Client
(from Actors)
View Open Orders
Sequence Diagram : View Open Orders


sd View Open Orders
(from Actors)
Client
:View Open Orders
:Account
:Transaction
loadAccountDetails()
loadOpenOrders()


sd Close Account
Basic Path – Client
The use case starts when a Client selects the Close Account command from the Account Details Screen. The system will load the Client's account details and check whether the Client has any outstanding orders. If not, the Client is shown a Confirmation Dialog. If the Client replies yes, the account is marked as closed. Note that none of the details are removed from the system, for auditing purposes.
Basic Path – Administrator
This scenario is the same as "Basic Path – Client" except that the user is Administrator and instead of closing the account for the current user, it could be for any user of the system.
No to Close
If the user replies "No" to the "Are You Sure?" dialog, a message is displayed confirming that nothing has been changed and then the use case terminates.
Outstanding Transactions
If the account has any outstanding transactions, a message is displayed to the effect that the account cannot be closed, giving the reasons.
Sequence Diagram : Close Account
Get Confirmation
Load Account Details
Client
Close Account
Mark Account Closed
Check for Outstanding Orders


sd Close Account
:Client
:Close Account
:Account
:Order
loadAccountDetails()
checkForOutstandingOrders()
get confirmation()
get confirmation()
return confirmation()
return confirmation()
markAccountClosed()

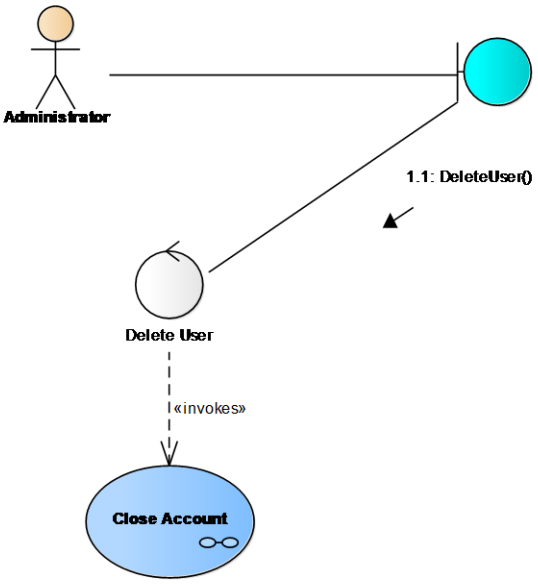
sd Delete User

Basic Path
This use case begins when an Administrator selects the Delete User command. The Administrator is prompted to enter a user ID. The ID is used to retrieve the user's name and address which are displayed on the screen. The administrator is then asked to confirm that they want to delete this particular user and asked to re-enter their password as an added security measure. The Close Account use case is executed to perform the closure and that use case will display the success or failure notification. The administrator is able to enter another user ID and repeat the process. The use case terminates when the administrator selects the Back or Exit button.

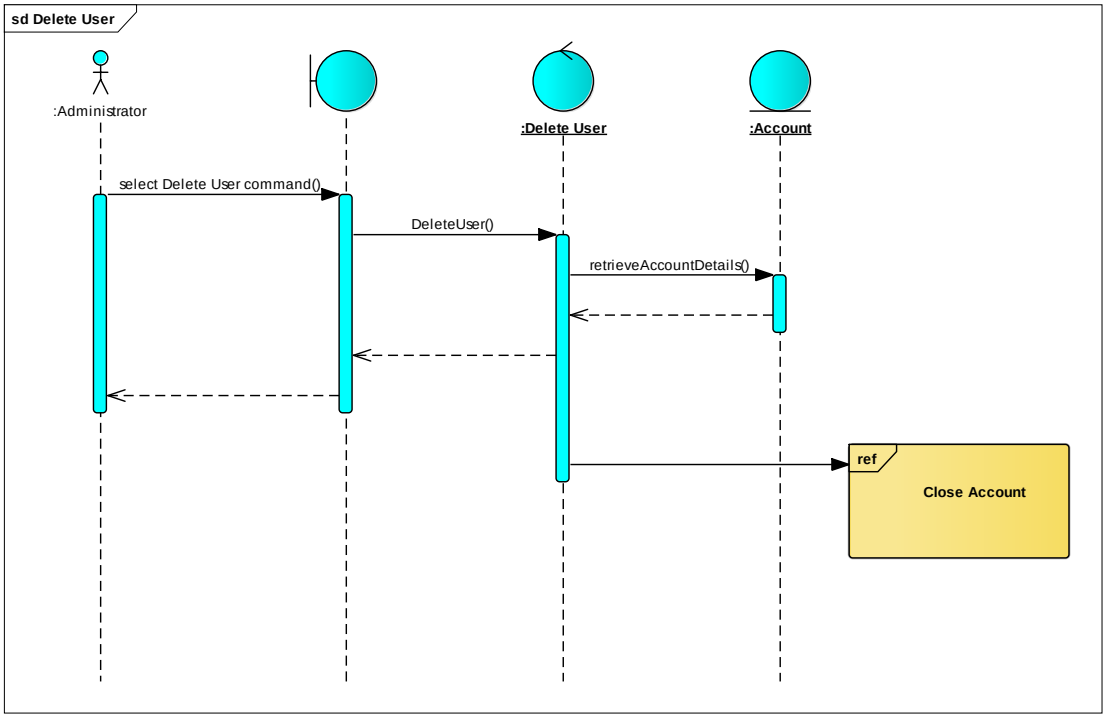
Bad ID
If the user ID doesn't match any existing account, an error message is displayed and the administrator is prompted to re-enter it.

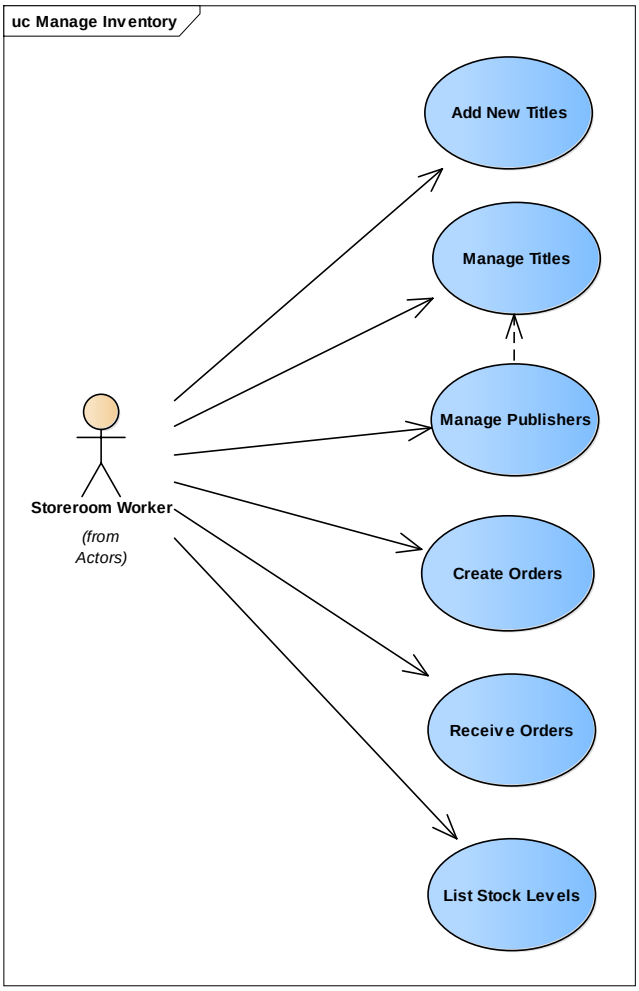
Bad password
If the administrator enters an incorrect password, an error message is displayed and the administrator is allowed a maximum of two more attempts before the use case terminates.

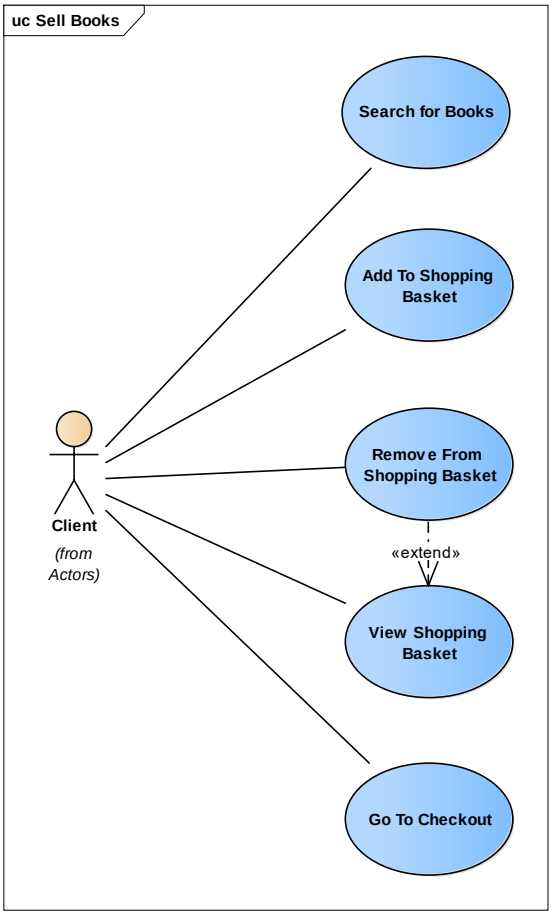
No confirm
If the administrator answers No to the Are You Sure request, it is assumed that the wrong user ID was entered so the user ID is cleared and the administrator is permitted to try again.

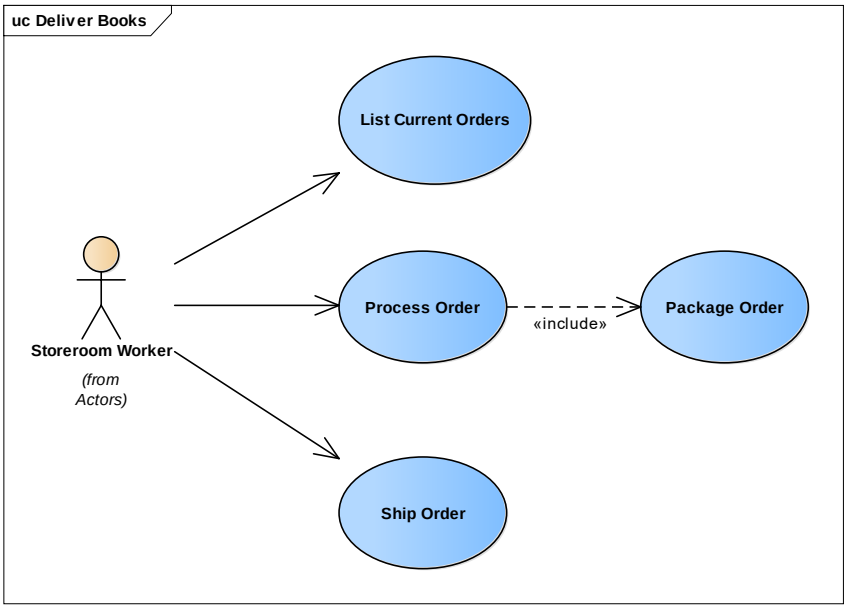


Sequence Diagram : Delete User

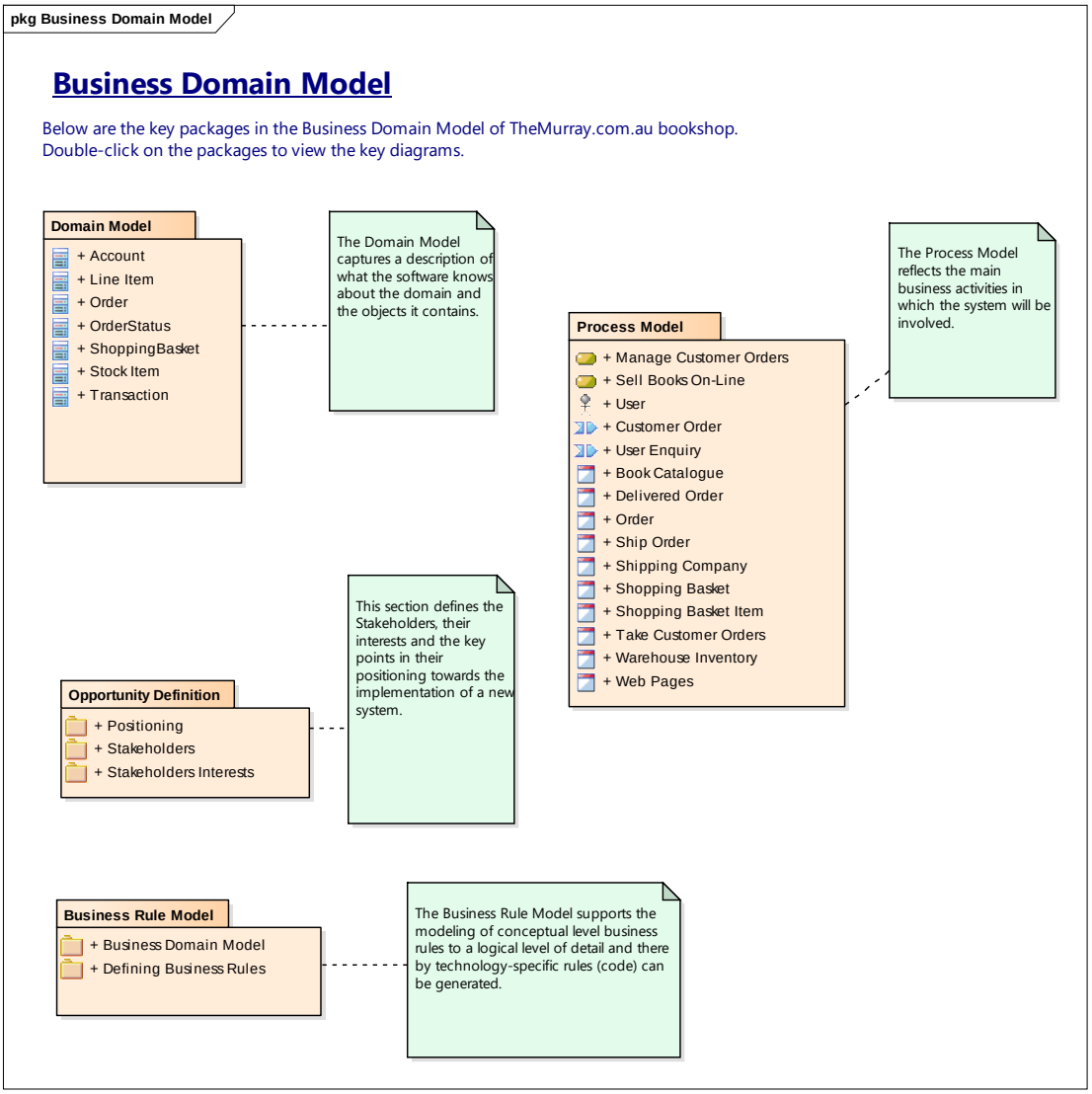

sd Delete User
:Administrator
:Delete User
:Account
select Delete User command()
DeleteUser()
retrieveAccountDetails()
ref
Close Account

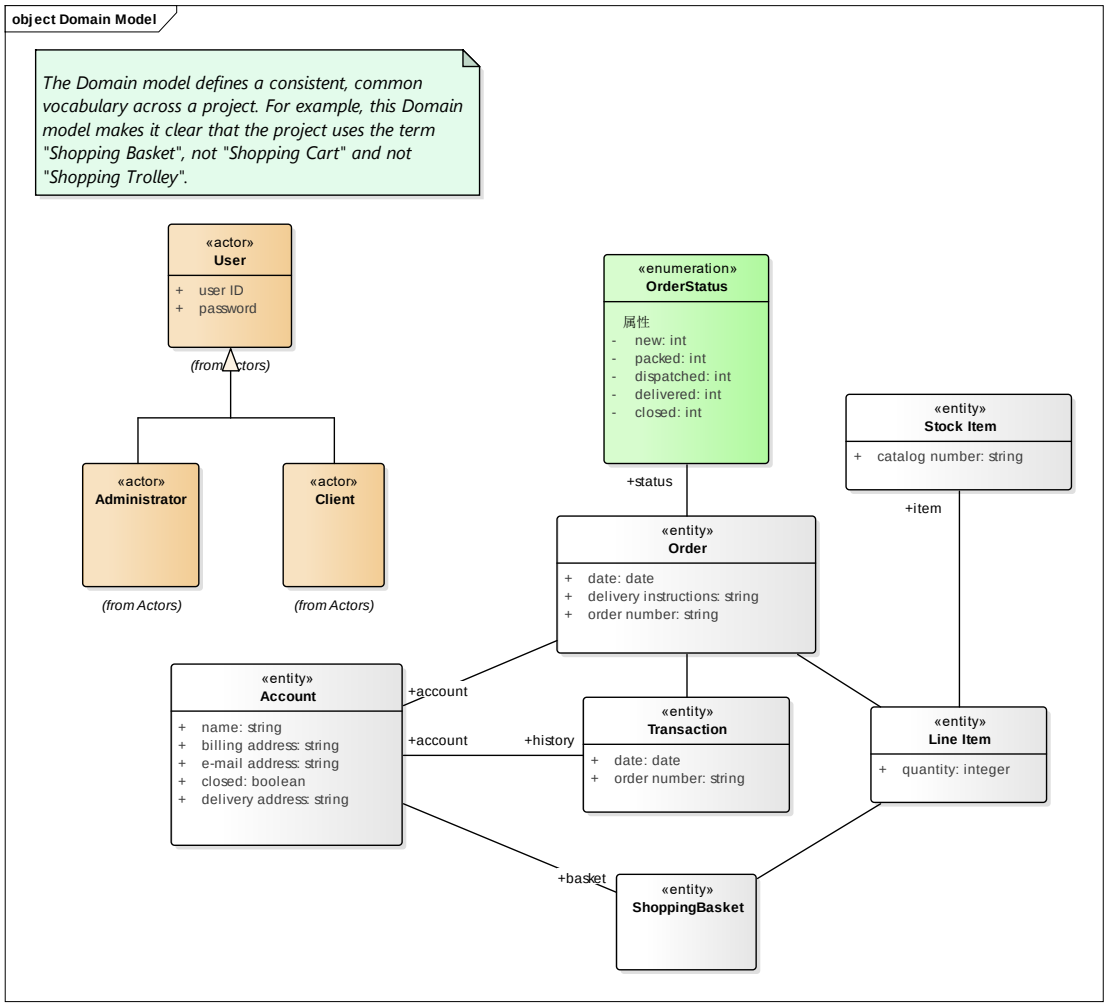

uc Manage Inventory
Add New Titles
Manage Titles
Manage Publishers
Create Orders
Receive Orders
List Stock Levels
Storeroom Worker
(from Actors)

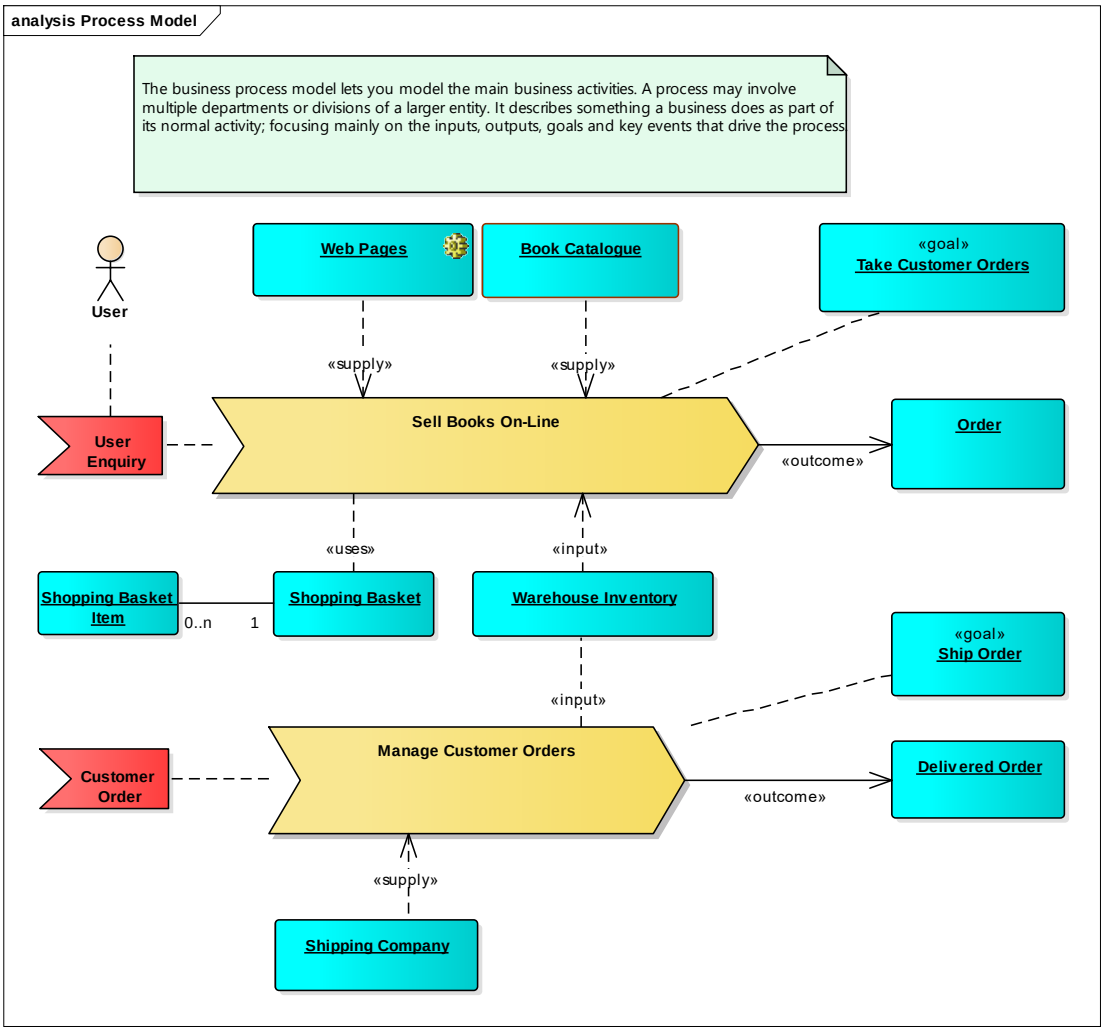

uc Sell Books
Search for Books
Add To Shopping Basket
Remove From Shopping Basket
«extend»
View Shopping Basket
Go To Checkout
Client
(from Actors)

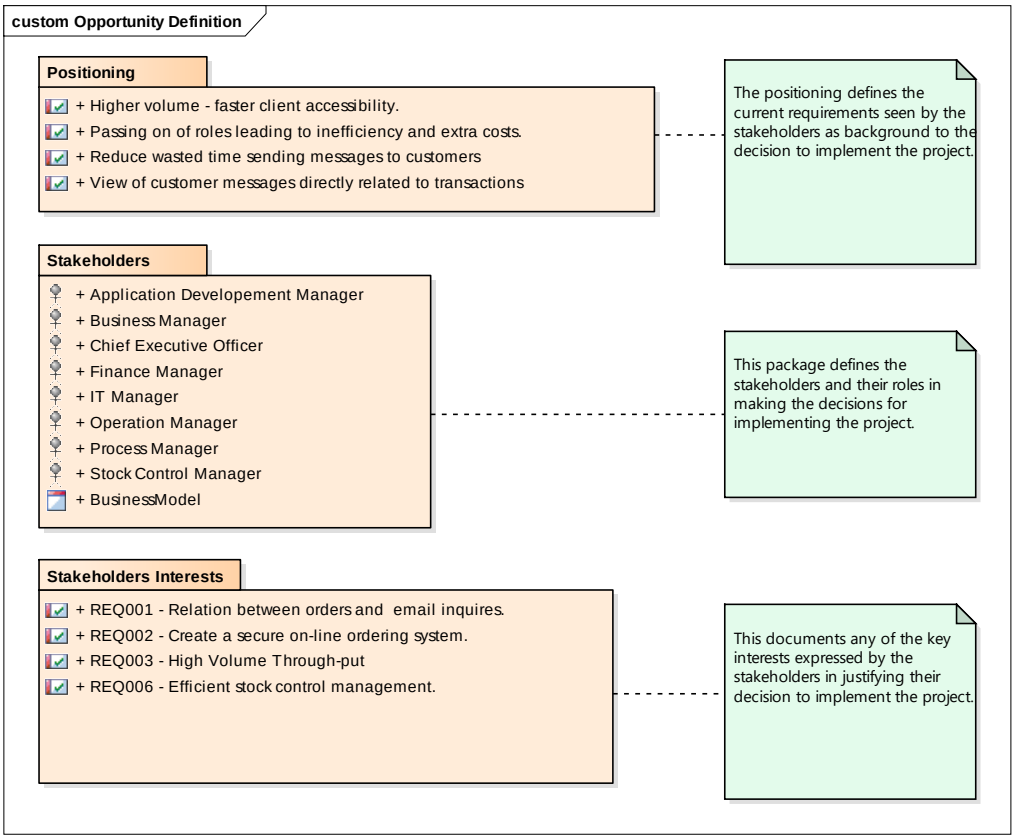

uc Deliver Books
List Current Orders
Process Order
«include»
Package Order
Ship Order
Storeroom Worker
(from Actors)

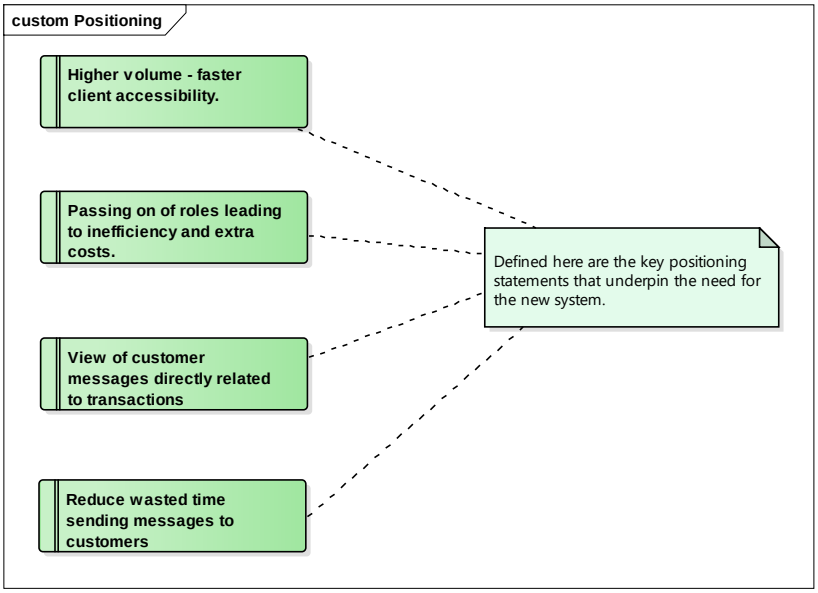
Business Domain Model

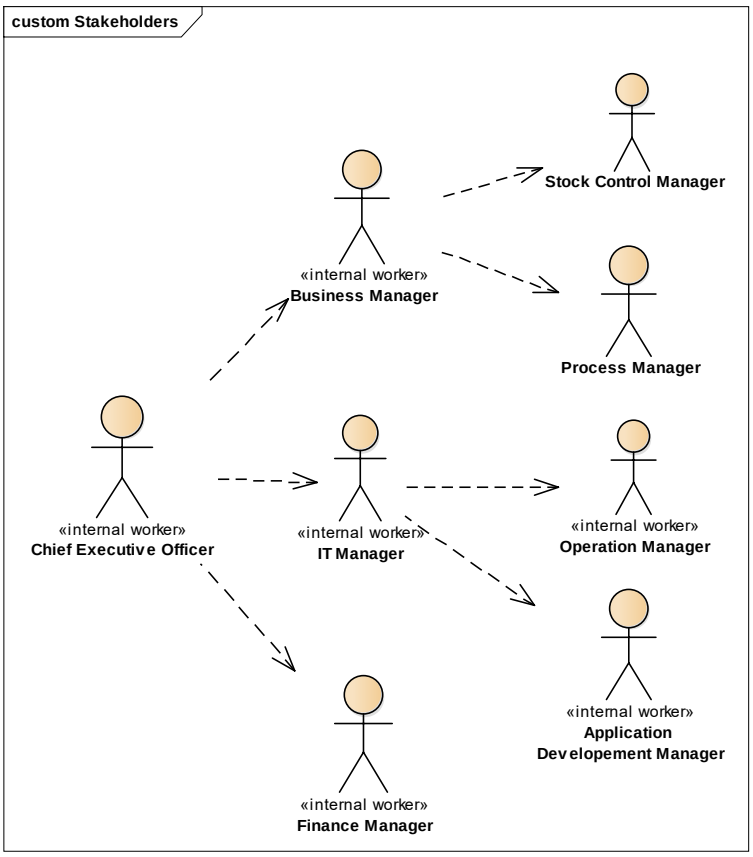


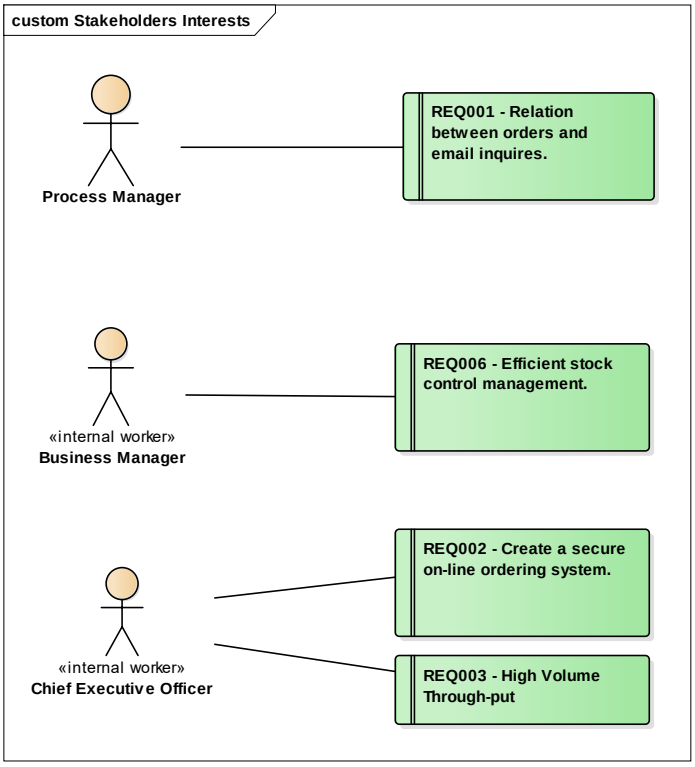

object Domain Model
The Domain model defines a consistent, common vocabulary across a project. For example, this Domain model makes it clear that the project uses the term "Shopping Basket", not "Shopping Cart" and not "Shopping Trolley".
«actor»
User
+ user ID
+ password
(from Actors)
«actor»
Administrator
(from Actors)
«actor»
Client
(from Actors)
«enumeration»
OrderStatus
属性
- new: int
- packed: int
- dispatched: int
- delivered: int
- closed: int
+status
«entity»
Stock Item
+ catalog number: string
+item
«entity»
Order
+ date: date
+ delivery instructions: string
+ order number: string
«entity»
Account
+ name: string
+ billing address: string
+ e-mail address: string
+ closed: boolean
+ delivery address: string
+account
+account
+history
«entity»
Transaction
+ date: date
+ order number: string
«entity»
Line Item
+ quantity: integer
+basket
«entity»
ShoppingBasket


analysis Process Model
The business process model lets you model the main business activities. A process may involve multiple departments or divisions of a larger entity. It describes something a business does as part of its normal activity; focusing mainly on the inputs, outputs, goals and key events that drive the process
User
Web Pages
Book Catalogue
«goal»
Take Customer Orders
«supply»
«supply»
User Enquiry
Sell Books On-Line
Order
«outcome»
«uses»
«input»
Shopping Basket Item
0..n
1
Shopping Basket
Warehouse Inventory
«goal»
Ship Order
«input»
Customer Order
Manage Customer Orders
Delivered Order
«outcome»
«supply»
Shipping Company


custom Opportunity Definition
Positioning
+ Higher volume - faster client accessibility.
+ Passing on of roles leading to inefficiency and extra costs.
+ Reduce wasted time sending messages to customers
+ View of customer messages directly related to transactions
The positioning defines the current requirements seen by the stakeholders as background to the decision to implement the project.
Stakeholders
+ Application Developement Manager
+ Business Manager
+ Chief Executive Officer
+ Finance Manager
+ IT Manager
+ Operation Manager
+ Process Manager
+ Stock Control Manager
+ BusinessModel
This package defines the stakeholders and their roles in making the decisions for implementing the project.
Stakeholders Interests
+ REQ001 - Relation between orders and email inquires.
+ REQ002 - Create a secure on-line ordering system.
+ REQ003 - High Volume Through-put
+ REQ006 - Efficient stock control management.
This documents any of the key interests expressed by the stakeholders in justifying their decision to implement the project.

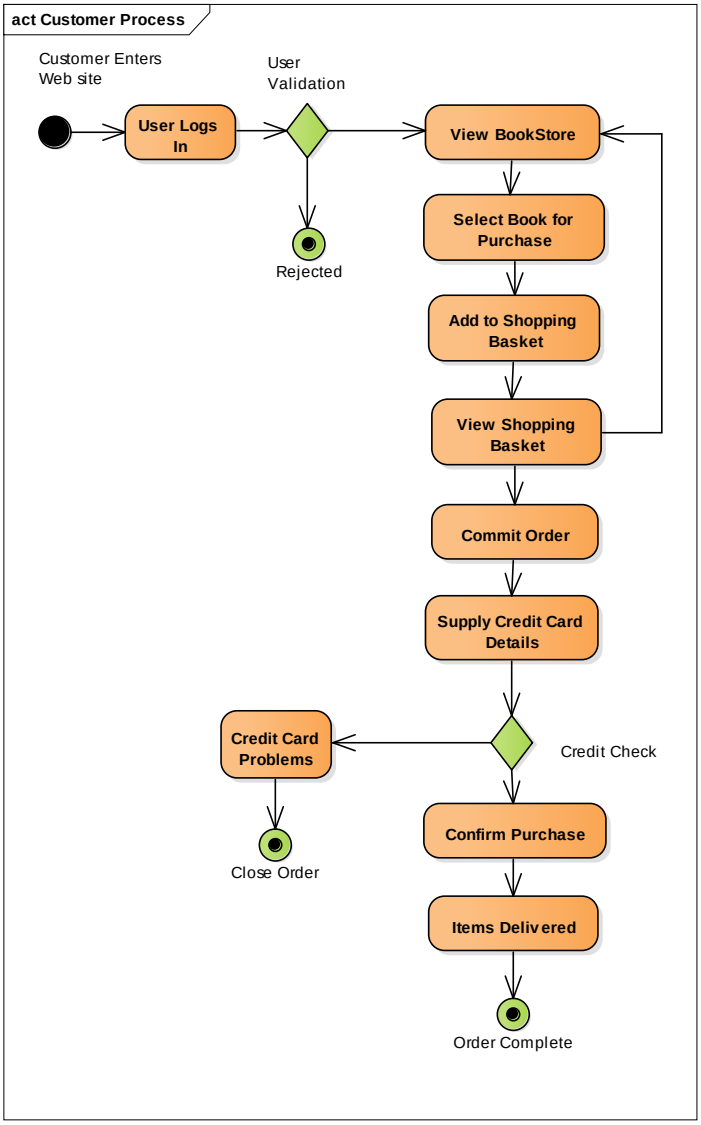

custom Positioning
Higher volume - faster client accessibility.
Passing on of roles leading to inefficiency and extra costs.
View of customer messages directly related to transactions
Reduce wasted time sending messages to customers
Defined here are the key positioning statements that underpin the need for the new system.

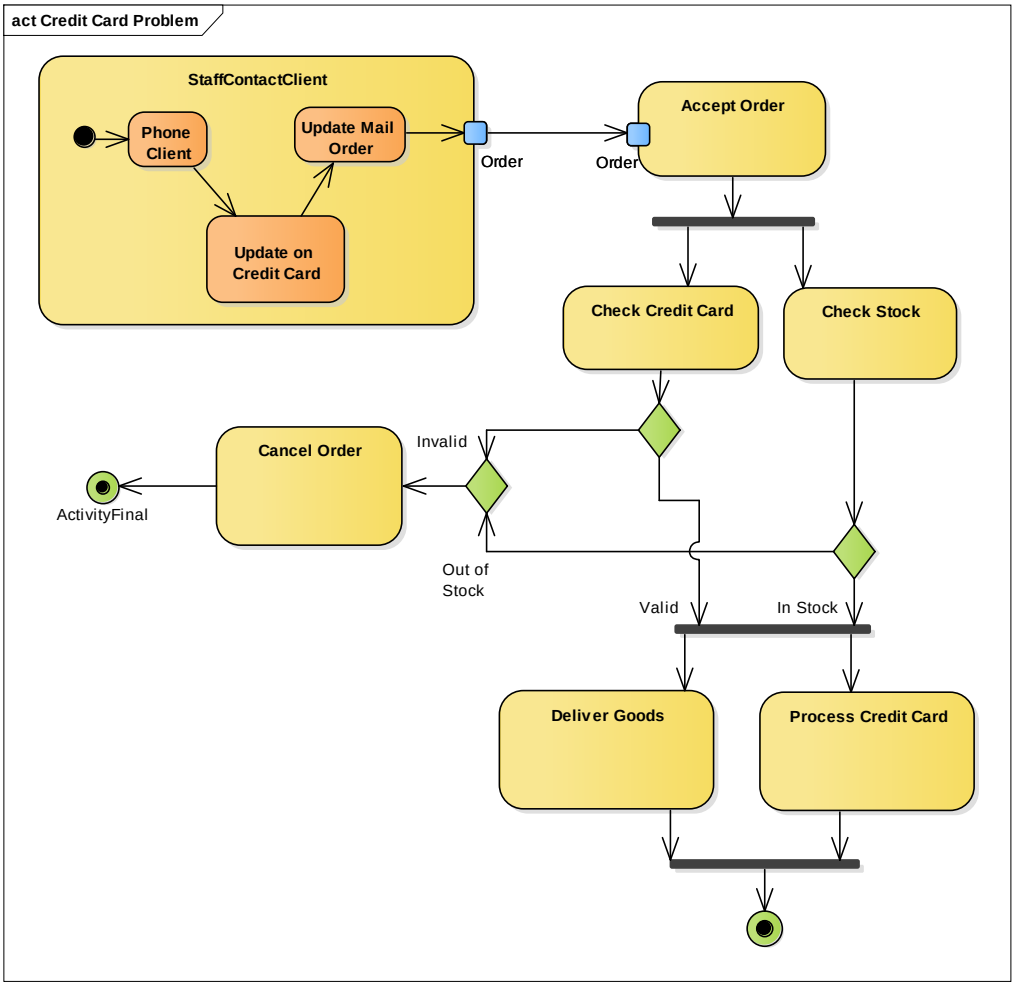

custom Stakeholders
Stock Control Manager
«internal worker»
Business Manager
Process Manager
«internal worker»
Chief Executive Officer
«internal worker»
IT Manager
«internal worker»
Operation Manager
«internal worker»
Application
Developement Manager
«internal worker»
Finance Manager

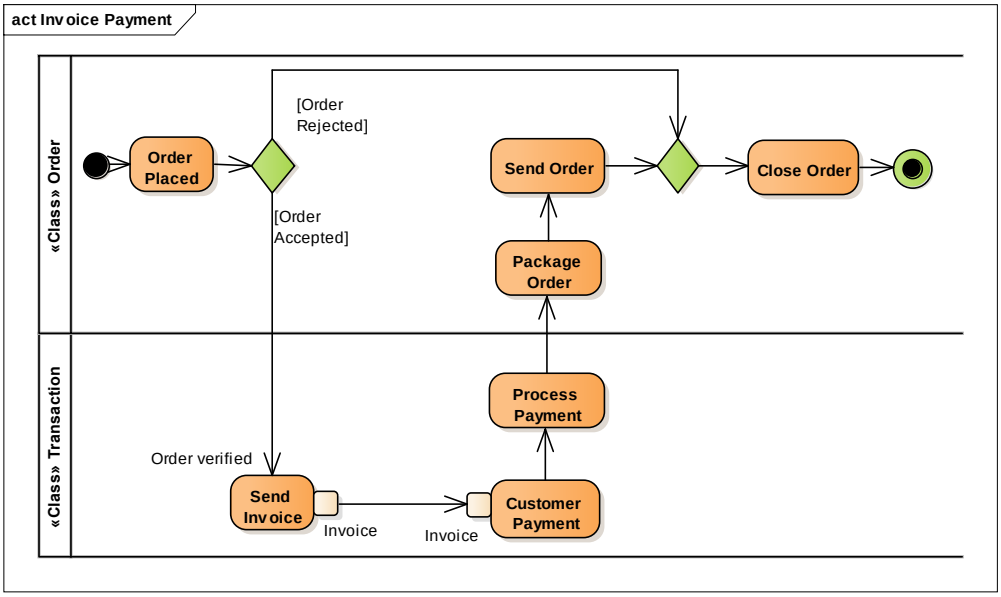

custom Stakeholders Interests
Process Manager
REQ001 - Relation between orders and email inquires.
«internal worker»
Business Manager
REQ006 - Efficient stock control management.
«internal worker»
Chief Executive Officer
REQ002 - Create a secure on-line ordering system.
REQ003 - High Volume Through-put

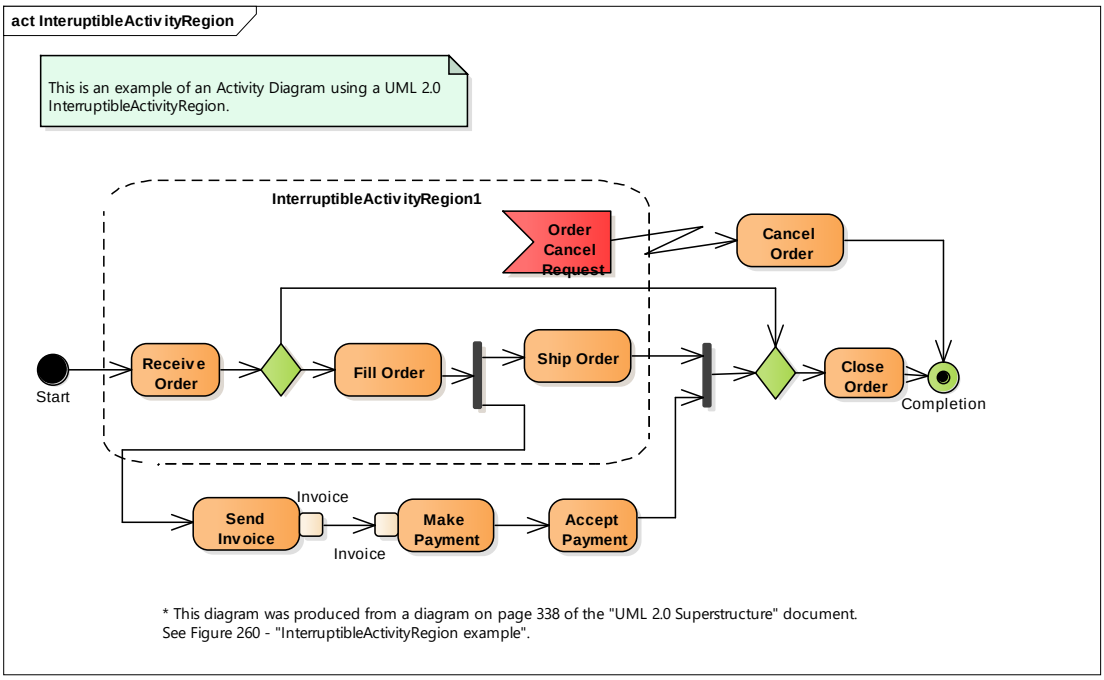
UML Behavior Diagrams

Activity Diagrams

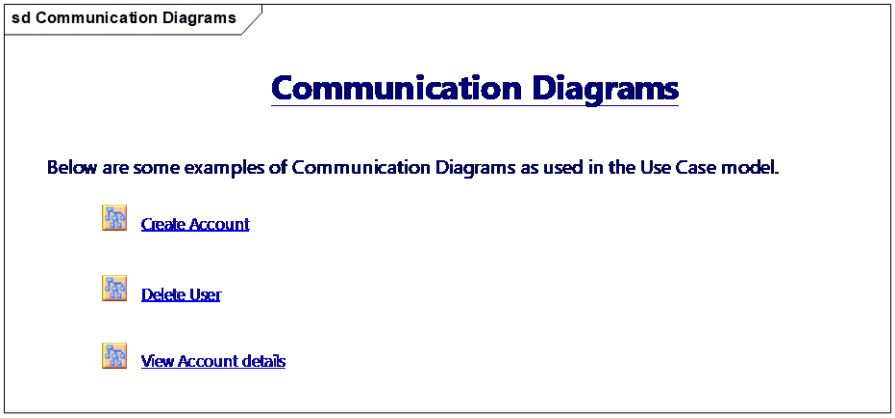



act Credit Card Problem
StaffContactClient
Phone Client
Update Mail Order
Update on Credit Card
Order
Order
Accept Order
Check Credit Card
Check Stock
Cancel Order
Invalid
ActivityFinal
Out of Stock
Valid
In Stock
Deliver Goods
Process Credit Card


act Invoice Payment
«Class» Order
«Class» Transaction
Order Placed
[Order Rejected]
[Order Accepted]
Send Order
Close Order
Package Order
Process Payment
Order verified
Send Invoice
Invoice
Invoice
Customer Payment



Communication Diagrams



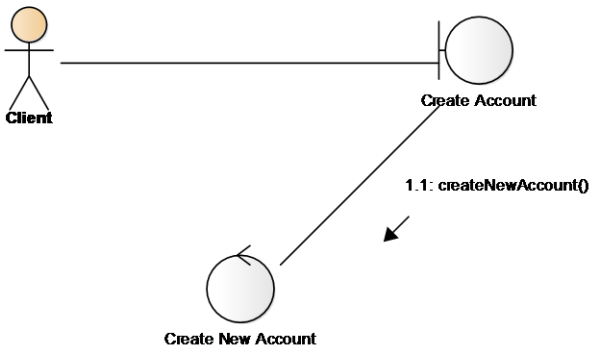
sd Create Account

Basic Path
1. Use case begins when the Client selects the "Create Account" command.
2. The Client enters name, address, e-mail address, password and password confirmation.
3. Client presses Submit button.
4. Submitted details are validated.
5. Processing message is displayed to the Client.
6. New account is created in the database.
7. Success message is displayed to Client.

No Submit
The Client may press the Exit or Back button at step 3. The use case terminates with no action performed.

Validation fails
At step 4, if validation fails then an error message is displayed and the Client is returned to step 2 with the invalid field or fields highlighted.

Cannot Create New Account
In step 6, if the database create fails, the Client is informed and the use case terminates.



Sequence Diagram : Create Account

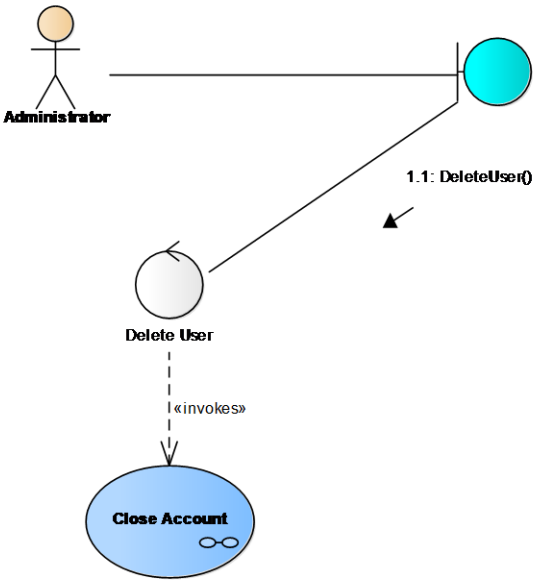
sd Delete User

Basic Path
This use case begins when an Administrator selects the Delete User command. The Administrator is prompted to enter a user ID. The ID is used to retrieve the user's name and address which are displayed on the screen. The administrator is then asked to confirm that they want to delete this particular user and asked to re-enter their password as an added security measure. The Close Account use case is executed to perform the closure and that use case will display the success or failure notification. The administrator is able to enter another user ID and repeat the process. The use case terminates when the administrator selects the Back or Exit button.

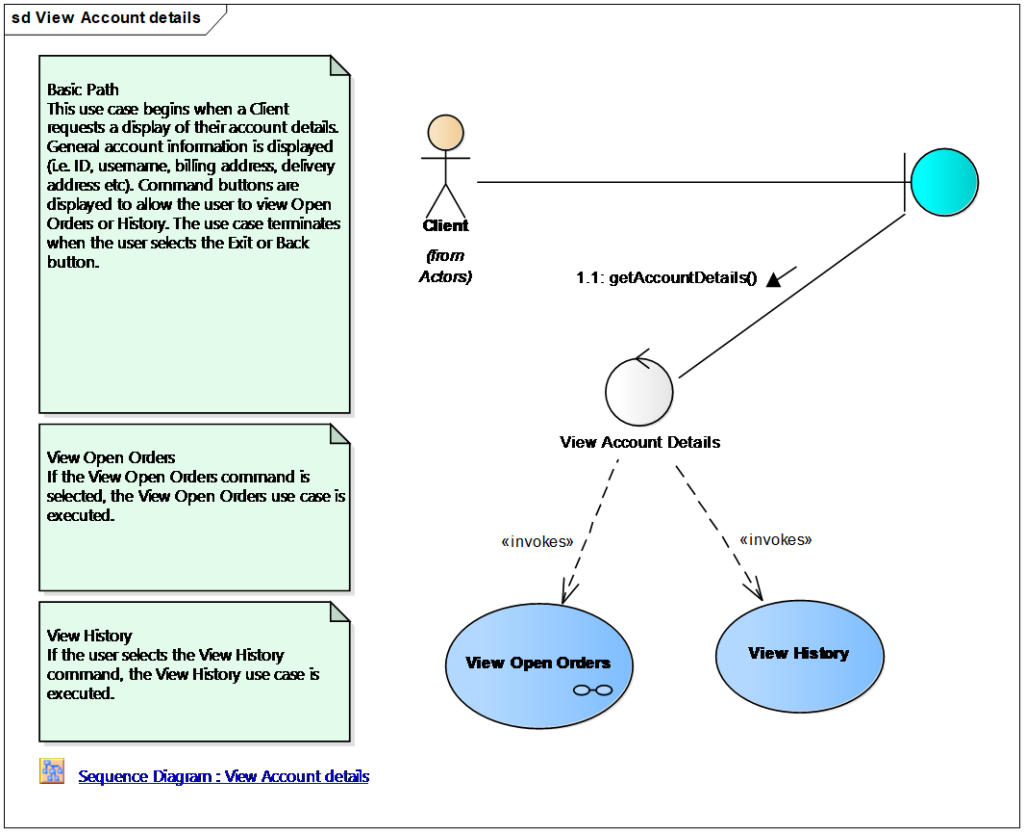
Bad ID
If the user ID doesn't match any existing account, an error message is displayed and the administrator is prompted to re-enter it.

Bad password
If the administrator enters an incorrect password, an error message is displayed and the administrator is allowed a maximum of two more attempts before the use case terminates.

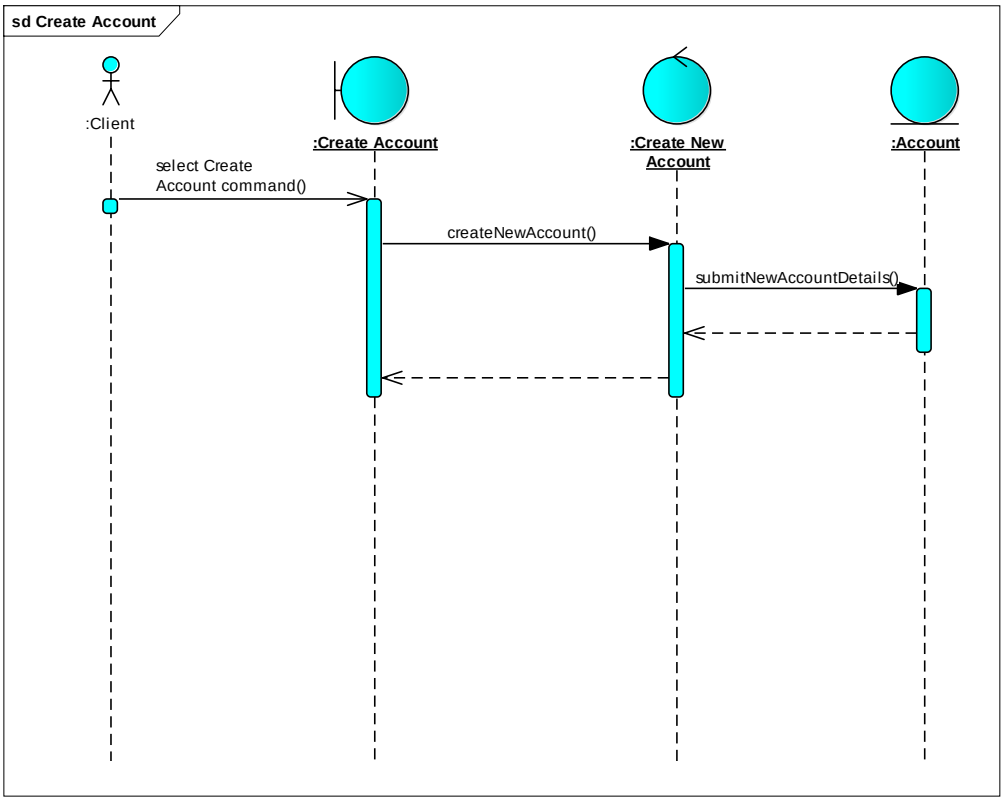
No confirm
If the administrator answers No to the Are You Sure request, it is assumed that the wrong user ID was entered so the user ID is cleared and the administrator is permitted to try again.

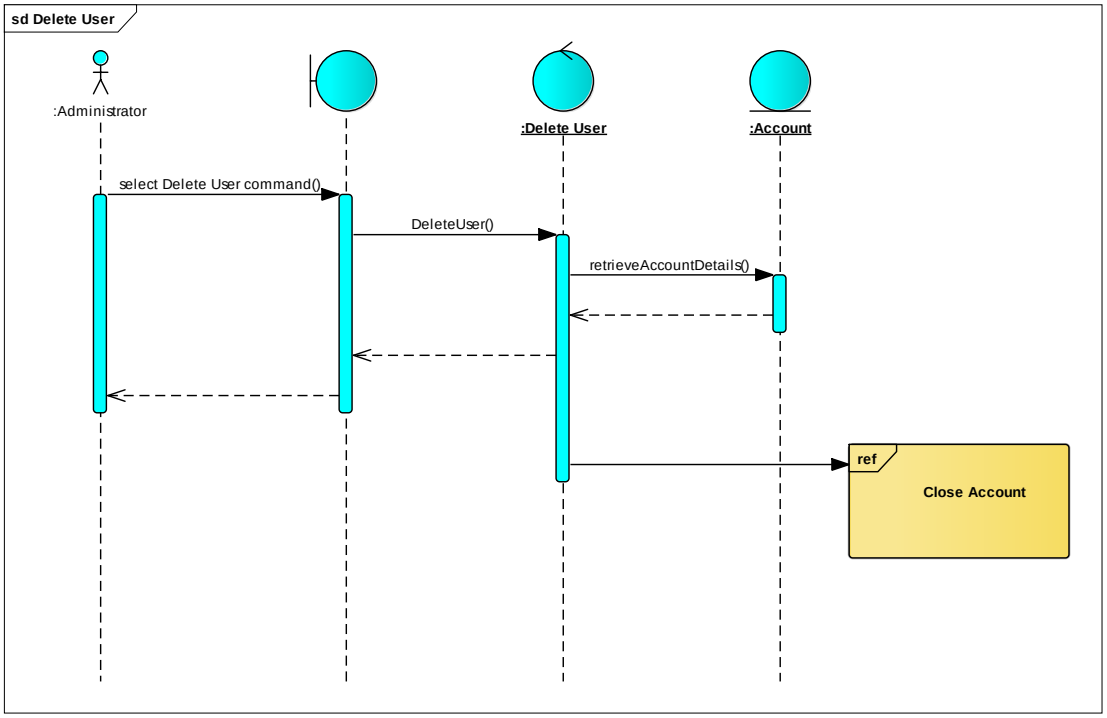


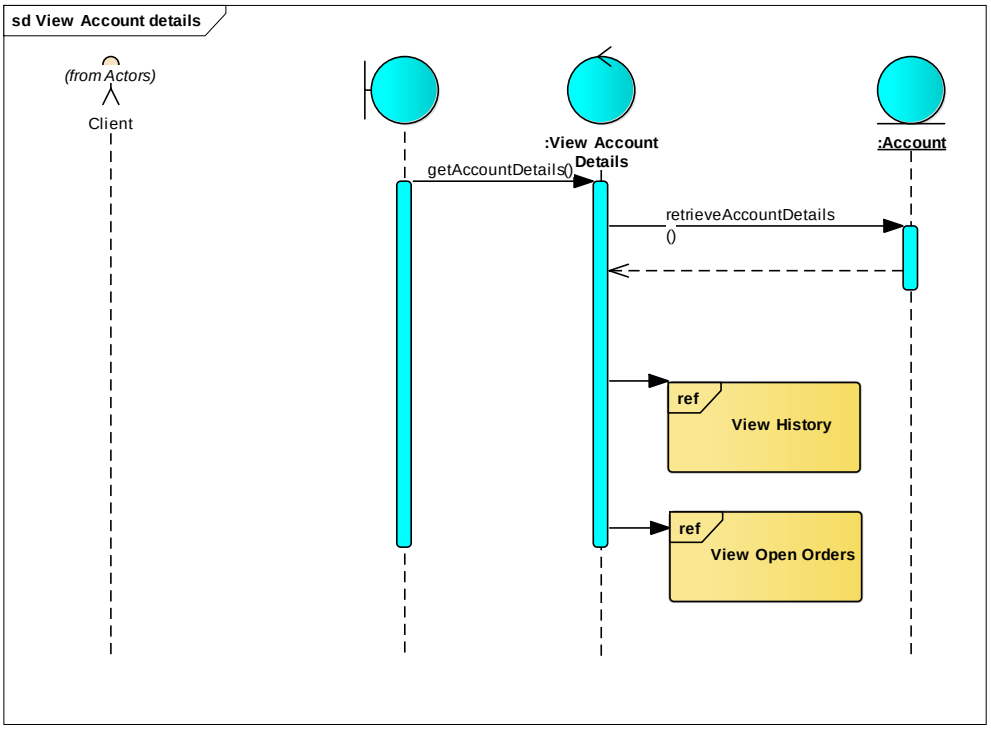
Sequence Diagram : Delete User



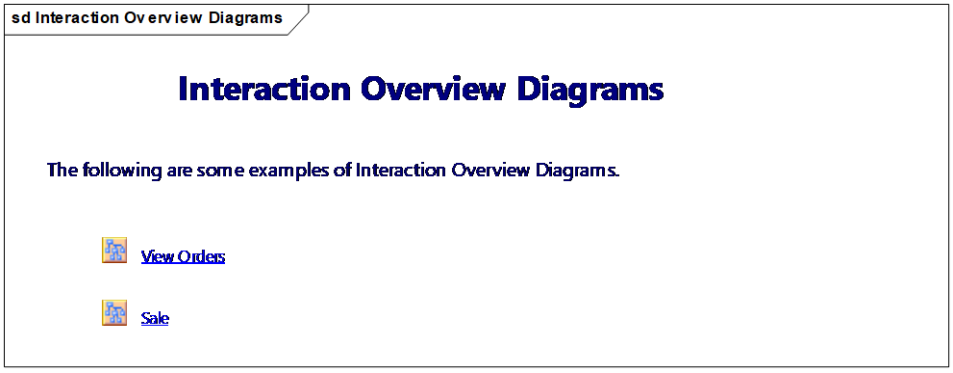
Sequence Diagrams

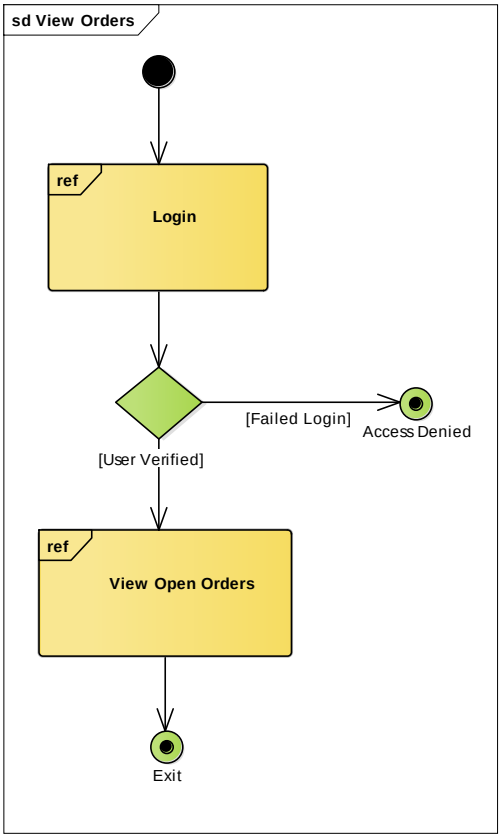


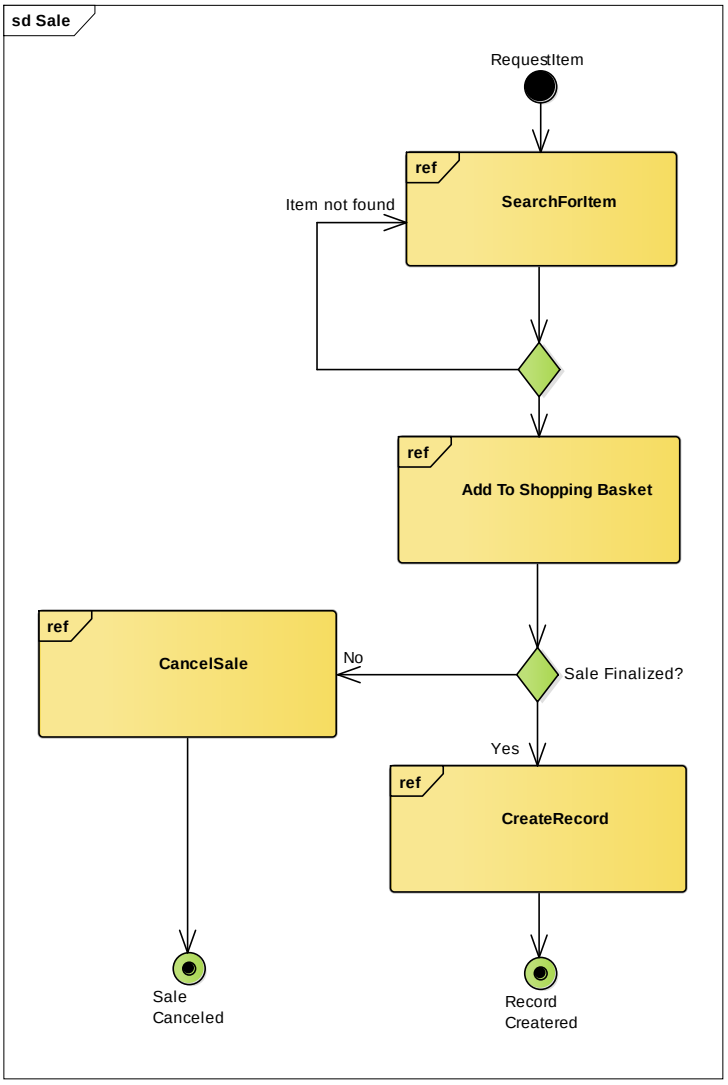

sd Delete User
:Administrator
:Delete User
:Account
select Delete User command()
DeleteUser()
retrieveAccountDetails()
ref
Close Account


sd View Account details
(from Actors)
Client
:View Account Details
:Account
getAccountDetails()
retrieveAccountDetails ()
ref
View History
ref
View Open Orders

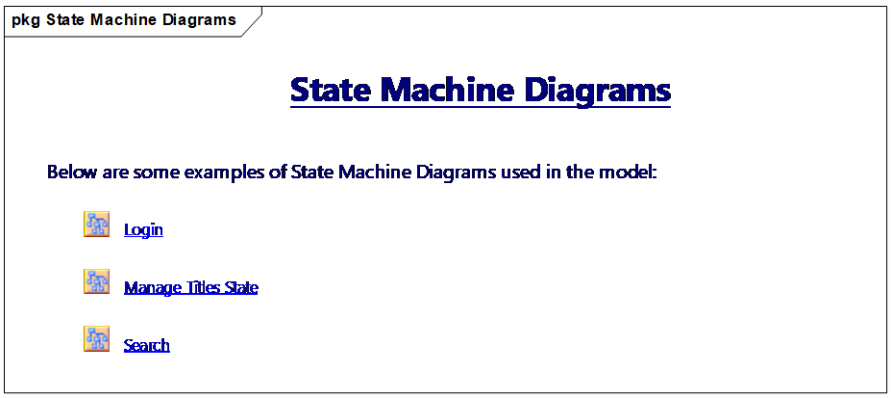
Interaction Overview Diagrams

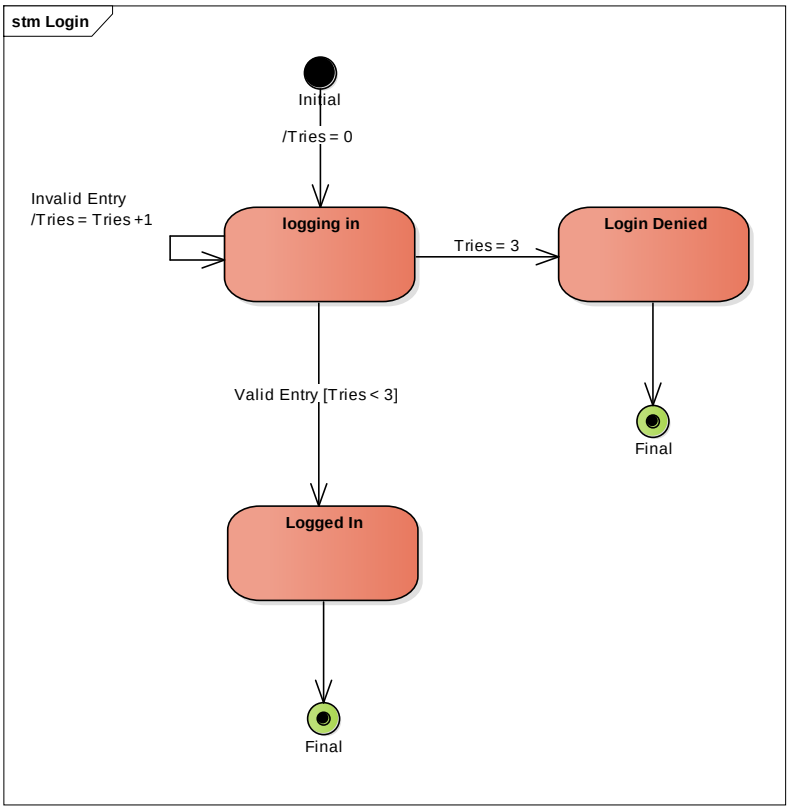


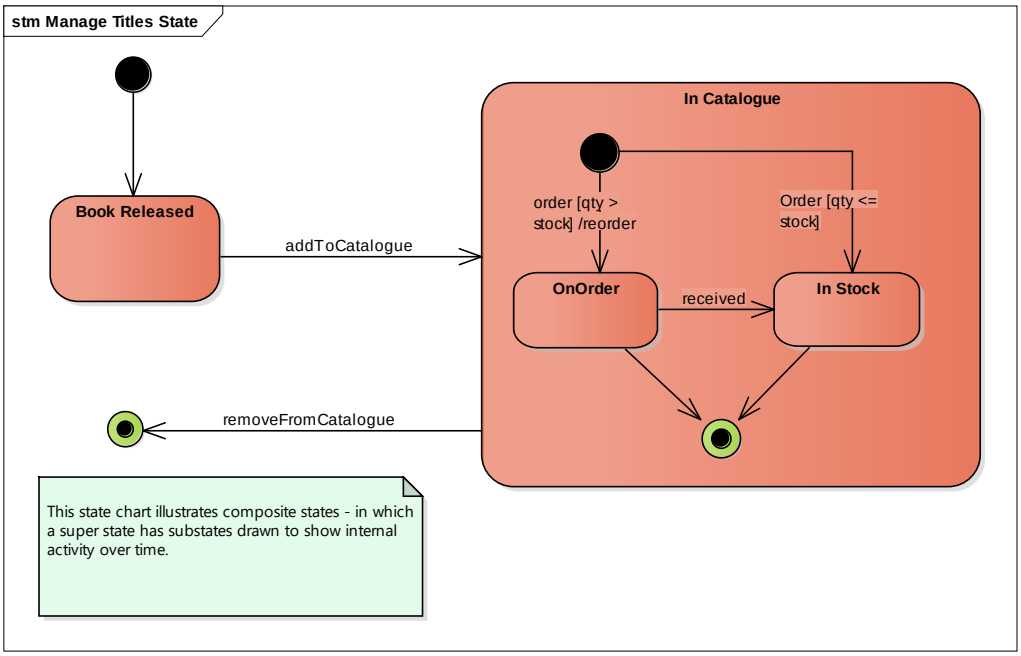


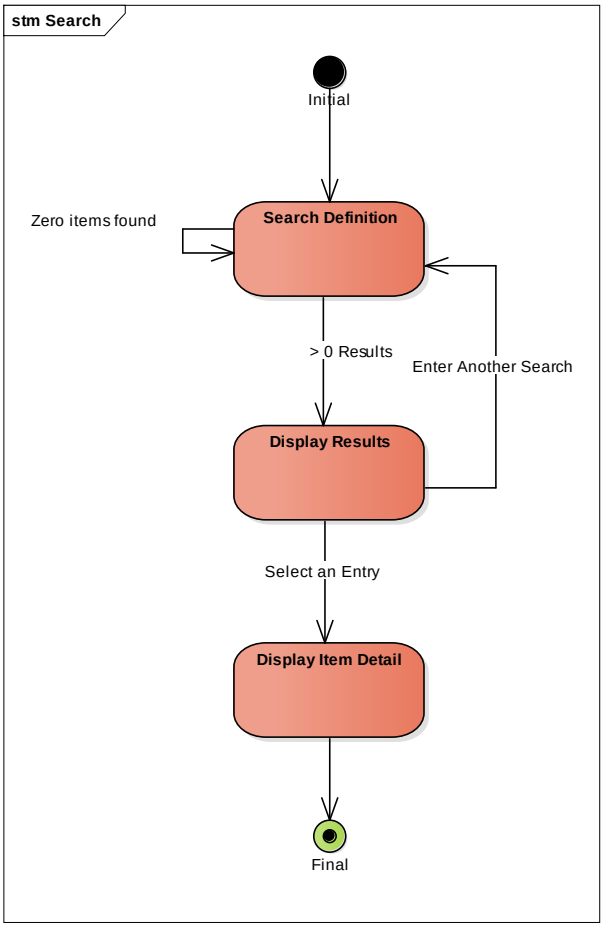


State Machine Diagrams



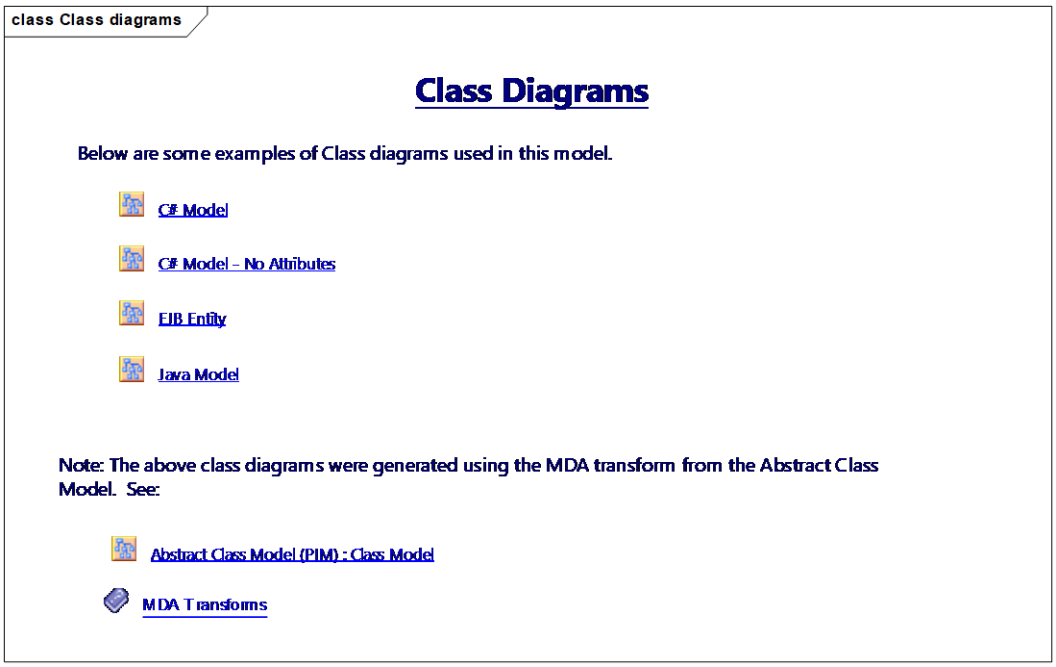

stm Login
Initial
/Tries = 0
Invalid Entry
/Tries = Tries +1
logging in
Tries = 3
Login Denied
Valid Entry [Tries < 3]
Final
Logged In
Final

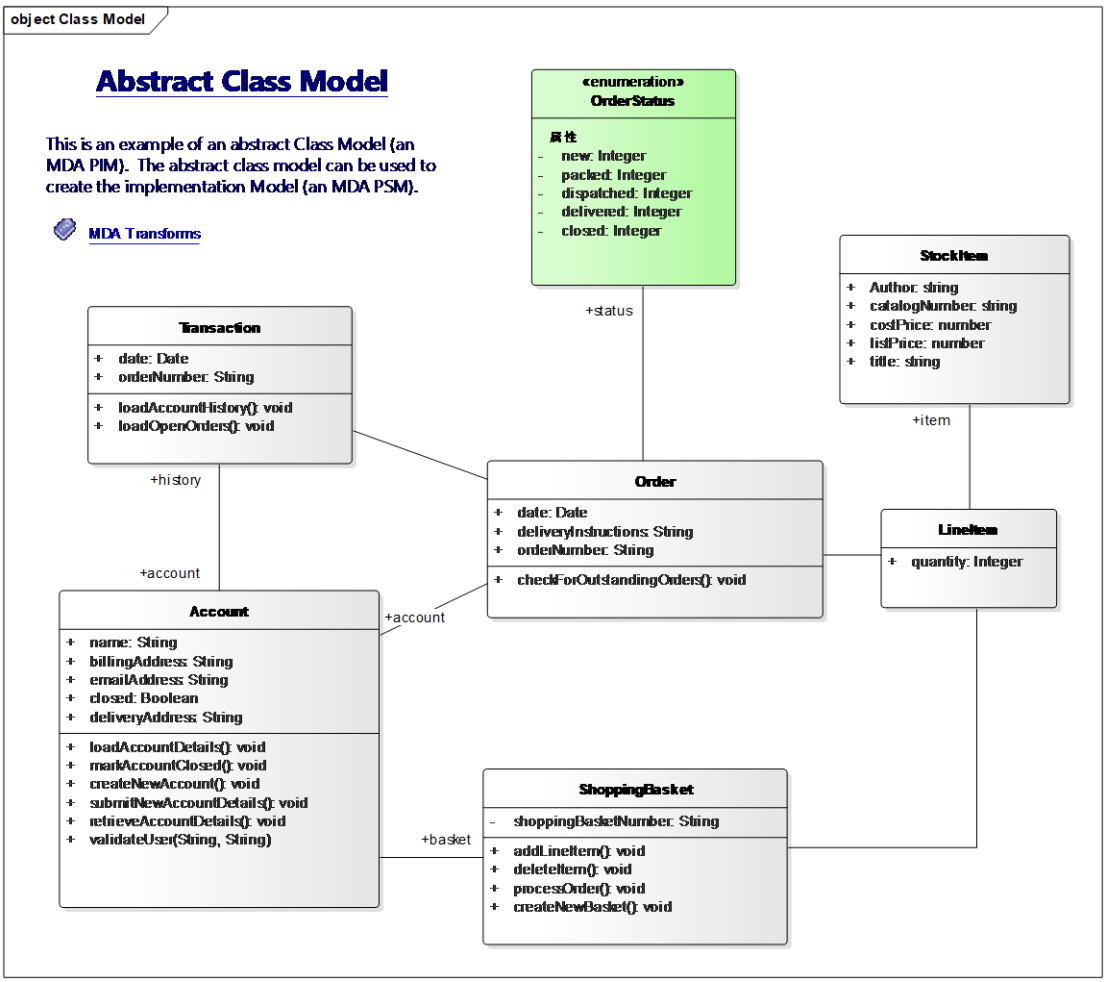

stm Manage Titles State
Book Released
addToCatalogue
In Catalogue
order [qty >
stock] /reorder
Order [qty <=
stock]
OnOrder
received
In Stock
removeFromCatalogue
This state chart illustrates composite states - in which a super state has substates drawn to show internal activity over time.

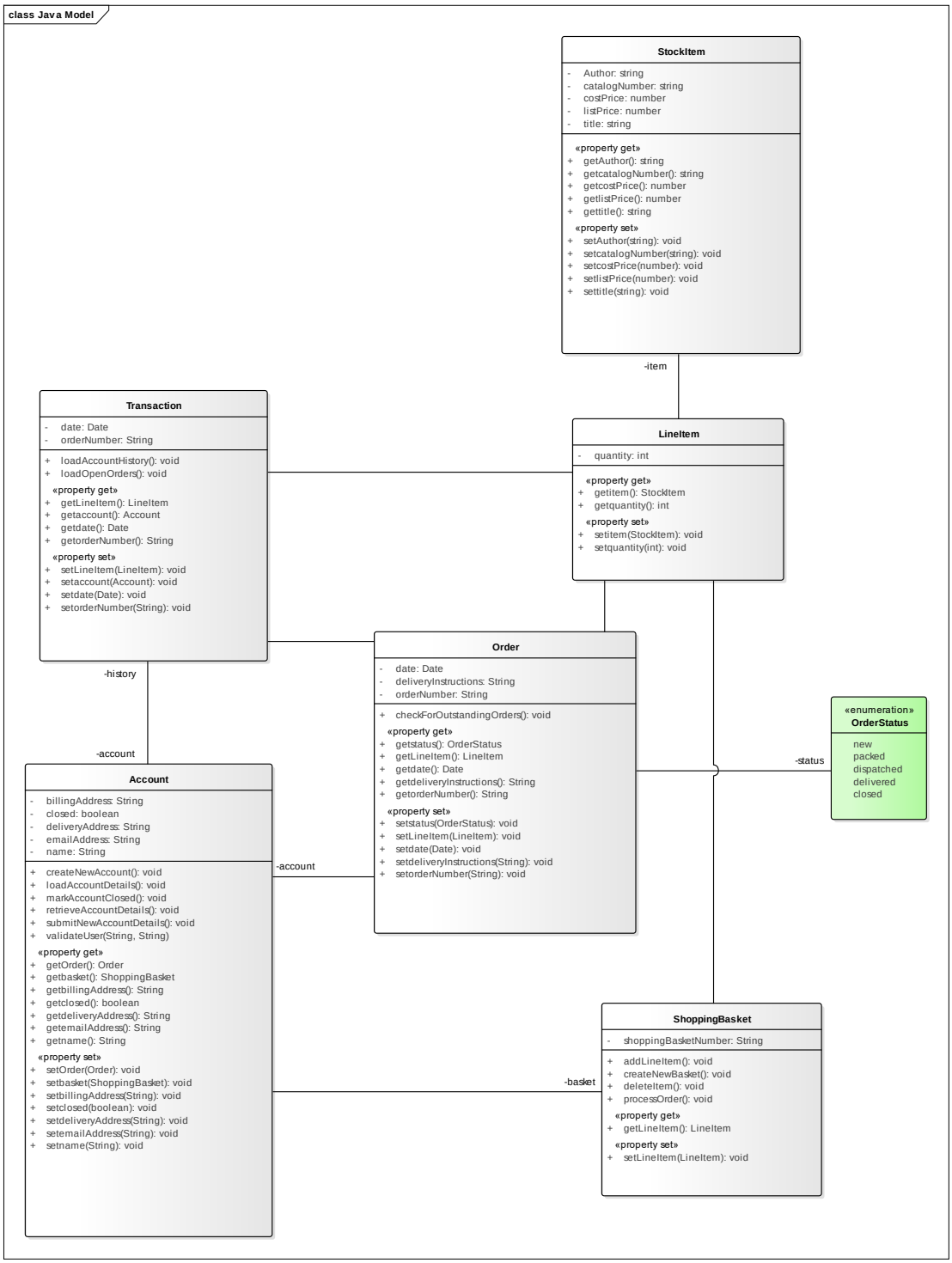

stm Search
Initial
Search Definition
Zero items found
> 0 Results
Enter Another Search
Display Results
Select an Entry
Display Item Detail
Final

Structural Diagrams

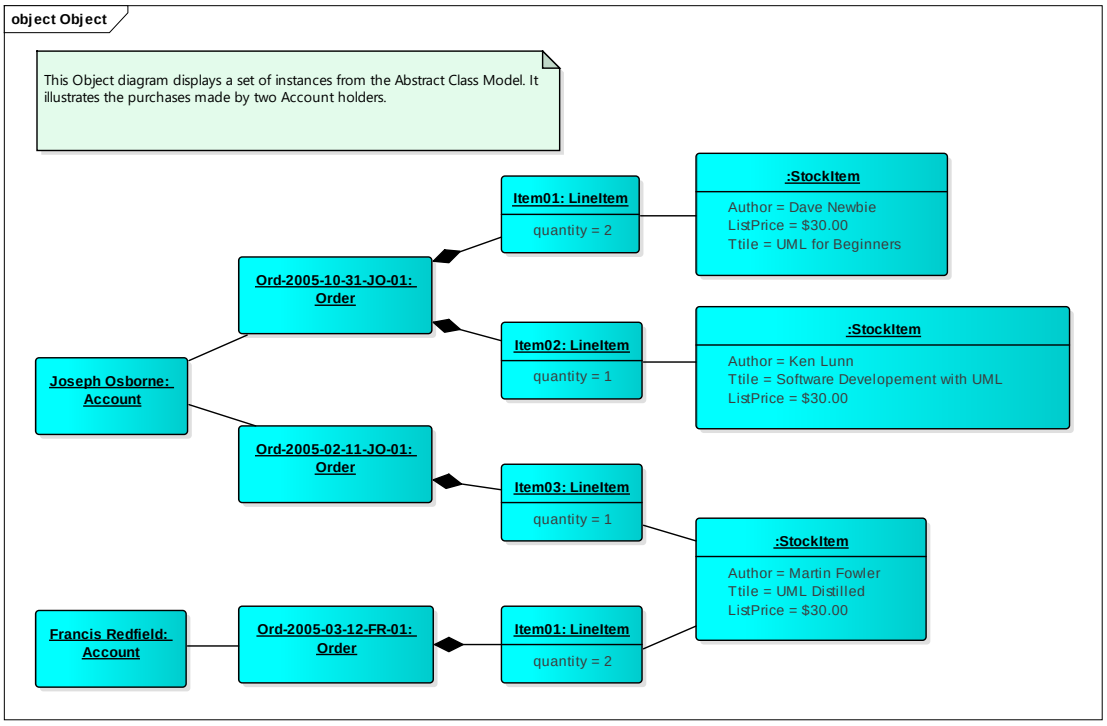
Class Diagrams



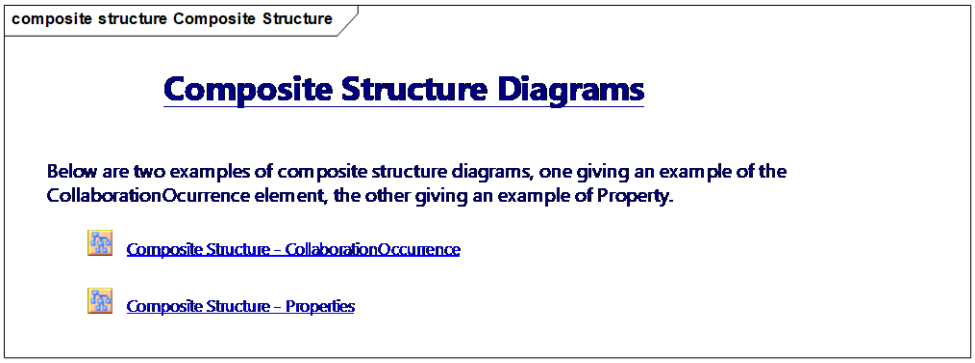

object Class Model
Abstract Class Model
This is an example of an abstract Class Model (an MDA PIM). The abstract class model can be used to create the implementation Model (an MDA PSM).
MDA Transforms
«enumeration»
OrderStatus
属性
- new: Integer
- packed: Integer
- dispatched: Integer
- delivered: Integer
- closed: Integer
+status
StockItem
+ Author: string
+ catalogNumber: string
+ costPrice: number
+ listPrice: number
+ title: string
+item
Transaction
+ date: Date
+ orderNumber: String
+ loadAccountHistory(): void
+ loadOpenOrders(): void
+history
+account
Order
+ date: Date
+ deliveryInstructions: String
+ orderNumber: String
+ checkForOutstandingOrders(): void
LineItem
+ quantity: Integer
Account
+account
+ name: String
+ billingAddress: String
+ emailAddress: String
+ closed: Boolean
+ deliveryAddress: String
+ loadAccountDetails(): void
+ markAccountClosed(): void
+ createNewAccount(): void
+ submitNewAccountDetails(): void
+ retrieveAccountDetails(): void
+ validateUser(String, String)
ShoppingBasket
- shoppingBasketNumber: String
+basket
+ addLineItem(): void
+ deleteItem(): void
+ processOrder(): void
+ createNewBasket(): void

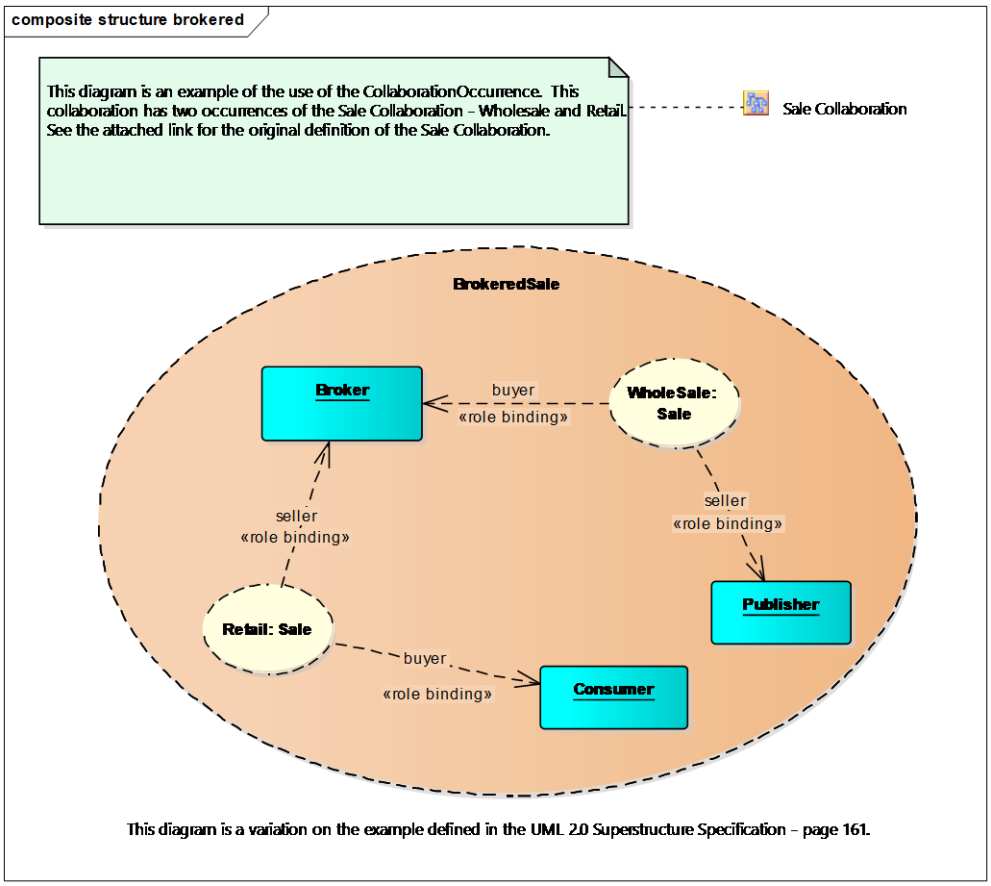

class Java Model
StockItem
- Author: string
- catalogNumber: string
- costPrice: number
- listPrice: number
- title: string
«property get»
+ getAuthor(): string
+ getcatalogNumber(): string
+ getcostPrice(): number
+ getlistPrice(): number
+ gettitle(): string
«property set»
+ setAuthor(string): void
+ setcatalogNumber(string): void
+ setcostPrice(number): void
+ setlistPrice(number): void
+ settitle(string): void
-item
Transaction
- date: Date
- orderNumber: String
+ loadAccountHistory(): void
+ loadOpenOrders(): void
«property get»
+ getLineItem(): LineItem
+ getaccount(): Account
+ getdate(): Date
+ getorderNumber(): String
«property set»
+ setLineItem(LineItem): void
+ setaccount(Account): void
+ setdate(Date): void
+ setorderNumber(String): void
LineItem
- quantity: int
«property get»
+ getitem(): StockItem
+ getquantity(): int
«property set»
+ setitem(StockItem): void
+ setquantity(int): void
-history
-account
Order
- date: Date
- deliveryInstructions: String
- orderNumber: String
+ checkForOutstandingOrders(): void
«property get»
+ getstatus(): OrderStatus
+ getLineItem(): LineItem
+ getdate(): Date
+ getdeliveryInstructions(): String
+ getorderNumber(): String
«property set»
+ setstatus(OrderStatus): void
+ setLineItem(LineItem): void
+ setdate(Date): void
+ setdeliveryInstructions(String): void
+ setorderNumber(String): void
-status
«enumeration»
OrderStatus
new
packed
dispatched
delivered
closed
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»
+ getOrder(): Order
+ getbasket(): ShoppingBasket
+ getbillingAddress(): String
+ getclosed(): boolean
+ getdeliveryAddress(): String
+ getemailAddress(): String
+ getname(): String
«property set»
+ setOrder(Order): void
+ setbasket(ShoppingBasket): void
+ setbillingAddress(String): void
+ setclosed(boolean): void
+ setdeliveryAddress(String): void
+ setemailAddress(String): void
+ setname(String): void
-account
-basket
ShoppingBasket
- shoppingBasketNumber: String
+ addLineItem(): void
+ createNewBasket(): void
+ deleteItem(): void
+ processOrder(): void
«property get»
+ getLineItem(): LineItem
«property set»
+ setLineItem(LineItem): void

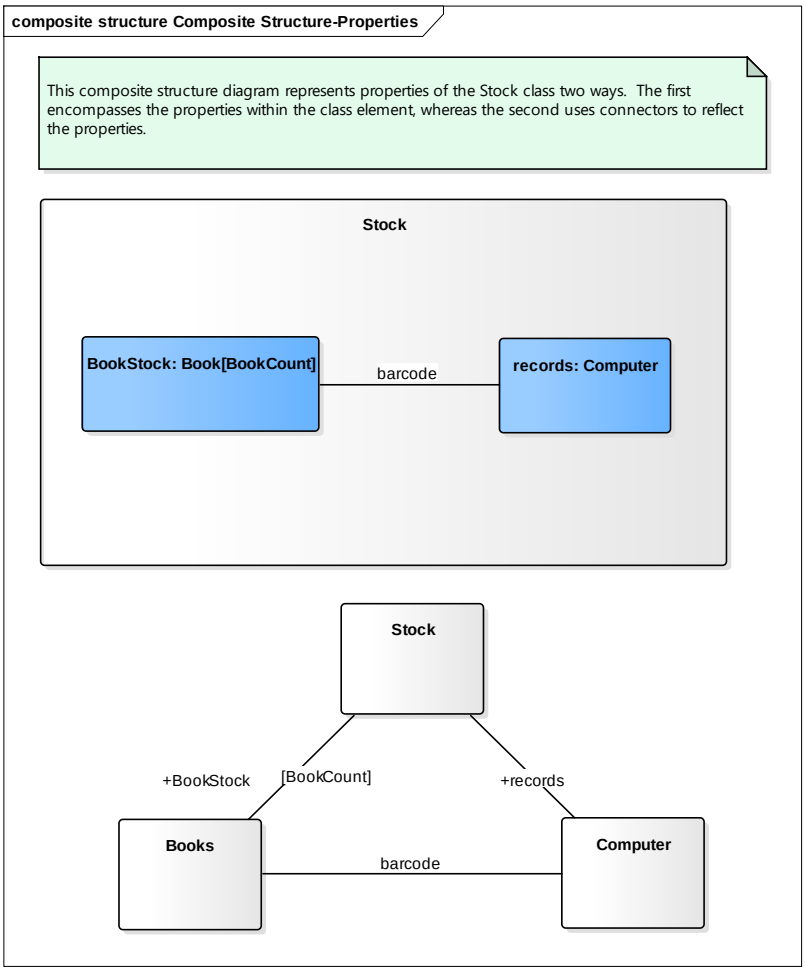
Object Diagram



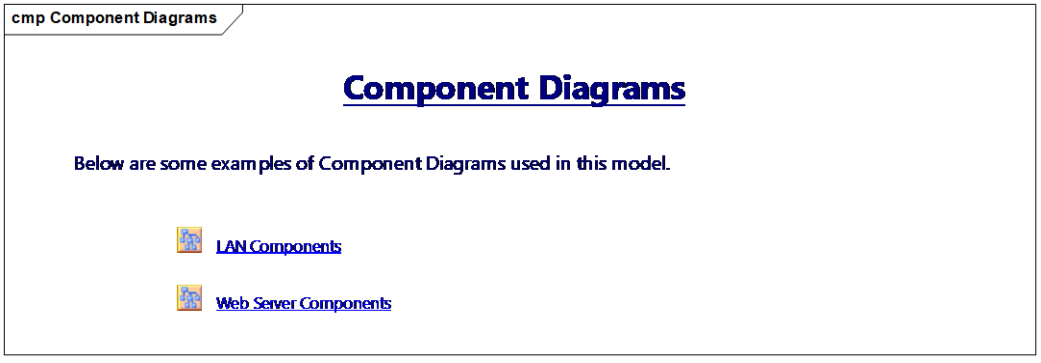
Composite Structure Diagram

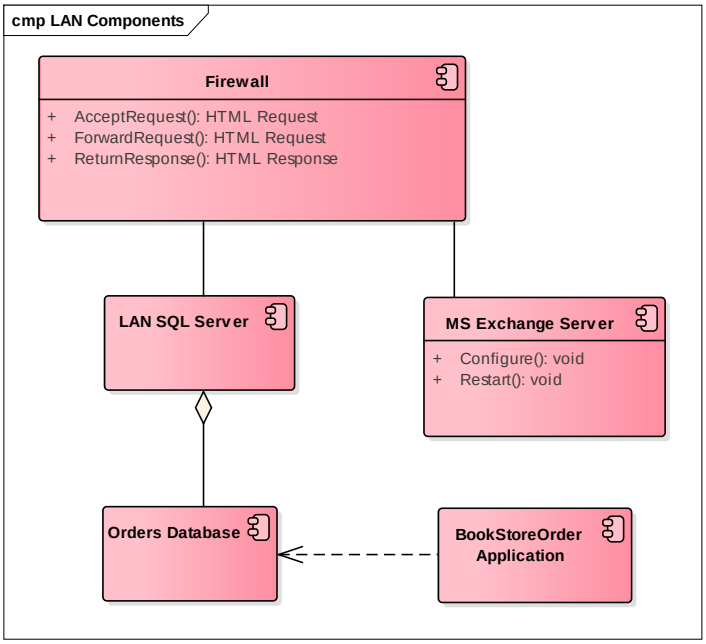


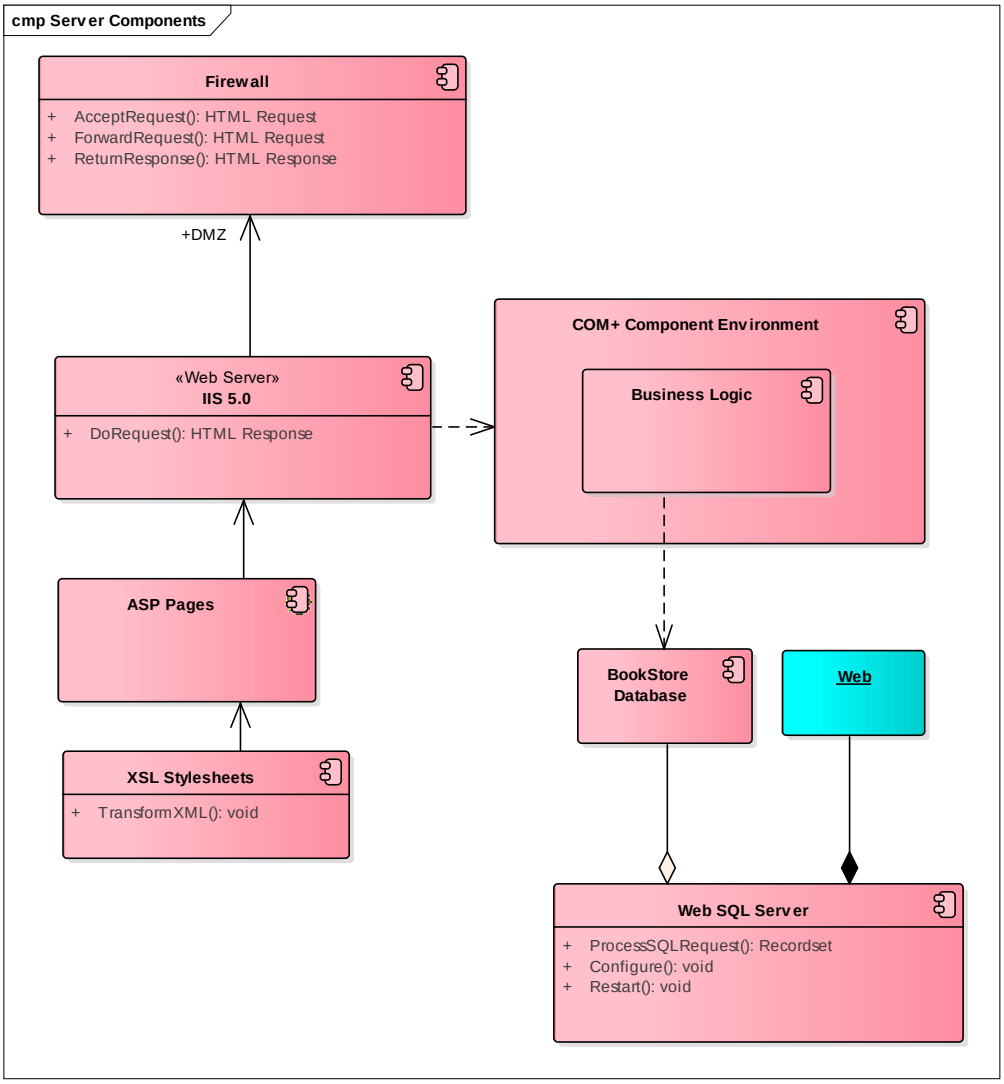

composite structure brokered
This diagram is an example of the use of the CollaborationOccurrence. This collaboration has two occurrences of the Sale Collaboration - Wholesale and Retail. See the attached link for the original definition of the Sale Collaboration.
Sale Collaboration
BrokeredSale
Broker
buyer
«role binding»
WholeSale: Sale
seller
«role binding»
seller
«role binding»
Publisher
Retail: Sale
buyer
«role binding»
Consumer
This diagram is a variation on the example defined in the UML 2.0 Superstructure Specification - page 161.



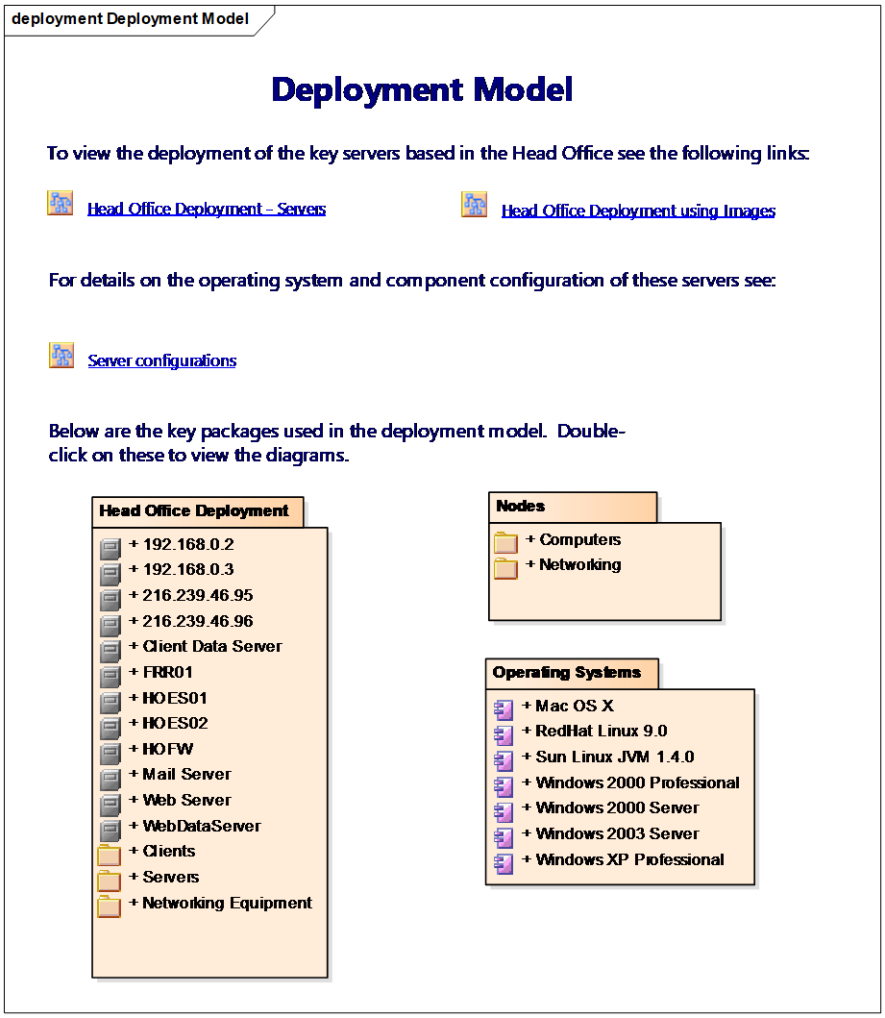
Component Diagrams

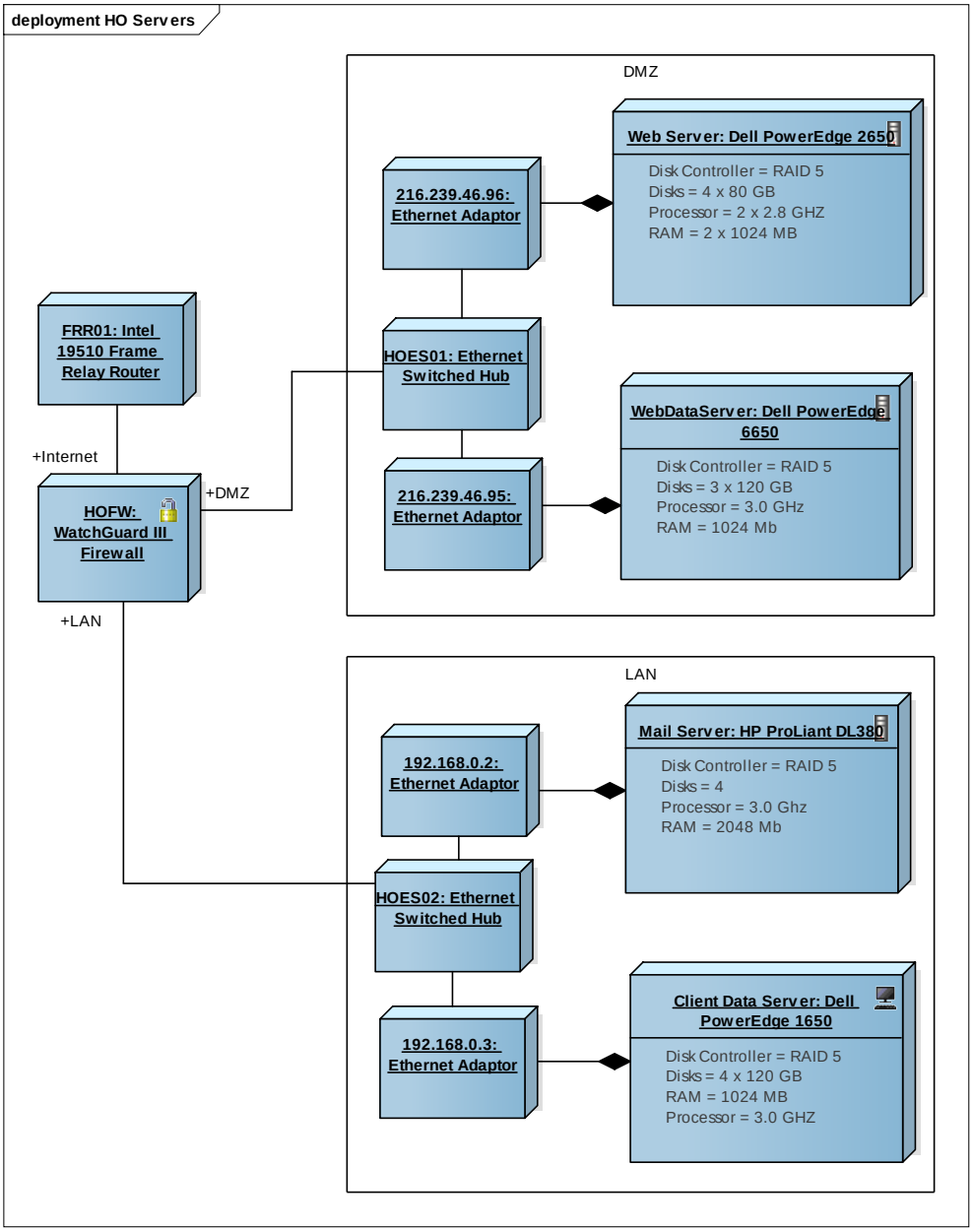


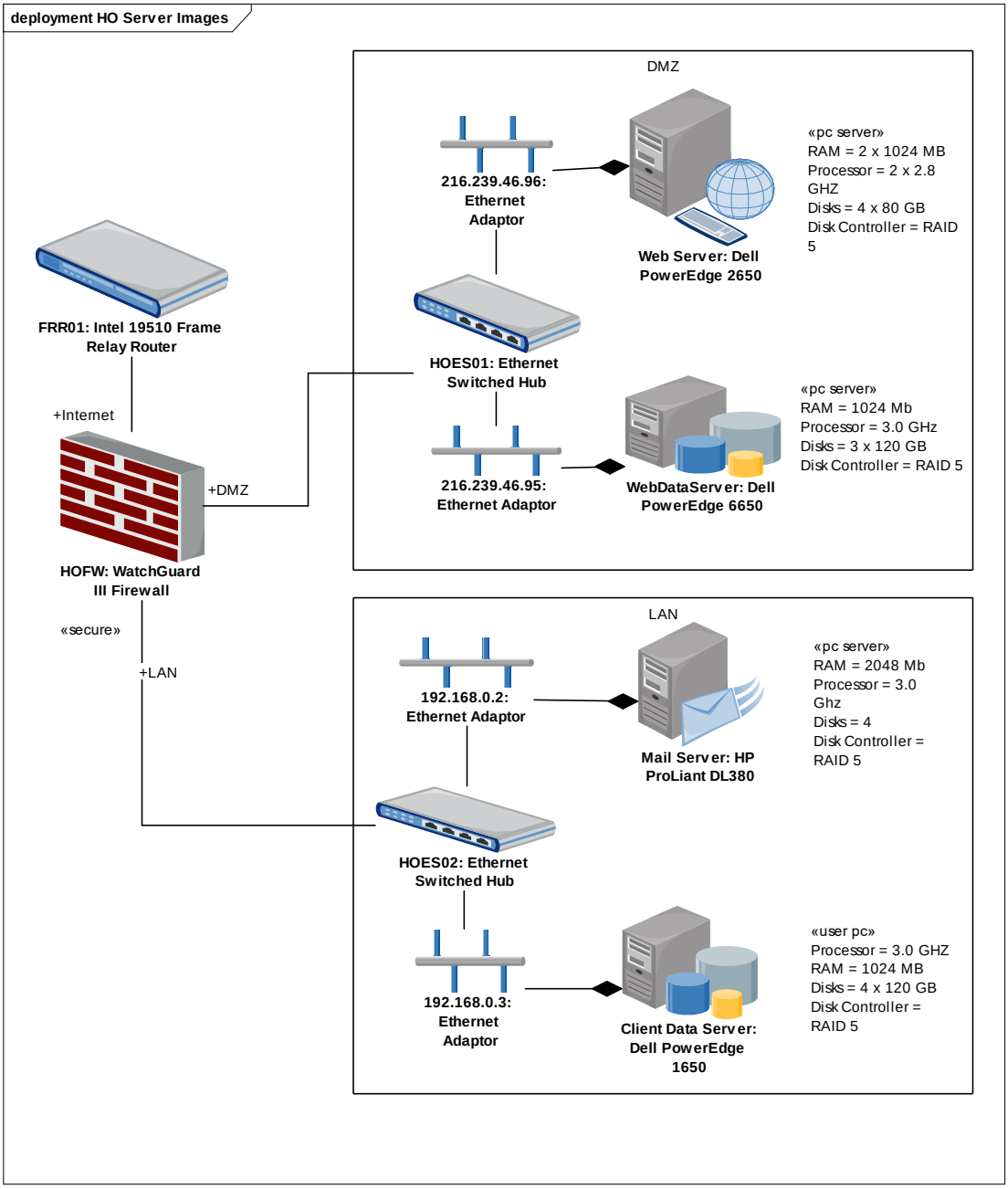

cmp LAN Components
Firewall
+ AcceptRequest(): HTML Request
+ ForwardRequest(): HTML Request
+ ReturnResponse(): HTML Response
LAN SQL Server
MS Exchange Server
+ Configure(): void
+ Restart(): void
Orders Database
BookStoreOrder Application


cmp Server Components
Firewall
+ AcceptRequest(): HTML Request
+ ForwardRequest(): HTML Request
+ ReturnResponse(): HTML Response
+DMZ
«Web Server»
IIS 5.0
+ DoRequest(): HTML Response
COM+ Component Environment
Business Logic
ASP Pages
BookStore Database
Web
XSL Stylesheets
+ TransformXML(): void
Web SQL Server
+ ProcessSQLRequest(): Recordset
+ Configure(): void
+ Restart(): void

Deployment Model



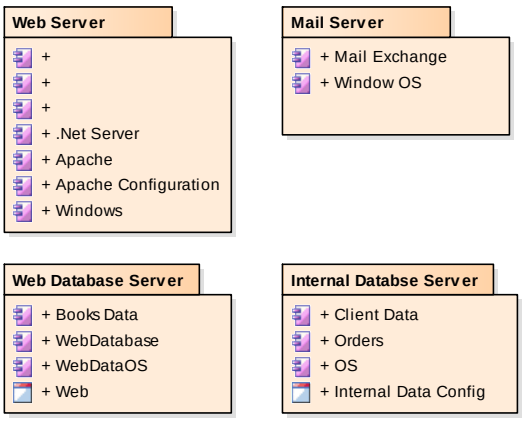

deployment HO Servers
DMZ
Web Server: Dell PowerEdge 2650
Disk Controller = RAID 5
Disks = 4 x 80 GB
Processor = 2 x 2.8 GHZ
RAM = 2 x 1024 MB
216.239.46.96: Ethernet Adaptor
FRR01: Intel 19510 Frame Relay Router
HOES01: Ethernet Switched Hub
WebDataServer: Dell PowerEdge 6650
Disk Controller = RAID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


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

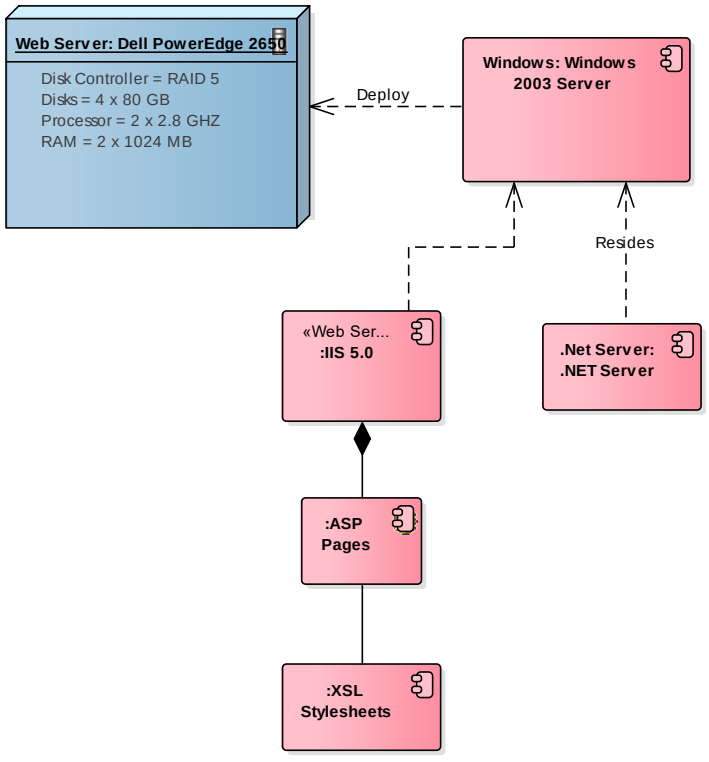
deployment Servers

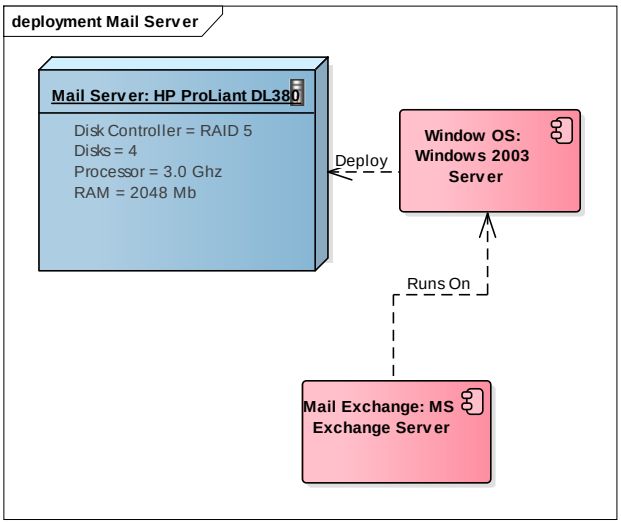
Server Configurations

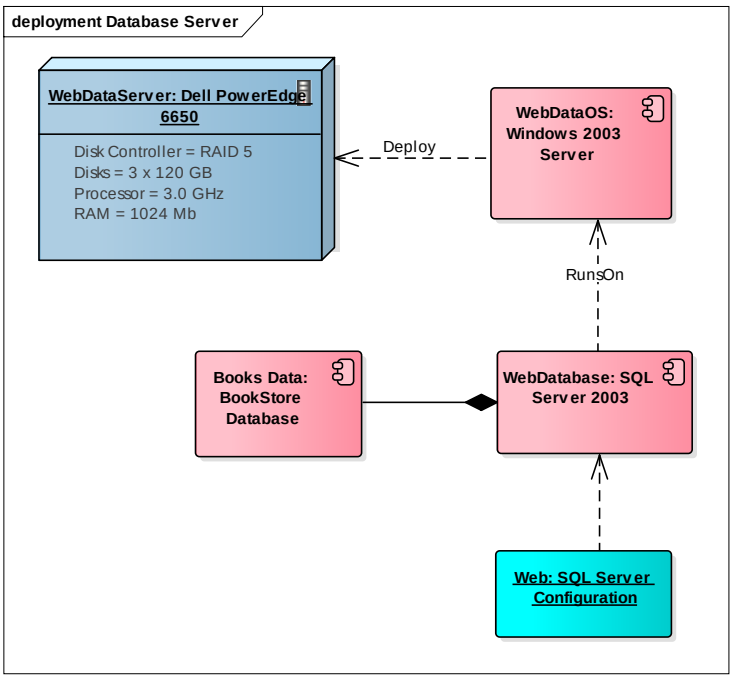
Below are the packages containing the server configurations. These packages include diagrams with the details of the components deployed on these servers. To access a server configuration diagram double click on the package.

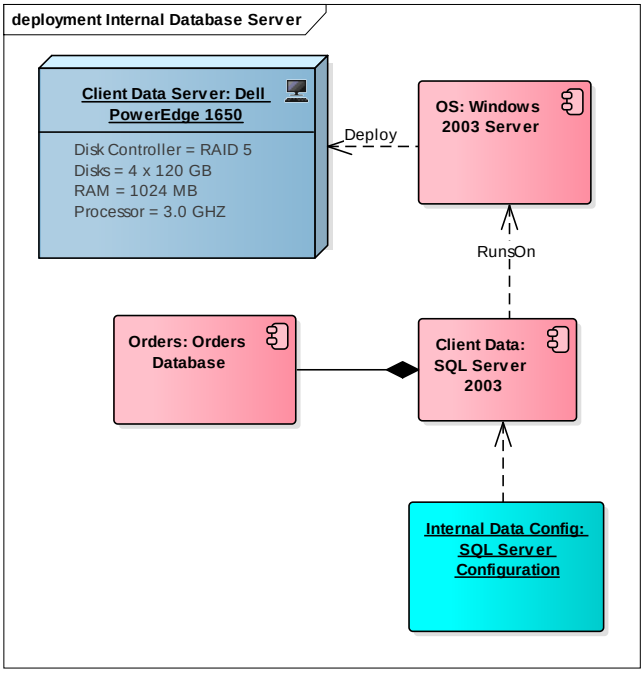


deployment Web Server

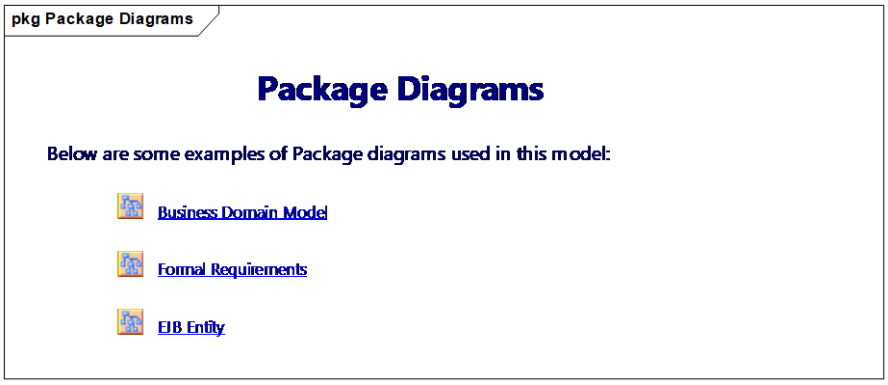



deployment Mail Server
Mail Server: HP ProLiant DL380
Disk Controller = RAID 5
Disks = 4
Processor = 3.0 Ghz
RAM = 2048 Mb
Deploy
Window OS: Windows 2003 Server
Runs On
Mail Exchange: MS Exchange Server


deployment Database Server
WebDataServer: Dell PowerEdge 6650
Disk Controller = RAID 5
Disks = 3 x 120 GB
Processor = 3.0 GHz
RAM = 1024 Mb
Deploy
WebDataOS: Windows 2003 Server
RunsOn
Books Data: BookStore Database
WebDatabase: SQL Server 2003
Web: SQL Server Configuration



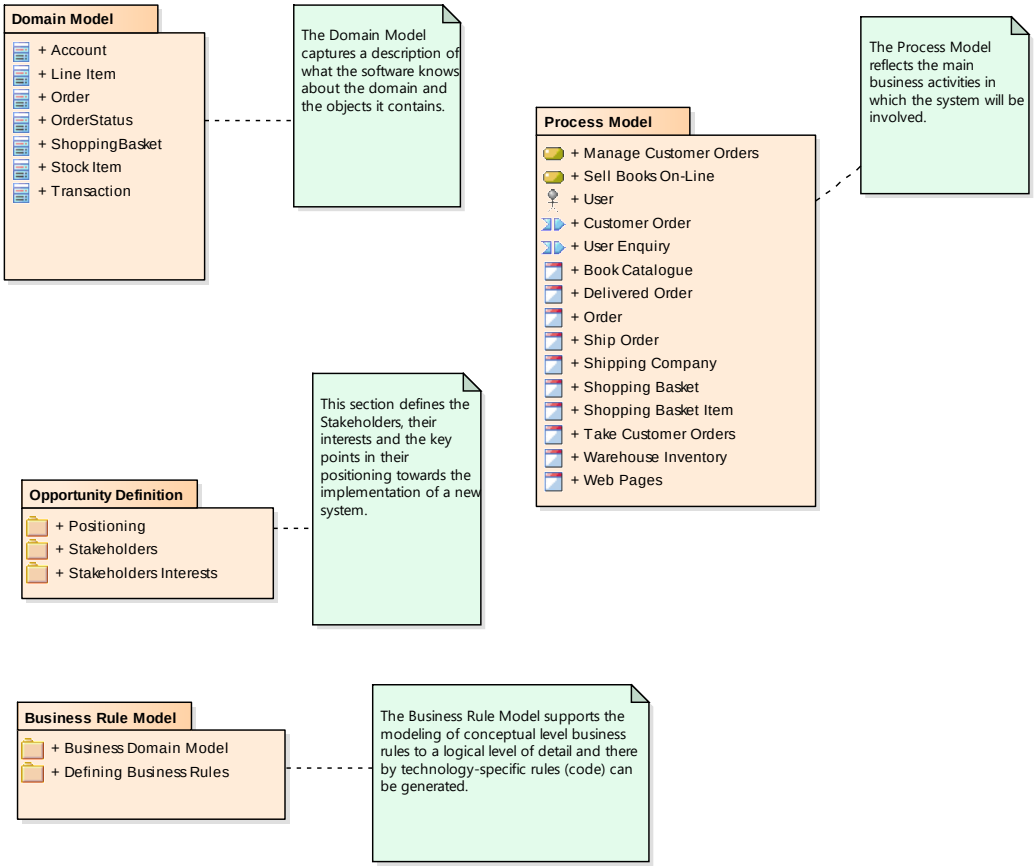
Package Diagram



pkg Business Domain Model

Business Domain Model

Below are the key packages in the Business Domain Model of TheMurray.com.au bookshop.
Double-click on the packages to view the key diagrams.



pkg Functional Requirements

Formal Requirements

Enterprise Architect allows you to document requirements graphically using the Requirement element. The Requirement element is available from the 'Requirements' Toolbox folder.

Using a Requirement element in the UML model, allows relationships to be drawn between requirements. It also allows for direct traceability to other aspects of the model such as Use Cases, Test Cases and other Analysis or Design elements.

The requirement element can be used to model or document any requirements, ranging from formal business requirements through to performance or security requirements.

Double-click on the package elements below to view the diagrams for these packages.

See Help:Requirements

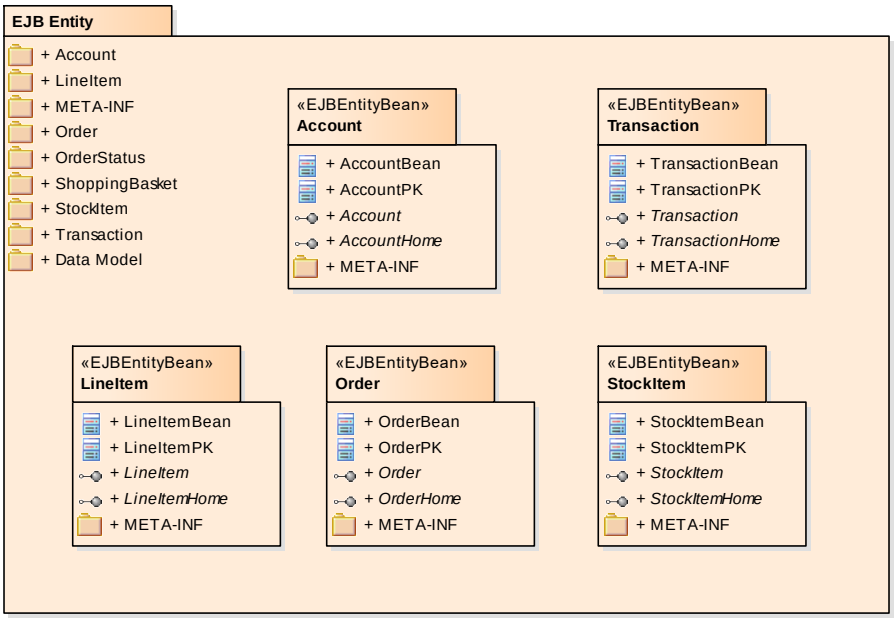
Manage Users
- + REQ011 - Manage User Accounts
- + REQ016 - Add Users
- + REQ017 - Remove User
- + REQ018 - Report on User Account
- + REQ024 - Secure Access
- + REQ025 - Store User Details
- + REQ026 - Validate User

Manage Inventory
- + REQ019 - Manage Inventory
- + REQ020 - Receive Books
- + REQ021 - List Stock Levels
- + REQ022 - Order Books
- + REQ023 - Store and Manage Books
- + REQ027 - Add Books
- + REQ032 - Update Inventory

Take Orders
- + REQ012 - Provide Online Sales
- + REQ014 - ShoppingBasket
- + REQ015 - Process Credit Card Payment

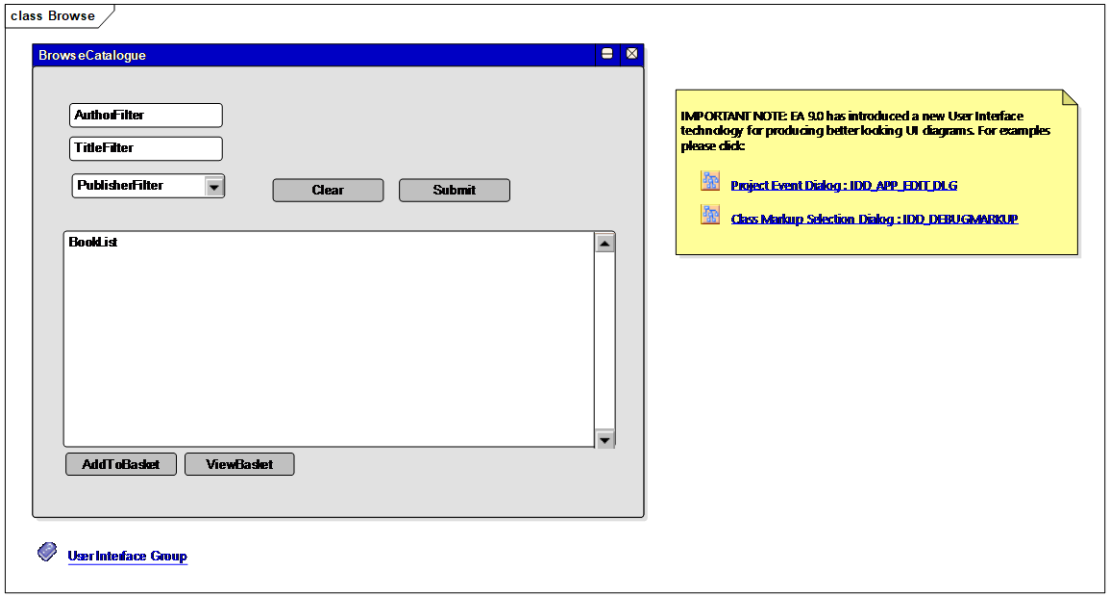
Fulfill Orders
- + REQ013 - Manage Deliveries
- + REQ028 - Process Order
- + REQ029 - Ship Order
- + REQ030 - Package Order
- + REQ031 - List Current Orders
- + REQ033 - Retrieve Books

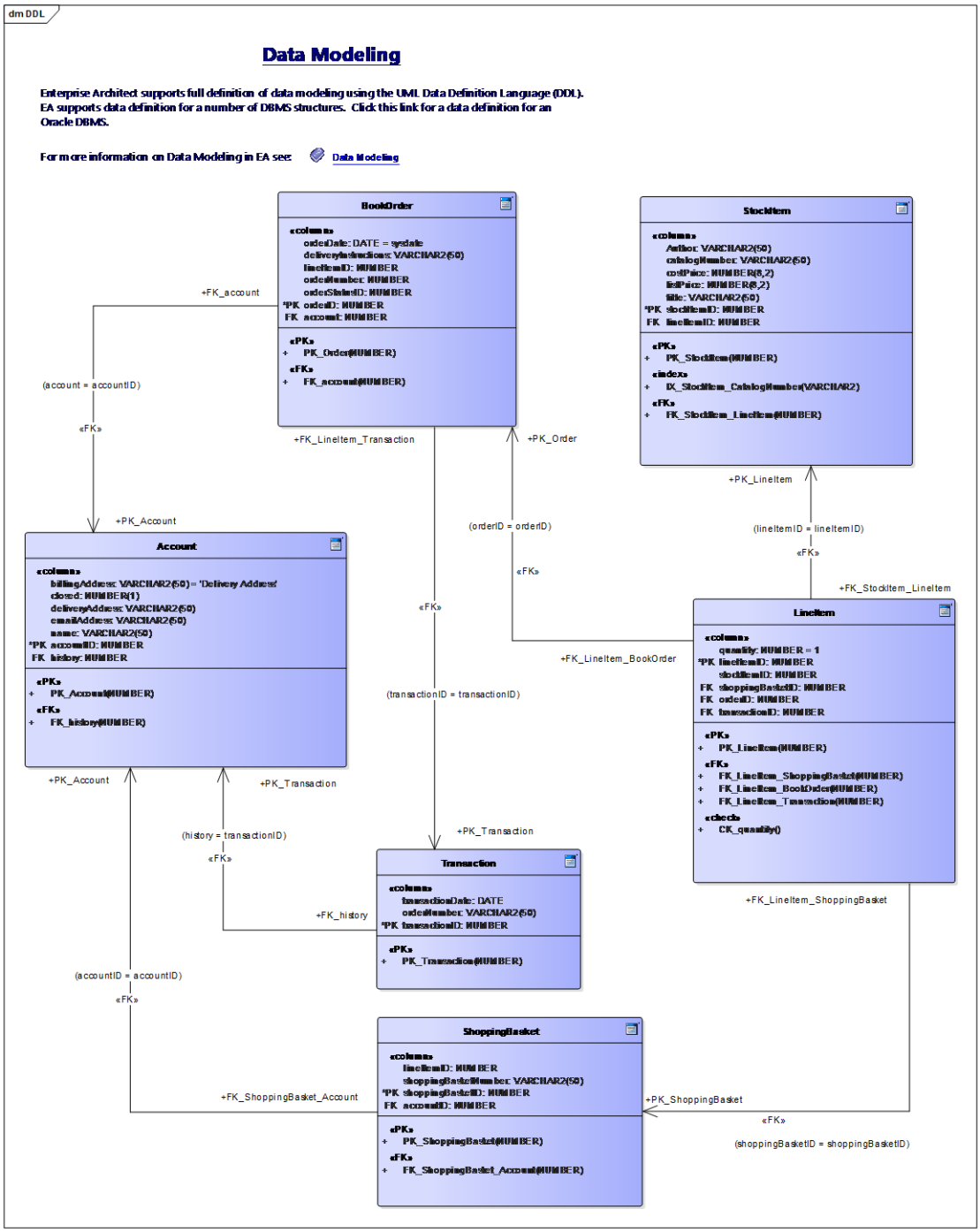
pkg EJB Entity



(from Software Engineering)

UI


class Browse
BrowseCatalogue
AuthorFilter
TitleFilter
PublisherFilter
Clear
Submit
BookList
AddToBasket
ViewBasket
User Interface Group
IMPORTANT NOTE: EA 9.0 has introduced a new User Interface technology for producing better looking UI diagrams. For examples please click:
Project Event Dialog : IDD_APP_EDIT_DLG
Class Markup Selection Dialog : IDD_DEBUGMARKUP

Database Engineering

Data Modeling