

# *Shopping Cart*

## ***Analysis & Design***

**Author: John Smith**

Version: 1.7

Status: Draft

Publication: 06/01/2009

Copyright: Modeliosoft

**Modeliosoft**

21, avenue Victor Hugo 75016 Paris France

## Table des matières

<b>1</b>	<b>Introduction .....</b>	<b>5</b>
<b>2</b>	<b>Use Cases .....</b>	<b>6</b>
2.1	Actors .....	6
2.2	Use Cases.....	6
2.2.1	Use Case "Actors" .....	7
2.2.2	Use Case "Online Shopping Overview" .....	7
2.2.3	Use Case "Account Management Overview" .....	8
2.2.4	Use Case "Catalog Management Overview" .....	8
2.2.5	Use Case "Product Management Overview" .....	9
2.2.6	Use Case "Order Management Overview" .....	9
2.2.7	Use Case "Use Cases Overview" .....	9
<b>3</b>	<b>Package Index.....</b>	<b>11</b>
<b>4</b>	<b>Package "ShoppingCart" .....</b>	<b>13</b>
<b>5</b>	<b>Package "Analysis" .....</b>	<b>14</b>
<b>6</b>	<b>Package "Actors" .....</b>	<b>15</b>
<b>7</b>	<b>Package "Processes" .....</b>	<b>16</b>
<b>8</b>	<b>Package "Use Cases" .....</b>	<b>17</b>
<b>9</b>	<b>Package "Online Shopping" .....</b>	<b>18</b>
<b>10</b>	<b>Package "Account Management" .....</b>	<b>19</b>
<b>11</b>	<b>Package "Catalog Management" .....</b>	<b>20</b>
<b>12</b>	<b>Package "Product Management" .....</b>	<b>21</b>
<b>13</b>	<b>Package "Order Management" .....</b>	<b>22</b>
<b>14</b>	<b>Package "Domains" .....</b>	<b>23</b>
<b>15</b>	<b>Package "OnlineStore" .....</b>	<b>24</b>
15.1	Class "Account" .....	24
15.2	Class "ShoppingCart" .....	26
15.3	Class "CartItem" .....	27
15.4	Class "Catalog" .....	27
15.5	Class "OnlineStore" .....	28
<b>16</b>	<b>Package "Client" .....</b>	<b>29</b>
16.1	Class "Client" .....	29

16.2	Class "ApplicationForm" .....	30
16.3	Class "ApplicationField" .....	30
<b>17</b>	<b>Package "Products" .....</b>	<b>31</b>
17.1	Class "Product" .....	31
17.2	Class "Supplier" .....	32
17.3	Class "Order" .....	33
17.4	Class "OrderLineItem" .....	33
<b>18</b>	<b>Package "Bank" .....</b>	<b>34</b>
18.1	Class "Payment" .....	34
<b>19</b>	<b>Package "Base Types" .....</b>	<b>35</b>
19.1	DataType "Date" .....	35
19.2	DataType "Address" .....	35
19.3	DataType "CreditCard" .....	36
<b>20</b>	<b>Package "Design" .....</b>	<b>37</b>
<b>21</b>	<b>Package "Logical Architecture" .....</b>	<b>38</b>
<b>22</b>	<b>Package "Presentation" .....</b>	<b>39</b>
22.1	Class "CatalogBrowser" .....	39
22.2	Class "CartBrowser" .....	39
22.3	Class "CatalogAdmin" .....	40
22.4	Class "AccountAdmin" .....	40
<b>23</b>	<b>Package "Business" .....</b>	<b>41</b>
23.1	Class "Catalog" .....	41
23.2	Class "ShoppingCart" .....	41
23.3	Class "PaymentManager" .....	42
23.4	Class "AccountManager" .....	43
<b>24</b>	<b>Package "Exchange Types" .....</b>	<b>44</b>
24.1	Class "ItemsList" .....	44
24.2	Class "ProductDescription" .....	44
<b>25</b>	<b>Package "Implementation" .....</b>	<b>45</b>
25.1	Class "CartItem" .....	45
25.2	Class "ShoppingCart" .....	45
25.3	Class "Account" .....	45
<b>26</b>	<b>Package "Entity" .....</b>	<b>46</b>
26.1	Class "Product" .....	46
26.2	Class "CartItem" .....	46
26.3	Class "Account" .....	47
<b>27</b>	<b>Package "Deployment" .....</b>	<b>48</b>
27.1	Class "MySql" .....	49

27.2	Class "Struts" .....	49
28	Package "Documents" .....	50

# 1 Introduction

This project contains the model for the Online Shopping Cart system, including all the models used to specify and realize the IT system.

This model addresses the development of an online shopping cart system, that allows suppliers to sell their products directly online to customers. It is provided as a UML example covering the analysis of the business context and processes, and the design and deployment of a possible implementation. The goal is to show the power UML brings to analysing and designing a robust system that corresponds to the initial requirements.

This model can be used to create demos of the Modelio tool. It is documented in depth, in order to produce pertinent documents using Modelio DocumentPublisher. Some examples of documents are also included in this example, in the "Documents" package.

## 2 Use Cases

### 2.1 Actors

Actor	Description
<b>Client</b>	Person purchasing products online The client, also known as customer, is the person that logs onto the shopping cart online system to purchase products of his/her choice.
<b>Administrator</b>	Person responsible for the system. The administrator is the person in charge of managing and administering the system. He also assumes the role of supervisor in the sense that he enforces the "Terms of use" of the site, and has the right to revoke clients privileges by deleting their account.
<b>Supplier</b>	The person selling his products through the online shopping cart system. The supplier is the person or company providing the goods on sale in the online shopping cart system. It is his responsibility to populate the product catalog, and to update any information relative to the products on the site.
<b>Bank</b>	The supplier's bank The bank gets involved to debit the client's account and credit the supplier's account during a purchase.
<b>System</b>	Dummy actor representing the system. This actor is a dummy actor used to represent the system in interactions diagrams for the use cases.

*Table 1 Table of Actors*

### 2.2 Use Cases

Use-Cases	Description
<b>Manage Shopping Cart</b>	How the user manages his/her shopping cart.
<b>Check out</b>	Check out the current shopping cart.
<b>File Complaint</b>	Unsatisfied customer files a complaint.
<b>Manage Client Account</b>	Client account management operations.
<b>Manage Product Catalog</b>	Manage products in the catalog.
<b>Recall Product</b>	Recall a defective product.
<b>Advertise Product</b>	Online product advertizing.
<b>Ship Order</b>	Ship the order to the client
<b>Handle Complaint</b>	Handle a client's complaint.

*Table 2 Table of Use Cases*

### 2.2.1 Use Case "Actors"

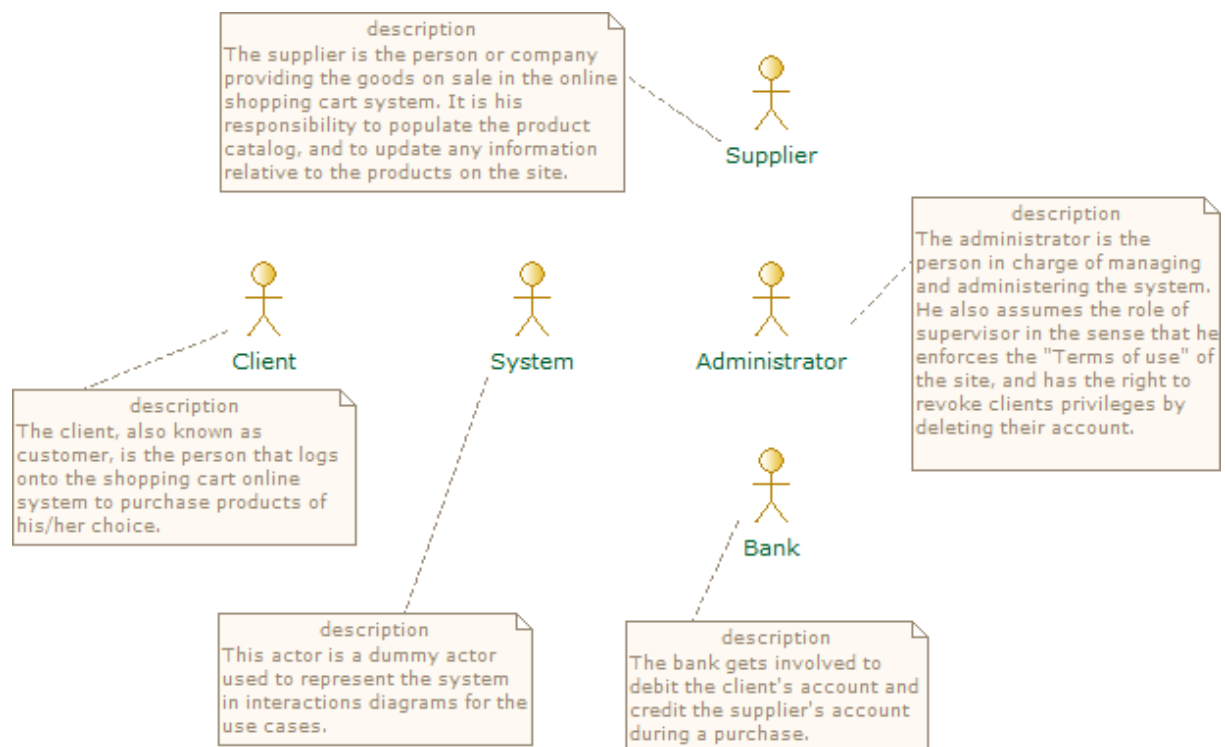


Figure 1 : Actors

### 2.2.2 Use Case "Online Shopping Overview"

This diagram shows the actors and their interactions with the use cases they can play against the system. This domain contains the use cases related to the client's shopping activities.

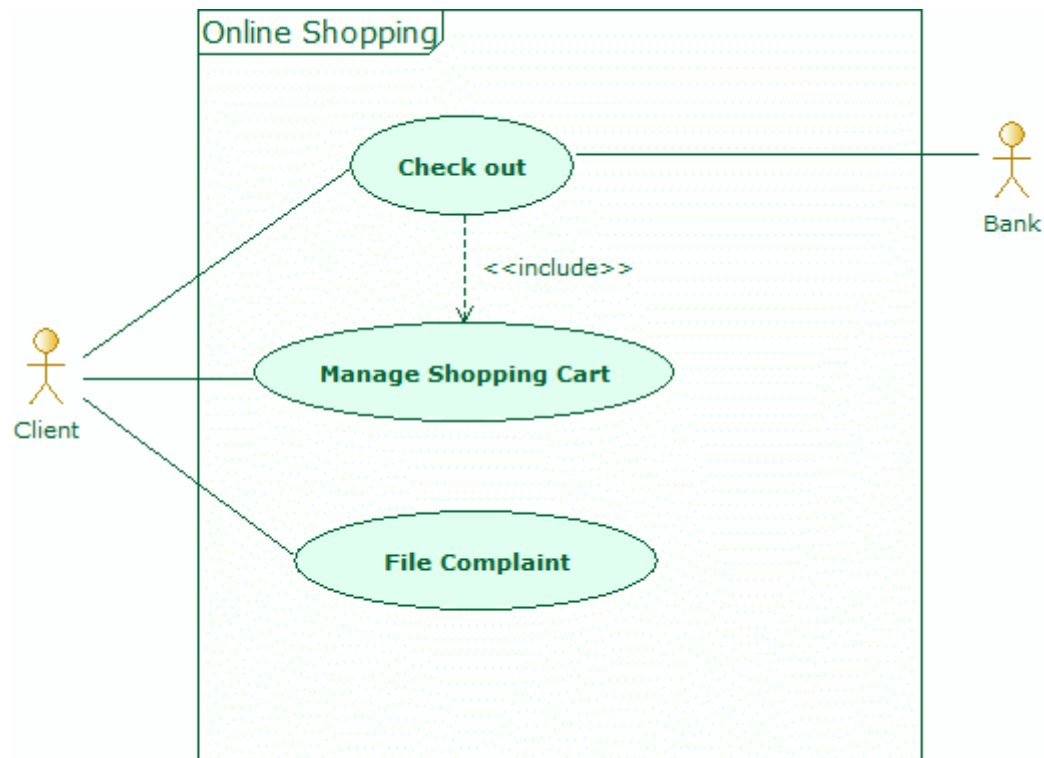


Figure 2 : Online Shopping Overview

### 2.2.3 Use Case "Account Management Overview"

This diagram shows all the use cases and actors related to the management of the client's accounts.

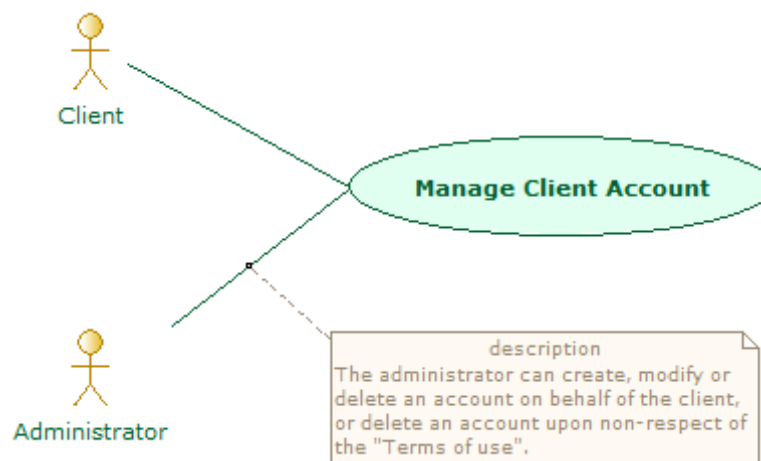
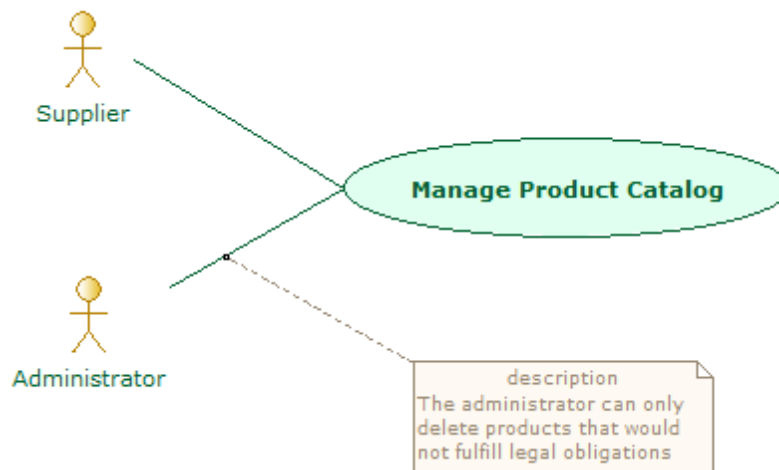


Figure 3 : Account Management Overview

### 2.2.4 Use Case "Catalog Management Overview"

This diagram shows the use cases and the actors related to the management of the product catalog for the online shopping cart.

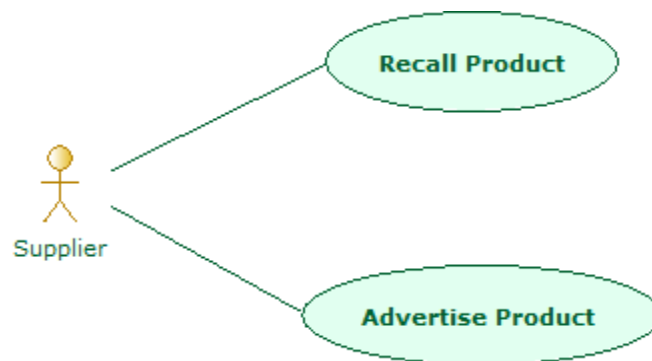




**Figure 4 : Catalog Management Overview**

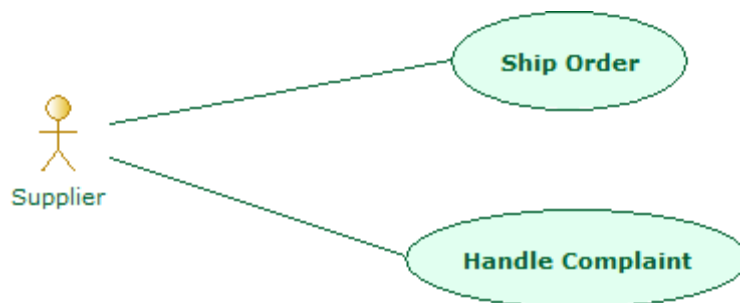
### 2.2.5 Use Case "Product Management Overview"

This diagram shows all use cases related to the management of the products on offer for sale in the online shopping cart system.



**Figure 5 : Product Management Overview**

### 2.2.6 Use Case "Order Management Overview"



**Figure 6 : Order Management Overview**

### 2.2.7 Use Case "Use Cases Overview"

This diagram includes all the use cases identified for the online shopping cart system. Use cases are organised in containers regrouping related use cases.

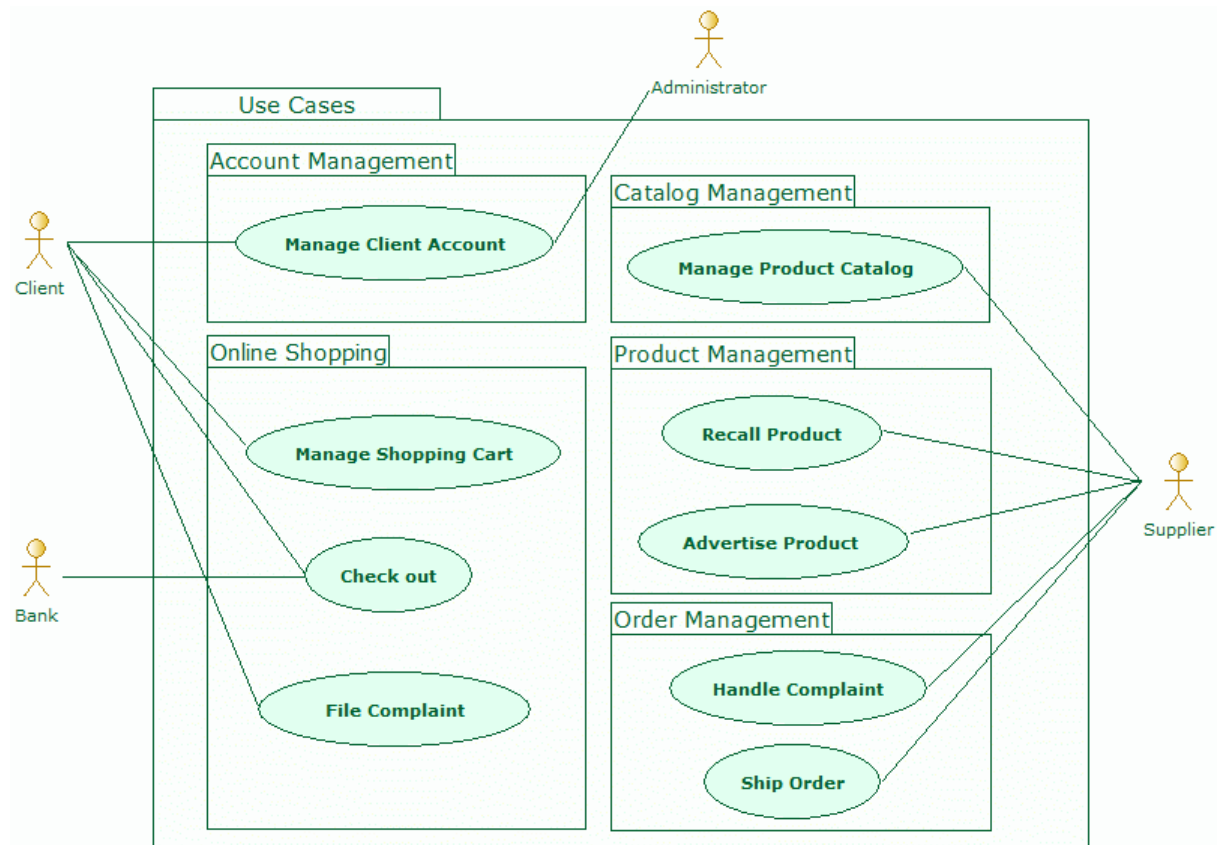


Figure 7 : Use Cases Overview

## 3 Package Index

<a href="#"><u>ShoppingCart</u></a>	This project contains the model for the Online Shopping Cart system, including all the models used to specify and realize the IT system.
<a href="#"><u>Analysis</u></a>	Analysis models for the Online Shopping Cart system
<a href="#"><u>Actors</u></a>	Persons or entities external to the system interacting with the system.
<a href="#"><u>Processes</u></a>	Main processes describing the online shopping system core business.
<a href="#"><u>Use Cases</u></a>	Use case for interacting with the shopping cart system.
<a href="#"><u>Online Shopping</u></a>	Use cases related to the client's shopping activities
<a href="#"><u>Account Management</u></a>	Use cases related to the management of accounts: creation, deletion...
<a href="#"><u>Catalog Management</u></a>	Use cases related to the management of the product catalog.
<a href="#"><u>Product Management</u></a>	Use cases related to the management of products for sale online.
<a href="#"><u>Order Management</u></a>	Supplier order management use cases.
<a href="#"><u>Domains</u></a>	This package contains the data and entity models for the system.
<a href="#"><u>OnlineStore</u></a>	This package contains all the classes directly related to the online store.
<a href="#"><u>Client</u></a>	This package contains all the classes related to the client.
<a href="#"><u>Products</u></a>	This package contains the classes related to the products.
<a href="#"><u>Bank</u></a>	In this package, the classes related to the bank are located.
<a href="#"><u>Base Types</u></a>	Non business-related base types.
<a href="#"><u>Design</u></a>	This package contains the models explaining the way the system is designed.
<a href="#"><u>Logical Architecture</u></a>	The Online Shopping Cart logical architecture.
<a href="#"><u>Presentation</u></a>	User interface components layer.
<a href="#"><u>Business</u></a>	Business layer and its components
<a href="#"><u>Exchange Types</u></a>	Exchange types included in the component service contract.
<a href="#"><u>Implementation</u></a>	Internal part of the component used to implement its functionalities.

<u>Entity</u>	Persistence layer components.
<u>Deployment</u>	Deployment model for the Online Shopping Cart System.
<u>Documents</u>	Contains the different documents describing the model.

## 4 Package "ShoppingCart"

This model addresses the development of an online shopping cart system, that allows suppliers to sell their products directly online to customers. It is provided as a UML example covering the analysis of the business context and processes, and the design and deployment of a possible implementation. The goal is to show the power UML brings to analysing and designing a robust system that corresponds to the initial requirements.

This model can be used to create demos of the Modelio tool. It is documented in depth, in order to produce pertinent documents using Modelio DocumentPublisher. Some examples of documents are also included in this example, in the "Documents" package.

Name	Summary
<u>Analysis</u>	Analysis models for the Online Shopping Cart system
<u>Design</u>	This package contains the models explaining the way the system is designed.
<u>Deployment</u>	Deployment model for the Online Shopping Cart System.
<u>Documents</u>	Contains the different documents describing the model.

*Table 3 Owned Packages of Package "ShoppingCart"*

## 5 Package "Analysis"

from Package *ShoppingCart*

This package contains the models used to analyse the business and its context in order to properly scope what the system must do.

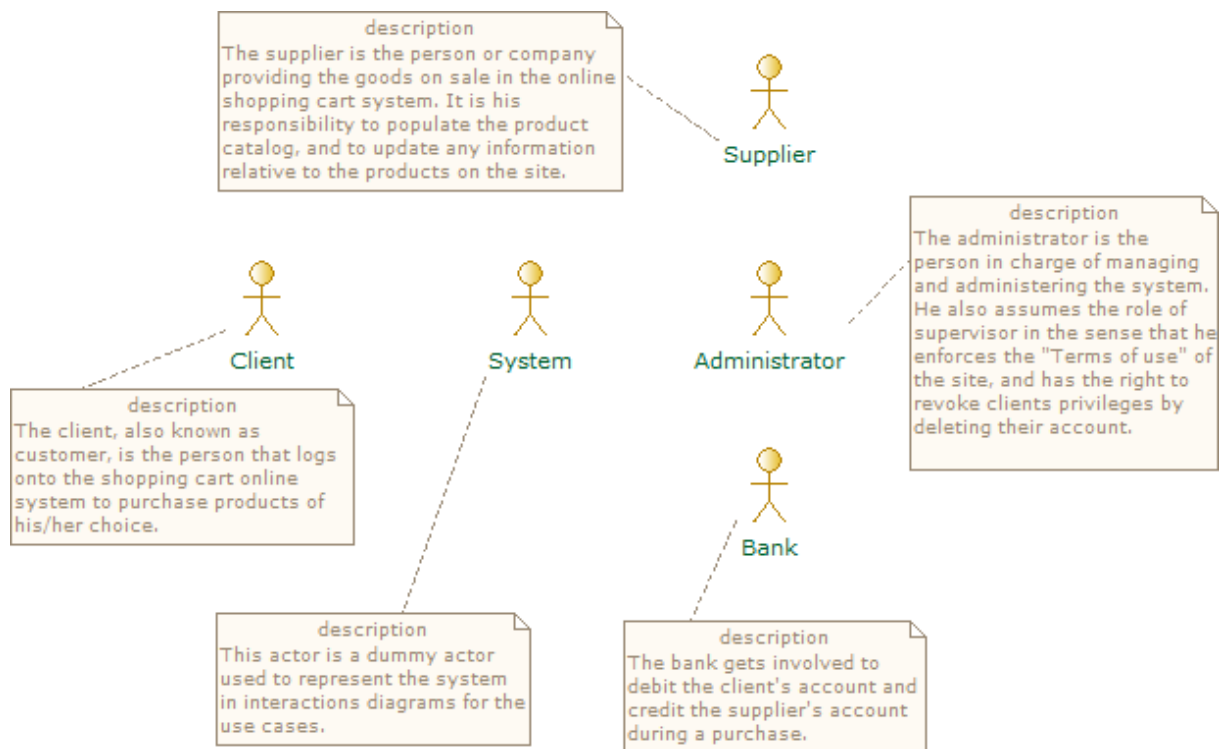
Name	Summary
<u>Actors</u>	Persons or entities external to the system interacting with the system.
<u>Processes</u>	Main processes describing the online shopping system core business.
<u>Use Cases</u>	Use case for interacting with the shopping cart system.
<u>Domains</u>	This package contains the data and entity models for the system.

*Table 4 Owned Packages of Package "Analysis"*

## 6 Package "Actors"

from Package ShoppingCart.[Analysis](#)

This package contains all the persons, systems or protocols that interact with the system during its life.

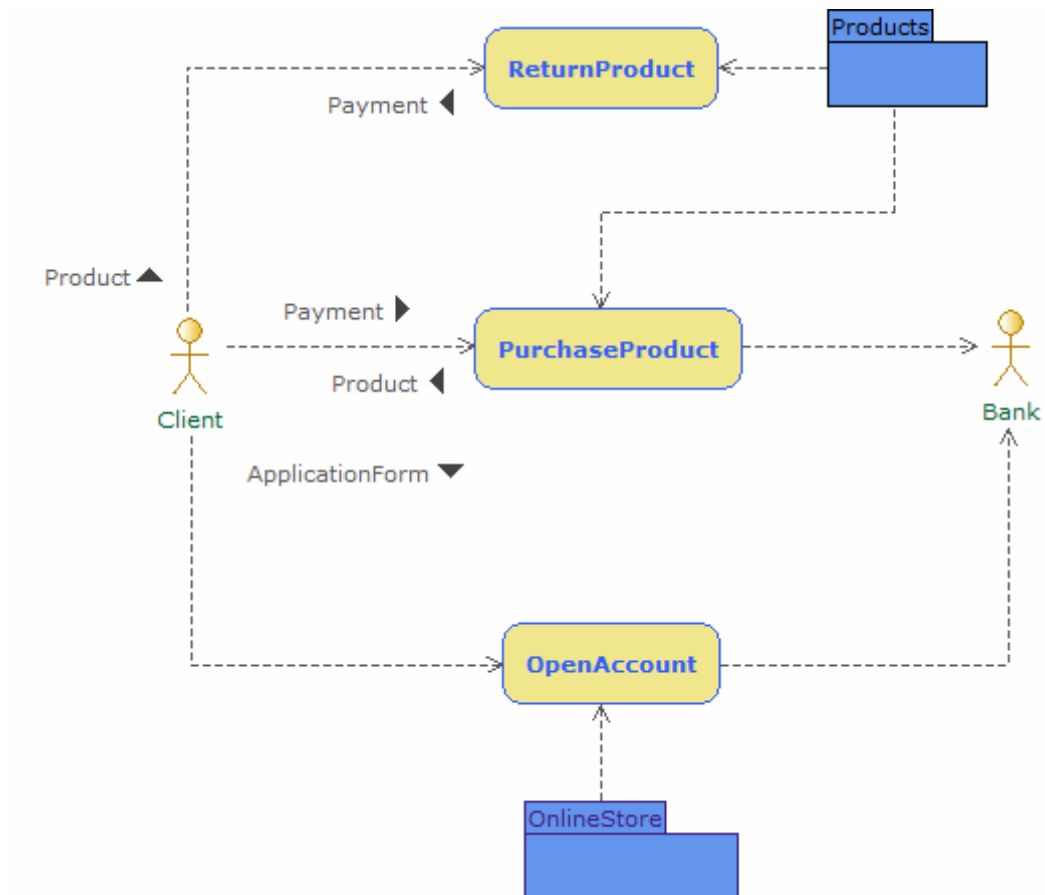


**Figure 8 Actors**

## 7 Package "Processes"

from Package *ShoppingCart*.[Analysis](#)

This package contains the analysis of the three main business processes the online shopping system must implement. They are described using UML activity diagrams.



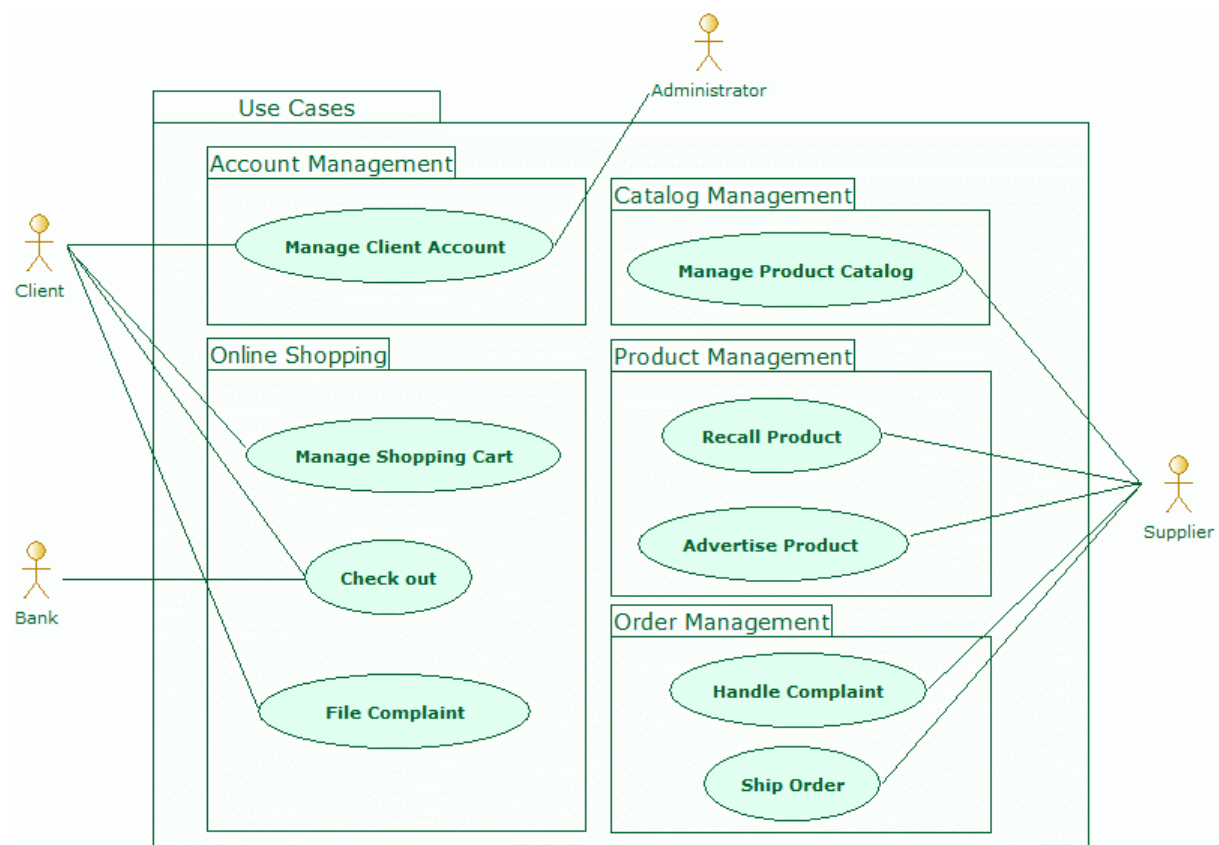
**Figure 9 Processes Overview**



## 8 Package "Use Cases"

from Package *ShoppingCart*.[Analysis](#)

This package contains a description of all the use cases that can be played by external actors to use the system.



**Figure 10 Use Cases Overview**

This diagram includes all the use cases identified for the online shopping cart system. Use cases are organised in containers regrouping related use cases.

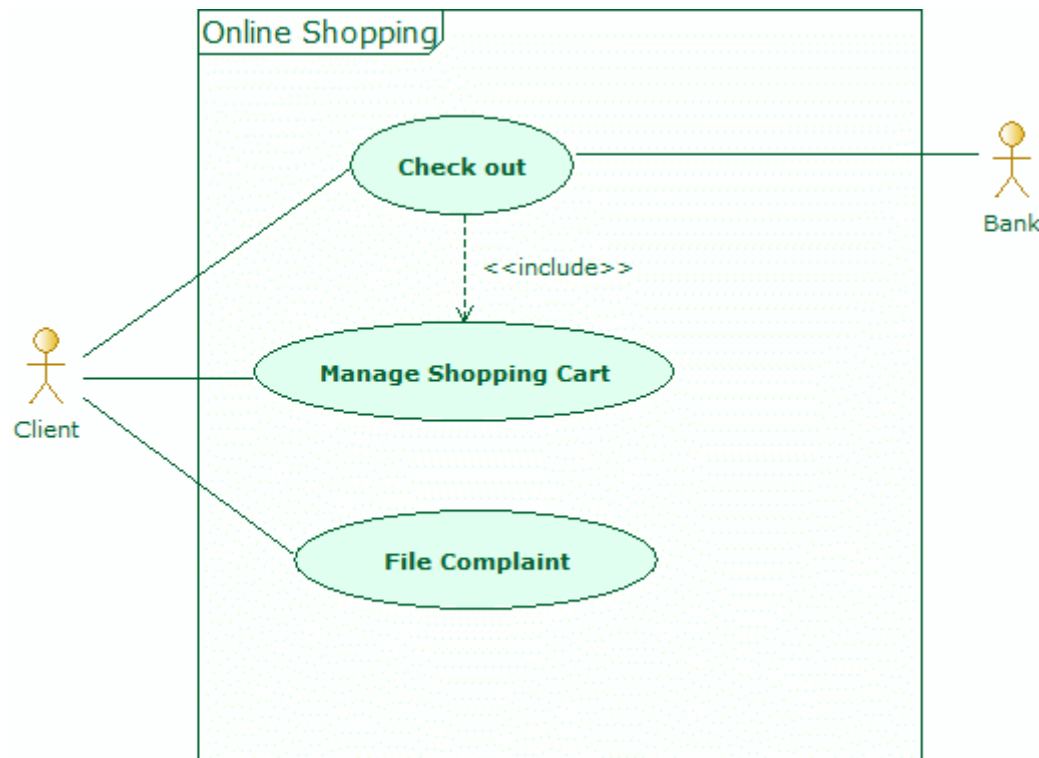
Name	Summary
<b><u>Online Shopping</u></b>	Use cases related to the client's shopping activities
<b><u>Account Management</u></b>	Use cases related to the management of accounts: creation, deletion...
<b><u>Catalog Management</u></b>	Use cases related to the management of the product catalog.
<b><u>Product Management</u></b>	Use cases related to the management of products for sale online.
<b><u>Order Management</u></b>	Supplier order management use cases.

**Table 5 Owned Packages of Package "Use Cases"**

## 9 Package "Online Shopping"

from Package *ShoppingCart.Analysis*. Use Cases

Here we find the use cases related to the shopping activities. These are the main interactions between the system and the client during buying sessions.



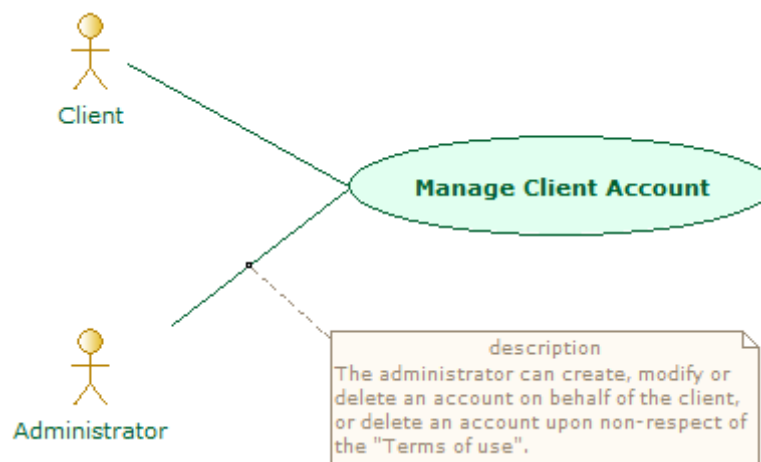
**Figure 11 Online Shopping Overview**

This diagram shows the actors and their interactions with the use cases they can play against the system. This domain contains the use cases related to the client's shopping activities.

## 10Package "Account Management"

from Package *ShoppingCart.Analysis*. Use Cases

This package contains the use cases related to the management of client's accounts. All the interactions between the system and the client that relate to accounts are described here.



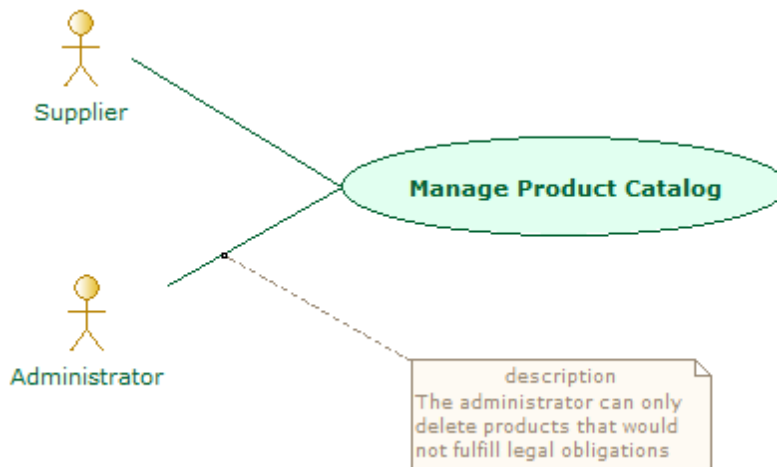
**Figure 12 Account Management Overview**

This diagram shows all the use cases and actors related to the management of the client's accounts.

## 11Package "Catalog Management"

from Package *ShoppingCart.Analysis*. Use Cases

The use cases contained in this package describe the interactions between users and system to manage the product catalog.



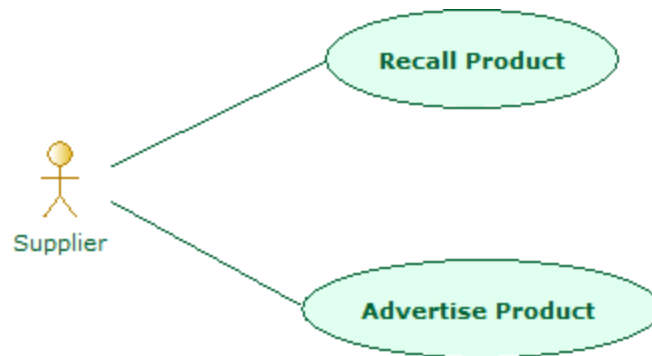
**Figure 13 Catalog Management Overview**

This diagram shows the use cases and the actors related to the management of the product catalog for the online shopping cart.

## 12Package "Product Management"

*from Package ShoppingCart.Analysis.* Use Cases

The use cases contained in this package describe how suppliers interact with the system to update product descriptions or decide on advertizing or product recalls.



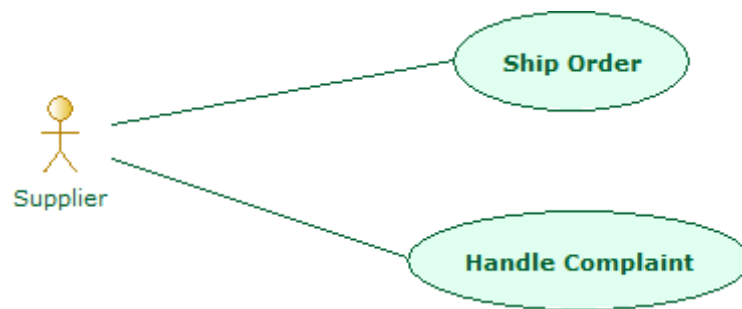
**Figure 14 Product Management Overview**

This diagram shows all use cases related to the management of the products on offer for sale in the online shopping cart system.

## 13Package "Order Management"

*from Package ShoppingCart.Analysis.*Use Cases

This package contains the use cases related to the management of the orders. Complete descriptions of the expected steps to carry out in order to manage orders are provided.

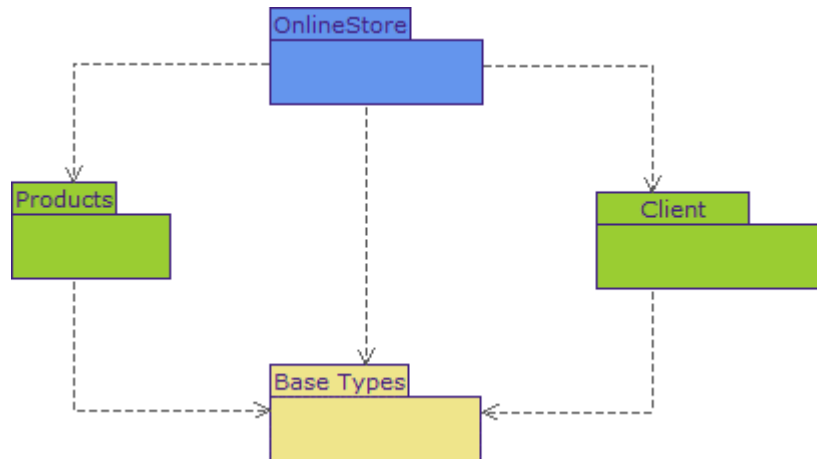


*Figure 15 Order Management Overview*

## 14Package "Domains"

from Package *ShoppingCart*.[\*Analysis\*](#)

This package contains the structural models for all entities required by the Online Shopping Cart system. The static model describes in details the data structure used manipulate entities such as Account or Client.



**Figure 16 Functional Domains**

This diagram shows the different object domains and the dependences that exist between them.

Name	Summary
<b><u>OnlineStore</u></b>	This package contains all the classes directly related to the online store.
<b><u>Client</u></b>	This package contains all the classes related to the client.
<b><u>Products</u></b>	This package contains the classes related to the products.
<b><u>Bank</u></b>	In this package, the classes related to the bank are located.
<b><u>Base Types</u></b>	Non business-related base types.

**Table 6 Owned Packages of Package "Domains"**

## 15Package "OnlineStore"

from Package *ShoppingCart.Analysis*.[Domains](#)

This package contains all the classes related to the online store and the shopping cart. These are the main classes of the online shopping system, supporting the product cart.

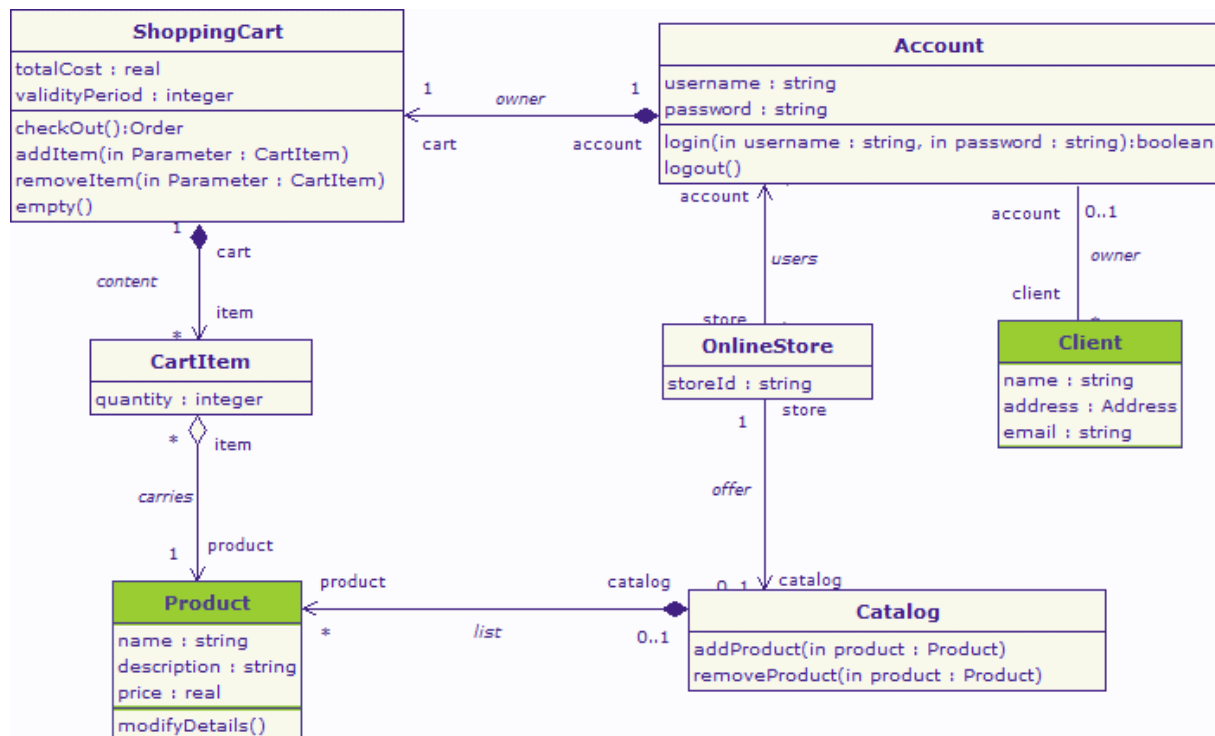


Figure 17 OnlineStore

Name	Summary
<a href="#"><u>Account</u></a>	The account held by the customer.
<a href="#"><u>ShoppingCart</u></a>	The container in which customers can add product for purchase.
<a href="#"><u>CartItem</u></a>	One entry in the shopping cart, containing a reference to a product and a quantity
<a href="#"><u>Catalog</u></a>	The reference listing containing all the products on offer for sale.
<a href="#"><u>OnlineStore</u></a>	The main class, representing the online store.

Table 7 Owned Classes of Package "OnlineStore"

### 15.1 Class "Account"

from Package *ShoppingCart.Analysis*.[Domains](#).[OnlineStore](#)

The class Account represents the account held by a customer. it allows a customer to login to the system to carry out purchases. In addition to a reference to a Client, it contains the credentials for the customer.



### State Machine "Account States"

This state machine describes the different states an account can be in. It is important to note that an account needs to go through a certain number of states before being activated.

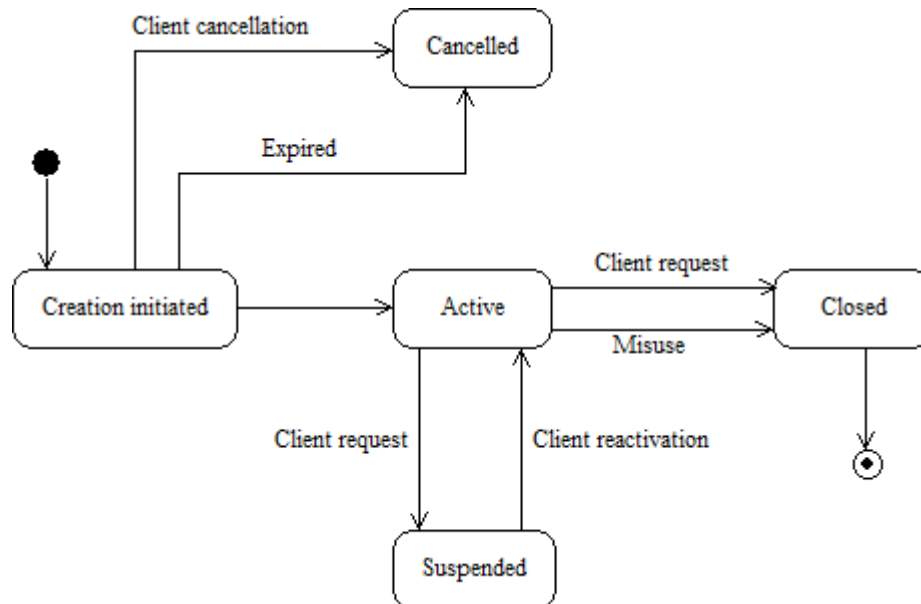


Figure 18 State diagram

Name	Description
<b>boolean login (IN username string,IN password string)</b>	Method invoked by the system in response to the client hitting the login button. this method verifies the credentials provided by the client at login time. A boolean return value indicates success or failure.
<b>logout ()</b>	This method is called in response to the client pressing the logout button. It makes sure the session is terminated, preventing any user from accessing the reserved pages.

Table 8 Operations of Class "Account"

Name	Description
<b>username : [1..1] string</b>	This is the unique identifier for each customer. The username is used during all exchanges with the customer via the online shopping cart site, to provide the users with anonymity.
<b>password : [1..1] string</b>	The password the customer must provide in order to authenticate one self on the shopping cart site. The password is encrypted by the system for security and confidentiality.

Table 9 Attributes of Class "Account"

Name	Description
<b>owner-&gt;client : [0..*] <u>Client</u></b>	Each client has one and only one account. A client cannot exist if he does not have an account. An account cannot exist if it is not attached to a client.

Name	Description
<b>owner-&gt;cart : [1..1] <u>ShoppingCart</u></b>	An account is associated to a single shopping cart. It is composition to clearly state that a shopping cart cannot exist without the associated account.
<b>users-&gt;store : [1..1] <u>OnlineStore</u></b>	This association shows that the OnlineStore class is in relation with 0 or many accounts, the accounts of the users currently active in the system.

Table 10 Associations of Class "Account"

## 15.2 Class "ShoppingCart"

from Package *ShoppingCart.Analysis.Domains*.OnlineStore

This is the container holding the products the customer wants to purchase. It persists its content to permanent storage until the customer proceeds to the check out. This means that items added to the shopping cart can be retrieved between sessions by the customer. Each customer has one cart that is restored every time the customer logs in.

Name	Description
<b>Order checkOut ()</b>	This method is called when the client decides to purchase the content of his/her shopping cart.
<b>addItem (IN Parameter CartItem)</b>	This method is called when the client wants to add an item to his/her shopping cart.
<b>removeItem (IN Parameter CartItem)</b>	This method is used to remove a specific item from the shopping cart. It is necessary to provide the id of the item to be removed.
<b>empty ()</b>	Instead of calling removeItems for all items in the cart one by one, this method enables the system to remove all items at once.

Table 11 Operations of Class "ShoppingCart"

Name	Description
<b>validityPeriod : [1..1] integer</b>	The validity period is expressed in number of days for which the shopping cart can contain items without a check out operation.
<b>totalCost : [1..1] real</b>	This is the total cost for the shopping cart at a given time. Although this field is calculated, it is directly persisted on the ShoppingCart class to avoid calculation delays during queries.

Table 12 Attributes of Class "ShoppingCart"

Name	Description
<b>owner-&gt;account : [1..1] <u>Account</u></b>	An account is associated to a single shopping cart. It is composition to clearly state that a shopping cart cannot exist without the associated account.
<b>content-&gt;item : [0..*] <u>CartItem</u></b>	This association illustrates the relation between cart items and carts. Carts contain items. This relation is a composition to express the idea that items cannot exist without a cart, at anytime.

Table 13 Associations of Class "ShoppingCart"

## 15.3 Class "CartItem"

from Package *ShoppingCart.Analysis.Domains*.[\*OnlineStore\*](#)

A cart item is an entry in the shopping cart describing the product desired and the quantity required. In addition to a reference to a product, the cart item manages the calculation for the total price of the item. This is usually equal to the product unit price multiplied by the quantity.

Name	Description
<b>getTotalCost ()</b>	Since the value returned by this method is calculated on the basis of the product unit cost and the quantity, it is not persisted in a class variable. This method interrogates the Product class to obtain the unit cost.

*Table 14 Operations of Class "CartItem"*

Name	Description
<b>quantity : [1..1] integer</b>	The quantity of products in this item. It can also be that the cart contains several items referencing the same product. But for large quantities, it is useful to be able to specify it in the CartItem class.

*Table 15 Attributes of Class "CartItem"*

Name	Description
<b>content-&gt;cart : [1..1] <a href="#"><u>ShoppingCart</u></a></b>	This association illustrates the relation between cart items and carts. Carts contain items. This relation is a composition to express the idea that items cannot exist without a cart, at anytime.
<b>carries-&gt;product : [1..1] <a href="#"><u>Product</u></a></b>	Each CartItem contains a reference to the product associated with the item. This association expresses this relation, in order to be able to reference the product from the item.

*Table 16 Associations of Class "CartItem"*

## 15.4 Class "Catalog"

from Package *ShoppingCart.Analysis.Domains*.[\*OnlineStore\*](#)

The catalog is the reference containing all the products on sale online. It contains a complete description of each product, so the client can be fully informed about the product before purchasing it. The catalog organizes products in categories to ease browsing for the customer.

Name	Description
<b>addProduct (IN product Product)</b>	This method is used to add new products to the catalog. It is called as a result of a supplier publishing a new product for sale. It takes the new product to add as argument.
<b>removeProduct (IN product Product)</b>	This method enables a supplier or administrator to remove a product from the catalog. The argument is the product to remove.

*Table 17 Operations of Class "Catalog"*

Name	Description
<b>offer-&gt;store : [1..1]</b> <b><u>OnlineStore</u></b>	This association has been added to allow the OnlineStore class to manipulate catalogs as required. An online store can have several catalogs, and for example decide to publish one at a given time, and a different one at a later stage, hence the "many" cardinality.
<b>list-&gt;product : [0..*]</b> <b><u>Product</u></b>	This relation illustrates the fact that catalogs have products in them. This association enables the catalog class to maintain a reference on products listed in the catalog, in order to interrogate the Product class for product descriptions and details.

Table 18 Associations of Class "Catalog"

## 15.5 Class "OnlineStore"

from Package *ShoppingCart.Analysis.Domains*.**OnlineStore**

This class is an abstraction for the online store. It allows for all the other classes to be connected together in a structured way, and provides the glue that permit other entities to be combined.

Name	Description
<b>storeId : [1..1] string</b>	This variable uniquely identifies the OnlineStore instance. This might become useful for a system that would support distinct stores on the same site for example.

Table 19 Attributes of Class "OnlineStore"

Name	Description
<b>offer-&gt;catalog : [0..1]</b> <b><u>Catalog</u></b>	This association has been added to allow the OnlineStore class to manipulate catalogs as required. An online store can have several catalogs, and for example decide to publish one at a given time, and a different one at a later stage, hence the "many" cardinality.
<b>users-&gt;account : [0..*]</b> <b><u>Account</u></b>	This association shows that the OnlineStore class is in relation with 0 or many accounts, the accounts of the users currently active in the system.

Table 20 Associations of Class "OnlineStore"

## 16Package "Client"

from Package *ShoppingCart.Analysis*.*Domains*

This package contains all the classes related to the client.

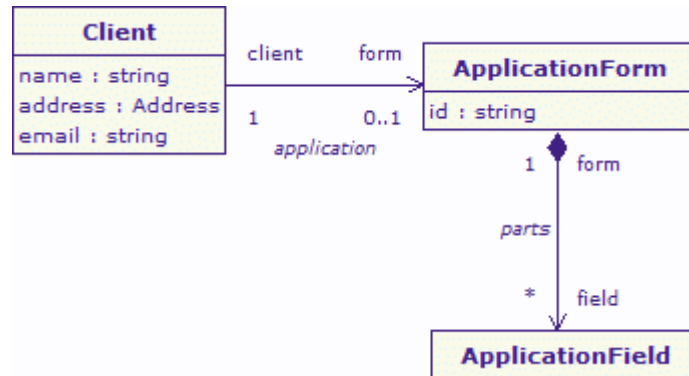


Figure 19 Client

Name	Summary
<u>Client</u>	The Client class represents the client or customer.
<u>ApplicationForm</u>	Form to fill to open a new account
<u>ApplicationField</u>	Entry in an application form.

Table 21 Owned Classes of Package "Client"

### 16.1 Class "Client"

from Package *ShoppingCart.Analysis*.*Domains*.*Client*

The Client class represents the client or customer. Its main role is to carry all the data attached to the client, like its name and address.

Name	Description
<b>name</b> : [1..1] string	This attribute holds the full name of the client, in accordance with the Home Office registry.
<b>address</b> : [1..1] Address	This is the complete address of the client. This address must be the same as the billing address provided in the customer's credit details.
<b>email</b> : [1..1] string	This is the email address used to communicate with the customer in all circumstances.

Table 22 Attributes of Class "Client"

Name	Description
<b>owner-&gt;account</b> : [0..1] <u>Account</u>	This association represents the relation between the Account class and the Client class. Each client has one and only one account.

Name	Description
<b>application-&gt;form : [0..1]</b> <u><b>ApplicationForm</b></u>	Each client has zero or one application form. This is the initial request for an account creation.

Table 23 Associations of Class "Client"

## 16.2 Class "ApplicationForm"

from Package *ShoppingCart.Analysis.Domains*.*Client*

The ApplicationForm class regroups all the fields that need to be populated by the customer when applying for an account with the store.

Name	Description
<b>id : [1..1] string</b>	The unique identifier for the application form.

Table 24 Attributes of Class "ApplicationForm"

Name	Description
<b>application-&gt;client : [1..1] <u>Client</u></b>	Each client has zero or one application form. This is the initial request for an account creation.
<b>parts-&gt;field : [0..*]</b> <u><b>ApplicationField</b></u>	An application form is made up of a certain number of application fields. Depending on the nature of the application, the number of fields may vary. The composition indicates the strong influence on the application field's lifecycle.

Table 25 Associations of Class "ApplicationForm"

## 16.3 Class "ApplicationField"

from Package *ShoppingCart.Analysis.Domains*.*Client*

An application field is a simple question/answer pair of text corresponding to all the details required by the system to register a client. This class allows for free form applications to be created.

Name	Description
<b>question : [1..1] string</b>	The question asked to the client while filling the form. For simple details like the name or the address, this variable will have a simple text value like "Name:" or "Address:".
<b>answer : [1..1] string</b>	This variable is used to store the answer provided by the client to the corresponding question, for later processing by the component or actor concerned.

Table 26 Attributes of Class "ApplicationField"

Name	Description
<b>parts-&gt;form : [1..1]</b> <u><b>ApplicationForm</b></u>	An application form is made up of a certain number of application fields. Depending on the nature of the application, the number of fields may vary. The composition indicates the strong influence on the application field's lifecycle.

Table 27 Associations of Class "ApplicationField"

## 17Package "Products"

from Package *ShoppingCart.Analysis.Domains*.[Domains](#)

The classes included in this package relate to the products on sale in the online shopping cart. All concepts related to the suppliers are modelled in this package.

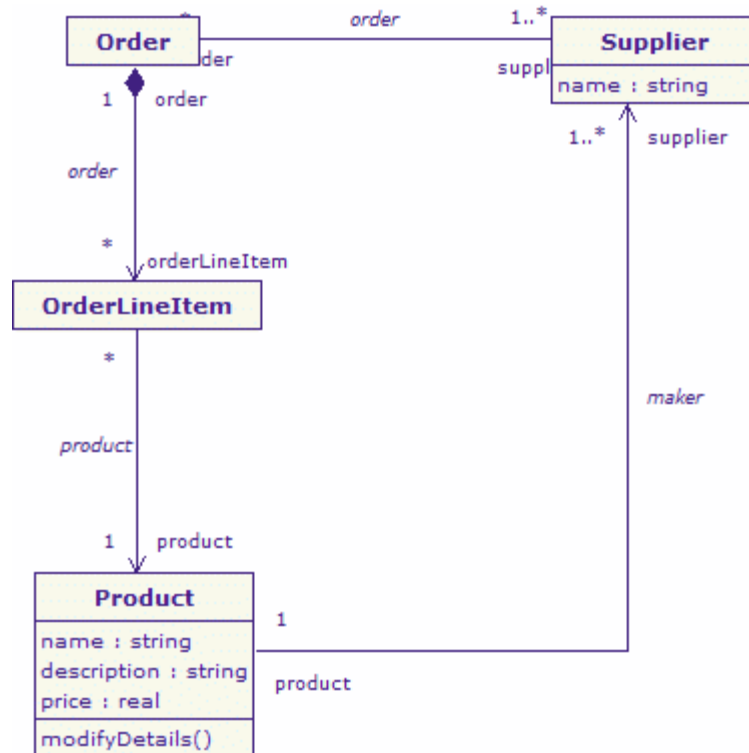


Figure 20 Products

Name	Summary
<b><u>Product</u></b>	Class representing the different products suppliers offer for sale.
<b><u>Supplier</u></b>	Person or organisation providing goods for sale to the system.
<b><u>Order</u></b>	Document received by the supplier as a result of a client purchasing products online.
<b><u>OrderLineItem</u></b>	Represents one line in an order document.

Table 28 Owned Classes of Package "Products"

### 17.1 Class "Product"

from Package *ShoppingCart.Analysis.Domains*.[Products](#)

Class representing the different products suppliers offer for sale. The complete product's details are included to allow for a comprehensive description to be provided to the buyer.

Name	Description
<b>modifyDetails ()</b>	Through this method, client can change the details for the product. This method does not provide a mean to modify the product's name, which is invariable. In the case the product changes name, a new Product instance must be created.

Table 29 Operations of Class "Product"

Name	Description
<b>name : [1..1] string</b>	This variable holds the name of the product. It is impossible to change the name of the product. The only access to the variables of this class is via the method modifyDetails(), which does not write this variable.
<b>description : [1..1] string</b>	Complete text provided by the supplier, and describing the product in details.
<b>price : [1..1] real</b>	The current price for the product. The price can be modified, providing the impact on customers is controlled. In no cases, the price can be changed once the client has pressed the "Check Out" button.

Table 30 Attributes of Class "Product"

Name	Description
<b>carries-&gt;item : [0..*] <u>CartItem</u></b>	Each CartItem contains a reference to the product associated with the item. This association expresses this relation, in order to be able to reference the product from the item.
<b>list-&gt;catalog : [0..1] <u>Catalog</u></b>	This relation illustrates the fact that catalogs have products in them. This association enables the catalog class to maintain a reference on products listed in the catalog, in order to interrogate the Product class for product descriptions and details.
<b>product-&gt;undefined : [0..*] <u>OrderLineItem</u></b>	In the same manner a shopping cart item has a reference to a product, an OrderLineItem has also a reference to a product.
<b>maker-&gt;supplier : [1..*] <u>Supplier</u></b>	Association modeling the relationship between products and supplier. In this model, a supplier does not have products, it's the product that has one and only one supplier. The same product may not have several supplier.

Table 31 Associations of Class "Product"

## 17.2 Class "Supplier"

from Package *ShoppingCart.Analysis.Domains*.Products

Person or organisation providing goods for sale to the system. The system itself is only responsible for the selling, the supplier taking also responsibility for shipping and receiving products.

Name	Description
<b>name : [1..1] string</b>	The name of the supplier. this must be equal to the social name of the organisation or the Home Office record in the case of an individual.

Table 32 Attributes of Class "Supplier"



Name	Description
<b>order-&gt;order : [0..*]</b> <u>Order</u>	The supplier holds a certain number of orders at any one time, representing the orders still to be fulfilled.
<b>maker-&gt;product : [1..1]</b> <u>Product</u>	Association modeling the relationship between products and supplier. In this model, a supplier does not have products, it's the product that has one and only one supplier. The same product may not have several supplier.

Table 33 Associations of Class "Supplier"

### 17.3 Class "Order"

from Package *ShoppingCart.Analysis.Domains*.Products

Document received by the supplier as a result of a client purchasing products online. The supplier uses orders to track the purchase requests originating from the online shopping system.

Name	Description
<b>order-&gt;orderLineItem : [0..*]</b> <u>OrderLineItem</u>	Famous pattern of object-oriented modeling, this association between Order and OrderLineItem is a composition to show that items can only exist if belonging to an order.
<b>order-&gt;supplier : [1..*]</b> <u>Supplier</u>	This association connects the Supplier to its orders. Orders are received by the supplier originated from the system, as a result of a client purchasing a product online.

Table 34 Associations of Class "Order"

### 17.4 Class "OrderLineItem"

from Package *ShoppingCart.Analysis.Domains*.Products

Represents one line in an order document. In the reality of the Online Shopping system, the OrderLineItem instances are in a one-to-one relation to the shopping cart items.

Name	Description
<b>id : [1..1]</b> undefined	Unique identifier for the OrderLineItem.

Table 35 Attributes of Class "OrderLineItem"

Name	Description
<b>order-&gt;order : [1..1]</b> <u>Order</u>	Famous pattern of object-oriented modeling, this association between Order and OrderLineItem is a composition to show that items can only exist if belonging to an order.
<b>product-&gt;product : [1..1]</b> <u>Product</u>	In the same manner a shopping cart item has a reference to a product, an OrderLineItem has also a reference to a product.

Table 36 Associations of Class "OrderLineItem"

## 18Package "Bank"

from Package *ShoppingCart.Analysis*.[Domains](#)

This package contains all the classes in relation to the bank and payments. Since the bank system is separate from the online shopping cart, the classes contained in this package represent the interface to the bank payment system, and in particular the exchange data types used to converse with the bank.

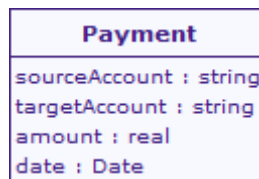


Figure 21 Bank

Name	Summary
<b>Payment</b>	This class represents a payment.

Table 37 Owned Classes of Package "Bank"

### 18.1 Class "Payment"

from Package *ShoppingCart.Analysis*.[Domains](#).[Bank](#)

This class represents a payment. it carries all the details necessary to the actual financial transaction and serves as reference for historisation.

Name	Description
<b>sourceAccount : [1..1] string</b>	The account of the buyer to be debited.
<b>targetAccount : [1..1] string</b>	The account of the seller to be credited.
<b>amount : [1..1] real</b>	The amount to be transferred between the accounts of the seller and the buyer.
<b>date : [1..1] Date</b>	The date at which the payment occurs.

Table 38 Attributes of Class "Payment"

## 19 Package "Base Types"

from Package *ShoppingCart.Analysis.Domains*. [Domains](#)

This package contains the definition of non-primitive base data types useful to give a type to domain class attributes.

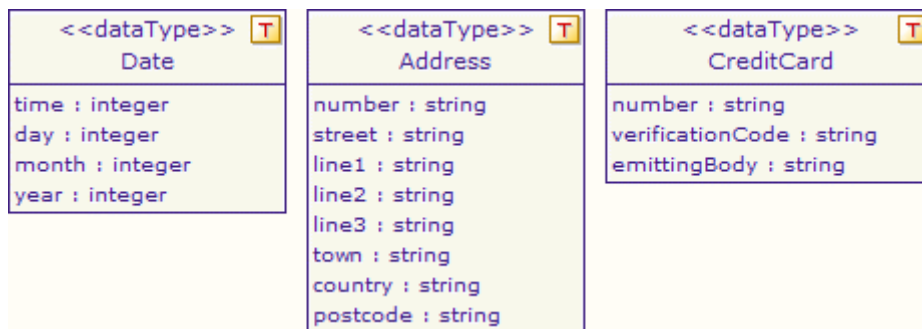


Figure 22 Base Types (1)

Name	Summary
<a href="#"><u>Date</u></a>	Datatype for a date.
<a href="#"><u>Address</u></a>	Datatype for an address
<a href="#"><u>CreditCard</u></a>	Datatype for a credit card.

Table 39 Owned DataTypes of Package "Base Types"

### 19.1 DataType "Date"

from Package *ShoppingCart.Analysis.Domains*. [Base Types](#)

Datatype for a date. This datatype includes all the necessary fields to fully represent arbitrary dates.

Name	Description
<b>time : [1..1] integer</b>	Number of seconds since midnight the previous day.
<b>day : [1..1] integer</b>	Day of the year. This value is bound by 0 and 366.
<b>month : [1..1] integer</b>	The number of the month, January being represented by 1 and December by 12.
<b>year : [1..1] integer</b>	The year in full, with a limit of 4 digits (no support for dates after December 31st 9999).

Table 40 Attributes of DataType "Date"

### 19.2 DataType "Address"

from Package *ShoppingCart.Analysis.Domains*. [Base Types](#)

Datatype for an address. The fields of this datatype have been designed to support addresses in arbitrary countries.

Name	Description
<b>number</b> : [1..1] string	the street number. This field accepts alphanumerical characters (10a, 12c...)
<b>street</b> : [1..1] string	The name and type of street.
<b>line1</b> : [1..1] string	Additonal line of text to fully describe the address.
<b>line2</b> : [1..1] string	Additonal line of text to fully describe the address.
<b>line3</b> : [1..1] string	Additonal line of text to fully describe the address.
<b>town</b> : [1..1] string	The name of the town
<b>country</b> : [1..1] string	The name of the country
<b>postcode</b> : [1..1] string	The postcode or zipcode depending on the country.

Table 41 Attributes of DataType "Address"

### 19.3 DataType "CreditCard"

from Package *ShoppingCart.Analysis.Domains*.[\*Base Types\*](#)

Datatype for a credit card. This type contains the 16-digit card number uniquely identifying the credit card.

Name	Description
<b>number</b> : [1..1] string	The 16-digit number uniquely identifying the credit card. In order to support alphanumerical characters in the future, the type of this variable is "string" rather than an integer.
<b>verificationCode</b> : [1..1] string	The three-digit code indicated on the back of the card, and used to verify authenticity. In order to support alphanumerical characters in the future, this variable is typed by "string".
<b>emittingBody</b> : [1..1] string	The organisation that emitted the card, usually a financial body.

Table 42 Attributes of DataType "CreditCard"

## 20Package "Design"

from Package *ShoppingCart*

This section describes the logical architecture chosen to implement the online shopping cart system. It is based on a component approach in accordance with the SCA architecture. Components are organized in several layers according to their level of stability. The layer "Entity" contains the most stable components, while the layer "Presentation" contains the least stable ones. Components use other components to provide the required functionalities. A component can only make use of the interfaces of components belonging to a layer of greater stability, and this to insure the highest level of reusability possible.

Name	Summary
<u>Logical Architecture</u>	The Online Shopping Cart logical architecture.

*Table 43 Owned Packages of Package "Design"*

## 21Package "Logical Architecture"

from Package *ShoppingCart.Design*

The logical architecture contained in this package correspond to the technological choices made by the architects to implement the shopping cart system.

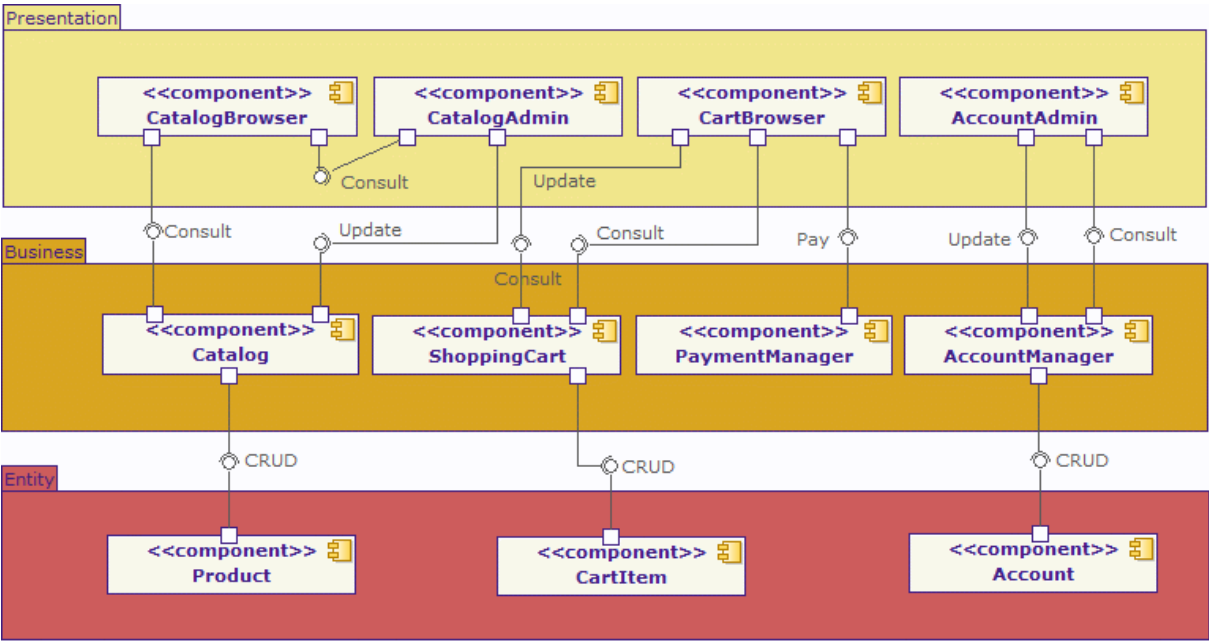


Figure 23 Logical Architecture

Name	Summary
<u>Presentation</u>	User interface components layer.
<u>Business</u>	Business layer and its components
<u>Entity</u>	Persistence layer components.

Table 44 Owned Packages of Package "Logical Architecture"

## 22Package "Presentation"

from Package *ShoppingCart.Design.Logical Architecture*

Regroups the components in charge of providing a user interface to the online shopping cart system.

Name	Summary
<u><a href="#">CatalogBrowser</a></u>	Interface component allowing users to browse the catalog.
<u><a href="#">CartBrowser</a></u>	Interface component providing the functionalities to browse the user cart.
<u><a href="#">CatalogAdmin</a></u>	This component provides the administration interface for the catalog.
<u><a href="#">AccountAdmin</a></u>	This component provides the Online Shopping Cart system administration interface.

*Table 45 Owned Classes of Package "Presentation"*

### 22.1 Class "CatalogBrowser"

from Package *ShoppingCart.Design.Logical Architecture.Presentation*

This user-interface component is in charge of providing all the widgets and functionalities to browse the product catalog. In order to provide a good ergonomomy, the component implements pagination and ordering functionalities on the list of products.

Name	Requires	Provides
Please provide content here		
Please provide content here		

*Table 46 Ports of Component "CatalogBrowser"*

### 22.2 Class "CartBrowser"

from Package *ShoppingCart.Design.Logical Architecture.Presentation*

This GUI component allows users to browse their shopping cart. In addition to all the functions required to visualize the items, this component provides the ability to add and remove items, and presents the check out button.

Name	Requires	Provides
Please provide content here		
Please provide content here		
Please provide content here		Interface <u><a href="#">Consult</a></u>

*Table 47 Ports of Component "CartBrowser"*

## 22.3 Class "CatalogAdmin"

*from Package ShoppingCart.Design.Logical Architecture.* Presentation

This component provides the graphical interface for administering the catalog. It reuses the CatalogBrowser component, and adds the functionalities required to add and remove products from the catalog. Its access is reserved for the suppliers and makers of products.

Name	Requires	Provides
Please provide content here		
Please provide content here		Interface <u>Consult</u>

*Table 48 Ports of Component "CatalogAdmin"*

## 22.4 Class "AccountAdmin"

*from Package ShoppingCart.Design.Logical Architecture.* Presentation

This component comprises all the functions to administer the user accounts. The system administrator controls access to the site by granting accounts to users. It is also possible to revoke accounts for users not respecting the conditions of use of the site.

Name	Requires	Provides
Please provide content here		
Please provide content here		Interface <u>Consult</u>

*Table 49 Ports of Component "AccountAdmin"*



## 23 Package "Business"

from Package *ShoppingCart.Design.Logical Architecture*

In this package, the components in charge of implementing the business behaviour of the system are described.

Name	Summary
<u>Catalog</u>	This component manages the system catalog.
<u>ShoppingCart</u>	This component provides the shopping cart management functions.
<u>PaymentManager</u>	Component implementing the payment related operations.
<u>AccountManager</u>	User accounts management component.

Table 50 Owned Classes of Package "Business"

### 23.1 Class "Catalog"

from Package *ShoppingCart.Design.Logical Architecture.Business*

This component provides all the functions related to the catalog. It bears all of the query operations that can be carried out on the catalog, including set operations returning lists of catalog items based on filtered queries.

Name	Summary
<u>Consult</u>	
<u>Update</u>	

Table 51 Owned Interfaces of Class "Catalog"

Name	Requires	Provides
Please provide content here	Interface <u>Consult</u>	
Please provide content here	Interface <u>Update</u>	
Please provide content here		Interface <u>CRUD</u>

Table 52 Ports of Component "Catalog"

### 23.2 Class "ShoppingCart"

from Package *ShoppingCart.Design.Logical Architecture.Business*

This component implements the Shopping Cart. It contains all the elements required to completely describe the component's internal and external parts. Its role is to manage the shopping cart and to provide business-level validation for the cart content.

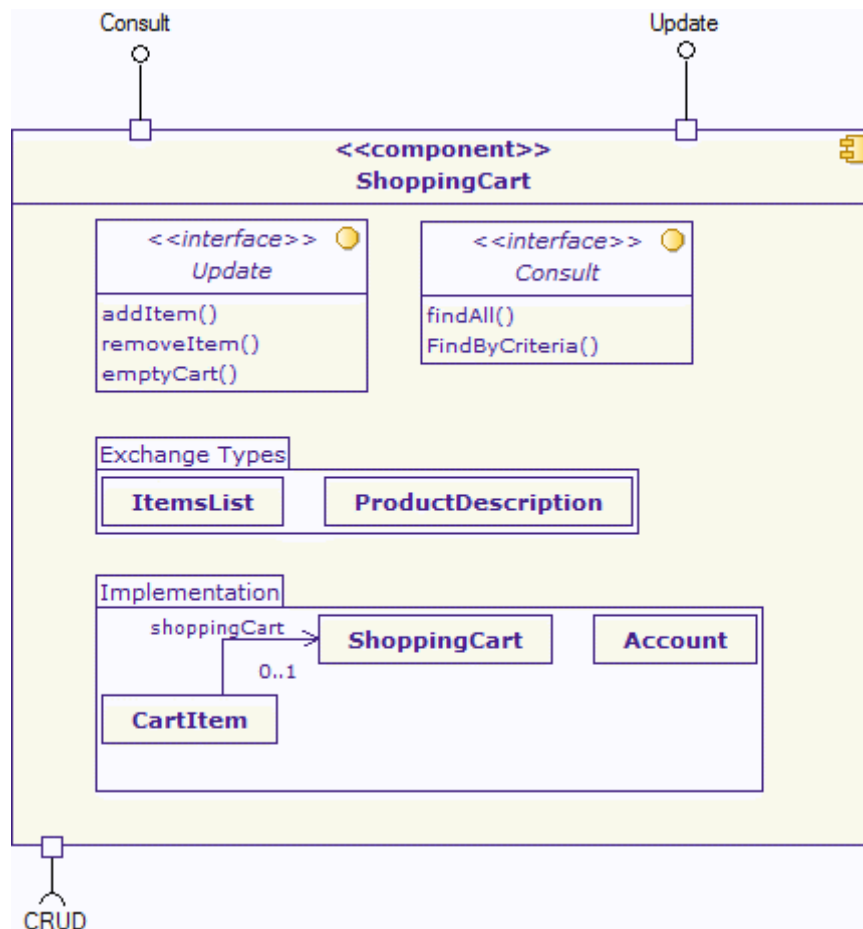


Figure 24 ShoppingCart Component View

Name	Summary
<u>Update</u>	
<u>Consult</u>	

Table 53 Owned Interfaces of Class "ShoppingCart"

Name	Requires	Provides
Please provide content here	Interface <u>Update</u>	
Please provide content here	Interface <u>Consult</u>	
Please provide content here		Interface <u>CRUD</u>

Table 54 Ports of Component "ShoppingCart"

### 23.3 Class "PaymentManager"

from Package *ShoppingCart.Design.Logical Architecture*.Business

This component provides all the operations related to payments. It represents the integration point with the external system pertaining to the bank that handles actual payments.

Name	Summary
<u>Pay</u>	

Table 55 Owned Interfaces of Class "PaymentManager"

Name	Requires	Provides
Please provide content here	Interface <u>Pay</u>	

Table 56 Ports of Component "PaymentManager"

## 23.4 Class "AccountManager"

from Package *ShoppingCart.Design.Logical Architecture*.Business

This component provides the interfaces that allow for the management of user accounts. It includes the authorisation checks necessary to make sure only the allowed customers can log into the system to carry out purchases.

Name	Summary
<u>Update</u>	
<u>Consult</u>	

Table 57 Owned Interfaces of Class "AccountManager"

Name	Requires	Provides
Please provide content here	Interface <u>Update</u>	
Please provide content here		
Please provide content here		Interface <u>CRUD</u>

Table 58 Ports of Component "AccountManager"

## 24Package "Exchange Types"

*from Component ShoppingCart.Design.Logical Architecture.Business.ShoppingCart*

This package contains the description of the exchange types included in the service contract for the ShoppingCart component. The exchange types are deduced from the domain analysis, and are the only data types accepted and produced by the component.

Name	Summary
<u>ItemsList</u>	
<u>ProductDescription</u>	

*Table 59 Owned Classes of Package "Exchange Types"*

### 24.1 Class "ItemsList"

*from Package ShoppingCart.Design.Logical  
Architecture.Business.ShoppingCart.Exchange Types*

### 24.2 Class "ProductDescription"

*from Package ShoppingCart.Design.Logical  
Architecture.Business.ShoppingCart.Exchange Types*

## 25Package "Implementation"

*from Component ShoppingCart.Design.Logical Architecture.Business.*[ShoppingCart](#)

This package contains the elements used to implement the functionalities provided by the component. It is the internal part of the component, that can change without affecting clients of the component services.

Name	Summary
<a href="#"><u>CartItem</u></a>	
<a href="#"><u>ShoppingCart</u></a>	
<a href="#"><u>Account</u></a>	

Table 60 Owned Classes of Package "Implementation"

### 25.1 Class "CartItem"

*from Package ShoppingCart.Design.Logical  
Architecture.Business.ShoppingCart.*[Implementation](#)

Name	Description
undefined->shoppingCart : [0..1] <a href="#"><u>ShoppingCart</u></a>	

Table 61 Associations of Class "CartItem"

### 25.2 Class "ShoppingCart"

*from Package ShoppingCart.Design.Logical  
Architecture.Business.ShoppingCart.*[Implementation](#)

Name	Description
undefined->undefined : [undefined..undefined] <a href="#"><u>CartItem</u></a>	

Table 62 Associations of Class "ShoppingCart"

### 25.3 Class "Account"

*from Package ShoppingCart.Design.Logical  
Architecture.Business.ShoppingCart.*[Implementation](#)

## 26 Package "Entity"

from Package *ShoppingCart.Design.Logical Architecture*

This package represents the most stable layer of the logicla architecture, and contains the components in charge of persisting the business entities required by the system.

Name	Summary
<u>Product</u>	Product persistence component.
<u>CartItem</u>	CartItem persistence component.
<u>Account</u>	Account persistence component.

*Table 63 Owned Classes of Package "Entity"*

### 26.1 Class "Product"

from Package *ShoppingCart.Design.Logical Architecture.Entity*

This is a low-level component that is responsible for implementing the persistence for Product instances. It contains the mapping between the object instances and the physical persistence relational database.

Name	Summary
<u>CRUD</u>	

*Table 64 Owned Interfaces of Class "Product"*

Name	Requires	Provides
Please provide content here		

*Table 65 Ports of Component "Product"*

### 26.2 Class "CartItem"

from Package *ShoppingCart.Design.Logical Architecture.Entity*

This is a low-level component that is responsible for implementing the persistence for CartItem instances. It contains the mapping between the object instances and the physical persistence relational database.

Name	Summary
<u>CRUD</u>	

*Table 66 Owned Interfaces of Class "CartItem"*

Name	Requires	Provides
Please provide content here		

Table 67 Ports of Component "CartItem"

## 26.3 Class "Account"

from Package *ShoppingCart.Design.Logical Architecture*. Entity

This is a low-level component that is responsible for implementing the persistence for Account instances. It contains the mapping between the object instances and the physical persistence relational database.

Name	Summary
<u>CRUD</u>	

Table 68 Owned Interfaces of Class "Account"

Name	Requires	Provides
Please provide content here		

Table 69 Ports of Component "Account"

## 27Package "Deployment"

from Package *ShoppingCart*

This package contains the deployment model for the online shopping cart system. It currently contains a simple deployment scenario that will be used to measure performances and charges.

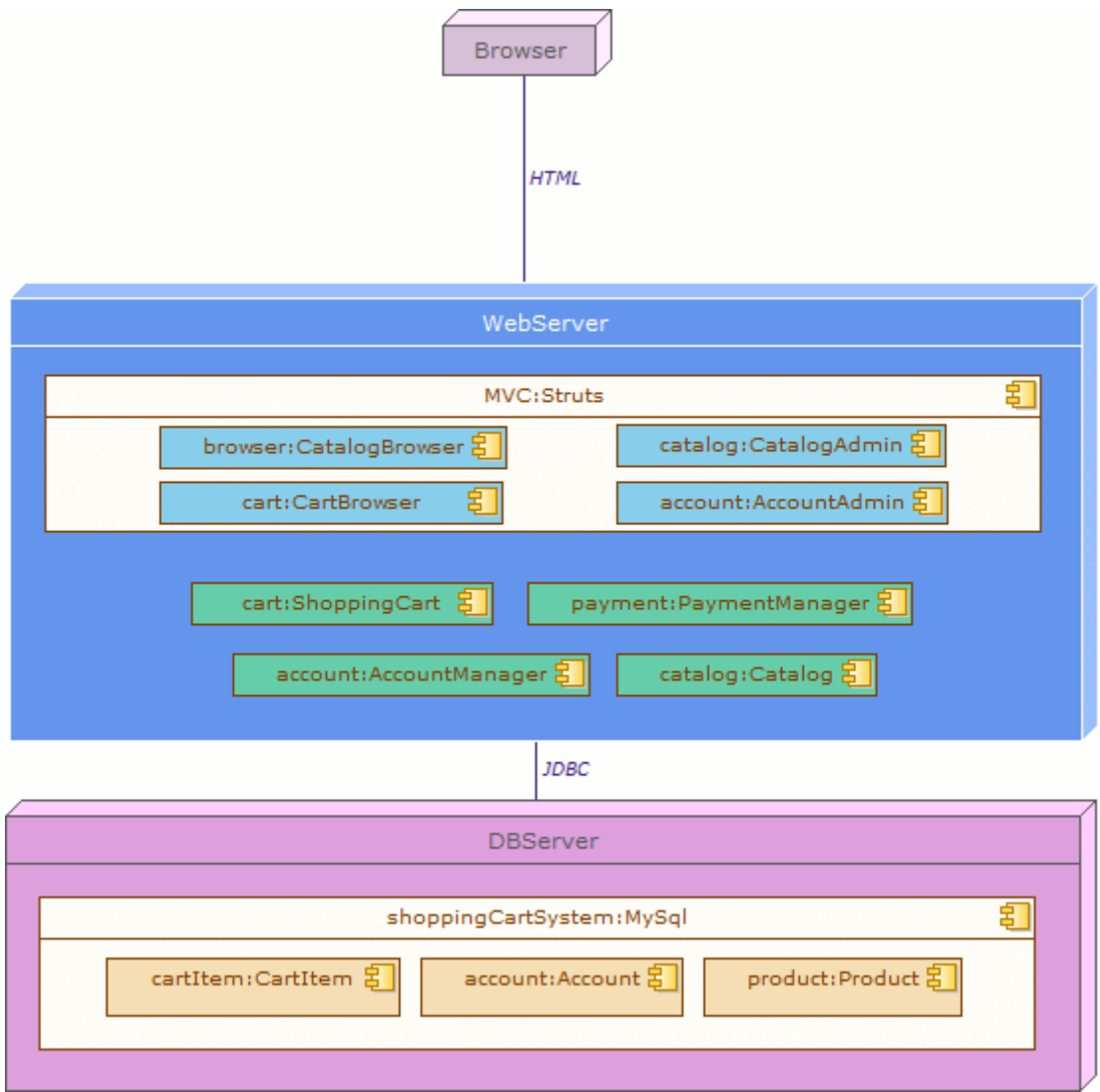


Figure 25 Deployment

Name	Summary
<u>MySQL</u>	Chosen database server.
<u>Struts</u>	Graphical framework component.

Table 70 Owned Classes of Package "Deployment"



## 27.1 Class "MySql"

*from Package ShoppingCart.*Deployment

This component represents the database chosen for persistence. In this case, we choose MySQL as the relational database, for its ease-of-use combined with extensive functionalities.

## 27.2 Class "Struts"

*from Package ShoppingCart.*Deployment

This component represents the chosen graphical framework to implement the actual web pages displayed to the user by the system.

## 28Package "Documents"

*from Package ShoppingCart*

Stereotypes: DocumentPublisher.DocumentFolder

This package regroups all the documents automatically generated by the DocumentPublisher module, and addressing different audiences.