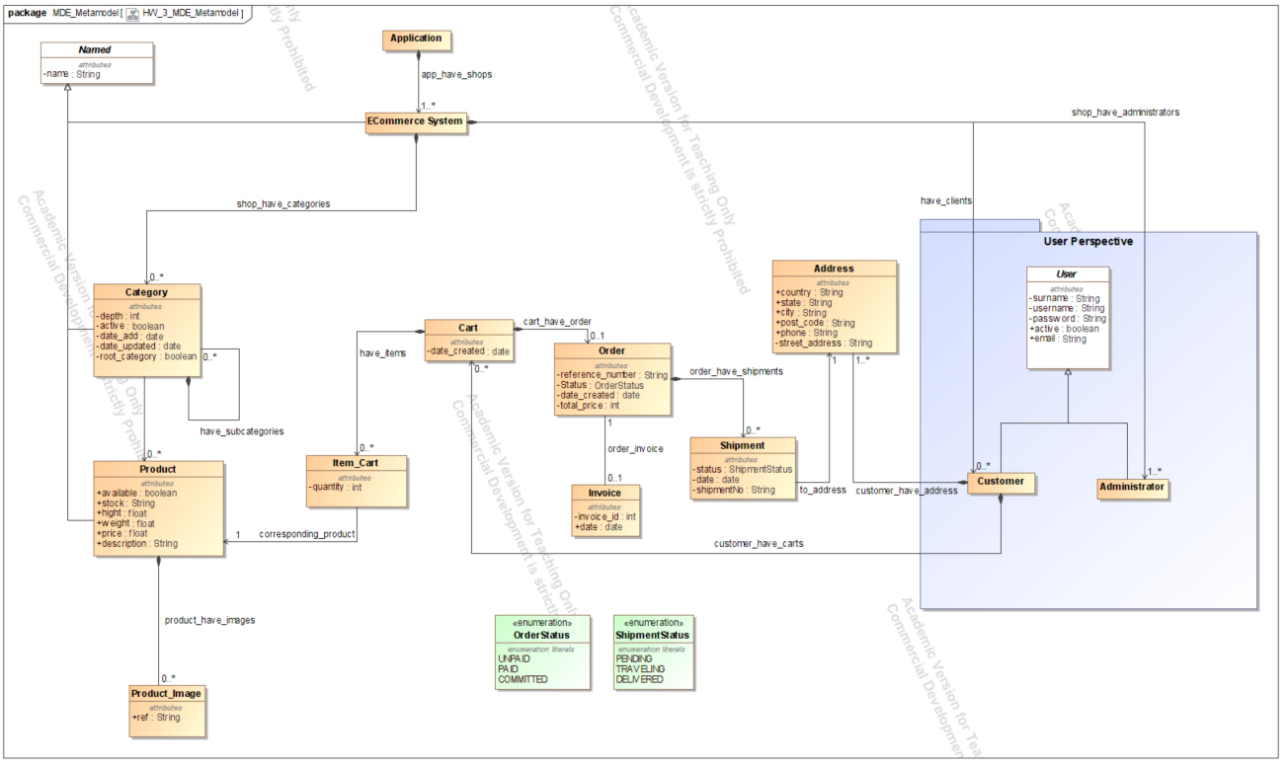
Model Driven Engineering

Homeworks Report

Syed Ghulam Mustufa
Federico Di Menna

Domain

The domain which we have selected is the **Ecommerce data domain**. In the following we added a diagram that represents our first version of the domain metaclasses with their relations. We basically have three main categories of elements: User related metaclasses, Shop related metaclasses and Order related metaclasses.



Metamodel definition

All the project implementation can be found on [this repo](#).

Homework 1

We started from the metamodel illustrated before and we defined the structure and the editors in MPS. The code in in the repo under the path */homewok1/**.

The constraints which we defined in this homework are the following (we grouped them by metaclasses):

- *User*
  - **username** has to match the following regex: `^[a-zA-Z0-9]([._-](?![._-])|[a-zA-Z0-9]){3,18}[a-zA-Z0-9]$`
  - **email** has to match the following regex: `^[A-Za-z0-9+_.-]+@(.+)$`
- *Address*
  - **post_code** has to have length < 10
  - **post_code** has to contain one or more digits
- *Category*
  - **depth** value has to be > 0
- *ItemCart*
  - **quantity** value has to be positive
- *Products*
  - **price** has to be a positive floating point number
- *ProductImage*
  - **ref** length < 2048 (URL standard max length).

Also we defined custom color rules that allows us to see the shipment and order statuses with different colors. The color rules match the following representation:

- Order Status
  - UNPAID
  - PAID
  - COMMITTED
- Shipment Status
  - PENDING
  - TRAVELING
  - DELIVERED

We defined the following two model instances:

- Clothes Shop
- Music Shop

Homework 2

We defined our metamodel in EMF. We defined the two model instance as:

- Music Shop
- Clothes Shop

These are located in the *homework2/it.disim.univaq.ecommerce.model* package in the repo.
Both the model contains at least an instance of each concept we defined at metaclass level. This means that any concept at the metamodel level can be instantiated in our models.

Then, we used OCL to add these additional data:

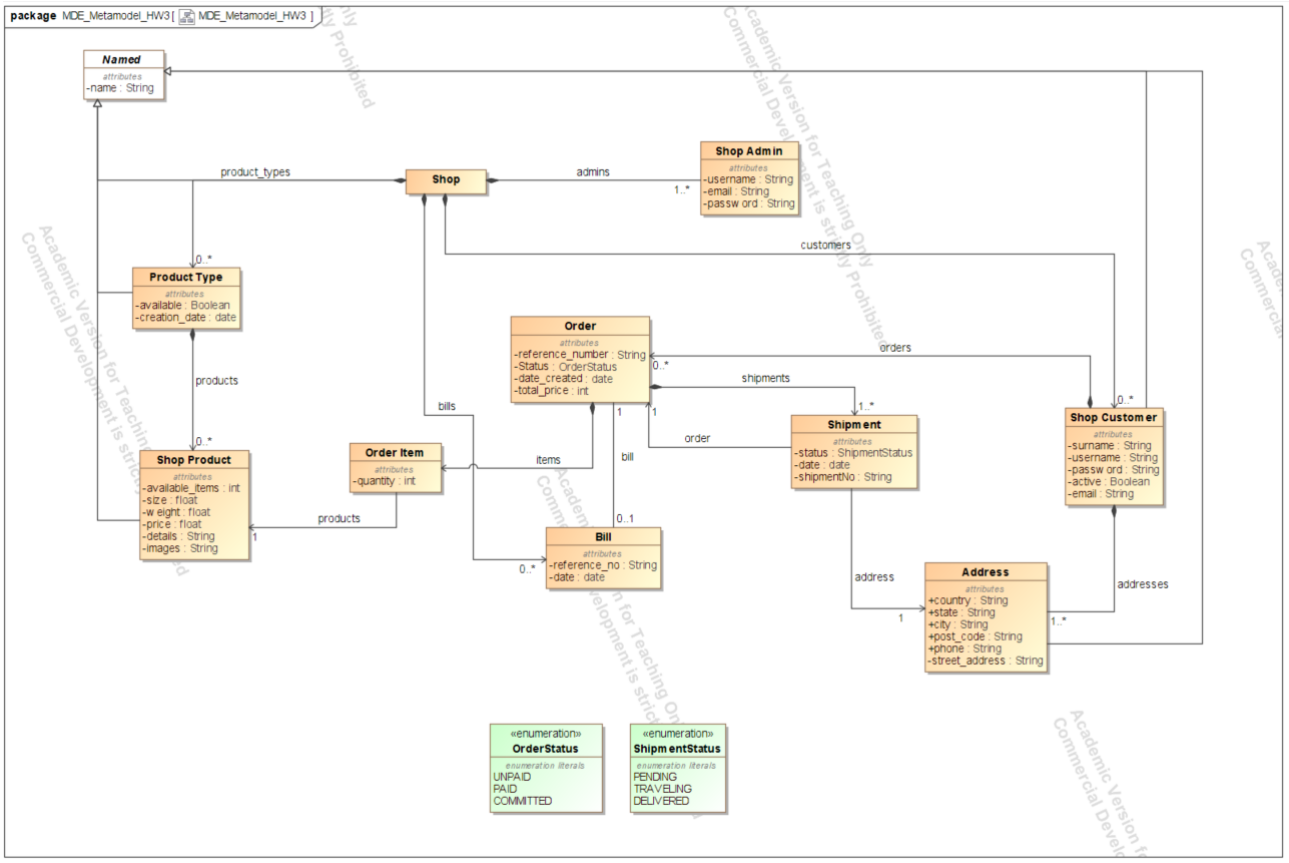
- ***Constraints***

a. *validPassword* → User.password has length >= 8. A standard simple password checking constraint.
b. *validStock* → To put an ItemCart in a cart, we have to be sure that the corresponding product has a stock value greater than the quantity value of the item.
c. *validOrderStatus* → this is splitted in two sub constraints. The meaning is that if we do not have the order in **COMMITTED** status the shipments status has to be **PENDING.** This is because the shipments of that order cannot start before that order has been committed.

- ***Operations***
  a. *Category.isLeaf()* → returns if the category has no children and so is a leaf in the category tree.
  b. *Category.getSubcategoriesNumber()* → returns the number of the children categories of a specific category.
  c. *Product.isProductAvailable()* → returns true if the stock of the product is > 0
  d. *Cart.totalProducts()* → returns the total number of products in a cart (summing the quantities of each item in the cart).
- ***Derived fields***
  a. *Cart.total_price* → this is calculated summing the price for each contained item cart
  b. *ItemCart.total_price_item_car*t → obtained multiplying the price of the corresponding product * quantity.
  c. *Order.total_price_order* → It derives from the total price of the cart
  d. *Invoice.total_price_invoice* → It derives from the total price of the order
  e. *Category.parent_category* → this calculates the parent category of the given one.

Homework 3

For this homework we refined the initial metamodel updating it to the following figure.
As we can see we removed the category tree and put all the categories at the same layer of depth. The name category has been changed with Product Type. We do not have the shop contained in an application but the shop now is a root element. Many attributes and metaclasses have been changed in terms of name.
Another structural refinement was to remove the cart metaclass, since the order corresponds to 0 or 1 cart we substituted directly the cart with the order. In this way the customer owns directly ordered instances. We also remove the inheritance from the abstract class User for the Customer and Administrator classes. Since the Admin requires very few attributes compared to the Customer we used two separated classes without inheritance.
We also added the shipment backward relation to the order, that is not present in the old model. The ProductImage metaclass has been removed and the images refs have been inserted in a field *images* of the Shop Product metaclass.



Refined metamodel

The M2M transformation is defined in the file of the repo with the following path: *homework3/Ecommerce2Shop/Ecommerce2Shop.atl*, and it is able to transform a model conforms to the old metamodel to a model conforms to the new one.

The M2T transformation is stored in the project under the following path: homework3/it.disim.univaq.ecommerce.model.acceleo. The mlt files that we used are *generate.mtl* and *generateDetails.mtl*.

Homework 4

For this homework, we create an XText language for our metamodel created in homework 2. We generated an XText project based on the ecommerce metamodel which contains all the concepts presented in the metamodel. Furthermore, we refined the language which is automatically created by XText by adding a couple of rules.

For the second task of hw2, we created a Sirius project based on our metamodel. We created a diagram which contains a fraction of the metaclasses. The diagram contains Nodes for the concepts and edges representing the references we used in our metamodel. The concepts are the following:

1. EcommerceSystem

2. Admin
3. Customer
4. Category
5. Product
6. ProductImage

In order to represent the metaclasses in the diagram we had to use **containers** to represent the containment references in the metamodel.