CREATING GLOBAL TRANSACTIONS

SAMPLE: SHOPPING CART

Last Update: Aug 2010

Table of Contents

1 Goal

In this tutorial we will continue walking through the development of a very simple application using Integranova. The tutorial covers the basic tools and steps involved in the development of applications from scratch as a way of introducing the reader to Integranova.

Following with the tutorial series, we will now expand the Tutorial 06.

In the Tutorial 06, we learned how to modify the application behavior according to the user who is logged in the application. In this new tutorial we will learn how to:

- ✓ Create a service which will delete the unfinished Purchase Orders created before a given date, independently of the customers who created them.
- ✓ Update the catalogue prices with a given percentage.
- ✓ Add a selected article to the shopping cart (purchase order), creating the shopping cart if it doesn't exist yet.

By the end of this tutorial, you will have learned how to create global services which can affect several instances that are not related among them.

In this tutorial we will start from the model we created in the previous tutorial in the series: Tutorial_06_Final.oom.

2 Concepts

Global transactions are similar to local transactions but their scope is the whole model, without being limited to objects related among them. They are on the top of the conceptual model and can access to any object of the model at any time.

Global Transactions require a formula where can be included any element that can be included in Local Transaction and other Global Transactions too.

In this tutorial, there are three cases where we must use global transactions instead of local ones:

- ✓ **Delete purchase orders:** In this case we are going to work with several instances of purchase orders that are not related among them. When you want to delete instances of a class without any structural relationship, you must use global transaction.
- ✓ **Update articles prices**: In this case we are going to work with several instances of article that are not related among them. When you want to modify instances of a class without any structural relationship, you must use global transaction.
- ✓ **Add articles to shopping cart**: In this case, we want the customers to add articles to their purchase orders without taking into account if it exists or not. A customer can see their purchase orders as shopping carts. Previously to add an article, the shopping cart (purchase order) must be created if it doesn't exist. But, by definition, the first action in a *local creation transaction* must always create the new element. This is the reason to create a global transaction to manage the process.

3 Deleting unfinished Purchase Orders

Administrators must have the chance of deleting all incomplete purchase orders created before a given date. The customers can create purchase orders, but it is possible that they never pay them. Then, the purchase process is incomplete and these purchase orders should be deleted to lighten the system data.

In order to create this new global transaction you should follow the next steps:

1- Open the *Object Interaction Diagram*:

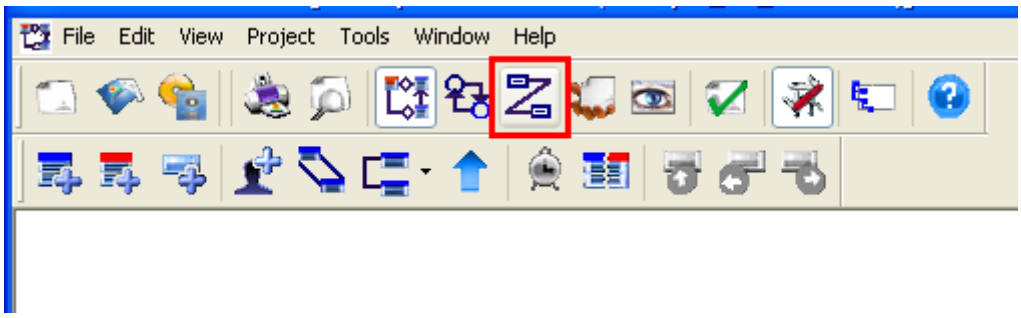

Figure 1 Open the *Object Interaction Diagram*

2- Open the *Global Service* window selecting the *Global Service …* option from *Edit* menu:

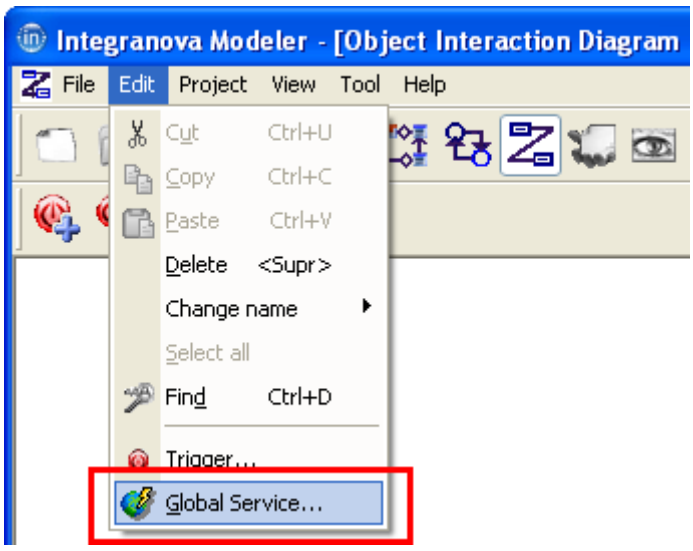

Figure 2 *Global Service* option from *Edit* menu

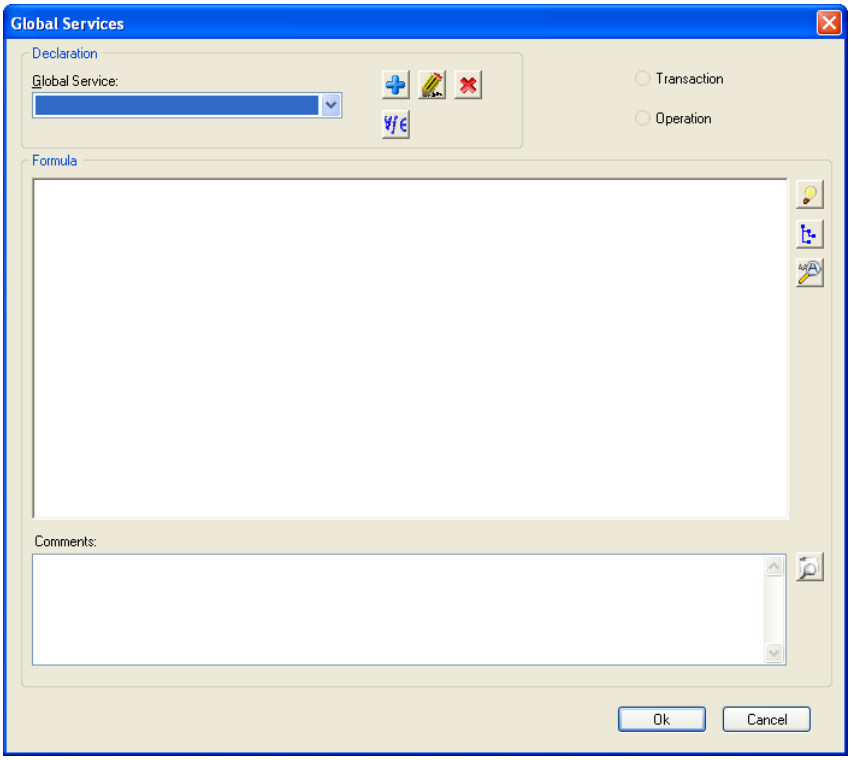

Figure 3 Global Services window

3- Click on *Add*  button to open the *Global Services Declaration* form:

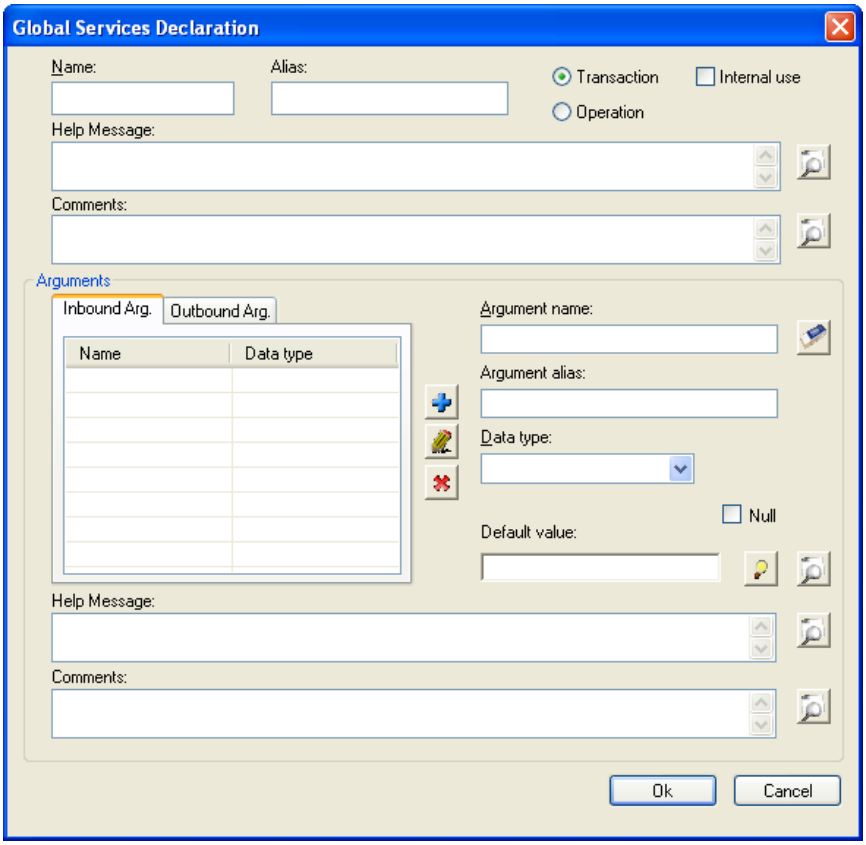

Figure 4 *Global Services Declaration* form

4- Create the new '*TDELETEPURCHASEORDERS*' global transaction fulfilling the fields with the values shown in this figure:

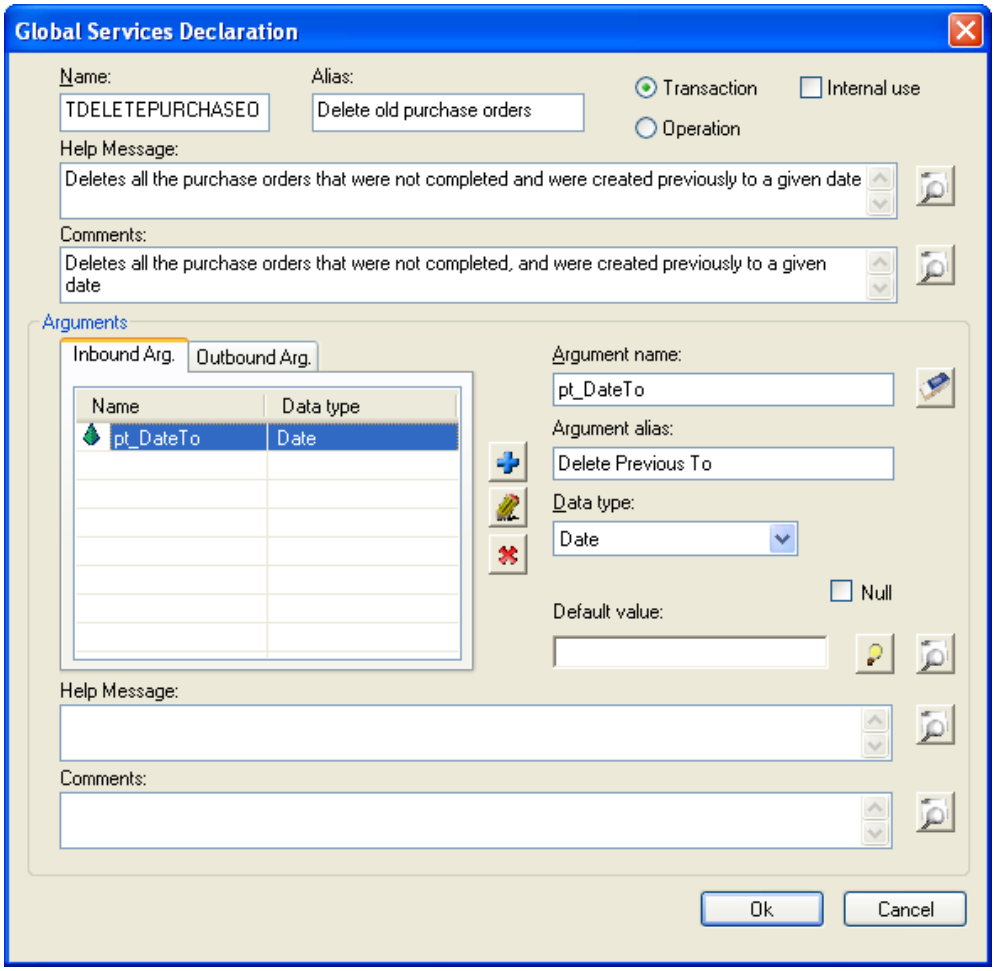

Figure 5 'TDELETEPURCHASEORDERS' global transaction declaration

5- Click on *OK* button

6- Only purchase orders in *Open* status, and earlier to the data given by the administrator will be deleted. Fill out the transaction formula as following:

```
FOR ALL PurchaseOrder WHERE (PurchaseOrder.CreationDate < pt_DateTo AND
PurchaseOrder.Status = "Open") DO
PurchaseOrder.TDELETE(PurchaseOrder)
```

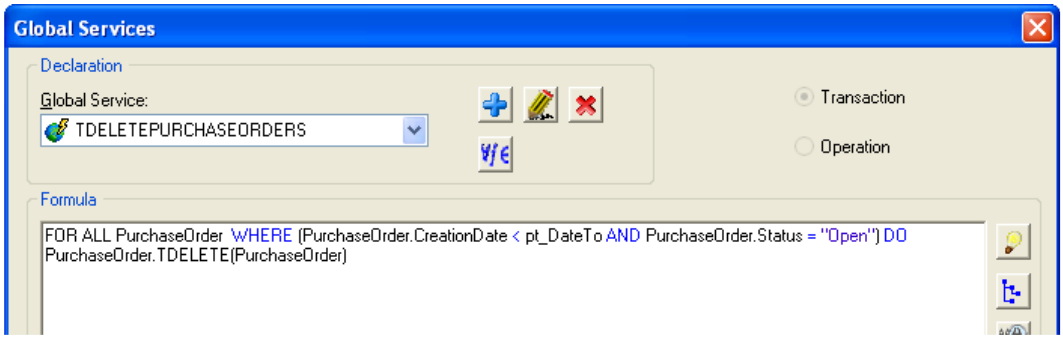

Figure 6 Global Transaction formula

Note: In these global transactions, the word 'PurchaseOrder' represents the 'PurchaseOrder' class and not a role path.

7- Click on *OK* button to save the formula and close the *Global Service* window

8- Assign the agent visibility over this new transaction. Only administrators will be able to use this service:

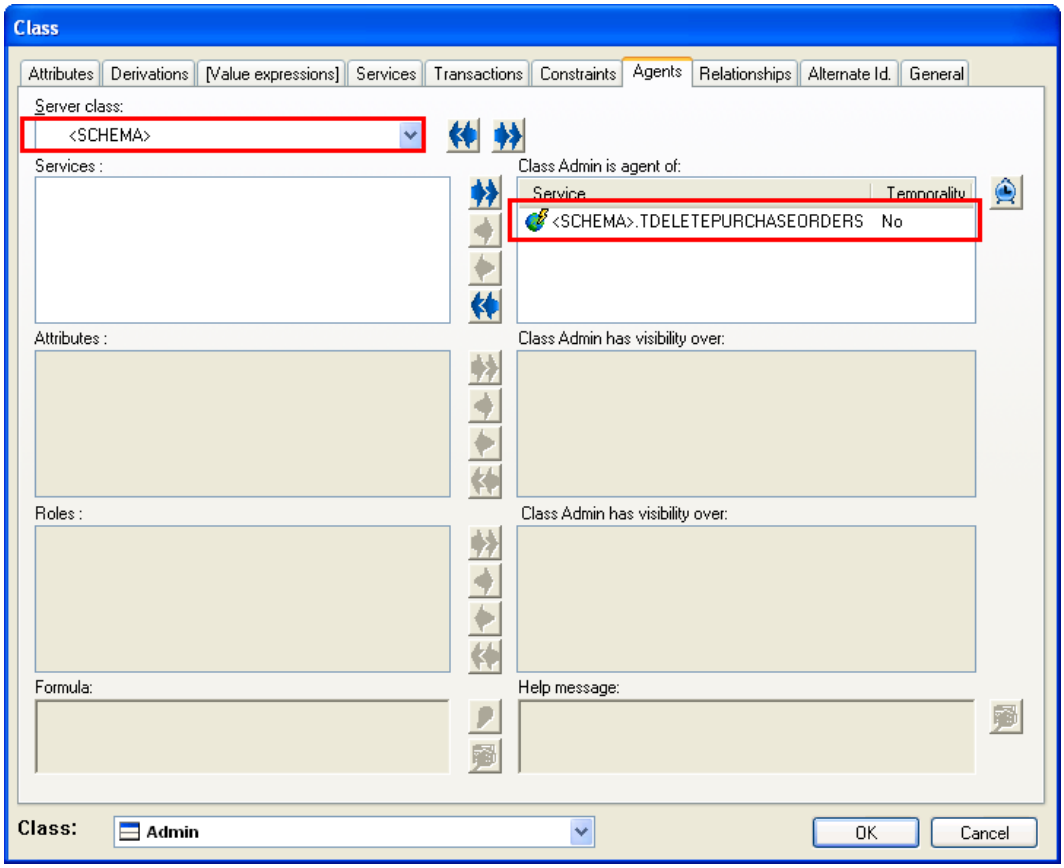

Figure 7 Assign agent visibility over the global transaction

Note: As you can see in the above figure, global services are grouped in the "***<SCHEMA>***" server class.

4 Updating the catalogue prices

All articles prices can be increased in a given percentage, for instance, due to a taxes increase. In this case, the administrator should be able to update the prices in a simple step, instead of modifying every single price.

1- First, we need to add a new event '*Updateprice*' in the 'Article' class in charge of updating its price:

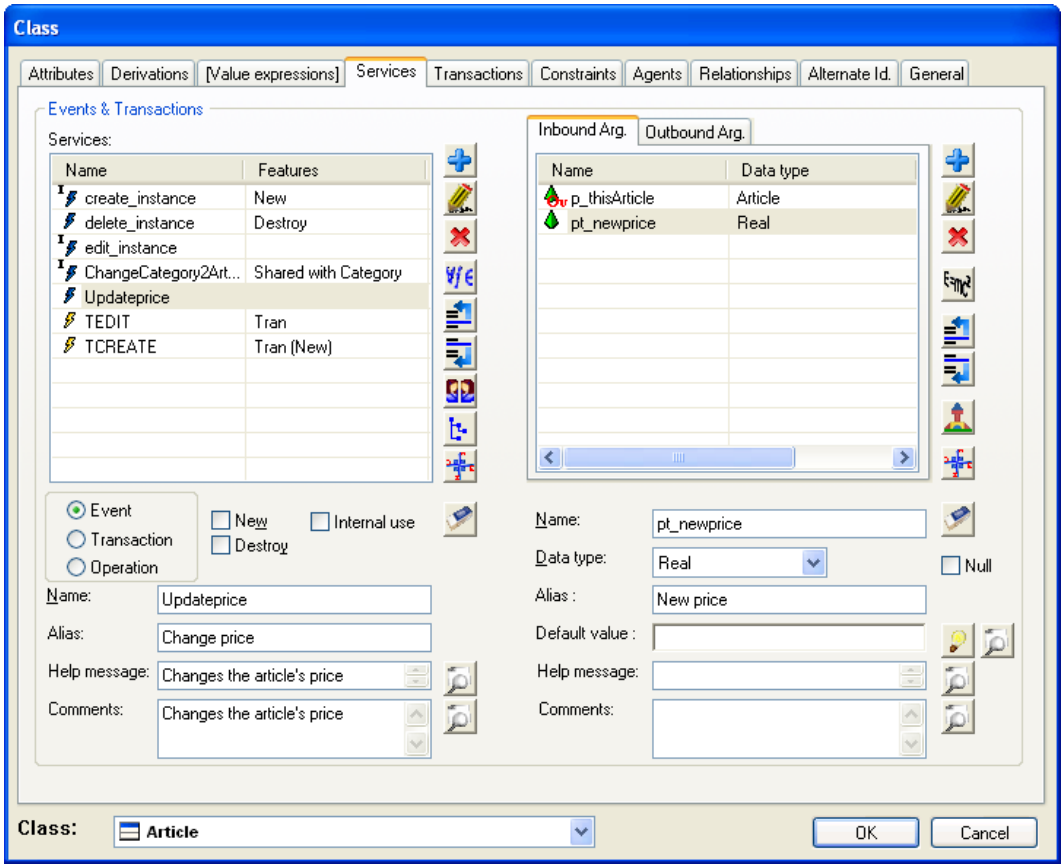

Figure 8 New 'Updateprice' event

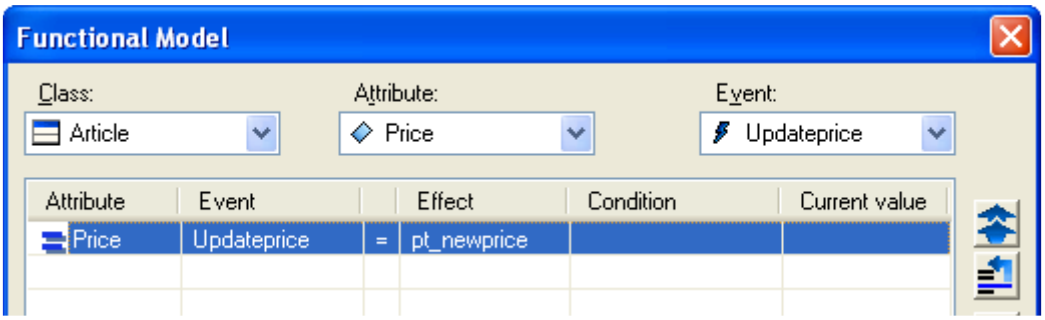

Figure 9 'Updateprice' event valuation

2- Create a global transaction called 'TUPDATEARTICLESPRICE' using the following data:

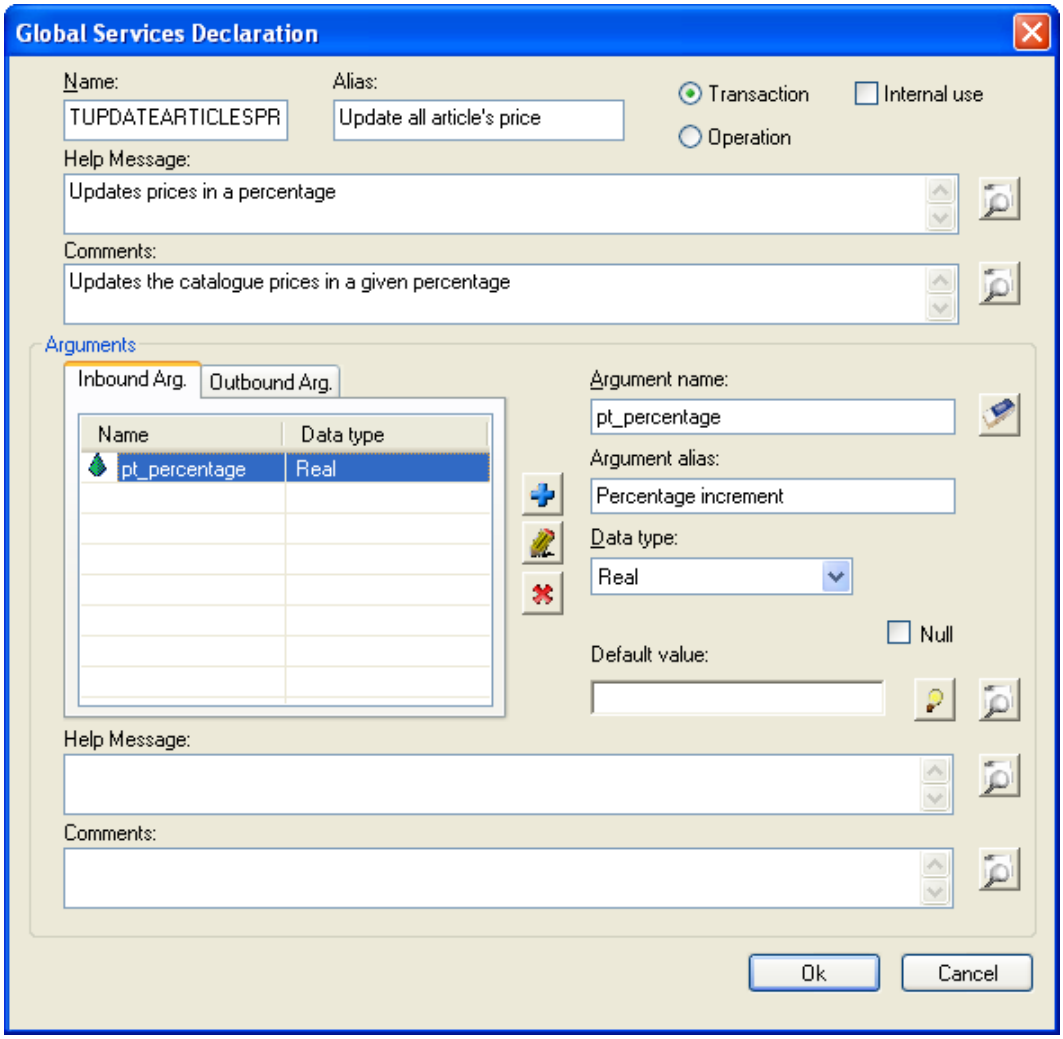

Figure 10 'TUPDATEARTICLESPRICE' global transaction

3- This transaction will only update the prices of the active articles applying the percentage indicated by the administrator:

```
FOR ALL Article WHERE (Article.ArtActive = TRUE) DO
Article.Updateprice(Article, Article.Price + Article.Price *
pt_percentage / 100)
```

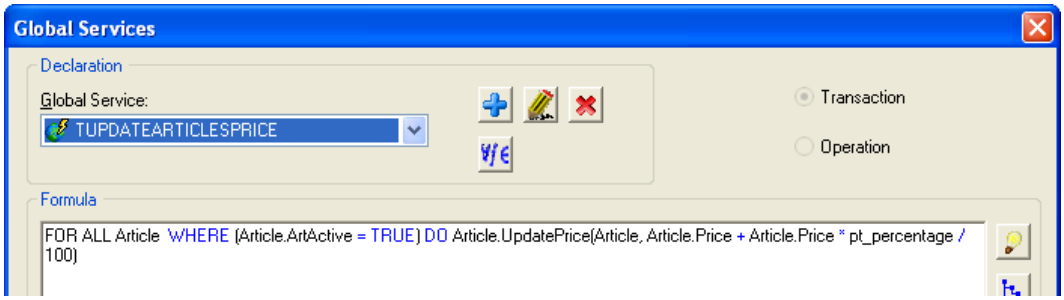

Figure 11 'TUPDATEARTICLESPRICE' global transaction formula

4- Assign the agent visibility over this new transaction. Only administrators will be able to use it.

5 Adding articles to shopping cart

At the beginning, the customer have not any shopping carts, it is only created when adding its first article. But the following articles should be added to this shopping cart, instead of creating a new one every time.

Besides, we have to avoid the customer creating specifically the shopping cart.

1- To distinguish the current purchase order (or shopping cart) from the old ones you need to establish a new relationship between *Customer* and *Purchase Order* classes.

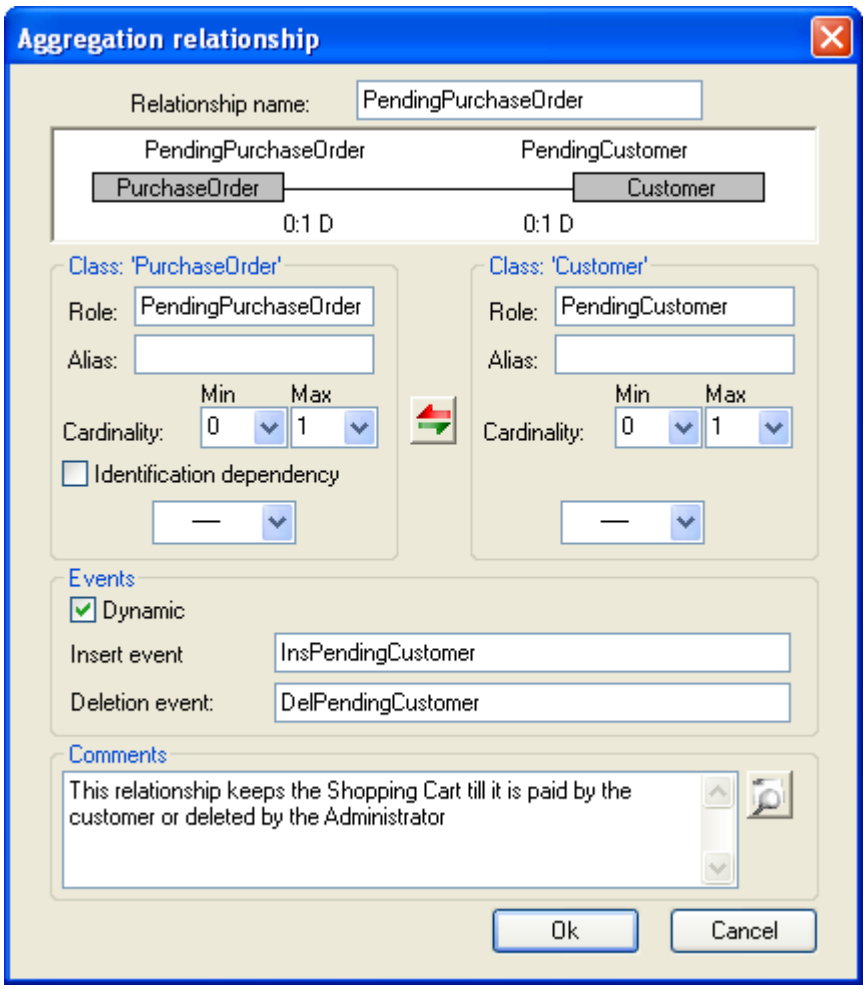

Figure 12 New relationship definition between 'PurchaseOrder' and 'Customer' classes

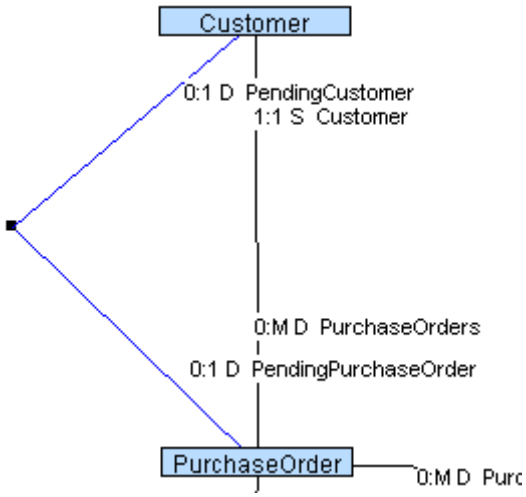

Figure 13 New relationship between 'PurchaseOrder' and 'Customer'

2- After adding this relationship, you have to do some changes in the model:

- Mark the new shared events as internal in both classes. They must not be directly executed by any agent.

- Now, the 'create_instance' events of 'Customer' and 'Purchase Order' classes have new arguments that correspond to the *Pending Purchase Order* and the *Pending Customer* respectively. You need to create or adapt the transactions which use them to take into account the new arguments:

  - 'TCREATE' transaction in 'Customer' class: We need to add this new transaction in order to ensure that the new *Pending Purchase Order* argument will be initialized with NULL value (newly created customer does not have any purchase order yet):

    Its formula will be:

    ```
    create_instance(NULL, p_Address, p_BillingAddress, p_Phone,
    p_CellPhone, p_PostalCode, p_City, p_Country, p_FirstName,
    p_Surname, p_Email, p_Passwod)
    ```

  - 'TCREATE' transaction in 'PurchaseOrder' class. In this transaction the new *Pending Customer* argument is also initialized with the connected agent:

    ```
    create_instance(AGENT_Customer, NULL, AGENT_Customer)
    ```

- After paying, the shopping cart process is finished. The shopping cart (or *Current purchase order*) should not be modified adding new articles, but we want to maintain it as a purchase order of the Customer. Instead of deleting, it we can delete the relationship between the Customer and the *Current purchase order*. This will be done modifying the 'TPAY' transaction formula. You should call to the shared event named 'DelPendingCustomer' in charge of deleting this relationship:

```
IF( EXIST( PaymentType ) = TRUE) {
   DelPaymentType2PurchaseOrder(THIS, PaymentType)
}.

InsPaymentType2PurchaseOrder(THIS, pt_PaymentType).

DelPendingCustomer(THIS, PendingCustomer).

Pay(THIS)
```

3- Create a global transaction called 'TADDARTICLETOCART'. To execute this transaction, a customer needs to fulfill two arguments, the desired article and its number of units:

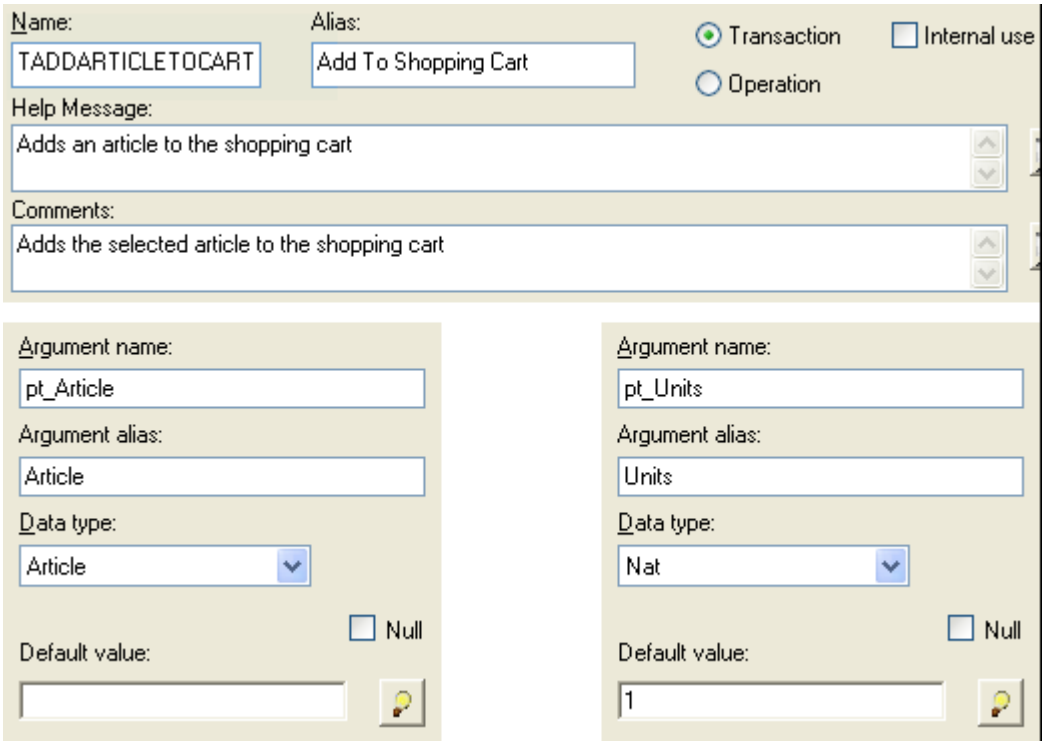

Figure 14 'TADDARTICLETOCART' global service data

4- This transaction will create a new *Pending Purchase Order* (*shopping cart*) if it doesn't exist. After that, a new purchase order line will be added to the pending shopping cart of the logged customer. (We do not know if it has been recently created or it previously existed).

```
IF (EXIST (AGENT_Customer.PendingPurchaseOrder) = FALSE) {
  PurchaseOrder.TCREATE()
}.

POLine.create_instance(AGENT_Customer.PendingPurchaseOrder, pt_Article,
pt_Units, pt_Article.Price)
```

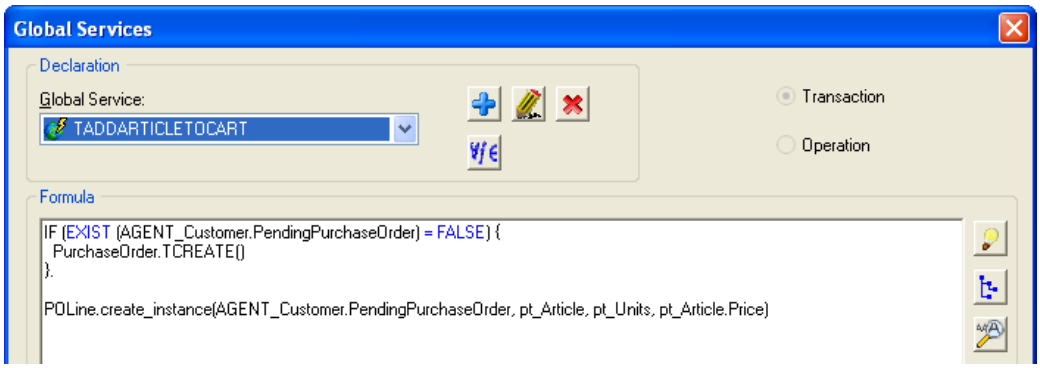

Figure 15 'TADDARTICLETOCART' global transaction formula

5- Assign the agent visibility over this new transaction. Only customers will be able to use it.

6- Finally, mark the 'create_instance' event of 'POLine' class as internal.

6 Default user interface

Integranova Modeler offers the possibility of creating a customized user interface using the features associated to the *Presentation Model*. All these features will be presented from Tutorial 9 in the series tutorials to the last one (Tutorial 15).

Meanwhile, **Integranova Modeler** provides a default user interface where all the functionality defined in the model can be accessed. It will be possible to access to the application using any of the agents defined in the model.

The main menu will have as many menu entries as classes defined in the model (the name will be the alias of the class). Each menu entry will have the following subentries: one scenario to show **one instance** of the class, another scenario to **list all the instances of the class** and as many scenarios as **non-internal services** we have defined in the class.

If there are global services defined in the model, a new menu entry labeled as "Global Services" is added to the main menu, where all the **global services** defined are available.

When an agent is connected to the application, he only will have access to the services that he can execute and the roles and attributes over which he has visibility. The menu entries which the agent has no permission (no execution, no visibility, no navigation) will be hidden.