EXECUTION FLOW BASED ON CONDITIONS

SAMPLE: SHOPPING CART

Last Update: Jul 2011

Table of Contents

1 Goal

In this tutorial we will finish the development of a very simple application using Integranova. Following with the tutorial series, we will now expand the Tutorial 14, with this new tutorial (Tutorial 15) that will build on the previous one.

In previous series we have widely seen service interaction units and some of their main properties. In this tutorial series we are going to work on defining execution flows based on whether these services execution succeed or failed. This concept is named *Conditional Navigation.*

By the end of this tutorial we will be able to define an execution flow based on conditions checked after a service execution.

With this last tutorial we will have seen the main concepts that Integranova offers to define and develop applications.

2 Execution flow through the concept of conditional navigation

When dealing with a system like the one we are defining along these series, for instance, an online shop, it is quite common that once we finish adding items to our cart, we go to a kind of checking out scenario. In this sort of scenarios, we are usually provided with a review listing all the items we have added to our cart, the total amount, discount if so applied… It is also very usual that reached that point, we are provided with some options (services) that allow us going back to add, change units or delete any specific item, delete the whole current cart or if we agree the terms, end up buying.

That’s the kind of situation we are going to model now.

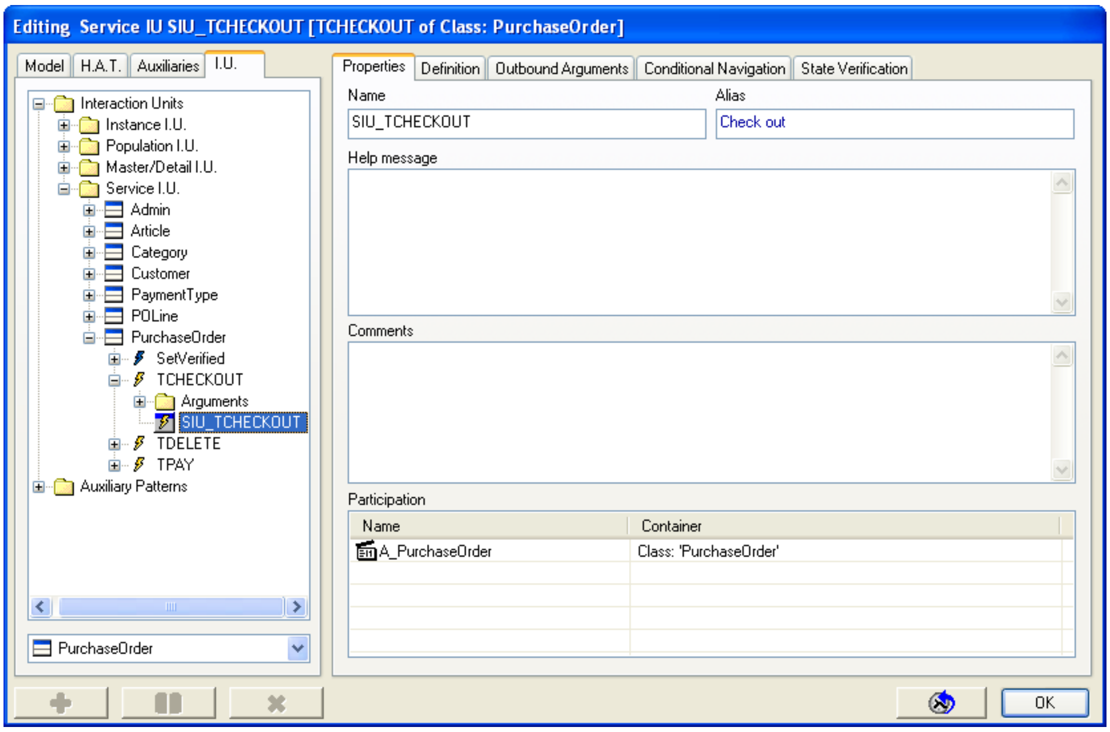
Let’s comment the execution flow that we would like to achieve. In our system, a customer is able to add items (articles) to the shopping cart. Anytime he can ‘close’ the shopping cart executing the *Check Out* service. If this service is correctly executed, no system error raises, then the system will show the user the total amount of the shopping cart (purchase order) and if a discount was applied. In addition to this, when closing the prior scenario, the system will automatically check a defined condition formula, and if fulfilled will open a sort of questioning scenario asking for where to take the customer to:

- ✓ Possibility of getting the user back to the first scenario where the customer will be able to add more articles, edit some article’s units or delete existent ones.
- ✓ Delete the whole shopping cart, if the agent connected changes his mind
- ✓ Carry on with the payment process

As you can see, the system will define a sort of execution flow, guiding the user through a well defined process.

This, in Integranova is named *Conditional Navigation* and we will see now how to define it in the model.

So, let’s go back to work and open the Tutorial_14.oom model file. We are going to open the Presentation Model, go to the *U.I.* tab, expand the *Interaction Unit*s’ folder, expand the *Service I.U* folder, expand the ‘PurchaseOrder’ class, expand the ‘TCHECKOUT’ transaction service and finally select the ‘SIU_TCHECKOUT’:

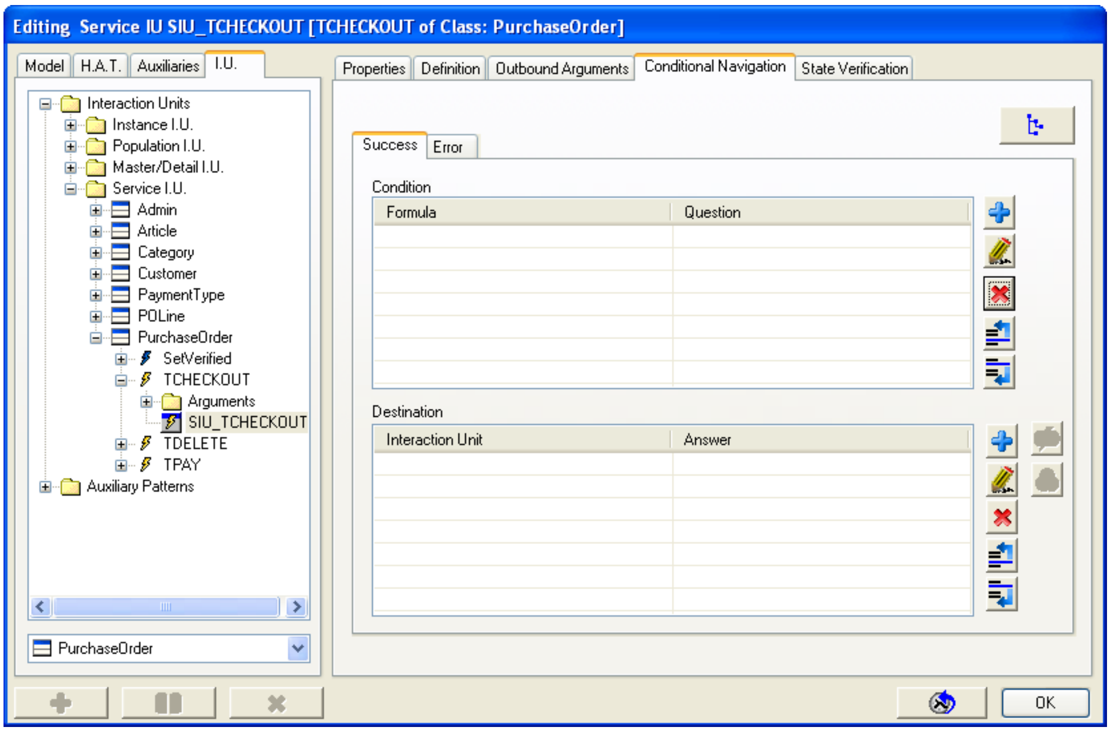

Editing Service IU SIU_TCHECKOUT [TCHECKOUT of Class: PurchaseOrder]

Name: SIU_TCHECKOUT
Alias: Check out
Help message
Comments
Participation

Name	Container
A_PurchaseOrder	Class: 'PurchaseOrder'

Figure 1 Service Interaction Unit TCHECKOUT of class PurchaseOrder

We have deeply seen the properties of the Service Interaction Units in previous series, so we will just focus on the *Conditional Navigation* tab. Then, select *Conditional Navigation* tab:


Editing Service IU SIU_TCHECKOUT [TCHECKOUT of Class: PurchaseOrder]

Condition

Formula	Question

Destination

Interaction Unit	Answer

Figure 2 Conditional Navigation

As we can see, initially it is empty. There are two tabs: *Success* and *Error*. These tabs separate the condition formulas in order of whether they have to be evaluated when the service succeeds or fails.

We are going to define a condition formula for the success case. This means that if the service 'TCHECKOUT' is correctly executed, then the condition we are going to define will be automatically checked. If we see again the Figure 2, we can see we have two areas. In the top, the *Condition* area is the area in which we can define our conditions plus a question. For instance, imagine we define a condition, whichever, that is fulfilled, and as a question we ask the user: "What would you like to do now?"

In the bottom area, the *Destination* area is where we can define the possible target or destination scenarios that the user can select as an answer to the question defined in the *Condition* area.

Let's define in the model, our particular case. Press the *Add* button in the top area for the *Success* tab:

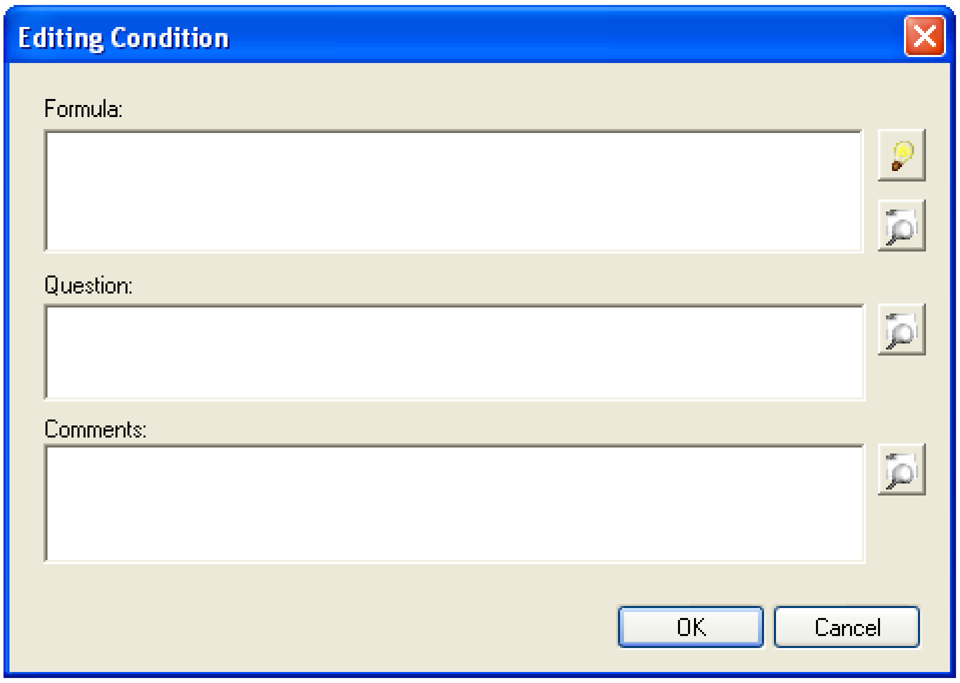


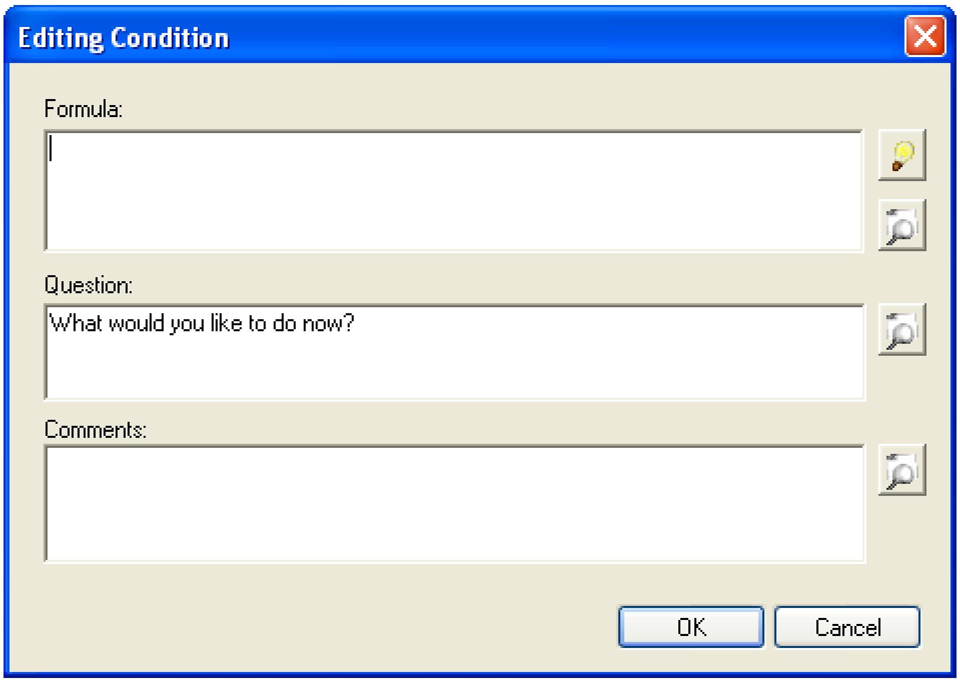
Figure 3 Adding a condition

The *Editing Condition* dialog appears asking for a formula, a question and some comments.

If it is not needed to add any formula, we can leave the *Formula* textbox empty. An empty condition formula means that anytime the service is executed the question entered in the *Question* text box, will be shown automatically to the application user. On the other hand, if we had a condition formula defined, obviously it would have to be fulfilled in order to show the question to the final user.

In our case, we would like to ask the user where to go in an automatic way, so we don't define any formula. However, we do define a question. The question will be: "What would you like to do now?"

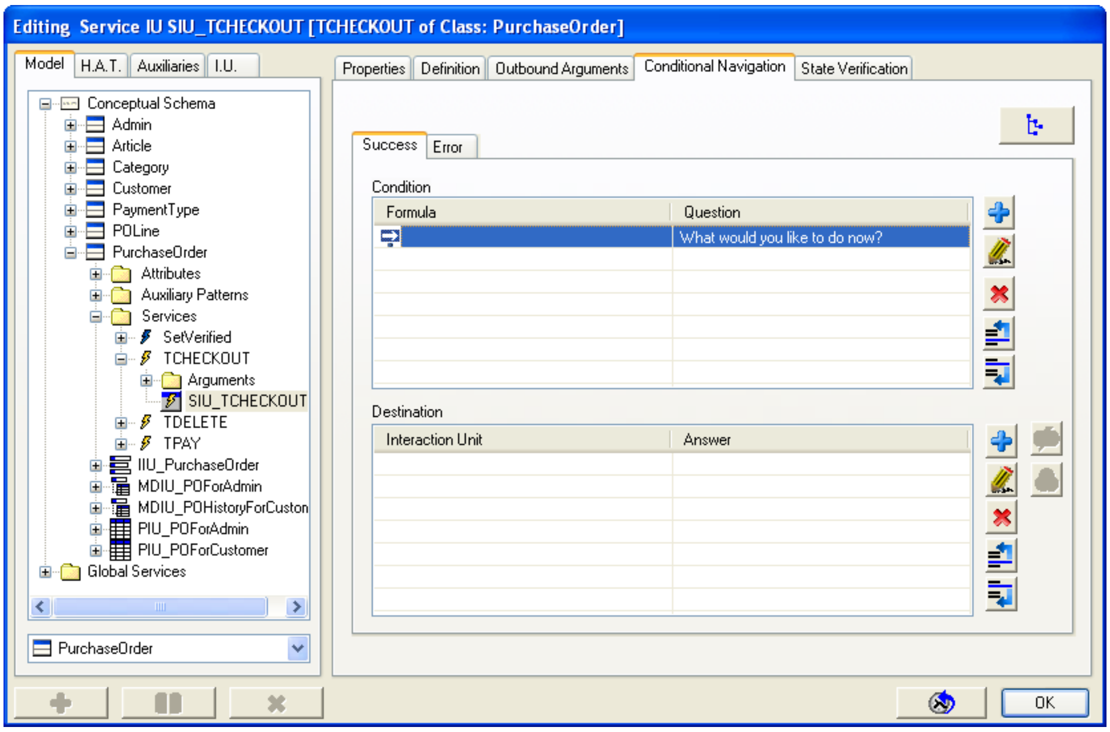
Then let's enter it:



What would you like to do now?

Figure 4 Defining a question

We press the *OK* button and we have our first success condition defined:



Conceptual Schema
- Admin
- Article
- Category
- Customer
- PaymentType
- POLine
- PurchaseOrder
  - Attributes
  - Auxiliary Patterns
  - Services
    - SetVerified
    - TCHECKOUT
      - Arguments
      - SIU_TCHECKOUT
    - TDELETE
    - TPAY
  - IIU_PurchaseOrder
  - MDIU_POForAdmin
  - MDIU_POHistoryForCuston
  - PIU_POForAdmin
  - PIU_POForCustomer
- Global Services

Formula	Question
	What would you like to do now?

Interaction Unit	Answer

Figure 5 Conditional navigation success conditions

We can add as many different conditions as needed. Obviously, as we have already defined an empty condition formula, if we had to define some other we would need to mandatory specify a formula condition. Otherwise, **Integranova Modeler** will warn us.

Note that if more than one condition is defined, it is also possible to change the conditions order, making use of the buttons in the right side of the top *Condition* area.

Now, let's define the possible destination scenarios that will be offered to the user. So we select our just created condition, go at the bottom *Destination* area and press the *Add* button:

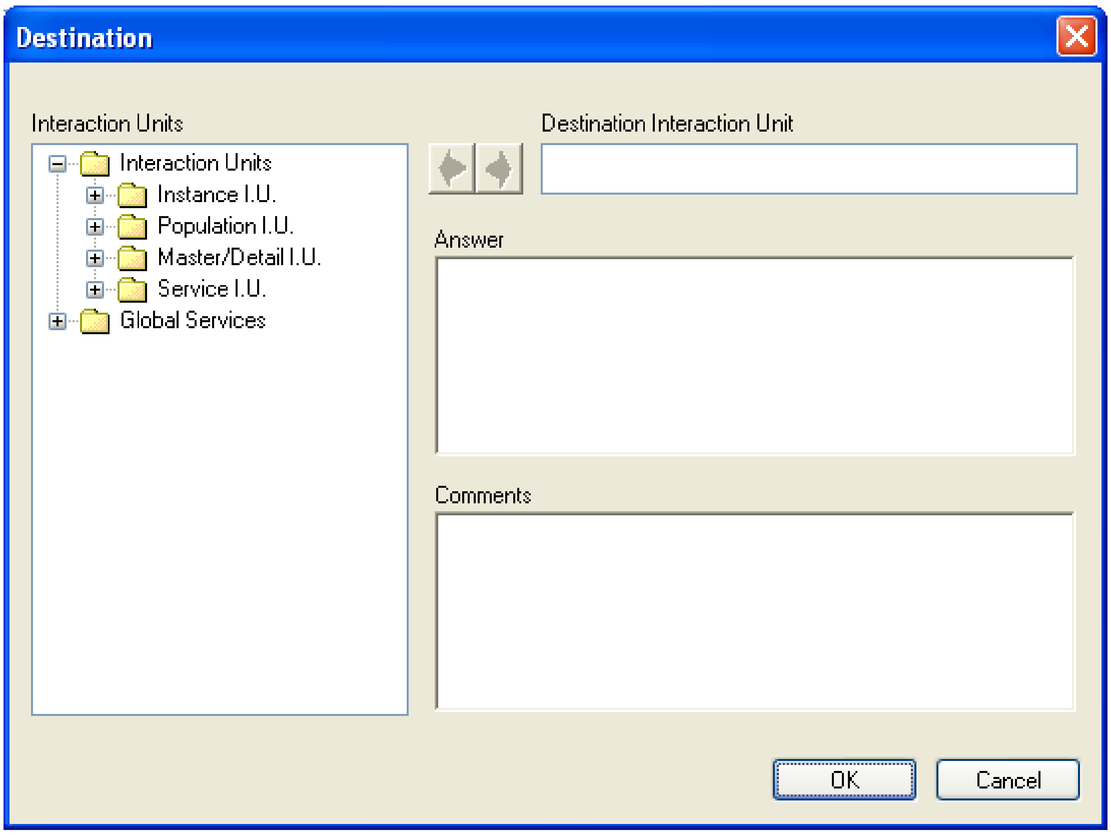


Figure 6 Defining a destination scenario

In the Figure 6, you can see in the left side of the dialog. The possible interaction units we can choose as the destination interaction unit. The only thing we have to do here is to expand the folders as needed, select the interaction unit we want to, and eventually add it to the *Destination Interaction Unit* text box using the *Add Destination IU* button.

In addition to this, we enter an answer in the *Answer* textbox. An answer, as previously stated, is a sort of option given to the final user as text that provides enough information to the user to know the next scenario to be selected. If there are more than one possible destination (later on we will add some more) we differentiate them through these answers.

Well for our case, we would like the first option to be the possibility of returning back to the scenarios where we can add more items, or change units or delete some. This is going back to the scenario 'MDIU_POLinesShoppingCart' of 'POLine' class. As an answer to reach that scenario we will introduce: "Return to the shopping cart".

So, we expand the *Master/Detail I.U* folder, we expand the 'POLine' class and we select the MDIU_POLinesShoppingCart interaction unit:

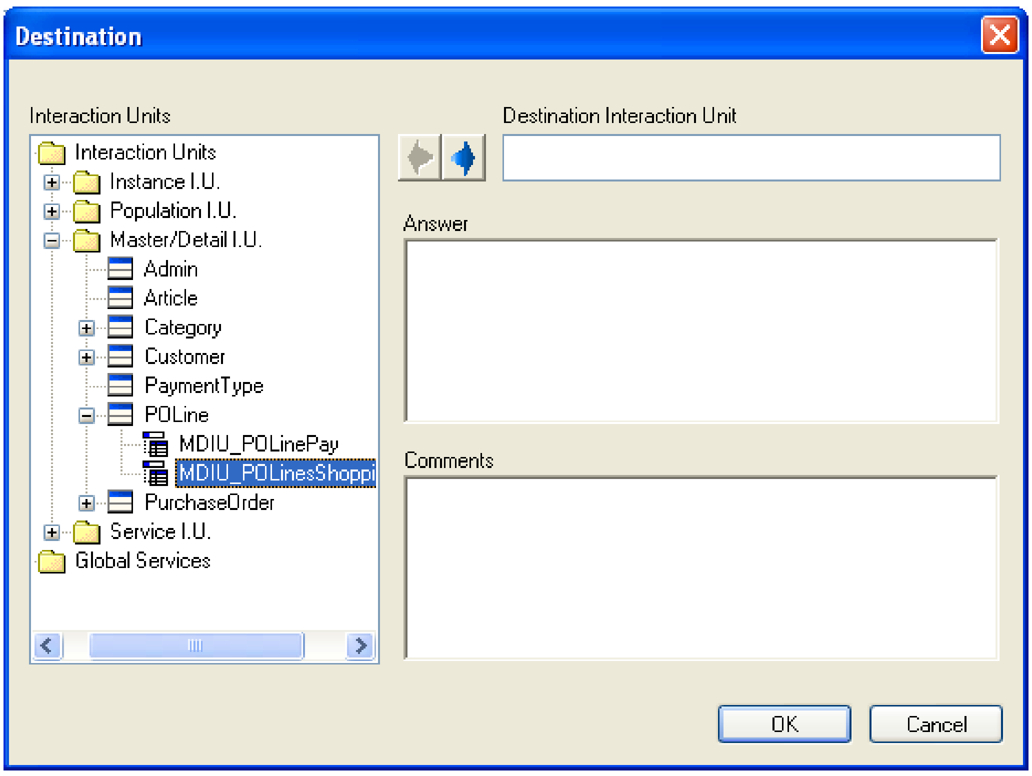
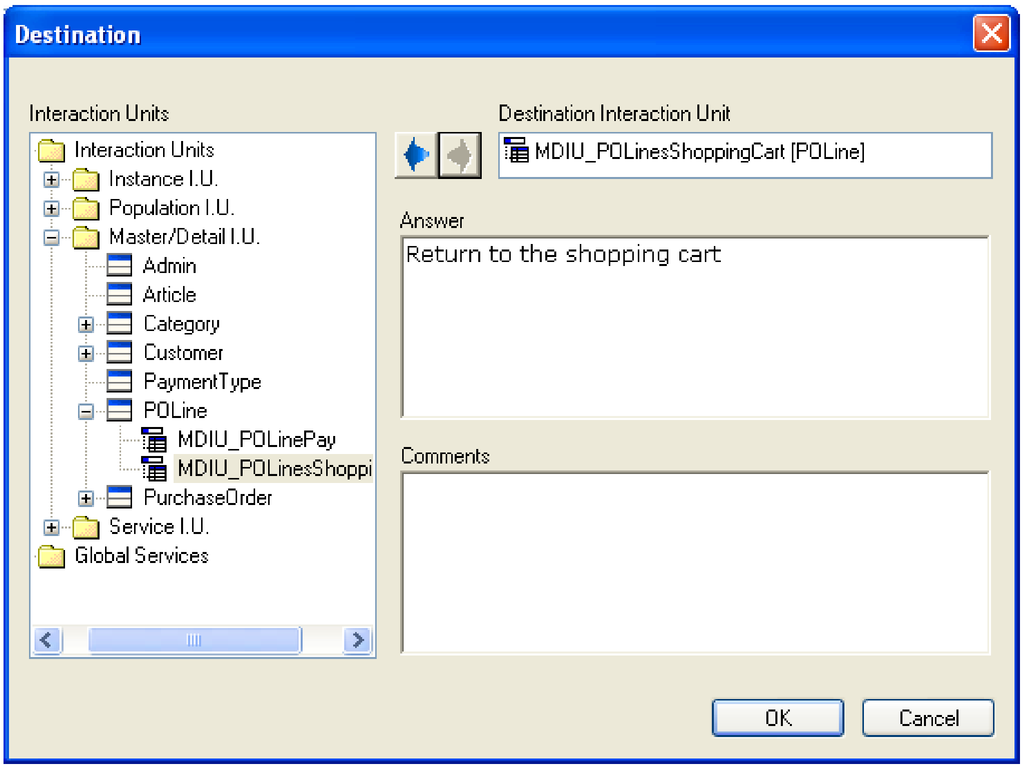


Figure 7 Selecting the Destination interaction unit

Notice that once selected an interaction unit, the *Add Destination IU* button is automatically enabled. Then, click on this button to add the interaction unit selected and enter the answer in the *Answer* textbox:

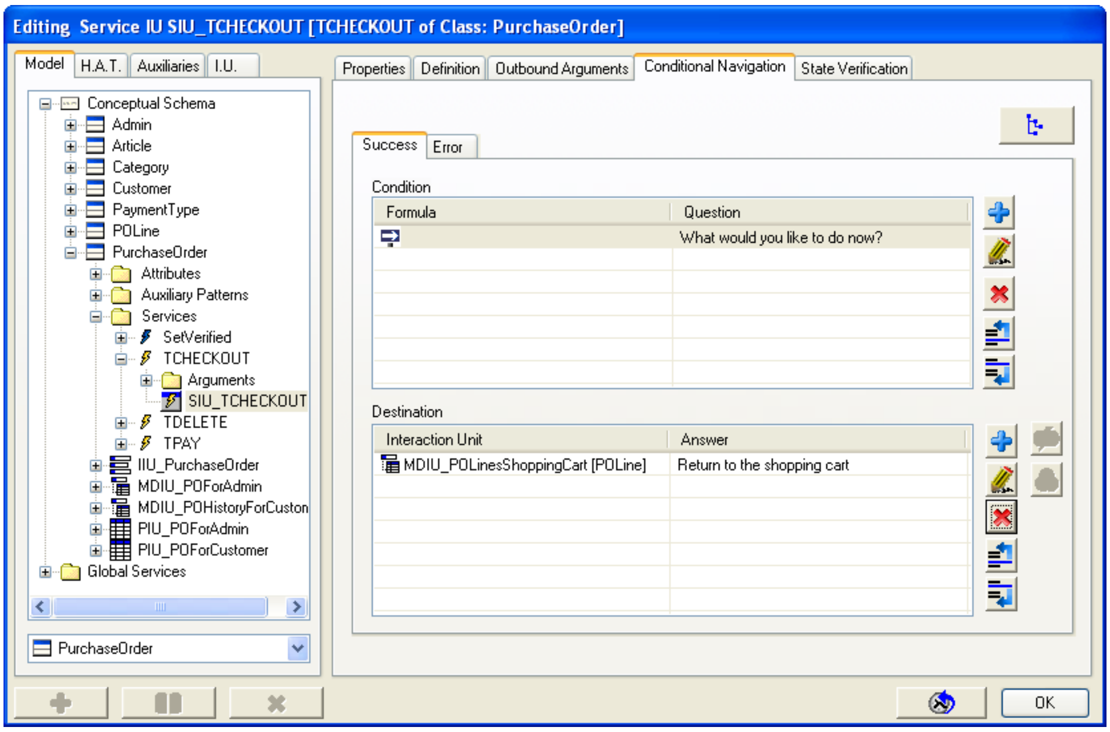


Destination Interaction Unit: MDIU_POLinesShoppingCart [POLine]

Answer: Return to the shopping cart

Figure 8 Defining a destination interaction unit

When pressing the *OK* button, we will have our first destination added:



Editing Service IU SIU_TCHECKOUT [TCHECKOUT of Class: PurchaseOrder]

Condition

Formula	Question
	What would you like to do now?

Destination

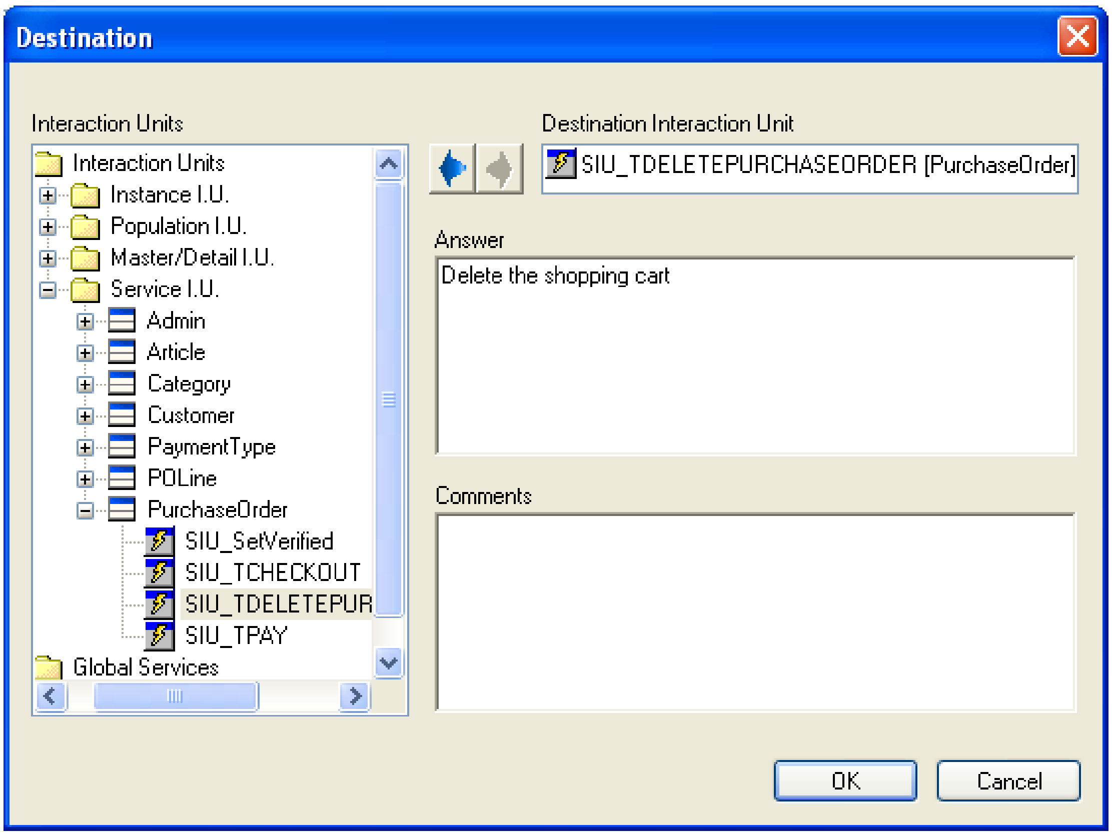
Interaction Unit	Answer
MDIU_POLinesShoppingCart [POLine]	Return to the shopping cart

Figure 9 Defining conditional navigation

Let's carry on with the two destinations left we need to add.

The second destination unit will be the possibility to delete the purchase order (shopping cart). Then, we press again the *Add* button as to add a new destination, and now we repeat expanding the folders as needed till we identify the interaction unit we would like to use. In this case, we are going to select the 'SIU_TDELETEPURCHASEORDER' from 'PurchaseOrder' class. We enter the answer: "Delete the current purchase order".

Finally, we have our second destination interaction unit ready to use:



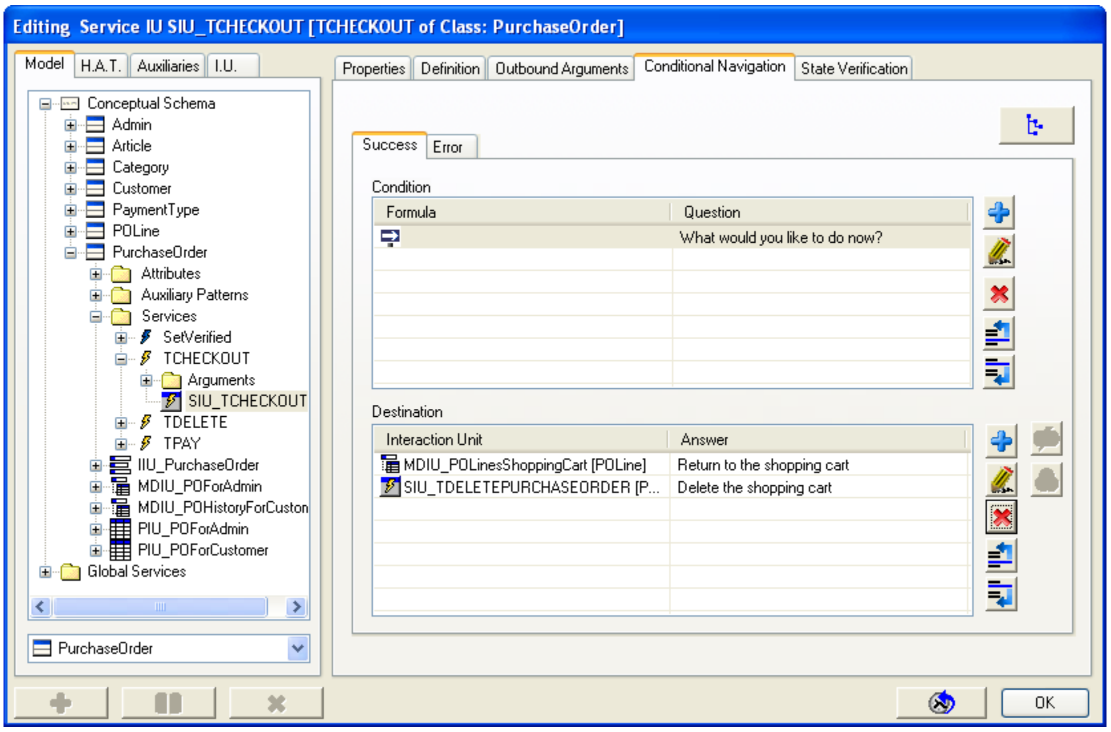
Destination Interaction Unit: SIU_TDELETEPURCHASEORDER [PurchaseOrder]

Answer: Delete the shopping cart

Comments:

Figure 10 Destination interaction unit

And, when pressing the *OK* button, this destination interaction unit is added to the list of possible destinations:



Condition

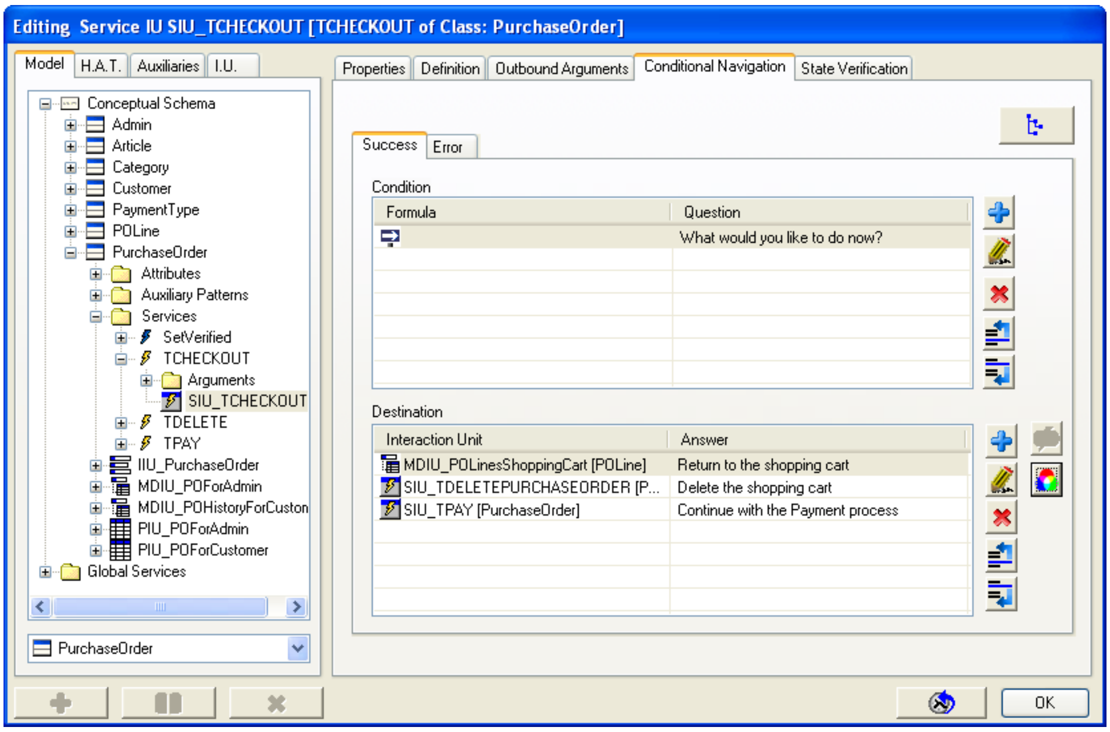
Formula	Question
	What would you like to do now?

Destination

Interaction Unit	Answer
MDIU_POLinesShoppingCart [POLine]	Return to the shopping cart
SIU_TDELETEPURCHASEORDER [P...	Delete the shopping cart

Figure 11 Second possible destination added

Eventually, carry on with our requirements; we are going to create our last possible destination interaction unit. In this case, we would like our customer to pay the amount of the shopping cart. So we are going to offer him the possibility of carry on with the payment process. Then, we press again the *Add* button, this time, we choose the 'SIU_TPAY' Service Interaction Unit of 'PurchaseOrder' class, and enter the answer that will be shown: 'Continue with the Payment process'. Once this is created we have our third possible answer created:


Editing Service IU SIU_TCHECKOUT [TCHECKOUT of Class: PurchaseOrder]

Model | H.A.T. | Auxiliaries | I.U.

- Conceptual Schema
  - Admin
  - Article
  - Category
  - Customer
  - PaymentType
  - POLine
  - PurchaseOrder
    - Attributes
    - Auxiliary Patterns
    - Services
      - SetVerified
      - TCHECKOUT
        - Arguments
        - SIU_TCHECKOUT
      - TDELETE
      - TPAY
    - IIU_PurchaseOrder
    - MDIU_POForAdmin
    - MDIU_POHistoryForCuston
    - PIU_POForAdmin
    - PIU_POForCustomer
- Global Services

PurchaseOrder

Properties | Definition | Outbound Arguments | Conditional Navigation | State Verification

Success | Error

Condition

Formula	Question
	What would you like to do now?

Destination

Interaction Unit	Answer
MDIU_POLinesShoppingCart [POLine]	Return to the shopping cart
SIU_TDELETEPURCHASEORDER [P...	Delete the shopping cart
SIU_TPAY [PurchaseOrder]	Continue with the Payment process

OK

Figure 12 Third possible destination added

3 Initialize destination IU defined in conditional navigation

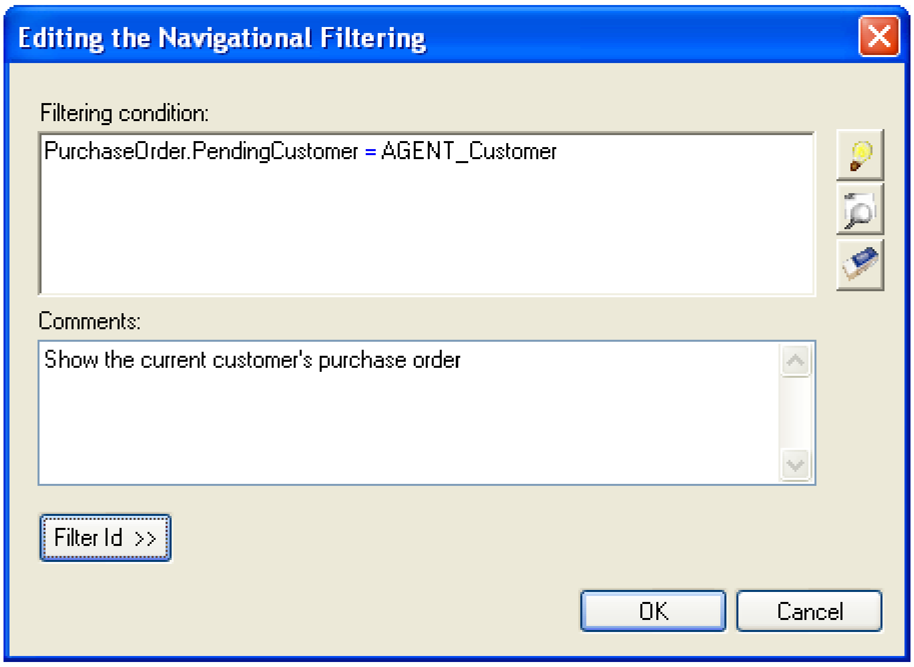
We have learnt how to add possible destinations interaction units, which will be offered by the system any time a purchase order check out service is executed. So we have a kind of predefined execution flow. In addition to this, we could enrich this process making the destination interaction units to be somehow **initialized**. For example, if there are Services Interaction Units offered, we could make them appear with some inbound arguments already initialized. Or, if we had, for instance, a Population Interaction Unit, we could apply a navigational filtering that restricts the instances to those needed.

We are going to further see this in this section.

Let's see how we could do this, following the order in which we have defined the possible destination interaction units.

The first interaction unit in our list of possible destinations is a Master/Detail Interaction Unit that allows returning to the shopping cart that is representing. We could define here a navigational filtering (the same way we learnt in previous series) and initialize this target interaction unit in a way that only shows the current shopping cart of the logged customer. Then, to define this, we need to select this destination and press the

Navigational filtering button placed in the right side of the dialog. When doing so, the *Navigational filtering* dialog is shown. Then, we enter the formula in which we specify our requirement:


Editing the Navigational Filtering

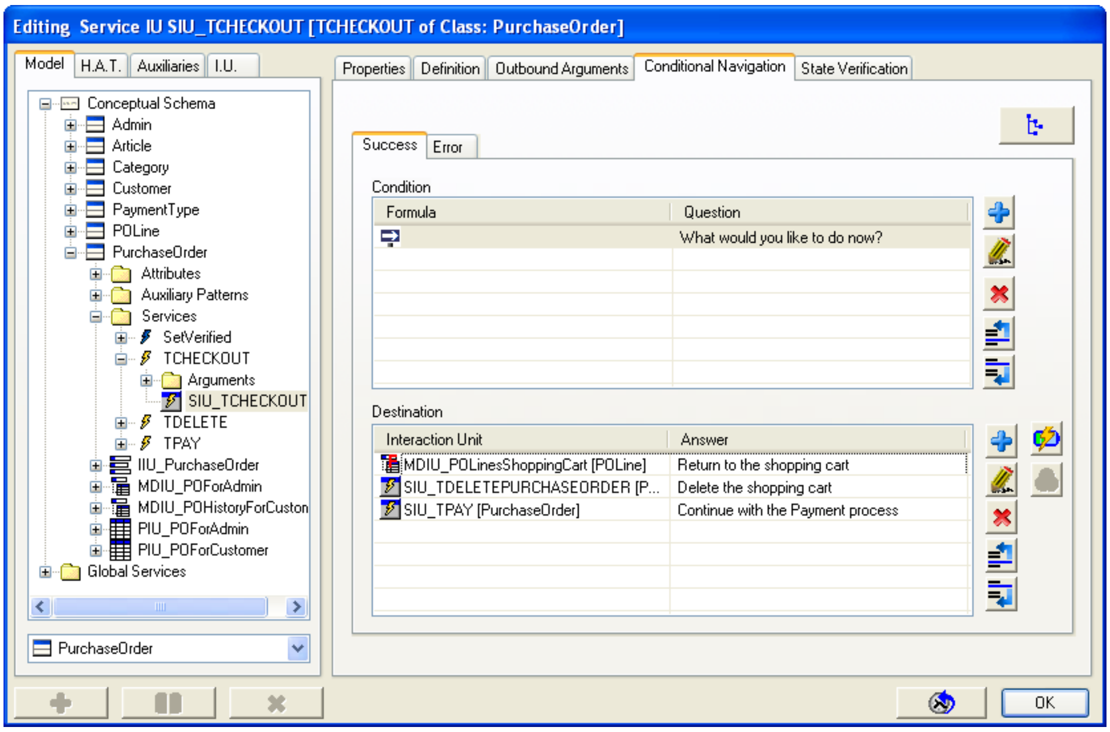
Filtering condition:
PurchaseOrder.PendingCustomer = AGENT_Customer

Comments:
Show the current customer's purchase order

Figure 13 Navigational filtering for M/D shopping cart

By means of this formula we are telling the system to show just the current shopping cart of the current customer logged in the application.

After pressing the *OK* button, the destination interaction unit shows a red 'F' over, which means that a navigational filtering has been defined for this interaction unit:


Editing Service IU SIU_TCHECKOUT [TCHECKOUT of Class: PurchaseOrder]

Condition

Formula	Question
	What would you like to do now?

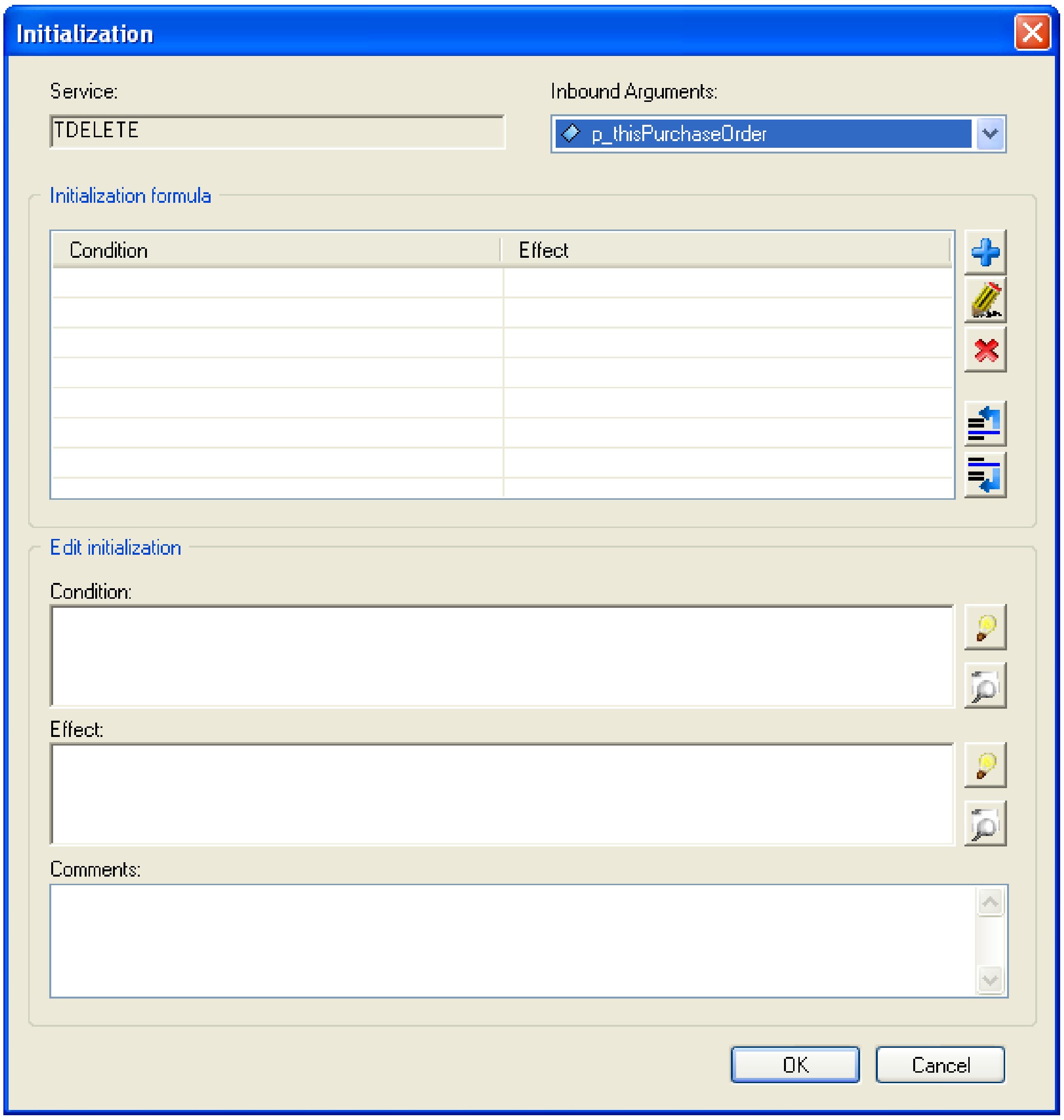
Destination

Interaction Unit	Answer
MDIU_POLinesShoppingCart [POLine]	Return to the shopping cart
SIU_TDELETEPURCHASEORDER [P...	Delete the shopping cart
SIU_TPAY [PurchaseOrder]	Continue with the Payment process

Figure 14 Navigational filtering icon over MDIU_POLinesShoppingCart

The two destination interaction units left are Service Interaction Units. If you select one of these units, you will see that the button of the navigational filtering disables. This is normal due to it is not possible to define a navigational filtering over a Service Interaction Unit. It won't have sense. However, it is possible to initialize its inbound arguments. To do so, we can see that when selecting one of these two Service Interaction Units, the *Arguments Initialization* button enables. By means of this button we can define or better said, initialize the inbound arguments of the destination Service Interaction Unit.

Let's then define initializations for the "Delete the shopping cart" option. So, we select the 'SIU_TDELETEPURCHASEORDER' Service Interaction Unit, and click on the *Arguments initialization* button:



Service: TDELETE

Inbound Arguments: p_thisPurchaseOrder

Condition:

Effect:

Comments:

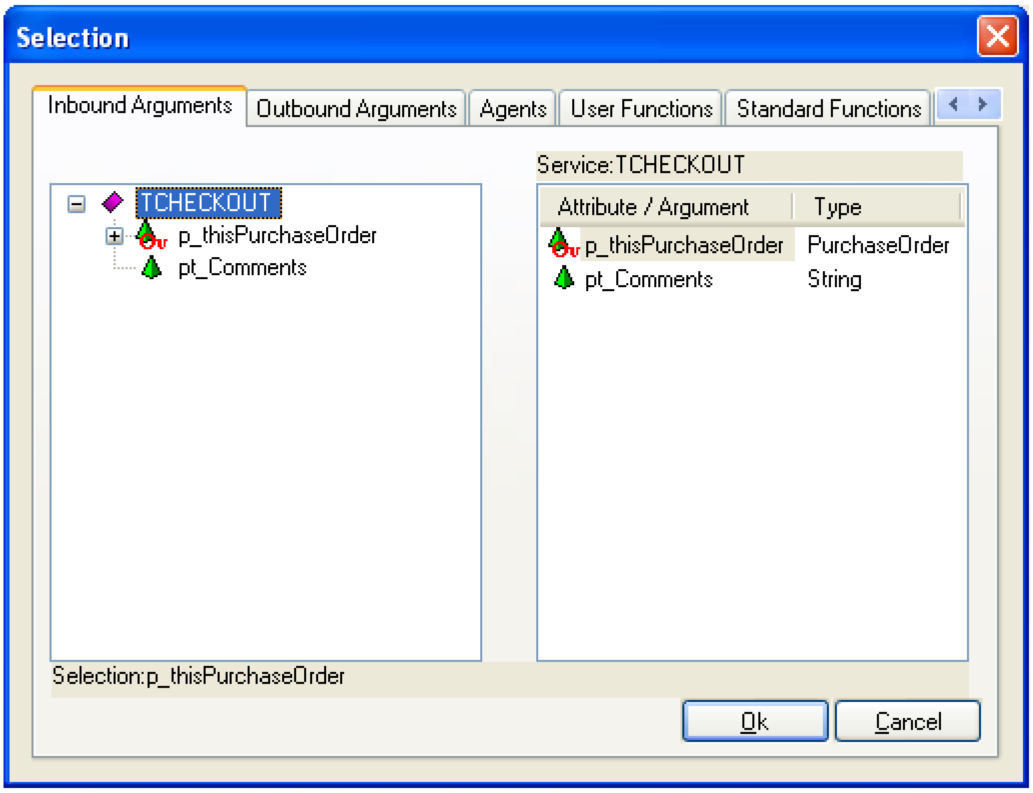
Figure 15 Arguments initialization for delete purchase order service

The *Initialization* dialog is open where we are going to define the value (the effect) that the inbound arguments of this 'TDELETE' service are going to take. This is called *Arguments Initialization.*

The service that allows deleting the current purchase order (shopping cart), which is 'TDELETE' of 'PurchaseOrder' class, only expects one inbound argument. We can check this if we expand the *Inbound Arguments* dropdown list of the dialog above. This inbound argument is an object-valued of 'PurchaseOrder' class (you can see this in the *Services* tab of the class properties).

In this definition, we can add a condition, and of course an effect is the value that will take the inbound argument selected in the dropdown list.

After that brief explanation, let's see how we could initialize that inbound argument. When we have started this conditional navigation process, we were checking out a purchase order (the shopping cart). After checking out that purchase order, we are now able to delete it thanks to the destination Service Interaction Unit we have added in this conditional navigation process, right? And as it is the same object, I mean the same purchase order instance that we prior checked out. We can initialize this inbound argument from the same object that was represented by the inbound argument of the 'TCHECKOUT' service. We can see this in the Figure 15. When pressing the *Help* button in the *Effect* textbox, we are presented, as usual, the wizard options:



Service:TCHECKOUT

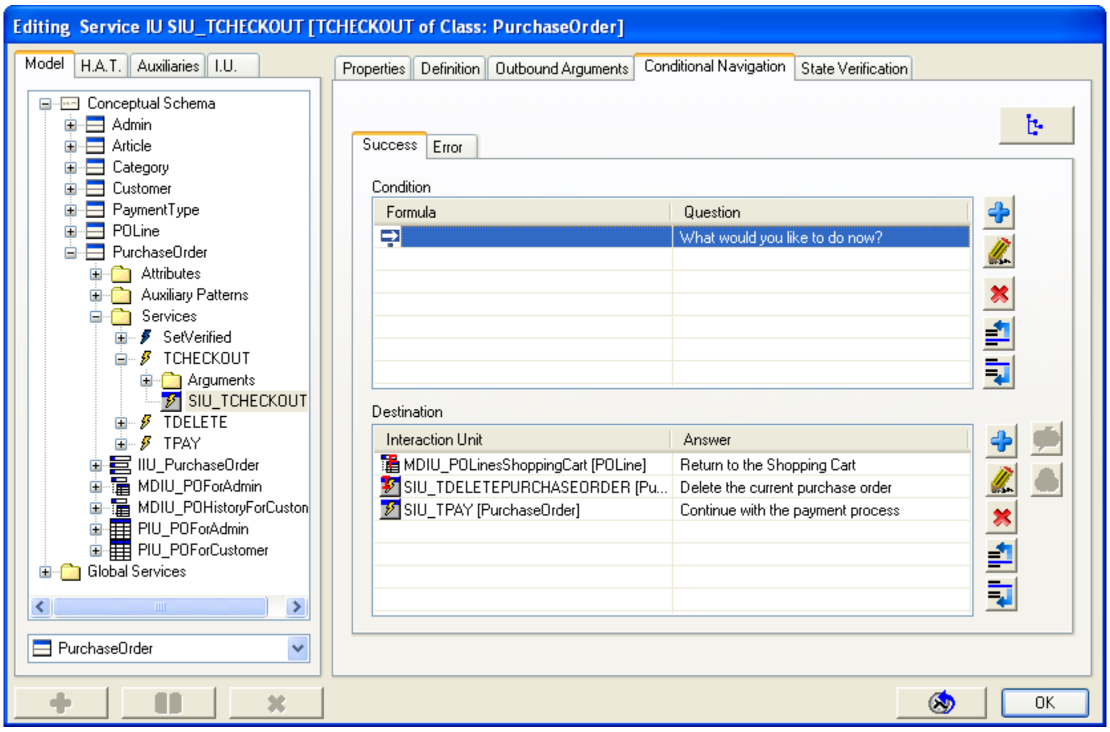
Attribute / Argument	Type
p_thisPurchaseOrder	PurchaseOrder
pt_Comments	String

Selection:p_thisPurchaseOrder

Figure 16 Defining an effect formula for the arguments initialization

As we can see in the Figure 16, we will select, from the *Inbound Arguments* tab, the 'p_thisPurchaseOrder' object valued inbound argument that was used in the 'TCHECKOUT' service.

After pressing the *OK* button, see how an icon with a red "I" appears over the Service Interaction Unit:



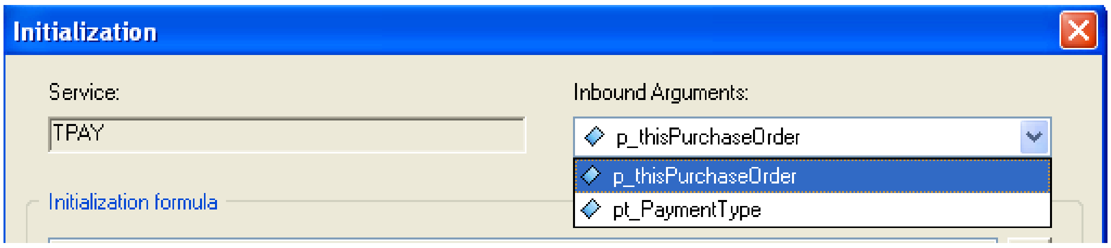
Formula	Question
	What would you like to do now?

Interaction Unit	Answer
MDIU_POLinesShoppingCart [POLine]	Return to the Shopping Cart
SIU_TDELETEPURCHASEORDER [Pu...	Delete the current purchase order
SIU_TPAY [PurchaseOrder]	Continue with the payment process

Figure 17 Arguments initialization icon over SIU_TCHECKOUT

And finally let's do the same, applying arguments initialization over the 'SIU_TPAY' Service Interaction Unit.

If you remember from previous series, the 'TPAY' transaction service of 'PurchaseOrder' class, has two inbound arguments. Both inbound arguments are object-valued arguments. One requests a purchase order, the purchase order that it is going to be paid (current shopping cart) and the second one requests a payment type:



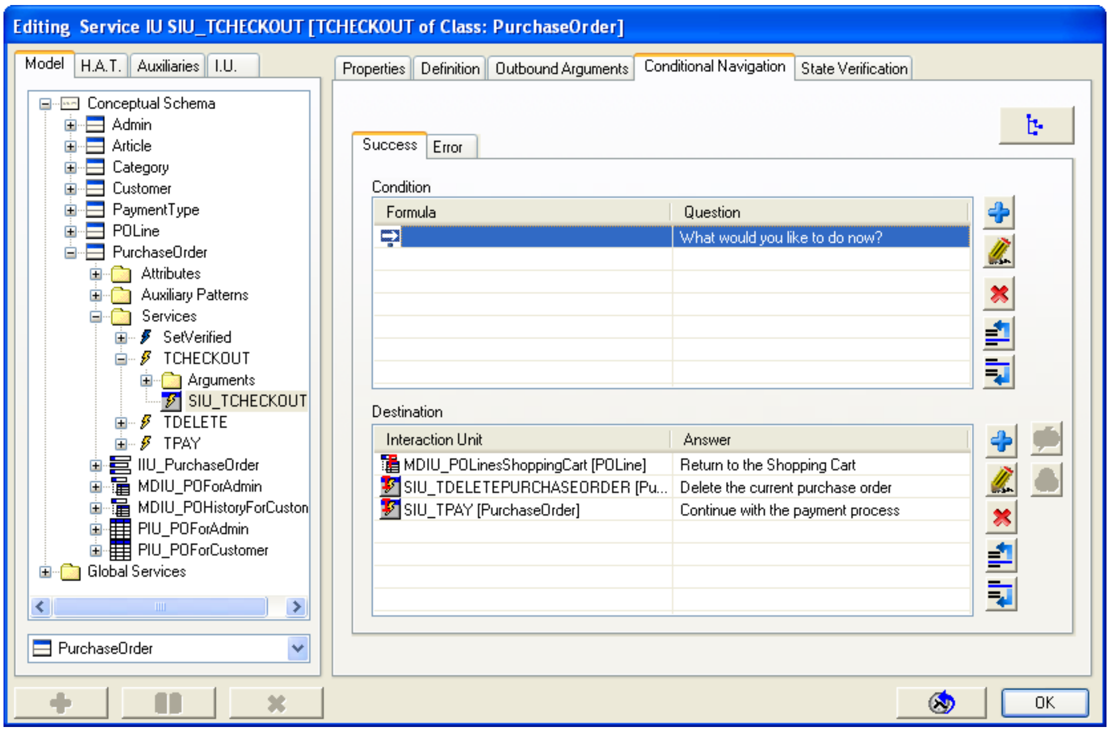
Service: TPAY

Inbound Arguments: p_thisPurchaseOrder
- p_thisPurchaseOrder
- pt_PaymentType

Figure 18 Service TPAY has two inbound arguments

In this case we would like to leave the 'pt_PaymentType' not initialized as we would like each customer to introduce the payment type they want to.

On the other hand, the inbound argument that requests the purchase order, the 'pt_thisPurchaseOrder' argument, is clear because, the same way we learnt in the previous arguments initialization case, it is going to be the same object used in the inbound argument of the check out service. So again, we proceed as in the previous example, selecting the 'p_thisPurchaseOrder' inbound argument of the 'CHECKOUT' service in the *Effect* formula, and eventually press the *OK* button:



Editing Service IU SIU_TCHECKOUT [TCHECKOUT of Class: PurchaseOrder]

Conditional Navigation — Success

Condition

Formula	Question
	What would you like to do now?

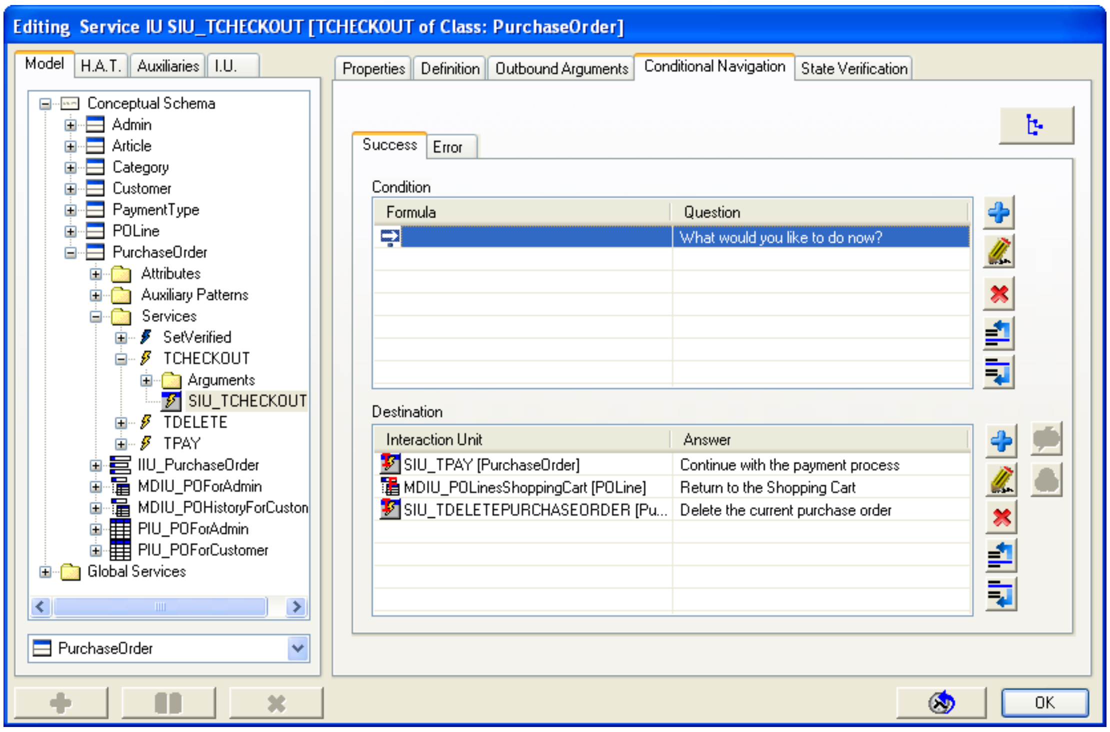
Destination

Interaction Unit	Answer
MDIU_POLinesShoppingCart [POLine]	Return to the Shopping Cart
SIU_TDELETEPURCHASEORDER [Pu...	Delete the current purchase order
SIU_TPAY [PurchaseOrder]	Continue with the payment process

Figure 19 Arguments initialization icon over SIU_TPAY

That way we have the destination interaction units already defined, and also initialize with the information we need.

However, we could go further on this. Perhaps we could think over the order of these destinations interaction units. We could place the "Continue with the payment process" option as the first one of the offered targets. This makes sense as it normally is the most common option to be chosen once checking out a purchase order. So making use of the *Move up* and *Move down* buttons we can define that order. Let's move the payment option to the first one:



Editing Service IU SIU_TCHECKOUT [TCHECKOUT of Class: PurchaseOrder]

Model | H.A.T. | Auxiliaries | I.U.

- Conceptual Schema
  - Admin
  - Article
  - Category
  - Customer
  - PaymentType
  - POLine
  - PurchaseOrder
    - Attributes
    - Auxiliary Patterns
    - Services
      - SetVerified
      - TCHECKOUT
        - Arguments
        - SIU_TCHECKOUT
      - TDELETE
      - TPAY
    - IIU_PurchaseOrder
    - MDIU_POForAdmin
    - MDIU_POHistoryForCuston
    - PIU_POForAdmin
    - PIU_POForCustomer
- Global Services

PurchaseOrder

Properties | Definition | Outbound Arguments | Conditional Navigation | State Verification

Success | Error

Condition

Formula	Question
	What would you like to do now?

Destination

Interaction Unit	Answer
SIU_TPAY [PurchaseOrder]	Continue with the payment process
MDIU_POLinesShoppingCart [POLine]	Return to the Shopping Cart
SIU_TDELETEPURCHASEORDER [Pu...	Delete the current purchase order

OK

Figure 20 Applying order to these destination interaction units

As we have just seen, it is possible to define execution processes in which lead the application user to a sort of execution steps and options through the *Conditional Navigation* feature.