

# RDF Tunnel Direct Invocation

**SCENARIO:  
TRANSFORMATION OF CONCEPTS**

## Scenario Description

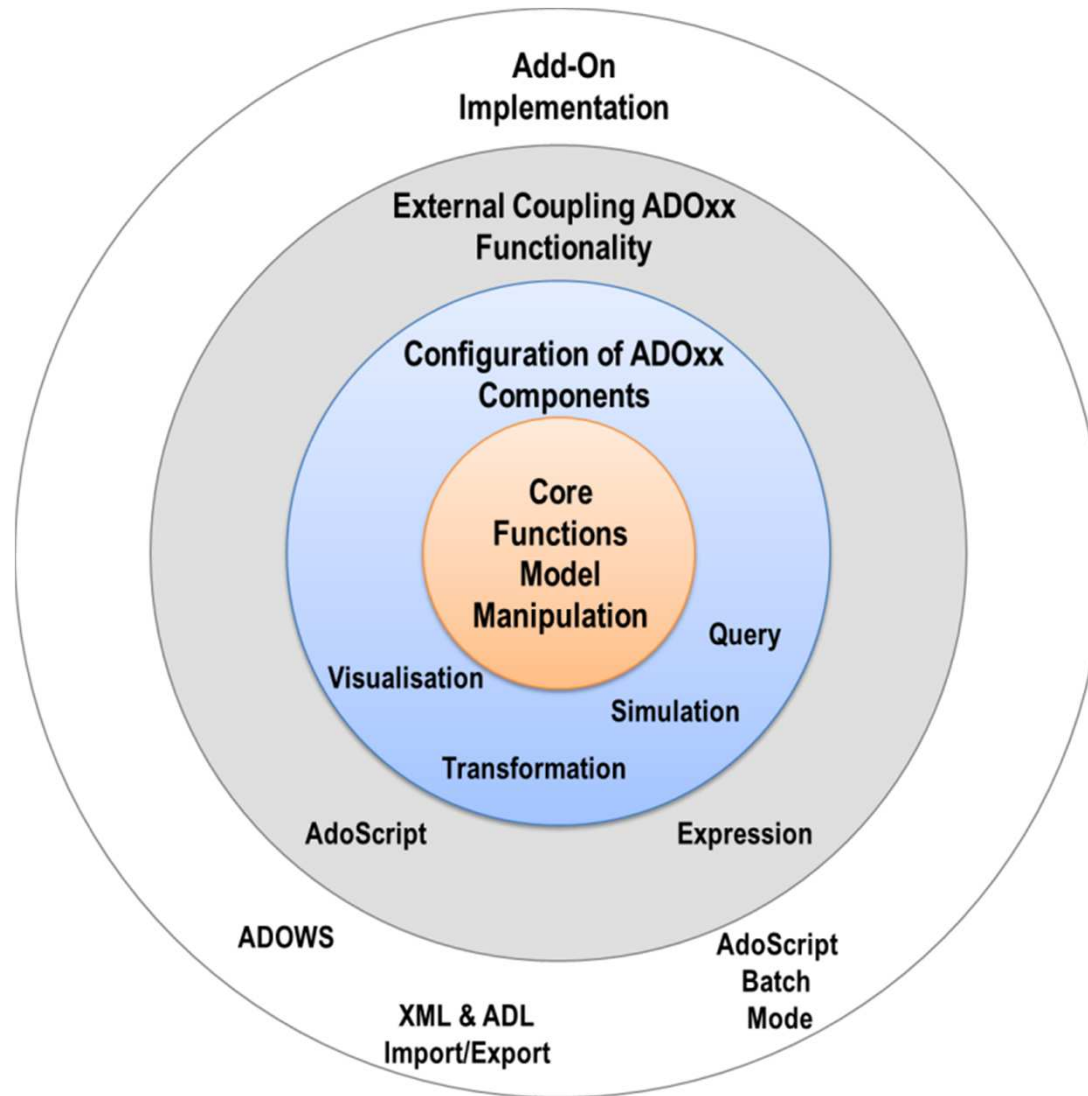
### **Case:**

Transformation of objects from class “Actor”, who is in a certain space, has a certain role and participates to task(s) in a certain process. That actors going to be transformed into objects from class “Agent”. Attribute values of Agent objects are set with information collected from models in source library.

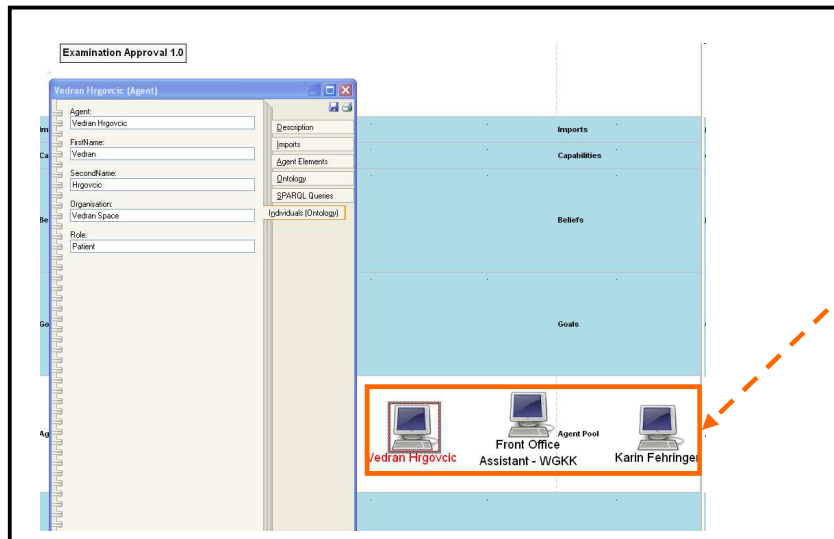
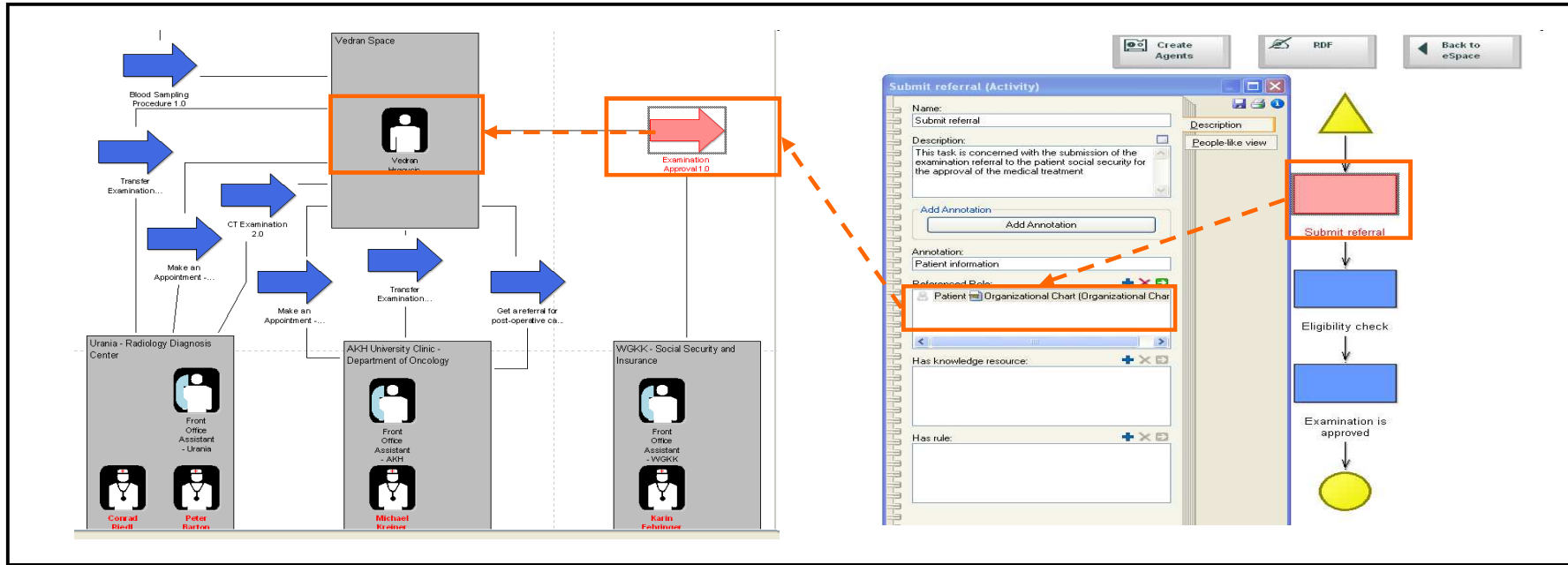
### **GOAL:**

Demonstrate how to transform concepts in one Modelling Language into another modelling language with using ADOWS

# ADOxx Functionality on Meta Level

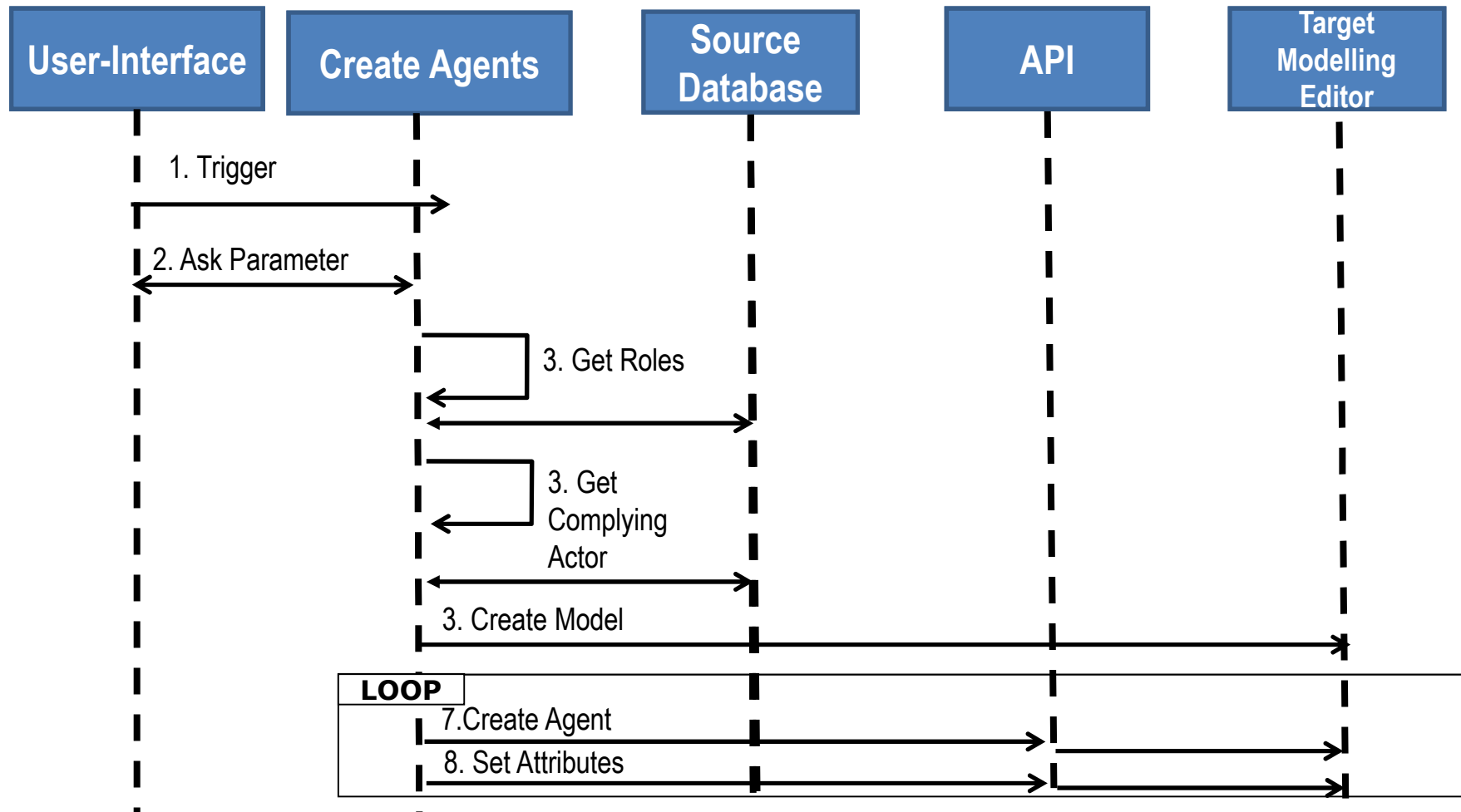


# Description of Algorithm

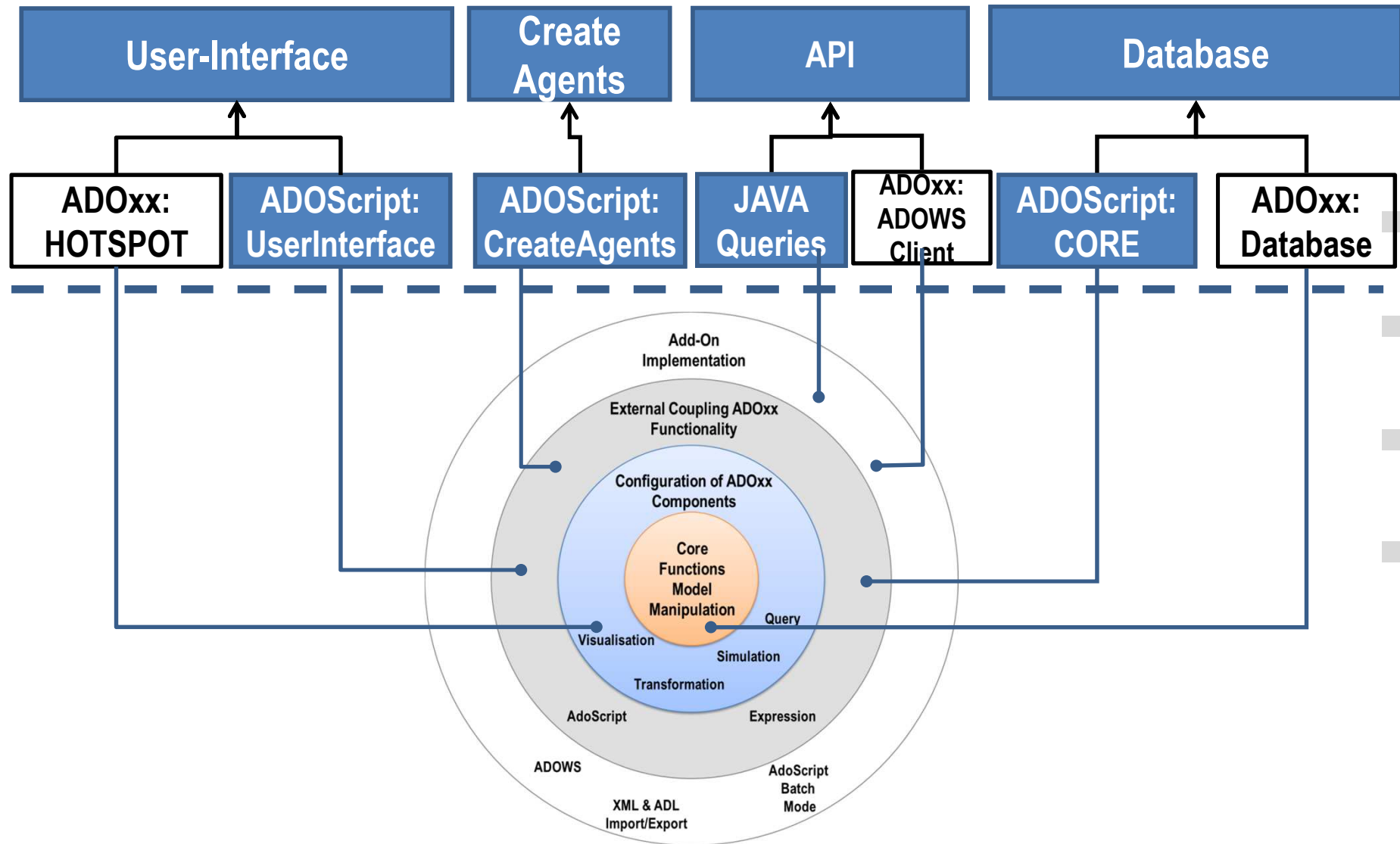


API (java)  
ADOWS

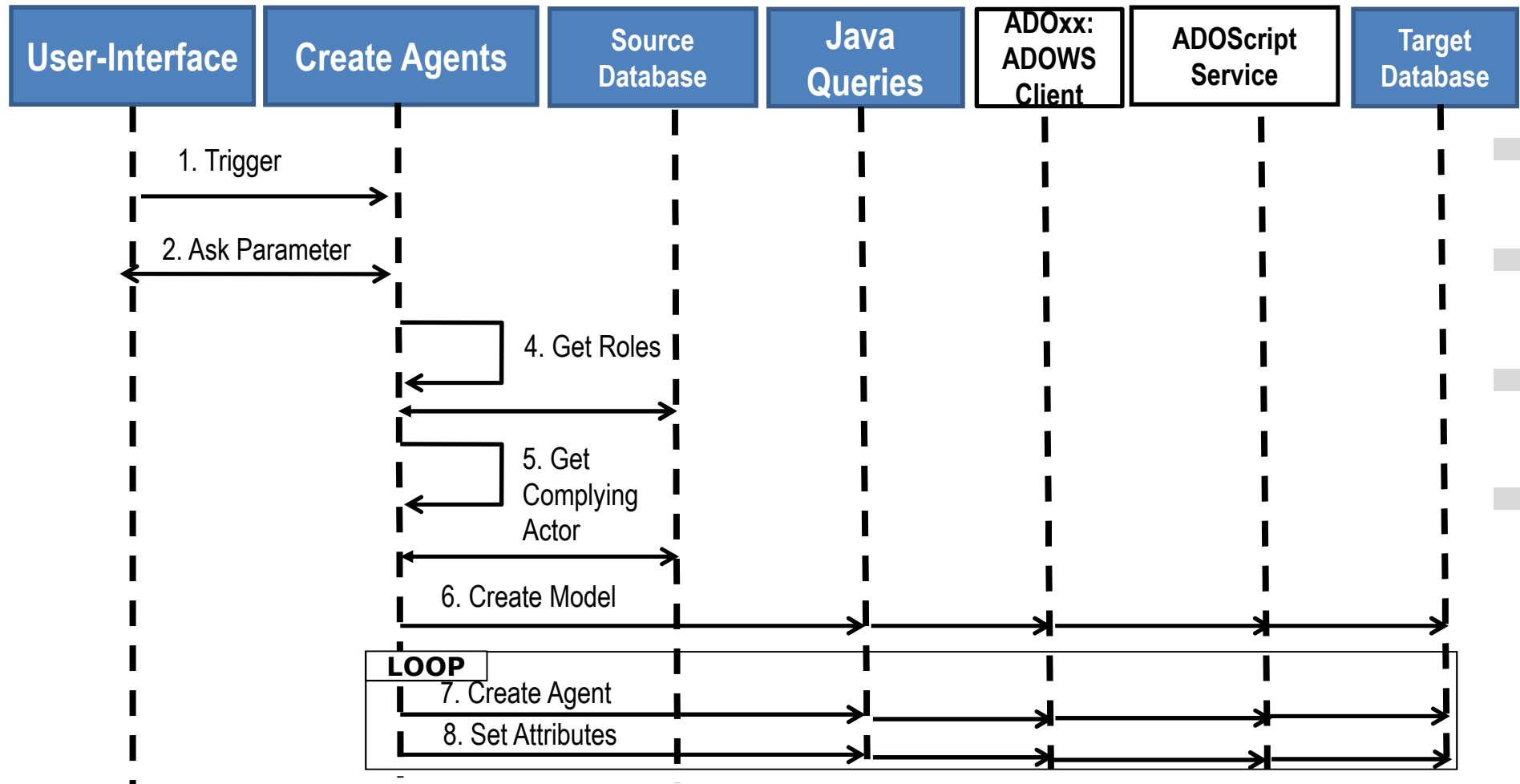
# Description of Algorithm



# Mapping ADOxx Functionality



# ADOxx Realisation Approach



## Added Value of Metamodelling Platform

Used meta-modelling functionality for realisation of the scenario:

- **ADOScript:** ADOScript can retrieve model information, sends request to the API
- **ADOxx Visualisation Component:** is provided by the platform and enables configuration of the user interface of model editor
- **ADOScript Service:** ADOScript Service: ADOScript Service listens a certain port to get and interpret requests
- **ADOWS Client:** is provide by ADOxx.org in order to achieve ADOWS and JAVA Integration



# ADOxx Realisation Hands-On

## 1. Source Modelling Language

1. Model Types “Space Model”, “Process Model”, “Organizational Chart”
2. New class “Activity”, “Process Start”, “Process End”, “Space”, “Role”, “Actor”, “Interaction Process”

## 2. Configure ADOxx

1. Configure Model Graphrep with Hotspots

## 3. Implement API with Java

1. Implement Queries
2. Export Queries with ADOWS Client as runnable JAR file

## 4. Implement Algorithm with ADOscript




1. ADOscript User Interface
2. Retrieve required information from models
3. Invoking API

## 5. Target Modelling Language

1. Model Types “Agent Model”
2. New class “Agent”

# Used ADOxx Functionality: Implementing an Algorithm

Introduction
Setup of Implementation Environment
Modelling Language Implementation
Classes
Relations
Class Attributes and Attributes
GRAPHREP
ATTRREP
CLASS Cardinality
CONVERSION
Model Pointer
Attribute Facets
Model Types

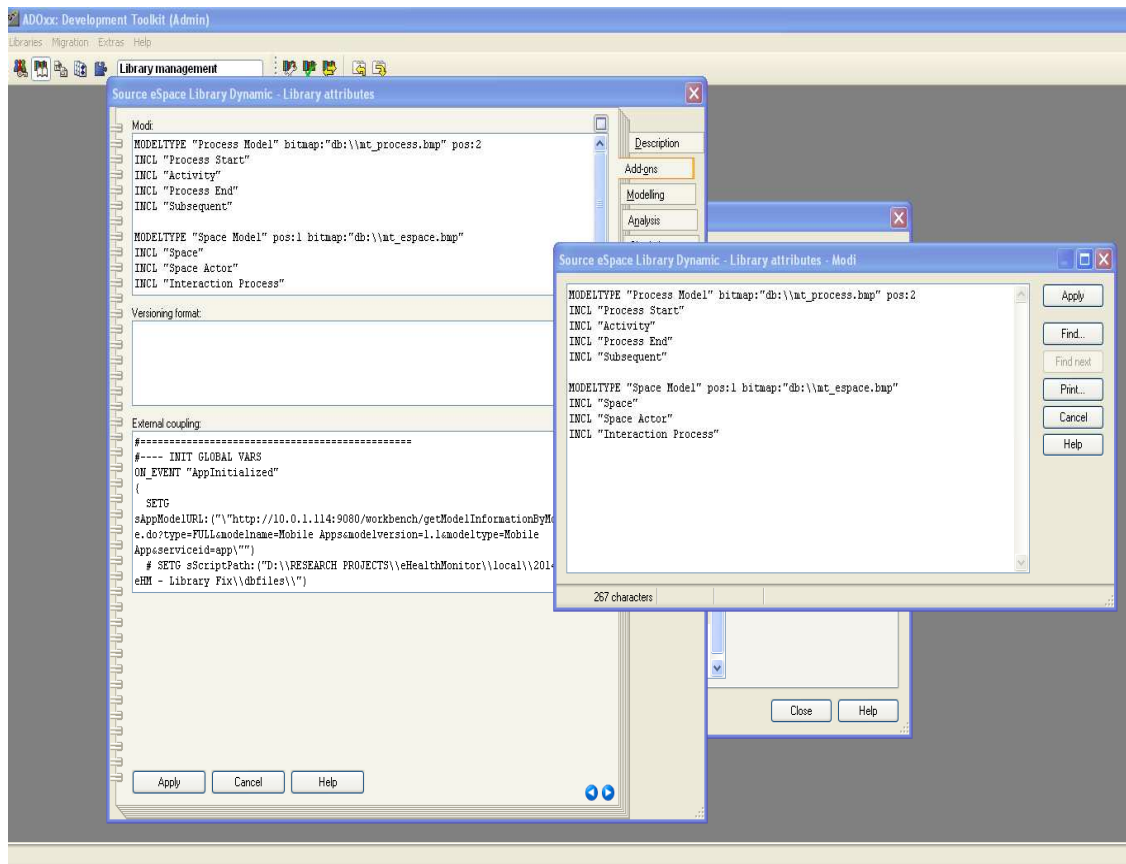
Mechanisms & Algorithms Implementation
<b>Core Functions for Model Manipulation</b>
Database 
<b>Visualisation</b>
Query
Transformation
Configuration of ADOxx Components
Visualisation
Query
<b>External Coupling ADOxx Functionality</b> 
<b>ADOscript Triggers</b>
ADOscript Language Constructs
Visualisation ADOscript
Visualisation Expression
Query ADOscript
Transformation ADOscript 
<b>ADD-ON Implementation</b>
<b>ADOxx Web-Service</b>
XML / ADL Import – Export
ADOscriptBatch Mode

# HANDS-ON

RDF Tunnel Direct Invocation

**SCENARIO:  
TRANSFORMATION OF CONCEPTS**

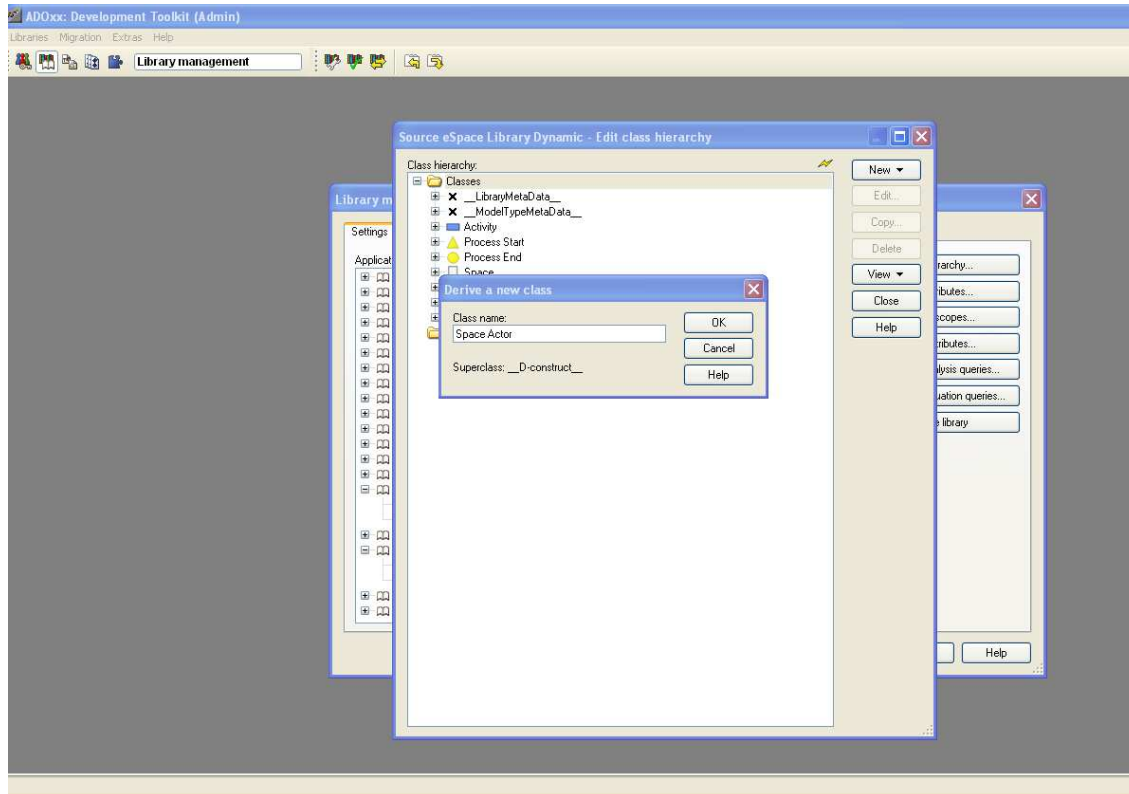
# Define new Modeltypes Space Model”, “Process Model”, “Organizational Chart Model”



## New Modeltypes:

- Select “Source eSpace Library Dynamic” and open Library attributes.
- Got to Add Ons
- Add the Modeltypes “Space Model” and “Process Model” in the Modi attribute
- Select “Source eSpace Library Static” and open Library attributes.
- Got to Add Ons
- Add the Modeltype “Organizational Chart Model” in the Modi attribute
- When the classes are defined, you need to INCLUDE them

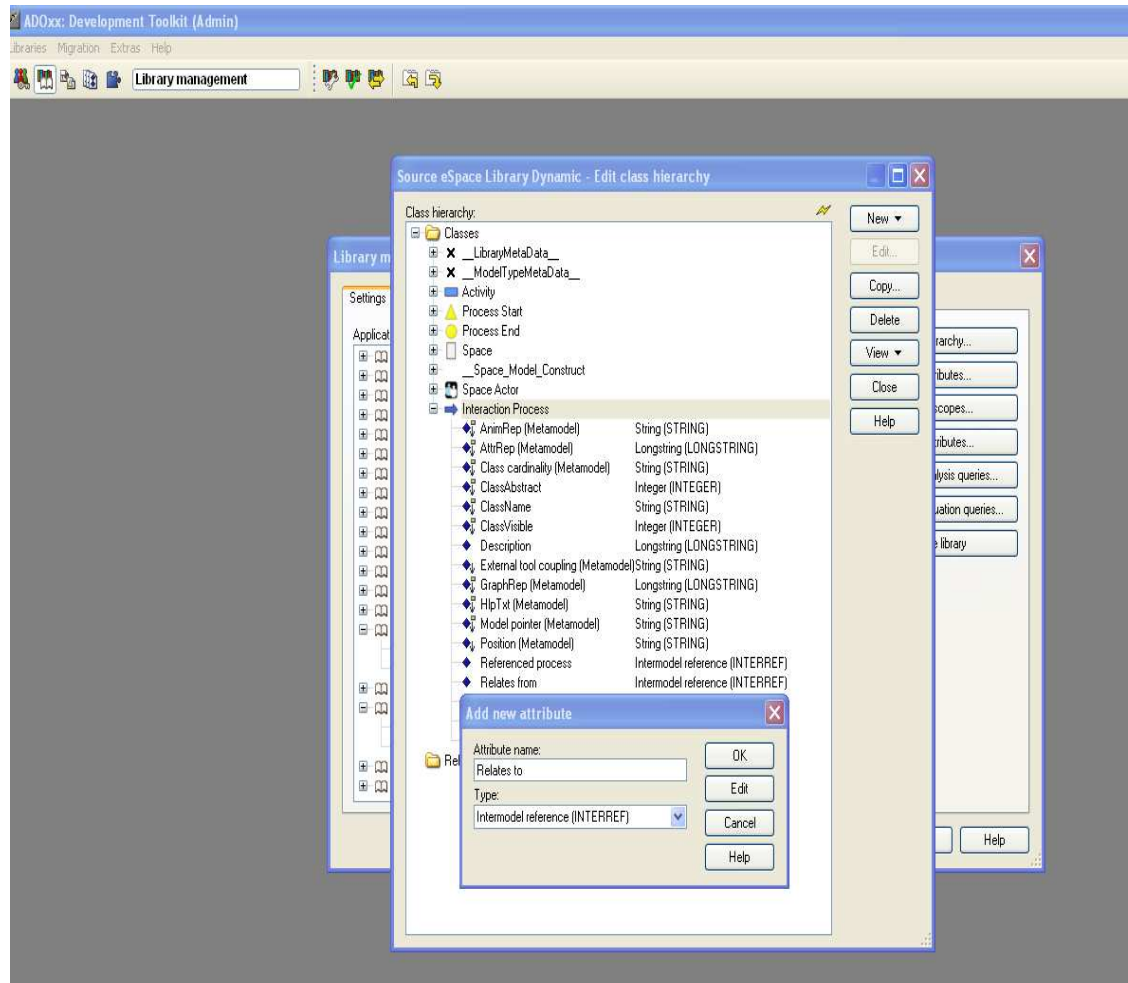
# Create New Classes



## Create New Classes

- Select “Source eSpace Dynamic Library” and open Library attributes.
- Open Class hierarchy, view “Metamodel” and “Class hierarchy” in the View button, select \_\_D-construct\_\_ and click new class.
- Name new classes: “Activity”, “Process Start”, “Process End”, “Space Actor”, “Interaction Process”
- “Activity”, “Process Start”, “Process End”, “Space”, “Space Actor”, “Interaction Process” are now sub-classes of \_\_D-construct\_\_
- Select \_\_D\_aggregation\_\_
- Create class “Space”
- Select “Source eSpace Static Library” and open Library attributes.
- Open Class hierarchy, view “Metamodel” and “Class hierarchy” in the View button, select \_\_S-construct\_\_ and click new class.
- Name new classes: “Role”
- “Roel” are now sub-classes of \_\_D-construct\_\_

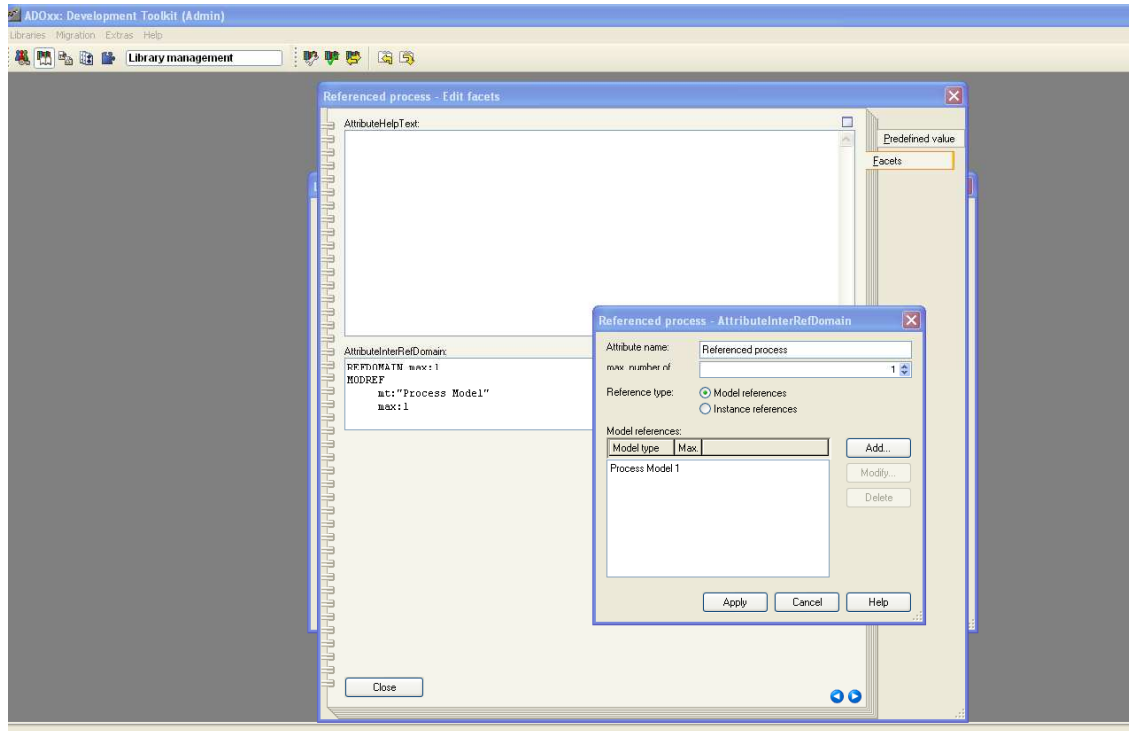
# Add Attributes for Classes



## Add Attributes

- Select “Space Actor” and click Newattribute.
- Make “Referenced Role” as type INTERREF.
- Select “Interaction Process” and click New, attribute.
- Make “Referenced Process” “Relates from” and “Relates to” as type INTERREF.
- Select “Activity” and click New, attribute.
- Make “Referenced Role” as type INTERREF

# Edit INTERREF



## Specification of INTERREF” Relates from” and “Relates to”

- EDIT Facet
- Select AttributeInterrefDomain
- Select “Instance References” and Space Model
- Max number of references is 1
- Select Space
- Max number of references is 1

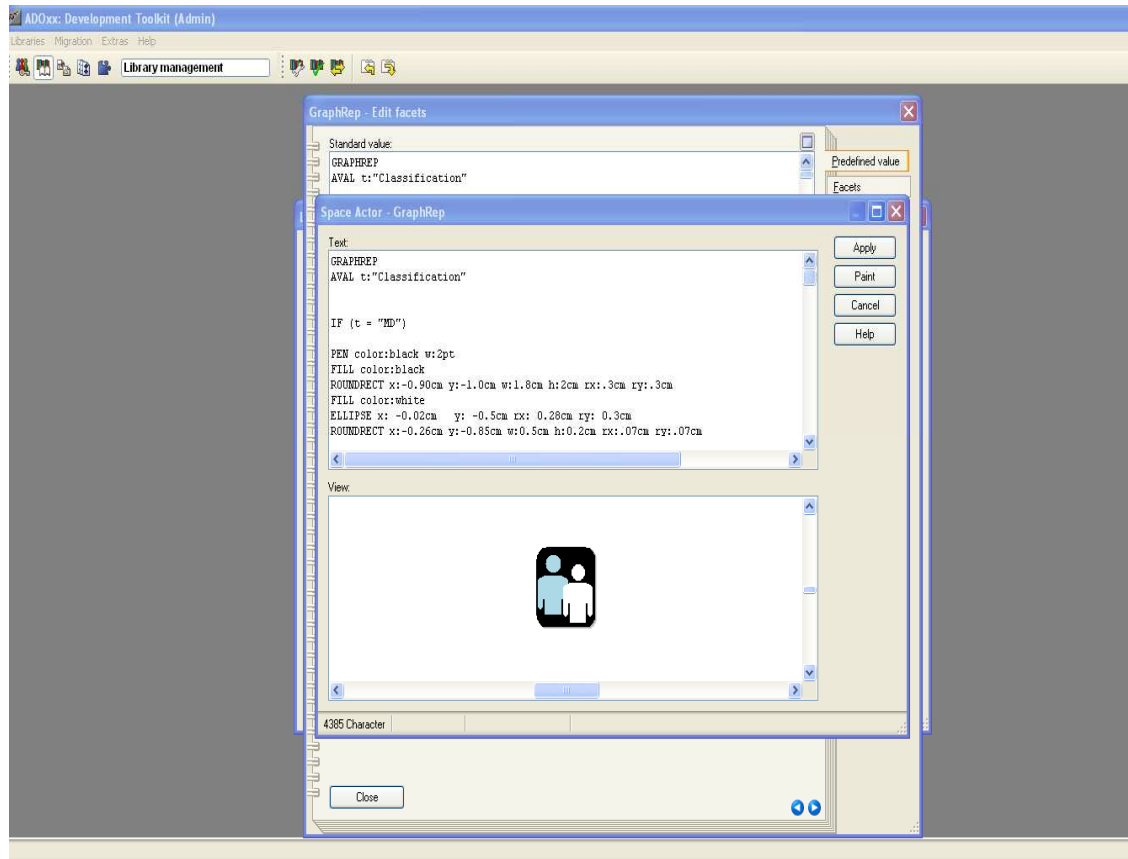
## Specification of INTERREF” Referenced Process”

- EDIT Facet
- Select AttributeInterrefDomain
- Select “Model reference”
- Max number of references is 1
- Select Process Model
- Max number of references is 1

## Specification of INTERREF” Referenced Role”

- EDIT Facet
- Select AttributeInterrefDomain
- Select “Instance References” and Organizational Chart
- Max number of references is 1
- Select Role
- Max number of references is 1

# Add GRAPHREP

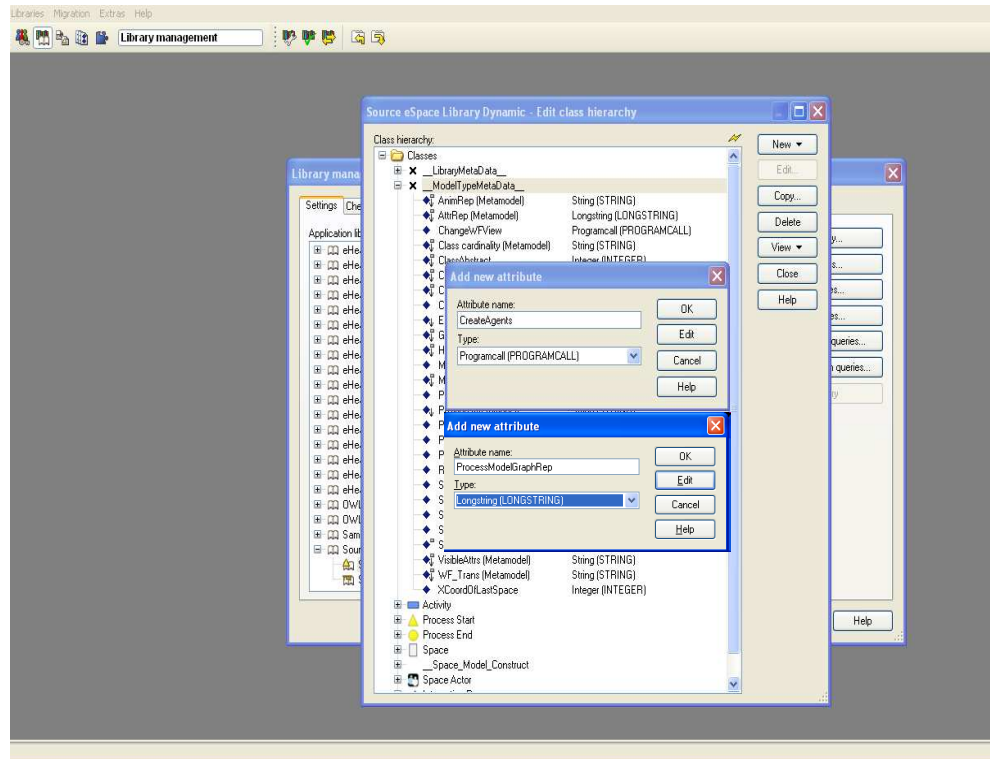


## Specification of GRAPHREP

- Select "Space Actor"
- Click on Attribute "GraphRep"
- Open the GraphRep Editor
- Enter text, paint it and apply.
- Repeat the steps for class "Interaction Process", "Activity" "Process Start", "Process End" and "Role"



# Add Hotspot



## Programcall

ITEM "CreateAgents"

EXECUTE file:("db:\\DirectInvocationScript.asc")

## Add Hotspot

- Select “\_\_ModelTypeMetaData\_\_”
- Add New Attributes “CreateAgents” as type PROGRAMCALL and “ProcessModelGraphRep” as type LONGSTRING
- Copy GraphRep code into standard value of “ProcessModelGraphRep”
- Select “CreateAgents” and enter “CreateAgents” into Standard value, open facets the and copy programcall code into EnumerationDomain

## GraphRep Code

GRAPHREP layer:-1

GRADIENT\_RECT x:6.6cm y:.6cm w:3.2cm h:1.2cm style:diagcross  
color2:lightgray color1:gray color3:lightgray color4:gray

GRADIENT\_RECT x:6.8cm y:.7cm w:2.8cm h:1.0cm style:vert  
color2:lightgray color1:white

PEN w:0.05cm color:darkgray

FILL style:null

RECTANGLE x:6.6cm y:.6cm w:3.2cm h:1.2cm

FONT color:("darkslategray") bold h:9pt

TEXT "Create Agents" x:8.3cm y:.8cm line-break:words w:c:2.0cm

FONT "Wingdings 2" bold h:18pt color:("darkslategray")

TEXT "9" x:7.2cm y:1.1cm w:c h:c

FONT color:("darkslategray")

HOTSPOT "CreateAgents" x:6.6cm y:.6cm w:3.2cm h:1.2cm

# Implement and Import ADOscript File into Database

```
SET sTempAPIResultsFile:("API_results.txt")
CC "Modeling" GET_ACT_MODEL
# --> modelid: intValue.
SET actModelId:(modelid)

CC "Core" GET_CLASS_ID classname:("Interaction Process")
SET n_intproc_classid:(classid)

CC "Core" GET_ATTR_ID classid:(n_intproc_classid) attrname:("Relates from")
SET n_space_relfrom_attrid:(attrid)

CC "Core" GET_ATTR_ID classid:(n_intproc_classid) attrname:("Relates to")
SET n_space_relto_attrid:(attrid)

CC "CoreUI" MODEL_SELECT_BOX modeltype:("Space Model") title:("Select Space Model") boxtext:("Please select a Space Model") oktext:("Select")
SET s_selected_space_modelid:(modelids)
SET s_model_selection:(endbutton)

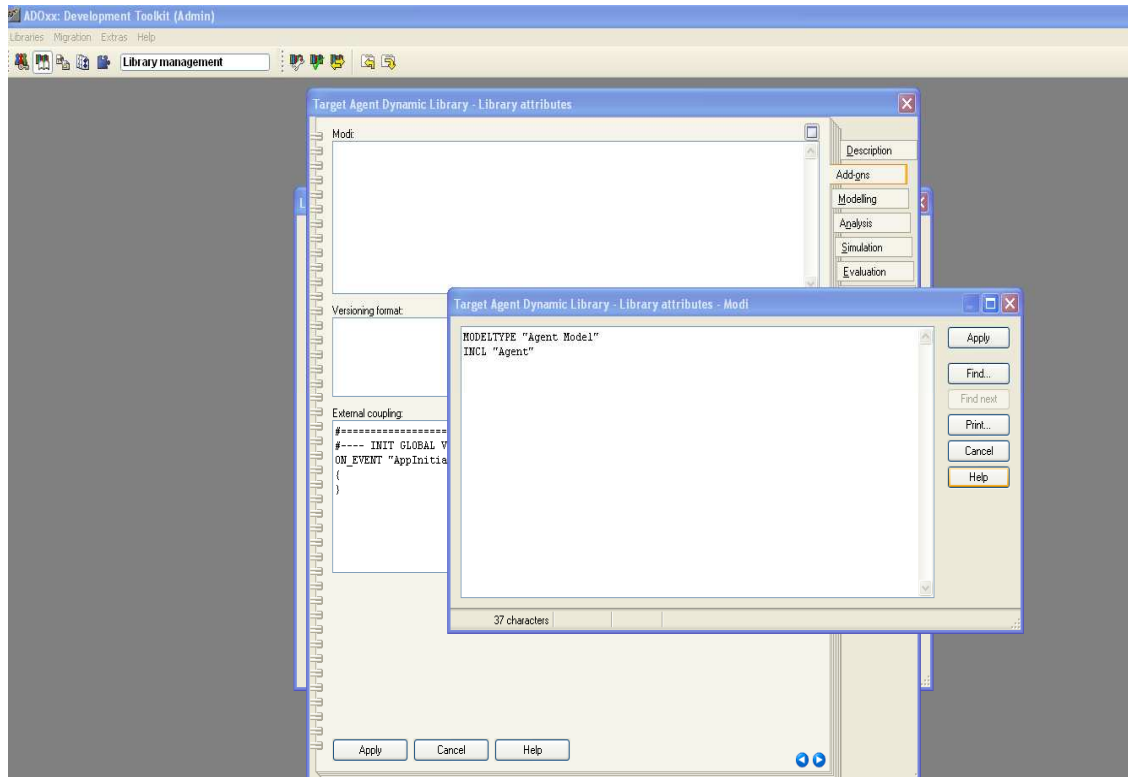
IF (s_model_selection ="ok")
{
    CC "Core" LOAD_MODEL modelid:(VAL s_selected_space_modelid)

    CC "AdoScript" PERCWIN_CREATE title:"Please wait! Agents are being created..."
    #####in MAS#####
    #CC "Core" GET_CLASS_ID classname:("Agent") bp-library
    #--> RESULT ecode: intValue classid: intValue isrel: intValue
    #SET n_agent_classId:(classid)

    SYSTEM ("cmd /c java -jar tools\\DirectInvocationAPI.jar \"getClassIdByName\" \"Agent\"")
    CC "AdoScript" FREAD file: (sTempAPIResultsFile) binary: 0 base64: 0
    SET n_agent_classId:(text)
}

...
```

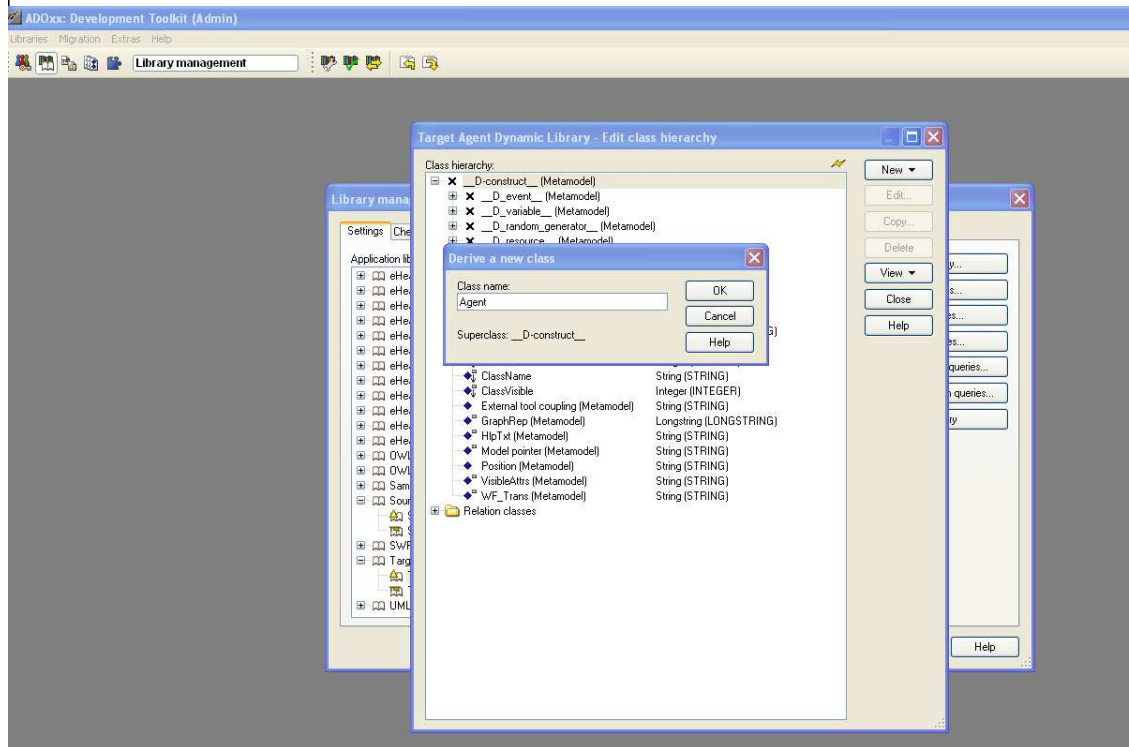
# Define new Modeltype “Agent Model”



## New Modeltypes:

- Select “Target Agent Library Dynamic” and open Library attributes.
- Go to Add Ons
- Add the Modeltype “Agent Model” in the Modi attribute
- When the classes are defined, you need to INCLUDE them

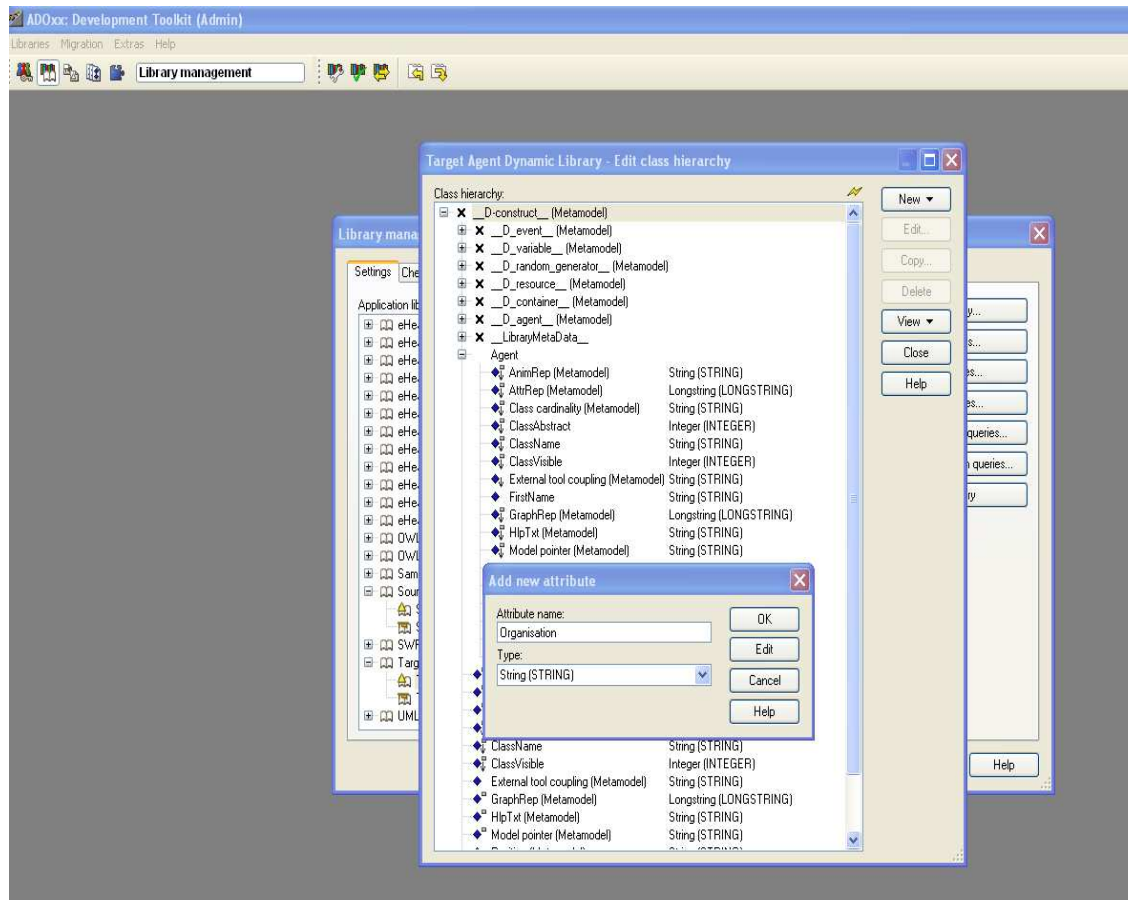
# Create New Classes



## Create New Classes

- Select “Target Agent Dynamic Library” and open Library attributes.
- Open Class hierarchy, view “Metamodel” and “Class hierarchy” in the View button, select `__D-construct__` and click new class.
- Name new class: “Agent”
- Agent is now sub-class of `__D-construct__`

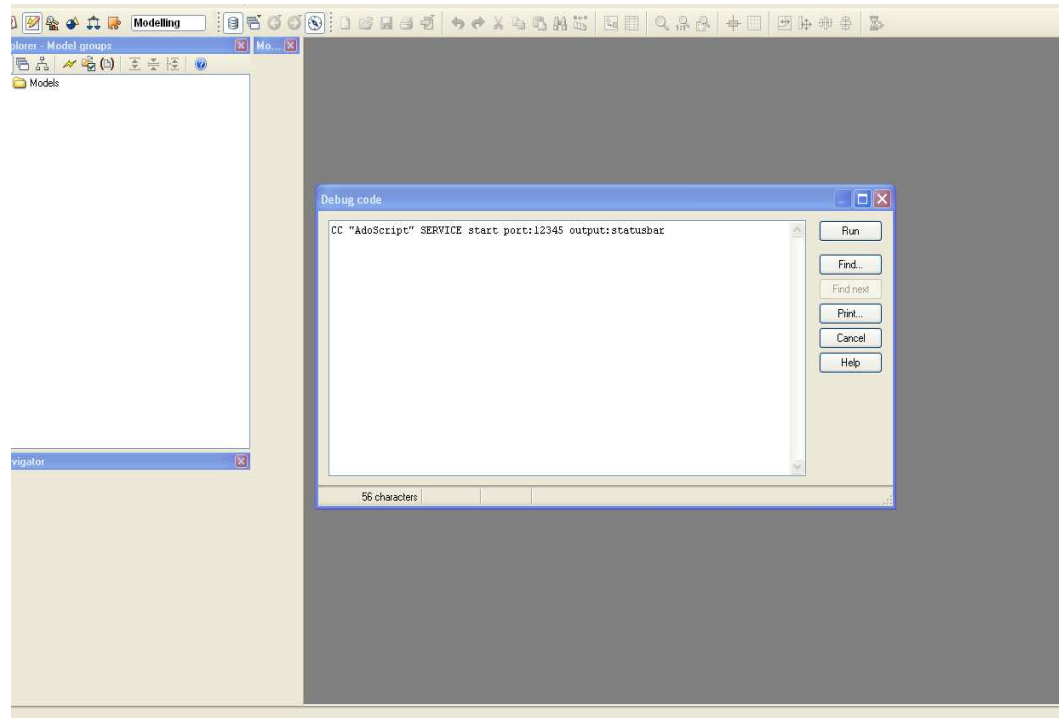
# Add Attributes for Classes



## Add Attributes

- Select “Agent” and click Newattribute.
- Make “FirstName”, “SecondName”, “Role”, “Organisation” as type String.

# Start Adoxx Web Service in Target Library

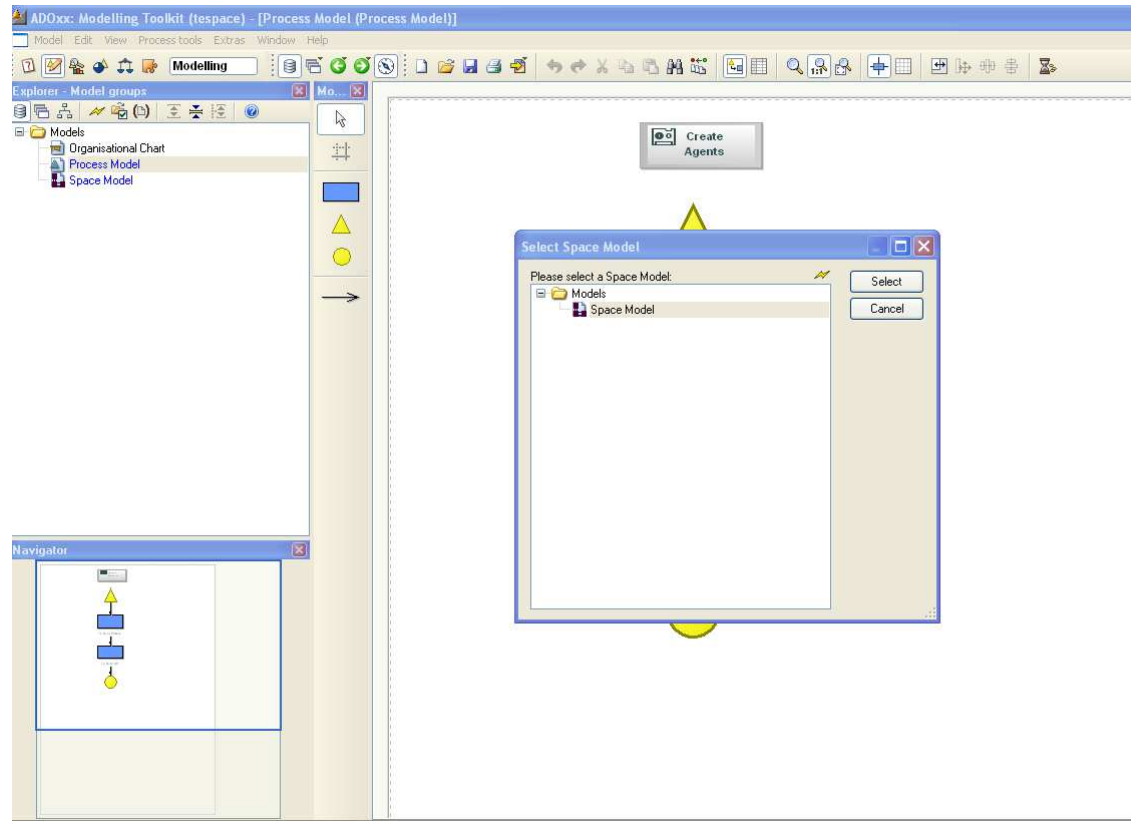


## Start ADOWS

- With using ADOScript Debug Shell enter AdoScript code:

*CC "AdoScript" SERVICE start  
port:12345 output:statusbar*

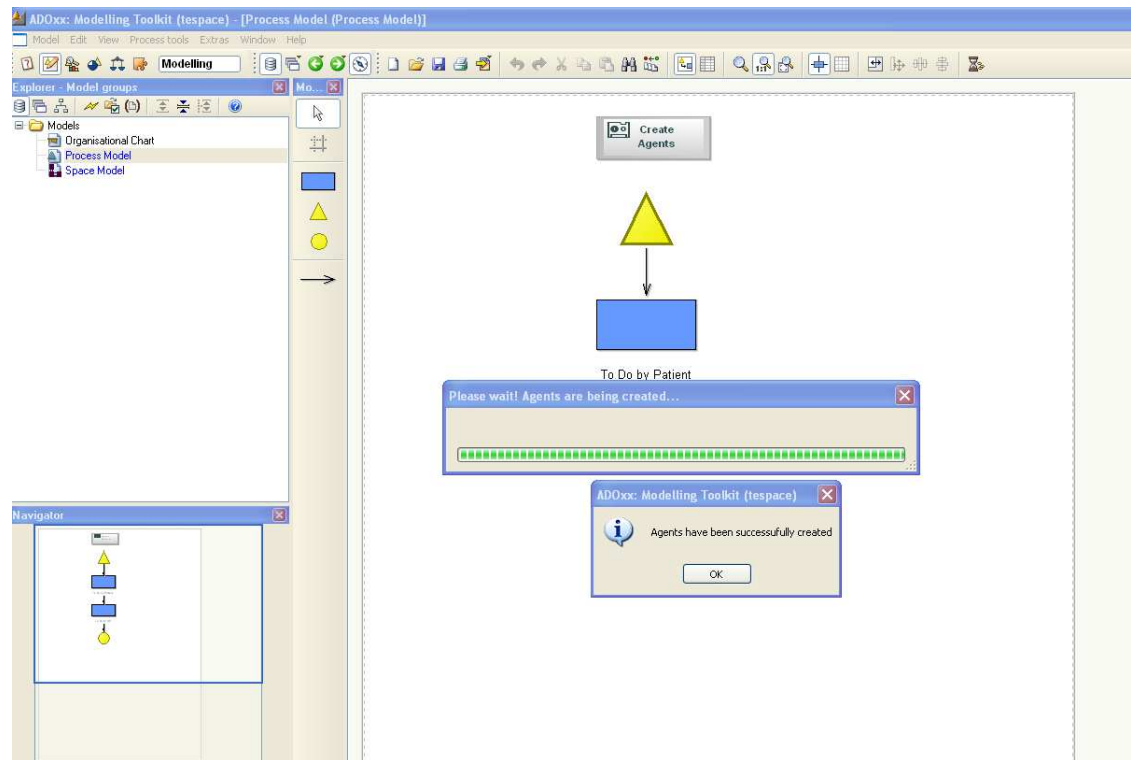
# Results



## Start Creating Agents

- Click “Create Agents” Button
- Select corresponding Space Model

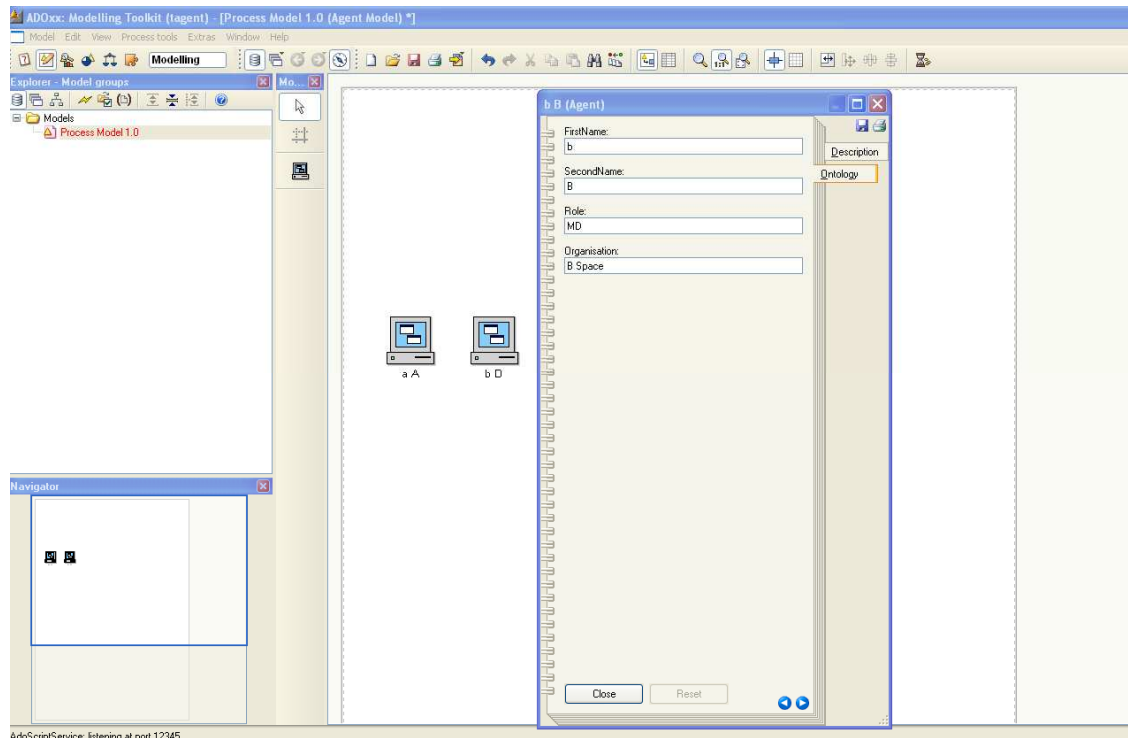
# Results



- AdoScript retrieves required information from three models and invokes API

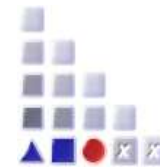


# Results



- Agents are created with corresponding information

# Further Questions?



[www.adoxx.org](http://www.adoxx.org)

[tutorial@adoxx.org](mailto:tutorial@adoxx.org)

