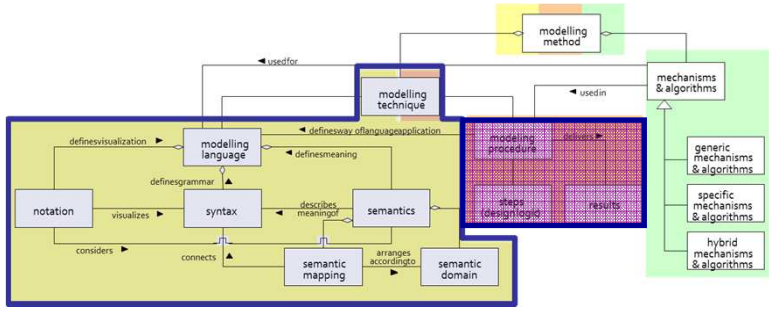


View Switch

SCENARIO:

Realize View Switch utilizing Dynamic GraphRep and AdoScript

Scenario Description

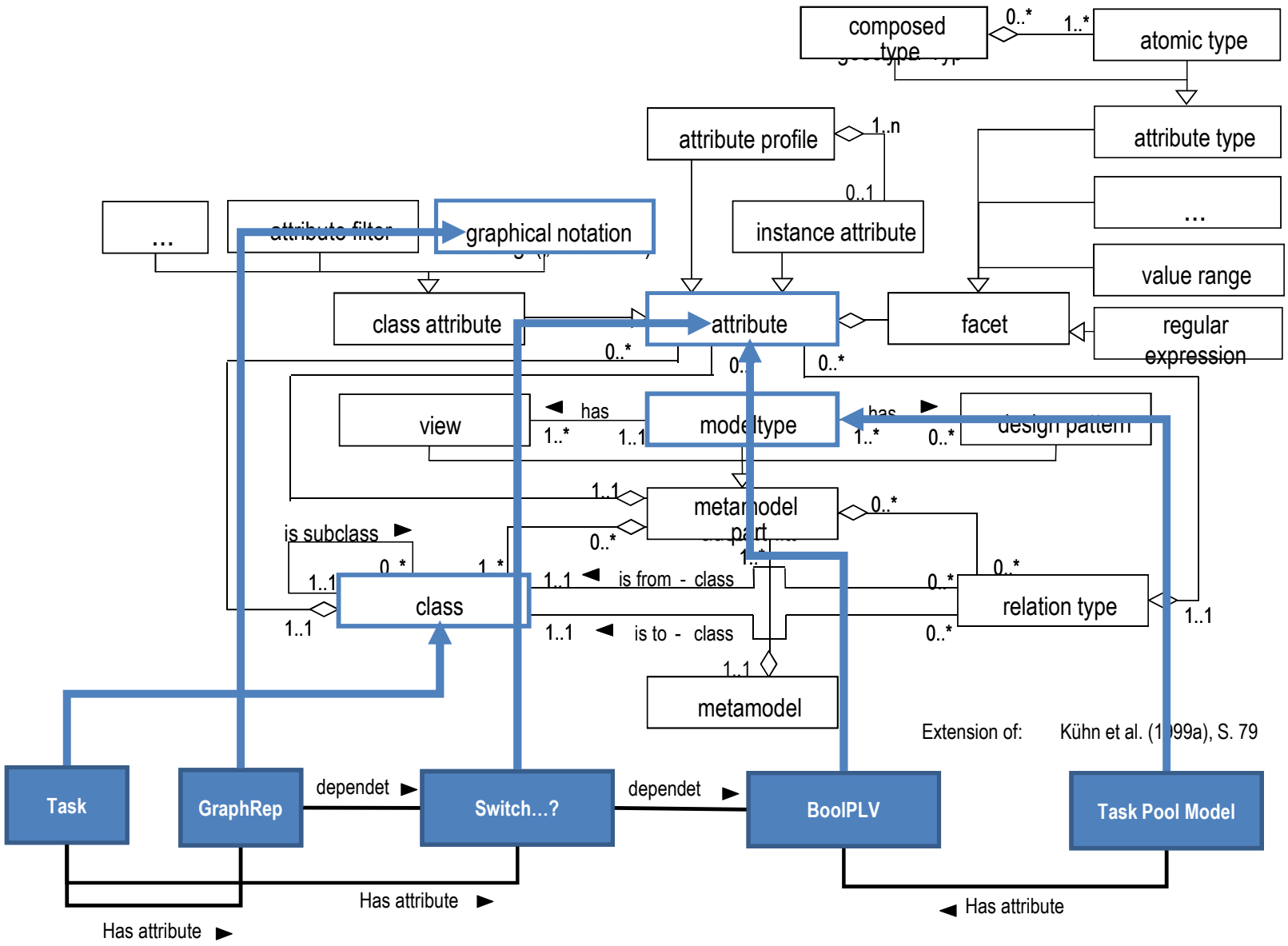
Case:

Realization of view switch between technical view and people-like view. Technical view is the view, which contains technical notations as defined in modelling language and the people-like view is the view which is more detailed and more basic-user friendly. This view switch realized with using dynamic graphical representation and mechanism executes switching.

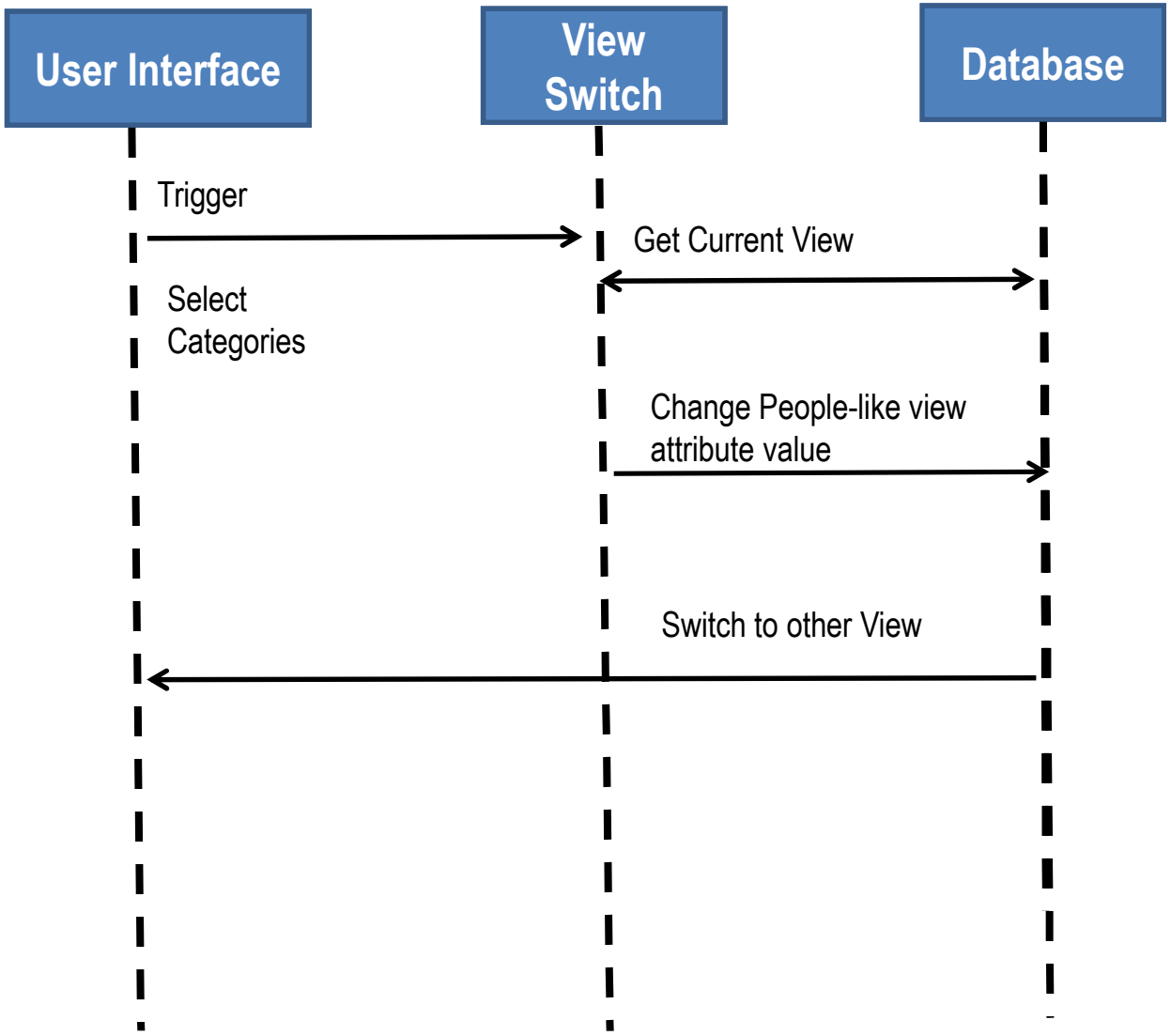
GOAL:

Demonstrate how a switch between views via utilizing dynamic GraphRep and AdoScript.

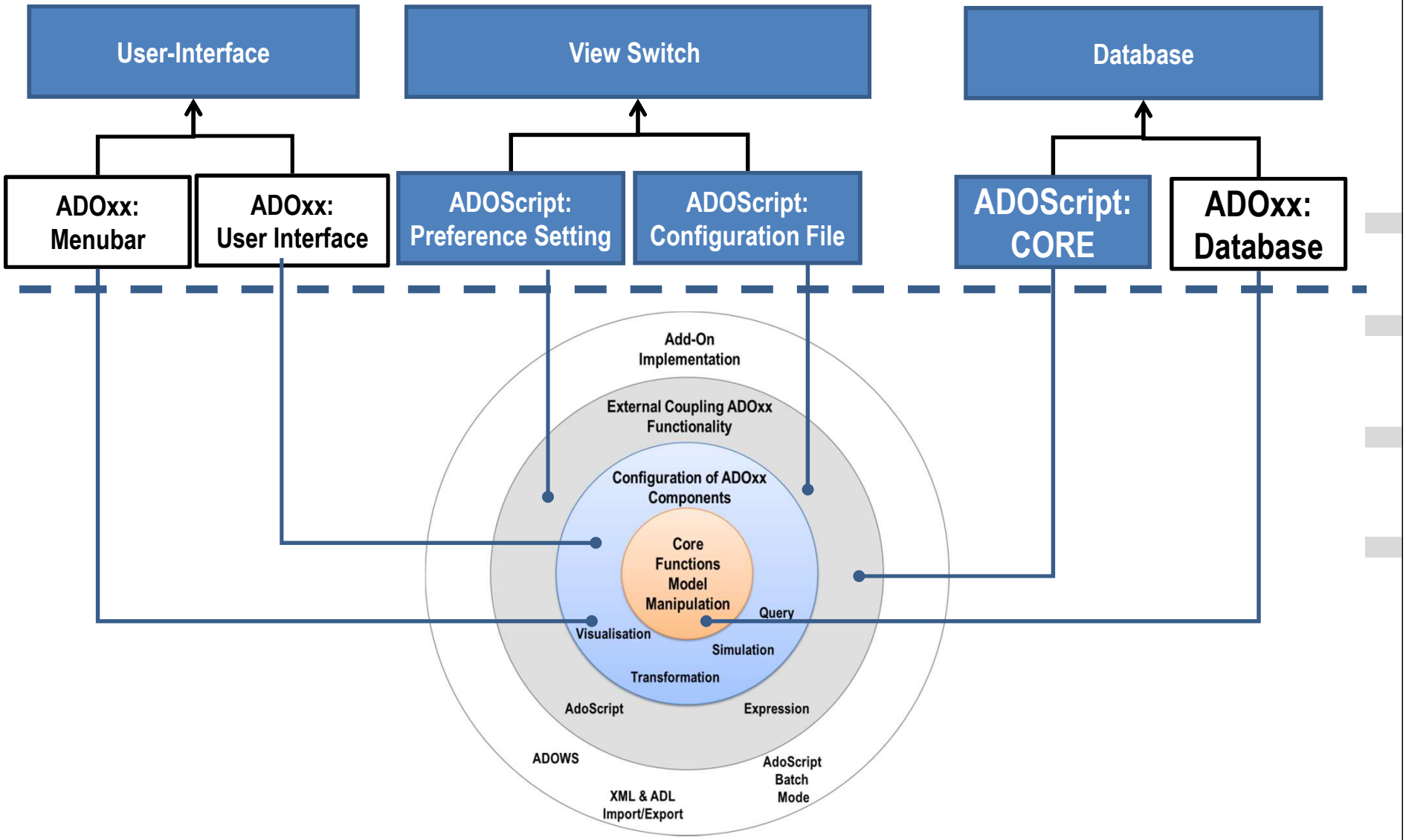
Meta Model of Meta Modelling Language



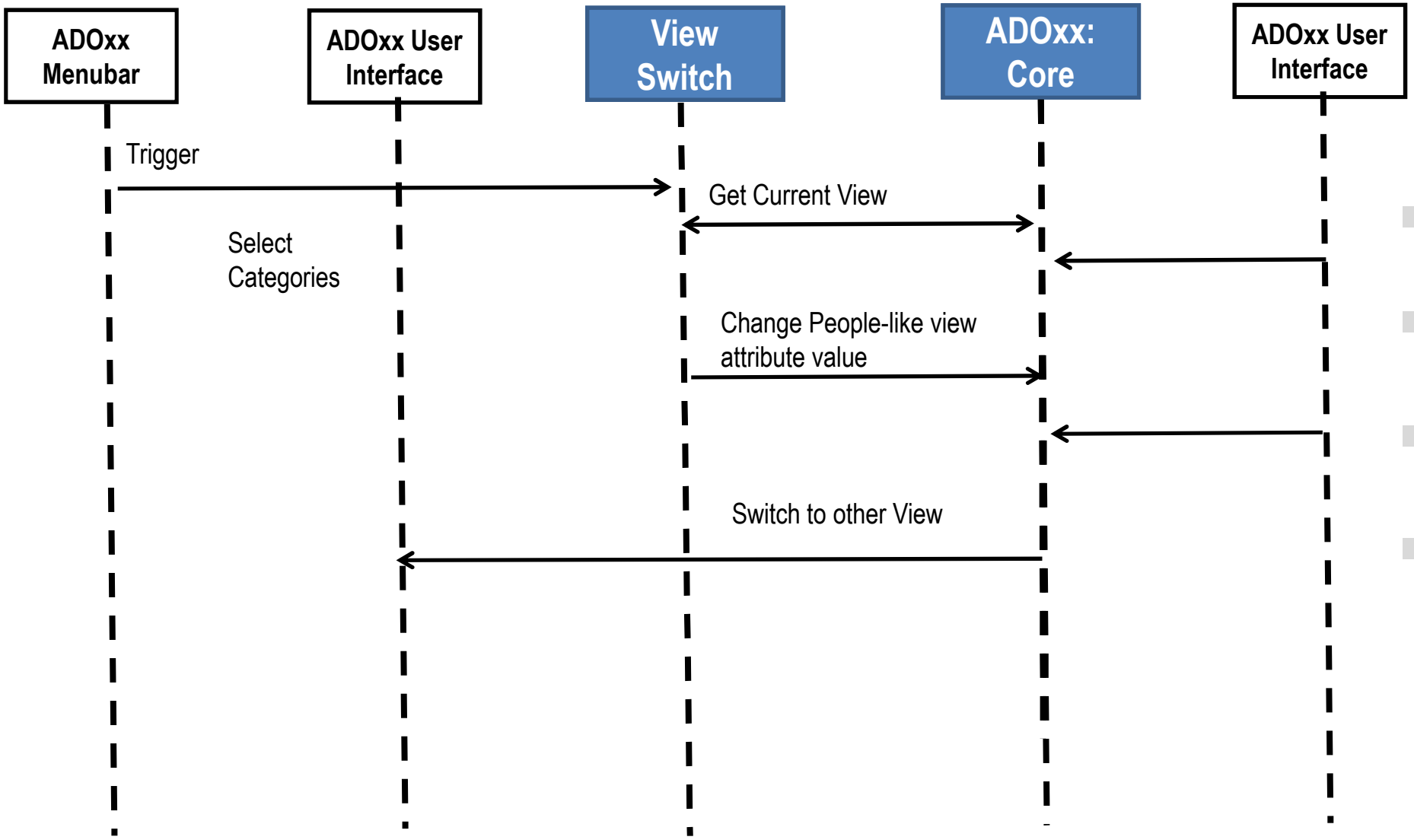
Description of Algorithm



Mapping ADOxx Functionality



Description of Algorithm



Added Value of Metamodelling Platform

Used meta-modelling functionality for realisation of the scenario:

- **ADOScript:** ADOscript can retrieve model information and establish interaction between ADOxx and XSLT Processor.

- **ADOxx Visualisation Component:** is provided by the platform and enables configuration of the user interface of model editor

- **GraphRep:** is a class attribute defined in Meta-meta model of the platform, which enables definition of graphical notation of concepts.

- **AttrRep (NOTEBOOK):** is a class attribute deined in Meta-meta model of the platform, which enables definition of notebooks of concepts

ADOxx Realisation Hands-On

1. Realisation of Modelling Language

1. Define Model Type “Task Pool Model”
2. New class “Task”, “__ModelTypeMetaData__”
3. Add Attributes
4. Implement and Configure GraphRep

2. Implement Algorithm with ADOscript

1. View Switch

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**

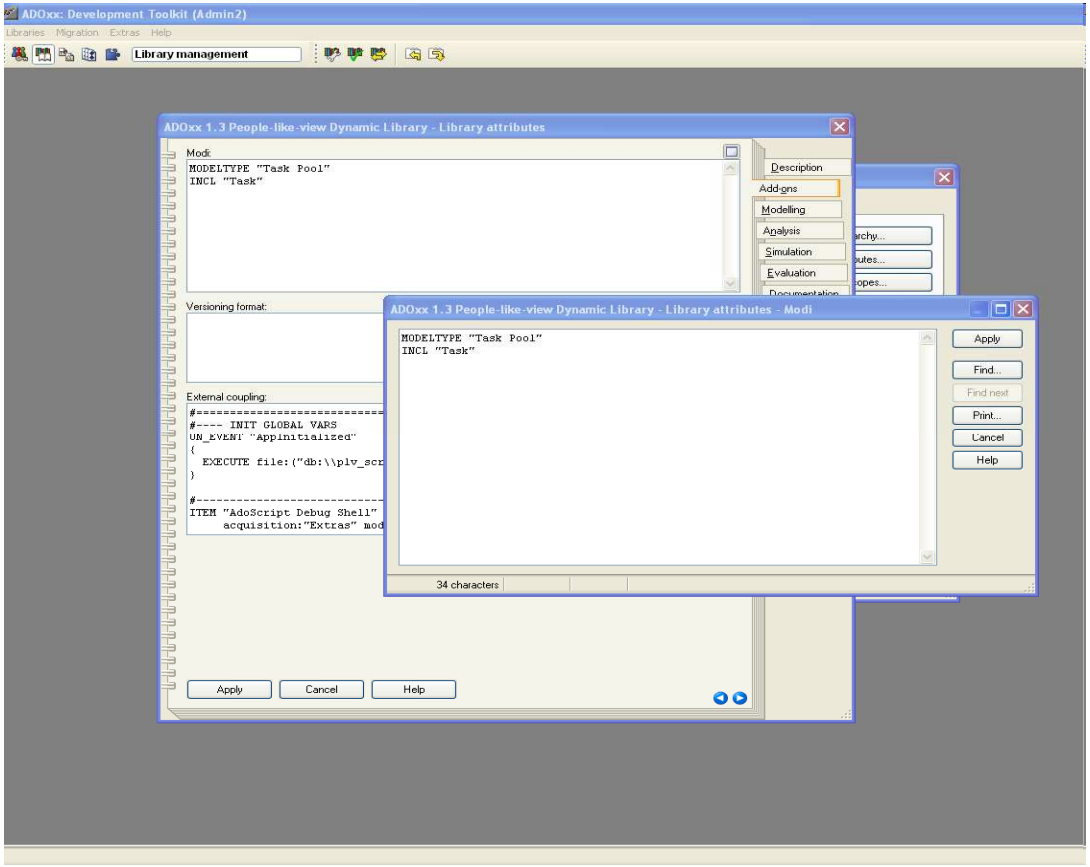
- Mechanisms & Algorithms Implementation
  - Core Functions for Model Manipulation
    - **Database**
    - Visualisation
    - Query
    - Transformation
  - Configuration of ADOxx Components
    - Visualisation
    - Query
  - **External Coupling ADOxx Functionality**
    - **ADOscript Triggers**
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

HANDS-ON

People-like View

SCENARIO:
Realize View Switch utilizing Dynamic GraphRep and AdoScript

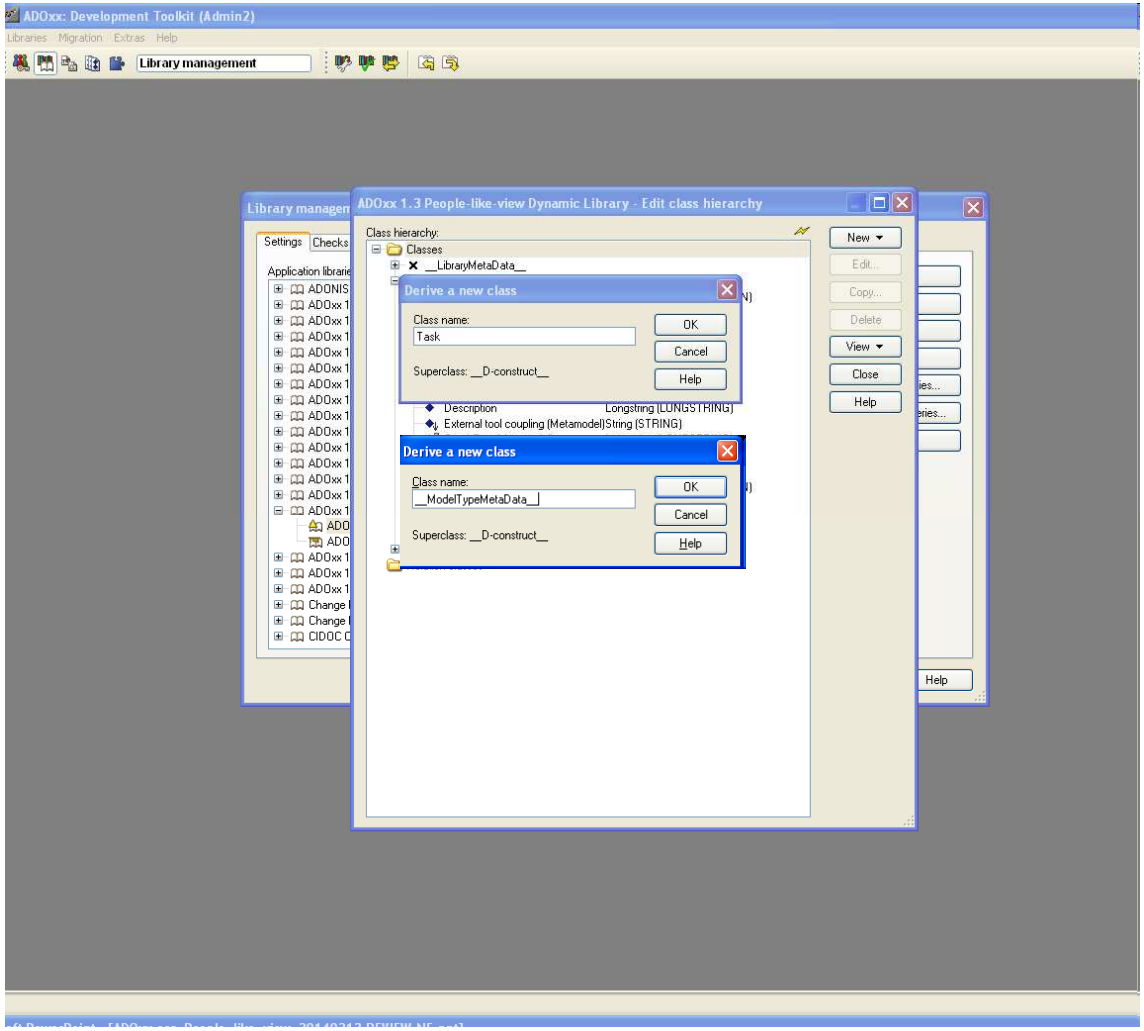
Define Model Types “Task Pool Model”



New Modeltypes:

- Select “People-like view Dynamic Library” and open Library attributes.
- Got to Add Ons
- Add the Modeltype “Task Pool Model” in the Modi attribute
- When the classes are defined, you need to INCLUDE “Task”

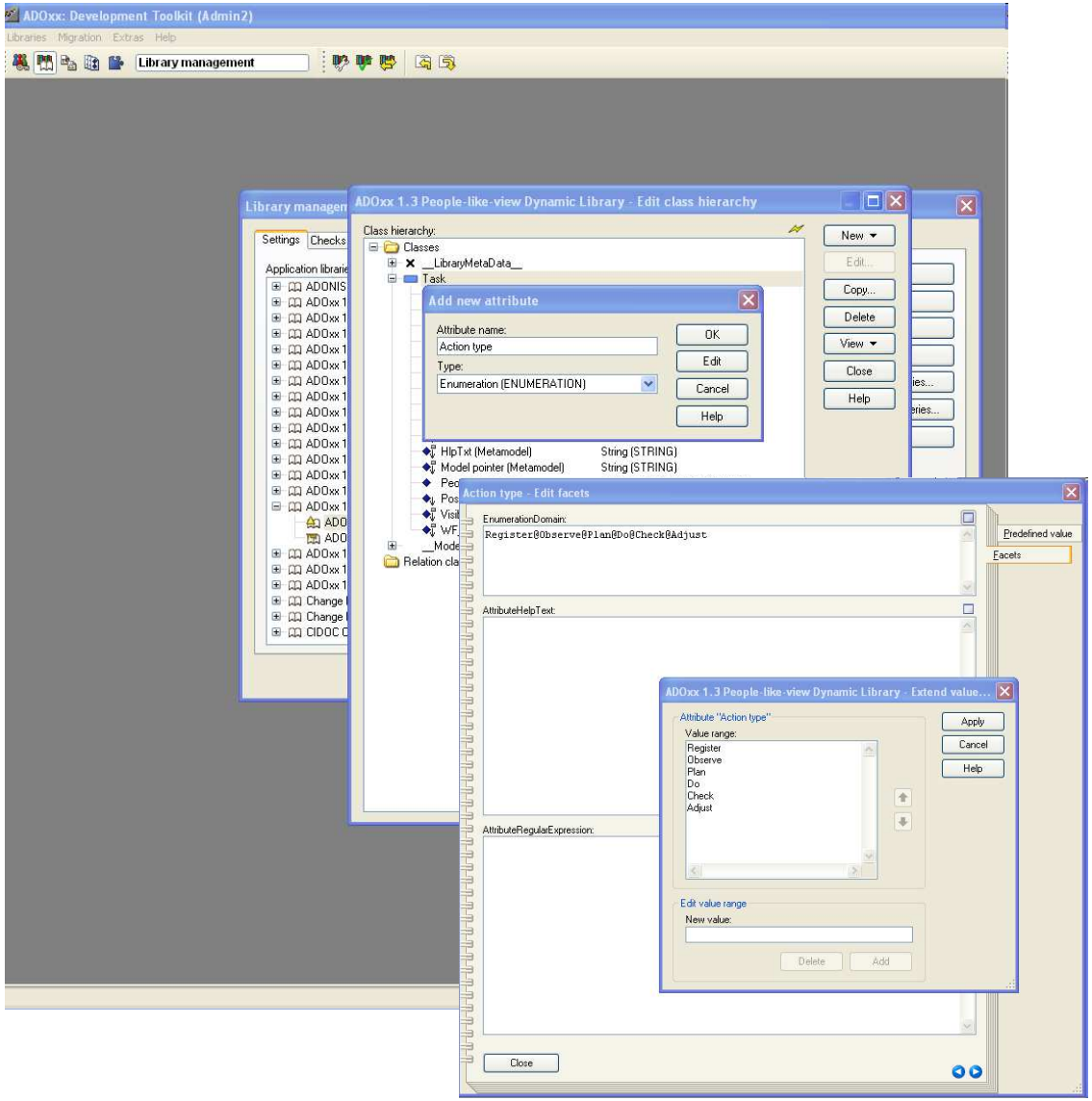
Create New Classes



Create New Classes

- Select “People-like view 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: “Task”, “__ModelTypeMetaData__”,
- They are now sub-classes of __D-construct__

Add Attributes



Add Attributes

- Select “Task” and click Newattribute.
- Add “Action type” and “People-like view” as type ENUMERATION and set EnumerationDomain attributes {Register@Observe@Plan@Do@Check @Adjust} and {No@Yes} respectively.
- Select “__ModelTypeMetaData__” and click New attribute”
- Add “BoolPLV” as type ENUMERATION and set EnumerationDomain attribute {No@Yes}

Implement and Configure GraphRep

```
GRAPHREP
AVAL plv:"People-like view"
FILL r:102 g:153 b:255
AVAL actionType:"Action type"
IF (plv = "No")
RECTANGLE x:-1.4cm y:-.7cm w:2.8cm h:1.4cm
ATTR "Name" y:1.2cm w:c:2.8cm h:t
ELSE
SET filename:("db:\\plan.jpg")
        IF (actionType = "Register")
        {
                SET filename:"db:\\register.jpg"
        }
        IF (actionType = "Observe")
        {
                SET filename:"db:\\observe.jpg"
        }
        IF (actionType = "Plan")
        {
                SET filename:"db:\\plan.jpg"
        }
        IF (actionType = "Do")
        {
                SET filename:"db:\\do.jpg"
        }
        IF (actionType = "Check")
        {
                SET filename:"db:\\check.jpg"
        }
        IF (actionType = "Adjust")
        {
                SET filename:"db:\\adjust.jpg"
        }
```

Define technical view representation of task

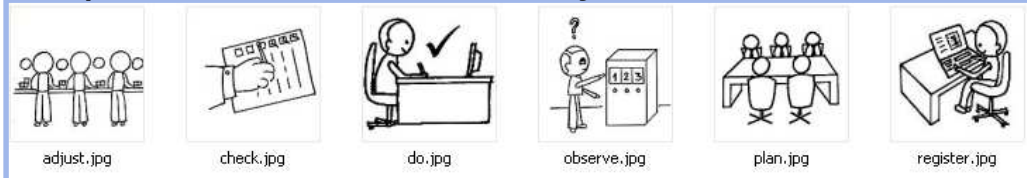
Define People-like view representation of task according to task type

Implement and Configure GraphRep

```
SET we:5.6cm
SET he:2.8cm
PEN w:0.07cm color:black
FILL style:null
SET bigRectangle:(CM (5.6))
RECTANGLE x:((-1)*(we/2)) y:((-1)*(he/2)) w:(bigRectangle) h:(CM 3)  # a border marking the object's size
TABLE  w:(we/2) h:(he/2) rows:1 cols:1 w1:100% h1:100%  # get the current size of the object
BITMAPINFO (filename)  # get the bitmap size
STRETCH off
F (bmpwidth / CMS tabw1 < bmpheight / CMS tabh1) {
# use maximum height, space left and right
SET w:(tabh1 * 2 * (bmpwidth / bmpheight))
BITMAP (filename) x:((-1)*(w)) y:(((-1)*(tabh1)) + (CM 0.05)) w:(w) h:(he)
} ELSE {
# use maximum width, space at top and bottom
SET h:(tabw1  * 2 * (bmpheight / bmpwidth))
BITMAP (filename) x:((-1)*(tabw1 * 2)) y:((-1)*(h)) w:(tabw1 * 2) h:(h)
}
FONT bold
ATTR "Name" x:0.5cm y:((((-1)*(he/2))+0.5cm)) w:l:2.0cm h:b:1.5cm line-break:rigorous
FONT
ATTR "Description" x:0.5cm y:(((-1)*(he/2)+1.5cm)) w:l:2.0cm h:c:2cm line-break:rigorous
ENDIF
```

Define how represent notation and description together

Import Pictures into Library



Implement and Import ADOscripts File into Database

plv_script_globals.asc.asc

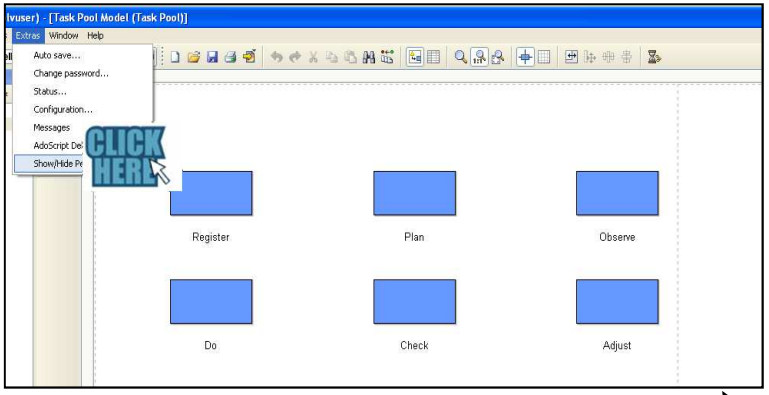
```
SETG c_MOD_TYPE_PROCESS:"Task Pool"

SETG c_CLASS_NAME_1:"Task"

SETG c_ATTR_NAME_SHOW_PEOPLE_LIKE_VIEW: "People-like view"
SETG c_ATTR_NAME_HIDE_SUBSEQUENT_PEOPLE_LIKE_VIEW: "People-like view"

SETG c_str_SHOW_PEOPLE_LIKE_TEXT_SHOW:"People-like view will be shown for the model."
SETG c_str_SHOW_PEOPLE_LIKE_TEXT_HIDE:"People-like view will be hidden for the model."
```

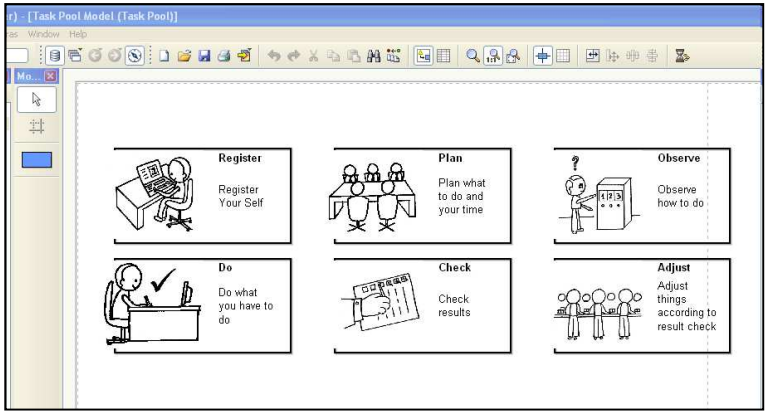
Implement and Import ADOscripts File into Database

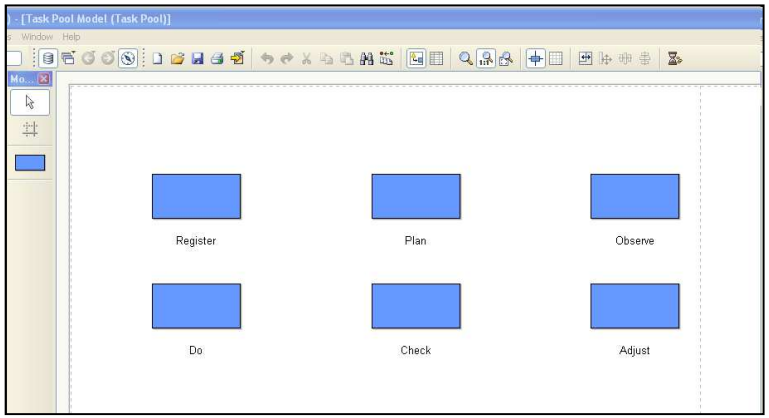
showHidePeopleLikeView.asc (please find whole code in People-oriented View package)

```
CC "Modeling" GET_ACT_MODEL
  #--> RESULT modelid:intValue
SETL id_ActModel:(modelid)

CC "Core" GET_ATTR_VAL objid:(id_ActModel) attrname:("BoolPLV")
SET s_peoplelikeview_attrval:(val)

IF (s_peoplelikeview_attrval = "No")
{
		SET plvOption:("Yes")
		CC "Core" SET_ATTR_VAL objid:(id_ActModel) attrname:("BoolPLV") val:("Yes")
}
ELSE
{
		SET plvOption:("No")
		CC "Core" SET_ATTR_VAL objid:(id_ActModel) attrname:("BoolPLV") val:("No")
}

CC "AQL"  EVAL_AQL_EXPRESSION modelid:(id_ActModel) expr:("<\"" + c_CLASS_NAME_1 + "\">")
SET ocount:(tokcnt (objids, " ")) j:0
WHILE(j < ocount)
{
   SET objid:(VAL token (objids, j, " "))
		CC "Core"  SET_ATTR_VAL objid:(objid) attrname:(c_ATTR_NAME_SHOW_PEOPLE_LIKE_VIEW) val:(plvOption)
   SET j:(j + 1)
}
IF (plvOption = "No")
{
		CC "AdoScript" INFOBOX (c_str_SHOW_PEOPLE_LIKE_TEXT_SHOW)
}
IF (plvOption = "Yes")
{
		CC "AdoScript" INFOBOX (c_str_SHOW_PEOPLE_LIKE_TEXT_HIDE)
}
```

Result


Auto save...
Change password...
Status...
Configuration...
Messages
CLICK HERE
Register
Plan
Observe
Do
Check
Adjust


Register
Register Your Self
Plan
Plan what to do and your time
Observe
Observe how to do
Do
Do what you have to do
Check
Check results
Adjust
Adjust things according to result check


Register
Plan
Observe
Do
Check
Adjust

Further Questions?

www.adoxx.org

tutorial@adoxx.org