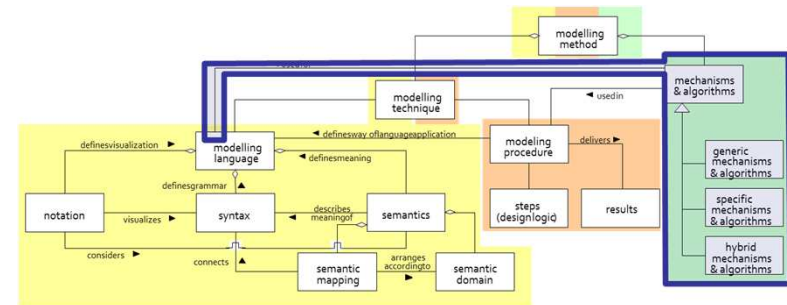




Export of OWL Models as RDF

SCENARIO: Configuration of ADOxx Component

Scenario Description

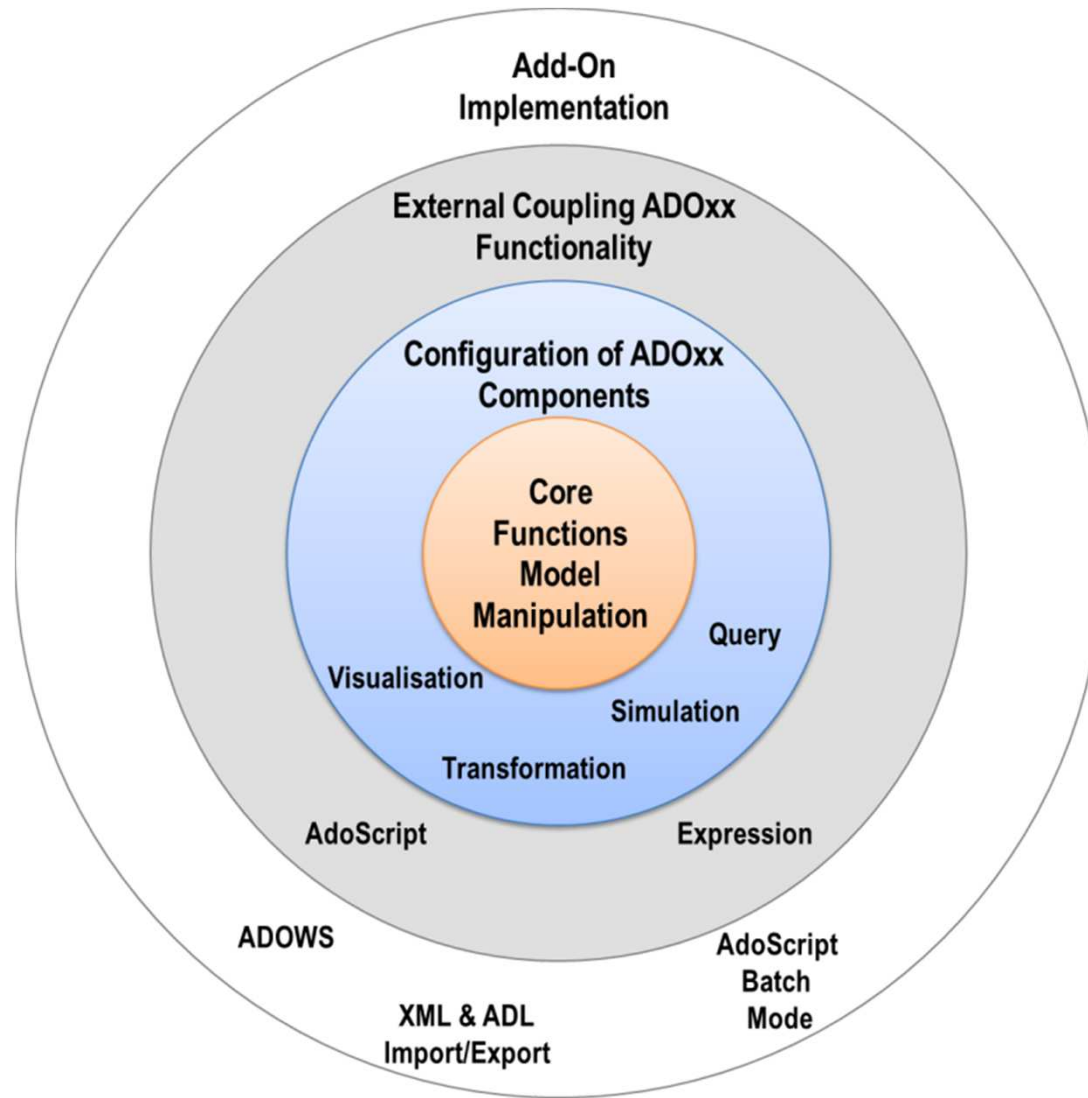
Case:

Export of OWL Model in RDF format via configuration of ADOxx Components. ADOscript invokes ADOxx Import/Export and ADOxx Documentation Components, establishes interaction with a XSLT Processor and XSLT Processor transforms OWL Model XMLs complying ADOxx XML Schema into RDF Files complying RDF Schema.

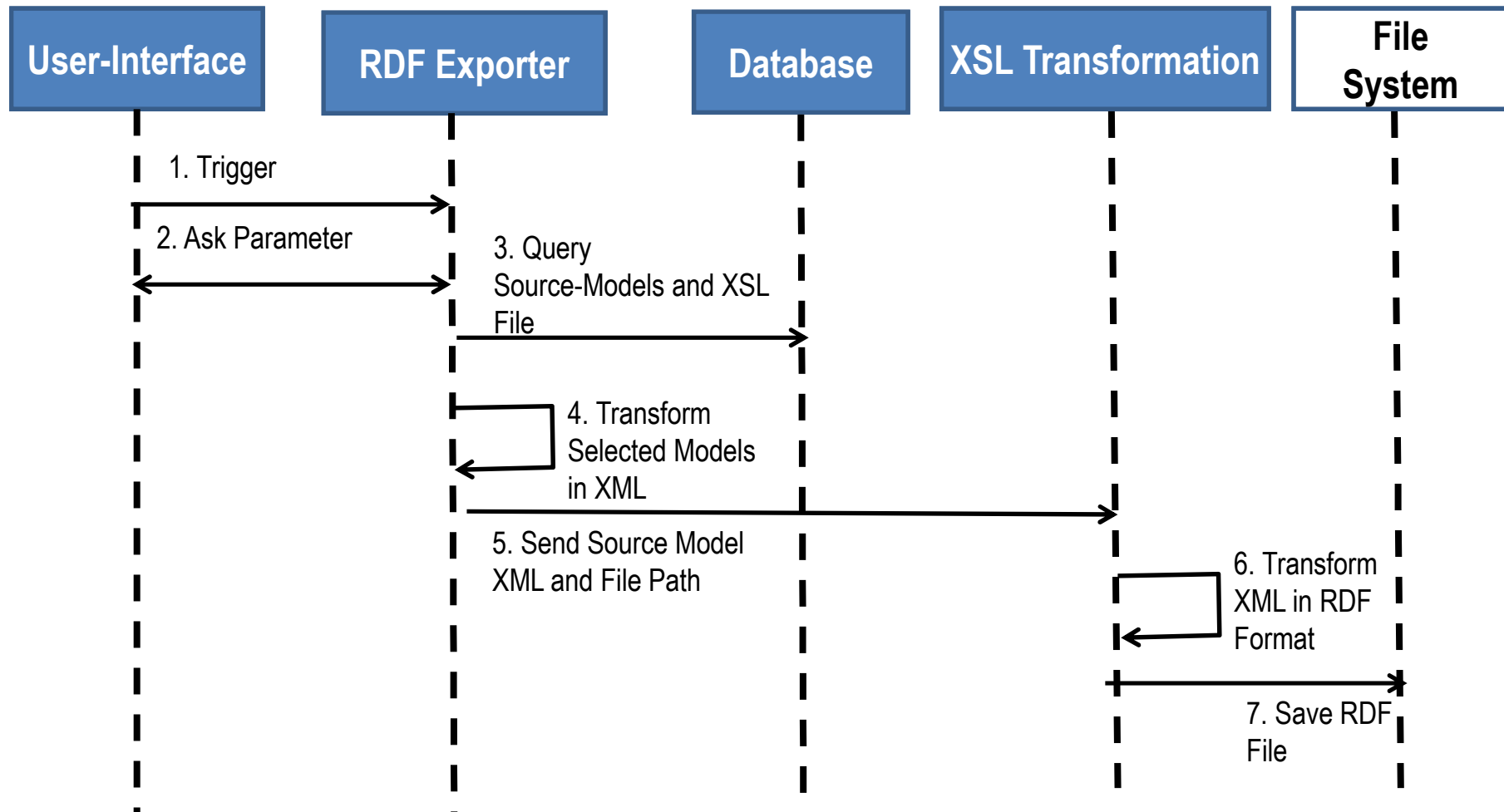
GOAL:

Demonstrate how to use AdoScript to invoke ADOxx Components and to establish interaction with a external system in order to transform Model format.

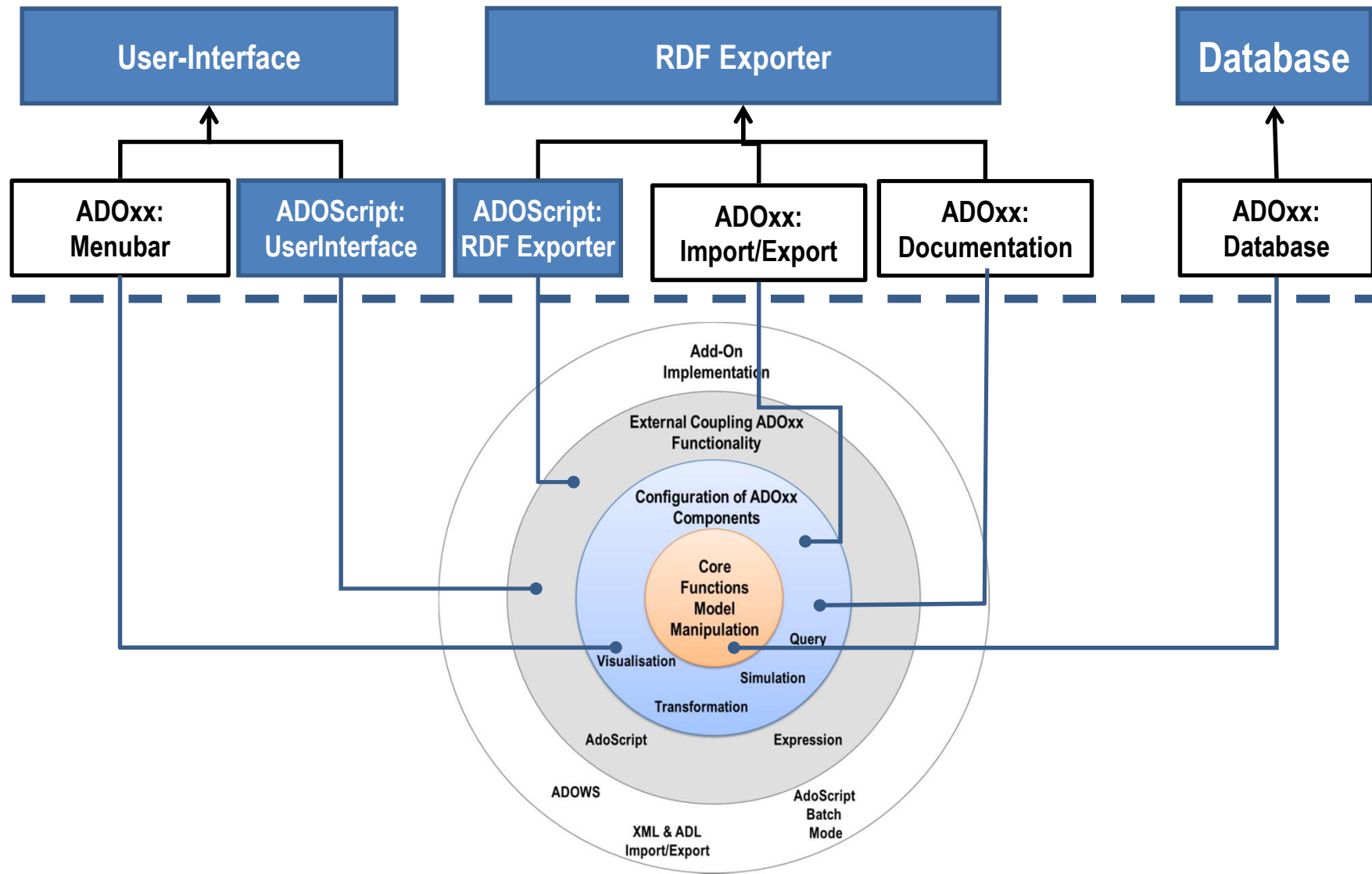
ADOxx Functionality on Meta Level



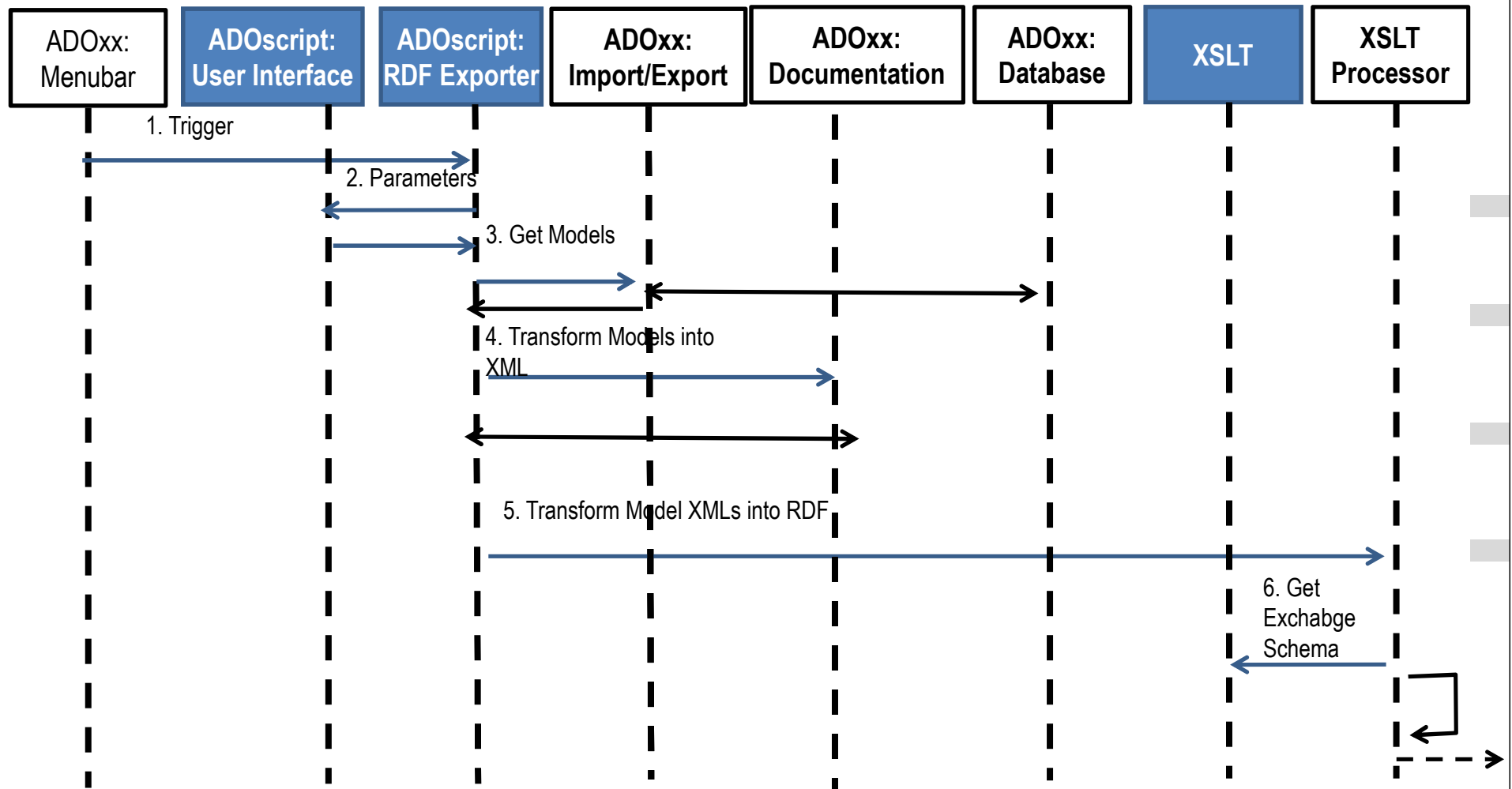
Description of Algorithm



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 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
- **ADOxx Import/Export Component:**
 - **ADOxx Import/Export Component:** is provided by the platform and can retrieve models from database .
 - **ADOScripts** can invoke the ADOxx Import/Export Component
- **ADOxx Documentation Component**
 - **ADOxx Documentation Component:** is provided by the platform and can transform the models in required format, in our case in XML format
 - **ADOScripts** can invoke the ADOxx Documentation Component

ADOxx Realisation Hands-On

- **Implementation of XSL File**
- **Configure ADOxx**

1. Configure Menubar

1. Implement Algorithm with ADOscript

1. ADOscript User Interface
2. Invoking Import/Export Component with ADOscript
3. Invoking Documentation Component with ADOscript
4. Invoking XSLT Processor with ADOscript

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

Export of OWL Models as RDF

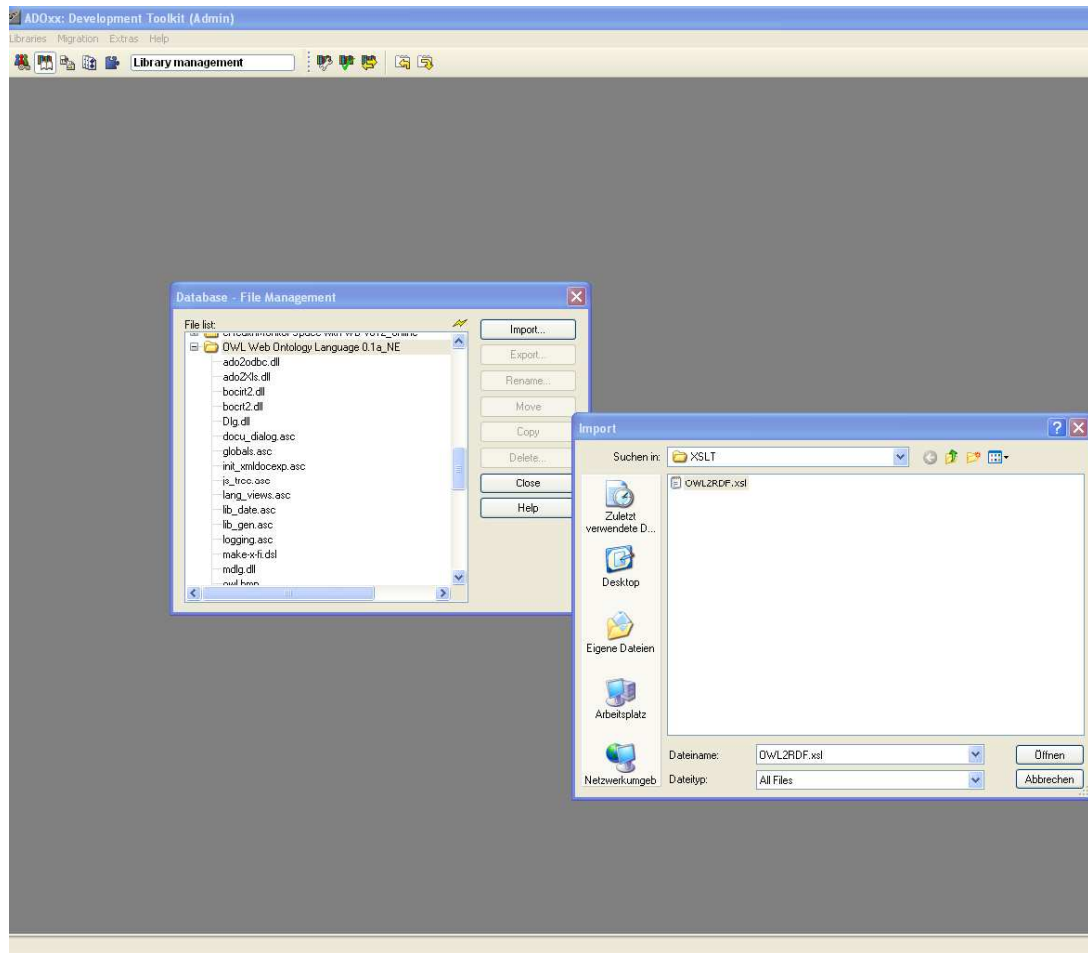
SCENARIO:
Configuration of ADOxx Component

Implement XSL File

```
43      <!-->
44      <!-->
45      <!-->
46      <!-->
47      <!-->
48      <!-->
49      <!-->
50      <!-->
51      </rdf:RDF>
52    </xsl:template>
53
54    <xsl:template name="BuildClasses">
55      <xsl:if test="count(//instance[@class='Class'])>0">
56        <xsl:for-each select="//instance[@class='Class']">
57          <xsl:element name="rdf:Description">
58            <xsl:attribute name="rdf:about">
59              <xsl:value-of select="$BaseURI"/></xsl:value-of>
60              <xsl:value-of select="./@name"/></xsl:value-of>
61            </xsl:attribute>
62            <xsl:if test="count(record[@name='Annotations']/row) > 0">
63              <xsl:call-template name="GetClassAnnotationLabel"/></xsl:call-template>
64              <xsl:call-template name="GetClassAnnotationComments"/></xsl:call-template>
65              <xsl:call-template name="GetClassAnnotationSeeAlso"/></xsl:call-template>
66            </xsl:if>
67            <xsl:element name="rdf:type">
68              <xsl:attribute name="rdf:resource">
69                <xsl:value-of select="$RdfSchemaURI"/></xsl:value-of>
70                <xsl:text>Class</xsl:text>
71              </xsl:attribute>
72            </xsl:element>
73            <xsl:call-template name="GetSuperClasses">
74              <xsl:with-param name="name">
75                <xsl:value-of select="./@name"/></xsl:value-of>
76              </xsl:with-param>
77            </xsl:call-template>
78          </xsl:element>
79        </xsl:for-each>
80      </xsl:if>
81    </xsl:template>
82
83    <xsl:template name="BuildObjectProperties">
84      <xsl:if test="count(//instance[@class='Object property'])>0">
```

length : 8827 lines : 231

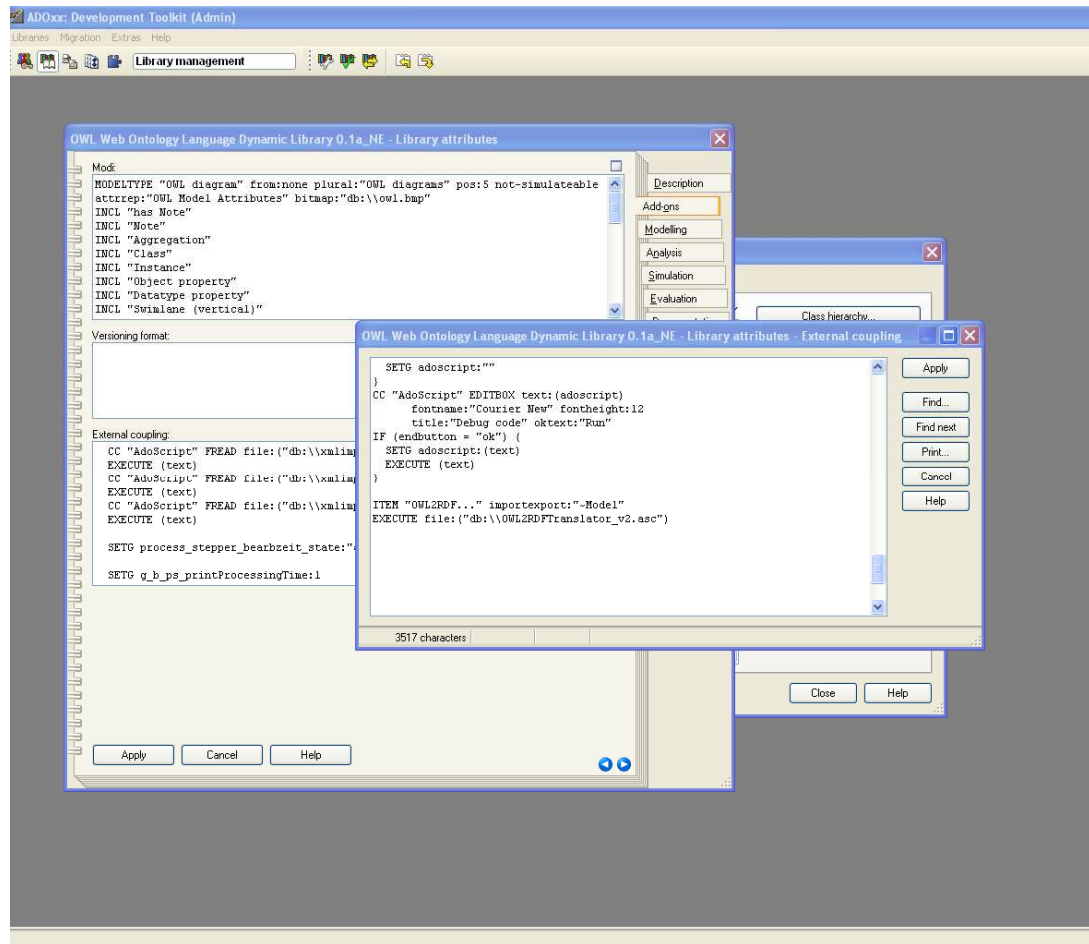
Copy XSL File into Database



Add Menubar

- Open Extras from Menubar
- Open File management
- Select ADOxx OWL Application Library
- Import XSL file

Add Menubar



Add Menubar

- Select Dynamic Library.
- Open Library Attributes
- Select Add-On
- Open External Coupling
- Add Menubar in External Coupling

ITEM "OWL2RDF..." importexport:"~Model"
EXECUTE file:("db:\\OWL2RDFTranslator_v2.asc")

Copy and Configure ADOscript

```
SET sXSLTfileName: "OWL2RDF.xsl"  
SET sTempFileName: "__rdf_temp.xml"
```

#Show Export Dialog and Invoke Import/Export Component

```
CC "ImportExport" SHOW_EXPORT_DLG mode: "xml" title: "XML_MODELS export"  
filedescription: "XML files" fileextension: "*.xml"
```

```
IF (endbutton = "ok") {
```

```
    SET sModelIDs: (modelids)  
    SET sModelGroups: (mgroupids)  
    SET sOutFilename: (filename)
```

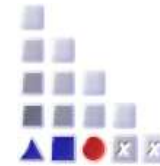
```
    SET nPosFileName: (bsearch ( sOutFilename , "\\" , (LEN sOutFilename)-1 ))  
    SET sExportFolder: (copy ( sOutFilename , 0 , nPosFileName+1 ))  
    SET sXSLTfilePath: (sExportFolder + sXSLTfileName)  
    SET sTempFilePath: ( sExportFolder + sTempFileName)  
    CC "AdoScript" FILE_COPY from: ("db:\\" + sXSLTfileName) to: (sXSLTfilePath)
```

#Invoke Documentation Component

```
CC "Documentation" XML_MODELS modelids: (sModelIDs) mgroupids: (sModelGroups)  
attrprofs: (attrprofids) apgroups: (apgroupids)
```

```
...
```

Further Questions?



www.adoxx.org

tutorial@adoxx.org

