FROM MODEL EDITORS TO MODELLING TOOLS: OPERATIONALIZING MODELLING METHODS WITH ADOxx

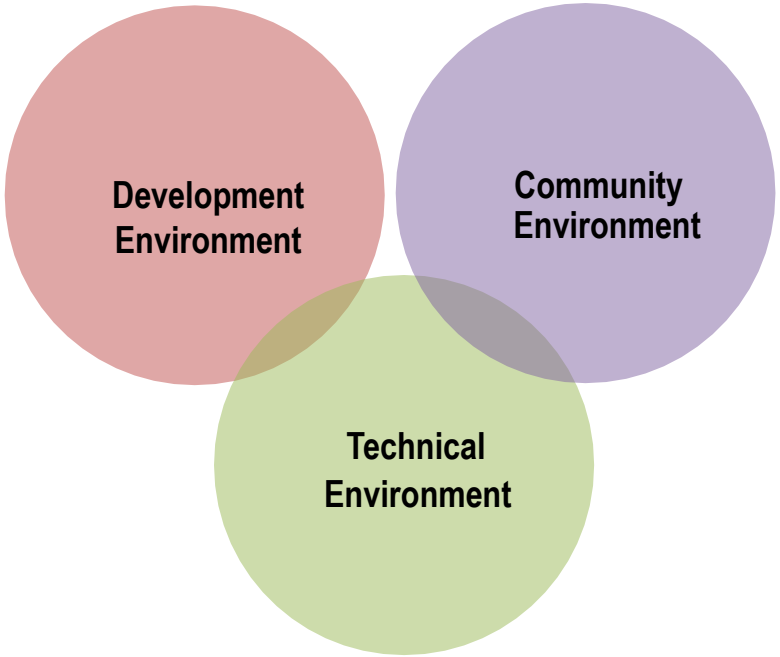
OMiLAB: Approach

- A **research and experimental laboratory** for the conceptualization, development and deployment of modelling methods and the models designed with them.
- Project space for Engineering of modelling methods and **modelling tools**
- A space for a community of researchers and practitioners sharing a common understanding about **model value**

Organisation: University of Vienna,
Faculty of Computer Science
Research Group: Knowledge Engineering

OMiLAB: Environment

- **Development environment** consists of
  - Core (Open Use): ADOxx on OMiLAB
  - Add-Ons (Open Source): implemented community tools such as Model Annotator, GraphRep Generator, Model Publisher, Method Publisher, OM-Repository, Meta-Model Browser, MLEA – Modelling Language Engineering Assistant
- **Technical environment** supports
  - virtual and physical accessibility
  - packaging and deployment capabilities
- **Community environment** provides
  - Web-platform based on Liferay
  - Community events like conferences, workshops, summer schools
  - Publications like books, conference and journal papers
  - Project networking activities
  - Newsletters, media and OM-TV



Agenda

- **Model Value**
- Definition of Model Structure on ADOxx
- Processing of Model Structure on ADOxx
  - Visualisation Functionality
  - Transformation Functionality
  - Analysis Functionality
- Conclusion

Model Definitions

- **Model as mappings of reality**
  …models as mappings of parts of reality for a particular purpose…

- **Model as a construction**
  ...the result of a construction of a modeler who declares for model users a representation of an original as significant at a given time using a language…

(Source: Schütte and Becker, 1998)

Model with Different Values

Representation Characteristic

"Models as a representation of natural or artifical originals, that again can be models." [1] (translated)

Abstraction Characteristic

"Models in general do not capture all attributes of the represented original, but only those that seem relevant to the modeller or model user." [1] (translated)

Pragmatic Characteristic

Models meet their substituion function for specific subjects, within a pre-determined time interval and with limitations on defined intellectual and/or real operations. [1] (translated)

(Source: Stachowiak 1973)

Introduction of Terms

- **Modelling Language:**
  Modelling constructs (object types) and their relations (relation types) to each other to declare a model.
- **Metamodel:**
  The model of the syntax of the modelling language
- **Meta2 Model:**
  Model of abstract syntax of a language to describe meta models.
- **Modelling Technique:**
  A modelling language and proceeding instructions for creation of a model in this modelling language.
- **Mechanisms und Algorithms:**
  Provision of functionalities to process models such as manipulation, visualisation, query, transformation or simulation depending on the modelling language and modelling procedure.

Cf. (Karagiannis and Kühn, 2002; Karagiannis and Höfferer, 2006; Kühn 2004; Karagiannis and Visic, 2011)

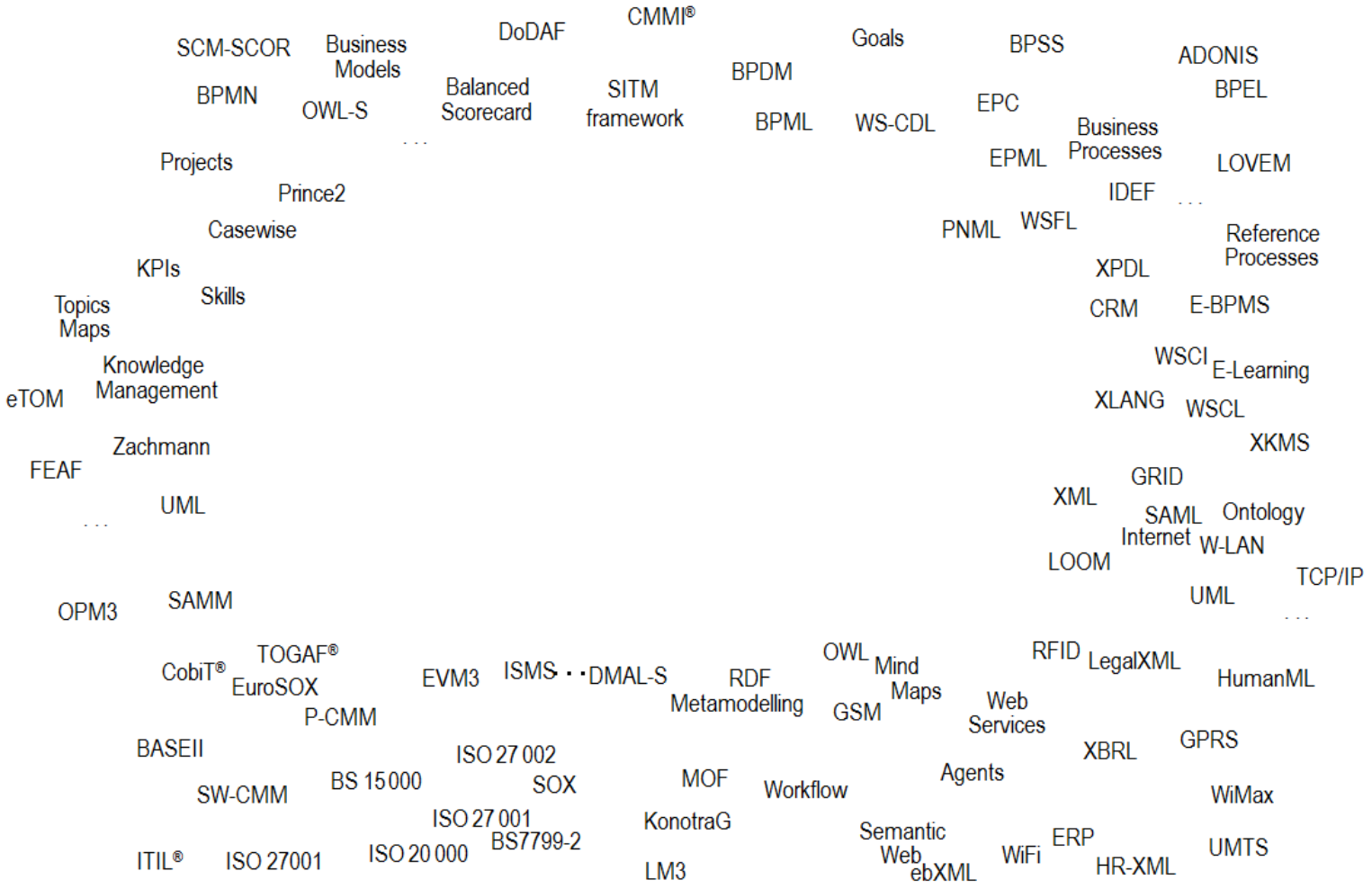
Model Values: An Example

THE RESULTS OF MODELLING CAN BE USED FOR GENERATING SOFTWARE, BUT ALSO ACT AS A BASIS OF ENTERPRISE KNOWLEDGE PLATFORMS

MACHINE PROCESSABLE

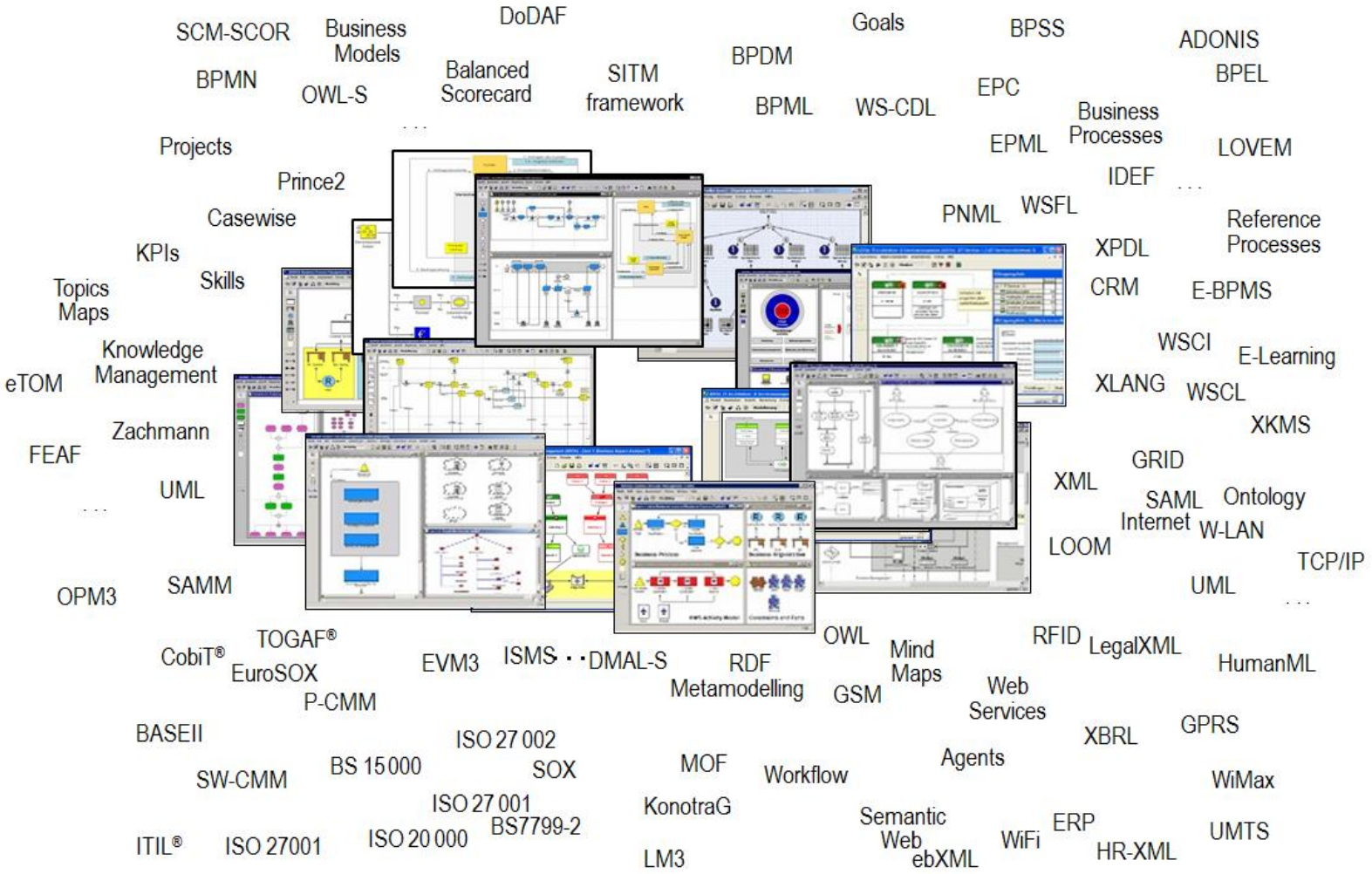
Cf. (Karagiannis, 2012 – Presentation at FInES – "Translating Knowledge Into Growth: Views from ICT Research to Support Future Business Innovation")

Some machine-processable formats ...



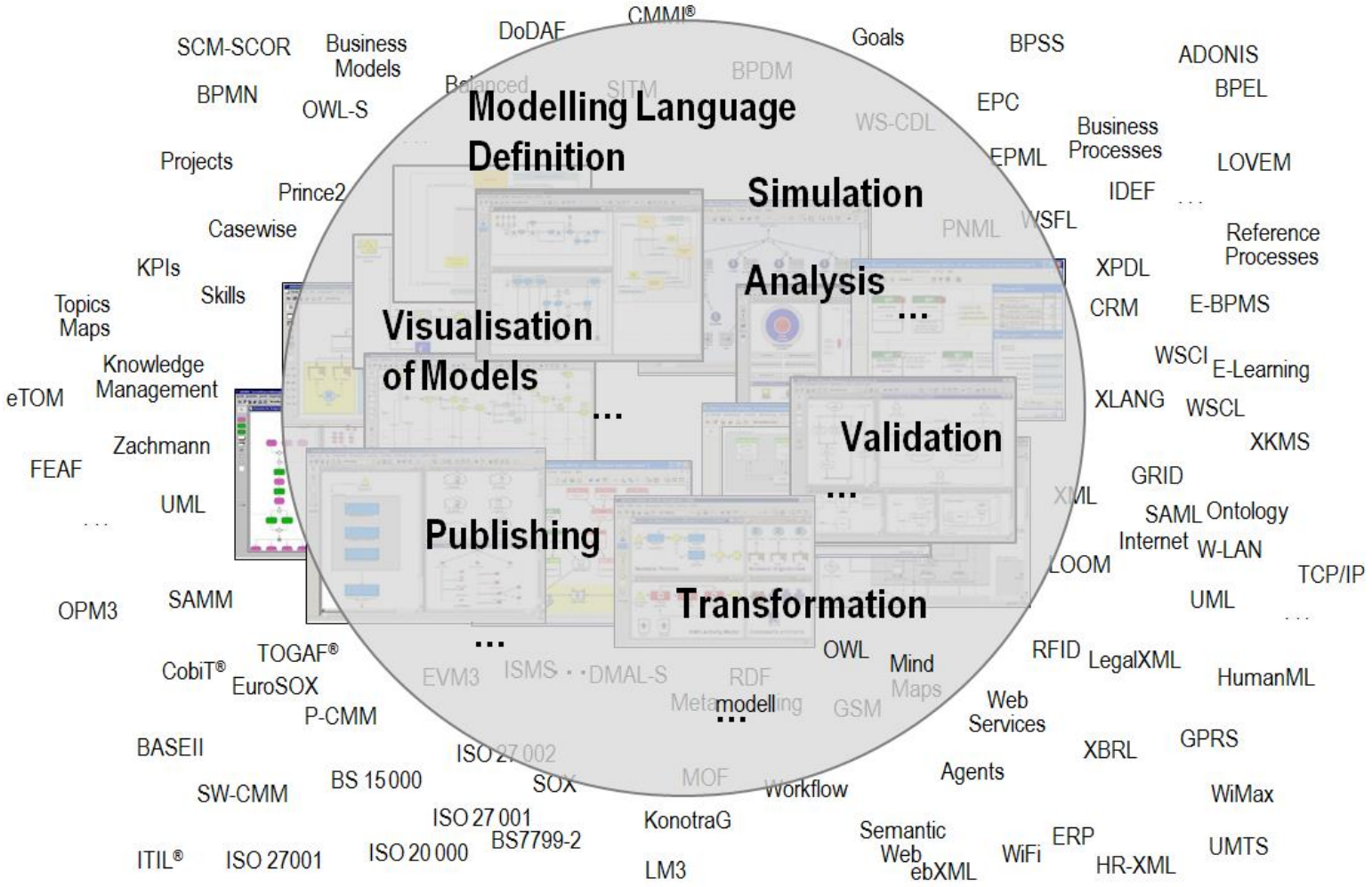
Cf. (Karagiannis and Kühn, 2002; Karagiannis and Höfferer, 2006)

... From an editor implementation, to ...



Cf. (Karagiannis and Kühn, 2002; Karagiannis and Höfferer, 2006)

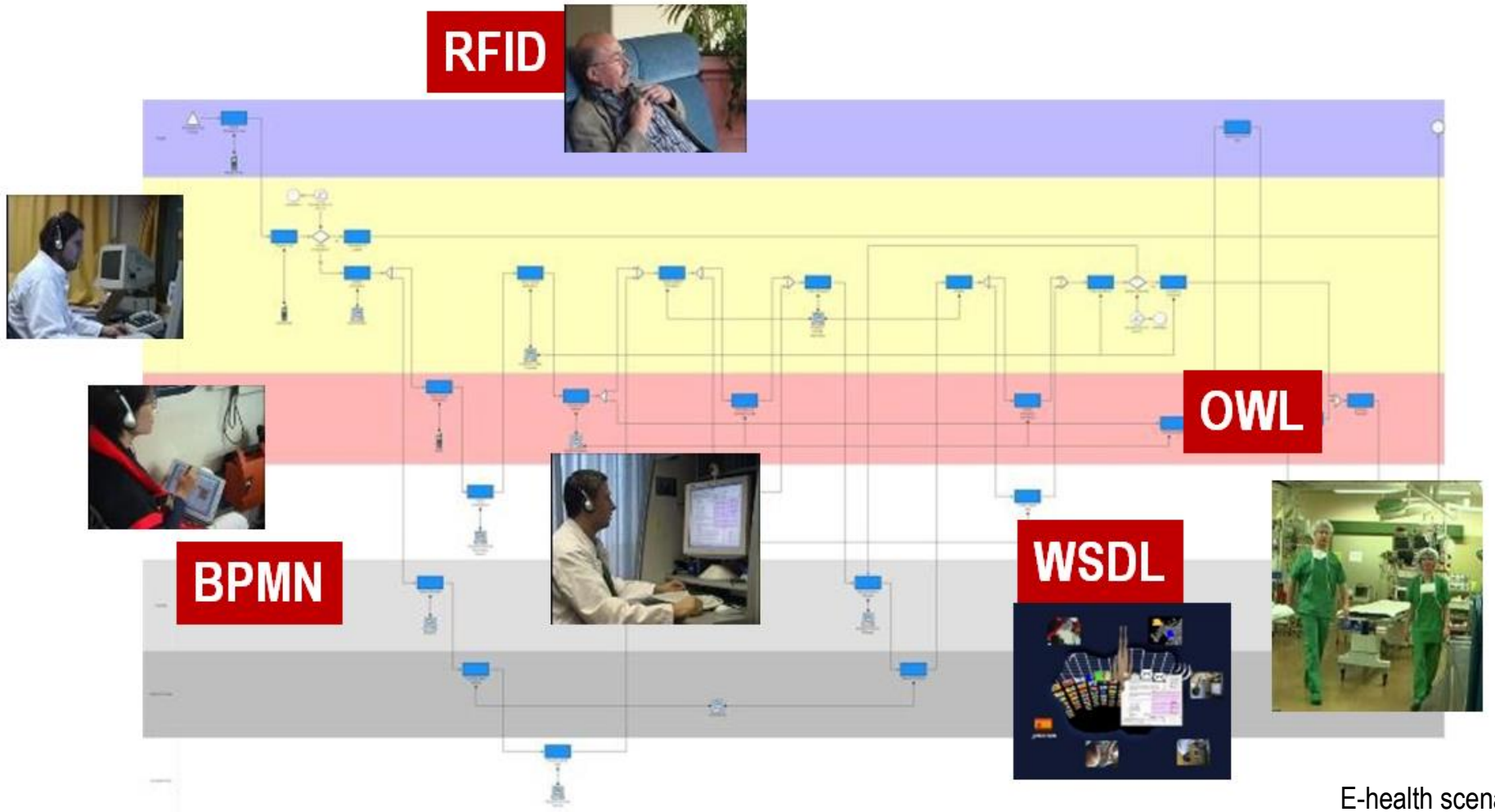
... to full-fledged modelling tool support ...



Cf. (Karagiannis and Kühn, 2002; Karagiannis and Höfferer, 2006)

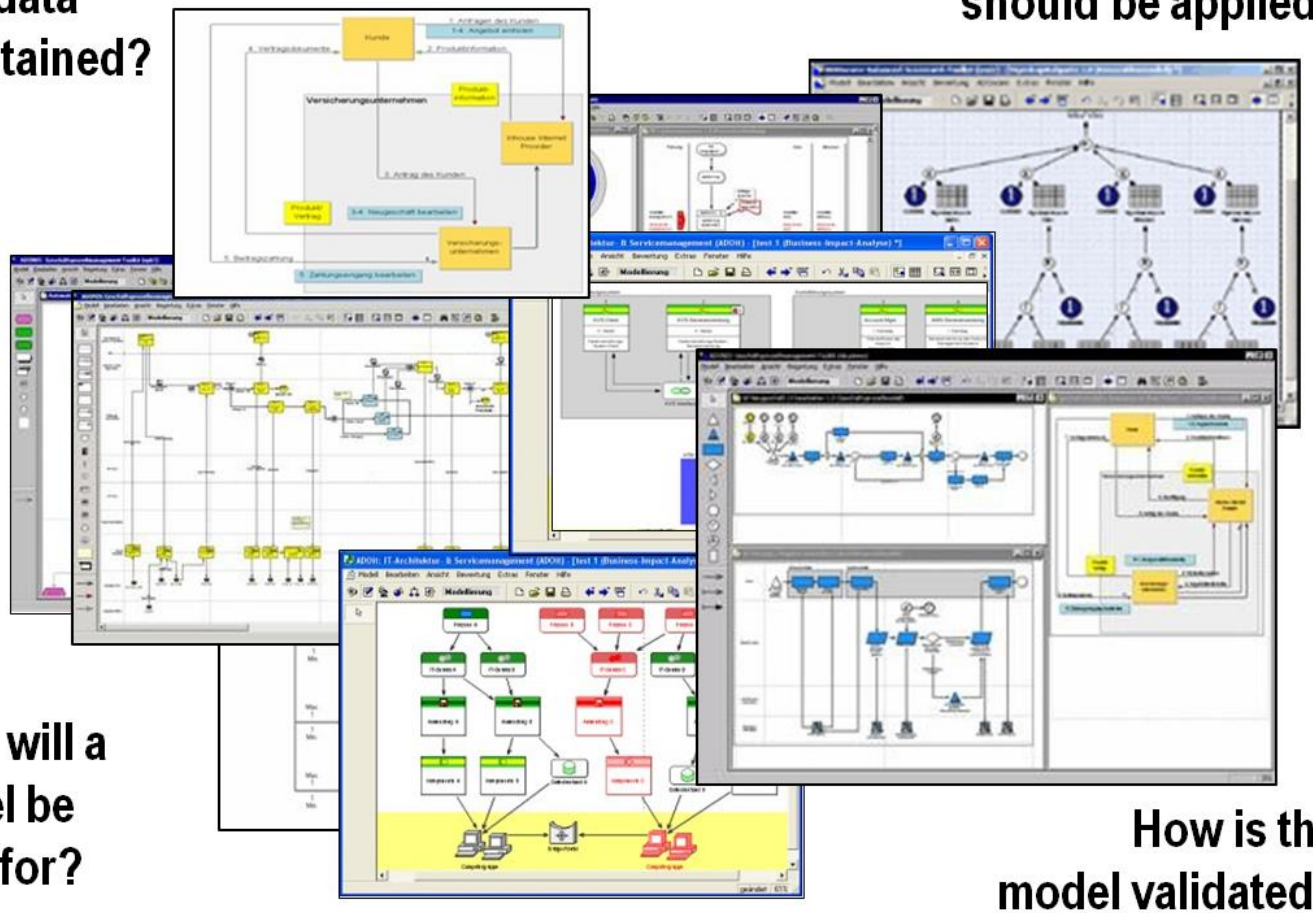
Scenario: Mobile eHealth Analysis and Simulation

AKOGRIMO Project



E-health scenario

The necessary information for model processing

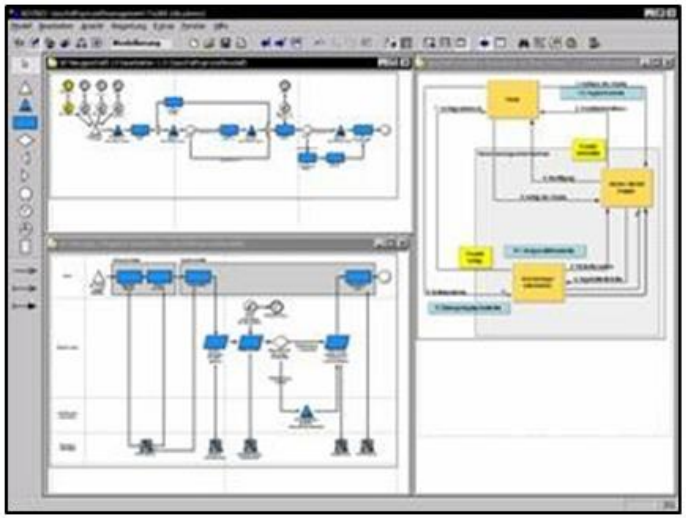


Some functionalities of modelling tools

Visualisation of models

User interaction like: drag and drop, zoom, grid snap, print, etc.

Simulation of models



Modelling language definition

Publishing in multiple formats

Transformation of models

Exchange of models

Analyse models and evaluate the results

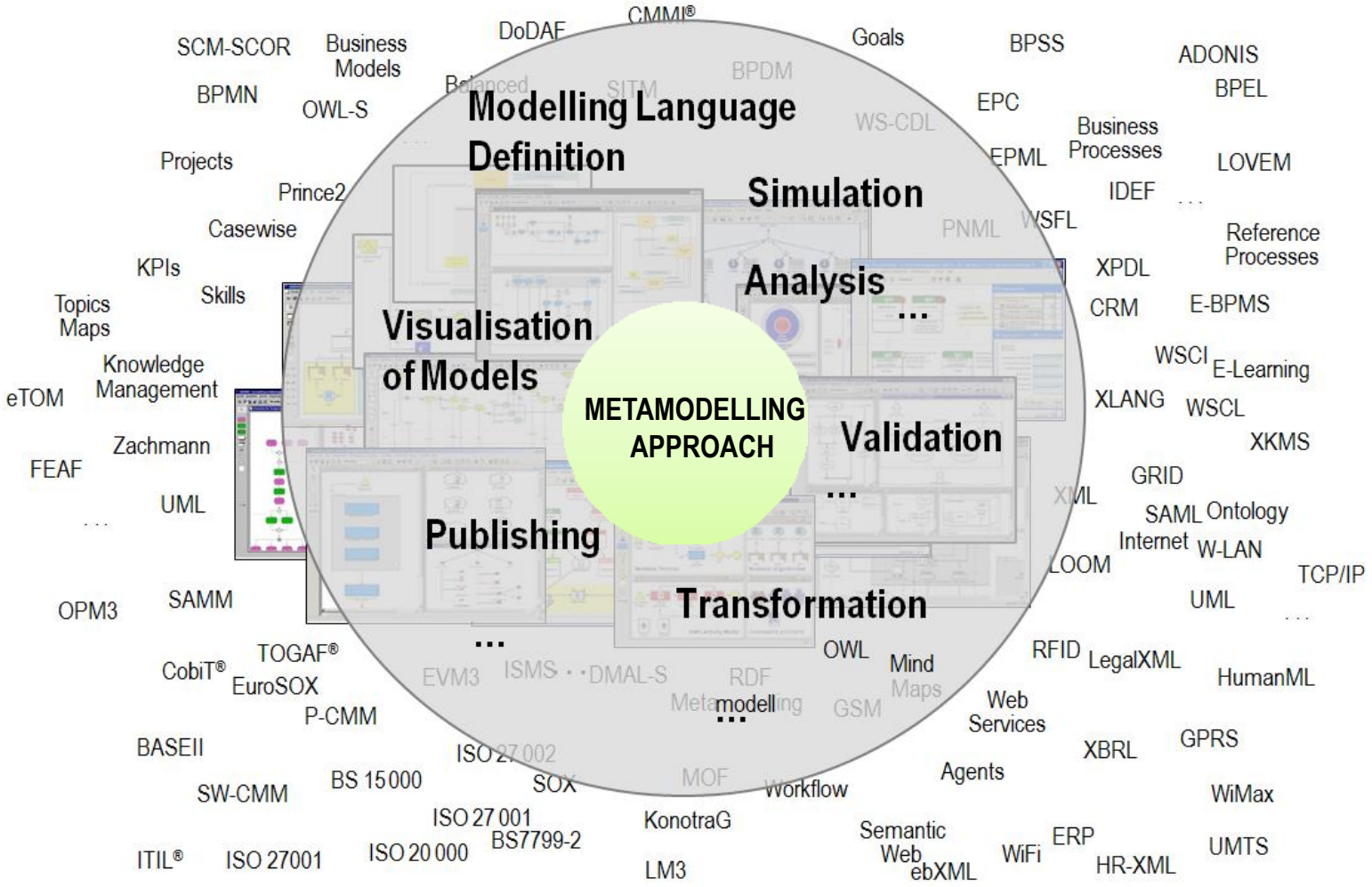
User access rights

Storage and Manipulation of Models

Security and Safety

Cf. (Karagiannis and Kühn, 2002; Karagiannis and Höfferer, 2006; Fill, 2009)

A Metamodel-based Realisation Approach



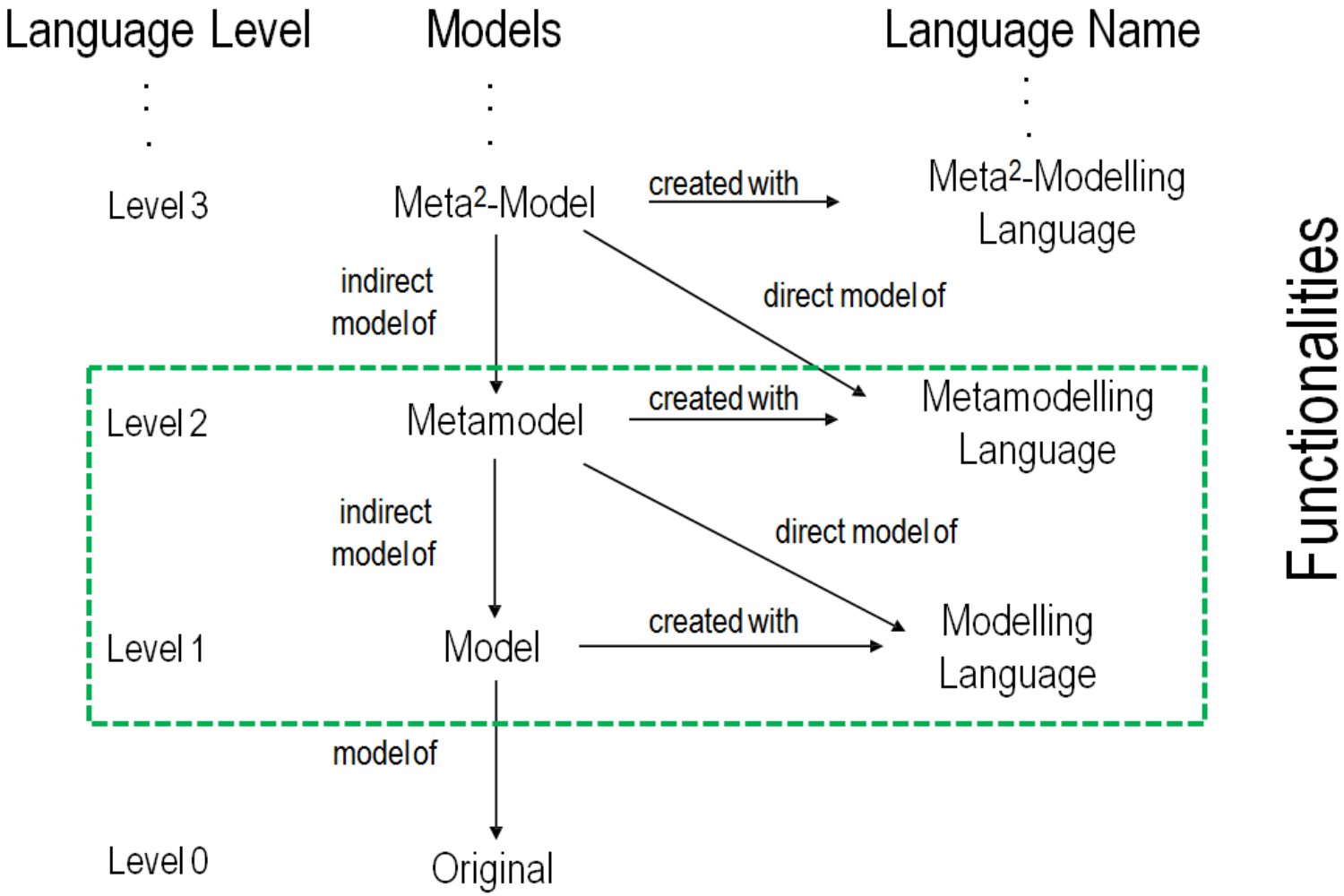
Cf. (Karagiannis and Kühn, 2002; Karagiannis and Höfferer, 2006)

...FOCUS OF TUTORIAL

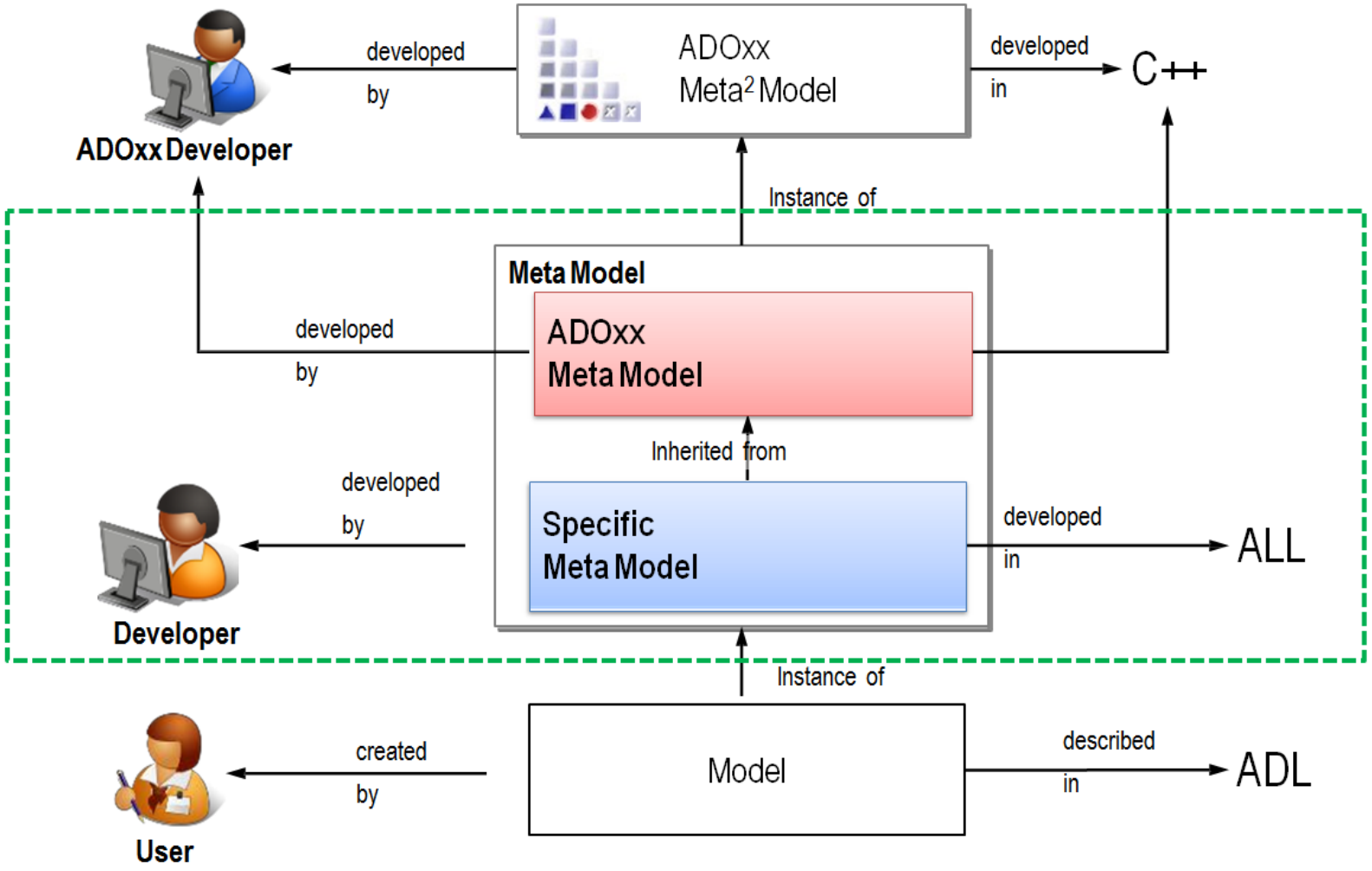
Agenda

- Model Value
- **Definition of Model Structure on ADOxx**
- Processing of Model Structure on ADOxx
  - Visualisation Functionality
  - Transformation Functionality
  - Analysis Functionality
- Conclusion

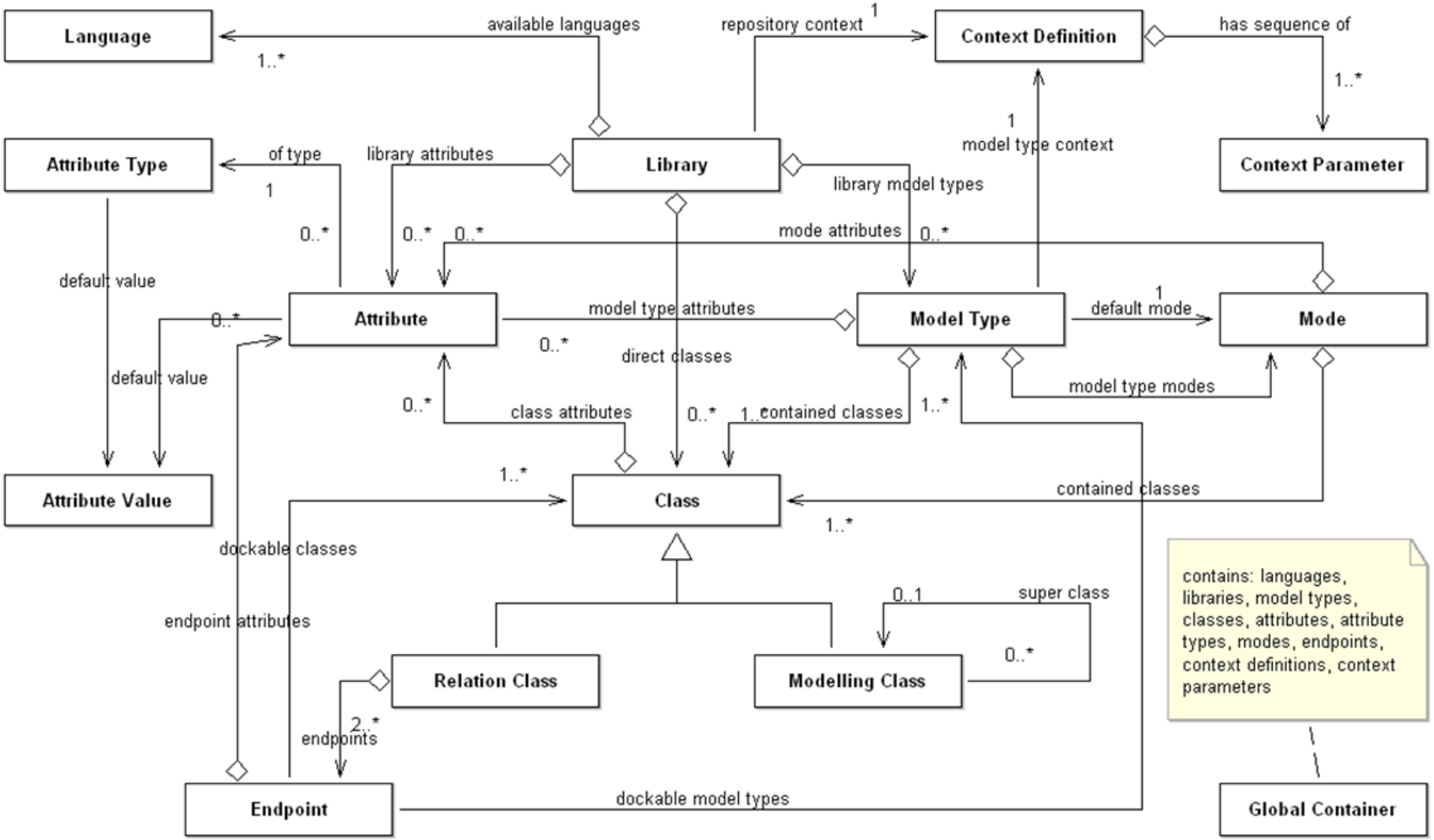
Definition of Model Structure and Functionalities



ADOxx Platforms Hierarchy

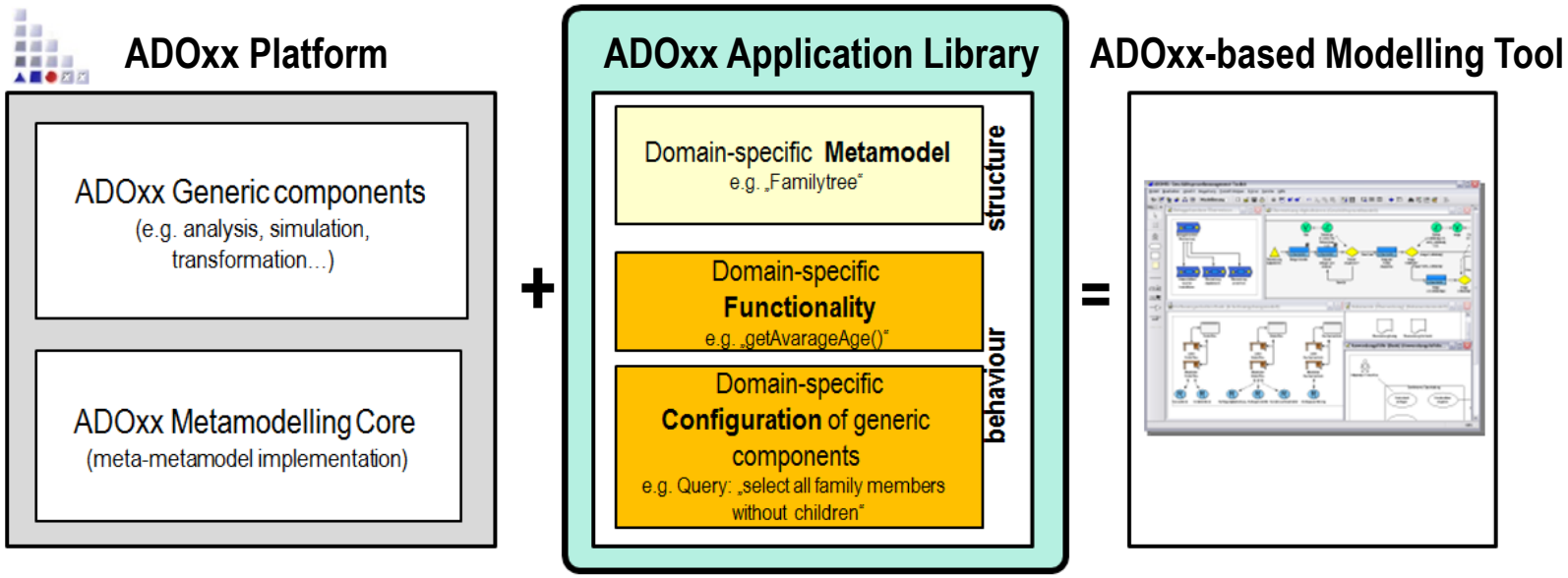


ADOxx Meta²-Model

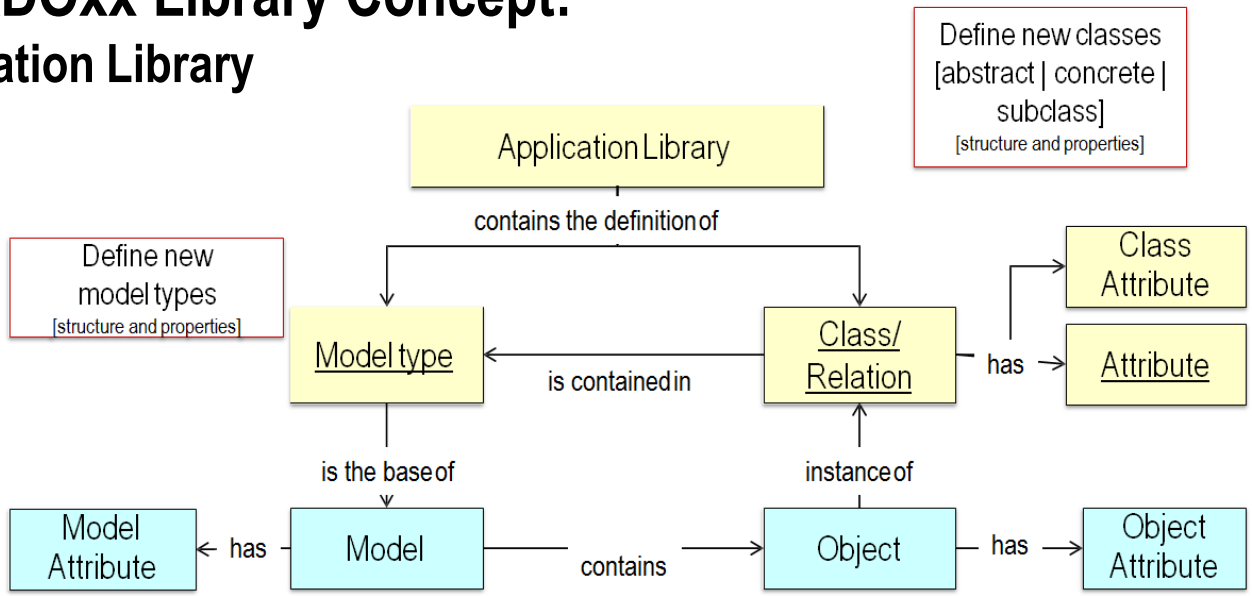


The ADOxx Library Concept:
More than a Metamodel

- The ADOxx Application Library is a concept which encapsulates both the structural and the behavioural aspects of metamodelling
- The ADOxx Application Library consists of:
  – Structural part: Metamodel definition
  – Behavioural part: Metamodel-specific functionality, Configuration of generic functionalities
- The ADOxx Application Library is a self-contained platform configuration package containing all neccessary artefacts to configure a fully-fledged modelling tool



The ADOxx Library Concept:
Application Library



- **Model Types:**
  A model type is a well-defined sub collection of classes and relation classes of a meta model.
- **Classes:**
  A class is a construct that is used as a template to create objects of that class. The objects of a class are alternatively called "instances"
- **Attributes:**
  An attribute is a property of a modelling construct such as a model, object or relation. Each attribute has a type and a value.
- **Relations:**
  A relation class is a construct that is used as a template to create relations between objects. A relation class is defined between classes. A relation is always a directed connection between objects, i.e. each relation has a from-side and a to-side.

Cf. (Junginger et al., 2000; Kühn, 2004; Fill, Redmond, Karagiannis, 2012)

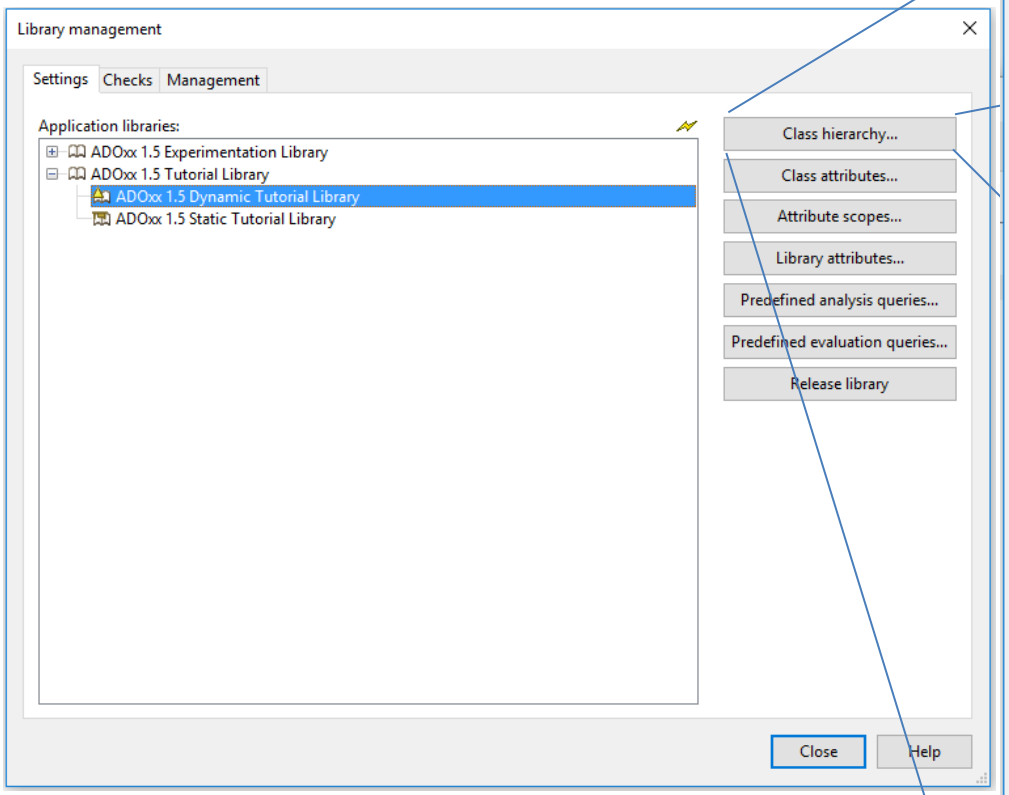
Class Types in ADOxx

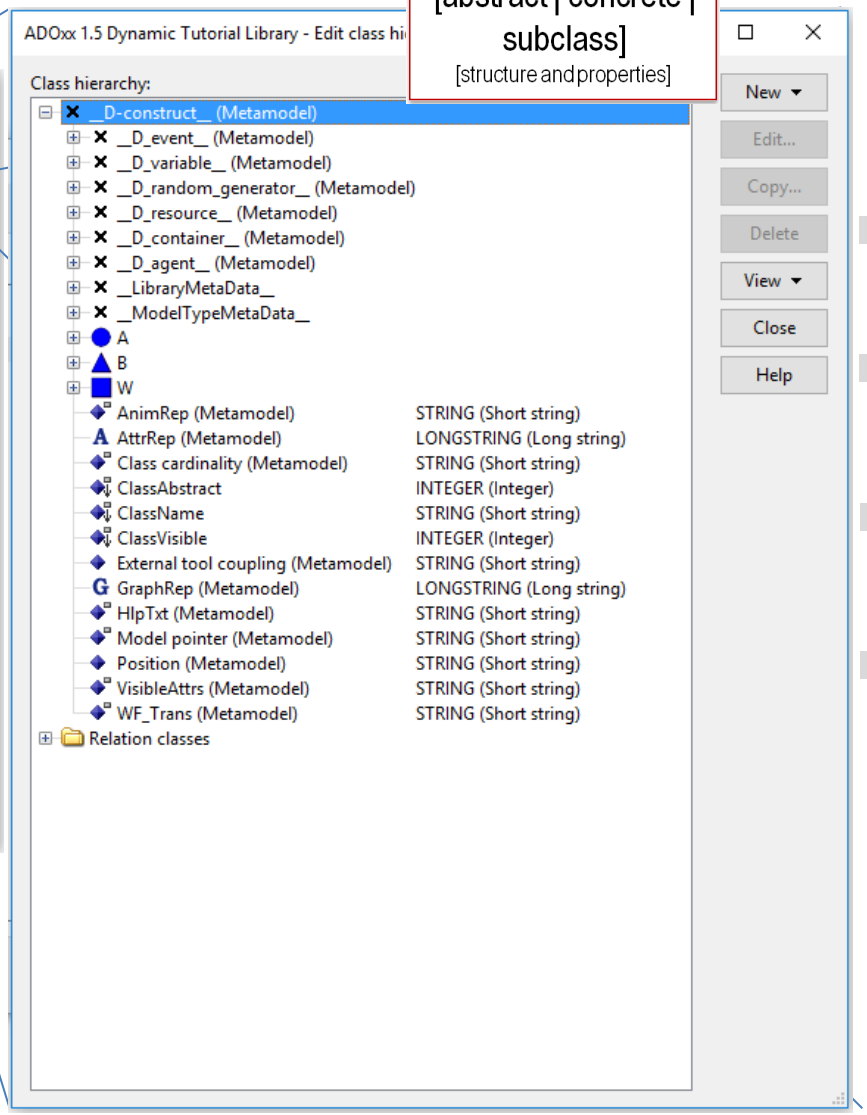
Define new classes
[abstract | concrete | subclass]
[structure and properties]

- **Abstract Classes**
  - Abstract classes are self-defined classes enabling to structure the meta model and define syntax in form of attributes and semantic, which is inherited by sub-classes.
  - Abstract classes either inherit from the root class of the meta model, or from any other class of the meta model. Hence, they inherit the behaviour from their super-class – which is often a pre-defined abstract class from the ADOxx meta model.
  - Abstract classes enable an efficient meta model, hence they may not be in every ADOxx meta model.
  - Nomenclature: _ Class Name _
- **(Concrete) Classes**
  - Classes are self-defined classes defining a concrete modelling class that can be used, when applying the corresponding modelling language. Hence all model objects in every model created on ADOxx is an instance of a class.
  - Classes inherit the semantic and the attributes from the Pre-defined abstract class and additionally - in case of inheriting - from the abstract class.
  - Classes enable the realisation of a concrete meta model.
  - Nomenclature: Class Name

Cf. (Fill, Redmond, Karagiannis, 2012)

Demonstration: Class Definition 1



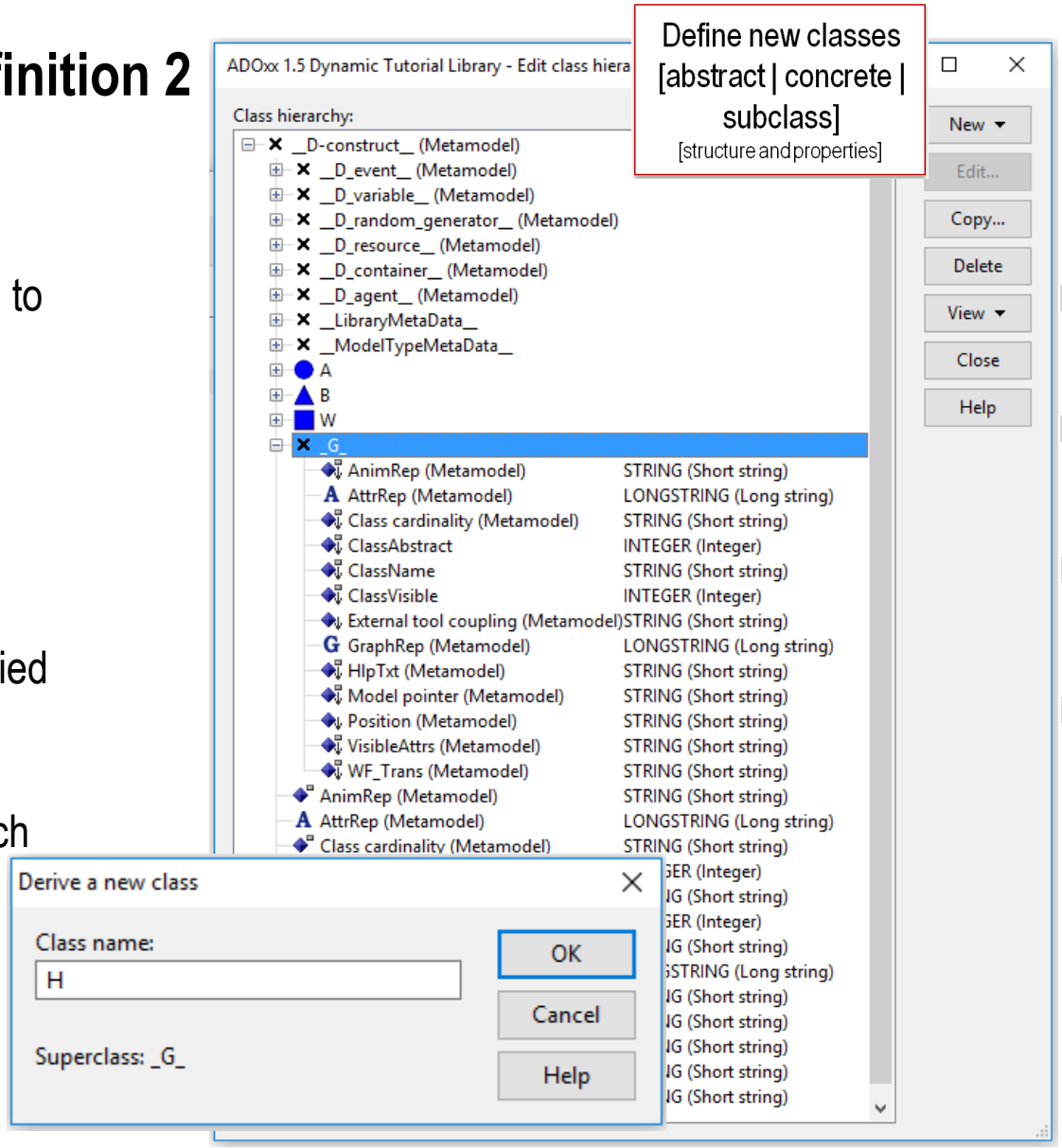


1. Open the Library Management Component
2. Expand the Application Library and select the library
3. Press “Class Hierarchy” to add/delete/copy classes

Demonstration: Class Definition 2

1. Add a new concrete class below the abstract element that is used to define a concrete class
2. Select the abstract class, click “New” -> “New class”
3. Name the new class

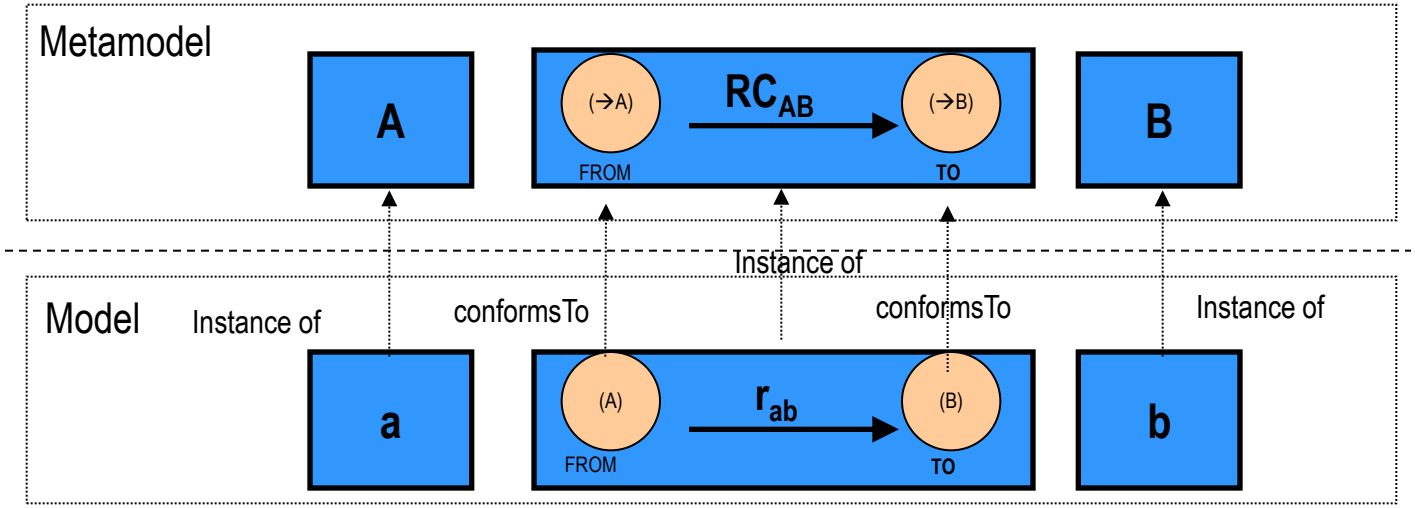
The new created class can be identified on instance level by the “Name” attribute. This attribute is automatically/implict available for each class



Relation Types

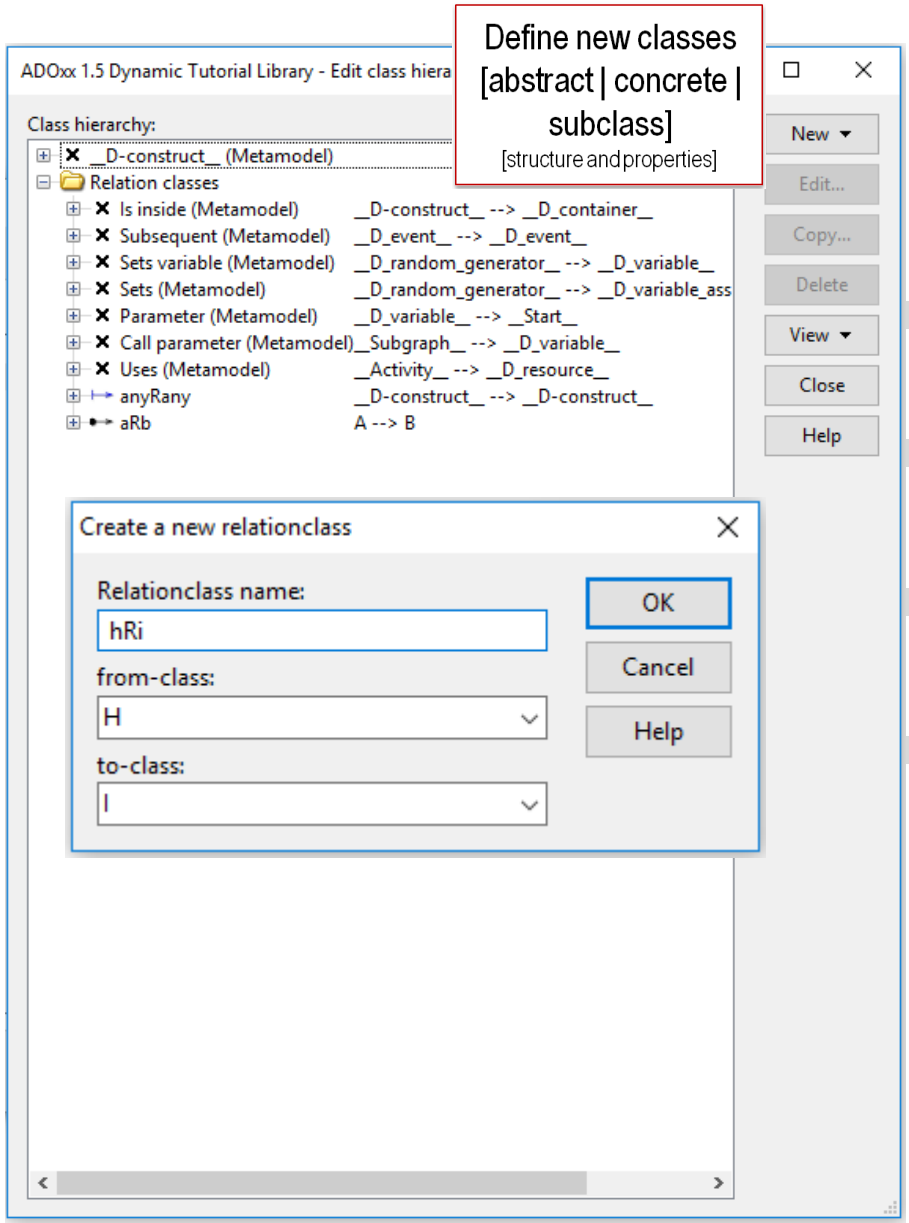
Define new classes
[abstract | concrete | subclass]
[structure and properties]

- Relations in ADOxx are expressed either as a class “Relation Class” or as a pointer in form of an attribute called “InterRef”.
- Relation as Class “RC”
  – describes relationship between two objects from two or more classes within one model.
  – has start and endpoints define which (abstract) classes a relation can connect
  – Cardinality and attribute defined the semantic of the relations class
- Relation as Attribute “InterRef”
  – Is a special configuration of a Relation Class and describes the relationship between two objects from two or more classes within or across models.
  – Is a pointer represented as an attributed in the class the relation starts from, with defined classes the relation can point to.
  – Cardinality defines the semantic of the InterRef



Demonstration: Relation Class Definition

- Add two new relation classes to connect classes
  - Click “New” -> “New relation class”
  - Name new relation class
  - Define from-class
  - Define to-class



Definition of Attributes

Define new classes
[abstract | concrete | subclass]
[structure and properties]

- Attributes for classes and relation classes have to be defined in the definition section of the class/relation class with 'TYPE'.
- The following attribute types are possible:

INTEGER	**integer**
DOUBLE	**floating number**
STRING	**string – max. 3699 symbols**
LONGSTRING	string – max. 32000 symbols
TIME	time
DATE	date
DATETIME	date and time
ENUMERATION	enumeration for selecting a characteristic
ENUMERATIONLIST	enumeration for selecting one or several characteristics
DISTRIBUTION	statistical distribution
PROGRAMCALL	**enumeration for selecting a program**
RECORD	a table of attributes
EXPRESSION	a formula
INTERREF	**reference on a model or an instance**
ATTRPROFREF	a preset set of attribute values

Demonstration: Attribute Types and their Appearance

Numerical Attributes: Integer (INTEGER)

Define new classes [abstract | concrete | subclass]
[structure and properties]

- An attribute of the type "Integer" is defined as an integer from -1,999,999,999 to 1,999,999,999.
- An ADOxx integer is limited to 10 digits plus an optional sign ('+' or '-')
- The standard value of attributes of this type is "0" or a value defined

Demonstration: Attribute Types and their Appearance

Numerical Attributes: Floating number (DOUBLE)

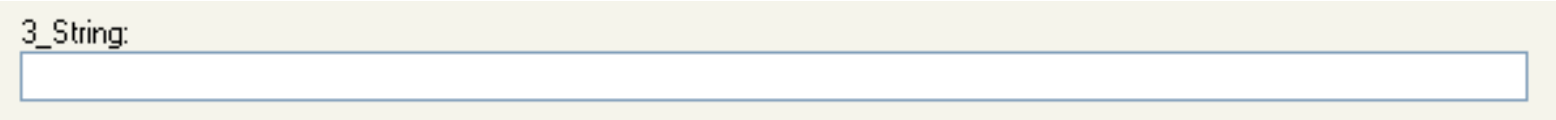
Define new classes [abstract | concrete | subclass]
[structure and properties]

- The amount of decimal places is defined by the attribute definition
- An attribute of the type "Double" is defined for a float within +/-999,999,999,999,999 for an integer (without decimal places) or +/-999,999,999.999999 for figures with 6 decimals.
- The corresponding attribute value is displayed to 6 decimal places. That means that a double value should not exceed a total of 15 significant digits with at last 6 decimal digits!
- The standard value of attributes of this type is "0.000000" or a value defined in the application library.

Demonstration: Attribute Types and their Appearance

String Attributes: String (STRING)

Define new classes
[abstract | concrete | subclass]
[structure and properties]



- An attribute of the type "String" is defined for texts up to 3700 characters of any type.
  - Hint: The maximum number of characters is 250 for name. That concerns classes, relation, instances, attributes, application models, libraries and application libraries.
  - Model names have a special rule!
- The standard value of attributes of this type is "" (no entry) or a value defined in the application library.

Wrap up: Definition of Model Structure on ADOxx

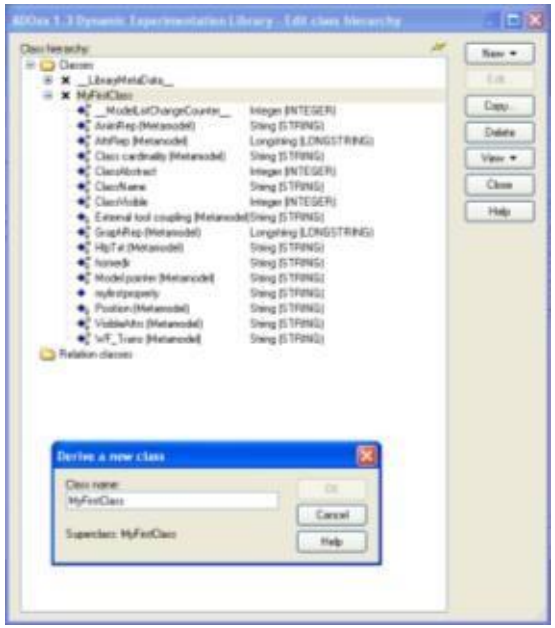
- Introduction of ADOxx Library Concept
- Demonstration of implementation of model structure

RESULT ACCOMPLISHED:

- Implemented model structure on ADOxx

```
//---------------------------------------
CLASS <MyFirstClass> : <__LibraryMetaData__>
//---------------------------------------

            CLASSATTRIBUTE <ClassAbstract>
            VALUE 0


            CLASSATTRIBUTE <ClassVisible>
            VALUE 1


//--- Class <MyFirstClass> - Instance attributes-
            ATTRIBUTE <myfirstproperty>
            TYPE STRING
            VALUE ""

            FACET <MultiLineString>
            VALUE 0
            FACET <AttributeHelpText>
            VALUE ""
            FACET <AttributeRegularExpression>
            VALUE ""


//--- Class <MyFirstClass> - default values-
```

Model Structure Definition using ADOxx Library Language (ALL)

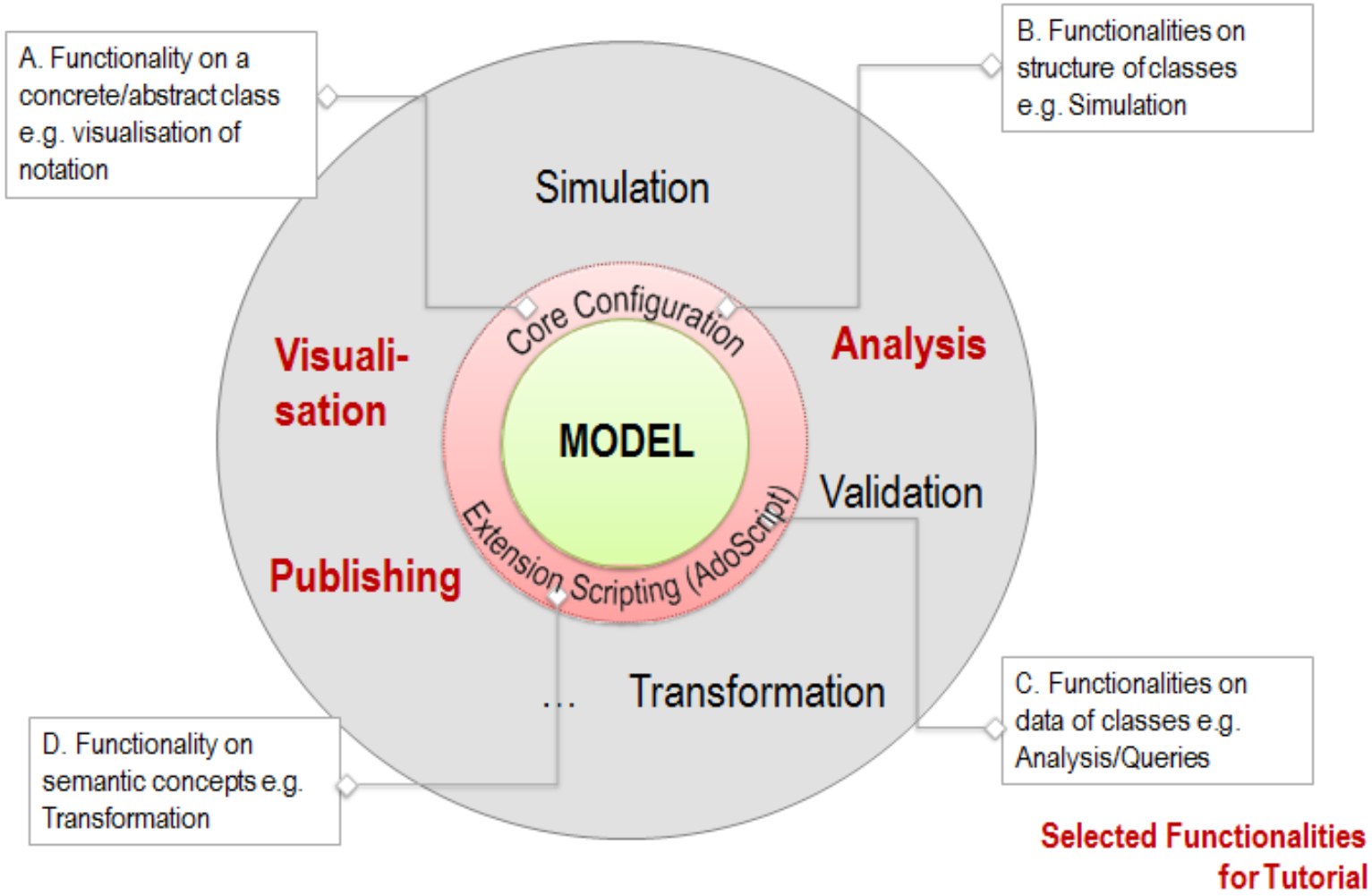


Model Structure Definition using Development Environment

Agenda

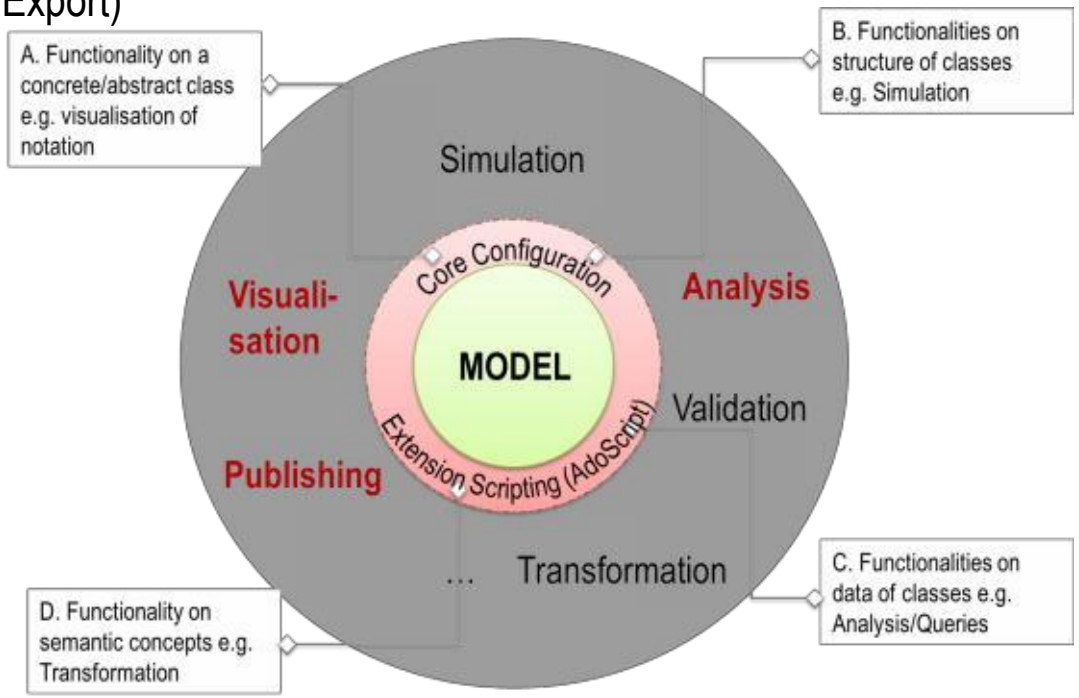
- Model Value
- Definition of Model Structure on ADOxx
- **Processing of Model Structure on ADOxx**
  - Visualisation Functionality
  - Transformation Functionality
  - Analysis Functionality
- Conclusion

Model Processing Classification



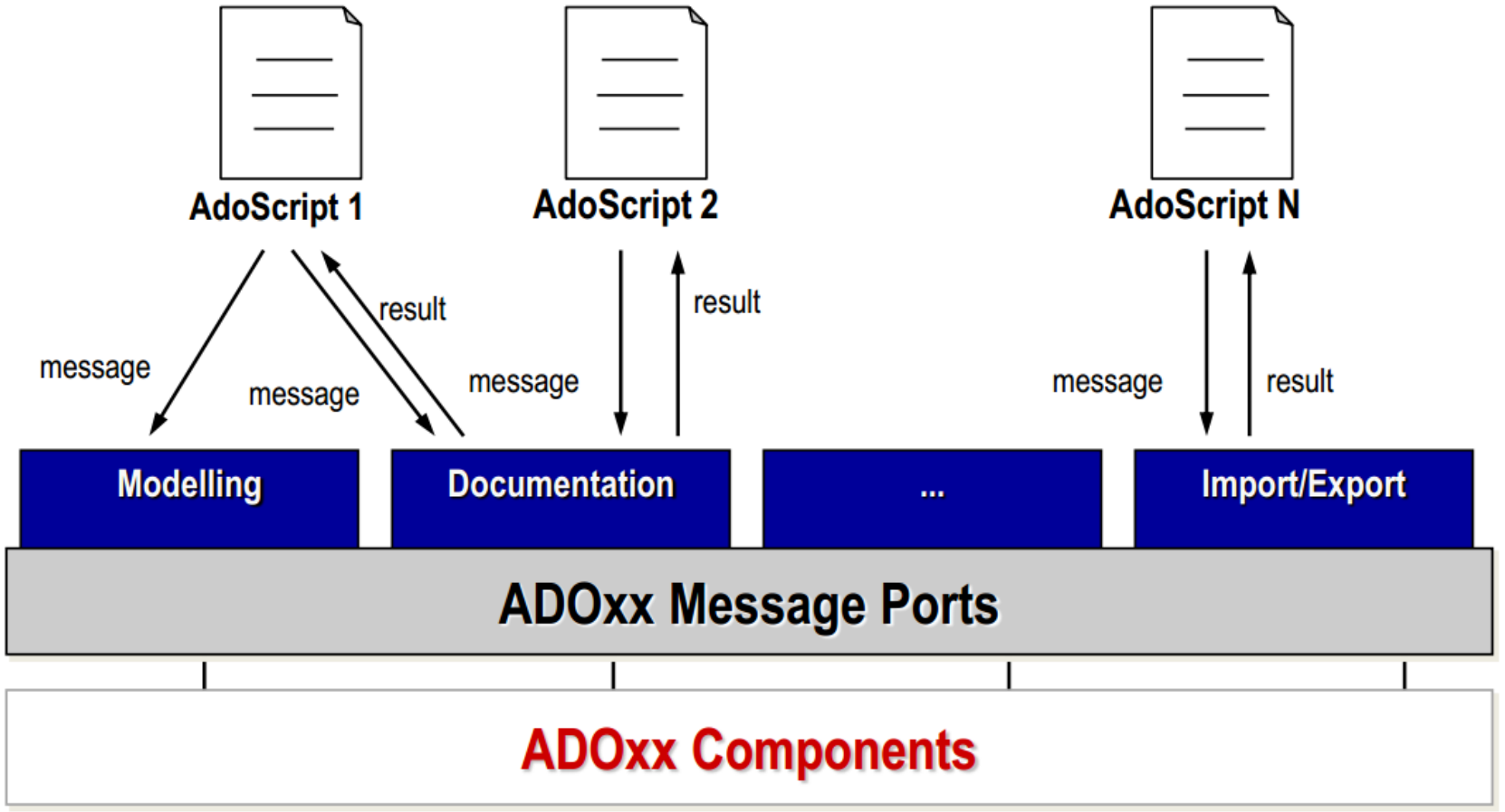
Core Configuration

- User and Access Right Management
- File Management
- Library Persistence (DB and File Persistence)
- Model Persistence (DB and File Persistence)
- Serialization Functionality (Import/Export)
- ...



Extension Scripting (AdoScript)

AdoScript: The ADOxx DSL



Programmable through Scripting APIs

- Method-specific development of functionalities through scripting
- Function calls/APIs of the platform are possible through scripting.

Component APIs
Messageport **Acquisition**
Messageport **Modeling**
Messageport **Analysis**
Messageport **Simulation**
Messageport **Evaluation**
Messageport **ImportExport**
Messageport **Documentation**
Messageport **AQL**

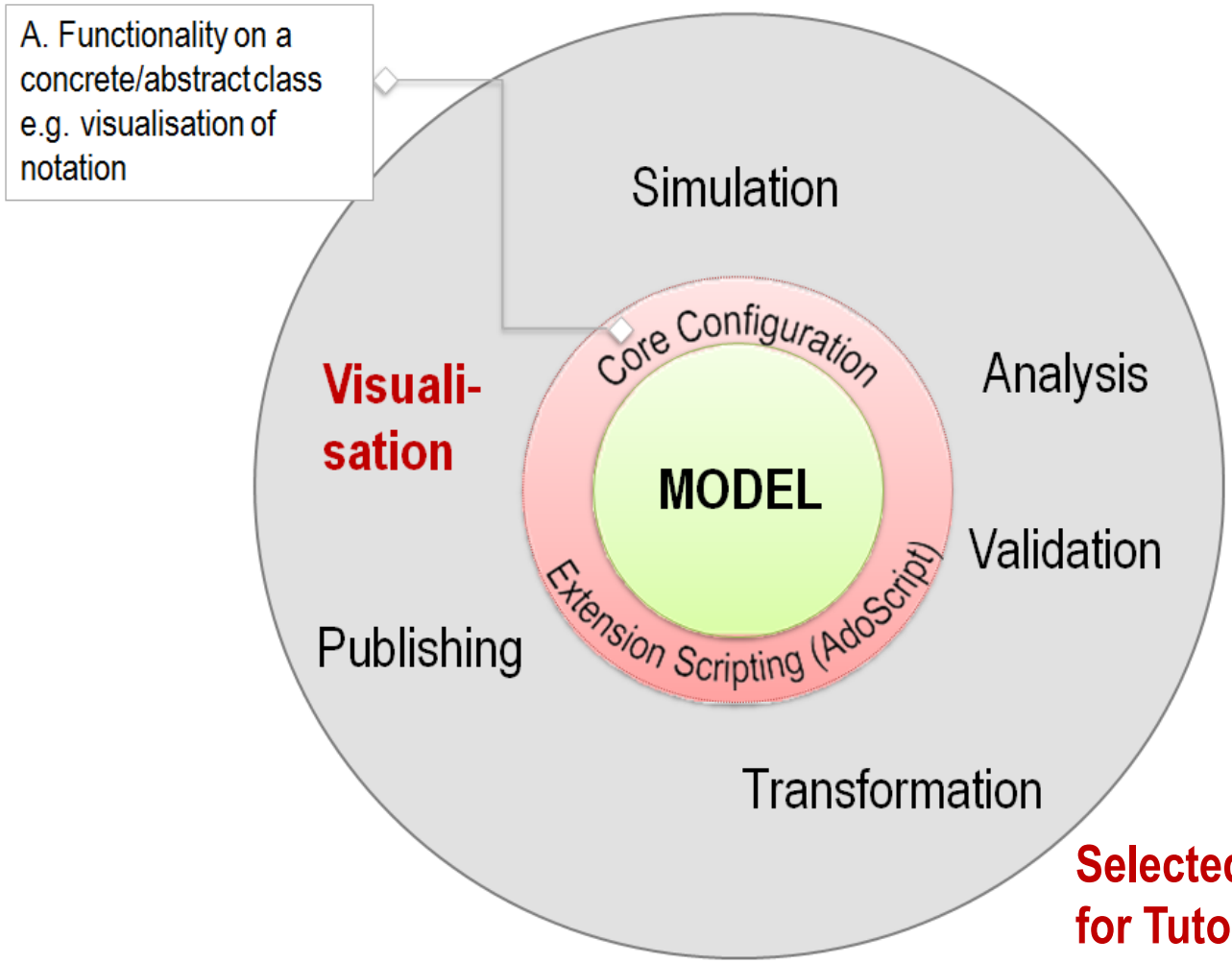
UI APIs
Messageport **AdoScript**
Messageport **CoreUI**
Messageport **Explorer**

Manipulation APIs
Messageport **Core**
Messageport **DB**
Messageport **UsrMgt**

Application APIs
Messageport **Drawing**
Messageport **Application**

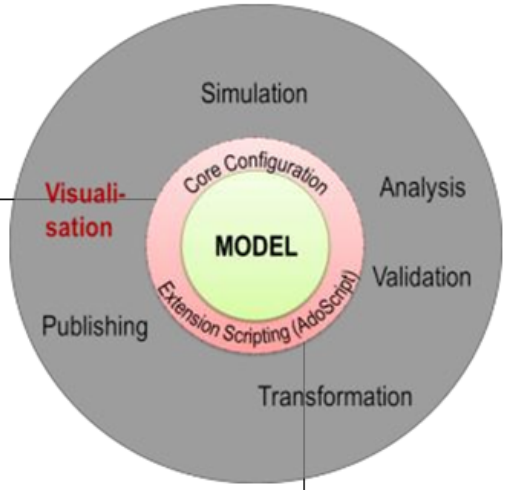
About 400 APIs are available.

Model Processing Functionality: Visualisation



Selected Functionalities for Tutorial

Object Visualisation



Platform Functionality

- Object visualisation
- Model visualisation
  - Tabular view incl. view concept
  - Graphical view incl. view concept
  - Machine-generated models
    - Model analysis visualisation
    - Information visualisation
  - Human-generated models
    - Support functionality (automatic & user-defined)

OMiLAB Development Tools

- OMiLAB GraphRepGenerator
- AdoScript Syntax Highlighter

OPEN SOURCE

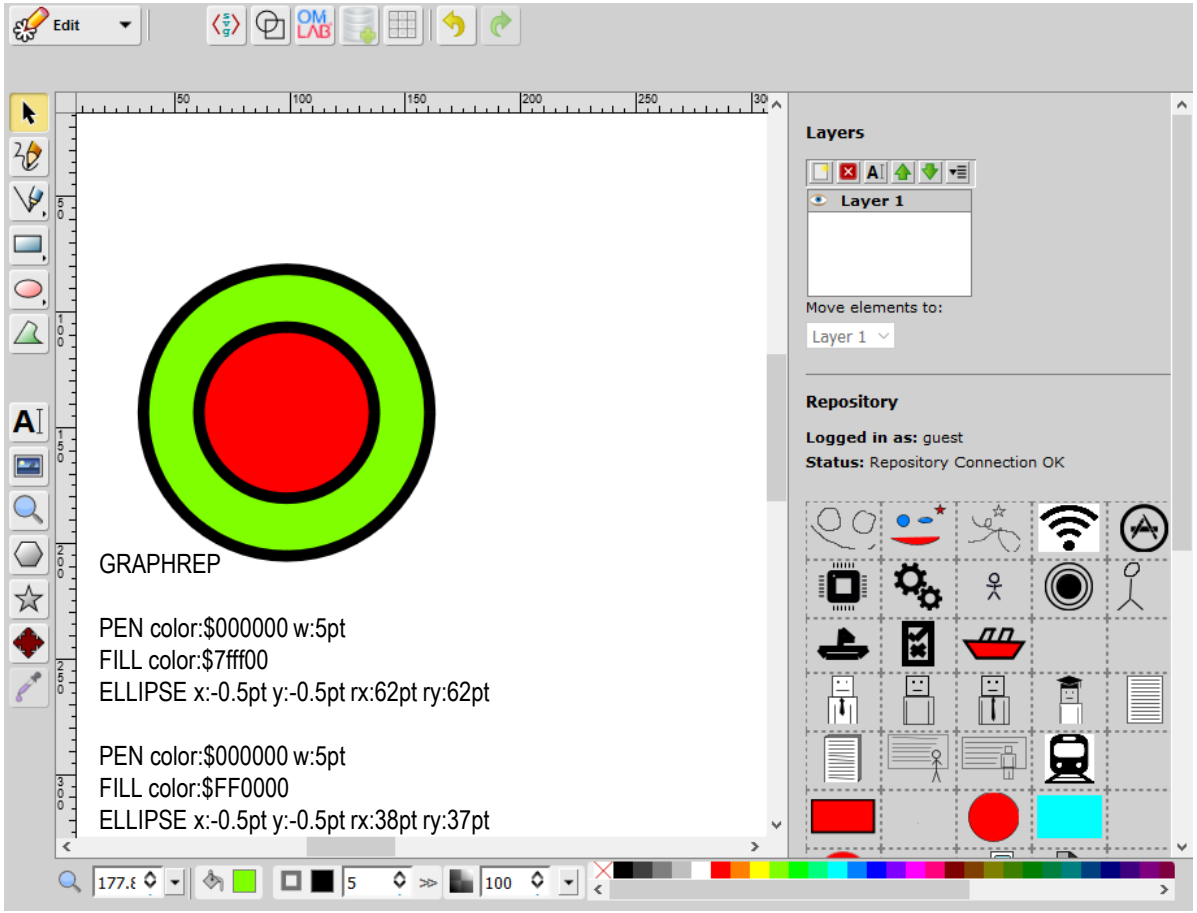
Platform Technologies

- GraphRep
- AdoScript

OPEN USE

Demonstration: Implementation of Object Visualisation

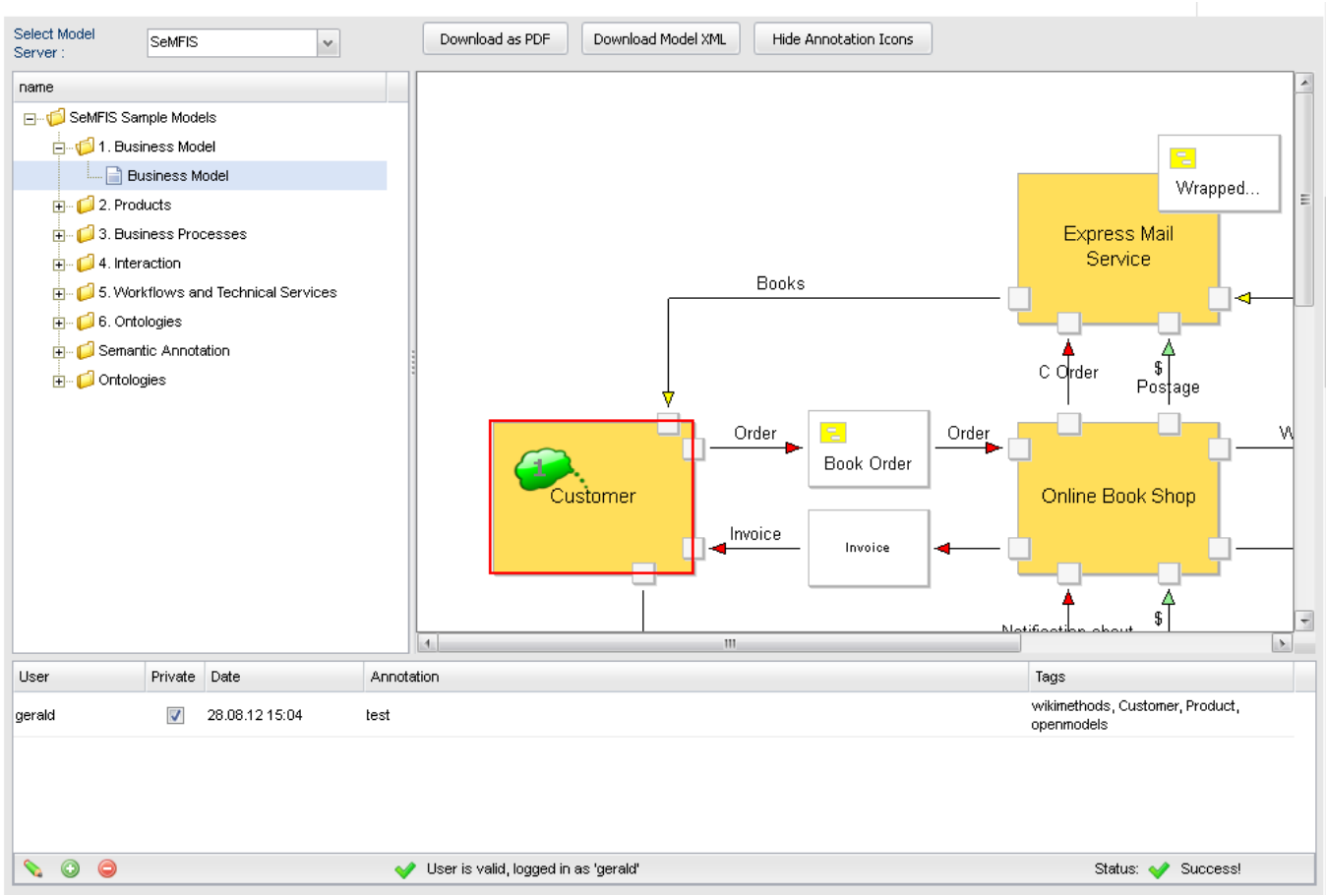
USE OMiLAB Development Tool



http://austria.omilab.org/psm/content/Graphrep/iframe?view=Developer-Online

Demonstration: Implementation of Object Visualisation

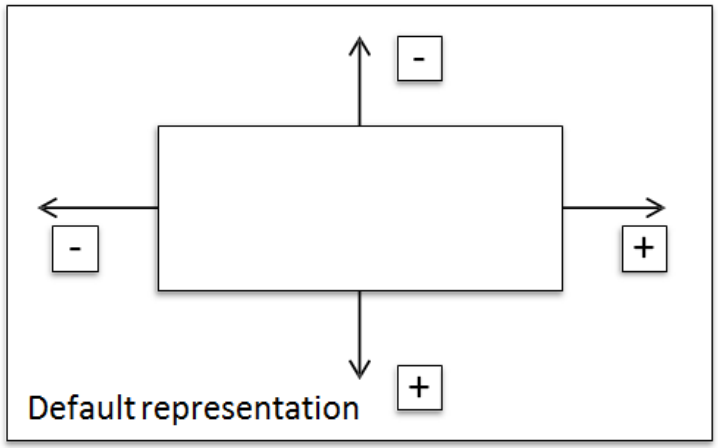
CONTRIBUTE to OMiLAB Developement

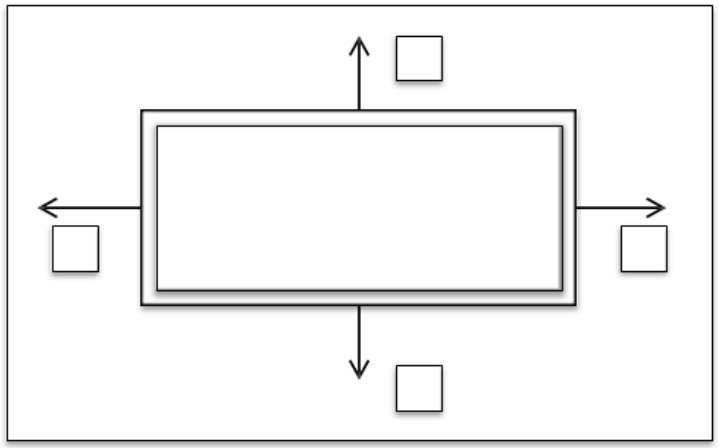


http://austria.omilab.org/psm/development

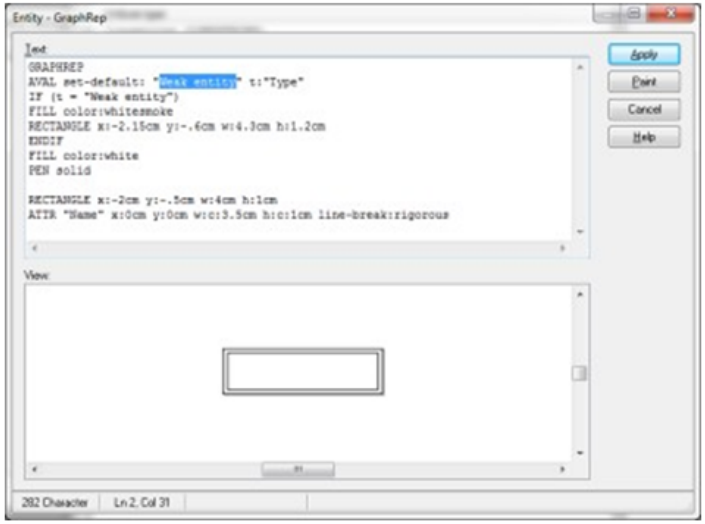
Demonstration: Implementation of Object Visualisation

DEVELOPMENT on ADOxx Platform

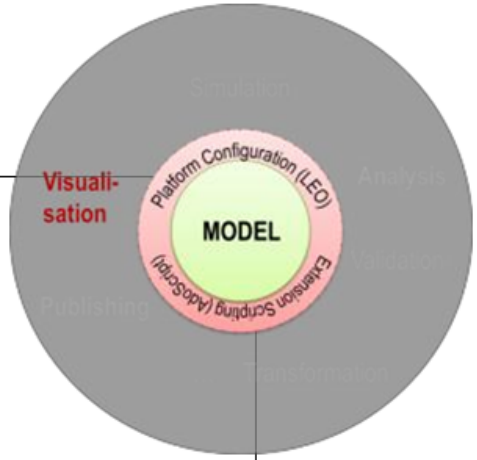




```
GRAPHREP
AVAL t:"Type"
IF (t = "Weak entity") (
  FILL color:whitesmoke
  RECTANGLE x:-2.15cm y:-.6cm w:4.3cm h:1.2cm
)                                  Conditional representation
FILL color:white                   Default representation
PEN solid

RECTANGLE x:-2cm y:-.5cm w:4cm h:1cm
ATTR "Name" x:0cm y:0cm w:c:3.5cm h:c:1cm line-
break:rigorous
                                   Name representation
```



Model Analysis Visualisation



Platform Functionality

- Object visualisation
- Model visualisation
  - Tabular view incl. view concept
  - Graphical view incl. view concept
  - Machine-generated models
    - Model analysis visualisation
    - Information visualisation
  - Human-generated models
    - Support functionality (automatic & user-defined)

OMiLAB Development Tools

- OMiLAB GraphRepGenerator
- AdoScript Syntax Highlighter

OPEN SOURCE

Platform Technologies

- GraphRep
- AdoScript

OPEN USE

Demonstration: Model Analysis Visualisation

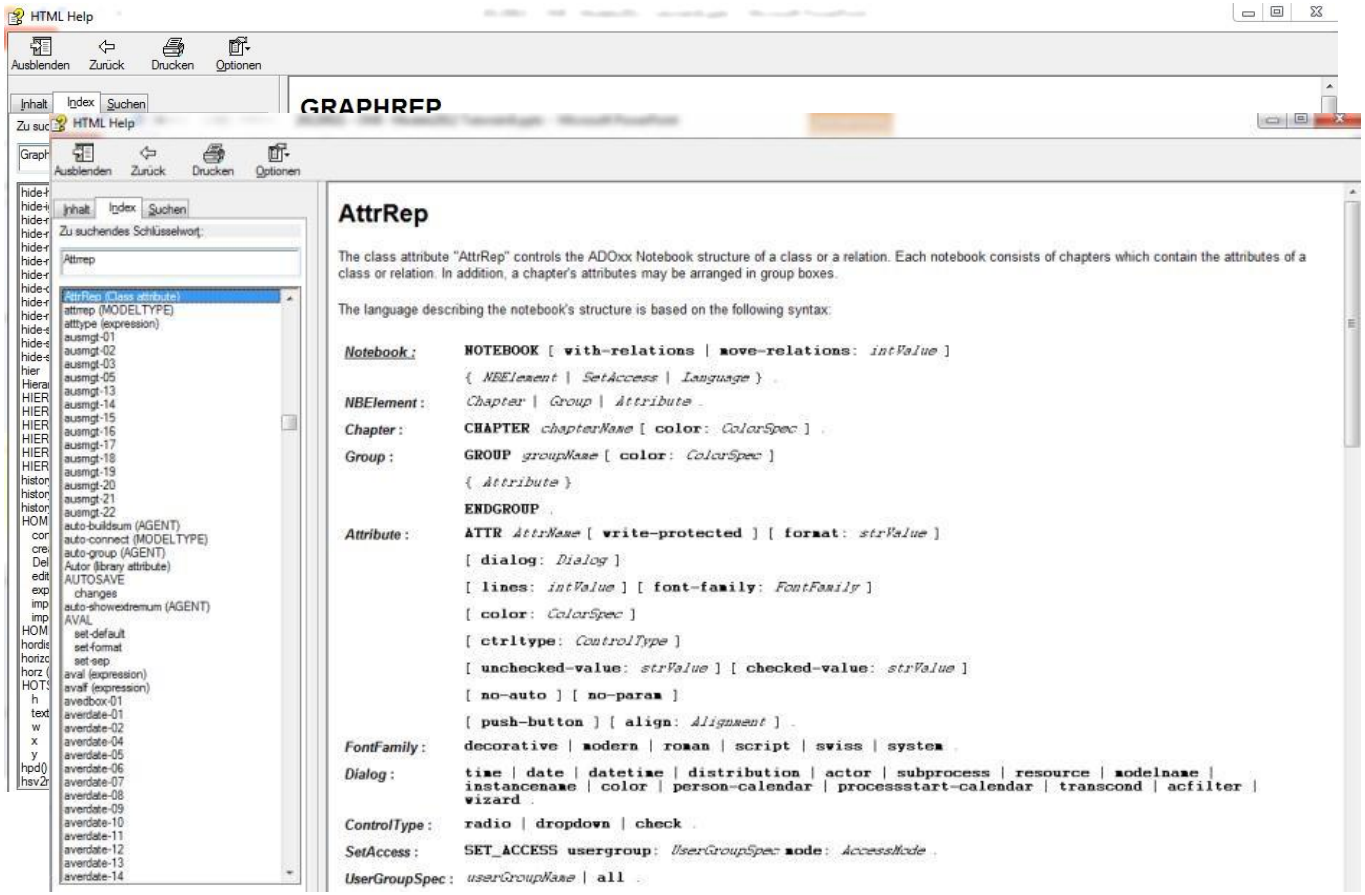
DEVELOPMENT on ADOxx Platform

```
## Get active M
CC "Modeling" G
SETL nStartmode

# make an info
into a string
CC "AdoScript"
   title:"Start

## count how ma

# get the id of
CC "Core" debug
SETL nClassID:(

#--------------
#--------------

# get all objec
CC "Core" debug
   classid:(nClassID)
```

```
IF (LEN(objids
  CC "AdoScrip
  EXIT
}
SETL debug nCo

CC "AdoScript"
   title:"Coun

## Creating a
CC "CoreUI" MO
   title:"Ziel
   Modellgrupp

# ------------
# ------------
# ------------
# ------------
# ------------
```

```
CC "Core" CREATE_MODEL modeltype:"Result-Type 1"
   modelname:"My First own result" version:"1.0"
   mgroups:(mgroupids)

# open the new created model AND to make the new
IF (ecode = 0) {
  CC "Modeling" CREATE_WINDOW_FOR_LOADED_MODEL mo
}

## Create objects in the new model

# get the model id of the new model
CC "Modeling" GET_ACT_MODEL
SETL nResultmodelID:(modelid)

# make an info box for debugging reasons - conver
into a string
CC "AdoScript" INFOBOX ("Hello " + STR(nResultmoc
   title:"Result model id!"
```

Summary: GraphRep & AttrRep Syntax



https://www.adoxx.org/live/adoxx-notation-language-graphrep
https://www.adoxx.org/live/adoxx-attribute-notation-language-attrrep

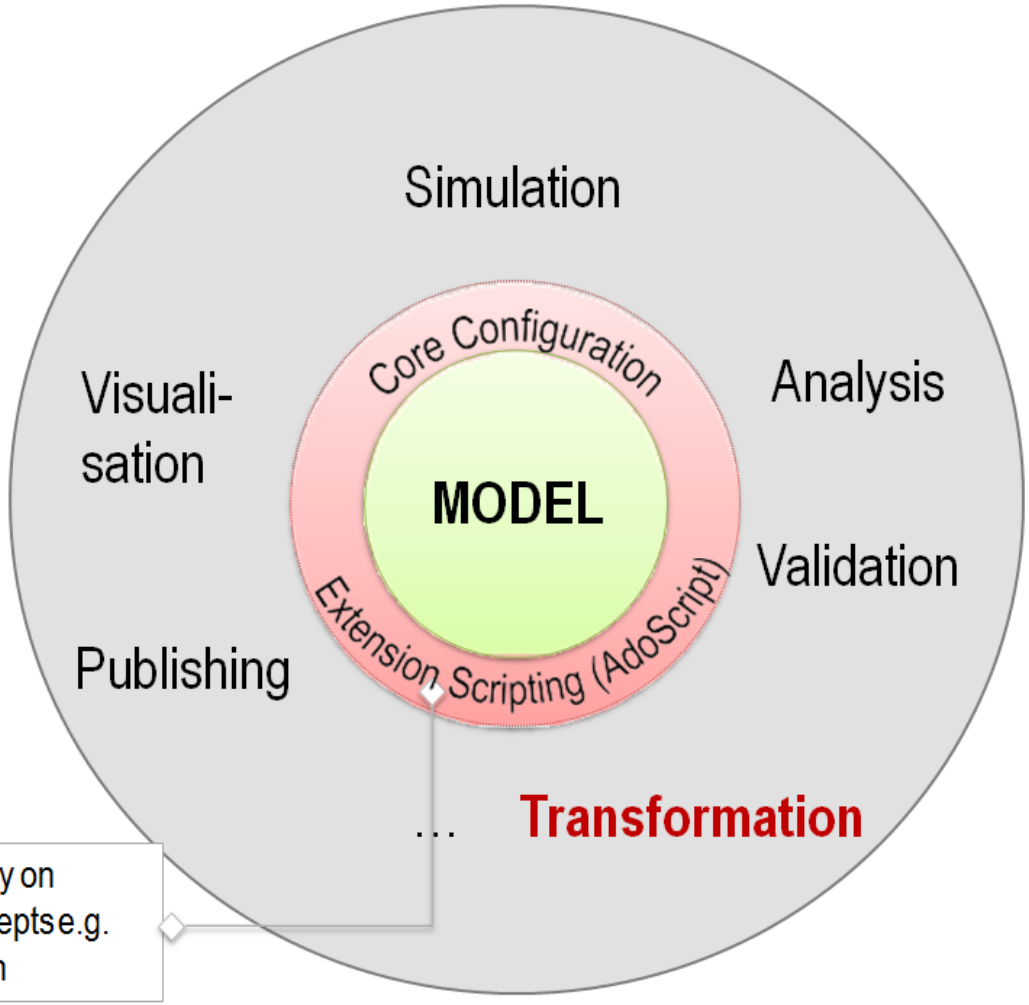
Wrap up: Visualisation Functionality

- Introduction of visualisation platform functionality
- Definition of tools and services to support development
- Technology Overview to support visualisation functionality

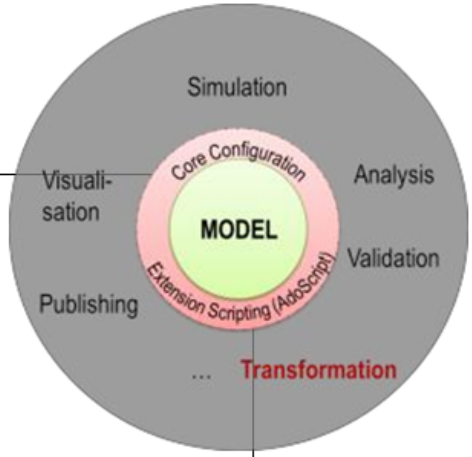
RESULT ACCOMPLISHED:

- Implemented Object Visualisation
- Implemented Script Functionality
- Modeltypes and View Definition
- Attribute Representation

Model Processing Functionality: Transformation



Model Transformation



Platform Functionality

- Objecttransformation
- Modeltransformation
  - SAME PLATFORM
  - DIFFERENT PLATFORM
    - Inbound Interfaces
    - Outbound Interfaces

OMiLAB Development Tools

- Model Publishing Engine
- AdoScript Syntax Highlighter

OPEN SOURCE

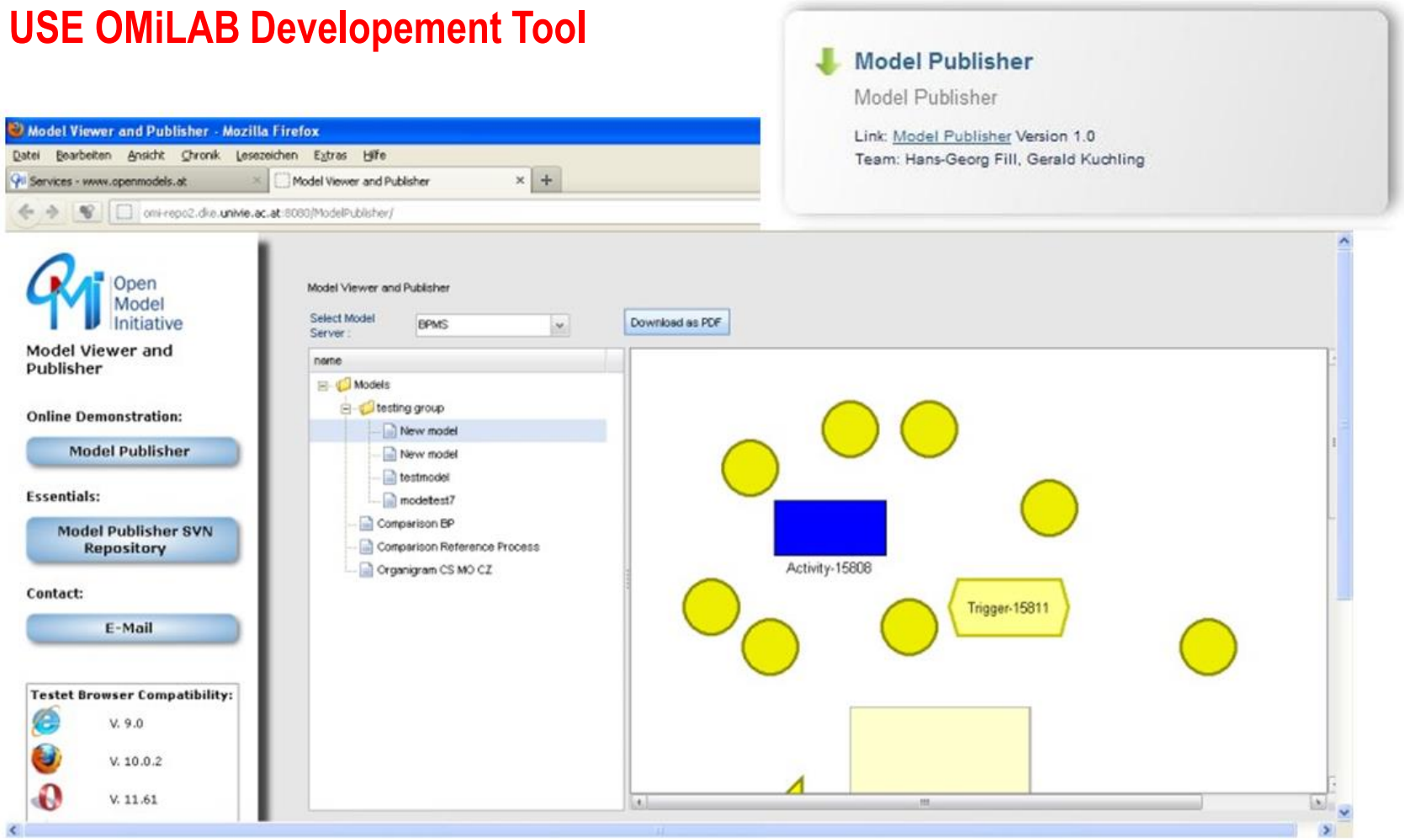
Platform Technologies

- Platform configuration
- AdoScript

OPEN USE

Demonstration: Implementation of Model Transformation

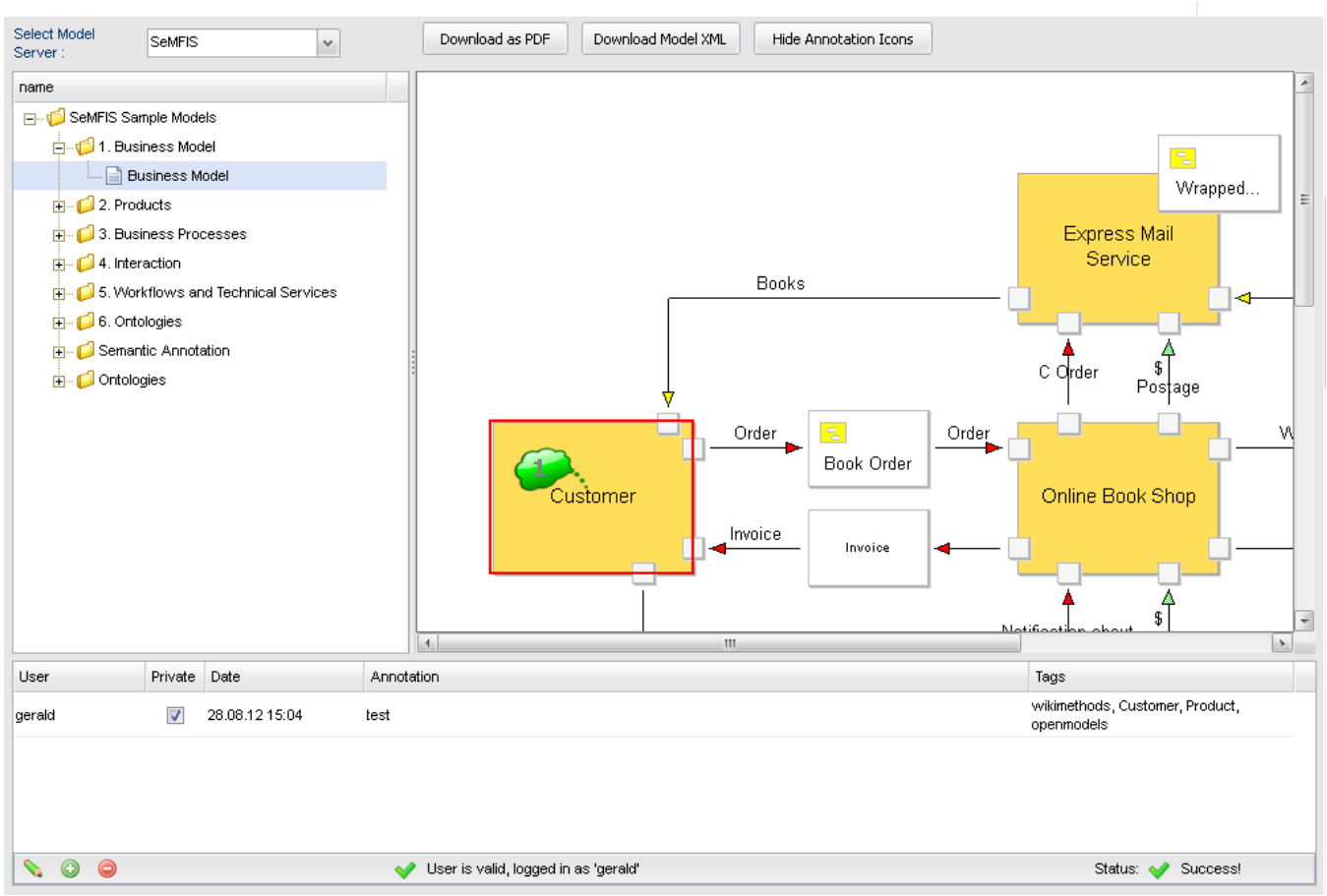
USE OMiLAB Developement Tool



http://austria.omilab.org/psm/development

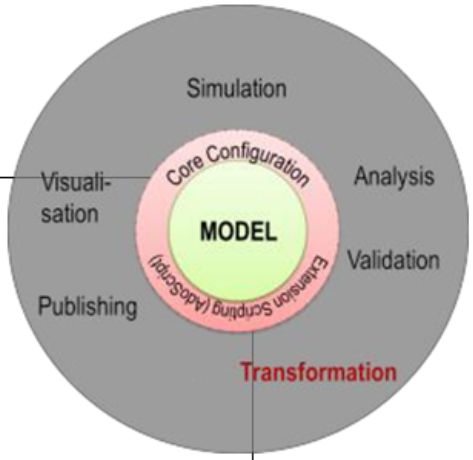
Demonstration: Implementation of Model Transformation

CONTRIBUTE to OMiLAB Developement



http://austria.omilab.org/psm/development

Model Transformation (Different Platform)



Platform Functionality

- Objecttransformation
- Modeltransformation
  - SAME PLATFORM
  - DIFFERENT PLATFORM
    - Inbound Interfaces
    - Outbound Interfaces

OMiLAB Development Tools

- Model Publishing Engine
- AdoScript Syntax Highlighter

OPEN SOURCE

Platform Technologies

- Platform configuration
- AdoScript

OPEN USE

Demonstration: Core Functionality for Serialisation as XML and ADL

USE functionality on ADOxx Platform

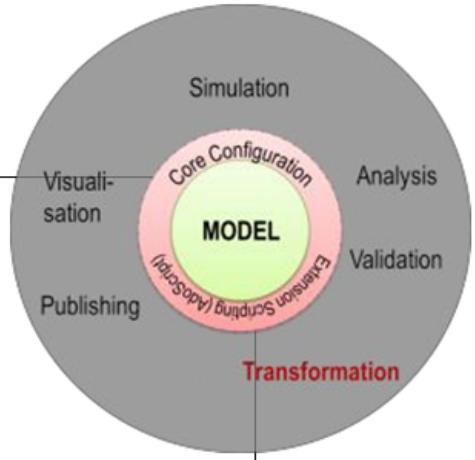
XML Export Sample

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE ADOXML [View Source for full doctype...]>
<ADOXML version="3.1" date="28.06.2012" time="13:32" database="adoxx13" username="sample1" adoversion="Vers
- <MODELS>
  - <MODEL id="mod.13813" name="model-1" version="1.1" modeltype="Sample" libtype="bp" applib="ADOxx 1.3 Dyn
    + <MODELATTRIBUTES>
    - <INSTANCE id="obj.13814" class="E" name="E1">
        <ATTRIBUTE name="Position" type="STRING">NODE x:4cm y:11cm w:2cm h:2cm index:1</ATTRIBUTE>
        <ATTRIBUTE name="External tool coupling" type="STRING" />
        <ATTRIBUTE name="a1" type="INTEGER">0</ATTRIBUTE>
        <RECORD name="a2" />
        <ATTRIBUTE name="a3" type="STRING" />
        <ATTRIBUTE name="b1" type="INTEGER">0</ATTRIBUTE>
        <RECORD name="b2" />
        <ATTRIBUTE name="b3" type="STRING" />
        <ATTRIBUTE name="e1" type="INTEGER">0</ATTRIBUTE>
        <RECORD name="e2" />
        <ATTRIBUTE name="e3" type="STRING">11</ATTRIBUTE>
        <ATTRIBUTE name="a4" type="INTEGER">0</ATTRIBUTE>
        <ATTRIBUTE name="b4" type="STRING" />
      </INSTANCE>
    + <INSTANCE id="obj.13817" class="A" name="A1">
    + <INSTANCE id="obj.13826" class="B" name="B1">
    + <INSTANCE id="obj.13832" class="C" name="C-13010">
    + <INSTANCE id="obj.13835" class="D" name="D-13013">
    + <INSTANCE id="obj.16408" class="B" name="B-16408">
    + <INSTANCE id="obj.16604" class="V" name="V1">
    + <INSTANCE id="obj.17004" class="W" name="W1">
    + <INSTANCE id="obj.17007" class="B" name="B-16408-17007">
    + <INSTANCE id="obj.17291" class="E" name="E-17291">
    + <INSTANCE id="obj.17294" class="E" name="E-17294">
    + <INSTANCE id="obj.17297" class="E" name="E-17297">
    + <INSTANCE id="obj.17328" class="E" name="D-13013-17321">
    + <INSTANCE id="obj.17334" class="E" name="C-13010-17318">
    + <CONNECTOR id="con.13841" class="aRb">
    + <CONNECTOR id="con.13842" class="aRb">
    + <CONNECTOR id="con.13843" class="aRb">
    + <CONNECTOR id="con.13844" class="aRb">
    + <CONNECTOR id="con.13845" class="aRb">
    + <CONNECTOR id="con.16607" class="Is inside">
    </MODEL>
  </MODELS>
</ADOXML>
```

ADL Export Sample

```
INSTANCE <E1> : <E>
    ATTRIBUTE <Position>
    VALUE "NODE x:4cm y:11cm w:2cm h:2cm index:1"
    ATTRIBUTE <External tool coupling>
    VALUE ""
    ATTRIBUTE <a1>
    VALUE 0
    ATTRIBUTE <a2>
    VALUE
    ATTRIBUTE <a3>
    VALUE ""
    ATTRIBUTE <b1>
    VALUE 0
    ATTRIBUTE <b2>
    VALUE
    ATTRIBUTE <b3>
    VALUE ""
    ATTRIBUTE <e1>
    VALUE 0
    ATTRIBUTE <e2>
    VALUE
```

Model Transformation (Same Platform)



Platform Functionality

- Objecttransformation
- Modeltransformation
  - SAME PLATFORM
  - DIFFERENT PLATFORM
    - Inbound Interfaces
    - Outbound Interfaces

OMiLAB Development Tools

- Model Publishing Engine
- AdoScript Syntax Highlighter

OPEN SOURCE

Platform Technologies

- Platform configuration
- AdoScript

OPEN USE

Demonstration: Implementation of Model Transformation 1

DEVELOP on ADOxx Platform

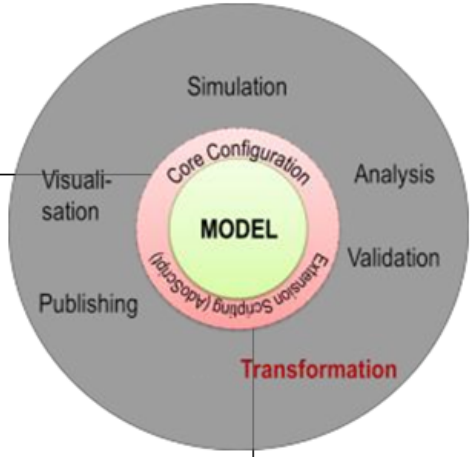
```
## Open Model
CC "Modeling" GET_ACT_MODEL
SETL nSourceModelID:(modelid)

SETL sClassnameSource:("A")
SETL sClassnameTarget:("E")

# BEGIN set new model
CC "CoreUI" MODEL_SELECT_BOX mgroup-sel without-models
   title:"Zielmodellgruppe"
   boxtext:"Selektieren Sie die Ziel-Modellgruppe in der Datenbank:"

CC "Core" CREATE_MODEL modeltype:"Sample"
   modelname:"My First sample"
   version:"1.0"
   mgroups:(mgroupids)
SETL nTargetModelID:(modelid)

# END set new model

CC "Core" GET_ALL_OBJS_OF_CLASSNAME modelid:(nSourceModelID)
classname:(sClassnameSource)
SETL sObjIDs:(objids)
```

Demonstration: Implementation of Model Transformation 2

DEVELOP on ADOxx Platform

```
# BEGIN set x, y pos
SETL nXpos:(5)
SETL nYpos:(5)
FOR i in:(sObjIDs) {
  # get class ID from class name
  CC "Core" GET_CLASS_ID classname:(sClassnameSource)

  # get the attribute "Name"
  CC "Core" GET_ATTR_ID classid:(classid) attrname:("Name")

  # and show it
  CC "Core" GET_ATTR_VAL objid:(VAL (i)) attrid:(attrid)
  CC "AdoScript" INFOBOX (val)
  SETL sAttrName:(val)

  #Make new model
  CC "Core" GET_CLASS_ID classname:(sClassnameTarget)
  SETL nClassTargetID:(classid)
  CC "Core" debug CREATE_OBJ modelid:(nTargetModelID) classid:(nClassTargetID)
     objname:(sAttrName)
  CC "Core" debug SET_ATTR_VAL objid:(objid) attrname:("Position") val:("x:"
     + STR (nXpos) +"cm y:" + STR (nYpos) + "cm")
  SETL nXpos:(nXpos + 2.5)
}
```

Object Transformation



Platform Functionality

- Objecttransformation
- Modeltransformation
  - SAME PLATFORM
  - DIFFERENT PLATFORM
    - Inbound Interfaces
    - Outbound Interfaces

OMiLAB Development Tools

- Model Publishing Engine
- AdoScript Syntax Highlighter

OPEN SOURCE

Platform Technologies

- Platform configuration
- AdoScript

OPEN USE

Objecttransformation using CONVERSION

DEVELOP on ADOxx Platform

- If you define __*Conversion*__ for the class „A" with

  CLASS „B"
  ATTR „ba1"
  ATTR „ba2" from: „aa3"

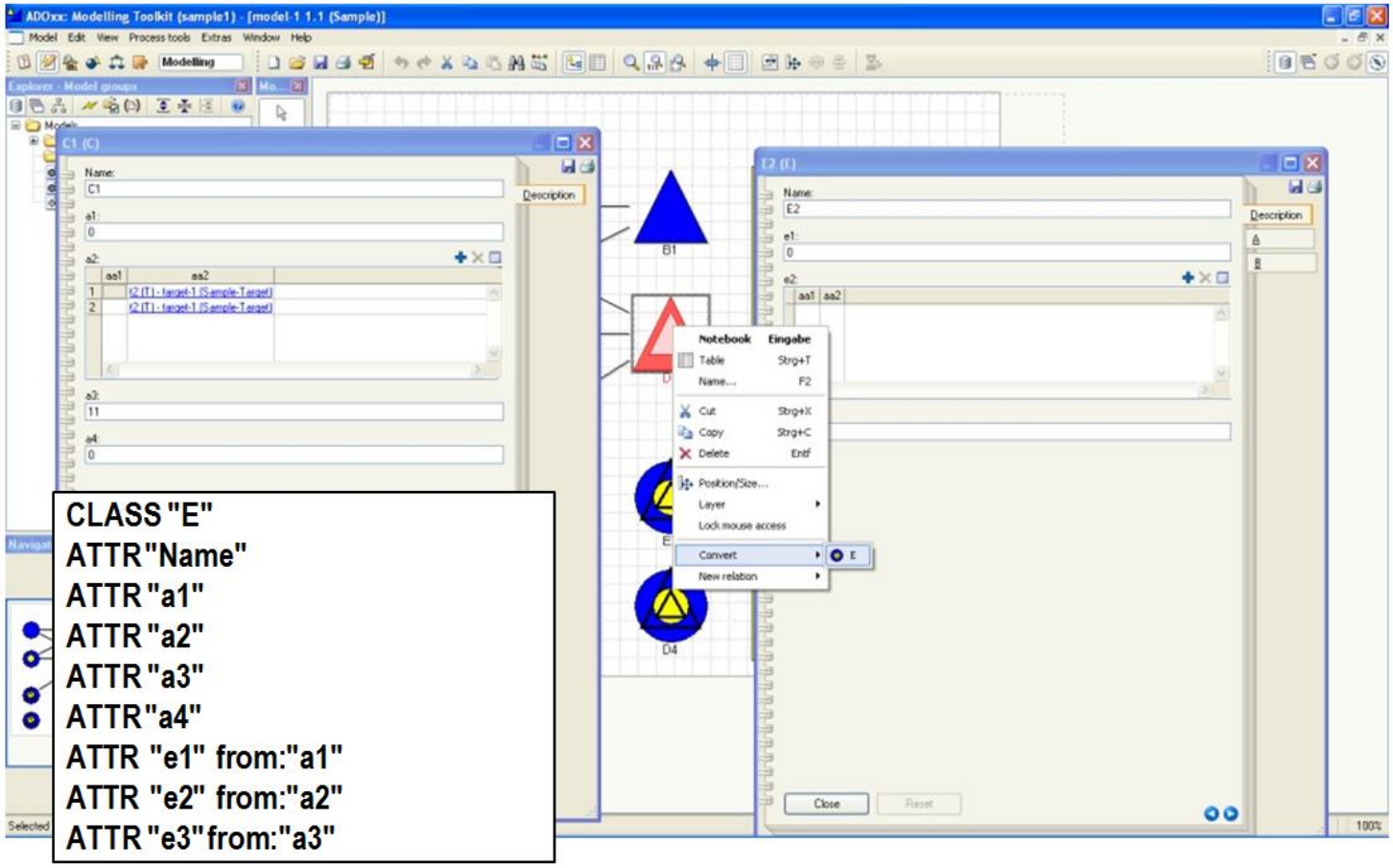
Conversion :	{ *ClassConversion* } .
ClassConversion :	**CLASS** *className* { *AttrConversion* } .
AttrConversion :	**ATTR** *attrName* [**from:***attrName*] .

- this means that
  - objects of class „A" can be converted to objects of class „B",
  - the aa1 is assigned from A to ba1 in B as the have the same name,
  - the aa3 from A is assigned to Ba2 from B as they have different names,

Demonstration: Objecttransformation

DEVELOP on ADOxx Platform

Instances of C -> E



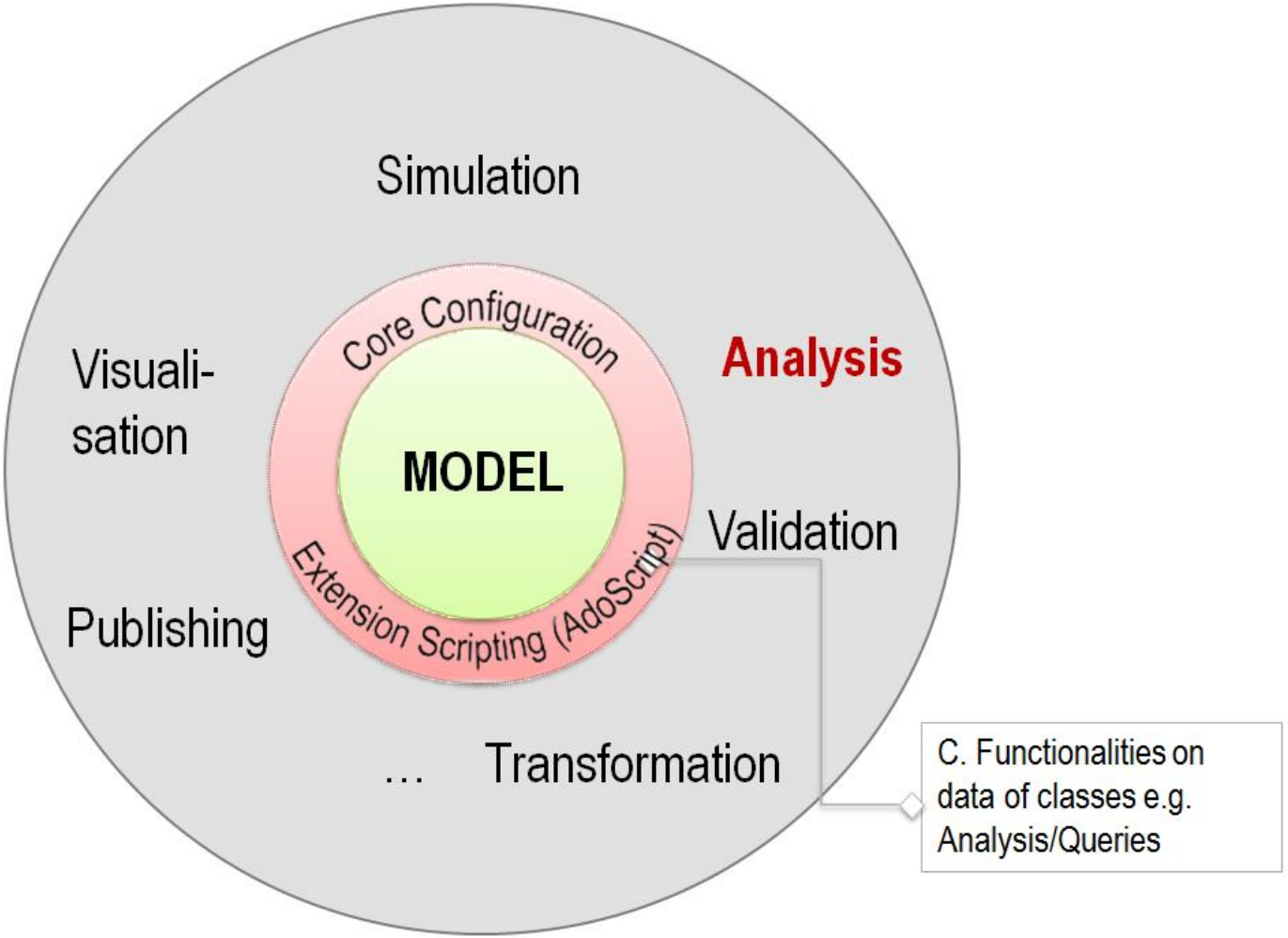
Wrap up: Transformation Functionality

- Use the basic transformation mechanisms to use it for publishing
- Use scripting mechanisms for model transformation

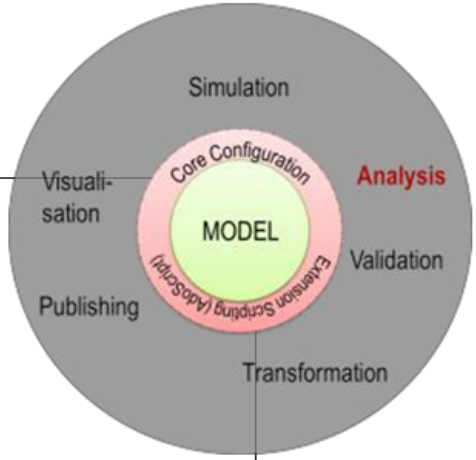
RESULT ACCOMPLISHED:

- Publishing example using the OMiLAB service
- Transformation of scripts

Model Processing Functionality: Analysis



Query of Model Content



Platform Functionality

- Query of model content
- Quality validation
- Consistency checks
- Population through analysis

OMiLAB Development Tools

- ReSTAPI

OPEN SOURCE

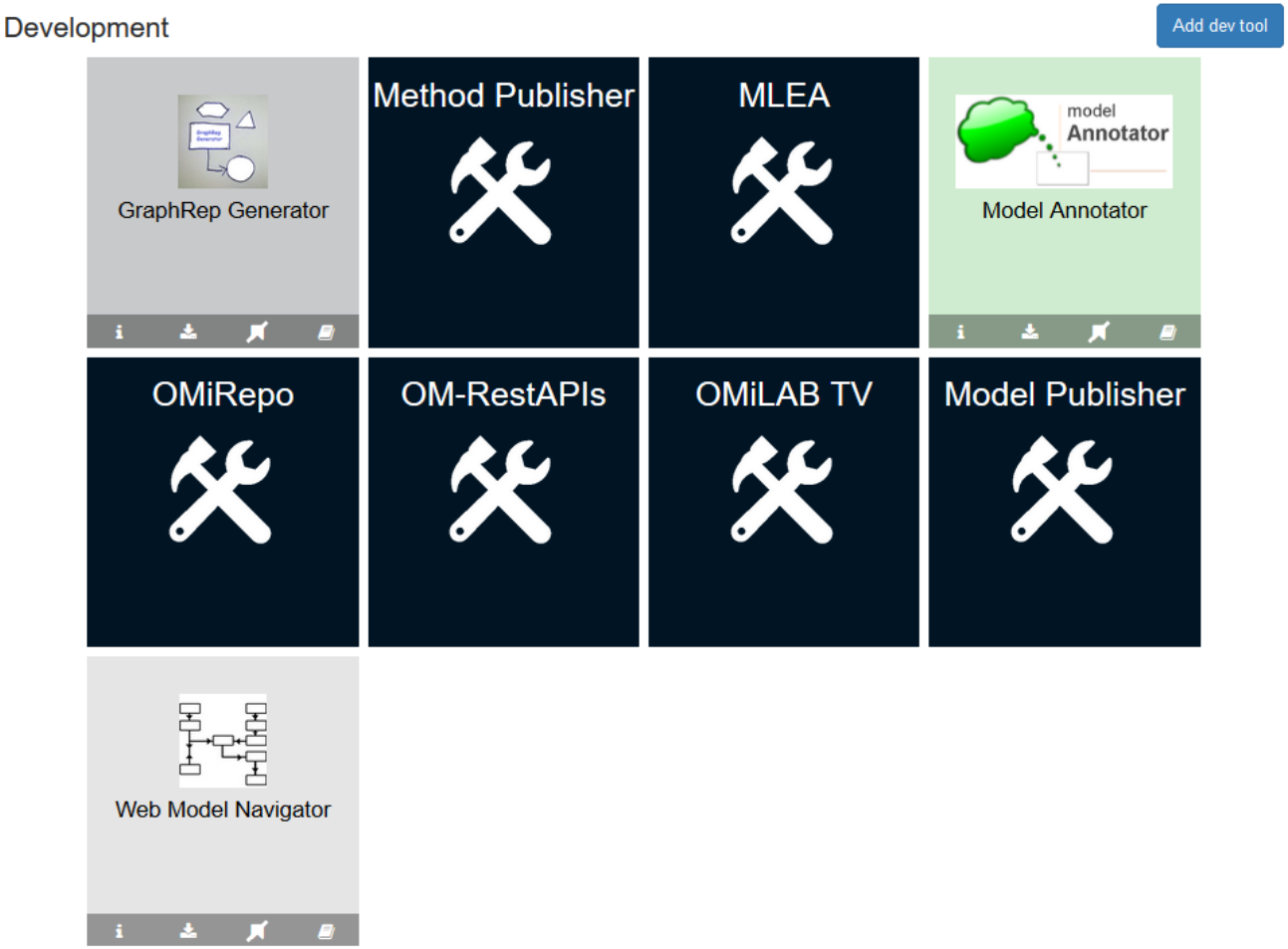
Platform Technologies

- AQL
- AdoScript

OPEN USE

Demonstration: Analysis Functionality

CONTRIBUTE to OMiLAB Developement



http://austria.omilab.org/psm/development

Demonstration: Analysis Functionality

DEVELOP on ADOxx Platform

Example 1: Get all objects of class "A" in a certain model

```
CC "Modeling" GET_ACT_MODEL
#-->RESULT modelid:intValue
CC "AQL" EVAL_AQL_EXPRESSION expr:"<\"A\">" modelid:(modelid)
IF (ecode = 0) {
  CC "AdoScript" INFOBOX ("Found objects: " + objids)
}
ELSE {
  CC "AdoScript" INFOBOX "An error has occured!"
}
```

Example 2: Get all models of modeltype "Sample"

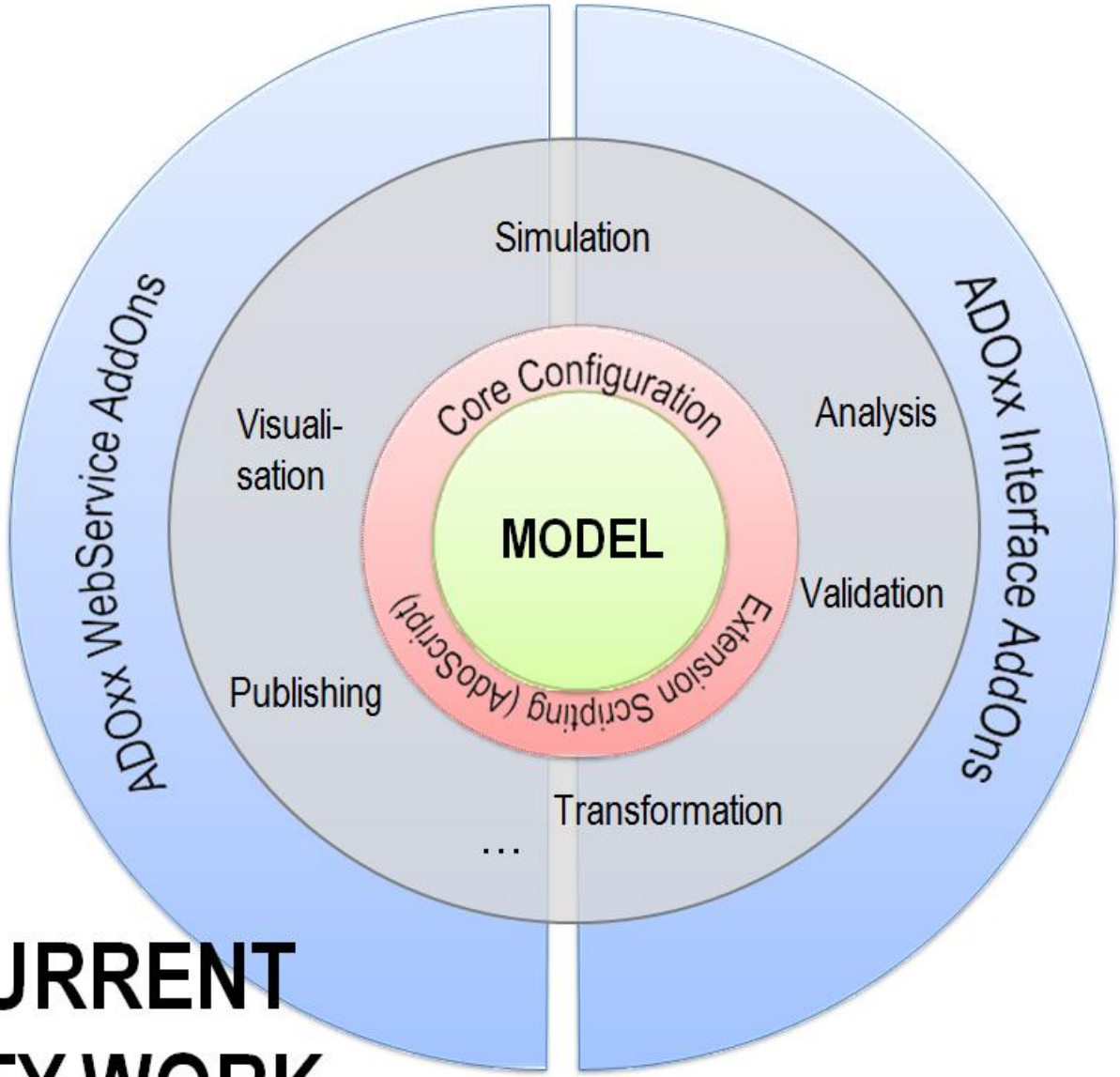
```
CC "AQL" EVAL_AQL_EXPRESSION expr:"<\"Sample\">" modelscope
IF (ecode = 0) {
  CC "AdoScript" INFOBOX ("Found models: " + objids)
}
ELSE {
  CC "AdoScript" INFOBOX "An error has occured!"
}
```

Wrap up: Analysis Functionality

- ReST API for model analysis
- Script <-> AQL combination to run analysis

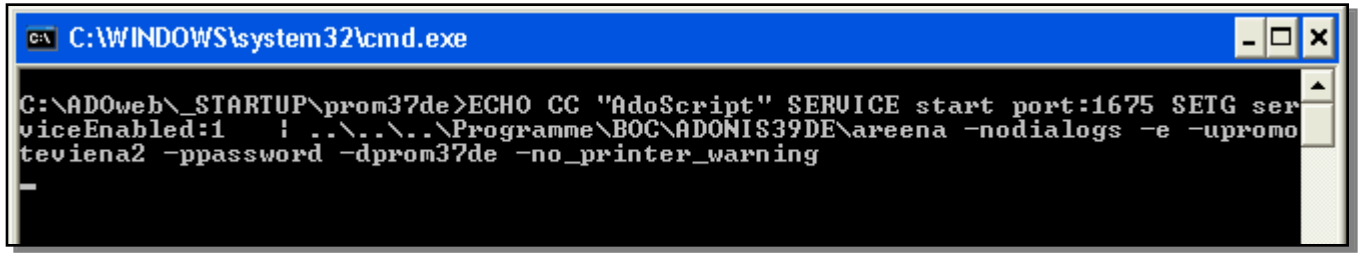
RESULT ACCOMPLISHED:

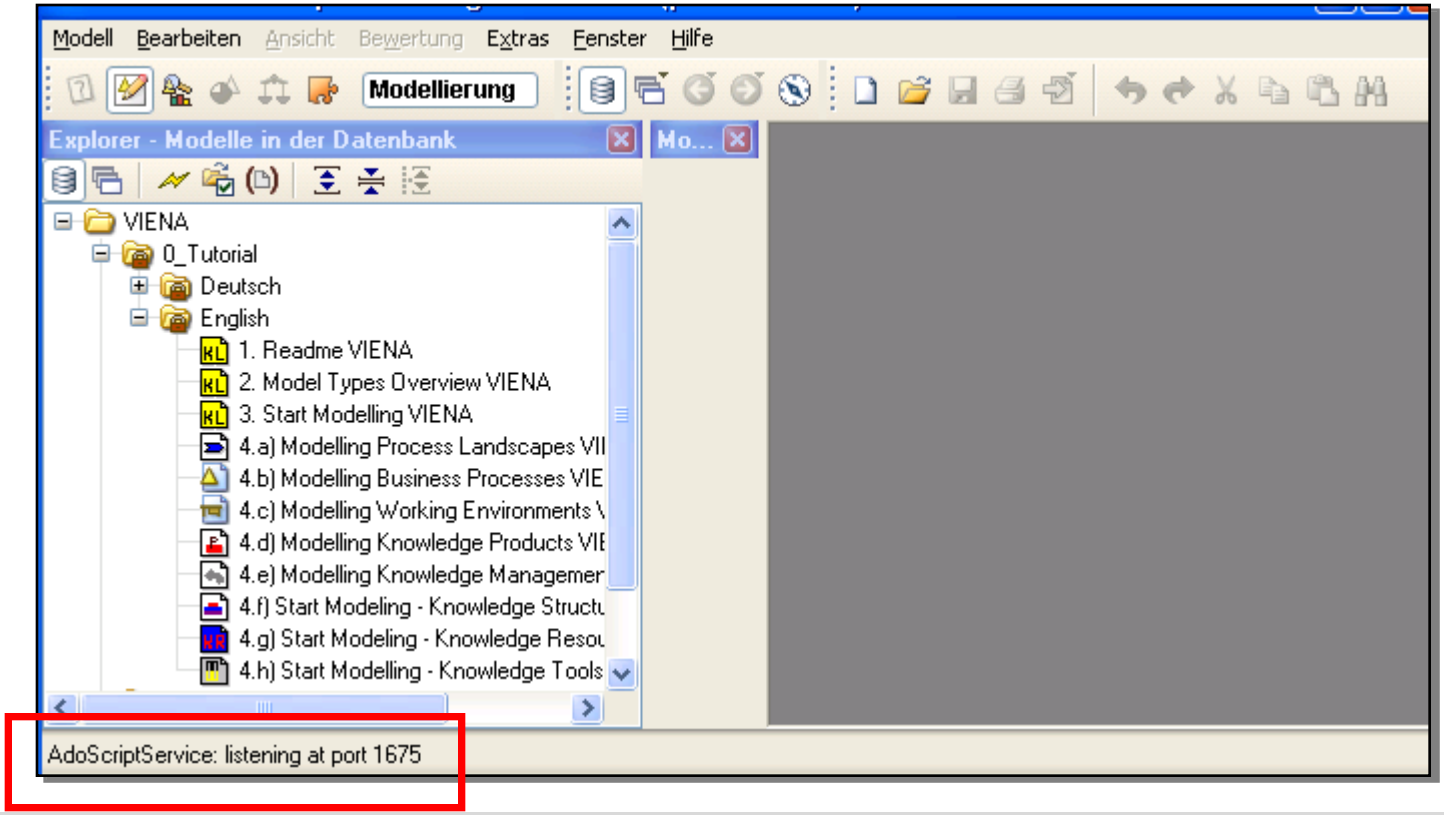
- Implemented API integration with demonstration environment
- AQL queries in script



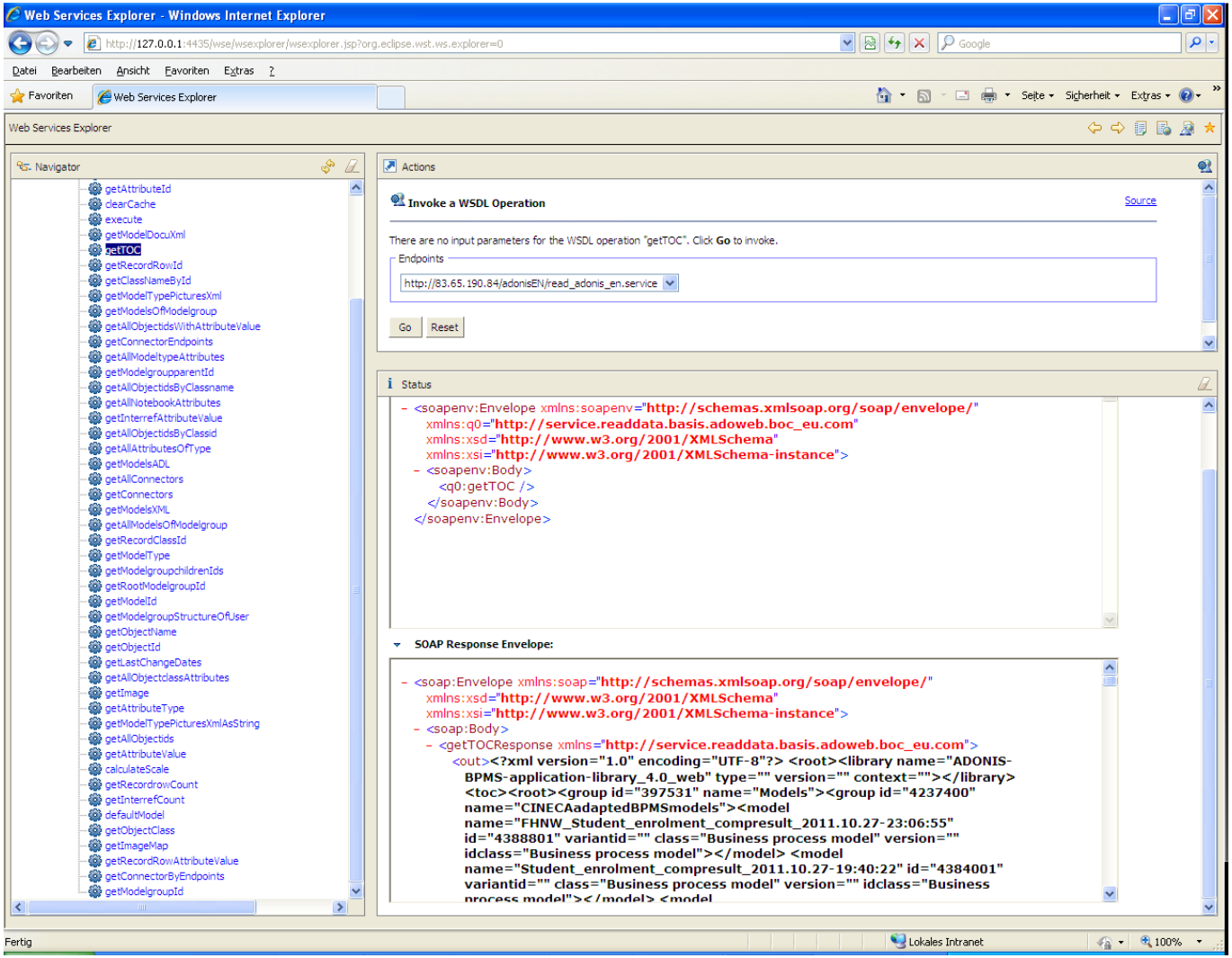
ADOXX SELECT CURRENT COMMUNITY WORK

ADOxx WebService Port

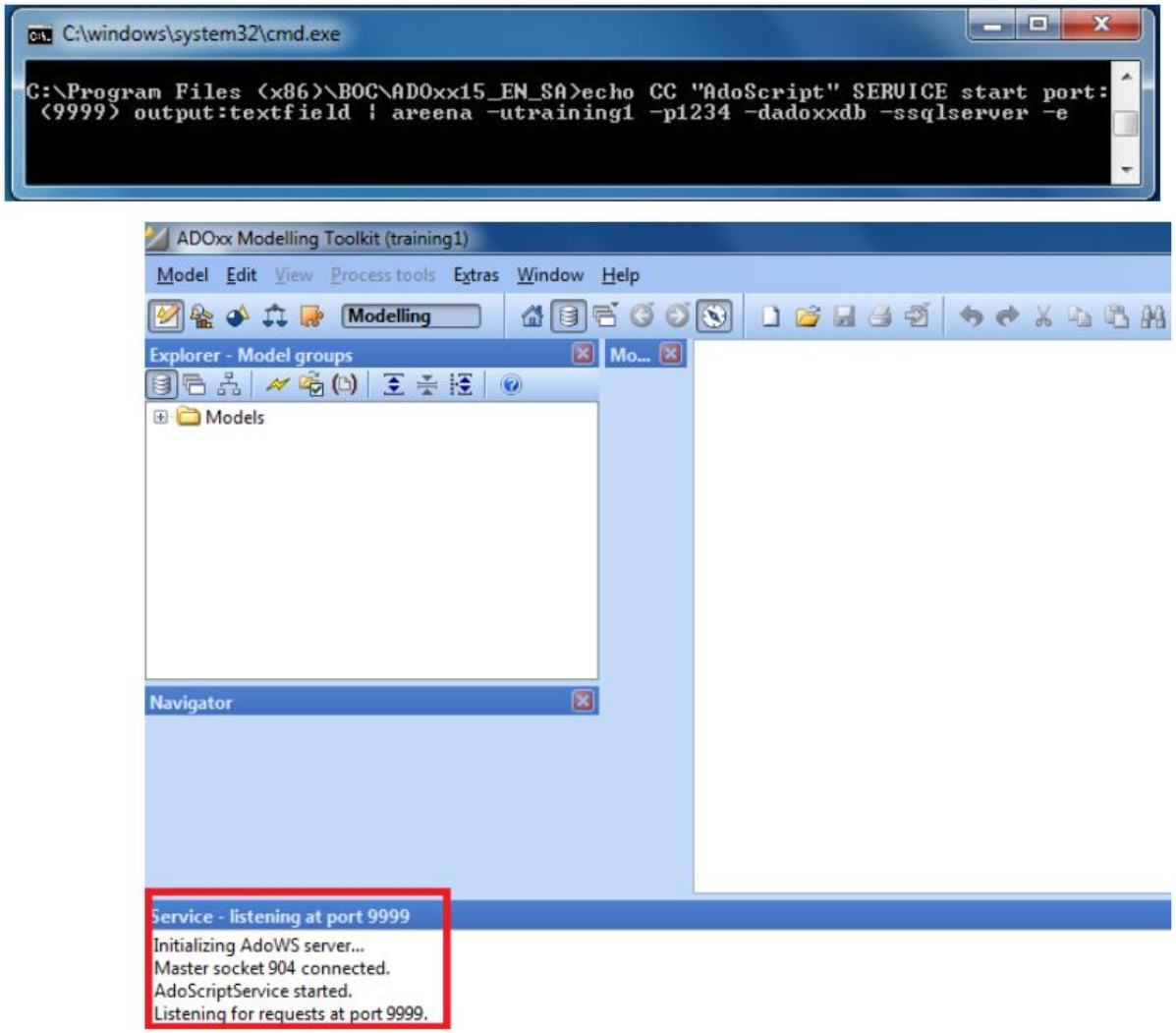




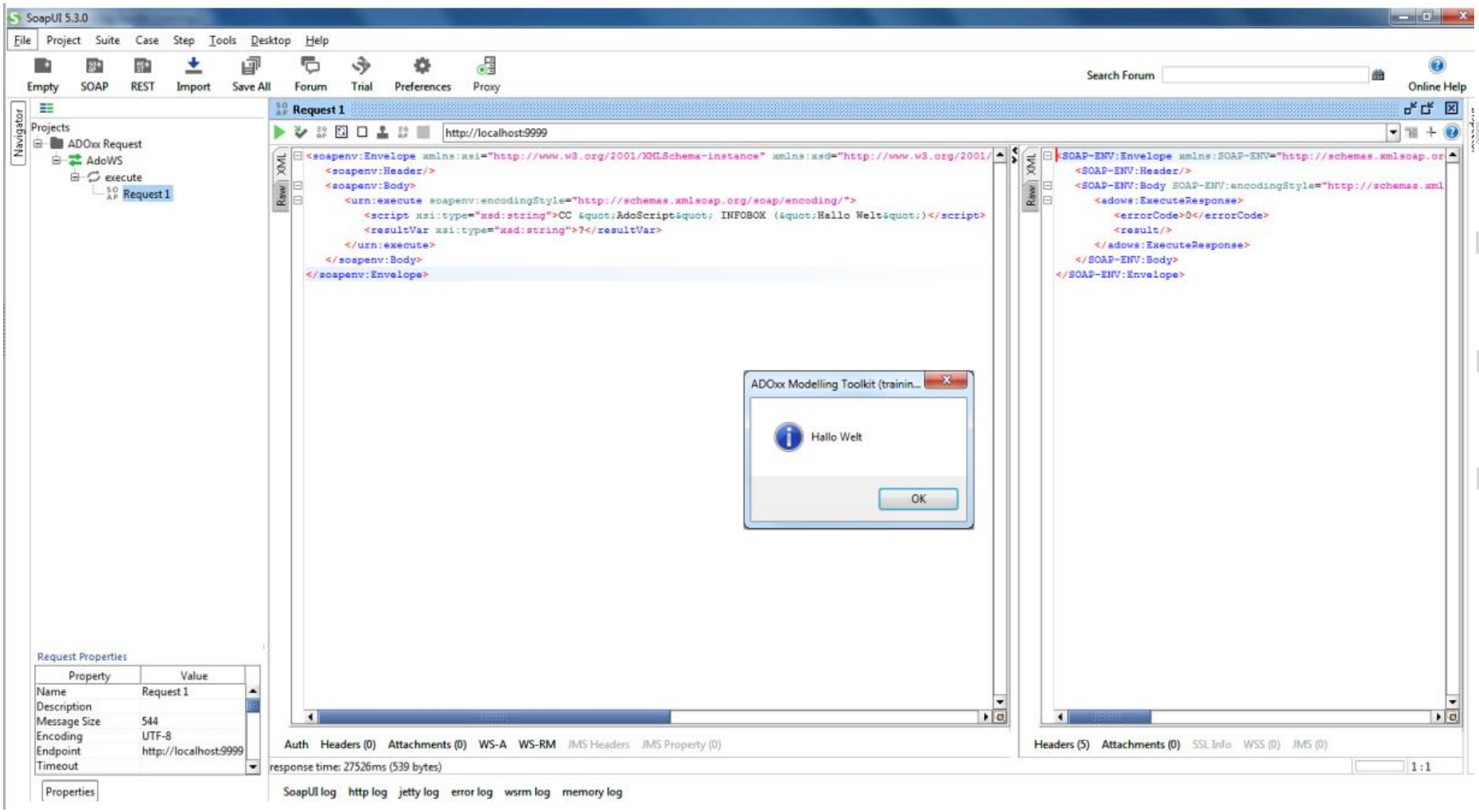
ADOxx WebService Interaction



ADOxx WebService: Example using SoapUI (1)



ADOxx WebService: Example using SoapUI (2)



CONCLUSION

The Method Conceptualisation Process ...

- Capturing of fundamental concepts, relationships in between and properties adhering to them, usually obtained through the analysis of a selected domain.

- Description of such conceptualisations varies depending on the addressed audience, with different expectations, like End User, Modeller, Developer, ….

- From a development perspective, a method conceptualization needs to be formal enough to enable developer continue along the life-cycle

- A model of the method (language)
  that facilitates a coherent view on the core concepts involved

...results in a Modelling Method Tool

- When the realisation of a modelling method is expected to result in an application software/tool, a domain expert's (i.e., method developer) viewpoint needs to be "augmented" with the viewpoint of a software developer (i.e., method engineer).

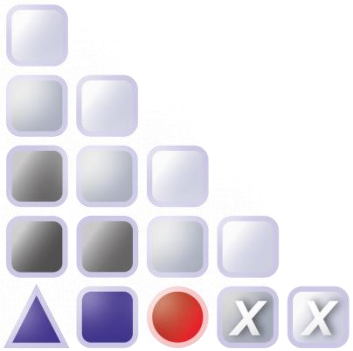
- Typically, a method developer rarely considers design, implementation or deployment of relevant artefacts when "conceptualising" a modelling method.

- A method engineer on the other hand is usually not an expert in the domain that is addressed by a certain modelling method.

Further Questions?

OMiLAB®

www.omilab.org



www.adoxx.org

tutorial@adoxx.org

REFERENCES

OMiLAB References on the ADOxx Meta Modelling Approach

- Fill, H.-G., T. Redmond, et al. (2012). FDMM: A Formalism for Describing ADOxx Meta Models and Models. Proceedings of ICEIS 2012, Wroclaw, Poland. L. Maciaszek, A. Cuzzocrea and J. Cordeiro. 3: 133-144.
- Junginger, S., H. Kühn, et al. (2000). "Ein Geschäftsprozeßmanagement-Werkzeug der nächsten Generation - ADONIS: Konzeption und Anwendungen (German: ADONIS: A next generation business process management tool - Concepts and Applications)." Wirtschaftsinformatik 42(5): 392-401.
- Karagiannis, D., Visic, N. (2011). Next Generation of Modelling Platforms, BIR Conference 2011, Riga, Latvia, 6.
- Karagiannis D. (2012). Modelling Aspects for Next-Generation Modelling Systems. Presentation at FInES "Translating Knowledge Into Growth: Views from ICT Research to Support Future Business Innovation" - Panel 3 Next Generation Enterprise Systems - Characteristics, Properties, and Architectural Design Principles. Aalborg May 9, 2012. URL: http://de.slideshare.net/FInESCluster/p3-2dimitris-karagiannisv2
- Kühn, H. and S. Junginger (1999). An Approach to use UML for Business Process Modeling and Simulation in ADONIS. Proceedings of the 13th European Simulation Multiconference (ESM 99) - Modeling and Simulation: A Tool for the Next Millenium. H. Szczerbicka. Warsaw, Poland: 634-639.
- Kühn H. (2004). Methodenintegration im Business Engineering, PhD Thesis, University of Vienna, April 2004.
- Schwab, M., D. Karagiannis, et al. (2010). i* on ADOxx(R): A Case Study. Proceedings of the 4th International i* Workshop - iStar10 - CAiSE Workshop Proceedings, Springer.
- Utz W., Woitsch, R. and Karagiannis D. (2011): Conceptualisation of Hybrid Service Models: An Open Models Approach, The 5th International IEEE Workshop on Requirements Engineering for Services (REFS 2011), Munich, IEEE
- Wende, C., Assmann, U., Zivkovic, S., and Kühn, H. (2011). Feature-based Customisation of Tool Environments for Model-Driven Software Development. In: Almeida, E., Kishi, T., Schwanninger, C., John, I., and Schmid, K. (Eds.): Proceedings of the 15th International Software Product Lines Conference (SPLC 2011), Munich, August 21-26, 2011, IEEE Computer Society, pp. 45-54.
- Zivkovic, S.; Kühn, H.; Murzek, M.: An Architecture of Ontology-aware Metamodelling Platforms for Advanced Enterprise Repositories. In: Camp, O.; Hammoudi, S.; (Eds.): Proceedings of the 1st International Workshop on Advanced Enterprise Repositories (AER 2009), Milano, Italy, May 6th, 2009. pp. 95-104.

OMiLAB References on General Aspects of Modelling and Modelling Methods

- Bork, D. and E. Sinz (2010). Design of a SOM Business Process Modelling Tool based on the ADOxx meta-modelling Platform. Proceedings of the Fourth International Workshop on Graph-Based Tools, EASST.
- Fill, H.-G. (2009). Visualisation for Semantic Information Systems, Gabler.
- Fill, H.-G. (2011). On the Conceptualization of a Modeling Language for Semantic Model Annotations. Advanced Information Systems Engineering Workshops, CAiSE 2011. C. Salinesi and O. Pastor. London, UK, Springer. LNBIP Vol. 83: 134-148.
- Fill, H.-G. (2012). An Approach for Analyzing the Effects of Risks on Business Processes Using Semantic Annotations. European Conference on Information Systems 2012, AIS.
- Fill, H.-G. (2012a). SeMFIS: A Tool for Managing Semantic Conceptual Models. Workshop on Graphical Modeling Language Development, Kgs. Lyngby, Denmark.
- Karagiannis, D. and H. Kühn (2002). Metamodeling Platforms. Third International Conference EC-Web 2002 – Dexa 2002. K. Bauknecht, A. Min Tjoa and G. Quirchmayr. Aix-en-Provence, France, Springer: 182.
- Karagiannis, D. and P. Höfferer (2006). Metamodels in Action: An Overview. ICSOFT 2006 - First International Conference on Software and Data Technologies. J. Filipe, B. Shishkov and M. Helfert. Setúbal, Insticc Press: IS-27-IS-36.
- Karagiannis, D., W. Grossmann, et al. (2008) "Open Model Initiative - A Feasibility Study.“, URL: http://cms.dke.univie.ac.at/uploads/media/Open_Models_Feasibility_Study _SEPT_2008.pdf
- Karagiannis, D., H.-G. Fill, et al. (2008). Metamodeling: Some Application Areas in Information Systems. UNISCON. R. Kaschek and et al., Springer: 175-188.
- Karagiannis, D., Moser, C., Mostashari, A. (2012): “Compliance Evaluation with Heatmaps”, accepted for CAiSE 2012, Gdańsk, Poland, 25th – 29th June, 2012.
- Kühn, H. and M. Murzek (2005). Modelling: From Craftsmanship to Automation. Proceedings of the 4th International Conference on Business Informatics Research (BIR 2005). P. Backlund, S. Carlsson and E. Soederstroem.
- Rausch, T.; Kühn, H.; Murzek, M.; Brennan, T. (2011). BPMN for Business Professionals: Making BPMN 2.0 Fit for Full Business Use. In: Shapiro, R.; White, A. S.; Palmer, N.; zur Muehlen, M.; Allweyer, T.; Gagné, D. et al (Eds.): BPMN 2.0 Handbook, 2011, Future Strategies, pp. 167-180.

OMiLAB References on Industrial Applications of Modelling

- Abazi, F., H.-G. Fill, et al. (2011). Formalising Knowledge-intensive Nuclear Fuel Process Models Using Pattern Theory. KSEM 2011, Springer.
- Fill, H.-G., A. Gericke, et al. (2007). Modeling for Integrated Enterprise Balancing (German: Modellierung für Integrated Enterprise Balancing). Wirtschaftsinformatik 06/2007: 419-429.
- Fill, H.-G., A. Eberhart, et al. (2011). An Approach to Support the Performance Management of Public Health Authorities using an IT based Modeling Method. Proceedings of the 10th International Conference on Wirtschaftsinformatik WI 2.011, Zürich, CH.
- Karagiannis, D., J. Mylopoulos, et al. (2007). Business Process-Based Regulation Compliance: The Case of the Sarbanes-Oxley Act. 15th IEEE International RE Conference, 315-321, IEEE.
- Karagiannis, D., F. Ronaghi, et al. (2007). Business-oriented IT management: Developing e-business applications with E-BPMS. ICEC 2008, ACM.
- Kühn, H., Murzek, M., Specht, G., Zivkovic, S. (2010). Model-Driven Development of Interoperable, Inter-Organisational Business Processes, in: Yannis Charalabidis (Ed.), "Interoperability in the Digital Public Services and Administration: Bridging E-Government and E-Business", pp. 119-143, IGI Global.
- Orensanz, D. et al. (2011): D2.2: Immigration Policy 2.0 Definition of Services and Service Bundles, Public deliverable: http://www.immigrationpolicy2.eu/trunk/Deliverables/D.2.2_Definition%20of%20 Services%20and%20Service%20Bundles_v1.1.pdf (access 24-09-2012)
- Zivkovic, S., Miksa, K., and Kühn, H. (2011). A Modelling Method for Consistent Physical Devices Management: An ADOxx Case Study. In: Salinesi, C., Pastor, O. (Eds.): Proceedings of Advanced Information Systems Engineering Workshops. 1st International Workshop on Conceptualization of Modelling Methods (CMM 2011). 2011, LNBIP 83, Springer, pp. 104-118.

Additional References

- Harel, D. and B. Rumpe (2000). Modeling Languages: Syntax, Semantics and All That Stuff - Part I: The Basic Stuff. Rehovot, Israel, The Weizmann Institute of Science: 28p.
- Harel, D. and B. Rumpe (2004). "Meaningful Modeling: What's the Semantics of "Semantics"?" IEEE Computer October 2004: 64-72.
- Koch, S., S. Strecker, et al. (2006). Conceptual Modelling as a New Entry in the Bazaar: The Open Model Approach. Open Source Systems, IFIP International Federation for Information Processing. 203/2006: 9-20.
- Schmidt, D. (2006). "Model-Driven Engineering." IEEE Computer 39(2): 25-34.
- Schütte, R. and J. Becker (1998). Subjektivitätsmanagement bei Informationsmodellen (German: Management of subjectivity for information models). Modellierung 98, Münster, GI-Workshop.
- Schütte, R. and T. Rotthowe (1998). The Guidelines of Modeling - An Approach to Enhance the Quality in Information Models. ER'98. T. W. Ling, S. Ram and L. M. Lee, Springer: 240-254.
- Stachowiak, H. (1973). Allgemeine Modelltheorie (German: General Model Theory), Springer.