



Meta-Modelling as a Concept: The Conceptualisation of Modelling Methods

Tutorial Team

Robert Woitsch, Hans-Georg Fill,
Sabin Popescu, Vedran Hrgovic

AGENDA

PART I:

- Motivation
- Foundations & Technologies
- Conceptualization & Development
- Best Practices

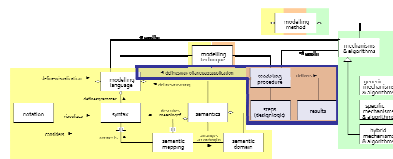
PART II:

- Hands-On Session

PART III:

- Conclusion
- Outlook

1001



Scenario Description

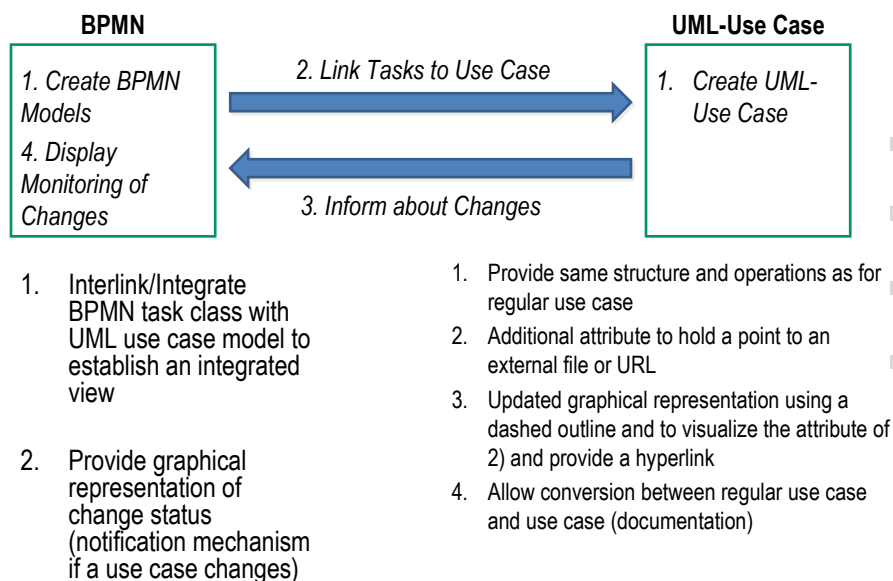
Case:

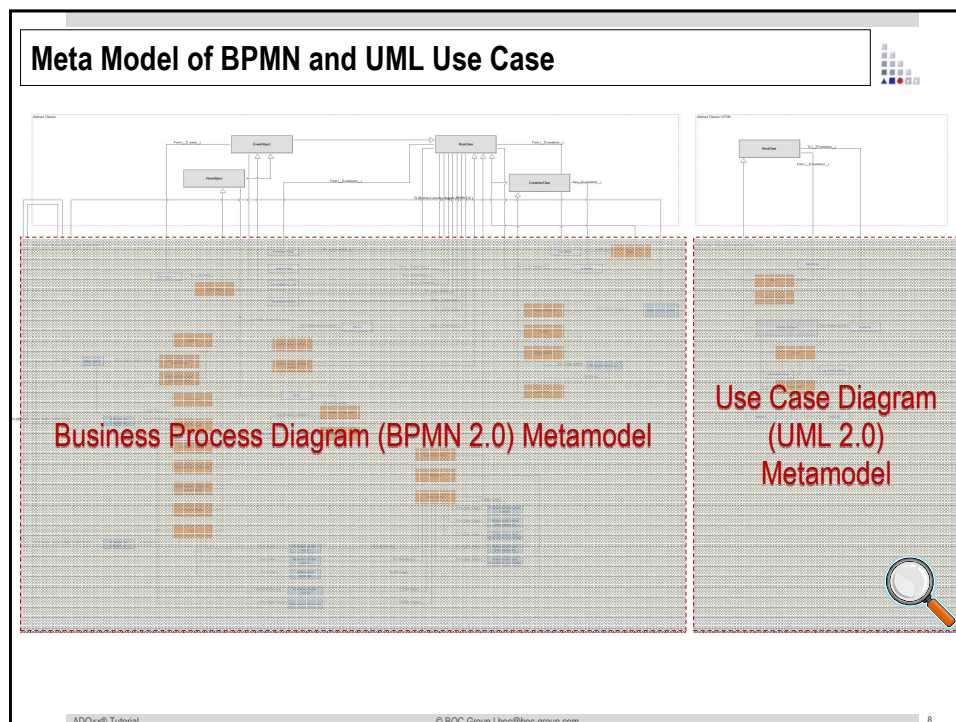
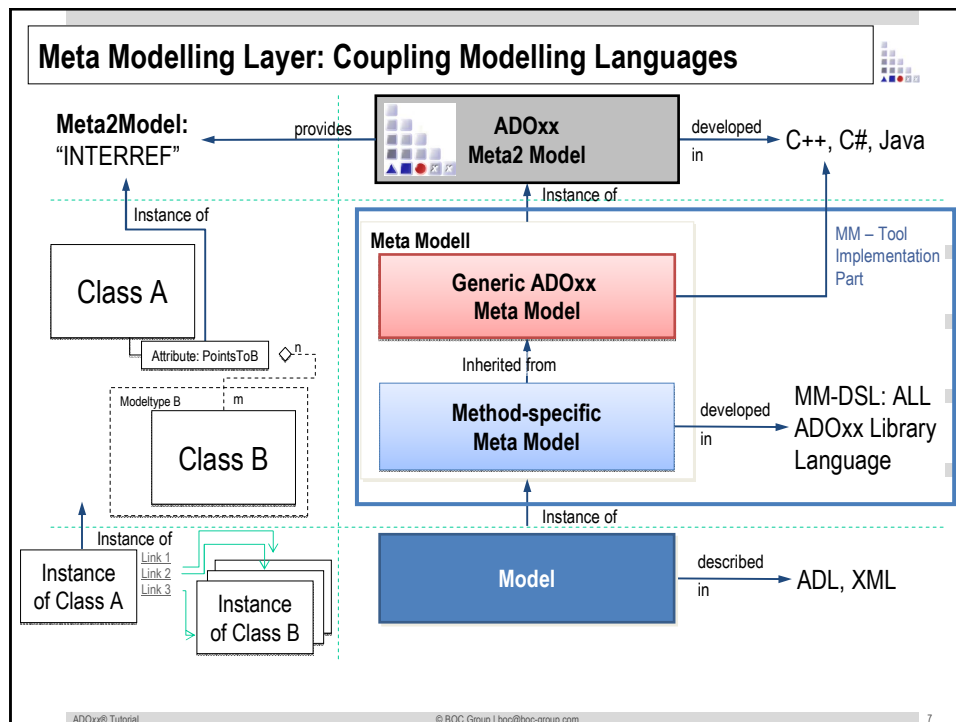
Two implementations of different modelling approaches are combined to support a common modelling procedure ("Vorgehensmodell"). The modelling approach BPMN and UML - use case models are implemented in a coupled way to enable an integrated view from processes to use cases.

GOAL:

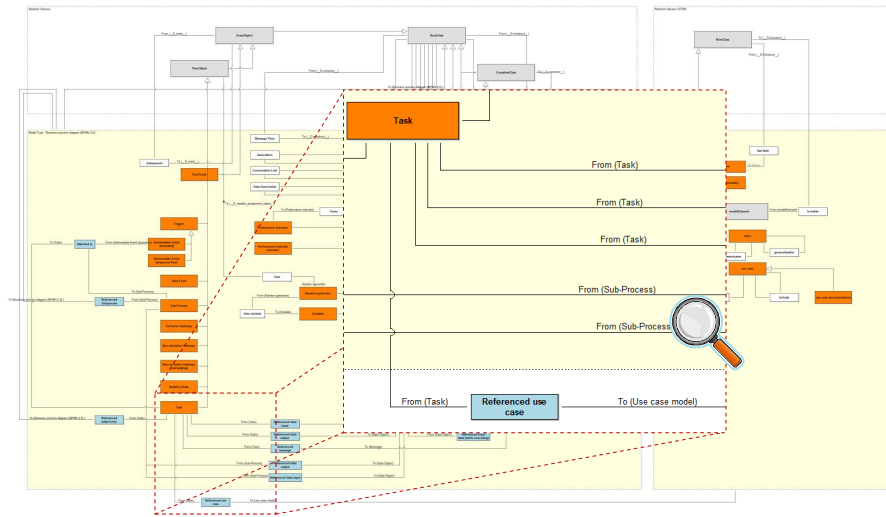
- Demonstrate how a combined usage of two modelling approaches is realized
- Develop functionality based on combined view

Coupling of BPMN and UML-Use Case

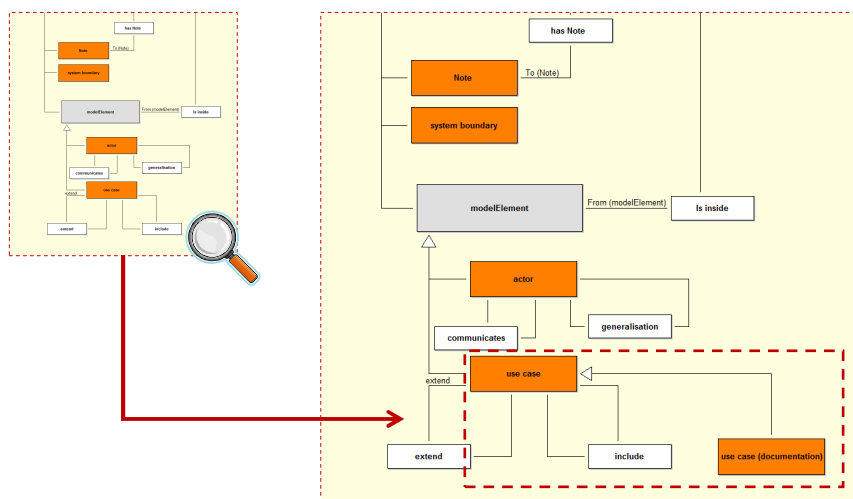




Integrated View of BPMN and UML Use Case Meta Model



Specialisation: CLASS "USE CASE (Documentation)"



Update Listener

1. Enable debugging facility for AdoScript Development
2. Listen on change events in Use case diagram
3. Update automatically related BPMN activities and their state

Applied ADOxx Functionality

- **ADOxx Constructs for Modelling Language Extensions**
 - INTERREF attribute: the **attribute type INTERREF** enables to combine two modelling classes using a pointer within one meta model.
 - Hyperlink: the **hyperlink and model-pointer** functionality enables to navigate better within models.
 - **Attribute Depended Graphical Representation**: depending on the value of an attribute, the graphical representation changes and hence a status can be presented in the model.
- **ADOxx Constructs for Mechanism and Algorithms Development**
 - **Event-Listener**: the platform provides a set of event-listener and hence changes in the model can be identified.
 - Use AdoScript **Core Operations to parse model** and update the corresponding model accordingly.

ADOxx Realisation Approach Overview

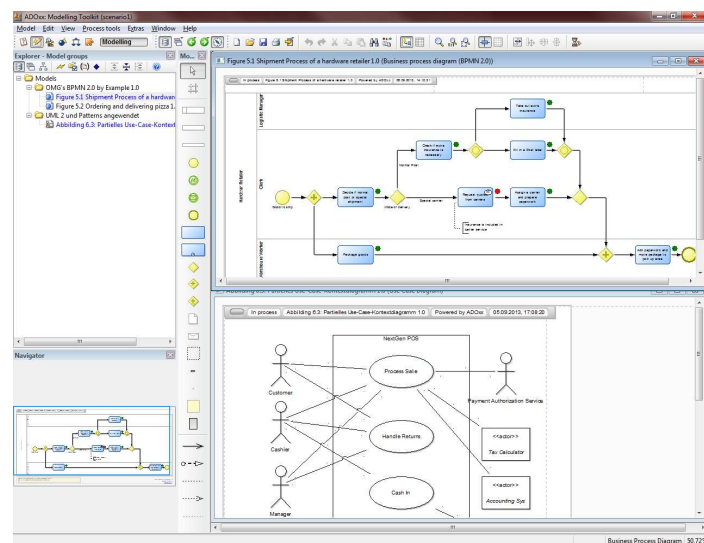
- **Modelling Language**

- New class for UML Use Case
- Change GRAPHREP, ATTRREP of new UML Use Case
- Define CONVERSION of UML Use Case to new UML Use Case
- Change GRAPHREP, ATTRREP from BPMN Activity
- Attribute dependent representation for status

- **Mechanism and Algorithms**

- AdoScript identifies changes in specification and changes status
- **Outlook on ADD-ON:** Document changes can be tracked and AdoScript can be invoked from outside.

Implementation Result



Used ADOxx Functionality: Coupling Modelling Languages	
Introduction	Mechanisms & Algorithms Implementation
Setup of Implementation Environment	Core Functions for Model Manipulation
Modelling Language Implementation	Database
Classes	Visualisation
Relations	Query
Class Attributes and Attributes	Transformation
GRAPHREP	Configuration of ADOxx Components
ATTRREP	Visualisation
CLASS Cardinality	Query
CONVERSION	External Coupling ADOxx Functionality
Model Pointer	ADOscript Triggers
Attribute Facets	ADOscript Language Constructs
Model Types	Visualisation AdoScripts
	Visualisation Expression
	Query ADOscript
	Transformation ADOscript
	ADD-ON Implementation
	ADOxx Web-Service
	XML / ADL Import – Export
	ADOscript Batch Mode

HANDS-ON

Coupling BPMN and UML Use Case Diagrams

4. SCENARIO: COUPLING MODELLING LANGUAGES TO SUPPORT MODELLING PROCEDURE

"USE CASE (Documentation)" Requirements

1. Provide same structure and operations as for regular use case
2. Additional attribute to hold a point to an external file or URL
3. Updated graphical representation using a dashed outline
4. Updated graphical representation to visualize the attribute of 2) and provide a hyperlink
5. Allow conversion between regular use case and use case (documentation)

SPECIALISATION OF CLASS "USE CASE (Documentation)"

The screenshot displays the 'BPMN-UC Integrated View for Requirements Engineering Dynamic Library v0.2 - Edit class hierarchy' window. On the left, a class hierarchy tree is shown with 'use case' highlighted. A red dashed box labeled '1' encloses the tree, and a red dashed box labeled '2' encloses the 'Derive a new class' dialog. The dialog has 'Class name: use case (documentation)' and 'Superclass: use case'. Below the dialog, a 'Click on New -> New class ...' button is shown. To the right of the dialog, a text box says 'Enter new class name "use case (documentation)" Check that class is derived from Superclass: "use case"'. Below that, another text box says 'Confirm by clicking on OK -> Class has been created, all definitions/setup have been derived from superclass.'

1 Open Class Hierarchy of Sublibrary
"BPMN-UC Integrated View for Requirements Engineering
Dynamic Library v0.2"

Switch View to display (Button "View")
a) Metamodel
b) Class hierarchy

Drill-down to class "use case" and highlight/mark
_D-construct
- modelElement
- use case

2 Click on New -> New class ...

Derive a new class

Class name: use case (documentation)

Superclass: use case

Enter new class name "use case (documentation)"
Check that class is derived from Superclass: "use case"

Confirm by clicking on OK -> Class has been created, all
definitions/setup have been derived from superclass.

ADD NEW ATTRIBUTE TO NEW CLASS

1 Select and highlight the new class
Click on New -> New attribute ...

2 Enter attribute name and select attribute type
Confirm with OK to save the attribute

ADOxx® Tutorial © BOC Group | boc@boc-group.com 19

ADD NEW ATTRIBUTE TO NOTEBOOK

1 Select class attribute "AttrRep"
Click on Edit ...

2 Add attribute to notebook by adding the line
ATTR "Collaborative documentation"
in chapter "Description"
Confirm with Close to save the change.

ADOxx® Tutorial © BOC Group | boc@boc-group.com 20

ADD NEW ATTRIBUTE TO NOTEBOOK



NOTEBOOK

ATTRREP CODE

```
CHAPTER "Description"
  ATTR "Name"
  ATTR "Presentation"
  ATTR "Description" lines:5
  ATTR "Comment" lines:5
  ATTR "External documentation"
  ATTR "Collaborative documentation"
CHAPTER "Conditions"
  ATTR "Preconditions"
  ATTR "Endconditions - case of success"
  ATTR "Endconditions - case of failure"
  ATTR "Postconditions"
CHAPTER "Details"
  ATTR "Details"
  ATTR "Details - others" lines: 5
  ATTR "Prototyping information" lines:15
CHAPTER "Extension points"
  ATTR "Extension points"
```

UPDATE GRAPHREP OF NEW CLASS



1 Select class attribute „GraphRep“

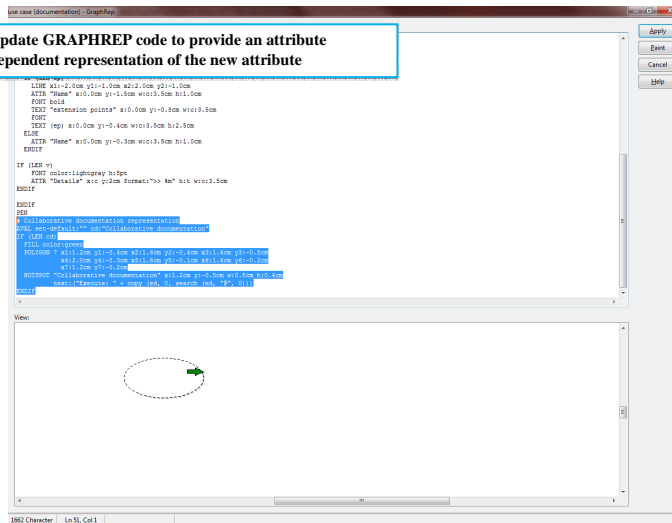
Click on Edit ...

2 Open Development Wizard/Support

UPDATE GRAPHREP OF NEW CLASS

3

Update GRAPHREP code to provide an attribute dependent representation of the new attribute



UPDATE GRAPHREP OF NEW CLASS

GRAPHREP CODE

```
GRAPHREP
sizing:asymmetrical
PEN w:0.02cm
FILL color:white

AVAL v:"Details"
AVAL ep:"Extension points table"
AVAL set-default:"Default" d:"Presentation"
PEN style:dash
IF (d = "Default")
    ELLIPSE x:0.00cm y:0.00cm rx:2.00cm
    ry:1.00cm
    IF (LEN ep)
        LINE x1:-1.45cm y1:-0.7cm x2:1.45cm y2:-
        0.7cm
        FONT bold
        TEXT "extension points" x:0.0cm y:-0.5cm
        w:c:3.5cm
        FONT
        TEXT (ep) x:0.0cm y:-0.1cm w:c:3.5cm
        h:1.0cm
        ATTR "Name" x:0.0cm y:1.3cm w:c:3.5cm
    ELSE
        ATTR "Name" x:0.0cm y:0.0cm w:c:3.5cm h:c
    ENDIF
ENDIF
IF (LEN v)
    FONT color:lightgray h:8pt
    ATTR "Details" x:c y:1cm format:">> %m"
    h:t w:c:3.5cm
ENDIF
```

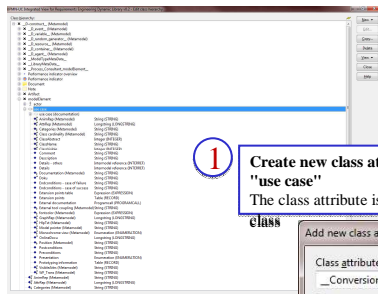
```
ELSIF (d = "Rectangle")
    RECTANGLE x:-2.0cm y:-2.0cm w:4.0cm h:4.0cm
    ELLIPSE x:1.5cm y:-1.7cm rx:0.3cm ry:0.15cm

    IF (LEN ep)
        LINE x1:-2.0cm y1:-1.0cm x2:2.0cm y2:-1.0cm
        ATTR "Name" x:0.0cm y:-1.5cm w:c:3.5cm h:1.0cm
        FONT bold
        TEXT "extension points" x:0.0cm y:-0.8cm w:c:3.5cm
        FONT
        TEXT (ep) x:0.0cm y:-0.4cm w:c:3.5cm h:2.5cm
    ELSE
        ATTR "Name" x:0.0cm y:-0.3cm w:c:3.5cm h:1.0cm
    ENDIF

    IF (LEN v)
        FONT color:lightgray h:8pt
        ATTR "Details" x:c y:2cm format:">> %m" h:t w:c:3.5cm
    ENDIF
ENDIF

PEN:
# Collaborative documentation representation
AVAL set-default:"" cd:"Collaborative documentation"
# Note that an empty PROGRAMCALL attribute has the value ""
IF (LEN cd > 1)
    FILL color:green
    POLYGON 7 x1:1.2cm y1:-0.4cm x2:1.6cm y2:-0.4cm x3:1.6cm y3:-
    0.5cm
    x4:2.0cm y4:-0.3cm x5:1.6cm y5:-0.1cm x6:1.6cm y6:-
    0.2cm
    x7:1.2cm y7:-0.2cm
    HOTSPOT "Collaborative documentation" x:1.2cm y:-0.5cm
    w:0.8cm h:0.4cm
    text:{"Execute: " + copy (ed, 0, search (ed, "e",
    0))}
ENDIF
```

DEFINE CONVERSION RULES

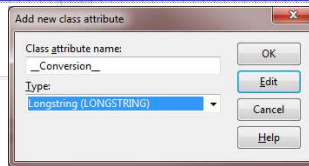


1

Create new class attribute "__Conversion_" for class "use case"

The class attribute is used to define conversion rules for each

class



Define the conversion rules for use case -> use case (documentation)

Define the conversion rules for use case (documentation)-> use case

DEFINE CONVERSION RULES

CONVERSION RULES

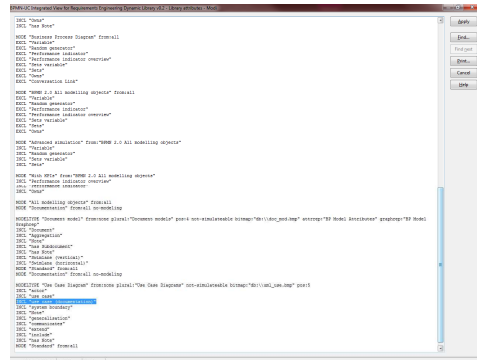
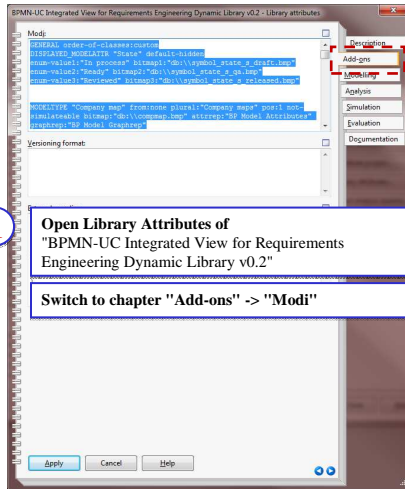
```
CLASS "use case
(documentation)"
ATTR "Name"
ATTR "Presentation"
ATTR "Description" lines:5
ATTR "Comment" lines:5
ATTR "External documentation"
ATTR "Preconditions"
ATTR "Endconditions - case of
success"
ATTR "Endconditions - case of
failure"
ATTR "Postconditions"
ATTR "Details"
ATTR "Details - others" lines:
5
ATTR "Prototyping information"
lines:15
ATTR "Extension points"
```

**use case -> use case
(documentation)**

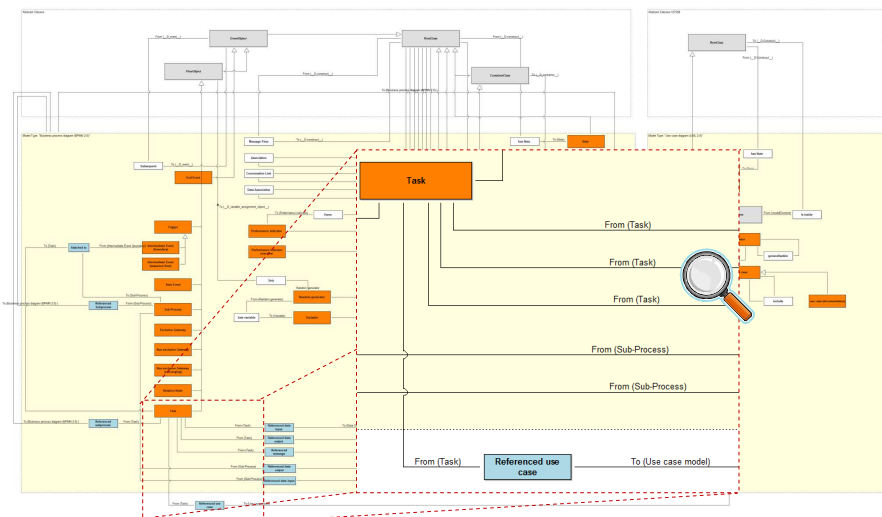
```
CLASS "use case"
ATTR "Name"
ATTR "Presentation"
ATTR "Description" lines:5
ATTR "Comment" lines:5
ATTR "External documentation"
ATTR "Preconditions"
ATTR "Endconditions - case of
success"
ATTR "Endconditions - case of
failure"
ATTR "Postconditions"
ATTR "Details"
ATTR "Details - others" lines: 5
ATTR "Prototyping information"
lines:15
ATTR "Extension points"
```

**use case (documentation) ->
use case**

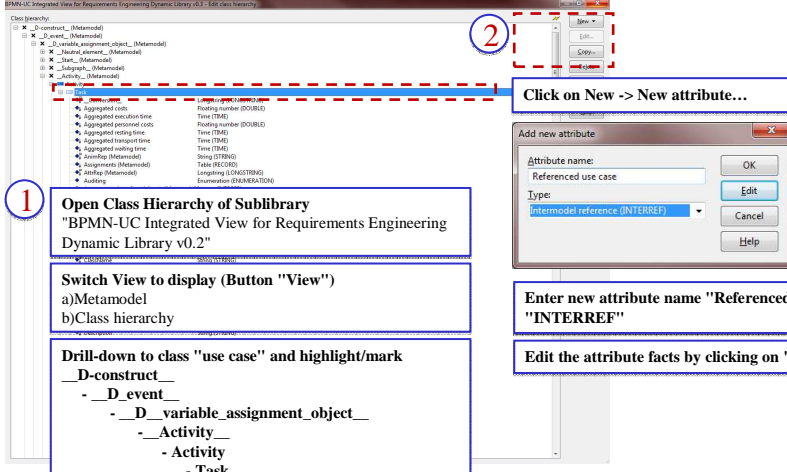
ADD CLASS TO MODELTYPE



INTEGRATED VIEW OF BPMN AND UML USE CASE



UPDATE OF CLASS "Task" INTERREF Use Case



1 Open Class Hierarchy of Sublibrary "BPMN-UC Integrated View for Requirements Engineering Dynamic Library v0.2"

Switch View to display (Button "View")

- Metamodel
- Class hierarchy

Drill-down to class "use case" and highlight/mark

- D-construct
 - D-event
 - D-variable_assignment_object
 - Activity
 - Activity
 - Task

2 Click on New -> New attribute...

Add new attribute

Attribute name: Referenced use case

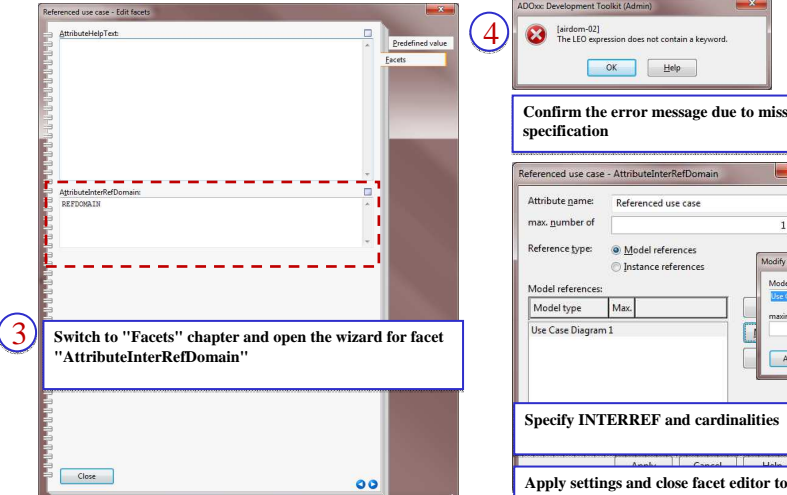
Type: Intermodel reference (INTERREF)

Enter new attribute name "Referenced use case" of type "INTERREF"

Edit the attribute facts by clicking on "Edit"

ADOxx® Tutorial © BOC Group | boc@boc-group.com 29

UPDATE OF CLASS "Task" INTERREF Use Case



3 Switch to "Facets" chapter and open the wizard for facet "AttributeInterRefDomain"

4 Confirm the error message due to missing facet specification

Referenced use case - AttributeInterRefDomain

Attribute name: Referenced use case

max. number of: 1

Reference type: ☒ Model references ☐ Instance references

Model references:

Model type: Use Case Diagram

maximum number of references: 1

Use Case Diagram 1

Specify INTERREF and cardinalities

Apply settings and close facet editor to save changes.

ADOxx® Tutorial © BOC Group | boc@boc-group.com 30

Add another attribute "State" of type ENUMERATION

6

Click on "Edit" to add the enumeration values

"Apply" and "Close" to save the changes.

Select class attribute "AttrRep"

Click on Edit ...

2

Add attribute to notebook by adding the line

```
ATTR "Referenced use case"
ATTR "State"
```

in chapter "Description"

Confirm with Close to save the change.

ADD NEW ATTRIBUTES TO NOTEBOOK

ATTRREP CODE

NOTEBOOK

```

AVAL sTaskType: "Task type"
AVAL sLoopCharacteristic: "Loop type"

#-----
CHAPTER "Description"
#-----
ATTR "Name"
ATTR "Show name"
ATTR "Referenced use case"
ATTR "State"
# center, below,
ATTR "Task type" ctrltype:drop-down
# Not specified@Service@Send@Receive@User@Manual@Business rule@Script@Call
activity
ATTR "Global task" ctrltype:check checked-value:"Yes" unchecked-value:"No"
enabled:((sTaskType = "User") OR (sTaskType = "Manual") OR (sTaskType =
"Business rule") OR (sTaskType = "Script"))
ATTR "Order"
ATTR "Description" lines:5 mandatory:1
ATTR "Comment" lines:5
ATTR "Open questions" lines:5
ATTR "Id" write-protected
...

```

UPDATE GRAPHREP OF CLASS

1 Select class attribute „GraphRep“

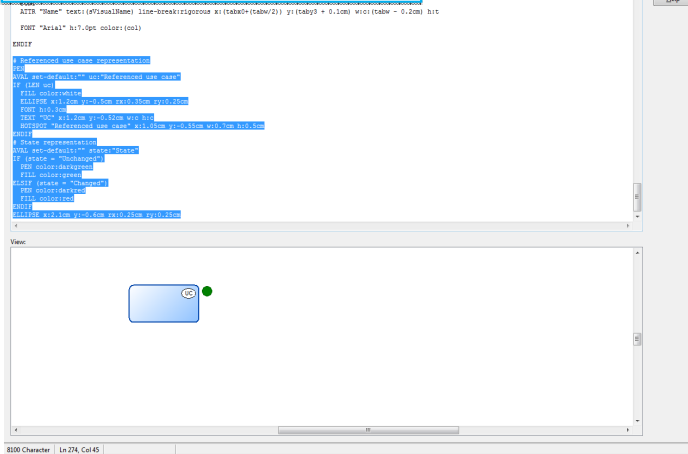
Click on Edit ...

2 Open Development Wizard/Support

UPDATE GRAPHREP OF CLASS

3

Update GRAPHREP code to provide an attribute dependent representation of the new attribute



UPDATE GRAPHREP OF CLASS

GRAPHREP CODE

```

# Referenced use case representation
PEN
AVAL set-default:" uc:"Referenced use case"
IF (LEN uc)
  FILL color:white
  ELLIPSE x:1.2cm y:-0.5cm rx:0.35cm ry:0.25cm
  FONT h:0.3cm
  TEXT "UC" x:1.2cm y:-0.52cm w:c h:c
  HOTSPOT "Referenced use case" x:1.05cm y:-0.55cm w:0.7cm h:0.5cm
ENDIF
# State representation
AVAL set-default:" state:"State"
IF (state = "Unchanged")
  PEN color:darkgreen
  FILL color:green
ELSIF (state = "Changed")
  PEN color:darkred
  FILL color:red
ENDIF
ELLIPSE x:2.1cm y:-0.6cm rx:0.25cm ry:0.25cm
  
```

-

[illegible]

DEBUGGING AdoScript in Modelling Toolkit

ITEM "Shell window" modeling:"Extras"

Menu Entry

SET endbutton:"ok"

Script Code for Shell Window

WHILE (endbutton = "ok")

{

CC "AdoScript" EDITBOX text:(adoscript)

title:"Enter the code you want to test..."

oktext:"Go!"

IF (endbutton = "ok")

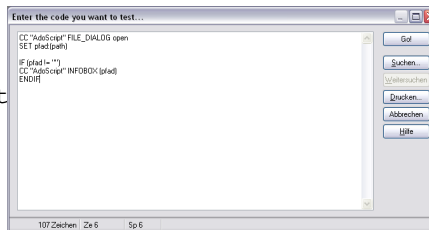
{

SET adoscript:global:(text)

EXECUTE (text)

}

}



Attribute Change Event Listener

1

Event listener for attribute changes

Define the event listener in the ExternalCoupling Attribute

"AfterEditAttributeValue" listens to any change in attribute values of the model

2

```
ON_EVENT "AfterEditAttributeValue" {  
  #preset: modelid, instid, attridattrtypeid
```

```
  # Only for use case diagrams
```

```
  CC "Core" GET_MODEL_INFO modelid:(modelid)
```

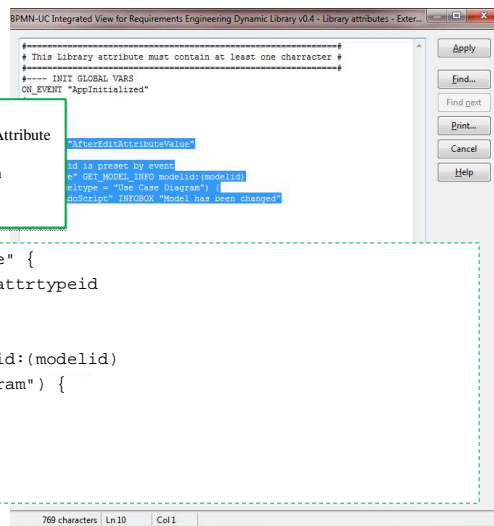
```
  IF (modeltype = "Use Case Diagram") {
```

```
    # Update logic
```

```
    ...
```

```
  }
```

```
}
```



State Update Logic

```
CC "Core" GET_INCOMING_INTERREFS modelid:(modelid)
# Parse the LEO text the first time to get the number of elements in it
LEO parse:(reftext) get-elem-count:elemcount
# Go through each element in the LEO text
FOR curelem from:0 to:(elemcount - 1) {
  # Access the information of the current element
  LEO parse:(reftext)
    set-cur-elem-index:(curelem)
    get-int-value:srcobjid:"srcobjid"
    get-int-value:srcmodelid:"srcmodelid"
  # to update instance attributes, the model must be loaded - check if so
  CC "Core" IS_MODEL_LOADED modelid:(srcmodelid)
  SET isInitialLoaded:(isloaded)
  IF (NOT (isInitialLoaded)) {
    CC "Core" LOAD_MODEL modelid:(srcmodelid)
  }
  #set the change value and save model
  CC "Core" SET_ATTR_VAL objid:(srcobjid) attrname:"State" val:"Changed"
  CC "Core" SAVE_MODEL modelid:(srcmodelid)
  #if initially closed, close again
  IF (NOT (isInitialLoaded)) {
    CC "Core" DISCARD_MODEL modelid:(srcmodelid)
  }
}
```

Any Questions?

