

# ADOxx TUTORIAL: eHEALTH SAMPLE

## Agenda



- ▶ **eHealth Scenario**
- ▶ Hybrid Modelling Approach Introduction
- ▶ Hands On Sessions:
  - ▶ Definition of Model Structure on ADOxx
  - ▶ Model Processing
  - ▶ Amalgamation
- ▶ Q&A

## eHealth Scenario



### Characteristics of eHealth Models

- ▶ Interplay of different actors:
  - ▶ directly involved (e.g. patients, medical professionals)
  - ▶ semi-directly involved (e.g. family caregivers)
  - ▶ indirectly involved (e.g. insurance companies)
- ▶ Obeying to different rules:
  - ▶ imposed rules (e.g. law – privacy issues)
  - ▶ “lived” rules (e.g. management decisions)
- ▶ Importance of Interoperability
  - ▶ many different standards for all aspects (e.g. PHR's)
  - ▶ importance of “leveled” semantics (e.g. understanding diagnosis)

Human Centric  
Aspect

Semantic  
Interoperability

**Focus of the Tutorial's Practice Slot**

### Goals

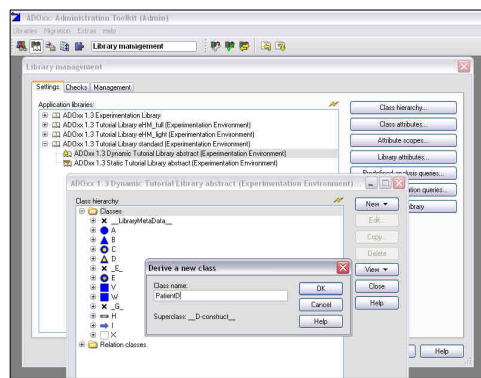
- (1) Define Model Structure using ADOxx for depicting interplay between Patients and Medical Professionals
- (2) Apply hybrid modelling approach to extend the Model Structure by importing two eHealth aspects: Management & Semantics
- (3) Implement functionality to allow:
  - (1) Defining Diagnosis of the Patients
  - (2) Informing Medical Professionals of Diagnosis Changes
  - (3) Calculate the number of Patients per doctor and allow automatic assignment if Patient is not treated yet
  - (4) Create and assign PHR's to Patients (if none is available)

## Agenda

- ▶ eHealth Scenario
- ▶ **Hybrid Modelling Approach Introduction**
- ▶ Hands On Sessions:
  - ▶ Definition of Model Structure on ADOxx
  - ▶ Model Processing
  - ▶ Amalgamation
- ▶ Q&A

## Definition of Modelling Language on ADOxx

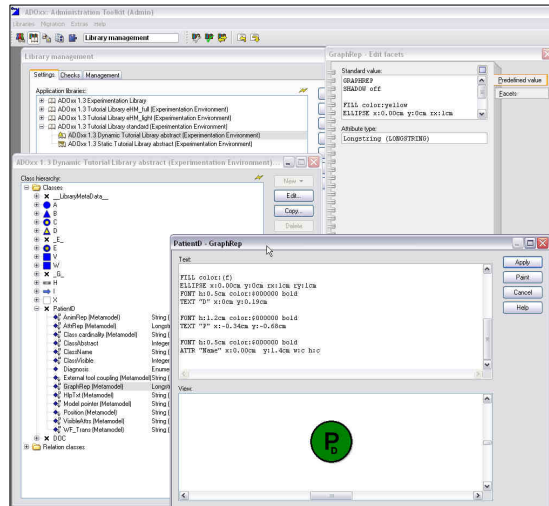
Define Classes,  
Attributes and Model  
Types



1. Open Library Management
2. Define a new Class named "PatientID" in the "Dynamic Tutorial Library"

## Definition of Modelling Language on ADOxx

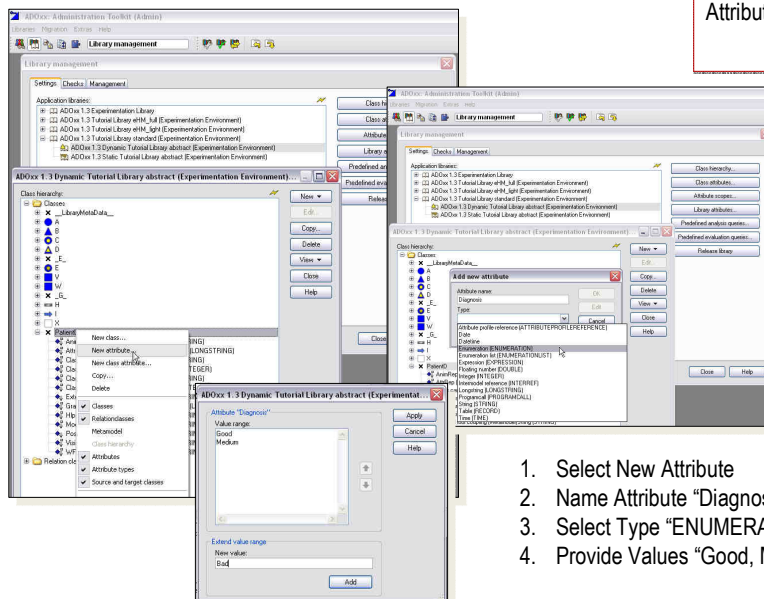
Define Classes,  
Attributes and Model  
Types



1. Select GraphRep Attribute of the Patient class
2. Write GraphRep code to visualise the class

## Definition of Modelling Language on ADOxx

Define Classes,  
Attributes and Model  
Types



1. Select New Attribute
2. Name Attribute "Diagnosis"
3. Select Type "ENUMERATION"
4. Provide Values "Good, Medium, Bad"

## Definition of Modelling Language on ADOxx

Define Classes,  
Attributes and Model  
Types

The screenshot shows the ADOxx 1.3 Dynamic Tutorial Library abstract (Experimentation Environment) window. The class hierarchy on the left lists various classes under 'Classes' and 'Relation classes'. The 'GraphRep - Edit facets' dialog is open, showing the 'Standard value' and 'Defined value' tabs. The 'Standard value' tab is selected, showing the 'GraphRep' attribute type. The 'Defined value' tab is also visible, showing the 'GraphRep' attribute type. The 'GraphRep' attribute type is selected in the 'Standard value' tab.

1. Select GraphRep Attribute in the PatientD class
2. Extend the existing GraphRep code to change the display colour depending on the value of the attribute "Diagnosis"

## Definition of Modelling Language on ADOxx

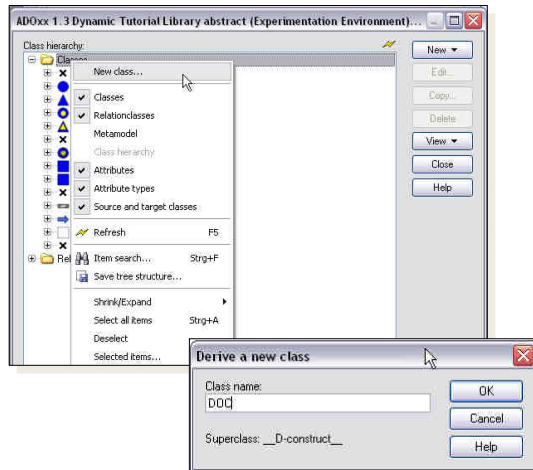
Define Classes,  
Attributes and Model  
Types

The screenshot shows the ADOxx 1.3 Dynamic Tutorial Library abstract (Experimentation Environment) window. The class hierarchy on the left lists various classes under 'Classes' and 'Relation classes'. The 'AttrRep - Edit facets' dialog is open, showing the 'Standard value' and 'Defined value' tabs. The 'Standard value' tab is selected, showing the 'AttrRep' attribute type. The 'Defined value' tab is also visible, showing the 'AttrRep' attribute type. The 'AttrRep' attribute type is selected in the 'Standard value' tab.

1. Select AttrRep Attribute of the PatientD class
2. Write AttrRep code to make attributes visible in the Notebook

## Definition of Modelling Language on ADOxx

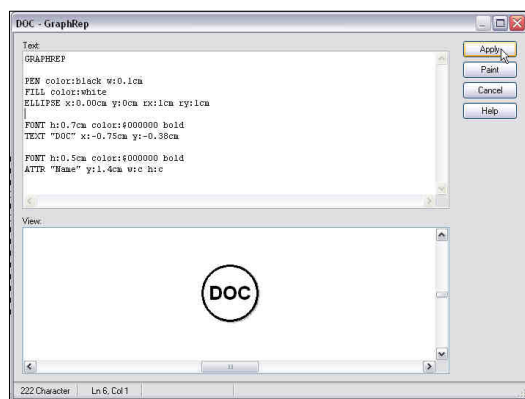
Define Classes,  
Attributes and Model  
Types



1. Define a new Class named "DOC" in the "Dynamic Tutorial Library"

## Definition of Modelling Language on ADOxx

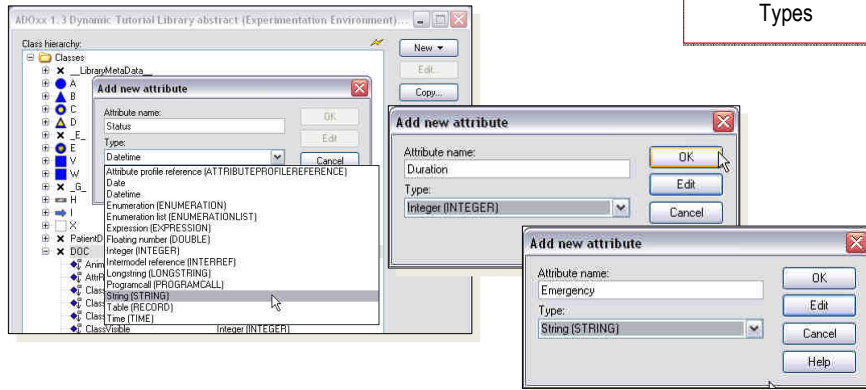
Define Classes,  
Attributes and Model  
Types



1. Select GraphRep Attribute of the DOC class
2. Write GraphRep code to visualise the class

## Definition of Modelling Language on ADOxx

Define Classes,  
Attributes and Model  
Types

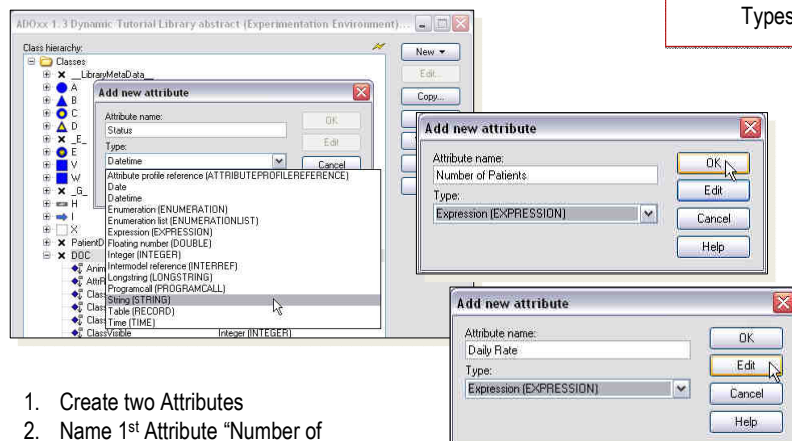


1. Select New Attribute
2. Name Attribute "Duration"
3. Select Type "INTEGER"

1. Select New Attribute
2. Name Attribute "Emergency"
3. Select Type "String"

## Definition of Modelling Language on ADOxx

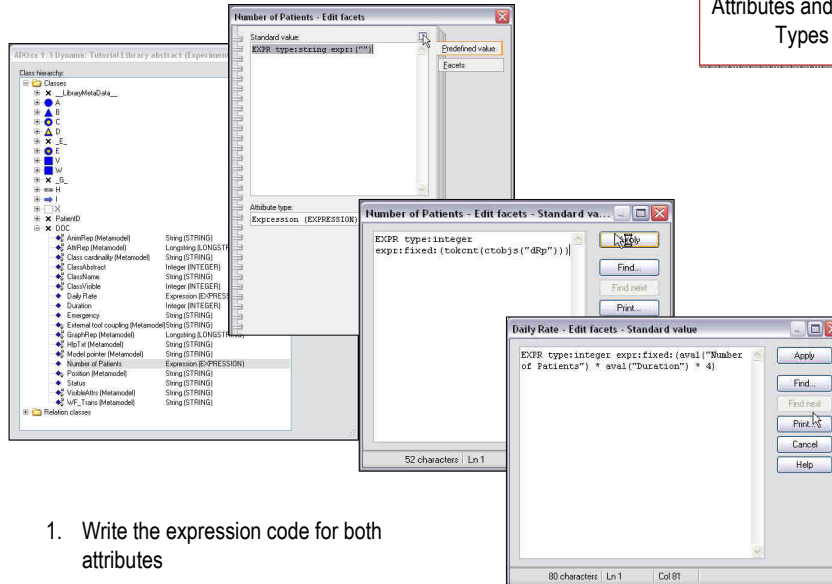
Define Classes,  
Attributes and Model  
Types



1. Create two Attributes
2. Name 1st Attribute "Number of Patients" and 2nd Daily Rate
3. Select Type "Expression"

## Definition of Modelling Language on ADOxx

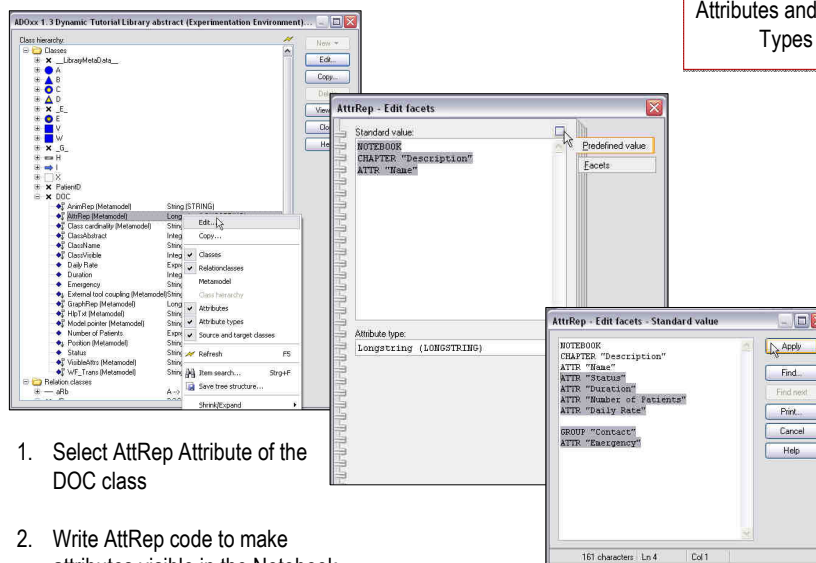
Define Classes,  
Attributes and Model  
Types



1. Write the expression code for both attributes

## Definition of Modelling Language on ADOxx

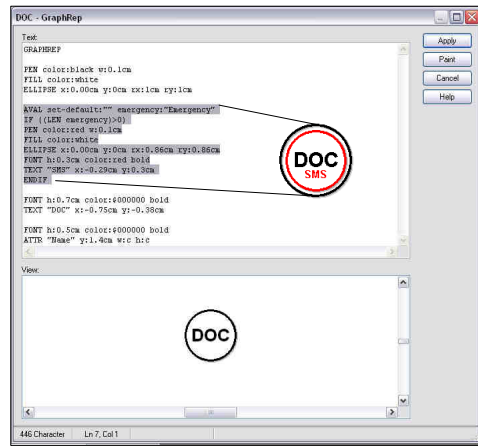
Define Classes,  
Attributes and Model  
Types



1. Select AttRep Attribute of the DOC class
2. Write AttRep code to make attributes visible in the Notebook

## Definition of Modelling Language on ADOxx

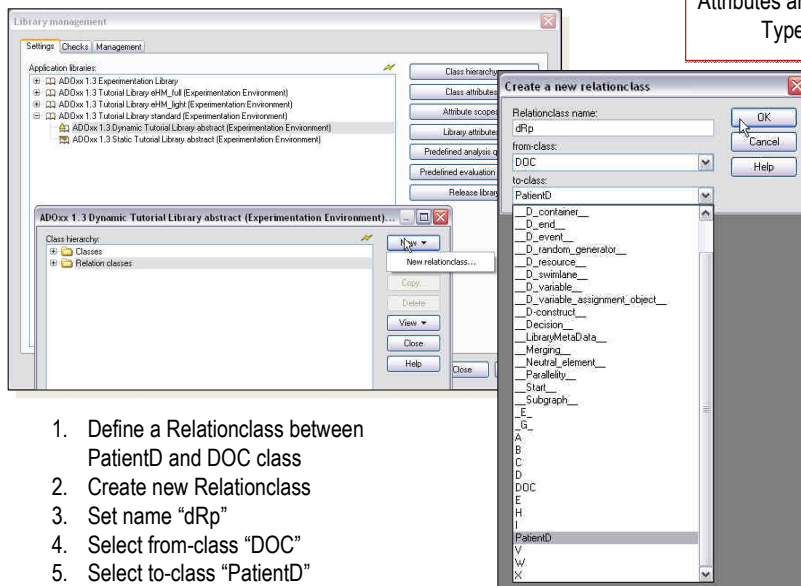
Define Classes,  
Attributes and Model  
Types



1. Select GraphRep Attribute in the DOC class
2. Extend the existing GraphRep code to visualise accessibility by SMS

## Definition of Modelling Language on ADOxx

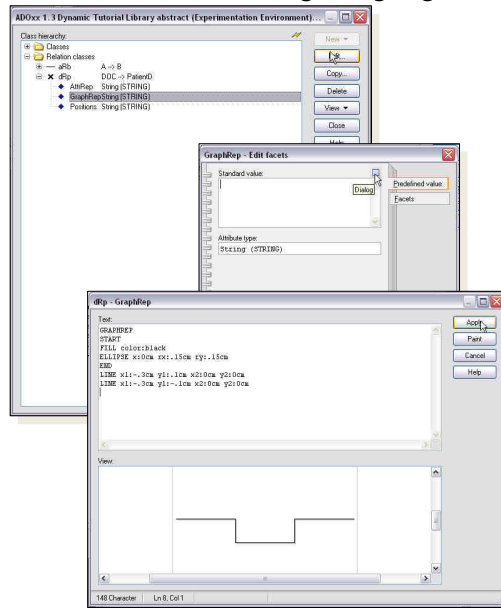
Define Classes,  
Attributes and Model  
Types



1. Define a Relationclass between PatientID and DOC class
2. Create new Relationclass
3. Set name "dRp"
4. Select from-class "DOC"
5. Select to-class "PatientID"

## Definition of Modelling Language on ADOxx

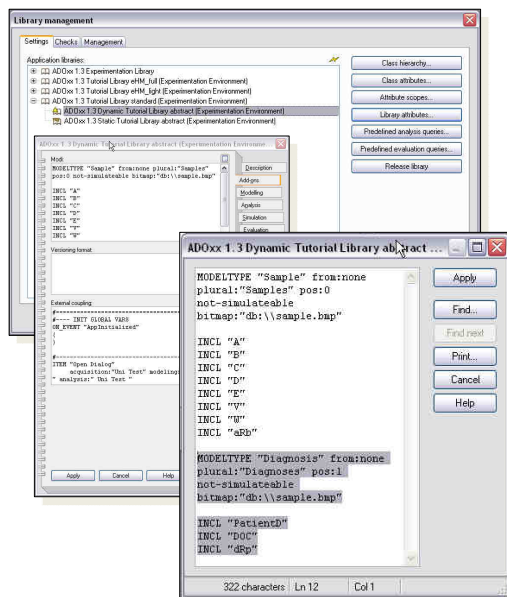
Define Classes,  
Attributes and Model  
Types



1. Select GraphRep Attribute
2. Open Dialog
3. Write GraphRep code to visualise the relationclass "dRp"

## Definition of Modelling Language on ADOxx

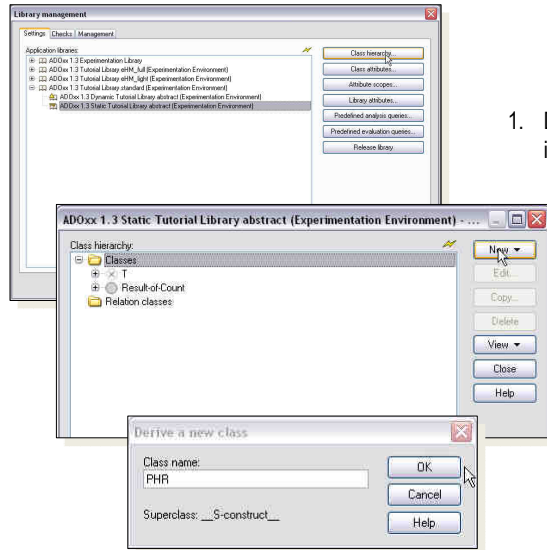
Define Classes,  
Attributes and Model  
Types



1. Select Library attributes
2. Click on tab Add-ons
3. Expand the "Model"
4. Write the Model Type definition Code for "Diagnosis"

## Definition of Modelling Language on ADOxx

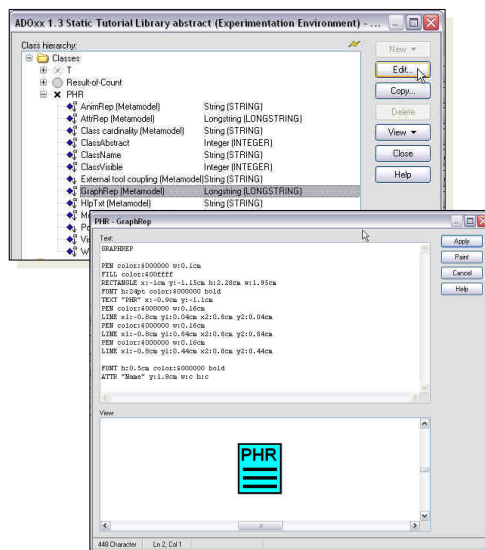
Define Classes,  
Attributes and Model  
Types



1. Define a new Class named "PHR" in the "Static Tutorial Library"

## Definition of Modelling Language on ADOxx

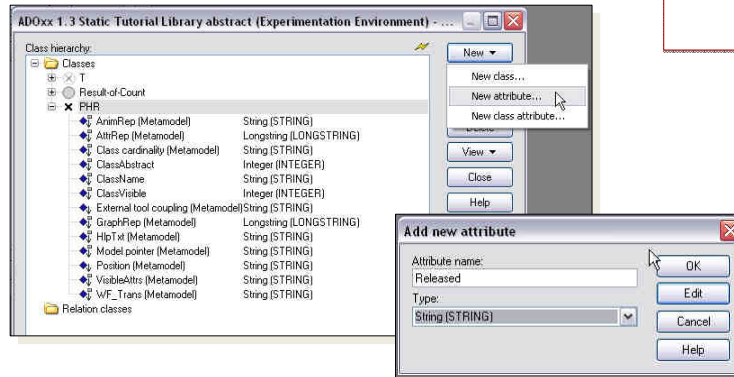
Define Classes,  
Attributes and Model  
Types



1. Select GraphRep Attribute of the PHR class
2. Write GraphRep code to visualise the class

## Definition of Modelling Language on ADOxx

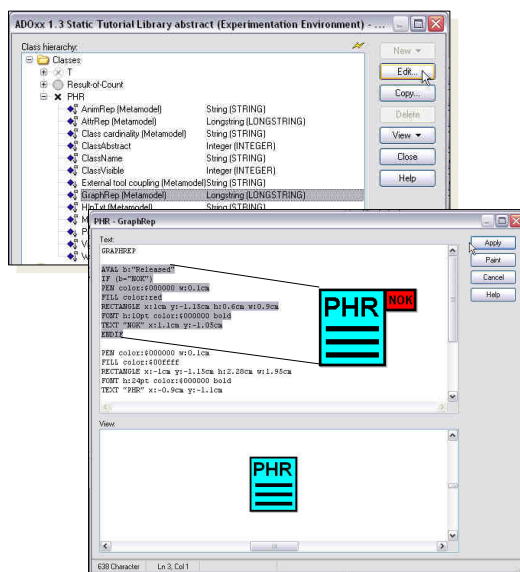
Define Classes,  
Attributes and Model  
Types



1. Select New Attribute
2. Name Attribute "Released"
3. Select Type "STRING"

## Definition of Modelling Language on ADOxx

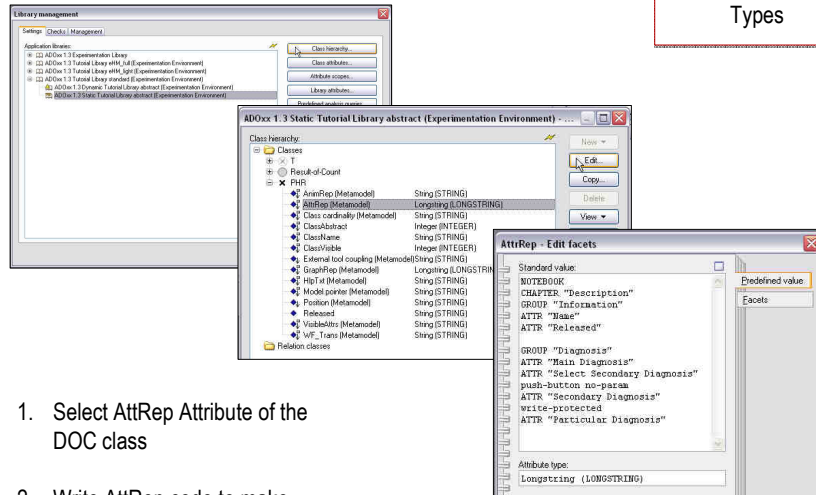
Define Classes,  
Attributes and Model  
Types



1. Select GraphRep Attribute in the PHR class
2. Extend the existing GraphRep code to visualise if PHR has been released or not

## Definition of Modelling Language on ADOxx

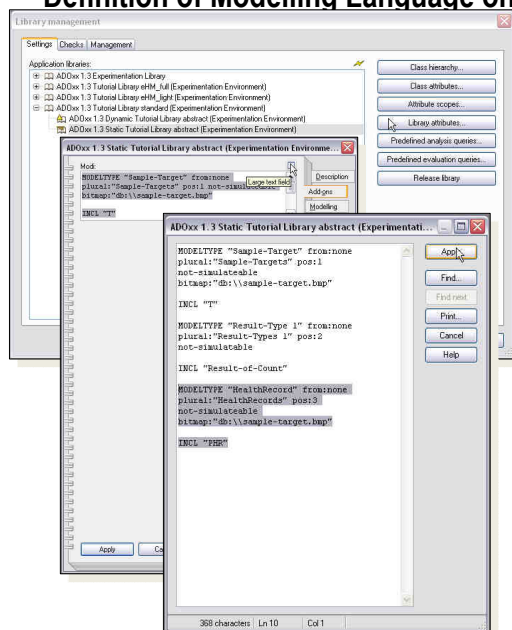
Define Classes,  
Attributes and Model  
Types



1. Select AttRep Attribute of the DOC class
2. Write AttRep code to make attributes visible in the Notebook

## Definition of Modelling Language on ADOxx

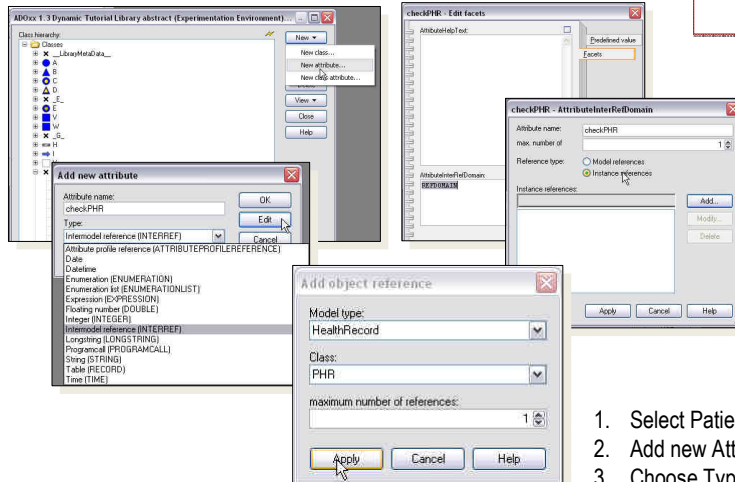
Define Classes,  
Attributes and Model  
Types



1. Select Library attributes
2. Click on tab Add-ons
3. Expand the "Mod"
4. Write the Model Type definition Code for "HealthRecord"

## Mechanisms and Algorithms Implementation on ADOxx

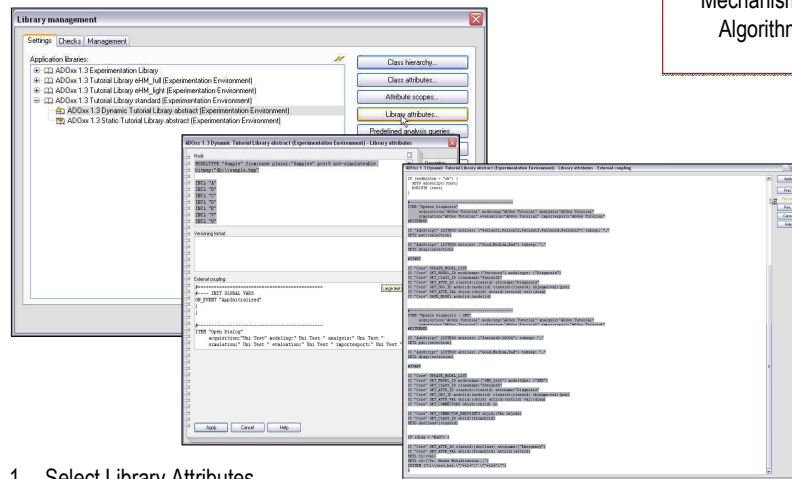
Define Classes,  
Attributes and Model  
Types



1. Select PatientID Class
2. Add new Attribute
3. Choose Type "INTERREF"
4. Point to PHR class in Model Type HealthRecord

## Mechanisms and Algorithms Implementation on ADOxx

Mechanisms &  
Algorithms



1. Select Library Attributes
2. Select Add-ons
3. Create new Menu Entry
4. Write AdoScript allowing change of Patient diagnosis and sending notification to assigned Medical Professional

## Mechanisms and Algorithms Implementation on ADOxx

```

ITEM "Update Diagnosis"
  acquisition:"ADOxx Tutorial" modeling:"ADOxx Tutorial" analysis:"ADOxx
Tutorial"
  simulation:"ADOxx Tutorial" evaluation:"ADOxx Tutorial" importexport:"ADOxx
Tutorial"
#EXTENDED

CC "Modeling" GET_ACT_MODEL
#--> RESULT modelid: intValue

#get all patients
SET aql_str:("<\PatientD">")
CC "AQL" debug EVAL_AQL_EXPRESSION expr:(aql_str) modelid: (modelid)
SET patientIDs: (objids)

#for each patient get name, then generate list
SET patientNames: ""
FOR patientID in: (patientIDs)
{
  CC "Core" GET_OBJ_NAME objid: (VAL patientID)
  #--> RESULT ecode: intValue objname: strValue
  SET patientNames: ((patientNames)+(objname)+",")
}

CC "AdoScript" LISTBOX entries: (patientNames) toksep: ","
SETL pat:(selection)

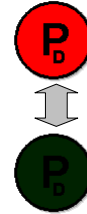
CC "AdoScript" LISTBOX entries: ("Good,Medium,Bad") toksep: ","
SETL diag:(selection)

#START

CC "Core" UPDATE_MODEL_LIST
CC "Core" GET_MODEL_ID modelname:("Patients") modeltype: ("Diagnosis")
CC "Core" GET_CLASS_ID classname:"Patient"
CC "Core" GET_ATTR_ID classid:(classid) attrname:"Diagnosis"
CC "Core" GET_OBJ_ID modelid:(modelid) classid:(classid) objname:val:(pat)
CC "Core" SET_ATTR_VAL objid:(objid) attrid:(attrid) val:(diag)
CC "Core" SAVE_MODEL modelid:(modelid)
  
```

### Mechanisms & Algorithms

#### "Update Diagnosis"



## Mechanisms and Algorithms Implementation on ADOxx

```

ITEM "Update Diagnosis - SMS"
  acquisition:"ADOxx Tutorial" modeling:"ADOxx Tutorial" analysis:"ADOxx Tutorial"
simulation:"ADOxx Tutorial" evaluation:"ADOxx Tutorial"
importexport:"ADOxx Tutorial"
#EXTENDED

CC "Modeling" GET_ACT_MODEL
#--> RESULT modelid: intValue

#get all patients
SET aql_str:("<\PatientD">")
CC "AQL" debug EVAL_AQL_EXPRESSION expr:(aql_str) modelid: (modelid)
SET patientIDs: (objids)

#for each patient get name, then generate list
SET patientNames: ""
FOR patientID in: (patientIDs)
{
  CC "Core" GET_OBJ_NAME objid: (VAL patientID)
  #--> RESULT ecode: intValue objname: strValue
  SET patientNames: ((patientNames)+(objname)+",")
}

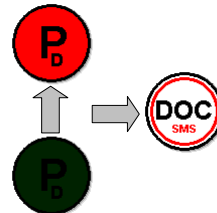
CC "AdoScript" LISTBOX entries: (patientNames) toksep: ","
SETL pat:(selection)

CC "AdoScript" LISTBOX entries: ("Good,Medium,Bad") toksep: ","
SETL diag:(selection)
  
```

1/2

### Mechanisms & Algorithms

#### "Update Diagnosis - SMS"



## Mechanisms and Algorithms Implementation on ADOxx

### Mechanisms & Algorithms

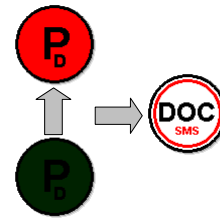
2/2

```
#START
CC "Core" UPDATE_MODEL_LIST
CC "Core" GET_MODEL_ID modelname:("Patients") modeltype:
("Diagnosis")
CC "Core" GET_CLASS_ID classname:"PatientD"
CC "Core" GET_ATTR_ID classid:(classid) attrname:"Diagnosis"
CC "Core" GET_OBJ_ID modelid:(modelid) classid:(classid)
objname:val:(pat)
CC "Core" SET_ATTR_VAL objid:(objid) attrid:(attrid) val:(diag)

SET dr_aql_str: ("{" + (pat) + "}" <- "dRp")
CC "AQL" EVAL_AQL_EXPRESSION expr:(dr_aql_str) modelid:
(modelid)
SET docIDs: (objids)

IF ((diag) = "Bad")
{
FOR docID in: (docIDs)
{
CC "Core" GET_CLASS_ID objid:(VAL docID)
#--> RESULT ecode: intValue classid: intValue isrel: intValue .
SETG docClass:(classid)
CC "Core" GET_ATTR_ID classid:(docClass) attrname:("Emergency")
CC "Core" GET_ATTR_VAL objid:(VAL docID) attrid:(attrid)
IF ((LEN val)>0)
{
SETL t1:(val)
SETL t2:("Dr. Notification...")
SYSTEM ("C:\\test.bat \"\"+t1+\"\" \"\"+t2+\"\"")
EXIT
}
}
}
}
```

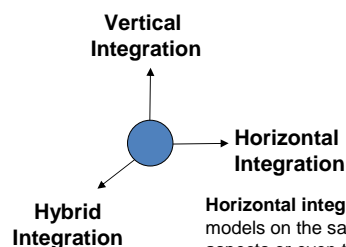
### "Update Diagnosis - SMS"



## Integration Approaches



**Vertical integration** represents a top-down or bottom-up integration approach. Metamodels and models with different level of details are integrated.



**Horizontal integration** integrates enterprise models on the same level of details. Different aspects or even the same aspects but from different viewpoints are integrated.

**Hybrid integration** is the combination of the vertical and horizontal integration approach.

Source: Kühn H. (2004). Methodenintegration im Business Engineering, PhD Thesis, University of Vienna, April 2004.

## Heterogeneity Dimensions

### ► Syntactical

- Heterogeneity of formats
- Unstructured, semi-structured and structured formats
- Example: Serialization formats: XML-based, object-oriented, relational formats, etc.

### ► Structural

- *Representational heterogeneity* (different modelling primitives available – expressive power of language may vary) - Example: Some languages support multiple inheritance, others are restricted to single parent class.
- *Schematic heterogeneity* (concepts are described differently, while seen from different viewpoints) - Example: the concept "Performer" is defined as a simple attribute of a class, whereas the same concept can be described with two classes: the "Worker" and its generalisation "Processor".

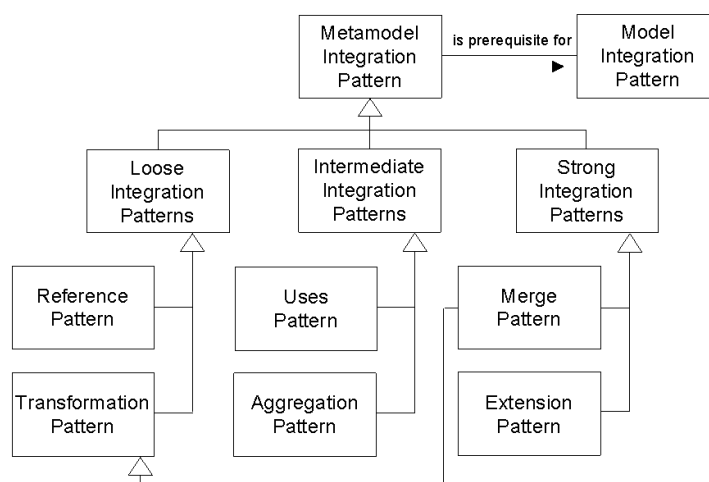
### ► Semantic

- differences in the meaning of the concepts under consideration
- Semantically equivalent concepts (e.g. synonyms)
- Semantically related concepts (relation types such as "is-a", "has-a", "type-of" and "associate" etc)
- Unrelated concepts (completely orthogonal semantics – homonyms?)

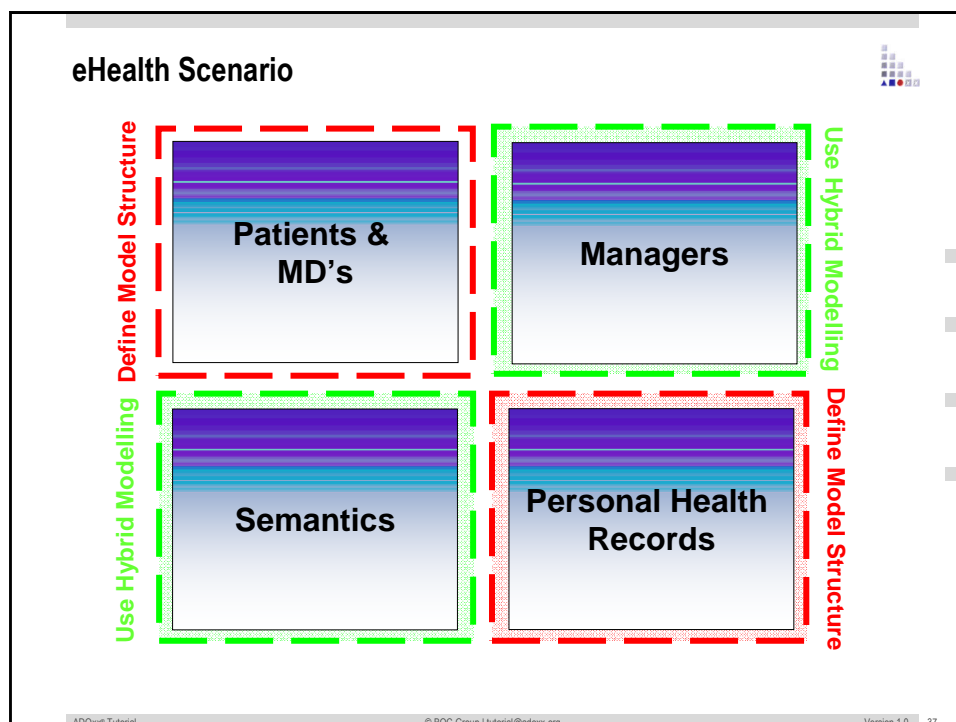
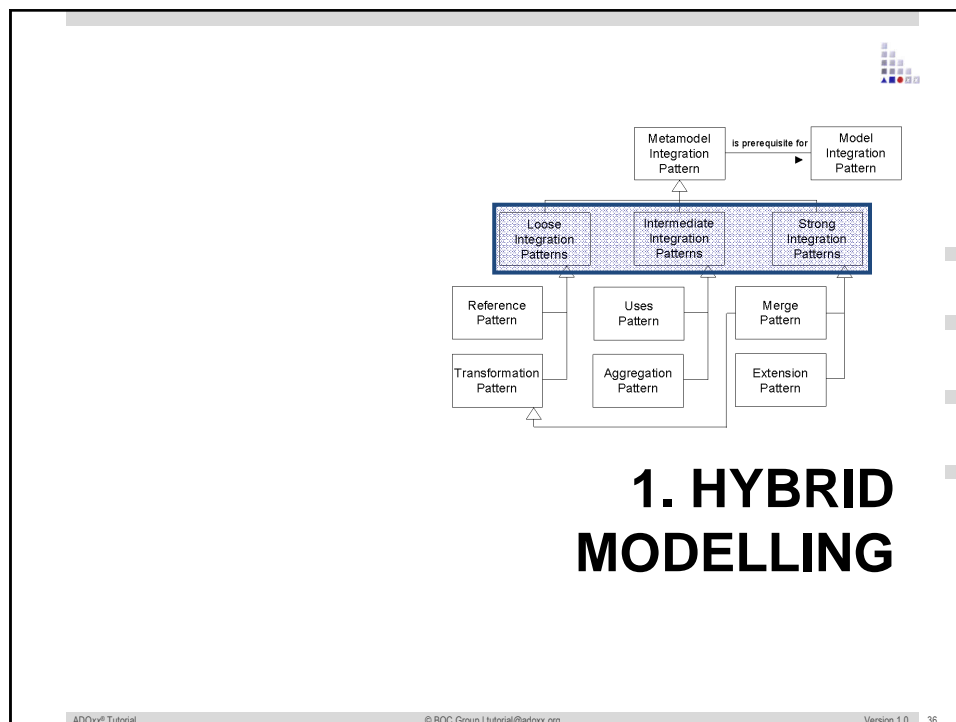
**Heterogeneity dimensions occur on all level of metamodeling hierarchy**

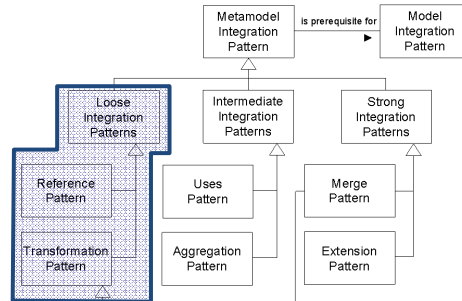
Source: Kühn H. (2004). Methodenintegration im Business Engineering, PhD Thesis, University of Vienna, April 2004.

## Integration Patterns



Source: Kühn H. (2004). Methodenintegration im Business Engineering, PhD Thesis, University of Vienna, April 2004.

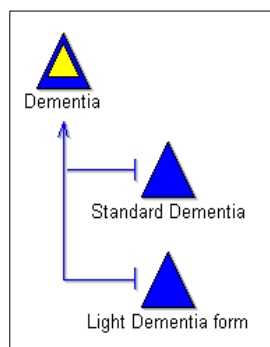




# 1. HYBRID MODELLING

## Sample of Loose Pattern Coupling on ADOxx

Hybrid Modelling



```
#RootConcept
GRAPHREP

FILL color:blue
POLYGON 3 x1:-0.5cm y1:0.5cm x2:0cm y2:-0.5cm x3:0.5cm y3:0.5cm
FILL color:yellow
POLYGON 3 x1:-0.3cm y1:0.3cm x2:0cm y2:-0.3cm x3:0.3cm y3:0.3cm

ATTR "Name" x:0cm y:0.55cm w:c
```

```
#Concept
GRAPHREP

FILL color:blue
POLYGON 3 x1:-0.5cm y1:0.5cm x2:0cm y2:-0.5cm x3:0.5cm y3:0.5cm

ATTR "Name" x:0cm y:0.55cm w:c
```

```
GRAPHREP
PEN color:blue
EDGE

START
LINE x1:0cm y1:0.2cm x2:0cm y2:-0.2cm
END
LINE x1:-.3cm y1:.1cm x2:0cm y2:0cm
LINE x1:-.3cm y1:-.1cm x2:0cm y2:0cm
```



## Sample of Loose Pattern Coupling on ADOxx

Hybrid Modelling

AdoScript code for creating a menu entry that starts every Manager that is active.

```
ITEM "Start Active Managers"
  acquisition:"ADOxx Tutorial" modeling:"ADOxx Tutorial" analysis:"ADOxx Tutorial"
  simulation:"ADOxx Tutorial" evaluation:"ADOxx Tutorial" importexport:"ADOxx Tutorial"
  "

CC "Core" UPDATE_MODEL_LIST
CC "Core" GET_MODEL_ID modelname:("Management") modeltype: ("Management")
#--> RESULT ecode: intValue modelid: intValue
SET
CC "AQL" EVAL_AQL_EXPRESSION expr:("<"Manager">[?"Active\" like \"*yes*\"]")
modelid:(modelid)
#--> RESULT ecode: intValue objids: strValue .
#CC "Core" ECODE_TO_ERRTEXT ecode:(ecode)
SET mgrIdStrings: (objids)

FOR mgrID in: (mgrIdStrings)
{
  CC "Core" GET_ATTR_VAL objid:(VAL mgrID) attrname:("Action")
  #--> RESULT ecode: intValue val: anyValue
  SET crt_action:""
  SET crt_action:("SET mgr_id: \"\"+(mgrID)+\"\" \n \"+(val))
  CC "AdoScript" debug EXECUTE (crt_action)
}
```

## Sample of Loose Pattern Coupling on ADOxx

Hybrid Modelling

```
CC "AdoScript" INFOBOX ("Manager: " + (mgr_id))

#variable mgr_id (string) 'passed' when calling the CC
"AdoScript" EXECUTE statement for starting the manager
CC "Core" GET_ATTR_VAL objid:(VAL mgr_id)
attrname:("Referenced Model")
#--> RESULT ecode: intValue val: anyValue
SET modelid: (val)

SET model_name:"Patients"
SET model_type: "Diagnosis"

CC "Core" GET_MODEL_ID modelname: (model_name) modeltype:
(model_type)
# --> RESULT ecode: intValue modelid: intValue .
SET myModelid: (modelid)

CC "Core" GET_CLASS_ID relation classname: "dRp"
#--> RESULT ecode: intValue classid: intValue
SET rel_id: (classid)

CC "AQL" EVAL_AQL_EXPRESSION expr:("<"PatientD">")
modelid: (myModelid)
#--> RESULT ecode: intValue objids: strValue .
#CC "Core" ECODE_TO_ERRTEXT ecode:(ecode)
SET patientIDs: (objids)
```

1/2

AdoScript code for designating a DOC to every Patient that is unattended

2/2

```
FOR patientID in: (patientIDs)
{
  CC "Core" GET_OBJ_NAME objid: (VAL patientID)
  # --> RESULT ecode: intValue objname: strValue
  SET pat_name: (objname)

  CC "AQL" EVAL_AQL_EXPRESSION
  expr:("<"\""+(pat_name)+"\">\"<- \"dRp\"") modelid:
(myModelid)
  #--> RESULT ecode: intValue objids: strValue .
  SET docs:""
  SET docs: (objids)

  IF (LEN (docs) = 0)
  {
    CC "AQL" EVAL_AQL_EXPRESSION
    expr:("<"DOC\">[?"Number of Patients\" < 3 ]")
    modelid: (myModelid)
    #--> RESULT ecode: intValue objids: strValue .
    SET free_docs: (objids)
    FOR my_doc in: (free_docs)
    {
      CC "Core" debug CREATE_CONNECTOR modelid:
(myModelid) fromobjid: (VAL my_doc) toobjid:(VAL
patientID) classid: (rel_id)
      # --> RESULT ecode: intValue objid: id .
      IF ((ecode) = 0)
      {
        EXIT
      }
    }
  }
}
```

## Sample of Loose Pattern Coupling on ADOxx

### Hybrid Modelling

```
CC "AdoScript" INFOBOX ("Manager: " + (mgr_id))

#variable mgr_id (string) 'passed' when calling the CC "AdoScript"
EXECUTE statement for starting the manager
CC "Core" GET_ATTR_VAL objid:(VAL mgr_id) attrname:("Referenced
Model")
#--> RESULT ecode: intValue val: anyValue
SET modelid: (val)

SET model_name: "Patients"
SET model_type: "Diagnosis"

CC "Core" GET_MODEL_ID modelname: (model_name) modeltype: (model_type)
# --> RESULT ecode: intValue modelid: intValue .
SET myModelid: (modelid)
CC "Core" GET_CLASS_ID relation classname: "dRp"
#--> RESULT ecode: intValue classid: intValue
SET rel_id: (classid)
CC "AQL" EVAL_AQL_EXPRESSION expr:("<"PatientD">") modelid:
(myModelid)
#--> RESULT ecode: intValue objids: strValue .
#CC "Core" ECODE_TO_ERRTEXT ecode:(ecode)
SET patientIDs: (objids)

FOR patientID in: (patientIDs)
{
  CC "Core" GET_OBJ_NAME objid: (VAL patientID)
  # --> RESULT ecode: intValue objname: strValue
  SET pat_name: (objname)

  CC "Core" GET_ATTR_VAL objid:(VAL patientID) attrname:("CheckPHR")
  #--> RESULT ecode: intValue val: anyValue
  SET checkPHR: ""
  SET checkPHR: (val)
  CC "AdoScript" INFOBOX ("checkPHR : " + (checkPHR) +
  (STR (LEN(checkPHR))))
}
```

1/2

```
2/2
IF ((LEN checkPHR) = 0)
{
  CC "Core" GET_MODEL_ID modelname: ("PHR")
  modeltype: ("HealthRecord")
  # --> RESULT ecode: intValue modelid: intValue
  SET phr_modelid: (modelid)
  CC "Core" GET_CLASS_ID classname: ("PHR")
  #--> RESULT ecode: intValue classid: intValue
  SET phr_classid: (classid)
  CC "Core" debug CREATE_OBJ modelid: (phr_modelid)
  classid: (phr_classid) objname: ("PHR - " +
  (patientID))
  # --> RESULT ecode: intValue objid: id
  SET phr_id:(objid)

  #create INTERREF from Patient to PHR
  CC "Core" GET_CLASS_ID objid: (VAL patientID)
  #--> RESULT ecode: intValue classid: intValue isrel:
  intValue .
  SET patient_classid: (classid)
  CC "Core" GET_ATTR_ID classid: (patient_classid)
  attrname: ("CheckPHR")
  #--> RESULT ecode: intValue attrid: id
  SET checkPHR_attrid: (attrid)
  CC "Core" ADD_INTERREF objid: (VAL patientID)
  attrid: (checkPHR_attrid) tobjid: (phr_id)
  #--> RESULT ecode: intValue .
}

CC "Core" GET_MODEL_ID objid: (phr_id)
#--> RESULT ecode: intValue modelid: intValue .
CC "Core" SAVE_MODEL modelid:(modelid)
```

AdoScript code for creating a PHR for every Patient that has none

## Sample of Loose Pattern Coupling on ADOxx

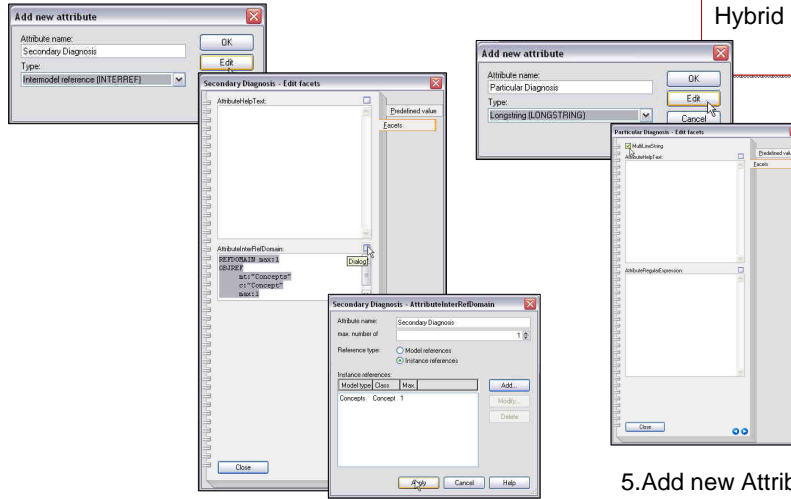
### Hybrid Modelling

The screenshot shows the ADOxx Library management interface. The 'Main Diagnosis' attribute is being added to the 'PHR' class hierarchy. The 'Main Diagnosis' attribute is being added to the 'PHR' class hierarchy. The 'Main Diagnosis' attribute is being added to the 'PHR' class hierarchy.

1. Select PHR Class
2. Add new Attribute "Main Diagnosis"
3. Choose Type "INTERREF"
4. Point to RootConcept class in Model Type Concepts

## Sample of Loose Pattern Coupling on ADOxx

Hybrid Modelling

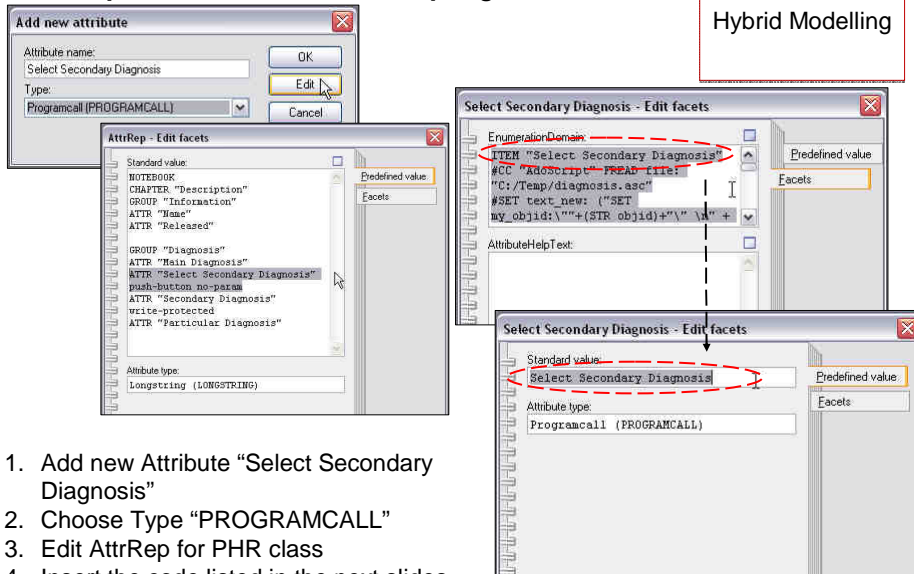


1. Select PHR Class
2. Add new Attribute "Secondary Diagnosis"
3. Choose Type "INTERREF"
4. Point to Concept class in Model Type Concepts

5. Add new Attribute "Particular Diagnosis"
6. Choose Type "LONGSTRING"
7. Select Multiline Facet

## Sample of Loose Pattern Coupling on ADOxx

Hybrid Modelling



1. Add new Attribute "Select Secondary Diagnosis"
2. Choose Type "PROGRAMCALL"
3. Edit AttrRep for PHR class
4. Insert the code listed in the next slides
5. Set the Predefined Value

## Sample of Loose Pattern Coupling on ADOxx

Hybrid Modelling

```
ITEM "Select Secondary Diagnosis"
SET my_objid: (STR objid)
SET myobjjid: (VAL my_objid)

CC "Modeling" GET_ACT_MODEL
#--> RESULT modelid: intValue
SET myModelID: (modelid)

CC "Core" GET_CLASS_ID objid: (myobjjid)
#--> RESULT ecode: intValue classid: intValue isrel: intValue
CC "Core" GET_ATTR_ID classid: (classid) attrname: ("Main Diagnosis")
#--> RESULT ecode: intValue attrid: id
CC "Core" GET_INTERREF objid: (myobjjid) attrid: (attrid) index: 0
# --> RESULT ecode: intValue tmodeltype: strValue tmodelname: strValue tmodelver: strValue (
type:"modelreference" | ( type:"objectreference" toclassname: strValue tobjname: strValue ) ) .

SET aql_str:((("{")+ (tobjname)+ ("":\")+ (tclassname)+ ("":\")+ (tmodelname)+ ("":\")+ (tmodeltype)+ ("":\})<-
\belongs to\"))
CC "AQL" EVAL_AQL_EXPRESSION expr: (aql_str) modelid: (tmodelid)
#--> RESULT ecode: intValue objids: strValue
SET conceptIDs:(objids)

# create the main TreeListBox with all its parameters
CC "AdoScript" TLB_CREATE title:"Select Secondary Diagnosis" oktext:"OK" canceltext:"Cancel" boxtext:"Please
choose one of the following Secondary Diagnoses" no-help:1 button-w:60 max-w:500 max-h:367 min-w:200 min-h:150
sorted: 1 checklistbox:1

FOR conceptID in:(conceptIDs)
{
CC "Core" GET_OBJ_NAME objid: (VAL conceptID)
#--> RESULT ecode: intValue objname: strValue
SET conceptName: (objname)
# insert some entries (as you like - ID should be unique)
CC "AdoScript" TLB_INSERT id:(VAL conceptID) text:(conceptName)
}

# and finally show it
CC "AdoScript" TLB_SHOW
```

AdoScript code to be executed when  
clicking the „Select Secondary  
Diagnosis Button“ in PHR 1/2

## Sample of Loose Pattern Coupling on ADOxx

Hybrid Modelling

```
IF (ecode = 0)
{
#CC "AdoScript" INFOBOX ("Selected ids: " + selectedids + "\n" + "You pushed
the following button: " + endbutton)

CC "Core" GET_CLASS_ID classname: ("PHR")
#--> RESULT ecode: intValue classid: intValue
SET myclassid: (classid)
CC "Core" GET_ATTR_ID classid: (myclassid) attrname: "Secondary
Diagnosis"
#--> RESULT ecode: intValue attrid: id
SET myattrid: (attrid)

#first delete the existing INTERREF, if any
CC "Core" GET_INTERREF_COUNT objid: (myobjjid) attrid: (myattrid)
#--> RESULT ecode: intValue count: intValue .
IF ((count)>0)
{ CC "Core" REMOVE_INTERREF objid: (myobjjid) attrid: (myattrid)
index: 0}

FOR mySelectedID in: (selectedids)
{
CC "Core" ADD_INTERREF objid: (myobjjid) attrid: (myattrid) tobjid:
(VAL mySelectedID)
#Adds only the first selected Secondary Diagnosis
EXIT
}
}
ELSE
{
CC "AdoScript" INFOBOX ("You cancelled the dialog without selecting a
Secondary Diagnosis!")
}
```

2/2

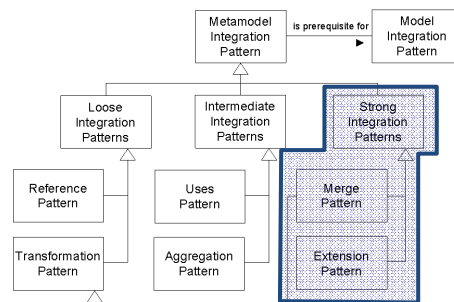
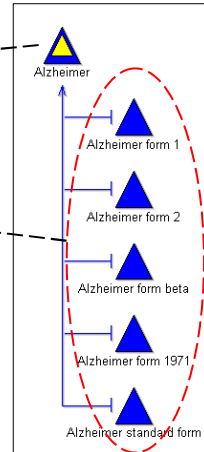
AdoScript code to be executed when clicking the  
„Select Secondary Diagnosis Button“ in PHR

## Sample of Loose Pattern Coupling on ADOxx

Notebook of extended PHR

Hybrid Modelling

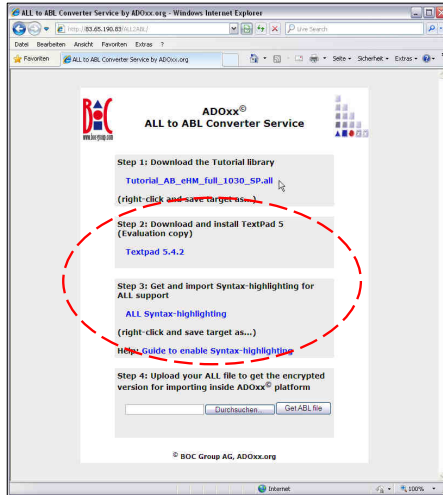
Window for selecting secondary diagnosis related tot the main diagnosis.





## Sample of Strong Pattern Coupling on ADOxx

Hybrid Modelling



- Using a text editor of your choice add the desired classes and model
- We recommend using TextPad 5 with ALL Syntax highlighting

## Sample of Strong Pattern Coupling on ADOxx

Hybrid Modelling

What is the ALL format?

```
//*****
//
// Date: 02.12.2012 19:29
//
// Generated by ADOxx - Library export -- V 2.0
//*****
//
// The file contains the following libraries:
//
// ADOxx 1.3 Tutorial Library eHM_light (Experimentation Environment)
// ADOxx 1.3 Dynamic Tutorial Library eHM_light (Experimentation Environment)
// ADOxx 1.3 Static Tutorial Library eHM_light (Experimentation Environment)
//*****
//
// Defined model types:
//
// 1. Sample
//   Class A
//   Class B
//   Class C
//   Class D
//   Class E
//   Class V
//   Class W
//   Relationclass aRb
//
// 2. Sample-Target
//   Class T
//
//*****
//
// VERSION <4.0>
```

Header

- description of the contents of the library
- name and version of the tool that generated the library

## Sample of Strong Pattern Coupling on ADOxx

```
//=====
//=====
APPLICATION LIBRARY <ADOxx 1.3 Tutorial Library (Experimentation Environment)>
//=====
//=====
//=====
RECORDCLASS <a> : <RecordClass>
//=====
=
CLASSATTRIBUTE <ClassAbstract>
VALUE 0

CLASSATTRIBUTE <ClassVisible>
VALUE 1

CLASSATTRIBUTE <AttrRep>
VALUE "NOTEBOOK"
CHAPTER \"Description\"
ATTR \"aa1\"
ATTR \"aa2\" "
```

### Hybrid Modelling

#### The common section

- describes the main library
- contains definitions of RecordClasses.

## Sample of Strong Pattern Coupling on ADOxx

```
//=====
//=====
BUSINESS PROCESS LIBRARY <ADOxx 1.3 Dynamic Tutorial Library (Experimentation Environment)>
//=====
//=====
ATTRIBUTE <Version number>
VALUE ""
ATTRIBUTE <Date last changed>
VALUE "02.12.2012, 11:05"
ATTRIBUTE <Last user>
VALUE "Admin"
ATTRIBUTE <Keywords>
VALUE "ADOxx - EMPTY LIBRARY"
ATTRIBUTE <Comment>
VALUE ""
ATTRIBUTE <Description>
VALUE ""
ATTRIBUTE <Modis>
VALUE "MODELTYPE \"Sample\" from:none plural:\"Samples\" pos:0 not-simulateable
bitmap:\"db:\\sample.bmp\""
INCL \"A\"
INCL \"B\"
INCL \"C\"
INCL \"D\"
INCL \"E\"
INCL \"V\"
INCL \"W\"
INCL \"aRb\"
MODELTYPE \"Diagnosis\" from:none plural:\"Diagnoses\" pos:1 not-simulateable bitmap:\"db:\\sample.\"
\"bmp\"
INCL \"PatientD\"
INCL \"DOC\"
INCL \"dRp\"
```

### Hybrid Modelling

#### Dynamic Library Section

- describes the dynamic library
- contains definitions of Library Attributes, Classes, Relation Classes, Predefined Queries

#### IMPORTANT

- Library Attribute „MODI“
- contains definitions of the model types and classed they use

## Sample of Strong Pattern Coupling on ADOxx

```
=====
ATTRIBUTE <External coupling>
VALUE #-----
#---- INIT GLOBAL VARS
ON_EVENT \"AppInitialized\"
{
}
#-----

#-----
ITEM \"AdoScript Debug Shell\"
acquisition:\"Extras\" modeling:\"Extras\" analysis:\"Extras\"
simulation:\"Extras\" evaluation:\"Extras\" importexport:\"Extras\"
#-----
IF (type (adoscript) = \"undefined\") {
SETG adoscript:\"\"
}
CC \"AdoScript\" EDITBOX text (adoscript)
fontname:\"Courier New\" fontheight:12
title:\"Debug code\" oktext:\"Run\"
IF (endbutton = \"ok\") {
SETG adoscript;text
EXECUTE (text)
}
#-----
ITEM \"Update Diagnosis\"
acquisition:\"ADOxx Tutorial\" modeling:\"ADOxx Tutorial\" analysis:\"ADOxx Tutorial\"
simulation:\"ADOxx Tutorial\" evaluation:\"ADOxx Tutorial\" importexport:\"ADOxx Tutorial\"
#EXTENDED
CC \"Modeling\" GET_ACT_MODEL
#--> RESULT modelid: intValue

#get all patients
SET aql_str:(\"<\\\"PatientD\\\">\")
CC \"AQL\" debug EVAL_AQL_EXPRESSION expr:(aql_str) modelid: (modelid)
SET patientIDs: (objjds)
```

2/4

ADOxx® Tutorial

© BOC Group | tutorial@adox.org

Version 1.0 | 58

Hybrid Modelling

Dynamic Library  
Section

### IMPORTANT

- Library Attribute „External Coupling“
- contains definitions of event handlers
- contains definitions of customized menu items and the AdoScript code for these menu items

## Sample of Strong Pattern Coupling on ADOxx

```
=====
CLASS <DOC> : < D-construct >
=====
CLASSATTRIBUTE <ClassAbstract>
VALUE 0
CLASSATTRIBUTE <ClassVisible>
VALUE 1
CLASSATTRIBUTE <GraphRep>
VALUE \"GRAPHREP\"
PEN color:black w:0.1cm
FILL color:white
ELLIPSE x:0.00cm y:0cm rx:1cm ry:1cm
AVAL set-default:\"\" emergency:\"Emergency\"
IF ((LEN emergency)>0)
PEN color:red w:0.1cm
FILL color:white
ELLIPSE x:0.00cm y:0cm rx:0.86cm ry:0.86cm
FONT h:0.3cm color:red bold
TEXT \"SMS\" x:-0.29cm y:0.3cm
ENDIF
FONT h:0.7cm color:$000000 bold
TEXT \"DOC\" x:-0.75cm y:-0.38cm
FONT h:0.5cm color:$000000 bold
ATTR \"Name\" y:1.4cm w:c h:c\"

CLASSATTRIBUTE <AttrRep>
VALUE \"NOTEBOOK\"
CHAPTER \"Description\"
GROUP \"Description\"
ATTR \"Name\"
ATTR \"Status\"
ATTR \"Duration\"
ATTR \"Number of Patients\"
ATTR \"Daily rate\"

GROUP \"Contact\"
ATTR \"Emergency\"
```

3/4

ADOxx® Tutorial

© BOC Group | tutorial@adox.org

Version 1.0 | 59

Hybrid Modelling

Dynamic Library Section

CLASS subsection

- contains definitions of classes and their attributes

## Sample of Strong Pattern Coupling on ADOxx

```

=====
RELATIONCLASS <dRp>
  FROM <DOC>
  TO <PatientD>
=====
//--- Relationclass <dRp> - Instance attributes-----
  ATTRIBUTE <Positions>
  TYPE STRING
  VALUE ""
    FACET <MultiLineString>
    VALUE 0
    FACET <AttributeHelpText>
    VALUE ""
    FACET <AttributeRegularExpression>
    VALUE ""

  ATTRIBUTE <GraphRep>
  TYPE STRING
  VALUE "GRAPHREP"

START
FILL color:black
ELLIPSE x:0cm rx:15cm ry:15cm
END
LINE x1:-.3cm y1:1cm x2:0cm y2:0cm
LINE x1:-.3cm y1:-.1cm x2:0cm y2:0cm
"

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

  ATTRIBUTE <AttrRep>
  TYPE STRING
  VALUE ""

```

Hybrid Modelling

Dynamic Library Section

RELATIONCLASS subsection

- contains definitions of relation classes and their attributes

4/4

ADOxx® Tutorial

© BOC Group | tutorial@adoxx.org

Version 1.0 | 60

## Sample of Strong Pattern Coupling on ADOxx

```

=====
//-----
WORKING ENVIRONMENT LIBRARY <ADOxx 1.3 Static Tutorial Library (Experimentation Environment)>
//-----
//-----
  ATTRIBUTE <Version number>
  VALUE ""
  ATTRIBUTE <Date last changed>
  VALUE "30.11.2012, 10:43"
  ATTRIBUTE <Last user>
  VALUE "Admin"
  ATTRIBUTE <Keywords>
  VALUE ""
  ATTRIBUTE <Comment>
  VALUE ""
  ATTRIBUTE <Description>
  VALUE ""
  ATTRIBUTE <Modis>
  VALUE "MODELTYPE \"Sample-Target\" from:none plural:\"Sample-Targets\" pos:1 not-
simulateable bitmap:\"db:\\\"
    \"\\sample-target.bmp\"

INCL \"T\"

MODELTYPE \"Result-Type 1\" from:none plural:\"Result-Types 1\" pos:2
not-simulatable

INCL \"Result-of-Count\"

MODELTYPE \"HealthRecord\" from:none plural:\"HealthRecords\" pos:3 not-simulateable bitmap:\"db:\\\"
    \"sample-target.bmp\"

INCL \"PHR\"

```

Hybrid Modelling

Static Library Section

- describes the static library
- contains definitions of Library Attributes, Classes, Relation Classes, Predefined Queries

**IMPORTANT**

- Library Attribute „MODI“
- contains definitions of the model types and classed they use
- has no „External Coupling“ attribute

1/2

ADOxx® Tutorial

© BOC Group | tutorial@adoxx.org

Version 1.0 | 61

## Sample of Strong Pattern Coupling on ADOxx

```

=====
CLASS <PHR> << S-construct >>
=====
    CLASSATTRIBUTE <ClassAbstract>
        VALUE 0
    CLASSATTRIBUTE <ClassVisible>
        VALUE 1
    CLASSATTRIBUTE <GraphRep>
        VALUE "GRAPHREP"

AVAL b:"Released"
IF (b!="NOK")
PEN color:$000000 w:0.1cm
FILL color:red
RECTANGLE x:1cm y:-1.15cm h:0.6cm w:0.9cm
FONT h:10pt color:$000000 bold
TEXT \"NOK\" x:1.1cm y:-1.05cm
ENDIF
<...>
"

CLASSATTRIBUTE <VisibleAttrs>
    VALUE ""

    CLASSATTRIBUTE <AttrRep>
        VALUE "NOTEBOOK"

CHAPTER \"Description\"
GROUP \"Information\"
ATTR \"Name\"
ATTR \"Released\"

```

Hybrid Modelling

Static Library Section

CLASS subsection

- contains definitions of classes and their attributes

RELATIONCLASS subsection

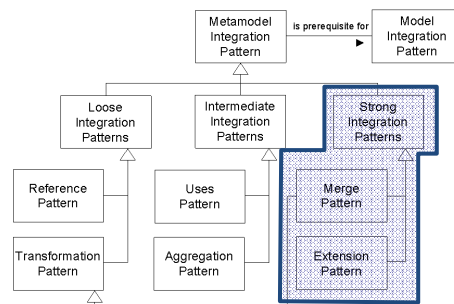
- contains definitions of relation classes and their attributes

2/2

ADOxx® Tutorial

© BOC Group | tutorial@adoxx.org

Version 1.0 | 62



# 1. HYBRID MODELLING

ADOxx® Tutorial

© BOC Group | tutorial@adoxx.org

Version 1.0 | 63

## Sample of Strong Pattern Coupling on ADOxx

Hybrid Modelling  
Hands-On

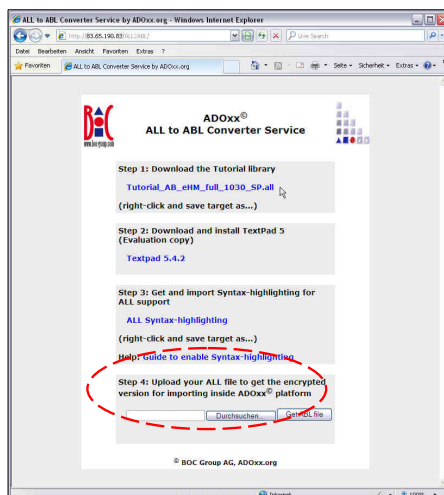
Extending the ALL file

- change the „MODI“ attribute in the „dynamic library“ section, either by typing the code you need or copy/pasting it from another source
- extend the „External Coupling,“ attribute (make sure that in the end the values of both Attributes lie between quotation marks)
- copy or type the new classes and their attributes at the end of the „CLASS“ subsection of the ALL file
- copy or type the new classes and their attributes at the end of the „CLASS“ subsection of the ALL file
- extend the „static library“ section the same way
- make sure that the names of the classes you are adding have not already been used in the ALL file; if that's the case, change the names of the new classes

Hint: Use the „Tutorial Library (Sample eHealth)-Supplement.all“ to see additional classes

## Sample of Strong Pattern Coupling on ADOxx

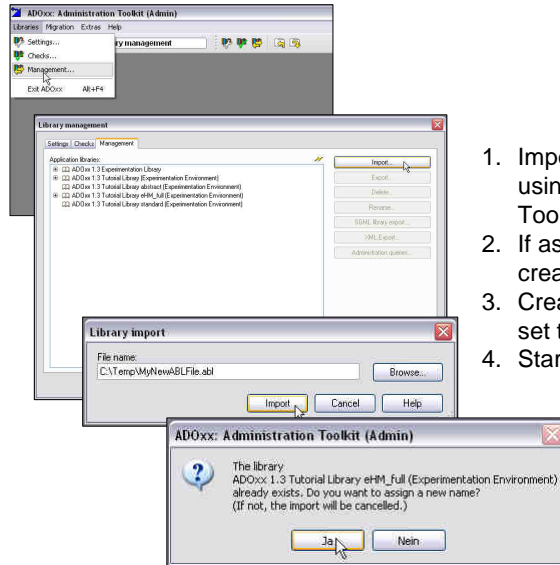
Hybrid Modelling  
Hands-On



1. Upload the extended ALL file to our webservice and get your new ABL file

## Sample of Strong Pattern Coupling on ADOxx

Hybrid Modelling  
Hands-On



1. Import the newly downloaded file using the ADOxx Administration Toolkit
2. If asked provide a new name to the created Libraries
3. Create new users for the library and set the proper user rights
4. Start modelling on your new library

## References

- Kühn H. (2004). Methodenintegration im Business Engineering, PhD Thesis, University of Vienna, April 2004.





In case of any questions, please contact

**tutorial@adoxx.org**