



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

Nesat Efendioglu



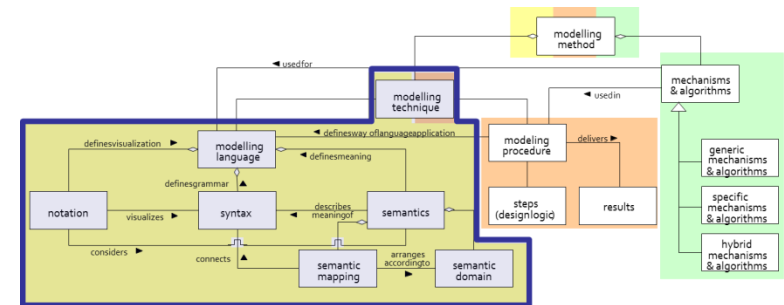
## Selected Scenarios for Training specific Hands-On:

### 1. Realising a **Modelling Language**

- Case: Entity Relationship Model

### 2. Implementing an **Algorithm**

- Case: Structural Similarities of Business Processes



# CASE: Entity Relationship Model

## 1. SCENARIO: REALISING A MODELLING LANGUAGE

# Scenario Description



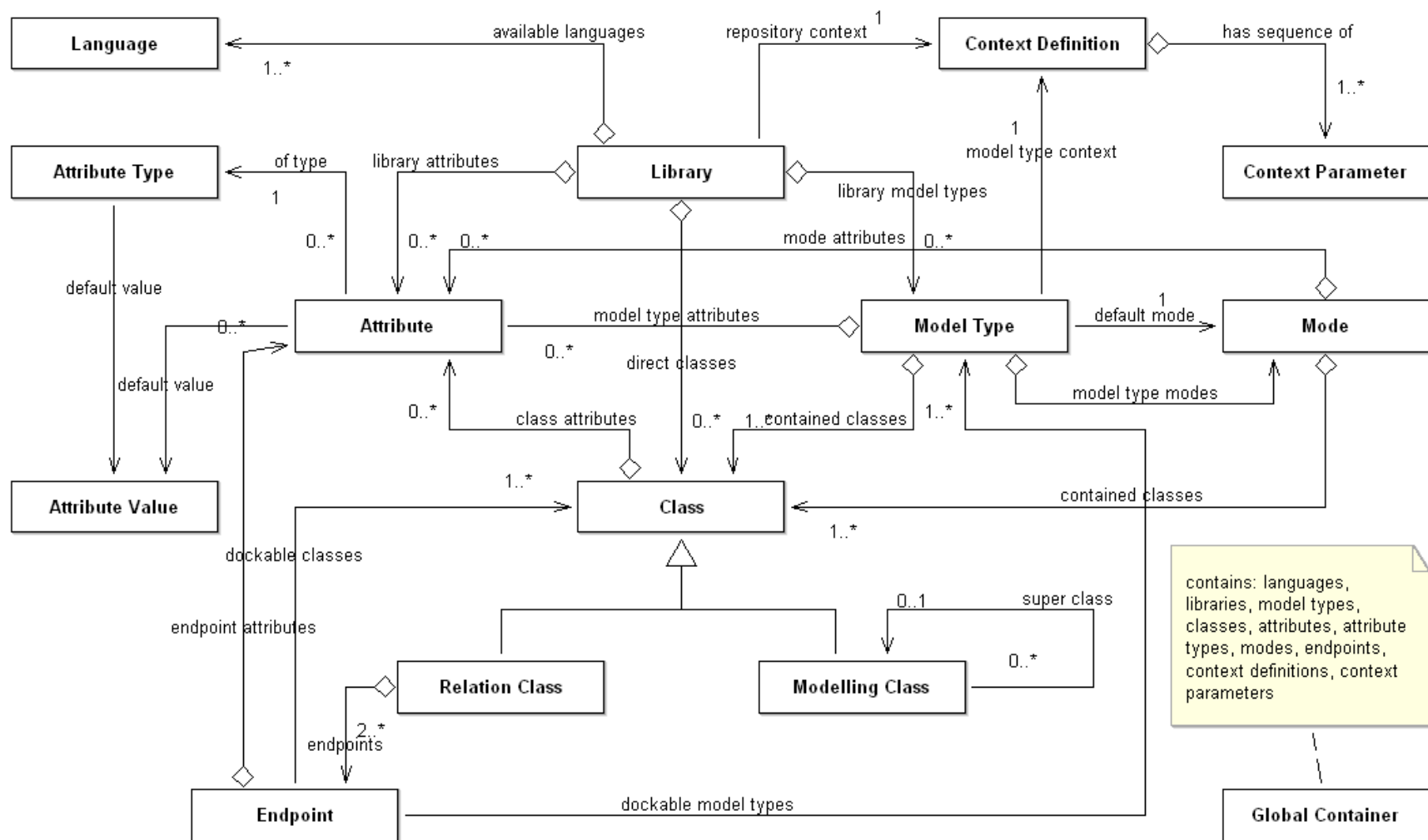
## Case:

Realise a modelling tool for the Modelling Language “Entity Relationship Model”.

## Goal:

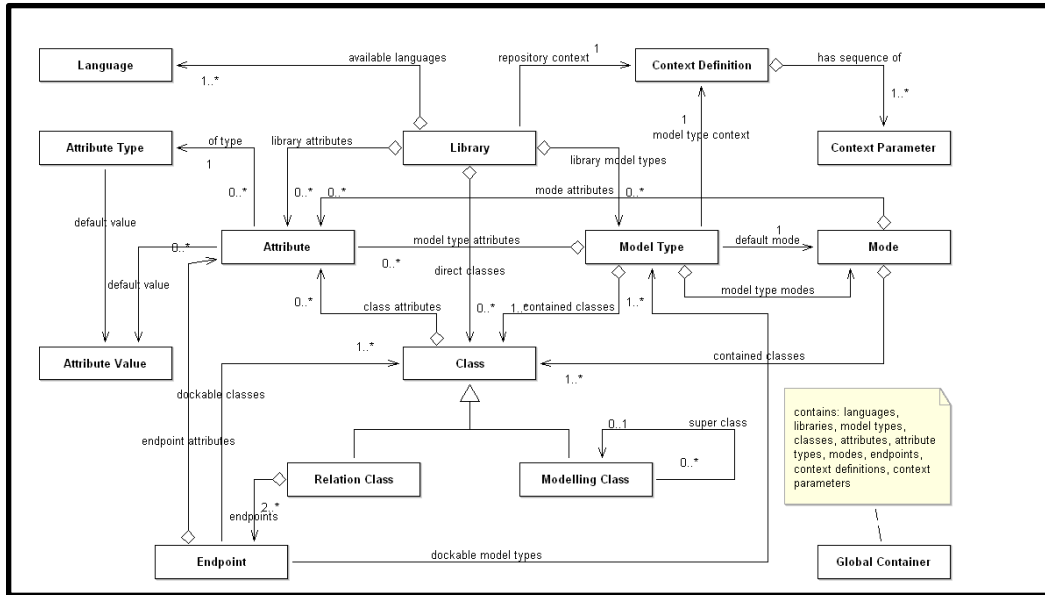
Demonstrate the development of a model editor for a defined modelling languages using common constructs from ADOxx Meta<sup>2</sup>Model.

# ADOxx Meta2Model



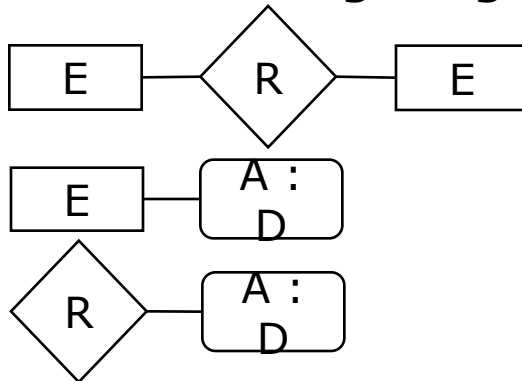
contains: languages, libraries, model types, classes, attributes, attribute types, modes, endpoints, context definitions, context parameters

# Mapping Meta2Model with ER-Meta Model



**How to map  
generic  
Meta<sup>2</sup>Model to a  
concrete  
Modelling  
Language ?**

## ER – Modelling Language Definition

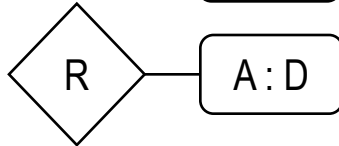
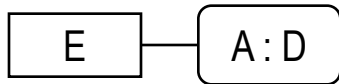
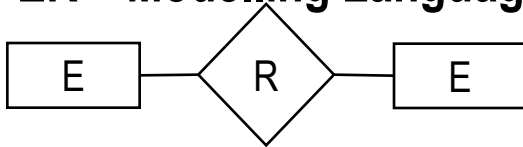


$E(A_1:D_1, \dots, A_m:D_m)$   
 $E(A_1, \dots, A_m)$   
 $R(A_1:D_1, \dots, A_m:D_m)$   
 $R(A_1, \dots, A_m)$

# Modelling Language Meta Model

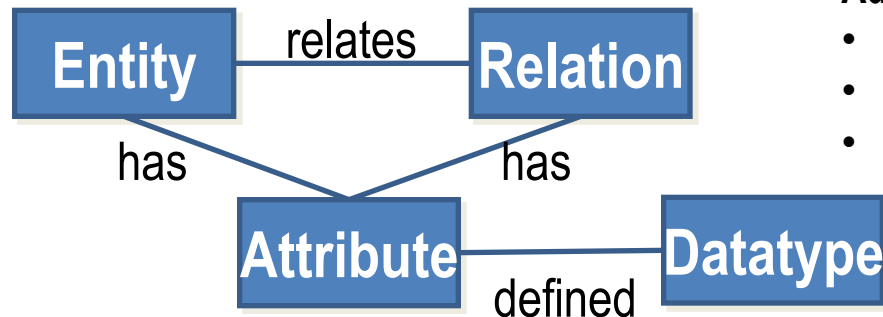


## ER – Modelling Language Definition



$E(A_1:D_1, \dots, A_m:D_m)$   
 $E(A_1, \dots, A_m)$   
 $R(A_1:D_1, \dots, A_m:D_m)$   
 $R(A_1, \dots, A_m)$

## Conceptualised ER – Meta Model



### Additional Aspects:

- Name
- Cardinality
- Key Attribute

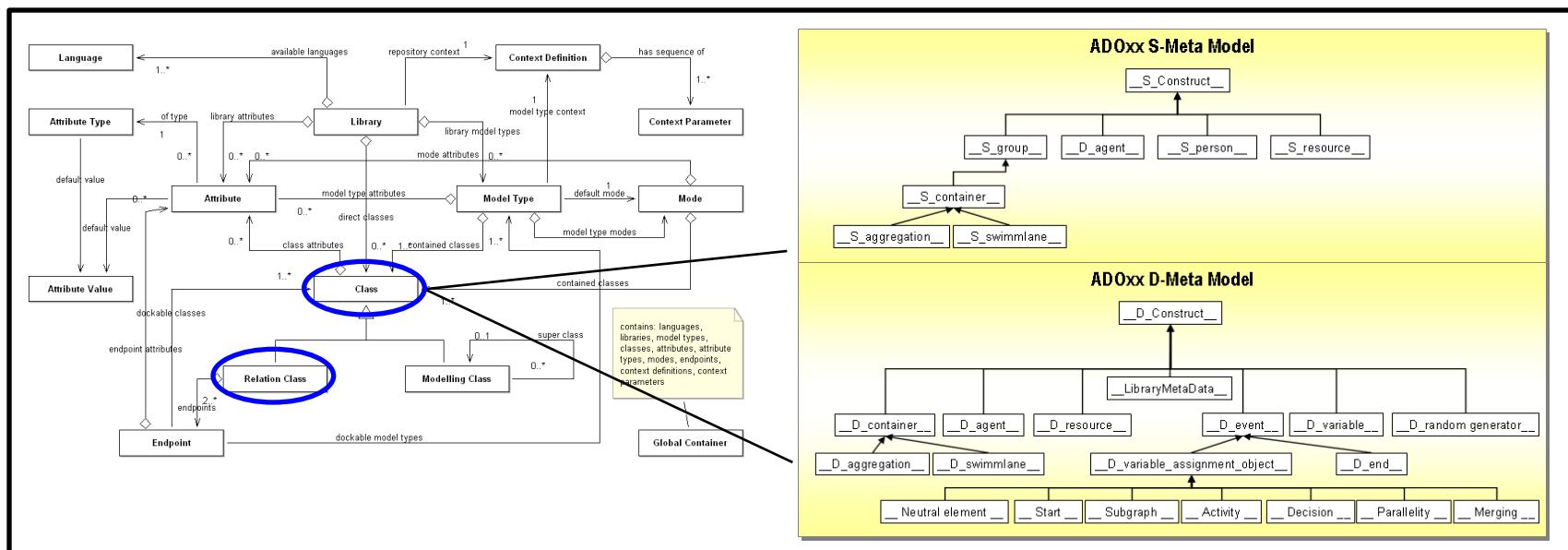
### Specification for Operationalization:

CLASS: Entity, Relation, Attribute,  
RELATIONCLASS: relates, has,  
Attribute: name (String), cardinality (String),  
key (Boolean), datatype (List)



Conceptualisation

# Operationalization of “CLASS” Concept



## Operationalisation : ER Modelling Language

**CLASS:** Entity, Relation, Attribute, `_ER-Concept_`  
**RELATIONCLASS:** relates (Entity->Relation),  
**Attribute:** has (`_ER-Concept_->Attribute`),  
 name (STRING), cardinality (STRING),  
 key (INTEGER), datatype (ENUMERATION)





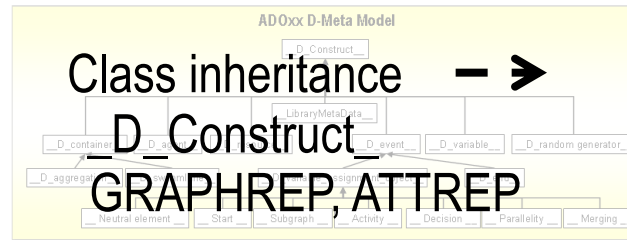
# Operationalisable Meta Model



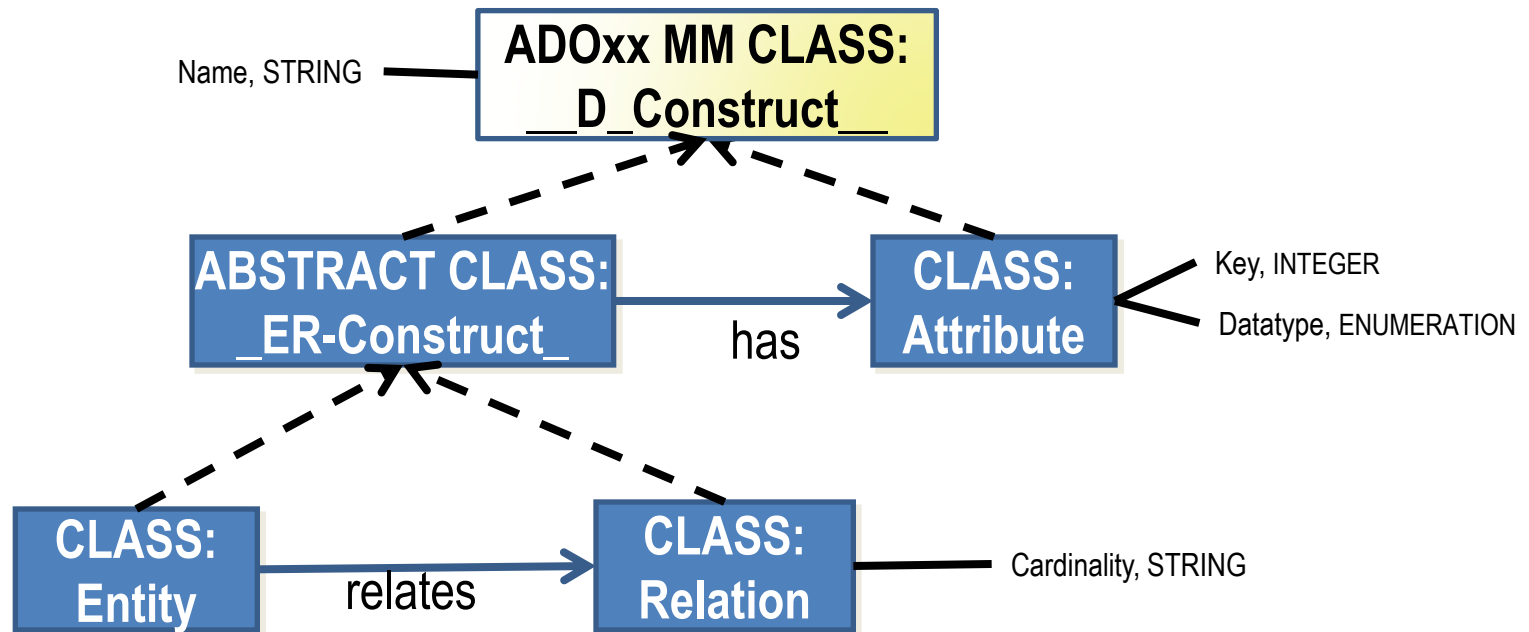
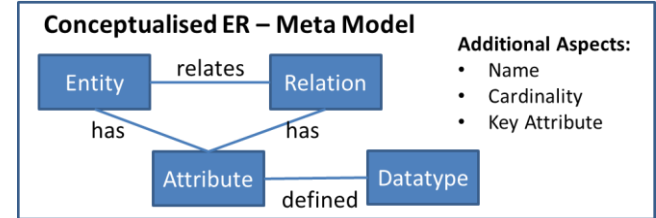
## Meta<sup>2</sup>Model

ABSTRACT CLASS  
CLASS  
RELATIONCLASS  
ATTRIBUTE  
ATTRIBUTETYPE  
{STRING,...ENUMERATION}

## Operationalisation MetaModel



## ER-MetaModel





Used meta-modelling functionality :

- **Meta<sup>2</sup>Model**: MODELTYPE, GRAPHREP, ATTREP, ATTRIBUTE TYPE, CLASS

- **ADOxx Meta2Model Component**:

- Model Editor incl. Menubar
- Query engine incl. AQL syntax
- ADOscript interpreter and ADOscript syntax
- Database



1. Defining **MODELTYPES**
2. Inheriting **CLASSES** from ADOxx Meta Model
3. Implementing **GRAPHREP**
4. Inherit **RELATIONCLASSES** from ADOxx Meta Model
5. Defining **ATTRIBUTES** and **ATTREP**

# Used ADOxx Functionality: Realising a Modelling Language



Introduction

Setup of Implementation Environment

## Modelling Language Implementation

**Classes**

**Relations**

**Class Attributes and Attributes**

**GRAPHREP**

**ATTRREP**

CLASS Cardinality

**CONVERSION**

**Model Pointer**

Attribute Facets

**Model Types**

## Mechanisms & Algorithms Implementation

### Core Functions for Model Manipulation

Database

Visualisation

Query

Transformation

### Configuration of ADOxx Components

Visualisation

Query

### External Coupling ADOxx Functionality

ADOscript Triggers

ADOscript Language Constructs

Visualisation AdoScripts

Visualisation Expression

Query ADOscript

Transformation ADOscript

### ADD-ON Implementation

ADOxx Web-Service

XML / ADL Import – Export

ADOscript Batch Mode



# HANDS-ON

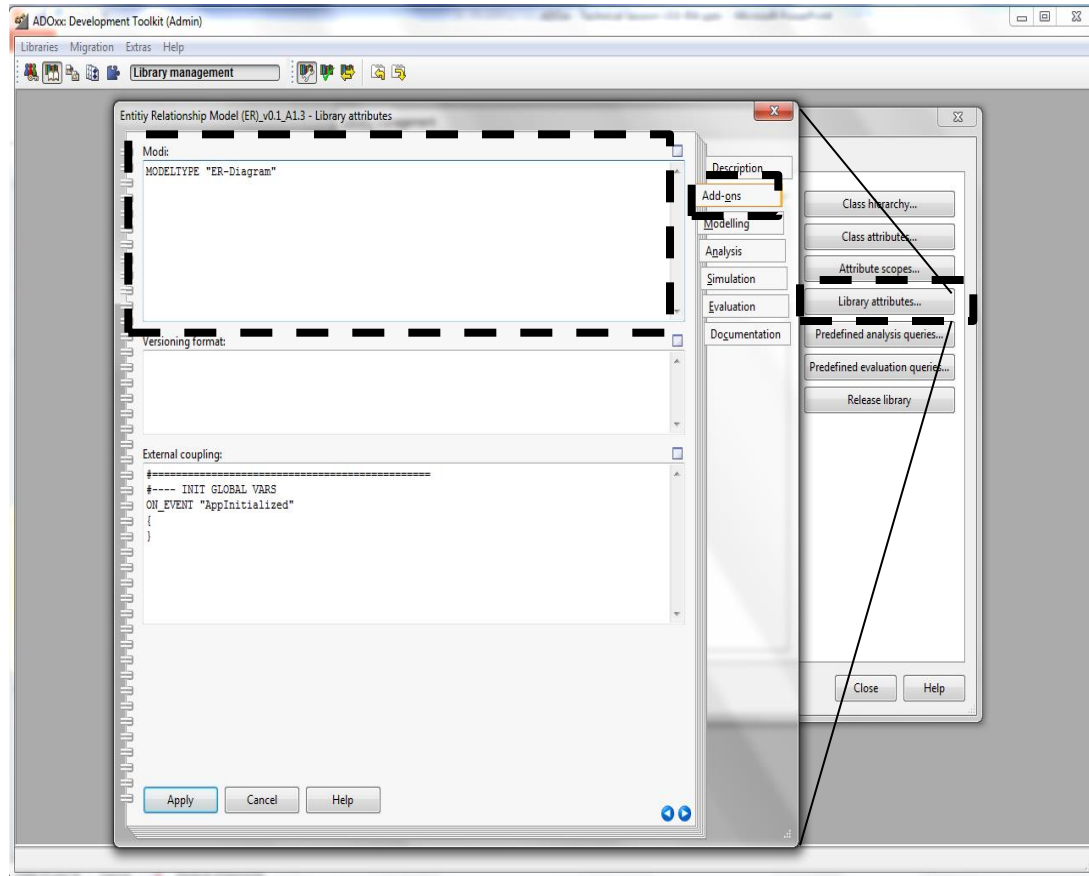
CASE: Entity Relationship Model

## **1. SCENARIO: REALISING A MODELLING LANGUAGE**



1. Defining **MODELTYPES**
2. Inheriting **CLASSES** from ADOxx Meta Model
3. Implementing **GRAPHREP**
4. Inherit **RELATIONCLASSES** from ADOxx Meta Model
5. Defining **ATTRIBUTES** and **ATTREP**

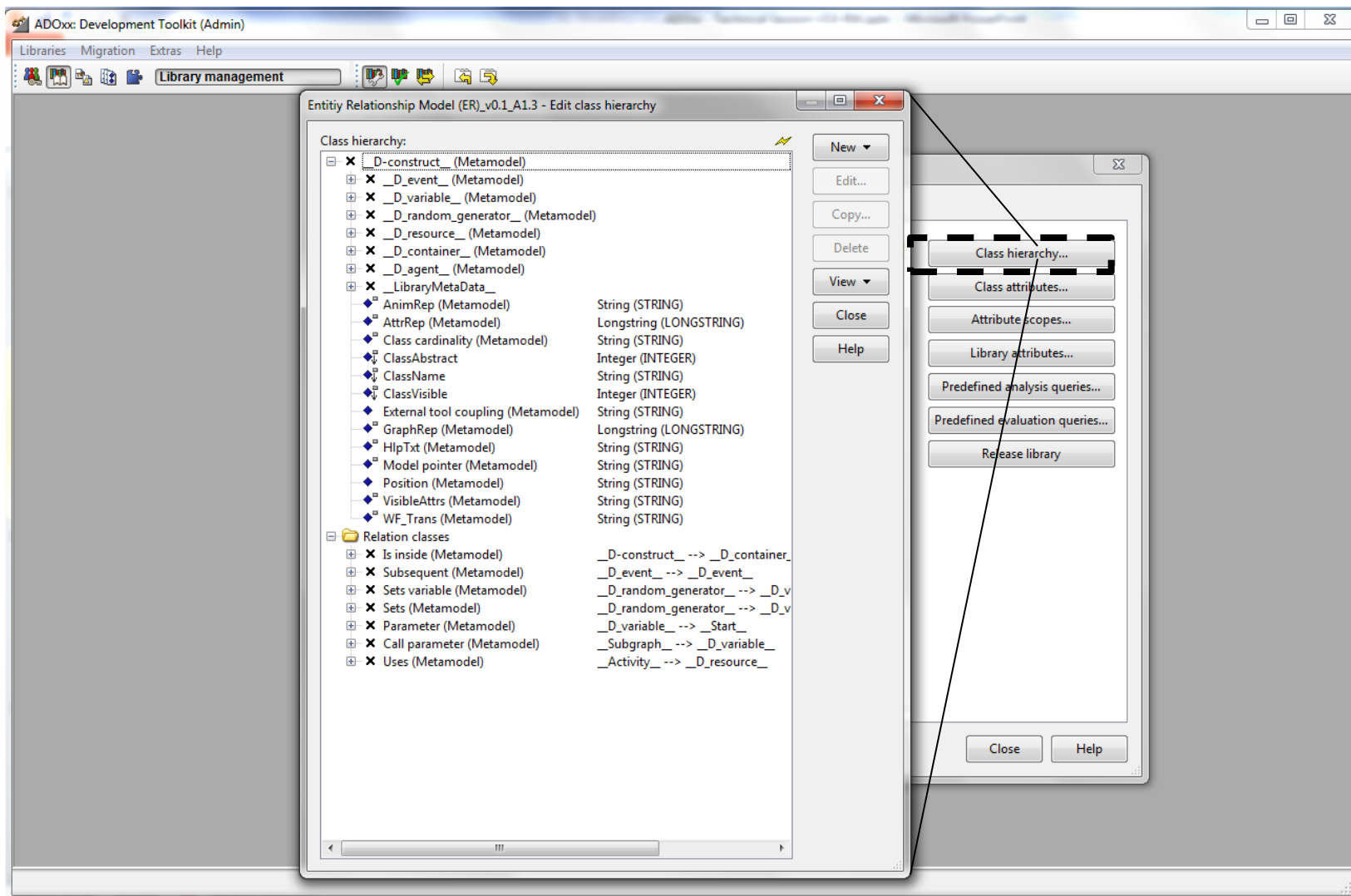
# Define Modeltype



## Define the Model-Type:

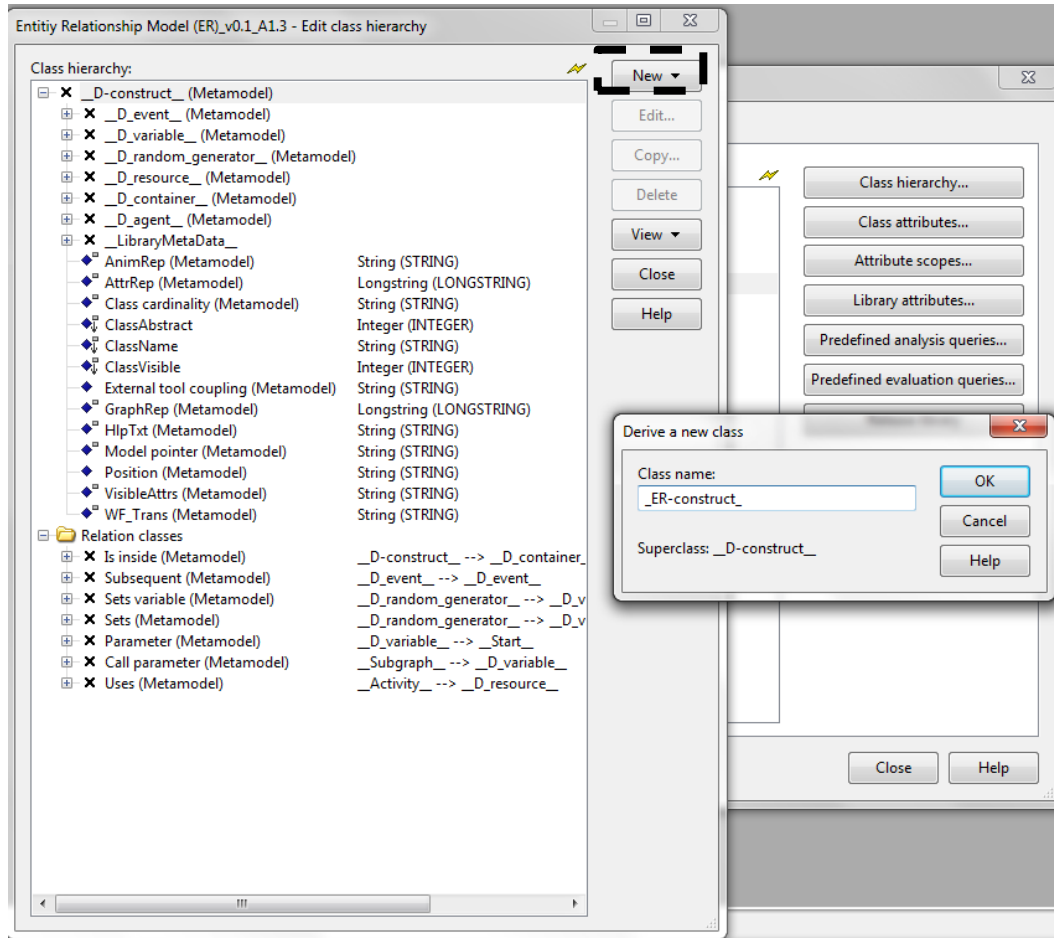
1. Click “Library attribute” of the ER-library
2. Go to “Add-on” chapter
3. Define the Modeltype in the Modi textfield.
4. “MODELTYPE “ER-Diagram”

# Library Management





# Define new class

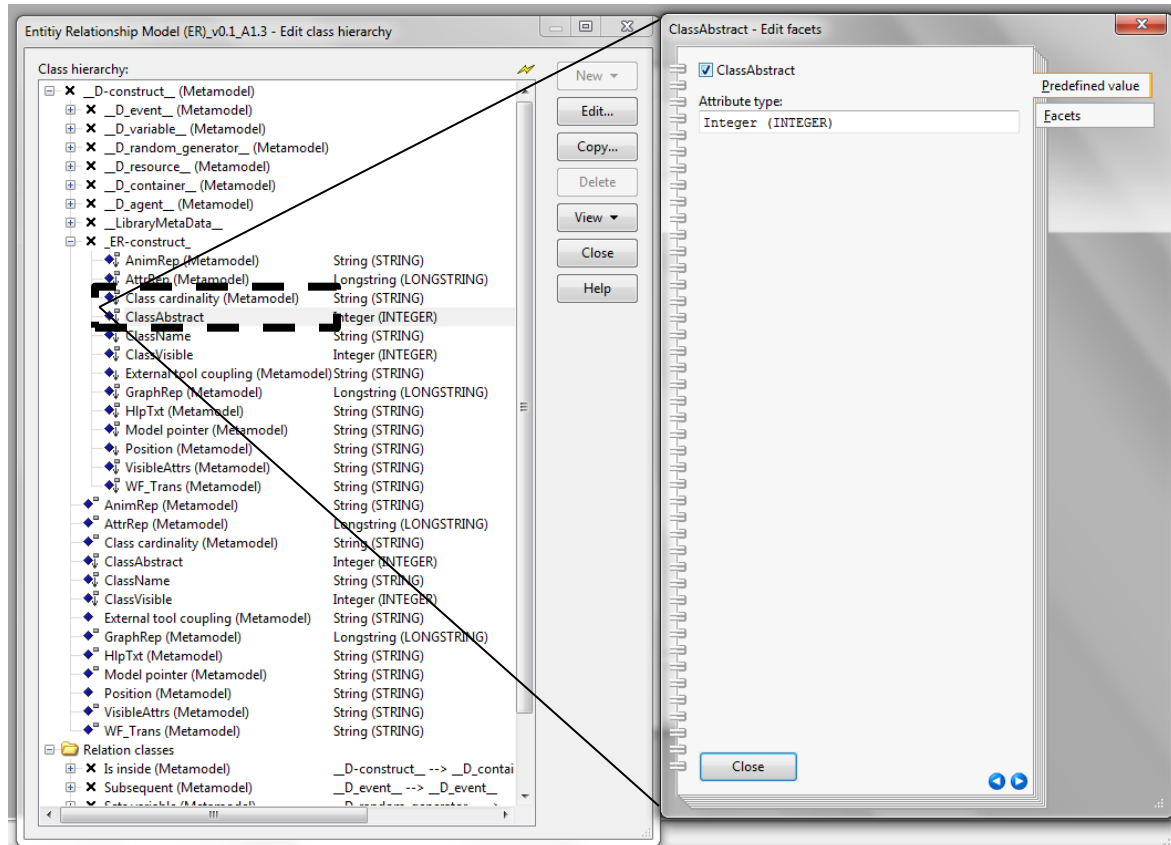


## Define Abstract Class:

Add a new abstract class below the root element that is used to define “\_ER-construct\_” related issues

1. Select root class, click “New” -> “New class”
2. Name new class as an abstract class  
Naming convention for abstract classes Pre- and Post-fix is “\_”.

# Make Class Abstract

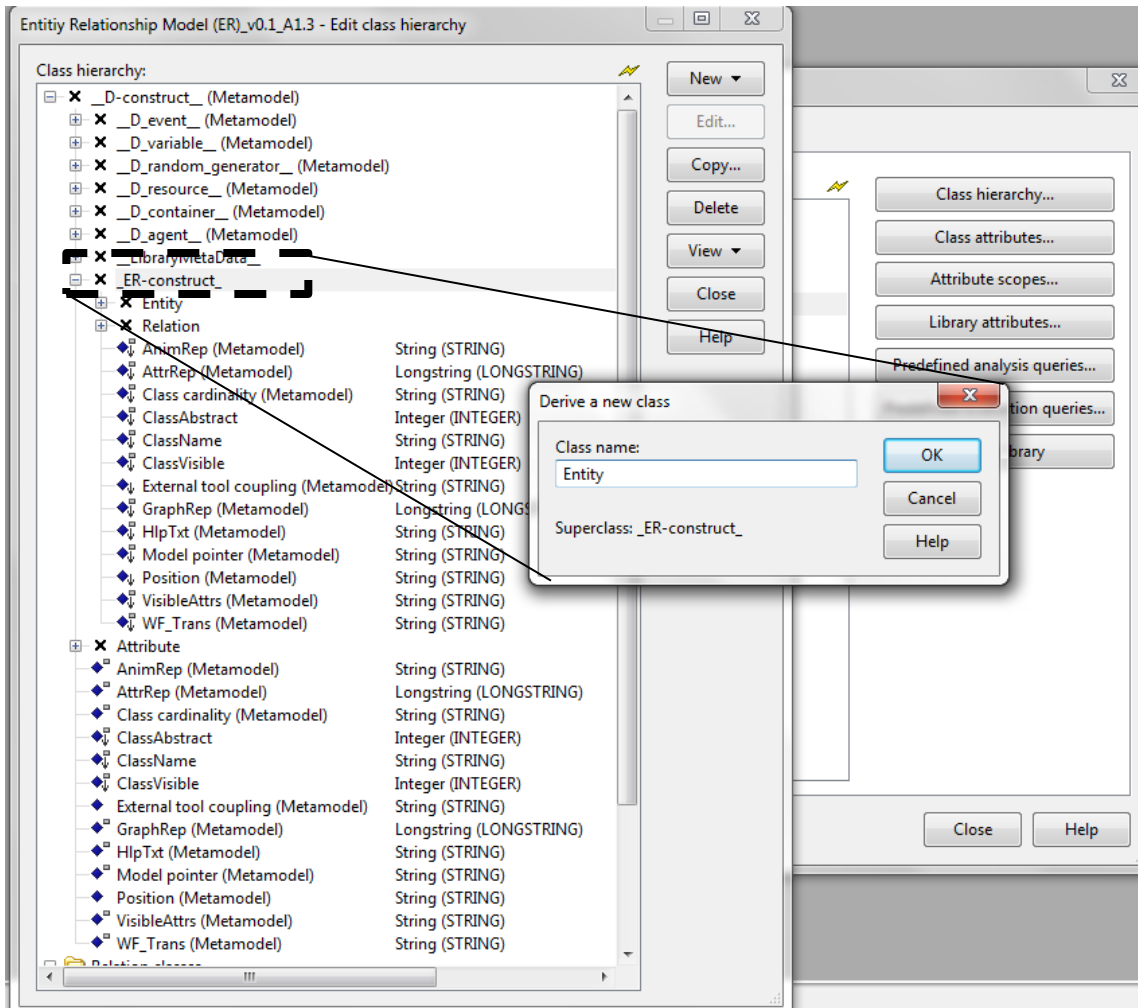


Make class abstract using  
“ClassAbstract” attribute

-> Effect:

class can not be instantiated  
in the modelling tool.

# Define ER-Classes



## Make new classes:

Entity and Relation inherited from

`_ER-construct_`

and

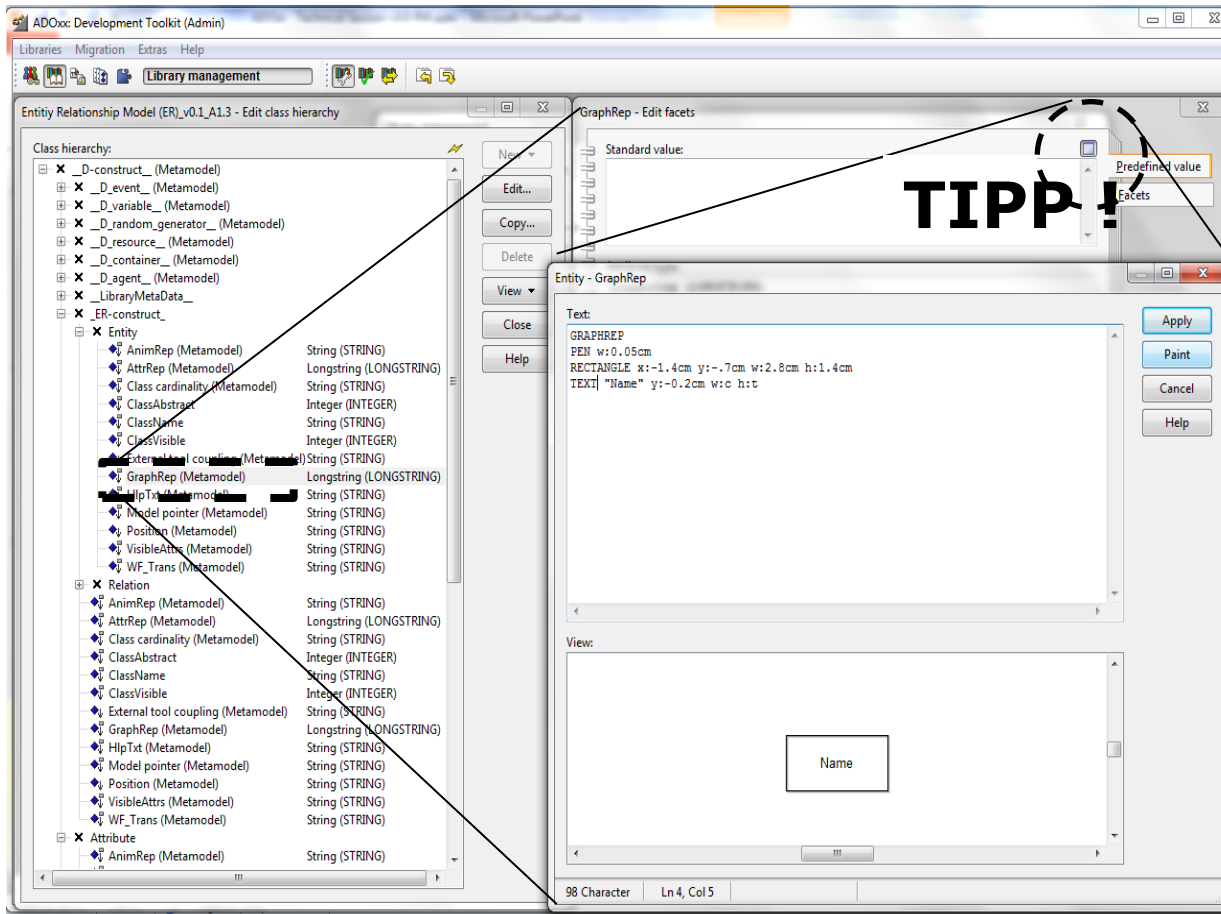
Attribute

Inherited from

`__D-construct__`

Leave them as classes, by not making them abstract.

# Implement GRAPHREP



## Implement GRAPHREP:

Edit Attribute GRAPHREP

Open GRAPHREP Editor

Write GRAPHREP Code in Text field

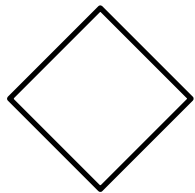
View current GRAPHREP with “Paint”

Store the GRAPHREP with “Apply”

# Implementing GRAPHREP



Entity



Relation



Attribute

GRAPHREP

PEN w:0.05cm

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

GRAPHREP

PEN w:0.05cm

FILL color:white

POLYGON 4 x1:-1.9cm y2:1.6cm x3:1.9cm y4:-1.6cm

ATTR "Name" w:c:2.5cm h:c:0.8cm line-break:rigorous

GRAPHREP

PEN w:0.05cm

ELLIPSE x:0.00cm y:0.00cm rx:3.00cm ry:0.80cm

ATTR "Name" y:-0.2cm w:c:2.8cm h:t

# Advanced Attribute Dependent Graphical Notation

AVAL k:"key"

IF ( k = 1)

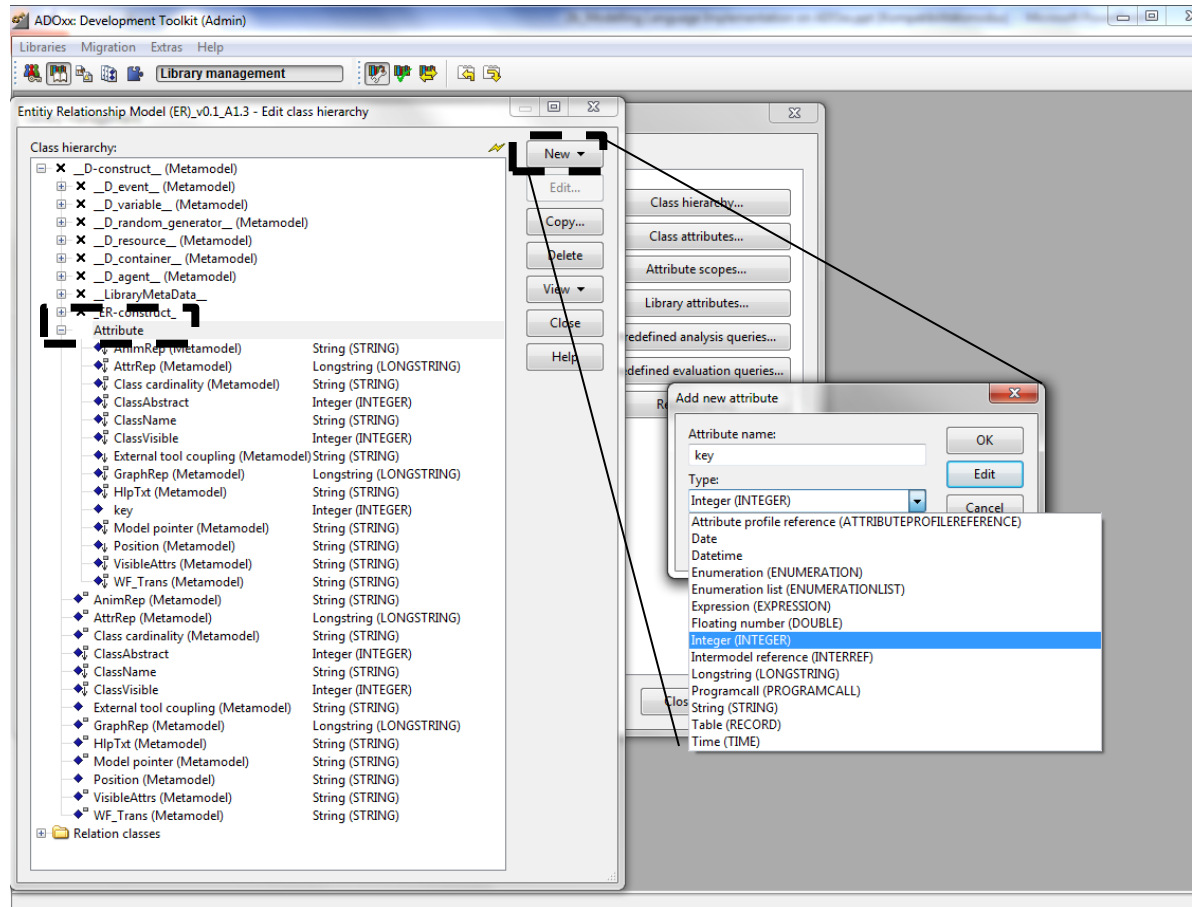
{

LINE x1:-1.7cm y1:-0.5cm x2:-1.7cm y2:0.5cm

LINE x1:-1.5cm y1:-0.6cm x2:-1.5cm y2:0.6cm

}

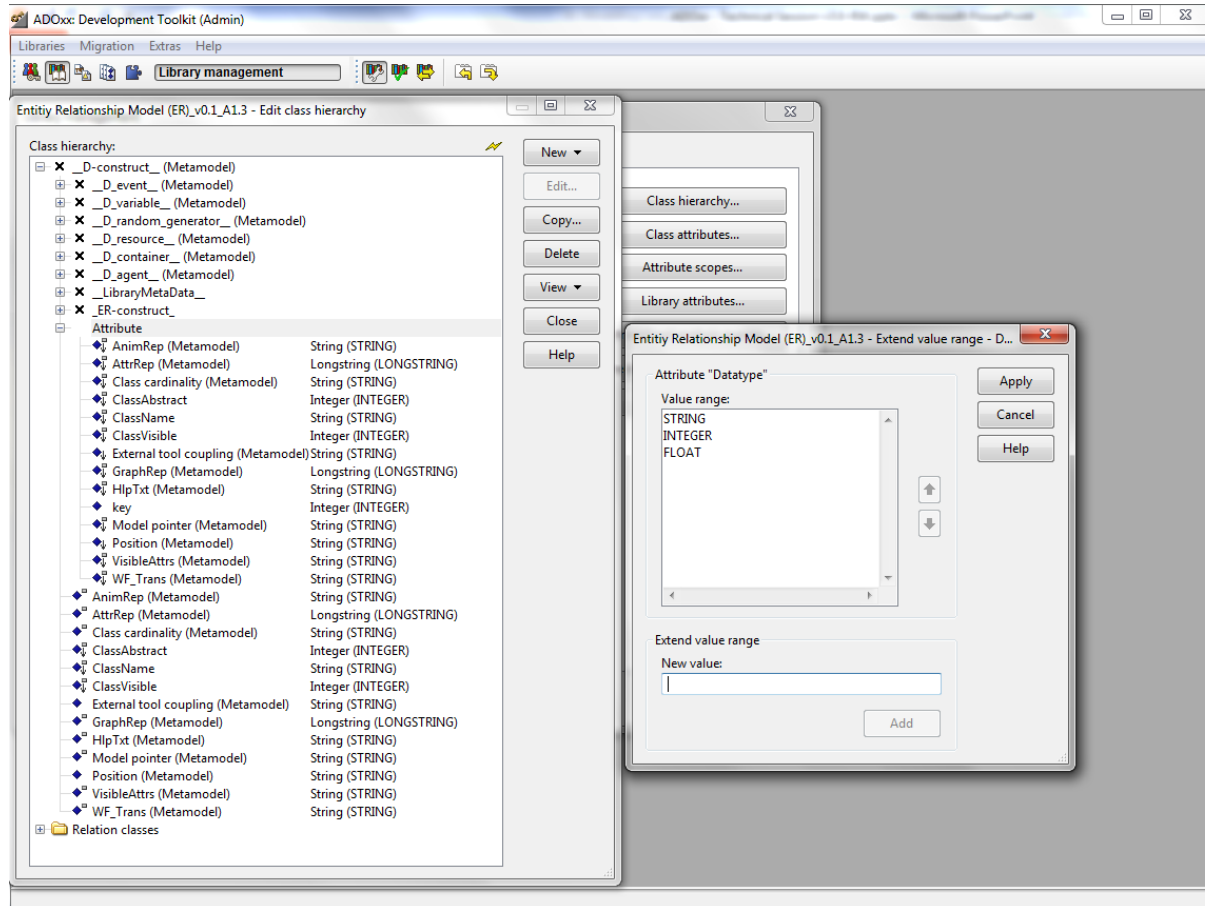
# Attribute Dependent GRAPHREP



## New Attribute

1. Select Class in which the new attribute has to be added
2. Add new attribute
3. Attribute name "key"
4. Type "INTEGER"

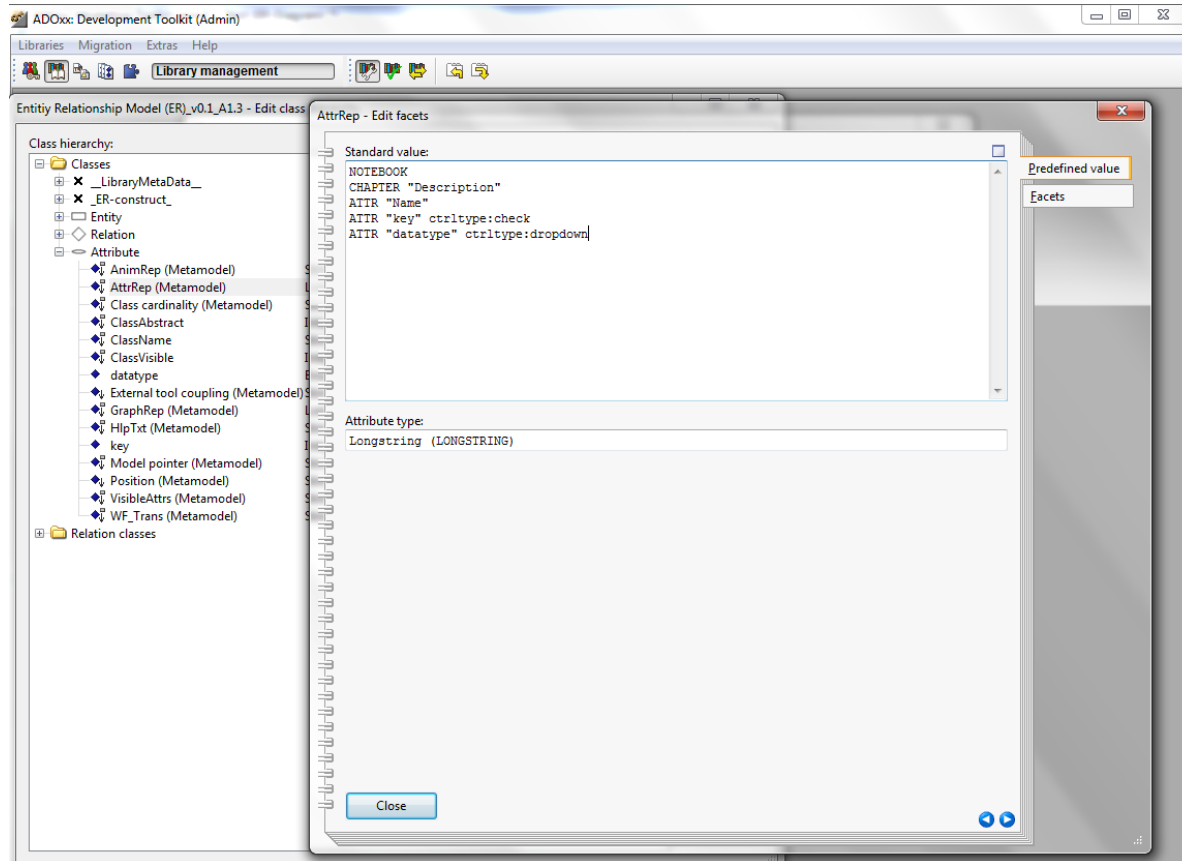
# Attribute Definition



## New Attribute

1. Select Class in which the new attribute has to be added
2. Add new attribute
3. Attribute name "Datatype"
4. Type "ENUMERATION"
5. Add value range

# Implement ATTREP

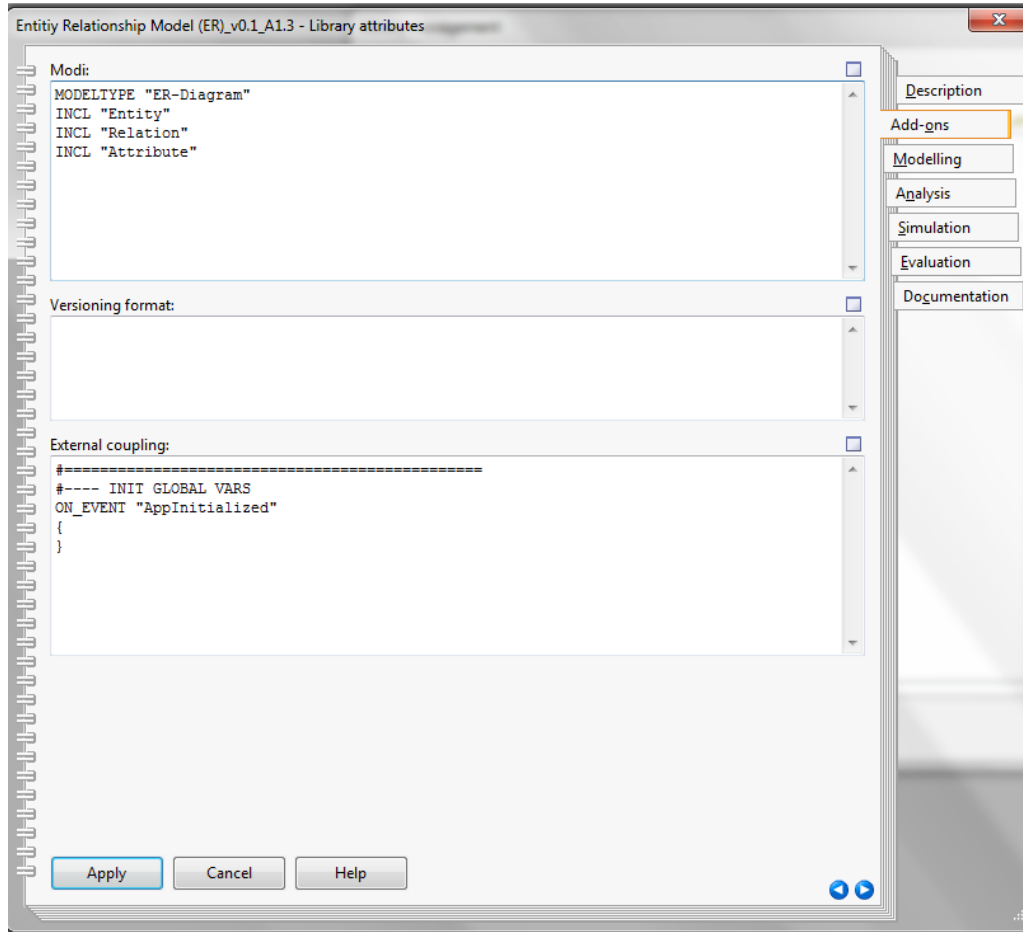


## Implement ATTREP

1. Select Class
2. Edit ATTREP Attribute
3. Define NOTEBOOK
  - Chapter
  - ATTR
  - Group
  - ctrltype



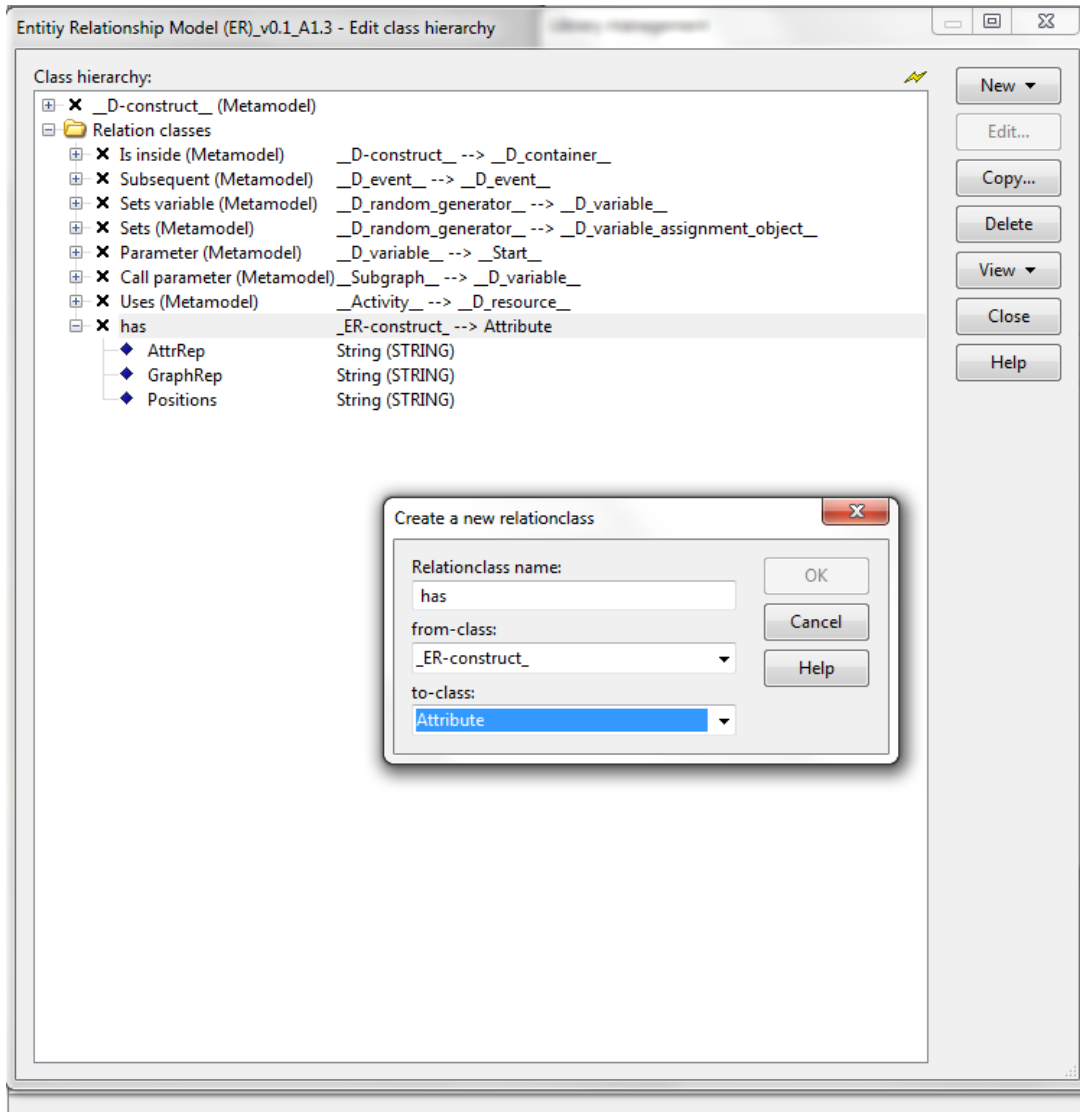
# Modeltype: Inclusion of Classes



## Define the Model-Type:

1. Click “Library attribute” of the ER-library
2. Go to “Add-on” chapter
3. Define the Modeltype in the Modi textfield.
4. “MODELTYPE “ER-Diagram”
5. Include classes:
  - INCL “Entity”
  - INCL “Relation”
  - INCL “Attribute”

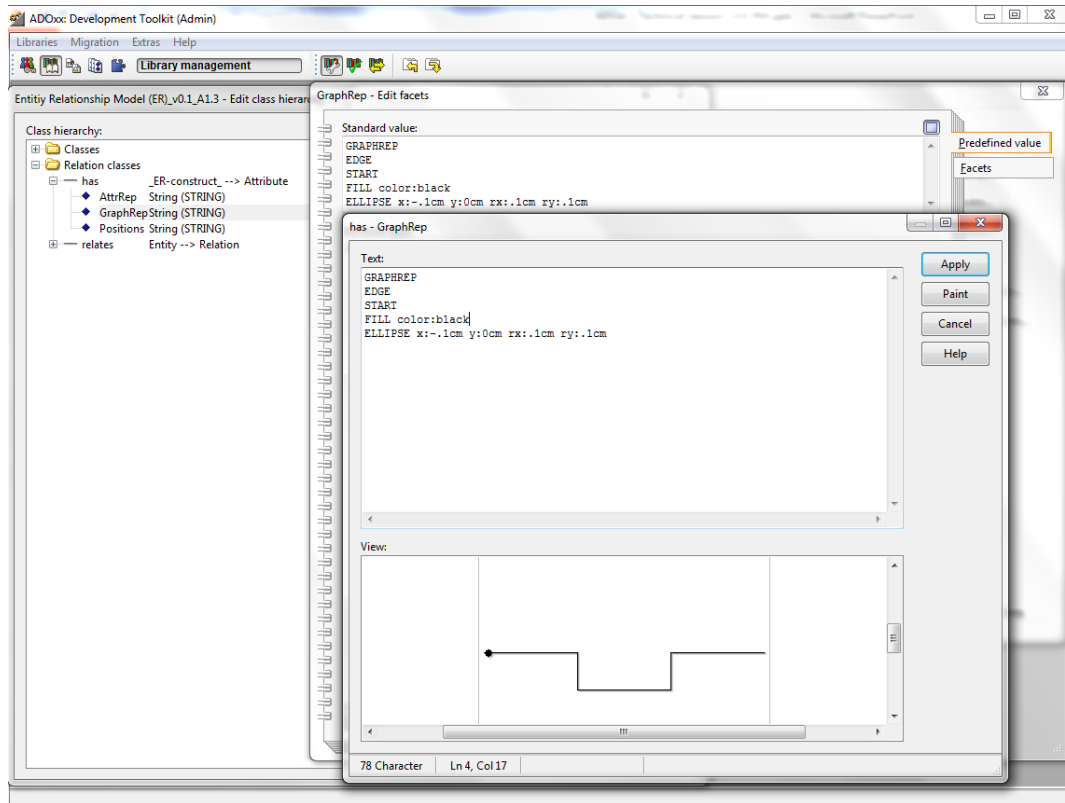
# Definition of RELATIONCLASS



## Define RELATIONCLASS:

1. Click new RelationClass
2. Define:
  - Name
  - From-class
  - To-class

# Implement RELATION GRAPHREP



## Define RELATION GRAPHREP:

1. Edit Attribute GRAPHREP
2. Open GRAPHREP Editor
3. Write GRAPHREP Code in Text field
4. View current GRAPHREP with "Paint"
5. Store the GRAPHREP with "Apply"

```
GRAPHREP
EDGE
START
FILL color:black
ELLIPSE x:-.1cm y:0cm rx:.1cm ry:.1cm
```



Any Questions?

