ADO*xx* 1.5 Cookbook for Implementing a Modelling Language

Abstract: The document contains an exercise whose purpose is to help a user new to ADO*xx* to get familiar with the ADO*xx* Development Toolkit. As an example, a very simple modeling method is defined and then implemented by implementing the modeling language.

Introduction

This cookbook introduces key aspects when developing a modelling language using the meta model approach.

In particular the principles of:

- Creating classes and inheriting from classes,
- The use of abstract class,
- Demonstration of analysis features on classes, inherited classes and abstract classes,
- Demonstration of built in functionality using Aggregation as a sample,
- Demonstration of Special Class Attributes.

Figure 1 introduces the ADOxx Dynamic Metamodel and the classes A, B, C, D, E, V, W and _E_ to be created as well as the relation classes, which are included in the ADOxx Dynamic Metamodel (gray) and those, which will be created in this cookbook (yellow).

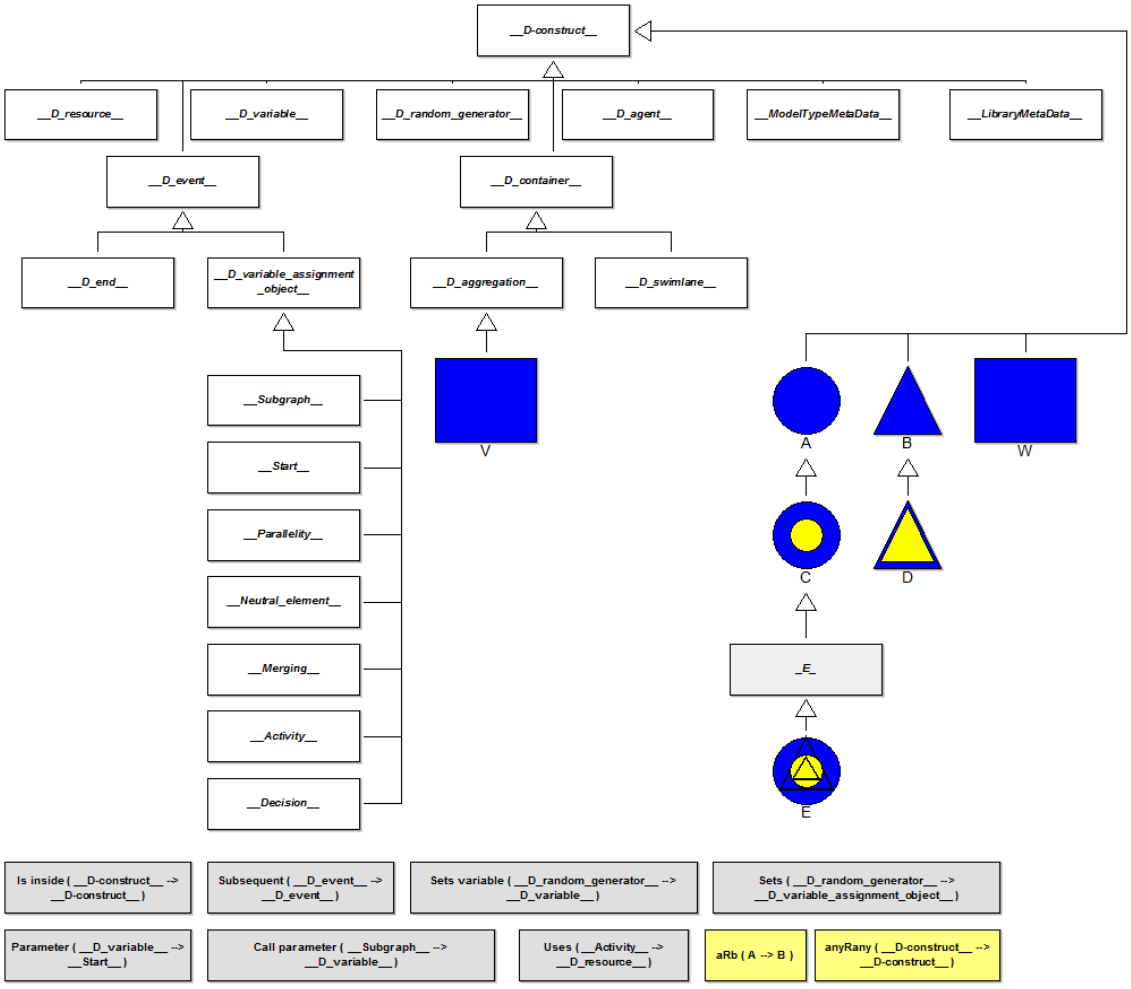


Figure 1: Modelling Language Implementation Cookbook Dynamic Metamodel

This sample basically introduces the creation and the inheriting of classes.
Furthermore the introduction of abstract classes and the behavior of the analysis e.g when querying all instances of class A can be tested.

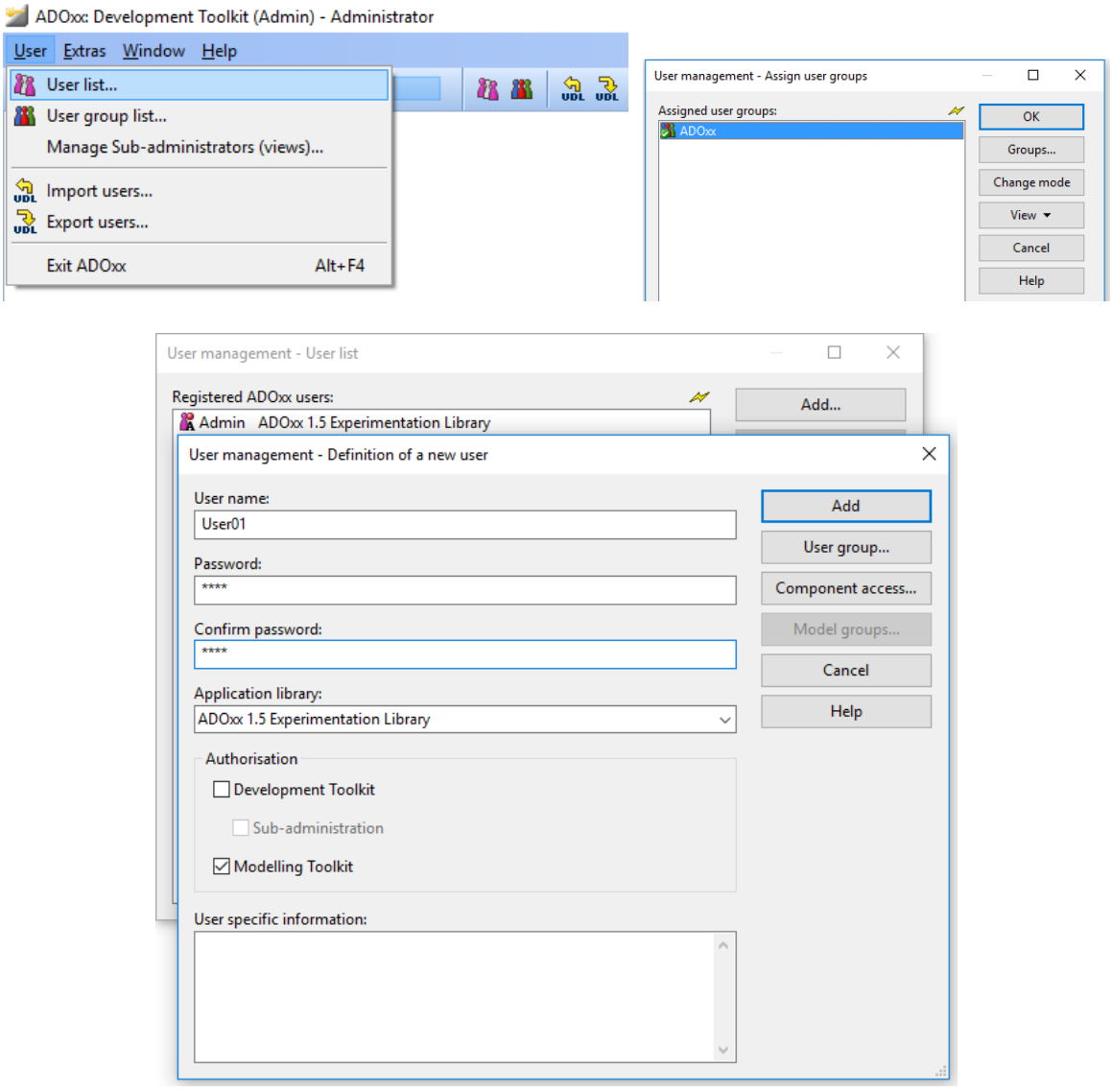
To demonstrate that built in functionality is inherited as well, we propose class V and W, whereas class V is inherited from a special abstract class "__D_aggregation__" and hence inherits all functionalities like the "is-inside" relation from that abstract class. Class W is identically created, but is not inherited from the special class "__D_aggregation__" but is inherited directly from the super class "__D-construct__", hence it does not inherit this special "is-inside" feature.

The following pages provide a hands-on description to implement the tutorial library and to test aforementioned behaviors.

Phase 1. PREPARATION

In this phase you will install and set up ADO*xx* 1.5. If you have already installed ADO*xx* 1.5, you can skip number 1. and 2.

1. Download the latest version of ADOxx from the **adoxx.org** portal
   Link: http://www.adoxx.org/live/download
2. Install ADOxx to your computer at a location of your choice. If needed, consult the Installation Guide.
3. Download and unpack the tutorial package from the **adoxx.org** portal.
   Link: http://www.adoxx.org/live/tutorial Method Engineering Tutorial Package
4. Unpack the tutorial package to allocation of your choice. This location will be referred to as "**<mypath>\ Method-Engineering-Package**".
5. Log into the ADOxx Development Toolkit with the username "Admin" and the password "password".
6. Create an user for modeling and testing purposes in the "User Administration" area



Phase 2. IMPLEMENT MODELLING LANGUAGE

2.1. Introduction to the modeling language

After having installed and set up ADO*xx* 1.5, you can start developing a basic modeling method. The classes and relation classes for the modeling language implemented in this exercise are described below.

(Relation-) Class	Shape	Colour	Size
A	Circular	Color: blue	Radius: 1.0 cm
	Attribute Name	**Type**	**Chapter**
	Name	STRING	Description
	a1	INTEGER	Description
	a2	RECORD	Description
	a3	STRING	Description
	a4	INTERREF	References
B	Triangular	Color: blue	Height: 1.6cm, Width: 1.6cm
	Attribute Name	**Type**	**Chapter**
	Name	STRING	Description
	b1	INTEGER	Description
	b2	RECORD	Description
	b3	STRING	Description
C	Circular Circular	Colour1: blue Colour2: yellow	Radius: 1.0 cm Radius: 0.5 cm
	Attribute Name	**Type**	**Chapter**
	Name	STRING	Description
	a1	INTEGER	Description
	a2	RECORD	Description
	a3	STRING	Description
	a5	INTERREF	References
D	Triangular Triangular	Color: blue Color: yellow	Height: 1.6cm, Width: 1.6cm Height: 0.7cm, Width: 0.8cm
	Attribute Name	**Type**	**Chapter**
	Name	STRING	Description
	b1	INTEGER	Description
	b2	RECORD	Description
	b3	STRING	Description
_E_	abstract class		
	Attribute Name	**Type**	**Chapter**
	Name	STRING	Description
	a1	INTEGER	Description
	a2	RECORD	Description
	a3	STRING	Description

(Relation-) Class	Shape	Colour	Size
	a4	INTEGER	Description
	a5	INTERREF	References
	b1	INTEGER	Description
	b2	RECORD	Description
E	Circular Triangular Circular Triangular	Color: blue Color: blue Color: yellow Color: yellow	Radius: 1.0cm Height: 1.6cm, Width: 1.6cm Radius: 0.5cm Height: 0.7cm, Width: 0.8cm
	Attribute Name	**Type**	**Chapter**
	Name	STRING	Description
	a1	INTEGER	A
	a2	RECORD	A
	a3	STRING	A
	a4	INTERREF	A, References
	b1	INTEGER	B
	b2	RECORD	B
	b3	STRING	B
	e1	INTEGER	Description
	e2	RECORD	Description
	e3	STRING	Description
	e4	ATTRIBUTEPROFILEREFERENCE	References
V derived from class __Aggregation__	Rectangular	Color: blue, red or yellow, depending on the value of the attribute "Type-Selection"	Height: 6.0cm, Width: 6.0cm
	Attribute Name	**Type**	**Chapter**
	Name	STRING	Description
	Type-Selection	ENUMERATION	Description
	Values: **type-1**, type-2, type-3		
	V-Text	LONSTRING	Description

W derived from class __D_construct__	Rectangular	Color: blue, red or yellow, depending on the value of the attribute "Type-Selection"	Height: 6.0cm, Width: 6.0cm
	Attribute Name	**Type**	**Chapter**
	Name	STRING	Description
	Type-Selection	ENUMERATION	Description
	Values: **type-1**, type-2, type-3		
	V-Text	LONSTRING	Description
aRb	arrow	black	1 pt
	connects:	A -> B	
	Attribute Name	**Type**	**Chapter**
	Name	STRING	Description

(Relation-) Class	Shape	Colour	Size
	Comments	LONGSTRING	Description
anyRany	line	darkblue	1 pt
	connects:	any -> any	
	Attribute Name	**Type**	**Chapter**
	Name	STRING	Description
	Comments	LONGSTRING	Description
RC1	**record class**	**no representation**	
	Attribute Name	**Type**	**Chapter**
	RC1_a1	STRING	
	RC1_a2	INTEGER	
	RC1_a3	TIME	
AP1	**attribute profile class**	**no representation**	
	Attribute Name	**Type**	**Chapter**
	AP1_a1	STRING	
	AP1_a2	INTEGER	
	AP1_a3	TIME	

Slides for this phase are located in the file "***2_Modelling Language Implementation_PUBLIC.pdf***", located in the folder *"<mypath>\Method-Engineering-Package\Tutorial_Slides\"*

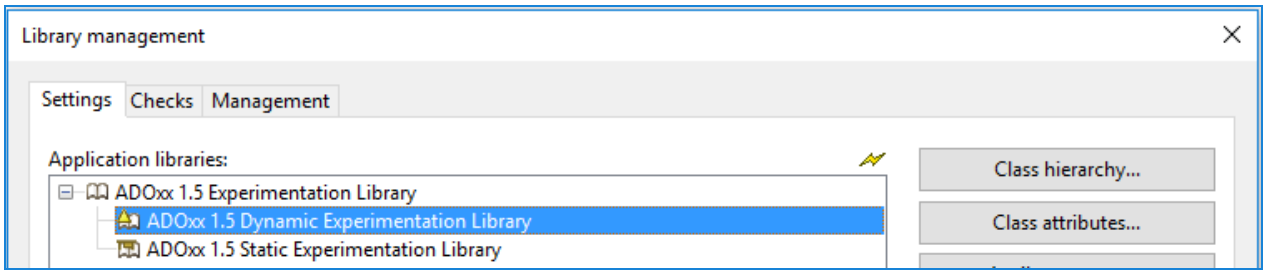
2.2. Create the classes

Note: If you are familiar with creating classes, then you can download the library, which already includes the classes and continue with section **2.3.**

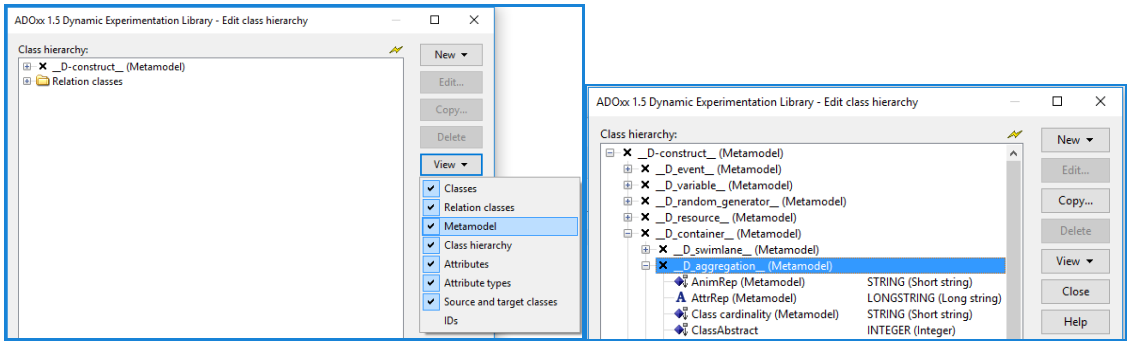
You can find the library in the Method Engineering Tutorial Package under "*<mypath>\Method-Engineering-Package\Tutorial_Libraries\Modelling Language Implementation Libraries*"

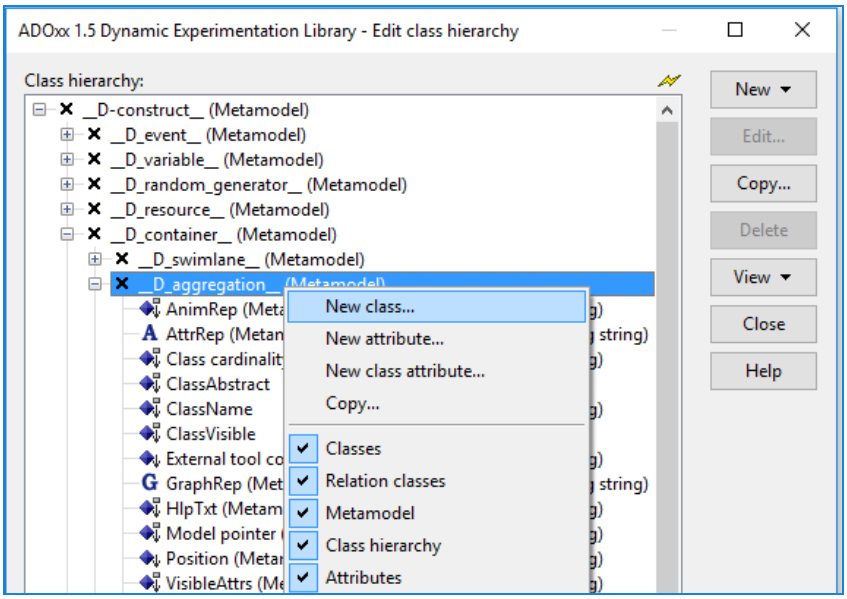
For help with steps 1 to 12, use the slides 15 – 34.

1. Open the Class hierarchy window: click the icon "Library Management" -> "Libraries" -> "Settings…" -> expand "ADOxx 1.5 Experimentation Library" -> select ADOxx 1.5 Dynamic Experimentation Library -> click the "Class hierarchy" button



2. Create a concrete class "A", derived from class "__D_construct__".
3. Create a concrete class "C", derived from class "A".
4. Create an abstract class "_E_", derived from class "C".
5. Create a concrete class "E", derived from class "_E_".
6. Create a concrete class "B", derived from class "__D_construct__".
7. Create a concrete class "D", derived from class "B".
8. Create a concrete class "V", derived from class "__D_aggregation__". The class "__D_aggregation__" is derived from class "__D_container__", who is derived from class "__D_construct__".





9. Create a concrete class “W”, derived from class “__D_construct__”

2.3. Create the relation classes

Note: If you are familiar with creating relation classes, then you can download the library, which already includes the relation classes as well as the implementations of previous sections and continue with section **2.4.**

You can find the library in the Method Engineering Tutorial Package under “*<mypath>\Method-Engineering-Package\Tutorial_Libraries\Modelling Language Implementation Libraries*”

10. Create a relation class “anyRany” from class “__D_construct__” to class “__D_construct__”
11. Create a relation class “aRb” from class “A” to class “B”

2.4. Define the model types

Note: If you are familiar with the definition of model types, then you can download the library, which already includes the model types as well as the implementations of previous sections and continue with section **2.5.**

You can find the library in the Method Engineering Tutorial Package under “*<mypath>\Method-Engineering-Package\Tutorial_Libraries\Modelling Language Implementation Libraries*”

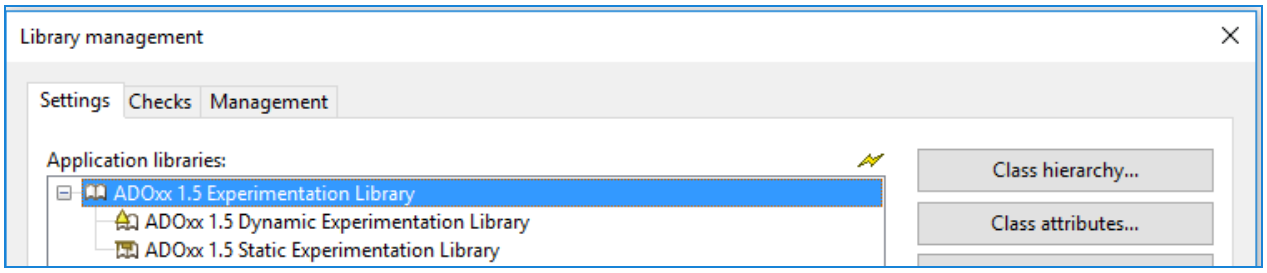
12. Define a model type “Sample” containing the classes “A”, “B”, “C”, “D” and “E” and the relation classes “anyRany” and “aRb”
13. Define a model type “Practice” containing the classes “A”, “B”, “V” and “W” and the relation classes “anyRany”

2.5. Define the record class and attribute profile class

Note: If you are familiar with the definition of record and attribute profile classes, then you can download the library, which already includes the record and attribute profile classes as well as the implementations of previous sections and continue with section **2.6.**

You can find the library in the Method Engineering Tutorial Package under “*<mypath>\Method-Engineering-Package\Tutorial_Libraries\Modelling Language Implementation Libraries*”

Go to the 'main' (superior) library



14. Define a record class ”RC1”
    - Select Class Hierarchy
    - Create Record Class “RC1”
15. Define an attribute profile class ”AP1”
    - Select Class Hierarchy
    - Create Attribute Profile Class “AP1”

2.6. Create the attributes for the classes

Note: If you are familiar with the creation of attributes, then you can download the library, which already includes the attributes as well as the implementations of previous sections and continue with section **2.7.**

You can find the library in the Method Engineering Tutorial Package under “*<mypath>\Method-Engineering-Package\Tutorial_Libraries\Modelling Language Implementation Libraries*”

16. For class “A”, create the following attributes:
    a. attribute “a1” of type INTEGER,
    b. attribute “a2” of type RECORD, referring to record class “RC1”
    c. attribute “a3” of type STRING,
    d. attribute “a4” of type INTERREF, referring to a class “V” in a model type “Practice”
    (Right click on a4: Edit->Facets->AttributeInterRefDomain:
    REFDOMAIN
    OBJREF
    mt:"Practice"
    c:"V"
    max:1
)
17. For class “B”, create the following attributes:
    a. attribute “b1” of type INTEGER,
    b. attribute “b2” of type RECORD, referring to record class “RC1”
    c. attribute “b3” of type STRING,
18. For class “C”, create attribute “__Conversion__“ of type LONGSTRING and add the following text to the Standard value of this attribute:

```
CLASS "E"
ATTR "Name"
ATTR "a1"
ATTR "a2"
ATTR "a3"
ATTR "a4"
ATTR "e1" from:"a1"
```

ATTR "e2" from:"a2"

ATTR "e3" from:"a3"

19. For class "_E_", create the following attributes:
    a. attribute "b1" of type INTEGER,
    b. attribute "b2" of type RECORD, referring to record class "RC1"
20. For class "E", create the following attributes:
    a. attribute "e1" of type INTEGER,
    b. attribute "e2" of type RECORD, referring to record class "RC1"
    c. attribute "e3" of type STRING,
    d. attribute "e4" of type PROFREF, referring to attribute profile class "AP1"
21. For class "V" create the following attributes:
    a. attribute "Type-Selection" of type ENUMERATION, with the values "type-1", "type-2", "type-3"
    b. attribute "V-Text" of type STRING
22. For class "W" create the following attributes:
    a. attribute "Type-Selection" of type ENUMERATION, with the values "type-1", "type-2", "type-3"
    b. attribute "W-Text" of type STRING
23. For record class RC1 create the following attributes:
    a. attribute "RC1_a1" of type STRING
    b. attribute "RC1_a2" of type INTEGER
    c. attribute "RC1_a3" of type TIME
24. For Attribute profile AP1 create the following attributes:
    a. attribute "AP1_a1" of type STRING
    b. attribute "AP1_a2" of type INTEGER
    c. attribute "AP1_a3" of type TIME
25. For relation class aRb create the following attribute:
    a. Attribute "Denomination" of type STRING
26. For relation class anyRany create the following attribute:
    a. Attribute "Denomination" of type STRING

2.7. Add graphical representation and attribute representation for the classes and relation classes

Note: If you are familiar with the addition of graphical and attribute representations for the classes, then you can download the final Tutorial library and continue with **Phase 3.**

You can find the library in the Method Engineering Tutorial Package under "*<mypath>\Method-Engineering-Package\Tutorial_Libraries*"

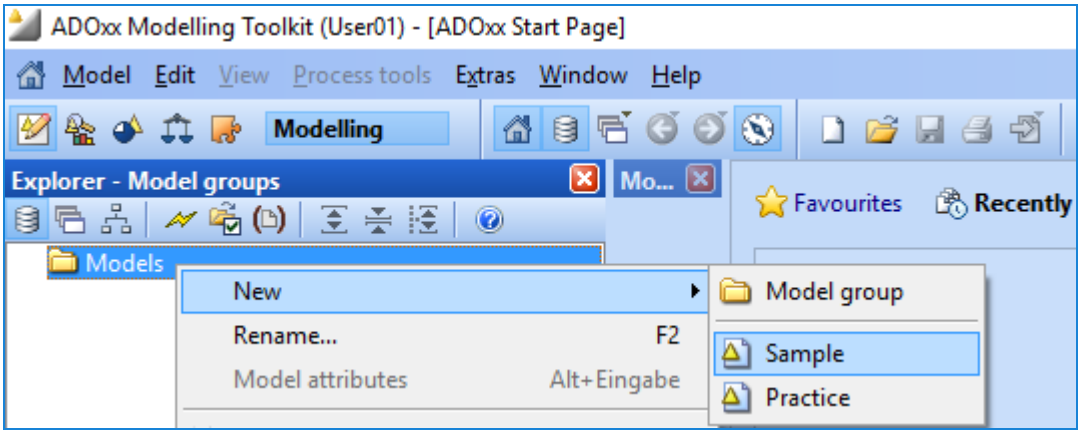
27. Write the code for the GraphRep of each concrete class and relation class you have defined (see ADOxx GraphRep Repository); after changing the contents of the GraphRep class attribute, you can have a preview of the graphical representation by clicking the "Paint" button.

    Note: By clicking on the class name in the table in section **2.1.** you will be directed to the graphical representation used in this cookbook for this particular class.

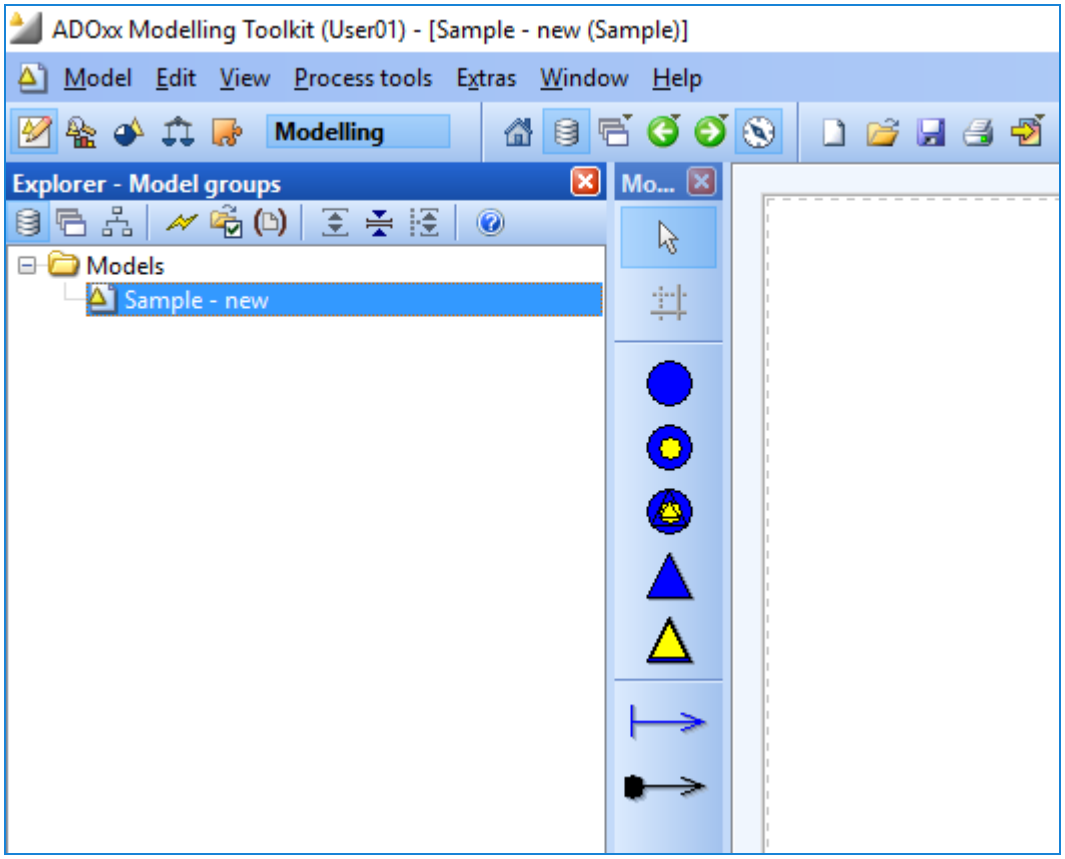
28. For each concrete class and relation class in your modeling language edit the AttrRep class attribute to define how the information about each object will be displayed (see http://www.adoxx.org/live/adoxx-attribute-notation-language-attrrep).

Phase 3. TESTING THE MODELLING LANGUAGE

1. Open the ADOxx Modelling Toolkit
2. Right click on the model group “Models” and select the model type “Sample”



3. All the classes and relation classes you mentioned in the definition of the model type (Phase 2. step 13) are displayed in the Modelling bar
4. Enter a name for the newly created model



5. To add an object to the model, click on the class in the modeling bar and then click on the model canvas.