

asreco

CENTRAL EUROPE

LIDS7 MDG Extension for EA

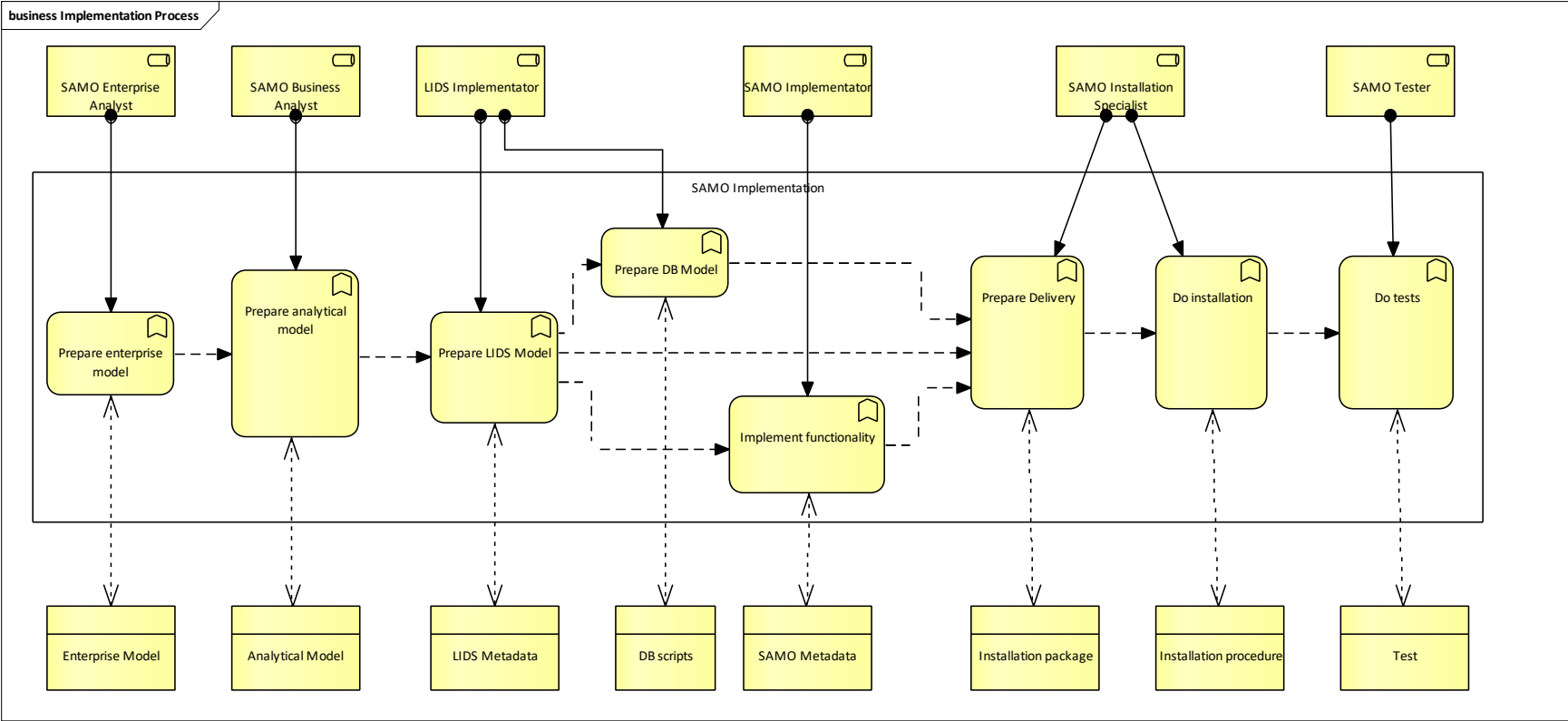
Workshop – SAMO Logical Data model

Radim Bystřický, Vojtěch Kučera – GIS&Utility

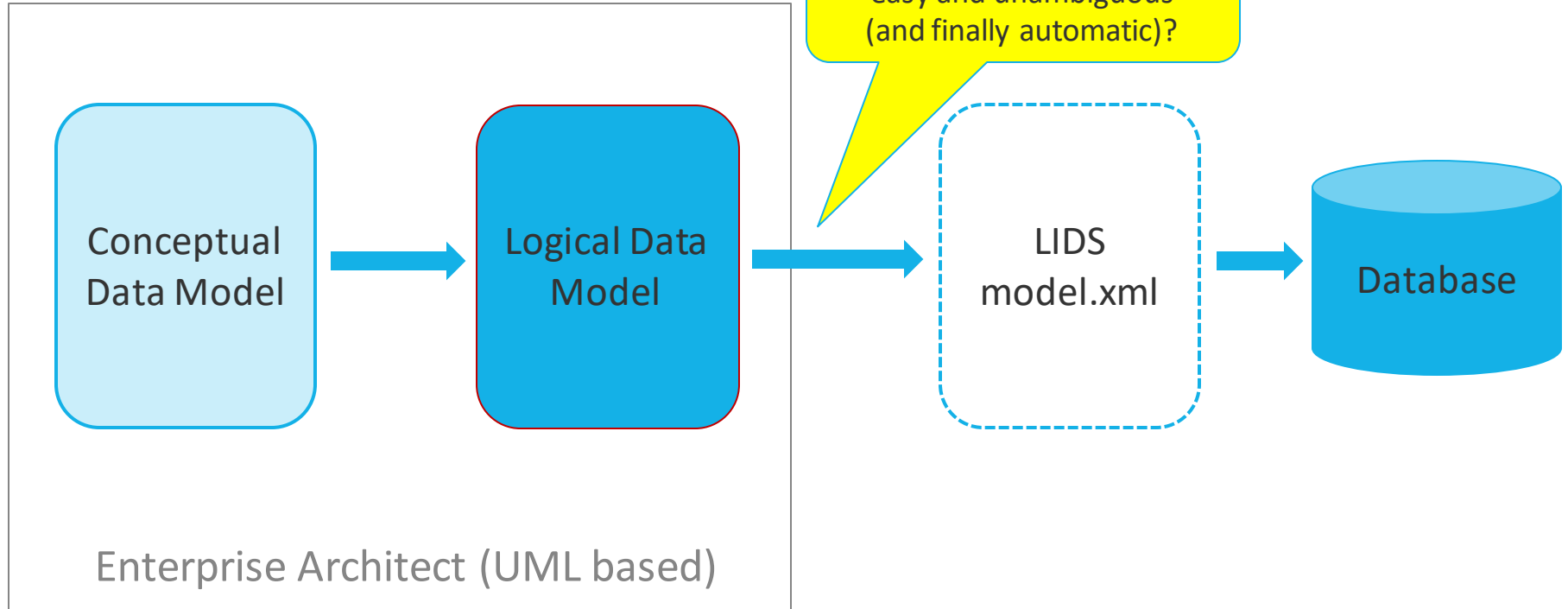
Content:

1. Introduction to LIDS7 MDG Technology
2. How to model Codelists
3. How to model Feature Types
4. Data Types
5. Naming Convention for LIDS model
6. Basic transformation rules
7. Advanced transformation rules
8. Patterns
9. „Production line“ for LIDS model

Implementation line



SAMO Data Model



Introduction to LIDS7 MDG Technology („LIDS Extension“)

SPARX MDG Technology

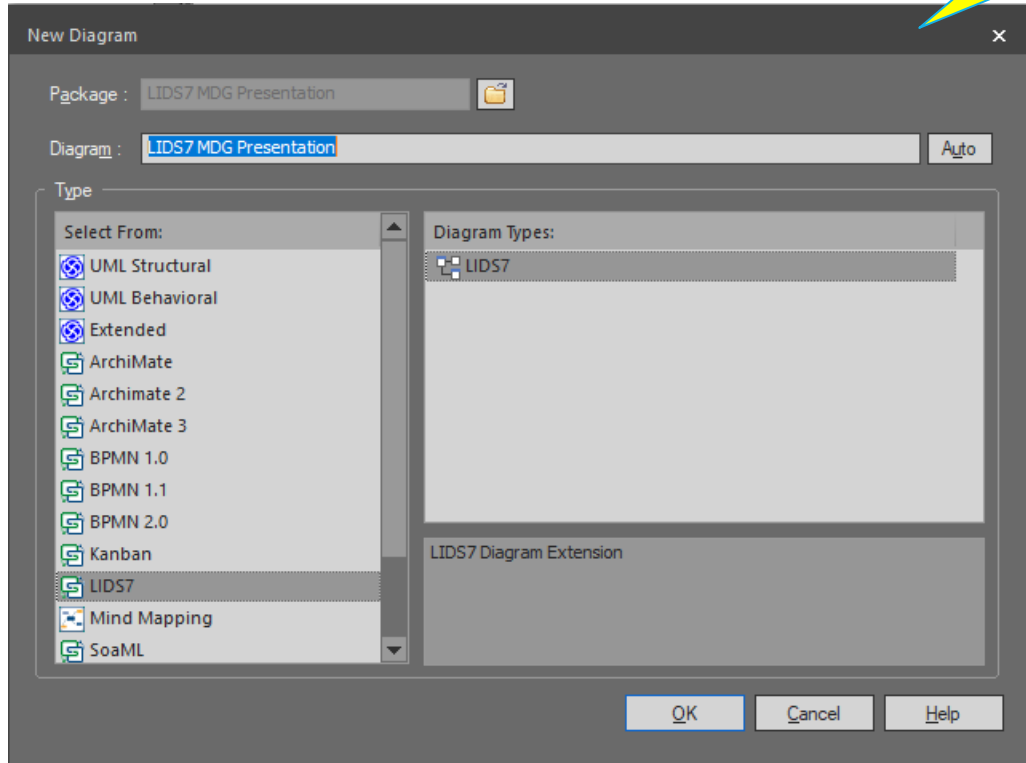
„**MDG Technology** is a vehicle for providing access to the resources of either a commercially-available technology or a technology that you have created yourself. Such resources include a wide range of facilities and tools, such as **UML Profiles**, code modules, scripts, **Patterns**, images, **Tagged Value Types**, report templates, linked document templates and **Toolbox pages**.

Using Enterprise Architect, you can develop models based on the standard UML specifications, and you can **extend the core UML** structures using UML-supported mechanisms such as **Tagged Values**, **Stereotypes**, **Profiles** and **Design Patterns**. These facilities are within the Enterprise Architect core technologies, and you can activate and use further Model Driven Generation (MDG) Technologies that are either integrated with the system or available from external locations.“

LIDS7 MDG Technology in EA

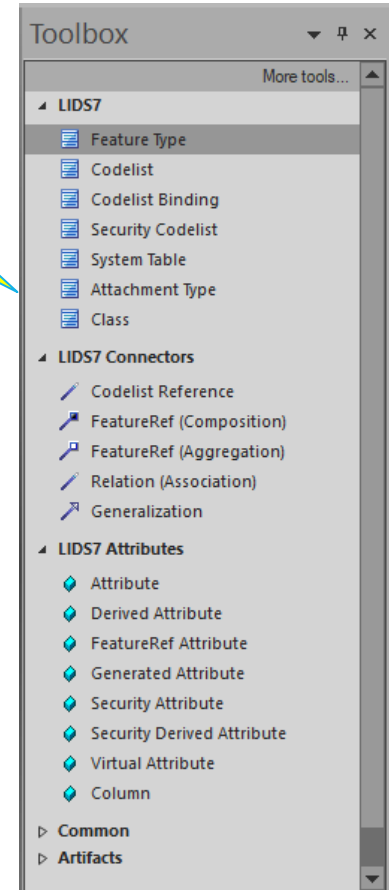
The screenshot displays the SAMOarchiv - Enterprise Architect interface. The main workspace shows a UML diagram with three classes: «featureType» boProject (light blue), «featureType» boVariant (light blue), and «codeList» boCategory (light green). The boProject class has attributes: + code: Code_T, + name: Name_T, + startDate: DateOnly_T, + endDate: DateOnly_T, + is_OrdeDone: Boolean_T = false, and «derivedAttribute» + fullName: Name_T. It has a «container =» compartment. The boVariant class has attributes: + cl_boCategory: CodeList_T, + code: Code_T, + name: Name_T, and «featureRefAttributes» + fr_boProject: FeatureType_T. It also has a «container =» compartment. The boCategory class has attributes: - id: ColumnId_T, - code: Code_T, and - description: Description_T. It has a «column» compartment and «dbName =», «hierarchical = false», and «size = small» compartments. Relationships include a composition between boProject and boVariant, a «featureRefAssoc» association between boProject and boVariant, and a «codeListRef» association between boVariant and boCategory. The Project Browser on the left shows the project structure, and the Toolbox in the center lists various LIDS7 elements like Feature Type, CodeList, and Connectors. A yellow callout bubble points to the Toolbox with the text "Special Toolbox". Another yellow callout bubble points to the diagram with the text "Special diagram".

LIDS7 Diagram + Toolbox



Add new diagram

Related
Toolbox

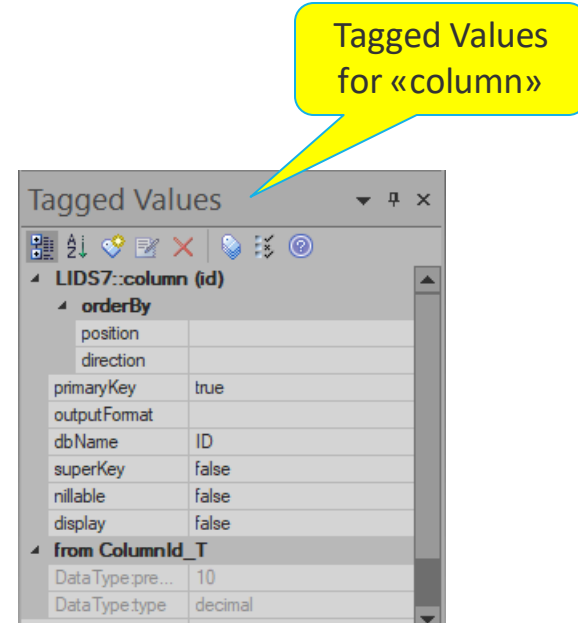
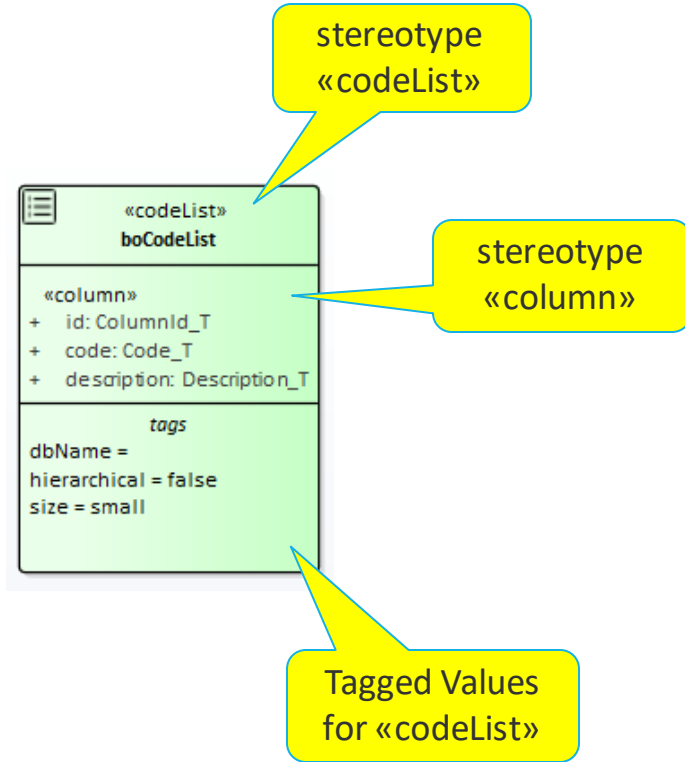


LIDS7 MDG Technology

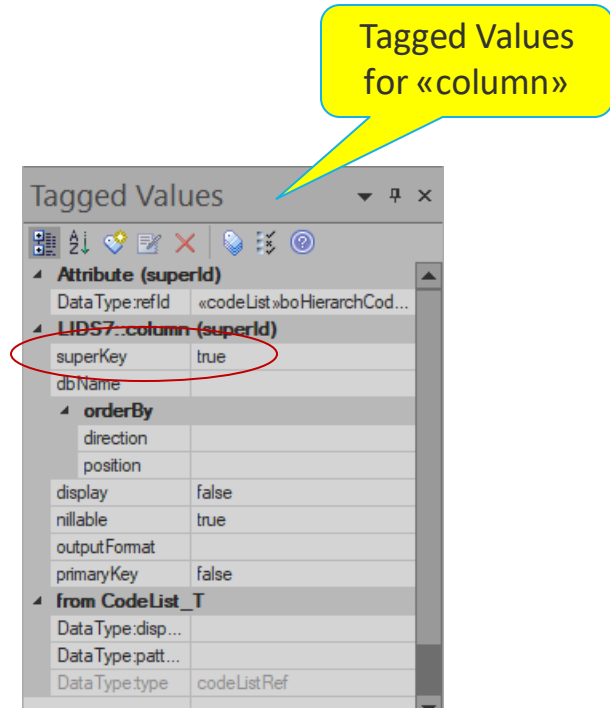
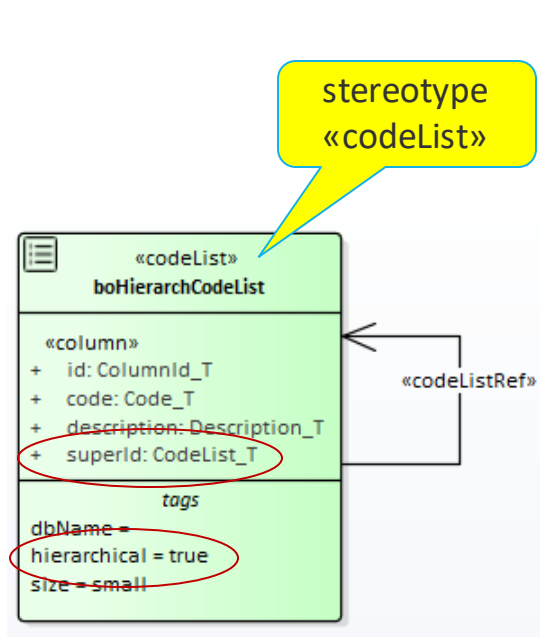
- New specific <<**stereotypes**>> with specific set of **Tagged values**
 - Class: featureType, codeList, ...
 - Attribute: column, derivedAttribute, featureRefAttribute, ...
 - Association: featureRefAssoc, codeListRef, ...
- New **diagram** with specific **Toolbox**
- Set of specific **Data Types**
- **Patterns**
- + **Name Conventions** (doc)
- + Specific transformation rules

How to model Codelists?

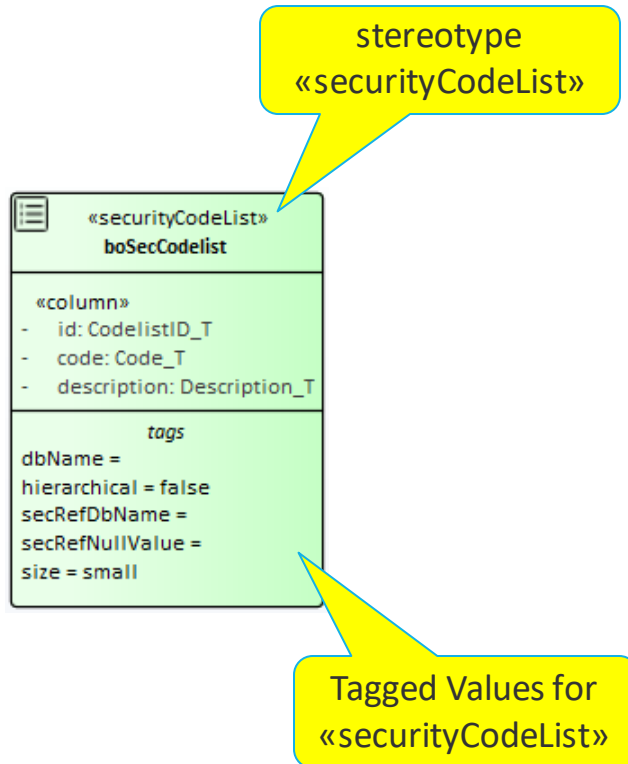
Codelist



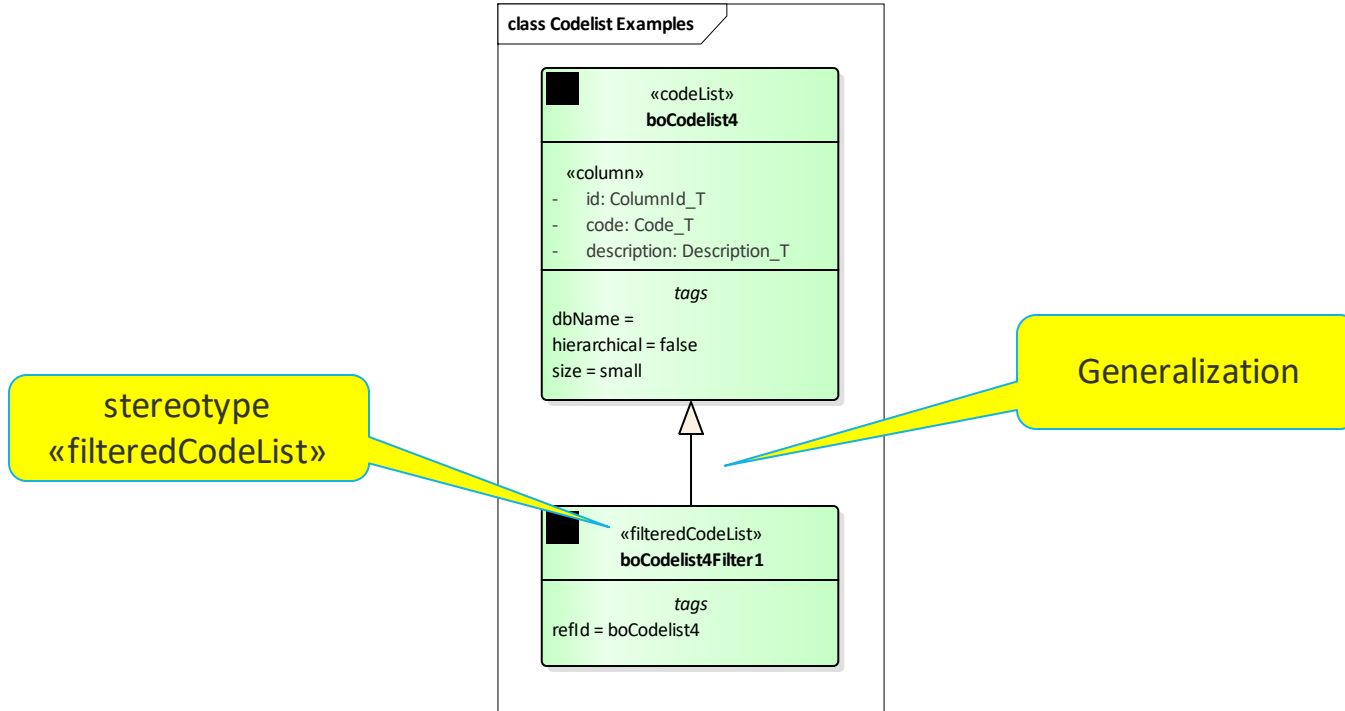
Hierarchical Codelist



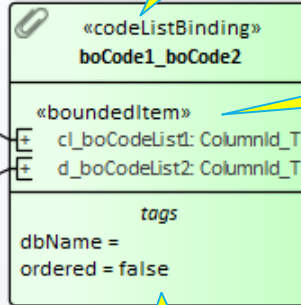
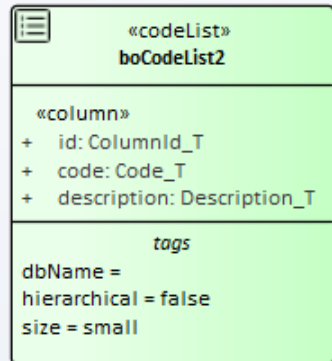
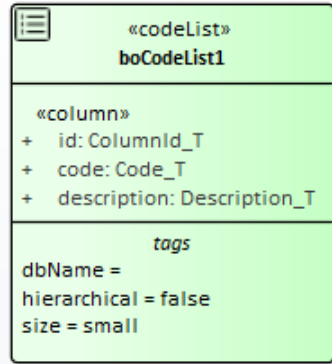
Security Codelist



Filtered Codelist



Codelist Binding



Tagged Values for
«codeListBinding»

stereotype
«codeListBinding»

stereotype
«boundedItem»

Tagged Values for
«boundedItem»

- Usage of binding in Feature Type must be done manually („boundedAttributesArray“)

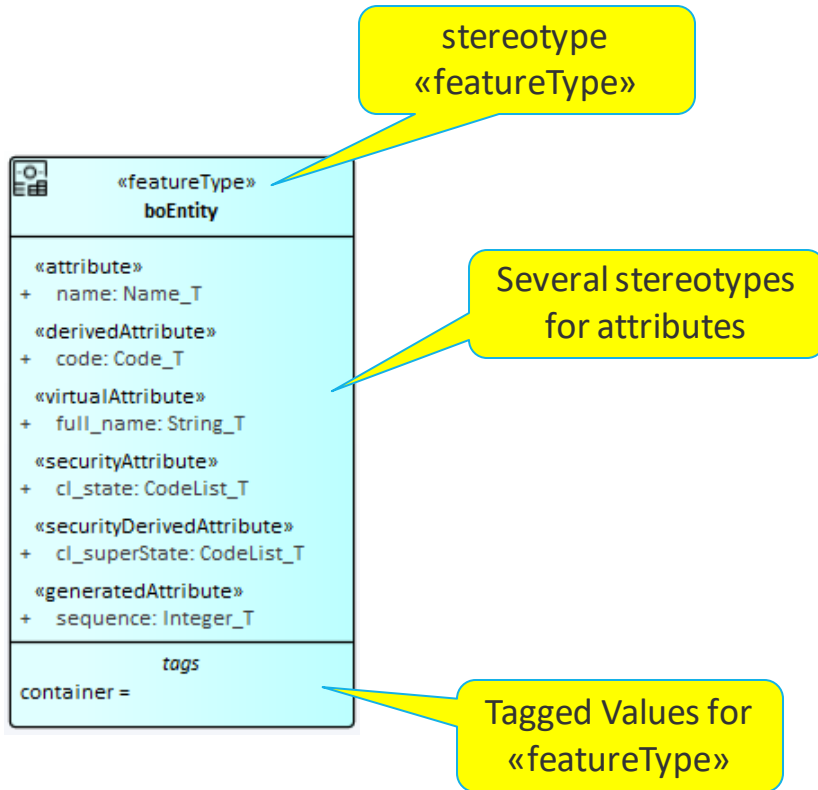
Tagged Values

LIDS7::boundedItem (cl_boCodeList1)	
dbName	
codeListRef	«codeList»boCodeList1
from ColumnId_T	
Data Type.pre...	10
Data Type.type	decimal



How to model Feature Types?

Feature Type

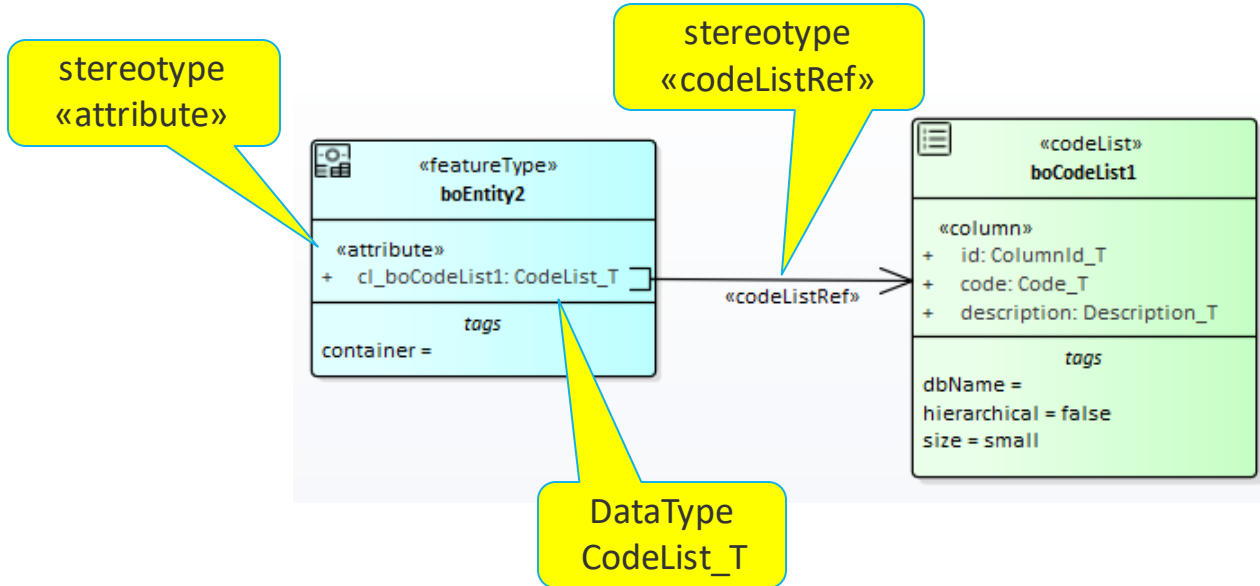


The screenshot shows a dialog box titled "Tagged Values" with a toolbar. It displays the tagged values for the attribute LIDS7::attribute (name). The values are organized into two sections: "LIDS7::attribute (name)" and "from Name_T".

LIDS7::attribute (name)	
dbName	
nullable	true

from Name_T	
Data Type.type	string
Data Type.minLen...	0
Data Type.maxLe...	100
Data Type.pattern	

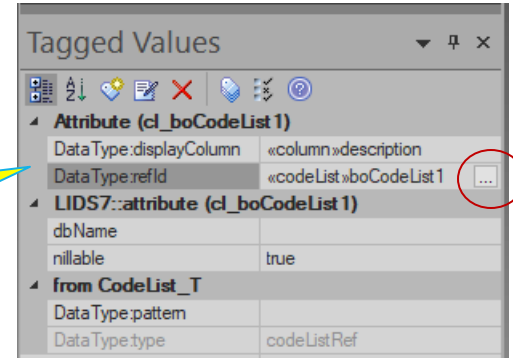
Feature Type and relation to Codelist



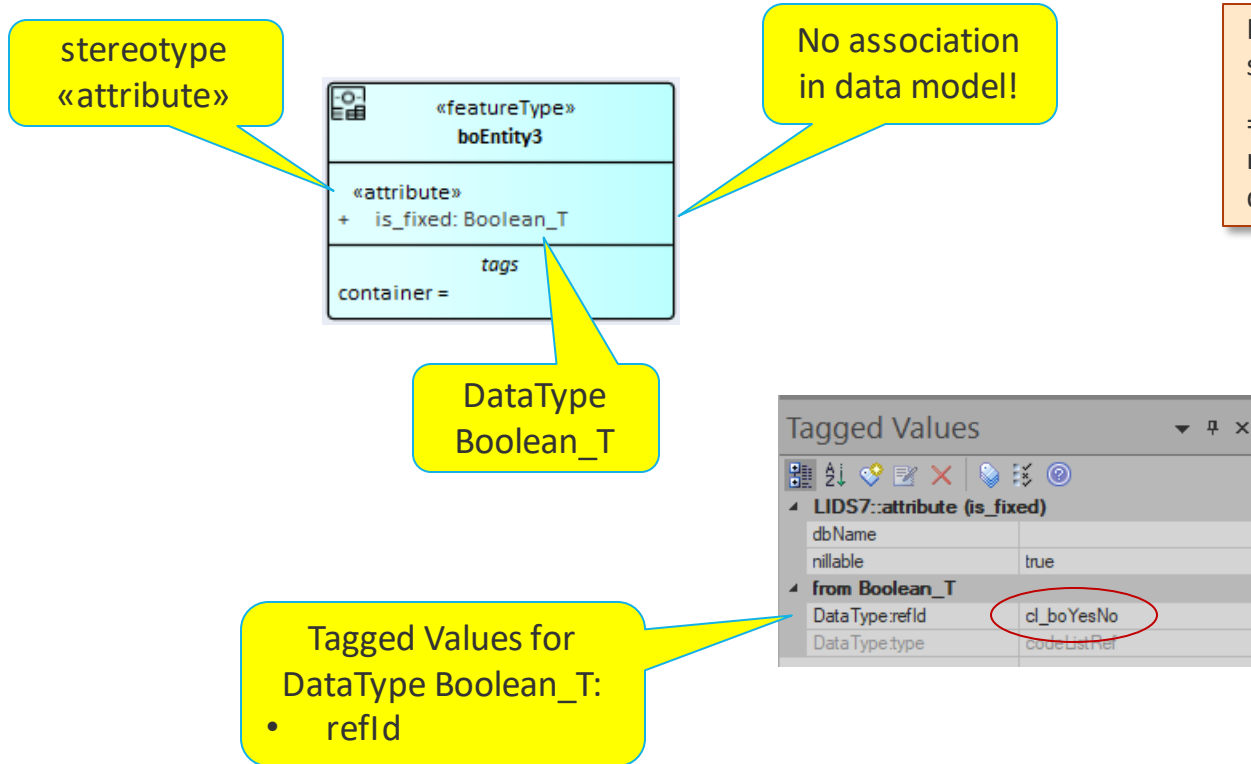
Reference to the codelist is stored in tag of attribute
=> association in model is created just for better understanding (!)

Tagged Values for DataType CodeList_T:

- refId
- displayColumn



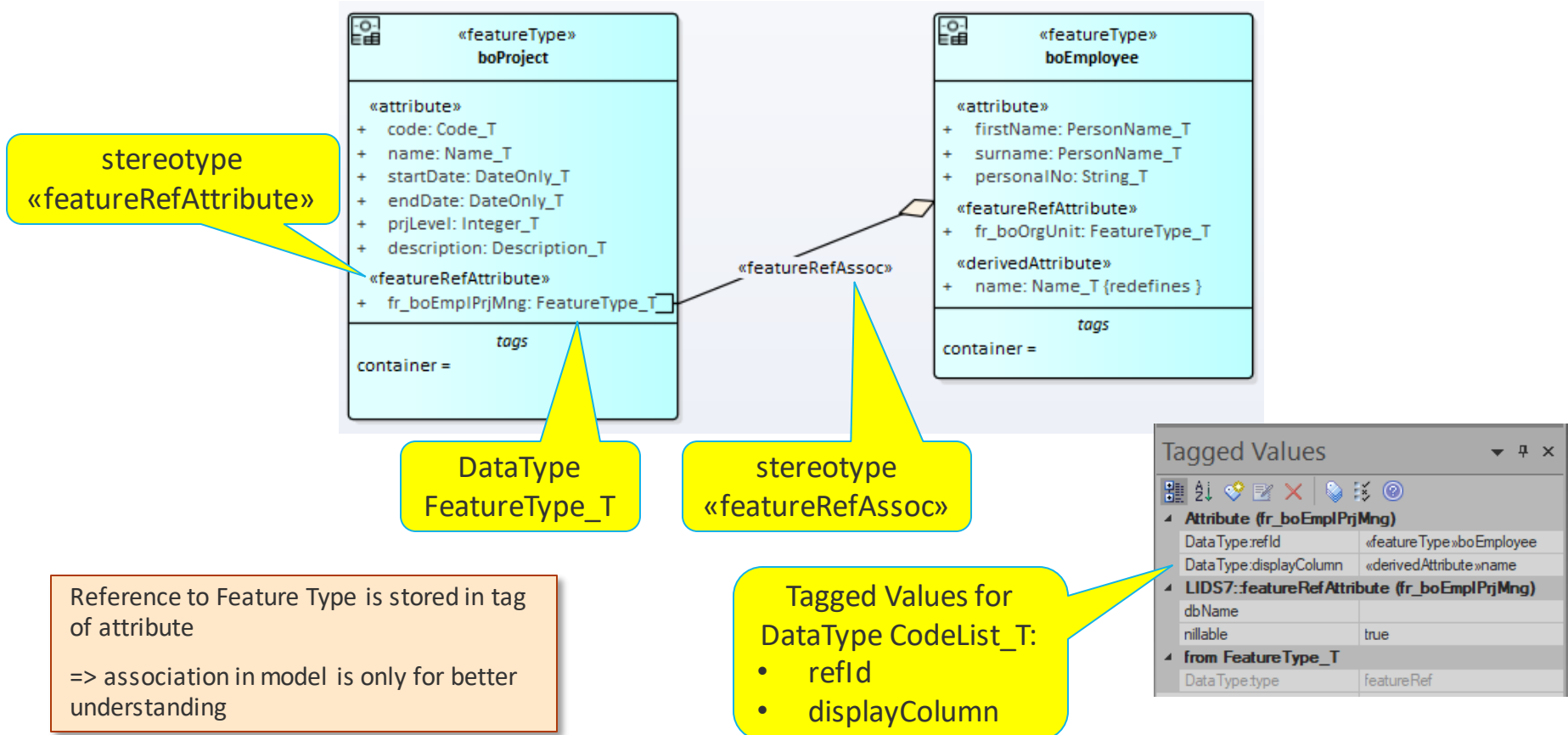
Feature Type and Boolean attribute



Reference to „boolean“ codelist is stored in tag of attribute

=> association in model is not necessary and it is not recommended due to simplicity of model

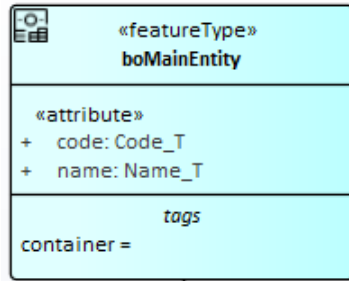
FeatureRef relation – variant Aggregation



Tagged Values

Attribute (fr_boEmplPrjMng)	
Data Type.refId	«feature Type»boEmployee
Data Type.displayColumn	«derivedAttribute»name
LIDS7:featureRefAttribute (fr_boEmplPrjMng)	
dbName	
nillable	true
from FeatureType_T	
Data Type.type	featureRef

FeatureRef relation – variant Composition



stereotype
«featureRefAssoc»

stereotype
«featureRefAttribute»

DataType
FeatureType_T

Tagged Values for
DataType CodeList_T:

- refId
- displayColumn

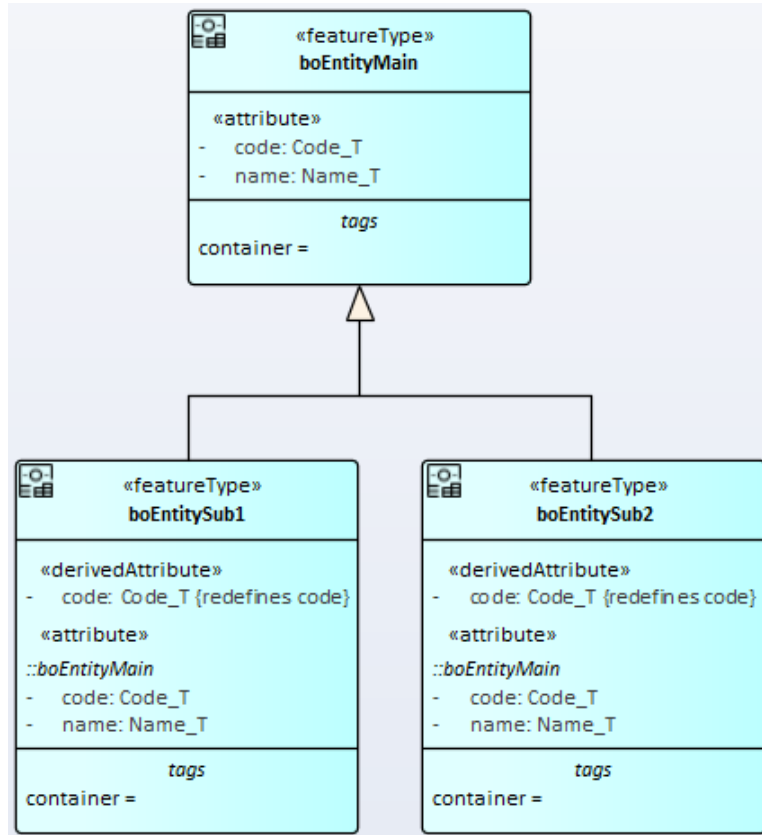
Reference to Feature Type is stored in tag of attribute

=> association in model is only for better understanding

Composite and Aggregation is used for better understanding. Now, there is no impact on generating model.xml.

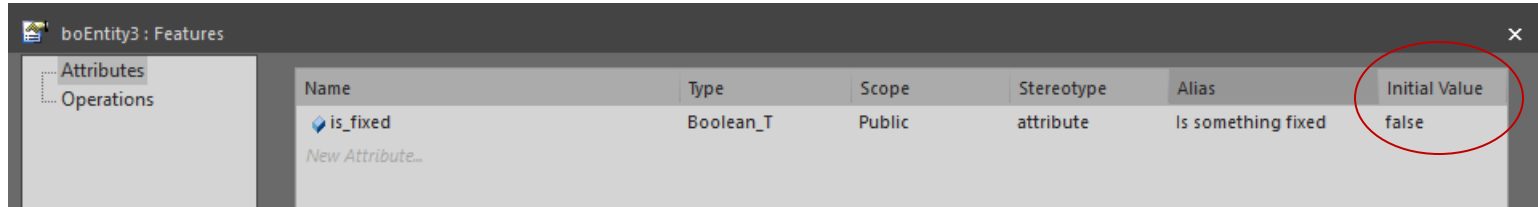
Tagged Values	
Attribute (fr_boMainEntity)	
Data Type:refId	«featureType»boMainEntity
Data Type:displayColumn	«attribute»code
LIDS7:featureRefAttribute (fr_boMainEntity)	
dbName	
nullable	true
from FeatureType_T	
Data Type:type	featureRef

Feature Type and Generalization

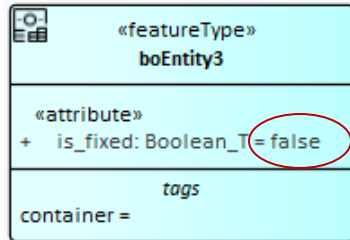


- Standard UML modelling
- „root“ ft_5000002 is not necessary to model in EA
- Generalization is a big challenge for modelling in EA

Default value of attribute

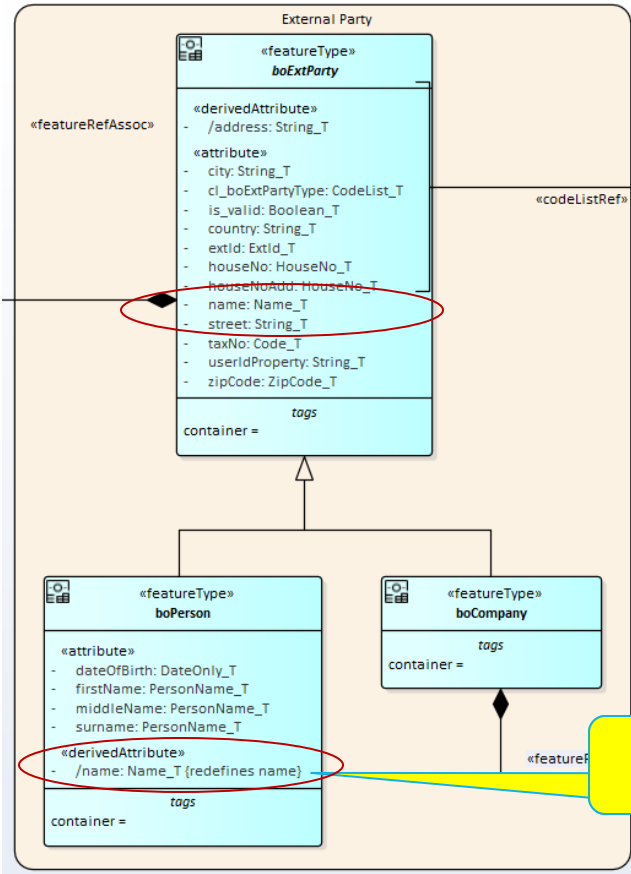


Name	Type	Scope	Stereotype	Alias	Initial Value
is_fixed	Boolean_T	Public	attribute	Is something fixed	false
<i>New Attribute...</i>					



For default value in model.xml
is used item Initial Value in EA

Redefinition of attributes



What you can redefine:

- Default values.
- Whether the attribute is mandatory (nillable flag in model).
- Forbidden operations (READ, QUERY, UPDATE).
- Data type (you can change a codelist data type to a different codelist, different codelist must be a filtered codelist based on previous codelist).
- Derived attribute query (you can define deriving query in children feature types, you can change deriving query in children feature types).
- You can change the type of the attribute from normal (parent) to derived in any child.
- You can make multiple redefinition of one attribute.

The screenshot shows the 'boPerson : Features' window. The 'Attributes' table lists the following attributes:

Name	Type	Scope	Stereotype	Alias	Initial Value
dateOfBirth	DateOnly_T	Private	attribute	Date of Birth	
firstName	PersonName_T	Private	attribute	First Name	
middleName	PersonName_T	Private	attribute	Middle Name	
surname	PersonName_T	Private	attribute	Surname	
name	Name_T	Private	derivedAttribute	Name	

The `name` attribute is circled in red. The 'Attribute (name)' details panel shows it is a derived attribute. The 'Redefines' tab shows the redefined property path: `«attribute»Detailed Model (ADM)::Common Components::Logical Data Model::External Party::boExtParty.name`.

A yellow callout box labeled 'Redefined attribute' points to the `name` attribute in the table.

Data Types

Data Types

- Mandatory
 - Several universal data types – string, decimal, ...
 - A lot of specific data types with the specific default values of tags

Example for
DataType Code_T

<p>«dataType» Code_T</p>
<p><i>tags</i> DataType:maxLength = 30 DataType:minLength = DataType:pattern = DataType:type = string</p>
<p><i>notes</i> String data type - typically it is used for formatted unique identification which is based on sequence (e.g. Invoice number) or specific unique identification for codelist item.</p>

25 dataTypes

dataType	Code_T	Name_T	Description_T
«dataType» String_T	«dataType» Code_T	«dataType» Name_T	«dataType» Description_T
<p><i>tags</i> DataType:maxLength = 30 DataType:minLength = 0 DataType:pattern = DataType:type = string</p>	<p><i>tags</i> DataType:maxLength = 30 DataType:minLength = 0 DataType:pattern = DataType:type = string</p>	<p><i>tags</i> DataType:maxLength = 30 DataType:minLength = 0 DataType:pattern = DataType:type = string</p>	<p><i>tags</i> DataType:maxLength = 30 DataType:minLength = 0 DataType:pattern = DataType:type = string</p>
<p><i>notes</i> String data type - typically it is used for formatted unique identification which is based on sequence (e.g. Invoice number) or specific unique identification for codelist item.</p>	<p><i>notes</i> String data type - typically it is used for formatted unique identification which is based on sequence (e.g. Invoice number) or specific unique identification for codelist item.</p>	<p><i>notes</i> String data type - typically it is used for the most common names which are assigned to business entities.</p>	<p><i>notes</i> String data type - typically it is used for the most descriptive names, especially for product name.</p>
«dataType» Phone_T	«dataType» Code_T	«dataType» LangText_T	
<p><i>tags</i> DataType:maxLength = 20 DataType:minLength = 0 DataType:pattern = DataType:type = string</p>	<p><i>tags</i> DataType:maxLength = 2000 DataType:minLength = 0 DataType:pattern = DataType:type = string</p>	<p><i>tags</i> DataType:maxLength = 4000 DataType:type = string</p>	
<p><i>notes</i> String data type - phone number.</p>	<p><i>notes</i> String data type - ISO country code.</p>	<p><i>notes</i> String data type - typically it is used for additional description of items.</p>	
«dataType» HouseNo_T	«dataType» ZIPCode_T	«dataType» Area_T	
<p><i>tags</i> DataType:maxLength = 30 DataType:minLength = 0 DataType:pattern = DataType:type = string</p>	<p><i>tags</i> DataType:maxLength = 10 DataType:minLength = 0 DataType:pattern = DataType:type = string</p>	<p><i>tags</i> DataType:maxLength = 200 DataType:minLength = 0 DataType:pattern = DataType:type = string</p>	
<p><i>notes</i> String data type - house number.</p>	<p><i>notes</i> String data type - ZIP code.</p>	<p><i>notes</i> String data type for geographical location (country/region/city).</p>	
«dataType» Product_T	«dataType» Price_T	«dataType» Quantity_T	
<p><i>tags</i> DataType:maxLength = 30 DataType:minLength = 0 DataType:pattern = DataType:type = string</p>	<p><i>tags</i> DataType:maxLength = 10 DataType:minLength = 0 DataType:pattern = DataType:type = decimal</p>	<p><i>tags</i> DataType:maxLength = 10 DataType:minLength = 0 DataType:pattern = DataType:type = decimal</p>	
<p><i>notes</i> Decimal data type.</p>	<p><i>notes</i> Decimal data type - for price.</p>	<p><i>notes</i> Decimal data type for quantity (amount of commodity).</p>	
«dataType» YearNo_T	«dataType» Duration_T	«dataType» Calendar_T	
<p><i>tags</i> DataType:maxLength = 10 DataType:minLength = 0 DataType:pattern = DataType:type = string</p>	<p><i>tags</i> DataType:maxLength = 10 DataType:minLength = 0 DataType:pattern = DataType:type = decimal</p>	<p><i>tags</i> DataType:maxLength = 10 DataType:minLength = 0 DataType:pattern = DataType:type = decimal</p>	
<p><i>notes</i> Decimale data type for year (e.g. 2020).</p>	<p><i>notes</i> Decimale data type for length (quantity - duration in number, but not in time) (e.g. 1200).</p>	<p><i>notes</i> Decimale data type for month/calendar key.</p>	
«dataType» ShortText_T	«dataType» ShortDate_T	«dataType» ShortTime_T	
<p><i>tags</i> DataType:maxLength = 10 DataType:minLength = 0 DataType:pattern = DataType:type = string</p>	<p><i>tags</i> DataType:maxLength = 10 DataType:minLength = 0 DataType:pattern = DataType:type = string</p>	<p><i>tags</i> DataType:maxLength = 10 DataType:minLength = 0 DataType:pattern = DataType:type = string</p>	
<p><i>notes</i> Short data type for 200 data. The price, month, day, day, etc. should be used for short text (e.g. Invoice number, etc.).</p>	<p><i>notes</i> Short data type for 200 data. The price, month and day, etc. should be used for short text (e.g. Invoice number, etc.).</p>	<p><i>notes</i> Short data type for 200 data. The price, month and day, etc. should be used for short text (e.g. Invoice number, etc.).</p>	
«dataType» Calendar_T	«dataType» Calendar_T	«dataType» Calendar_T	
<p><i>tags</i> DataType:maxLength = 10 DataType:minLength = 0 DataType:pattern = DataType:type = string</p>	<p><i>tags</i> DataType:maxLength = 10 DataType:minLength = 0 DataType:pattern = DataType:type = string</p>	<p><i>tags</i> DataType:maxLength = 10 DataType:minLength = 0 DataType:pattern = DataType:type = string</p>	
<p><i>notes</i> Decimale data type for year (e.g. 2020).</p>	<p><i>notes</i> Decimale data type for year (e.g. 2020).</p>	<p><i>notes</i> Decimale data type for year (e.g. 2020).</p>	

<p>«dataType» String_T</p> <p>tags</p> <p>DataType:contentType = DataType:maxLength = 100 DataType:minLength = DataType:pattern = DataType:type = string</p> <p>notes</p> <p>UDS? String data type</p>	<p>«dataType» Code_T</p> <p>tags</p> <p>DataType:maxLength = 30 DataType:minLength = DataType:pattern = DataType:type = string</p> <p>notes</p> <p>String data type - typically it is used for formatted unique identification which is based on sequence (e.g. Invoice number) or specific unique identification for codelist item.</p>	<p>«dataType» Name_T</p> <p>tags</p> <p>DataType:maxLength = 100 DataType:minLength = DataType:pattern = DataType:type = string</p> <p>notes</p> <p>String data type - typically it is used for the main name which is displayed in browses and tiles.</p>	<p>«dataType» Description_T</p> <p>tags</p> <p>DataType:maxLength = 150 DataType:minLength = DataType:pattern = DataType:type = string</p> <p>notes</p> <p>String data type - typically it is used for basic description of item, especially for codelist item.</p>
<p>«dataType» PersonName_T</p> <p>tags</p> <p>DataType:maxLength = 50 DataType:minLength = DataType:pattern = DataType:type = string</p> <p>notes</p> <p>String data type - typically it is used for the person first name, surname, middle name and so on.</p>	<p>«dataType» ExtId_T</p> <p>tags</p> <p>DataType:maxLength = 50 DataType:minLength = DataType:pattern = DataType:type = string</p> <p>notes</p> <p>String data type - it is used for external identification of the item in other information system.</p>	<p>«dataType» LongText_T</p> <p>tags</p> <p>DataType:maxLength = 4000 DataType:type = string</p> <p>notes</p> <p>String data type - typically it is used for additional description of item.</p>	
<p>«dataType» Phone_T</p> <p>tags</p> <p>DataType:maxLength = 20 DataType:minLength = DataType:pattern = DataType:type = string</p> <p>notes</p> <p>String data type - phone number. Pattern?</p>	<p>«dataType» Url_T</p> <p>tags</p> <p>DataType:maxLength = 2048 DataType:minLength = DataType:pattern = DataType:type = string</p> <p>notes</p> <p>String data type - URL reference. Pattern?</p>	<p>«dataType» Email_T</p> <p>tags</p> <p>DataType:maxLength = 255 DataType:minLength = DataType:pattern = DataType:type = string</p> <p>notes</p> <p>String data type - e-mail address Pattern?: [0-9a-zA-Z]([-.\ w]*[0-9a-zA-Z]*@[0-9a-zA-Z]([-.\ w]*[0-9a-zA-Z])?.)+[0-9a-zA-Z]{2,4}</p>	
<p>«dataType» HouseNo_T</p> <p>tags</p> <p>DataType:maxLength = 10 DataType:minLength = DataType:pattern = DataType:type = string</p> <p>notes</p> <p>String data type - house number.</p>	<p>«dataType» ZipCode_T</p> <p>tags</p> <p>DataType:maxLength = 10 DataType:minLength = DataType:pattern = DataType:type = string</p> <p>notes</p> <p>String data type - zip code.</p>	<p>«dataType» Json_T</p> <p>tags</p> <p>DataType:contentType = application/json DataType:maxLength = 32767 DataType:type = string</p> <p>notes</p> <p>String data type for application JSON content (contentType="application/json")</p>	

Data Types based on ,string‘

Data Types based on ,decimal'

class Data Types

«dataType»
Decimal_T

tags

DataType:maxExclusive =
 DataType:maxInclusive =
 DataType:minExclusive =
 DataType:minInclusive =
 DataType:pattern =
 DataType:precision = 28
 DataType:scale =
 DataType:type = decimal

notes

LIDS? Decimal data type

«dataType»
Integer_T

tags

DataType:maxExclusive =
 DataType:maxInclusive =
 DataType:minExclusive =
 DataType:minInclusive =
 DataType:pattern =
 DataType:precision = 10
 DataType:scale = 0
 DataType:type = decimal

notes

Decimal data type - only integer (scale = 0)

«dataType»
Price_T

tags

DataType:maxExclusive =
 DataType:maxInclusive =
 DataType:minExclusive =
 DataType:minInclusive =
 DataType:pattern =
 DataType:precision = 16
 DataType:scale = 4
 DataType:type = decimal

notes

Decimal data type for price

«dataType»
Quantity_T

tags

DataType:maxExclusive =
 DataType:maxInclusive =
 DataType:minExclusive =
 DataType:minInclusive =
 DataType:pattern =
 DataType:precision = 16
 DataType:scale = 4
 DataType:type = decimal

notes

Decimal data type for quantity (physical amount of something)

«dataType»
YearNo_T

tags

DataType:maxInclusive =
 DataType:minInclusive = 0
 DataType:pattern =
 DataType:precision = 4
 DataType:scale = 0
 DataType:type = decimal

notes

Decimal data type for year, e.g. 2018.

«dataType»
Duration_T

tags

DataType:maxExclusive =
 DataType:maxInclusive =
 DataType:minExclusive =
 DataType:minInclusive =
 DataType:pattern =
 DataType:precision = 16
 DataType:scale = 4
 DataType:type = decimal

notes

Decimal data type for specific quantity - duration. It is number, but it should be visible as time, for example 2,5 in database means 2:30 h.

«dataType»
ColumnId_T

tags

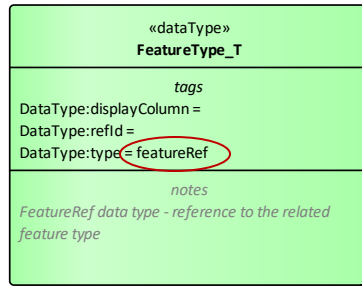
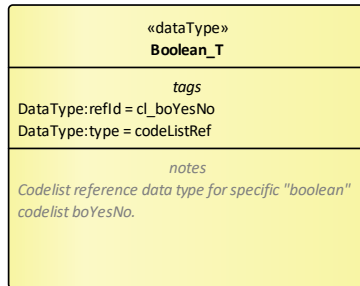
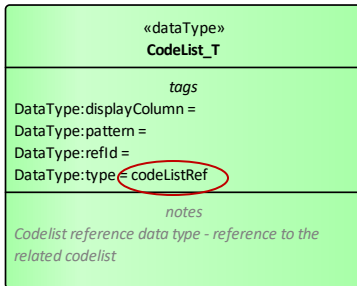
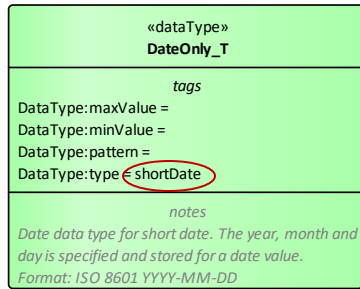
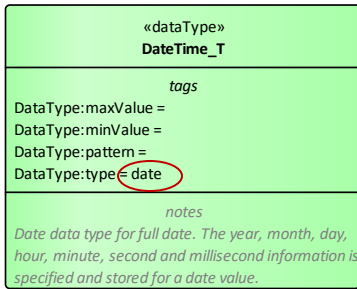
DataType:precision = 10
 DataType:type = decimal

notes

Decimal data type for codelist primary key

Other Data Types

class Data Types



Naming Convention for LIDS model

Naming Conventions (example I)

featureType	ft_	Id	ft_{dom}FeatureTypeName <i>Note: It is recommended to make the identifier at most 15 (max 20) characters long (including prefix).</i>	ft_boDefect ft_eleCable
		DB	<i>Not required (Feature Type is represented in database via the assigned container DB name).</i>	
featureAttribute	at_	Id	at_{dom}FeatureTypeName_attributName	at_boDefect_code at_eleCable_description
		DB	attributName	CODE DESCRIPTION
container	ct_	Id	ct_{dom}ContainerName If the container is intended just for one feature type, then the name should be the same as for the feature type. If the container is for more than one feature types that are connected via relation Generalization, then the name should be derived from the "highest" feature type in hierarchy.	ct_boDefect ct_eleCable
		DB	CT_{dom}ContainerName	CT_BODEFECT CT_ELECABLE

Keep it as short as possible

One underscore!

No underscores in DB!

Naming Conventions (example II)

featureAttribute variant: Codelist reference	Id	<p>at_{dom}FeatureTypeName_cl_{dom}CodelistName{diff}</p> <p>Where optional {diff} is used in one reference to the same code necessary, the CodelistName may be truncated.</p>	at_boDefect_cl_boDefectType
	DB	cl_{dom}CodelistName{diff}	CL_BODEFECTTYPE
featureAttribute variant: Feature reference	Id	<p>at_{dom}FeatureTypeName_fr_{dom}FeatureTypeNameRef{diff}</p> <p>Where FeatureTypeNameRef is feature type being referred to. Optional suffix {diff} is used in case when more than one reference to the same feature is needed. If necessary, the FeatureTypeNameRef may be truncated.</p> <p>Especially in case of a self-reference, the FeatureTypeNameRef may be omitted and for suffix {diff} is recommended "superId" or "sup" (if a shortened version is required).</p> <p><i>Note: Identifier may be at most 30 characters long (including prefix).</i></p>	<p>at_boDefect_fr_boWorkOrder</p> <p>at_boOrgUnit_fr_sup</p>
	DB	fr_{dom}FeatureTypeNameRef{diff}	FR_BOWORKORDER

cl_prefix

fr_prefix

No SID!

is_prefix

Name Conventions (example III)

is_prefix

<i>featureAttribute</i> variant: <i>Boolean</i>	<i>Id</i>	at_{dom}FeatureTypeName_is_Name The similar convention as for the codelist reference is	at_boDefect_is_fixed
		used here because the Boolean value is modeled as reference to the specific codelist with two values Yes/No.	
	<i>DB</i>	is_Name	IS_FIXED

- Description of all naming conventions is available as a documentation of SAMO implementation.
- Naming conventions can be set by each project

Basic transformation rules

Codelist

codeList : boDefectType

Properties

- General
- Templates

Rules

- Requirements
- Constraints
- Scenarios

Related

- Files
- Links

boDefectType

CodeList contains the types of defect.

id (without prefix cl_)

description

name („label“)

Stereotype: codeList

Status: Proposed

Alias: Defect Type

Keywords:

Language: Java

Version: 1.0

Phase: 1.0

Package: LIDS7 Extension Example

Created: 31.12.2018 9:50:02

Modified: 06.01.2019 18:06:45

Main Details Advanced LIDS7 Tags

OK Cancel Apply Help

«codeList» boDefectType
«column» - id: CodelistID_T - code: Code_T - description: Description_T
tags dbName = hierarchical = false size = small

Codelist (II)

codeList : boDefectType

boDefectType

LIDS7: codeList (boDefectType)

dbName	
hierarchical	false
size	small

CodeList contains the types of defect.

Main Details Advanced **LIDS7** Tags

OK Cancel Apply Help

```
«codeList»
boDefectType

«column»
- id: CodeListID_T
- code: Code_T
- description: Description_T

tags
dbName =
hierarchical = false
size = small
```

Codelist Column

The screenshot shows the 'boDefectType : Features' tool with a table of attributes and their database mappings. The table has columns: Name, Type, Scope, Stereotype, Alias, and Initial Value.

Name	Type	Scope	Stereotype	Alias	Initial Value
id	CodelistID_T	Private	column	Id	
code	Code_T	Private	column	Code	
description	Description_T	Private	column	Description	

Annotations:

- A yellow callout points to the 'id' attribute, stating: **id (without prefix ca_)**
- A yellow callout points to the 'Alias' column, stating: **name („label“)**
- A yellow callout points to the 'dbName' property in the 'LIDS7::column (code)' section, stating: **dbName (only when it is not compatible with naming rules)**

On the right, a green box shows the UML class diagram for the codelist:

```
«codeList»
boDefectType

«column»
- id: CodelistID_T
- code: Code_T
- description: Description_T

tags
dbName =
hierarchical = false
size = small
```

Feature Type

featureType : boDefect

Properties

- General
- Templates

Rules

- Requirements
- Constraints
- Scenarios

Related

- Files
- Links

boDefect

id (without prefix ft_)

Main entity for evidence of the defects.

description

name („label“)

Alias: Defect

Stereotype: featureType

Status: Proposed

Keywords: Radim BYSTRICKÝ

Priority: Easy

Language: Java

Version: 1.0

Phase: 1.0

Package: LIDS7 Extension Example

Created: 17.12.2018 10:27:53

Modified: 06.01.2019 19:16:02

Main Details Advanced LIDS7 Tags

OK Cancel Apply Help

```
classDiagram
    class "«featureType» boDefect" {
        - code: Code_T
        + name: SemanticID_T
        + cl_boDefectType: CodeList_T
        «generatedAttribute»
        + codeSeq: Integer_T
    }
    class tags {
        container =
    }
```

Feature Type (II)

featureType : boDefect

boDefect

Properties

- General
- Templates

Rules

- Requirements
- Constraints
- Scenarios

Related

- Files
- Links

Main entity for evidence of the defects.

LIDS7:featureType (boDefect)

- container

container
(only when it is not compatible with naming rules)

Main Details Advanced LIDS7 Tags

OK Cancel Apply Help

«featureType»
boDefect

«attribute»

- code: Code_T
- + name: SemanticID_T
- + cl_boDefectType: CodeList_T

«generatedAttribute»

- + codeSeq: Integer_T

tags

container =

Feature Attribute

The screenshot displays the 'boDefect : Features' configuration window. The main table lists attributes with their properties:

Name	Type	Scope	Stereotype	Alias	Initial Value
code	Code_T	Private	attribute	Code	
name	SemanticID_T	Public	attribute	Name	
cl_boDefectType	CodeList_T	Public	attribute	Defect Type	
codeSeq	Integer_T	Public	generatedAttri...	Auxiliary number	

Annotations in yellow callouts point to specific fields:

- id (without prefix at_)**: Points to the 'code' attribute name.
- name („label“)**: Points to the 'Name' alias for the 'code' attribute.
- dbName (only when it is not compatible with naming rules)**: Points to the 'dbName' field in the 'LIDS7::attribute (code)' configuration panel.

The 'Attribute (code)' configuration panel shows the following properties:

Containment	Not Specifi...
Static	False
Property	
Const	False
Is Literal	False
Transient	False
Derived	False
Derived Union	False
Is ID	False
Multiplicity	[1]
Is Collection	False
Container Type	

The 'Notes' panel shows the following configuration for 'LIDS7::attribute (code)':

dbName	
nillable	true
from Code_T	
Data Type.pattern	
Data Type.minLength	0
Data Type.type	string
Data Type.maxLength	30

On the right, a UML diagram shows the '«featureType» boDefect' structure:

```
«featureType»
boDefect
- code: Code_T
+ name: SemanticID_T
+ cl_boDefectType: CodeList_T
«generatedAttribute»
+ codeSeq: Integer_T

tags
container =
```

Description of attribute

Name	Type	Scope	Stereotype	Alias	Initial Value
dateOfBirth	DateOnly_T	Private	attribute	Date of Birth	
firstName	PersonName_T	Private	attribute	First Name	
middleName	PersonName_T	Private	attribute	Middle Name	
surname	PersonName_T	Private	attribute	Surname	
name	Name_T	Private	derivedAttribute	Name	

New Attribute...

Use Notes for „decoration“

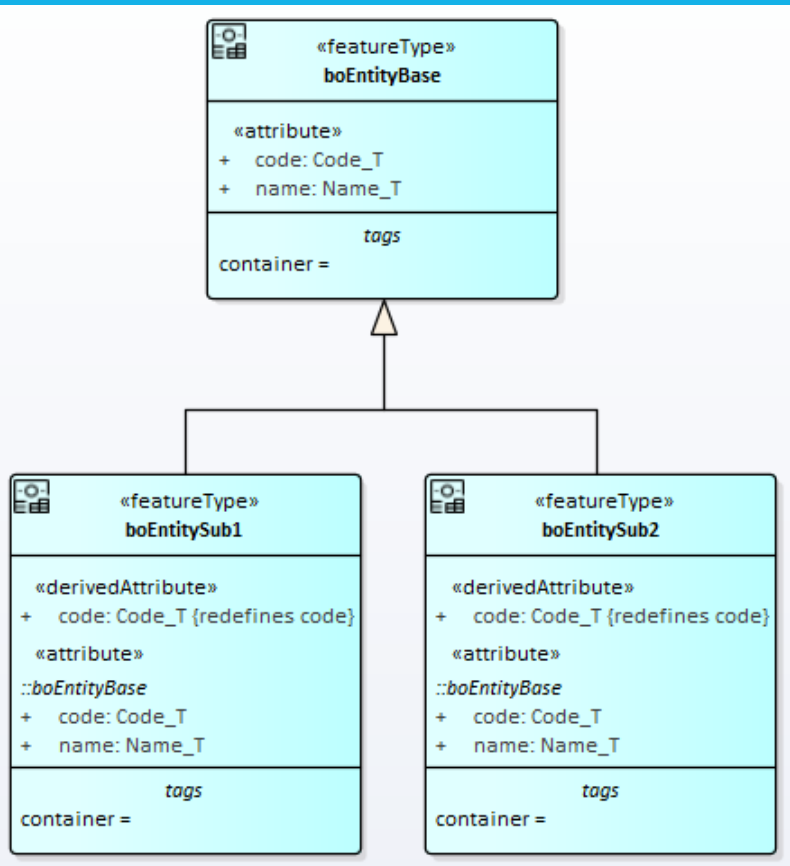
Notes Constraints Redefines Tagged Values

derivedAttribute = {surname} {firstname} {middlename}

```
- <ber:derivedAttribute name="Name" id="at_boPerson_name" dbName="NAME" nillable="true">  
  <ber:description>derivedAttribute = {surname} {firstname} {middlename}</ber:description>  
</ber:derivedAttribute>
```

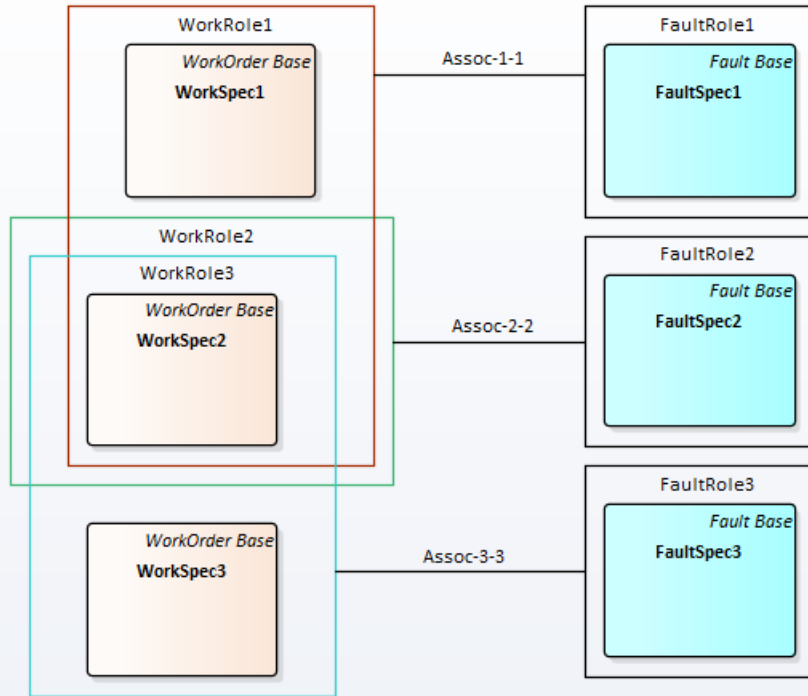
Advanced transformation rules

Containers



- If the feature type is not a part of the „inheritance tree“ then the container is generated as 1:1 with this feature type
- If the feature type is a part of the „inheritance tree“ then the container is generated for the „root“ feature type and all other feature types are included in this container
- It is possible to use the tagged value ‚container‘ to define a specific container for the feature type

The container ct_boEntityBase contains all feature types that are related via the generalization



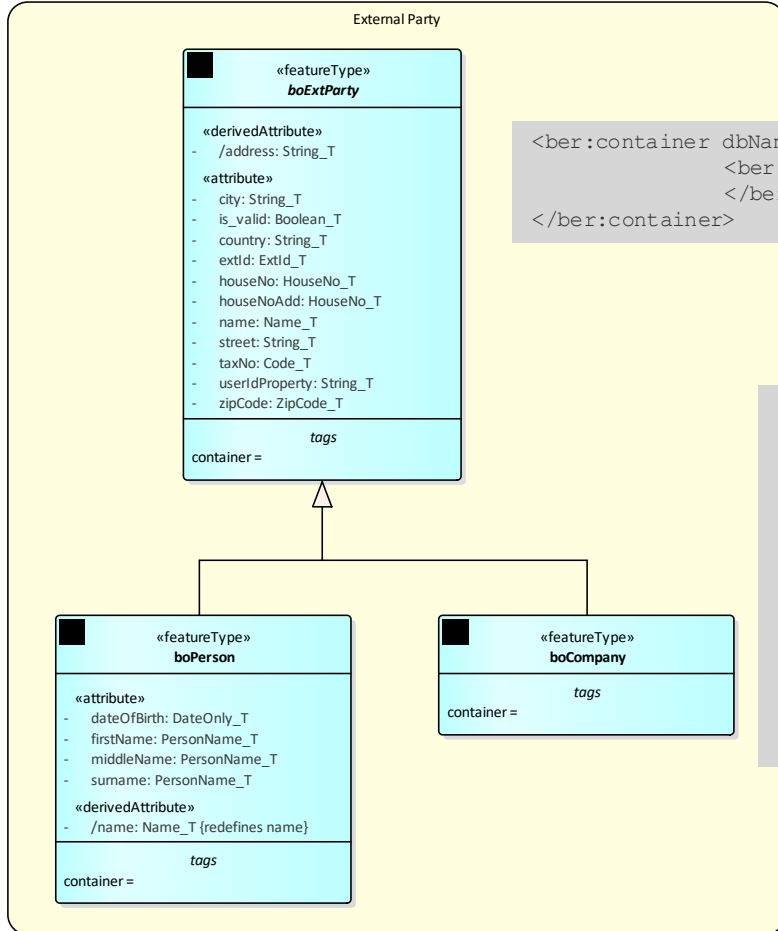
In standard modeling you need several associations and roles for the same „business“ relation „WorkOrder for fixing Fault“ (in the case there exists an association between the complex „inheritance“ structures).

Attention

- Redesign of roles in the model can lead to a loss of information about previous relations between objects!

Recommendations for now:

- Design one role with all feature types in the „inheritance“ tree
- Allow all-to-all combination
- Refactoring later – LIDS should allow to set the combinations in special extension



One container

```

<ber:container dbName="CT_BOEXTPARTY" id="ct_boExtParty" name="External Party" versioned="true">
  <ber:description>Table containing the information about given external party.
</ber:description>
</ber:container>
  
```

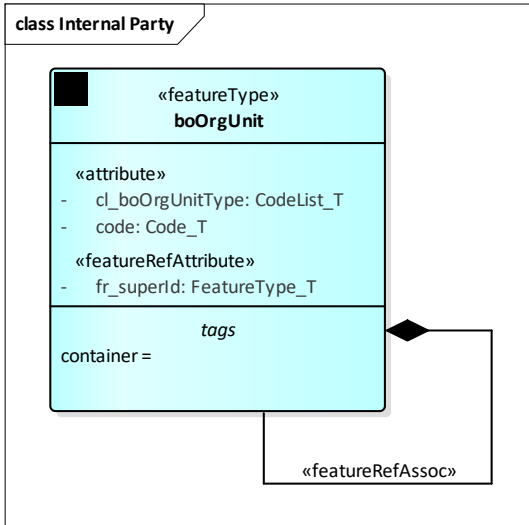
```

<!-- Role :: BOEXTPARTY -->
<ber:relationRole dbName="BOEXTPARTY" id="rt_boExtParty" name="External Party">
  <ber:ftItem refId="ft_boExtParty"/>
  <ber:ftItem refId="ft_boCompany"/>
  <ber:ftItem refId="ft_boPerson"/>
</ber:relationRole>
<!-- Role :: BOCOMPANY -->
<ber:relationRole dbName="BOCOMPANY" id="rt_boCompany" name="Company">
  <ber:ftItem refId="ft_boCompany"/>
</ber:relationRole>
<!-- Role :: BOPERSON -->
<ber:relationRole dbName="BOPERSON" id="rt_boPerson" name="Person">
  <ber:ftItem refId="ft_boPerson"/>
</ber:relationRole>
  
```

Three „default“ roles

Roles - self-reference

For self-reference association you need two different roles



```
<ber:relationRole dbName="BOORGUNIT" id="rt_boOrgUnit_superior" name="Organizational Unit">
  <ber:ftItem refId="ft_boOrgUnit"/>
</ber:relationRole>
<ber:relationRole dbName="BOORGUNIT" id="rt_boOrgUnit" name="Organizational Unit">
  <ber:ftItem refId="ft_boOrgUnit"/>
</ber:relationRole>
```

```
<ber:featureRefRelationAssoc id="as_boOrgUnit_boOrgUnit" name="boOrgUnit -&gt; boOrgUnit">
  <ber:masterRole cardinality="1" refId="rt_boOrgUnit_superior"/>
  <ber:childRole cardinality="unbounded" refId="rt_boOrgUnit"/>
</ber:featureRefRelationAssoc>
```

Patterns

Patterns

Add Pattern Basic Patterns::LIDS7: Codelist Pattern to Diagram

Pattern Basic Patterns::LIDS7: Codelist Pattern (Ver: 1.0)

```
«codeList»  
boCodeList  
  
- column =  
- id: CodelistID_T  
- code: Code_T  
- description: Description_T  
  
tags  
dbName =  
hierarchical = false  
size = small
```

Pattern Elements: Use Auto Names

Name	Type	Action	Default
boCodeList	Class	Create	boCodeList

Element Notes:

Import as Package Fragment

OK Cancel Help

Resources

- Document Generation
- MDG Technologies
- Matrix Profiles
- Favorites
- Stylesheets
- UML Profiles
- Patterns
 - Basic Patterns
 - LIDS7: Codelist Pattern
 - LIDS7: Codelist

View Pattern Details
Add Pattern to Diagram
Delete Pattern

Start Page *Logical Data Model x LID

```
«codeList»  
boCodeList  
  
«column»  
- id: CodelistID_T  
- code: Code_T  
- description: Description_T  
  
tags  
dbName =  
hierarchical = false  
size = small
```

Patterns with relation

Add Pattern Basic Patterns::LIDS7: Codelist with Reference Pattern to Diagram

Pattern Basic Patterns::LIDS7: Codelist with Reference Pattern (Ver: 1.0)

Createc new codelist incl. relation to feature.

Pattern Elements:

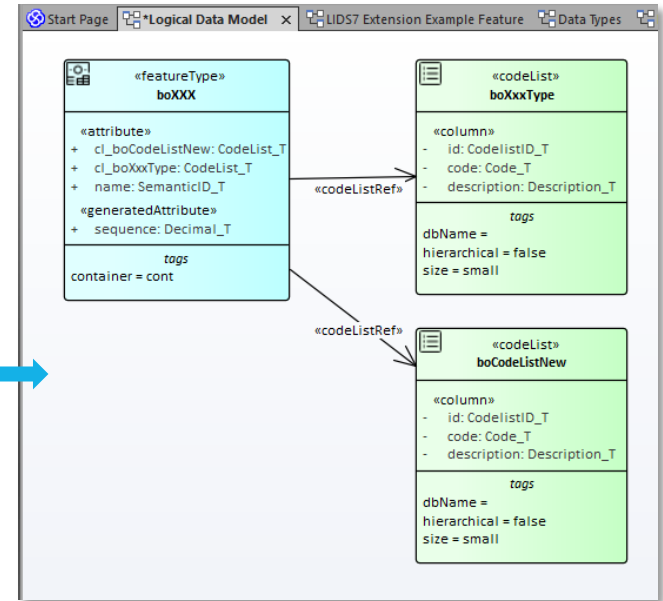
Name	Type	Action	Default
boFT	Class	Merge	boXXX
boCodeListNew	Class	Create	boCodeListNew

Use Auto Names

Element Notes:

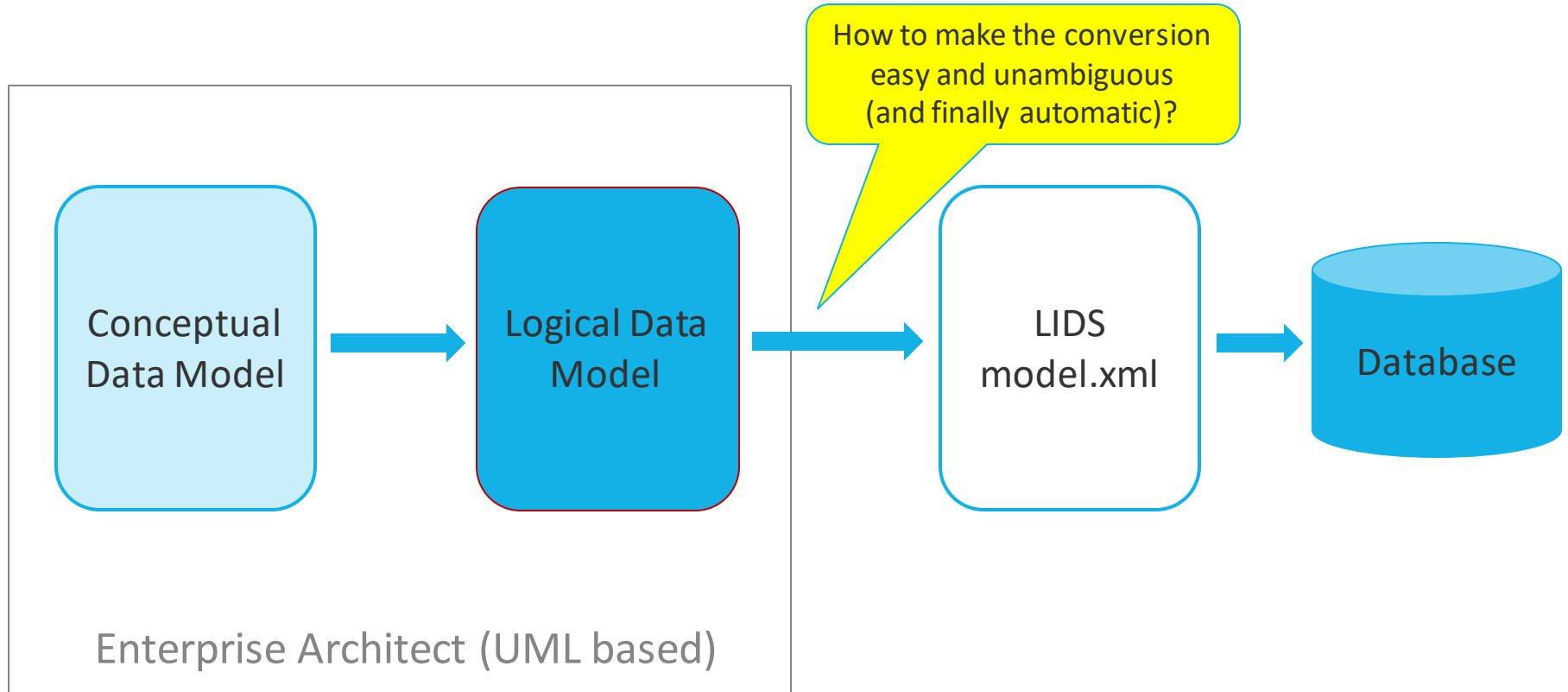
Import as Package Fragment

OK Cancel Help

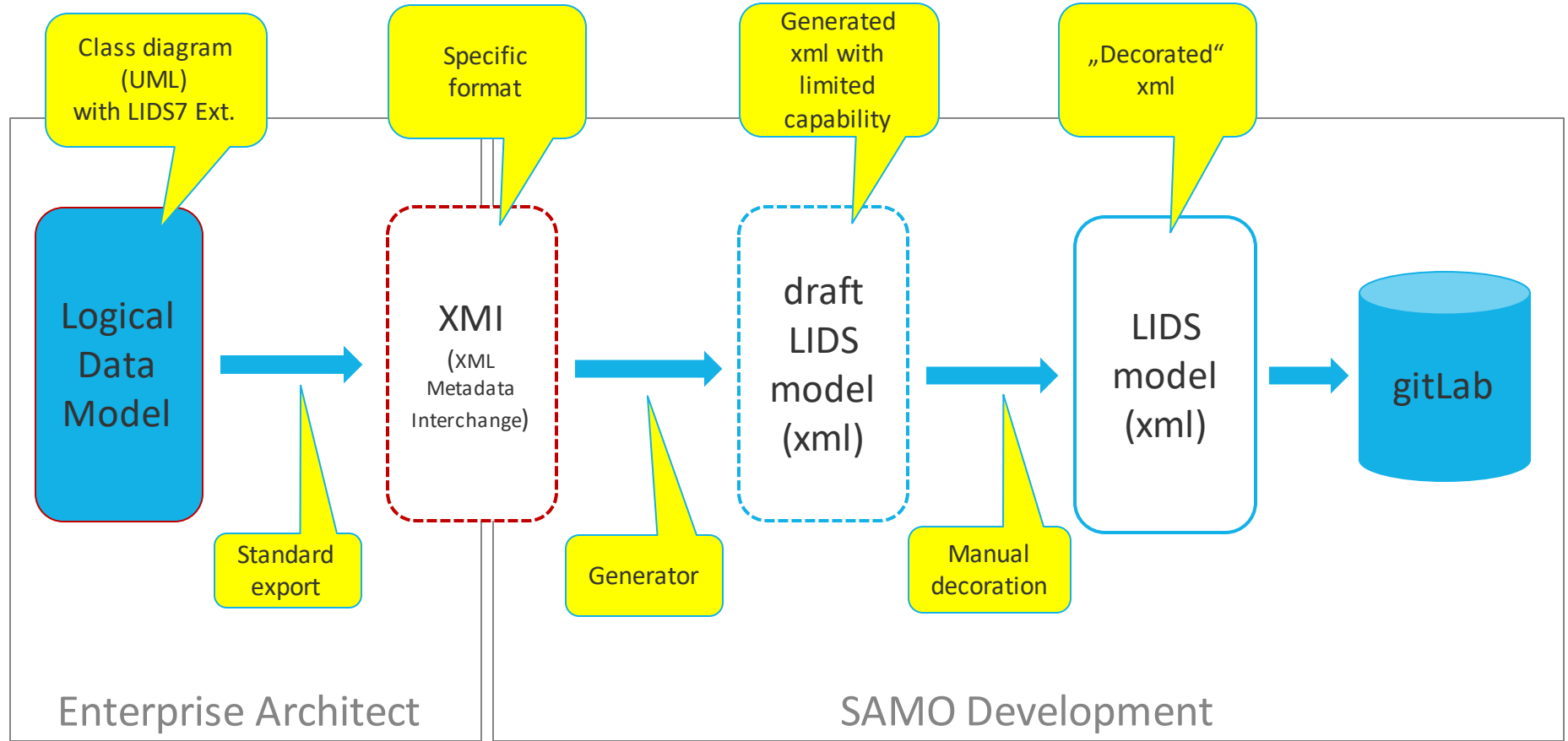


„Production line“ for LIDS model

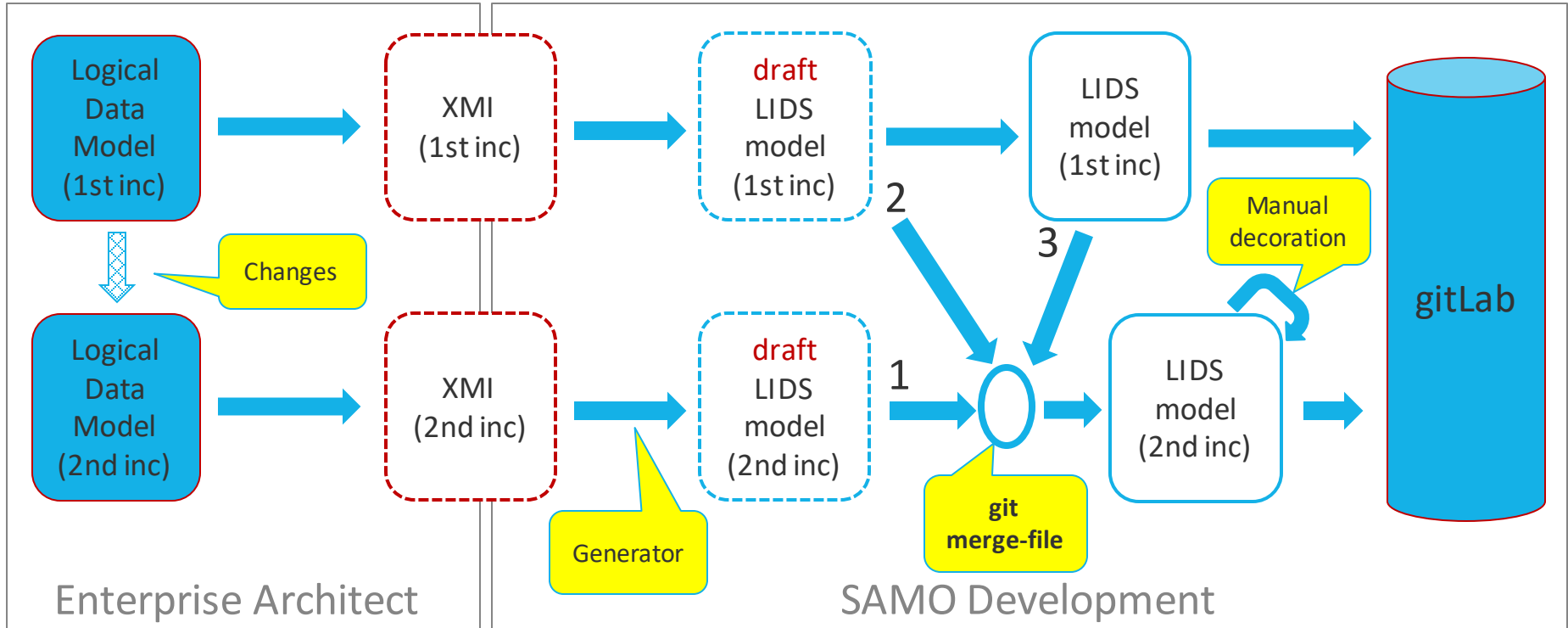
SAMO Data Model



Automatization - concept



Automatization – increments



EA2LIDS Add-in

The screenshot shows the Enterprise Architect software interface. The top menu bar includes 'Start', 'Design', 'Layout', 'Specialize', 'Publish', 'Construct', 'Simulate', 'Code', 'Execute', 'Configure', and 'Find Command...'. The 'Specialize' menu item is highlighted with a red box and the number '2'. Below the menu bar, the 'Tools' section contains 'Portals', 'Database Builder', 'Schema Composer', 'Scripting', 'Integration', 'Manage Publish', 'ArchiMate', 'Manage Windows', and 'Gitlab Extension'. The 'Gitlab Extension' toolbar contains an 'Upload package' button, which is highlighted with a red box and the number '3'. The Project Browser on the left shows a tree view of the project structure, with the 'LDD_EPA External Party' folder highlighted by a red box and the number '1'. The Toolbox on the right shows a list of LIDS7 components, including Feature Type, Codelist, Security Codelist, System Table, Attachment Type, Class, LIDS7 Connectors, LIDS7 Attributes, and Component.

1. Select package in project browser
2. Navigate into Specialize category in top menu or right-click to see the context menu
3. Click on Gitlab Extension module and then on Upload package button

EA2LIDS Add-in

The screenshot shows a 'Package Upload' window with a progress bar at the top containing seven steps: Configure, Upgrade, Git pull, Export XML, Run generator, Git merge file, and Git push. The 'Run generator' step is highlighted with a red box and a '1' next to it. Below the progress bar, there are two sections: one with a Git commit link and file paths (highlighted with a red box and a '2'), and a large text area showing log output (highlighted with a red box and a '3'). The log output includes error messages, file paths, and Git commands. A red box with a '4' is also present in the top right corner of the window, containing the text 'ea2lids generator 12'.

```
Package Upload
```

Configure Upgrade Git pull Export XML Run generator Git merge file Git push 1

ea2lids generator 12
4

Git commit: https://gitlab/samo/customers/asseco-ce/eam/ea2lids_ewr/commit/9b68608 2

Generated file: C:\dev\ea2lids_models\ea2lids_ewr\models_lids_generated\LDD_EPA_External_Party.xml

Decorated file: C:\dev\ea2lids_models\ea2lids_ewr\models_lids_decorated\LDD_EPA_External_Party.xml

```
[ERROR] Attribute length exceeded 30 character limit: at_boExtPartyCont_fr_boExtParty
[INFO] Writing package to:
C:\dev\ea2lids_models\ea2lids_ewr\models_lids_generated\LDD_EPA_External_Party.xml
[STEP] Git merge file
[INFO] Running command: c:\dev\ea2lids\generator\Main.py pipeline-merge -p "LDD_EPA_External_Party" -r
https://gitlab/samo/customers/asseco-ce/eam/ea2lids\_ewr.git
[INFO] OK
C:\dev\ea2lids_models\ea2lids_ewr\models_lids_decorated\LDD_EPA_External_Party.xml
[STEP] Git push
[INFO] Running command: c:\dev\ea2lids\generator\Main.py pipeline-push -p "LDD_EPA_External_Party" -r
https://gitlab/samo/customers/asseco-ce/eam/ea2lids\_ewr.git -g https://gitlab/samo/customers/asseco-ce/eam/ea2lids\_ewr
[INFO] git add .
[INFO] git commit
[INFO] [dev 9b68608] Automatic package upload: LDD_EPA_External_Party
3 files changed, 402 insertions(+), 435 deletions(-)

[INFO] git push
[INFO] remote:
remote: To create a merge request for dev, visit:
remote: https://gitlab/samo/customers/asseco-ce/eam/ea2lids\_ewr/merge\_requests/new?merge\_request%5Bsource\_branch%5D=dev
remote:
to ssh://gitlab:20022/samo/customers/asseco-ce/eam/ea2lids_ewr.git
4d02214..9b68608 dev -> dev

Commit ID: https://gitlab/samo/customers/asseco-ce/eam/ea2lids\_ewr/commit/9b68608
[INFO]
"Life is what happens to us while we are making other plans" AS
```

1. You can see pipeline steps in top left corner of the window as colored rectangles with name of the step below them. If pipeline step was successful, the color is green, red if otherwise.
2. Link to gitlab commit and links to generated file and decorated file are in section below pipeline steps.
3. Large text area is supposed to show log messages. There is a standalone log section for every pipeline step with same name as the pipeline step.
4. Generator version is located in top right corner.

Automatization - support

- Add-in EA
- Git support
- Impact
 - All team members have to work in the same way
 - Logical Data Model in EA is the **development artefact** (!)
 - No „sketching“
 - Problem has to be fixed in the source (= EA)
 - Manual „decoration“ only for structures that are not supported in Generator

Asseco Central Europe, a.s.

Trenčianska 56/A
821 09 Bratislava
Slovak Republic
Phone: +421 2 20 838 400

Budějovická 778/3a
140 00 Praha 4
Czech Republic
Phone: +420 234 292 500

sales@asseco-ce.com

asseco.com/ce

Solutions
for demanding
business.

asreco

Legal disclaimer

This presentation is the property of Asseco Central Europe (Asseco CE) business group. Information presented serves for marketing purposes only and constitutes neither an offer to sell nor a solicitation to buy. Asseco CE accepts no liability whatsoever for any loss arising directly or indirectly from the use of, reliance of any information contained in this presentation or for any omission of the information. The processing, copying, recording on information carriers, as well as making this presentation or any part thereof available in any way to third parties requires the prior consent of Asseco CE member.