SAMO Delivery

SAMO/LIDS Naming Convention

Version: 0.9

Last updated: 07. 01. 2019

Author(s): Radim Bystřický

asseco.com/ce

Identification	This document contains the basic rules for creating and working with SAMO analytical model. The rules are to be followed by all the employees working on the SAMO project in order to ensure compatibility and quality of all the product parts.
Owner	Utility BU, Head of Analysis Department (Radim Bystřický)
Classification of information	INTERNAL

Version history

Version	Release date	Author(s)	Description of changes
09	7.1.2019	Radim Bystřický	Update of the documentation into the new document format. Minor changes in the document content.
10	18.1.2019	Radim Bystřický	Added recommendations and examples for codelists and feature types
11	6.2.2019	Radim Bystřický	Change for container – dbName is with prefix CT_

Distribution

No.	Name and surname / Place of distribution

Content

1 LIDS Model Convention

1.1 Introduction

This document contains the proposal of terminology convention for basic LIDS metadata attributes.

Basic principles

- Name should be **singular**.
- **camelCase** convention is used for LIDS model (it means no underscore in name, except predefined prefixes and constructions - see the table below).
- For database is used simple **UPPERCASE** convention (it means no underscore in name, except predefined prefixes and constructions - see the table below).
- Defined prefixes are mandatory because the objects must have **a unique identification** in the LIDS model.
- Expresion {**dom**} is the abbreviation of the business domain/field of interest (not mandatory).
    - „**bo**“ is intended for SAMO business objects.
    - The abbreviation of the utility is often used for assets, e.g. „**ele**“, „**gas**“, ...

1.2 Codelist

The information below contains the rules applicable for codelists of LIDS Model.

Metadata	Prefix	Part	Name Convention	Examples
codeList	**cl_**	*Id*	**cl_{dom}CodeListName**	cl_boDefectType cl_eleMaterial
		DB	**cl_{dom}CodeListName**	CL_BODEFECTTYPE CL_ELEMATERIAL
codeListColumn	**ca_**	*Id*	**ca_{dom}CodeListName_columnName**	ca_boDefectType_code ca_eleMaterial_id
		DB	**columnName**	CODE ID
codeListBinding	**cb_**	*Id*	**cb_{dom}BindingCodeListName** Where **BindingCodeListName** should contain a pair of codelist names the values of which are the content of the bounded codelist. In the case of triple (and more) bindings, the name needs to be defined as a specific one and should describe the purpose of the binding. *Note: Identifier may be at most **30** characters long (including prefix).*	cb_boDefectType_eleMaterial
		DB	**cb_{dom}BindingCodeListName**	CB_BODEFECTTYPE_ELEMATERIAL
boundedItem		*Id*	*Not required*	
		DB	**cl_{dom}CodeListName**	CL_BODEFECTTYPE
filteredCodeList	**cf_**	*Id*	**cf_{dom}CodeListName_filter** Where: • The 1st part has the name of the base codelist; • The 2nd part has the name of the purpose of filtration (e.g. in the example, the codelist is derived from docelist cl_boDefectType and filtration is performed for types which are considered as damage).	cf_boDefectType_damage cf_eleMaterial_cable
		DB	*Not required (filtered codelist does not exist in the DB)*	

1.2.1 Standardized Codelist Structure

Recommendation for columns naming in the standardized codelist:

Id	Name	DB	Description
ca_xxx_id	Id	ID	Primary key.

ca_xxx_code	Code	CODE	Unique code that is understandable for the user.
ca_xxx_description	Description	DESCRIPTION	Description that is understandable for the user.
ca_xxx_superId	Superior Id	SUPERID	Self-reference to the superior id (usable only when codelist is defined as hierarchical).

1.3 Feature Type and Container

Metadata	Prefix	Part	Name Convention	Examples
featureType	**ft_**	*Id*	**ft_{dom}FeatureTypeName** *Note: It is recommended to make the identifier at most **15 (max 20)** characters long (including prefix).*	ft_boDefect ft_eleCable
		DB	*Not required (Feature Type is represented in database via the assigned container DB name).*	
featureAttribute	**at_**	*Id*	**at_{dom}FeatureTypeName_attributName**	at_boDefect_code at_eleCable_description
		DB	**attributName**	CODE DESCRIPTION
featureAttribute variant: Codelist reference		*Id*	**at_{dom}FeatureTypeName_cl_{dom}CodeListName{diff}** Where optional **{diff}** is used in case when more than one reference to the same codelist is needed. If necessary, the **CodelistName** may be truncated.	at_boDefect_cl_boDefectType
		DB	**cl_{dom}CodelistName{diff}**	CL_BODEFECTTYPE
featureAttribute variant: Feature reference		*Id*	**at_{dom}FeatureTypeName_fr_{dom}FeatureTypeNameRef{diff}** Where **FeatureTypeNameRef** is the name of the feature type being reffered 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. 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). *Note: Identifier may be at most **30** characters long (including prefix).*	at_boDefect_fr_boWorkOrder at_boOrgUnit_fr_sup
		DB	**fr_{dom}FeatureTypeNameRef{diff}**	FR_BOWORKORDER
featureAttribute variant: Virtual attribute		*Id*	The same rule as for the ordinary attribute is applied, with the rule that the identifier may be at most **30** characters long (including prefix).	
featureAttribute variant: Boolean		*Id*	**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.	
		DB	**is_Name**	IS_FIXED
featureForm	**fmd_**	*Id*	**fmd_{dom}FeatureTypeName{_formName}** Where **formName** briefly describes the purpose of the form. Form for determination of columns in a tabular statement of entity is without formName. Form for the editing detail of the entity has name with suffix **_detail**.	fmd_boDefect fmd_boDefect_detail
relationRole	**rt_**	*Id*	**rt_{dom}RoleTypeName** Where **RoleTypeName** briefly describes the purpose of the role. The role may represent a container or feature type or group of feature types.	rt_boDefect rt_eleCable
		DB	**{dom}RoleTypeName**	BODEFECT ELECABLE
relationAssoc	**as_**	*Id*	**as_{dom}AssociationName** Where **AssociationName** should contain a pair of role names that create an association. The aim is for the "name" to be "readable", e.g. AS_TRAFOSTATION_DEFECT in the sense of "the given Trafo station has a Defect" or in the order "AS_higher level_subordinate" role, e.g. AS_LIGHTING_INSPECTION (0..N inspection is recorded for the given light).	as_trafostation_boDefect as_powerCable_boDefect as_eleCable_route
		DB	**as_{dom}AssociationName**	AS_TRAFOSTATION_BODEFECT AS_POWERCABLE_BODEFECT AS_ELECABLE_ROUTE
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
groupContainer	**gc_**	*Id*	**gc_{dom}GroupContainerName** Where **GroupContainerName** briefly describes the purpose of the group container.	gc_bo
		DB	**gc_{dom}GroupContainerName**	GC_BO

1.3.1 Standardized “strong” entity

For “strong” entity is characteristic that it has proper unique user’s identification, workflow and several subordinated “weak” entities.

In the table below, see the recommendation for columns naming in the standardized feature type that represents a “strong” entity:

Id	Name	DB	Description
at_xxx_code	Code	CODE	Unique code that is understandable for the user (typically it is an expression derived from the other attribute).
at_xxx_name	Name	NAME	Short description that can represent the entity in many application contexts.
at_xxx_description	Description	DESCRIPTION	Long description.
at_xxx_codeSeq	Auxiliary number	CODESEQ	Auxiliary number that is generated via number generator and is used in derived expression for attribute Code.
at_xxx_cl_xxxState	State	CL_XXXSTATE	Reference to the state codelist (usable only when entity has defined the workflow). Often this attribute is defined as security attribute.
at_xxx_startDate **at_xxx_endDate**	Start date End date	STARTDATE ENDDATE	Date when the entity has stared its lifecycle. Date when the entity has finished its lifecycle. When another pair of date items is needed, additional prefix should be used. For example for planned starting date should be used planStartDate.

1.3.2 Standardized “weak” entity

Recommendation for feature type naming when the feature type represents a “weak” entity.

Feature Type	Id	Description
Non-abstract “strong” entity	ft_boDefect	Name of “strong” entity is created in standard way. If there exist more descendants from abstract ancestor then an suitable suffix can be used, e.g. ft_boDefectEle, ft_boDefectGas, and so on.
Subordinated “weak” entity	ft_boDefAsset ft_boDefEmpl ft_boDefCost	Names for the subordinate “weak” entities are created in this way: • From the name of superior entity is created suitable unique abbreviation (3-5 characters) • Subordinate entity is named as: **ft_{dom}{abbr}FeatureTypeName** *Note: “weak” entity is typically related to the “strong” entity via composite relation.*
abstract ancestor	ft_boDefBase	**ft_{dom}{abbr}Base** Name of abstract ancestor should contain the suffix “**base**” to distinguish it from non-abstract “real” descendants that are used in application.

1.3.3 Specific attribute convention

Id	Name	DB	Description

at_boDefect_approvedBy	Approved By	APPROVEDBY	To express an “action” that is performed by a user, use the suffix “**By**”.
at_boDefect_detectionDate	Date of Detection	DETECTIONDATE	To express a “date” value with a certain meaning, use the suffix “**Date**”.
at_boDefect_sumRealCosts	Sum of Real Costs	SUMREALCOSTS	To express a statistical value with a certain meaning, use the appropriate prefix, such as: “**sum**”, “**cnt**”, “**max**”, “**min**”, and so on. This is often used for naming of derived attributes.

1.4 Other Metadata

Metadata	Prefix	Part	Name Convention	Examples
numberGenerator	**seq_**	*Id*	**seq_{dom}NumberGeneratorName** This is the recommended usage of the featureType identificator for the generator name if it makes sense, for example when it is used for generating the unique attribute ‘code’.	seq_boDefect
		DB	**seq_{dom}NumberGeneratorName**	SEQ_BODEFECT
attachmentType	**att_**	*Id*	**att_{dom}AttachmentName** This is the recommended usage of the featureType identification for the attachment name if it makes sense, but the attachment type is often quite general, for example “photo”, and so on.	att_boProtocol att_eleDocument

2 Sparx Enterprise Architect Convention

When the CASE tool Sparx Enterprise Architect is used for analysis, the naming rules for codelists, feature types, attributes etc. must be followed as well. On the other hand, in Enterprise Architect are typically used the specific stereotypes that indicate the type of target metadata object in LIDS model. So it is not necessary for analyst to use all the prefixes when he defines the identificators in Enterprise Architect. The mandatory prefixes are added during the transformation from logical data model to physical LIDS model.

Below you can see several examples how to use items in Enterprise Architect when data model is developed.

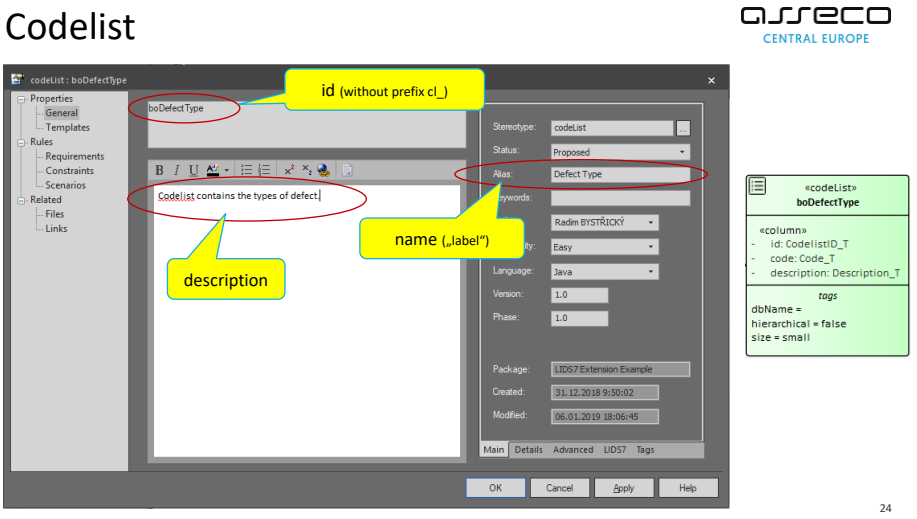


Image 1: Enterprise Architect – Codelist Example I.

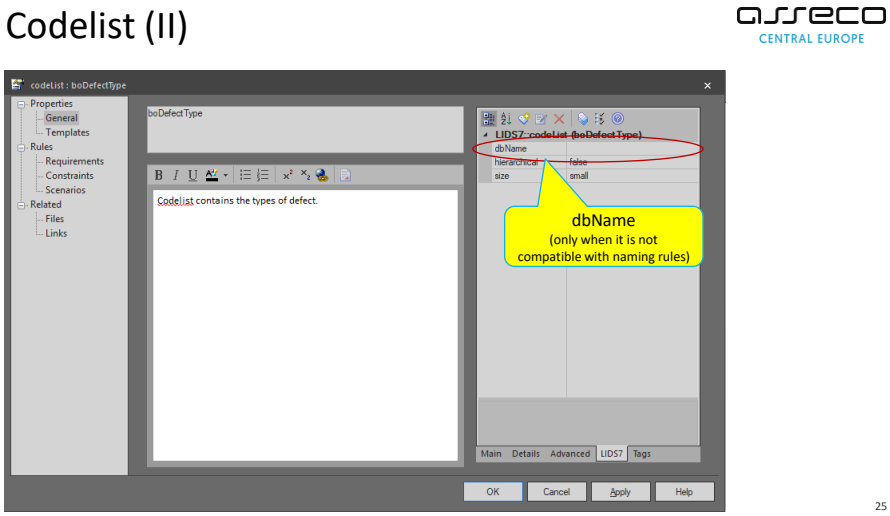
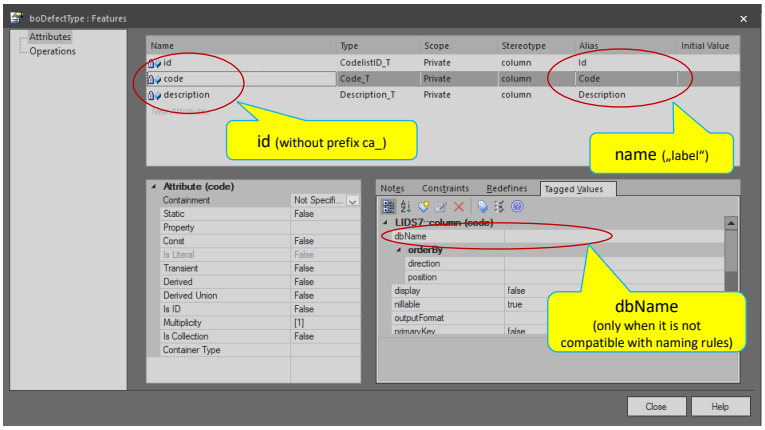


Image 2: Enterprise Architect – Codelist Example II.

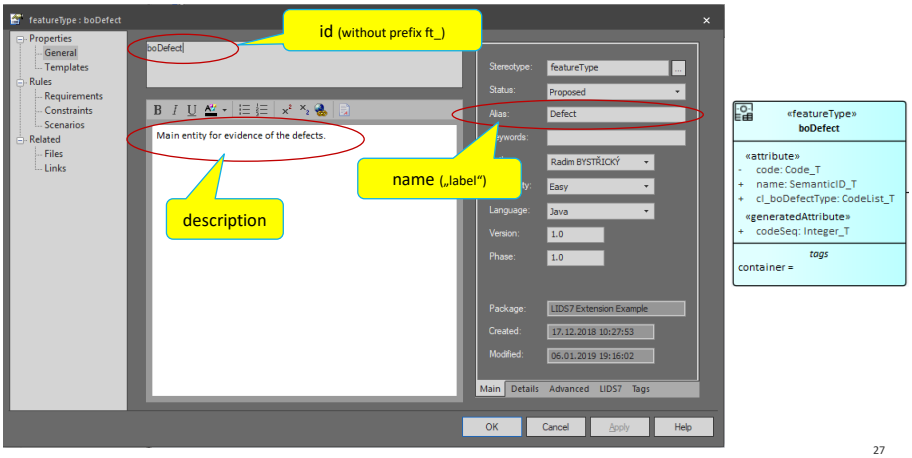
Codelist Column



26

Image 3: Enterprise Architect – Codelist Column Example

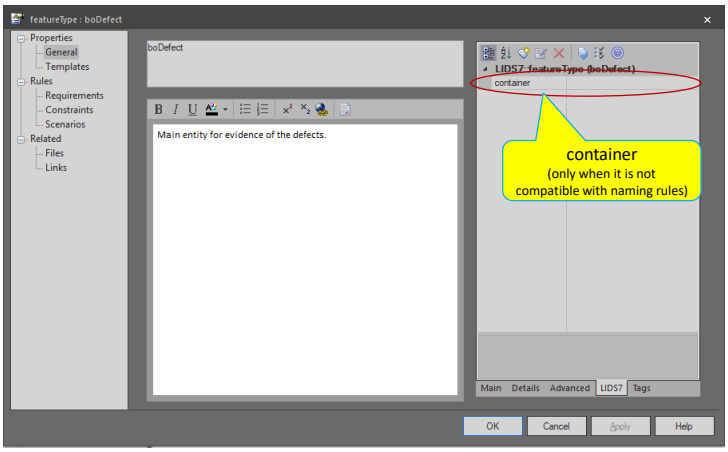
Feature Type



27

Image 4: Enterprise Architect – Feature Type Example I.

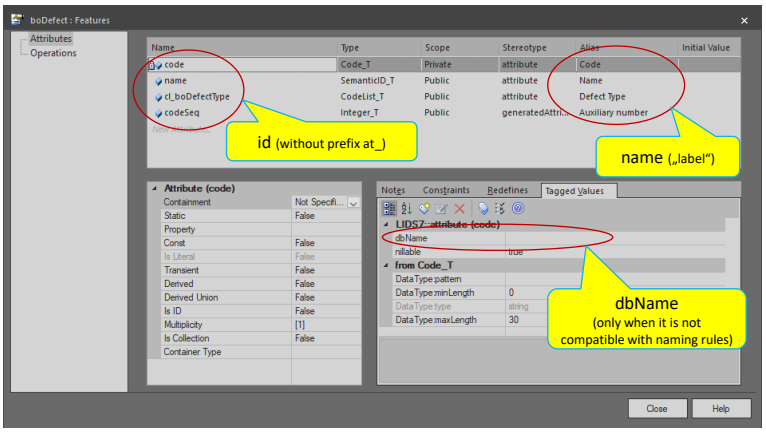
Feature Type (II)



28

Image 5: Enterprise Architect – Feature Type Example II.

Feature Attribute



29

Image 6: Enterprise Architect – Feature Attribute Example

3 Linked Documents

In the table below are listed the mentioned or important documents related to the topic of this document.

Document name	Description	Attached document / link to document

4 Open issues

In the table below are listed open issues (issues connected to this document without solution).

No.	Description and communication	Issue owner	Resolution

4.1 Closed issues

The table below contains all the issues that have been successfully closed.

No.	Description and communication	Issue owner	Resolution