TabLan.15926

Data Mapping Specification

Ver. 2.3

06.08.2013

Moscow

Contents

Revision history

Version 2 (published 08.03.2012)
First English release of the methodology.

Version 2.1 (published 30.09.2012)
Updates to terminology and code samples with the release of .15926 Editor version 1.00 and .15926 Platform announcement.
Changes to the mapping of infinite cardinality (*) – non-numeric values in cardinality template roles avoided.

Version 2.2 (published 19.11.2012)
Diagram and sample file changes as new Methodology reference data schema elements became available in TechInvestLab.ru RDL sandbox.
Corrected mistakes in the statement modality classification mapping for classes of classes.
References to the release of open-source TabLan extension for .15926 Editor version 1.10.
Meta-data namespace changed to *http://techinvestlab.ru/meta#* .

Version 2.3 (published 06.08.2013)
Namespace for PCA reference data library changed to *http://posccaesar.org/rdl/* as PCA changed namespace in library export file to the one used at endpoint.
Namespace for p7tpl templates changed to *http://standards.iso.org/iso/ts/15926/-8/ed-1/tech/reference-data/p7tpl#* as it was changed in files accompanying official ISO release of Part 8.

1. General information

This specification (hereafter the Specification) defines rules for transformation of TabLan.15926 (table-based data description language) data tables into the ISO 15926 – compliant data model in RDF/OWL. TabLan.15926 language (or simply TabLan) and its usage are described in an accompanying document **«TabLan.15926 The Methodology of a Technical Document Semantic Modeling»** (hereafter the Methodology).

The transformation rules defined in this Specification (the mapping) are programmed for computer execution in Python using API of .15926 Builder module. Some general notes on the approach to mapping used for .15926 Platform are in the Annex 1 to this Specification.

The software module doing the mapping is distributed as open-source extension to the .15926 Editor (available at http://techinvestlab.ru/dot15926Editor). The Editor also does a visualization of ISO 15926 models. The software is developed by TechInvestLab.ru. License, terms of distribution and usage can be found with software package.

TabLan extension of the .15926 Editor available at the link above performs TabLan tables' transformation into RDF/OWL abiding by many restrictions and conventions described in the Methodology and in the Specification. Open-source extension can be modified for a specific project according to the rules of a particular organization using the Methodology.

Transformation rules defined below are illustrated using the diagram language of the ISO 15926 Part 7, augmented with a set of additional notation elements described in the Annex 2 of this Specification.

It is assumed that that the reader of this Specification is familiar with the Parts 1, 2, 7 and 8 of the standard **ISO 15926 «Industrial automation systems and integration — Integration of life-cycle data for process plants including oil and gas production facilities»**.

2. Items in the model

During the transformation of TabLan data description tables into ISO 15926 – compliant data model the following data items are created:

- Instances of the ISO 15926 Part 2 entity types (individuals, classes, relationships).

  Please notice that in diagrams below Part 2 entity types shown for some instances of the model are most general entity types recommended by the Methodology for particular kinds of items.

- Instances of the ISO 15926 Part 7 templates.

  The set of templates used by .15926 Editor in data transformation should be available in the Part 8 compliant template definition file on user's computer.

  For mapping in .15926 Editor TabLan extension **p7tpl_enhanced.owl** file is distributed together with the software.

  The **p7tpl.owl** file is planned for distribution with the ISO 15926 Part 8, as noted in the Annex D of the Part 8 draft. The templates in this file correspond to the templates of the Sections 6.3 and 7.5 and of the Annex C of the ISO 15926 Part 7, but do not completely coincide with them, and the names may differ.

The **p7tpl_enhanced.owl** file distributed with the software for use in TabLan transformation is the original **p7tpl.owl** file enhanced by one additional template ***DescriptionByInformationObject***, defined in the Annex 3 of this Specification.

This mixed approach (with Part 2 type and template instances) was chosen in search of a balance between two major factors: compactness of a template form and the need to have explicit Part 2 relationships in data model for classification purposes. Unfortunately formally template instances can not be further classified or described in any other way with other templates. The mixed approach is also a good illustration for .15926 Builder API capabilities.

3. Reference data, project data and metadata namespaces

In a RDF/OWL file created according to this Specification the following namespaces are used:

a. Namespaces defined by W3C consortium:

rdfs: http://www.w3.org/2000/01/rdf-schema#
rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
xsd: http://www.w3.org/2001/XMLSchema#
owl: http://www.w3.org/2006/12/owl2-xml#
owllist: http://www.co-ode.org/ontologies/list.owl#

b. Namespaces required by Part 8 of the ISO 15926 (namespace prefixes in a RDF/OWL file may be different from those listed below and used for visualization):

dm: http://rds.posccaesar.org/2008/02/OWL/ISO-15926-2_2003#
rdl: http://posccaesar.org/rdl/
A model built with default configuration of the .15926 Editor software can include entities from a local copy of the PCA RDL reference data library. Therefore all its namespace conventions are used, including PCA RDL namespace for Part 2 data model.
p7tpl: http://standards.iso.org/iso/ts/15926/-8/ed-1/tech/reference-data/p7tpl#
meta: http://standards.iso.org/iso/ts/15926/-8/ed-1/tech/reference-data/metadata#

c. A namespace for meta-data declarations in data models developed by TechInvestLab.ru. In this namespace properties **label_en** and **label_ru** are declared for national language identifications (so far there is no agreement on the single best implementation of ISO 15926 in multilingual environment):

til: http://techinvestlab.ru/meta#

d. A namespace for new reference and project data created in TabLan table transformation with the.15926 Editor can be specified in transformation settings or left at default value:

http://example.org/item#

New reference and project data URIs are formed in this namespace using identifier made by concatenating string "id" with the UUID compliant to RFC 4122 / ITU-T X.667 / ISO/IEC 9834-8.

These namespaces are defined for a default configuration of the TabLan extension in .15926 Editor. For a specific project namespaces used by extension should be modified according to the rules of a particular organization using the Methodology.

4. Table transformation

The TabLan data description tables are maintained as electronic spreadsheets in an Office Open XML format (**.xlsx**), supported by Microsoft Excel, OpenOffice or LibreOffice software tools. Tables in the old format of Microsoft Excel (.xls) are not supported by transformation software! A sample spreadsheet file is distributed with the Methodology.

The header of each table is supposed to be in the first row of the table. Tables are processed row-by-row.

The rows where the **Section** cell is filled can be used for comments, as all other fields in the row with non-empty **Section** cell are ignored in transformation.

All columns without header or with a header different from the ones defined in the Methodology are ignored and can be used for comments.

If information in a cell marked *optional* is absent, items (Part 2 type instances or template instances) dependant on it are not created.

If information in a cell marked *mandatory* is absent, the table translation is stopped with the error message (software control is not implemented in some cases, where a cell is mandatory or optional depending on the value in some other cell).

If the content of the cell is not recognized (broken naming rules for Part 2 types, different spelling of item names in different cells, unexpected data type, etc.) the software issues an error message.

4.1. Document structure model table transformation

Data description table of the structural model is located on the **Structure Model** tab of the model spreadsheet file. The header of the **Structure Model** data description table:

Section	Item ID	Item Class	Item Text	Item of Whole	Item Subject	Classifier Class

Section	*optional*
Item ID	*mandatory*
Item Class	*mandatory*
Item Text	*mandatory*
Item of Whole	*mandatory* for items classified as Element Collection and Elementary Statement
Item Subject	*mandatory* for items classified as Elementary Statement
Classifier Class	*optional*

The Transformation of a **Structure Model** table leads to the creation of:

An instance of the ClassOfInformationObject type with:
- unique name: **Item ID**
- attribute meta:annTextDefinition=**Item Text**

An unnamed instance of the Specialization type with roles hasSubclass=**Item ID**, hasSuperclass=**Item Class**

An unnamed instance of the Classification type with roles hasClassified=Item ID, hasClassifier= **Classifier Class**

Templates' instances:

p7tpl.ClassOfArrangementOfIndividual(**Item ID**, **Item of Whole**)
p7tpl.DescriptionByInformationObject (**Item Subject**, **Item ID**)

Additional row processing:

A completely filled row with some **Item ID** may be followed by one or more rows with only **Item Subject** and/or **Classifier Class** cells filled. For such incomplete rows additional relationships are created as described above between the **Item ID** described in the preceding row and additional **Item Subject** or **Classifier Class** instances.

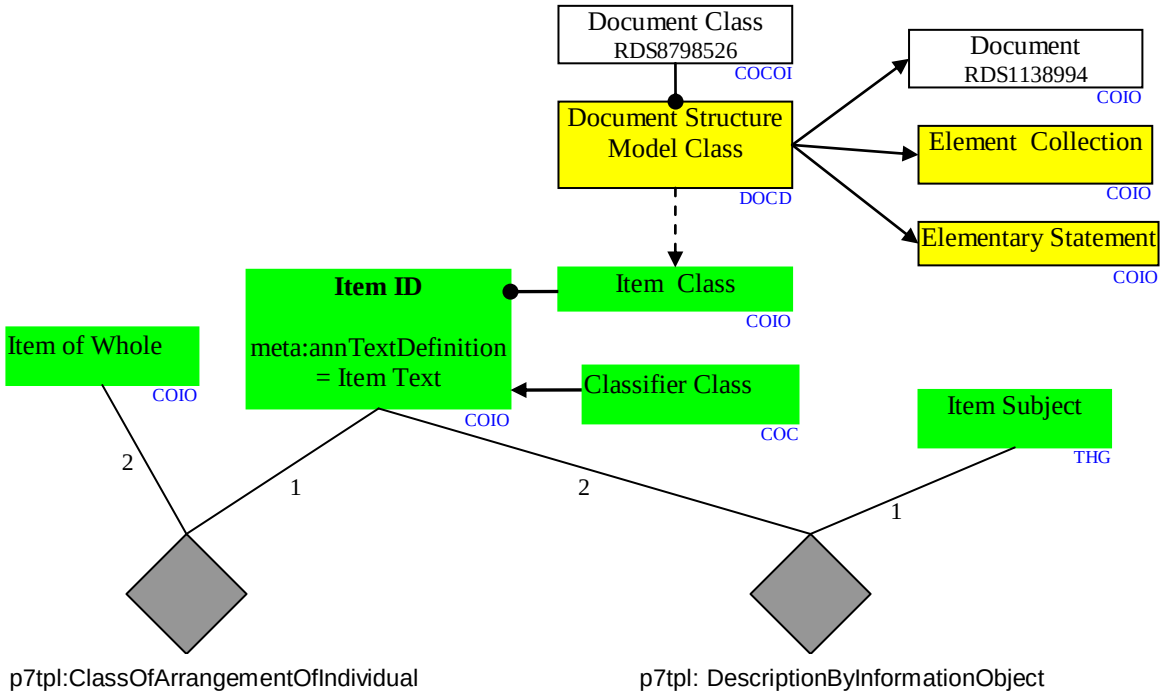


Diagram 1. Transformation of a document structure model table

4.2. Core reference data descriptions table transformation

Core reference data description table is located on the **Class Definitions** tab of the model spreadsheet file. The header of the **Class Definitions** data description table:

Section	URI	Russian Unique Class Name	English Unique Class Name	Part 2 Type	Superclass	Classifier Class	Source

Section *optional*
URI *optional*
Russian Unique Class Name *mandatory* in the absence of English Unique Class Name
English Unique Class Name *mandatory* in the absence of Russian Unique Class Name
Part 2 Type *mandatory*
Superclass *optional*
Classifier Class *optional*
Source *mandatory*

The Transformation of a **Class Definitions** table leads to the creation of:

An instance (denoted below **X)** of the type identified in the cell **Part 2 Type** with:
- unique name: **English Unique Class Name** or **Russian Unique Class Name** in the absence of English Unique Class Name
- attribute til:label_en=**English Unique Class Name** (if present)
- attribute til:label_ru=**Russian Unique Class Name** (if present)
- attribute meta:annSource=**Source**

An unnamed instance of the Specialization type with roles hasSubclass=**X**, hasSuperclass=**Superclass**

An unnamed instance of the Classification type with roles hasClassified=**X**, hasClassifier=**Classifier Class**

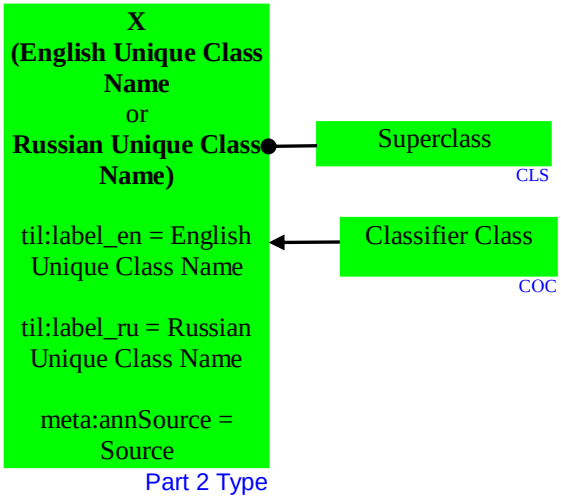


Diagram 2. Transformation of a core reference data description table

URI cell processing:

If URI cell in a row is empty, the item defined in this row is created in the project namespace and recorded in the RDF/OWL file.

If URI cell is not empty, the item described in this row is considered as taken from the external source of reference data, and is not created in the project namespace. In particular, the Russian Unique Class Name for such item will not be saved in the model after the transformation. Two relationships listed above are created in the model and recorded in the RDF/OWL file.

4.3. Properties data description table transformation

Properties table is located on the **Class Properties** tab of the model spreadsheet file. The header of the **Class Properties** data description table:

Section	Class Name	Property Name	Indirect Property Name	Property Min	Property Max	Source

Section	*optional*
Class Name	*mandatory*
Property Name	*mandatory*
Indirect Property Name	*optional*
Property Min	*mandatory*
Property Max	*mandatory*
Source	*mandatory*

The Transformation of a **Class Properties** table leads to the creation of:

An instance of the Property type with unique name **Property Min** (if such instance was not created earlier while processing previous rows)

An instance of the Property type with unique name **Property Max** (if such instance was not created earlier while processing previous rows)

An unnamed instance (denoted below ***range***) of the PropertyRange type with
- attribute meta:annSource=**Source**

An unnamed instance of the Specialization type with roles hasSubclass=***range***, hasSuperclass=**Property Name** and with
- attribute meta:annSource=**Source**

A templates' instance:
p7tpl.LowerUpperOfPropertyRange(***range***, **Property Min**, **Property Max**) with
- attribute meta:annSource=**Source**

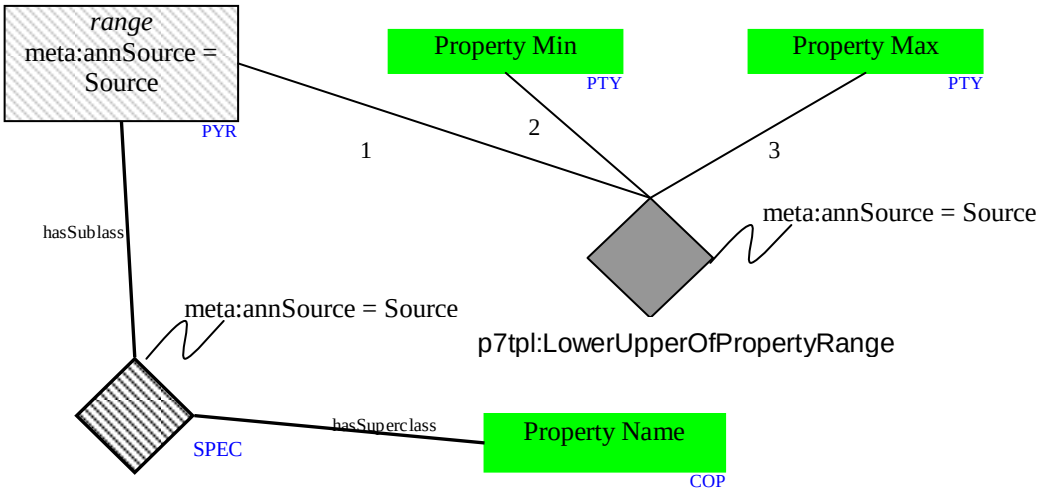


Diagram 3.1. Transformation of a properties table
(creation of a property range)

If the cell **Indirect Property Name** is not empty:

A template instance is created:
p7tpl.PropertyRangeRestrictionOfClass(**Class Name**, **Indirect Property Name**, ***range***) with
- attribute meta:annSource=Source

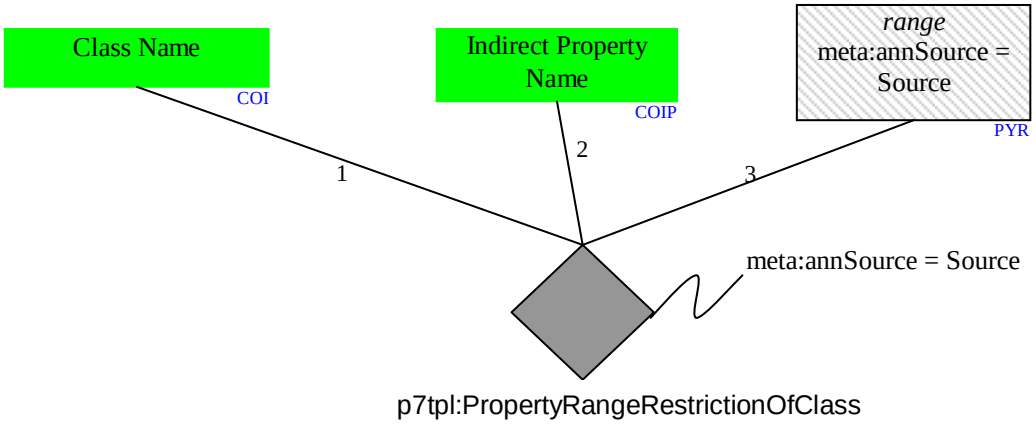


Diagram 3.2. Transformation of a properties table
(indirect property of a class)

If the cell **Indirect Property Name** is empty:

An unnamed instance of ClassOfClassification type is created with roles hasClassOfClassified=**Class Name**, hasClassOfClassifier= ***range*** and with
- attribute meta:annSource=**Source**

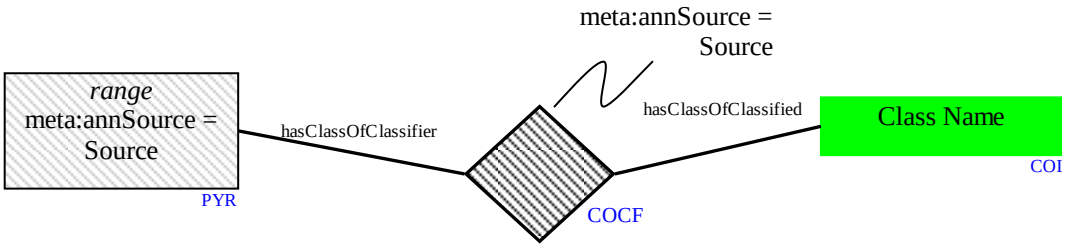


Diagram 3.3. Transformation of a properties table
(in the absence of indirect property)

4.4. Classifiers description table transformation

Classifiers description table is located on the **Classifier Definition** tab of the model spreadsheet file. The header of the **Classifier Definition** data description table:

Section	Class Name	Superclass Name	Classifier Relationship Class	Class of Classifier Classes	Source

Section	*optional*
Class Name	*mandatory*
Superclass Name	*optional*
Classifier Relationship Class	*optional*
Class of Classifier Classes	*optional*
Source	*mandatory*

The Transformation of a **Classifier Definition** table leads to the creation of:

An unnamed instance (denoted below ***classifier relationship***) of the Specialization type with roles hasSubclass=**Class Name**, hasSuperclass=**Superclass Name** and with

- attribute meta:annSource= **Source**

An unnamed instance of the Classification type with roles hasClassifier=**Classifier Relationship Class**, hasClassified=***classifier relationship*** and with

- attribute meta:annSource= **Source**

An unnamed instance of the Classification type with roles hasClassifier=**Class of Classifier Classes**, hasClassified=**Class Name** and with

- attribute meta:annSource= **Source**

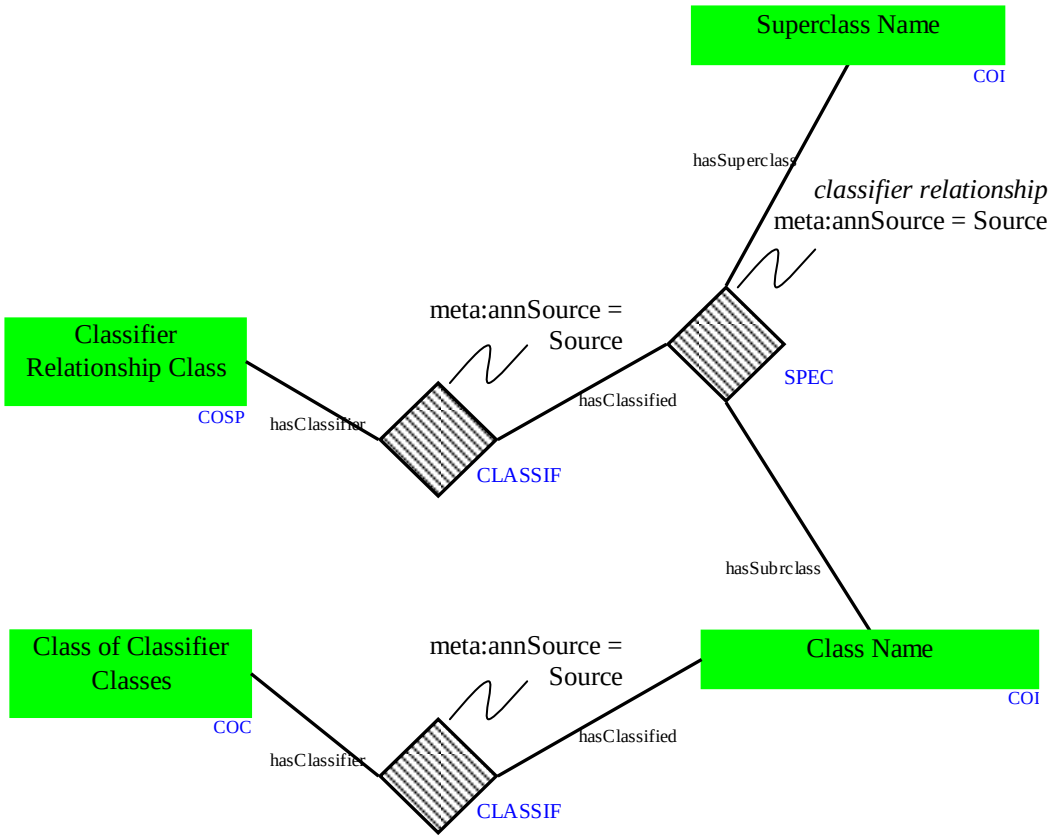


Diagram 4. Transformation of a classifiers description table

4.5. Breakdowns description table transformation

Breakdowns description table is located on the **Breakdown Definition** tab of the model spreadsheet file. The header of the **Breakdown Definition** data description table:

Section	Class Name	Whole Class Name	Breakdown Relationship Class	Class of Breakdown Classes	Source

Section	*optional*
Class Name	*mandatory*
Whole Class Name	*optional*
Breakdown Relationship Class	*optional*
Class of Breakdown Classes	*optional*
Source	*mandatory*

The Transformation of a **Breakdown Definition** table leads to the creation of:

An unnamed instance (denoted hereafter ***breakdown relationship***) of the ClassOfCompositionOfIndividual type with roles hasClassOfPart=**Class Name**, hasClassOfWhole=**Whole Class Name** and with

- attribute meta:annSource= Source

An unnamed instance of the Classification type with roles hasClassifier= **Breakdown Relationship Class**, hasClassified=***breakdown relationship*** and with

- attribute meta:annSource= **Source**

An unnamed instance of the Classification type with roles hasClassifier=**Class of Breakdown Classes**, hasClassified=**Class Name** and with

- attribute meta:annSource= **Source**

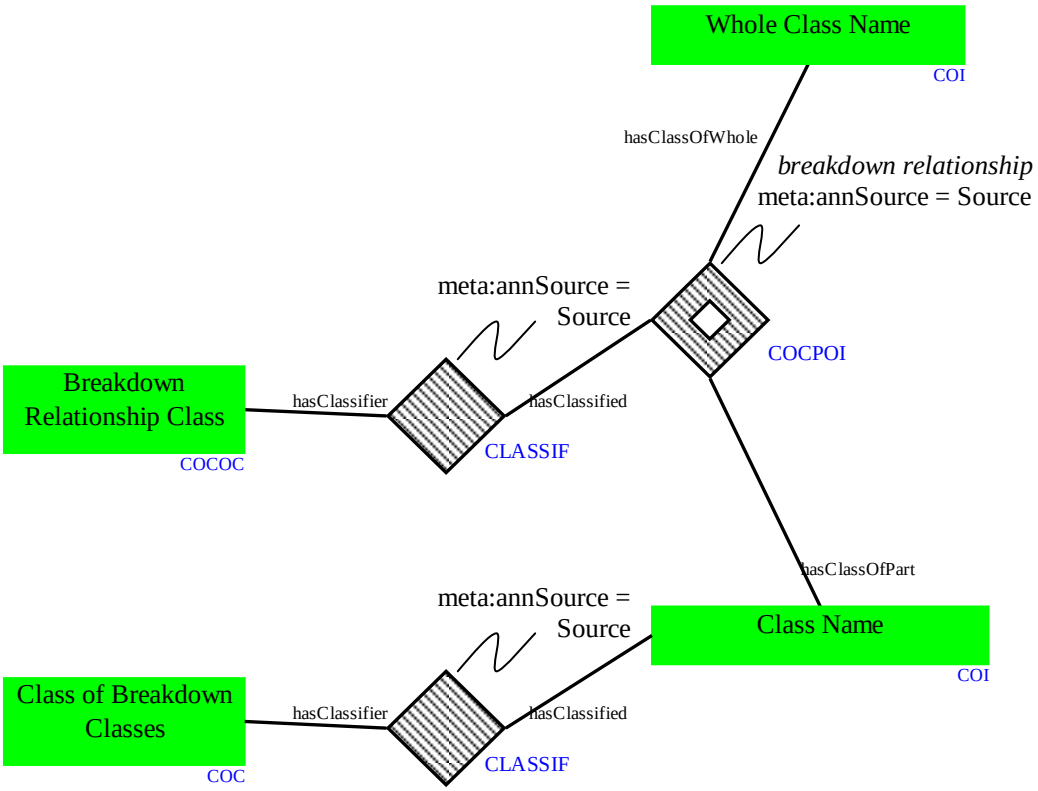


Diagram 5. Transformation of a breakdowns description table

4.6. Other relationships description table transformation

Other relationships description table is located on the **Other Relationships** tab of the model spreadsheet file. The header of the **Other Relationships** data description table:

Section	Role 1	Role 1 Cardinality	Relates to	Role 2	Role 2 Cardinality	Source

Section *optional*
Role 1 *mandatory*
Role 1 Cardinality *optional*
Relates to *mandatory*
Role 2 *mandatory*.
Role 2 Cardinality *optional*
Source *mandatory*

Cardinality cells processing:

The text strings in the cells **Role 1 Cardinality** and **Role 2 Cardinality** have to be in the format ***(#:#)*** where # is a natural number (including 0) or * in a second position for infinity (cardinality ***(*:*)*** should be simply omitted). These strings are transformed into separate cardinality values for **Role1CMin**, **Role1CMax**, **Role2CMin**, **Role2CMax**.

Cardinalities are uniformly processed if the string in the **Relates to** cell of this row is:

«**participates in**»
«**is a predecessor in time of**»
«**has as part**»
«**is performed by**»
«**is related to**»

Cardinalities are ignored for rows with other content of **Relates to** cell.

Main class of relationship created in row transformation is denoted below ***relation***.

If cardinality is processed and both min and max cardinality values for a role are numbers, for the corresponding end of ***relation*** the following entity is created:

Template instances:

p7tpl.CardinalityEnd1MinMax(***relation***, **Role1CMin**, **Role1CMax**) with
- attribute meta:annSource=**Source**

or

p7tpl.CardinalityEnd2MinMax(***relation***, **Role2CMin**, **Role2CMax**) with
- attribute meta:annSource=**Source**

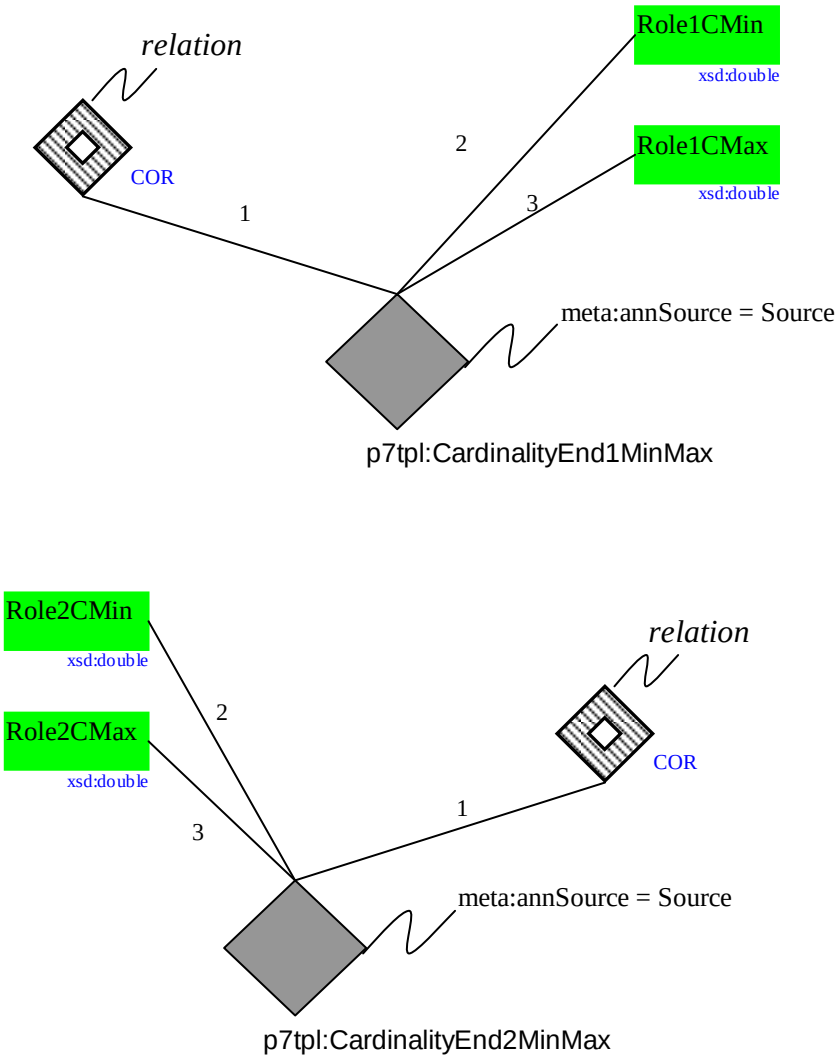


Diagram 6.0.1. General schema of transformation for cardinality with min and max values

If cardinality is processed, and max cardinality value for a role is infinity (marked as *), for the corresponding end of ***relation*** the following entity is created:

Template instances:

p7tpl. CardinalityEnd1Min(***relation***, **Role1CMin**) with

- attribute meta:annSource=**Source**

or

p7tpl.CardinalityEnd2Min(***relation***, **Role2CMin**) with

- attribute meta:annSource=**Source**

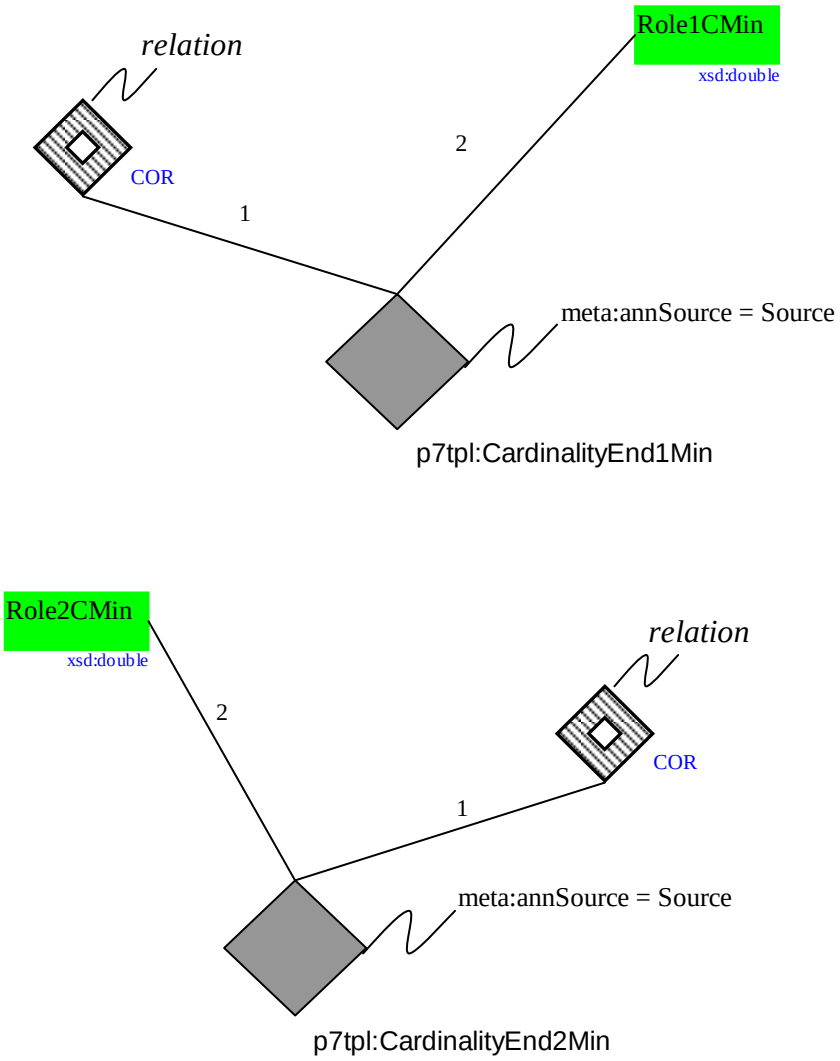


Diagram 6.0.2. General schema of transformation for cardinality with min value only

Relates to cell processing:

The specific way of the row transformation is defined by the string in the **Relates to** cell of this row.

4.6.1. «**is described by**» relationship

The transformation of such a row leads to the creation of:

An unnamed instance (denoted below ***document instance***) of the PossibleIndividual type with
- attribute meta:annSource=**Source**

An unnamed instance of the Classification type with roles hasClassified=***document instance***, hasClassifier=**Role 2**

An unnamed instance of the Description type with roles hasSign=***document instance***, hasRepresented=**Role 1** with
- attribute meta:annSource=**Source**

This transformation mirrors the axiom of the p7tpl.DescriptionByInformationObject template. However the full Part 2 model is instanced here and in the Transformation of a **Requirements** data description table – because in **Requirements** it is necessary to add additional classification to the singled out "required" relationship, which is impossible for template instance.

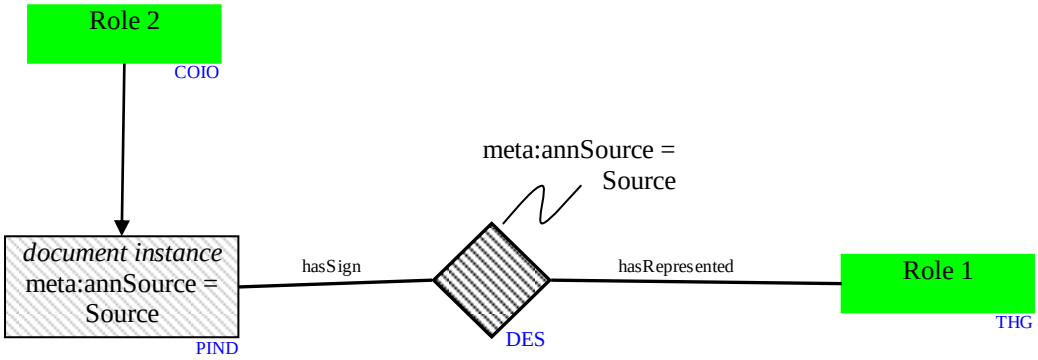


Diagram 6.1. Transformation of an «is **described by**» relationship row

4.6.2. «**participates in**» relationship

The transformation of such a row leads to the creation of:

An unnamed instance (denoted below ***participation***) of the ClassOfCompositionOfIndividual type with roles hasClassOfPart=**Role 1**, hasClassOfWhole=**Role 2** with

- attribute meta:annSource=**Source**

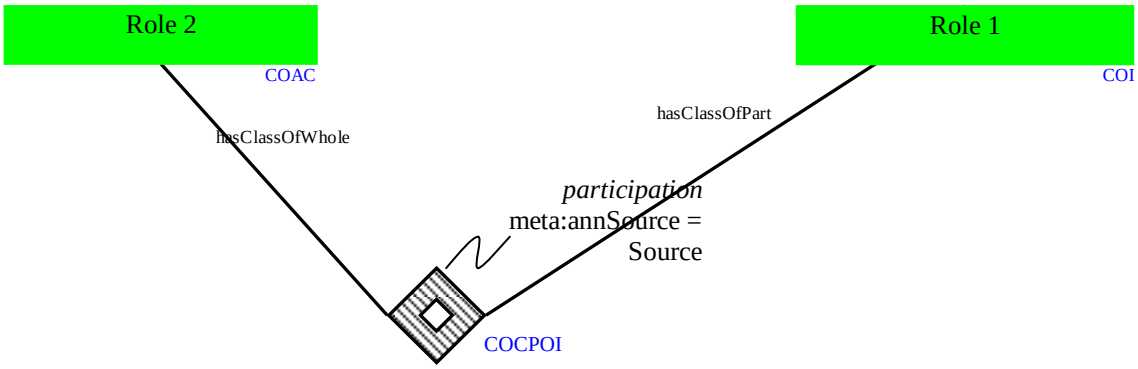


Diagram 4.6.2. Transformation of a «**participates in**» relationship row

4.6.3. «**is a predecessor in time of**» relationship

The transformation of such a row leads to the creation of:

An unnamed instance (denoted below ***sequence***) of the ClassOfTemporalSequence type with roles hasClassOfPredecessor= **Role 1**, hasClassOfSuccessor= **Role 2** with

- attribute meta:annSource=**Source**

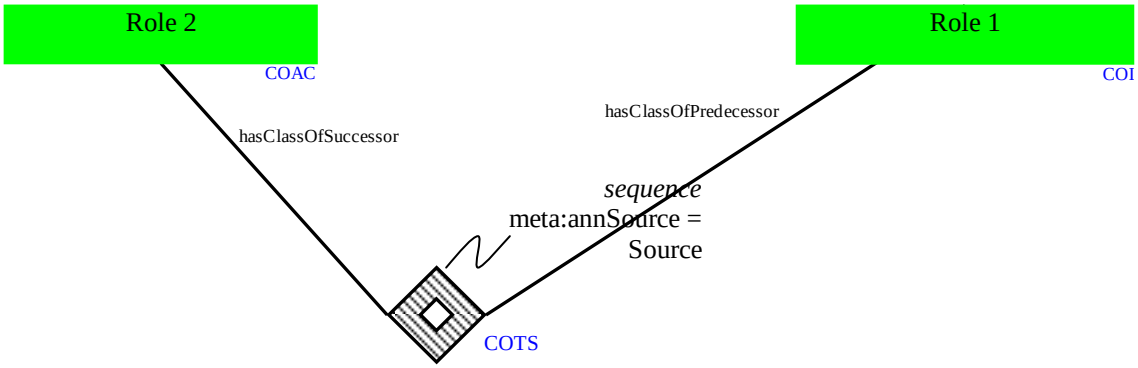


Diagram 6.3. Transformation of an «is **a predecessor in time of**» relationship row

4.6.4. «**has as part**» relationship

The transformation of such a row leads to the creation of:

An unnamed instance (denoted below ***composition***) of the ClassOfCompositionOfIndividual type with roles hasClassOfWhole=**Role 1,** hasClassOfPart=**Role 2**with

- attribute meta:annSource=**Source**

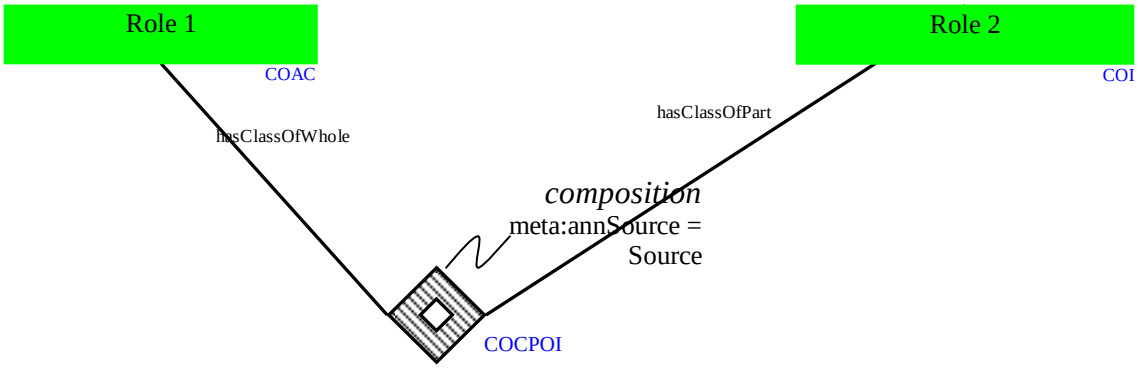


Diagram 6.4. Transformation of a «**has as part**» relationship row

4.6.5. «**is disjoint**» relationship

The transformation of such a row leads to the creation of:

An unnamed instance (denoted below ***set***) of the EnumeratedSetOfClass type with

- attribute meta:annSource=**Source**

An unnamed instance of the Classification type with roles hasClassified=**Role 1**, hasClassifier=***set*** with

- attribute meta:annSource=**Source**

An unnamed instance of the Classification type with roles hasClassified=**Role 2**, hasClassifier=***set*** with
- attribute meta:annSource=**Source**

A template instance p7tpl.IntersectionOfSetOfClass(***set***, ***EmptyClass***) with
- attribute meta:annSource=**Source**

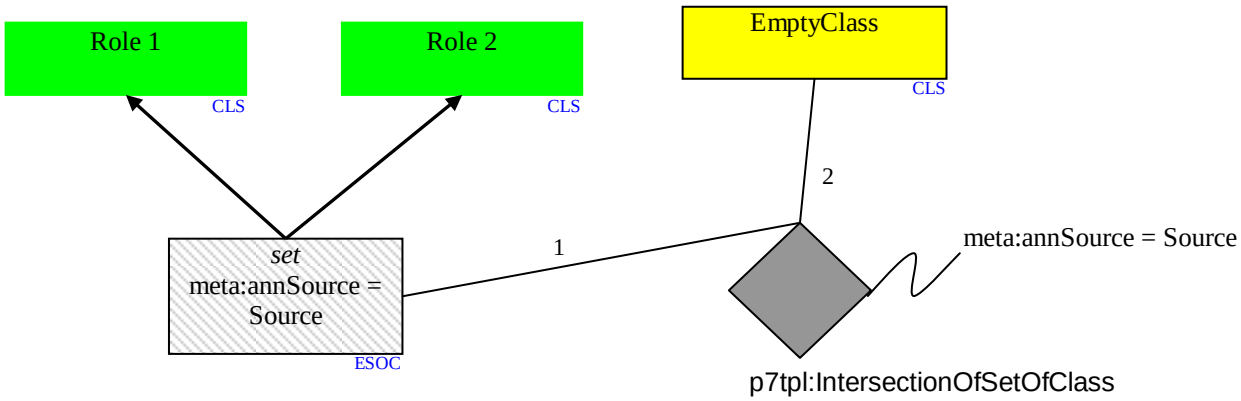


Diagram 6.5. Transformation of an «is **disjoint**» relationship row

4.6.6. «**is performed by**» relationship

The transformation of such a row leads to the creation of:

An unnamed instance (denoted below ***performance***) of the ClassOfCompositionOfIndividual type with roles hasClassOfWhole=**Role 1,** hasClassOfPart=**Role 2** with
- attribute meta:annSource=**Source**

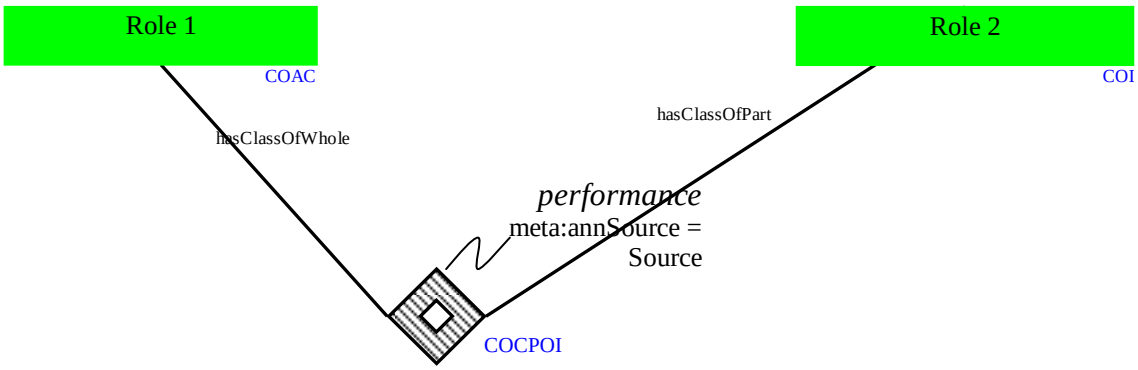


Diagram 6.6. Transformation of an «is **performed by**» relationship row

4.6.7. «**is related to**» relationship

The transformation of such a row leads to the creation of:

An unnamed instance (denoted below ***relation***) of the OtherRelationship type with roles hasEnd1=**Role 1,** hasEnd2=**Role 2** with
- attribute meta:annSource=**Source**

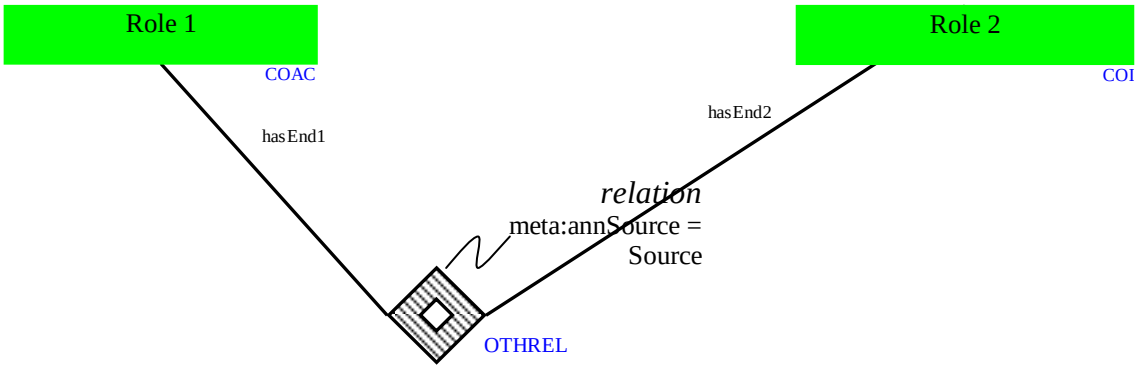


Diagram 6.7. Transformation of an «is **related to**» relationship row

4.7. Requirements description table transformation

Requirement description table is located on the **Requirements** tab of the model spreadsheet file. The header of the **Requirements** data description table:

Section	Statement Classification	Role 1	Role 1 Cardinality	Relates to	Role 2	Role 2 Cardinality	Source

Section	*optional*
Statement Classification	*mandatory*
Role 1	*mandatory*
Role 1 Cardinality	*optional*
Relates to	*mandatory*
Role 2	*mandatory*
Role 2 Cardinality	*optional*
Source	*mandatory*

Cardinality cells processing:

The text strings in the cells **Role 1 Cardinality** and **Role 2 Cardinality** have to be in the format ***(#:#)*** where # is a natural number (including 0) or * in a second position for infinity (cardinality ***(*:*)*** should be simply omitted).

For **Requirement description table** cardinalities are processed in exactly the same manner as for **Other relationships description table** (including processing or ignoring cardinalities depending on the **Relates to** cell content).

Statement Classification cells processing:

During a transformation of a **Requirements** data description table one relationship (or class of relationship) is singled out as "required" and classified by (or specialized from) a member of ***Requirement Status Class*** referred to in the **Statement Classification** cell.

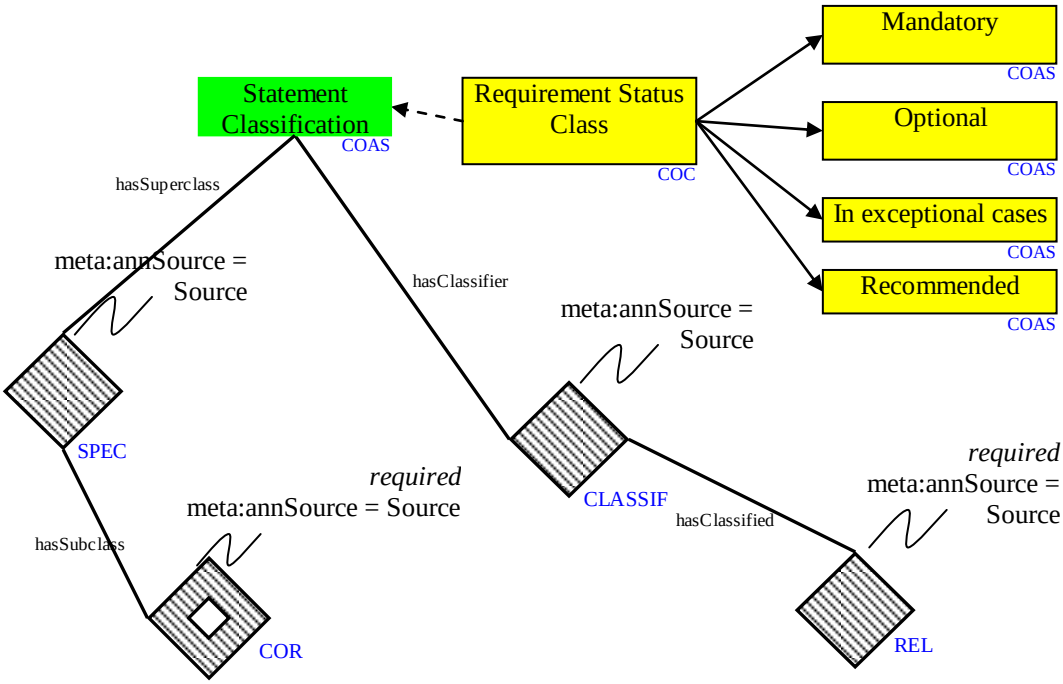


Diagram 7.0. General schema of "required" relationship or class of relationship classification

Classification of template instances in this manner is impossible, therefore template instances have limited usability for requirements representation.

Relates to cell processing:

The specific way of the row transformation is defined by the string in the **Relates to** cell of this row.

4.7.1. «**is classified as**» requirement

The transformation of such a row leads to the creation of:

An unnamed instance (denoted below ***required***) of the Classification type with roles hasClassified=**Role 1**, hasClassifier=**Role 2** with

- attribute meta:annSource=**Source**

An unnamed instance of the Classification type with roles hasClassified=***required***, hasClassifier= **Statement Classification** with

- attribute meta:annSource=**Source**

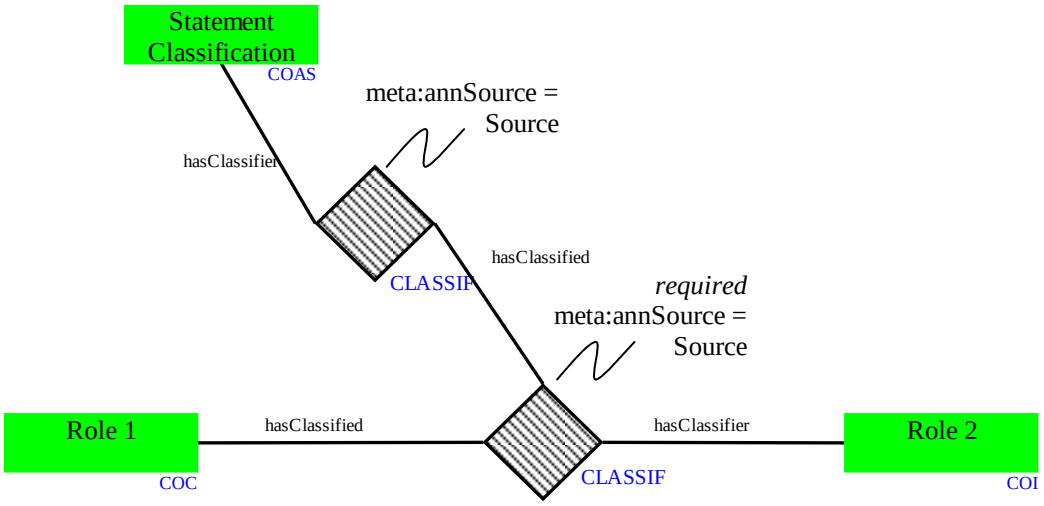


Diagram 7.1. Transformation of an «is **classified as**» requirement row

4.7.2. «**is subclass of**» requirement

The transformation of such a row leads to the creation of:

An unnamed instance (denoted below ***required***) of the Specialization type with roles hasSubclass=**Role 1**, hasSuperclass=**Role 2** with

- attribute meta:annSource=**Source**

An unnamed instance of the Classification type with roles hasClassified=***required***, hasClassifier= **Statement Classification** with

- attribute meta:annSource=**Source**

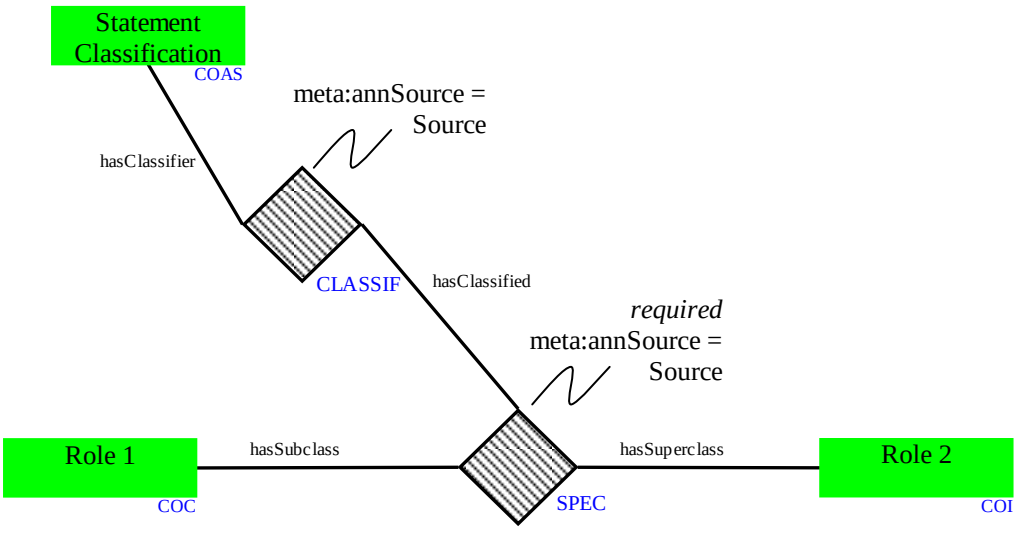


Diagram 7.2. Transformation of an «**is subclass of**» requirement row

4.7.3. «**complies to description in**» requirement

The transformation of such a row leads to the creation of:

An unnamed instance (denoted below ***document instance***) of the PossibleIndividual type with

- attribute meta:annSource=**Source**

An unnamed instance of the Classification type with roles hasClassified=***document instance***, hasClassifier=**Role 2**

An unnamed instance (denoted below ***required***) of the Description type with roles hasSign=***document instance***, hasRepresented=**Role 1** with

- attribute meta:annSource=**Source**

An unnamed instance of the Classification type with roles hasClassified=***required***, hasClassifier= **Statement Classification** with

- attribute meta:annSource=**Source**

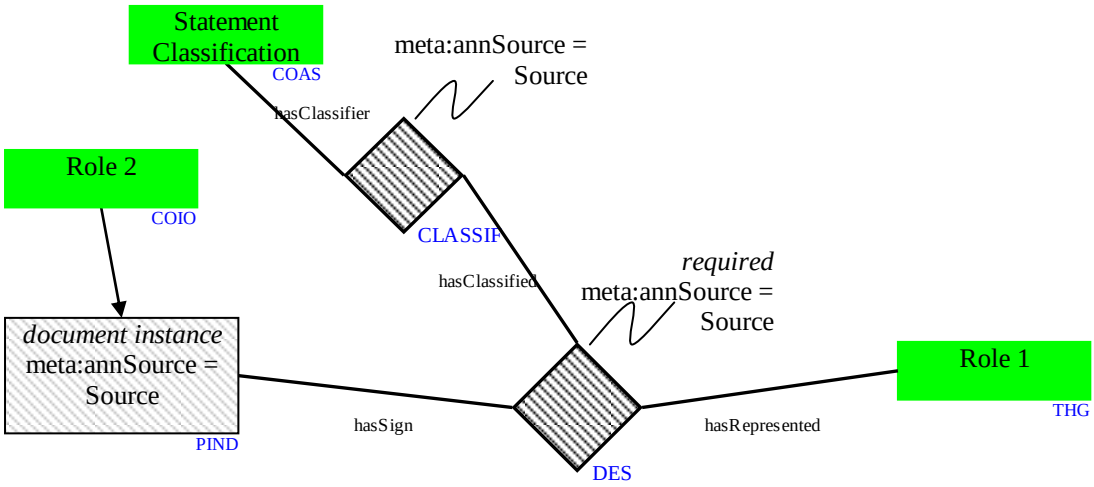


Diagram 7.3. Transformation of a «**complies to description in**» requirement row

4.7.4. «**participates in**» requirement

The transformation of such a row leads to the creation of:

An unnamed instance (denoted below ***required***) of the ClassOfCompositionOfIndividual type with roles hasClassOfPart=**Role 1**, hasClassOfWhole=**Role 2** with

- attribute meta:annSource=**Source**

An unnamed instance of the Specialization type with roles hasSubclass=***required***, hasSuperclass= **Statement Classification** with

- attribute meta:annSource=**Source**

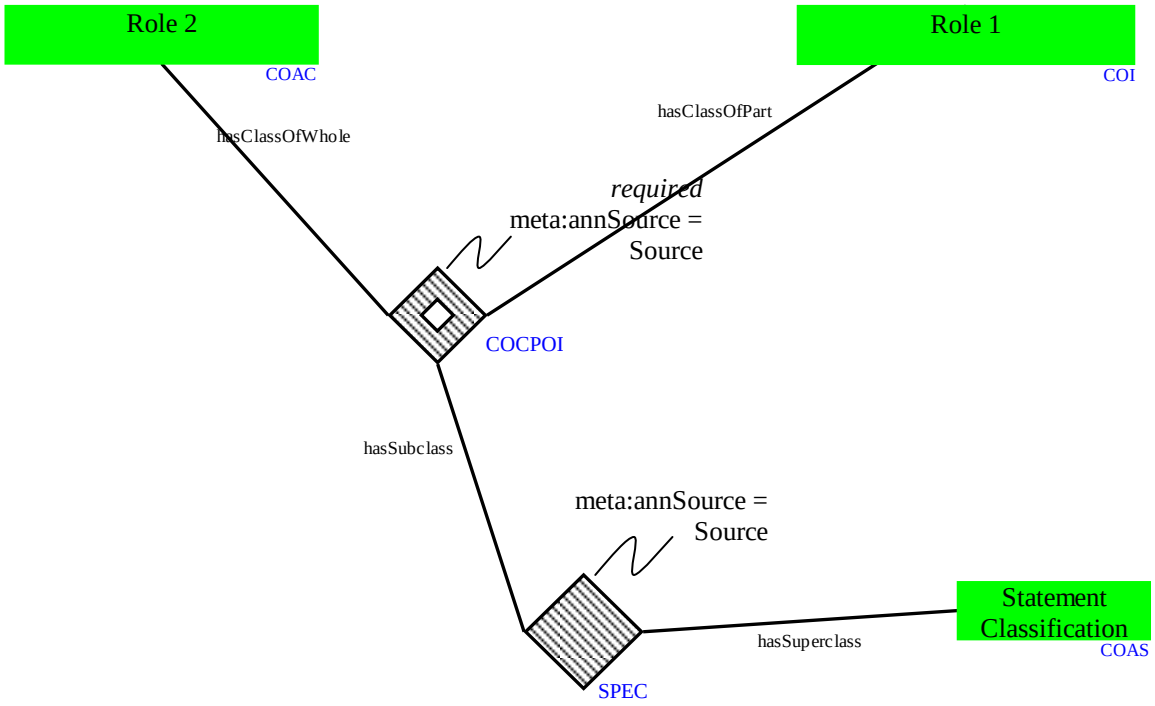


Diagram 7.4. Transformation of a «**participates in**» requirement row

4.7.5. «**is a predecessor in time of**» requirement

The transformation of such a row leads to the creation of:

An unnamed instance (denoted below ***required***) of the ClassOfTemporalSequence type with roles hasClassOfPredecessor= **Role 1**, hasClassOfSuccessor= **Role 2** with

- attribute meta:annSource=**Source**

An unnamed instance of the Specialization type with roles hasSubclass=***required***, hasSuperclass= **Statement Classification** with

- attribute meta:annSource=**Source**

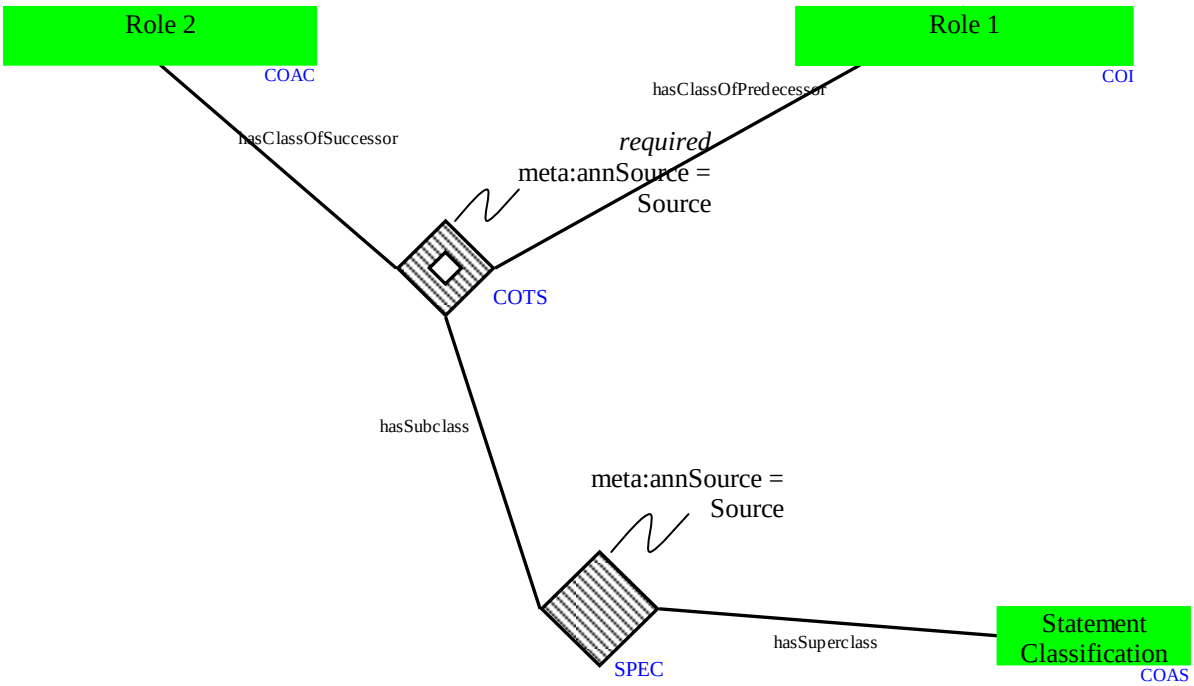


Diagram 7.5. Transformation of an «is **a predecessor in time of**» requirement row

4.7.6. «**has as part**» requirement

The transformation of such a row leads to the creation of:

An unnamed instance (denoted below ***required***) of the ClassOfCompositionOfIndividual type with roles hasClassOfWhole=**Role 1,** hasClassOfPart=**Role 2**with

- attribute meta:annSource=**Source**

An unnamed instance of the Specialization type with roles hasSubclass=***required***, hasSuperclass= **Statement Classification** with

- attribute meta:annSource=**Source**

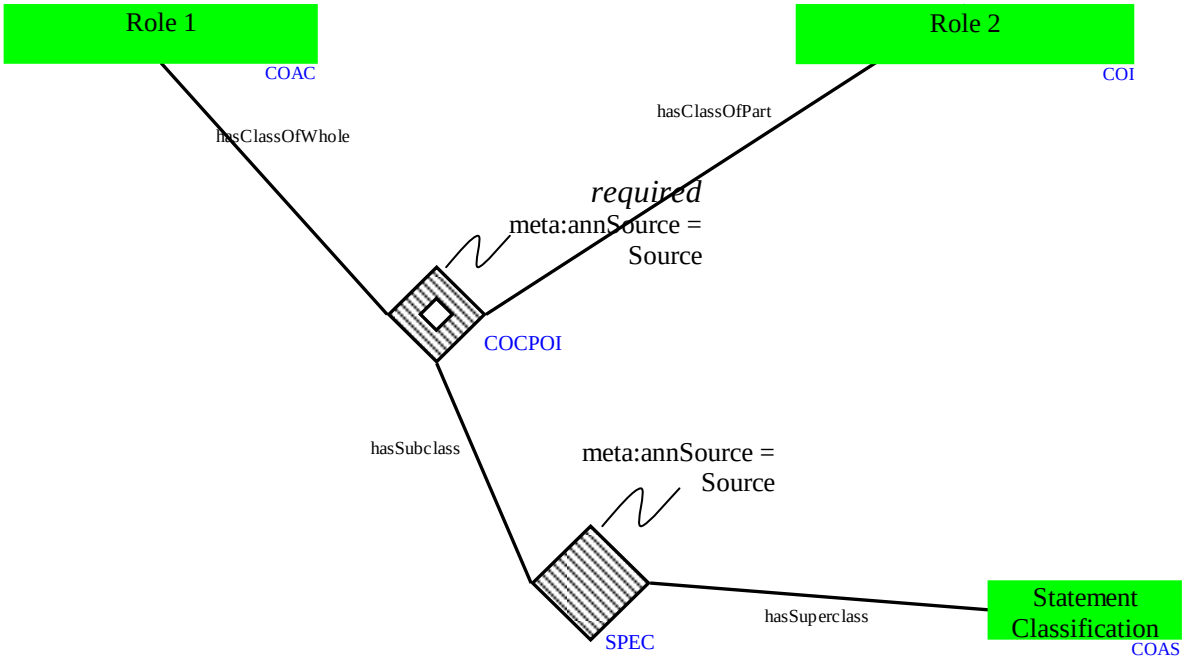


Diagram 7.6. Transformation of a «**has as part**» requirement row

4.7.7. «**is performed by**» requirement

The transformation of such a row leads to the creation of:

An unnamed instance (denoted below ***required***) of the ClassOfCompositionOfIndividual type with roles hasClassOfWhole=**Role 1,** hasClassOfPart=**Role 2** with

- attribute meta:annSource=**Source**

An unnamed instance of the Specialization type with roles hasSubclass=***required***, hasSuperclass= **Statement Classification** with

- attribute meta:annSource=**Source**

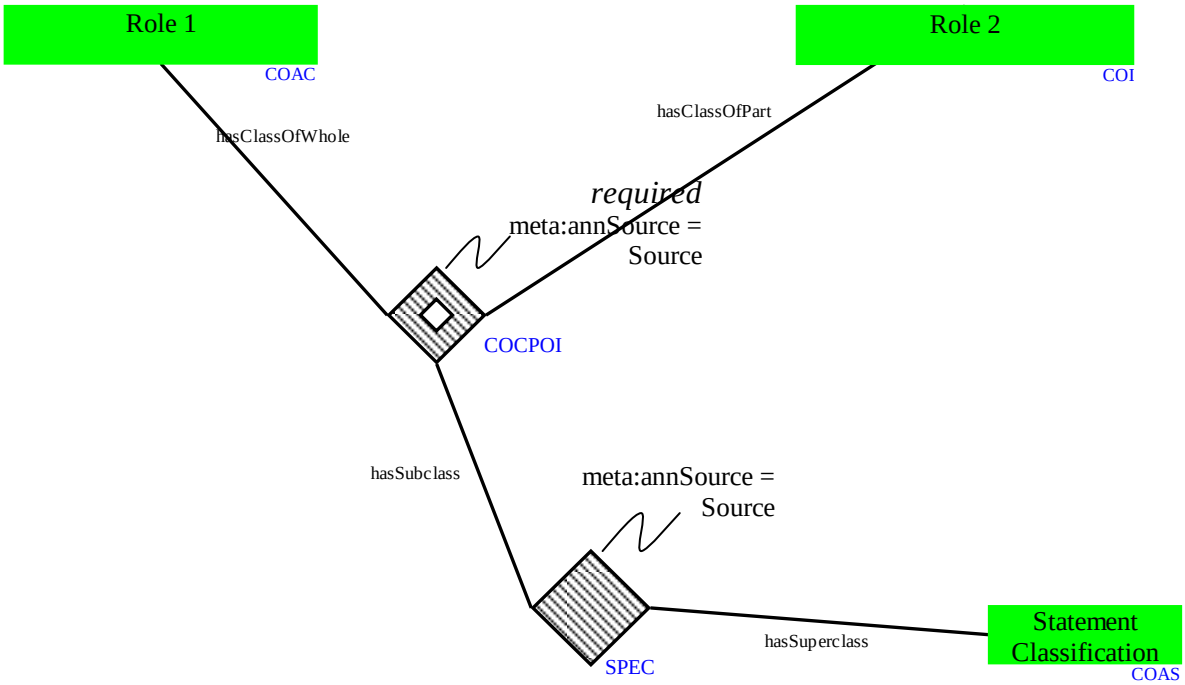


Diagram 7.7. Transformation of an «is **performed by**» requirement row

4.7.8. «**is related to**» requirement

The transformation of such a row leads to the creation of:

An unnamed instance (denoted below ***required***) of the OtherRelationship type with roles hasEnd1=**Role 1,** hasEnd2=**Role 2** with

- attribute meta:annSource=**Source**

An unnamed instance of the Specialization type with roles hasSubclass=***required***, hasSuperclass= **Statement Classification** with

- attribute meta:annSource=**Source**

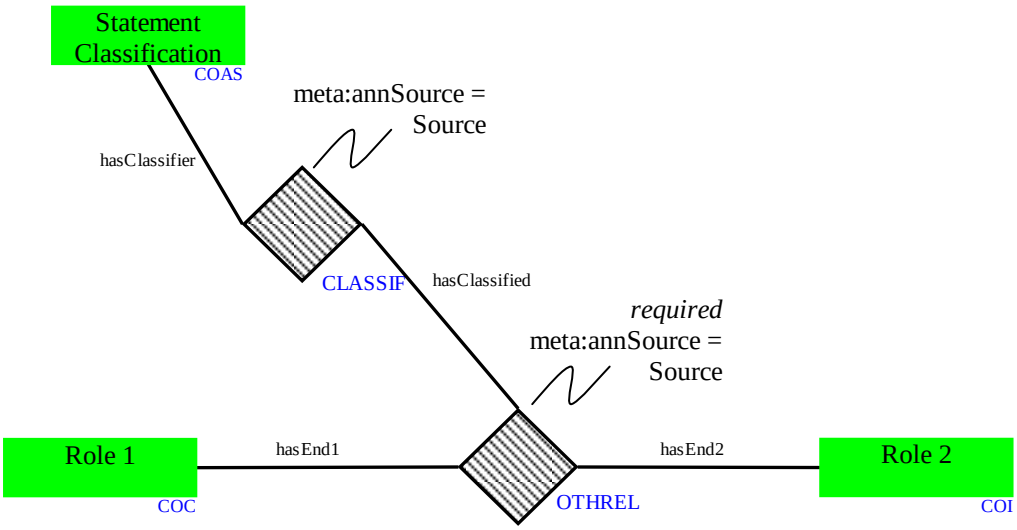


Diagram 7.8. Transformation of an «is **related to**» requirement row

Annex 1. Notes on mapping for .15926 Platform

Specific methodologies and languages for mapping between data model schemes are left out of the ISO 15926 standard scope and completely entrusted to implementers. For many years many different approaches to data mapping were developed in academic and business communities by ontologists and computer scientists working with different standards. Some common deficiencies can be noted for Common Logic and various First Order Language variations, XSLT and other restricted languages. Below are our reasons for choice of a general-purpose programming language (enhanced by domain specific language constructs) for this task.

From a logician's viewpoint every "big mapping" for two large and non-homogeneous data sources is effectively an algorithm bringing together many "small" mappings. "Small" mapping in turn can be a simple dictionary/table (like "kg"<->"KILOGRAMM", etc.) or a complex algorithm in itself.

From a software engineer's viewpoint the mapping involves various external system settings, program states, computer resources (memory, processing power, network latency and bandwidth), interactions with user of users. If mapping for an online façade is not completed in a reasonable and predictable time, it's useless. If business user sees some cryptic XSLT error messages instead of a human-readable report, the system can not gain much popularity.

Using specialized restricted languages it is possible to specify quite effectively atomic rules ("small" mappings); but there are problems in the encoding of conditions and circumstances for them ("big" mapping). These are often left in plain text format as gathered from specialists in legacy systems, without further formal and executable specification. This in turn leads to process fragmentation and manual execution of important tasks in the data conversion pipeline without an ability to do a throughput testing for all potential choices and forks. Practically the result of each data conversion has to be put through validity test anew.

In a modern programming language you can specify and test any type of condition, internal or external, from a single process. Validity and performance tests for time constraints, usability and business value can be done for one mapping or for a complete data pipeline.

Another source of process fragmentation for restricted languages is the need to pre-convert native data to elementary facts, flat data table structure, XML or other special format. And perform a conversion in the opposite direction after the mapping is completed. In a modern programming language it is possible to integrate seamlessly connections to legacy databases, reference data sources and other data providers. It is possible also to read the mapping rules themselves "on the fly" from an external data sources (dictionaries or tables, etc).

Finally, established general-purpose programming languages allow for highly effective code reuse, be it a user interfaces or specific ISO 15926 data mapping tasks. In Python (Ruby, C++, Scala, and others) it is possible to do a task, identify appropriate code patterns and metaphors, abstract them to libraries and further improve them.

Thus only by using general-purpose programming languages it is possible to build higher abstraction language layers and define domain-specific constructs, rising above an "assembler" level of table/XSLT/SPARQL. The goal is to allow business user to do very advanced tasks with data using only engineering discipline-specific or geometry-specific terms, patterns and metaphors (of ISO 15926, STEP or other high-level standard).

We at TechInvestLab.ru are using Python programming language augmented by domain-specific constructs with special libraries and syntax tricks.

Below is a sample mapping from spreadsheet table to ISO 15926 graph using specialized API of .15926 Platform Builder component (for full Builder documentation see .15926 Editor documentation):

```
class_id = r.known('Class Name', ids)
superclass = r.known_or_empty('Superclass Name', ids)
rel_class = r.known_or_empty('Classifier Relationship Class', ids)
class_class = r.known_or_empty('Class of Classifier Classes', ids)
source = r.nonempty('Source')
if superclass:
   rel_id = part2.Specialization(hasSubclass=class_id, hasSuperclass=superclass, annSource=source)
   if rel_class:
       part2.Classification(hasClassified=rel_id, hasClassifier=rel_class, annSource=source)
if class_class:
   part2.Classification(hasClassified=class_id, hasClassifier=class_class, annSource=source)
```

Two samples below use API calls of another Platform component – Scanner (this API is realizing a domain-specific semantic query language **SearchLan.15926**, full description is included in .15926 Editor documentation):

```
entities = find(type = part2.Scale, label = icontains("celsius"))
```

or

```
outfile = file('scales_celsius.txt', 'w')
scales=find(type=part2.Scale, label=icontains('celsius'))
for entity in scales:
   designations=find(id=entity, hasDesignation=out)
   for property in designations:
      outfile.write(entity+' : '+property+'\n')
outfile.close()
```

Such queries can be used to build a mapping *from* ISO 15926 graph.

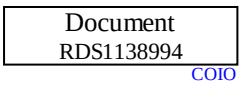
Using these domain-specific APIs bidirectional mapping between ISO 15926 data model (Part 2 or template level) and high level pattern representation of data can be built (for sample implementation see .15926 Editor distribution).

Annex 2. Extensions of the Part 7 diagram language

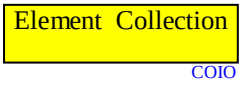
In this Specification ISO 15926 Part 7 instance diagrams are used for illustration of ISO 16926 data models with the following additional notation elements and conventions:

1. ISO 15926 entity types are shortened according to the list of acronyms https://www.posccaesar.org/wiki/SigMmt/Templates/Acronyms .

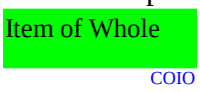
2. White rectangles are elements of the Methodology reference data schema from the external reference data libraries



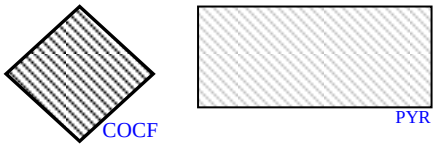
3. Yellow rectangles are elements of the Methodology reference data schema from TechInvestLab.ru RDL sandbox.



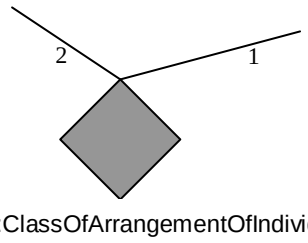
4. Green rectangles are elements corresponding to the column names of the data description tables.



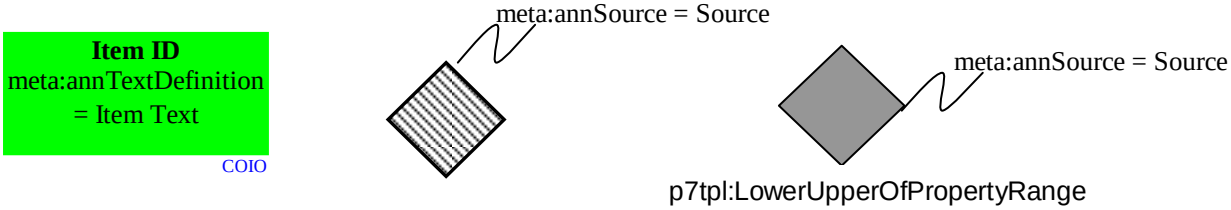
5. Hatched elements are unnamed elements created during the data description table transformation.



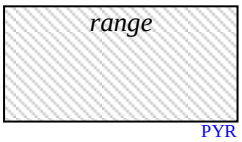
6. Grey diamonds signed by template names are template instances with roles shown as connection lines to role fillers with role numbers indicated.



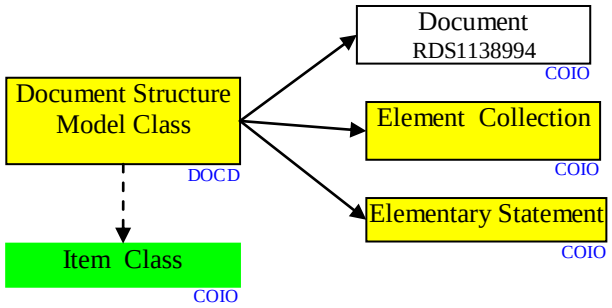
6. Additional attributes assigned to items of the data model are written with the namespace used and the attribute names, and are placed inside the rectangles for classes and individuals and attached as notes for template instances, relationships and classes of relationship.



7. Names written in *italic* are dummy names given to unnamed elements to reference them in the Specification.



8. Dashed arrows are classifications restrictions, signifying that a class in a model is selected from members of a class of class defined in the Methodology reference data schema.



Annex 3. Template DescriptionByInformationObject

Name: **DescriptionByInformationObject**

Description: DescriptionByInformationObject(x1, x2) means that x1 is a [thing] and x2 is a [class_of_information_object] and x1 is described by x2.

Reason for creation: Entities belonging to types Description or ClassOfDescription have specific role restrictions in the Part 2 data model. Therefore it is impossible to use them for description by an entity belonging to ClassOfInformationObject type. Desired semantics can be achieved by stating that there exists a member of ClassOfInformationObject entity in question, which is an individual and is connected to the described entity by an instance of Description entity type.

Signature:

Order	Role name	Role type
1	hasRepresented	Thing
2	hasPattern	ClassOfInformationObject

FOL axiom:

DescriptionByInformationObject(x1, x2) ↔
Thing (x1)∧
ClassOfInformationObject (x2)∧
∃ y（PossibleIndividual(y)∧
ClassificationOfIndividual(y,x2)∧
Description(x1,y))

Diagram :