.15926 Editor

Version 1.5beta

(incomplete documentation)

Volume 4

Patterns and Mapping

February 23, 2013

Volume 4. Patterns and Mapping

Contents

Other documentation volumes:

Volume 1. Getting Started
Volume 2. APIs: Scanner and Builder
Volume 3. Extensions

License

Parts of .15926 Editor (built-in extensions and extension samples) are released as a source code under the BSD 2-Clause license.

Other parts of the software (released in binary and in text form) are not covered by the license above and are distributed "as is" and free of charge for evaluation purposes only!

Elephant icon by Martin Berube is used for .15926 software according to terms at http://www.iconarchive.com/show/animal-icons-by-martin-berube/elephant-icon.html

12. Patterns

12.1. Approach to patterns

It is well known that ISO 15926 (and Semantic Web technologies in general) allow for many ways to express the same ontological relationship between things. Part 2 type instances, base or specialised templates of unrestricted complexity, OWL relationships or custom RDF properties – all these constructs can be used in data modelling. Several implementations of ISO 15926 exist already and more are planned.

Part 2 relational instances, corresponding base templates, core, specialized and any number of custom templates can be all used to express the same fact (sometimes among other facts) – that class A is specialization of B, or individual X is part of individual Y, for example. On the other hand, the same concept of "connectedness" or "identification" can be applied to quite diverse things –to classes of classes, to classes of individuals, or to individuals – physical objects, events, activities.

Complex multi-role relationship can be expressed as custom template signature, later it can be expanded (lifted) to expose entities composing template axiom, to base template level, and finally it can be fully expanded to a complex network of Part 2 type instances. The process of template 'contraction' can go in other direction. Data graphs in this sequence can be also considered as representations of the same pattern.

Today slightly different concepts of "pattern" are developed by several ISO 15926 communities to utilize an idea of recurring patterns in various domains and of multiple methods to do the same modelling. Patterns are the cornerstone of PCA modelling methodology (https://www.posccaesar.org/wiki/FiatechJord), patterns are defined for use in mapping in ISO 15926 Information Patterns (IIP) project by members of iRING User Group (http://iringug.org/wiki/index.php?title=ISO_15926_Information_Patterns_%28IIP%29).

Pattern representation technology and tools developed in .15926 project will be able to support patterns in all possible areas: template expansion/contraction, pattern modelling, pattern mapping, etc.

For .15929 Platform pattern is a formally defined list of alternative ways (representations) to express particular ontological relation between two or more entities.

> For example, concept of connection between two things can be expressed for various types of things by usage of six Part 2 type instances (ConnectionOfIndividual, DirectConnection, ClassOfDirectConnection, ClassOfIndirectConnection, ClassOfConnectionOfIndividual, IndirectConnection), six corresponding templates from Part 8 initial template set, or through several templates currently discussed by the Special Interest Group (SIG) Modeling, Methods and Technology (MMT) of POSC Caesar Association (ClassOfDirectConnectionDefinition, etc.), available at http://15926.org). One single pattern "connected to" is based on this list. Two things are considered to form this pattern if they occupy corresponding roles in any of relational entities listed above.

Pattern recognition in data sources is a powerful mechanism of .15926 Platform. Search for patterns in SearchLan.15926 (documented in **Volume 2**) is already supported in the Editor. From version 1.4 the Editor supports mapping to patterns from Excel spreadsheet. Other pattern-based instruments will be developed later, including template expansion for data set transformations.

12.2. Patterns in the Editor

Standardization of ISO 15926 pattern format and mapping rule description language are expected in the future. Today .15926 Editor keeps patterns libraries as open JSON files.

Released version of .15926 Editor has an initial set of predefined patterns. More pattern libraries are released by TechInvestLab.ru with some etensions.

Initial libraries for the Editor are located in files with ***.patt*** extension, placed in the *<installation_folder>/patterns* folder. You can add new ***.patt*** files to this folder, modify existing patterns definitions, or add ***.patt*** files to your project from other locations on your computer..

Always use new files to record new patterns and rename initial libraries files if you are changing definitions in it. An installer may overwrite it while upgrading the software!

Template definitions used in patterns should be present in the project to facilitate pattern search and mapping. There template definition data sources in the project should be assigned predefined specified module names, as documented in the released libraries.

Initial pattern sets depend on three template data sources:

- Part 8 initial set in a module named ***p7tpl*** (the file ***p7tpl.owl*** accompanying ISO 15926-8 is included with the distribution in folder <*samples*>);
- PCA MMT SIG set in a module named ***mmttpl*** (the work-in-progress version of PCA MMT SIG available for download from http://posccaesar.org/sandbox/p8iwg/);
- IIP template set in a module named ***iiptpl*** available as **IIP Sandbox (templates)** from *Files* menu.

An absence of a registered module or an absence of particular template definition in a module will lead to the inability to recognize presence of some patterns for visualization, querying or mapping.

12.3. Patterns definition

More documentation to follow

12.4. Examples

More documentation to follow

12.4.4. IIP pattern modelling

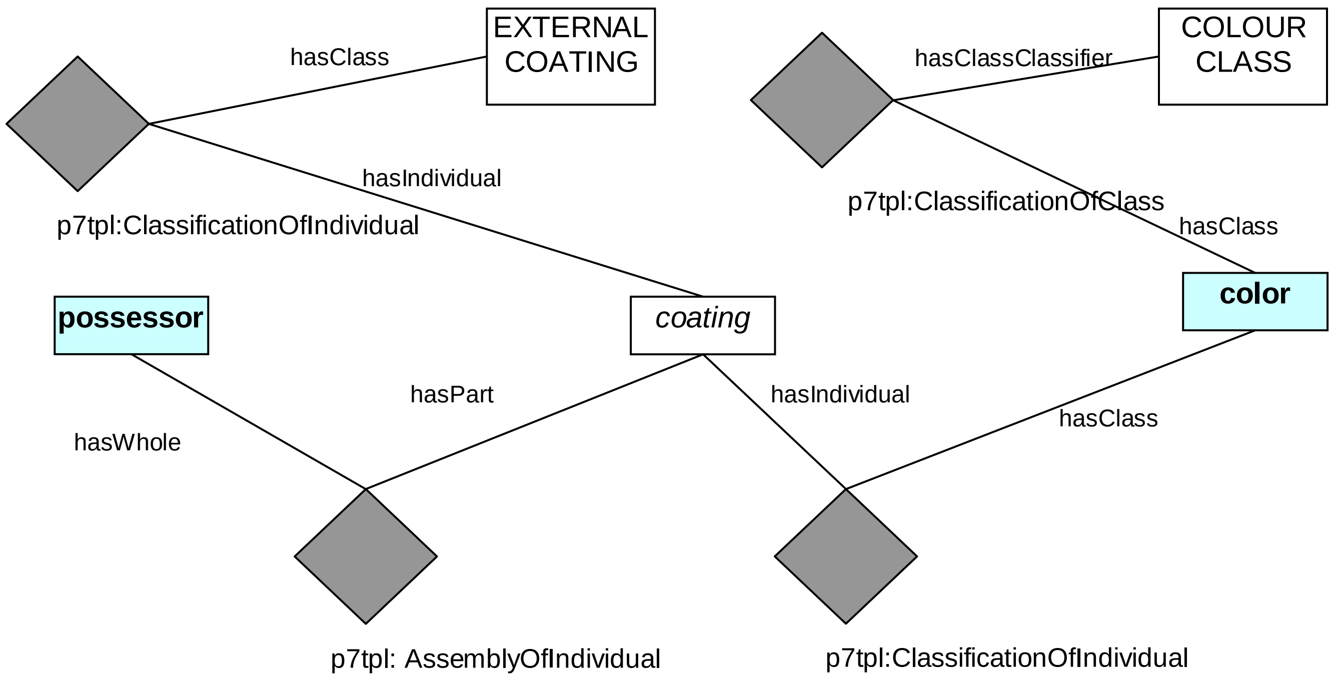
Combining methods of pattern construction described above, let's model a pattern representing one of the patterns proposed in IIP project of iRING User Group (http://iringug.org/wiki/index.php?title=ISO_15926_Information_Patterns_%28IIP%29). Look at the Coating Color pattern described as:

P0003	0	Coating Color	The color name for the coating	EXTERNAL COATING	AssemblyOfIndividual
	1			COLOUR CLASS	ClassificationOfIndividual

We can use this pattern to identify relationships in ISO 15926 dataset for import into some engineering database which has a color list for objects. Or we can use it to create ISO 15926 data set as export from such database. Depending on the goal we'll use pattern search functionality of Scanner.15926 (available already), or pattern writing functionality of Builder.15926 (see spreadsheet adaptor description below).

To identify (or create) this pattern for an entity-in-focus *possessor* (individual), we've to find (or create) another individual (*coating*) classified by EXTERNAL COATING class and connected to our entity-in-focus by an instance of AssemblyOfIndividual template, and then look for (or create) classification of *coating* individual by a class which is a member of COLOUR CLASS.

Selection of templates for classification has to be agreed for pattern modeling, in this example we'll use base templates only, specialized to the most specific entity types available (thus ClassificationOfClass and ClassificationOfIndividual, not Classification).



We'll have two roles for our pattern: **possessor** and **color**. Other parts of a pattern will be representing two classifier classes and four template instances connecting them in a pattern.

If you look at other parameters of Coating Color pattern you will notice that it is applicable to restricted list of several commodity types: EQUIPMENT, INSTRUMENT, PIPING NETWORK SYSTEM, PIPING NETWORK SEGMENT, PIPING SPECIALTY COMPONENT, PIPE SPOOL and VALVE. These are restrictions on the type of **possessor** role, and are given as a list of restricting classes. Some of the restricting classes exist in the PCA RDL, others can be found only in the iRING Sandbox, this a mix of URIs in two different namespaces.

More documentation to follow

13. Spreadsheet mapper

Spreadsheet mapper is a powerful adapter designed for the transformation of Excel spreadsheets to ISO 15926 compliant RDF.

To get an ISO 15926 representation of a spreadsheet we need a data model of the spreadsheet. This data model should be is described by the pattern in one of the pattern libraries of the Editor, using reference data and templates from some RDL(s). The mapping is defined between pattern roles and spreadsheet columns. Then the adapter imports spreadsheet data into the local data source in RDF format.

Spreadsheet mapper is included into the .15926 Editor as an open source extension released under the **BSD 2-Clause License**. **Other parts of the software (released in binary and in text form) are not covered by this license.**

13.1. Mapping basics

The adapter works only with Microsoft Excel installed and running on the computer, using Excel VBA access. It is tested with Microsoft Excel starting from 2002 version.

The adapter can work with files in*.xls* and *.xlsx* formats only. The adapter uses many specific features of these formats, storing mapping data both in open and hidden data fields and attributes of the spreadsheet. Saving spreadsheet in other formats will most probable destroy the mapping data and will require a new mapping process.

Spreadsheet adapter allows mapping of only one pattern for a worksheet at a time! You can save mappings to files and load them from files to make several imports from one worksheet sequentially with different patterns, but only the last mapping will be stored in the worksheet itself.

The adapter is called via *Import – Build patterns from Excel* menu command. Make sure that spreadsheet for data import is open in Excel before you call the adapter! The spreadsheet should have all the needed worksheets present and named and all necessary column names should be already defined in Row 1 of each worksheet. Column names should start from a letter or a numbers, avoid use of other symbols as mapper will add more columns to your spreadsheet with names starting with "!" and "$".

If you find out that you need to insert a worksheet, rename it or insert more columns for your data – you have to close the adapter panel, make necessary changes and call adapter again, in some cases you'll be obliged to redefine the mapping after that.

You can add or change data in the spreadsheet, change mapping and repeat import, search data source and browse other data sources in the Editor – all without closing the adapter panel.

Previously imported data may be edited or changed when the import is repeated, but never removed. Changes to the data source incurred by the spreadsheet import can not be undone via Editor's *Undo*.

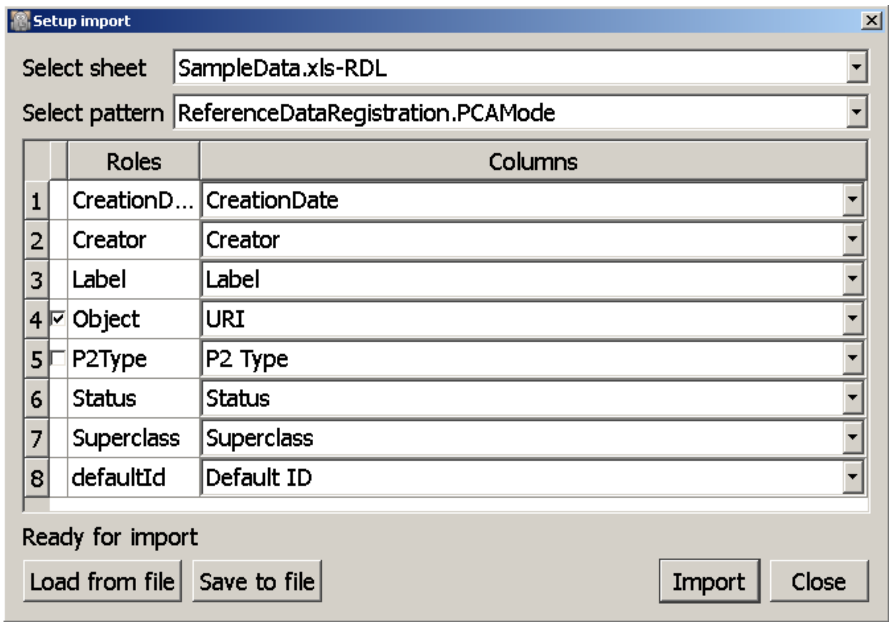
The mapping and some overhead information (classifiers, generated URIs for pattern Scolem variables, etc.) are stored in the open and in the hidden data fields and attributes of the spreadsheet. Do not change any fields except the ones you've designed for information entry, and the adapter will open your worksheets again preserving all previously defined mappings.

13.2. Mapping and import

Select the worksheet and the pattern in the mapping panel to start the mapping. You can select any worksheet of any file open on your computer at the time adapter is called. You can select any pattern option of any pattern in pattern libraries present in the Editor. Patterns are listed alphabetically with option names (unnamed options are numbered sequentially). Always name pattern options designed for mapping!

Once worksheet and pattern are selected, you will be presented with a list of pattern roles (left column) and will be able to choose corresponding worksheet column names (right column) to map them.

Data transformation during the import depends in the mapping, on the content of the imported spreadsheet and on the data contained in the project data sources. Please pay attention to the rules describing data changes for incremental entry of data and sequential imports of the same spreadsheet!



Select sheet: SampleData.xls-RDL

Select pattern: ReferenceDataRegistration.PCAMode

		Roles	Columns
1		CreationD...	CreationDate
2		Creator	Creator
3		Label	Label
4	✓	Object	URI
5	☐	P2Type	P2 Type
6		Status	Status
7		Superclass	Superclass
8		defaultId	Default ID

Ready for import

Mapping and import rules:

1. You can leave a pattern role unmapped by leaving column selection field empty. Unmapped roles will be ignored. If unmapped role corresponds to some pattern part or to the role of some pattern part – corresponding entity will be omitted, and entities linked to it may be omitted also.

2. Sequential imports can not delete any previously imported entity, property or relationship in your data source, whatever changes are made to the spreadsheet. You can edit any imported property or relationship; edited information will replace the old one.

3. All pattern roles defined with **'self'** value in corresponding pattern parts represent entities which potentially may be created by the adapter. Such roles have a checkbox before their names.

3.1. Mark this checkbox if you want to create new data entities for this part of a pattern. The cells in the column mapped to the marked pattern role may be left empty, may contain special value ***new*** or contain text which will be treated as an URI.

- If the cell is empty – corresponding entity will be not be created for this row. If according to the pattern structure it should occupy a mandatory role in some relational entity of a known type (Part 2 type instance or a template instance) – this one will be skipped in turn, and entities linked to it may be skipped also.

- If the cell contains special value ***new*** – new entity will be created in the data source. Its URI will generated in the *Namespace for new entities* of the data source according to the rules described in **Volume 1. Getting started**.

- If the cell contains any other text – it will be interpreted as an URI.

  If an entity with this URI already exists in the data source – data for it will be added. All changed properties and relationships will replace the old ones, but there is no way to delete properties or relationships by changing the spreadsheet.

  If there is no entity with this URI in the data source – it will be created with this URI. The adapter will not check whether URI is well-formed or not!

3.2. If checkbox at the role name is not marked, the adapter will not create new entity for the corresponding part of the pattern. If this part of the pattern is restricted by a **'type'** key – the adapter will attempt to build a full classifier for this restriction by searching for all occurrences of **Classification** pattern in **all** project data sources. The set of classes identified in this process will be exported to the spreadsheet and built into the drop-down list for all cells in the column.

If such classifier is present as a part in the pattern, try to import an empty worksheet immediately after you've defined the mapping. It will create drop-down menus for all cells of the corresponding columns which will simplify further manual data entry.

4. The cells in all other mapped spreadsheet columns may be left empty, or may contain strings, including labels of some reference data entity or URIs.

4.1. If the column is mapped to the pattern role which represents annotation property:

- If the cell is empty – the property will be omitted. If such property for the entity already exists – it will be left intact.

- If the cell contains any string – the property will be created with the specified value.

4.2. If the column is mapped to the pattern role which represents relational entity role or other object property, values of its cells will be interpreted as references to entities in the project. The adapter will create relational entity of a known type (Part 2 type instance or a template instance) only if all its mandatory roles are successfully filled by the entities imported from spreadsheet according to the mapping. The adapter will always create any other object property with specified value.

To find a referenced entity for the role (property) the adapter will:

- Attempt to search for an entity with exactly the same ***rdfs:label*** property. If search brings such an entity – it will be placed in the referred role.

- If search brings nothing – adapter will use a value in the cell as an URI of an entity without checking whether it is well-formed or not.

5. Entities corresponding to the parts of the pattern which represent Scolem variables (parts which are not present in the signature) will be always created at import, except for the following cases:

- When they are unambiguously identified by **'uri'** key with a single value. Such entities will be used to populate relations to which they are assigned, but entities themselves will not be created in the data source.

- When one or more of mandatory roles of a relational entity (Part 2 type instance or a template instance) can not be filled after iterative attempts to do data transformation for the worksheet.

6. For partially filled worksheet the adapter will iterate data transformation process attempting to create as many entities as possible.

7. The adapter is creating parts of the pattern in the order they are listed in the pattern definition dictionary. It is advisable to arrange pattern parts in the definition dictionary in such a way that an entity definition part (the part with **'self'** value for pattern role or Scolem variable key) appears before this particular pattern role or Scolem variable is used to reference a property key in another part representing relational entity.

8. The adapter is writing generated URI to the worksheet (overwriting **new** strings) at the moment this URI is generated. It is possible to reference newly generated URI in some other cell and use it in the same import (see an example below).

13.3. Saving and restoring the mapping

Although the adapter will store your mapping in the spreadsheet itself, the way is provided to save the mapping to a file and restore it from a file. It allows to do several imports with different patterns from the same worksheet as described above. This feature is also useful if you are designing spreadsheet form to be filled by other parties and returned to you for processing, and you can not be sure it will not be reformatted or otherwise corrupted in the process.

To save the mapping to a file press *Save to file* button on the adapter panel. The mapping is saved to a JSON file. The mapper will ask you for the file name and location.

To restore the mapping from a file select the worksheet for which the mapping was saved and press *Load from file* button on the adapter panel. The mapper will ask you to locate the JSON file and will restore the mapping.

Remember that you have to save a separate mapping for each worksheet kind!

13.4. Mapping example

Please refer to the **Sample Mapping and Adapter Prototyping Walk-through Guide** (file **dot15926AdapterSample.pdf**) in the <*documentation*> folder.