OpenBudgets.eu: Fighting Corruption with Fiscal-Transparency

Project Number: 645833 Start Date of Project: 01.05.2015 Duration: 30 months

Deliverable D1.7

Extraction and transformation of relevant code lists

Dissemination Level	Public
Due Date of Deliverable	Month 7, 30.11.2015
Actual Submission Date	30.11.2015
Work Package	WP 1, Data Structure Definition for Budgets and Public Spending
Task	T1.2 Definition of code lists
Type	Demonstrator
Approval Status	Final
Version	1.0
Number of Pages	28
Filename	D1.7 Extraction and transformation of relevant code lists.docx

Abstract: This document presents the methodology followed in order to convert code lists to a format acceptable according to the OpenBudgets.eu requirements. Two different tools, OpenRefine and UnifiedViews, are used to support the conversion process, which is described.

Project funded by the European Union's Horizon 2020 Research and Innovation Programme (2014 – 2020)

History

Version	Date	Reason	Revised by
0.1	23.10.2015	First Version – initial contributions by OKFGR	Lazaros Ioannidis
0.2	26.10.2015	Requirements, Code list selection, Transformation Process	Charalampos Bratsas, Lazaros Ioannidis, Panagiotis-Marios Philippides
0.3	27.10.2015	Example Open Refine	Panagiotis-Marios Philippides, Charalampos Bratsas
0.4	28.10.2015	Pipeline Definition Unified Views	Sotirios Karampatakis
0.5	19.11.2015	Review	Jindřich Mynarz
1.0	17.11.2015	Review Corrections	Lazaros Ioannidis

Author List

Organisation	Name	Contact Information
OKFGR	Sotirios Karampatakis	s.karampatakis@gmail.com
OKFGR	Lazaros Ioannidis	larjohn@gmail.com
OKFGR	Panagiotis-Marios Philippides	filippidis.okfgr@gmail.com
OKFGR	Charalampos Bratsas	Charalampos.bratsas@okfn.org
UEP	Jindřich Mynarz	jindrich.mynarz@vse.cz

Executive Summary

Fiscal datasets may contain values that can be classified into list of codes. Such code lists may represent budget status, budget function, involved state agencies, geographic information and others. Many countries have established their own code lists, while, at the same time, international organizations propose more generic lists.

Code lists are useful in order to constrain the described domain. Their significant value in the context of OpenBudgets.eu appears mainly when the code list properties of datasets can be used as hierarchical data cube dimensions. Furthermore, an organized and linked representation of code lists can be used to compare trends between multiple datasets, for instance coming from different countries.

When code lists are inherent to the containing dataset, they need to be extracted, by following the appropriate process. In any way, a flat list of codes can then be further transformed into a richer representation and later be linked to other, similar code lists. The last step would also include the transformation of the original dataset, in order to replace code list terms with reference to their equivalent in the enriched representation.

The representation selected for this deliverable is SKOS, a standardized specification for organizing knowledge into vocabularies. In this document we present two software tools, OpenRefine and UnifiedViews for processing code lists coming into various formats. OpenRefine's strength is its rapid approach to the targeted RDF output through a graphical interface. On the other hand, UnifiedViews provides more advanced options and supports automated execution of the extraction and conversion pipelines.

A generic approach of designing an extraction and conversion pipeline is presented in this document. Sample pipelines were designed and executed for a selection of code lists identified in the previous work of WP1. The pipeline design required for each code list is highly dependent on the input format and the desired output. The resulting SKOS representations indicate that UnifiedViews can be considered feature-complete for the code list extraction and transformation requirements of OpenBudgets.eu.

Abbreviations and Acronyms

LOD	Linked Open Data
SKOS	Simple Knowledge Organization System
RDF	Resource Description Framework

Table of Contents

LIST OF FIGURES **6**
LIST OF TABLES **6**
1. INTRODUCTION **7**
2. REQUIREMENTS **9**
1. SKOS REPRESENTATION 10
3. CODE LISTS SELECTION **11**
4. TRANSFORMATION PROCESS DESIGN **12**
1. EXAMPLE USING OPENREFINE: GREEK BUDGET REVENUES CODES (KAE ESODON) 12
5. PIPELINE DEFINITION IN UNIFIED VIEWS **16**
1. SKOSIFYING 16
6. CONCLUSIONS **20**
7. APPENDIX: EXTRACTED CODE LISTS **20**
1. STATISTICAL CLASSIFICATION OF ECONOMIC ACTIVITIES IN THE EUROPEAN COMMUNITY 20
2. CLASSIFICATION OF PRODUCTS BY ACTIVITY 21
3. GEOGRAPHICAL STANDARD CODE LIST 21
4. TRANSACTIONS IN FINANCIAL ASSETS AND LIABILITIES 22
5. CLASSIFICATION OF BALANCING ITEMS AND NET WORTH 23
6. DISTRIBUTIVE TRANSACTIONS 23
7. FINANCIAL ASSETS 24
8. NOMENCLATURE OF THE CLASSIFICATION OF SECTORS 24
9. NON-FINANCIAL ASSETS 25
10. OTHER CHANGES IN ASSETS 25
11. TRANSACTIONS IN PRODUCTS 26
12. ISO 4217 WORLD CURRENCY 26
13. ORGANIZATION IDENTIFIER 27
14. MULTIANNUAL FINANCIAL FRAMEWORK/POLITICAL CATEGORIES 28

List of Figures

Figure 1: Code Lists extraction and transformation process 9
Figure 2: Greek Budget Revenues Codes in XLS format 13
Figure 3: Greek Budget Revenues Codes in Open Refine 14
Figure 4: Greek Budget Revenues Codes in an appropriate structure to SKOSify 14
Figure 5: SKOS properties and relationships of the Code 100 of the Greek Budgets Revenues Codes 15
Figure 6: SKOSifying the Greek Budget Revenues Codes in Open Refine 15
Figure 7: RDF format of the Greek Budget Revenues Codes 16
Figure 8: CPC pipeline 17
Figure 9: CPC sample lines 18
Figure 10: Basic configuration for data extraction of Tabular DPU 18
Figure 11: Basic mappings 19
Figure 12: Hierarchy based on code 19
Figure 13: Proper relations mapping 19

List of Tables

Table 1: Sample countries list 8
Table 2: Code lists extraction and transformation requirements 9

1. Introduction

Budget datasets of the European Union countries include fields that refer to economic concepts. Some of these fields have a specific range of values and each budget line can have one of these values, for every such field. To this end, statistics agencies have created appropriate code lists, which are lists of codes that contain all the values a specific field can get. For example, a field that refers to the status of the budget can get one of the values "draft", "revised", "approved", "executed" etc., while a field that refers to the ministry planned to carry out a particular expense, can get as value one of the country's ministries. However, instead of the ministry's name, this budget line value will be a specific code that corresponds to that ministry, based on the code list.

All European countries use code lists for their budget representations, but each country may have its own code lists to represent the values of the respective fields, or it may use code lists in different fields of the budget. Additionally, there are international authorities that provide their own code lists for budget concepts, among other.

In the deliverable D1.6, an extensive survey of the code lists used by international authorities and national agencies was conducted. Two hundred forty code lists that refer to, or are, directly or indirectly, related to budget concepts and could be included in budget representation fields were recorded. These lists were classified under their scope (international/national) and under certain concepts that they refer to, i.e. functions, products, activities etc.

In this deliverable, an additional survey conducted over European budget datasets, indicated the most commonly used fields of the budget that use code lists, in European and national level, as detailed below. Based on this survey, particular code lists of the deliverable D1.6 were selected to be extracted and transformed in RDF, forming the basis for linking various budget datasets, as well as connecting additional code lists used in national budgets, in a next stage.

While code lists are easily discovered within almost every budget dataset, their actual value, as hierarchical dimension properties, does not emerge, because, many times, the lists' integration into the dataset does not allow the user to extract information in order to use it for a purpose other than plain-text filtering.

In other circumstances, code list terms are found within datasets not in a formal representation (i.e. a unique identifier), but with literals. This is common with countries, cities and other geographical attributes found in fiscal datasets. The extraction of the code lists from fiscal datasets allows us to:

1. Replace the literal representation with a more machine-readable one. This leads to easier deduplication and possibly mistyping elimination
2. Organize the code list to an explicitly hierarchical format, if appropriate. Many code list are already hierarchical but there are no documented links between terms, so the hierarchy is denoted by naming conventions only.
3. Link the similar code lists into families and subsequently link the containing datasets, making comparisons and data analysis more straightforward. Even standardized code lists are often overridden in order to accommodate each state's specific needs, which in turn overlap with similar needs in other states.

The extraction of a code list from a fiscal dataset usually results in a flat list of distinct terms. A flat codes list may contain hierarchical relations information within the terms' names. A semantically complete representation would include these relations explicitly. Many times, such flat lists have already been published by their author, so the extraction process is more straightforward and involves downloading the list from a remote server and validating its contents against a format specification. The latter is necessary, in cases where the provided data is claimed to carry a specific format (for instance CSV), but contains syntax errors. To be useful at a later stage, the validated list can be transformed to a format that provides semantic relationships, initially among the terms of the same list. A semantically complete representation

of the code list can then be used to generate any of the simpler representations that may require smaller amounts of information.

Generally speaking, in order to achieve the code list extraction, we need to create a software system that takes as input a fiscal dataset and after processing it, it yields a set of code lists that are contained in the dataset. In this abstract definition we need to also add to the input information on which dataset columns consist of code list terms, and also information on the format of the code list as a whole.

A second software system would then be placed in front of the former system, in order to transform the distinct code list terms into a richer representation. The added data features can be hierarchical relationships between terms and additional attributes that the terms can have, usually coming from external sources. This process may also require integration with a separate system that takes care of linking the various code lists based on similarity of terms across datasets.

The result of these two processes will be a semantic representation of objects, with each object matching a specific term and containing a label, a set of relationship with other terms and a set of additional attributes. For instance, a countries list can be like the following:

Table 1: Sample countries list

ID	Label (country name)	Neighbouring countries	Population
1	Greece	{Albania, FYROM, Bulgaria, Turkey}	10,815,197
2	Czech Republic	{Germany, Poland, Slovakia, Austria}	10,541,466
3	United Kingdom	{Ireland}	63,181,775
4	France	{Belgium, Luxemburg, Germany, Switzerland, Italy, Monaco, Spain, Andorra}	67,107,000
5	Spain	{France, Andorra, Portugal}	46,815,916
6	Germany	{Denmark, Poland, Czech Republic, Austria, Switzerland, France, Luxemburg, Belgium, Netherlands}	81,083,600
…			

A third process could then create updated dataset copies, where the code list attributes are replaced by their respective semantically represented terms, using a unique identifier. The whole process orchestration can be seen in the following diagram:

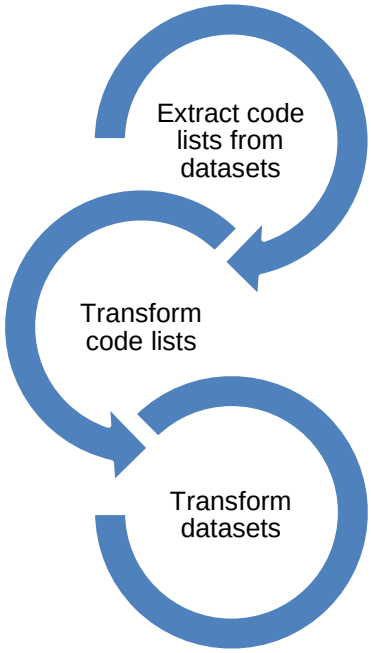


Figure 1: Code Lists extraction and transformation process

2. Requirements

The process mentioned before was specified and implemented using a set of requirements. The requirements were separated into two groups: functional and non-functional. Functional requirements define the exact functions a system is supposed to accomplish, while non-functional requirements support the functional ones by judging the quality aspects of the implementation and the operation of the system. In D4.2, sufficient level of details to design and develop the overall platform were provided. Many of those requirements relate to the Code Lists extraction and transformation, as this process defines how the code lists can be used later on. Most of those requirements relate to the output format of the process. The reader can overview the requirements in the following table:

Table 2: Code lists extraction and transformation requirements

Functional		Non-functional	
F001	RDF Data Structure for Fiscal Data	N002	Localized labels
F005	RDF Modelling for code lists	N003	Scalability
F008	Code lists' mapping support	N006	Seamless integration with other platforms
F010	Ability to attach concrete targets to spending	N008	Clear coding
F011	Link ability		
F019	Semantic Search		
F020	Exploration of processed datasets		
F043	Entity comparison		
F045	Functional comparison		
F047	Filter by administration type		

F048	Get top level aggregates	
F060	Break down functionality	
F015	Loading from an API	

Based on the requirements shown above, SKOS representation was selected as the representation language, as it is expressed in RDF (inherently supporting localized labels), it is standardized and platform independent and allows for hierarchical ordering of terms. A components pipeline was designed in order to represent a set of scenarios and serve as the reference implementation for further needs on extraction and transformation of code lists. The reuse of existing components is crucial on the design and the development of such a pipeline. Various implementations were reviewed, with the most suitable ones to be OpenRefine and Unified Views. Both tools were tested in order to assess which one would better satisfy both the functional and the non-functional requirements of the pipeline.

Open Refine has an RDF extension that enabled us to build the SKOS model and export the data in RDF format. RDF extension was created and released by DERI and can be downloaded for free from this website: http://refine.deri.ie/rdfExport

UnifiedViews (http://www.unifiedviews.eu/) is a Java-based Linked (Open) Data Management Suite to schedule and monitor required tasks (e.g. perform reoccurring extraction, transformation and load processes) for smooth and efficient Linked (Open) Data Management to support web-based Linked Open Data portals (LOD platforms) as well as sustainable Enterprise Linked Data integrations inside of organizations. We used UnifiedViews instead of OpenRefine, because of the scheduled, automated procedures it offered.

1. SKOS representation

Simple Knowledge Organization System (SKOS) is an RDF vocabulary, W3C recommended, designed for representation of thesauri, classification schemes, taxonomies, subject-heading systems, and generally any other type of structured controlled vocabulary. SKOS is part of the Semantic Web family of standards built upon RDF and RDFS, and its main objective is to enable easy publication and use of such vocabularies as linked data. Because SKOS is based on the Resource Description Framework (RDF), these representations are machine-readable and can be exchanged between software applications and published on the World Wide Web.

"SKOS has been designed to provide a low-cost migration path for porting existing organization systems to the Semantic Web. SKOS also provides a lightweight, intuitive conceptual modelling language for developing and sharing new KOSs. It can be used on its own, or in combination with more-formal languages such as the Web Ontology Language (OWL). SKOS can also be seen as a bridging technology, providing the missing link between the rigorous logical formalism of ontology languages such as OWL and the chaotic, informal and weakly-structured world of Web-based collaboration tools, as exemplified by social tagging applications"[1].

In basic SKOS, conceptual resources (concepts) are identified with URIs, labelled with strings in one or more natural languages, documented with various types of note, semantically related to each other in informal hierarchies and association networks, and aggregated into concept schemes.

[1] http://www.w3.org/TR/skos-primer/

3. Code Lists Selection

The selected code lists had to be relevant to fiscal concepts, or to concepts that are used in fiscal datasets. We focused on code lists that meet the task requirements and the needs of the OpenBudgets.eu data model and also refer to economic or other concepts that are included in this model. Additionally, we focused on code lists that are widely used and can be the "bridge" code lists, forming a "linking backbone" to connect additional lists of limited level of use, for instance national level code lists, under the same concept.

More specifically, in order to select the relevant code lists to extract and transform, we examined the fields and concepts that are used in fiscal datasets in Europe and European countries. The most commonly used fields in these datasets are:

- country
- region/area
- year
- organisation/organisational unit/department
- function
- activity
- programme
- fund
- status
- economic classification of revenues and expenses
- other economic classification (accounts/chapters/categories)

Not all of these fields of fiscal datasets had code lists values.

We also considered the Openbudgets.eu data model's dimensions, in order to end up with the concepts and fields that we had to find and extract code lists for. Combining these two kinds of sources, we focused on European Union datasets, to identify code lists that are used in them. The main source of European datasets was the European Union Open Data Portal (https://open-data.europa.eu/en/data) where we selected the "national accounts", "economics" and "finance" tags to specify and examine the relevant datasets (https://open-data.europa.eu/en/data/group/eurovoc_domain_100146?vocab_concepts_eurovoc=http%3A%2F%2Feurovoc.europa.eu%2F56&groups=eurovoc_domain_100148&groups=eurovoc_domain_100146).

In each one of these datasets, there is a metadata page, where a list of the classifications that are used by the datasets, is presented. We considered this metadata information to end up finding what code lists are mostly used, in order to set the final relevant code lists for the deliverable.

The code lists that we selected are:

- Classification of the Functions of Government (COFOG) - UNSD
- Classification of the Outlays of Producers According to Purpose
- (COICOP) - UNSD
- Geographical Standard Code List (GEO) - Eurostat
- Statistical Classification of Products by Activity, Version 2.1 (CPA 2.1) - Eurostat
- Statistical Classification of Economic Activities in the European Community, Rev. 2 (NACE Rev. 2) - Eurostat
- Sectors Codes (CL_SECTOR) - Eurostat
- Organization Identifier (IATI)
- Multiannual Financial Framework Political Categories (MFF/CATPOL) - Eurostat
- European System of National and Regional Accounts (ESA 2010):
    - assets and liabilities
    - balancing items and net worth
    - distributive transactions

- financial assets
- institutional sectors
- non-financial assets
- other changes in assets
- transactions in non-produced, non-financial assets
- transactions in products

COFOG and COICOP were already in SKOS-RDF form, so there was no need to transform these code lists.

We also examined national fiscal datasets, to identify additional, supplementary code lists. However, in many cases, there are many different code lists for the same field or concept, in a country's datasets (i.e. between two different municipalities, or departments). Moreover, based on the values of national fiscal datasets and on the list of classification of national statistical agencies websites, we tried to identify the code lists that are used in these datasets, but, with no result. National statistical agencies websites contain mainly international classifications, from United Nations Statistics Division, or Eurostat and even the national classifications that they include were not identified in the corresponding country's fiscal datasets, or they refer to different concepts. This kind of inconsistency and the lack of any metadata, limited the extraction of national code lists possibilities, thus, there were no attempt to extract code lists that may be used only in a municipality's dataset and nowhere else.

In cases where it was clear which national code lists are used in national fiscal datasets, these code lists where extracted and transformed in RDF too. The Greek budget revenues and expenditure codes are two such code lists.

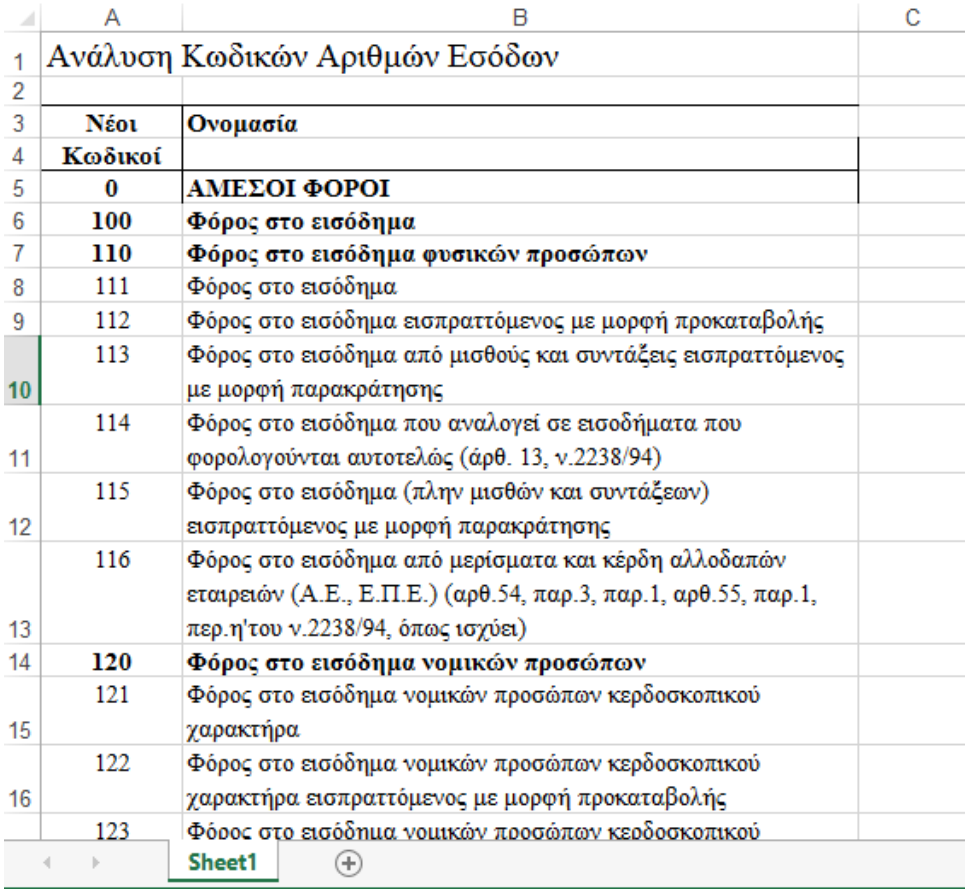
Thus, we have identified the fields and concepts that use code lists in European fiscal datasets. Next, we extracted the most appropriate code lists that meet the needs of the datasets that will be published under OpenBudgets.eu and also, to have a linking backbone for connecting additional code lists. This list can be further enriched with additional code lists, depending on the needs of the next deliverables, as well as the publication of additional fiscal datasets that contain officially recorded code lists, and last, but not least, experts' contribution.

4. Transformation Process Design

The conversion of code lists can be performed automatically, at least in a large part, using UnifiedViews. Prior to feeding data to UnifiedViews, it is useful to get familiar with the dataset, by projecting it to one or more subsets in tabular format. By decomposing the dataset's entities into tables, it is easier to also plan their conversion in UnifiedViews, later. OpenRefine accepts into its input a handful of data formats, ranging from CSV and Excel – like spreadsheets, to JSON and RDF. OpenRefine's graphical editor is an appropriate tool to test transformations on the dataset in a live manner. Following, an example is presented, where OpenRefine is used to design and prototype the transformation process for the Greek Budget Revenues Codes dataset.

1. Example using OpenRefine: Greek Budget Revenues Codes (KAE Esodon)

The Greek Budget Revenues Codes have the format below, where there are many hierarchical relationships between the codes, namely "broader-narrower" relations.



	A	B	C
1	Ανάλυση Κωδικών Αριθμών Εσόδων		
2			
3	**Νέοι Κωδικοί**	**Ονομασία**	
5	**0**	**ΑΜΕΣΟΙ ΦΟΡΟΙ**	
6	**100**	**Φόρος στο εισόδημα**	
7	**110**	**Φόρος στο εισόδημα φυσικών προσώπων**	
8	111	Φόρος στο εισόδημα	
9	112	Φόρος στο εισόδημα εισπραττόμενος με μορφή προκαταβολής	
10	113	Φόρος στο εισόδημα από μισθούς και συντάξεις εισπραττόμενος με μορφή παρακράτησης	
11	114	Φόρος στο εισόδημα που αναλογεί σε εισοδήματα που φορολογούνται αυτοτελώς (άρθ. 13, ν.2238/94)	
12	115	Φόρος στο εισόδημα (πλην μισθών και συντάξεων) εισπραττόμενος με μορφή παρακράτησης	
13	116	Φόρος στο εισόδημα από μερίσματα και κέρδη αλλοδαπών εταιρειών (Α.Ε., Ε.Π.Ε.) (αρθ.54, παρ.3, παρ.1, αρθ.55, παρ.1, περ.η'του ν.2238/94, όπως ισχύει)	
14	**120**	**Φόρος στο εισόδημα νομικών προσώπων**	
15	121	Φόρος στο εισόδημα νομικών προσώπων κερδοσκοπικού χαρακτήρα	
16	122	Φόρος στο εισόδημα νομικών προσώπων κερδοσκοπικού χαρακτήρα εισπραττόμενος με μορφή προκαταβολής	
	123	Φόρος στο εισόδημα νομικών προσώπων κερδοσκοπικού	

Sheet1

Figure 2: Greek Budget Revenues Codes in XLS format

The data were in a spreadsheet (XLS) file and has two columns, the codes column and the names column. Open Refine is the most appropriate tool to clean this kind of data, because of its cleaning and filtering capabilities. A semi-structured data file can be easily edited using transformations, facets and clustering features of this tool, in order to make the data cleanly structured.

First, we created a new Open Refine project and we selected the XLS file with the budget revenues codes. In order to delete the blank rows, we deselected the “store blank rows” option during the creation of the project process. We also detected errors in the values of the codes column, by select the Facet Numeric option for this column. We deleted five rows of non-numeric values.

Then, we created a new column with the broadest codes of the list. We distinguished these ten codes, by selecting the star of their corresponding row. From the codes column, we selected the “edit column” option and then the “add column based on this columns” option. We named the new column “Κωδικοί 0” (“codes 0”) and we deleted the codes from the codes column, by selecting the “edit cells”, “common transforms” and “blank out cells” options consecutively.

With a similar process, we created a new column for the names of the “Κωδικοί 0” (“codes 0”) column codes. We named this new column “Ονομασία: Κωδικοί 0” (“Name: codes 0”) and we deleted the values of names of this column from the “names” column. Then, we selected the “edit cells” and “fill down” options from the “codes 0” column to have this result:

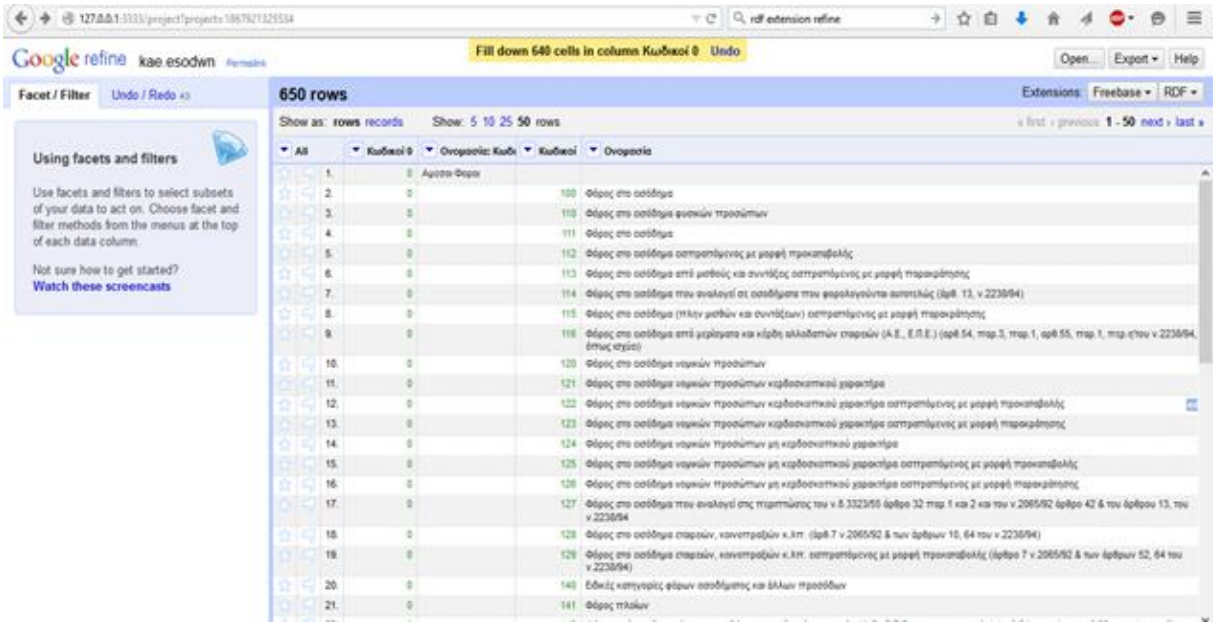


Figure 3: Greek Budget Revenues Codes in Open Refine

There are four levels of hierarchy in this code list. We repeated this procedure for each level and the result is shown in the next picture:

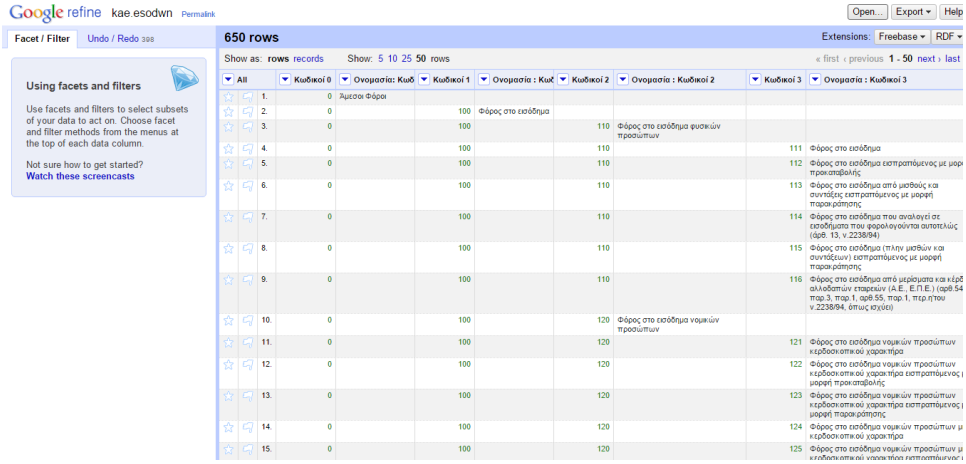


Figure 4: Greek Budget Revenues Codes in an appropriate structure to SKOSify

Having installed the RDF extension for Open Refine, we selected the “RDF” option at the top right of the window and then, the “Edit RDF Skeleton” option. Open Refines creates automatically its own RDF skeleton. We didn't need this skeleton, thus we deleted it (“Reset RDF Skeleton” option) and we created a new one, based on SKOS vocabulary. First, we have to load the SKOS vocabulary, because it is not included in the available prefixes.

We created the SKOS model using concepts and properties such as Concept, topConceptOf, prefLabel, narrower, broader. A visualization example of a code is shown in the next picture, where the code 100 is a skos:Concept, has a property skos:prefLabel which takes its name, and skos:broader and skos:narrower properties for broader and narrower level of revenues categories respectively.

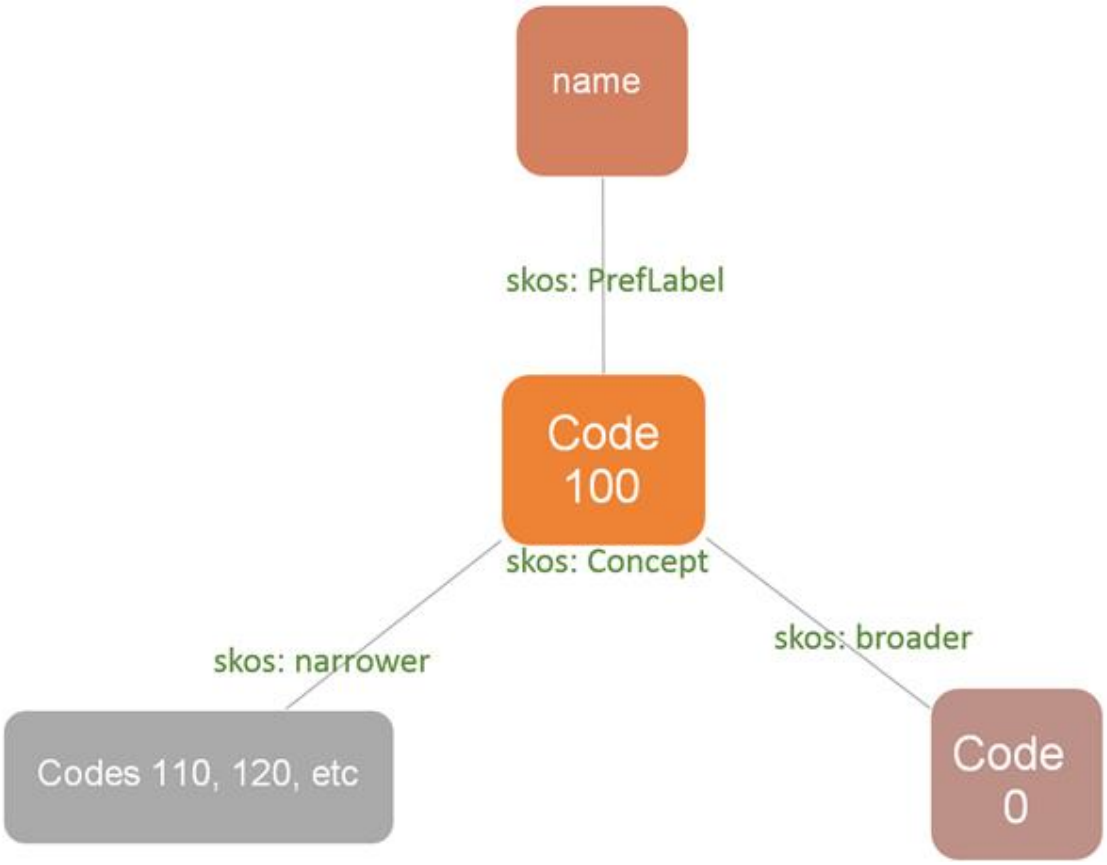


Figure 5: SKOS properties and relationships of the Code 100 of the Greek Budgets Revenues Codes

In this code list, there are four levels of hierarchy, so there would be four trees in the RDF skeleton. This skeleton is shown below:

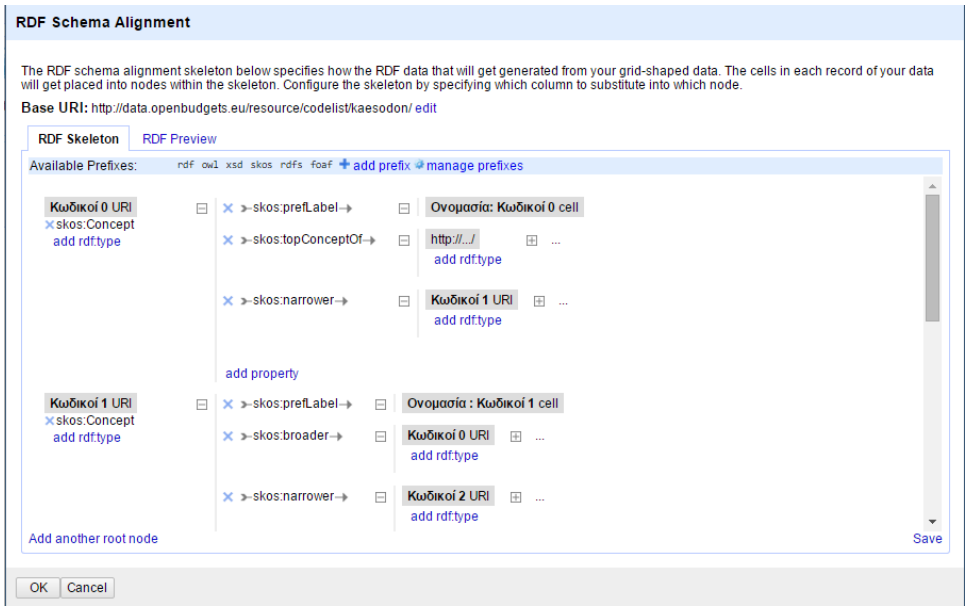


Figure 6: SKOSifying the Greek Budget Revenues Codes in Open Refine

An RDF preview after the transform of the code list is shown in the next picture:

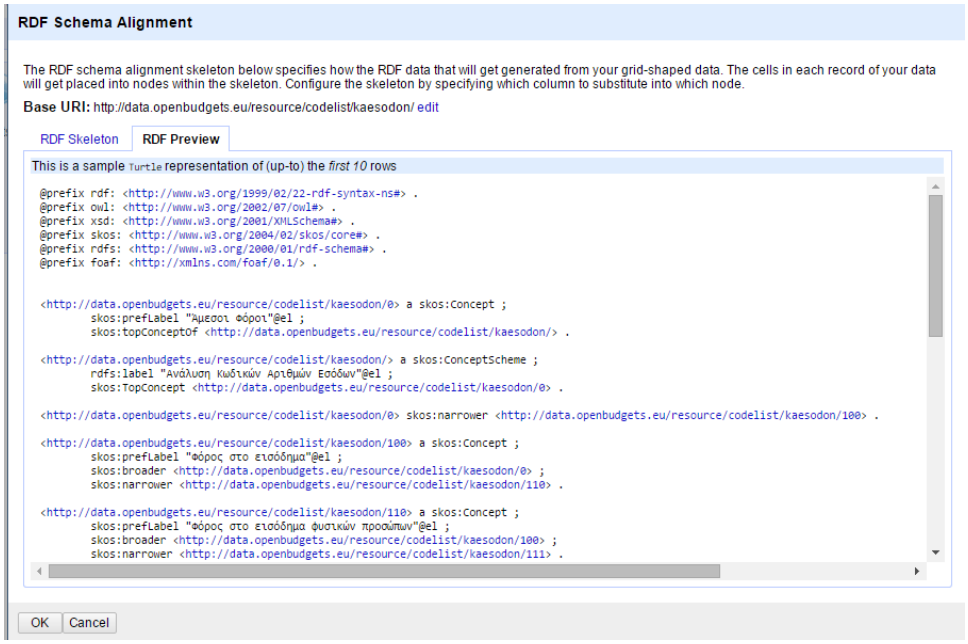


Figure 7: RDF format of the Greek Budget Revenues Codes

For simple code lists, or ones that do not change regularly, OpenRefine can be used to perform the required transformation, as long as the data is closer to a flat table. This is due to OpenRefine currently not being able to repeat the same conversion process with a refresh dataset, as it does not officially support command line or API invocation. Lately, BatchRefine[2] attempts to fill this gap, but is still inefficient in terms of features in comparison to Unified Views. In the following paragraph a more generic approach is described as a pipeline that can be adapted and reused with arbitrary code lists and produce the desired SKOS output.

5. Pipeline definition in Unified Views

In this deliverable we performed a conversion of some of the code lists described in deliverable D 1.6 onto the RDF format. The ontology we used was SKOS ontology as required. We used Unified Views as the core tool in order to perform the description of Code Lists in SKOS form. All converted code lists can be found on GitHub link https://github.com/openbudgets/Code-lists/tree/master/UnifiedViews/skosified and the according pipelines on https://github.com/openbudgets/Code-lists/tree/master/UnifiedViews/pipelines.

1. SKOSifying

In the following examples we demonstrate the pipeline procedures that was used to describe the Code Lists into the SKOS representation. In most cases, a different pipeline structure was needed in order to successfully transform the Code lists into SKOS format.

The following pipeline was used in order to describe CPC code list.

[2] https://github.com/fusepoolP3/p3-batchrefine

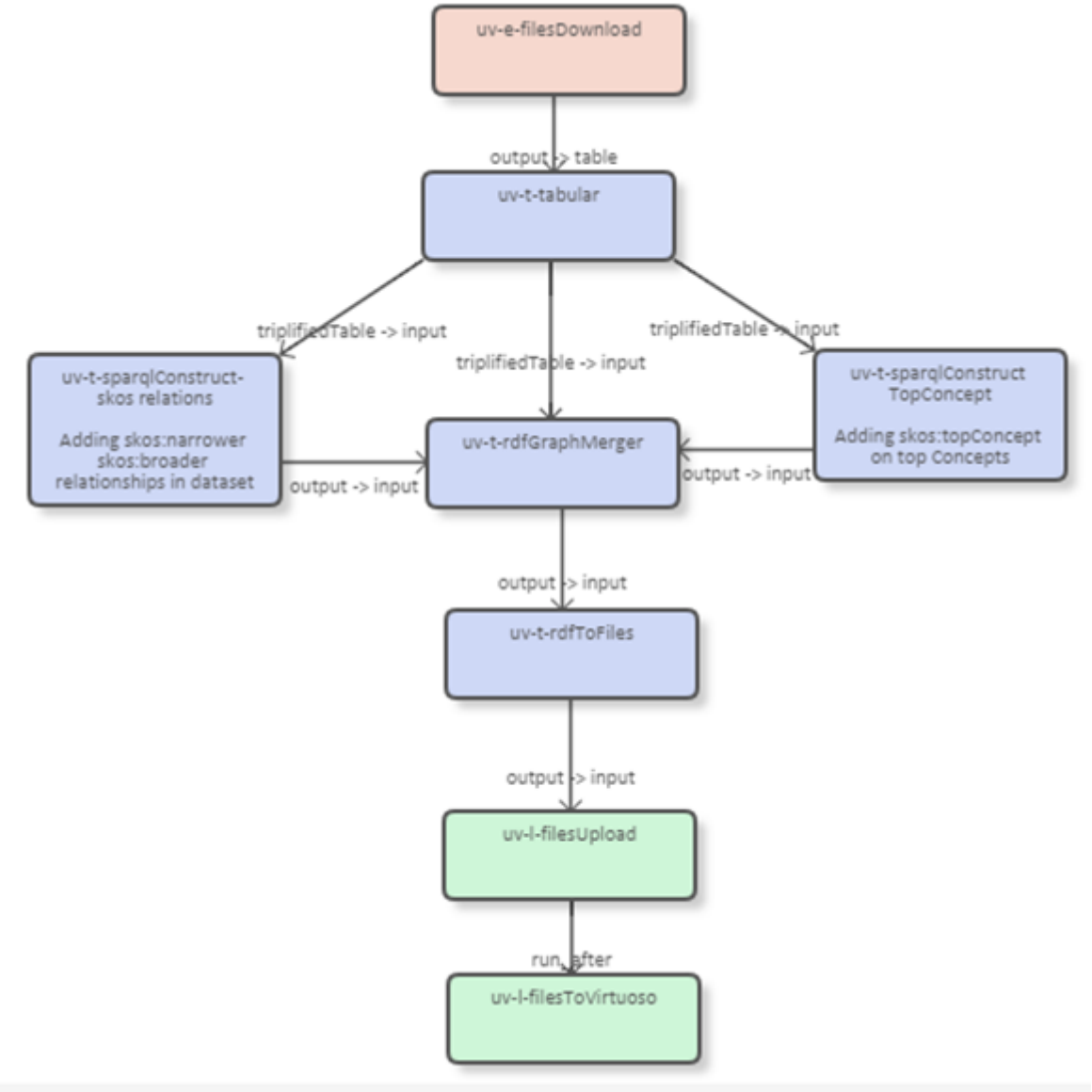


Figure 8: CPC pipeline

Using UnifiedViews to convert a dataset into RDF triples is simply shown above as a working procedure. The first block in red colour is the plugin to download the raw Code List. The most common format of Code lists is CSV-formatted files, so the raw data is then passed on Tabular plugin. In Tabular we can configure how the file is read and also configure how the basic data description will be performed. Here, we also define the data model we will use.

CPC code list consists of a CSV formatted file, containing two columns and 4409 rows.

```
"Code", "Description"
"0","Agriculture, forestry and fishery products"
"01","Products of agriculture, horticulture and market gardening"
"011","Cereals"
"0111","Wheat"
"01111","Wheat, seed"
"01112","Wheat, other"
```

```
"0112","Maize (corn)"
"01121","Maize (corn), seed"
"01122","Maize (corn), other"
"0113","Rice"
"01131","Rice, seed"
"01132","Rice paddy, other (not husked)"
```

Figure 9: CPC sample lines

The first column contains the code, and the second it's description. Tabular plugin can be used in order to construct the basic structure of the converted RDF Code list.

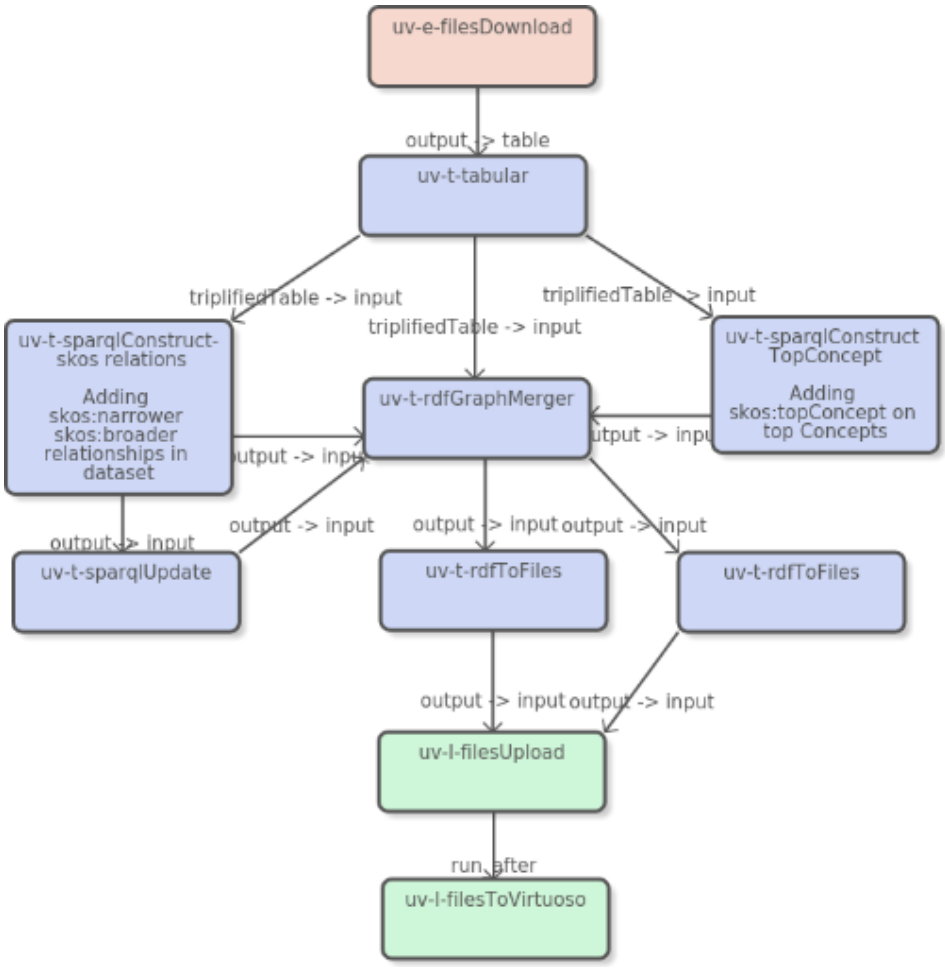


Figure 10: Basic configuration for data extraction of Tabular DPU

Here we configure Tabular on how to read the CSV file, and then how to create the base Entity. We choose the "Code" column in addition with resource URI base http://data.openbudgets.eu/resource/codelist/cpc/ to be the name of the entity. Moreover, here we define the class of the entity. For every entity in the dataset, i.e. every row except header, we choose the fundamental element of the SKOS vocabulary, the skos:Concept class.

The next step is to define the basic mappings. Tabular is powerful enough to construct the main properties of each entity by simply define the column name the type of data it contains and then the property URI that will be used to describe data.

Figure 11: Basic mappings

But when we have to define more complex mappings such as semantic relations between entities, Tabular is inadequate. For the next step we used two instances of the SPARQL Construct DPU. Semantic relations on CPC code list are described by the "Code" column as shown in the next figure.

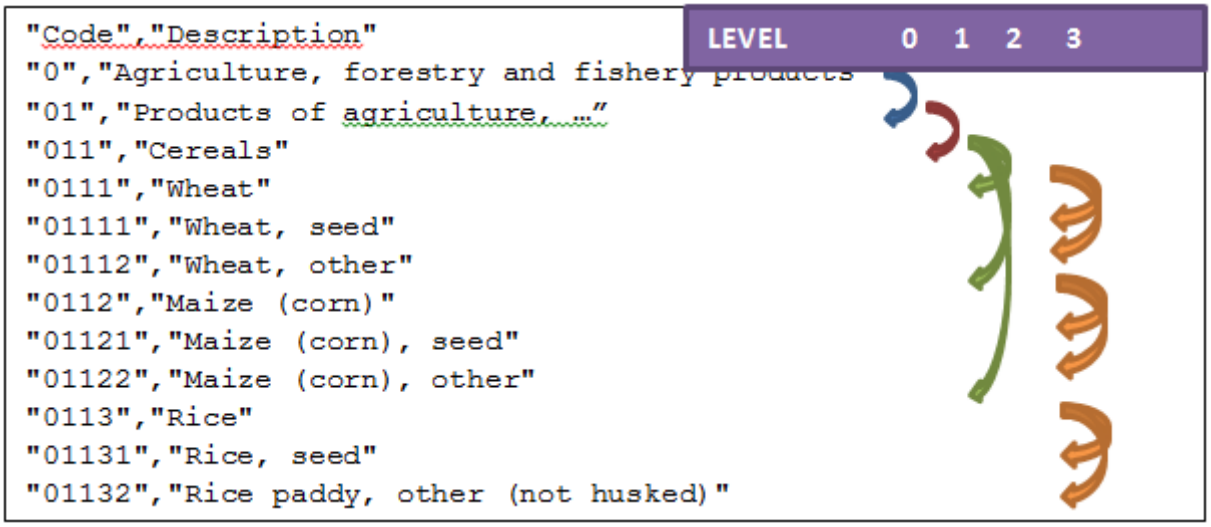


Figure 12: Hierarchy based on code

The relations shown are described by SKOS with the skos:narrower semantic relation property. The inverted relations are described by the skos:broader relation. This kind of mapping cannot be constructed by Tabular plugin. The solution is to use the SPARQL Construct DPU with the following query.

```
construct
{
?s1 <skos:narrower> ?s2 .
?s2 <skos:broader> ?s1 .}
where
{
?s1 <skos:notation> ?o1 .
?s2  <skos:notation> ?o2
filter(strlen(?o2)=(strlen(?o1)+1) && substr(str(?o2),1,strlen(?o1))=str(?o1)).
}
```

Figure 13: Proper relations mapping

With this query we can construct the proper semantic relations between code list entities. The query exploits the fact that code string length denotes its level. And if the part of the string minus the last digit is the equal with another one, then creates a semantic relation between them. Another instance of SPARQL CONSTRUCT DPU is used in order to create semantic relations to the topConcept class of the code list which is usually the base URI.

After the extraction of all triplets are merged into one RDF graph and finally extracted into a single RDF formatted file, or even uploaded into a Virtuoso server.

6. Conclusions

The extraction and transformation of code lists related to fiscal datasets is a vital process for the effective discovery of information within the datasets. The process of extraction and transformation can be automated in order to produce standard SKOS representation of the code lists. Two tools were tested for the purpose, with UnifiedViews qualifying as the more versatile, given the versatility of the datasets and its ability to let the user easily reuse the process as the code lists change.

The extraction and conversion pipeline creation approach was followed with 14 of the code lists collected in the previous task of this Work Package. The conversion resulted in valid SKOS representation of the code lists, which will be useful for the next work packages of OpenBudgets.eu as well as in other contexts, while being uploaded online with a free licence.

7. Appendix: Extracted Code Lists

1. Statistical classification of economic activities in the European Community

Full title	**Statistical classification of economic activities in the European Community**
Abbreviation	**CPC**
Raw Code List Link	http://ec.europa.eu/eurostat/ramon/nomenclatures/index.cfm?TargetUrl=LST_CLS_DLD&StrNom=CPC_2&StrLanguageCode=EN&StrLayoutCode=HIERARCHIC
SKOSified Code List Link	https://github.com/openbudgets/Code-lists/blob/master/skosified/cpc/cpc.ttl
Pipeline GitHub Link	https://github.com/openbudgets/Code-lists/blob/master/UnifiedViews/pipelines/CPC-RC1.zip
Including Hierarchical relations	YES
Base URI	http://data.openbudgets.eu/resource/codelist/cpc/
Entity Class	skos:Concept
Mappings	
Column/Data	Property
Code	skos:notation

Description	skos:prefLabel

2. Classification of Products by Activity

Full title	**Classification of Products by Activity**
Abbreviation	**CPA**
Raw Code List Link	http://ec.europa.eu/eurostat/ramon/nomenclatures/index.cfm?TargetUrl=LST_CLS_DLD&StrNom=CPA_2008&StrLanguageCode=EN&StrLayoutCode=HIERARCHIC
SKOSified Code List Link	https://github.com/openbudgets/Code-lists/blob/master/skosified/cpa/cpa.ttl
Pipeline GitHub Link	https://github.com/openbudgets/Code-lists/blob/master/UnifiedViews/pipelines/CPA2-RC1.zip
Including Hierarchical relations	YES
Base URI	http://data.openbudgets.eu/resource/codelist/cpa/
Entity Class	skos:Concept
Mappings	
Column/Data	Property
Code	skos:notation
Description	skos:prefLabel
Level	skos:note
This item includes	skos:definition
This item also includes	skos:definition
This item excludes	skos:definition

3. Geographical Standard Code List

Full title	**Geographical Standard Code List**
Abbreviation	**GEO**
Raw Code List Link	http://ec.europa.eu/eurostat/ramon/nomenclatures/index.cfm?TargetUrl=LST_CLS_DLD&StrNom=CL_GEO&StrLanguageCode=EN&StrLayoutCode=HIERARCHIC
SKOSified Code List Link	https://github.com/openbudgets/Code-lists/tree/master/skosified/cl_geo

Pipeline GitHub Link	https://github.com/openbudgets/Code-lists/blob/master/UnifiedViews/pipelines/CL-GEO%20rows%200-2000.zip
Including Hierarchical relations	TRUE
Base URI	http://data.openbudgets.eu/resource/codelist/cl-geo/
Entity Class	skos:Concept
Mappings	
Column/Data	Property
Code2	skos:notation
Description	skos:prefLabel
Code	skos:definition
Level	skos:note
NUTS level	skos:note

4. Transactions in financial assets and liabilities

Full title	**Transactions in financial assets and liabilities**
Abbreviation	ESA_F 2010
Raw Code List Link	http://metaweb.stat.ee/classificator_publish_list.htm?siteLanguage=en
SKOSified Code List Link	https://github.com/openbudgets/Code-lists/blob/master/skosified/ESA2010_assets_and_liabilities/esa2010-assets-and-liabilities.ttl
Pipeline GitHub Link	https://github.com/openbudgets/Code-lists/blob/master/UnifiedViews/pipelines/ESA2010_assets_and_liabilities.zip
Including Hierarchical relations	YES
Base URI	http://data.openbudgets.eu/resource/codelist/esa2010-assets-and-liabilities/
Entity Class	skos:Concept
Mappings	
Column/Data	Property
col1	skos:notation
col2	skos:prefLabel

5. Classification of balancing items and net worth

Full title	Classification of balancing items and net worth
Abbreviation	ESA_B 2010
Raw Code List Link	http://metaweb.stat.ee/classificator_publish_list.htm?siteLanguage=en
SKOSified Code List Link	https://github.com/openbudgets/Code-lists/blob/master/skosified/ESA2010_balancing_items_and_net_worth/esa2010-balancing-items-and-net-worth.ttl
Pipeline GitHub Link	https://github.com/openbudgets/Code-lists/blob/master/UnifiedViews/pipelines/ESA2010_balancing%20items%20and%20net%20worth.zip
Including Hierarchical relations	YES
Base URI	http://data.openbudgets.eu/resource/codelist/esa2010-balancing-items-and-net-worth/
Entity Class	skos:Concept
Mappings	
Column/Data	Property
col1	skos:notation
col2	skos:prefLabel

6. Distributive transactions

Full title	Distributive transactions
Abbreviation	ESA_D 2010
Raw Code List Link	http://metaweb.stat.ee/classificator_publish_list.htm?siteLanguage=en
SKOSified Code List Link	https://github.com/openbudgets/Code-lists/blob/master/skosified/ESA2010_distributive_transactions/esa2010-distributive-transactions.ttl
Pipeline GitHub Link	https://github.com/openbudgets/Code-lists/blob/master/UnifiedViews/pipelines/ESA2010_distributive_transactions.zip
Including Hierarchical relations	YES
Base URI	http://data.openbudgets.eu/resource/codelist/esa2010-distributive-transactions/
Entity Class	skos:Concept

Mappings	
Column/Data	Property
col1	skos:notation
col2	skos:prefLabel

7. Financial assets

Full title	**Financial assets**
Abbreviation	ESA_AF 2010
Raw Code List Link	http://metaweb.stat.ee/classificator_publish_list.htm?siteLanguage=en
SKOSified Code List Link	https://github.com/openbudgets/Code-lists/blob/master/skosified/ESA2010_financial_assets/esa2010-financial-assets.ttl
Pipeline GitHub Link	https://github.com/openbudgets/Code-lists/blob/master/UnifiedViews/pipelines/ESA2010_financial_assets.zip
Including Hierarchical relations	YES
Base URI	http://data.openbudgets.eu/resource//codelist/esa2010-financial-assets/
Entity Class	skos:Concept
Mappings	
Column/Data	Property
col1	skos:notation
col2	skos:prefLabel

8. Nomenclature of the Classification of Sectors

Full title	**Nomenclature of the Classification of Sectors**
Abbreviation	ESA_S 2010
Raw Code List Link	http://metaweb.stat.ee/classificator_publish_list.htm?siteLanguage=en
SKOSified Code List Link	https://github.com/openbudgets/Code-lists/blob/master/skosified/ESA2010_institutional_sectors/esa2010-institutional-sectors.ttl
Pipeline GitHub Link	https://github.com/openbudgets/Code-lists/blob/master/UnifiedViews/pipelines/ESA2010_institutional%20sectors.zip

Including Hierarchical relations	TRUE
Base URI	http://data.openbudgets.eu/resource/codelist/esa2010-institutional-sectors/
Entity Class	skos:Concept
Mappings	
Column/Data	Property
col1	skos:notation
col2	skos:prefLabel

9. Non-financial assets

Full title	**Non-financial assets**
Abbreviation	ESA_AN 2010
Raw Code List Link	http://metaweb.stat.ee/classificator_publish_list.htm?siteLanguage=en
SKOSified Code List Link	https://github.com/openbudgets/Code-lists/blob/master/skosified/ESA2010_non-financial_assets/esa2010-non-financial-assets.ttl
Pipeline GitHub Link	https://github.com/openbudgets/Code-lists/blob/master/UnifiedViews/pipelines/ESA2010_non-financial_assets.zip
Including Hierarchical relations	YES
Base URI	http://data.openbudgets.eu/resource/codelist/esa2010-non-financial-assets/
Entity Class	skos:Concept
Mappings	
Column/Data	Property
col1	skos:notation
col2	skos:prefLabel

10. Other changes in assets

Full title	**Other changes in assets**
Abbreviation	ESA_K 2010
Raw Code List Link	http://metaweb.stat.ee/classificator_publish_list.htm?siteLanguage=en
SKOSified Code List Link	https://github.com/openbudgets/Code-lists/blob/master/skosified/ESA2010_other_

	changes_in_assets/esa2010-other-changes-in-assets.ttl
Pipeline GitHub Link	https://github.com/openbudgets/Code-lists/blob/master/UnifiedViews/pipelines/ESA2010_other_changes_in_assets.zip
Including Hierarchical relations	YES
Base URI	http://data.openbudgets.eu/resource//codelist/esa2010-other-changes-in-assets/
Entity Class	skos:Concept
Mappings	
Column/Data	Property
col1	skos:notation
col2	skos:prefLabel

11. Transactions in products

Full title	**Transactions in products**
Abbreviation	ESA_P 2010
Raw Code List Link	http://metaweb.stat.ee/classificator_publish_list.htm?siteLanguage=en
SKOSified Code List Link	https://github.com/openbudgets/Code-lists/blob/master/skosified/ESA2010_transactions_in_products/esa2010-transactions-in-products.ttl
Pipeline GitHub Link	https://github.com/openbudgets/Code-lists/blob/master/UnifiedViews/pipelines/ESA2010_transactions_in_products.zip
Including Hierarchical relations	YES
Base URI	http://data.openbudgets.eu/resource/codelist/esa2010-transactions-in-products/
Entity Class	skos:Concept
Mappings	
Column/Data	Property
col1	skos:notation
col2	skos:prefLabel

12. ISO 4217 World Currency

Full title	ISO 4217 World Currency

Abbreviation	ISO 4217
Raw Code List Link	http://www.currency-iso.org/dam/downloads/lists/list_one.xls
SKOSified Code List Link	http://ontologycentral.com/2009/05/currency/iso-4217.rdf https://github.com/openbudgets/Code-lists/blob/master/UnifiedViews/skosified/currencies/currencies.ttl
Pipeline GitHub Link	https://github.com/openbudgets/Code-lists/blob/master/UnifiedViews/pipelines/Currency-RC1.zip
Including Hierarchical relations	TRUE
Base URI	http://data.openbudgets.eu/codelist/currency/
Entity Class	http://data.openbudgets.eu/ontology/Currency
Mappings	
Column/Data	Property
Currency	skos:prefLabel
Alpabetic code	skos:notation
Numeric code	skos:altLabel

13. Organization Identifier

Full title	Organization Identifier
Abbreviation	ISO 4217
Raw Code List Link	http://iatistandard.org/201/codelists/OrganisationIdentifier/
SKOSified Code List Link	https://github.com/openbudgets/Code-lists/blob/master/UnifiedViews/skosified/OrganizationIdentifier/OrganizationIdentifier.ttl
Pipeline GitHub Link	https://github.com/openbudgets/Code-lists/blob/master/UnifiedViews/pipelines/Organization%20Identifier.zip
Including Hierarchical relations	TRUE
Base URI	http://data.openbudgets.eu/resource/codelist/organization-identifier/
Entity Class	skos:Concept

Mappings	
Column/Data	Property
code	skos:notation
name	skos:prefLabel

14. Multiannual Financial Framework/Political Categories

Full title	Multiannual Financial Framework Political Categories
Abbreviation	MFF/CATPOL
Raw Code List Link	http://eur-lex.europa.eu/budget/data/General/2014/en/Final-budget-2014-EN.zip
SKOSified Code List Link	
Pipeline GitHub Link	
Including Hierarchical relations	YES
Base URI	http://data.openbudgets.eu/resource/codelist/organization-identifier/
Entity Class	skos:Concept
Mappings	
Column/Data	Property
code	skos:notation
name	skos:prefLabel