Deliverable Number: 3.13
Deliverable Title: Report on multilingual coding platform
Work Package: 3 – Maximizing equivalence through translation

Deliverable type: Other
Dissemination status: Public

Submitted by SHARE ERIC

Authors: Maurice Martens (CentERdata)

Date Submitted: June 2019
V 1.1
SERISS (Synergies for Europe’s Research Infrastructures in the Social Sciences) aims to exploit synergies, foster collaboration and develop shared standards between Europe’s social science infrastructures in order to better equip these infrastructures to play a major role in addressing Europe’s grand societal challenges and ensure that European policymaking is built on a solid base of the highest-quality socio-economic evidence.

The four year project (2015-19) is a collaboration between the three leading European Research Infrastructures in the social sciences – the European Social Survey (ESS ERIC), the Survey of Health, Ageing and Retirement in Europe (SHARE ERIC) and the Consortium of European Social Science Data Archives (CESSDA AS) – and organisations representing the Generations and Gender Programme (GGP), European Values Study (EVS), and the WageIndicator Survey.

Work focuses on three key areas: Addressing key challenges for cross-national data collection, breaking down barriers between social science infrastructures, and embracing the future of the social sciences.

Please cite this deliverable as: Martens, M. (2019) Report on multilingual coding platform. *Deliverable 3.13 of the SERISS project funded under the European Union’s Horizon 2020 research and innovation programme* GA No: 654221. Available at: [www.seriss.eu/resources/deliverables](http://www.seriss.eu/resources/deliverables)
Contents

1 Introduction .......................................................................................................................... 5
  1.1 Purpose of the tool .......................................................................................................... 5
  1.2 Current status .................................................................................................................. 6
  1.3 This document ................................................................................................................ 6
  1.4 Further information ......................................................................................................... 6

2 Software Requirements Specification ................................................................................. 6
  2.1 General Requirements ..................................................................................................... 6
     2.1.1 Software Perspective ................................................................................................. 6
     2.1.2 Software functions .................................................................................................... 6
  2.2 API .................................................................................................................................. 11
     2.2.1 Good to know ........................................................................................................... 11
     2.2.2 Common arguments .................................................................................................. 12
     2.2.3 Mode 1: List of countries ......................................................................................... 12
     2.2.4 Mode 2: Search (used in autocomplete) .................................................................... 12
     2.2.5 Mode 3: Tree view .................................................................................................... 13
     2.2.6 Mode 4: Retrieving data of specific coding ............................................................... 13
  2.3 Operating Environment .................................................................................................. 14
     2.3.1 Design and Implementation Constraints .................................................................. 14
     2.3.2 User Documentation .................................................................................................. 14
     2.3.3 Assumptions and Dependencies .............................................................................. 14
  2.4 External Interface Requirements .................................................................................... 14
     2.4.1 User Interfaces .......................................................................................................... 14
     2.4.2 Hardware Interfaces .................................................................................................. 14
     2.4.3 Software Interfaces .................................................................................................. 14
     2.4.4 Communication Interfaces ....................................................................................... 14
  2.5 Other Non-functional Requirements .............................................................................. 14
     2.5.1 Performance Requirements ..................................................................................... 14
     2.5.2 Security Requirements ............................................................................................. 15
     2.5.3 Software Quality Attributes ..................................................................................... 15

3 Usability Specification ........................................................................................................ 15

4 System Architecture and Design ....................................................................................... 15
  4.1 System Overview ............................................................................................................. 15
  4.2 Design Constraints ......................................................................................................... 15
  4.3 System Architecture ....................................................................................................... 16
     4.3.1 System Hardware Architecture ............................................................................... 16
     4.3.2 System Software Architecture ............................................................................... 16
  4.4 File and Database Design ............................................................................................... 16
     4.4.1 Database Management System Files ....................................................................... 16
1 Introduction

As part of Synergies for Europe’s Research Infrastructures in the Social Sciences (SERISS) programmers from CentERdata, Tilburg University have been working with several survey infrastructures to develop the Translation Management Tool (TMT). The TMT was originally developed for the Survey of Health, Ageing and Retirement in Europe (SHARE). Under SERISS, the TMT has been adapted to support other large scale studies that need to translate survey questionnaires including ESS, EVS and GGP.

Another task in this work package was to extend the Translation Management Tool to also support the building of sets needed for coding of socio-economic indicators, like Education, Industry and Occupation. However, during the design and implementation of this extension it became clear it was not very convenient or even sensible to provide this functionality as an integral part of the TMT.

Translation for surveys is a different task than the development of these sets. The tasks involved are done by completely different experts, with different workflows. Questionnaire translation is something that is mostly done for a single study or for a limited number of questionnaires. The multilingual coding tools are developed by a team of experts independent of the study and made available for common reuse. Once made available, items have to be classified in international standard classifications, but not always (re-)translated. Guided by the experiences under SERISS WP8, a separate tool has therefore been developed that manages coded items and their classifications. This tool links directly to the coding module as developed under WP8.

In this document, we will describe the tool we developed for the experts who will use it to maintain these datasets, the SurveyCodings-backend.

The SurveyCodings-backend is an online environment that centrally stores sets of classified strings. It is highly configurable for usage in questionnaires. This environment has been developed in PHP with an underlying MySQL database. We refer to it as the ‘backend’ because there is also a ‘frontend’. The frontend is a website plus service that is used to disseminate this data and allow easy access for questionnaire developers. This website can be accessed at https://www.surveycodings.org.

The SurveyCodings-backend is configured per domain. For example, we can collect a large set of occupation titles. The titles are phrased in a language, they are relevant in a certain culture or country. We can have one or more classifications per title. Perhaps we can think of some synonyms, or can we assume synonyms are codable items themselves? We also might know of one or more search trees we can build on the domain to structure and guide respondents in their search efforts. To store this information the SurveyCodings-backend was designed.

1.1 Purpose of the tool

The SurveyCodings-backend, developed by CentERdata, University of Tilburg, is designed to allow stakeholders – experts in the area of classifications - to manage a set of classified and sometimes translated strings. The SurveyCodings-backend has six main aims:

- To define a validate set of coded items, versioned, reusable
- To classify, define classifications and link them to the sets
- To allow for design of tree views on subsets of the coded strings
- To facilitate a platform where coding experts can meet
- To provide a platform where users can report problems and suggest adaptations and expansions
- To provide a platform that can be connected to directly by online questionnaires
1.2 Current status
The SurveyCodings-backend has been developed during the SERISS project. It will need updates in the future, to comply with future server requirements, to add new contexts, but also to support new coding domains, and new features to maintain and support the challenges, new developments and trends that come up in the survey industry. To provide optimal support for the studies using the SurveyCodings data sets and service, it was decided to centrally host it and allow access to it as a service. Each survey, study or questionnaire that uses its content can connect via the API (api.surveycodings.org) or download sets form the public website (www.surveycodings.org). This setup allows for optimal synergy and a quick response time when technical challenges present themselves.

1.3 This document
This document provides information on the SurveyCodings-backend software. It is an introduction for developers taking over the development or maintenance of the system to learn about its architecture and design principles. Since the service is live and being updated when new trends occur or problems are found and corrected, this document might be outdated, we advise readers to visit the SurveyCodings website (https://www.surveycodings.org) where the latest information can be found.

This report provides details on the SurveyCodings-backend software specification. It also provides a high level overview of the system architecture and its inputs and outputs. It summarises the field-testing the tool has undergone to date and the future testing required.

1.4 Further information
More information on this tool is available online at https://www.surveycodings.org.

2 Software Requirements Specification

2.1 General Requirements

2.1.1 Software Perspective
The centralized collection of validated lists presents a unique challenge to systematically improve and create methods that can be used in surveys. The SurveyCodings-backend is a place where experts can meet and discuss the items and classifications assigned to these items. The conceptual design is open, and can easily be extended to include alternative domains. With little efforts the API, and connected tooling can be reused.

2.1.2 Software functions
The SurveyCodings-backend collects and manages sets of translated strings, synonyms of those strings, classified in several classifications for various domains. You can navigate between the domains by clicking on the top tabs. Currently ‘Education’, ‘Fields of Education”, “Industry” and “Occupation” are available.

Figure 1: Domain Tabs
On the left side of each of these pages a general menu is available.

Figure 2: Menu per domain

The codings menu item will give you a full overview of the codings within the domain. Within the page it is possible to filter the list and make available certain contexts only.

Figure 3: Codings overview

Within the codings page, an alternative interface is available that allows the user to define a tree interface from the available codings. This can be done per context or globally depending on how the structure can be defined. For example, educational systems differ by design in each country, while for many countries the labour force system can be represented in similar structures.
In the codings menu, it is possible to generate XML files that describe subsets for reuse in other tools as well.

The second menu item is Contexts. For many domains, the contexts could be defined as a Language-Country pair (a Locale), but this is not always the case. The context might also be on a Language level, where there is no Country-specific difference. Alternatively, within a Language-Country context, there might also be differences at, for example, a regional level. Then the context of the domain shifts to that regional level. This is, for example, the case in the United Kingdom, where educational systems differ in England and Scotland. The locale en_GB however is the same.
Figure 6: Contexts overview

For each context a set of properties is defined. Also if you zoom into a context, the list of codings attached to the context can be reviewed.

Figure 7: Context view

It is possible to classify the codings. The system allows the user to define these classifications themselves.
Once a classification is added, Classification Elements can be added to it.

These Classification Elements can be assigned to multiple codings. It is possible to assign one Classification Element per Classification to a coding as can be seen once a coding is further inspected. In this interface it is possible to add source information, deviations and other notes.
2.2 API

Linked to the SurveyCodings-backend is an application programming interface (API). An API is a set of functions and procedures allowing the creation of applications that access the features or data of an operating system, application, or other service.

There are four modes of data retrieval. Each mode is used in different situations. You can view a complete example when inspecting the html and JavaScript code on the database live search pages throughout this website.

2.2.1 Good to know

- Each mode has a specific set of combination of options. Please see below the description of each mode for the correct options.
- **Always** tell the API which database to use. This is done with the **type** argument. See below for details.
- The API returns all data in **JSON** format. The first argument starts with a question mark ?, the following arguments start with &. See the examples on how to write a complete URL correctly.
2.2.2 Common arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Options</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>education occupation industry</td>
<td>Tells the API in which database to search the codings. There are three different databases with codings. With argument type=[database] you can tell the API which database to use. Always use this argument.</td>
<td>type=education type=occupation type=industry</td>
</tr>
<tr>
<td>Mode</td>
<td>country search tree coding</td>
<td>Tells the API what to do.</td>
<td>mode=country mode=search mode=tree mode=coding</td>
</tr>
</tbody>
</table>

2.2.3 Mode 1: List of countries

Tell the API to return a JSON format of all countries in the education database that have codings. The arguments type and mode are used to retrieve the requested data. See the above section on what they do.

Example: [https://api.surveycodings.org/codings/search.json?type=education&mode=country](https://api.surveycodings.org/codings/search.json?type=education&mode=country)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Options</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>mode</td>
<td>country</td>
<td>The above URL tells the API to return a JSON format of all countries.</td>
<td>mode=country</td>
</tr>
</tbody>
</table>

2.2.4 Mode 2: Search (used in autocomplete)

Tell the API to return a JSON list of all codings for a specific context where the description matches a given search string.

Example education database: [https://api.surveycodings.org/codings/search.json?type=education&mode=search&context=52800&search=pr](https://api.surveycodings.org/codings/search.json?type=education&mode=search&context=52800&search=pr)


<table>
<thead>
<tr>
<th>Argument</th>
<th>Options</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode</td>
<td>search</td>
<td>Tells the API to search for codings based on a text search.</td>
<td>mode=search</td>
</tr>
</tbody>
</table>
2.2.5 Mode 3: Tree view

Tell the API to return a JSON tree representation of all codings for a specific context.

Example education database:
https://api.surveycodings.org/codings/search.json?type=education&mode=tree&context=52800

Example occupation/industry database:
https://api.surveycodings.org/codings/search.json?type=occupation&mode=tree&context=nl_NL
https://api.surveycodings.org/codings/search.json?type=industry&mode=tree&context=nl_NL

<table>
<thead>
<tr>
<th>Argument</th>
<th>Options</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>mode</td>
<td>tree</td>
<td>Tells the API to return a full tree of all codings.</td>
<td>mode=tree</td>
</tr>
<tr>
<td>context</td>
<td>[integer] or [string]</td>
<td>Tells the database in which context to look for codings. The context for education is an integer. The context for industry and occupation are strings.</td>
<td>Example education: context=52800 Dutch codings. Example occupation: context=nl_NL Dutch codings. Example industry: context=nl_NL Dutch codings.</td>
</tr>
</tbody>
</table>

2.2.6 Mode 4: Retrieving data of specific coding

Tell the API to return detailed data about the requested coding.

Returns an empty set if

<table>
<thead>
<tr>
<th>Argument</th>
<th>Options</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>mode</td>
<td>coding</td>
<td>Tells the API to return the details of a specific coding.</td>
<td>mode=coding</td>
</tr>
<tr>
<td>coding</td>
<td>[integer]</td>
<td>Tells the database which coding is requested. This is the name of the coding.</td>
<td>coding=52800079</td>
</tr>
</tbody>
</table>

2.3 Operating Environment
The SurveyCodings-backend is a PHP based web application running on a WAMPP or LAMPP stack. It is built on top of the CakePHP 2.4 framework.

2.3.1 Design and Implementation Constraints
The SurveyCodings-backend is designed as a service installed centrally on a website. This allows for optimal configurability, support and synergy.

2.3.2 User Documentation
The expert users of the SurveyCodings-backend are trained to use the system. At this stage, there is no user documentation available.

2.3.3 Assumptions and Dependencies
- The SurveyCodings-backend does not depend on outside libraries.
- To allow for import of Excel sheets a strictly defined format should be used.
- Exports in Word and Excel should be viewed with Office.
- A stable PHP version is mandatory to work correctly.

2.4 External Interface Requirements
2.4.1 User Interfaces
The SurveyCodings-backend runs online and can be used in modern web browsers, like Mozilla or Chrome. There is no need to install anything on the user’s local machine.

2.4.2 Hardware Interfaces
The SurveyCodings-backend is set up as a service, there are no hardware requirements to be met for using it.

2.4.3 Software Interfaces
Not relevant.

2.4.4 Communication Interfaces
The SurveyCodings-backend imports strictly defined Excel formats, and exports XML format. Also various scripts such as, Stata syntaxes can be generated. An API has been developed on top of the model, to be used by external online surveys.

2.5 Other Non-functional Requirements
2.5.1 Performance Requirements
The SurveyCodings-backend is able to handle multiple users working at the same time

Safety Requirements
The various SurveyCodings datasets are versioned.
2.5.2 Security Requirements
The SurveyCodings-backend does not handle personal data. Therefore, data security is not of paramount importance. However, to protect the integrity of the system we use HTTPS. This protects intruders from tampering with the communications between the system and users’ browsers. The database management system is restricted to local clients. Users log in to the SurveyCodings-backend using encoded passwords. Within the SurveyCodings-backend, top level administrative actions can only be done by users belonging to the Administrator group.

2.5.3 Software Quality Attributes
Attributes that should be prioritized when developing the software are:

- Usability - the tool must be intuitive to use;
- Responsiveness - users must be able to enter data in real time;
- Interoperability - the tool must be operational in different countries.

3 Usability Specification
Since the system is intended for a small group of expert users, usability was not a key requirement. We did however put in effort to set up a usable environment.

4 System Architecture and Design
4.1 System Overview
The system has a client server architecture where the clients who make use of a pc use a web browser to interact with the application.

![Client-Server model](image)

Figure 11: Client-Server model

4.2 Design Constraints
The design constraints used for developing the software are:

- Import and export of data should take place via a known format usable by different software tools;
- The webserver used is limited to Apache;
- Database support is limited to MySQL;
- The graphical user interface is created via HTML with the help of the HTML, CSS and JavaScript;
- The PHP guidelines for development are followed.

4.3 **System Architecture**

4.3.1 **System Hardware Architecture**
The current hardware architecture is partitioned in 2 main servers with the following specifications

1. Apache web server
2. MySQL database server

The apache web server hosts the main system interface. It handles the different data downloading and uploading actions via HTTPS.

The MySQL database server stores the data provided by the web server.

4.3.2 **System Software Architecture**
The SurveyCodings-backend uses the CakePHP framework which enforces a Model View Controller architecture so the application can be maintained more easily.

![MVC architecture](image)

**Figure 12: MVC architecture**

4.4 **File and Database Design**

4.4.1 **Database Management System Files**
For each version of a coding domain, we define a database; the entities used in the system are:

<table>
<thead>
<tr>
<th>Entity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>classifications</td>
<td>Definition of a classification.</td>
</tr>
<tr>
<td>classifications_elements</td>
<td>Elements belonging to a classification.</td>
</tr>
<tr>
<td>classifications_elements_codings</td>
<td>Link table that defines per code several classification_elements.</td>
</tr>
<tr>
<td>classifications_elements_properties</td>
<td>Additional properties for classification_elements if relevant.</td>
</tr>
<tr>
<td>Term</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>classification_properties</td>
<td>Extra properties for classifications if relevant.</td>
</tr>
<tr>
<td>classification_sources</td>
<td>Sources of the chosen classification.</td>
</tr>
<tr>
<td>classification_trees</td>
<td>The tree structure of a classification, how classification elements are ordered hierarchically within a classification.</td>
</tr>
<tr>
<td>classification_types</td>
<td>The set with classification types.</td>
</tr>
<tr>
<td>codings</td>
<td>Codings is the set of classified strings, either per context, if different per context, or by default a source string in English, if translatable.</td>
</tr>
<tr>
<td>codings_translation</td>
<td>If codes can be translated, while maintaining valid classification, the translations are defined in the codings_translation table.</td>
</tr>
<tr>
<td>contexts</td>
<td>The context is the area in which the codable item is valid. Typically, this would be language, country, locale or region.</td>
</tr>
<tr>
<td>languages</td>
<td>The languages available to the system.</td>
</tr>
<tr>
<td>nodes</td>
<td>The elements available in the tree structures are defined as nodes, where each node knows its parent node and tree_id.</td>
</tr>
<tr>
<td>notes</td>
<td>Set of notes that can be attached to codes.</td>
</tr>
<tr>
<td>roles</td>
<td>Definition of the roles available in the SurveyCodings-backend, together with an acos, aros table, the rights are defined per role.</td>
</tr>
<tr>
<td>sources</td>
<td>Set of potential sources a code could have originated from.</td>
</tr>
<tr>
<td>synonyms</td>
<td>If codes have synonyms that do not add to the codes set, but should be matched against, then they are stored here.</td>
</tr>
<tr>
<td>translations</td>
<td>The set of translations of translation_elements_types like</td>
</tr>
<tr>
<td>Entity Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>translation_element_types</td>
<td>Set of element_types that are used in the interface to show context-relevant information.</td>
</tr>
<tr>
<td>trees</td>
<td>Definition of a tree, if translatable, then for all context, else per context.</td>
</tr>
<tr>
<td>tree_reference</td>
<td>Identifies the user that created the tree, possibly with remark or comments.</td>
</tr>
<tr>
<td>users</td>
<td>Users known in the system that have rights to the relevant domain.</td>
</tr>
</tbody>
</table>

The relations between these entities are visualised in the Entity Relationship Diagram below:

![Entity Relationship Diagram](image)

**Figure 13: Entity Relationship Diagram**