German Medical Data Sciences: Bringing Data to Life
R. Röhrig et al. (Eds.)

doi:10.3233/SHTI210056

ISO 21526 Conform Metadata Editor for FAIR Unicode SKOS Thesauri

Mark R. STÖHR[a,1], Andreas GÜNTHER[a] and Raphael W. MAJEED[a]
[a] *UGMLC, German Center for Lung Research (DZL),*
Justus-Liebig-University, Giessen, Germany

Abstract. Metadata repositories are an indispensable component of data integration infrastructures and support semantic interoperability between knowledge organization systems. Standards for metadata representation like the ISO/IEC 11179 as well as the Resource Description Framework (RDF) and the Simple Knowledge Organization System (SKOS) by the World Wide Web Consortium were published to ensure metadata interoperability, maintainability and sustainability. The FAIR guidelines were composed to explicate those aspects in four principles divided in fifteen sub-principles. The ISO/IEC 21526 standard extends the 11179 standard for the domain of health care and mandates that SKOS be used for certain scenarios. In medical informatics, the composition of health care SKOS classification schemes is often managed by documentalists and data scientists. They use editors, which support them in producing comprehensive and valid metadata. Current metadata editors either do not properly support the SKOS resource annotations, require server applications or make use of additional databases for metadata storage. These characteristics are contrary to the application independency and versatility of raw Unicode SKOS files, e.g. the custom text arrangement, extensibility or copy & paste editing. We provide an application that adds navigation, auto completion and validity check capabilities on top of a regular Unicode text editor.

Keywords. Metadata, classification, data visualization, semantic web, data management, health information interoperability

1. Introduction

1.1. Background

Metadata repositories are an indispensable component of data integration infrastructures and support semantic interoperability between knowledge organization systems. The underlying controlled vocabulary varies in scope, notation format, storage type and complexity. This yields two key aspects for long-term functional appropriateness: First, the ability to interconnect with other metadata vocabularies in form of mappings and translations and secondly, the maintainability of the vocabulary itself. Common standards were published to approach these issues: The ISO/IEC 11179 standard defines a schema for representing metadata. It has been extended and clarified by the ISO/IEC 21526 standard to meet the requirements of healthcare [1,2]. The World Wide Web Consortium (W3C) defined multiple standards: The Resource Description Framework

[1] Corresponding Author, Mark R. Stöhr, UGMLC, Justus-Liebig-University, Klinikstraße 36, 35392 Gießen, Germany, E-Mail: mark.stoehr@innere.med.uni-giessen.de

(RDF) provides a common framework to share and reuse data across the semantic web [3]. The Web Ontology Language (OWL) provides a language to describe ontology resources, their relations and properties expressed in RDF [4]. The Simple Knowledge Organization System (SKOS) is a lightweight data model formally defined as OWL ontology with purpose to describe thesauri and other classification schemes [5]. It is considered a compromise between poorly structured taxonomies and extensively formalized ontologies. A large collection of example SKOS datasets can be found on the W3C homepage, showing that SKOS has a wide scope of application including national libraries, social sciences, MeSH terms and drug administration forms [6]. The ISO/IEC 21526 standard explicitly "mandates the use of SKOS to provide user-interface surfaced content classification". The use of SKOS leads directly to metadata with improved compliance with three of the four FAIR principles [7]: Resources are registered with unique identifiers (Findability), data is available in a "broadly applicable language for knowledge representation" (Interoperability) and resources are annotated with preferred labels, descriptions and notations (Reusable). The fourth principle "Accessibility" can be achieved by loading SKOS resource definition files into a dedicated application, e.g. a triple store like Apache Jena Fuseki with SPARQL interface [8,9].

Implementing the SKOS standard for representing a classification scheme means to use one or more of RDF's serializations formats, the most popular ones being RDF/XML, Notation3, Turtle and JSON. We prefer Turtle, which is highly human readable and still machine parsable at reasonable costs. Since it is written as Unicode text, it can on the one hand easily be shared between people and applications and on the other hand be maintained in simple text editors. Nevertheless, especially when handling large classification schemes, the latter can lead to negative user experiences: Resource definitions and their relations become confusing, syntax errors will possibly be unrecognized until the documents are parsed by a machine. All known tools for maintaining SKOS data make use of (server) applications with additional databases and user interfaces to operate on them. Although the tools offer interfaces that allow the import and export of various Unicode formats, they thereby negate the simplicity, independency (no server needed, offline editing possible) and high flexibility (e.g. extensibility, copy & paste editing, commenting) of raw Unicode SKOS data. The decision for an editing tool depends on the target users. Conway et al. claim that in the field of medicine such tools are mostly used by physicians [10], whereas we made contrary experiences: In the domain of medical informatics, the development of classification schemes necessitates a certain amount of knowledge about metadata management and compliance with design patterns. For example, hierarchical subordinated elements may inherit context information, different catalogues in different versions need to be merged (e.g. TNM classification) and annotated codes may require accurate post-coordination. Therefore, in many cases editing is performed by data managers and medical documentalists. For such clientele, an editor should support users at their task while not impeding any Unicode text editor functionality.

In this paper, we elaborate a solution to fill a disregarded niche of Unicode SKOS classification scheme maintenance.

1.2. Requirements

Throughout over one hundred iterations of editing SKOS data and uploading it into our metadata repository, we identified various requirements for an assisting Unicode editor: (1) We need RDF syntax highlighting as well as syntax verification, (2) a presentation

of the defined SKOS hierarchy and (3) the possibility to include multiple Unicode files at once. The editor should (4) support navigation within the classification scheme: selecting a node in the concept tree or selecting a concept's reference to another concept should navigate to the respective text segment. Hovering a concept identifier should (5) offer context information about the SKOS resource, e.g. label and hierarchical classification. (6) Changes should be adopted immediately and (7) proper suggestions (auto completion) should be offered during typing. (8) We require basic semantic verification, e.g. checks for unique preferred labels, an existing English label and no loops in hierarchy. Improper text segments should be visualized for the user. For the purpose of sustainability and due to limited resources, we do not want to develop a completely new editor from scratch. Thus, we require (9) a platform independent existing text editor with extension capabilities. The editor itself should work (10) server-independently/stand-alone, which on the one hand corresponds to the proven approach of editing offline and publishing/versioning online and on the other hand prevents accessibility issues like server downtime or network problems.

2. State of the art

Although we seek a server-independent editor, we still want to take the most prominent ones and their SKOS editing capabilities into account: Protégé, developed by the Stanford University, incorporates a large toolset for complex OWL ontology maintenance [11]. Like its browser-based counterpart "Web Protégé" [12], Protégé only considers the "subclass" and "is a" relation for hierarchy evaluation. Thus, SKOS' "broader" and "narrower" relations are supported, but they will not result in a proper tree view. An SKOS plugin was developed in 2011, but it is not supported by the latest Protégé version and development has been discontinued [13]. The University of Utah developed a web-based SKOS editor, which unfortunately is no longer accessible [10]. Additionally, there are various commercial vendors like PoolParty [14] and TopQuadrant. Those focus on creating and maintaining thesauri according to the SKOS standard. They offer additional tools like text mining, graph visualization and use deep learning algorithms for content classification. Again, the actual data is either stored on a web server or in a binary database.

On the other hand, we have a large amount of text editors. To name only few popular ones, there are Atom, Notepad++, Vim and Sublime. Many of them provide an extension interface, making them potential candidates for powerful SKOS editors.

3. Concept

The most appealing text editor is the Visual Studio Code editor by Microsoft. It is free to use, has a strong community, is highly extensible, platform independent, provides built-in Git commands, the IntelliSense feature offers smart completion capabilities and our working group already made positive experiences with it. Additionally, an extension for RDF syntax highlighting and syntax verification is already available. Visual Studio Code extensions are written in typescript.

In order to build a Visual Studio Code extension that meets the requirements, we seek to make use of as many built-in features as possible. Defining SKOS classification schemes has lot in common with program code writing: Both have clear grammar rules

and include references to other text segments. Visual Studio Code offers interfaces to implement custom features for code editing, context information, validation and navigation, e.g. “Go to Reference”, search and replace, “Tree View” panels, tooltip on hover, word suggestion, diagnostics and word completion (“IntelliSense”).

We will make use of many of SKOS’ classes (“Concept”, “Collection”, “ConceptScheme”), hierarchical relations (“broader”, “narrower”, “inScheme”, “member”, “topConceptOf”, “hasTopConcept”) and properties (“prefLabel”, “notation”).

Regarding the overall architecture of our extension, the user interface consists of two panels: The actual text editor and the tree view, which depicts the SKOS resources in hierarchical order. Editing the text document will trigger the extension to parse the whole text document and update these definitions and the tree view. We will implement a button to load all turtle files from the current working directory. During text editing, the user will be offered suggestions based on the context, e.g. after typing “skos:broader” the extension will suggest all known concepts. Hovering a resource will show helpful information in a tooltip, e.g. the label and the hierarchical classification. Using the “Right Click → Go to Implementation” function as well as clicking a resource in the tree view will jump to its defining paragraph(s). Using the “Right Click → Go to Reference” function will show all text segments that reference the respective resource. We use the Visual Studio Code “Diagnostic Collection” to highlight invalid text segments with proper severity level.

For text document parsing, we decided to use regular expressions. We defined them according to the RDF grammar rules (https://www.w3.org/TR/turtle/#sec-grammar-grammar). The downside of this approach is that regular expressions do not support recursion. To solve this issue, we decided not to parse nested blank nodes. Throughout our research, we did not come across any thesaurus making use of nested blank nodes.

4. Implementation

During development, two users intensively tested the editor for usability, performance and functional appropriateness. As data basis, we made use of various publicly available Turtle SKOS metadata sets.

Figure 1 shows some of the extension’s key features: On the left side is a tree view panel, which shows all defined “Concept”, “Collection” and “ConceptScheme” instances in a hierarchical arrangement. Clicking on a tree view item will focus the respective text segment or offer a selection if more than one segments describe the item. Located on the right side is an outline panel. The middle section shows the actual text editor with active syntax highlighting. Green text sections are comments that will be ignored by the parser and can be used to take notes or to blank out definitions. After typing “skos:broader”, the editor offers a list of all available preferred labels and additionally a descriptive window showing the selected item’s hierarchical classification and all defining text paragraphs. Picking a selection will insert the selected item’s identifier.

To prove the editor’s functionality, we loaded three different SKOS datasets: (1) The “STW Thesaurus for Economics” by the Leibniz Information Centre for Economics [15], (2) the UNESCO Thesaurus [16] and (3) the German Center for Lung Research (DZL) thesaurus [17]. The first two thesauri can be downloaded and both consist of one Unicode file with approximately 3.4 megabyte and 3.2 megabyte in size. The DZL thesaurus consists of 63 Unicode files with a total of approximately 700 kilobyte. All

thesauri were parsed successfully and displayed correctly. To test the loading performance, we used a computer with an i7-8700 CPU. Loading the STW thesaurus took approximately 59 seconds, the UNESCO thesaurus approximately 20 seconds and the DZL thesaurus took less than 2 seconds. For comparison: Loading the STW thesaurus into an Apache Jena Fuseki server instance took less than one second. After every text editing, the refreshing takes the same amount of time. During that refresh process, interaction with the tree view is delayed, but text editing including suggestions still works seamlessly. Additionally, we wanted to test the extension for the "NIH NLM Value Sets" thesaurus found in BioPortal [18], but had to realize that with over 37 megabytes in size it was too large for our parser in its current version.

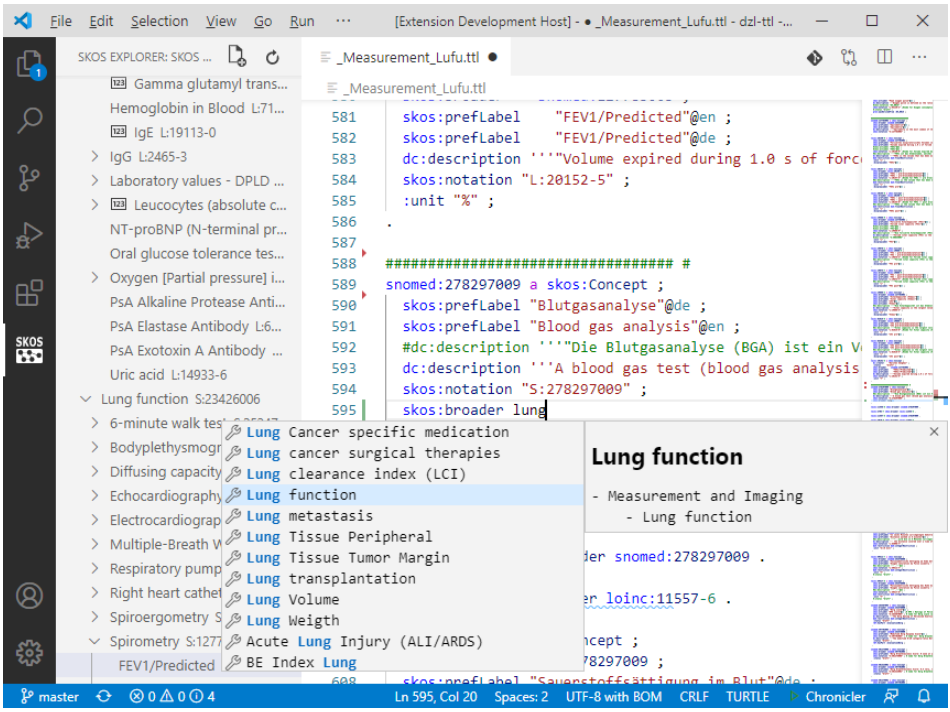

Figure 1. Screenshot of the editor's user interface. Left side: Tree view. Middle: Text editor. During text input, a suggestion window with additional context information pops up.

The implementation in its current version has been tested for usability by a medical documentalist and a medical information scientist. Both are experienced in editing text-based SKOS thesauri. They became a short introduction into the developed editor's features. For testing, the DZL thesaurus has been extended by a disease area specific catalog. The users shared their experiences in a short qualitative evaluation interview naming the advantages and disadvantages they found. Beneath the ability to navigate through the thesaurus, the users especially appreciated the semantic checks. Proper indication for duplicates and ambiguous preferred labels raised awareness for concepts that are included in more than one sub-catalog. Auto completion for known SKOS resources helped preventing typing errors. On the other hand, navigating with the "Go to Reference" and "Go to Implementation" features turned out to be less intuitive for non-software architects.

5. Lessons learned (Discussion)

The Visual Studio Code documentation helped us to develop an editor extension that meets our requirements entirely for relatively small classification schemes. For larger thesauri (up to a size of around 3 megabytes or 7000 SKOS resources) like the "STW Thesaurus for Economics" and the UNESCO Thesaurus, the functionality is assured, but maintenance is not as fluent. This is because the editor is not yet built for performance. After every change, the whole file is parsed again. Thus, refreshing is delayed and editing feels less fluent. For smaller thesauri, the user experience is positive, especially the semantic verifications have been found very useful. An additional benefit that occurred during tests was the fact, that the built-in Git module automatically indicated changes made since the last thesaurus upload. Of course, this only works, if the thesaurus is versioned with Git. When searching for datasets to test our application, we found many thesauri in RDF XML format. For more compatibility, additional parsers should be developed. The current parser only works for Turtle files and ignores nested blank nodes. During our research, we did not find any thesaurus making use of nested blank nodes. Another suggestion for future development could be an additional form-based interface for editing SKOS resources. Visual Studio Code provides so-called "Web Views" for custom HTML-based interfaces.

6. Conclusion

We identified the necessity to complement the collection of SKOS thesaurus editors with a server-independent standalone Unicode SKOS thesaurus editor. Our main purpose was to keep the versatility of raw Turtle formatted text files (e.g. custom text arrangement, extensibility, copy & paste editing, versioning) and to add navigation, auto completion and validity check capabilities on top. For thesauri divided in multiple files, the presented editor works as intended. It offers benefits for usability like clarity, validity checks and fast navigation. For larger thesauri, improvements in performance are required. The extension is publicly accessible:
https://marketplace.visualstudio.com/items?itemName=markstoehr.skos-ttl-editor

Conflict of Interest

The authors state that they have no conflict of interests.

References

[1] S.M.N. Ngouongo, M. Löbe, J. Stausberg, The ISO/IEC 11179 norm for metadata registries: does it cover healthcare standards in empirical research?, *Journal of biomedical informatics* **46(2)** (2013), 318–327, DOI: 10.1016/j.jbi.2012.11.008.

[2] International Organization for Standardization, ISO/TS 21526:2019 Health informatics — Metadata repository requirements (MetaRep) [Last accessed: 25/02/2020], https://www.iso.org/standard/71041.html.

[3] World Wide Web Consortium, Resource Description Framework, 2014 [Last accessed: 25/02/2020], https://www.w3.org/RDF/.

[4] World Wide Web Consortium, Web Ontology Language, 2012 [Last accessed: 25/02/2020], https://www.w3.org/OWL/.

[5] World Wide Web Consortium, Simple Knowledge Organization System, 2009 [Last accessed: 25/02/2020], https://www.w3.org/2004/02/skos/.

[6] World Wide Web Consortium, SKOS/Datasets, 2018 [Last accessed: 26/03/2020], https://www.w3.org/2001/sw/wiki/SKOS/Datasets.

[7] M.D. Wilkinson, et al., The FAIR Guiding Principles for scientific data management and stewardship, *Scientific data* **3** (2016), DOI: 10.1038/sdata.2016.18.

[8] The Apache Software Foundation, Jena Fuseki, 2011 [Last accessed: 14/07/2020], https://jena.apache.org/documentation/fuseki2/.

[9] W3C Semantic Web, SPARQL Protocol And RDF Query Language (SPARQL), 2013 [Last accessed: 14/07/2020], https://www.w3.org/TR/2013/REC-sparql11-query-20130321/.

[10] M. Conway, et al., Developing a web-based SKOS editor, *Journal of biomedical semantics* **7** (2016), 5, DOI: 10.1186/s13326-015-0043-z.

[11] T. Tudorache, et al., Supporting Collaborative Ontology Development in Protégé, *ISWC 2008: The Semantic Web* (2008), 17–32.

[12] M. Horridge, et al., WebProtege: a collaborative Web-based platform for editing biomedical ontologies, *Bioinformatics (Oxford, England)* **30(16)** (2014), 2384–2385, DOI: 10.1093/bioinformatics/btu256.

[13] S. Jupp, S. Bechhofer, R. Stevens, A Flexible API and Editor for SKOS, *ESWC 2009: The Semantic Web: Research and Applications* **2009**, 506–520.

[14] T. Schandl, A. Blumauer, PoolParty: SKOS Thesaurus Management Utilizing Linked Data, *ESWC 2010: The Semantic Web: Research and Applications* (2010), 421–425.

[15] Leibniz Information Centre for Economics, STW Thesaurus for Economics, 2019 [Last accessed: 25/02/2020], http://zbw.eu/stw/version/latest/about.

[16] UNESCO, UNESCO Thesaurus [Last accessed: 25/02/2020], https://skos.um.es/unescothes/.

[17] German Center for Lung Research, CoMetaR - Collaborative Metadata Repository, 2019 [Last accessed: 25/02/2020].

[18] M. Salvadores, NIH NLM Value Set as a SKOS terminology, 2015 [Last accessed: 27/03/2020], https://bioportal.bioontology.org/ontologies/NLMVS.