IMPLEMENTATION OF A CLASSIFICATION SERVER TO SUPPORT METADATA ORGANIZATION FOR LONG TERM PRESERVATION SYSTEMS

by Sándor Kopácsi†, Rastislav Hudak, Raman Ganguly

Abstract: *In this paper we describe the implementation of a classification server that helps in metadata organization for a long term reservation system of digital objects. After a short introduction to classifications and knowledge organization, the requirements of the system to be implemented are summarized. Some Simple Knowledge Organization System (SKOS) management tools we have evaluated are briefly presented. These include Skosmos, the solution we have selected for implementation. Skosmos is an open source, web-based SKOS browser based on the Jena Fuseki SPARQL server. We present the main steps of the installation of the applied tools and some potential problems with the classifications used, as well as possible solutions.*

Keywords: *long term preservation; metadata; classification; SKOS; Skosmos; Jena Fuseki*

IMPLEMENTIERUNG EINES KLASSIFIKATIONSSERVERS FÜR METADATENORGANISATION FÜR LANGZEITARCHIVIERUNGSSYSTEME

Zusammenfassung: *In diesem Artikel beschreiben wir die Implementierung eines Klassifikationsservers für Metadatenorganisation in einem Langzeitarchivierungssystem für digitale Objekte. Nach einer kurzen Einführung in Klassifikationen und Wissensorganisationen stellen wir die Anforderungen an das zu implementierende System vor. Wir beschreiben sämtliche Simple Knowledge Organization System (SKOS) Management Tools, die wir untersucht haben, darunter auch Skosmos, die Lösung, die wir für die Implementierung gewählt haben. Skosmos ist ein open source, webbasierter SKOS Browser, basierend auf dem Jena Fuseki SPARQL Server. Wir diskutieren einige entscheidende Schritte während der Installation der ausgewählten Tools und präsentieren sowohl die potentiell auftretenden Probleme mit den verwendeten Klassifikationen als auch mögliche Lösungen.*

Schlüsselwörter: *Langzeitarchivierung; Metadaten; Klassifikation; SKOS; Skosmos; Jena Fuseki*

Contents

1. Introduction

Long term preservation of digital objects is a key issue for libraries and research institutes today, because they need to ensure that the digital content of books, documents, pictures, research data, etc. remains accessible and usable within a required period of time [1]. Digital preservation includes the activities of planning, resource allocation, and application of preservation methods and technologies [2].

When we store digital objects in an archiving system, it is fundamental to assign well-defined metadata to make the discoverability of the object easier. Metadata can provide title, authors, keywords, and other important information about a document. Metadata can also store technical details on format and structure, ownership and access rights information, as well as the history of preservation activities on the digital object.

When the data provider of the digital object is allowed to add non-standardized values as metadata, it can be challenging to tell the appropriate keywords, a process that sometimes requires guessing. If we want to avoid ambiguities, misspellings, etc. it is better to select terms from pre-defined controlled vocabularies.

Controlled vocabularies, or rather classifications in multiple topics, are available in several data sources among which we can select. If we want to provide all relevant classifications inside our archiving system and make them available to users so they can select terms during upload or search, a classification server that handles vocabularies and classifications relevant to us seems to be a favourable solution. It is beneficial because we can then ensure that the required classifications are always available within a defined access time in our Classification server.

Classification servers store information according to classification or knowledge organization schemas, usually in the structure of Resource Description Framework (RDF) and/or as Simple Knowledge Organization System (SKOS), and should be organized as Linked Data.

2. Classification and knowledge organization systems

2.1. Classification

A classification is a form of categorization that collects objects or items according to their subjects usually arranged in a hierarchical tree structure. This knowledge organization technique can take many different forms, such as controlled vocabularies, taxonomies, thesauri, ontologies, and some others (see Fig. 1). There are different interpretations (e.g. [3], [4], [5]) of these types of classifications, from which we can establish the following explanations.

A **controlled vocabulary** is a closed list of words or terms that have been included explicitly, and that can be used for classification. It is controlled because only terms from the list may be used, and because there is control over who can add terms to the list, when and how.

A **taxonomy** is a collection of controlled vocabulary terms organized into a hierarchical structure by applying parent-child (broader/narrower) relationships. Each term in taxonomies is in one or more relationships (e.g. whole/part, type/instance) to other terms in the taxonomy.

A **thesaurus** is a more structured, much richer taxonomy, that uses associative relationships (like "related term") in addition to parent-child relationships.

An **ontology** is a more complex type of thesaurus usually expressed in an ontology representation language that consists of a set of types, properties and relationship types. In an ontology, there are various customized relationship pairs that contain specific meaning, such as "owns" and its reciprocal "is owned by", instead of simple "related term" relationships.

Fig. 1 shows the above mentioned concepts in order of complexity. Controlled Vocabulary is the least complex classification; all other categories are a subset of it. For example Taxonomy is a subset of Controlled Vocabulary with the additional requirement of an hierarchical structure. We can say that every Taxonomy is a Controlled Vocabulary, but not every Controlled Vocabulary is a Taxonomy, and so forth.

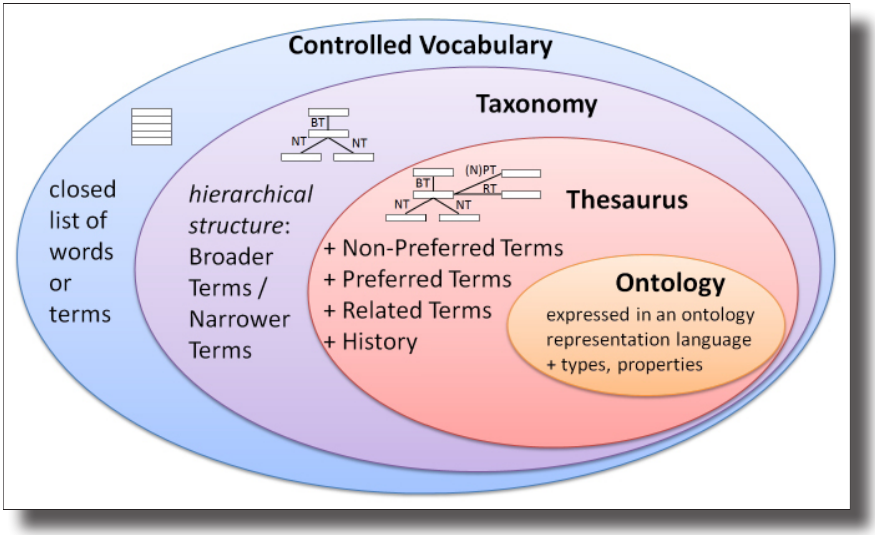


Fig. 1: Categories of classification

2.2. Knowledge organization systems and Linked Data

Classifications can be considered as a collection of organized knowledge, therefore the technical background of classification is based on Knowledge Organization Systems (KOS). In knowledge organization systems we usually store knowledge in form of triplets, such as object-predicate-subject, or object-attribute-value.

Classifications can be represented in Simple Knowledge Organization Systems (SKOS[1]) as a Resource Description Framework (RDF) vocabulary. The Simple Knowledge Organization System is a W3C recommendation designed for the representation of thesauri, classification schemes, taxonomies, subject-heading systems, or any other type of structured and controlled vocabulary.

Using RDF allows knowledge organization systems to be used in distributed, decentralized metadata applications. "Decentralized metadata is becoming a typical scenario, where service providers want to add value to metadata harvested from multiple sources." [6]

Each SKOS concept is defined as an RDF resource, and each concept can have RDF properties attached, which include one or more preferred terms, alternative terms or synonyms, and language specific definitions and notes. Established semantic relationships are expressed in SKOS and intended to emphasize concepts rather than terms/labels. [7]

A special query language, called SPARQL (a recursive acronym for SPARQL Protocol and RDF Query Language)[2], can be used to query and update data sources stored as RDF. SPARQL contains capabilities for querying required and optional graph patterns along with their conjunctions and disjunctions. SPARQL also supports extensible value testing and constraining queries by source RDF graph. [8]

SKOS – as a modern, well established standard – can (potentially) support formal alignments and hierarchical grouping of concepts using different SKOS relations (e.g. skos:exactMatch, skos:closeMatch, skos:narrower, skos:broader, skos:related), translation of concept labels, and URI-based mapping to similar concepts in other KOS.

3. Application of the classification server

The Classification server that we have integrated from available tools is an independent component of Phaidra[3], the Digital Asset Management Platform with long-term archiving functionality developed by the University of Vienna.

Phaidra is an acronym for Permanent Hosting, Archiving and Indexing of Digital Resources and Assets. Phaidra is implemented at several local Austrian institutions and also internationally, including universities in Serbia, Montenegro and Italy. Phaidra provides academic, research and management staff the possibility to archive digital objects for an unlimited period of time, to permanently secure them, to supplement them with metadata, as well as to archive objects - and to provide world-wide access to them.

We are going to apply the Classification server both during the ingestion and in the search phase of Phaidra usage. During the ingestion phase the Phaidra user uploads new items to the archiving system and can assign metadata to them from controlled vocabularies. In the search phase the user looks for objects that might have been previously supplied with classification terms from existing vocabularies. We also need the Classification server for resolving the classification terms that were saved together with the objects, when displaying them.

4. Requirements of the classification server

The first step in development was to establish requirements of the Classification server that supports different classifications and controlled vocabularies for Phaidra. The requirements were grouped into the categories of General Requirements and Technical Requirements. At this level of development we have not explicitly distinguished functional and non-functional requirements, but there are constraints that are either more functional or more of an administrative feature of the system among the General Requirements and Technical Requirements. Each requirement was prioritized between 1 and 3, where 1 meant the highest priority (= most important), while 3 meant the lowest priority (= least important).

4.1. General requirements

The General Requirements (see Table 1) are related to the main goals of the system that we were going to achieve by the implementation of the Classification server. Some of them (GR-1, GR-2, GR-3, GR-4) are functional requirements, but others (GR-5 and GR-7) are rather administrative issues.

	Requirement The Classification server	Priority 1: highest 3: lowest
GR-1	should resolve the URIs of the different terms.	3
GR-2	should support multiple languages .	2
GR-3	should support multiple versions of classifications.	1
GR-4	should return the list of sub terms (narrower concepts).	1
GR-5	should be Phaidra independent.	1
GR-6	should have no assumptions about content, which means that the set of classifications can differ on instances that are locally managed.	2
GR-7	does not require too much development effort and have low costs.	2

Tab. 1: General requirements

4.2. Technical requirements

All the Technical Requirements (see Table 2) can be considered functional requirements, and some of them (TR-1, TR-2, TR-3) are related to the input and output format of the system. TR-4 was a rather important requirement, because we definitely wanted our system to provide a SPARQL endpoint through which other systems can access our Classification server by using SPARQL queries[4].

	Requirement The Classification server	Priority 1: highest 3: lowest
TR-1	should return the terms in multiple formats (such as XML, JSON, RDF, TTL).	2
TR-2	should support standard import formats for vocabularies (e.g. SKOS/RDF, TTL, N-Triples).	1
TR-3	should support Linked Data (in SKOS/RDF/XML formats).	1
TR-4	should provide a SPARQL endpoint.	1
TR-5	should provide a comprehensive search needed for Phaidra.	1
TR-6	should also support classifications/vocabularies that do not yet support linked data (do not have URIs).	2
TR-7	should be able to use external terminology services, e.g. dewey.info, so that we do not necessarily have to import it locally.	3

Tab. 2: Technical requirements

5. Tool selection for the implementation

5.1. Evaluation of available tools

Before implementation we tested and evaluated some popular tools (ThManager[5], TemaTres[6], SKOS Shuttle[7], PoolParty[8], Protégé[9], Skosmos[10] for managing classifications. We also collected information about other tools (like HIVE[11], iQvoc[12], CATCH[13]), but they did not fit our requirements, thus they were not investigated further.

All of the evaluated tools had advantages and disadvantages, but the most important selection criterion for us was to find an open source tool that can provide a SPARQL Endpoint. A comparison of the evaluated tools can be seen in Table 3 where we included the open source products only. Another important selection criterion was to find a tool that is based on the stable and widespread Apache Jena technology and which can be accessed via REST API[14].

	Implemented in	Input	Multilingual	Backend	SPARQL Endpoint	REST API	Last update
ThManager	Apache Jena	SKOS RDF	yes	SPARQL	available	N/A	2006
TemaTres	N/A	SKOS, tabulated text	yes	MySQL	available	available	2017
Protégé	Java Swing	OWL, Excel, CSV	yes	SPARQL	available	N/A	2016
Skosmos	Apache Jena	SKOS Core	yes	SPARQL	available	available	2017

Tab. 3: Comparison of the evaluated tools

Based on the above criteria, Skosmos with Jena Fuseki seemed to be the best solution, which is why we selected it for our Classification server implementation.

5.2. Skosmos with Jena Fuseki

Skosmos[15], developed by the National Library of Finland, is an open source web application for browsing controlled vocabularies. Skosmos was built on the basis of prior development (ONKI[16], ONKI Light[17]) for developing vocabulary publishing tools in the FinnONTO (2003–2012) research initiative from the Semantic Computing Research Group.

Skosmos is a web-based tool for accessing controlled vocabularies used by indexers describing documents, and by users searching for suitable keywords. Vocabularies are accessed via SPARQL endpoints containing SKOS vocabularies.

Skosmos provides a multilingual user interface for browsing vocabularies. The languages currently supported in the user interface are English, Finnish, German, Norwegian, and Swedish. However, vocabularies in any language can be searched, browsed and visualized, as long as proper language tags for labels and documentation properties have been provided in the data.

Skosmos provides an easy to use REST API for read only access to the vocabulary data. The return format is mostly JSON-LD, but some methods return RDF/XML, Turtle, RDF/JSON with the appropriate MIME type. These methods can be used to publish the vocabulary data as Linked Data. The API can also be used to integrate vocabularies into third party software. For example, the search method can be used to provide autocomplete support and the lookup method can be used to convert term references to concept URIs. [9]

The developers of Skosmos recommend using the Jena Fuseki[18] SPARQL server and triple store with the Jena text index for large vocabularies. The Jena text extension can be used for faster text search. In addition to using a text index, caching of requests to the SPARQL endpoint with a standard HTTP proxy cache such as Varnish[19] can be used to achieve better performance for repeated queries, such as those used to generate index view.

6. Implementation of the Classification server

The classification server was implemented using Skosmos as a frontend for handling SKOS vocabularies, and Jena Fuseki as a SPARQL RDF store containing SKOS vocabulary data (see Fig. 2). The input of the system and the possible connections to the users or directly to Phaidra are discussed in the sections below.

Alternatively, instead of Fuseki, we could use other SPARQL 1.1 compliant RDF stores, but the performance of other tools did not seem to be sufficient with large vocabularies since there is no text index support for generic SPARQL 1.1.

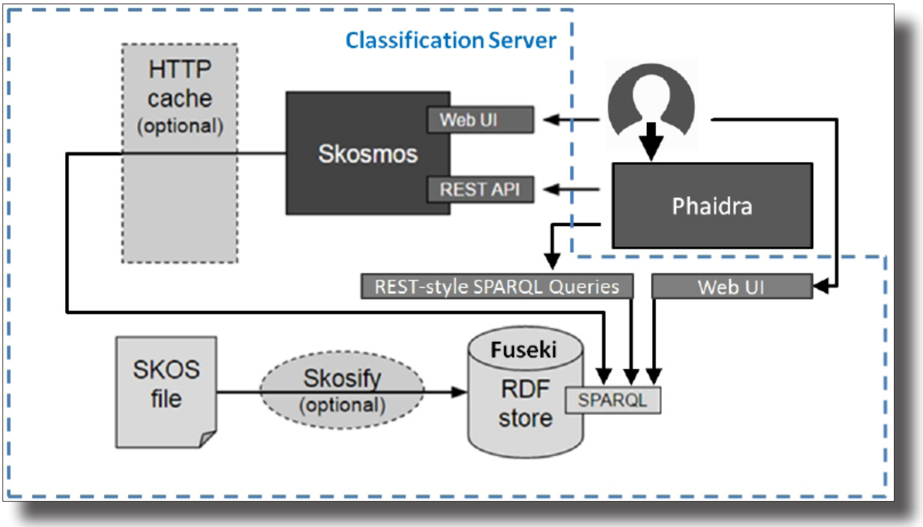


Fig. 2: System architecture (original source: [10])

6.1. Installation of Skosmos and Jena Fuseki

Skosmos and Fuseki require Apache and PHP running on the server. We have installed them on a Windows 7 environment (Professional 64 bit, Service Pack 1, Intel Core i7-56000 CPU, 2.6 GHz, 16 GB RAM) using Java 1.8 (jre1.8.0_40), with XAMPP (xampp-win32-1-8-3-4-VC11), as well as on a CENTOS 6.5 and on a Ubuntu 16.04 virtual machine (Intel Xeon CPU E5-2670 0 @ 2.60GHz).

A detailed installation guide can be found on GitHub[20] for the Linux version, but there are some deviations on the Windows version, as well as some important issues that are worth highlighting.

6.2. Configuration of Skosmos and Fuseki

Configuration of Skosmos. Skosmos can basically be configured in two files, config.inc for setting some general parameters, and vocabularies.ttl to configure the vocabularies shown in Skosmos.

In config.inc one can set the name of the vocabularies file, change the timeout settings, set interface languages, set the default SPARQL endpoint, and set the SPARQL dialect if the Jena text index is needed.

Vocabularies are managed in the RDF store accessed by Skosmos via SPARQL. The available vocabularies are configured in the vocabularies.ttl file that is an RDF file in Turtle syntax.

Each vocabulary is expressed as a skosmos:Vocabulary instance (subclass of void:Dataset). The local name of the instance determines the vocabulary identifier used within Skosmos (e.g. as part of URLs). The vocabulary instance has the following properties: title of vocabulary (in different languages), the URI namespace for vocabulary objects, language(s) and the default language that the vocabulary supports, the URI of the SPARQL endpoint containing the vocabulary, and the name of the graph within the SPARQL endpoint containing the data of the individual vocabulary.

In addition to vocabularies, the vocabularies.ttl file also contains a classification for the vocabularies expressed as SKOS. The categorization is used to group the vocabularies shown in the front page of Skosmos. You can also set the content of the About page in about.inc, and add additional boxes to the left and to the right of the front page in left.inc and in right.inc.

Configuration of Fuseki. Fuseki stores data in files. It is also possible to configure Fuseki for in-memory use only, but with a large dataset, this requires a lot of memory. The in-memory use of Fuseki is usually faster.

The Jena text enabled configuration file specifies the directories where Fuseki stores its data. The default locations are /tmp/tdb and /tmp/lucene. To flush the data from Fuseki, simply clear or remove these directories.

The Jena text extension can be used for faster text search, and Skosmos needs to have a text index to work with vocabularies of medium to large size. The limit is a few thousand concepts, depending on the performance of the SPARQL endpoint and on how much latency is acceptable for users.

If Fuseki is started in the TDB with ./fuseki-server --config config.ttl it runs using text indexing. To use Fuseki in TDB, the TDB location for Jena text index should be set, and the Lucene text directory in config.ttl. If Fuseki is run in memory with ./fuseki-server --update --mem /ds, then there is no text indexing by default.

It is also possible to use in-memory TDB and text indexing, but it requires a Fuseki configuration file (config.ttl) with special "file names" that are actually in-memory (for TDB: tdb:location "--mem--"; and for Jena text: text:directory "mem").

Timeout settings. If there is more data than Skosmos is able to handle, some queries can take a very long time. The slow queries are usually the statistical queries (number of concepts per type, number of labels per language) as well as the alphabetical index.

Short execution timeout for PHP scripts can trigger Runtime IO Exceptions. To change the timeout values, check PHP and Apache's time out settings (e.g. in php.ini the max_execution_time). It is highly recommended to find this setting and change it to a higher value (say to 5 or 10 minutes).

Skosmos also has a HTTP_TIMEOUT setting in config.inc, that should only be used for external URI requests, not for regular SPARQL queries, because there might be unknown side-effects. The EasyRdf HTTP client has a default timeout of 10 seconds. It is also recommended to change this value.

It is also recommended that users change the timeout value of their browsers from which they are planning to access Skosmos on the client-side. This is possible in Firefox and Internet Explorer, but not in Google Chrome.

6.3. Getting and setting vocabularies

The basic usage of our Classification server is to store the classifications locally (if its access time is acceptable), and we also provide the links to the remote SPARQL endpoints of the classifications if they are available.

If certain vocabularies are planned for local use, they have to be in SKOS format, and should be uploaded to the local SPARQL server, that is to Jena Fuseki.

Downloading and converting vocabularies. Vocabularies can be downloaded from the original dataset provider (e.g. from Getty, COAR, Statistics Austria, etc.), or in case of a small dataset, they can be created manually. The vocabularies need to be expressed using SKOS Core representation in order to publish them via Skosmos directly, but SKOS-XL representations or even files in Excel can also be easily converted to SKOS Core. For the SKOS-XL to SKOS Core conversion we use the owlart converter[21]. SKOS-XL labels can be converted to SKOS Core labels by executing SPARQL Update queries as well. If the classification is available in Excel or CVS, then VBA macros can convert it to SKOS Core structures. The format of the file that is accepted by Fuseki can be rdf/xml (.rdf or .xml), turtle (.ttl) or N-Triples (.nt). For SKOS files from external resources or files converted from other formats it is recommended to pre-process the vocabularies using a SKOS proofing tool, like Skosify[22]. This ensures, e.g., that the broader/narrower relations work in both directions, and that related relationships are symmetric. Skosify reports and tries to correct a lot of potential problems in SKOS vocabularies. It can also be used to convert non-SKOS RDF data into SKOS. An online version of the Skosify tool is also available, where the default options can be used after selecting the vocabulary to be checked.

Uploading files to Fuseki. If Skosmos is used for accessing classifications in the local SPARQL triple store, then the datasets have to be uploaded to Fuseki. First, it has to be considered if Fuseki will run either in memory or in a predefined folder, usually called TDB. If Fuseki runs in memory, then all uploads and updates (if allowed) will be temporary. If Fuseki runs in TDB, then uploads and updates will remain there even if we exit Fuseki and restart it.

In a SPARQL triple store there is always a default (unnamed) graph, and there can also be multiple named graphs. In other words, there is only one default graph (with no name), but there can be any number of named graphs in a SPARQL endpoint/dataset. The URI namespaces can be used as graph names (e.g. http://vocab.getty.edu/tgn/ would store Getty's TGN data).

The datasets can be uploaded to Fuseki online, when Fuseki is running, or offline, when Fuseki is not running. To upload the dataset online the control panel of the web interface of Fuseki or command line instructions can be used. For offline upload the datasets can be directly loaded to TDB.

On uploading datasets online to Fuseki through its control panel, one can set the Graph to "default" or to a graph name provided. If a graph name is used, it should be the name of its dataset in skosmos:sparqlGraph (e.g. http://vocab.getty.edu/tgn/) in vocabularies.ttl.

The Fuseki file upload handling is not very good at processing large files. It loads the dataset into memory first, and to the on-disk TDB database (and also the Lucene/Jena text index) only afterwards. It can run out of memory on the first step ("OutOfMemoryError: java heap space" is a typical error message when this happens). If we give several GB of memory to Fuseki (for example by setting JVM heap to 8 GB: export JVM_ARGS=-Xmx8000M) it should be possible to upload large (several hundreds of MB) files, although it might take a while and it is recommended to restart Fuseki afterwards to free some memory.

6.4. Some examples and problems of adding individual vocabularies

In this section we are going to describe some examples for individual vocabularies that we are using in our classification server that show typical problems and solutions.

Getty vocabularies[23] contain structured terminology for art and other cultural, archival and bibliographic materials. They provide authoritative information for cataloguers and researchers, and can be used to enhance access to databases and web sites.

Getty has its own SPARQL endpoint, but it is not responding correctly to our Classification server. There seems to be some incompatibility between Skosmos (in practice, the EasyRdf library which is used to perform SPARQL queries) and the Getty SPARQL endpoint.

Even if we could access the Getty SPARQL endpoint, it would most likely be extremely slow to use with Skosmos, since it does not have a text index that Skosmos could use. The lack of text index prevents any actual use of Skosmos with the Getty endpoint.

Therefore, we tried to upload Getty vocabularies to our own local Fuseki SPARQL endpoint with the Jena text index. But unfortunately Getty vocabularies do not work well in Skosmos due to their very large size.

There are two sets of each Getty vocabulary, the „explicit“ set and the „full“ set (Total Exports). With the „explicit“ set, which is smaller, we had to configure Fuseki to use inference so that the data store can infer the missing triples. With the full set this is not needed, but the data set is much larger so we had difficulties loading it. We could finally upload the full set of Getty's vocabularies using the tdbloader utility of Jena Fuseki.

The downloaded export file of the full set includes all statements (explicit and inferred) of all independent entities. It is a concatenation of the Per-Entity Exports in N-Triples format. Because it includes all required Inference, it can be loaded to any repository (even one without RDFS reasoning).

We had to download the External Ontologies (SKOS, SKOS-XL, ISO 25964), from http://vocab.getty.edu/doc/#External_Ontologies to get descriptions of properties, associative relations, etc. We downloaded the GVP Ontology from http://vocab.getty.edu/ontology.rdf. Finally we loaded the full.zip export files (.aat, .tgn and .ulan) from http://vocab.getty.edu/dataset/. This way we were able to make some Getty vocabularies available in our Classification server, but due to their huge size, they are rather slow.

COAR Resource Type Vocabulary[24] defines concepts to identify the genre of a resource. Such resources, like publications, research data, audio and video objects, are typically deposited in institutional and subject repositories or published in journals.

The main problem with COAR is that it only represents labels using SKOS XL properties. Skosmos currently does not support SKOS-XL. Unfortunately, the remote endpoint of COAR[25] cannot be used either, because the COAR endpoint data currently is not in SKOS Core, but in SKOS-XL. Since we wanted to use COAR data in our Classification server, we had to convert it to SKOS Core labels using owlart[26].

ÖFOS[27] is the Austrian version of the Field of Science and Technology Classification (FOS 2007[28]), maintained by Statistik Austria. The Austrian classification scheme for branches of science (1-character and 2-character) is a further development modified for Austrian data.

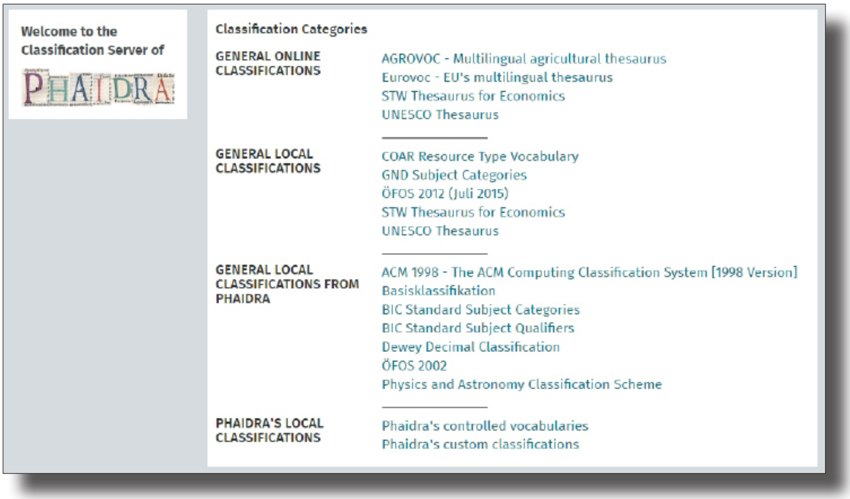
The ÖFOS can be downloaded in PDF and CSV format, but no SKOS structure (in RDF/XML, Turtle or N-Triples) or Linked Open Data representation through a SPARQL Endpoint is available.

Since we received it directly from Statistik Austria[29] in Excel format, the simplest way of converting it to SKOS was using VBA macros. These macros simply read the content of the Excel file, extend it with the appropriate RDF and SKOS labels, and write it to the desired RDF/XML or turtle format.

7. Available services and usage of the classification server

Currently our Classification server makes four general on-line classifications from external triple stores (AGROVOC, Eurovoc, STW, UNESCO), some other general local classifications (e.g. COAR Resource Type Vocabulary, GND, ÖFOS, etc.) and two local, Phaidra specific classifications available. The local classifications have been uploaded to our local triple store in order to make them accessible from Skosmos.

The Classification server is available at http://vocab.phaidra.org from any web browser. The operation of the server is quite simple: on the opening page of Skosmos (see Fig. 3) the user simply has to click on one of the classifications, and then the selected classification will be opened. First it shows the vocabulary information, like Title, Creator, Date Issued, Rights, etc. Alphabetical and hierarchical view is available for browsing the classifications. Depending on the configuration in some cases the Change History of the vocabulary or the Group of Concepts can also be seen. The user can search for specific contents either in the remote or local triple store servers or simply in a selected classification.



Welcome to the Classification Server of PHAIDRA

Classification Categories

GENERAL ONLINE CLASSIFICATIONS	AGROVOC - Multilingual agricultural thesaurus Eurovoc - EU's multilingual thesaurus STW Thesaurus for Economics UNESCO Thesaurus
GENERAL LOCAL CLASSIFICATIONS	COAR Resource Type Vocabulary GND Subject Categories ÖFOS 2012 (Juli 2015) STW Thesaurus for Economics UNESCO Thesaurus
GENERAL LOCAL CLASSIFICATIONS FROM PHAIDRA	ACM 1998 - The ACM Computing Classification System [1998 Version] Basisklassifikation BIC Standard Subject Categories BIC Standard Subject Qualifiers Dewey Decimal Classification ÖFOS 2002 Physics and Astronomy Classification Scheme
PHAIDRA'S LOCAL CLASSIFICATIONS	Phaidra's controlled vocabularies Phaidra's custom classifications

Fig. 3: Opening page of the classification server

8. Connecting the preservation system to the classification server

Our Preservation System Phaidra requires the Classification server when the user ingests new items and wants to add metadata from a controlled vocabulary. Another scenario is when the user searches for some documents classified with terms from a controlled vocabulary and wants to display or resolve them.

The connection between Phaidra and the Classification server was realized using the REST API of Skosmos and/or the REST-style SPARQL Queries of Jena Fuseki (see Fig. 1). These are read-only interfaces over HTTP to the data stored in the Classification server, where requests can be built in the URL. The returned data is in UTF-8 encoded JSON-LD format.

Skosmos provides a REST-style API and Linked Data access to the underlying vocabulary data. The REST URLs must begin with the /rest/v1 prefix. Most of the methods return the data as UTF-8 encoded JSON-LD, served using the application/json MIME type. The data consists of a single JSON object which includes JSON-LD context information (in the @ context field) and one or more fields which contain the actual data.

Jena Fuseki provides REST-style SPARQL HTTP Update, SPARQL Query, and SPARQL Update using the SPARQL protocol over HTTP. Fuseki implements W3C's SPARQL 1.1 Query, Update, Protocol and Graph Store HTTP Protocol.

9. Conclusions and future plans

In the presented work we have successfully completed our research objectives, i.e. to collect some available methods and tools for classification, with which we could implement a Classification server. The selected tools (Skosmos and Jena Fuseki) seemed to be a good choice, despite the difficulties during implementation, as well as with the upload of certain classifications.

The classification server has fulfilled the requirements that we have set up. The current stable version contains 14 internal and 4 external classifications at the moment. There have been hundreds of visits since the official launch of the Classification server, and we are receiving positive feedbacks from users continuously.

The general online external classifications (AGROVOC, Eurovoc, STW and UNESCO) are currently using the SPARQL server of the National Library of Finland, because this server is operated by Skosmos, and this way we can guarantee that these vocabularies work well from our Classification server. In the near future we are going to redirect these external requests to their original data source. The connection between the Classification server and Phaidra is still in development at the moment.

Sándor Kopácsi[†], PhD
ehem. Universität Wien, Zentraler Informatikdienst

Mag. Rastislav Hudak
Universität Wien, Zentraler Informatikdienst
E-Mail: rastislav.hudak@univie.ac.at

Dipl.-Ing. (FH) Raman Ganguly
ORCID: http://orcid.org/0000-0002-9837-0047
Universität Wien, Zentraler Informatikdienst
E-Mail: raman.ganguly@univie.ac.at

[†] Sándor Kopácsi regrettably died in August 2017, just before the publication of this issue.

References

1. Digital Preservation Coalition: Introduction: Definitions and Concepts. Digital Preservation Handbook. York, UK. (2008). http://www.dpconline.org/advice/preservationhandbook/introduction/definitions-and-concepts
2. Day, Michael: The long-term preservation of Web content. Web archiving, Springer, pp. 177–199. (2006).
3. Lars Marius Garshol: Metadata? Thesauri? Taxonomies? Topic Maps! http://www.ontopia.net/topicmaps/materials/tm-vs-thesauri.html#N429
4. Hedden Information Management: Taxonomies, Thesauri, and Controlled Vocabularies. https://www.hedden-information.com/taxonomies.htm
5. The Web Graph Database: What are the differences between a vocabulary, a taxonomy, a thesaurus, an ontology, and a meta-model? http://infogrid.org/trac/wiki/Reference/PidcockArticle
6. W3C Semantic Web: Introduction to SKOS. https://www.w3.org/2004/02/skos/intro
7. Zeng, M. L., & Chan, L.M.: Semantic Interoperability. Encyclopedia of Library and Information Sciences 4th Edition, p. 8. (2015).
8. Bob DuCharme: Learning SPARQL, 2nd Edition, Querying and Updating with SPARQL 1.1, O'Reilly Media (2013).
9. Suominen, O., Ylikotila, H., Pessala, S., Lappalainen, M., Frosterus, M., Tuominen, J., Baker, T., Caracciolo, C., Retterath, A.: Publishing SKOS vocabularies with Skosmos. Manuscript submitted for review (2015).
10. Osma Suominen: Publishing SKOS concept schemes with Skosmos. AIMS Webinar 6th April 2016, Slide 25. (2016).

Notes

1 https://www.w3.org/2004/02/skos/intro
2 https://www.w3.org/TR/rdf-sparql-query/
3 https://phaidra.univie.ac.at/
4 https://www.w3.org/TR/rdf-sparql-query/
5 http://thmanager.sourceforge.net/
6 http://www.vocabularyserver.com/
7 https://ch.semweb.ch/leistungen/thesaurus-services/en-thesauri/?ucl=en
8 https://www.poolparty.biz/
9 http://protege.stanford.edu/

10 http://skosmos.org/
11 http://onlinelibrary.wiley.com/doi/10.1002/bult.2011.1720370407/full
12 http://iqvoc.net/
13 http://www.cs.vu.nl/STITCH/repository/
14 Representational State Transfer that relies on stateless, client-server, cacheable HTTP communication.
15 http://skosmos.org/
16 http://onki.fi/en/
17 http://light.onki.fi/fi/
18 https://jena.apache.org/documentation/serving_data/
19 https://varnish-cache.org/
20 https://github.com/NatLibFi/Skosmos/wiki/Installation
21 https://bitbucket.org/art-uniroma2/owlart/downloads
22 https://code.google.com/p/skosify/
23 http://vocab.getty.edu/
24 https://www.coar-repositories.org/activities/repository-interoperability/ig-controlled-vocabularies-for-repository-assets/deliverables/
25 http://vocabularies.coar-repositories.org/sparql/repositories/coar
26 See the description of "Setting and getting vocabularies" in section 6.3.
27 http://www.statistik.at/KDBWeb/kdb.do?FAM=OESTERR&&NAV=EN&&KDBtoken=null
28 https://www.oecd.org/science/inno/38235147.pdf
29 https://www.statistik.at/web_de/statistiken/index.html