Abstract Metadata about documents, artifacts, and other objects is traditionally created by filling in metadata fields (e.g., dc:subject) with values taken from controlled vocabularies, such as keyword thesauri and classifications (e.g., LCSH). When this practice is applied to Linked Data a new problem is encountered: Linked data typically comes from different organizations and domains where mutually incompatible thesauri and classifications are used in annotations. This breaks links between the annotations, which creates data silos in linked data clouds. This paper argues that to solve the problem it is not enough to link data using primarily owl:sameAs mappings, as is customary today in, e.g., the Linked Open Data cloud, but one has to link annotation vocabularies, i.e., ontologies, into a Linked Open Ontology cloud. Since class hierarchies (using, e.g., rdfs:subClassOf) form the backbone of annotation ontologies, a key problem here is how to create and maintain a system of interlinked hierarchical ontologies so that the transitive subclass relations are not broken, when reasoning across ontologies in fundamental tasks such as query expansion and property inheritance. As a solution, we present the steps necessary for transforming thesauri into a cloud of ontologies and maintaining the system when ontologies are updated. Our approach has been used and evaluated in practice in building a cloud called KOKO of sixteen ontologies, with a total of 47,000 concepts, forming a basis for the Finnish national Linked Data architecture. KOKO has been published as an ontology service and is in use in, e.g., collection managing systems for both data indexing and semantic search.

1 Introduction

Libraries, archives, museums, and other organizations have been using classifications and thesauri for content indexing (annotation) for a long time. There exists vast amounts of high-quality annotations describing vast amounts of documents but these descriptions are divided into silos without machine-traversable links between the values used in the metadata. With the stronger need on cross-domain interconnectivity brought forth by the Web and Linked Data [12] in particular, interoperability between semantically heterogeneous data repositories of different distributed content providers has become a critical challenge for the Semantic Web. Enabling the different thesauri and vocabularies to link to one another in meaningful ways would allow the different organizations to benefit from each other’s work by enriching a common pool of linked knowledge [15].

1.1 Why a Linked Open Ontology Cloud?

The Linked Data movement\(^1\) has mainly focused its efforts on building cross-domain interoperability by creating and using (typically) owl:sameAs mappings between the different datasets in the Linked Open Data (LOD) cloud. However, when linking data, in addition to the data itself also ontologies are needed for interoperability, and this calls for more refined ontology alignment techniques [7] for concept class hierarchies than when mapping class instances only.

This paper focuses on light-weight ontologies intended for annotation. In this paper a light-weight ontology is a hierarchy of concepts with subsumption, partitive, and associative relations like in a traditional thesaurus [2]. They can

---

\(^1\) [http://linkeddata.org/](http://linkeddata.org/)
typically be represented using RDFS\textsuperscript{2}, simple OWL\textsuperscript{3} constructs, or SKOS \cite{24}.

The research hypothesis of this paper is that the LOD cloud should be complemented by developing a light-weight “Linked Open Ontology” (LOO) cloud. The idea of LOO is to provide a shared cross-domain ontology for data annotations based on a set of interlinked domain ontologies. Developing such a linked structure is in many ways different from linking datasets. A major difference between using LOD and LOO clouds is that in LOO the linked structure is used for reasoning based on the hierarchical subclass relation, the backbone of ontologies \cite{28}. This fundamental task requires special consideration at the ontology boundaries as otherwise cross-domain reasoning and ontology-based query and document expansion \cite{3,16} in applications may fail \cite{14}. 

For example, assume that the concept “Mirror” is present in a given ontology A of daily utensils and has the subclass “Make-up-mirror”:

\begin{verbatim}
a:Make-up-mirror rdfs:subClassOf a:Mirror.
\end{verbatim}

In a related ontology B of furniture, the class Mirror is used as a subclass of the class Furniture:

\begin{verbatim}
b:Mirror rdfs:subClassOf b:Furniture
\end{verbatim}

The notion of mirror looks like the same in both ontologies, i.e., \texttt{a:Mirror owl:sameAs b:Mirror}, and possibly the same URI for Mirror has already been used in both ontologies. Since the ontology A did not define make-up mirrors are furniture, the concept could have been used to correctly annotate hand-held make-up mirrors as well as, e.g., make-up mirrors integrated into a desk. Reasoning and query expansion works fine in A and B separately, but when using A and B linked together, expanding a search query for “furniture” would return falsely the hand-held make-up mirrors (in addition to pieces of furniture). Even in this fundamental case of subclass reasoning, larger contexts have to be considered when linking ontologies.

There are also other difficulties specific to developing a LOO cloud. For example, the principle of dividing a shared concept \( X \) into subclasses in different ontologies may be different. For example, the concept of “clothes” may divided into subclasses based on the gender or the age of their wearers. Then, from a human perspective, \( X \) may have a confusing mixture of subclasses in the linked ontology, hampering its use in user interfaces (e.g., as a search facet). We argue, that addressing issues like these are hard to automate, and therefore LOO development in practice requires more coordination of collaboration between the developers of linked ontologies than between the developers of linked datasets of instances. In this way better quality links can be created; various aspects of data quality are becoming a more and more critical issue of Linked Data\textsuperscript{4}. Coordinating collaboration also facilitates larger scale ontology development, which prevents the creation of interoperability problems, and minimizes redundant ontology development work in overlapping areas of ontologies.

Our approach therefore emphasizes the systematic development process of a coherent aggregated ontology from a set of component ontologies that are maintained by different domain communities. This proactive idea developing a larger ontology based on different subdomain ontologies \cite{14} is different from traditional ontology mapping, where one takes a set of existing, independently developed ontologies and tries to map them together \textit{afterwards}. In contrast to simply mapping individual ontologies together we take the mappings as an integral part of the ontologies. The ontologies are considered both as individual entities but also as integral parts of the cloud forming a greater whole. This aspect is taken into account at all levels of the development and publication of the ontologies, thus leading to a different process and underlying philosophy compared to the usual approach of linking independently developed ontologies.

1.2 A National Effort in Building a LOO Cloud

In order to test and evaluate these hypotheses in practice, a LOO cloud called KOKO of cross-domain ontologies (e.g., health, cultural heritage, agriculture, seafaring, government, defense, etc.) has been realized in Finland on a national level during the FinnONTO research project (2003–2012). Various libraries, archives, museums and governmental actors have annotated vast amounts of documents, each with their own thesauri. The aim of the KOKO cloud is to maintain backwards compatibility with the existing annotations while allowing for better interoperability between datasets. Thus, the system is based on transforming a set of legacy thesauri in use into light-weight ontologies and interlinking them with each other. Currently KOKO encompasses 16 ontologized thesauri with more to be integrated into the system in the future. The current version of KOKO is a harmonized global ontology of some 47,000 concepts aligned into a single hierarchy.

In the center of the KOKO cloud is the General Finnish Ontology YSO. It is based on the General Finnish Thesaurus YSA, which has been developed since 1987 in the National Library of Finland and used across various memory organizations in Finland. YSO provides the upper hierarchy, originally inspired by DOLCE \cite{8}, as well as the common concepts needed in many domains. YSO is then complemented by ontologies for specific domains and these domain ontologies are developed by the experts responsible for the original thesauri allowing us to preserve the know-how while allowing for asynchronous updating based on each organization’s resources. From an end-user’s point of view, KOKO ontology is seen and used as a single ontology without bound-

---

\textsuperscript{2} http://www.w3.org/TR/rdf-schema/

\textsuperscript{3} http://www.w3.org/standards/techs/owl#stds

\textsuperscript{4} Cf. e.g. http://pedantic-web.org/fops.html
aries; for the ontology developers, domain boundaries are needed in order to divide responsibilities of distributed ontology work based on domain expertise needed in different parts of the KOKO cloud.

KOKO ontology was originally published in the ontology library system ONKI\(^5\) [33,34] operating as a living lab research environment. Over the years ONKI has been integrated into, e.g., several museums, libraries, and web portals. In 2013 the National Library of Finland launched a joint project with the Ministry of Finance and the Ministry of Education and Culture to build a permanent, national ontology service Finto\(^6\) based on the ONKI system. The National Library also took on the responsibility of coordinating further development of the KOKO cloud. Finto ontology service provides a centralized publication channel for the ontologies with common interfaces for accessing them in various applications. Finnish archives are among the first to integrate the ontologies via Finto into their common search registry service for metadata on both digital as well as conventional documents with the AHAA project [18] that is being deployed in year 2014.

1.3 Challenges in Developing a Linked Ontology Cloud

Several issues need to be taken into account when moving from developing individual thesauri into developing and maintaining a cloud of interlinked ontologies. In this paper the following challenges are discussed.

- **Creation of ontologies.** How is an existing legacy thesaurus transformed into an ontology? How is a new ontology mapped into the linked ontology system? How are the URIs of the concepts formed?
- **Development of ontologies.** How are the overlapping parts of ontologies recognized in order to minimize the duplicate work of the ontology developers? How are the changes in one ontology communicated to other related ontologies? How are the errors and other quality issues recognized in a system of several ontologies?
- **Publication of ontologies.** How should the linked ontology system be presented to the end-user in order to make it comprehensible? What kind of services for using the ontologies are needed for different user groups?

1.4 Structure of the Paper

This paper is divided into four main parts. Section 2 presents a model for creating and updating a linked ontology cloud and lists a set of seven principles guiding the process. The following Sections 3–5 describe the development cycle of the cloud in more detail with an emphasis on how a single domain ontology is handled by the process. Throughout, the KOKO cloud provides an illustrative use case on how the process has been applied to practice. Since traditional, comparative evaluation of the process is difficult, the extensive application to practice has been used as a proof of concept with the process being adjusted based on real life experiences. The main user groups for this have been, on one hand, the ontology developers and on the other hand, the systems that KOKO has been integrated into. Finally, contributions of the paper are summarized and related work presented in Section 6.

2 A Model for Managing a Linked Ontology Cloud

Our ontology development work started by a field study on how thesauri in use are actually developed. The result was that thesauri are typically developed by independent expert groups focusing on concepts in their own domains of interest, with little collaboration between the groups. The situation seems to be more or less similar in other countries, too. Obviously, this model leads to redundant work in developing overlapping areas of thesauri and, at the same time, to interoperability problems between the thesauri, since different parties define their concepts without considering each others’ choices. To address these problems we propose a more coordinated collaborative model for developing a linked ontology cloud, which is depicted in Fig. 1. Note that in the following discussion the bolded numbers in parentheses refer to the numbered parts in the figure.

2.1 Ontology Creation Phase

First, existing thesauri (1) and ontologies (2) are selected for building blocks of the ontology cloud. A thesaurus is converted into RDF format using a shared ontology schema (3) and aligned with a general upper ontology (GUO) (4). Aligning domain ontologies with a GUO forms the basis for interoperability by providing a complete concept hierarchy and is much easier to maintain than direct, pair-wise mappings between domain ontologies [11]. This idea was suggested, e.g., by the IEEE Standard Upper Ontology (SUO)\(^7\) working group.

The alignment can be done in a semi-automatic fashion by first generating equivalence mappings automatically and then correcting them manually. In addition to equivalence mappings, subsumption and partitive relations might be used in the case of light-weight ontologies. For example, the concept “antique furniture” in a museum domain ontology may be aligned with the concept “furniture” in the upper ontology using a subclass-of relation. In order to create

---

\(^5\) http://onki.fi/

\(^6\) http://finto.fi/en/

\(^7\) http://suo.ieee.org/
a complete, fully connected linked ontology, all concepts of a domain ontology should be mapped to the GUO either directly or through other concepts in the domain ontology. The transformation is discussed in more detail in [14].

In our case study, a natural basis for a GUO was the General Finnish Thesaurus YSA that was transformed into the General Finnish Ontology YSO. YSA is developed by the National Library in Finland and was used already as a reference in many specialized thesauri.

2.2 Ontology Integration Phase

The ontology integration phase begins after the domain ontologies have been aligned with the GUO (5). There may be mutually overlapping parts between the domain ontologies because the alignments were made between the domain ontologies and the GUO only. To facilitate the integration of domain ontologies (6), processes, and tools for discovering overlapping parts of the ontologies are needed. Based on the analysis, it is possible to eliminate redundant development work by deciding between domain ontology developers which ontologies maintain which overlapping parts.

Changes in the GUO create pressure for the domain ontologies to be updated accordingly to ensure the consistency of the cloud (7). For example, in our case study system, some 2,000 changes are made annually in the upper ontology YSO. The changes should be taken into account in the development process of the domain ontologies. Fortunately, not all changes in the GUO are relevant for all domain ontologies, but only those related to them via equivalency, subsumption or partitive relations.

2.3 Cloud Publication Phase

After a development cycle of an ontology cloud has been completed, its logical consistency and other quality aspects should be validated (8) making sure that the resulting ontology adheres to the constraints of the properties and classes used. Some of the problems encountered can be fixed automatically [32] (e.g., overlaps in disjoint semantic relations or cycles in the concept hierarchy) but they should nonetheless be communicated to the developers for fixing at the source. However, automatic validation has difficulty in finding problems on the semantic level, which usually requires manual checking. Finally, the ontology cloud can be published as services for humans and machines, e.g., via user interfaces, APIs, and downloadable files (9).

2.4 Principles for Building a Linked Ontology Cloud

We have applied this model to creating a LOO cloud based on 16 ontologized thesauri of various domains (cf. Table 1). Below, some lessons learned during the work are summarized as a suggestion for a list of practical working principles. We consider the proposed principles novel, as we are not aware of previous ontology design patterns focused on managing an ontology cloud as a whole.

I The ontology cloud consists of one general upper ontology and several domain ontologies that are linked to the upper ontology with subsumption, equivalence, associative and partitive relations. 
Reason: This means that the domain ontologies do not have to be linked to each other pairwise, as the upper ontology acts as a semantic glue for joining all the ontologies together. Minimizing the links between the domain ontologies simplifies their development since a given developer needs to only concern herself with her own domain ontology and the GUO.

II Every concept in a domain ontology has a subsumption or equivalency relation to a concept in the GUO or a subsumption relation to a concept in the same domain ontology. 
Reason: This means that every concept in a domain ontology needs to be able to trace a subsumption relation to a concept in the general upper ontology. This ensures a consistent concept hierarchy for the whole cloud and that domain ontologies can not define new top-level concepts.
<table>
<thead>
<tr>
<th>Name of the ontology</th>
<th>Domain</th>
<th>Number of concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>YSO</td>
<td>General upper ontology (GUO)</td>
<td>27,200</td>
</tr>
<tr>
<td>AFO</td>
<td>Agriculture and forestry</td>
<td>7,000</td>
</tr>
<tr>
<td>JUHO</td>
<td>Government</td>
<td>6,300</td>
</tr>
<tr>
<td>KAUNO</td>
<td>Literature</td>
<td>5,000</td>
</tr>
<tr>
<td>KITO</td>
<td>Literary research</td>
<td>850</td>
</tr>
<tr>
<td>KTO</td>
<td>Linguistics</td>
<td>900</td>
</tr>
<tr>
<td>KULO</td>
<td>Cultural research</td>
<td>1,500</td>
</tr>
<tr>
<td>LIITO</td>
<td>Economics</td>
<td>3,000</td>
</tr>
<tr>
<td>MAO</td>
<td>Museum artifacts</td>
<td>6,800</td>
</tr>
<tr>
<td>MERO</td>
<td>Seafaring</td>
<td>1,300</td>
</tr>
<tr>
<td>MUSO</td>
<td>Music</td>
<td>1,000</td>
</tr>
<tr>
<td>PUHO</td>
<td>Military</td>
<td>2,000</td>
</tr>
<tr>
<td>TAO</td>
<td>Design</td>
<td>3,000</td>
</tr>
<tr>
<td>TERO</td>
<td>Health</td>
<td>6,500</td>
</tr>
<tr>
<td>TSR</td>
<td>Working and employment</td>
<td>5,100</td>
</tr>
<tr>
<td>VALO</td>
<td>Photography</td>
<td>2,000</td>
</tr>
</tbody>
</table>

Table 1 The ontologies comprising the LOO cloud KOKO

III If a concept in a domain ontology has an equivalency to a concept in the GUO, it may not have broader concepts in the domain ontology, which lack an equivalency relation to a concept in the GUO.

**Reason:** This is needed to avoid having dependencies from the GUO towards a domain ontology by forbidding domain ontologies from introducing broader concepts to concepts in the GUO (through inference over equivalency relation). Otherwise domain ontology developers could propagate contradictions or other unwanted concept (re)definitions into the GUO, especially in cases where more than one domain ontology is involved.

IV A concept in a domain ontology may not have an equivalency to a concept in another domain ontology. A concept in a domain ontology may have an associative relation or have a broader concept in a another domain ontology at the discretion of the developers.

**Reason:** This means that the developers need to monitor the changes to the other domain ontology but this is allowed since it does not affect the other domain ontology directly. The use of broader and associative relations extends the target domain ontology but does not affect its semantics (strongly), whereas the equivalency relation would possibly introduce new broader concepts to concepts in the target ontology and thus redefine their semantics.

V A domain ontology is focused on a clearly bounded domain and is as self-contained as possible.

**Reason:** This allows the domain ontology developer to concentrate on the area of her expertise. This also minimizes dependencies between domain ontologies and facilitates ontology development work.

VI The GUO and the domain ontologies use a shared ontology schema and are based on domain-independent standards.

**Reason:** Thus the ontologies can be easily merged and used together with various datasources outside of those directly linked to the cloud, using standard software, e.g., SKOS or OWL tools.

VII The resulting ontology cloud should be logically consistent, e.g., by ensuring the integrity of the concept subsumption hierarchy over ontology boundaries (since transitivity is assumed).

**Reason:** This allows for reasoning and query expansion over the whole cloud.

3 Creating a Thesaurus-based Ontology for the Linked Ontology Cloud

The creation of a cloud of linked ontologies begins with the formation of the general upper ontology. This ontology provides a completely connected hierarchy of general concepts including the topmost division, e.g., between abstract, endurant and perdurant concepts [8]. This forms the basic structure for the domain ontologies to map into thus saving them from having to repeat the higher parts of the hierarchy.

![Fig. 2 Ontology Creation Phase](image-url)
format into RDF, typically SKOS [24]. This is normally a straightforward operation but in some cases the original thesaurus can include relations that are not easy to convert to SKOS. In these cases it can be a good idea to retain the original relations in the form of a temporary predicate which can then be harmonized in the publication phase of the process.

In the FinnONTO case we used ad-hoc scripts for the transformation since the domain ontologies were in different formats. We also transformed them originally into a custom OWL-based format since at the beginning of the project SKOS had not yet been established as a standard way of representing light-weight RDF ontologies. We continued this practice in part because of existing tools, such as Protégé8, but also because some thesauri used relations not present in SKOS. An example of this was the Agriforest9 thesaurus of agriculture and forestry, which uses different relations to differentiate between names of concepts that were derived from different sources. These were preserved for the benefit of the ontology developers but were then combined into a common relation for publication.

The next step of the process is the automatic mapping (4) to the GUO (1). This can be done simply through string comparison between labels or by also utilizing the structure of the ontologies. The decision between the two is mostly dependent on the nature of the original thesaurus. Since different thesauri may have different labeling conventions (for example plural vs. singular) some sort of stemming or lemmatization can also be needed for the label comparison. The result is a preliminary mapping which needs to be checked manually in the next part (5 and 6) by a domain expert ontologist since the aim is to produce as good and reliable a result as possible. Aside from checking the mapping, the development part also entails the ontologization work proper, which needs to be done when changing from terms into concepts [14,20]. Each term in the original thesaurus becomes one or more concepts in the ontology depending on whether the term has multiple meanings or not. When a single term corresponds to several concepts, we have added some kind of definition in parentheses to the preferred label for each concept in order to make it easier to differentiate between them.

In the FinnONTO project we did the preliminary mapping between a domain ontology and the GUO by label comparison using lemmatization and custom scripts while most of the actual ontologization work was done by the experts from the organizations that maintained the thesauri. Now there exist specialized mapping tools, such as SILK10, which could be utilized for the initial mapping.

Finally, the URI scheme of the ontology needs to be considered. Human-readable URIs pose difficulties in that they are language-dependant and, most importantly, in the case where further development ends up changing the label of the concept, the persistent URI can not keep up with the changes thus resulting in inconsistencies. This is especially awkward if the label is then used for a different concept. Therefore the use of non-human-readable URIs is encouraged, e.g. with the local name consisting of a letter and a string of numbers, and there needs to be a system (7) tracking the use of these URIs. This is especially pertinent when concepts are deleted due to developmental decisions or simple human mistakes and new concepts are introduced.

For FinnONTO we employed our own tool called Purify11, which changes all temporary local names in a given namespace to the letter followed by a string of numbers format. It keeps a log of the mappings between the possible temporary URIs used during the early stages of the development since human-readable URIs were found to be useful at the beginning of the ontologization. Finally, Purify makes sure that no two resources get the same URI and that if the URI generation needs to be repeated, the result will be the same every time.

With the first version of the domain ontology completed (8) and successfully mapped to the GUO, the next step is to integrate it into the ontology cloud proper. Table 1 lists all the ontologies completed for our LOO cloud KOKO as of summer 2014. The name of the ontology is followed by the number of concepts and a description of the domain of the ontology.

4 Maintaining a Cloud of Interlinked Ontologies

There are two main components to the integration of the domain ontologies to the cloud. One is concerned with managing the domain ontologies with respect to one another and the other deals with the changes imposed by the developments of the GUO. This phase is depicted in Fig. 3, continuing from Fig. 2 and is explained in depth below.

4.1 Avoiding Overlap Between Domain Ontologies

The Principles IV and V presented in Section 2 posit that in order to keep relations between domain ontologies to a minimum, the domains covered need to be precisely set. With minimal links between domain ontologies, they can be developed independently from one another, but it also means that this curation of the ontological domains is an on-going process. Overlaps can come into being especially in the border areas of two domains where a given set of concepts can not be unequivocally said to belong to either one of two different ontologies’ domains.

8 http://protege.stanford.edu/
9 http://www-db.helsinki.fi/agri/agrisanasto/Welcome_eng.html
10 http://wifo5.tis.informatik.uni-mannheim.de/bizeti/silk/
11 http://puri.onki.fi/info/
In Fig. 3, we can see that the process of discovering these overlaps between a given domain ontology (1) and the rest of the domain ontologies in the cloud (3) makes use of a tool (4) to help the domain ontology developers in finding and even analyzing the overlaps. This can be simple string matching of labels and reporting on the overlapping concepts but can also make use of the ontological structure and relations to find overlapping concepts [7].

Once an overlap between two or more domain ontologies has been discovered, a dialog (5) should be started between the developers of those ontologies. The possible end results are as follows:

a) The concepts that overlap are really the same concept and the developers of the domain ontologies and the GUO can agree that the concept is general enough to be included in the GUO. In this case the concept can be removed from the domain ontologies.

b) The concepts that overlap are really the same concept but the developers can agree that the concept is not needed in both (or all) of the domain ontologies and agree on a single domain ontology that should host the concept in question and handle changes and development further on that concept and its subconcepts. This is most common in situations where the concept is clearly in the domain of one ontology and only included in the other due to historical reasons, for example.

c) The concepts that overlap are really the same concept and the ontology developers wish for it to remain present in both (or all) of the ontologies. A note should be made that if the concept is changed or developed further, the other developers should be informed as needed.

d) The concepts that overlap are actually different concepts but might share a preferred label. In this case the labels should be differentiated from one another if possible so as to avoid confusion when the ontologies are used together.

Great care should be exercised when choosing option c) since it clashes with Principle IV and can easily lead to increasing complexity in development. This should be avoided as much as possible, since one of the main goals of the presented system is to make asynchronous development of ontologies possible and having the same concept in two domain ontologies means that both sets of developers need to agree on possible further development.

In the FinnONTO project we used simple label matching for finding the overlaps since the light-weight ontologies do not offer much structure to use as basis for the discovery task. Furthermore, identical labels pose the most difficulty for the annotators using the ontology since they would need to decide between different concepts with same names based on their place in the hierarchy. Having concepts with different labels that end up being the same is much less of a problem since, when the mistake is discovered, it is relatively easy to combine the annotations made using the duplicated concepts.

We implemented a basic tool called KOAN for comparing the labels and presenting the results in a simple format. The process found an overlap of typically 3-10% even between ontologies of seemingly very distinct domains. Table 2 shows some of the comparison results between ontologies by showing the percentage amount of overlap. So from the first row, second cell, we can see that the agriculture and forestry domain ontology AFO contains 8% of the concepts of the government domain ontology JUHO. Comparing to Table 1 we can see that even ontologies from seemingly very distinct domains can have a lot of overlap between them and maintaining this overlap in multiple places at once creates a lot of unnecessary work and can lead to inconsistencies in the transitive relations. Our aim is to implement a systematic process for the elimination of these overlaps.

After the homonymous concepts have been been given unique labels, the logical next step would be to tackle the challenges of identifying the concepts that are the same but have differing labels. To this end we intend to implement either a stand-alone tool or continue the development of the KOAN tool by integrating an overlap detection algorithm.
4.2 Keeping the Domain Ontologies Up to Date with Regards to the GUO

The second part of the cloud management process, as depicted on the right hand side of Fig. 3, is the handling of the changes in the upper ontology. As an implication of Principle II, the domain ontologies react to the changes in the upper ontology. In other words, when the GUO developers release a new version (6), the domain ontologies need to be updated.

The structure of the cloud aims to allow for asynchronous updating of the domain ontologies, which can result in long intervals between the updates of a given domain ontology. Additionally, the ontologies are often developed in different organizations and possibly using different ontology editors creating a challenge in how to communicate the changes between the developers. The two main approaches for conveying the changes are push and pull synchronization [5]. The push version propagates the changes in one ontology immediately to the other ontologies, whereas in the pull version the change listing is requested when needed by the ontology developer. The push approach is good for situations where the ontologies are updated frequently, so that the ontology developer can quickly ensure the consistency of the ontology after the changes. However, if the ontology reacting to the changes in another is updated infrequently, due to different organizations having different amount of resources into the work, it would be more preferable that the changes could be acquired when needed and not propagated immediately. A long update interval also means that the amount of changes can build up over time and when the update process of the domain ontology is started, the number of changes that need to be checked can be in the thousands.

In order to ease the work of the domain ontology developer, some kind of a tool (7) needs to be used for propagating the changes. It would also be beneficial to order or categorize the changes of the GUO somehow according to their relevance to the domain ontology in question. A set of criteria for estimating relevance should take into account the differences in ontologies and the relations between them as well as the likely changes that are going to occur in development.

For the FinnONTO project we created a change propagation tool MUTU and a set of accompanying relevance criteria as described in [27]. The basic idea is that a change in the GUO is likely to be relevant to domain ontology developers if it concerns a concept that has been directly linked to from the domain ontology (a connecting concept). If a concept in the GUO has been marked as equivalent or as a superclass to a concept in the domain ontology, any change to it is likely to be of interest to the domain ontology developers. Furthermore, if the concept hierarchy of an ancestor changes for a connecting concept in GUO, this is likely to be relevant for the domain ontology developer due to the transitive nature of the relation. If a concept is removed, rare though that is, that is deemed as interesting due to the fact that this concept could have been in use by the domain ontology users but has not been duplicated to the ontology itself. Finally, if a concept has been added that has the same label as a concept already existing in the domain ontology, this is always relevant.

MUTU simply finds out the changes between the new version of the GUO and the one that was used for the mapping of the domain ontology, lists these changes according to the type of the change and relevance and adds helper classes to the development version of the domain ontology for grouping the concepts in the ontology editor suite that need to be checked by the developer (8). MUTU also allows the domain ontology developer to configure it somewhat according to her needs based on, e.g., specific priority of the languages or blocking changes from certain properties deemed irrelevant to the domain ontology development.

5 Publishing a Linked Open Ontology Cloud

Once the individual component ontologies have been developed and the links between them have been curated, the ontology cloud needs to be published. The aim is to provide the users with a single, unified whole that can be used essentially as a single ontology like any other. The process of publishing a Linked Open Ontology Cloud is depicted in Fig. 4.

5.1 Validation, Merging and Publication

The publication phase starts when a new version of a domain ontology (1) is ready and the developer wants to publish it in the LOO cloud. The first step is to clean up the ontology for publication (2), by removing structures needed only in the development of the ontology. For example, temporary concepts that are used for grouping concepts that are under development and editorial notes for internal use of the ontology developers can be removed. Similarly, temporary ontology-specific predicates should be converted to the common schema.

In the FinnONTO case the domain ontologies were developed in a Protégé project with the general upper ontology included. To help the ontology developer to focus on the concepts of the domain ontology, the topmost concepts of the domain ontology (the ones that do not have subclass-of relation to a concept in the domain ontology) are connected with subclass-of relation to a temporary concept acting as a root concept for the ontology. This root concept is removed in the clean up process as the goal is to present the resulting ontology cloud to the end-users as a single complete hierarchy with a single root concept.
Before a new version of the domain ontology is merged into the cloud, it is validated (5) for compliance with Principle VII, e.g., by checking the logical consistency and spotting violations of best practices. The results of the validation are communicated to the ontology developer who can then fix the problems. A validator may also fix some of the problems automatically, e.g., by removing cycles in the concept hierarchy.

For validation of the ontologies we are using the Skosify tool [32], which converts RDFS/OWL/SKOS vocabularies into proper SKOS format. Overlaps in disjoint semantic relations (skos:related and skos:broaderTransitive) are checked and inconsistencies are corrected automatically by removing skos:related relations in problematic cases. In cases where a concept has more than one skos:prefLabel per language one of the labels is arbitrarily selected for use as skos:prefLabel while the others are simply converted into skos:altLabels. A check is performed in order to detect overlaps in disjoint label properties (skos:prefLabel, skos:altLabel, and skos:hiddenLabel) and the superfluous ones are removed. As a best practice, the cycles in the concept hierarchy are removed, and finally extra whitespace is removed from concept labels. The violations are also reported to the ontology developer so he may fix the problems in a controlled way instead of relying on the automatic fixing procedure. We also used Skosify to convert the ontologies from the custom OWL format to SKOS for publication.

After the validation, the new version of the domain ontology is merged (6) with the GUO (3) and the other domain ontologies (4). The idea is to build a single representation of the linked ontology cloud (7) by merging equivalent concepts and giving them a single URI. The differing ontology schemas are harmonized so that the end-user does not get confused with the ontology-specific structures and naming conventions.

When annotating using the linked ontology cloud, the annotator should be making choices between concepts as opposed to ontologies. To this end, the cloud is merged (6) into a single whole (7) by taking all the concepts linked with skos:exactMatch relations between them and combining them. In case a more clear division between the development version of individual ontologies and the published version of the cohesive cloud needs to be made, new URIs can also be assigned to the concepts of the cloud. Here care needs to be taken in order to ensure that the same concepts always map to the same URIs even in cases where that concept might at first originate from the upper ontology and later have been moved to a domain ontology or vice versa.

In FinnONTO we called this realization of the cloud KOKO and gave all the concepts new URIs in a new namespace. The combination of the ontologies listed in Table 1 resulted in a combined cloud of some 47,000 concepts. In order to allow the end-user to select only from a single domain, we assigned domain ontology-specific concept types as subclasses of the skos:Concept.

Once the ontology cloud is merged, it is ready for publication (8). In accordance to the “open” part of the name LOO, the cloud is published as Linked Data [12], so that individual concept URIs can be referenced from various data sets and URIs act as URLs serving information about the concepts in human- and machine-readable forms. The LOO cloud can be published for humans via user interfaces for searching, browsing and visualizing the ontologies, and as widgets for integrating ontologies into applications. For machine use, various APIs can be provided to facilitate even deeper integration of the ontologies. Finally, the ontology cloud is published as a SPARQL endpoint enabling ad hoc queries for more complex needs.

As part of the FinnONTO project, we built the ONKI Ontology Service[12] [34], which was further developed by the National Library of Finland in the form of Finto thesaurus and ontology service[13]. The service acts as a repository for vocabularies, thesauri, and ontologies, providing support for publishing, finding, and accessing them in a centralized fashion. The main user groups of Finto are ontology developers, content indexers and information searchers. Ontology developers need a way to visualize the ontology they are developing, and especially in context of the LOO cloud, where the domain ontology developers map their ontologies only to the general upper ontology, there is a need for getting an overview of the whole cloud. This way the developers can see how their domain ontologies are situated in the LOO cloud and discover possible overlaps with other domain ontologies.

---

12 http://onki.fi/
13 http://finto.fi/en/
Content indexers and information searchers need ways of finding ontologies and concepts suited for their needs. The Linked Open Ontology Cloud approach eliminates the need for finding ontologies and making selections between them, as one ontology system covering all domains of life aims to fulfill the needs of everyone. For finding suitable concepts, Finto provides an ontology browser which visualizes the ontologies as a tree hierarchy and shows other relations between concepts. The user may use autocompletion search for finding concepts based on their labels. In addition to dereferenceable URIs, machines are served with REST API, and a SPARQL endpoint\textsuperscript{14}. The general ontology service software powering Finto is developed by the National Library of Finland as an open source project\textsuperscript{15} and can be freely used to set up a similar service. The ontology cloud is published under a permissive Creative Commons Attribution license\textsuperscript{16}.

5.2 Use Cases

During the FinnONTO project, the adoption of KOKO was hampered by the uncertainty regarding its future. With ontology service project of the National Library of Finland, the development and publication of KOKO was also given governmental resources thus securing its future. However, due to the length of the process of securing funding, the widespread adoption of KOKO is only beginning.

One of the pilot users of KOKO has been the Swedish language division of Finland’s national public service broadcasting company, Svenska YLE. They have used KOKO in the annotation of their web content and have complemented it with Freebase\textsuperscript{17} for instance references such as people and organizations. The pilot has been a success and they are considering adopting the system onto the Finnish part of YLE as well as their archive. KOKO has also been in use in Espoo City Museum where it has been used for the annotation of museum artefacts. Again, the pilot use has been considered succesful and several other museums are considering adopting KOKO.

The multi-disciplinary nature of KOKO means that it is especially suited to organizations that deal with potentially all kinds of topics such as media organizations, museums, and archives. Furthermore, using the concepts from domain ontologies links the content to more in-depth data from the specialist organizations that originally developed the domain ontologies for their data annotation needs. KOKO is also being deployed in the project for developing a common information retrieval service for all the archives in Finland.

6 Conclusions and Discussion

We described a process for creating and managing a Linked Open Ontology Cloud. The emphasis was in developing a model for enabling the re-use of existing annotations and domain expertise in well-established thesauri.

6.1 Contributions

The realization of the Linked Open Ontology Cloud was achieved by facilitating an environment of interconnected but easily manageable set of cross-domain ontologies allowing for distributed ontology development. This model provides an alternative way of linking datasets together as opposed to the direct linking between resources inherent in the LOD cloud approach. This can be more efficient since the mappings between ontologies can be re-used for different datasets.

Furthermore we presented a detailed description of managing the overlaps between domain ontologies and the inconsistencies resulting from the asynchronous updating of the GUO compared to the domain ontologies. Finally, we implemented a LOO in practice with sixteen ontologies and a full set of tools for the development cycle from a set of thesauri to a fully realized Linked Open Ontology Cloud. Based on this work, we accompanied the LOO model with a set of seven principles that guide the process of building and managing an ontology cloud. The principles are general enough to be applied to other ontology clouds in addition to the one we have built.

A central challenge in the linking process is in maintaining the integrity of the transitive relations across different ontologies and we have achieved this through the use of a general upper ontology acting as a central hub for the linking of various domain ontologies. We consider proactive linking as an integral part of the development process itself as opposed to simply mapping independently developed ontologies.

The model was piloted by implementing it into practice in extensive scale in various organizations encompassing many different domains. The model evolved based on lessons learned during the multi-year implementation and development process. Feedback was gathered from ontology developers as well from the systems integrating the linked ontology cloud. Based on the success of the prototype, the model is now being applied on a national scale in archives, libraries, and museums, as well as in ministries and other governmental agencies.

\textsuperscript{14} http://api.finto.fi/sparql; The KOKO cloud is available in the named graph http://www.yso.fi/onto/koko/.
\textsuperscript{15} http://github.com/NatLibFi/Skosmos/
\textsuperscript{16} http://creativecommons.org/licenses/by/3.0/
\textsuperscript{17} https://www.freebase.com/
6.2 Related Work

The limitations of \texttt{owl:sameAs} type linking in Linked Data are widely recognized (e.g., [10]) caused by the strength and the symmetric nature of the statement. Linking the data through ontologies allows for deeper interoperability due to the inferred knowledge gained through the shared semantics [13]. Our approach follows this principle and provides a framework for interlinked ontology cloud development while retaining backwards compatibility with existing annotations.

A well-known example of highly linked ontologies can be found in BioPortal\footnote{http://bioportal.bioontology.org/}, an ontology repository that contains features supporting collaborative (inter-)ontology development. In addition to uploading new ontologies to BioPortal, users can also create and upload mappings between concepts of different ontologies [26]. The mappings can be used to bridge overlapping ontologies or to extend general ontologies with specialized ones. Users can comment and create discussion threads on ontologies, their parts, and mappings, thus supporting a collaborative and open inter-ontology development process. However, a set of mapped ontologies does not appear as a single whole to the user, though one can browse the ontologies by following the mappings between concepts. Moreover, the mappings are not utilized in the ontology development process to propagate the changes of an (upper) ontology to ontologies extending it, as they are in our LOO model.

Linked Open Vocabularies (LOV)\footnote{http://lov.okfn.org/dataset/lov/} is a related approach for interlinking vocabularies, concentrating on the ones used in the LOD datasets. The focus of LOV is to gather, describe, and classify the vocabularies, and to interlink them on a high level (e.g., ontology A extends ontology B). In contrast, in the LOO approach ontologies are interlinked on the concept level.

To move from a group of ontologies into a coherent ontology system, the ontologies need to be reconciled. Ontology reconciliation [11] is a broad term, covering ontology merging, alignment and integration. Most of the reconciliation methods are automatic or semi-automatic, which can lead to lower quality, especially if the ontologies were originally expert-made [6]. Our approach emphasizes the importance of manual development work in the reconciliation process.

In ontology modularization [1] ontologies are divided into smaller interlinked parts to facilitate distributed development and re-use of the ontologies. Our approach merges ontologies based on existing thesauri to form a Linked Open Ontology Cloud, while the development of the individual ontologies continues in a modularized way. Furthermore, we present the resulting interlinked cloud as single whole so that the end-users do not have to make selections between ontologies.

There have been several previous efforts on building a general upper ontology [23] that can be used as a foundational basis for domain ontologies. Some of the upper ontologies have been developed from scratch, while, e.g., the Suggested Upper Merged Ontology SUMO [25] was created by merging existing ontologies. We used the General Finnish Ontology YSO, transformed from an existing general thesaurus YSA, as the upper ontology in our Linked Open Ontology Cloud. In contrast to the previous work on upper ontologies, which focuses on the creation of an upper ontology, our work emphasizes the model for managing the domain ontologies as part of the ontology cloud and keeping them up to date and synchronized with the changes of the upper ontology.

Related to the principles of forming and managing the ontology cloud presented in this paper, similar guidelines have been presented in the context of the OBO Foundry initiative [29], however in OBO Foundry the focus is on coordinating the development of different ontologies under shared principles, but not on merging them into a single ontology cloud and the challenges therein. General, domain-independent ontology design principles have been proposed by several researchers [9,35]. Our model uses their ideas as a foundation, e.g., by organizing orthogonal concept domains into separate ontologies and supporting ontologies of different granularity levels in the form of a GUO and the domain ontologies extending it. The principles presented in this paper extend the general ontology design principles by covering modelling issues related to the management of changes in a linked ontology cloud.

Keeping domain ontologies up to date with the changes of the GUO is closely related to the topic of ontology evolution, which concentrates on dealing with the changes in ontologies over time. Different change types have been listed in [31], whereas [19] considers more abstract change patterns constructed from atomic changes. [4] mentions that users would have liked to see explicitly when a concept created by them had been modified, indicating that not all the changes occurring in an ontology are equally relevant for all developers providing the impetus for our work on building a set of priorities for different changes. The detection of changes in an updated ontology can be done using logs [30] or by comparing two versions of the ontology [21], which was the approach we chose. Additionally, extra challenges of distributed ontology development have been addressed in [19,22,17]. In our approach there is no assumption that all the ontology developers use the same ontology editor, as the support tools are implemented separately from the ontology development suite.
Acknowledgements

This work is part of the National Semantic Web Ontology project in Finland20 FinnONTO (2003–2012), funded by the National Technology and Innovation Agency (Tekes) and a consortium of 35 public organizations and companies.

References


