

# A Flexible API and Editor for SKOS

[Extended Abstract]

Simon Jupp  
School of Computer Science  
The University of Manchester

Sean Bechofer  
School of Computer Science  
The University of Manchester  
first.last@manchester.ac.uk

Robert Stevens  
School of Computer Science  
The University of Manchester

## Keywords

SKOS, OWL, Protégé

## ABSTRACT

This poster presents a programmatic interface (SKOS API) and plugin for Protégé 4 for editing and working with the Simple Knowledge Organisation System (SKOS). The SKOS API has been designed to work with SKOS models at a high level of abstraction to aid developers of applications that use SKOS. We discuss SKOEd, a tool for authoring and editing SKOS artefacts. A key aspect to the design of the API and editor is how SKOS relates to OWL and what existing OWL infrastructure can be exploited to work with SKOS.

## 1. INTRODUCTION

SKOS is an emerging standard and specification for representing and publishing classification schemes, thesauri, taxonomies and subject heading systems<sup>1</sup>. In order to support adoption and use of SKOS there is now a need for tools that can support its use in the form of APIs, editors, browsers and validators.

We chose to take advantage of existing OWL infrastructure in the form of the OWL API [?] and the OWL ontology editor Protégé 4 [?] to build an API and editor for SKOS. Building a SKOS implementation using the OWL API provides many benefits including the ability to manipulate and extend the SKOS data model, which is itself an OWL ontology. One of the major issues with this approach is the SKOS data model has components that take a SKOS vocabulary into OWL Full whilst the OWL API and Protégé 4 is restricted to the OWL 2<sup>2</sup> sub language. To handle this we propose an OWL 2 model for SKOS that covers all the major functionalities of SKOS and has the benefit of safe interoperation of SKOS with OWL 2 ontologies.

## 2. API DESIGN

Many Knowledge Organisation System (KOS) authors may prefer not to be exposed to the underlying technical details such as RDF and OWL. We suggest that a SKOS data model that provides an appropriate abstraction from the technical issues of concrete representations is of benefit. This approach proved successful in designing an API for OWL [?]

<sup>1</sup><http://www.w3.org/2006/07/SWD/>

<sup>2</sup><http://www.w3.org/TR/owl2-syntax/>

and subsequent tools such as an OWL Validator [?] and editor, Protégé-OWL<sup>3</sup>. Such tools are of benefit to OWL users, and the aim is that building a suite of tools for working with SKOS will have similar benefits.

A key feature of the SKOS API is to provide access and support for extending the OWL data model used to describe SKOS, and to have built in support for interpreting the SKOS semantics. Many tools exist for working with RDF, including APIs<sup>4</sup> and graphical editors<sup>5</sup>, however, these often operate at too low a level, exposing the underlying RDF structures which may be unfamiliar or unintuitive to KOS developers. Tools exist for building KOS, yet few have direct support for SKOS or provide any programmatic interface to the model. With these considerations in mind, we built an API and editor by exploiting current OWL tools. This was achieved by extending an existing API, the OWL API, which is already designed to work with OWL at a high level of abstraction.

## 3. API FUNCTIONALITY

- **Parsing and Rendering:** We take advantage of the OWL API serialisation and parsing mechanisms which supports handling of multiple concrete syntaxes for SKOS. For the most part RDF/XML will be the default and the most common.
- **Change control:** Manipulation of a SKOS via the addition and removal of objects and assertions. This includes a mechanism for tracking changes that are made to a KOS.
- **Extension to the model:** The OWL API provides interfaces for manipulating the underlying OWL ontology and SKOS model, our implementation provides support for using the OWL API objects with the SKOS API.
- **Inference:** We can exploit the power of DL reasoners to build an inferred model of a SKOS data set. The API provides support for working solely with an asserted model, or querying an inferred model.

## 4. SKOED

We have a basic list of functional requirements for SKOEd;

<sup>3</sup><http://protege.stanford.edu/>

<sup>4</sup><http://jena.sourceforge.net/>

<sup>5</sup>[http://www.altova.com/products/semanticworks/semantic\\_web\\_rdf\\_owl\\_editor.html](http://www.altova.com/products/semanticworks/semantic_web_rdf_owl_editor.html)

- Ability to create, edit and delete SKOS entities such as Concepts, Concept Schemes etc. . .
- Assert SKOS relationships between the SKOS entities, for example `skos:inScheme`, `skos:hasTopConcept`.
- Visualise the broader/narrower hierarchy in the form of a navigable tree.
- Assert and edit data properties such as `skos:prefLabel`, along with support for multilingual labeling.
- Support for meta-data such as SKOS documentation properties and other standard vocabularies like Dublic Core.
- Provide alternate renderings in the editor, especially for multilingual thesauri.
- Access to the SKOS data model and support for extending it.
- Query interface that supports SKOS entailments.
- Change tracking, the ability to undo/redo actions and track those changes.
- Ability to parse and render a range of ontology formats, including RDF/XML.

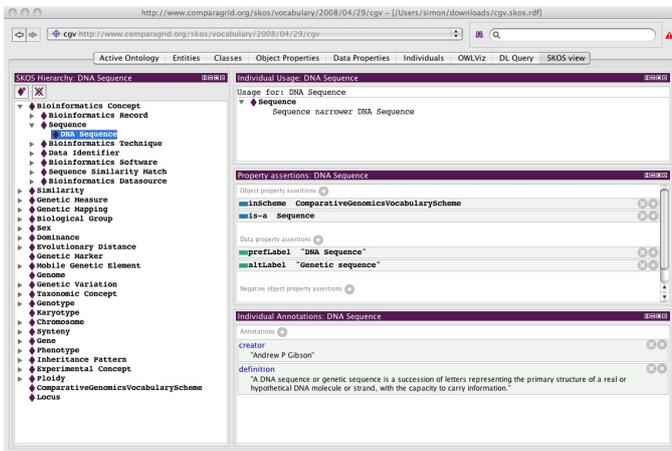


Figure 1: A screenshot of SKOSEd plugin in Protégé. The left hand panel shows the concept hierarchy, the right hand panel shows asserted information and meta-data about the selected Concept.

SKOSEd (See Figure ??) is a plugin for the OWL ontology editor Protégé 4 that has begun to fulfil these requirements. SKOSEd’s purpose is to provide support for working with SKOS. The decision to use the Protégé environment as a platform for SKOSEd follows on from our decisions and justifications for using the OWL API as a basis for the SKOS API. As SKOS has an OWL T-box, a standard installation of Protégé is already a capable SKOS editor, however, the interface is not necessarily suitable or intuitive to KOS authors. SKOSEd provides a suite of *Views*, that expose the structure of SKOS, along with some support for editing and authoring.

## 5. DISCUSSION

In this poster, we have presented an API for working with SKOS, along with a graphical editor SKOSEd, implemented using the API. We discussed the requirements for such tools and some justifications for reusing existing OWL infrastructure to provide support when working with SKOS. We highlighted two key issues that we believe are desirable when

working with SKOS; 1/ the ability to extend the SKOS model via an API or editor; and 2/ the exploitation of DL reasoners to compute entailments for richer querying of data represented in SKOS. By reusing an existing API (the OWL API) and ontology editing software (Protégé 4), we were able to rapidly develop early tool support for the emerging SKOS standard.

By adopting the Protégé framework we hope that the existing plugin will be extended with additional plugins provided by the community, to improve the way we work with not only SKOS, but combinations of both the SKOS and OWL languages. The SKOS API provides a flexible set of interfaces for working with SKOS, along with a concrete implementation using the OWL API. The API is independent from the editor and additional tools are planned, including a re-implementation of the interface to provide some persistent storage mechanism, such as a database backend, or extensions to support alternate querying interfaces, such as SPARQL. Additional plans include the use for the SKOS API as a basis for SKOS validation services.

SKOSEd is available as open source<sup>6</sup>. We encourage the reader to download and use the application and welcome further comments or suggestions for enhancements.

**Acknowledgements:** We would like to acknowledge Matthew Horridge for his initial work on the SKOSEd plugin and valuable expertise on the OWL API and Protégé 4. SJ is supported by the Sealife project (IST-2006-027269).

## 6. REFERENCES

- [1] *Proceedings of the World Wide Web Conference, WWW2004*. ACM Press, 2004.
- [2] S. Bechhofer and J. J. Carroll. Parsing OWL DL: Trees or Triples? In *Proceedings of the World Wide Web Conference, WWW2004* [?], pages 266–275.
- [3] S. Bechhofer, R. Volz, and P. Lord. Cooking the Semantic Web with the OWL API. In *2nd International Semantic Web Conference, ISWC*, volume 2870, 2003.
- [4] M. Horridge, S. Bechhofer, and O. Noppens. Igniting the OWL 1.1 Touch Paper: The OWL API. In *Proceedings of OWLEd 2007: Third International Workshop on OWL Experiences and Directions*, 2007.
- [5] M. Horridge, D. Tsarkov, and T. Redmond. Supporting early adoption of owl 1.1 with protégé-owl and FaCT++. In *OWLED 2006 OWL: Experiences and Directions 2006*, 2006.

<sup>6</sup><http://code.google.com/p/skoseditor/>