Proceedings of the Workshop "Ontologies come of Age in the Semantic Web" (OCAS2011)

10^{th} International Semantic Web Conference

Bonn, Germany, October 24, 2011

edited by Ken Baclawski, John Bateman, Alexander García Castro, Christoph Lange, and Kim Viljanen

October 24, 2011

Preface

This volume contains the papers presented at OCAS2011 (`http://ocas.mywikipaper.org`), the Workshop "Ontologies come of Age in the Semantic Web" held on October 24, 2011 in Bonn, Germany.

There were 17 submissions. Each submission was reviewed by at least 2, and on the average 3.6, program committee members. The committee decided to accept 12 papers, 9 of which were presented during the workshop.

The program also included one keynote, given by Deborah McGuinness.

We would like to thank

- our peer reviewers for carefully reviewing the submissions and giving constructive feedback
- our keynote speaker for giving the audience further inspiration on the question whether ontologies have come of age
- the sponsors of the OCAS challenge for their prizes
- EasyChair for their reviewing and proceedings generation support
- CEUR-WS.org for publishing the workshop proceedings

November 16, 2011
Bremen

Ken Baclawski
John Bateman
Alexander García Castro
Christoph Lange
Kim Viljanen

Program Committee

Ken Baclawski	VIStology, Inc.
John Bateman	University of Bremen
Oscar Corcho	Universidad Politécnica de Madrid
Mike Dean	Raytheon BBN Technologies
Li Ding	Rensselaer Polytechnic Institute
Alexander Garcia	Postdoctoral fellow
Leyla Garcia	Universitaet der Bundeswehr
Raul Garcia	Universidad Politecnica de Madrid
Benjamin Good	The Genomics Institute of the Novartis Research Foundation
Michael Gruninger	University of Toronto
Peter Haase	fluid Operations
William Hogan	University of Arkansas for Medical Sciences
Matthew Horridge	The University of Manchester
Michael Kohlhase	KWARC
Oliver Kutz	University of Bremen, SFB/TR 8 Spatial Cognition
Christoph Lange	Jacobs University Bremen
Riichiro Mizoguchi	University of Osaka
Fabian Neuhaus	NIST
Raul Palma	Poznan Supercomputing and Networking Center
Carlos Pedrinaci	Knowledge Media Institute, The Open University
Steve Pettifer	The University of Manchester
Nigam Shah	Stanford University
Carlos Toro	Vicomtech
Jouni Tuominen	Aalto University School of Science
Kim Viljanen	Aalto University, School of Science
Boris Villazón-Terrazas	Universidad Politecnica de Madrid

Additional Reviewers

L

Le Pendu, Paea
N

Normann, Immanuel

Contents

Generic Multilevel Approach Designing Domain Ontologies based on XML Schemas

Thomas Bosch[1] and Brigitte Mathiak[2],

[1] GESIS - Leibniz Institute for the Social Sciences, Square B2, 1,
68159 Mannheim, Germany
[2] GESIS - Leibniz Institute for the Social Sciences, Lennéstr. 30,
53113 Bonn, Germany
{Thomas.Bosch, Brigitte.Mathiak}@gesis.org

Abstract. Designing an ontology for a specific domain is a time-consuming process. In many cases, information sources like XML Schemas serve as a basis for ontology engineers to conceptualize the intended ontologies. The ontology design process is sped up significantly when XML Schemas are transformed automatically into generated ontologies. An XML Schema Metamodel Ontology has been designed to represent the components of the XML Schema abstract data model. The generated ontologies' classes are defined as sub classes of this ontology. The classes specified for the generated ontologies are intended to be further supplemented with additional semantic and domain specific information defined in domain ontologies. The resulting ontologies are as usable as ontologies that were constructed completely manual, but with a fraction of necessary effort.

Keywords: Semantic Web, Ontology Design, XML Schema

1 Introduction

XML has reached wide acceptance as a data exchange format in e-business. Data and metadata structured by ontologies can be published in the increasingly popular LOD cloud to get linked with a huge number of other RDF datasets [1]. As RDF is an established standard there is a plethora of tools which can be used to interoperate with data and metadata represented in RDF. An effective and efficient cooperation between e-business partners is only possible if they agree on a common syntax and have a common understanding of the domain classes. XML Schema and OWL support differing modeling goals. The data model of XML describes a node labeled tree [2], the syntactic structure of XML document instances. OWL, however, is based on the subject-predicate-object triples from RDF [3], based upon formal logic, and describes semantic information about domain classes as well as their relations and therefore allows the sharing of conceptualizations. XML represents a large set of information in many domains. This fact has driven the development of general-purpose tools for converting XML Schemas to OWL ontologies. The direct mapping from XML and XML Schema to RDF and OWL is not sufficient, since it only

transports information about the syntactic structure of XML document instances. Semantic information has to be added in a further step. The aim of this paper is to bridge the gap between XML and OWL by lifting the syntactic level of XML documents to the semantic level of OWL ontologies. The process of designing domain ontologies is extremely time-consuming. XML Schemas describing specific domains are often existent in early stages of the ontology design process. In this paper, the authors describe a generic multilevel approach which accelerates the process of designing domain ontologies from scratch based on already available XML Schemas. The intention is to create generated ontologies automatically based on any possible XML Schemas of an underlying domain data model using XSLT transformations. Initially defined generated ontologies are linked to an ontology of the appropriate domain used to specify supplementary semantic information not covered in the XML Schemas. Domain experts enrich the domain ontology with additional semantics needed for tasks typically performed in the particular domain.

2 Designing Domain Ontologies based on XML Schemas

Figure 1 sketches the devised underlying concept of the generic multilevel approach for designing domain ontologies based on XML Schemas.

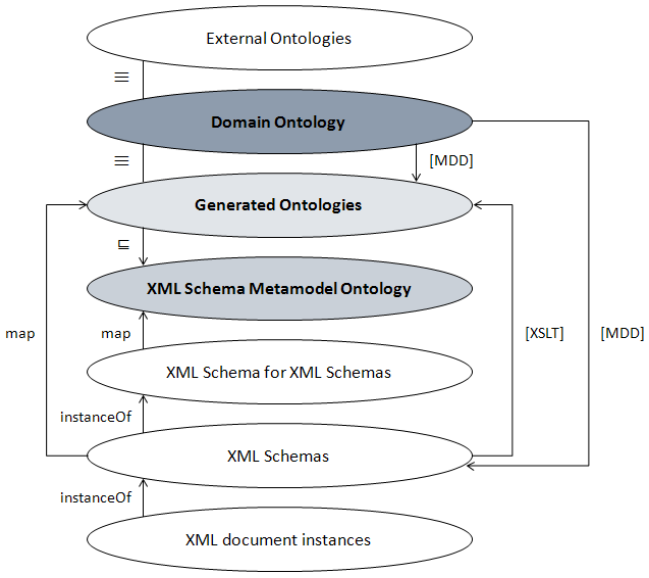

Fig. 1. Generic multilevel approach for designing domain ontologies based on XML Schemas

XSLT transformations map any XML Schemas to generated ontologies automatically. The XML Schema Metamodel Ontology serves as a basis for this process. Domain ontologies are related to generated ontologies in order to append semantic information not expressed in the XML Schemas. Further, you may integrate external ontologies'

semantics. The relationships between the separate levels of XML and between the distinct ontologies are delineated. You can derive generated ontologies and corresponding XML Schemas simultaneously, automatically and model-driven from the data model of the domain ontologies. The ensuing paragraphs present the different levels of XML, the individual ontologies and their relationships in more detail.

2.1 XML Schema and the XML Schema Metamodel Ontology

XML [4] documents are commonly used to store and transfer information in distributed environments. XML documents may be instances of XML Schemas [5] determining their terminology and syntactic structure. The W3C has defined XML Schema, the class of XML documents, recursively using the XML Schema language to describe the XML Schema language [6], just like XML Schema documents are XML documents describing XML documents. Generated ontologies are based on the components of the XML Schema abstract data model, the meta-model of XML Schema. Table 1 outlines the mappings between the XML Schema meta-model and the XML Schema Metamodel Ontology. In order to visualize OWL language constructs, Description Logic syntax is used.

Table 1. Mapping of the XML Schema meta-model to the XML Schema Metamodel Ontology

XML Schema for XML Schemas	XML Schema Metamodel Ontology
meta-element information items	classes: <meta-element information item>
attributes of meta-element information items	datatype properties and associated universal restrictions: <domain meta-element information item> ⊑ ∀<attribute>_<domain meta-element information item>_String.String
texts contained in meta-element information items	datatype property and associated universal restriction: <domain meta-element information item> ⊑ ∀valueXSD_<domain meta-element information item>_String.String
texts contained in XML document instances' components	datatype property and associated universal restriction: <domain meta-element information item> ⊑ ∀valueXML_<domain meta-element information item>_String.String
attributes of meta-element information items referring to meta-element information items	object properties and associated universal restrictions: <domain meta-element information item> ⊑ ∀<attribute>_<domain meta-element information item>_<range meta-element information item>.<range meta-element information item>
attributes 'type' and 'base'	object properties and associated universal

	restrictions: <domain meta-element information item> ⊑ ∀type|base_<domain meta-element information item>_Type.Type
meta-element information items' part-of relationships	object properties and associated universal restrictions: <domain meta-element information item> ⊑ ∀contains_<domain meta-element information item>_<range meta-element information item>.<range meta-element information item>

The authors have mapped the meta-element information items corresponding to the XML Schema abstract data model components (e.g. 'element') directly to classes of the XML Schema Metamodel Ontology (e.g. 'Element'). Attributes of meta-element information items have been mapped to datatype properties '<attribute>_<domain meta-element information item>_String' (e.g. 'name_Element_String') with the classes representing the meta-element information items as domains and the built-in primitive datatype 'string' as range. Universal restrictions on datatype properties have been defined, since all range individuals of these datatype properties have to be of the primitive datatype 'string': <domain meta-element information item> ⊑ ∀ <attribute>_<domain meta-element information item>_String.String (e.g. Element ⊑ ∀ name_Element_String.String). As XML Schemas' components can not only have child elements as content, but also plain text, the datatype property 'valueXSD_<domain meta-element information item>_String' (e.g. 'valueXSD_Documentation_String') and the associated universal restriction <domain meta-element information item> ⊑ ∀ valueXSD_<domain meta-element information item>_String.String (e.g. Documentation ⊑ ∀ valueXSD_Documentation_String. String) have been added, since the class representing the meta-element information item is a sub-class of the anonymous super-class of all the individuals which have only relationships along this datatype property to individuals of the class 'String' corresponding to the built-in datatype 'string' or have no relationships along this datatype property. The XML Schema Metamodel Ontology includes the datatype property 'valueXML_<domain meta-element information item>_String' and the corresponding universal restriction <domain meta-element information item> ⊑ ∀ valueXML_<domain meta-element information item>_String.String, since XML document instances' components may contain text. Considering the OWL assertional knowledge, the XML document fragment <VariableName ... lang="en">EF1 </VariableName> is mapped to the property assertions valueXML_Attribute_String (Lang-Individual, 'en') and valueXML_Element_String (VariableName-Individual, 'EF1'). Attributes of meta-element information items such as 'ref' referring to meta-element information items, have been transferred to object properties '<attribute>_<domain meta-element information item>_<range meta-element information item>' with corresponding universal restrictions <domain meta-element information item> ⊑ ∀ <attribute>_<domain meta-element information item>_<range meta-element information item>.<range meta-element information item> (e.g. Attribute ⊑ ∀ ref_Attribute_Attribute.Attribute). Diverse meta-element information

items include the attributes ‘type’ or ‘base’. These attributes may have simple ur-type, simple type or complex type definitions as possible attribute values. According to the specified naming conventions, each attribute would be transformed into three different object properties with the ranges ‘AnySimpleType’, ‘SimpleType’ and ‘ComplexType’. XSLT transformations, building generated ontologies automatically based on XML Schemas, would have to determine the range of the object properties belonging to specific ‘type’ and ‘base’ attributes at runtime. It is complicated and error-prone to determine if type references either point to simple or complex type definitions which are part of external XML Schemas’ namespaces. During the transformation process, XML Schemas with appropriate target namespaces have to be available and it has to be iterated over each simple and complex type. Due to these reasons, the authors have decided to map the attributes ‘type’ and ‘base’ to the object properties ‘type|base_<domain meta-element information item>_Type’ with the class ‘Type’ as range. ‘Type’ represents the super-class of all three possible type definitions. As a consequence, each specific type definition can be in the range of the ‘type’ and ‘base’ object properties. Universal restrictions on these object properties have been specified as well: <domain meta-element information item> $\sqsubseteq$ $\forall$type|base_<domain meta-element information item>_Type.Type (e.g. Element $\sqsubseteq$ $\forall$ type_Element_Type.Type). Part-of relationships to child meta-element information items as content of meta-element information items have been transferred to object properties ‘contains_<domain meta-element information item>_<range meta-element information item>’. Universal restrictions on each object property have been defined, because the range of relationships along these object properties is assumed as fixed: <domain meta-element information item> $\sqsubseteq$ $\forall$ contains_<domain meta-element information item>_<range meta-element information item>.<range meta-element information item> (e.g. ComplexType $\sqsubseteq$ $\forall$ contains_ComplexType_SimpleContent. SimpleContent).

2.2 Generated Ontologies

Executing an XSLT script, the declarations and definitions of any XML Schemas are transformed into classes of generated ontologies directly and automatically. As all components of the normative XML Schema for XML Schemas are included in the XML Schema Metamodel Ontology, this works with all valid XML Schemas. The mapping process takes seconds and requires no human interaction. The generated ontologies’ classes are defined as sub-classes of the XML Schema Metamodel Ontology. Hence, all generated ontologies are based on the same reusable classes. Like heavyweight ontologies [7], the generated ontologies consist of a hierarchy of classes as well as relations with domains and ranges. Moreover, the generated ontologies include universal restrictions on object properties, hasValue restrictions on datatype properties and complex classes consisting of the union of multiple classes’ individuals, if universal restrictions on object properties have more than one class in the range. Table 2 depicts the mappings between XML Schemas and the generated ontologies.

Table 2. Mapping of XML Schemas to generated ontologies

XML Schemas	Generated Ontologies
element information items	sub-classes of XML Schema Metamodel Ontology's classes: <element information item> ⊑ <meta-element information item>
values of element information items' attributes	hasValue restrictions on XML Schema Metamodel Ontology's datatype properties: <element information item> ⊑ ∃<attribute>_<domain meta-element information item>_String.{<String>}
texts contained in element information items	hasValue restrictions on XML Schema Metamodel Ontology's datatype properties: <element information item> ⊑ ∃valueXSD_<domain meta-element information item>_String.{<String>}
values of element information items' attributes referring to other element information items	universal restrictions on XML Schema Metamodel Ontology's object properties: <domain element information item> ⊑ ∀<attribute>_<domain meta-element information item>_<range meta-element information item>.<range element information item>
values of attributes 'type' and 'base'	universal restrictions on XML Schema Metamodel Ontology's object properties: <domain element information item> ⊑ ∀type|base_<domain meta-element information item>_Type.<range element information item>
element information items' part-of relationships	universal restrictions on XML Schema Metamodel Ontology's object properties: <domain element information item> ⊑ ∀contains_<domain meta-element information item>_<range meta-element information item>.<union of range element information items>

XML Schemas' element information items are mapped to sub-classes of the XML Schema Metamodel Ontology's classes: <element information item> ⊑ <meta element information item>. The element information item 'element' with the assigned name 'VariableName' (<xs:element name="VariableName" ... />), for example, is tranferred to the class 'VariableName' with 'Element' as super-class (VariableName ⊑ Element), since all 'VariableName' individuals are also part of the 'Element' class extension. Values of element information items' attributes are transformed into hasValue restrictions on the XML Schema Metamodel Ontology's datatype properties

<element information item> ⊑ ∃ <attribute>_<domain meta-element information item>_String.{<String>}, as the element information item is the sub-class of the anonymous super-class of all the individuals which have at least one relationship along the datatype property ‘<attribute>_<domain meta-element information item>_String‘ to the specified individual of the primitive datatype ‘string’. For instance, the value of the attribute ‘name’ of the element information item ‘element’ (<xs:element name="VariableName" ... />) is converted to the datatype property hasValue restriction VariableName ⊑ ∃ name_Element_String.{'VariableName'}, since each element ‘VariableName’ has at least one associated name, namely ‘VariableName’. Texts contained in element information items are mapped to hasValue restrictions on the XML Schema Metamodel Ontology’s datatype property <element information item> ⊑ ∃ valueXSD_<domain meta-element information item>_String.{<String>}. For example, the text included in the element information item ‘documentation’ (<xs:documentation>Indicates the language of content.</xs:documentation>) is translated into the datatype property hasValue restriction Documentation1 ⊑ ∃ valueXSD_Documentation_String. {'Indicates the language of content.'}. As element information items may contain more than one element information item of the same meta-element information item, the contained element information items’ identifiers are sequential (e.g. Documentation1). Values of element information items’ attributes referring to other element information items are converted to universal restrictions on the XML Schema Metamodel Ontology’s object properties <domain element information item> ⊑ ∀ <attribute>_<domain meta-element information item>_<range meta-element information item>.<range element information item>. The reference to the element information item ‘attribute’ called ‘lang’ (<xs:attribute ref="lang"/>) is transformed into the object property universal restriction Lang-Reference ⊑ ∀ ref_Attribute_Attribute.Lang. The values of the attributes ‘type’ and ‘base’ are transferred to universal restrictions on XML Schema Metamodel Ontology’s object properties: <domain element information item> ⊑ ∀ type|base_<domain meta-element information item>_Type.<range element information item>. The attribute ‘type’ of the element information item ‘element’ named ‘VariableName’ (<xs:element name="VariableName" type="NameType"/>), for example, is converted to the object property’s universal restriction VariableName ⊑ ∀ type_Element_Type.NameType. Element information items’ part-of relationships are realized by universal restrictions on XML Schema Metamodel Ontology’s object properties <domain element information item> ⊑ ∀ contains_<domain meta-element information item>_<range meta-element information item>.<union of range element information items>. The complex type definition ‘InternationalStringType’ includes only one ‘simpleContent’ element information item (<xs:complexType name="InternationalStringType"> ...<xs:simpleContent>...</xs:simpleContent></xs:complexType>). As a consequence, the range of the object property can only consist of individuals of one class (InternationalStringType ⊑ ∀ contains_ComplexType_SimpleContent.Simple Content1). If element information items like ‘extension’ have more than one element information item as content (e.g. <xs:extension...><xs:attribute name="translated"...> ...</xs:attribute><xs:attribute name="translatable"..>..</xs:attribute></xs:extension>), the domain element information items can only have relationships along the object

property to individuals of the complex class consisting of the union of individuals of multiple classes representing the contained range element information items (Extension1 ⊑ ∀ contains_Extension_Attribute.(Translated ⊔ Translatable)).

2.3 Domain Ontologies and Integration of Other Ontologies

In domain ontologies, the semantics of classes are specified as exactly as needed using formal logic [7]. Each data model of a specific domain can be expressed in the form of a domain ontology. Classes of any number of generated ontologies of a given domain can be annotated as equivalent to classes of the domain ontology (<domain ontology class> ≡ <generated ontology class>). Thus, the information of a particular domain stored in generated ontologies and in corresponding XML Schemas can be reused during early stages of the domain ontology design process. Ontology engineers can add further domain specific semantic information to the domain ontology subsequently in a continuous way. You can perform queries on domain ontologies using the semantics of the particular domain without knowledge of complex XML Schemas' structures. Requests on domain ontologies are propagated to the underlying generated ontology or ontologies (if the domain data model consists of more than one XML Schema) via equivalence relationships. Hence, there is no need to query each associated generated ontology individually using different classes, object and datatype properties. Classes of domain ontologies can be annotated as being equivalent to existing similar and widely adopted classes of external ontologies (<domain ontology class> ≡ <external ontology class>; other types of relationships are also possible). Due to this, reasoners may use additional semantic information defined in external ontologies for deductions [8].

3 Related Work

The XML Schema Metamodel Ontology, although much more complex, corresponds to the general database ontology designed by Kupfer et al. [8]. These ontology engineers have defined a database schema-to-ontology mapping, which means that specific database ontologies are generated automatically from any database schemas. Kupfer et al. have specified the conceptual model of the general database ontology as follows: databases can consist of multiple tables and tables can comprise diverse attributes. The authors used the three classes 'Database', 'Table', and 'Attribute' as well as the object property 'consistsOf' to describe database schemas. The classes' identifiers serve as links to all tables and attributes of the underlying database schemas. Kupfer et al. depicted domain ontologies in the context of the developed general database ontology. Using domain ontologies, semantic information about specific domains is annotated and added supplementary to database ontologies. The relation between database ontologies' classes and classes of domain ontologies has been conceptualized using the object property 'containsDataAbout'.
Several strategies lifting the syntactic level of XML documents to the semantic level of OWL ontologies can be distinguished. The authors have clustered appropriate tools

implementing these transformations into three classes depending on the kind of conversion either at the instance, the conceptual, or both the instance and the conceptual level. At the instance level, Klein has developed the so-called RDF Schema mapping ontology enabling a one-way mapping of XML documents to RDF. Relevant XML documents' content can be identified [9]. Extending this approach, Battle has introduced a bidirectional mapping of XML components to RDF [10]. The WEESA system implements an automatic transformation from XML to RDF using an OWL ontology, manually created from corresponding XML Schemas and manually defined rules. XML document instances are not mapped to OWL equivalents [11]. O'Connor and Das developed an approach transforming XML documents to individuals of an OWL ontology describing the serialization of the XML document. SWRL [12] is used to map these instances to individuals of a domain ontology [13].
At the conceptual level you can distinguish between approaches converting XML schema languages to RDFS or OWL. Several languages for writing schemas like DTD [4], XML Schema [5], DSD [14] and Relax NG [15] exist. The prototype OntoLiFT [16] offers a generic means for converting arbitrary XML schema languages to RDFS ontologies semi-automatically. In a first step, XML schema languages are transformed into regular tree grammars consisting of non-terminals, terminals, start symbols and production rules [17]. In a second step, non-terminals as well as terminals are converted to RDFS classes and production rules are mapped to RDF properties. In comparison with our approach, OntoLiFt converts any XML schema language and not just XML Schema to ontologies. Anicic et al. evolved an approach based on meta-models transforming between the different models of XML Schema and OWL [18].
At the instance and the conceptual level, there are methods transforming XML to RDF and XML Schema to either RDFS or OWL. Within the EU-funded project called 'Harmonise' the interoperability of existing standards for the exchange of tourism data has been achieved by the transformation of XML documents and XML Schemas into RDF and RDFS ontologies which have been mapped to each other [19]. Using the approach of O'Connor and Das [20], XML document instances are transformed to OWL ontologies even though associated XML Schemas not exist. As a consequence, unstructured contents can be mapped to OWL ontologies as well. XML Schemas can also be mapped to OWL ontologies, as XML Schema documents are represented in XML, too. New OWL ontologies can be generated from scratch and existing ones can be extended. O'Connor and Das evolved XML Master, a language describing OWL ontologies declaratively. XML Master combines the Manchester OWL Syntax [21] and XPath [22] to refer to XML content. O'Connor and Das criticize the limited and unsatisfactory number of OWL constructs supported by current tools converting XML Schemas to OWL ontologies. Thus, all OWL constructs are covered. One shortcoming associated with this method is that you have to write mapping language expressions manually and therefore you cannot transform XML documents and XML Schemas to OWL ontologies automatically. Another drawback is that ontology engineers have to be familiar with the Manchester OWL Syntax and XPath in order to express the mappings. Ferdinand et al. propose both mappings from XML to RDF and XML Schema to OWL which are independent of each other. This means, OWL individuals do not necessarily correspond to the OWL conceptual model, since XML documents' declarations and definitions may be transferred to differing OWL

constructs [23]. In addition, another system can be stated transferring XML Schema components to OWL language constructs at the terminological level and XML document instances to OWL individuals at the assertional level. XPath expressions are applied selecting XML documents' content [24]. Besides that, the approach of Tous et al. is very similar to this method [25]. The authors of [26] devised a mapping between XML and RDF and between XML Schema and OWL .The authors assume that XML documents are structured like relational databases. Thus, XML documents' relational structures are discovered and represented in OWL. Relations correspond to classes, columns to properties, and rows to instances. XML data model elements are mapped automatically to components of the OWL data model. Named simple and complex types, for instance, are transferred to classes. Elements, containing other elements or having at least one attribute, are converted to classes and object properties between these classes. Both elements, including neither attributes nor sub-elements, and attributes, assumed to represent database columns, are transformed into datatype properties with the surrounding element as their domain. Also, XML cardinality contraints are transformed into equivalent OWL cardinality restrictions.
Many approaches try to extrac semantics from XML Schemas. The suggested approach, in contrast, only gains information about the syntactic structure of XML document instances contained in XML Schemas. Generated ontologies are connected with domain ontologies which are enriched with semantic domain specific information in a further step. The majority of the tools attempt to convert either schemas to ontologies at the conceptual level or XML to RDF at the instance level. The method, presented in this paper, follows a complete approach transforming XML document instances' content to OWL individuals as well as XML Schemas to OWL. In comparison with our approach, many others transform XML to RDF and/or XML schema languages to ontologies in a manual or at most in a semi-automatic and not automatic manner. Furthermore, diverse existent methods generate RDFS ontologies and not the more expressive OWL ontologies.

4 Conclusion

The aim of this paper is to bridge the gap between XML and OWL. XML Schema and OWL ontologies follow differing modeling goals. While XML Schema describes the syntactic structure of XML document instances, OWL is based on formal logic and describes the semantics of data models. In this paper, the authors demonstrated a generic multilevel approach designing domain ontologies when XML Schemas are provided as input sources for the ontology design process. This process of designing ontologies from scratch requires considerable effort. The normal procedure of ontology engineers is specifying the domain ontologies' semantics in collaboration with domain experts already at the beginning of the process. Applying our approach, ontology engineers are allowed to pursue a different path. They can rely on existent information located in XML Schemas of a given domain data model. To realize this, generated ontologies are built in an automatic way based on already available XML Schemas. Therefore, time-consuming work is already done and can be reused by the ontology engineers who do not have to define the domain data model anew. The

generated ontologies' classes are based on super-classes of the XML Schema Metamodel Ontology. This ontology consists of classes representing the components of the XML Schema abstract domain model and the corresponding element information items. The components of the XML Schema abstract data model are used to describe XML Schemas recursively using XML Schema language constructs. Based on interviews with domain experts, the information stored in the generated ontologies can be extended in a continuous manner. This supplemental semantic domain specific information is defined in domain ontologies whose classes are linked to the generated ontologies' classes via equivalence relationships. Domain ontologies' classes can be annotated as equivalent to classes of widely adopted external ontologies. As a consequence, reasoners may use additional semantics for deductions.

5 Future Work

A complete use case designing a specific domain ontology using the devised multilevel approach based on already existing XML Schemas will be described in detail. The underlying data model of the application domain is called the Data Documentation Initiative (DDI) [27]. DDI in its current version 3 is an international standard for describing data from the social, behavioral and economic sciences. Furthermore, more use cases from different domains will be shown to prove the generality of the developed approach. The main benefit associated with this approach, saving time for ontology engineers in the process of designing domain ontologies from scratch, will be evaluated as well.
The authors will develop an XSLT framework to implement a complete stylesheet-driven approach to generate OWL ontologies. So far, XSLT transformations build generated ontologies automatically based on arbitrary XML Schemas. Moreover, the authors will write XSLT transformations, converting XML documents without corresponding XML Schemas determining their syntactic structure to generated ontologies. The first step is creating suitable XML Schemas out of XML document instances automatically. These XML Schemas will then be converted to generated ontologies in a second step. Another XSLT stylesheet will convert XML document instances' data to OWL instances according to the generated ontologies. Generated ontologies and corresponding XML Schemas will be derived automaticly from designed domain ontologies using XSLT transformations. These scripts will be evolved realizing the model-driven development of generated ontologies and underlying XML Schemas associated with the domain ontologies.

References

1. Linked Data, http://linkeddata.org
2. The XML data model, http://www.w3.org/XML/Datamodel.html
3. Resource Description Framework (RDF): concepts and abstract syntax, http://www.w3.org/TR/2002/WD-rdf-concepts-20021108/

4. Extensible Markup Language (XML) 1.0 (fifth edition) - W3C recommendation 26 November 2008, http://www.w3.org/TR/2008/REC-xml-20081126/
5. XML Schema part 0: primer second edition - W3C recommendation 28 October 2004, http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/
6. XML Schema part 1: structures second edition - W3C recommendation 28 October 2004, http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/
7. Stuckenschmidt, H.: Ontologien: Konzepte, Technologien und Anwendungen. Springer-Verlag, Berlin Heidelberg (2009)
8. Kupfer, A., Eckstein, S., Störmann B., Neumann K., Mathiak B.: Methods for a synchronised evolution of databases and associated ontologies. In: Proceeding of the 2007 Conference on Databases and Information Systems IV. (2007)
9. Klein, M.C.A.: Interpreting XML documents via an RDF Schema ontology. In: 13th International Workshop on Database and Expert Systems Applications, Aix-en-Provence (2002)
10. Battle, S.: Gloze: XML to RDF and back again. In: 1st Jena User Conference, Bristol (2006)
11. Reif, G., Gall, H, Jazayeri, M.: WEESA - web engineering for Semantic Web applications. In: 14th World Wide Web Conference, Chiba (2005)
12. SWRL: a Semantic Web Rule Language combining OWL and RuleML, http://www.w3.org/Submission/SWRL/
13. O'Connor, M.J., Das, A.K.: Semantic reasoning with XML-based biomedical information models. In: 13th World Congress on Medical Informatics, Cape Town (2010)
14. Karlund, N., Moller, A., Schwartzbach, M.I.: DSD: a schema language for XML. In: ACM SIGSOFT Workshop on Formal Methods in Software Practice. (2000)
15. Clark, J., Cowan, J., Fitzgerald, M., Kawaguchi, J., Lubell, J., Murata, M., Walsh, N., Webber, D.: Information technology – document schema definition language (DSDL) – part 2: regular-grammar-based validation – RELAX NG. ISO/IEC 19757-2:2003(E). (2003)
16. Volz, R., Oberle, D., Staab, S., Studer R.: OntoLiFT Prototype – WonderWeb: ontology infrastructure for the Semantic Web. Karlsruhe (2003)
17. Murata, M., Lee, D., Mani, M., Kawaguchi, K.: Taxonomy of XML schema languages using formal language theory. In: ACM Transactions on Internet Technology. vol. 5, New York (2005)
18. Anicic, N., Ivezic, N., Marjanovic, Z.: Mapping XML Schema to OWL. In: Enterprise Interoperability, Part V, pp. 243--252, Springer, Berlin (2007)
19. Dell'Erba, M., Fodor, O., Ricci, F., Werthner, H.: Harmonise: a solution for data interoperability. In: Proceedings of the 2nd IFIP Conference on E-Commerce, E-Business, E-Government I3E. (2002)
20. O'Connor, M. J., Das, A. K.: Acquiring OWL ontologies from XML documents. In: Proceedings of the Sixth International Conference on Knowledge Capture, New York (2011)
21. OWL 2 Web Ontology Language Manchester Syntax, http://www.w3.org/TR/owl2-manchester-syntax/
22. XML Path Language (XPath) 2.0 (second edition), http://www.w3.org/TR/xpath20/
23. Ferdinand, M., Zirpins, C., Trastour, D.: Lifting XML Schema to OWL. In: Web Engineering - 4th International Conference, Munich (2004)
24. Kobeissy, N., Genet, M.G., Zeghlache, D.: Mapping XML to OWL for seamless information retrieval in context-aware environments. In: International Conference on Pervasive Services, Istanbul (2007)
25. Tous, R., Garcia, R., Rodriguez, E., Delgado, J.: Architecture of a semantic XPath processor. Application to digital rights management. In: 6th E-Commerce and Web Technologies, Copenhagen (2005)
26. Bohring, H., Auer, S.: Mapping XML to OWL Ontologies. In: Leipziger Informatik Tage, vol. 72, Leipzig (2005)
27. Data Documentation Initiative, http://www.ddialliance.org

Extending Ontologies with Free Keywords in a Collaborative Annotation Environment

Matias Frosterus, Mika Wahlroos, and Eero Hyvönen

Semantic Computing Research Group (SeCo)
Aalto University School of Science, Dept. of Media Technology, and
University of Helsinki, Dept. of Computer Science
http://www.seco.tkk.fi/
firstname.lastname@aalto.fi

Abstract. Semantic web technologies have introduced the idea of annotating content in terms of concepts taken from ontologies. Since concepts are defined in terms of properties and relations to other concepts, descriptions grow up into larger RDF graphs that can be used as a basis for data integration and intelligent information retrieval. Since ontologies do not typically contain all the possible concepts needed for annotation, it is usually necessary to offer the annotator the possibility to introduce new free keywords or tags in addition to the predefined ontology concepts. The problem then is that free keywords/tags do not have ontological connections to the rest of the RDF graph, unless such relations are defined by the annotator. We present a process for integrating free keywords into the ontological framework, and a practical tool implementation of it, discussing the challenges and possibilities introduced by the system. We also describe a case study performed for the Finnish Defence Forces, where the tool is used for creating a faceted semantic search portal featuring the free keywords and the ontological concepts at the same time.

1 Introduction

1.1 Position Statement

A large amount of metadata is being produced through free keywords, or tags, on the web allowing for a robust, easy-to-use, and flexible annotation of content. Ontologies offer an easy way to impose structure and meaning to the free keywords linking the annotated material into the larger framework of the Semantic Web.

1.2 The challenges of free tagging

A common practice in community-based annotation is to allow the users to create the needed terms, or tags, freely when describing objects. This facilitates flexibility in annotations and makes it easier for novice users to describe things. On the other hand, in the professional metadata world (e.g., in museums, libraries, and archives) using shared pre-defined thesauri is usually recommended for enhancing interoperability between annotations of different persons, and enhancing search precision and recall in end-user

applications. Both approaches are usually needed, and can also be supported to some extent by e.g. suggesting the use of existing tags.

A more advanced approach than using thesauri is to use ontologies [6] for harmonizing content indexing. Then indexing is based on language-independent concepts referred to by URIs, and keywords are labels of the actual underlying concepts. Defining the meaning of indexing terms by their properties and relations to other concepts allows for better interoperability of contents and their use by machines. This is important in application areas, such as semantic search, recommending, linking, and automatic indexing. With even a little extra work, e.g. by just systematically organizing concepts along subclass hierarchies and partonomies, substantial benefits can be obtained [2].

Free keywords are needed in many situations:

1. There can be omissions in the ontology that should be added, but are not currently there.
2. Concepts for new things and phenomena that have not yet been added to the ontology may be needed in annotations.
3. The number of concepts, e.g., the names of plants, can be too numerous to be included in the ontology, but can still be needed in annotations.
4. Instance data, e.g., persons, places, events etc. can be needed in annotations.

There is a need for a system that integrates new free keywords into the wider framework of ontologies in an annotation environment. As a solution, we present a system and its implementation for introducing free keywords into ontologies. The next section presents a general overview of the process. After this a specific implementation in a case study done for the Finnish Defence Forces is presented. Finally, we conclude with discussing related and future work.

2 Using Free Keywords in Annotations

Our key problem is how to incorporate metadata with free keywords into an ontology-driven annotation environment in a simple way that does not require ontology modeling knowledge from the annotators. This requires that the free keywords must be turned into a compatible, machine-readable RDF form, and that the relations between the free keywords and the existing ontologies must be established.

The first step in the process depicted in Figure 1 is to go through the free keywords used in the annotations (1) and match as many as possible to existing ontological concepts (2). Keywords should be transformed into the base form and the strings compared to the labels in the ontology.

Keywords that did not match to ontology concepts are then made into RDF objects with the original keyword as the label (3). The class for these should be kept separate from the class of the concepts in the ontology since these have not been approved by ontology developers, and are therefore less reliable than the proper ontological concepts. At this stage, the keyword object can be used in further annotations, and the list can be edited and pruned as needed. However, at this point it does not offer much additional usability compared to existing tagging systems based on using isolated tags.

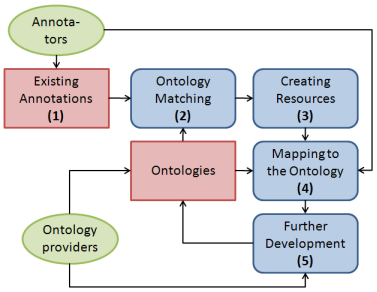

Fig. 1. The process of utilizing free annotations in an ontology-driven annotation environment

In order to take full advantage of using ontologies, the keyword objects should be mapped to the existing ontology (4), typically through the `rdfs:subClassOf` property. Also other relations such as partonomy or equivalence can be used. The keyword objects also do not need to be connected directly to the ontology, but rather can be connected to other keyword objects that are in turn connected to the ontology. When the ontology is developed further (5), the keywords that have been used the most make for prime candidates to be included into the next version of the ontology.

There should be a way, however, for the annotators to keep some keywords out from ontological development if the annotator knows that the keyword will not be of interest to the ontology developers or if the keyword itself is such that it is not wanted to be accessible to the wider public. This latter case is more likely in situations where the annotators are working with sensitive data. The same mechanics can be used by the ontology developers themselves to mark free keywords that they have reviewed but not deemed fit for the ontology.

When new free keywords are needed, the annotator can align them with other ontological concepts straightaway and thus make its meaning explicit within the annotation framework used, leading to less ambiguity. Furthermore, by using literal properties, the annotator can provide detailed explanations of the concept to human readers, and include e.g. labels in different languages, acronyms, and synonyms for the keyword.

A system realizing the process should fulfill the following requirements:

- facilitate finding ontological concepts and free keyword objects for annotations,
- allow the creation of new free keyword objects,
- facilitate the mapping of new free keyword objects to each other and to ontological concepts, and
- instantly show new keyword objects to other annotators and allow their use.

Finally, all of this should be doable without technical expertise, with the application hiding the complexities of the RDF model in the background.

3 Case Study: The Finnish Defence Forces' Norms

The process was implemented in a project done for the Finnish Defence Forces' norms database. The norms comprise of documents describing procedures and regulations as

well as the associated metadata in XML format. The goal of the project was to implement a faceted search portal for the norms utilizing the semantic web technologies.

Metadata about documents included annotations about the subject of the norms using keyword from the Defence Administration's Thesaurus as well free keywords chosen by the annotators. The free keywords contained some spelling mistakes as well as multiples of some keywords (i.e. a singular and a plural form of the same keyword).

For the ontology we used the Finnish Defence Administration's Ontology PUHO[1] which is a domain ontology comprised of concepts relevant to the Finnish Defence Forces developed from the Defence Administration's Thesaurus that has been in use for the annotations of the organization's documents. PUHO extends the General Finnish Upper Ontology YSO[2] so it was also included in the project. For easy use in different applications, the ontology is hosted in the ONKI ontology service[9], which contains several different interfaces for easy integration into other systems and applications.

The metadata was transformed into RDF using a custom conversion process which involved matching keywords present in the metadata with concepts defined in the ontology. Lemmatized forms of the keywords were first obtained in order to identify different inflected forms of the same word, and the lemmatized keywords were then matched with similarly lemmatized labels of ontological concepts using strict string matching. Keywords that did not match the label of any ontological concept were included as new RDF resources with their own URIs.

Once the conversion was ready, the RDF was loaded into the SAHA3 metadata editor [4], which is easily configurable to different schemas, can be used by multiple annotators simultaneously, and works in a normal web browser, therefore needing no special software to be installed. The support for multiple annotators is implemented in a robust way with synchronization and locks which guarantee that the annotators don't interfere with each other's work. The tool also includes a chat channel in case online dicussions between annotators are needed. Using SAHA3, the annotators can collaboratively clean up the free keywords as needed and map them to the ontology, and SAHA3 realizes the requirements set in section 2. SAHA3 is available as open source at Google Code[3].

For the publication of the metadata, SAHA3 is integrated with the multi-faceted search portal generator HAKO that provides easy access to the datasets from different faceted viewpoints. The facets are built automatically based on the properties of the metadata according to a simple configuration description, and the faceted search application is complemented by free text search. HAKO works in a normal web browser allowing easy access to the data from anywhere. For machine use, SAHA3 and HAKO have two machine APIs: one for using the content as an ONKI ontology service [7] for annotation work, and one for using the content via a SPARQL end-point, which can be used by other applications to access all the metadata as needed.

In our case, one of the facets was the subject of the norms featuring both the ontological concepts from PUHO as well as the new free keyword objects. The hierarchical

[1] http://onki.fi/en/browser/overview/puho
[2] http://www.seco.tkk.fi/ontologies/yso/
[3] http://code.google.com/p/saha/

facet contains both types of concepts integrated so that a user does not see a difference between them since the inner workings of the system are of no interest to the user.

4 Discussion and Related Work

This paper presented a process of bringing free keyword annotations into the framework of an ontology-driven annotation system, detailing the different steps necessary as well as the requirements for the tools that facilitate this process. A case study where this was done was presented and the tools used.

Folksonomies and ontologies have been combined before [3, 8, 5] but much of the focus has been on blogs and similar domains where the annotations have been done by the public within a completely free framework, as opposed to professional annotators working with free keywords in tandem with a controlled vocabulary. Others have built domain ontologies based on partially controlled and partially free tagging data and discussed the need to merge future development of the controlled tag vocabulary with the ontology [1]. Our work is more focused on the process of bringing the free keywords into the ontological framework as opposed to using them to build new ontologies or to permanently extend existing ones.

In addition to processes for manually defining relations between isolated tags and ontological concepts, ontologies have also been derived from folksonomies using automatic or semi-automatic methods based on machine learning [3]. Much of the work has focused on discovering implicit semantic relations between tags based on statistical analysis of connections between users, tags, and the objects tagged by the users. The focus of our work is on relatively sparse free keyword data which may not lend itself well to using statistical analysis of the tagging data as the primary technique.

Next, our goal is to try to devise ways to facilitate mapping the free keywords into the ontology easier by trying to reason possible relations from their usage alongside the ontology terms. This could also be used to find out relations between the keywords themselves. We also intend to evaluate the benefits of the system described in the case study from the perspective of practical use cases in document management and search of the norms database.

Acknowledgements This work is part of the National Semantic Web Ontology project in Finland[4] FinnONTO (2003–2012), funded currently by the National Technology and Innovation Agency (Tekes) and a consortium of 35 public organizations and companies.

References

1. Mihai Codescu, Gregor Horsinka, Oliver Kutz, Till Mossakowski, and Rafaela Rau. OSMonto - an ontology of OpenStreetMap tags. In *State of the map Europe (SOTM-EU) 2011*, 2011.
2. Eero Hyvönen, Kim Viljanen, Jouni Tuominen, and Katri Seppälä. Building a national semantic web ontology and ontology service infrastructure—the FinnONTO approach. In *Proceedings of the ESWC 2008, Tenerife, Spain*. Springer–Verlag, 2008.

[4] http://www.seco.tkk.fi/projects/finnonto/

3. Hak Lae Kim, Simon Scerri, John G. Breslin, Stefan Decker, and Hong Gee Kim. The state of the art in tag ontologies: a semantic model for tagging and folksonomies. In *Proceedings of the 2008 International Conference on Dublin Core and Metadata Applications*, pages 128–137. Dublin Core Metadata Initiative, 2008.
4. Jussi Kurki and Eero Hyvönen. Collaborative metadata editor integrated with ontology services and faceted portals. In *Workshop on Ontology Repositories and Editors for the Semantic Web (ORES 2010), the Extended Semantic Web Conference ESWC 2010, Heraklion, Greece*. CEUR Workshop Proceedings, http://CEUR-WS.org, 2010.
5. Alexandre Passant. Using ontologies to strengthen folksonomies and enrich information retrieval in weblogs. In *ICWSM'2007*, 2007.
6. S. Staab and R. Studer, editors. *Handbook on ontologies (2nd Edition)*. Springer–Verlag, 2009.
7. Jouni Tuominen, Matias Frosterus, Kim Viljanen, and Eero Hyvönen. ONKI SKOS server for publishing and utilizing SKOS vocabularies and ontologies as services. In *Proceedings of the 6th European Semantic Web Conference (ESWC 2009)*, 2009. Springer–Verlag.
8. Céline Van Damme, Martin Hepp, and Katharina Siorpaes. FolksOntology: An Integrated Approach for Turning Folksonomies into Ontologies. In *Bridging the Gep between Semantic Web and Web 2.0 (SemNet 2007)*, pages 57–70, 2007.
9. Kim Viljanen, Jouni Tuominen, and Eero Hyvönen. Ontology libraries for production use: The Finnish ontology library service ONKI. In *Proceedings of the ESWC 2009, Heraklion, Greece*. Springer–Verlag, 2009.

Folksonomies behind the scenes

Leyla Jael García-Castro[1], Alexander García[2]

[1] Universität der Bundeswehr München, Werner-Heisenberg-Weg 39,
85779 Neubiberg, Germany
w31blega@unibw.de
[2] University of Arkansas for Medical Sciences, Biomedical Informatics.
agarcia@uams.edu /alexgarciac@gmail.com

Abstract. In this position paper we analyze the similarities amongst folksonomies, semantic wikis, and ontology building; we also propose an alignment and orchestration of ontologies representing these scenarios. We argue that such alignment enables a more direct application of folksonomy-based approaches over these set-ups. The rationale behind folksonomies is shared across environments such as collaborative ontology building and semantic wikis. The three of them aim to facilitate knowledge sharing across communities; a social environment in which individual and community objectives are achieved supports them all.

Keywords: Ontology engineering, social web, semantic web, folksonomy, wiki

1 Introduction

Social tagging systems (STS) have become increasingly popular within the Web 2.0 era; they allow users to freely associate terms, *i.e.* tags, to resources. Tags support a variety of tasks such as information retrieval, personal organization strategies, and share-ability. Conceptual structures emerging from STS are known as folksonomies [1, 2]; they have been used mainly to improve the retrieval on tagged resources [3-5] as well as to discover shared conceptualizations and make explicit the semantic behind tags [6, 7]. Simplicity and immediate benefits for end users, *e.g.* bookmarks available online, are part of the rationale behind the fast adoption of STS [8].

Ontologies are shared conceptualizations that aim to represent an abstraction of a particular domain [9, 10]. Ontologies play a central role in Semantic Web because they are intended to enable data and information exchange in a machine-accessible format; establishing in this way common vocabularies and semantic interpretations of concepts [11]. Whilst agreements in folksonomies are implicit and mainly reached by common use and popularity; agreements in ontologies are explicit as well as documented and supported on evidences.

Wikis enable users to collaboratively create, share, and edit information via a browser interface, thus the final content is the result of everybody's effort [12]. Semantic Wikis, introduced in 2004 [13], aim to facilitate ontology content integration in Wiki as well as to support the evolution of knowledge: moving from term lists to logical constraints while granting users the freedom over the creative

process. Most of the existing semantic Wikis rely on RDF and mainly support subject-predicate-object structures [13]. Semantic Wikis exhibit a similar structure as folksonomies supporting both semantic annotations and selection within documents: a semantic annotation takes as subject a document or a portion of it and relates it to an object by means of a semantic qualifier, *e.g. skos:broader*. This structure could also be extracted from links in traditional Wikis; in this case the annotations would establish a *relatedTo* relation.

In this position paper we analyze the similarities amongst folksonomies, semantic wikis, and ontology building; we also propose an alignment and orchestration of ontologies representing these scenarios. We argue that such alignment enables a more direct application of folksonomy-based approaches over these set-ups.

2 Folksonomies behind the scenes

Annotation Ontology. The Annotation Ontology (AO) [14] represents the annotation process within social environments. AO is built upon the Annotea Project (http://www.w3.org/2001/Annotea/); it is also compatible with Newman's tagging ontology (www.holygoat.co.uk/projects/tags/), the Meaning of a Tag ontology (http://moat-project.org/), and the Simple Knowledge Organization System (SKOS) (http://www.w3.org/2004/02/skos/). AO supports both *free* as well as *semantic annotations*, namely qualified annotations in AO. It enables users to freely attach terms to resources –*free annotations*, as well as terms related to ontological entities – *semantic annotation*. Annotations can be attached to the entire resource as well as to portions of it, *e.g.* text, images, or tables. As annotations on specific parts of a document do not necessarily apply to the whole document, implementations should take care of it by enabling users to define whether or not such annotations should be also global. AO also supports the curation process over the annotations and offers different types of annotations such as notes, comments, erratum, etc. It offers provenance support by reusing the Provenance Authoring and Versioning ontology (PAV, http://swan.mindinformatics.org/spec/1.2/pav.html).

Collaborative Ontology Building. The Changes and Annotations Ontology (ChAO) [15] provides a model to track the modifications on ontology classes, properties and instances. It contains two main classes: *Change* represents changes –add, edit, and delete, in the ontology, and *Annotation* that stores related information such as comments, examples, explanations and votes. ChAO is currently in use by the collaborative Protégé project (http://protegewiki.stanford.edu/wiki/Collaborative_Protege). The ontology building process entails negotiation practices, *i.e.* ontologies are social agreements to accomplish shared objectives. When building ontologies, people are pursuing: (i) retrieving related information, (ii) sharing information, and (iii) improving and broadening both knowledge and performance. Interestingly, these are also the motivations for tagging resources [16]. We consider the ontology building process as a structured folksonomy in which the main document being tagged is that one representing the ontology; also, the participants are aware of the purpose of their contributions, *i.e.* annotations.

Mapping ChAO and AO makes it possible to use the flexibility of folksonomies

into the ontology building process; Fig. 1 shows the proposed mapping. Users in ChAO are identified by user names or accounts whilst AO uses *foaf:Agent* for that purpose; this brings benefits such as a unique URI to identify a contributor participating in different ontology developments, regardless of the methodology or editor. AO can also facilitate reusing information from other folksonomies as it is compatible with Newman's ontology, MOAT, and SKOS.

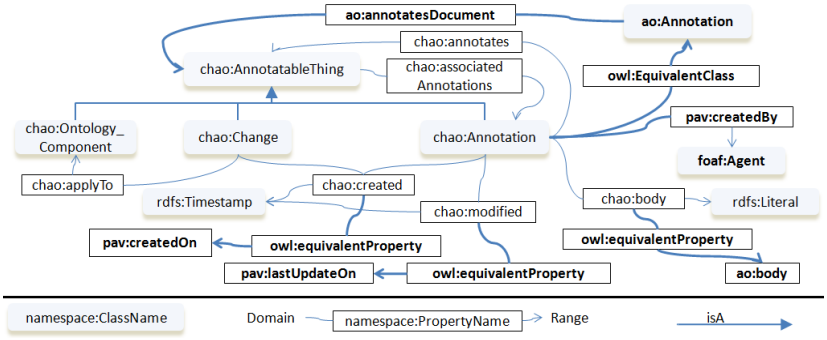

Fig. 1 AO and ChAO mapping

Fig. 2 shows an annotation from the ChAO and AO perspectives: a user named Daniel works on the Pizza ontology document; he creates a property #hasTopping, *chao:Property_Created*, on a *chao:Ontology_Component*, and adds an annotation, *chao:Annotation*, explaining why the property was created. From the AO perspective Daniel is represented as a *foaf:Agent*; he creates an annotation, *ao:Annotation*, corresponding to a creation, *ao:hasTopic*, on a portion of the ontology named #hasTopping, *ann:context*. In AO, the property #hasTopping is represented by means of XPointer (www.w3.org/TR/xptr/) selector, *i.e.* an element in an XML document.

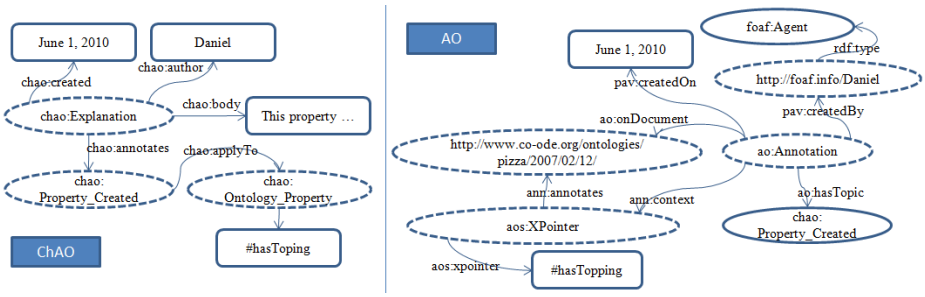

Fig. 2 Changes on an ontology component, ChAO and AO perspectives

During the ontology building, contributors perform activities such as adding, editing, and deleting ontological entities; those activities produce classes, properties or axioms that can be seen as portions of the ontology, easily identified by a URI. Annotations such as comments, notes, and votes are attached to particular entities. Consequently, it is possible to identify a contributor attaching annotations to pieces of

a document, *i.e.* the ontology; this process is fully supported by AO. Conjugation of ChAO, AO, PAV and FOAF makes it possible to use SPARQL in order to answer questions such as who has worked on this class, on which ontologies have contribute Andy and Tony, or what ontologies have been created for a particular domain.

Wikis. SweetWiki [17] proposes an ontology to represent the wiki structure; concepts include *document*, *page*, *tag*, *link*, *backward link*, *contributor*, *version*, *attached file*, etc. These concepts are found in both semantic and non semantic wikis; most of them are also covered by well known vocabularies such as Dublin Core, SKOS, SIOC, and FOAF. Semantic annotations within semantic wikis follow the structure subject-predicate-object [13]; by using AO, it is possible to model those annotations as qualified ones: the contributor corresponds to the annotator *–pav:createdBy* and *foaf:Agent*, the subject to the wiki page or a portion of it *–ao:onDocument* and *ann:context*, the topic to the type of the annotation *–skos:broader*, *dcterms:isVersionOf*, *sioc:attachment*, etc., and the object to the annotation *–ao:body*. AO currently supports semantic annotations corresponding to *skos:exactMatch*, *skos:closeMatch*, *skos:broader*, and *skos:narrower*; however, it is possible to extend AO in such a way that other qualifiers are also allowed. This extension is based on Hypertag [18] and consists of a new type of annotation *Relationship* that relates either two resources, or one resource as *subject* and the other one as *object*; it also enables both reusing relations, *e.g. dcterms:isVersionOf*, or creating new ones.

3 Conclusions and Future Work

We have presented an alignment between folksonomies and collaborative ontology building that facilitates collaboration across decentralized settings, pacing with dynamics on evolving domains, and monitoring the quality and consistency of the model by using the wisdom of crowds. The proposed alignments will likely facilitate the use of folksonomy-based approaches in wiki environments and vice versa as well as the knowledge emergence from both of them. Semantic annotations as well as provenance do not solve all semantic issues that folksonomies lack of; however, it reduces the gap between the social and the semantic web. The proposed alignment makes it easy to integrate knowledge gathered from social platforms into knowledge elicitation phases in ontology development methodologies. The alignment fits into methodologies with a collaborative component reusing non-structured or semi-structured existing knowledge such as Mature Project [19], NeOn Methodology [20], and Melting Point [21]. Integrated to the Mature Project, our approach makes it easier to consolidate and axiomatize the ontology built by the community as it includes semantic links; facilitating in this way the extraction of hierarchies as well as *ad hoc* relationships and mappings. Similarly, our approach facilitates reusing and reengineering non-ontological resources; one of the phases proposed by the NeOn methodology, as well as the conceptualization activity proposed in the Melting Point. The aforementioned methodologies reuse knowledge while our alignment facilitates to add and extract semantics from social environments. This combination makes possible the evolution of the extracted ontology as this one becomes part of the ontologies used to qualify annotation, thus new mappings and relations can emerge.

In this a way, users contribute to the ontology building process without being aware of the process in which they are taking part.

We have identified three scenarios that could benefit from such an alignment, improving the way that content from folksonomies is currently exploited. The first scenario belongs to the bioinformatics domain and is related to the collaborative annotation of proteins. In such a scenario, documents representing protein sequences are enhanced by semantic annotations that can be applied to the whole sequence or a portion of it as well as to other annotations on the protein. In this way, users provide content in the form of annotations, facilitating the publication of experimental data related to proteins. It also enables the immediate discovery of such information as annotations are modeled with AO and linked to protein specialized vocabularies; thus, it will be available as part of the Linked Open Data cloud. It will also facilitate ontology evolution by using an AO extension that enables the representation of relationships.

The second scenario belongs to the biological domain, it is related to annotations in laboratory notebooks. Tags4Labs [22] is a prototype supporting the annotation of experimental data for some of the processes routinely run at the Center for International Tropical Agriculture (CIAT) biotechnology laboratory. With the proposed alignment it will be also possible to use annotations as an enrichment mechanism for those ontologies being used to annotate experimental procedures. The third scenario belongs to the medical domain and is related with the annotation of medical images. Ceballos et al. [23] (http://72.167.51.20:8888/webprotege/) propose and environment in which medical images can be annotated with ontological terms or just by "tagging". With the proposed alignment it will be also possible to enrich existing ontologies by capturing the evidence behind a "tag" so that ontology engineers can decide on the inclusion of the term in the ontology. Also, other users will be able to access such information, making it easier for them to evaluate the relevance of the term and its corresponding use.

Documents should be able to "know about" their own content for automated processes in order to "know what to do" with them. With the proposed alignment we aim to make it possible, *i.e.* both knowledge discovery and knowledge emergence.

References

1. Jaschke, R., Hotho, A., Schmitz, C., Ganter, B., Stumme, G.: Discovering shared conceptualizations in folksonomies. Web Semantics: Science, Services and Agents on the World Wide Web 6 (2008) 38-53

2. Helic, D., Strohmaier, M., Trattner, C., Muhr, M., Lerman, K.: Pragmatic evaluation of folksonomies. International World Wide Web Conference. ACM, Hyderabad, India (2011)

3. Begelman, G., Keller, P., Smadja, F.: Automated Tag Clustering: Improving search and exploration in the tag space. World Wide Web Conference - Collaborative Web Tagging Workshop, Scotland (2006)

4. Heymann, P., Garcia-Molina, H.: Collaborative Creation of Communal Hierarchical Taxonomies in Social Tagging Systems Stanford University (2006)

5. Yeung, C.A., Gibbins, N., Shadbolt, N.: Understanding the Semantics of Ambiguous Tags in Folksonomies. International Workshop on Emergent Semantics and Ontology Evolution, Korea (2007)

6. Angeletou, S., Sabou, M., Specia, L., Motta, E.: Bridging the Gap Between Folksonomies and the Semantic Web: An Experience Report. European Semantic Web Conference - Bridging the Gap between Semantic Web and Web 2.0 Workshop, Austria (2007)
7. Van Damme, C., Hepp, M., Siorpaes, K.: FolksOntology: An Integrated Approach for Turning Folksonomies into Ontologies. European Semantic Web Conference - Workshop "Bridging the Gap between Semantic Web and Web 2.0", Austria (2007)
8. Hotho, A., Jäschke, R., Schmitz, C., Stumme, G.: Information Retrieval in Folksonomies: Search and Ranking. The Semantic Web: Research and Applications (2006) 411-426
9. Gruber, T.: What is an Ontology? , Vol. 2009 (1992) Retrieved May. 23, 2009, from http://www-ksl.stanford.edu/kst/what-is-an-ontology.html
10. Guarino, N., Poli, R., Gruber, T.: Toward Principles for the Design of Ontologies Used for Knowledge Sharing. International Workshop on Formal Ontology. Kluwer Academic Publishers (1993)
11. Gendarmi, D., Lanubile, F.: Community-Driven Ontology Evolution Based on Folksonomies. On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops, Vol. 4277. Springer (2006)
12. Schaffert, S., Gruber, A., Westenthaler, R.: A Semantic Wiki for Collaborative Knowledge Formation. Semantics 2005, Vienna, Austria (2005)
13. Kuhn, T.: AceWiki: A Natural and Expressive Semantic Wiki. Semantic Web User Interaction, Florence, Italy (2008)
14. Ciccarese, P., Ocana, M., Garcia Castro, L.J., Das, S., Clark, T.: An Open Annotation Ontology for Science on Web 3.0. BmC Bioinformatics (accepted) (2010)
15. Noy, N., Chugh, A., Liu, W., Musen, M.: A Framework for Ontology Evolution in Collaborative Environments (2006)
16. Braun, S., Schmidt, A., Walter, A., Nagypal, G., Zacharias, V.: Ontology Maturing: a Collaborative Web 2.0 Approach to Ontology Engineering. International World Wide Web Conference - Workshop on Social and Collaborative Construction of Structured Knowledge (CKC), Canada (2007)
17. Buffa, M., Gandon, F.: SweetWiki: semantic web enabled technologies in Wiki. International Symposium on Wikis. ACM, Odense, Denmark (2006)
18. García-Castro, L.J., Hepp, M., García, A.: Tags4Tags: Using Tagging to Consolidate Tags. International Conference on Database and Expert Systems Applications, Linz, Austria (2009)
19. Braun, S., Schmidt, A., Walter, A., Nagypal, G., Zacharias, V.: Ontology Maturing: a Collaborative Web 2.0 Approach to Ontology Engineering. International World Wide Web Conference - Workshop on Social and Collaborative Construction of Structured Knowledge (CKC), Canada (2007)
20. Suárez-Figueroa, M., Dellschaft, K., Montiel-Ponsoda, E., Villazón-Terrazas, B., Yufei, Z., Aguado-de-Cea, G., García, A., Fernández-López, M., Gómez-Pérez, A., Espinoza, M., Sabou, M.: NeOn Methodology for Building Contextualized Ontology Networks (NeOn Deliverable D5.4.1.). NeOn Project (2008)
21. Garcia, A., O'Neill, K., Garcia, L.J., Lord, P., Stevens, R., Corcho, O., Gibson, F.: Developing Ontologies within Decentralised Settings. In: Chen, H., Wang, Y., Cheung, K.-H. (eds.): Semantic e-Science, Vol. 11. Springer US (2010) 99-139
22. Garcia Castro, A., Giraldo, O., Garcia Castro, L.J.: Annotating experimental records using ontologies. International Conference on Biomedical Ontology, Buffalo, NY, USA (2011)
23. Ceballos, O., Garcia Castro, A., Garcia Castro, L.J., Millan, M.: Anotación semántica de imágenes médicas. Acta Biológica Colombiana 15 (2010)

Ontologies Come of Age with the iKUP Browser

Simon Jupp[1], Julie Klein[2], Panagiotis Moulos[2], Joost Schanstra[2], and Robert Stevens[1]

[1] School of Computer Science, The University of Manchester, UK
[2] Institut National de la Santé et de la Recherche Médicale, Toulouse, France
first.last@manchester.ac.uk

The iKUP browser, a web-based interface to the kidney and urinary pathway knowledge base (KUPKB) [4], shows ontologies coming of age by enabling biologists to ask questions of integrated data that form hypotheses that are being tested in the laboratory. That is, *semantically* annotated data have been delivered to the target users in a form that they can use to change how they undertake their job. iKUP uses a browsing approach to query the KUP data as its users are usually not bioinformaticians that will design and use sophisticated scripts or workflows, and they are almost certainly not users familiar with semantic web technologies. The KUPKB contains data from high-throughput experiments on the kidney and the urinary system. The experimental data is richly *interconnected* to other biological data to form a single integrated repository for querying and exploration. The KUPKB uses multiple biomedical ontologies that act as a controlled vocabulary for standardised annotation of the datasets. These ontologies' semantics are used to ask queries that return *intelligent* answers that form part of the biologists' hypothesis generation process. By reducing the data to common representation languages like the Web Ontology Language (OWL) and the Resource Description Framework (RDF), we have shown semantic web technologies offering novel opportunities for data analysis.

The iKUP browser supports renal biologists in finding data integrated from many disparate data sources, providing a simple interface to survey a large set of the KUP domains 'omics experiments simultaneously. At present, biologists must gather and integrate many data sets by hand and this integration is vital as genes, proteins, and small molecules have to be co-ordinated across many 'omic levels through investigations reported by many people. By hand, this kind of integration and querying is long, tedious and error prone. Users can use the iKUP browser to search for a molecule (mRNA, miRNA, Protein) or list of molecules. The query box exploits the label and synonym tags for molecules to guide the user with their search *via* a dynamic suggestion box that pops up as users type. Once users confirm the molecules they are looking for are present in the KUPKB, the application performs a SPARQL query based on these search terms to generate the results. The results show known information about each molecule from the range of datasets in the KUPKB. For each result, the user can see where anatomically the molecule is active and under what conditions.

The metadata for each experiment is captured using ontologies. Datasets are collected in spreadsheet based templates that have ontologies embedded inside them. By using a spreadsheet template we provide a lightweight and familiar mechanism for the community of renal biologist to submit data. By embedding

the ontologies inside the spreadsheets, we are able to collect ontological annotations from the users by stealth. These *semantic spreadsheets* are created using RightField [7] which provides a mechanism to embed ontology based restrictions on values inside the spreadsheet. These consistent and precise metadata help to integrate the different experiments and are presented to the user on querying the KUPKB *via* iKUP. Every search result is accompanied by a navigable polyhierarchy that is generated from the ontological metadata. This hierarchy provides a faceted browser that exploits the semantics of the ontology to perform *intelligent* filtering of the data. For example, filtering the results on experiments on the glomerulus of the kidney will include in the results any experiments where the sample is glomerular or any part of the glomerulus, thus spanning from gross anatomy to the cellular level.

The KUPKB is public and open-access, and biologists are encouraged to submit new datasets to the KUPKB. So far, over 160 experiments have been sourced from the literature and public databases. All biological identifiers are normalised to a common identification scheme based on *de facto* URIs sourced from linked data resources, such as Bio2RDF [1] and OBO ontologies. To provide deeper querying and more flexibility, the knowledge base is also accessible through a SPARQL endpoint. The iKUP application is online at http://www.kupkb.org and the SPARQL endpoint is at http://www.e-lico.eu/kupkb/sparql. Many of the datasets and ontologies are sourced from public repositories, however, a specific KUP Ontology (KUPO) was developed and is available from http://www.e-lico.eu/kupo. The source code for the KUPKB website is open source and available from http://code.google.com/p/kupkb-dev. Both the KUPO and the KUPKB were built by engaging our non-Semantic Web savvy users in the building process using tools such as Populous [3] and RightField. Using such tools enables iKUP's users to engage with the building process, delivering the annotated data for the KUPKB and extensions to the KUPO without having to have knowledge of OWL, RDF, SPARQL and related tools. We have taken this approach in an attempt to make the KUPKB sustainable and scalable.

The KUPKB was designed to fulfil specific requirements from members of the KUP community. These requirements include the integration of experimental datasets with existing biological databases about genes, proteins, miRNAs, metabolites and diseases. This type of data integration provides a constant challenge for bioinformaticians. One goal of the KUPKB was to show how many of these challenges can be overcome when data are aligned to a common representation and reuse is maximised. To do this we have reused portions of Bio2RDF and made the KUPO as an application ontology, reusing and extending many OBO ontologies [6].

The KUPKB utilises the state of the art in semantic web technology wherever possible. At its core is the RDF triple store that stores the data. After an evaluation we decided to use a Sesame framework [2] backed with a BigOWLIM storage and inference layer [5]. The iKUP web application is built using the Google Web Toolkit (GWT). Whilst the primary search on the iKUP website is powered by SPARQL, we also exploit the ontologies using the Java based OWL

API to classify results and handle the faceted browsing. We load all the ontologies into the OWL API and use the OWL-DL reasoner HermiT for classification. This provides us with the necessary inference to generate the hierarchy and drive the dynamic filtering of results on the web site. We can develop the whole application using a single programming language as Java interfaces are provided for Sesame, the OWL API and GWT. The importance of having such APIs in place can not be underestimated. The iKUP browser is a demonstration that semantic technologies are sufficiently matured so that they can deliver competitive data integration solutions.

The KUPKB and iKUP show ontologies coming of age by fulfilling some of their promise. The KUPKB has used ontologies to provide a common semantic framework for a broad range of previously semantically heterogeneous data. The use of Semantic Web technologies provides the means to integrate and query these data. The key to the coming of age is the iKUP user interface; without a simple means to access these integrated data, our biologist users would not and could not use the KUPKB; ontologies come of age when they deliver meaningful use to their intended users. This has now happened with the KUPKB, with biologists testing hypotheses generated *via* the iKUP in laboratories. Though the KUPKB is relatively small, it does what semantic technologies are supposed to do and show what is possible with biology's rich resource of data once issues of heterogeneity are taken away and the means of delivery to its users is taken into account.

Acknowledgments: This work is funded by the e-LICO project—EU/FP7/ICT-2007.4.4.

References

1. Franois Belleau, Marc-Alexandre Nolin, Nicole Tourigny, Philippe Rigault, and Jean Morissette. Bio2rdf: Towards a mashup to build bioinformatics knowledge systems. *Journal of Biomedical Informatics*, 41(5):706 – 716, 2008. Semantic Mashup of Biomedical Data.
2. Jeen Broekstra, Arjohn Kampman, and Frank van Harmelen. Sesame: A generic architecture for storing and querying rdf and rdf schema. In Ian Horrocks and James Hendler, editors, *The Semantic Web - ISWC 2002*, volume 2342 of *Lecture Notes in Computer Science*, pages 54–68. Springer Berlin / Heidelberg, 2002.
3. Simon Jupp, Matthew Horridge, Luigi Iannone, Julie Klein, Stuart Owen, Joost Schanstra, Robert Stevens, and Katy Wolstencroft. Populous: A tool for populating ontology templates. *CoRR*, abs/1012.1745, 2010.
4. Simon Jupp, Julie Klein, Joost Schanstra, and Robert Stevens. Developing a kidney and urinary pathway knowledge base. *Journal of Biomedical Semantics*, 2(Suppl 2):S7, 2011.
5. Atanas Kiryakov, Damyan Ognyanov, and Dimitar Manov. Owlim - a pragmatic semantic repository for owl. In Mike Dean, Yuanbo Guo, Woochun Jun, Roland Kaschek, Shonali Krishnaswamy, Zhengxiang Pan, and Quan Z. Sheng, editors, *WISE Workshops*, volume 3807 of *Lecture Notes in Computer Science*, pages 182–192. Springer, 2005.

6. Smith, Barry, Ashburner, Michael, Rosse, Cornelius, Bard, Jonathan, Bug, William, Ceusters, Werner, Goldberg, Louis J., Eilbeck, Karen, Ireland, Amelia, Mungall, Christopher J., Leontis, Neocles, Rocca-Serra, Philippe, Ruttenberg, Alan, Sansone, Susanna-Assunta, Scheuermann, Richard H., Shah, Nigam, Whetzel, Patricia L., and Lewis, Suzanna. The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature Biotechnology*, 25(11):1251–1255, November 2007.
7. Katy Wolstencroft, Stuart Owen, Matthew Horridge, Olga Krebs, Wolfgang Mueller, Jacky L. Snoep, Franco du Preez, and Carole Goble. Rightfield: embedding ontology annotation in spreadsheets. *Bioinformatics*, 27(14):2021–2022, 2011.

Dynamic *Is-a* Hierarchy Generation for User-Centric Semantic Web

Kouji Kozaki[1], Keisuke Hihara[1], Riiciro Mizoguchi[1]

[1]The Institute of Scientific and Industrial Research, Osaka University
8-1 Mihogaoka, Ibaraki, Osaka, 567-0047 Japan
{kozaki, hihara, miz}@ei.sanken.osaka-u.ac.jp

Abstract. In ontological theories, *is-a* hierarchy must represent the essential property of things and hence should be single-inheritance, since the essential property of things cannot exist in multiple. However, we cannot avoid multi-perspective issues when we build an ontology because users often want to understand things from their own viewpoints. Especially, in the Semantic Web, the variety of user issues in capturing target domains. In order to tackle this multi-perspective issue, we should adopt an approach of dynamically generating *is-a* hierarchies according to the viewpoints of users from an ontology using single-inheritance. This article discusses a framework for dynamic *is-a* hierarchy generation and its implementation as an extended function of Hozo. Through the function, users can understand an ontology from their own viewpoints.

Keywords: ontology, *is-a* hierarchy generation, viewpoint, multi-perspective

1 Introduction

Ontologies are designed to provide systematized knowledge and machine readable vocabularies of domains for Semantic Web applications. The competences of semantic technologies strongly depend on the ontology which they use. Ontology is defined as "An explicit specification of conceptualization" [1], and it clearly represents how the target world is captured by people and systems.

Semantics of concepts (classes) are defined clearly through the description of their relationships between other concepts in an ontology. In particular, the most important relationship is an *is-a* (sub-class-of) relationship which represents a relation between a generalized concept and a specialized concept. Class hierarchies according to *is-a* relationships are called *is-a* hierarchies, and they form the foundation of ontologies. That is, *is-a* hierarchies in an ontology reflect how the ontology captures the essential conceptual structure of the target world.

Therefore, in ontological theories, an *is-a* hierarchy should be single-inheritance because the essential property of things cannot exist in multiple. Imagine that objects, processes, attributes, etc. have their own unique and essential properties. The use of multiple-inheritance for organizing things necessarily blurs what the essential property of things is. This observation is strongly supported by the fact that both of

the well-known upper ontologies, DOLCE and BFO, use single-inheritance hierarchies.

Nicola Guarino criticizes the careless usage of *is-a* relationships without enough ontological consideration as *is-a* overloading [2] and proposes an ontology development methodology, called OntoClean, which defines concepts based on meta-properties such as rigidity and anti-rigidity. DOLCE is developed based on the OntoClean methodology using single-inheritance *is-a* hierarchy. BFO is the upper ontology used by the OBO Foundry[1] which aims to create a suite of orthogonal interoperable reference ontologies in the biomedical domain. BFO also uses single-inheritance hierarchy, and it is recommended in the guidelines of OBO Foundry to avoid careless usage of multiple-inheritance.

However, we cannot avoid multi-perspective issues when we build an ontology across multiple domains. It is because domain experts often want to understand the target world from their own domain-specific viewpoints. In many cases, their interests are different even if they are experts in the same domain. In some domains, there are many ways to categorize the same kinds of concepts, and different taxonomies are used depending on the purpose and situation.

For example, in the medical domain, a disease is interpreted from various viewpoints. Consider diabetes as an example. Clinician may pay attention to body parts with the abnormalities and classify diabetes as an abnormal blood sugar level. On the other hand, certain specialists may pay attention to the main condition and may classify diabetes as an abnormality in metabolism, and other specialists may classify diabetes as a lifestyle disease. Staffs administering medical care implicitly understand which *is-a* hierarchy should be used for disease interpretation in correlation with their respective interpretations. This suggests that one *is-a* hierarchy of diseases cannot cope with such a diversity of viewpoints since a single-inheritance hierarchy necessarily represents one viewpoint.

Many efforts are under taken to solve these multi-perspective issues. The OBO Foundry proposes a guideline for ontology development stating that we should build only one ontology in each domain [3]. This is an effort to exclude the multi-perspective nature of domains from ontologies. Ontology mapping is used as an approach to acceptance of multiple ontologies based on the different perspectives in a domain. It aims to make clear the relationships between different ontologies. Someone may consider that multiple-inheritance is an easy way to solve these multi-perspective issues. Because multiple-inheritance causes some ontological problems as mentioned above, our ontology development tool, named Hozo[2], allows the user to use a multiple-inheritance only when he/she represents clearly from which upper-concepts the essential property is inherited[3]. However, if we define every possible *is-a* hierarchy using multiple-inheritances, they would be very verbose and the user's viewpoints would become implicit.

In order to tackle these multi-perspective issues, the authors take a user-centric approach based on ontological viewpoint management. It dynamically generates *is-a* hierarchies according to the viewpoint of users from an ontology using single-

[1] http://www.obofoundry.org/

[2] http://www.hozo.jp

[3] It is represented by two kinds of *is-a* relationships: (essential) *is-a* and (non-essential) *IS-A*.

inheritance. The main strategy is composed of: (1) fixing the conceptual structure of an ontology using single-inheritance based on ontological theories and (2) reorganizing some conceptual structures from the ontology on the fly as visualizations to cope with various viewpoints. Based on this strategy, the authors consider a framework for dynamic *is-a* hierarchy generation according to the interests of the user and implement the framework as an extended function of the ontology development tool Hozo [4]. In this article, we discuss the framework for dynamic *is-a* hierarchy generation and its application to a medical ontology. It would solve the conflicting requirements of multi-perspective and single-inheritance in a good ontology, and it could contribute to a user-centric Semantic Web.

The rest of this paper is organized as follows: In section 2, we introduce dynamic *is-a* hierarchy generation according to viewpoints. In section 3, we discuss implementation of the framework as an additional function of Hozo. In section 4, we shows its application to a medical ontology for dynamic *is-a* hierarchy generation of disease. In section 5, we discuss related work. Finally, we present concluding remarks with future work.

2 Dynamic *Is-a* Hierarchy Generation according to Viewpoints

2.1 Ontology Representation in Hozo

We implement the dynamic *is-a* hierarchy generation system as an additional function of Hozo [4]. Fig.1 shows an example of ontology defined using Hozo. Ontologies are represented by nodes, slots and links. The nodes represent concepts (classes), *is-a* links represent *is-a* (subclass-of) relations, and slots represents *part-of* (denoted by "p/o") or *attribute-of* (denoted by "a/o") relations. A slot consists of its kind ("p/o" or "a/o"), *role concept*, *class restriction*, *cardinality*. Roughly speaking, a slot corresponds to *property* in OWL and its role name represent name of property. Its

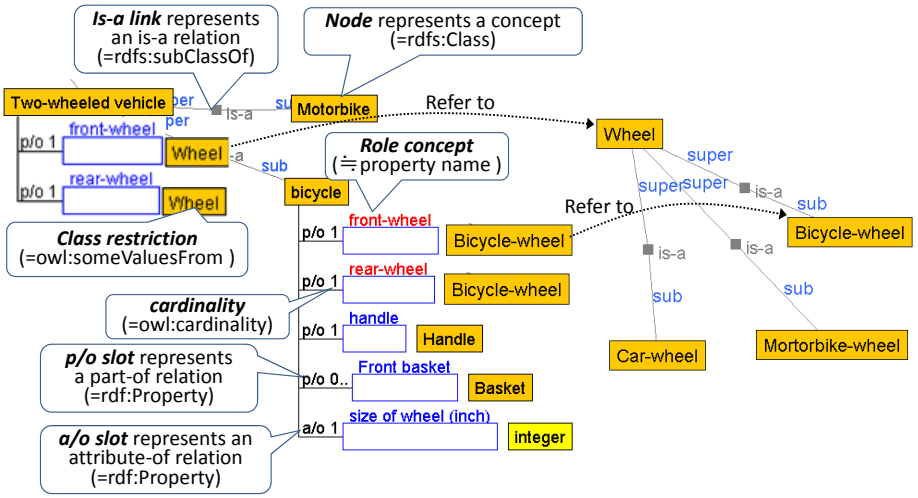

Fig.1 An example of ontology defined using Hozo.

class restriction and cardinality correspond to owl:someValuesFrom and owl:cardinality respectively. Their restrictions refer to other concepts which are defined elsewhere. However, semantics of Hozo's ontology includes some concepts related to role which are not supported in OWL because it is designed based on an ontological theory of role [5]. While we have designed three levels of role representation model in OWL to capture the semantics level-wise [6], we use the simplest model described above in this paper.

In the target ontologies, concepts (classes) are defined by several slots which represent properties and restrictions for them. These definitions are inherited from super-concepts (super-classes) to their sub-concepts (sub-classes) along with *is-a* links. Furthermore, in some sub-concepts, some inherited definitions are specialized according to *is-a* hierarchies of concepts which are referred by their restrictions. For example, *bicycle* in Fig.1 inherits *front-wheel* from *Two-wheeled vehicle* and its class-restriction could be specialized from *Wheel* to *Bicycle-wheel*. This research focuses on these characteristics of *is-a* hierarchies and considers an approach to reorganize *is-a* hierarchies of concepts based on *is-a* hierarchies of concepts referred to by their definitions.

2.2 Dynamic *Is-a* Hierarchy Generation through Transcription of a Hierarchical Structure

Fig.2 outlines a framework for dynamic *is-a* generation. It generates *is-a* hierarchies by reorganizing the conceptual structures of the target concepts selected by a user based on the user's viewpoint. The viewpoint is represented by an *aspect* and a *base hierarchy*. By aspect, we mean something which the user is interested in and selects from the definition of the target concept to generate an *is-a* hierarchy. By base hierarchy, we mean a conceptual structure of concepts which are referred to by the definition selected as the aspect. Because sub-concepts of the target concept could be defined by specializing their inherited definitions according to the base hierarchy, we could reorganize the *is-a* hierarchy of the target concepts according to the following steps:

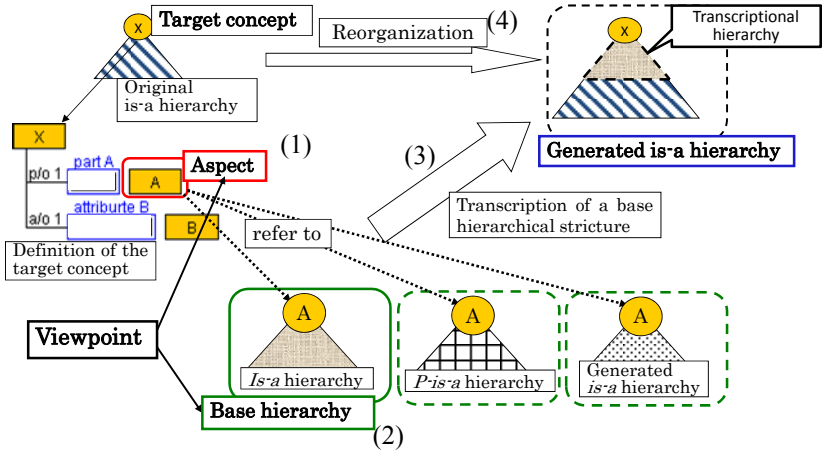

Fig.2 A framework for dynamic *is-a* generation.

Step 1: Selection of an aspect

The user selects something as an aspect from the definition of the target concept for dynamic *is-a* hierarchy generation (see. Fig.2(1)). Because any concept is defined in terms of slots each of which consists of a role-concept, a role-holder [5] and a class-restriction, he/she can select one of them as an aspect. In this paper, we consider only a case where the user selects a class restriction as an aspect for simplicity.

Step 2: Selection of a base hierarchy

The user selects a base hierarchy from hierarchies of concepts which the aspect is referring to (see. Fig.2(2)). In Hozo, three kinds of conceptual hierarchies could be the base hierarchy as follows: the *is-a* hierarchy of concepts referred to by the aspect, the *p-is-a* hierarchy generated by the system according to part-whole relationships of the concepts referred to, and dynamically generated *is-a* hierarchies using the proposed method. A *p-is-a* hierarchy is obtained by abstracting parts from a part-of hierarchy [7]. The detail of the *p-is-a* hierarchy is discussed in section 2.3.2.

Step 3: Transcription of a hierarchical structure

The system defines new sub-concepts of the top concept of target concepts by specializing the definition of the top concept according to the class restriction selected as an aspect and base hierarchy (see. Fig.2(3)). Then, their concept names are automatically determined by the system using a template such as "<the target concept name> with <the specialized aspect> as <the role name of the aspect>". As a result, an *is-a* hierarchy which has the same conceptual structure with the base hierarchy is generated. We call the generated hierarchy a *transcriptional hierarchy* and the operations to generate it a *transcription of a hierarchical structure*.

The scope of a transcription of the base hierarchy could be managed by specifying the number of the target layers rather than to use all concepts of the base hierarchy for transcription.

Step 4: Reorganization of *is-a* hierarchy based on a transcriptional hierarchy

The system reorganizes the *is-a* hierarchy by comparing the original *is-a* hierarchy and the transcriptional hierarchy generated in step 3. The system compares the sub-concepts of the target concept (we call them *existing sub-concepts*) with the concepts on the transcriptional hierarchy (we call them *generated sub-concepts*) according to the aspect and the base hierarchy. When an existing sub-concept's definition specified by the aspect subsumes the definition of a generated sub-concept, the existing sub-concept is classified into sub-concepts of the generated sub-concept. If an existing concept is classified into sub-concepts of multiple generated sub-concepts, the existing concept is classified into the lowest sub-concepts. As a result, all existing concepts are classified into sub-concepts of the generated concepts on the transcriptional hierarchy according to the aspect and the base hierarchy[4].

Through the above four steps, the system can dynamically generate *is-a* hierarchies by reorganizing existing sub-concepts according to the transcriptional hierarchies of base hierarchies.

[4] The result of reorganization corresponds to the result of classification using DL-reasoner while it is implemented by procedural ways in Hozo.

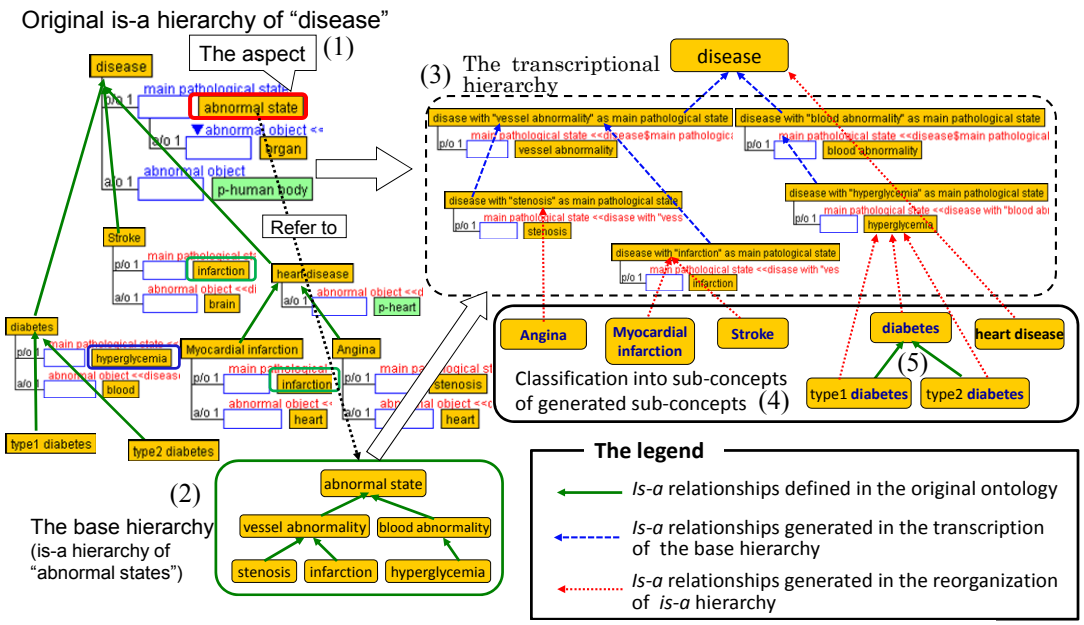

Fig.3 An example of dynamic *is-a* generation of disease in the case that is-a hierarchy of abnormal state is selected as the base hierarchy.

Although DL-reasoners can classify classes (concepts) automatically by reasoning, the result of classification is only an *is-a* hierarchy which is determined uniquely according to the definitions of the classes. Therefore, it is different from our dynamic reorganization according to the users' viewpoints. DL-reasoners can generate a different *is-a* hierarchy only when class definitions in the ontology have changed.

2.3 Examples of Dynamic *Is-a* Generations

§ 1 In the Case of that an *Is-a* Hierarchy is Selected as a Base Hierarchy

As an example, we consider a dynamic *is-a* generation of diseases which is defined in terms of several slots such as “main pathological state”, “abnormal object” and so on (see. Fig.3). Here, we suppose the user selects the class-restriction of “main pathological state” as an aspect (Fig.3(1)) and the *is-a* hierarchy of “abnormal state” as a base hierarchy (Fig.3(2)).

First, sub-concepts of “disease” such as “disease with vessel abnormality as main pathological state” and “disease with blood abnormality as main pathological state” are dynamically generated by specializing the definition of “disease” according to the class restriction selected as the aspect and the base hierarchy. After repetitions of generations of sub-concepts, the transcriptional hierarchy of “disease” is obtained (Fig.3(3)). Then, existing sub-concepts of “disease”, such as “myocardial infarction” and “angina pectoris” are classified into sub-concepts of generated sub-concepts on the transcriptional hierarchy through comparisons between definitions of them (Fig.3(4)). When more than one existing sub-concepts are classified into the same generated sub-concept, they could be organized based on the original *is-a*

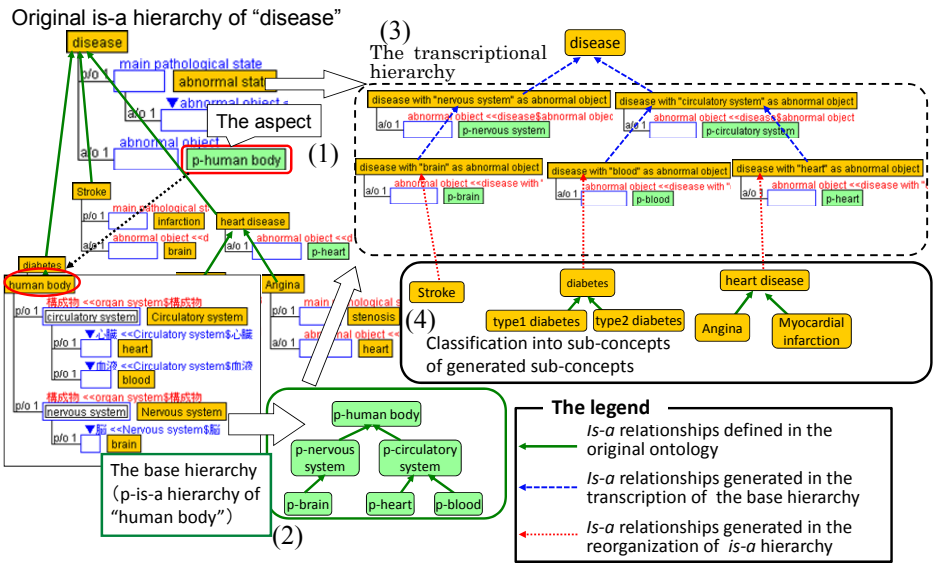

Fig.4 An example of dynamic *is-a* generation of disease in the case that p-is-a hierarchy of human body is selected as the base hierarchy.

relationships between them. In the case shown in Fig.3(5), because *is-a* relationships between "disease with hyperglycemia as main pathological state" and "type1/type2 diabetes" can be identified by reasoning, "type1/type2 diabetes" are classified into sub-concepts of diabetes according to the original *is-a* relationships.

§ 2 In the Case of that an p-*is-a* Hierarchy is Selected as a Base Hierarchy

In the next example, we suppose the user selects the class-restriction of "abnormal object" as the aspect and the p-*is-a* hierarchy of "human body" as the base hierarchy for a dynamic *is-a* generation of disease in the same ontology with the previous example (Fig.4(1),(2)).

In the property inheritance mechanism of ordinary *is-a* relationship, when a super-class and its sub-class have the same slot, the class restriction of the sub-class's slot must be a sub-class of the super-class's one as well. However, in some case, the class restriction of the sub-class's slot must be a part of the super-class's. For example, when <disease of a pulmonary valve *is-a* disease of a heart>, both "disease of a pulmonary valve" and "disease of a heart" have a slot of "site of the disease" and the class restriction of the former must be a part of the latter, that is <pulmonary valve *part-of* heart>.

To cope with such cases, on the basis of our latest theory of roles, we introduced "p-" operator in Hozo which automatically generates a generic concept representing all the parts of the entity to which the operator is attached. The operator enables parts to be inherited by ordinary

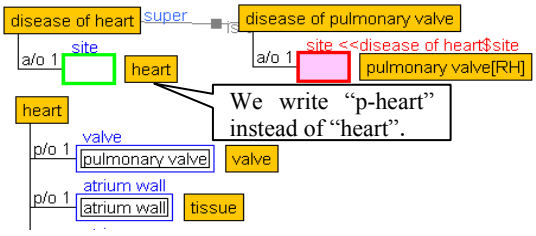

Fig.5 An example of usage of p-operator.

property inheritance mechanism. In the case of Fig.5, for example, we write "p-heart" instead of "heart", then the slot of its subclass inherits not subclass of "heart" but its parts. When p-X is used, Hozo automatically generates a generic concept representing all of the defined parts of X including all parts which have X as their ancestor. This is valid because each part is a subclass of "X's parts class" which coincides with p-X. According to mereology, the theory of parts, p-X includes itself which is not the very X as an entity but X as its part.

On the basis of this theory, Hozo automatically generates *is-a* relationships between p-X such as <p-pulmonary valve *is-a* p-heart>. As a result, an *is-a* hierarchy of p-X is generated according to part-of hierarchy of X. The *is-a* hierarchy of p-X is called p-*is-a* hierarchy[5] and could be selected as a base hierarchy for a dynamic *is-a* generation.

In the case of Fig.4, since the class restriction of "abnormal object" is "p-human body", we can select it as an aspect and p-*is-a* hierarchy as a base hierarchy for dynamic *is-a* generation. Then, sub-concepts of "disease" such as "disease with p-nervous system as abnormal object" and "disease with p-circulatory system as abnormal object" are dynamically generated according to the aspect and the base hierarchy. As a result, the transcriptional hierarchy of "disease" based on p-*is-a* hierarchy of "p-human body" is obtained (Fig.4(3)). The existing sub-concepts of "disease" are classified into the transcriptional hierarchy like Fig.4(4).

In addition to these examples, we can select *is-a* hierarchies which are generated using the proposed method as a base hierarchy to generate another *is-a* hierarchies. That is, our dynamic *is-a* generation could be executed recursively.
The dynamic *is-a* generation is applicable to reorganizations of a portion of an *is-a* hierarchy of "disease". For example, we can select a middle-level concept (e.g. "disease of heart" as the target concept for the dynamic *is-a* generation.

In these ways, we can dynamically generate *is-a* hierarchies of diseases according to the selected aspects and base *is-a* hierarchies from various viewpoints.

3. Implementation

We implemented a prototype of dynamic *is-a* hierarchy generation system as an extended function of Hozo. The system was developed using HozoCore, which is Java API for ontologies built using Hozo, and Hozo OAT (Ontology Application Toolkit), which is Java library for GUI of ontology-based applications developed using HozoCore.

The new function consists of three modules: *is-a* hierarchy viewer, viewpoint setting dialog, and dynamic *is-a* generation module. The *is-a* hierarchy viewer shows an *is-a* hierarchy of an ontology in a tree representation (Fig.6). The user selects a target concept on the *is-a* hierarchy for a dynamic *is-a* generation. The definition of the selected target concept is shown on the viewpoint setting dialog. In the dialog, the user selects a viewpoint for the dynamic *is-a* generation by choosing an aspect, a kind

[5] To deal with *p-is-a* hierarchies in OWL, we can represent them by some design pattern of ontologies such as SEP triple proposed by Udo Hahn and his group [8].

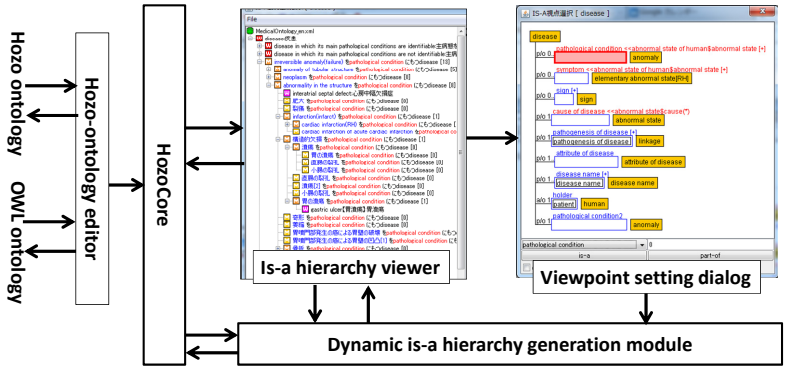

Fig.6 The architecture of the dynamic *is-a* hierarchy generation system.

of base hierarchy and the number of target layers of a transcriptional hierarchy according to his/her interests. The dynamic *is-a* generation module generates an *is-a* hierarchy according to the specified viewpoint. The generated *is-a* hierarchy is shown on the *is-a* hierarchy viewer and could be saved as an ontology file.

While the target of the system is an ontology in Hozo's format, it also can support an ontology in OWL because Hozo can import/export OWL ontologies. When the generated *is-a* hierarchy is exported in the OWL format, its generated sub-concepts in the transcriptional hierarchy are represented by *owl:equivalentClass* which have restriction on properties selected as the aspect. The user can reorganize the *is-a* hierarchy of the exported OWL ontology based on the transcriptional hierarchy by reasoning using a DL reasoner.

5. Application of Dynamic *Is-a* Generation to a Medical Ontology

We applied dynamic *is-a* hierarchy generation system to a medical ontology which we are developing in a project supported by Japanese government [7, 9]. In our medical ontology, diseases are defined by specifying typical *disorder roles*, such as *pathological condition, symptom,* played by *abnormal state*. Fig.7(a) shows the framework to define diseases. Its *disorder roles* are represented as slots with class-restrictions for constraining slot values. These slots are used as aspects for dynamic generation of *is-a* hierarchies of diseases.

For example, when we select the *pathological condition* of disease as an aspect and the *is-a* hierarchy of abnormal state as the base hierarchy, the *is-a* hierarchy of disease is generated (Fig.7(c)). In the generated *is-a* hierarchy, concepts which have names represented by "disease which has X as pathological condition" (e.g. *disease which has abnormality in the structure as pathological condition*) are sub-concepts generated through the dynamic *is-a* hierarchy generation. Their concept names are automatically determined by the system using a template. Exiting sub-concepts are reorganized as sub-concepts of them. For instance, *acute cardiac infarction* is classified into a sub-concept of *disease which has cardiac infarction as pathological condition*. From the generated *is-a* hierarchy, we can understand diseases according to the classification of pathological conditions.

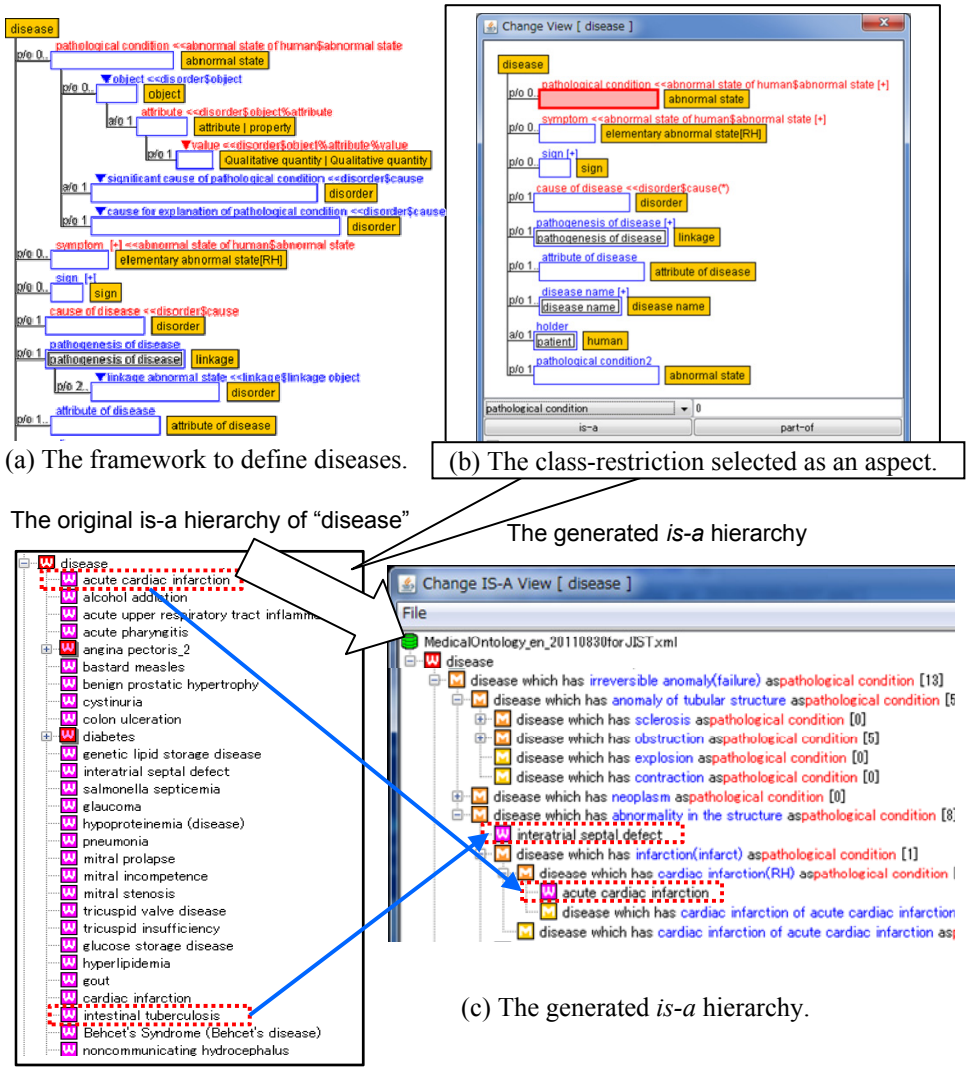

Fig.7 Application of dynamic *is-a* generation to a medical ontology.

On the other hand, when we select the *object of pathological condition* as an aspect and p-*is-a* hierarchy of the *human body* as a base hierarchy, the system generates the *is-a* hierarchy of disease which is similar to the part-whole hierarchy of the human body. For instance, *acute cardiac infarction* is classified into a sub-concepts of *disease which has a pathological condition in the myocardium*.

Moreover, we have developed a medical information system to consider how the dynamic *is-a* hierarchy generation function can be used in other systems [10]. It is used as an index for semantic navigation in the system. We also performed an informal evaluation of the implemented system in a workshop[6] and received

[6] The number of participants was about 25. It includes not only the members of the medical ontology development but also others who work in the medical domain.

favorable comments from medical experts. They especially liked the dynamic *is-a* hierarchy reorganization, which is the first solution to the multi-perspective issues of medical knowledge in the world.

5. Related Work

In order to avoid multiple-inheritance, some researchers took an approach that they developed ontologies using single-inheritance and reorganized them by reasoning using a DL-reasoner [11]. It corresponds to reorganization of *is-a* hierarchy based on a transcriptional hierarchy in step 4 of the proposed method. However, the approach needs that the transcriptional hierarchy is developed in advance while it is dynamically generated by the system in the case of the proposed method.

Faceted Classification is used to represent classifications from multiple-perspectives. In the Semantic Web, some researchers proposed Faceted Search for semantic portals [12, 13]. They use Faceted Classification according to the user's choose of facets from the definition of ontologies to provide user-centric semantic search. In order to formalize the Faceted Classification, Bene Rodriguez-Castro proposed an ontology design pattern to represent Faceted Classification in OWL [14]. Although the proposed method use a similar technique to Faceted Classification for transcription of a hierarchical structure, it is different from Faceted Classification since we focus on considerations of ontological meaning of generated *is-a* hierarchies. Introduction of a p-*is-a* hierarchy is one of the results of the ontological investigations.

However, there are some rooms to ontological investigate on a method of dynamic *is-a* hierarchy generation. For instance, we need to investigate *is-a* hierarchies of role-concepts and role-holders [5] while this paper concentrated on *is-a* hierarchies of basic concept (normal type). Dynamic *is-a* hierarchy generation based on more complicated viewpoints is also important subject to be considered. For example, we are considering viewpoints to cope with a new disease model based on an ontological consideration of causal chains [9]. Because the latest version of our medical ontology based on the new disease model has more rich definitions than previous one, it would support more complicated viewpoints for dynamic *is-a* hierarchy generation based on causal chains in diseases. We believe these ontological considerations would clarify the feature of the proposed method.

6. Concluding Remarks

In this paper, we discussed multi-perspective issues of *is-a* hierarchy construction in ontologies and proposed a method of dynamic generation of *is-a* hierarchies. The main idea is reorganization of *is-a* hierarchies from the original ontology according to viewpoints of users. The proposed method was implemented as a new function of Hozo, and we applied it to our medical ontology for a preliminary evaluation. As a result, we confirmed that it could generate several kinds of *is-a* hierarchies from the medical ontology according to the user's viewpoints. As of April 12, 2011, 6051 diseases have been defined in the medical ontology by 12 clinicians, and these

definitions are currently being refined. The demonstration of the dynamic *is-a* hierarchy generation is available at http://www.hozo.jp/demo/. The function is also supported by the latest version of Hozo.

Currently, we are improving the dynamic *is-a* hierarchy generation function in order to support more detailed disease model which we proposed in [9]. We are also developing a dynamic *is-a* hierarchy generation system for OWL ontologies using OWL-API while it is partly available through OWL import/export function of Hozo. Future work includes ontological investigations of the proposed method and evaluation of the system through application of several ontologies.

Acknowledgement

A part of this research is supported by the Ministry of Health, Labour and Welfare, Japan, the Japan Society for the Promotion of Science (JSPS) through its "Funding Program for World-Leading Innovative R&D on Science and Technology (FIRST Program)", and Grant-in-Aid for Young Scientists (A) 20680009.

References

1. Gruber,T.: A translation approach to portable ontology specifications, Proc. of JKAW'92:89-108 (1992)
2. Guarino, N: Some Ontological Principles for Designing Upper Level Lexical Resources, Proc. of International Conference on Lexical Resources and Evaluation (1998)
3. Smith, B. et al.:The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration, Nature Biotechnology, 25(11):1251-1255 (2007)
4. Kozaki, K. et al.: Hozo: An Environment for Building/Using Ontologies Based on a Fundamental Consideration of "Role" and "Relationship", Proc. of EKAW2002,LNCS 2473:213-218 (2002)
5. Mizoguchi, R. et al.: A Model of Roles within an Ontology Development Tool: Hozo, J. of Applied Ontology, 2(2):159-179 (2007)
6. Kozaki, K., et.al.: Role Representation Model Using OWL and SWRL, In Proc. of 2nd Workshop on Roles and Relationships in Object Oriented Programming, Multiagent Systems, and Ontologies, Berlin, July 30-31, (2007)
7. Mizoguchi, R. et al.: An Advanced Clinical Ontology, Proc. of ICBO:119-122 (2009)
8. Hahn U, et al.: Turning Lead into Gold?, Proc. of EKAW2002, LNCS 2473:182-196 (2002)
9. Mizoguchi, R. et al.: River Flow Model of Diseases, Proc. of ICBO2011: 63-70 (2011)
10. Kou, H., Ohta, M. Zhou, J., Kozaki, K., Mizoguchi, R., Imai, T., and Ohe, K.: Development of Fundamental Technologies for Better Understanding of Clinical Medical Ontologies, Proc. of KEOD 2010:235-240, Valencia, Spain, October 25-28 (2010)
11. Nico Adams, Edward O. Cannon, Peter Murray-Rust: ChemAxiom -An Ontological Framework for Chemistry in Science, Proc. of ICBO:15-18 (2009)
12. Osma Suominen, Kim Viljanen, and Eero Hyvönen: User-centric Faceted Search for Semantic Portals, Proc. of ESWC2007, LNCS 4519 :356-370 (2007)
13. Markus Holi: Crisp, Fuzzy, and Probabilistic Faceted Semantic Search, PhD Thesis, Aalto University, Finland (2010)
14. Bene Rodriguez-Castro, Hugh Glaser, and Les Carr: How to Reuse a Faceted Classification and Put it on the Semantic Web, Proc. of ISWC2010, LNCS 6496:663–678 (2010)

MUTU: An Analysis Tool for Maintaining a System of Hierarchically Linked Ontologies

Sini Pessala, Katri Seppälä, Osma Suominen, Matias Frosterus, Jouni Tuominen, and Eero Hyvönen

Semantic Computing Research Group (SeCo)
Aalto University School of Science, Dept. of Media Technology, and
University of Helsinki, Dept. of Computer Science
http://www.seco.tkk.fi/, firstname.lastname@aalto.fi

Abstract. We consider ontology evolution in a system of light-weight Linked Data ontologies, aligned with each other to form a larger ontology system. When one ontology changes, the human editor must keep track of the actual changes and of the modifications needed in the related ontologies in order to keep the system consistent. This paper presents an analysis tool MUTU, by which such changes and their potential effects on other ontologies can be found. Such an analysis is useful for the ontology editors for understanding the differences between ontology versions, and for updating linked ontologies when changes occurred in other components of an ontology system.

1 Facilitating changes across ontologies

1.1 Position statement

Ontologies are often linked into larger ontology systems systems of a general upper ontology complemented by domain-specific ontologies. This enables that the experts of various domains can create their field-specific ontologies that co-operate with the other ontologies.

When updating the upper ontology, the changes require modifications in the domain ontologies. Since the domain ontology developers are not specialized in the upper ontology, the changes to it should be conveyed to the domain ontology developers in a readable form.

1.2 Introduction

Ontologies define concepts and relations between them in a machine-processable form [5]. The backbone of ontologies is typically based on *rdfs:subClass* relation, that can also be used in Linked Data for aligning ontologies hierarchically into larger systems. In such systems, more domain-specific ontologies extend the concepts of more general (upper) ontologies. This idea has been implemented in the Finnish Collaborative Holistic Ontology (KOKO)[1]. KOKO has the Finnish General Upper Ontology YSO [1] as its

[1] http://www.seco.tkk.fi/ontologies/koko/

upper ontology, aligned with 15 more specific ontologies in domains such as cultural heritage, agriculture and forestry, sea faring, photography, and public governance.

Domain ontologies are typically separately developed by different expert groups, since the process requires field-specific knowledge often found in separate communities and organizations. In KOKO, all component ontologies are based on existing keyword thesauri developed by such distributed, independent expert groups. However, the ontologies have mutual implicit relations with each other via shared concepts and relations. The added value of KOKO is to explicate such alignment relations with the upper ontology YSO, linking the ontologies into a coherent global hierarchical system.

This paper concerns the problems of ontology evolution in such a system of Linked Data ontologies. When an ontology has been edited, possibly by several persons in a team, the ontology editor needs tools for seeing the actual changes of the ontology with regard to the earlier versions and examining the possible consequences of the changes to other ontologies. We built an analysis tool MUTU for the ontology editor to address such changes. The idea in MUTU is to provide the domain ontology developer with an analysis of changes with regard to a prior version of her ontology, and support her in performing the required modifications, when changes have occurred in other related ontologies, in particular the upper ontology gluing the component ontologies together.

2 Tool Description: MUTU

2.1 Adressing the Changes of the Upper Ontology with MUTU

MUTU has been developed as an answer to a real demand in collaboration with ontology developers. MUTU lists the changes occurred at the update of the upper ontology as an HTML file, and marks the modified concepts of the updated upper ontology grouping them to different categories, based on the type of change. The groupings help the domain ontology developer to identify which concepts still need to be checked. After the developer has performed the required modifications to the domain ontology, the groupings can be removed. MUTU supports lightweight RDFS, SKOS[2] and OWL ontologies. MUTU is adapted for a new upper and domain ontology pair by creating a configuration file containing a list of the ontology namespaces and property URIs.

MUTU is based on the assumption that the URI of a concept is the same across different ontology versions, thus concepts are identifiable by their URIs. This means that when a label of the concept changes, no new concept is created.

Interesting Change Categories of the Upper Ontology MUTU divides the changes in an upper ontology into eight categories, which were determined in collaboration with ontology developers. A change can be either an addition or a removal of a concept or an addition, a removal or a replacement of a property value of a concept. It is assumed that the schema of the ontologies remains unchanged simplifying the process compared with a more general approach as in, e.g., [3]. However, all of the changes are not relevant to the developer of the domain ontology, since the domain ontology extends only some

[2] http://www.w3.org/2004/02/skos/

parts of the upper ontology. Thus, we define that a change in the upper ontology is interesting, if it will most likely cause update procedures to the domain ontology. This means that different subgroups of the upper ontology changes are interesting depending on the domain ontology and the connection between it and the upper ontology.

Generally, the changes nearest to the extended parts are interesting, but the domain ontology developer needs to examine all of the changes. By dividing the interesting and rest of the changes into different groups, the domain ontology developer could only check the interesting changes. The uninteresting changes are also available for acquiring a broader picture of the changes of the upper ontology.

The domain ontology is connected to the upper ontologies via RDF triples where the subject belongs to the upper ontology and the object to the domain ontology or vice versa. The concepts of the upper ontology participating in the described triples are called connecting concepts. Respectively, connecting concepts that are connected via an equivalency property are called equivalent-connecting concepts. Next, we introduce the change categories and define the interesting changes in these categories.

1. **Added and removed concepts:** Added and removed concepts are concepts that do not exist in both of the upper ontology versions. An added or removed concept is interesting when it is an ancestor of a connecting concept, since it should be checked that the reasoning of the hierarchy is valid for the connecting concepts of the domain ontology. Additionally, some added concepts might already exist in the domain ontology, so these concept pairs should be marked equal.
2. **Hierarchy and partonomy changes:** Hierarchy and partonomy changes are in the values of the hierarchy building properties, e.g., *rdfs:subClassOf* or *skos:broader*. The changes of this category are interesting if the concept is an ancestor of a connecting concept with a continuous chain of that property.
3. **Label changes:** Label changes are changes in the value of label-related properties, such as *skos:prefLabel* or *skos:altLabel*. The label changes are interesting, if the preferred label of an equivalent-connecting concept has been replaced, since then the preferred label of the domain ontology concept might need to be updated.
4. **Associativity and equivalency changes:** Associativity and equivalency changes are the changes of the properties that change additional connections between concepts. Examples of these properties are *skos:related* and *skos:exactMatch*. These changes are interesting if they occur in a connecting concept.
5. **Other changes:** Other changes contain the changes of the properties that were not mentioned above. They are interesting if they occur in a connecting concept.

Additional features In addition to the change categories listed above, the ontology developers requested additional features to aid their work. These features were reporting on multiple concepts with the same label and blocking of uninteresting changes in properties and concepts.

The label-matching feature compares all of the labels of the updated upper ontology with the labels of the domain ontology, detecting the unwanted situations of having the multiple occurrences of the same concept without any relation between them, e.g., *owl:sameAs*. If some labels match and there is no marked equivalency, then the domain

ontology developer should see if the concepts should be marked equivalent. Additionally, the labels are lemmatized, since depending on the ontology, some labels might be in plural and the others in singular.

Blocking an uninteresting property means that the changes of that property are ignored, and blocking an uninteresting concept means that changes to the subconcepts of the concept are ignored. This prevents flooding the change listing with the modifications of irrelevant properties, whose change would not cause any need for updating the domain ontology. As an example, label changes in languages that are not used in the domain ontology are not relevant.

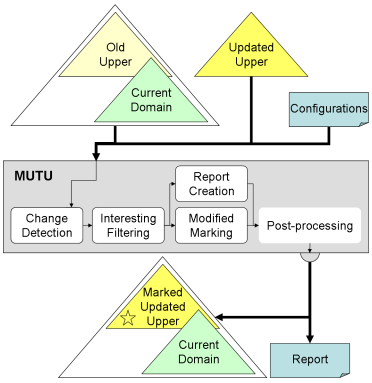

Fig. 1. Overview of the process of MUTU

MUTU Process The usage of MUTU is shown in Figure 1. The domain ontology developer inputs the current upper ontology, the old upper ontology, the domain ontology and a configuration file. MUTU detects the changes between the different versions of the upper ontology and detects the connecting concepts. Then MUTU separates the changes to interesting and the rest and prints this list to a report in HTML format. Finally, MUTU marks the modified concepts and outputs the HTML listing and marked upper ontology for the domain ontology developer.

The HTML listing contains the changes sorted by different modification types. By expanding one modification type, one can choose to expand either the interesting changes or the rest modifications and then see a list of the concepts with these changes and browse the changed property values. The marked upper ontology can be used in the ontology editor for keeping on track of which concepts are still unchecked. In the marked upper ontology, the changed concepts are marked as subclasses for structuring concepts representing different modification types. The changes are marked in the ontology so that the subconcepts of one modification type are the same as in the HTML listing.

2.2 Case Study: YSO and LIITO

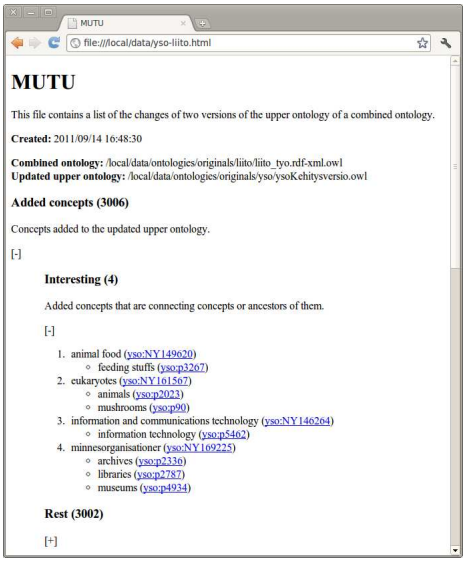

Fig. 2. The aFdded, interesting concepts of the HTML change list. The bulletin-listed concepts are the nearest connecting concept children of the added concept.

In our case study, we used the Finnish General Upper Ontology YSO [1] as the upper ontology and the Business Economy Ontology LIITO[3] as the domain ontology. Both ontologies are in active development and there is a new release of YSO quarterly. LIITO contains almost 3200 concepts, the old version of YSO contains 20700 concepts and the updated YSO contains 23600 concepts.

We have not yet conducted a formal evaluation of MUTU, but the first results of using MUTU with YSO and LIITO are promising. The outputs, the change list and marked updated upper ontology, were considered useful among the domain ontology developers. It was easy for the developers to see what changes had been made. A screenshot of the interesting added concepts are shown in figure 2.

When examining the change lists, some interesting findings were made: somewhat over 10% of the hierarchy changes occurred near to the connecting concepts. Most of these changes were of hierarchically close concepts. Another interesting finding was that since the upper ontology has not been updated before, over 500 concepts added to YSO that had labels similar to those in LIITO. This means that over 15% of the concepts of LIITO exist in the updated upper ontology without any marked connections to the concepts in the YSO. Nevertheless, this does not mean that all of these concepts

[3] http://onki.fi/en/browser/overview/liito

are duplicates, since for example “The Museum of Modern Arts” could mean a building in the upper ontology and an organisation in the domain ontology.

LIITO is developed using the Protégé[4] ontology editor, and for aiding the development process, the source file of LIITO contains a copy of YSO. The YSO copy does not automatically update when the original YSO updates, thus the old YSO version is replaced with a copy of the updated YSO version in the post-processing phase of MUTU.

LIITO and YSO contain labels in Finnish, Swedish, and English, so the label changes are divided into subgroups according to the languages to ease browsing.

3 Discussion and Related Work

Contributions This paper rises the issue of updating systems of separately-developed ontologies. To address this, we presented a tool for detecting the changes in an upper ontology and listing the relevant changes for the developer of the domain ontology. This supports the developer when reflecting the changes of the upper ontology into the domain ontology.

Related Work Changes and evolution of a single ontology is a widely researched area, see for example [4][7]. However, the situation where the changes in the upper ontology need to be reflected in the domain ontology has not been researched as extensively. Maedche & al. [2] discuss the evolution of distributed ontologies depending on other ontologies. They use change logging and when the domain ontology requests the modifications of the upper ontology, the changes of the upper ontology are merged to the change log of the domain ontology. Their main goal was to ensure the consistency of the ontologies, where as we are concerned of conveying the changes to the domain ontology developer.

Future work This paper described only the preliminary results of the idea and testing of MUTU, and its web interface is still under development. We intend to put the system to use with the domain ontologies of YSO. The web interface of the MUTU will be integrated into the ONKI ontology service[5] [8]. After the integration is complete, we will evaluate the application in collaboration with the actual domain ontology developers. The evaluation will be that the domain ontology developers use MUTU in their normal updating tasks and then analyze their use experiences with MUTU and give proposals for improvements.

In addition, we will enhance the change categories and interesting changes with the feedback and group single changes to more human-understandable composite changes similarly to the work of Stojanovic et al. [6]

[4] http://protege.stanford.edu/
[5] http://onki.fi/

Acknowledgements This work is part of the National Semantic Web Ontology project in Finland[6] FinnONTO (2003–2012), funded currently by the National Technology and Innovation Agency (Tekes) and a consortium of 35 public organizations and companies.

References

1. E. Hyvönen, K. Viljanen, J. Tuominen, and K. Seppälä. Building a national semantic web ontology and ontology service infrastructure—the FinnONTO approach. In *Proceedings of the European Semantic Web Conference (ESWC 2008)*, 2008.
2. A. Maedche, B. Motik, and L. Stojanovic. Managing multiple and distributed ontologies on the Semantic Web. *The International Journal on Very Large Data Bases (The VLDB Journal)*, 12(4):286–302, 2003.
3. A. Maedche, B. Motik, L. Stojanovic, and N. Stojanovic. User-driven Ontology Evolution Management. *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web*, pages 133–140, 2002.
4. P. Plessers, O. De Troyer, and S. Casteleyn. Understanding ontology evolution: A change detection approach. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(1):39–49, 2007.
5. Staab S. and Studer R., editors. *Handbook on Ontologies (2nd Edition)*. Springer–Verlag, 2009.
6. L. Stojanovic, N. Stojanovic, and S. Handschuh. Evolution of the metadata in the ontology-based knowledge management systems. In *German Workshop on Experience Management*, volume 2002, pages 65–77, 2002.
7. M. Tury and M. Bieliková. An approach to detection ontology changes. *Workshop proceedings of the sixth international conference on Web engineering (ICWE 2006)*, 2006.
8. K. Viljanen, Jouni T., and E. Hyvönen. Ontology Libraries for Production Use : The Finnish Ontology Library Service ONKI. In *Proceedings of the 6th European Semantic Web Conference (ESWC 2009)*, 2009.

[6] http://www.seco.tkk.fi/projects/finnonto/

An Ontology-Based Feature Recognition and Design Rule Checker for Engineering

Luis Enrique Ramos García[1] , Alexander Garcia[2] ,John Bateman[1]
[1] University of Bremen, FB 10
Cartesium, Enrique-Schmidt-Strasse. Bremen, Germany
bateman@uni-bremen.de, s_7dns7r@uni-bremen.de
[2] University of Arkansas for Medical Sciences
Little Rock, Arkansas, USA
alexgarciac@gmail.com

Abstract. This paper describes the design and implementation of an Ontology-Based System for Features Recognition and Design Rules Checking in the domain of sheet-metal engineering using Semantic Web technologies. The system was implemented by means of the Protégé Application Programming Interface (API), a rule engine and a reasoner. Using ontology in the core of the system enabled the representation of manufacturing rules and Automatic Features recognition. Rules and Queries were not hard coded in the program, giving the system a high level of maintainability and reusability. Features were classified as general and specific, easing the work of classifying newer features as they appear given their previous classification. Most common sheet metal features referred to in the literature were recognized by the system.

Keywords: Semantic Web, OWL, SWRL, SQWRL, CAD, CAM, CAPP

1 Introduction

Sheet metal parts are elements commonly used in several products manufactured in modern industry; aerospace, automotive, appliances, machine tools, etc. are only some of these. Although various software tools are available for making digital designs for such metal parts, trying to use those designs as input for process planning and manufacturing is not straightforward. The majority of standards for Computer Aided Design (CAD) do not represent certain features that are needed when manufacturing. For instance, declaring when a circle corresponds to an inner edge or an outer edge has significant consequences during manufacturing although not necessary when displaying designs. Facilitating the interoperability across the CAD-CAM (**C**omputer **A**ided **M**anufacturing) process should make it possible for designers to select optimal manufacturing conditions. For instance, to choose a sheet thickness that would prevent crashes when punching holes.

In order to achieve such interoperability, manufacturing features must be extracted from CAD files. Extracting these features entails identifying certain patterns from the CAD files that are to receive additional manufacturing-relevant enrichment. This information about features is used *a posteriori* to determine the machining tools and

manufacturing processes required to manufacture a given design [1]. Feature information can also then be used to pre-check designs in order to detect production rules violations. If these violations are not detected in the early stage of design, the life cycle of its development increases; raising both production costs and time to market [2]. Manual recognition of the targeted features is not a viable alternative; Automated Features Recognition (AFR) is thus required. However, even nowadays AFR is not fully integrated with CAD software tools. It is applicable only to parts with relatively simple geometry and still requires human intervention to obtain the features identified. Moreover, such techniques are generally supported only on expensive CAD software tools that are beyond the reach of small-scale industry [3].

To address these shortcomings of AFR from sheet metal designs and design checking, in this paper we propose an ontology-based framework that facilitates interoperability across the entire CAD-CAM process. Our system integrates both a CAD and a feature ontology; these ontologies were written in OWL (Web Ontology Language) and represent 2D primitives as lines, arcs and circles in the CAD ontology and as edges, slots, tab and holes in the features ontology. In order to provide rules and query support, OWL was complemented with SWRL [4] and SQWRL [5]. The result is an ontology-based system that allows us to automatically extract the most common manufacturing features referred to in the literature.

The paper is organized as follows. We begin with related work in Section 2. Section 3 then describes the components in the architecture of the implemented system. Section 4 shows the implementations and some results of the evaluation on a sample design and Section 5 concludes with an outlook for future work.

2 Related work

2.1 AFR & Design Advisory Systems

Henderson & Anderson [6] reported on early experiences with the use of production rules that were not hard-coded into the system using Prolog, while Meeran & Pratt [7] proposed to extract features from 2D drawings to generate process planning and machining instructions from CAD. The latter approach was also based on logic programming, adopting Prolog as the implementation language. These authors mentioned that Prolog presented certain limitations when dealing with trigonometric functions.

Other authors, such as Vasilakis [8] and Krima *et al* [9], did not use logic programming. Rules were represented instead as a set of nested IF <*list of conditions*> THEN <*hypothesis*><*list of actions*>. In the case of Vasilakis [8], when a design rule was violated the system showed the rule violated but did not suggest any corrective action. Radhakrishnan [2] integrated a rule checker within a design adviser; this approach identifies distance violations amongst features. The general tendency in these contributions was the development of feature recognition and design advisory systems with hard-coded rules.

In summary, two main classes of approach can be identified from the literature. On the one hand there are approaches based on logic programming. Here, rules are separated from code, facilitating interoperability, shareability and maintainability; these approaches have generally used Prolog as implementation language. On the other hand, there are also approaches relying on procedural programming, commonly using C and C++ as implementation languages. As these are not rule-based, maintainability and share-ability are limited. Babic *et al* [10] consider that emerging approaches should be hybrids.

2.2 Ontology for CAD

The limitations of CAD software tools have motivated researchers to develop frameworks that overcome those limitations and some of these have already adopted ontological approaches. Vasilakis & Andersen [11] and Krima *et al* [12], for example, developed ontologies for the Standard for the Exchange of Product model data (STEP). Here, Krima *et al* [12] acknowledged the limitation of STEP and accordingly developed OntoSTEP. OntoSTEP is based on OWL precisely because of its support of logic reasoning and inference mechanisms.

Approaches addressing the problem of interoperability across CAD standards have been explored by Ghafour *et al* [13], Sun & Ding [14] and Ramos [15]. Ghafour *et al.* proposed a common Design Features Ontology written in OWL and one domain ontology from each specific standard, an approach similar to that of Sun & Ding. Ramos focused on an ontology for CAD primitives populating a CAD ontology. In addition, Grüninger & Delaval [16] developed an ontology for the cutting process of sheet metal parts; this approach was proposed as a theoretical step from which the planning process could be supported. Franke *et al* [17] developed an ontology-based tool for quality control in CAD design. They aimed to verify cases in which overlapping occurred in order to determine if this constituted a design mistake. Their architecture included their own software tool (OntoDMU) and the system HETS [18].

2.3 Sheet Metal Features and Rules for Manufacturing

Defining features in the manufacturing domain is not straightforward [1]; a feature is understood here as a part of a mechanical design that has a specific functionality and that is semantically significant within the manufacturing process. Farsi & Arezoo [19] classified sheet metal features according to whether those features were internal or external; internal features are holes and external features are different types of notches. Radhakrishnan *et al* [2] classified rules into two types: intrafeature rules and interfeature rules. Interfeature rules are related only to the feature itself and its constraints, indicating minimum values of its dimensional parameters. Interfeature rules describe restrictions across features and contours. In Fig. 1 some of the most common features and rules found in the literature are described. The figure includes features names and some of their most important parameters. The indicated rules are related directly with the production process. It is necessary to take such rules into account in order to provide the necessary structural strength to the design´s features. Improperly formed features will contribute to waste of material in the finished products due to buckle, cracks or breaks.

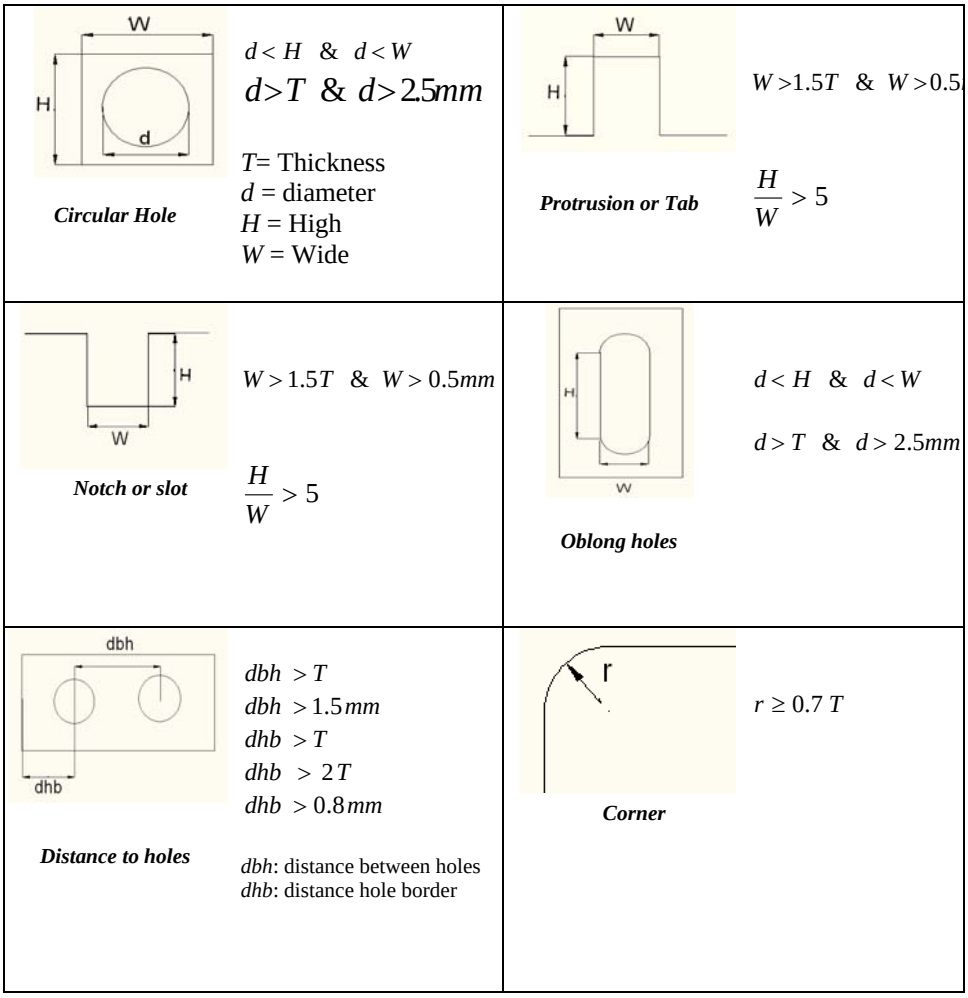

Fig. 1. Manufacturing features with their constraints

3 An Ontology-Based AFR and Design Checker system

The approach developed here builds on the state of the art we have seen and takes this further, relying on a modular architecture that makes use of the Protégé API. In the following subsection we describe the approach's components and the way in which they interact with each other. Although the organization and classes of the two ontologies are often comparable, it is important to see that they reflect fundamentally different *kinds* of ontological information. To the extent that similarities are found, then this renders the subsequent mapping process more straightforward, but there is

no requirement that the ontologies correspond precisely. Since the identification criteria for the CAD and the features ontologies are quite distinct—the former relating to descriptions of design, the latter to manufacturability —it is both theoretically and architecturally cleaner to maintain them separately. We return to this important consideration and the issues of perspectives below.

3.1 A CAD Ontology

The first component to be discussed is the CAD ontology. During the development of this ontology several CAD formats, such as DXF [20] and IGES [21], were considered. The resulting CAD ontology is under continued revision and update. Fig. 2 illustrates the current state of the CAD Ontology: http://bit.ly/q8baMm.

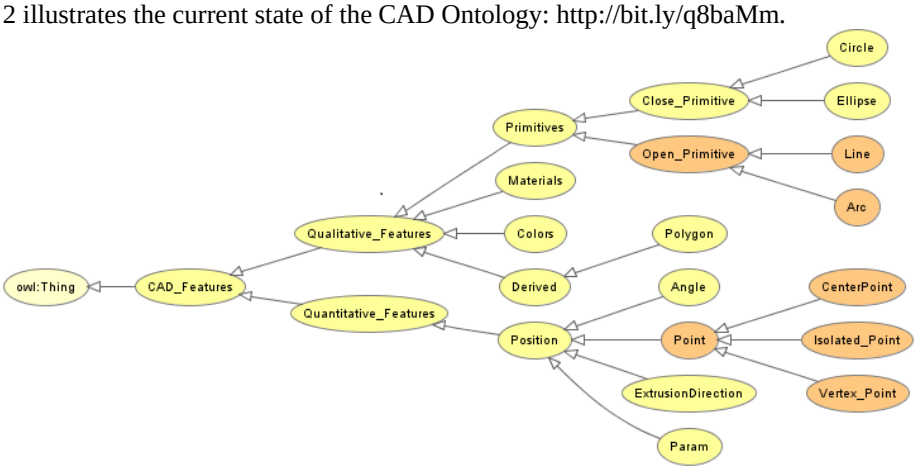

Fig. 2. CAD Ontology, http://bit.ly/q8baMm

As is illustrated in the figure, we divided *CAD_Features* into the two subclasses: *Qualitative_Features* and *Quantitative_Features*. Among the former, *Primitives* can be either *Open* or *Closed*. Under these classes, *Line*, *Arc*, *Circle* and *Ellipse* are understood as *Primitive* elements that are present in any CAD standard that provides at least a 2D representation. *Derived* elements are then Qualitative Features that may vary and are standard dependent; they appear as a combination of certain primitive elements. For example, a *Vertex* appears when two primitive lines are connected at one end. Definition of concepts, such as *Vertex*, is dimensional-space dependent.

3.2 Ontology of Features

For the development of the ontology of features, available at http://bit.ly/pyQGEE, we considered information such as that shown in Fig. 1. An extract of the resulting ontology is presented in Fig 3. There are two main concepts: these are *Shape_Features* and *NonGeometry*. Shape_Features was divided further into *Qualitative_Features* and *Quantitative_Features*, analogously to the top-level distinction in the CAD ontology. *Named_Feature* involves features found in the literature that can be precisely defined and identified using their properties or attributes. *Atomic_Feature* is then a *Named_Feature* consisting of exactly one

identifiable element which has meaning in a manufacturing domain. These elements can have *Open* or *Closed* edges.

Composed_Features include features that appear as a combination of *Atomic_Features*. *Partial_Composed* includes identifiable features that need to be part of a whole to be consistent. *Total_Composed* covers a feature that exists as a whole. *Partial_Composed* features have to be part of a *Total_Composed* feature. *hasPart* and *isPartof* were included following a standard mereological [22] point of view and its terminologies and definitions. Properties such as *properWidth*, *properHeight*, *properCircularHole* were included as part of SWRL rules. When these rules are evaluated, results are bound to true or false. If results are all true, the design passes its evaluation; otherwise the design is flagged as needing to be checked.

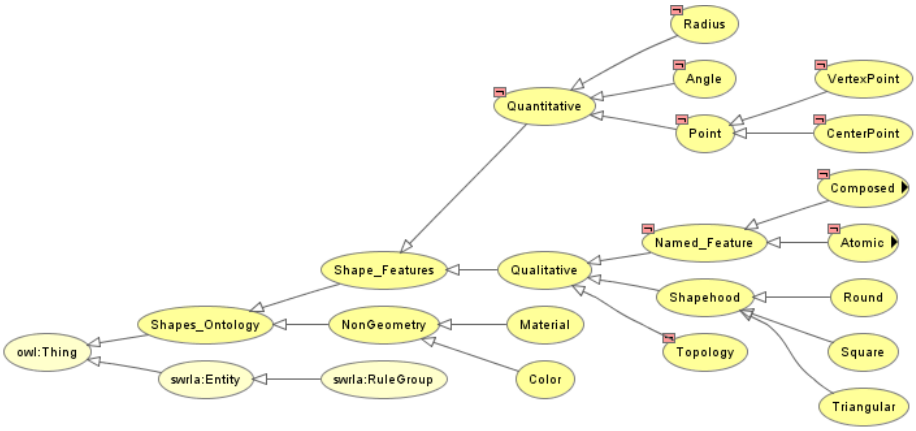

Fig. 3. Sheet features Ontology, http://bit.ly/pyQGEE

3.3 Mapping two domains of interpretation

One of the main issues of AFR is the level of abstraction involved in the manufacturing domain [23]. On the one hand, designers make designs from the functionality and usability point of view, with its own constraints and rules. On the other hand, manufacturing engineers evaluate a design from the manufacturing point of view, considering factory constraints. It is for this reason that, in general, quite distinct standards, formats and tools have been developed. In the two domains, two different interpretations of the requirements take place for the *same object*, so an interoperability channel between them is needed. This channel should facilitate the interchange of information about products as well as the evaluation of designing and manufacturing constraints.

In Fig. 4, we introduce the framework implemented for dealing with this situation. Knowledge transfer between domain ontologies has been facilitated by means of a third, mapping ontology, which keeps track of the related concepts of both domains. This mapping ontology was developed using the Prompt plug-in [24]. Given that in many cases the difference between the domains can be reduced to

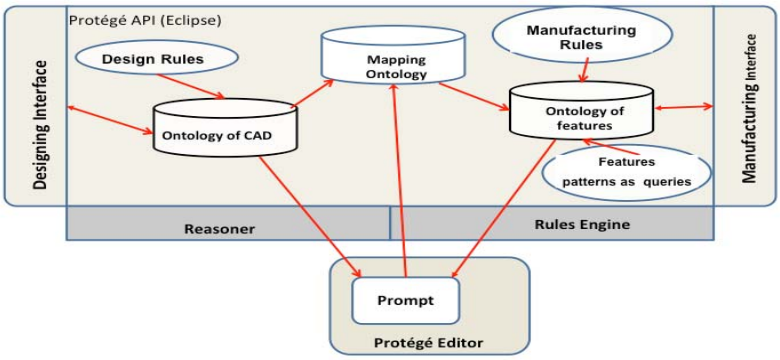

Fig. 4 System Architecture

terminological issues (factoring out often implicit identity criteria), we first made an internal comparison [25]. This consisted of comparing concepts based on attributes and properties. The ontology of CAD and the ontology of features were used as inputs and a mapping ontology was the output. Mapping was semi-automatic because the algorithm did not detect all relationships; human intervention was needed to complete the process. This preliminary mapping already gave us the necessary information to generate individuals of the features ontology and then to transfer the knowledge about individuals and their properties from the ontology of CAD to the ontology of features in a workable fashion.

3.4 Feature Patterns as Queries

Features were recognized by querying the features ontology with SQWRL. These queries were performed at two levels of abstraction. A first set of queries was implemented to facilitate a general classification of features and a second set was executed to facilitate specialized classification. These two sets of queries reduced the impact of not extracted or unidentified features referred to in the literature [26]. Within the first set of queries, all given *atomic features* were classified as *is part of*, a general pattern. Subsequently, having assigned all *atomic features* to some general pattern, we proceeded to identify specific patterns. If some atomic feature was not classified as *is part of* some specific pattern, then a new specific pattern had to be added.

For example, a protrusion pattern is formed by three connected edges having a specific slope. The query for extracting this pattern was expressed as follows:

```
Open_Primitive(?l1) ∧ Slope(?l1, "-0.0") ∧
Open_Primitive(?l2) ∧  Slope(?l2, "Infinity") ∧
Open_Primitive(?l3) ∧ Slope(?l3, "+0.0") ∧
hasendpoint(?l1, ?p1) ∧ hasstartpoint(?l2, ?p1) ∧
hasendpoint(?l2, ?p2) ∧ hasstartpoint(?l3, ?p2) ˚
```

```
sqwrl:makeSet(?s, ?l3)
→ sqwrl:select(?s, ?l1, ?l2, ?l3)
```

The number of sets obtained and their composition allows the generation of composed features. Once the query has been executed, composed features are related to atomic features by means of the *has-part* property.

3.5 Design and Manufacturing Rules Checking

Although AFR research is receiving considerable attention, a recognized feature still does not indicate anything about its quality. This aspect is fundamental in order to reduce the cost of manufacturing a given design [27]. To this end, two sets of SWRL were implemented to check manufacturing rules. The first set allowed us to enrich the features ontology with the necessary engineering information about features. For instance, the width of a *Partial Composed* feature was calculated with the following rule:

```
PartialComposed(?pc) ∧ hasEdge(?pc, ?e) ∧ slope(?e,
"0.0") ∧ hasEndPoint(?e, ?p2) ∧ hasYcoordinate(?p2,
?y2) ∧ hasStartPoint(?e, ?p1) ∧ hasYcoordinate(?p1,
?y1) ∧ swrlb:subtract(?v1, ?y2, ?y1) ∧ swrlb:abs(?v1) →
width(?pc, ?v1)
```

The second set of rules was implemented for checking the design. For instance, given the thickness of a given material, a minimum width is mandatory to make its manufacturability feasible. For some materials the width must be 1.5 times greater than the given thickness. This rule was expressed in SWRL as follows:

```
PartialComposed(?pc) ∧ width(?pc, ?w) ∧ heigh(?pc, ?h)
∧ thickness(?pc, ?t) ∧ swrlb:multiply(?v1, ?t, 1.5) ∧
swrlb:greaterThan(?w, ?t) → proper_width(?pc, true)
```

3.6 Reasoning framework

The reasoning capabilities of the Semantic Web framework gives us the possibility of classifying instances of classes against the ontologies of our application. Our reasoning was done in two steps. Firstly, using SWRLTab [28], rules and queries were edited, implemented and debugged. JESS [29], a rule engine for JAVA platforms, has been included in Protégé for enabling this tab. Each OWL Model and its corresponding SWRL rules were loaded into the JESS engine. The JESS engine was then run and the resulting inferences were uploaded into the OWL model. Thus, the amount of knowledge encoded in the OWL model was augmented. Results of querying the models were also considered for creating new individuals. This result was finally compared with the design in order to determine if the result was consistent with it. Here we used Pellet [30] for the automatic verification of the consistency of the OWL model and for classifying the instantiated features.

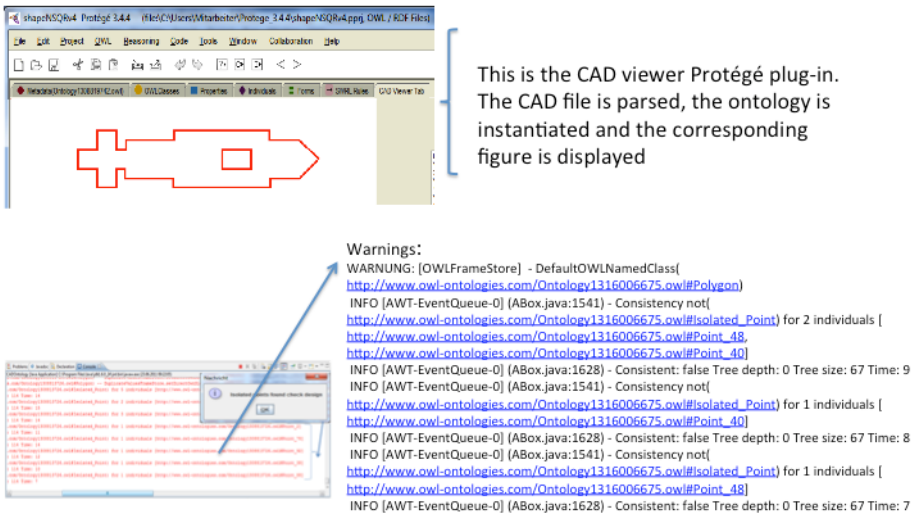

Fig. 5. Sample shape

Secondly, using the SWRL API [31], the OWL Reasoning API [32] and the Protégé API, a prototype was developed. *SWRLJessBridge*, *SWRLQueryAPI and ProtegePelletJenareasoner* were also used in this implementation; the overall architecture is presented in Fig 4. Rules and the results of running the rules were loaded into models at run time. Queries were made to the OWL models, and the results were again imported so as to add new knowledge into the OWL models. Finally, the reasoner verified the consistency of these ontologies and classified instanced features. Resulting from this process we obtained an OWL model of features containing all the necessary information pertaining to recognized features with their quality; our model also includes non-identified features.

4 Implementation and Results

The implementation was tested with several designs of shapes. In Fig. 5 one of these is displayed by means of a CADViewer plug-in for Protégé [15]. Our first evaluation of the design consisted of diagnosing the topological correctness of the model and verifying the connectivity of edges and vertex [33]. For this verification, we defined *Isolated_Point* as a subclass of *Point.* After populating the CAD ontology, the reasoner was invoked. If instances of *Isolated_Point* are found then the design is

considered as inconsistent; such a case is illustrated in Fig. 6. As soon as the problem was corrected, the system continued with the extraction of features and their validation.

The features described in Fig. 1 were recognized and extracted into the features ontology. Manufacturing rules were run on the OWL model adding newer knowledge. This new knowledge was then used by the reasoner to classify features into sets of proper and improper features.

5 Conclusions and future work

AFR is a research and application area devoted to bridging the gap between design and manufacturing. Most of the approaches used in AFR are divided into procedural and declarative approaches. We have developed an Ontology-Based AFR and Design Rules Checker System that combines the advantages of both approaches. An ontology of CAD and an ontology of features are the fundamental components of our system. The necessary interoperation between both ontologies was achieved by means of a mapping ontology generated in a semi-automatic manner using the Prompt Protégé plug-in.

SWRL was used to perform engineering calculations in order to add appropriate semantics to the features ontology. SQWRL was used for modeling feature patterns commonly referred to in the literature. Sheet metal features were extracted as the result of queries applied to the features ontology. SQWRL queries and SWRL rules were written at several levels of abstraction in order to make a progressive identification of features possible. By using such combination, SQWRL and SWRL, we have identified a significant number of features.

In this paper we have demonstrated that complex rule of mechanical and manufacturing engineering processes can be expressed using OWL and SWRL. Similarly, we have demonstrated that complex engineering information can be retrieved from the ontology using the SQWRL language. These rules and queries were also used for inferring knowledge and driving appropriate decision-making information. New features, rules and patterns can be integrated into our system by means of standard ontology editors. In the near future, we plan to integrate a recommendation ontology. Thus, as soon as a design violation is found, a recommendation will be presented to the user. We will also investigate how to deal with intersecting mechanical features.

The resulting set of recognized features could be used as an input in Computer Aided Process Planning (CAPP) systems, from which process planning will be obtained. We will continue this line of development as a Semantic Web CAPP system, focusing on the accuracy of the kind of planning generated with the Semantic Web Technologies taking into consideration other aspects of the engineering environment and investigating scalability.

References

1. Cayiroglu, I.: A new method for machining feature extracting of objects using 2D technical drawings. Comput. Aided Des. 41, 1008-1019 (2009).
2. Radhakrishnan, R., Amsalu, A., Kamran, M., Nnaji, B.O.: Design rule checker for sheet metal components using medial axis transformation and geometric reasoning. Journal of Manufacturing Systems. 15, 179-189 (1996).
3. Kumar, S., Singh, R.: Trends and Developments in Intelligent Computer Aided Design of Progressive Dies. AMR. 6-8, 241-248 (2005).
4. Horrocks, I., Patel, P.F.-S., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: A Semantic Web Rule LanguageCombining OWL and RuleML, http://www.w3.org/Submission/SWRL/#7.1, (2004).
5. Amar Das, M.O.: SQWRL: a Query Language for OWL. 6th International Workshop on OWL: Experiences and Directions (OWLED 2009). Vrije Universiteit Amsterdam, Chantilly, VA, United States (2009).
6. Henderson, M.R., Anderson, D.C.: Computer recognition and extraction of form features: A CAD/CAM link. Computers in Industry. 5, 329-339 (1984).
7. Meeran, S., Pratt, M.J.: Automated feature recognition from 2D drawings. Computer-Aided Design. 25, 7-17 (1993).
8. de Sam Lazaro, A., Engquist, D.T., Edwards, D.B.: An Intelligent Design for Manufacturability System for Sheet-metal Parts. Concurrent Engineering. 1, 117 -123 (1993).
9. Soman, A., Padhye, S., Campbell, M.I.: Toward an Automated Approach to the Design of Sheet Metal Components. AI. 17, 187-204 (2003).
10. Babic, B., Nesic, N., Miljkovic, Z.: A review of automated feature recognition with rule-based pattern recognition. Computers in Industry. 59, 321-337 (2008).
11. Andersen, O., Vasilakis, G.: Building an Ontology of CAD Model Information. Springer Berlin Heidelberg (2007).
12. Krima, S., Barbau, B., Fiorentini, X., Sudarsan, R., Sriram, R.: OntoSTEP: OWL-DL Ontology for STEP, http://www.nist.gov/customcf/get_pdf.cfm?pub_id=901544.
13. Ghafour, Abdul, S., Ghodous, P., Shariat, B., Perna, E.: An Ontology-based Approach for “Procedural CAD Models” Data Exchange. Proceeding of the 2006 conference on Leading the Web in Concurrent Engineering: Next Generation Concurrent Engineering. pags. 251–259. IOS Press, Amsterdam, The Netherlands, The Netherlands (2006).
14. Sun, L.-juan, Ding, B.: Ontology-based Semantic Interoperability among Heterogeneous CAD Systems. Information Technology Journal. 9, pp. 1635 - 1640 (2010).
15. Ramos, L.: Ontological CAD Data Interoperability Framework. Presented at the SEMAPRO 2010 , Florence, Italy Octubre (2010).
16. Grüninger, M., Delaval, A.: A First-Order Cutting Process Ontology for Sheet Metal Parts. Proceeding of the 2009 conference on Formal Ontologies Meet Industry. pp. 22-33. IOS Press (2009).
17. Franke, M., Klein, P., Schröder, L.: Ontological Semantics of Standards and PLM Repositories in the Product Development Phase. Proc. 20th CIRP Design Conference 2010. Alain Bernard.
18. Mossakowski, T., Maeder, C., Lüttich, K.: The Heterogeneous Tool Set, Hets, (2007). In TACAS 2007 (2007), O. Grumberg and M. Huth, Eds., vol 4424 of LNSC, Springer, pp. 519 - 522.
19. Farsi, M., Arezoo, B.: Feature Recognition and AND Design Advisory Systemfor Sheet Metal Components. Presented at the International Advanced Technologies Symposium , Turkey Mayo 13 (2009).
20. Autodesk: DXF Reference, http://images.autodesk.com/adsk/files/acad_dxf1.pdf, (2009).

21. U.S. Product Data Association, U.S.P.D.A.: Initial Graphics Exchange Specification 5.3, http://www.uspro.org/documents/IGES5-3_forDownload.pdf, (1997).
22. Varzi, A.: Mereology, http://plato.stanford.edu/entries/mereology/.
23. Shah, J.J., Rogers, M.T.: Functional requirements and conceptual design of the feature-based modelling system. Computer-Aided Engineering Journal. 5, 9-15 (1988).
24. Noy, N.F., Musen, M.A.: The PROMPT suite: interactive tools for ontology merging and mapping. International Journal of Human-Computer Studies. 59, 983-1024 (2003).
25. Euzenat, J.: State of the art on ontology alignment, http://www.starlab.vub.ac.be/research/projects/knowledgeweb/kweb-223.pdf, (2004).
26. Marchetta, M.G., Forradellas, R.Q.: An artificial intelligence planning approach to manufacturing feature recognition. Computer-Aided Design. 42, 248-256 (2010).
27. Li, X., Yoo, S.B.: Integrity validation in semantic engineering design environment. Computers in Industry. 62, 281-291 (2011).
28. Martin, O.: SWRLTab, http://protege.cim3.net/cgi-bin/wiki.pl?SWRLTab, (2011).
29. Friedman-Hill, E.: the Rule Engine for the JavaTM Platform. Sandia National Laboratories (2008).
30. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. Web Semant. 5, 51–53 (2007).
31. O'Connor, M., Nyulas, C., Shankar, R., Das, A., Musen, M.: The SWRLAPI: A Development Environment for Working with SWRL Rules. Proceedings of the International Workshop on OWL: Experiences and Directions (OWLED 2008) (2008).
32. Protege-OWL Reasoning API, http://protegewiki.stanford.edu/wiki/ProtegeReasonerAPI, (2009).
33. Tanaka, F., Kishinami, T.: STEP-based quality diagnosis of shape data of product models for collaborative e-engineering. Computers in Industry. 57, 245-260 (2006).

Socio-technical Ontology Development for Modelling Sensemaking in Heterogeneous Domains

Dhavalkumar Thakker, Fan Yang-Turner, Lydia Lau, Vania Dimitrova

School of Computing, University of Leeds
Leeds LS2 9JT, United Kingdom
{D.Thakker, F.Yang-Turner, L.M.S.Lau, V.G.Dimitrova}@ leeds.ac.uk

Abstract. Sensemaking is often associated with processing large or complex amount of data obtained from diverse and distributed sources. With information explosion from the web, sensemaking is becoming ubiquitous and ever more challenging. Semantic technologies have potential to support understanding of sensemaking process with the benefits they bring (e.g. reasoning, aggregation, automation). Conceptual models of sensemaking have been developed to understand its complex processes by social and information scientist. However, these frameworks are not applicable directly to system design. This paper describes a socio-technical approach for modelling sensemaking process in order to inform the development of intelligent services to aid sensemakers. We apply an *a priori* ontology modularisation methodology for handling complexity of heterogeneous domains and utilise well-known sensemaking theoretical framework to guide ontology development. This approach is applied in an EU project - Dicode, for the development of its sensemaking ontology.

Keywords: Ontology, Sensemaking, Modularisation, Semantic Annotation

1 Introduction

Semantic technologies, underpinned by ontologies, have been seen as one of the promising platforms for developing knowledge management systems [1, 2]. Examples of successful ontology developments can be found in a diverse range of domains such as multimedia [3] and life sciences [4, 5]. In these relatively well-defined and well-researched domains, ontological representations enhance the machine's reasoning capability on those knowledge bases.

With the recent proven successes of semantic web and ontologies, the field is ready to take on challenges offered by complex social-oriented domains which are less well-defined or scoped. Sensemaking is such a domain that involves cognitively-complex processes carried out by human and often requires injection of tacit knowledge. Moreover, sensemaking encompasses a range of behaviour surrounding the collection and organisation of information, may be across domains, for better understanding of a situation. Therefore, it is very challenging to derive a systematic and thorough understanding of the sensemaking processes from domain experts using traditional knowledge elicitation techniques.

Conceptual models of sensemaking have been developed to understand its complex processes by social and information scientists [6, 7, 8]. However, the problem of understanding and supporting sensemaking via technology remains challenging [9]. Initial work has already started in utilising semantic technologies for aiding sensemaking process in the domains of linked data [10], visualisation [11] and e-health [12]; which focus on applications that serve sensemaking rather than modelling sensemaking as a generic process.

This paper proposes a socio-technical approach for the development of an ontology which models sensemaking process in order to inform the design of intelligent aids to sensemakers. This is motivated by the vision of an EU project - Dicode[1], which aims to provide synergy between human and machine intelligence in collaboration and decision making within data-intensive environments. Theoretical frameworks on sensemaking, combined with an *a priori* ontology modularisation methodology, are used to guide the ontology development for sensemaking in heterogonous domains.

The paper is organised as follows. Section 2 explores the domain of sensemaking and the issues to be considered in developing a sensemaking ontology. Our socio-technical approach is proposed in section 3. Section 4 illustrates the application of the socio-technical approach in Dicode for the development of a multi-layered **Dicode ONtology** (referred as **DON** afterwards) for sensemaking. To better understand the potential benefits of semantics (e.g. using the ontology for reasoning, aggregation, automation) for applications in sensemaking domains, a proof-of-concept prototype "***Augmentor***" has been developed and discussed in Section 5. The concluding section summarises our contribution and future work.

2 Sensemaking: A Case Study

Sensemaking, as in "to make sense", is a process of transforming information into a knowledge product [8]. Sensemaking process involves interplay between foraging for information and abstracting the information into a representation called a schema that will facilitate a decision or solution [6]. It is often associated with processing large or complex amount of data obtained from diverse and distributed sources. There has been a recent increase of interest in sensemaking driven by the information explosion from the web that has rapidly changed our ability to assess large amounts of information [20].

Dicode project is aimed at supporting sensemaking and decision making in data-intensive and cognitively-complex settings. The solution foreseen in the Dicode project will bring together the reasoning capabilities of both the machine and the human. There are three use case partners involved to validate the transferability of Dicode solutions. They are from three different domains: (1) **Clinico-Genomic (CG)** research where clinical research professionals collaborate to explore scientific findings related to breast cancer using very large datasets; (2) **Rheumatoid Arthritis (RA)** clinical trial where medical personnel involved in the clinical trials collaborate and exchange their professional judgment within complex clinical decision making

[1] http://www.dicode-project.eu/

processes; and (3) **Public Opinion (PO)** monitoring where analysts watch social media to monitor public perceptions of their clients' branding, products or services.

These three use case partners were selected to address common challenges in sensemaking and decision making. All use cases experience the problem of information overload; all require sensemaking towards decision making based on cognitively intensive analysis and interpretation of data; all need to discuss and share interpretation and decision making rationale between specialists. They cover the full range of features and functionalities to be addressed by the project, from various sectors and domains and draw relevant information from large scale and real time data residing in heterogeneous sources. However, beyond these high level similarities, each use case comes from different domains (e.g. biomedical, medical, or public relations), deals with different type of data (e.g. structured database tables, semi-structured log data, unstructured blogs, forum discussions or tweets) from different data sources (e.g. biomedical analysis tools, image analysis software or social media monitoring tools) and with different work practices (e.g. organisational practices of research teams or market research teams for public opinion monitoring).

Both the similarities and the differences among the use cases bring forth several research challenges in terms of ontology development: (1) **Domain complexity**: Understanding sensemaking in these domains is difficult as it involves heterogeneous sources of knowledge, i.e. expertise from multiple disciplines. (2) **Knowledge scope expansion**: The conceptualisation process is generally dynamic and evolves with the increasing amount of tacit knowledge being made explicit. This means that certain concepts and relationships are unidentified in the beginning. Hence, it is not always possible to build an all-encompassing ontology in the very first instance. (3) **Systematic development**: Traditional knowledge elicitation techniques for conceptualisation that rely on domain experts are not sufficient as conceptualisation might result in ad-hoc modelling. To address these challenges in Dicode and for the development of DON, we apply a novel socio-technical sensemaking modelling approach presented in the next section.

3 The Proposed Socio-technical Approach

Socio-technical principles started in the age of shop floor automation [13]. They have since been applied to the design and implementation of computer-based systems and information technology [14, 15]. Underpinning our proposed socio-technical approach for modelling ontology for sensemaking is the concept of *a priori* modularisation. We have developed *a priori* modularisation methodology [16] that enables dividing the domain ontology into several modules from the outset in order to handle the complexity and dynamicity of ontology modelling in ill-defined domains. This modularisation methodology is devised for a class of problems that involve cognitively-complex processes carried out by humans and require tacit knowledge (e.g. decision making, sensemaking). The understanding of such domains involves inter-disciplinary domain experts who often utilise a theoretical framework to guide the articulation of their understanding. According to this methodology, ontology development begins with some *theoretical framework* and arrives to *case specific*

domain ontologies. We follow the three-layered development of domain ontologies which consist of an upper abstract layer, a middle reusable layers, and a lower case specific layer. Each layer may consist of one or more ontology modules (see Fig. 1).

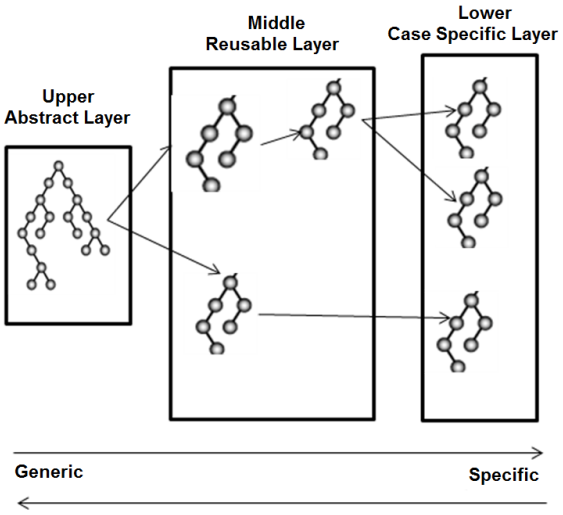

Fig. 1. A Multi-layered Ontology Development with *a Priori* Modularisation

Upper abstract layer: The chosen socio-driven theoretical framework(s) will have the most influence on this layer when the base concepts for the domain are defined following the theoretical framework. Conceptualisation at this level is conceived and developed *independently from its usage context* and avoids defining any concepts that are tied to a particular use case. The sensemaking theoretical frameworks selected for this approach are discussed in section 4.

Middle reusable layers: Middle layers, which *evolve organically through use*, are used to make the connection between the upper ontology layer and the case specific ontology layer. The concepts captured in this layer are likely to be expanded as more tacit knowledge used for interpreting the base concepts is being captured. This layer provides a context-rich bridge between the upper level concepts and the multiple case specific domain ontologies. The middle layers can expand into a number of sub-layers depending on the commonalities among specific cases. The concepts defined in the sub-level should be reusable and remain high level. Only thinking in terms of reusability [17] will keep this layer generic for any sensemaking domain.

Case specific layer: This layer defines the concepts that are specific to each use case (i.e. closer to the content and usage). During this stage, when commonalities in the use cases are discovered, those ontological statements will be moved to the middle layers. This may lead to the expansion of a module or even start a completely new module in the middle layers.

4. Dicode ONtology (DON) for Sensemaking

The following subsections present the main features of the three-layer **Dicode ONtology** (**DON**) developed for sensemaking.

4.1 Upper Abstract Layer

This upper abstract layer ontology covers base concepts that describe sensemaking process for Dicode. We here explain our choice of upper abstract layer sensemaking frameworks.

In Dicode, each use case involves group of professionals collaborating to address complex problems by combining experience and expertise towards a shared understanding. Hence, ***collaborative sensemaking*** is the ultimate target for our work.

We are inspired by the work of Paul and Reddy [18] on collaborative sensemaking. Their framework shows collaborative sensemaking activities are often initially split into tasks/sub-tasks and sub-tasks are performed by different group members (possibly by performing individual sensemaking), depending on their roles and expertise. Roles can be organizational or might be assigned informally. It also defines the collaboration triggers (e.g. ambiguity of information, role-based distribution of information, and lack of expertise) and characteristics of collaborative sensemaking (e.g. prioritizing relevant information, sensemaking trajectories, and activity awareness). The framework also highlights the need to bring together individual sensemaking activities prior to supporting the collaborative sensemaking activities.

To address ***individual sensemaking***, we adopted a notional model developed by Pirolli and Card [8], in which sensemaking process is defined as two interconnected loops: foraging loop and sensemaking loop. The foraging loop involves sensemaking operations such as searching and filtering information, gradually leading to the identification and organization of relevant knowledge. The sensemaking loop is an iterative development of a mental model from the schema/representation that best fits the evidence, which involves searching for support (e.g. using support systems) and using that schema to complete a final task.

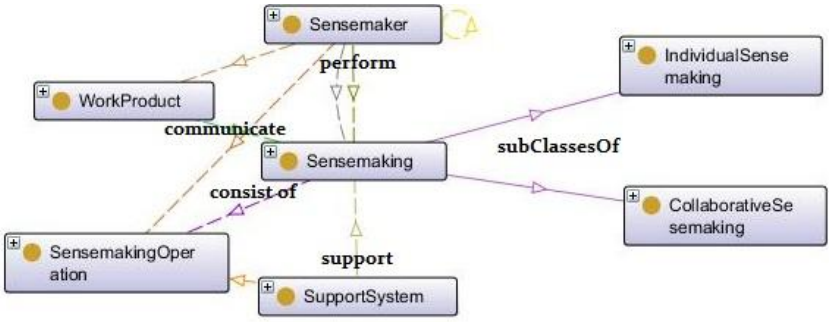

Fig. 2. Abstract Sensemaking Model

Fig. 2 and Table 1 outline the resulting high level base concepts drawn from these two frameworks. The upper abstract layer caters for the main elements of the individual sensemaking such as: actors (e.g. SENSEMAKERS), outcomes (WORK PRODUCT such as documents, diagrams), support services (SUPPORT SYSTEMS, such as data mining, semantic search), and sensemaking operations performed as part of TASKS by human or by machines and main axioms linking them.

Table 1. Abstract Sensemaking Model

Description Logic (DL) syntax
CollaborativeSensemaking ⊑ Sensemaking
IndividualSensemaking ⊑ Sensemaking
Sensemaking ⊑ ∃ consistOf.SensemakingOperation
SensemakingOperation ⊑ ∃ using. InformationSource
SensemakingOperation ⊑ ∃ on. Data
WorkProduct ⊑ ∃ communicate. Sensemaking
Representation ⊑ WorkProduct
Representation ⊑ ∃ represent. Data
Sensemaker ⊑ ∃ perform. Sensemaking
Sensemaker ⊑ ∃ carry out. SensemakingOperation
Sensemaker ⊑ ∃ have. Expertise
Sensemaker ⊑ ∃ create. WorkProduct
Sensemaker ⊑ ∃ utilise. InformationSource
SupportSystem ⊑ ∃ support. Sensemaking
SupportSystem ⊑ ∃ facilitate. SensemakingOperation
CollaborationTrigger ⊑ ∃ trigger. Collaborativ Sensemaking
Sensemaker ⊑ ∃ interactsWith. Sensemaker

The upper abstract layer also contains conceptualisation of collaborative sensemaking process: TRIGGERS triggering COLLABORATIVE SENSEMAKING, SENSEMAKERS interacting with other SENSEMAKERS and playing a ROLE and offering EXPERTISE, division of tasks into SHARED TASK and outcomes (SHARED UNDERSTANDING, SHARED REPRESENTATION).

This upper abstract layer is a starting point for extending into more specific ontologies.

4.2 Middle Reusable Layers

In the middle reusable layers, we defined concepts and respective modules that are used across three use cases. The middle layers in the sensemaking ontology include common concepts that expand the base concepts from the upper layer. The common concepts within all use cases are related to DATA, SENSEMAKING OPERATION, SENSEMAKER and REPRESENTATATION (see Table 2). For example, SENSEMAKING OPERATIONS were expanded with operations relevant to the Dicode use cases (e.g. ABSTRACTING, CLASSIFYING, COMPARING, FILTERING,

SEARCHING, VISUALISING); and DATA were specified (e.g. STRUCTURED DATA, UNSTRUCTURED DATA, QUALITATIVE DATA, QUANTITATIVE DATA).

Table 2. Conceptualising Representation of Middle Reusable Layers

Description Logic (DL) syntax
Representation ⊑ ∃ typeOfRepresentation. RepresentationType
Representation ⊑ ∃ communicate. SharedUnderstanding
SpatialRepresentation : RepresentationType
FacetedRepresentation : RepresentationType
ArgumentationalRepresentation : RepresentationType

4.3 Lower Case Specific Layer

Three case specific ontologies were derived to capture the specificity of sensemaking activities for each Dicode use case. For example, the sensemakers in each user case are represented such as: RADIOLOGIST, RADIOGRAPHER, and CLINICAN in the RA clinical trial use case; CLINICAL RESEARCHER in the CG research use case; and MARKETING RESEARCHER in the PO use case.

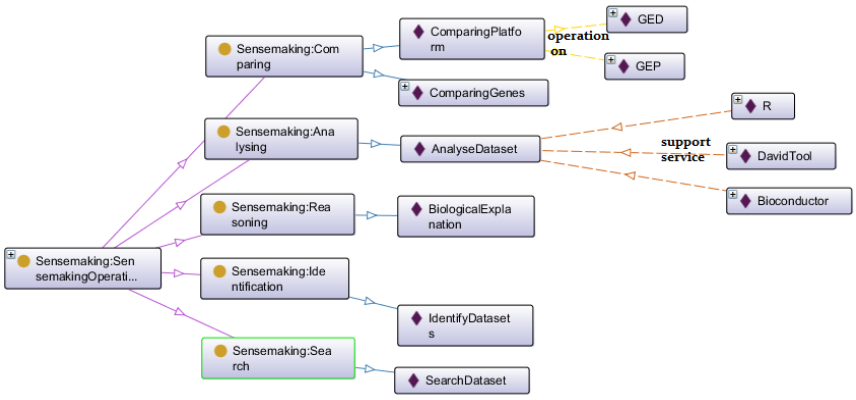

Fig. 3. Sensemaking Operations for Clinico-Genomic (CG) Use Case

Fig. 3 represents the case specific sensemaking operations related to the CG research use case (e.g. COMPARING PLAFTORM, COMPARING GENES, ANALYSE DATASET, BIOLOGICAL EXPLANATION, IDENTIFY DATASET, SEARCH DATASET) including the data such operations are performed on (e.g. GED - Gene Expression Data, GEP – Gene Expression Profile in the case of operation COMPARING PLATFORM) and support systems for such operations (e.g. R, DAVID TOOL and BIOCONDUCTOR for ANALYSING DATASET).

Concepts in the case specific layer were derived from several knowledge sources: interviews with stakeholders in each use case, relevant documentation, and user stories. The knowledge sources were analysed by a representative of domain experts following the guidance from the upper abstract layer and a knowledge glossary was built. The concepts from the glossary were then encoded in an ontology using an

intuitive ontology authoring tool ROO [19] which enables active involvement of domain experts.

Our modelling approach also allowed us to utilise relevant ontologies and datasets from Linked Data Cloud (such as DBpedia[2]) and public ontologies (such as RadLex[3], MeSH[4]) to enhance the coverage of the concepts in DON use case modules. For example, to improve the coverage of the BODY PART concept (from the RA clinical trial use case) we utilised RadLex and MeSH ontologies (see Fig. 4).

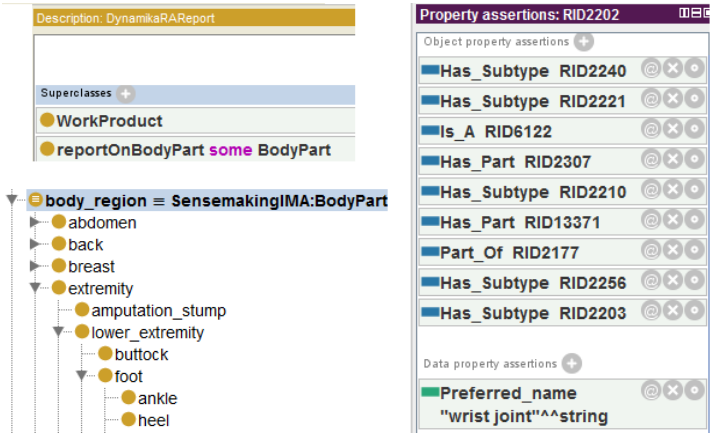

Fig. 4. Utilising external ontologies to specialise Body Part concepts in the RA clinical trial use case (left-top: Original, left-bottom: MeSH ontology, right: RadLex ontology)

5. Utilisation of DON: Augmentor Semantic Services in Dicode & Beyond

DON ontology is being used for semantic augmentation of medical diagnosis reports and user contributions to argumentative interactions. For semantic augmentation we have developed generic services: a) Semantic Annotation service – to tag content semantically, i.e. linking content to named entities and b) Semantic Query service – to search (and to facilitate browsing of) semantically tagged content.

A web based tool ***Augmentor*** is developed to illustrate the utilisation of these semantic services and to understand benefits ontologies can bring. In this section, we outline the implementation details for Augmentor.

[2] http://dbpedia.org/About
[3] http://www.rsna.org/radlex/
[4] http://www.nlm.nih.gov/mesh/

5.1. Architecture and Implementation

Fig. 5 shows the main architectural elements of the currently implemented Augmentor services and their interactions.

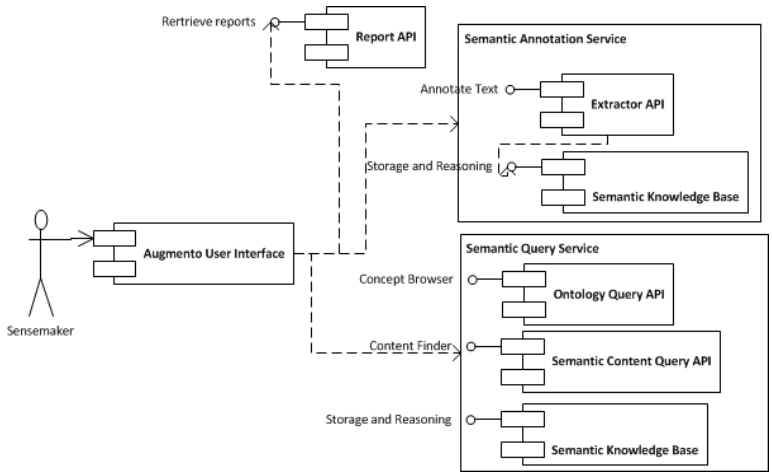

Fig. 5. UML Component Diagram for Augmentor Services

In addition to its front-end user interface, Augmentor consists of semantic annotation and semantic query services – both components are utilising DON ontologies. The interface also consumes an internal report API. The semantic reasoning and storage layer is part of the both services and works as an interface between the underlying semantic processing technologies (ontological knowledge bases, application logic, and text mining systems) and the services.

Through a URL, Augmentor retrieves metadata and selects textual content of the medical diagnosis reports kept in a web server. Semantic annotation service automatically tags content with DON concepts using text mining techniques based on GATE[5]. This service also augments the tags with the concepts from external ontologies. The content is tagged on the fly and stored in a semantic knowledge base driven by high performance OWLIM[6] semantic repository. Browsing and retrieval of heterogeneous content (comments, metadata, and knowledge base) in semantic query service is implemented using two sets of technologies: schema level reasoner API Jena[7] to browse the ontologies and content retrieval service based on OWLIM. The REST based implementation of these components allows utilising these services outside Augmentor user interface (see Fig. 6) by Dicode use case partners or other services in Dicode.

[5] http://gate.ac.uk/
[6] http://www.ontotext.com/owlim
[7] http://jena.sourceforge.net/

In Fig. 6, four parts are being highlighted (A to D) to show the result of semantic augmentation on a medical report. ***A)*** *Comments* - the medical report contains self-reflection note/comments from a sensemaker (radiographer). These notes can be used by other sensemakers to study this sensemaker's sensemaking process while conducting clinical study. ***B)*** *Concepts* - the important concepts that describe the comments are semantically tagged and linked to the knowledge base. Clicking on the hyperlinked concepts takes the sensemakers to other related reports. ***C)*** *Sensemaking Operations* - Augmentor gives indication of the sensemaking operations carried out by this sensemaker, which can be referenced by other sensemakers. ***D)*** *Resources* – Augmentor interface shows the connections of the report to relevant linked datasets.

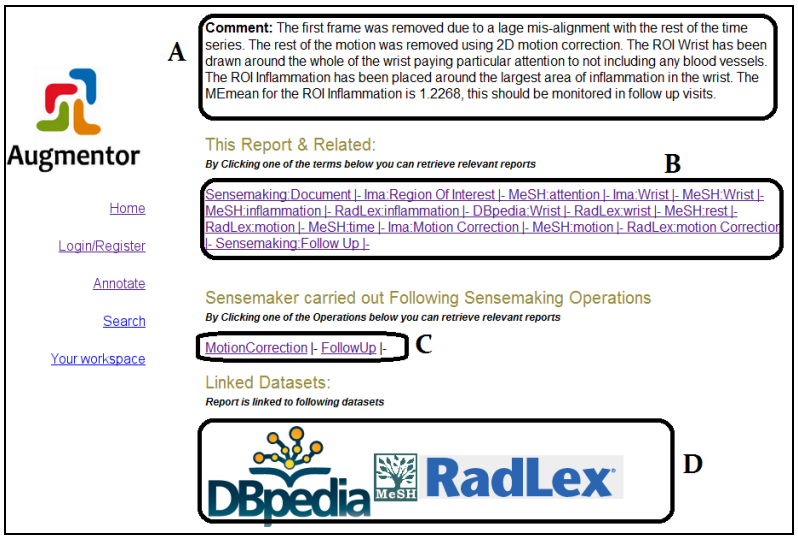

Fig. 6. Interface for Augmentor Services: a result of semantic Augmentation

5.2. Benefits of Modularisation in DON

While utilising DON in Augmentor services for semantic augmentation of content, the modularisation approach allows utilizing only relevant modules from the DON and corresponding knowledge base. It helps to constraint the annotation space for the semantic annotation service to be limited to the specific use case.

We also utilise DON in Augmentor services by providing a structure for browsing content related to sensemaking activities and knowledge bases, which can facilitate sensemakers in Dicode to take informed decisions. The browsing service requires storage and reasoning layer to support required reasoning and search functionalities. We utilise semantic technologies (e.g. semantic repositories, SPARQL[8]) to provide the storage and reasoning for developing such applications. We exploit the modularisation in DON ontologies by loading relevant ontologies and datasets to each

[8] http://www.w3.org/TR/rdf-sparql-query/

use cases into separate SPARQL named graphs on semantic repositories. This allows us querying and reasoning against a subset of ontology and knowledge bases instead of the whole. Hence, we benefit from the scalability offered by the modular design.

Beyond Dicode, with reusability-driven strategy in the modularisation, we have created a set of ontologies that can be utilised and extended seamlessly in other projects and applications that focus on sensemaking activities. DON is distributed as open source[9].

6. Conclusions & Future Work

In this paper, we have described a socio-technical approach for modelling sensemaking which is an example of cognitively complex domains. Underpinning our approach is an *a priori* modularisation methodology that enables the division of domain ontology into several modules from the outset in order to (i) systematically handle the complexity and dynamicity of ontology modelling in such domains; (ii) iteratively incorporate contributions from the social sciences into the ontology.

This approach can be followed for addressing key challenges of ontology engineering in cognitively-complex or ill-defined domains (which are becoming ubiquitous with the information explosion on the Web). In particular, utilising theoretical frameworks can be beneficial for the domain experts to guide the articulation of their understanding. We have demonstrated the application of the socio-technical approach in the context of the Dicode project where a multi-layered ontology (DON) is designed to address requirements from multiple use cases that involve sensemaking. The paper has also illustrated the use of DON in Augmentor to semantically augment and link medical diagnosis reports to assist sensemakers.

The next phase of this work includes: ***a) Further experimentations*** with ***DON*** in web service annotations and discovery. DON could be used as a common vocabulary between services and service developers and for enhancing the functionality of specific Dicode services such as social media mining or community mining. ***b) A user trial*** of DON and Augmentor. In particular, we are interested in the impact that the semantic-driven sensemaking services have in aiding sensemakers. ***c) Improvement to the functionalities*** of DON and Augmentor services. Augmentor will be further developed to cover remaining use cases from the Dicode to support clinical research professionals to make sense of scientific findings in the breast cancer domain and support market research analysts to make sense of the brand's public perceptions on social media.

Acknowledgements

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no ICT 257184 (DICODE project). Thanks go to Ronald Denaux for his reviews of the paper and useful suggestions for improving the paper.

[9] https://sites.google.com/site/ontomatic/don

References

1. Warren, P.: Knowledge Management and the Semantic Web: From Scenario to Technology. IEEE Intelligent Systems, IEEE Intelligent Systems, 21, pp. 53-59 (2006)
2. Stojanovic, N. and Handschuhs, S.: A Framework for Knowledge Management on the Semantic web, In: 11th International World Wide Web Conference, Honolulu, Hawaii, USA (2002)
3. Watanabe, K.: Introduction of Dublin Core metadata. Journal of Information Processing and Management, 43 (2001)
4. Harris, M.A., Clark, J.I., Ireland, A Lomax, J. and Ashburner, J.: The Gene Ontology (GO) project in 2006. Nucleic Acids Research, 34, pp. 322-326 (2006)
5. Rector, A. and Rogers, J.: Ontological Issues in using a Description Logic to Represent Medical Concepts: Experience from GALEN. IMIA WG6 Workshop: Terminology and Natural Language in Medicine. Phoenix Arixona (1999)
6. Russell, D. M., Stefik, M. J., Pirolli, P. and Card, S. K.: The cost structure of sensemaking. In INTERCHI '93 Conference on Human Factors in Computing Systems, Amsterdam (1993)
7. Dervin, B.: Sense-Making Theory and Practice: An overview of user interests in Knowledge seeking and use. Journal of Knowledge Management, 2(2), pp. 36-46 (1998)
8. Pirolli, P. and Card, S.: The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis. Proceedings of International Conference on Intelligence Analysis (2005)
9. Whittaker, S.: Making sense of sensemaking. In T. Erickson and D.W. McDonald (Eds.): HCI remixed: Reflections on works that have influenced the HCI community, pp. 173–178 (2008)
10. Omitola, T., Millard, I., Glaser, H., Gibbins, N. and Shadbolt, N: From Information to Sense-Making: Fetching and Querying Semantic Repositories. In: KES 2010, Part IV, Lecture Notes in Artificial Intelligence, 6279, Springer (2010)
11. Dadzie A-S, Iria, J., Petrelli, D. and Xia L: The xmediabox: Sensemaking through the use of knowledge lenses. In Extended Semantic Web Conference, pp. 811-815 (2009)
12. Ure, J., Proctor, R. Data Integration in eHealth: A Domain/Disease Specific Roadmap, Studies Health Technolgy Information, pp. 144-53 (2007)
13. Trist, E and Bamforth, K.: Some social and psychological consequences of the longwall method of coal getting, in: Human Relations, 4 (1), pp. 3-38 (1951)
14. Clegg, C.W.: Sociotechnical Principles for Systems Design, Applied Ergonomics, 31, pp. 463-477, (2000)
15. Scacchi, W.: Socio-technical design, In: Bainbrigde, W.S. (Ed.), The Encyclopaedia of Human-Computer Interaction, Berkshire Publishing Group (2004)
16. Thakker, D., Dimitrova, V., Lau, L., Denaux, R., Karanasios, S. and Yang-Turner, F: A Priori Ontology Modularisation in Ill-defined Domains. Accepted for the I-Semantics 2011: 7th International Conference on Semantic Systems. Graz, Austria (2011)
17. Simperl, E.: Reusing ontologies on the Semantic Web: A feasibility study. Data Knowledge Engineering. 68(10), pp. 905-925 (2009)
18. Paul, S.A. and Reddy, M.: A Framework for Sensemaking in Collaborative Information Seeking. Proceedings of 2nd International Workshop on Collaborative Information Seeking at CSCW 2010, Savannah, GA (2010)
19. Denaux, R., Dolbear, C. Hart, G., Dimitrova, V. and Cohn, A.G.: Supporting Domain Experts to Construct Conceptual Ontologies: A Holistic Approach. Web Semantics Science Services and Agents on the World Wide Web, 9(2), pp. 113-127 (2011)
20. Pirolli, P. and Russell, D. M.: Introduction to this special issue on sensemaking, Human-Computer Interaction Journal, 26(1-2), pp. 1-8 (2011)

iCAT: A Collaborative Authoring Tool for ICD-11

Tania Tudorache, Csongor Nyulas, Natalya F. Noy,
Timothy Redmond, Mark A. Musen

Stanford Center for Biomedical Informatics Research, Stanford University, US
{tudorache, nyulas, noy, tredmond, musen}@stanford.edu

Abstract. We present iCAT—the Collaborative Authoring Tool for the 11th revision of the International Classification of Diseases (ICD-11). ICD is a fundamental health-care resource developed by the World Health Organization (WHO) and applied in all United Nations countries for a variety of uses. A landmark change in this ICD-11 revision compared to previous ones is the use of OWL as a representation language and of Semantic Web technologies for the collaborative authoring of the ICD content. A community of international medical experts develops ICD-11 in a collaborative setting using the Web-based iCAT platform. Besides its extensive collaboration support, iCAT also fosters the interconnectedness of ICD-11 with other biomedical ontologies and terminologies by providing interlinking and reusing capabilities as a fundamental feature of the tool, while also storing the provenance metadata. The generic and extensible infrastructure as well as its declarative user interface allowed us to easily deploy iCAT in a production setting for the development of ICD-11 and two other WHO classifications. The declarative user interface allowed us to custom tailor the platform for domain experts use. iCAT is in production use since October 2009; it was used to author over 105,000 changes in the ICD-11 ontology, to create more than 40,000 cross-links to other biomedical ontologies, and to produce over 19,000 notes and discussions. The software is open source and a demo version of the platform is available at http://icatdemo.stanford.edu.[1]

1 ICD-11 – Using Semantic Web Technologies

The International Classification of Diseases (ICD) is the standard diagnostic classification developed by the World Health Organization (WHO) to encode information relevant for epidemiology, health management, and clinical use. Over the years, ICD has become an essential resource in all United Nations countries, who are applying ICD for a variety of uses, ranging from compiling basic health statistics, to billing, and to informing policy making.

To keep up to date with the scientific progress, WHO publishes new revisions of ICD every decade. ICD-10 is the latest revision that is currently in use. Our group has been collaborating with WHO since 2007 to support the development of ICD-11.[2] A large community of medical experts around the world are involved in the authoring of ICD-11 in a collaborative Web-based platform, called iCAT [2]. The most important change

[1] For the OCAS challenge we created a user account: ocas and password: ocas.
[2] http://www.who.int/classifications/icd/revision/en/index.html

in ICD-11 compared to previous revisions is the decision of WHO **to adopt a solid formalization of ICD-11** and **to use Semantic Web technologies for its development**. As a result, the underlying formalization of ICD-11 is OWL, and the platform used for the collaborative authoring of ICD-11, iCAT, a customization of a generic Web-based ontology editor, WebProtégé.[3]

2 The ICD-11 Semantic Web Platform—iCAT

With the radical change in the ICD-11 revision with respect to the underlying representation, we needed to develop tools and methods that, on the one hand, are suited for domain experts and on the other hand support the much richer Semantic Web content of ICD-11. Some of the main requirements for the platform are: a) support the **Web-based collaboration** of experts all around the world, b) provide features to **inter-connect and inter-link** ICD-11 with other biomedical ontologies and terminologies, and c) provide a **user interface tailored to the needs of the domain experts** that would hide or sugar-coat complex OWL formalization details.

As an overall strategy, we decided to implement **generic features** that can be used for the development of other Semantic Web applications in the more generic WebProtégé platform, and have only a very small number of plugins that are specific for ICD in iCAT (which is a customization of WebProtégé).

The development of the ICD-11 core ontology and of the Web platform took place in parallel, with the former informing the latter. A WHO assigned committee of ontology experts developed the core ontology in OWL. We defined **forms and templates** based on this core ontology that domain experts use to enter the actual ICD-11 content in iCAT. During the last 4 years, the core ontology has evolved significantly, and we had to make sure that our tools were adaptable enough to support seamlessly such frequent changes. Our solution was to implement a **declarative user interface**[4] as a generic feature in WebProtégé, and iCAT became mainly a specific UI configuration of WebProtégé. This feature allowed us to re-configure iCAT on-the-fly while the Web application is running, and also to define **user-specific views** of the ICD ontology.

Collaboration support: One of the main features of iCAT is its extensive support for collaboration in the development of Semantic Web content, including change tracking, contextualized threaded discussions, watches and notifications, an extensible access policy mechanism, and generation of statistics of the ontology-development process[5] [2]. We reused the functionalities of Collaborative Protégé [3], which are themselves implemented using Semantic Web technologies.

Ontology interconnectedness and reuse support: ICD-11, as many other ontologies, reuses terms from external ontologies and terminologies. We implemented a generic plugin for WebProtégé that enables simple import of terms from external ontologies stored in the BioPortal[6] ontology repository [1]. This plugin allows users to search for terms in BioPortal ontologies, to browse their details, and then to import them into the ontology with a single click. All these operations are supported through

[3] `http://protegewiki.stanford.edu/wiki/WebProtege`
[4] `http://protegewiki.stanford.edu/wiki/WebProtegeLayoutConfig`
[5] `http://protegewiki.stanford.edu/wiki/ChangeAnalysisTab`
[6] `http://bioportal.bioontology.org`

REST calls to BioPortal. Several properties in ICD-11 have values coming from external ontologies, such as *body part*, *morphology*, or *genomic linkages*. Based on the property range definitions in the ICD ontology, we configured the fields in the UI for these properties to search specific ontologies, or their subsets, in BioPortal. Some of the external ontologies and terminologies we have linked to ICD-11 include SNOMED-CT, The Gene Ontology (GO), the Online Mendelian Inheritance in Men (OMIM), the International Classification of Functioning, Disability and Health (ICF) and the International Classification of External Causes of Injuries (ICECI). During the last year, users have created more than 40,000 links between ICD-11 and terms from external biomedical ontologies. More than 14,000 of these links are for *body part* associated to a disease and take values from the SNOMED CT Anatomy branch. Another use case for interlinking ICD-11 with its previous revision, ICD-10 (also stored in BioPortal) is to keep track of the ICD-10 to ICD-11 mappings that are essential for transitioning existing medical software to the new ICD-11 coding system.

Infrastructure reuse: We implemented WebProtégé as a pluggable and extensible architecture that can be customized to the needs of a particular project. As a proof, iCAT is one particular customization of WebProtégé. We reused this generic infrastructure to deploy similar platforms to support the production development of two other WHO classifications: the International Classification of Traditional Medicine (ICTM) [7] and the International Classification of Patient Safety (ICPS). [8] These two platforms required only a new user interface configuration file and no code changes. Other customizations of WebProtégé are available on the WebProtégé demo server.[9]

3 Conclusions

We presented iCAT, a customization of the WebProtégé platform for the development of ICD-11. We discussed the advantages of using Semantic Web technologies for ICD-11 elsewhere [2]. iCAT has been in production use since 2009 and other customization of WebProtégé for real-world Semantic Web applications are available. All software is open source, pluggable, reusable and under active development.

Acknowledgments

We thank our WHO collaborators for developing the project requirements and for the fruitful collaboration. The work presented in this paper is supported by the NIGMS Grant 1R01GM086587.

References

1. N. Noy, T. Tudorache, C. Nyulas, and M. Musen. The Ontology Life Cycle: Integrated Tools for Editing, Publishing, Peer review, and Evolution of ontologies. In *AMIA Annual Symposium Proceedings*, volume 2010, page 552. American Medical Informatics Association, 2010.
2. T. Tudorache, S. Falconer, C. Nyulas, N. Noy, and M. Musen. Will Semantic Web Technologies Work for the Development of ICD-11? In *The 9th Intl. Semantic Web Conference (ISWC 2010)*, pages 257–272. Springer, 2010.
3. T. Tudorache, N. F. Noy, and M. A. Musen. Supporting Collaborative Ontology Development in Protege. In *Seventh International Semantic Web Conference, ISWC 2008*, Karlsruhe, Germany, 2008.

[7] `http://icatdemo.stanford.edu/ictm/`
[8] `http://icat-ps.stanford.edu`
[9] `http://webprotege.stanford.edu`