Automated UML-Based Ontology Generation in OSLO^2

Dieter De Paepe[1], Geert Thijs[2], Raf Buyle[1],
Ruben Verborgh[1], and Erik Mannens[1]

[1] IDLab, Department of Electronics and Information Systems, Ghent University – imec
Sint-Pietersnieuwstraat 41, B-9000 Ghent, Belgium
{firstname.lastname}@ugent.be
[2] Flanders Information
Koningin Maria Hendrikaplein 70, B-9000 Ghent, Belgium
geert.thijs@kb.vlaanderen.be

Abstract. In 2015, Flanders Information started the OSLO^2 project, aimed at easing the exchange of data and increasing the interoperability of Belgian government services. RDF ontologies were developed to break apart the government data silos and stimulate data reuse. However, ontology design still encounters a number of difficulties. Since domain experts are generally unfamiliar with RDF, a design process is needed that allows these experts to efficiently contribute to intermediate ontology prototypes. We designed the OSLO^2 ontologies using UML, a modeling language well known within the government, as a single source specification. From this source, the ontology and other relevant documents are generated. This paper describes the conversion tooling and the pragmatic approaches that were taken into account in its design. While this tooling is somewhat focused on the design principles used in the OSLO^2 project, it can serve as the basis for a generic conversion tool. All source code and documentation are available online.

Keywords: Linked Data, UML, ontology design, RDF generation, OSLO2

1 Introduction

In 2015, the Flemish government started a project to stimulate data reuse between Belgian governments and improving semantic interoperability between government services. They initiated the OSLO^2 project as a continuation of OSLO [2], the aim of OSLO^2 was to define multiple ontologies to model 4 core government domains. The project was led by the Flanders Information agency and involved local, regional, federal, European and private stakeholders in the process through working groups.

Governments are slowly finding their way to the Semantic Web. Working with Semantic Web still requires a lot of technical knowledge not familiar to domain experts. Yet, the input of these domain experts is essential to model an ontology in line with business requirements. As such it is very important

to either train domain experts or to use more familiar techniques for modeling. This led us to the decision to use UML, a well known formal modeling language. The UML model was familiar enough to domain experts to understand it and provide feedback, while also serving as a source from which both the ontology and corresponding documentation could be generated.

In this demo, we will demonstrate the tool developed for and used in the OSLO[2] project. This tool is capable of transforming a UML diagram, intended for communication with stakeholders and domain experts rather than being modeled specifically for RDF generation, into an ontology. The tool is currently a command line tool with a focus on the design choices made by Information Flanders. Nevertheless, it displays great potential and could be extended into a fully generic tool in future work. All source code is available at `https://github.com/Informatievlaanderen/OSLO-EA-to-RDF`.

2 Related Work

Specialized ontology design tools such as Protégé [5] or TopBraid Composer are well known in the Semantic Web world. Because of their ontology-centered design method, these tools are powerful in the hands of experienced users but more obscure for users not familiar with ontologies.

UML[3] is a modeling language standardized by the Object Management Group (OMG). It originally focused on object oriented software engineering, but grew to cover more uses later on such as interaction or object diagrams. UML can be serialized to a machine readable format using the XML Metadata Interchange (XMI) format, also a standard created by the OMG.

UML has been already been investigated as a tool for assisting ontology development and has several advantages, mainly related to its adoption [4]. Cranefield et al. describe the use of XSLT transformations to transform UML represented in XMI into RDF-XML and supporting Java classes [3]. Stuckenschmidt et al. describe how UML could be used as a way of visualizing RDF as well as serving as a basis for generating RDF, again using an XSLT transformation [6]. They also describe the mismatch between both worlds, most notably the fact that properties are first class citizens in RDF but not in UML. Lastly, ISO 19150-2 describes how UML from geographical standards can be converted into OWL ontologies [1].

We see two shortcomings in the approaches described. Firstly, none of them demonstrate how to integrate existing ontologies in the design. Secondly, they rely on modeling the UML diagram very close to the RDF model, which will make the intended structure unfamiliar to domain experts.

[3] http://www.omg.org/spec/UML/

3 RDF Transformation

3.1 Source Data Model

Because of its wide usability and simple representation, Flanders Information uses UML to model their data models. For this, they use Enterprise Architect (EA)[4], a commercial tool with extended UML features. EA uses an internal database to store the models and has the capability to export XMI.

Despite XMI being a common exchange format for UML, subtle differences cause information loss when importing this data into other tools, such as the freely available Visual Paradigm software[5]. Further difficulties arise when using XMI, as the XSLT transformations are very sensitive to the exact format of the source [6]. Lastly, the UML used in previously mentioned papers was mostly designed with an RDF model in mind, rather than following a data-modeling centric methodology. These reasons caused us to focus on the EA data model rather than the XMI format.

EA provides a Java API, giving access to the EA object model[6]. We found this API to be lacking in both usability and capabilities. Performance-wise, a noticeable load time is needed for the library and queries appear to become slower over time, possibly caused by the ActiveX COM implementation. Furthermore, the EA object model is not fully available through the API and no developer-friendly links are available between core classes. Instead, we created our own API for accessing the EA object model by directly querying the internal database.

3.2 Transforming UML to RDF

ISO 19150-2, Cranefield [3] and Stuckenschmidt [6] all describe similar ways of transforming UML into an ontology by mapping UML classes to RDF classes and UML attributes and associations to RDF properties. However, none provide any guidance for integrating existing terms into the ontology, a vital concept in the Linked Data world. Also, all seem to assume that the UML diagram will be created with the RDF model in mind: any UML attribute or association identifies a unique property, domain and range are determined by the related UML class, etc. While these assumptions are valid when strictly considering the modeling, they obstruct a workflow centered around expressive models, as demonstrated in Figure 1.

In order to support this more pragmatic way of working, we extended the conversion rules from existing literature with extra customization options. Specifically, we allow the addition or customization of information through the use of tags that are added to the UML elements. These tags are stored as meta data in the EA data model. For example, it is possible to add multiple labels to each

[4] http://www.sparxsystems.com/products/ea/
[5] https://www.visual-paradigm.com
[6] http://www.sparxsystems.fr/resources/user-guides/automation/enterprise-architect-object-model.pdf

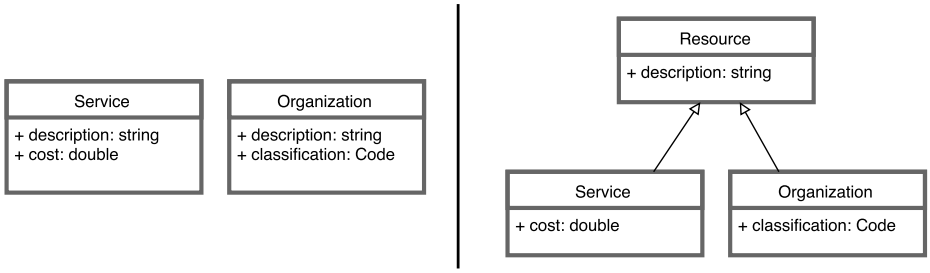

Fig. 1. Two UML versions of the same model. On the left as modeled in typical (non-RDF) contexts. On the right as modeled according to the RDF model.

element, to use a name for the RDF term that differs from the UML name or to specify the parent property of an association or attribute.

Through a configuration file, users can specify which tags map to which RDF terms, allowing them to customize the tool to their design needs instead of the other way around. The tool provides warnings for missing information or possible errors such as missing labels or multiple elements being mapped to the same URI. Nevertheless, any warnings can be ignored at the discretion of the user, giving them full control over the transformation.

A second addition to the tooling comprises the support for existing terms. In fact, because we were modeling 4 different domains that were reusing terms among themselves, we identified 3 types of terms being modeled:

In-scope terms Terms defined by ourselves and contained within the package (ontology) being converted;

Out-of-scope terms Terms defined by ourselves but contained outside the package being converted;

External terms Terms used in the model but defined in external ontologies.

Each of these types has a different presence in the generated ontology: in-scope terms require all information to be included, out-of-scope terms do not require any information to be included and external terms may need to include some information such as additional translations. Again, this behavior can be customized using the configuration file.

Lastly, we allow the user to specify an RDF file containing *user terms*, which are simply added to the resulting ontology. This can be used to add additional information about the ontology itself (such as authors, revision date, changelist...) since this information varies a lot between ontologies and may contain a deeper linking structure. A complete workflow of the tool is shown in Figure 2.

4 Demonstration

We will demonstrate our tool by converting different UML diagrams of varying complexity to RDF ontologies. We will focus on the integration with existing

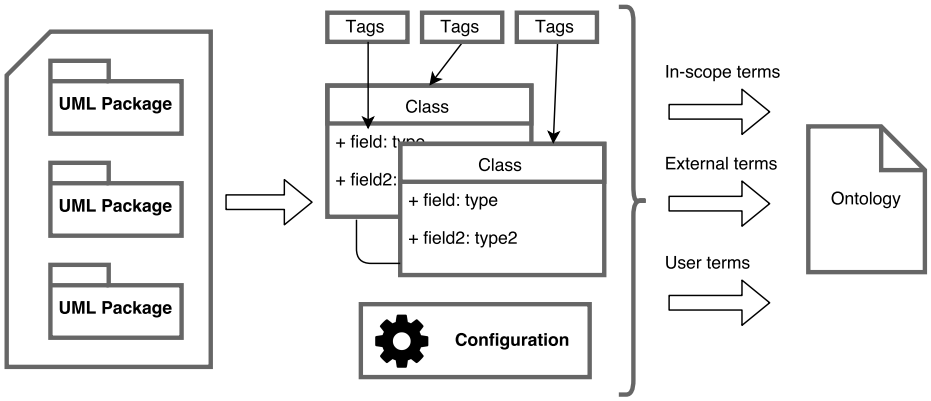

Fig. 2. Workflow of the tool. Starting from a project containing multiple, possibly interlinking packages, a single package is converted. RDF is generated based on the UML information, user specified tags present and the configuration. Depending on the type of the term, different statements are added to the ontology. Out-of-scope terms are not exported.

ontologies and UML intended for communication with stakeholders. An example transformation is available online at `https://github.com/Informatievlaanderen/OSLO-EA-to-RDF/blob/v1.0/Example.md`.

References

1. Geographic information – ontology – part 2: Rules for developing ontologies in the web ontology language (owl), `https://www.iso.org/standard/57466.html`
2. Buyle, R., De Vocht, L., Van Compernolle, M., De Paepe, D., Verborgh, R., Vanlishout, Z., De Vidts, B., Mechant, P., Mannens, E.: Oslo: Open standards for linked organizations. In: Proceedings of the International Conference on Electronic Governance and Open Society: Challenges in Eurasia. pp. 126–134. EGOSE '16, ACM, New York, NY, USA (2016), `http://doi.acm.org/10.1145/3014087.3014096`
3. Cranefield, S.: Uml and the semantic web. In: Proceedings of the First International Conference on Semantic Web Working. pp. 113–130. CEUR-WS. org (2001)
4. Kogut, P., Cranefield, S., Hart, L., Dutra, M., Baclawski, K., Kokar, M., Smith, J.: Uml for ontology development. The Knowledge Engineering Review 17(01), 61–64 (2002)
5. Noy, N.F., Sintek, M., Decker, S., Crubézy, M., Fergerson, R.W., Musen, M.A.: Creating semantic web contents with protege-2000. IEEE intelligent systems 16(2), 60–71 (2001)
6. Stuckenschmidt, K.F.M.S.H.: Uml for the semantic web: Transformation-based approaches. Knowledge Transformation for the Semantic Web 95, 92 (2003)