NeOn: Lifecycle Support for Networked Ontologies

Integrated Project (IST-2005-027595)

Priority: IST-2004-2.4.7 — "Semantic-based knowledge and content systems"

D4.2.4 Ontology Customization Plugin

Deliverable Co-ordinator: **Noam Bercovici**

Deliverable Co-ordinating Institution: **University Koblenz-Landau (UKO-LD)**

This is the forth documents about the ontology customization work in the NeOn project. The main purpose of this deliverable is to present the ontology customization plugin together with the use of it in the different NeOn use case

Document Identifier:	NEON/2009/D4.2.4/v1.0	Date due:	November 30, 2009
Class Deliverable:	NEON EU-IST-2005-027595	Submission date:	November 30, 2009
Project start date	March 1, 2006	Version:	v1.0
Project duration:	4 years	State:	Final
		Distribution:	Public

NeOn Consortium

This document is part of the NeOn research project funded by the IST Programme of the Commission of the European Communities by the grant number IST-2005-027595. The following partners are involved in the project:

Open University (OU) – Coordinator Knowledge Media Institute – KMi Berrill Building, Walton Hall Milton Keynes, MK7 6AA United Kingdom Contact person: Martin Dzbor, Enrico Motta E-mail address: {m.dzbor, e.motta}@open.ac.uk	**Universität Karlsruhe – TH (UKARL)** Institut für Angewandte Informatik und Formale Beschreibungsverfahren – AIFB Englerstrasse 11 D-76128 Karlsruhe, Germany Contact person: Peter Haase E-mail address: pha@aifb.uni-karlsruhe.de
Universidad Politécnica de Madrid (UPM) Campus de Montegancedo 28660 Boadilla del Monte Spain Contact person: Asunción Gómez Pérez E-mail address: asun@fi.ump.es	**Software AG (SAG)** Uhlandstrasse 12 64297 Darmstadt Germany Contact person: Walter Waterfeld E-mail address: walter.waterfeld@softwareag.com
Intelligent Software Components S.A. (ISOCO) Calle de Pedro de Valdivia 10 28006 Madrid Spain Contact person: Jesús Contreras E-mail address: jcontreras@isoco.com	**Institut 'Jožef Stefan' (JSI)** Jamova 39 SL–1000 Ljubljana Slovenia Contact person: Marko Grobelnik E-mail address: marko.grobelnik@ijs.si
Institut National de Recherche en Informatique et en Automatique (INRIA) ZIRST – 665 avenue de l'Europe Montbonnot Saint Martin 38334 Saint-Ismier, France Contact person: Jérôme Euzenat E-mail address: jerome.euzenat@inrialpes.fr	**University of Sheffield (USFD)** Dept. of Computer Science Regent Court 211 Portobello street S14DP Sheffield, United Kingdom Contact person: Hamish Cunningham E-mail address: hamish@dcs.shef.ac.uk
Universität Kolenz-Landau (UKO-LD) Universitätsstrasse 1 56070 Koblenz Germany Contact person: Steffen Staab E-mail address: staab@uni-koblenz.de	**Consiglio Nazionale delle Ricerche (CNR)** Institute of cognitive sciences and technologies Via S. Marino della Battaglia 44 – 00185 Roma-Lazio Italy Contact person: Aldo Gangemi E-mail address: aldo.gangemi@istc.cnr.it
Ontoprise GmbH. (ONTO) Amalienbadstr. 36 (Raumfabrik 29) 76227 Karlsruhe Germany Contact person: Jürgen Angele E-mail address: angele@ontoprise.de	**Food and Agriculture Organization of the United Nations (FAO)** Viale delle Terme di Caracalla 00100 Rome Italy Contact person: Marta Iglesias E-mail address: marta.iglesias@fao.org
Atos Origin S.A. (ATOS) Calle de Albarracín, 25 28037 Madrid Spain Contact person: Tomás Pariente Lobo E-mail address: tomas.parientelobo@atosorigin.com	**Laboratorios KIN, S.A. (KIN)** C/Ciudad de Granada, 123 08018 Barcelona Spain Contact person: Antonio López E-mail address: alopez@kin.es

Work package participants

The following partners have taken an active part in the work leading to the elaboration of this document, even if they might not have directly contributed to the writing of this document or its parts:

- FAO
- Atos Origin

Change Log

Version	Date	Amended by	Changes
0.1	01-09-2009	Noam Bercovici	Initial document
0.2	18-09-2009	Noam Bercovici	Addition of the use case
0.3	22-09-2009	Noam Bercovici	First draft of the plugin description
0.4	09-10-2009	Noam Bercovici	Refinement of the plugin description
0.5	25-11-2009	Noam Bercovici	Second version of the use case
0.6	08-01-2010	Noam Bercovici	Draft send to QA
1.0	22-01-2010	Noam Bercovici	Answer the QA comments

Executive Summary

In this deliverable we focus on the plugin itself, its functionalities and its uses. An exhaustive description of the ontology customization process is presented in the previous deliverable ([DDM+07], [BDS+08], [BS09]) This deliverable accompanies two software outputs in the form of a plugin for the NeOn Toolkit:

(i) SAIQL plugin

(ii) Customization plugin.

Contents

List of Figures

Chapter 1

Introduction

This deliverable is the last part of the work on the ontology customization task. In [DDM+07] we described the state of the art for customization techniques and we presented several use cases where they could be applied. This state of the art highlights two main ways to customize ontologies at the graphical level and draw the attention of the user to a specific part of the ontology, this approach is preferred by [KS03] in the fish eye projection. Another means of customization is using algebra operators to change the structure of the ontology as [Wie94]. Afterward, in [BDS+08] we proposed a method in between those two, using a query engine to evaluate customized views similar to the idea developed by J.F. Brinkley in [LTD06] and refined later in [LTDF07]. The previous deliverable [BS09] presents a first implementation of the customization tool; now this deliverable shows the final tools for ontology customization. The first part will introduce the use case where ontology customization plays a role, while the second part will show how to use the plugin in the case study situations.

1.1 The place of customization in WP4

We also mentioned in the previous works that a good visualization of ontologies is an important aspect of ontology construction tools. From the user studies made in D4.1.1 [DMG+06], visualization supported by the existing tools is rather poor, being too general and providing limited support for problem-specific needs. Concatenating visualization techniques with ontology customization operators (e.g., the one for pruning, described in the previous deliverable) may help to show only the relevant, more focused parts of ontologies, rather than showing the entire graphs with potentially thousands of nodes. Thus, a good level of detail in visualizing can be offset by showing only a customized ontology view - an approach that clearly complements the techniques for context-sensitive visualization that strive to show large and networked ontologies by abstracting the level of visualized details.

Access control presented in [DKG+07] and [Dzb09] benefits clearly from the customization technique for creating a customized view of an ontology. Those views represent the amount of information that the user is allow to access. In order to specify the view definition the administrator can use the customization tools presented in this deliverable with specific templates. The solution proposes to pre-compute a set of views, each of those views corresponding to a possible authorized access to the original ontology.

1.2 Relationship of this work to the other NeOn work packages

Obviously, since customization relates to such aspects as context or modularization, this work is related to other work packages in the NeOn project. In particular, we highlight a few points of overlap and where potential interactions can be found with the other work packages. First, one particular form of creating networked ontologies investigated in WP1 is their modularization. An ontology module is seen as a tuple of imported ontologies, certain import and export interfaces, and a set of mappings. Formal models of

modularization techniques have been presented in [dHR+07].

Relationship to WP2 is less obvious, however, in the process of ontology customization we use templates, which can be filled in with a concrete input from the user or which can be bootstrapped and executed on a user profile. In other words, these defined queries may be seen as a specific type of "ontology pattern ", albeit using those templates less as "ontology design patterns ", and more in the role of a driver for simplifying views on an ontology. In WP2 the research focus on a more generic set of patterns; mainly on those used during ontology design phase [Gan05].

With respect to research done in WP3, the relationship is somewhat clearer. Customizing an ontology by adjusting what is shown to a particular user (e.g., by pruning unnecessary nodes and branches) can, pragmatically, be seen as bringing and ontology into the context of a particular user. Even more importantly, the selection and use of particular templates by a given group of users may be seen as a valid contextual modifier for segmenting users into groups and for automatically constructing user profiles. The added value of our work is that SAIQL queries represent user preferences in terms of procedural knowledge, i.e., how they prefer to interact with a given ontology; whereas the majority of other user profiling techniques is focusing on what the user interacts with.

Chapter 2

Use Case for Ontology Customization

2.1 WP7 use case - view by language

Multilingual ontologies may be available in several languages e.g., fisheries ontologies described in[CG07] contain three or four languages. AGROVOC is annotated with sixteen languages, although for common applications not all of them are used. It is then useful to select only the languages used for the specific application at hand.

Most of the reference data is available in more than one language, usually English, French and Spanish. Often, two or three names are available in each language, such as a “short name,” a “long name” and an “official name.” In all cases, names are established on the basis of international agreements which result in a 1-to-1 correspondence between languages. They rendered each name (either short, long or official) by means of data-type properties, endowed with an RDF label xml:lang corresponding to the language at hand. The two-digit ISO codes of the language are also kept in the name of properties (as in “hasNameShortFR”) in order to ease the visualization of properties by human editors and to stay close to existing practices adopted in FAO.

This multiplication of different languages and different types of names has an undesirable effect. It makes the ontology size bigger which can slow down the application for no reason (cf table2.1). In this table, we see the advantage of having a view on the ontology which considers only the language the user need.

Languages Selected	Size
EN[1]	39 MB
EN + TH[2]	53 MB
EN + TH + FR[3]	65 MB
EN + TH + FR + ES[4]	79 MB
EN + TH + FR + ES + AR[5]	92 MB
EN + TH + FR + ES + AR + ZH[6]	110 MB
EN + TH + FR + ES + AR + ZH + JA[7]	130 MB

Table 2.1: size of agrovoc following different languages, ontology from Mars 2009

This use case is strongly dependent on the user who is choosing the type of name and the languages the ontology is shown in, reflecting the preferences of the user. It is for this reason that this use case can only be handled by the customization plug-in.

2.2 WP8 Nomenclature use case - view by domain

The main requirements of the Semantic Nomenclature case study are described in [GPDM+06] and [GPDLH07]. Although there is a lack of an international standard to describe drugs, in recent years we

have been witnesses of the increased attention of the health domain towards what is called Semantic Interoperability in eHealth. One of the key determinants for the success of the Semantic Interoperability is to have a common description of drugs. To address this, our partner, Atos, proposes the Nomenclature Ontology Network. This network is composed of general, domain and application ontologies. The Nomenclature Ontology Network is shown in the figure2.1 where the reader can see the different ontologies used and their different types. We see from this figure that Galen[RRP96] is used as a domain ontology.

Let's consider the case of a laboratory which produces cardiac medication. Galen provides more information than is needed by this user; only the axioms related to "Cardiac Pathology" would be interesting for him. We will see in the next chapter how to handle this case.

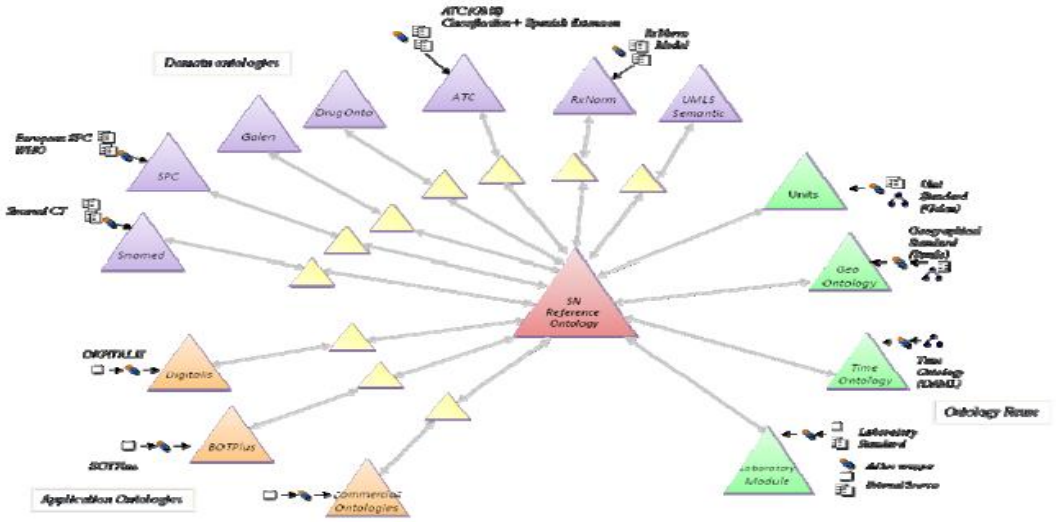

Figure 2.1: Nomenclature Ontology Network

Chapter 3

Ontology Customization Plugin

The previous deliverable [BS09] described the process and the architecture of the ontology customization plugin[1]. We explained how to build a customized view from an ontology and a view definition. The major deficiency in the previous version of the plugin was identified, which is the process of acquiring the view definition. It was noted that this step of the customization process could make the user uncomfortable with the tools. In the new version of the plugin we therefore focused on the graphical user interface and the relation to the user. This chapter will first present an overview of what the plugin does. Then we will show how the plugin can be used in the different use cases in FAO and later in the nomenclature case study. We will close this chapter by describing the user documentation of this plugin.

3.1 Functional Description

The main purpose of the ontology customization plugin is to propose a simple way to define a customized ontology view corresponding to the needs of an ontology engineer using the NeOn Toolkit. The plugin perspective integrates the ontology relation browser from Isoco by default. While the user is browsing the ontology he can select classes or properties which might be relevant for him. The user can easily define a customized view of the ontology from those classes or properties by using the template mechanism presented in the deliverable [BS09].
Altogether, the plugin supports the following functionalities:

1. A filtering mechanism using the annotation properties or/and the data-type properties;
2. A customized ontology view builder which proposes a friendly user interface for helping the user to compose the view definition;
3. A query engine for the query language SAIQL which gives the possibility to execute directly SAIQL query via the “expert section”.

3.2 Use of the customization plugin in the FAO use case study to evaluate a customized view of a multilingual ontology

Let’s consider Agrovoc, an ontology that is described in 16 different languages which are rarely used at the same time by the same user. Figure 3.1 shows how the user can use the ontology customization plugin to filter the agrovoc[2] ontology via the language used in the label. The type of ontology customization that we call filtering enables the user to keep the same ontology structure as the original one. This filter can be

[1] http://neon-toolkit.org/wiki/Ontology_Custamization
[2] Please note when we mention Agrovoc in this chapter we refer to a module of 50Mb used in our test.

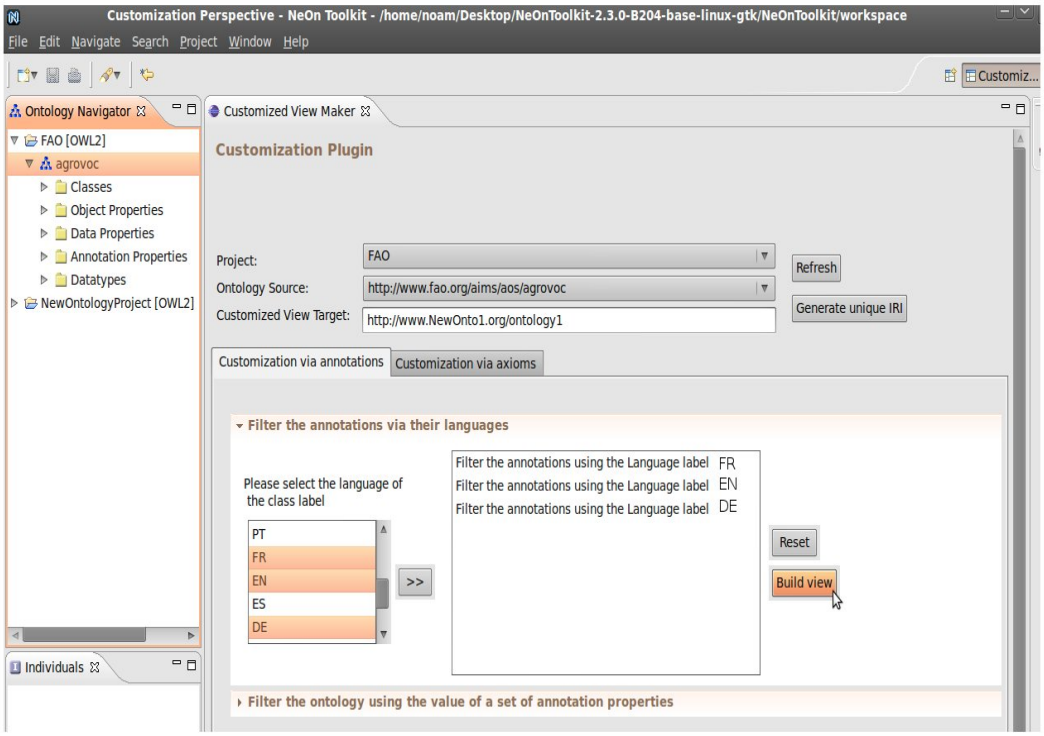

Figure 3.1: Ontology filtering operator

applied to the data-properties and the annotation properties. This filtering operation can be combined with some other functionalities of the plugin which are shown later in this deliverable.

3.3 Use of the customization plugin in the Nomenclature use case study

The main goal of this section is to show how to create an ontology view which is focusing on a specific region of the ontology. To achieve this, we will use the example of the Galen ontology used in the nomenclature use case. We will explain how to make a view focus on the “Cardiac Pathology”. In figure 3.2 you can see the structure of the Galen ontology and the table 3.1 brings some statistics concerning Galen before the customization process.

Languages Selected	Size
classes	2,000
class expressions	19,000

Table 3.1: some statistics about the Galen ontology used in the test

Later in this chapter, in section 3.4.2 we will explain the process of defining this customized view. Let us have a look at the view from the statistic description content shown in table3.2. Here, the view is lighter than the original ontology and therefore easier to work with than the original ontology. In figure 3.3 we can see that the path to go from “thing” to “Cardiac Pathology” is shorter in the view in comparison to the original ontology.

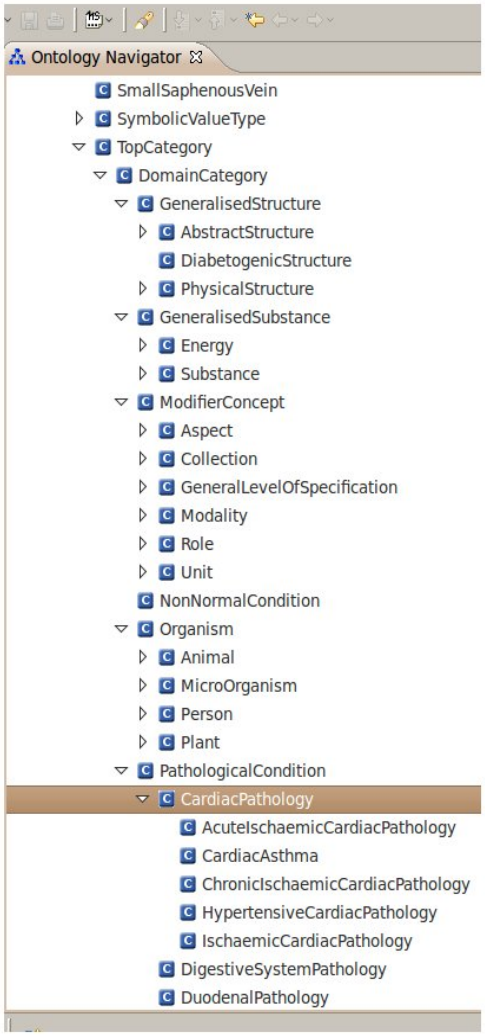

Figure 3.2: Taxonomy of Galen Ontology

Type	Number
classes	57
class expressions	1,050

Table 3.2: Some statistics about the view focusing on “Cardiac Pathology”

3.4 User Documentation

3.4.1 How to install it?

If you want to install the Ontology Customization plugin:

1. Within the NeOn-Toolkit open the Help-menu and select the entry Software Updates->Find and Install;
2. Select Search for new features to install in the appearing window and click Next;
3. Then select the NeOn-Toolkit Update Site and click Finish;
4. In the appearing window you can find the Ontology Customization-plugin under the modularization/customization category. Select the Customization feature. Click Next and follow the installation instructions shown by the toolkit.

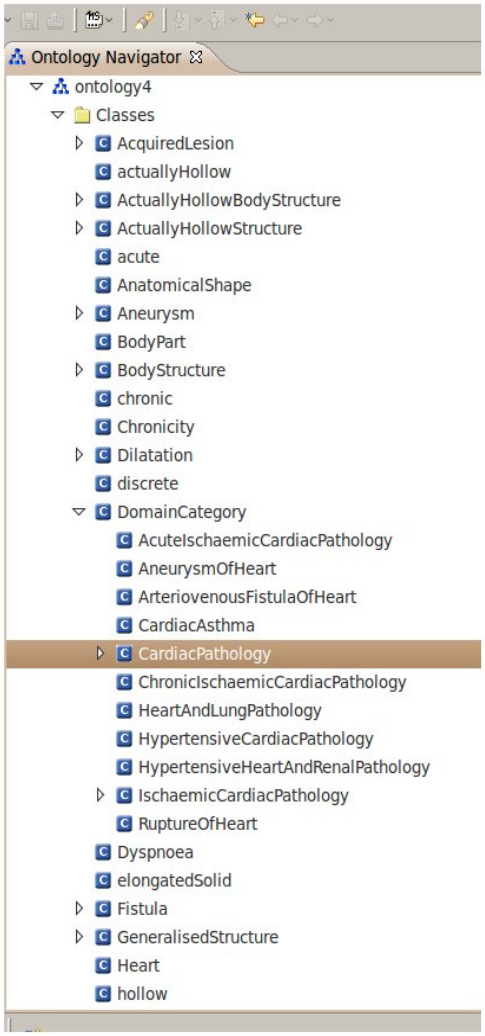

Figure 3.3: Taxonomy of the Galen view with “Cardiac Pathology” as the main class

If you want to update from an older version of the Ontology Customization plugin:

1. Within the NeOn-Toolkit open the Help-menu and select the entry Software Updates->Find and Install;
2. Select Search to update the currently installed features in the appearing window and click Finish;
3. If updates are available, select the parts you want to update and click Next and follow the update instructions shown by the toolkit.

For manual installation, download the latest version of the plugin with all the dependencies (cf. the list below) from the homepage. After downloading, you have to copy the java archive (.jar) in the folder "Plugins" located in the NeOn Toolkit’s directory. After manually installing the plugin you have to restart the NeOn-Toolkit.

Dependencies

The customization plugin depends on another plugin

The query engine SAIQL is used to evaluate the view : org.neontoolkit.saiql version 1.0

Ontology Visualization from isoco : org.neontoolkit.ontologyVisualization version 1.2

3.4.2 How to use the customization via a set of axioms

This function is helping the user to define customized view. First of all the user has to define a project and an ontology that he wants to customize as shown in figure 3.4 in the top of the plugin view. Afterwards, the user should specify a namespace for the customized view, either by choosing a namespace or by generating a new unique namespace by simply clicking on the button next to the field provided. Proceeding, there are two tabulations in the view, from which the user has to select the second one called “customization via axioms”. The user should find himself in a similar situation as shown in figure 3.4.

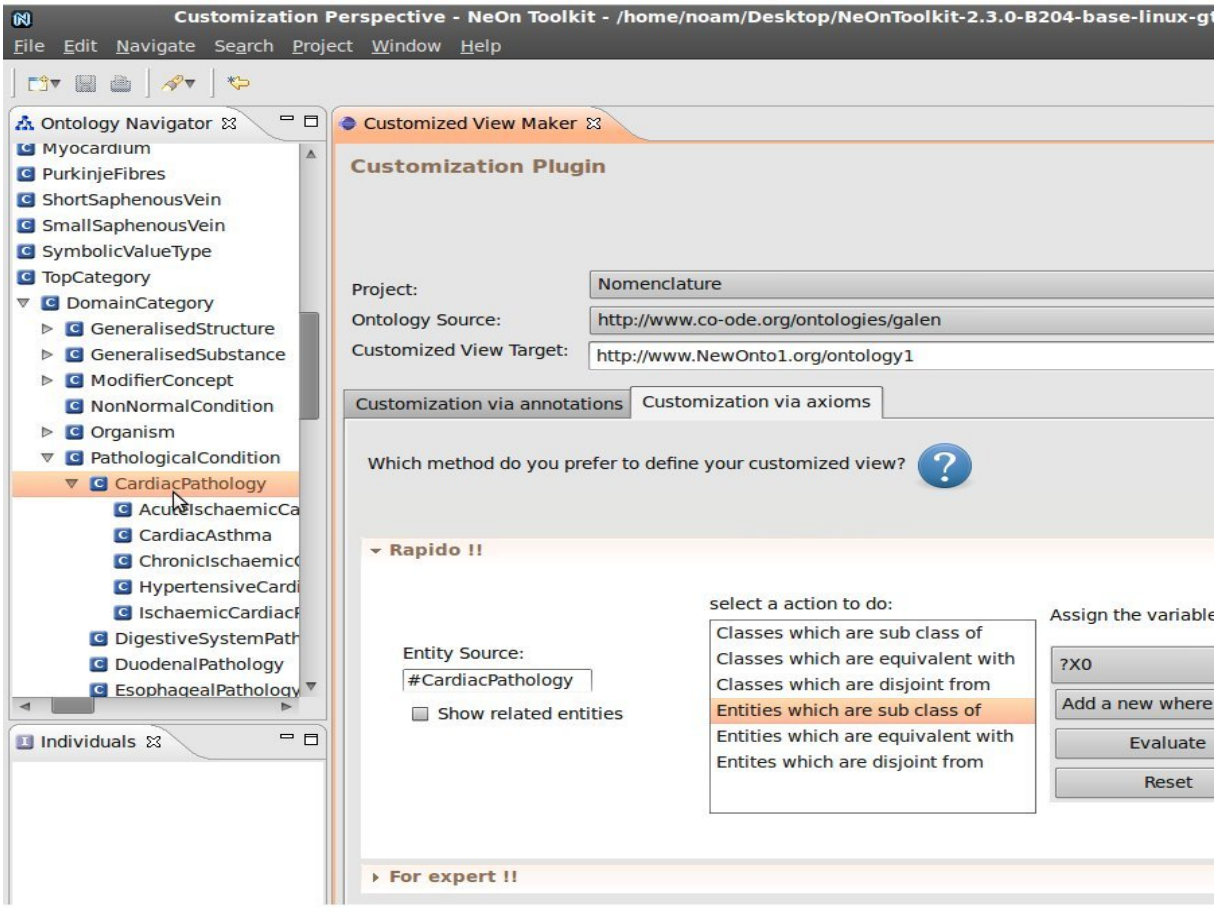

Figure 3.4: How to select a class to start the customization process

When the user choosed to expand the “Rapido” view, he will be able to define the view. Now the user can go through the classes, the properties or the individuals while browsing the ontology, either using the “ontology relation browser” or the navigator provided by the NeOn Toolkit. Doing so, the selected axiom will change. In the case of figure 3.4 we want to customize the Galen ontology using the class “Cardiac Pathology”. When the right class is selected, the user can select different ways of extracting the view. As a next step, the user should select a variable. Note that the variable will be generated following the type of the entity selected. A new variable will be added for each axiom. Nevertheless the user has the possibility to reuse, by selecting, the same variable in different axioms to restrict or to enrich the output view. If the user estimates that the current definition of the view is complete, the view can be evaluated by pressing the corresponding button or, in the event that the user wants to add something to the view definition, a simple click on the button “add axiom to the definition” restarts the previous step to add a new axiom.

In the “expert section” the user can see the generated query using the SAIQL[KSS06] language for OWL-DI ontology as shown in figure 3.5. With the help of this section the user can easily check whether the query is what he was expecting. Each change which occurs in one of those sections will automatically be forwarded

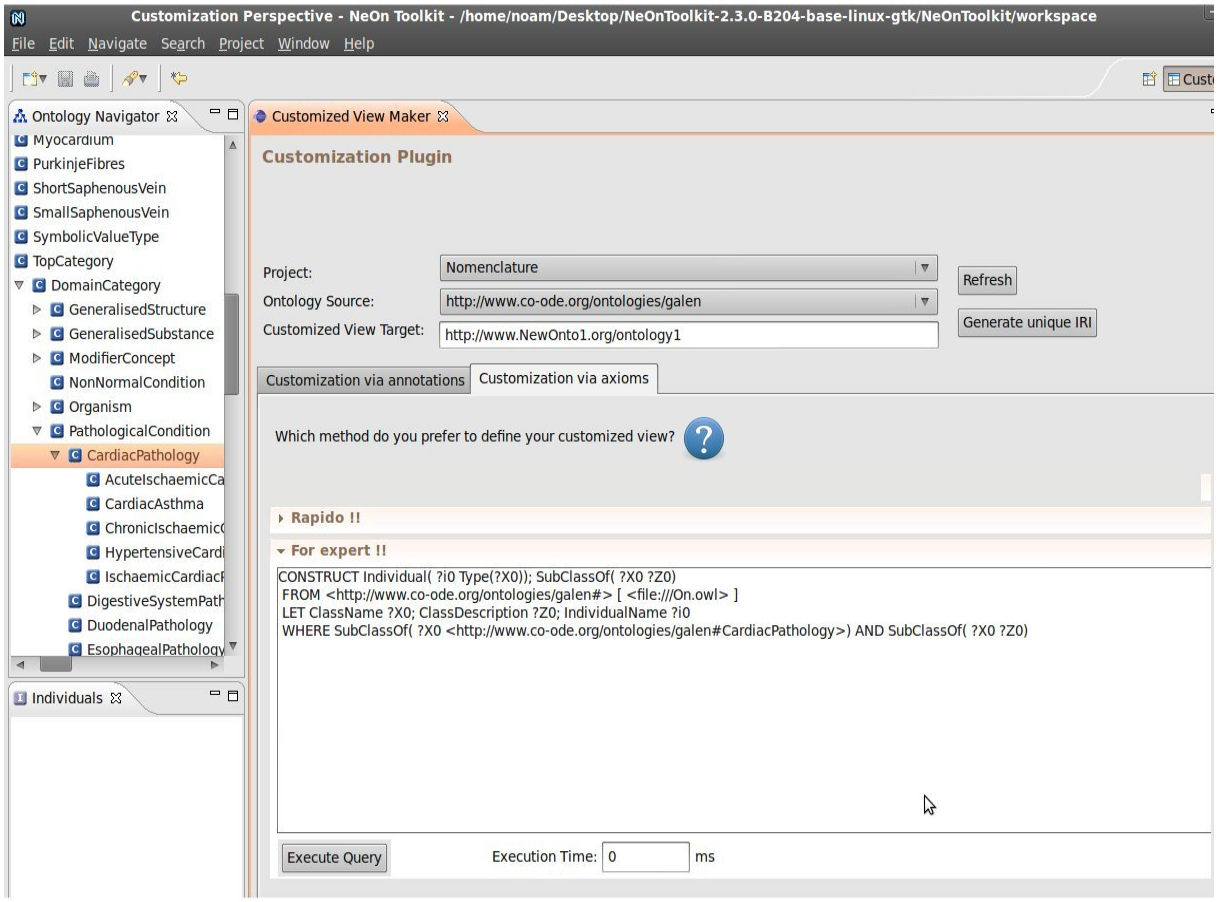

Figure 3.5: The expert section display of the view definition in the SAIQL syntax

to the other sections. This means that the user can easily switch between the different sections to define the view.

When the view definition is ready, the user can evaluate it using the “Evaluation” button. The result can take from a few seconds to several minutes depending on the complexity of the input ontology and the query. The output ontology will appear in the ontology tree view of the NeOn Toolkit under the namespace given by the user in the first step of the ontology customization process. Figure 3.6 shows the customized view of Galen focusing on the class “Cardiac Pathology”. The result can be saved as an ontology in the NeOn Toolkit and be re-used as an input for another plugin. For example, the user can easily share this new ontology using the cupboard plugin for NeOn.

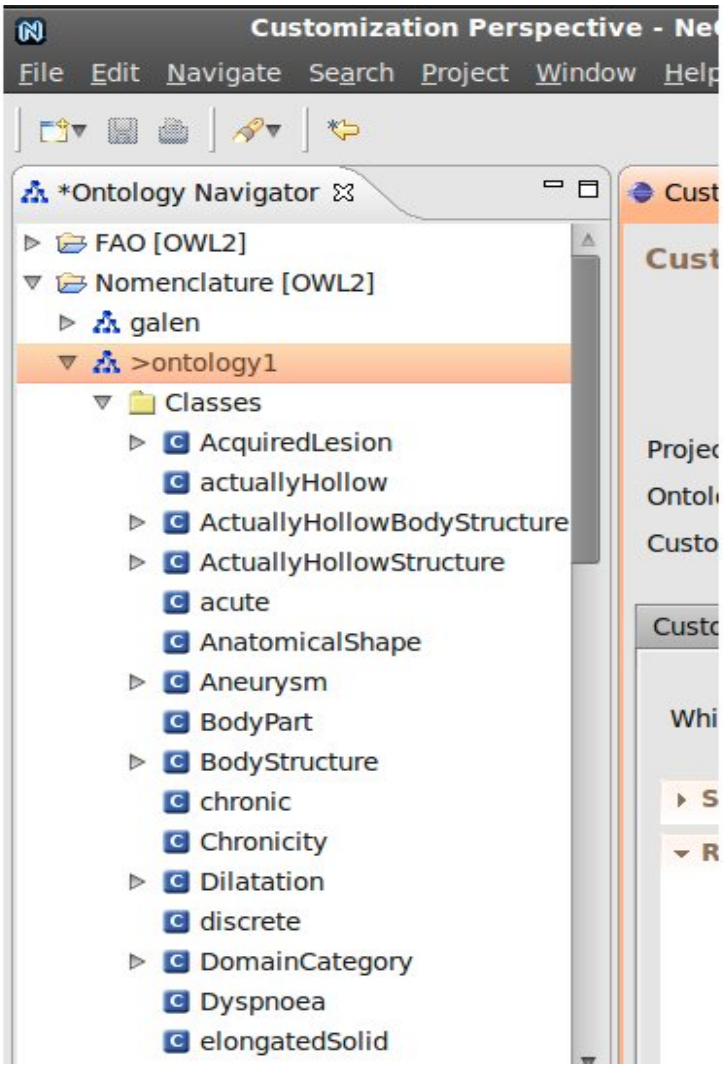

Figure 3.6: The customized view of the running example

Chapter 4

Conclusion

In this deliverable we have presented two plugins: the SAIQL plugin and the ontology customization plugin. The SAIQL plugin proposes a new means of querying an ontology to the user of the NeOn Toolkit. This new language aims at solving queries from the TBox and the ABox at one time. The ontology customization plugin gives the possibility to generate a customized view. This plugin is a front end user interface to the SAIQL plugin.

In this deliverable we focused on two use cases where the customization plays a major role. The nomenclature use case presents a frequent problem met by laboratories or medicine departments. Ideally, they would like to work with well used ontologies like Galen; however, ontologies such as Galen are covering more domains than are required by the user. There is a clear need for these ontologies to be used in sharing or reusing information, and we have explained in this deliverable why the user must be able to view only that which is related to his domain.

Bibliography

[BDS+08] Noam Bercovici, Martin Dzbor, Simon Schenk, Alexander Kubias, and Gerd GrÃűner. Ontology customization and module creation: query-based customization operators and model. Deliverable D4.2.2, NeOn Project, 2008.

[BS09] Noam Bercovici and Simon Schenk. Ontology customization prototype presentation. Deliverable D4.2.3, NeOn Project, 2009.

[CG07] Caterina Caracciolo and Aldo Gangemi. Revised and enhanced fisheries ontologies. Deliverable, FAO, 08 2007.

[DDM+07] Klaas Dellschaft, Martin Dzbor, Dunja Mladenic, Alexander Kubias, Carlos Buil Aranda, and Jose Manuel Gomez. Review of methodds and models for customizing/personalizing ontologies. Deliverable D4.2.1, NeOn Project, 2007.

[dHR+07] Matthieu d'Aquin, Peter Haase, Sebastian Rudolph, Jérôme Euzenat, Antoine Zimmermann, Martin Dzbor, Marta Iglesias, Yves Jacques, Caterina Caracciolo, Carlos Buil Aranda, and Jose Manuel Gomez. Neon formalisms for modularization: Syntax, semantics, algebra. Technical report, The NeOn Project, 02 2007.

[DKG+07] Martin Dzbor, Alexander Kubias, Laurian Gridinoc, Angel Lopez-Cima, and Carlos Buil Aranda. The role of access rights in ontology customization. Deliverable D4.4.1, NeOn Project, 2007.

[DMG+06] M. Dzbor, E. Motta, J. M. Gomez, C. Buil Aranda, K. Dellschaft, O. Grlitz, and H. Lewen. Analysis of user needs, behaviours & requirements wrt. user interfaces for ontology engineering. Deliverable D4.1.1, NeOn Project, 2006.

[Dzb09] Martin Dzbor. Realization of a prototype extension for access control in neon infrastructure. Deliverable D4.4.2, NeOn Project, 2009.

[Gan05] Aldo Gangemi. Ontology design patterns for semantic web content. In Yolanda Gil, Enrico Motta, V. Richard Benjamins, and Mark A. Musen, editors, *Proceedings of ISWC'05: the 4th International Semantic Web Conference, Galway, Ireland, November 6–10, 2005*, volume 3729 of *Lecture Notes in Computer Science*, pages 262–276. Springer, 2005.

[GPDLH07] Jose Manuel Gomez-Perez, Claire Daviaud, Tomas Pariente Lobo, and German Herrero. Software architecture for the neon pharmaceutical case studie. Deliverable, Intelligent Software Components (iSOCO), 2007.

[GPDM+06] Jose Manuel Gomez-Perez, Claire Daviaud, Berta Morera, Richard Benjamins, Tomas Pariente Lobo, German Herrero, and Gloria Tort. Analysis of the pharma domain and requirements. Deliverable, Intelligent Software Components (iSOCO), 09 2006.

[KS03] J. Komzak and P. Slavik. Scaleable GIS data transmission and visualisation. In *Proceedings of the International Conference on Information Visualization (IV)*, 2003.

[KSS06] Alexander Kubias, Simon Schenk, and Steffen Staab. Saiql query engine - querying owl theories for ontology extraction. 2007-06.

[LTD06] James F Brinkley Landon T Detwiler. Custom views of reference ontologies. In PubMed, editor, *American Medical Informatics Association Fall Symposium*, volume 2006; 2006: 909, 2006.

[LTDF07] James F Brinkley Landon T Detwiler and James F. Querying non-materialized ontology views. In PubMed, editor, *American Medical Informatics Association Fall Symposium*, 2007.

[RRP96] A. L. Rector, J. E. Rogers, and P. A. Pole. The galen high level ontology. pages 174–178. IOS Press, January 1996.

[Wie94] G. Wiederhold. An algebra for ontology composition. In *Proceedings of the Workshop on Formal Methods*, 1994.