ClickOn$_{\mathcal{A}}$: An Editor for *DL-Lite*$_{\mathcal{A}}$ based Ontology Design

Michael Wolters, German Nemirovski, and Andreas Nolle

Albstadt-Sigmaringen University, Business and Computer Science
Jakobstrasse 6, 72458 Albstadt, Germany
{wolters,nemirovskij,nolle}@hs-albsig.de
http://www.hs-albsig.de/win

Abstract. In recent years a lot of effort has been put into research dealing with accessing relational data with the help of ontologies. Solutions related to this issue focus on the Ontology-based Data Access (ODBA) architecture. Therefore the family of *DL-Lite* formalisms and in particular *DL-Lite*$_{\mathcal{A}}$ has been developed to enable ontology design for efficient processing of conjunctive queries targeting a large amount of data stored in relational data bases. However, due to strong limitations and the comparatively high complexity of constructs comprised by *DL-Lite*$_{\mathcal{A}}$, it can be quite a challenging task to design ontologies which are based on this formalism. This paper presents an editor for *DL-Lite*$_{\mathcal{A}}$ fulfilling ontology design. Thanks to its features, such as automated generation of axioms, on-the-fly inferring and graphical representation of relations between concepts, the editor simplifies the design process and improves the quality of the resulting ontologies significantly. The application and evaluation of the editor in two projects on data and knowledge integration for energy saving and urban planning confirmed these advantages.

Keywords: Ontology Editor, Ontology Design, Ontology Engineering, *DL-Lite*$_{\mathcal{A}}$, Graphical Representation, Inferring, Ontology based Data Access (OBDA)

1 Introduction

Visualization and GUI based editing of formal ontologies have been central to various projects since the introduction of the term ontology in the engineering and technological disciplines by Gruber [1]. After some time of evolving and proving to be competent with other systems the editor Protégé de facto became standard tool for ontology design. Yet, at least since the introduction of OWL 2 and OWL 2 profiles (OWL 2 QL, OWL 2 EL and OWL 2 RL) by the W3C consortium the limitations of available tools, including Protégé editor, became evident. Basically we have identified two challenges that have arisen in this context. Both of them concern validation or control of user input.

At first, when designing an ontology which has to follow the syntax declared in one of the OWL profiles or in one of the DL logics like $\mathcal{AL}$, $\mathcal{SHOIN}$,$\mathcal{SHIQ}$,

$\mathcal{EL}$ etc., the programming environment (i.e. ontology editor) should support the designer by taking these syntactical restrictions into account. The designer should be notified if a specification of an ontology element is not aligning with the definition of the selected profile or logic; for example if such specification uses a construct that is not available in the selected logic. Alternatively the environment should support only those constructs that are defined within the selected profile or logic, i.e. the GUI available for a designer must not contain constructs which aren't defined.

However, the grammar control of the ontology designer is only one challenging aspect. The second task concerns on-the-fly validation of semantics. Though, this issue appears to be more difficult to address since it requires on-the-fly reasoning, the degree of complexity and response time depends on the properties of the selected logic and the ontology being edited. In spite of the expected difficulties concerning the realization of these features, this demand seems to be just natural. Analogous to that in the area of software engineering, on-the-fly code validation belongs to the state of the art in this field and is implemented in numerous IDEs, such as eclipse[1].

In this paper we present the ontology editor *ClickOn*$_{\mathcal{A}}$, that implements an approach addressing two challenges mentioned above, i.e. taking into account the constructors and syntax available in a particular description logic (to be precise it is *DL-Lite*$_{\mathcal{A}}$) during the execution design/editing operations. The *DL-Lite* family and in particular *DL-Lite*$_{\mathcal{A}}$ has been developed as a fundamental element of the Ontology-based Data Access (ODBA) architecture to facilitate ontology design for efficient processing of conjunctive queries targeting a large amount of data stored in relational data bases.

ClickOn$_{\mathcal{A}}$ has been applied for the project SEMANCO[2] (Semantic Tools for Carbon Reduction in Urban Planning) focusing on implementation of analytic and data management services for carbon reduction in urban planning. Due to its features, such as automated generation of axioms, on-the-fly inferring and graphical representation of relations between concepts, *ClickOn*$_{\mathcal{A}}$ provides an extremely efficient -in terms of editing and evaluation time - environment for ontology design.

This paper is structured as follows: Chapter 3 outlines how *DL-Lite*$_{\mathcal{A}}$ specific aspects are applied in *ClickOn*$_{\mathcal{A}}$. The next chapter explains its features, like cooperative design, on-the-fly inferring and advantages in ontology visualization. Implementation details of the editor including the applied frameworks are introduced in section 5. In the following chapter an evaluation is described which proves the high efficiency of *ClickOn*$_{\mathcal{A}}$ compared to conventional ontology editors.

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

[2] The SEMANCO project is being carried out with the support of the Seventh Framework Programme "ICT for Energy Systems" 2011-2014, under the grant agreement Number 287534.

2 Related Work

Comparative studies for ontology design tools are provided in Khondoker [2] and in Kapoor and Sharma [3]. Descriptions of particular tools such as Protégé, WebODE and OntoEdit can be found in [2], [4], and [5] respectively.

Though the methodology issue is not reflected in this paper, it is important to understand that application of these tools brings maximal effect in the context of a particular methodology, where they are supplemented by process descriptions, templates and other tools. The methodical approach called Data Integration Driven Ontology Design, in whose context the *ClickOn*$_{\mathcal{A}}$ editor has been developed and its application is described in [6]. Some of the most recent methodical approaches are described in [7] and [8].

As stated above the *ClickOn*$_{\mathcal{A}}$ editor has been developed to design ontologies on the basis of the *DL-Lite*$_{\mathcal{A}}$ formalism, which belongs to the *DL-Lite* family of description logics. Another two logics of this family, DL-Lite$_{\mathcal{F}}$ and DL-Lite$_{\mathcal{R}}$ were described by Calvanese et al. in [9]. One year later Poggi et al. published the paper about *DL-Lite*$_{\mathcal{A}}$ [10]. [11] and [12] describe principles of query answering using *DL-Lite* ontologies and the *DL-Lite* capable reasoner *Quest*.

3 *DL-Lite*$_{\mathcal{A}}$ support by *ClickOn*$_{\mathcal{A}}$

In [14] *DL-Lite*$_{\mathcal{A}}$ is referred as the Description Logic of the *DL-Lite* familiy that is closest to OWL 2.0. It is a compromise between the maximum expressiveness of the ontology specification language and the optimal time performance when processing conjunctive queries referring to elements of ontologies specified using this language. Typically these are ABox queries which are aiming at collecting data properties related to specific individuals and stored in relational data sources.

Regarding OWL 2 the two most important aspects of *DL-Lite*$_{\mathcal{A}}$ are on the one hand a restriction that allows range/domain specifications for functional properties only by connecting concepts with data types. On the other hand, for non-functional properties connecting two concepts, domain and range have to be modeled through axioms using "owl:Restriction". For example the following axioms use subsumption ($\sqsubseteq$), existence quantification ($\exists$) and inversion ($^-$) to express the fact that the concept *Land_Vehicle* relates to the class *Wheel* via the *hasWheel* property.

$$\exists hasWheel \sqsubseteq Land\_Vehicle \tag{1}$$

$$\exists hasWheel^- \sqsubseteq Weel \tag{2}$$

In RDF/XML these statements are specified as follows:

```
<owl:Restriction>
  <rdfs:subClassOf rdf:resource=".../clickOnA.owl#Wheel"/>
  <owl:onProperty>
```

```
    <rdf:Description>
      <owl:inverseOf rdf:resource=".../clickOnA.owl#hasWheel"/>
    </rdf:Description>
  </owl:onProperty>
  <owl:someValuesFrom rdf:resource="...org/2002/07/owl#Thing"/>
</owl:Restriction>
```

and

```
<owl:Restriction>
  <rdfs:subClassOf rdf:resource=".../clickOnA.owl#Land_Vehicle"/>
  <owl:onProperty rdf:resource=".../clickOnA.owl#hasWheel"/>
  <owl:someValuesFrom rdf:resource="....org/2002/07/owl#Thing"/>
</owl:Restriction>
```

In fact if axioms are specified in this style, domains and ranges of the corresponding properties can be inferred by a reasoner software. *ClickOn$_\mathcal{A}$* was implemented to infer ranges and domains from axioms coded this way, converts them into a graphical presentation and thus hides the complex code from the user (Figure 1).

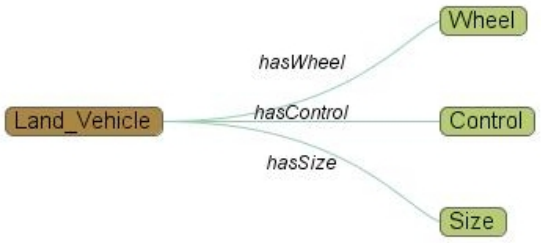

Fig. 1. Graphic representation of properties, domains and ranges by *ClickOn$_\mathcal{A}$*

Apart from interpretation and graphical representation of axiomatically specified domains and ranges, *ClickOn$_\mathcal{A}$* allows generation and modification of existing axioms. Here as well, the user designs without the complex axiomatic nature of the specification and invokes corresponding operations by selection of a context menu option, as shown in figure 2. Code that strictly aligns with the regulations of *DL-Lite$_\mathcal{A}$* is generated automatically. The editor's on-the-fly reasoning functionality enables that this code is interpreted as domain/range specifications.

Defining *DL-Lite$_\mathcal{A}$* conform code as shown above is probably the biggest problem when a conventional ontology editor like Protégé[3] is used. To specify one domain/range pair users have to switch between three different subwindows: "class description", "object property hierarchy" and "general class

[3] http://protege.stanford.edu/

Fig. 2. Context menu selection in *ClickOn*$_{\mathcal{A}}$ leads to generation of complex code

axioms". When a high number of suchlike relations have to be specified, errors are almost inevitable.

4 Usability and effective authoring

Furthermore *ClickOn*$_{\mathcal{A}}$ implements a set of features aiming at high usability and effectiveness of the authoring/design process. These features basically are

- Partitioning of the complex editing task into a group of smaller and simpler tasks
- Advanced ontology visualization, and support of user navigation along the ontology graph

4.1 Partitioning of the editing task

ClickOn$_{\mathcal{A}}$ considers the editing process as a combination of three sub-processes: i) editing of a subsumption taxonomy; ii) editing of a non-subsumption properties graph and iii) annotation of concepts. Correspondingly, three different perspectives, presented in the screen simultaneously are offered for users: the taxonomy perspective (figure 3, left), the properties graph (figure 3, right), and the meta data perspective (figure 3, upper part).

In the middle column between the taxonomy perspective and the properties graph there is a control panel that can be moved horizontally, enlarging one of the perspectives to the left or to the right side. The advantages of this visual structure become evident when *ClickOn*$_{\mathcal{A}}$ is used on a system with two monitors. In this case one monitor can be dedicated to each of these two perspectives.

A new concept can be created either in the taxonomy perspective or in the properties graph. In the former case this is done using the context menu, when the concept that is supposed to be subsumed by a new one is clicked with the right mouse button. In the property graph, new concepts can be created through a context menu as well. In this case however, instead of a new concept, a new property relation specified by a set of axioms, has to be created as described

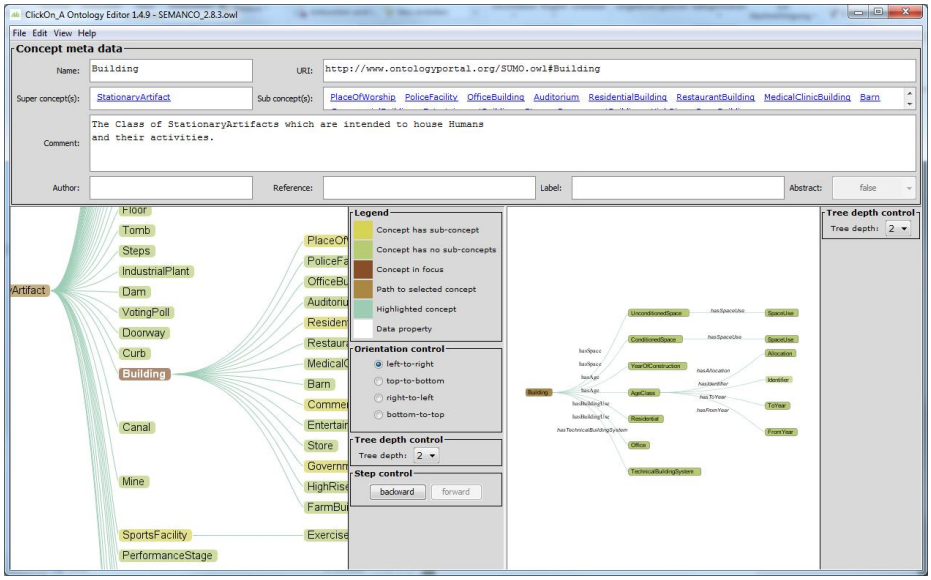

Fig. 3. Different editing perspectives in *ClickOn*$_{\mathcal{A}}$

in section 3. The label for this property is generated automatically. New concepts added in the property graph are initially subsuming the concept called *Suspense*, which is automatically generated by *ClickOn*$_{\mathcal{A}}$ as root of a temporary sub-taxonomy. Later, concepts subsuming *Suspense* are relocated in the taxonomy to a more suitable place.

Moreover, the meta data perspective (see figure 4) offers several fields for a concept's metadata. The fields "Comment", "Author", "Reference" and "Label" can be modified. Other fields contain values generated by the system. The initial value of the "Author" field is generated automatically depending on user settings.

Fig. 4. Meta data perspective

4.2 Ontology visualization and navigation support

To achieve high usability and optimized ontology authoring it is important that users are able to navigate through both inferred graphs (taxonomy and the properties graph) in a most natural and transparent way. In this context *ClickOn*$_{\mathcal{A}}$ offers the following settings that are shown on the control panel displayed in the middle of figure 3.

- All nodes of the ontology graph are colored with regard to their meaning (the legend on the control panel explains the different colors)
- Mouse click navigation: After a concept is clicked on with the left mouse button it is going to be centered on the screen and relations originating from the corresponding graph node and (optionally) form its sub nodes become visible (see node *Compactness_Ratio* in figure 5). Double clicking on a node results in a new tree presentation where the selected concept becomes the root of a taxonomy. (see node *CS_Geometry* in figure 5)
- Enlarging or collapsing parts of the ontology graph by a zoom function
- Controlling the tree depth (one, two or three levels can be displayed simultaneously)
- Changing graph direction from e. g. left to right or bottom down
- Forward and backward navigation within a selection history

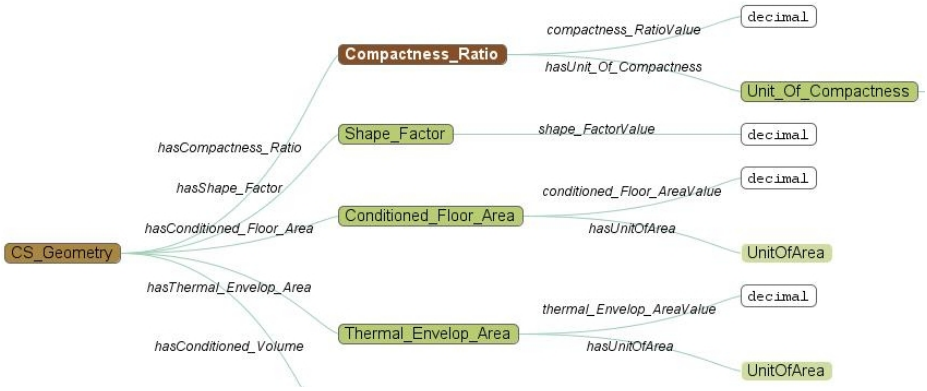

Fig. 5. Visualization of properties graph by *ClickOn*$_{\mathcal{A}}$

Additional navigation options can be found in the meta data perspective. There, all super- and sub-concepts of a selected node are listed. By clicking on one of them the editor navigates to the corresponding root node. Besides, the ontology editor offers a powerful option for searching concepts using autocomplete functionalities. If only a part of a concept name is entered by the user the system offers him a list of concept names containing the entered part (see figure 6).

In the current version of the *ClickOn$_\mathcal{A}$* editor is able to import and export ontologies as .owl files using the RDF/XML format.

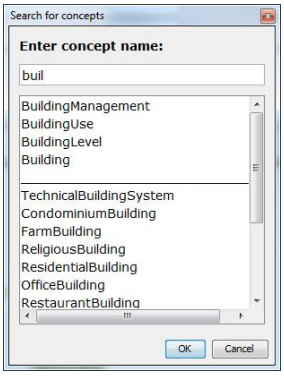

Fig. 6. Searching for concepts in *ClickOn$_\mathcal{A}$*

5 Implementation aspects

Implementation of *ClickOn$_\mathcal{A}$* has been carried out on the basis of a Java Swing graphical user interface and using basically three frameworks: *Prefuse*[4], *Apache Jena*[5] and *JDOM*[6]. The *Prefuse* framework expands the Java 2D graphic library and can easily be integrated into any Java Swing software. The functionalities of *Prefuse* are used in *ClickOn$_\mathcal{A}$* to display graphs of concepts and relations. The *Apache Jena* framework offers a lot of tools and libraries for reading, processing and writing RDF data in XML; for dealing with OWL ontologies and for rule-based reasoning. *JDOM* is an open source Java-based document object model for XML. In the *ClickOn$_\mathcal{A}$* *JDOM* is used to manage (access and modify) files written in the Web Ontology Language (OWL), whereby the OWL is a derivate of the XML, defined by means of a set of name spaces.

6 Evaluation of *DL-Lite$_\mathcal{A}$* capabilities of ontology editors

The available evaluations of ontology editors (for example [13]) focus basically on usability or language support (in terms of RDF, OWL, XML, etc.). So these cannot be helpful in an assessment of *DL-Lite* support provided (or not provided) by different tools.

[4] http://prefuse.org/
[5] http://jena.apache.org/
[6] http://www.jdom.org/

The advantages of *ClickOn*$_{\mathcal{A}}$ for editing *DL-Lite*$_{\mathcal{A}}$ conform ontologies can be illustrated by a comparison of design processes carried out with *ClickOn*$_{\mathcal{A}}$ and with different conventional ontology editors. The editors used for this evaluation are: *ClickOn*$_{\mathcal{A}}$, Protégé 4.2, NeoOn Toolkit 2.5.2[7], OntoStudio 3.2.0[8], Hozo 5.2.36[9], OWLGrEd 1.6.0[10] and TopBraid Composer 4.2.1[11].

This evaluation was carried out by three ontology experts. Taking into account that the *ClickOn*$_{\mathcal{A}}$ editor is designed especially for domain experts an evaluation with participants of this group would most probably show a much higher difference between the process time results, in favor of *ClickOn*$_{\mathcal{A}}$.

First the process of creating a new, tiny ontology containing 8 concepts has been evaluated. In the second step three properties with axiomatically defined ranges and domains were specified. The third step aimed at presentation of information inferred from axioms created in the previous step. The results of this evaluation regarding the duration of these three processes are displayed in the following table:

Table 1. Evaluation results measured in seconds

	Cli.	**Pro.**	**NeO.**	**Ont.**	**Hoz.**	**OWL.**	**Top.**
Creating 8 new concepts	90	80	70	70	75	90	75
New relations, *DL-Lite*$_{\mathcal{A}}$ conform	25	120	-	-	-	-	-
Presenting inferred relations	0	3	-	-	-	-	-

Besides *ClickOn*$_{\mathcal{A}}$ only Protégé was able to create the axioms required by the *DL-Lite*$_{\mathcal{A}}$ formalism. The creation of new ontology concepts using *ClickOn*$_{\mathcal{A}}$ results in a slightly longer time span because of its current implementation whereby renaming of newly created concepts requires an additional mouse click compared to the other editors. In Protégé users have to enter two axioms (see section 2) for each relation manually, whereas *ClickOn*$_{\mathcal{A}}$ generates these fully automatically. On-the-fly inferring supported by *ClickOn*$_{\mathcal{A}}$ enables immediate graphic presentation of relations created in the second process (line 3 in the table above). When

[7] http://neon-toolkit.org/wiki/Main_Page
[8] http://www.semafora-systems.com/de/produkte/ontostudio/
[9] http://www.hozo.jp/
[10] http://owlgred.lumii.lv/index.html
[11] http://www.topquadrant.com/products/TB_Composer.html

using Protégé, at first a reasoner, for example *Fact++*, has to be executed before any inferred information can be shown to the user. Unlike *ClickOn*$_{\mathcal{A}}$ Protégé isn't able to present this information graphically but only as additional textual description of a selected property (figure 7).

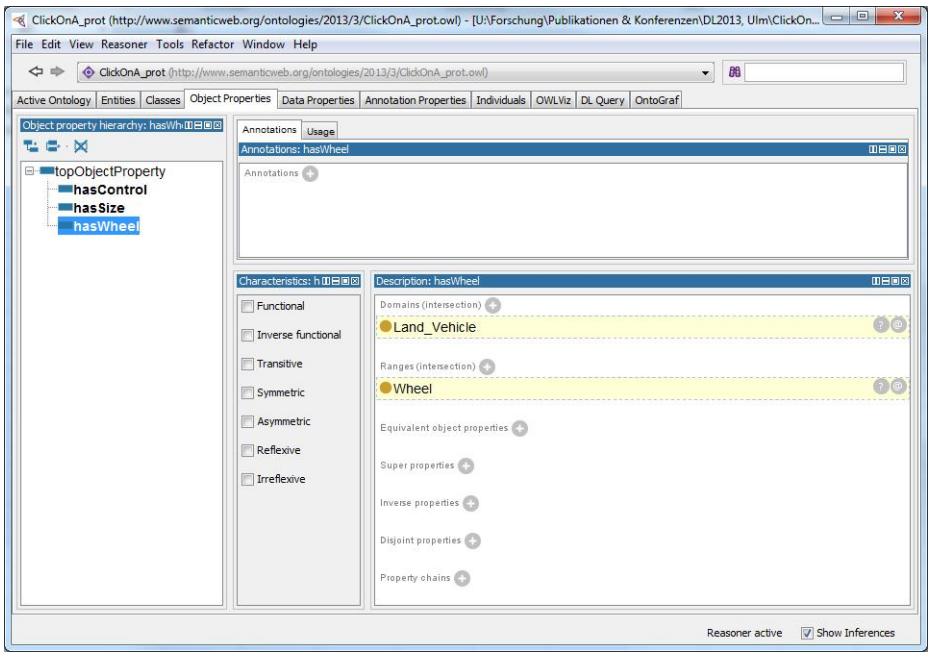

Fig. 7. Results of inferring shown in the ontology editor Protégé

However the advantages of *ClickOn*$_{\mathcal{A}}$ become evident when the size of the ontology to be edited grows. Comfortable navigation instruments offer perfect orientation, save time and reduce the number of errors significantly.

7 Conclusion

In this paper we have presented an innovative ontology editor. *ClickOn*$_{\mathcal{A}}$ is dedicated to the design of ontologies basing on the *DL-Lite*$_{\mathcal{A}}$ formalism. Due to its restrictive architecture application of *DL-Lite*$_{\mathcal{A}}$ may lead to an increasing complexity of the ontology code. In this context real time evaluation of user input becomes essential. Due to on-the-fly inferring, powerful graphical presentation and control options *ClickOn*$_{\mathcal{A}}$ is able to give very expressive and illustrative feedback to the user, immediately after his input. Furthermore graphical widgets enable automated rapid and error-free generating of complex axioms, and offer a high level of usability. In the nearest future *ClickOn*$_{\mathcal{A}}$ will be enhanced by mechanisms for versioning, user and change management.

References

1. Gruber, T.: Towards principles for the design of ontologies used for knowledge sharing. In: International Journal of Human Computer Studies, Vol. 43 (5–6), pp. 907–928. ELSEVIER, (1995)
2. Khondoker, M.R., Mueller, P.: Comparing Ontology Development Tools Based on an Online Survey. In: Proceedings of the World Congress on Engineering 2010, Vol. I WCE 2010, London (2010)
3. Kapoor, B. and Sharma, S.: A Comparative Study Ontology Building Tools for Semantic Web Applications. International journal of Web & Semantic Technology (IJWesT), Vol.1, Num.3. (2010)
4. Arprez, J.C., Corcho, O., Fernndez-Lpez, M. Gmez-Prez, A.: WebODE: a scalable workbench for ontological engineering. In: Proceedings K-CAP '01 Proceedings of the 1st international conference on Knowledge capture, pp. 6-13. ACM New York (2001)
5. Surez-Figueroa, M.C., Gmez-Prez, A., Motta, E. and Gangemi., A.: Ontology Engineering in a Networked World. Springer, Berlin (2012)
6. Nemirovski, G., Nolle. A., Sicilia, A., Ballarini, I., Corrado, V.: Data Integration Driven Ontology Design, Case Study Smart City. In: SemCity Wokshop, Madrid (accepted paper) (2013)
7. Fernandes, B.C.B., Guizzardi, R.S.S. and Guizzardi, G.: Using Goal Modeling to Capture Competency Questions in Ontology-based Systems. Journal of Information and Data Management, Vol. 2, No 3, pp. 527-540 (2011)
8. Fonou-Dombeu, J.V. and Huisman, M.: Combining Ontology Development Methodologies and Semantic Web Platforms for E-government Domain Ontology Development. International Journal of Web and Semantic Technology (IJWesT) Vol. 2, No. 2 (2011)
9. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. Journal of Automated Reasoning 39(3), pp. 385-429. Springer, New York (2007)
10. Poggi, A. et al: Linking Data to Ontologies. Journal on Data Semantics, X, pp. 133-173. Springer, Heidelberg (2008)
11. Rodriguez-Muro, M. and Calvanese, D.: High Performance Query Answering over *DL-Lite* Ontologies. In: Proc. of the 13th Int. Conf. on the Principles of Knowledge Representation and Reasoning. pp. 308-318 (2012)
12. Rodriguez-Muro, M. and Calvanese, D.: Quest, an OWL 2 QL Reasoner for Ontology-Based Data Access. In: Proc. of the 9th Int. Workshop on OWL: Experiences and Directions (OWLED 2012), CEUR-WS.org (2012)
13. Alatrish, E. S.: Comparision of Ontology Editors. e-RAF Journal on Computing, Vol. 4 (2012)
14. Corona, C., Di Pasquale, E., Poggi, A., Ruzzi, M. and Savo, D. F.: When *DL-Lite* met OWL... (extended abstract). In: Proc. of OWLED 2008, CEUR-WS.org (2009)