MOPAS 2012

The Third International Conference on Models and Ontology-based Design of Protocols, Architectures and Services

ISBN: 978-1-61208-196-0

April 29 - May 4, 2012

Chamonix / Mont Blanc, France

MOPAS 2012 Editors

Petre Dini, Concordia University, Canada / China Space Agency Center, China

MOPAS 2012

Foreword

The Third International Conference on Models and Ontology-based Design of Protocols, Architectures and Services [MOPAS 2012], held between April 29th and May 4th, 2012 in Chamonix / Mont Blanc, France, proposed a new context for presenting achievements, surveys and perspectives in the areas of design, architecture and implementation based on ontologies and related models.

We take here the opportunity to warmly thank all the members of the MOPAS 2012 Technical Program Committee. The creation of such a high quality conference program would not have been possible without their involvement. We also kindly thank all the authors who dedicated much of their time and efforts to contribute to MOPAS 2012. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

Also, this event could not have been a reality without the support of many individuals, organizations, and sponsors. We are grateful to the members of the MOPAS 2012 organizing committee for their help in handling the logistics and for their work to make this professional meeting a success.

We hope that MOPAS 2012 was a successful international forum for the exchange of ideas and results between academia and industry and for the promotion of progress in the field of models and ontology-based design and protocols, architectures and services.

We are convinced that the participants found the event useful and communications very open. We also hope the attendees enjoyed their stay in the French Alps.

MOPAS Advisory Committee:

Raj Jain, Washington University in St. Louis, USA
Marc Lacoste, Orange Labs, France

MOPAS 2012 PROGRAM COMMITTEE

MOPAS Advisory Committee

Raj Jain, Washington University in St. Louis, USA
Marc Lacoste, Orange Labs, France

MOPAS 2012 Technical Program Committee

Mourad Alia, Orange Business Services- Grenoble, France
João Caldeira, Instituto de Telecomunicações / University of Beira Interior / Polytechnic Institute of Castelo Branco, Portugal
Shu-Ching Chen, Florida International University - Miami, USA
Vanea Chiprianov, Telecom Bretagne, France
Michel Diaz, LAAS, France
Rachida Dssouli, Concordia University, Canada
Antti Evesti, VTT Technical Research Centre of Finland, Finland
Bin Guo, Northwestern Polytechnical University, China
Robert C. H. Hsu, Chung Hua University, Taiwan
Zahid Iqbal, University Graduate Center (UNIK) - Kjeller, Norway
Brigitte Jaumard, Concordia University - Montreal, Canada
Achilles Kameas, Hellenic Open University-Patras, Greece
Marc Lacoste, Orange Labs, France
Thomas D. Lagkas, University of Western Macedonia, Greece
Vlad Nicolicin Georgescu, Université de Nantes / SP2 Solutions - La Roche sur Yon, France
Peera Pacharintanakul, TOT, Thailand
Arun Prakash, Fraunhofer Institute for Open Communication Systems (FOKUS) - Berlin, Germany
Neeli R. Prasad, Aalborg University, Denmark
Vasco Soares, Instituto de Telecomunicações / University of Beira Interior / Polytechnic Institute of Castelo Branco, Portugal
Shensheng Tang, Missouri Western State University, USA
Zoltan Theisz, evopro Informatics and Automation, Ltd., Hungary
Dimitrios D. Vergados, University of Piraeus, Greece
Roberto Willrich, Santa Catarina Federal University, Brazil
Nataša Živic, University of Siegen, Germany

Copyright Information

For your reference, this is the text governing the copyright release for material published by IARIA.

The copyright release is a transfer of publication rights, which allows IARIA and its partners to drive the dissemination of the published material. This allows IARIA to give articles increased visibility via distribution, inclusion in libraries, and arrangements for submission to indexes.

I, the undersigned, declare that the article is original, and that I represent the authors of this article in the copyright release matters. If this work has been done as work-for-hire, I have obtained all necessary clearances to execute a copyright release. I hereby irrevocably transfer exclusive copyright for this material to IARIA. I give IARIA permission or reproduce the work in any media format such as, but not limited to, print, digital, or electronic. I give IARIA permission to distribute the materials without restriction to any institutions or individuals. I give IARIA permission to submit the work for inclusion in article repositories as IARIA sees fit.

I, the undersigned, declare that to the best of my knowledge, the article is does not contain libelous or otherwise unlawful contents or invading the right of privacy or infringing on a proprietary right.

Following the copyright release, any circulated version of the article must bear the copyright notice and any header and footer information that IARIA applies to the published article.

IARIA grants royalty-free permission to the authors to disseminate the work, under the above provisions, for any academic, commercial, or industrial use. IARIA grants royalty-free permission to any individuals or institutions to make the article available electronically, online, or in print.

IARIA acknowledges that rights to any algorithm, process, procedure, apparatus, or articles of manufacture remain with the authors and their employers.

I, the undersigned, understand that IARIA will not be liable, in contract, tort (including, without limitation, negligence), pre-contract or other representations (other than fraudulent misrepresentations) or otherwise in connection with the publication of my work.

Exception to the above is made for work-for-hire performed while employed by the government. In that case, copyright to the material remains with the said government. The rightful owners (authors and government entity) grant unlimited and unrestricted permission to IARIA, IARIA's contractors, and IARIA's partners to further distribute the work.

Table of Contents

Use of Persistent Meta-Modeling Systems to Handle Mappings for Ontology Design

Valéry Téguiak
LIAS/ISAE-ENSMA
Futuroscope, France
teguiakh@ensma.fr

Yamine Ait-Ameur
IRIT-ENSEEIHT
Toulouse, France
yamine@enseeiht.fr

Éric Sardet
CRITT Informatique
Futuroscope, France
sardet@ensma.fr

***Abstract*—To enable data intensive application including global information systems with heterogeneous models, the model mapping problem in which a source model is mapped to a target one should be addressed. Current work about mapping provides a finite set of mapping constructors available for writing mappings. In this case, adding a new concept in a meta-model describing mapped schemas could have the effect of building new types of mapping constructors. Thus, this paper attempts to provide a generic and systematic approach for modeling mapping constructors, so that new mapping constructors could be handled efficiently without requiring to rebuild completely the mapping repository system.**

Keywords-data integration; mapping; meta-modeling; model transformation; ontology engineering.

I. INTRODUCTION

The huge amount of data created by several application domains and development activities was at the origin of the emergence of several heterogeneous data models and modeling languages. The need of exploiting data and these models in an integrated manner led to several studies on data and model integration and heterogeneous modeling [1], [2]. Two particular and interesting studies are model mappings and mapping languages. Moreover, these approaches have also been developed in the context of the semantic web where model mappings were required for defining transformations, instance migration, etc.

In order to deal with various heterogeneous models used to represent the same real word domain, several mapping languages [8], [12], [20] have been proposed. Their central objective is to establish relationships between models. Most of these approaches run in central memory and do not address the scalability problem when dealing with huge amount of data, instances of those models.

However, many information systems rely on databases to ensure scalability. As a consequence, the need of managing mappings in a persistence context appeared. Therefore, the availability of a repository of mappings is required. Also, a way for exploiting mappings by interpreting, handling and manipulating such mapping operators is also required.

The work of Miller et al. [3] and Ling et al. [4] was precursor. It addressed the problem of mapping management in a database. The main assumption in this work consists in modeling mapping constructors as a finite set of operators. This assumption is acceptable if models to be mapped are encoded in a meta-model that will not evolve dynamically due to the fixed set of mapping constructors.

Nevertheless, offering the capability to manipulate and/or to modify the meta-model could offer more flexibility and extensions capabilities dynamically. Indeed, offering the capability for the meta-model to evolve by supporting the creation of new concepts would also offer the capability to dynamically define on the fly new mapping constructors. As a consequence, the definition of mapping constructors in a generic way becomes possible.

Moreover, because the size of models and instances are growing drastically, the traditional approaches for mapping models need to scale up. Therefore, offering persistent settings for managing such mappings and instances becomes a necessity if one wants to address real sized problems.

This paper focuses on the definition of a generic infrastructure for managing mappings in a database context. It uses a specific database architecture that supports definition of meta-models and their instances. This database infrastructure consists of: (1) a space for representing mapping constructors, (2) a space for representing models and mappings between these models and finally (3) a space for representing data (instances of models).

This paper is organized as follows. Section II outlines related work on mappings. Then, our contribution using constructive data models to model mappings is presented in Section III. In Section IV, we discuss how to represent a graph of mappings in a persistent context. Once our persistent solution for handling mappings in a database structure, through model repositories, is presented in Section V, we briefly present, in Section VI, how this approach has been set up to encode the transformation process for building ontologies starting from texts. This work has been conducted in the context of the DaFOE4App (Differential and Formal Ontologies Editor for Application) project [22]. We finally conclude and give some perspectives of this work.

II. RELATED WORK

Many proposals address model mapping and data translation problems. These proposals can be splitted into two categories: hard encoded and rule-based approaches.

 ISBN: 978-1-61208-196-0

A. Hard encoded approach

By the term *hard encoded*, we refer to approaches where both mappings and mapped models representations are hidden within a framework as a program. It means that these representations are not exposed as declarative and user-comprehensible rules. This leads to several difficulties. First, models and mappings can only be extended by the framework designers. Secondly, because of the program-based representation of models and mappings, any extension requires changes at the framework code level. As a consequence, correctness of these representations has to be accepted by users as a dogma. For example, the approach of Papotti and Torlone [14] can be said to be hard encoded. In that context, the expressed transformations are imperative programs, which have the weaknesses described above. The instance translation process is achieved by firstly converting the source data into XML, and then by performing an XML-to-XML translation expressed in XQuery to reshape instances in order to be compatible with the target schema, and finally, by converting the XML representation into the target model.

B. Rule-based approach

Weaknesses of the hard encoded approach can be solved using a rule-based approach. This approach attempts to provide a generic way to handle models, mappings and data translation without using a hard encoded program. For example, the approach proposed by Bernstein et al. [13] is a rule-based one. In that approach, they focus on a flexible mapping based on inheritance hierarchies, and in the incremental regeneration of mappings each time the source schema is modified. Other rule-based approaches are driven by a dictionary of schemas, models and translation rules. Among them, we can quote the work of Bowers and Delcambre [11] that proposes Uni-Level Description (UDL). UDL is a meta-model in which models and translations can be described and managed in a uniform process environment for models, schemas and instances. UDL is used to express specific model-to-model translations of both schemas and instances. Like the approach of Atzeni et al. [16], translations are expressed as Datalog rules and the source and target models are stored in a generic relational dictionary.

Our approach is also considered as a rule-based approach. But, compared to the previous quoted approaches, we provide a more abstract level where, in addition, the dictionary is explicitly represented and becomes manageable. Indeed, the dictionary representation according to a meta-meta-model allows the user for example, to modify mapping models without modifying the underlying program.

Furthermore, Kalfoglou and Schorlemmer [7] address the problem of mapping discovery which consists of an automatic synthesis of an alignment between models. In our proposal, we assume that the discovery process has already been achieved. Indeed, our work deals with mapping specification and instance mediation in database environment. More discussions on topics around mapping problems and provided solutions can be founded in [4], [9], [15], [18]. As illustrated in Figure 1, mappings can be composed transitively. This requirement has been formalized in [10], [19], where an approach to use composition among models has been proposed. Because this paper focuses on a repository for storing mappings, we do not discuss handling composition between mappings (composition is handled by a query engine in our framework). Furthermore, [3], [4] introduce the notion of *value correspondence* as a proposal of representation for mapping operators.

III. Our approach

In our approach, modeling mapping consists in creating mapping constructors (Model level mapping, Entity level mapping, Attribute level mapping, etc.). In this section, we present a formal model for mapping constructors. Furthermore, before connecting domain models using mappings, these models should be represented in a way allowing them to be managed efficiently. The meta-modeling-based approach is often used for this purpose. We use a meta-model called Entity-Attribute meta-model (E-A meta-model) to handle domain models. Using this E-A meta-model, a model *m* is formally defined by $m = \langle E, A, I, T, dom, range, its\_entity \rangle$ where:

- ***E*** represents the set of entities of the model *m*;
- ***A*** represents the set attributes used to describe entities of the model *m*;
- ***I*** represents the set of entity instances of the model *m*;
- ***T*** is a set of primitive types (Int, String, Boolean, etc.);
- $dom : A \rightarrow E$ defines the domain of an attribute;
- $range : A \rightarrow E \cup T$ defines the range of an attribute;
- $its\_entity : I \rightarrow E$ returns the entity associated to a given instance.

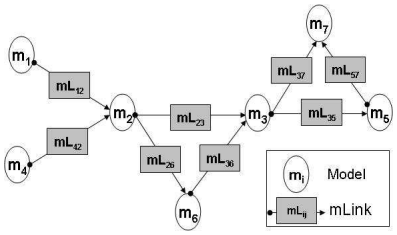

Figure 1. An mLink graph of model mappings.

A. Model level mapping: mLink

Correspondences between models are represented by a directed acyclic graph whose nodes are models. Formally,

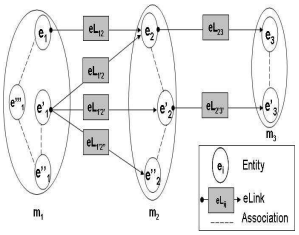

Figure 2. eLink graph.

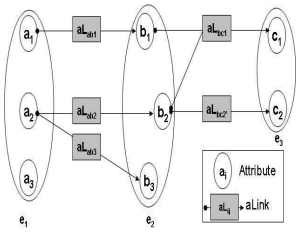

Figure 3. aLink graph.

if ***M*** represents a set of models, the graph G_m of correspondences between models (Cf. Figure 1) is defined by $G_m = (N_m, L_m)$ where:

- $N_m \subseteq M$ represents the set of nodes of the correspondence graph;
- $L_m = \{(m_s, m_t, \alpha_m) \in N_m \times N_m \times [0;1]\}$ represents a set of correspondences between a source model (m_s) and a target model (m_t). Here, α_m is a confidence degree of this correspondence.

Through this paper, we will use the term ***mLink***, formally defined as element of L_m, to refer to a correspondence or a mapping between models.

B. Entity level mapping: eLink

Correspondences between entities of the models are represented by a directed acyclic graph whose nodes are entities. Formally, if ***E*** represents a set of entities, the graph G_e of correspondences between entities (Cf. Figure 2) is defined by $G_e = (N_e, L_e)$ where:

- $N_e \subseteq E$ is the set of nodes of the correspondence graph;
- $L_e = \{(e_s, e_t, \alpha_e, mL) \in N_e \times N_e \times [0;1] \times L_m\}$ represents correspondences between a source entity (e_s) and a target entity (e_t) built in the context of the *mLink* *mL* with confidence degree α_e.

Through this paper, we will use the term ***eLink***, formally defined as element of L_e, to refer to a correspondence between entities.

C. Attribute level mapping: aLink

The arity of correspondences between attributes is n:0 reflecting the fact that one needs zero or more attributes of the source entity to compute an attribute of the target entity. Formally, if ***A*** represents the set of attributes, the graph G_a of correspondences between attributes (Cf. Figure 3) is defined by $G_a = (N_a, L_a)$ where:

- $N_a \subseteq A$ represents the set of nodes of the correspondence graph;
- $L_a = \{(A_s, a_t, \alpha_a, \varphi, eL) \in N_a^k \times N_a \times [0;1] \times \Phi \times L_e\}$ represents a set of correspondences from a set of sources attributes $A_s = (a_{s1}, a_{s2}, ..., a_{sk})$ to a target one ($a_t \in N_a$) where:
  - the eLink *eL* represents the context in which the correspondence has been created;
  - φ is an expression used to write a_t in terms of a_{si} ($1 \leq i \leq k$).
  - α_a represents the confidence degree of the correspondence.

Through this paper, we will use the term ***aLink***, formally defined as element of L_a, to refer to a correspondence between attributes.

Unlike other work performed on mappings [3], [4], [12], our approach does not presuppose the existence of a finite set of mapping constructors but it aims at providing a formal support to dynamically create new mapping constructors or evolving existing one. Indeed, all mapping constructors presented above use a numeric confidence degree α. This is the most used approach for handling fuzzy mappings. However, what will happen if a user (who is not the framework designer) want to create another fuzzy property for mapping by annotating each mapping with a quality value ("Weak", "Average", "Good", "Very", "Good", "Excellent", etc. for example)? Thus, our approach propose to model mapping constructors so that mapping constructors like *mLink*, *eLink*, *aLink*, etc., could be managed in a generic way and easily extended at runtime. As our work is conducted in a database context, we discuss in the following sections, modeling possibilities of a database repository to store such a graph-based representation of mapping.

IV. MODEL REPOSITORIES

In the recent years, several works [6], [17] investigated the problem of representing ontologies and their instances within a database. We reuse this approach for models in general and the resulting database (that we simply call Model Based-Database (MBDB)) can be represented according in three main approaches. In this section, we present a taxonomy of these approaches. Our goal here is to discuss how the graph of mappings presented in Section III can be stored in each of these database types.

A. MBDB of type 1

In MBDB of type 1, information is represented using a single schema composed of a single table of triples (subject, predicate, object). This table, referred as vertical table [5] is used both for model level data and instance level data. For model level data, the three columns of this table respectively represent the Identifier of an element of the model, a predicate and the value taken by the predicate (Cf. Figure 4). Furthermore, in order to implement our mapping representation proposal with MBDB of type 1, we apply the following rules:

- use RDF Schema as the meta-model for representing domain models. For instance, the triplet (e_1, Type, Entity) means that the concept e_1 is an RDF Class (called Entity in the E-A meta-model);

- extend RDF Schema with new concepts representing mapping constructors (*mLink*, *eLink*, *aLink*);
- extend RDF Schema with new properties representing parameters of L_m, L_e, L_a. For instance, "$rdf : m_s$" is used to represent the source model of a *mLink* while "$rdf : eL$" is used to identify a *mLink* representing the context in which a given *eLink* is created.

Putting all these previous rules together results in a database as illustrated in Figure 4. Unfortunately, it clearly appears that, this approach is not enough scalable because of the number of auto-join operations required on the underlying vertical table.

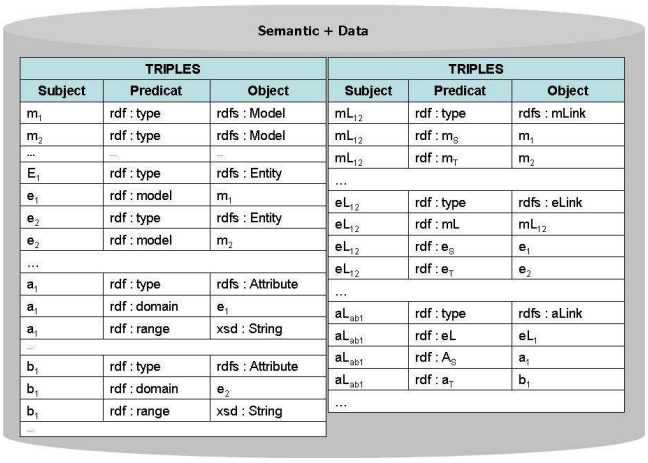

Figure 4. MBDB of type 1.

B. MBDB of type 2

MBDB of type 2 store separately model level data and the instance level data in two distinct schemes [6]. In classical databases, the system catalog part plays the role of the model level where models are stored as meta-data. The main problem with this representation comes from the fact that the meta-model provided by the DataBase Management System (DBMS) cannot be manipulated. Indeed, this meta-model is frozen and therefore, it cannot be evolved according to some particular requirements. For example, how do we represent mapping concepts like *mLink*, *eLink*, *aLink*, *etc.* (not available in this meta-model) using this type of database? However, as illustrated in Figure 5, one can use the semantic part of the database to represent mappings. Even if this approach provides a solution independent of a particular DBMS, the meta-model of the semantic part is also frozen and cannot be extended at runtime in order to provide new mapping constructors.

C. MBDB of type 3

MBDB of type 3 [17] propose to add another schema to MBDB of type 2. This schema stores the E-A meta-model in a reflexive meta-meta-model. Thus, for the meta-model, the

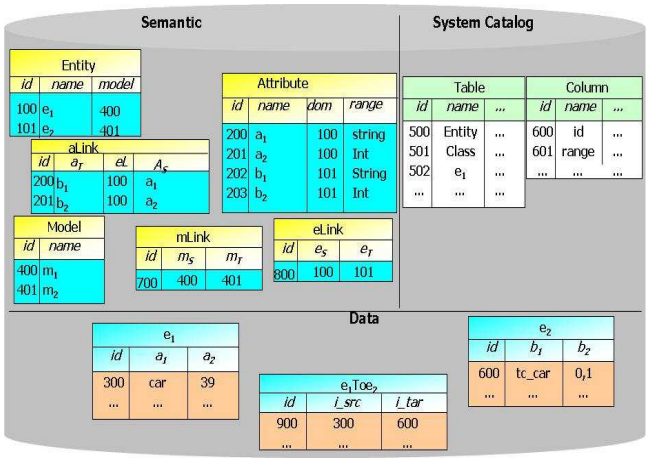

Figure 5. MBDB of type 2.

meta-meta-model plays the same role as the one played by the system catalog. Compared to MBDB of type 2, adding a meta-meta-model in MBDB of type 3 provides possibility to extend the meta-model. So, thanks to that meta-meta-model, MBDB of type 3 could be reused to reach our goal related to the representation of mapping concepts in database. Figure 6 illustrates the OntoBD [17] architecture as an example of a MBDB of type 3. OntoDB is based on 4 main parts:

- **the meta-base part:** it corresponds to the catalog system of databases. It contains system tables used to manage all the data contained in the database;
- **the data part:** it represents domain objects also called data;
- **the semantic part:** it contains models defining the semantics of data. More precisely, domain models of information system stored in the semantic part;
- **the meta-schema part:** the meta-schema part records the E-A meta-model.

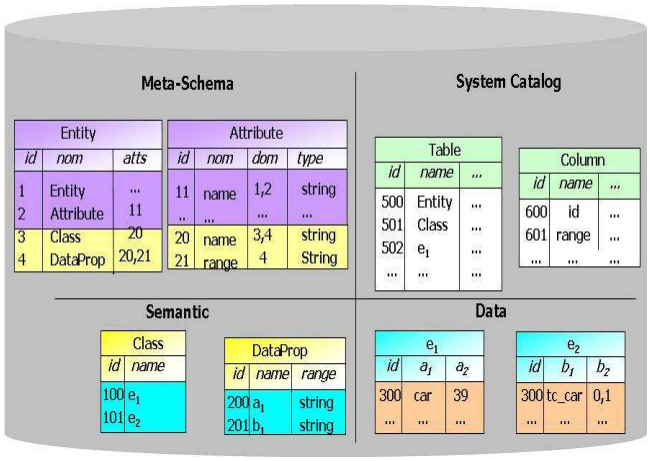

Figure 6. OntoDB: a MBDB of type 3.

The next Section presents how the approach led by MBDB of type 3 has been used to persist the graph of mappings.

V. Handling Mappings in a Database

In this section, we give some details about the implementation of our proposal. Thanks to its extension facility, MBDB of type 3 are therefore better suited to implement the mapping approach presented in this paper. Indeed, a type 3 MBDB provides enough flexibility for the extension of both the E-A meta-model and the mapping model. It is therefore possible, for example, to create new mapping constructors. The resulting database repository is illustrated in Figure 7 where labels purposes (instead of object identifier) are used only for more readability. Compared to Figure 6, we have extended the semantic part with mapping constructors. All these mapping constructors are created as instances of the meta-schema. That provides for creating dynamically new mapping constructors. The resulting infrastructure is obtained in 3 steps.

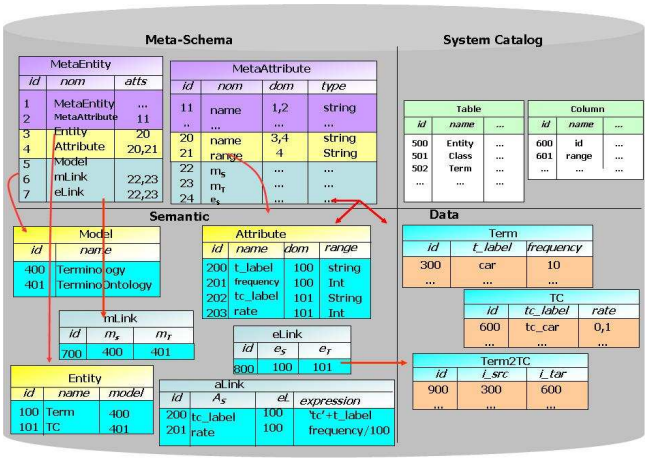

Figure 7. MBDB with mappings.

1) **Setup of the mapping management infrastructure.** This infrastructure consists in building a repository for mapping constructors. After creating tables in the meta-schema part, we populate them. For example, the statements of Table I insert new rows in tables *MetaEntity* and *MetaAttribute* tables of Figure 7. As a consequence, a physical repository for each constructor is automatically built in the semantic part. A table is created for each meta-entity with its attribute found in the meta-attribute table. At this level, we use classical SQL queries and the designer needs to know the meta-schema tables structure. The MQL language [21] provides the user with high level operators that hide implementation details available in such SQL queries. We do not give details of this language to keep this paper in reasonable size. Notice that, for readability purposes all statement examples provided in this paper use *''label"* as foreign key instead of URI.

Table I
Creation of the E-A meta-model

```
INSERT INTO MetaEntity (label) VALUES("mLink")
INSERT INTO MetaEntity (label) VALUES("eLink")
INSERT INTO MetaEntity (label) VALUES("aLink")
INSERT INTO MetaEntity (label) VALUES("Model")
...
INSERT INTO MetaAttribute (label, domain, type)
VALUES("m_s","mLink","Model")
...
INSERT INTO MetaAttribute (label, domain, type)
VALUES("m_t","mLink","Model")
...
INSERT INTO MetaAttribute (label, domain, type)
VALUES("α","mLink",INT)
...
```

2) **Model creation.** The models creation task consists in populating the *Entity*, *Attribute* and *Model* tables of the E-A meta-model (Cf. Figure 7). For each entity of the E-A meta-model, the physical structure (e_i tables) for storing data is created in the Data part.
3) **Mapping creation.** Creating a mapping consists in populating the *mLink*, *eLink*, *aLink* tables materializing mapping constructors (Cf. Figure 7). This explicit representation keeps traceability of mappings. However, it also keeps traceability between instances. Indeed, instance level mappings results from the instantiation of *eLink*. This instance mapping constructor is handled by creating, for each *eLink* a table ($e_1 TO e_2$ for example) storing those instances of the source entity that have been used to build the instance of the target entity.

An example of both of model and mapping creation tasks is given in Section VI, where the case study of modeling an ontology building process is detailed.

Notice that we have defined directed links from models and concepts to others. The reverse links can be easily built, if needed, using the same process. This capability will offer full traversals in the database from models to others. As a consequence, it is possible to trace the source concept used to produce target ones. The MQL language [21] offers high level operators for such traversals.

VI. Application

This section summarizes the use of the approach developed above in a particular context: building ontologies starting from text analysis.

A. Overview

The proposal of this paper has been applied in the DAFOE platform, a platform led by the ANR DaFOE4App project. This platform provides a stepwise methodology where the

 ISBN: 978-1-61208-196-0

first step is dedicated to linguistic analysis (called Terminology Step) in which users manage linguistic information (terms and relations between terms) extracted with natural language processing tools (NLP). After the linguistic analysis, the step for structuring linguistic information (called TerminoOntology step), in order to avoid possible ambiguity of terms, is performed. Finally, a formalization step (called Ontology Step) allows users to create *classes* and *properties* of the ontologies and to populate created classes. Each of these autonomous steps has been modeled as illustrated in Figure 8, 9 and 10 respectively. Notice that the main goal of the DaFOE platform is not to populate classes but to build ontologies that are intended to be exported into other systems that provide instance management facilities. Thus, instance management is out of the scope of the application domain.

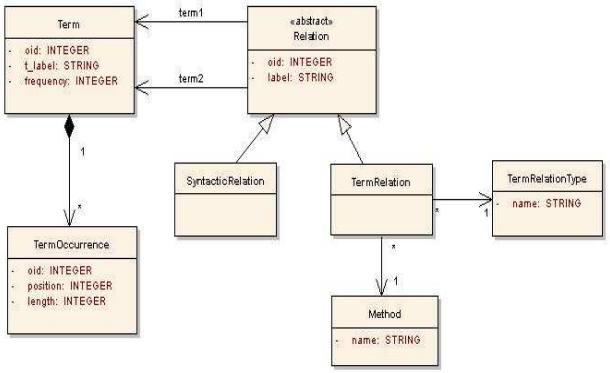

Figure 8. A simplified representation of the Terminology model.

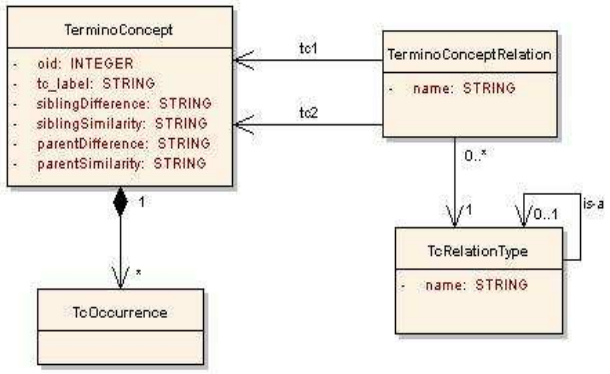

Figure 9. A simplified representation of the TerminoOntology model.

B. Setting up our approach

Applying our approach leads to the persistent infrastructure represented in Figure 11. It consists in writing correspondences between elements of the model of each step using mapping constructors. The developed approach in the

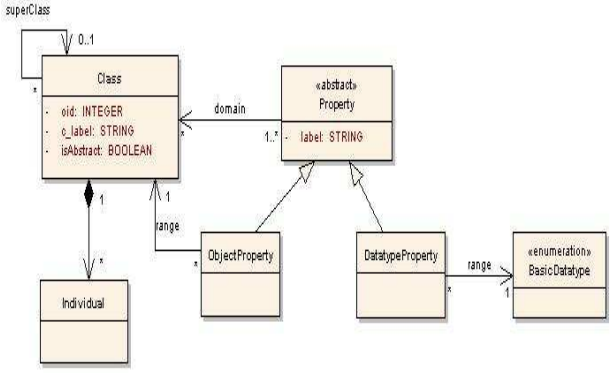

Figure 10. A simplified representation of the Ontology model.

DaFOE4App project has identified two bridges for switching between steps: a first one for producing termino-ontology concepts from texts and a second one for producing ontology concepts from termino-ontology concepts. According to our approach, bridge means the creation of *mLink*, *eLink* and *aLink* respectively after setting the models to be mapped. These two bridges are detailed below.

1) **Bridge 1. Terminology to TerminoOntology step.** Considering both *Terminology* and *TerminoOntology* steps through their models (Cf. Figure 8 and 9 respectively), a simplified mapping between these steps consists in:
mLink creation. The statement S_1 of Table II creates a *mLink* from the *Terminology* model to the *TerminoOntology* model. As a result, row 700 is inserted in table *mLink* (Figure 11).
eLink creation. The statement S_3 of Table III creates a *eLink* from the *Term* entity to the *TerminoConcept* entity to express that instances of the *Term* entity will be transformed as instances of the *TerminoConcept* entity. As a result of this statement, row 800 is inserted in the table eLink as illustrated in Figure 11. A *eLink* from the *TermRelation* entity of the *Terminology* model and the *TerminoConceptRelation* entity of the *TerminoOntology* model is also available.
aLink creation. The statement S_5 of table IV shows a *aLink* expresing that an instance of the *TerminoConcept* entity has the same *label* as its corresponding instance in the *Term* entity prefixed by *'tc_'*. Another *aLink* expresses that the rate of an instance of *TerminoConcept* entity equals to the frequency of corresponding instances in *Term* entity divided by 100. As a result, rows 200 and 201 are inserted in table *aLink* (Figure 11).
2) **Bridge 2. TerminoOntology to Ontology step.** For *TerminoOntology* and *Ontology* steps, a simplified mapping between their respective models consists in:
mLink creation. The statement S_2 of Table II creates a *mLink* from the *TerminoOntology* model to the *Ontology* model. As a result, row 701 is inserted in

Table II
STATEMENTS FOR MLINKS CREATION

Statement S_1-
INSERT INTO mLink (label, m_s, m_t, α_m)
VALUES("Terminology2TerminoOntology", "Terminology", "TerminoOntology", 0.8);

Statement S_2-
INSERT INTO mLink (label, m_s, m_t, α_m)
VALUES("TerminoOntology2Ontology", "TerminoOntology", "Ontology", 0.9);
...

Table III
STATEMENTS FOR ELINKS CREATION

Statement S_3-
INSERT INTO eLink (label, e_s, e_t, α_e, mL)
VALUES("Term2TerminoConcept", "Term", "TerminoConcept", 0.8, Terminology2TerminoOntology);

Statement S_4-
INSERT INTO eLink (label, e_s, e_t, α_e, mL)
VALUES("TerminoConcept2Class", "TerminoConcept", "Class", 0.8, "TerminoOntology2Ontology");
...

table *mLink* of Figure 11.

eLink creation. In the context of the previous created *mLink*, a *eLink* is created from the *TerminoConcept* entity to the *Class* entity to express that instances of the *TerminoConcept* entity will be transformed as instances of the *Class* entity. This *eLink* is created using statement S_4 represented Table III. A *eLink* from *TerminoConceptRelation* of the *TerminoOntology* model and the *Property* entity of the Ontology model is also available. As a result, row 801 is inserted in table *eLink* (Figure 11).

aLink creation. The statement S_7 of Table IV shows a *aLink* expressing that an instance of the *Class* entity has the same *label* as its corresponding instance in *TerminoConcept* entity. As result, rows 202 is inserted in table *aLink* (Figure 11).

Putting these mappings all together results in a stepwise design methodology for a database recording manipulated data and produced to build an ontology from texts according to the process defined in the DaFOE4App project.

VII. CONCLUSION

In this paper, we presented an approach for persisting mappings. Focusing on a specific type of databases i.e persistent meta-modeling systems, we proposed an extensible infrastructure for mapping management. Rather than freezing all the mapping constructors in a database, we have proposed to represent them as a model. This model is then used on the one hand to create new mapping constructors and on the other hand for the automatic generation of a persistent

Table IV
STATEMENTS FOR ALINKS CREATION

Statement S_5-
INSERT INTO aLink (label, A_s, a_t, α_a, φ, eL)
VALUES("TermLabel2TcLabel", ("term_label"), "tc_label", 0.8, "tc_label= "tc_" + term_label", "Term2TerminoConcept");

Statement S_6-
INSERT INTO aLink (label, A_s, a_t, α_a, φ, eL)
VALUES("TermLabel2TcLabel", ("frequency"), "rate", 0.8, "rate= "frequency/100", "Term2TerminoConcept");

Statement S_7-
INSERT INTO aLink (label, A_s, a_t, α_a, φ, eL)
VALUES("TcLabel2ClassLabel", ("tc_label"), "tc_label", 0.8, "class_label= tc_label", "TerminConcept2Class");
...

repository for mappings to ensure their traceability. As an assessment, our approach has been deployed and then implemented for the modeling process of building ontologies from texts in the context of the DaFOE4App project.

Furthermore, once models and mappings are created and models are populated with data, it would be interesting for example, to exploit these mappings when querying data. Indeed, because our mapping modeling is applied to models that represent the same real world domain, the domain related retrieving process needs to interpret mappings between models. Unfortunately, the resulting mapping graph as presented in this paper may be complex to manage with the classical SQL queries. For instance, as mappings are transitive thanks to mapping composition, one would want to use this capability to retrieve data transitively. Writing such a query could become complex. Thus, in continuity to this work, we have defined a SQL-like management and query language, namely MQL [21], that provides high level operators that makes easier querying data using mappings between models. This language, that hides implementation details regarding the database structure will be benchmarked in the context of engineering data retrieving where response time may be critical because of the huge amount of the underlying data. In this particular case, we are planing to improve performance analysis of the MQL query language.

ACKNOWLEDGMENT

The authors would like to thanks the partners of the ANR DaFOE4App project for their contribution.

REFERENCES

[1] Michael Genesereth, Arthur Keller and Oliver Duschka. Infomaster: an information integration system. In Proceedings of the ACM SIGMOD Record, pp. 539–542, 1997

[2] Pepijn Visser, Martin Beer, Trevor Bench-Capon, B. M. Diaz and Michael Shave. Resolving Ontological Heterogeneity in the KRAFT Project. In Proceedings of DEXA'99, pp. 668–677, 1999

 ISBN: 978-1-61208-196-0

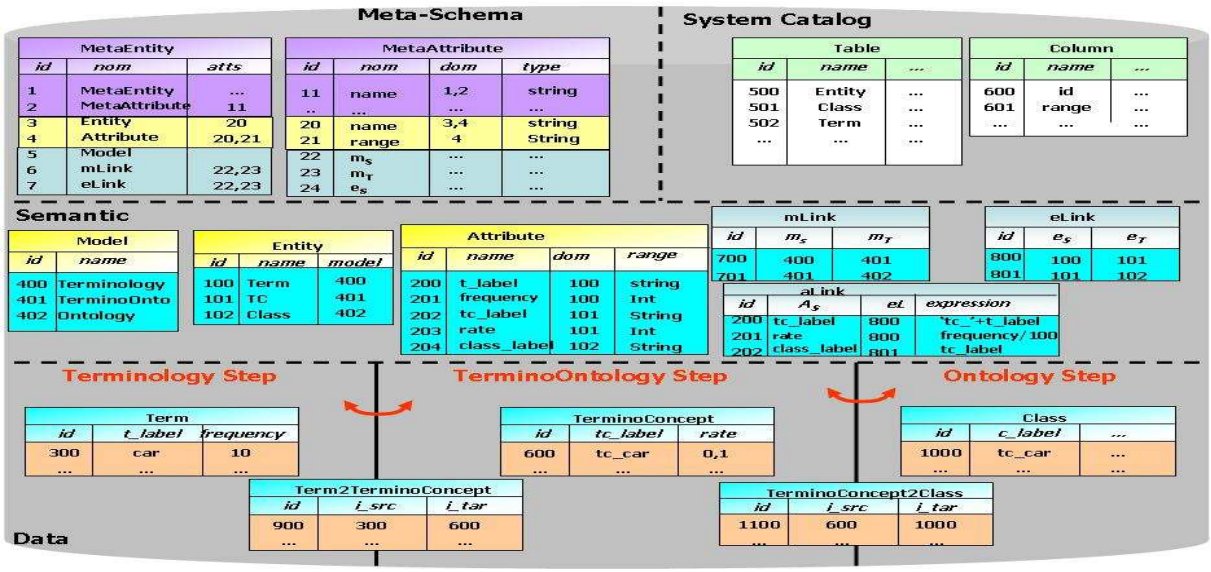

Figure 11. Mapping management in the DaFOEApp project.

[3] Renée Miller, Laura Haas and Mauricio Hernández. Schema Mapping as Query Discovery. In Proceedings of VLDB'00, pp. 77–88, 2000

[4] Yan Ling, Miller Renée, Haas Laura and Fagin Ronald. Data-driven understanding and refinement of schema mappings. In SIGMOD Record, pp. 485–496, 2001

[5] Rakesh Agrawal, Amit Somani and Yirong Xu. Storage and Querying of E-Commerce Data. In Proceedings of VLDB'01, pp. 149–158. Rome, Italy 2001

[6] Jeen Broekstra, Arjohn Kampman and Frank van Harmelen. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In Proceedings of ISWC'02, pp. 54-68, 2002

[7] Yannis Kalfoglou and Marco Schorlemmer. IF-Map: an ontology mapping method based on information flow theory. In JoDS'03, pp. 98–127, 2003

[8] Sergey Melnik, Erhard Rahm and Philip Bernstein. Developing metadata-intensive applications with Rondo. In Journal of Semantic Web, pp. 47–74, 2003

[9] Philip Bernstein. Applying Model Management to Classical Meta Data Problems. In SIGMOD Record, pp. 209-220. 2003

[10] Madhavan Jayant and Halevy Alon. Composing mappings among data sources. In Proceedings of VLDB'03, pp. 572–583, 2003

[11] S. Bowers and L. M. L. Delcambre. The Uni-Level Description: A uniform framework for representing information in multiple data models. In Proceedings of ER'03, pp. 45-58, 2003.

[12] Ian Horrocks, Peter Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof and Mike Dean. SWRL: a semantic web rule language combining OWL and RuleML, 2004, http://www.w3.org/Submission/SWRL/,2012-04-09 06:00:05 +0100

[13] Philip Bernstein, Sergey Melnik and Peter Mork. Interactive schema translation with instance-level mappings, In Proceedings of VLDB'05, pp. 1283–1286, 2005

[14] P. Papotti, R. Torlone. Heterogeneous data translation through XML conversion. J. Web Eng., pp. 189-204, 2005.

[15] Choi Namyoun, Song Il-Yeol and Han Hyoil. A survey on ontology mapping. In SIGMOD Record, pp. 34–41. New York, USA 2006

[16] Paolo Atzeni, Paolo Cappellari and Philip A. Bernstein. MIDST: model independent schema and data translation, In SIGMOD Record, pp. 1134-1136, 2007

[17] Hondjack Dehainsala, Guy Pierra and Ladjel Bellatreche. OntoDB: An ontology-based database for intensive applications. In Proceedings of DASFAA'07, pp. 497–506, 2007

[18] Jérôme Euzenat, Pavel Shvaiko. Ontology matching. In Springer-Verlag(eds.), 2007

[19] Alan Nash, Philip Bernstein and Sergey Melnik. Composition of Mappings Given by Embedded Dependencies. In ACM TODS'07, Volume 32 Issue 1, pp. 172–183, 2007

[20] Naouel Moha, Sagar Sen, Cyril Faucher, Olivier Barais and Jean-Marc Jézéquel: Evaluation of Kermeta for solving graph-based problems. In STTT'10 Journal, pp. 273–285, 2010

[21] Valéry Téguiak, Yamine Ait-Ameur, Éric Sardet and Ladjel Bellatreche. MQL: an extension of SQL for mappings manipulation. Internal report, LIAS/ISAE-ENSMA, 2011, http://www.lisi.ensma.fr/ftp/pub/documents/papers/2011/2011-Report-Teguiak_1.pdf, 2012-04-09 06:00:05 +0100

[22] Projet DaFOE4App. État de l'art et étude des besoins pour une plate-forme de construction d'ontologies. Livrable de projet, 2007, ftp://ftp.irit.fr/IRIT/IC3/Dafoe-livrableA.0.1.pdf, 2012-04-09 06:00:05 +0100

 ISBN: 978-1-61208-196-0

Cloud Computing Ontologies: A Systematic Review

Darko Androcec, Neven Vrcek, Jurica Seva
Faculty of Organization and Informatics
Varazdin, Croatia
e-mail: darko.androcec@foi.hr, neven.vrcek@foi.hr, jurica.seva@foi.hr

***Abstract*—The main objective of this study is to obtain a holistic view of Cloud Computing ontologies, their applications and focuses. The identification of primary studies in this systematic review is based on a pre-defined research protocol with a research question, inclusion and exclusion criteria and a search strategy. We summarize the selected studies into four main categories: Cloud resources and services description, Cloud security, Cloud interoperability and Cloud services discovery and selection. The analysis of the included studies indicates a number of challenges and topics for future research, including those specifically related to using ontologies to improve security and interoperability of Cloud Computing offerings.**

Keywords—Cloud Computing; ontology; systematic review; Cloud service

I. INTRODUCTION

Cloud Computing has become a new paradigm for the provision of computing infrastructure, platform or software as a service. Its main benefits are flexibility, pay-per-use model and significant cost reduction. Linthicum [1] concludes that the most comprehensive definition of the aforementioned paradigm is that provided by NIST. According to this definition, "Cloud computing is a pay-per-use model for enabling available, convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [1]. The minimum definition of Cloud Computing must contain scalability, pay-per-use utility model and virtualization [2]. Cloud Computing is primarily a new business paradigm [3] that enables on-demand access, elasticity, pay-per-use, connectivity, resource pooling and abstracted infrastructure [4].

There are a lot of Cloud Computing review papers in the current literature but to date no systematic review of Cloud Computing ontologies has been published. Therefore the primary aim of our research is to systematically select and review published work and provide an overview of Cloud Computing ontologies, their types, applications and focuses. The following research questions are stated: What are the main focus and application contexts of Cloud Computing ontologies covered in the scientific literature? What is the impact of the studies to scientific and professional community?

This paper proceeds as follows. Firstly, in Section 2, we describe the research method used in this review. Section 3 contains the overview data concerning the included studies. In the Section 4 we provide a detailed description of relevant reviewed papers and classify them into appropriate categories according to topics. Then, Section 5 presents a synthesis of this systematic review. Our conclusions are presented in the last section.

II. RESEARCH METHOD

Our research uses a systematic review method [5], which is a formalized process to assess and interpret all available research related to a specific research question. Guidelines that address specific problems of software engineering research are introduced in [5]. A systematic review has three main phases: planning the review, conducting the review and reporting the review.

We developed a review protocol in the planning phase. The background and the research question are specified in the introduction of our paper. Only full papers in English from peer-reviewed journals and conferences published from 2008 to 2011 were considered. Studies that are not related to the usage of ontology in Cloud Computing were excluded. In cases where several duplicated studies were found that existed in different versions, only the most complete version of the study was included. We focused on searching Google Scholar and the following electronic scientific databases: ScienceDirect, Current Contents, IEEE Xplore, SpringerLink and ISI Web of Science. These databases had been chosen since they provide the most important journals and conference proceedings covering Cloud Computing and ontology engineering. The following search term was used to find relevant studies: Cloud Computing AND ontology. Irrelevant studies were excluded based on the analysis of their titles, abstracts and keywords, whereas primary studies were obtained based on full text read. The search process was performed in November 2011, during which a total of 463 publications were identified. After filtering the publications list by reading titles, abstracts and keywords, full text reading of the articles that had not been excluded was performed to ensure that the content is related to our research question. Finally, 24 studies were identified as primary studies. Data extraction and synthesis were done by reading the full text of these 24 studies and extracting relevant data to Excel spreadsheets.

III. OVERVIEW DATA CONCERNING SELECTED STUDIES

In this section we describe the sources of publication and the citation status of the selected studies. Most of these

 ISBN: 978-1-61208-196-0

studies were published in conference proceedings, book chapters and journals. Table I. provides an overview of the distribution of the studies and the number of studies from a particular source type.

TABLE I. DISTRIBUTION PER PUBLICATION SOURCE TYPES

Source	Count
Conference proceedings	13
Book chapters	5
Journals	5
Workshops	1
Total	**24**

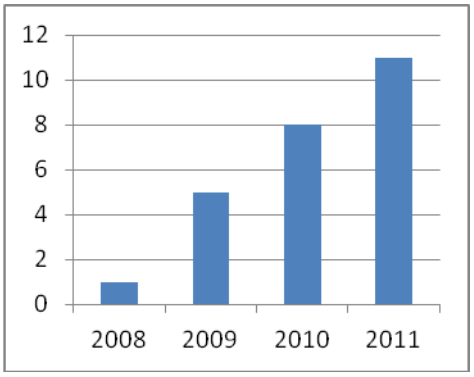

Figure 1. Number of studies by year of publication.

TABLE II. ACTIVE RESEARCH COMMUNITIES

Institution	Number of studies
Gwangju Institute of Science and Technology, South Korea	3
Wuhan Univ. of Technol., China	2
Victoria Univ. of Wellington, New Zealand	2

The obtained distribution is in line with expectations since Cloud Computing is a relatively new paradigm. The citation rates for the included studies were obtained from Google Scholar. The citation rates of the studies are quite low (most studies <10 citations). This result is in line with expectations since all the initially selected studies were published from 2008 to 2011, and 75% of those that were eventually included were published in last two years. According to Google Scholar data, the most cited publications from our selected set of studies are [6] with 191 citations and [7] with 38 citations. When the year of publication of the papers is concerned (Figure 1), we noticed an upward trend in the number of relevant publications about Cloud Computing ontology. In the selected set of studies we also looked for the authors' affiliation details in order to identify active research communities involved in work related to Cloud Computing ontologies (Table II.).

IV. RESULTS

Our examination of the selected studies was based on their similarities in terms of the main focus and application of Cloud Computing ontologies. We identified four main categories: Cloud resources and services description, Cloud security, Cloud interoperability and Cloud services discovery and selection.

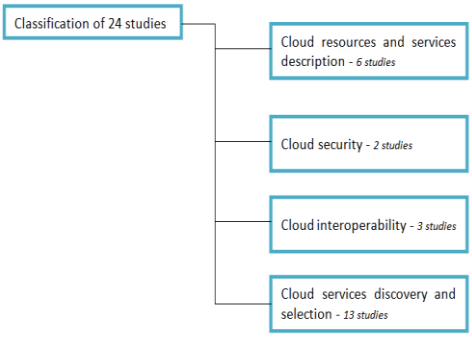

Figure 2. Classification of included studies.

Figure 2 illustrates these categories while also specifying their distribution across studies. The identified categories will be elaborated in the following subsections.

A. *Cloud Resources and Services Description*

The studies in this category use the Cloud ontologies to describe Cloud resources and services, classify the current services and pricing models or define new types of Cloud services.

One of the first attempts to establish a detailed Cloud ontology was presented in [6]. In that paper the authors proposed an ontology which demonstrates a dissection of the Cloud into five main layers: applications, software environments, software infrastructure, software kernel and hardware. Each layer encompasses one or more Cloud services which belong to the same layer if they have equivalent levels of abstraction. The authors of the ontology in [6] also discussed each layer's strengths, limitations and their dependency on preceding computing concepts.

Weinhardt et al. [7] proposed a Cloud business ontology model to classify current Cloud services and pricing models. Their ontology consists of three layers: infrastructure, platform and application as a service. Cloud users and providers can use it to map the existing Cloud services and set pricing schemes.

Bohm et al. [8] suggested various definitions of Cloud Computing, its billing models and a systematic description of its major actors and value network. They also reviewed the definitions, models and ontologies from the existing literature.

The concept of ontology as a service was proposed in [9]. Ontology as a service (OaaS) is a service where Cloud vendors provide the application and infrastructure to tailor the source ontology to the users' requirements. The authors of the study reported in [9] elaborated ontology extraction and sub-ontology merging process.

 ISBN: 978-1-61208-196-0

Sheng-Yuan et al. [10] proposed an ontology-supported ubiquitous interface agent and described the interaction with the backend information agent system in Cloud Computing.

A formal catalog representation of Cloud services was proposed in [11]. In this paper, Deng et al. introduced a range of Cloud services and their processes modeled by means of ontological representation.

B. Cloud Security

The studies in this category use ontologies to describe and improve Cloud security.

Takahashi, Kadobayashi and Fujiwara [12] built the ontology for cyber security operational information and applied it to Cloud Computing. Since the essential changes of cyber security information in Cloud Computing are data-asset decoupling, composition of multiple resources and external resource usage, they included data provenance and resource dependency information into their ontology.

The architecture of a deployed service in the Cloud Computing environment used for malware detection was presented in [13]. The authors used the ontology for malware and intrusion detection that represents the signatures for known and novel attacks as well as an ontological model for reaction rules creating the prevention system.

C. Cloud Interoperability

One of the biggest obstacles of Cloud Computing is provider lock-in that can be solved by means of interoperable Cloud services. The studies in this category show how to use ontologies to achieve interoperability among different Cloud providers and their services.

The FP7 mOSAIC project is aimed at creating and exploiting an open-source Cloud application programming interface and a platform for developing multi-Cloud oriented applications. The mOSAIC Cloud Ontology is described in [14]. The concepts in this ontology were identified by analyzing standards and proposals from literature and will be used for semantic retrieval and composition of Cloud services.

Bernstein and Vij [15] presented the InterCloud Directories and Exchanges mediator to enable connectivity and collaboration among Cloud vendors. Their ontology of Cloud Computing resources intended for facilitating work with heterogeneous providers of Cloud Computing services was defined using RDF.

A method for semantic interoperability aggregation in requirements refinement and a metric framework for calculating semantic interoperability capability based on ontologies are proposed in [16]. This methodology can provide a semantic representation mechanism for refining users' requirements in the Cloud Computing environment.

D. Cloud Services Discovery and Selection

This category consists of the studies that use ontologies to discover and select the best Cloud service alternative.

Along with a lack of standard definitions of resource requirements, managing Cloud resources implies resource information management issues and resource allocation compatibility problems. As a solution to the aforementioned problems, Yoo et al. [17] proposed a resource virtualization method using ontology.

Wang and Li [18] introduced the basic principles of the HCCloud (Heterogeneous Computing Cloud) design. The HCCloud is their architecture for the deployment and management of distributed applications in the Cloud where users can access services based on their requirements regardless of where the services are hosted. The resource selection mechanism starts with the user's requirements, calculates the similarity between resources and a particular candidate in the database of the Cloud resource ontology and ranks candidate resources accordingly.

Han and Sim [19] presented a Cloud service discovery system that uses Cloud ontology to determine the similarities between and among services. It is an agent-based discovery system that enables reasoning about the relations of Cloud services using three types of similarity reasoning to assist users in searching available Cloud services more efficiently. Their Cloud ontology consists of concepts of different Cloud services for IaaS (infrastructure as a service), PaaS (platform as a service) and SaaS (software as a service).

Zhou, Yang and Hugill [20] introduced a novel approach to reengineering enterprise software for Cloud Computing. They proposed the ontology for enterprise software and then partitioned it to decompose enterprise software from legacy system into potential service candidates during migration to the Cloud Computing environment.

In their study, Kang and Sim [21] proposed a Cloud ontology to semantically define the relationship among different Cloud services. The similarity among Cloud services is determined using concept similarity reasoning, object property similarity reasoning and data type property similarity reasoning. They also presented their own Cloud service search engine that uses the defined ontology. Users can specify functional, technical and cost requirements, and the search engine returns the list of relevant Cloud services. Sim [22] proposed the development of software agents for Cloud service discovery, service negotiation and service composition. An agent-based search engine for Cloud service discovery consists of a service discovery agent that uses the defined Cloud ontology and multiple Cloud crawlers. In the aforementioned study, Sim [22] also devised a complex Cloud negotiation mechanism and adopted a focus selection contract net protocol and service capability tables to automate Cloud service composition.

In [23] Dastjerdi, Tabatabaei and Buyya presented an architecture using ontology-based discovery to provide QoS aware deployment of virtual appliances on Cloud IaaS services. Virtual appliances are sets of virtual machines including operating systems, pre-configured and ready-to-run applications and embedded needed components. The proposed architecture can help users to deploy their virtual appliances on the most appropriate IaaS providers based on their definition of QoS requirements.

Yun et al. [24] introduced a tele-management system using a Cloud Computing platform for ubiquitous city. Ontology was used for context aware intelligence processing.

 ISBN: 978-1-61208-196-0

Semantic services discovery from the available Cloud providers is described in [25]. In this study, Wang et al. extended the Support Vector Machine (SVM)-based text clustering technique and proposed an iterative process to incrementally enrich domain ontology.

Ma, Schewe and Wang [26] extended their ASM-based model of Abstract State Services (ASSs) to a Cloud Computing model. They proposed a formalism of Clouds by federations of services and a description of Cloud services in form of ontology. These descriptions contain a technical description of services (types, pre- and post-conditions) and keywords which describe the application area and functionality of the annotated service.

The architecture to provide a semantic service for document management in Cloud Computing implemented by using techniques of web service and ontology was proposed in a book chapter by Wei and Junpeng [27].

In their paper, Ma, Jang, and Lee [28] proposed an ontology-based job allocation algorithm for a resource management system in Cloud Computing. They considered virtual machines as Cloud resources and built a Cloud ontology based on Cloud resource information and agreed SLAs (Service Level Agreements). The aforementioned ontology can be used to process complicated queries for searching Cloud resources. Its experimental results have verified that the ontology-based resource management system improves the efficiency of resource management for Cloud Computing when compared to the existing resource management algorithms.

The approach to developing semantic Cloud services which are annotated based on shared ontology was proposed in a book chapter by Chen, Bai, and Liu [29], along with a description of the usage of these annotations for semantics-based discovery of relevant Cloud services.

V. DISCUSSIONS

Research papers regarding Cloud Computing ontologies vary in terminology, descriptions and involved activities, but they also share a lot in common (focus, goal, application etc.). Our examination of the selected studies was based on their similarities in terms of the main focus and application of Cloud Computing ontologies. We divided them in the four categories: Cloud resources and services description, Cloud security, Cloud interoperability and Cloud services discovery and selection.

Since the bias in our selection of the studies to be included presented the main threat to validity of our research, we used a research protocol to define the research question, inclusion and exclusion criteria and our search strategy. The review protocol was prepared by the first author and reviewed by the other two authors.

Our review reveals that Cloud Computing ontologies are predominantly applied in the discovery and selection of the best service alternative in accordance with users' needs and the description of Cloud resources and services (80% of the relevant identified studies deal with these issues). The identified categories of themes provide an overview of Cloud Computing ontologies research as well as a basis for discovering possibilities for improvement in research and practice. Table III specifies the main achievements, limitations and challenges of these categories in the existing literature.

TABLE III. CURRENT STATE OF THE CLOUD COMPUTING ONTOLOGIES

Category	Achievements	Limitations and challenges
Cloud resources and services description	- general Cloud business ontology - dissection of the Cloud into layers - classification of the current Cloud services	- Cloud market is very dynamic, new Cloud services often emerge - detailed ontology of the Cloud resources and services is missing
Cloud security	- the ontology for cyber security operational information in Cloud Computing - the ontology for malware and intrusion detection deployed in the Cloud	- data and assets can be decoupled and manipulated independently in the Cloud Computing - external resources usage and composition of multiple resources - privacy and data security risks
Cloud interoperability	- the mOSAIC Cloud ontology that uses concepts from standards and proposals from literature to improve interoperability - ontology based Cloud Computing resources catalog to federate or interoperate resources	- lack of interoperability among Cloud Computing services - common Cloud API or an orchestration platform is currently not available (some on-going FP7 research projects such as mOSAIC plan to develop Cloud interoperability platforms) - detailed ontology focused on Cloud API resources and operations does not exists
Cloud services discovery and selection	- multiple Cloud services discovery and selection approaches were proposed	- user-friendly application for Cloud services discovery and selection is still missing

The analysis of the selected studies indicates a number of challenges and topics for future research based on identified limitations and challenges in the existing literature. The most promising area of future research is the use of ontologies to improve security and interoperability of Cloud Computing offerings, because the main obstacles of the Cloud Computing paradigm are provider lock-in and security/privacy issues. For example, interesting research challenge is using an ontology-based approach as a basis for creation of the mechanism to automatically determine and solve interoperability problems among two or more Cloud Computing services provided by different vendors. Ontologies can also be useful tool to annotate sensitivity of data and portions of data stored in Cloud services. Existing Cloud Computing ontologies are mostly general and detailed ontologies of each Cloud Computing layer (software as a

 ISBN: 978-1-61208-196-0

service, platform as a service, infrastructure as a service) are still missing.

Besides for researchers, this systematic review might have implications for practitioners. They can use this review as a source in searching for relevant approaches for Cloud services discovery and selection. The identified limitations of the current literature can inspire programmers and Cloud users (e.g., development of user-friendly application for Cloud services discovery and selection).

VI. CONCLUSION

Cloud Computing is a new paradigm for the provision of computing infrastructure, platform or software as a service. The main objective of the systematic review presented in our paper is to obtain a holistic perspective of Cloud Computing ontologies, their applications and focuses. We identified 24 primary studies using the systematic review methodology described in [5].

The main focus and application contexts of Cloud Computing ontologies covered in the scientific literature are: Cloud resources and services description, Cloud security, Cloud interoperability and Cloud services discovery and selection. The studies in the first category use the Cloud ontologies to describe Cloud resources and services, classify the current services and pricing models or define new types of Cloud services. The Cloud security category shows how to use ontologies to describe and improve Cloud security. Cloud interoperability consists of the studies that use ontologies to achieve interoperability among different Cloud providers and their services. Finally, the fourth category comprises the studies that focus on discovery and selection of the best Cloud service alternative using the previously defined ontology.

The analysis of the selected studies indicates a number of challenges and topics for future research, including those specifically related to using ontologies to improve security and interoperability of Cloud Computing offerings. The main obstacles of the Cloud Computing paradigm are provider lock-in and security/privacy issues, which researchers can overcome by using an ontology-based approach. Practitioners can use our work to find existing approaches or develop new applications inspired by identified limitations of the currently available solutions.

REFERENCES

[1] D. S. Linthicum, Cloud Computing and SOA Convergence in Your Enterprise: a step-by-step guide, 1st ed., Addison-Wesley, 2009.

[2] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A Break in the Clouds: Towards a Cloud Definition", ACM SIGCOMM Computer Communication Review, vol. 39, pp. 50-55, January 2009.

[3] A. Rosenthal, P. Mork, M. H. Li, J. Stanford, D. Koester, and P. Reynolds, "Cloud computing: A new business paradigm for biomedical information sharing", Journal of Biomedical Informatics, vol. 43, pp. 342-353, April 2010.

[4] D. Durkee, "Why Cloud Computing Will Never Be Free", Communications of the ACM, vol. 8, pp. 62-69, April 2010.

[5] B. Kitchenham, "Procedures for Performing Systematic Reviews", Technical Report 0400011T, Keele University, July 2004.

[6] L. Youseff, M. Butrico, and D. Da Silva, "Toward a Unified Ontology of Cloud Computing", GCE '08 Grid Computing Environments Workshop, pp. 1-10, November 2008.

[7] C. Weinhardt, A. Anandasivam, B. Blau, and J. Stosser, "Business Models in the Service World", IT Professional, vol. 11, pp. 28-33, March-April 2009.

[8] M. Bohm, S. Leimeister, C. Riedl, and H. Krcmar, "Cloud Computing – Outsourcing 2.0 or a new Business Model for IT Provisioning?" in Application Management, F. Keuper, C. Oecking, and A. Degenhardt, Eds. Gabler, 2011, pp. 31-56.

[9] A. Flahive, D. Taniar, and W. Rahayu, "Ontology as a Service (OaaS): a case for sub-ontology merging on the cloud", The Journal of Supercomputing, pp. 1-32, October 2011.

[10] Y. Sheng-Yuan, H. Chun-Liang, and L. Dong-Liang, "An ontology-supported ubiquitous interface agent for cloud computing — Example on Bluetooth wireless technique with Java programming", Proceedings of the Ninth International Conference on Machine Learning and Cybernetics, pp. 2971-2978, July 2010.

[11] Y. Deng, M. R. Head, A. Kochut, J. Munson, A. Sailer, and H. Shaikh, "Introducing Semantics to Cloud Services Catalogs", 2011 IEEE International Conference on Services Computing, pp. 24-31, July 2011.

[12] T. Takahashi, Y. Kadobayashi, and H. Fujiwara, "Ontological approach toward cybersecurity in cloud computing", Proceedings of the 3rd international conference on Security of information and networks, pp. 100-109, September, 2010.

[13] C. A. Martinez, G. I. Echeverri, and A. G. C. Sanz, "Malware detection based on Cloud Computing integrating Intrusion Ontology representation", 2010 IEEE Latin-American Conference on Communications (LATINCOM), pp. 1-6, September, 2010.

[14] F. Moscato, R. Aversa, B. Di Martino, T.-F. Fortis, and V. Munteanu, "An Analysis of mOSAIC ontology for Cloud Resources annotation", Proceedings of the Federated Conference on Computer Science and Information Systems, pp. 983-990, 2011.

[15] D. Bernstein, and D. Vij, "Intercloud Directory and Exchange Protocol Detail Using XMPP and RDF", 2010 6th World Congress on Services (SERVICES-1), pp. 431-438, July, 2010.

[16] K.-Q. He, J. Wang, and P. Liang, "Semantic Interoperability Aggregation in Service Requirements Refinement", Journal of Computer Science and Technology, vol. 25, pp. 1103-1117, November 2010.

[17] H. Yoo, C. Hur, S. Kim, and Y. Kim, "An Ontology-Based Resource Selection Service on Science Cloud" in Grid and Distributed Computing, D. Slezak, T. Kim, S. S. Yau, O. Gervasi, and B. Kang, Springer Berlin Heidelberg, 2009, pp. 221-228.

[18] B. X. N. Wang and C. Li, "A Cloud Computing Infrastructure on Heterogeneous Computing Resources", Journal of Computers, vol. 6, pp. 1789-1796, 2011.

[19] T. Han, K. M. Sim, "An Ontology-enhanced Cloud Service Discovery System", Proceedings of the International MultiConference of Engineers and Computer Scientists 2010 Vol I, March 2010.

[20] H. Zhou, H. Yang, and A. Hugill, "An Ontology-Based Approach to Reengineering Enterprise Software for Cloud Computing", 2010 IEEE 34th Annual Computer Software and Applications Conference (COMPSAC), pp. 383-388, July 2010.

 ISBN: 978-1-61208-196-0

[21] J. Kang and K. M. Sim, "Ontology and search engine for cloud computing system", 2011 International Conference on System Science and Engineering (ICSSE), pp. 276-281, June 2011.

[22] K. M. Sim, "Agent-based Cloud Computing", IEEE Transactions on Services Computing, vol. PP, pp. 1-13, October 2011.

[23] A. V. Dastjerdi, S. G. H. Tabatabaei, and R. Buyya, "An Effective Architecture for Automated Appliance Management System Applying Ontology-Based Cloud Discovery", 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, pp. 104-112, May 2010.

[24] C. H. Yun, H. Han, H. S. Jung, H. Y. Yeom, and Y. W. Lee, "Intelligent Management of Remote Facilities through a Ubiquitous Cloud Middleware", 2009 IEEE International Conference on Cloud Computing, pp. 65-71, September 2009.

[25] J. Wang, J. Zhang, P. C. K. Hung, Z. Li, J. Liu, K. He, "Leveraging Fragmental Semantic Data to Enhance Services Discovery", 2011 IEEE International Conference on High Performance Computing and Communications, pp. 687-694, September 2011.

[26] H. Ma, K.-D. Schewe, and Q. Wang, "An Abstract Model for Service Provision, Search and Composition", APSCC 2009. Services Computing Conference, pp. 95-102, December 2009.

[27] Y. Wei and C. Junpeng, "Semantic Service in Cloud Computing" in Advances in Information Technology and Education, T. Honghua and Z. Mark, Springer Berlin Heidelberg, 2011, pp. 156-160.

[28] Y. B. Ma, S. H. Jang, and J. S. Lee, "Ontology-Based Resource Management for Cloud Computing" in Intelligent Information and Database Systems, N. Nguyen, C.-G. Kim, and A. Janiak, Springer Berlin Heidelberg, 2011, pp. 343-352.

[29] F. Chen, X. Bai, and B. Liu, "Efficient Service Discovery for Cloud Computing Environments" in Advanced Research on Computer Science and Information Engineering, G. Shen and X. Huang, Springer Berlin Heidelberg, 2011, pp. 443-448.

 ISBN: 978-1-61208-196-0

An Ontology-based Representation of the Google+ API

Konstantinos Togias
School of Science and Technology
Hellenic Open University
Patras, Greece
ktogias@eap.gr

Achilles Kameas
School of Science and Technology
Hellenic Open University
Patras, Greece
kameas@eap.gr

***Abstract*—Social Networking Services (SNS) provide users with functionalities for developing their on line social networks, connecting with other users, sharing and consuming content. While most of popular SNS provide open Web 2.0 APIs, they remain disconnected from each other thus fragmenting user's data, social network and content. Semantic social web technologies such as public vocabularies and ontologies can be used for bridging the semantic gap between different SNS. Ontology-based representations of SNS APIs can help developers share knowledge about SNS APIs and can be used for linking APIs with public Social Semantic Web ontologies and vocabularies and for enabling automatic ontology-based service composition. In this paper, we study the API of Google+ SNS and create an ontology based representation of its structural and functional properties. The proposed ontology describes valuable structural and functional details of the API, in a machine processable format useful for understanding the API and appropriate for integrating into ontology based Mashups.**

Keywords*—*Semantics; Social Networking System; Web Mashup; Social Semantic Web.

I. INTRODUCTION

Social Networking Services (SNS) are web applications that allow users create and maintain an online network of close friends or business associates [1]. Typical examples of SNS are Facebook, Myspace, Twitter and the most recent Google+. While SNS have much common functionality they do not usually interoperate and therefore require the user to re-enter her profile and redefine her connections when registering for each service [1]. Also content shared in one SNS is not available to users of other SNS.

Web 2.0 is a widely-used term characterizing the modern web made popular by Tim O' Reilly. Web 2.0 is the network as platform, spanning all connected devices [2]. Web 2.0 applications consume data and services from other applications and enable the reuse and remixing of their own data and services through public Application Programming Interfaces (APIs). Experienced users and programmers use those APIs for creating new integrated web applications, popular known as mashups [3] that combine different data sources and APIs into an integrated end user experience.

Most SNS participate to the Web 2.0 ecosystem by providing their own open APIs. Those APIs provide a first step towards bringing down the walls between SNS. Nevertheless, every SNS use its own terms for defining concepts and representing resources, while it interconnects the resources it provides in its own custom way. Thus common concepts, resources and functionalities are described and provided in different ways in each SNS API.

The Social Semantic Web is the vision of a Web where all of the different collaborative systems and SNS, are connected together through the addition of semantics, allowing people to traverse across these different types of systems, reusing and porting their data between systems as required [1]. Social Semantic Web uses Semantic Web technologies in order to describe in an interoperable way users' profiles, social connections and content creation, sharing and tagging accross different SNS and Sites in the Web.

Ontologies have become the means of choice for knowledge representation in recent years as they provide common format and understanding on domain concepts, while being machine processable [4]. Hendler [5] supports that the ontology languages of the Semantic Web can lead directly to more powerful agent-based approaches. Furthermore, ontologies are used for representing and sharing knowledge about structural and behavioral properties of software [6], for building context-aware and pervasive applications [7], and for achieving context-aware web service discovery and automatic service composition in Service Oriented Software (SOA) [8][9].

Web 2.0 APIs, SOA technologies and Social Semantic Web approaches provide the basic means for bridging the gap between today's SNS and for unifying users' data, social networks and interactions scattered across various SNS. However, today's SNS APIs lack semantic representations, while existing Semantic Web Ontologies and Vocabularies do not provide links with the API resources and methods used for actually accessing and manipulating users, social networks and content within SNS. Thus, Social Semantic Web approaches, SOA service discovery and service composition techniques cannot be directly applied on them. Moreover, combining multiple SNS APIs for building Mashups require for developers to search, read and combine

 ISBN: 978-1-61208-196-0

information from miscellaneous documentation pages scattered across the web. Using Ontologies for describing those APIs can help addressing those shortcomings by providing common, machine processable representations suitable for both sharing knowledge between developers and achieving automatic service discovery and service composition in SNS Mashups.

In this work, we study the API provided by Google+, one of the most popular and most recent SNS and we propose an ontology based representation of its structural and functional characteristics. Our ontology is compatible with the technologies of the Semantic Web and aims to be useful for sharing knowledge about the Google+ API between developers of Web 2.0 Mashups and as part of future inter-operable ontology based social networking software.

The paper is organized as follows. Section 2 briefly reviews related work in the areas of Social Semantic Web, Web 2.0 Mashups, ontology representation of software properties, and Service Oriented Architectures (SOA). Section 3 presents the proposed ontology-based representation of Google+ API. Section 4 discusses the representation and visualization of the ontology, while Section 5 presents test queries run on the proposed ontology. Section 6 presents conclusions and suggestions for future work.

II. RELATED WORK

Berslin and Decker [10] and Berslin et al. [1] propose the use of Semantic Web mechanisms in order to bridge the isolation and fragmentation of todays SNS. Public vocabularies and ontologies can be used to give meaning to Social Networks and interconnect social websites. The FOAF ontology [11] provides a formal, machine readable representation of user profiles and friendship networks. The SIOC Core Ontology provides the main concepts and properties required to describe information from online communities (e.g., message boards, wikis, weblogs, etc.) on the Semantic Web [12]. The SIOC and FOAF ontologies are used in combination with metadata vocabularies like Dublin Core [13] and SKOS [14] for describing user-generated content on the Social Web. Zhou and Wu in [15] propose an ontology representing SNSs based on FOAF in order to resolve the problem of social data inconsistency and to achieve interoperability among multiple social network services. Their ontology defines some of the basic attributes of a generic SNS API, such as operations, arguments and responses, combined with some user profile and contact attributes borrowed by FOAF ontology, but it does not provide any structural description of the resources that can be accessed through it.

While the above approaches describe generic concepts about people, content and SNS, they do not describe the functional and structural aspects of specific SNS APIs necessary for building ontology based Mashups. Specialized ontology-based representations of the APIs of existing SNS could be used in combination with the above ontologies and vocabularies in order to bridge abstract concepts with specific resources and actions provided by each API.

Hartmann et al. [16], Zang et al. [3], and Wong and Hong [17] investigate how users with programming skills and programmers build Mashups that make use of public APIs provided by popular web 2.0 services. Most of those users are self-taught and depend on the documentation of the API they want to use. Some of the most common problems encountered when creating Mashups is the complexity of communicating data from one server to another and the lack of proper tutorials and examples in the documentation [3].

Dietrich and Elgar [6] propose that knowledge about structural and behavioural properties of software can be shared across the software engineering community in the form of design patterns expressed in the web ontology language (OWL). The inherent advantage of their approach is that it yields descriptions that are machine processable, but also suitable for a community to share knowledge taking advantage of the decentralized infrastructure of the Internet [6]. Ontology-based representations of SNS APIs can bring the same advantages for the community of Mashup developers.

Kurkovsky, Strimple and Nuzzi in [18] discuss the possibility of convergence of Web 2.0 and SOA, while Xiao et al [8][9] propose the use of ontologies for context-aware web service discovery and automatic service composition. The availability of ontology-based representations of SNS APIs can also help to build software able to automatically compose services that integrate data and functionality from SNS.

Our work takes into consideration the above works by providing an ontology-based representation of Google+ API, compatible with Semantic Web mechanisms and ontology based service discovery and composition approaches that can be used for knowledge sharing and as part of ontology-based Mashups that integrate Google+ functionality and data.

III. AN ONTOLOGY BASED REPRESENTATION OF THE GOOGLE+ API

Google+ is an SNS operated by Google Inc. The service was launched on June 28, 2011in an invite-only testing phase and went public on September 20, 2011. Google+ integrates longer existent Google social services such as Google Profiles and Google Buz, and introduces new features identified as Circles for organizing users' connections into custom groups and Hangouts for group video chat [19]. Google+ became popular form the very first days of its testing phase and in October 2011 reached 40 million users [19].

On September 15, 2011 Google released its first open API for Google+ [20]. Google+ API follows a RESTful API design, meaning that applications use standard HTTP methods to retrieve and manipulate Google+ resources. The API is currently read only, thus it provides only methods for retrieving and searching resources through the HTTP GET method. The API can be used free of charge, with applications being limited to a courtesy usage quota. Developers can request a higher limit for their applications for a fee. Many API calls require that the user of the application is granted permission to access their data. Google uses the OAuth 2.0 [21] protocol to allow authorized

 ISBN: 978-1-61208-196-0

applications to access user data. Resources in the Google+ API are represented using JSON [22] data formats. It also supports pagination and partial responses for sending only requested fields instead of the full representation of a resource. The API currently provides read only access to three main types of resources named "Persons", "Activities" and "Comments". Person resources represent Google+ API users, Activities resources stand for content shared by users and Comments resources are content posted as a replies to Activities. Google also provides free client libraries for various programming languages including Python, PHP, Ruby, Javascript and Java.

In order to describe the structural and be properties of Google+ API in a way that can be shared among Software Developers and automatically interpreted by software components, we have introduced an ontology based representation of its main characteristics, resources and actions. For designing our ontology we followed the steps described by Noy and MacGuinness in [23]:

A. *Specification of the domain and the purpose of the ontology*

The domain of the ontology is the Google+ API and more specifically its structural and functional properties. That is, the data interchange and auhentication methods it uses, the types of entities that can be accessed through it and their attributes, and the actions that can be performed through it on these entities. The purpose of the ontology is dual: On the one hand the ontology is playing the role of a shareable and browsable knowledge base for researchers and programmers that want to develop applications and Mashups that integrate Google+ data and functionality, while on the other hand, because of its machine interpretable format, it may be used for building inter-operable ontology based social networking software. Such software will be programmed in a higher level of abstraction and use automatic reasoning on ontologies for providing integration with Google+.

B. *Enumeration of important terms in the ontology*

For enumerating the important terms in the ontology we studied the Google+ API documentation available online [24]. Through the documentation pages we identified references to key terms such as "Authorization Protocol", "Value Type", and "Parameter". Other terms like "Action Type", "Field" and "Resource Type" where produced through generalization of the descriptions provided by the documentation.

C. *Considering reusing existing ontologies*

The FOAF ontology describes user profiles and friendship networks, while the SIOC Core Ontology provides the main concepts and properties required to describe information from online communities. Both describe concepts relative to SNS at a high level of abstraction. For describing Google+ API, we needed lower level concepts such us urls, resources and methods that are not provided by those ontologies. The ontology proposed by Zhou and Wu in [15] defines some of the basic attributes of a generic SNS API, such as operations, arguments and responses, without describing them further or defining relations between them and the resources accessed through them. Thus, there was no important gain in reusing concepts from these ontologies for building our ontology. However, we would like to connect our ontology with ontologies like those in the future.

D. *Specification of the classes of the ontology and class hierarchy*

The classes of an ontology describe the main concepts of its domain. Since the domain of our ontology is the Google+ API, its classes will represent the concepts that are necessary for describing its structural and functional properties. Based on the documentation of the API we defined the following classes: *API* (an API), *APIType* (an API type), *DataFormat* (a data interchange format), *AuthorizationProtocol* (an authorization protocol used to access the API), *ResourceType* (a resource type provided by the API), *Field* (a field of a resource; fields represent attributes of a resource), *Action* (an action that can be performed to Resource), *ActionType* (an action type), *Parameter* (a parameter of an action), *ValueType* (the type of the value contained in a field or a parameter) and *DataStructure* (the type of the data structure contained in a field or a parameter).

The domain of the ontology is found to be flat in terms of generalization. The concepts we used for describing the API are considered to belong all at the same level of generality. Thus the classes of the ontology are disjoint with each other and no subclasses where defined.

E. *Specification of the properties of the classes and property value types*

The properties of a class represent the characteristics of the corresponding concept. The API is described in terms of its type, the format in which it exchanges data, the authorization protocol it supports and the resource types it provides. It has a name property, a base url used to build http request urls, and a documentation url where developers can access the official documentation of the API. The API provides some types of Resources. A Resource type has a name and may have a specific documentation url. A Resource type consists of Fields and can provide Actions. A Field is characterized by the type of its value (e.g. String, Integer, or Resource) and the type of its data structure (a single value or a structure like a list). An Action can have required or optional parameters and be performed by an HTTP/1.1 GET, PUT, POST or DELETE method. The Action also has a url mask used to build the http request url, and may require authentication using a token that has been granted to the caller application. Finally, a Parameter has a name, and it (the parameter) may be required or not.

Figure 1 depicts the classes and object properties of the Google+ API ontology. While analyzing the Google+ API we found that in some cases the Field of a Resource Type provides a reference to another Resource Type. This type of connection between resource types through their fields is not clearly presented in the API documentation, and a developer has to study the detailed documentation of the responses of

 ISBN: 978-1-61208-196-0

various actions in order to detect it. We describe this type of connection in our ontology with the *connectsWith* object property of *Field* Class. There are also some common optional parameters that can be applied to any action. We used the *hasCommonParameter* object property connecting *API* and *Parameter* classes to describe this relation.

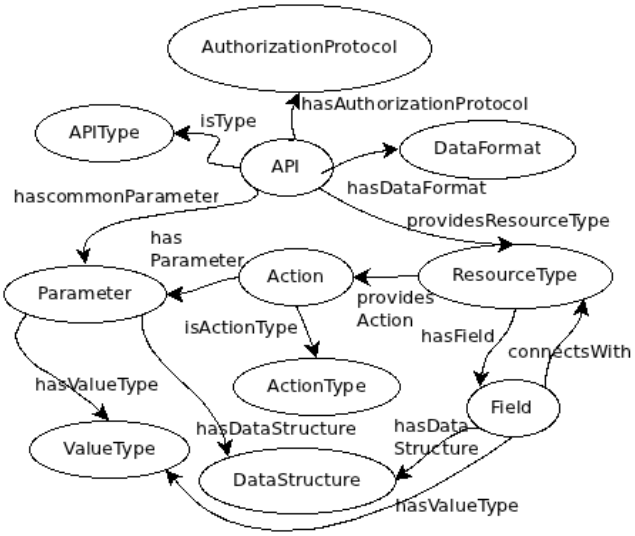

Figure 1. The classes and object properties of the ontology. An oval represents a class and an arrow stands for an object property.

F. *Specification of the value types and restrictions of the properties*

We defined the value types and restrictions of the properties of the ontology by analyzing the classes specified at the previous step. For example, the type property of the *API* class takes exactly one value that has to be instance of the *APIType* class, while the *connectsWith* property of the Field class can have at most one value of the type *ResourceType*. Figure 2 lists the properties and their value types for the *API* and *ResourceType* classes.

API		
hasAuthorizationProtocol	Instance*	AuthorizationProtocol
isAPIType	Instance*	APIType
hasCommonParameter	Instance*	Parameter
providesResourceType	Instance*	ResourceType
hasDataFormat	Instance*	DataFormat
baseUrl	String*	
documentationUrl	String*	
name	String*	

ResourceType		
providedBy	Instance*	API
hasField	Instance*	Field
providesAction	Instance*	Action
connectedTo	Instance*	Field
documentationUrl	String*	
name	String*	

Figure 2. Properties and value types of API and ResourceType classes.

G. *Creation of instances*

We defined the Instances of the Ontology based on the documentation of the API. We firstly created an instance of the *API* class representing the Google+ API. Since Google+ API is a Restful API, we created the *RestfullAPI* instance of the *APIType* class. The API uses the JSON data structure, so we created a *DataFormat* instance for it. The API also uses the OAuth authentication protocol for granting access to applications, so *OAuth* is an instance of the *AuthorizationProtocol* Class. Google+ API is currently read only, so all its actions are of *ActionType GET*, corresponding to the GET HTTP/1.1 method.

Based on our study of the parameters and return values of the Actions provided by the API, we identified 5 instances of the *ValueType* class: *String*, *UnsignedInteger*, *Boolean*, *DateTime* and *ResourceType*.

Two instances of the *DataStructure* class where also created: *SingleValue* and *List*.

The API explicitly specifies three main resource types (People, Activities and Communities), but with a more thorough study we identified a much larger number of resource types. The API does not currently provide actions for directly accessing all those resource types, but they can be indirectly accessed through the actions provided by the main three resource types. In our ontology we defined all the identified resource types as instances of *ResourceType* Class. Thus we created 25 instances of the *ResourceType* class: *Access* (identifies who has access to see an activity), *AccessItem* (an Access entry), *Activity* (a note that a user posts to her stream), *ActivityFeed* (list all of the activities in the specified collection for a particular user), *Actor* (the person who performes an activity), *Attachment* (the media objects attached to this activity), *CommentObject* (the object of a comment), *Circle* (a Google+ Circle), *Comment* (a comment is a reply to an activity), *CommentFeed* (list of all comments for an activity), *Email* (an email adderess for a person), *Embed* (if an attachment is a video, the embeddable link), *Name* (an object representation of the individual components of a person's name), *Object* (the object of an activity), *Organization* (an organization with which a person is associated), *PeopleFeed* (a list of all public profiles), *Person* (a person as represented in the Google+ API), *Place* (a place where a person has lived), *Plusoners* (people who +1'd an activity), *PreviewImage* (the preview image for photos or videos), *ProfileImage* (the representation of the person's profile photo), *Provider* (the service provider that initially published an activity), *Replies* (comments in reply to an activity), *Resharers* (people who reshared an activity) and *Url* (a URL for a person).

Finally, we created an instance of Field class for every property of every *ResourceType*, an instance of *Action* class for every action presented in the documentation of the API, and an instance of the *Parameter* class for every action parameter.

 ISBN: 978-1-61208-196-0

IV. REPRESENTATION AND VISUALIZATION OF THE ONTOLOGY

For the representation of the ontology we used the RDF/XML exchange syntax for the OWL ontology language. We used VIM text editor for editing the XML expressions of the classes and the properties and the specialized ontology editing software Protégé for checking the ontology, creating instances, and producing visualizations. Figure 3 is a visualization depicting the connections detected between the main resource types in the ontology. From this visualization we observe for example that a resource of type Person can be the Actor of an Activity or a Comment, or a member of a feed of people that Reshared or "PlusOned" (a term that is used by Google+ for evaluating other user's activities) the Object of an Activity.

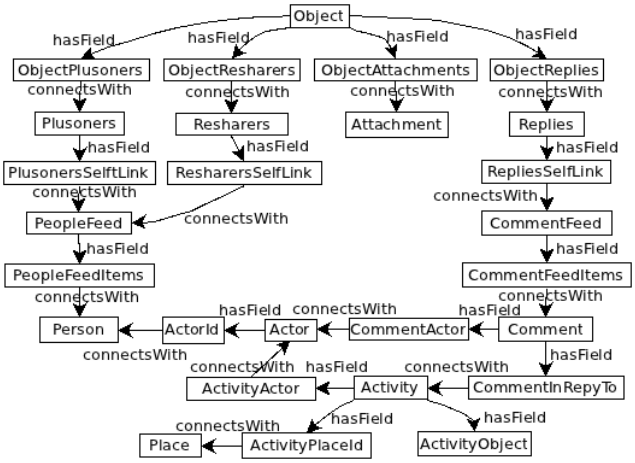

Figure 3. Connections between the main resource types in the ontology.

Figure 4 depicts all the fields of the Object resource type and their types.

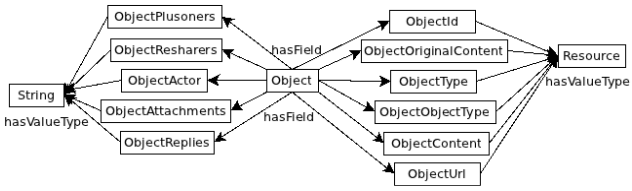

Figure 4. Fields of the Object resource type and their value types.

V. TEST QUERIES

In order to test the proposed ontology we run test queries regarding the completeness and correctness of the resulting ontology and validated the results. We queried for all class instances and their properties and cross-checked the returned results with the API documentation pages. We also made sure that all the identified instances were returned. Figure 5 depicts the query for getting the name, description and documentation url for all instances of *ResourceType* class.

We also the run two sets of usage test queries and verified the returned results. For the first set of queries, we tried to extract information useful for developers that wish to use the API for building Mashups. Such queries are: (1) What authentication protocol is supported by Google+ API? (2) What is the API's documentation url? (3) What actions and what parameters can be used for directly accessing a Person resource? (4) What resources can be directly accessed through the API? (5) What are the resource types that provide a second rank reference to the Person resource type (i.e. Have a field that connects to a resource type that has a field that connects to Person)?

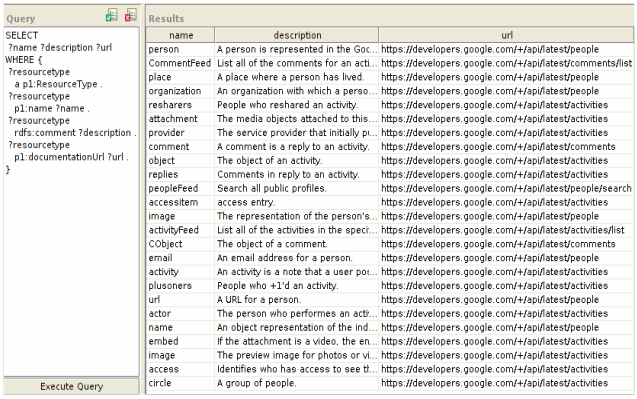

Figure 5. The SPARQL Query for getting the name, description and documentation URL for all ResourceType instances returns correct info for all the 25 identified resource types.

For the second set of queries we assumed that the ontology is used in ontology-based software for automatically invoking API's methods. Such software needs to extract low-level information about the actual method calls needed for performing an action and the structure of the data needed to be exchanged. Some example queries of this type are the following: (1) What is the APIs base url? (2) What is the APIs data format? (3) What is the urlMask of an Action? (4) What fields are contained in a Person resource type and what value type and data structure is each of them?

Moreover if such software is programmed in a higher level of abstraction, it may execute complex queries on the ontology in order to combine data form multiple API resources or to translate generic actions into sequences of API calls. For example: (1) What resource types that can be directly accessed through a GET Action provide a reference to an Email resource type? (2) What sequence of Actions can be called in order to get the image (PersonImage) of the Actor of an Activity?

We expressed the above queries in the SPARQL ontology querying language and executed using Protege. Figure 6 depicts a usage test query and the returned results.

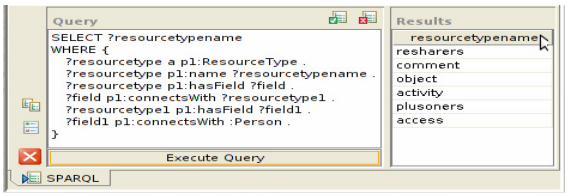

Figure 6. SPARQL query for getting all the resource types that provide second rank access to Person resource type.

VI. CONCLUSIONS AND FUTURE WORK

Ontology-based representations of SNS APIs can help developers comprehend the structure and functionalities of SNS and their APIs and share this knowledge. Moreover they can be used to link those APIs with public Social Semantic Web ontologies and vocabularies and for enabling automatic ontology-based service composition.

We studied the API provided by Google for its popular Google+ SNS and created an ontology based representation of its structural and functional properties. For designing the ontology we followed the methodology proposed by Noy and MacGuinness in [23]: First we specified the domain and the purpose of the ontology, then specified the classes of the ontology, the hierarchy, the properties and finally we created the instances. We tested the ontology with SPARQL queries. The proposed ontology reveals the existence of important resources and connections between them that are not clearly presented in the official documentation. We identified a total of 25 resource types in Google+ API connecting with each other in various ways. We have made the ontology publicly accessible in OWL format at http://goo.gl/Oefl2.

In this work, we focused on representation of the basic structural and functional features of Google+ API such as the resources it provides, the way they connect with each other and the actions they provide. We would like to extend the ontology with descriptions of the authentication process, the manipulation of paging and partial queries and bindings of the actions to client libraries method calls, in order to support automatic invocation of the API calls from ontology driven applications. In the near future we would also like to connect the ontology with ontologies and vocabularies like FOAF and SIOC that describe more abstract concepts about users, social networks and content. Finally, we would like to create ontology based representations for the APIs provided by other popular SNS such as Facebook and Twitter and to use them for building ontology-based mashups that automatically combine data and functionalities from multiple SNS.

REFERENCES

[1] J.G. Breslin, A. Passant, and S. Decker , "The Social Semantic Web", Springer-Verlang Berlin Heidelberg, 2009

[2] Tim O'Reilly, "What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software", Published in: International Journal of Digital Economics No. 65, March 2007, pp. 17-37.

[3] N. Zang, M.B. Rosson, and V. Nasser, "Mashups: who? What? Why?", In: CHI 2008: CHI 2008 extended abstracts on Human factors in computing systems, ACM, New York, 2008, pp. 3171-3176.

[4] T. R. Gruber, "Toward Principles for the Design of Ontologies Used for Knowledge Sharing", In International Journal of Human-Computer Studies, Vol 43 Issue 5-6, Nov./Dec. 1995, pp. 907-928.

[5] J. Hendler, "Agents and the Semantic Web", In IEEE Intelligent Systems, Vol. 16 No 2, 2001, pp. 30-37.

[6] J. Dietrich and C. Elgar, "Towards a web of patterns", In: Web Semantics: Science, Services and Agents on the World Wide Web, vol. 5, num. 2, Elsevier, 2011.

[7] B. Guo, D. Zhang, and M. Imai, "Toward a cooperative programming framework for context-aware applications", In Personal and Ubiquitous Computing, Vol 15, Issue 3, March 2011, pp. 221-233.

[8] H. Xiao et al, "An automatic approach for ontology-driven service composition", Proc. IEEE International Conference on Service-Oriented Computing and Applications (SOCA) 2009, Taipei, Taiwan, 14-15 December 2009, pp 1-8.

[9] H. Xiao et al, "An Approach for Context-Aware Service Discovery and Recommendation", Proc., IEEE International Conference on Web Services (ICWS), 5-10 July 2010, Miami, FL, 2010, pp. 163 – 170.

[10] J. Berslin and S. Decker, "The Future of Social Networks on the Internet: The Need for Semantics", IEEE Internet Computing, vol. 11, November 2007, pp. 86-90.

[11] The Friend of a Friend (FOAF) project, online at http://goo.gl/Rdpja, retrieved December 2011.

[12] U. Bojārs and J.G. Breslin (editors), "SIOC Core Ontology Specification", W3C Member Submission 12 June 2007, online at http://goo.gl/8OQV1, 2007, retrieved December 2011.

[13] Dublin Core Metadata Initiative, "Dublin Core Metadata Element Set", Version 1.1, online at http://goo.gl/MHLlw, 2010, retrieved December 2011.

[14] A. Miles and S. Bechhofer (editors), "SKOS Simple Knowledge Organization System Reference", W3C Recommendation 18 August 2009, online at http://goo.gl/ypDOU, 2009, retrieved December 2011.

[15] B. Zhou and C. Wu, "Social networking interoperability through extended FOAF vocabulary and service", Proc. 3rd International Conference on Information Sciences and Interaction Sciences (ICIS), 23-25 June 2010, Chengdu, China, 2010, pp. 50 – 55.

[16] B. Hartman, S. Doorley, and S.R. Klemmer, "Hacking, Mashing, Gluing: Understanding Opportunistic Design", in IEEE Pervasive Computing, vol. 7 issue 3, July 2008.

[17] J. Wong, J. and J. Hong, "What do we "mashup" when we make mashups?", Proc. WEUSE '08: Proceedings of the 4th international workshop on End-user software engineering, 2008.

[18] S. Kurkovsky, D. Strimple, and E. Nuzzi, "Convergence of Web 2.0 and SOA: Taking Advantage of Web Services to Implement a Multimodal Social Networking System", proc. 11th IEEE International Conference on Computational Science and Engineering - Workshops, 2008, pp. 227-232.

[19] Wikipedia, Google+, online at http://goo.gl/N5rou, retrieved December 2011.

[20] C. Chabot, "Getting Started on the Google+ API", The Google+ Platform Blog, online at http://goo.gl/EPfiM, September 15, 2011, retrieved December 2011.

[21] E. Hammer-Lahav (editor), "The OAuth 1.0 Protocol, Internet Engineering Task Force (IETF)", online at http://goo.gl/eN6VT, April 2010, retrieved December 2011.

[22] D. Crockford, "The Media Type for JavaScript Object Notation (JSON)", online at http://goo.gl/7oDGo, July 2006, retrieved December 2011.

[23] N. F. Noy and D. L. McGuinness, "Ontology development 101: a guide to creating your first ontology". Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880. Stanford Knowledge Systems Laboratory. Available at http://goo.gl/kr6n4, 2001, retrieved December 2011.

[24] Google, Inc., "Google+ API", online at http://goo.gl/q2jai, retrieved December 2011.

 ISBN: 978-1-61208-196-0