

AutoMap4OBDA: Automated Generation of R2RML Mappings for OBDA. *20th International Conference Knowledge Engineering and Knowledge Management, 2016*

AutoMap4OBDA: Automated Generation of R2RML Mappings for OBDA

Álvaro Sicilia¹, German Nemirovski²

¹ ARC Ingeniería i Arquitectura La Salle, Universitat Ramon Llull, Barcelona, Spain
asicilia@salleurl.edu

² Business and Computer Science Albstadt-Sigmaringen-University of Applied Sciences,
Albstadt, Germany
nemirovskij@hs-albsig.de

Abstract. Ontology-Based Data Access (OBDA) has become a popular paradigm for the integration of heterogeneous data. The key components of an OBDA system are the mappings between the data source and the target ontology. The great efforts required to create manual mappings are still a significant barrier to adopting the OBDA. Current relational-to-ontology mapping generators are far from providing 100% of the mappings required in real-world problems. To overcome this issue we present AutoMap4OBDA, a system which automatically generates R2RML mappings based on the intensive use of relational source contents and features of the target ontology. Ontology learning techniques are applied to infer class hierarchies, the string similarity metrics are selected based on the target ontology labels, and graph structures are applied to generate the mappings. We have used the RODI benchmarking suite to evaluate AutoMap4OBDA which outperforms the most advanced state-of-the-art mapping generators.

Keywords: Relational-to-ontology mappings, R2RML, ontology learning, OBDA

1 Introduction

In recent years, we have witnessed a growing need for businesses to integrate data stored in different formats, using different value scales, distributed in numerous sources, and built upon different schemas. This need arises, on the one hand, in a context of information integration, in particular in interdisciplinary projects, and also in the case of fusions or joint collaborations of companies and institutions. On the other hand, and thanks to the widely available initiatives for open data access such as the Linked Open Data, the quantity of the available data is steadily increasing.

The Ontology-Based Data Access (OBDA) approach has been developed to address this need. In OBDA settings, data queries formulated in terms of an ontology, are rewritten with respect to the native database schema, and forwarded to the database [17]. The main components of an OBDA system are a database – which contains the data –, an ontology – which represents the conceptualization of a domain –, map-

pings between the database and the ontology, and the query rewriter which transforms queries into an “understandable” form for the native database management system.

In this context, the development of mappings between the ontology and the database is one of the key issues. The manual mapping in OBDA systems is, nowadays, the most widely adopted solution in academic and industry communities in spite of it being extremely time-consuming and requiring high levels of human expertise [17]. The development of mappings between the ontology and relational data schemas is a process that requires knowledge from a specific domain, for instance medicine, or mechanical engineering, as well as skills in Entity Relationships modeling and ontology design. Finding users with this profile is difficult. Furthermore, creating hundreds of mapping rules manually is error prone. Therefore, creating mappings is still a significant barrier in the adoption of the OBDA approach [13].

Efforts for the automation of the mapping development tasks have been carried out in several studies such as [13]. Sequeda et al. proposed a process for the automated generation of relational-to-ontology mappings as an alternative to the manual process [18]. It starts with a reverse engineering step to derive an ontology from the database schema. Such automatically generated ontology is called putative ontology. It usually has a flat structure as compared to a domain ontology which is created by a community of experts or by a standardization body. In the second step, the putative ontology is aligned with the domain ontology used in an OBDA system by applying ontology matching methods. Thirdly, the alignments between both ontologies are used to derive the final mappings.

As shown in the results of the relational-to-ontology RODI benchmark conducted by Pinkel, current automated mapping generators can address simple mappings. However, all systems failed on advanced tests in which relational databases use design patterns that differ greatly from those used in ontologies [14, 15]. Our analysis – reported in section 5 – shows that the current mapping generators basically rely on the relational schema and do not fully take into account the contents of the database and the features of the target ontology, i.e. the one to be mapped onto the database. We believe that there is room for improvement in those systems if database and ontology features are extensively taken into account.

In this paper we present AutoMap4OBDA¹, a system for the automated generation of R2RML mappings between a relational database and existing target ontology specified in OWL. The main motivation behind the design of this system was to obtain mappings of significantly higher quality compared to the competitors’ results. To do so, the database content and features of the target ontology are taken into account during the mapping generation process. Moreover, AutoMap4OBDA has been designed to be used in OBDA scenarios (Section 2) and is able to generate fully compliant R2RML mappings without user intervention. The techniques used by AutoMap4OBDA are described in Section 3. An evaluation using the RODI benchmark suite has been described in Section 4. Finally, AutoMap4OBDA has been compared with the existing systems of the state-of-the-art (Section 5).

¹ <http://arc.salleurl.edu/automap4obda/>

2 Concepts of an OBDA System

The main purpose of an OBDA system is to provide access to data stored in a relational database by means of queries formulated in terms of a common domain specific vocabulary encapsulated as an ontology. This approach complies with the requirement of interoperability for systems and applications by facilitating the querying of the relational sources without taking into account their specific schemas.

The core purpose of a relational database is to store data and relations among data in tables. Data is grouped in relations (i.e., tables) of tuples (i.e., table rows), in which each tuple is composed of attributes (i.e., columns). The attributes are defined with a name and a set of permitted values for a particular domain. In this way, a relation is a set of n-tuples where each tuple has the same type of attributes. The data stored in relations should be uniquely identified and linked to related data through attributes.

An ontology is "a formal specification of a shared conceptualization" [3]. The term "shared conceptualization" indicates reaching a consensus among experts whereby the conceptualization represents the related knowledge domain. Ontologies incorporate concepts which are sets, collections, types of objects or kinds of things; object properties that are aspects, properties, features, characteristics that an object can have; and datatype properties that describe types of values. Furthermore, in an ontology items of a knowledge domain are specified by means of axioms.

In terms of R2RML, a declarative language recommended by the W3C for the definition of relational-to-ontology mappings, mappings are declared as a subject map which defines the Internationalized Resource Identifier (IRIs) generated from the logic table which can be defined as a base table, a view (i.e. the result set of a stored query), or a SQL query. Moreover, the data-to-object mappings are declared as predicate and object maps. The subject and objects maps describe how the IRIs should be generated using the columns specified in the logic table and the elements of a target ontology. Generating R2RML mappings consists of two main tasks: first, finding the correspondences between the elements of the database and the target ontology and second, obtaining the SQL views needed to generate the IRIs of the subject and predicate object maps.

3 AutoMap4OBDA: Automated Mappings for OBDA

AutoMap4OBDA is a system that automatically generates R2RML mappings from a relational database and an ontology (further referred as a target ontology). It is invoked from command line passing the connection parameters to the database (e.g., string connection, database name, user, and password), the path to the ontology file, and parameters to enable/disable some features (e.g., Applying ontology learning methods). The goal behind the design of AutoMap4OBDA is to make a relational-to-ontology matcher dependant not only on the database schema, but also on the database content and features of the target ontology. For example, class hierarchies are mined from the instances of the database.

The mapping generation process by AutoMap4OBDA is aligned on the common strategy described in the introduction to this paper: the generating of a putative ontology derived from the database, the alignment of the putative and the target ontology using ontology matching techniques, and the generating of the final R2RML mappings based on the alignment. The mapping generation process includes two steps more than the common strategy: augmenting the putative ontology and extending the alignment. In particular, the AutoMap4OBDA workflow comprises five steps carried out sequentially. Each step is carried out by a corresponding module (Fig. 1).

- *Reverse engineering methods*. The module contains the functionalities to derive a putative ontology from a relational schema.
- *Ontology learning*. The module contains the method to infer class hierarchies based on the database content.
- *Matcher*. The module contains two sub-modules, one to read the database schema and another one to implement the string similarity metric selection based on labels (i.e., StringAuto and PropString).
- *R2RML generator*. It contains functionalities to generate the R2RML mappings based on the findings of the matcher.
- *Ontology Tools*. It contains functionalities to outline features of the target ontology: maximum entropy, maximum class name length, and maximum number of subclasses among others.

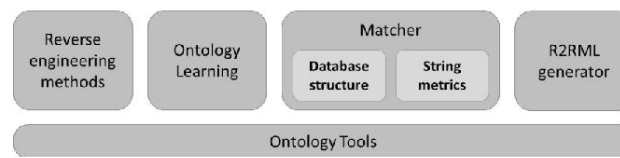


Fig. 1. Architecture of AutoMap4OBDA

Before starting the workflow, the target ontology is saturated. That is: specifications of domains and ranges of properties are rewritten in form of axioms.

3.1 Step 1: Generating the Putative Ontology from a Database Schema

The first step comprises generating a putative ontology from a database schema so that each ontology class corresponds to a database table. The name of the table is used to define the class name as well as the class label. Each attribute of a table corresponds to a data property in which the name of the attribute determines the name of the property. If the database table attribute is a foreign key, the domain of the resultant object property will be the class which corresponds to this table and its range will be the class which corresponds to the table whose primary key is referred by the foreign key. Moreover, a class is generated for each attribute and an object property which connects the table class and the attribute class. For each element of the putative

ontology (classes, object or data properties) a reference to its origin database element, i.e. a table or a column, is stored as an annotation.

3.2 Step 2: Augmenting the Putative Ontology Basing on the Database Content

In the second step, the putative ontology is augmented based on the database contents. A putative ontology derived from a database schema only, without taking into consideration data values would have a similar structure to the database schema. However, by learning patterns contained in data values the ontology structure can be enriched considerably [4]. In particular, the class hierarchy can be identified by applying data mining methods to the database contents. For example, if a table *Building* is related to a class *Building* of the target ontology, the values of the attribute *Building_Use* can be mined to find subclasses for class *Building* such as *Office* and *Residential*.

One of the main issues in this context is to identify the “categorizing” attributes where subclasses can be extracted from. Cerbah proposed two ways to identify candidates by “categorizing” attributes: Identification of lexical clues in attribute names and filtering through entropy-based estimation of data diversity [4]. In the first option, attributes are selected if their names can reveal a specific role in the table. For example, the attribute *Use* of the table *Building* might determine subclasses of the class *Building*. The problem is that the lexical clues are strongly related to the domain of the database and the language of the contents. Cerbah’s suggestion was that the user provided the clues. The second option is based on selecting attributes that have the most balanced distribution of instances. The attributes are characterized using the concept of entropy from information theory which is a measure of the uncertainty of a data source. The attributes with highest entropy – usually primary key attributes – and lowest entropy – attributes with highly repetitive content (up to a single value only) – are usually discarded. The problem with this option is that the discarded attributes may contain candidates to be subclasses. For example, the *Use* attribute of the table *Building* has few instances highly repeated such as *Residential* and *Office*.

In this step we have implemented a set of rules based on the properties of the database and the target ontology in order to select attributes that can be mined to obtain hierarchy classes. In particular, each attribute of a relational table and its values are discarded, that is are not represented within a putative ontology, if they match with one or more of the following rules:

- *The entropy of an attribute is greater than the maximum entropy of the ontology.* By applying this rule we avoid the generation of putative classes whose entropy would be greater than the entropy of any class in the target ontology. We define the entropy of an ontology as the maximum entropy of a class. In order to calculate the entropy of a class, a list of subclasses that can be reached at a certain depth, which is a fraction of the maximum depth of the ontology, is needed. The maximum depth of an ontology is the maximum depth value of all classes. The depth of a class is the maximum number of edges on the longest downward path between the given class and a subclass-leaf (a class which is not a super-class of any class). The

entropy of candidate attributes and the entropy of a set of subclasses are calculated according to definitions of Cerbah [4].

- *The number of different values of an attribute is greater than the maximum number of subclasses of any class of the ontology.* By applying this rule we avoid a situation in which a putative class has more subclasses than the maximum number of subclasses of the corresponding class in the target ontology. The maximum number of subclasses is determined as a number of subclasses of the given class that can be reached at a certain depth. This depth is a fraction of the maximum depth of the ontology obtained in the calculation of the ontology entropy.
- *The length of a value is greater than maximum class name length in the target ontology plus a factor.* This rule is to avoid generating classes that will not be matched in the next step since the length of the value is much greater than the maximum class name length of the ontology.
- *An attribute value is not a text.* Non text attribute values such as Numbers and Booleans cannot produce class names.
- *An attribute value contains a URL.* A URL cannot be a class name.

The values of attributes that have not been filtered become putative classes that are subclasses of the attribute class. The values of attributes are used as class identifiers and labels.

3.3 Step 3: Matching Using String Similarity Metrics

In this step the classes and data properties of the augmented putative ontology and the target ontology are matched. Full-featured ontology matcher systems use syntactic, semantic, and structural similarity metrics to find correspondences between entities of two ontologies (a source and a target one). Mature tools for ontology alignment can be found in the literature such as LogMap [9], however, when the source ontology is a putative ontology derived from a relational database, those systems may make it difficult to find correspondences between elements of two ontologies. The reason for that is that “hand-made” ontologies, like the target ontology in our settings, usually specify domain knowledge on a high-level of abstraction while putative ontologies derived from relational schemas describe the syntactical structure on a very low level of granularity. Moreover, different design patterns are usually used in ontologies and relational schemas. Therefore, the use of structural metrics may hinder the detection of correspondences.

In this context more simple String-based alignment techniques perform better. These techniques use a predefined string similarity metric. However, as stated by Chetham and Hitzler [5], for some types of ontologies, the performance of different string similarity metrics varies greatly in terms of precision and recall. To address this issue, the authors proposed a set of guidelines to choose the proper metric based on the number of words per entity label after tokenization, on the language of the ontologies and on embedded synonyms. Moreover, the selection takes into account whether the goal is to maximize precision or if it is to recall measures. These guidelines have been implemented by the authors in two matchers named StringAuto [5] and

PropString, an entirely string-based approach to aligning properties [6]. AutoMap4OBDA integrates both matchers to match the classes and properties instead of using a full-featured alignment system. The output of this step is a list of correspondences between the classes of the putative and the target ontology and a list of the correspondences between properties of both ontologies.

3.4 Step 4: Extending the Alignment According to the Target Ontology

In the process of generating the putative ontology, mappings are generated between the relational database and the putative ontology. Later, the alignment between the putative ontology and the target ontology is used to update the mappings. Finally, the modified mappings can be used to query the source by means of queries referring to the target ontology. The main drawback of this approach is that the mappings strongly reflect the relational database structure which might not be the same as the structure of the target ontology. This may lead to the generation of incorrect and incomplete mappings.

To overcome this issue, AutoMap4OBDA implements a method to rely on the structure of the target ontology (instead of the putative ontology) to extend the alignment with new correspondences. Thus, it uses the database schema to assure that SQL query can be obtained according to the new correspondences. Once classes of the putative and target ontologies are primarily aligned, additional correspondences between them can be established according to the object properties of the target ontology. This step takes as input correspondences found by the initial matching (step 3). Then, property paths between two classes – of the target ontology – matched in the step 3 are analyzed. These paths are built by the range/domain relations that connect two classes with each other. Such paths can contain arbitrary number of properties and classes connected by these relations. The idea is to find additional correspondences to the classes that are located within such chains. Two classes of the target ontology – previously matched to two putative classes in Step 3 – are taken as input for the step 4 if:

1. There is a property path connecting these two classes within the target ontology,
2. There is a relational path (over a set of foreign keys) between the pair of database tables that correspond to the two putative classes matched to the two classes of the target ontology.

Thanks to the first rule, the connectivity between classes is assured in terms of the target ontology since at least one object property will exist between those concepts. The second rule, assures that a SQL query can be obtained which involves the tables mapped onto the two target ontology classes. This SQL query will be required in Step 5 (described later) to specify the logic table of R2RML mappings. The second rule uses the database schema instead of the putative ontology because the putative ontology had been augmented in Step 2 increasing the difficulty to generate the SQL query.

In order to obtain the property paths, both the database schema and the target ontology are represented as two graph structures. A graph is generated from the database

elements where the tables and columns are nodes. The relations of foreign keys are edges between correspondent columns (Algorithm 1). Another graph is derived from the target ontology where the classes and properties are nodes while the edges are the domain and ranges of the properties (Algorithm 2). In both algorithms there is a guarantee that nodes are only inserted once.

Algorithm 1: Creating a graph from a relational database schema

```

INPUT: S = Database schema
OUTPUT: G = Graph

1 For each Table T in S do
2   G.addNode(T.name);
3 For each Column C in T.listOfColumns do
4   If C.isForeignKey then
5     G.addNode(T.name + "." + C.name);
6     G.addEdge(T.name, T.name + "." + C.name);
7     G.addEdge(T.name + "." + C.name, C.foreigntable + "." + C.foreignkey);
8     G.addEdge(C.foreigntable + "." + C.foreignkey, C.foreigntable);
9 return G;

```

Algorithm 2: Creating a graph from an ontology

```

INPUT: O = Ontology; R = Reasoner
OUTPUT: G = Graph

1 For each OWLClass C in O.getClassesInSignature() do
2   G.addNode(C);
3 For each OWLObjectProperty OP in O.getObjectPropertiesInSignature do
4   For each OWLClass CD in R.getObjectPropertyDomains(OP) do
5     For each OWLClass CR in R.getObjectPropertyRanges(OP) do
6       G.addNode(OP);
7       G.addEdge(CD, OP);
8       G.addEdge(OP, CR);
9 return G;

```

To find a path between a pair of tables according to the database schema and a pair of target concepts according to the target ontology an algorithm for finding the shortest path is invoked on the graphs created by algorithm 1 and 2 accordingly. Although it is not the most performing algorithm, Dijkstra's algorithm was selected because of its ease of implementation [7]. The extending alignment algorithm (Algorithm 3) iterates over matches found in step 3 and invokes Dijkstra's algorithm (*findPathBetween*) to obtain *ontP*, the shortest path between all pairs of classes (i.e., *S* and *T*) of the target ontology using the graph created with Algorithm 2. Moreover, *findPathBetween* is invoked to obtain *dbP*, the shortest path between tables of the database schema using the graph obtained through Algorithm 1. Then, if both paths exist, the components of the *ontP* are processed. The algorithm iterates over the components of the path and checks if the target nodes *nD* have been already matched to the putative classes by looking into the current list of matches *M*. In the case that they have not been matched then they are added to the list of new matches *M'*. The target nodes *nT* are aligned to a putative class from the target nodes. That is, they share the same table and column of the database (i.e., *nD.table* and *nD.column*).

The method applied in this step takes into account two mapping cases (Fig. 2). The first case occurs when the minimum path length between a pair of concepts of the target ontology has more nodes (concepts) than the minimum path length between the aligned concepts of the putative ontology. In this case, extra mappings (i.e., Triple-

Maps) are generated. The IRIs and SQL queries generated for these extra mappings are the same as those generated for the precedent concept (the leftmost concept in the upper chain on Figure 2).

Algorithm 3: Extending alignment

```

INPUT: M = ListOfMatches, dbG = Graph, ontG = Graph
OUTPUT: M' = ListOfNewMatches

1 For each Match S in M do
2 For each Match T in M do
3   ontP = findPathBetween(S.domainname, T.domainname, ontG);
4   dbP = findPathBetween(S.table, T.table, dbG);
5   If ontP.isEmpty() or dbP.isEmpty() then continue;
6   For int i = 0; i < ontP.getNodeList().size; i += 2 do
7     nD = ontG.getNodeSource(ontP[i]); //domain class
8     op = ontG.getNodeTarget(ontP[i+1]); //object property
9     nT = ontG.getNodeTarget(ontP[i+2]); //target class
10    If M.contains(nD) and !M.contains(nT) and !M'.contains(nT) then
11      nT.table = nD.table;
12      nT.column = nD.column;
13      ObjectPropertyList.add(op, nD, nT);
14      M'.add(nT);
15 return M';
    
```

In the example of Figure 2, the mappings for the concepts *Room* and *Wall* will have the same IRI and SQL queries as those generated for the mappings of the *Building* concept. The second case occurs when the minimum path length between a pair of concepts of the putative ontology has more nodes (i.e., concepts) than the minimum path length between the aligned concepts of the target ontology. This case is solved by generating an SQL query with multiple join clauses to connect all nodes of the path. Since a path between concepts of the database schema has been found for each pair of connected classes, the SQL query can be obtained by means of joining clauses.

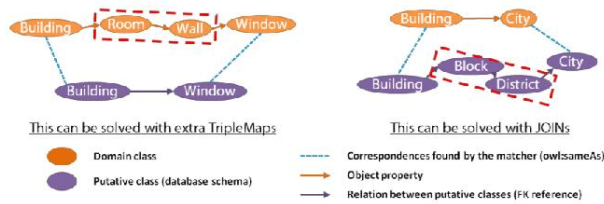


Fig. 2. Mapping generation cases. 1) Less source elements than target elements (Left). 2) More source elements than target elements (Right).

3.5 Step 5: Generating R2RML Mappings

The last step of the AutoMap4OBDA workflow is to generate the R2RML document according to the correspondences found by the matcher. For each correspondence a *TripleMap* is generated which contains a *predicateObjectMap* to define the data properties already aligned. Then, for each object property an extra *TripleMap* is generated. The logical tables of the *TripleMaps* are defined as SQL queries which are automati-

cally obtained according to the database schema paths found in step 4. That is, SQL join clauses between tables are automatically set by means of the foreign keys. The triples map in Listing 1 includes the SQL query obtained for the mapping in Figure 2 (right part).

```
<mapping1> a rr:TriplesMap;
rr:logicalTable [rr:sqlQuery "SELECT a.id, d.id FROM Building AS a
INNER JOIN Block AS b ON a.fkBlock=b.id INNER JOIN District AS c ON
b.fkDistrict=c.id INNER JOIN City AS d ON c.fkCity=d.id "];
rr:subjectMap [rr:template ".../building/{a.id}";rr:class ex:Building];
rr:predicateObjectMap [ rr:predicate ex:hasCity;
rr:objectMap [rr:template ".../city/{d.id}"]].
```

Listing 1. An example of a R2RML mapping in Figure 2

4 Evaluation

AutoMap4OBDA has been evaluated with relational-to-ontology benchmark suite RODI [14]. RODI offers basic test scenarios from conference, geographical, and oil and gas domains; and mixed scenarios in the conference domain where the database schema has to be matched to an ontology from a different scenario. Each scenario is composed of databases, ontologies, and a set of queries to test if the mappings generated by mapping generating system are performing well. The mappings are evaluated for each scenario by the percentage of successfully answered queries. RODI includes a wide range of relational-to-ontology mapping challenges classified as naming conflicts, structural heterogeneity, and semantic heterogeneity. RODI simulates real-world scenarios by creating different databases with modifications to reproduce design patterns and anti-patterns in databases. For a further explanation of the scenarios addressed by RODI refer to [14, 15].

We have assessed the performance of the state-of-the-art tools described in Section 5 related works against AutoMap4OBDA (**AM4O**). The results for BootOX (**B.OX**), IncMap (**IncM.**), ontop, MIRROR (**MIRR.**), COMA++ (**COMA**), and D2RQ have been obtained by RODI team [15]. While BootOX and IncMap are full relational-to-ontology matching systems, the other mapping generators cannot generate mapping according to an **existing** ontology. To overcome this issue, RODI benchmark includes an ontology matcher (LogMap [9]) to align the putative ontology generated by those systems with the target ontology.

The average execution time of AutoMap4OBDA – in an Intelcore i5 architecture with 10GB of RAM – has been less than 25 seconds per scenario for 15 scenarios, 57.96 seconds for the *Adjusted naming Conference* scenario, and 6.87 minutes for the *Oil&Gas* whose database has 70 tables with 250k records and the target ontology has 344 classes, 148 object properties, and 237 data properties.

The results show that AutoMap4OBDA comes out in the top position for eleven out of seventeen scenarios and in second position in three scenarios (Table 1). The scores are based on average of per-test F-measure. The results of AutoMap4OBDA in

the mixed scenarios such as *Target ontology: CMT*, *Target ontology: Conference*, and *Target ontology: SIGKDD* is not as good as the other scenarios because the level of semantic heterogeneity is much higher than in basic scenarios.

Table 1. Overall scores of the state-of-the-art tools and AutoMap4OBDA in RODI scenarios (scores based on average of per-test F-measure). Best numbers per scenario in bold.

Scenarios		B.OX	IncM.	ontop	MIRR.	COMA	D2RQ	AM4O
<i>Adjusted naming</i>	CMT	0.76	0.45	0.28	0.28	0.48	0.31	0.56
	Conference	0.51	0.53	0.26	0.27	0.36	0.26	0.56
	SIGKDD	0.86	0.76	0.38	0.30	0.66	0.38	0.86
<i>Restructured</i>	CMT	0.41	0.44	0.14	0.17	0.38	0.14	0.41
	Conference	0.41	0.41	0.13	0.23	0.31	0.21	0.54
	SIGKDD	0.52	0.38	0.21	0.11	0.41	0.28	0.72
<i>Combined case</i>	SIGKDD	0.48	0.38	0.21	0.11	0.28	0.28	0.62
<i>Missing FK</i>	Conference	0.33	0.41	-	0.17	0.21	0.18	0.49
<i>Denormalized</i>	CMT	0.44	0.40	0.20	0.22	-	0.20	0.52
<i>GeoData</i>	Classic Rel	0.13	0.08	-	-	-	0.06	0.44
<i>Oil&Gas domain</i>	User Queries	0.00	0.00	0.00	0.00	-	0.00	0.00
	Atomic	0.14	0.12	0.10	0.00	0.00	0.08	0.23
<i>Target ontology: Conference</i>	Conference	0.20	0.35	0.10	0.00	0.00	0.10	0.15
	CMT	0.33	0.33	0.19	0.00	0.14	0.19	0.38
<i>Target ontology: CMT</i>	CMT	0.20	0.34	0.05	0.00	0.05	0.05	0.39
	Conference	0.13	0.30	0.09	0.00	0.04	0.09	0.17
<i>Target ontology: SIGKDD</i>	CMT	0.51	0.57	0.19	0.00	0.24	0.26	0.41
	Conference	0.24	0.44	0.13	0.00	0.09	0.14	0.19
Average of the tests		0.36	0.37	0.15	0.10	0.20	0.18	0.43

In the other scenarios, AutoMap4OBDA outperforms other systems thanks to the methods described in this paper such as extending correspondences according to the target ontology. Indeed, full-featured alignments systems – such as those used in the evaluation of BootOx and ontop among others – have difficulties to match object properties when the structure of the source (putative) ontology is not similar to the structure of the target (domain) ontology. AutoMap4OBDA does not directly match the object properties of the putative and domain ontologies, but the object properties are set by the extending correspondences method described in step 4.

For example, the correspondences illustrated in Figure 3 can be found by AutoMap4OBDA but not by full-featured alignments systems. In *sigkdd_mixed* scenario, the target ontology has the object property *isCommitteeOf* whose domain is *Committee* and range is *Conference*. Moreover, in the putative ontology the correspondent object property is *committee* whose range class is *conferences* and whose domain classes are *best_paper_awards_commits*, *organizing_committees*, and *program_committees*. Those domain classes are subclasses of *committee* class in the domain ontology however an ontology matcher cannot match both object properties. In this case, AutoMap4OBDA fulfills the alignment of the object property in the extending correspondences method once it has aligned the classes correctly.



Fig. 3. Example of correspondences found by AutoMap4OBDA in *sigkdd_mixed* scenario

It is worth mentioning the *GeoData* scenario in which the method of *String similarity metric selection based on target ontology labels for ontology alignment* helped to find much more data properties than the other systems bringing a performance more than three times as high as the following system. No results have been achieved in *Oil&Gas domain User Queries* scenario. This is a real-world scenario where the queries go beyond returning a simple result list to determine whether all objects are of one class. The good result in the *Oil&Gas domain Atomic* scenario – compared with other systems – has been achieved thanks to the ontology learning method applied in step 2 of the AutoMap4OBDA workflow. In this scenario, AutoMap4OBDA was able to find several mappings where the values of the columns are used to set classes of the *subjectMap* as the following example. AutoMap4OBDA outperforms other mapping generators of the state-of-the-art because they cannot generate this kind of mapping in an automated way.

```
<mapping1_201> a rr:TriplesMap;
  rr:logicalTable [ rr:sqlQuery "SELECT pipnpdpipe FROM pipeline WHERE
  pipmedium = 'Oil'" ];
  rr:subjectMap [ rr:template "http://.../oilpipeline/{pipnpdpipe}";
  rr:class http://sws.ifi.uio.no/vocab/npd-v2#OilPipeline ].
```

Despite the good results AutoMap4OBDA is far from generating a full list of mappings derived from a relational database and a target ontology. Indeed, the average F-measure obtained in the RODI scenarios is 0.43 which is not a remarkable result but it is a step forward in relational-to-ontology mapping generators since it has improved the results by 0.06 with regard the next contender (IncMap) which has an average of 0.37.

5 Related Work

Approaches for ontology creation from existing databases without using any external resource are called direct mapping methods. Sequeda et al. not only surveyed those methods but also standardized the basic transformation rules implemented by those methods [18]. In most systems, the use of direct mapping methods implies that relational tables are mapped to concepts, columns to data properties, and relations implemented by means of foreign/primary keys to an object properties. Apart from the

application of transformation rules, direct mapping methods apply reverse engineering techniques to inspect the database schema constraints to unveil its semantics. The implementation of such basic transformation rules is described in [19]. Following this line of work there are D2RQ [2], MIRROR [12] and onto [16] systems, however they are not fully comparable with AutoMap4OBDA proposed in this paper since they do not generate mappings between a relational database and an existing ontology. These methods generate a putative ontology derived from a relational database and their corresponding mappings.

The prime example of systems that enhance direct mapping methods is RDBToOnto which applies ontology learning methods to identify of semantic patterns in the stored data with the final goal of producing expressive ontologies [4]. Indeed, the ontology learning methods implemented in AutoMap4OBDA are based on the techniques of RDBToOnto. In the same way, the direct mapping methods reported above, RDBToOnto cannot be used directly in OBDA settings.

The closest system to AutoMap4OBDA is BootOX, a relational-to-ontology mapping generator system which applies the similar transformation rules as direct mapping methods and providing support for different OWL profiles [10]. BootOX can produce a set of axioms based on one of the three OWL profiles – QL, RL, and EL – depending on the settings. Moreover, BootOX system includes LogMap as ontology matching system to align a putative ontology – BootOX authors call it a bootstrapped ontology – with a target ontology. The main difference between BootOX and AutoMap4OBDA is that BootOX does not take advantage – in an automatic mode – of the contents of the database to enrich class hierarchy of the putative ontology, instead BootOX proposes users to improve mappings generated in previous step through interaction with the system.

Alternatives to this kind of approach are systems such as COMA++ which represent the database and the target ontology as directed graphs whose nodes are to be matched [1]. Following this line of work, IncMap automatically maps the target ontology directly to the database using an intermediate graph structures – IncGraphs – and applies a flooding algorithm to merge the graphs aiming at finding the correspondences [13]. In contrast to AutoMap4OBDA, IncMap creates two intermediate graph structures which only rely on the ontology structure and database schema. The graph structures created by AutoMap4OBDA in *extending correspondences according to the target ontology* step are similar to those created by IncMap, however in AutoMap4OBDA graphs data properties and non-foreign key columns are not included as nodes in the graph. Moreover, it does not use ontology learning methods to search further correspondences between the data and classes of the target ontology.

The concept of intermediate graph structured is also present in Karma system, a semi-automatic mapping system which also offers a graphical user interface to verify and correct the mappings [11]. Although Karma inspects database contents to find the correspondences to the target ontology, human intervention is required to generate the final mappings. Therefore, it cannot work in fully automatic mode in the same way as AutoMap4OBDA.

6 Conclusions

We have presented AutoMap4OBDA, a full-featured mapping generator for OBDA scenarios. AutoMap4OBDA produces a putative ontology from a relational database and uses it as an intermediate element in the relational-to-ontology mapping process. AutoMap4OBDA requires a relational database and an ontology specified in OWL as its input. The evaluation presented in this paper confirms that AutoMap4OBDA is a step forward in relational-to-ontology mapping systems. With regards to non-synthetic scenarios such as GeoData and Oil&Gas, the performance of AutoMap4OBDA, requires improvement. Our conclusion of the evaluation is that the generation of relational-to-ontology mappings is a task that cannot be completely automated yet. An expert in the domain where the data originates from still should validate and complement the mappings automatically generated by systems. To do so, we have ensured that mappings generated by AutoMap4OBDA are fully compliant with the R2RML recommendation and that they can be loaded in Map-On, a graphical R2RML mapping editor to support users without ontology engineering and database skills in curation of relational-to-ontology mappings [20].

Providing support for non-relational data sources is an ambitious research line for mapping generators such as AutoMap4OBDA. This requires the addition into AutoMap4OBDA of other ways to extract class hierarchies from non-relational sources (e.g., XML files, graph and NoSQL databases) using ontology learning methods as well as an extension of the Extending correspondences step to handle non-relational sources. Supporting these kinds of sources will lead to using an alternative mapping language such as RDF Mapping Language (RML) [8]. The next step in the AutoMap4OBDA development will be enhancing ontology learning methods to facilitate the inferring of class hierarchies since the results have not been completely satisfactory. A translation feature will be also incorporated to address multi-language scenarios and semantic similarity techniques based on external resources.

7 Acknowledgements

This work was carried out within the research project ENERSI funded by Ministry of Economy and Competitiveness of the Government of Spain (Reference number RTC-2014-2676-3)

8 References

1. Aumüller, D., Do, H.H., Massmann, S., Rahm, E.: Schema and Ontology Matching with COMA++. In: Proceedings of the ACM SIGMOD international conference on Management of data, pp. 906–908. ACM (2005)
2. Bizer, C., Seaborne, A.: D2RQ - treating non-RDF databases as virtual RDF graphs. In: 3rd International Semantic Web Conference. Vol. 2004. Springer (2004)
3. Borst, W. N. Construction of Engineering Ontologies for Knowledge Sharing and Reuse. Technology. Vol. PhD (1997). Retrieved from <http://doc.utwente.nl/17864/>

4. Cerbah, F.: Mining the Content of Relational Databases to Learn Ontologies with Deeper Taxonomies. In: *Web Intelligence and Intelligent Agent Technology*, pp. 553-557. IEEE (2008)
5. Cheatham, M., Hitzler, P.: String similarity metrics for ontology alignment. In: *12th International Semantic Web Conference*, Springer Berlin Heidelberg (2013)
6. Cheatham, M., Hitzler, P.: The properties of property alignment. In: *9th International Conference on Ontology Matching*, pp. 13-24. Volume 1317. CEUR-WS. Org (2014)
7. Dijkstra, E. W.: A note on two problems in connexion with graphs. *Numerische Mathematik* 1(1), 269–271 (1959)
8. Dimou, A., Sande, M., Vander, Colpaert, P., Verborgh, R., Mannens, E., Van De Walle, R.: RML: A generic language for integrated RDF mappings of heterogeneous data. In: *7th Workshop on Linked Data on the Web* (2014)
9. Jimenez-Ruiz, E., Cuenca Grau, B.: LogMap: Logic-based and Scalable Ontology Matching. In: *International Semantic Web Conference*. Springer Berlin Heidelberg (2011)
10. Jiménez-Ruiz, E., Kharlamov, E., Zheleznyakov, D., Horrocks, I., Pinkel, C., Skjæveland, M.G., Thorstensen, E., Mora, J.: BOOTOX: Practical Mapping of RDBs to OWL 2. In: *International Semantic Web Conference*. Springer Berlin Heidelberg (2015)
11. Knoblock, C.A., Szekeley, P., Ambite, J.L., Goel, A., Gupta, S., Lerman, K., Mallick, P.: Semi-automatically mapping structured sources into the semantic web. In: *9th Extended Semantic Web Conference* (2012)
12. de Medeiros, L.F., Priyatna, F., Corcho, O.: MIRROR: Automatic R2RML Mapping Generation from Relational Databases. In *Engineering the Web in the Big Data Era*, pp. 326–343, (2015)
13. Pinkel, C., Binnig, C., Kharlamov, E., Haase, P.: IncMap: Pay-as-you-go Matching of Relational Schemata to OWL Ontologies. In: *8th International Conference on Ontology Matching*, pp. 37-48. Volume 1111. CEUR-WS. org. (2013)
14. Pinkel, C., Binnig, C., Jiménez-Ruiz, E., May, W., Ritzke, D., Skjæveland, M.G., Solimando, A., Kharlamov, E.: RODI: a benchmark for automatic mapping generation in relational-to-ontology data integration. In: *The Semantic Web. Latest Advances and New Domains*, pp. 21–37. Springer International Publishing (2015)
15. Pinkel, C., Binnig, C., Jimenez-Ruiz, E., Kharlamov, E., May, W., Nikolov, A., Skjæveland, M.G., Solimando, A., Taheriyani, M., Heupel, C., Horrocks, I.: RODI: Benchmarking Relational-to-Ontology Mapping Generation Quality. *J. SW* (2016). Under review at <http://www.semantic-web-journal.net/content/rodi-benchmarking-relational-ontology-mapping-generation-quality-0>
16. Rodriguez-Muro, M., Rezk, M.: Efficient SPARQL-to-SQL with R2RML Mappings. In: *Web Semantics: Science, Services and Agents on the World Wide Web* 33, pp. 141–169 (2015)
17. Savo, D.F., Lembo, D., Lenzerini, M., Poggi, A., Rodríguez-Muro, M., Romagnoli, V., Ruzzi, M., Stella, G.: MASTRO at work: Experiences on ontology-based data access. In *DL 2010*, pp. 20–31 (2010)
18. Sequeda, J., Garcia-Castro, A., Corcho, O., Tirmizi, S.H., Miranker, D.P.: Overcoming database heterogeneity to facilitate social networks: the Colombian displaced population as a case study. In: *18th International Conference on World Wide Web*. ACM (2009)
19. Sequeda, J., Arenas, M., Miranker, D.P.: On directly mapping relational databases to rdf and owl. In: *21st international conference on World Wide Web*, pp. 649–658. ACM (2012)
20. Sicilia, Á., Nemirovski, G.: Map-On: A web-based editor for visual ontology mapping. *J. SW*, (2016). Accepted for publication at <http://www.semantic-web-journal.net/content/map-web-based-editor-visual-ontology-mapping-0>

Map-On: A web-based editor for visual ontology mapping. *Semantic Web Journal*, 2016

Map-On: A web-based editor for visual ontology mapping

Editor(s): Krzysztof Janowicz, University of California, Santa Barbara, USA

Open review(s): Amruta Nanavaty, University of Illinois at Chicago, USA; Daniel Faria, Universidade de Lisboa, Portugal; Valerio Santarelli, Sapienza Università di Roma, Italy

Álvaro Sicilia^a, German Nemirovski^b and Andreas Nolle^b

^a*ARC Enginyeria i Arquitectura La Salle, Universitat Ramon Llull, Barcelona, Spain*

E-mail: asicilia@salleurl.edu

^b*Business and Computer Science Albstadt-Sigmaringen-University of Applied Sciences Albstadt, Germany*

E-mail: [\[nemirovskij,nolle\]@hs-albsig.de](mailto:[nemirovskij,nolle]@hs-albsig.de)

Abstract. This paper presents Map-On, a web-based editor for visual ontology mapping developed by the Architecture, Representation and Computation research group of La Salle, Ramon Llull University. The Map-On editor provides a graphical environment for the ontology mapping creation using an interactive graph layout. A point-and-click interface simplifies the mapping creation process. The editor automatically generates a R2RML document based on user inputs, particularly producing IRI patterns and SQL queries. It has been used in real scenarios alleviating the effort of coding R2RML statements which is one of the main barriers for adopting R2RML in research and in the industry.

Keywords: Ontology mapping editor, Semantic data integration, OBDA, Mapping visualization, R2RML

1. Introduction

In recent years, the interdisciplinary character of numerous projects and applications has led to an increasing need for integrating data that is related to different knowledge domains and stored in different formats. In this context, the community of experts and stakeholders that is currently working with heterogeneous data has grown considerably. These communities are usually composed by people with heterogeneous backgrounds skills and goals. In addition and thanks to widely spread initiatives such as the open data movement, the quantity and quality of the available open data is steadily increasing.

The ontology-based data access (OBDA) paradigm can be useful in scenarios where people with different backgrounds and skills wish to access heterogeneous data sources. In OBDA settings, the data sources are accessed using a high-level conceptual representation without the need to know how the data sources are actually organized [19, 4]. The main

components of an OBDA system are the ontology, which represents the conceptualization of the data sources domains, the mappings between the data sources and the ontology, and the query rewriter which receives queries in terms of the ontology and transforms them according to data sources. The queries can be processed over the ontology with reasoning purposes in order to allow of the use of hidden relations that are not explicitly defined in the data source.

In this context, the development of mappings between the ontology, which presents a generic conceptual view of the data, and the schemas of the integrated data sources is one of the key issues. In several approaches [16, 9], efforts have been made to automate the mapping tasks. However, due to the complexity of obtaining the proper semantics from the schemas, the manual mapping in OBDA systems is still currently the most widely adopted solution in academia and industry in spite of its time consumption and of the high requirements of human expertise

[23, 1]. Experts are required to have knowledge of the data schemas to be integrated and on the ontology architecture.

Mappings from relational databases to a RDF dataset are mostly implemented using R2RML, a declarative language recommended by the W3C. R2RML is supported by OBDA systems for data integration such as Mastro Studio [5] and systems for data integration [2]. As stated above, creating R2RML mappings require advanced skills and expertise in ontology design and formal logic. However, also domain experts and data owners, who usually lack the mentioned expertise, are involved in the setting up of OBDA scenarios. However, in these scenarios and for these kinds of users, the main barrier is often the lack of a visual representation of the mappings. In practice, a visual representation could help them to overcome the lack of expertise and complete the mapping task.

With this scenario in mind, we have developed Map-On, a graphical environment for ontology mapping to help different kind of users – domain experts, data owners, and ontology engineers – to create and maintain mappings between a database and a domain ontology using the R2RML recommendation. The ontology mapping environment offers visualizations of mappings based on a graph layout and supports the automated generation of IRI patterns and SQL queries for R2RML statements.

The paper is organized as follows. Section 2 gives an overview of the existing mapping editors. We briefly introduce the R2RML language in section 3. Afterwards, in section 4, the main features of Map-On are described, including its visual representation of ontology mappings, point-and-click interface and the automated generation of R2RML statements. Section 5 gives a complete overview of the tool architecture. A description of real-world deployments of Map-On is given in Section 6, and a discussion about the current limitations and future plans, is exposed in Section 7.

2. Related tools

Several tools have been developed to assist experts in specifying the mappings between the data sources and the ontologies. For instance, OntopPro extends Protégé by providing a mapping editor that integrates a SPARQL query interface with the Quest query engine [22]. In spite of the fact that the editor works with a proprietary mapping format, it can ex-

port and import R2RML files. Similar functionalities are included in the mapping editor presented by Sengupta et al. in [24], in this case with native support of R2RML. Generally, the users of these tools are technicians, experts in formal and especially description logics. These tools do not provide graphic visualization of any kind. This shortcoming makes these tools inadequate for domain experts and data owners, who despite being involved in the data integration process, do not have the necessary background in ontology engineering.

A significant help for non-experts in ontology engineering can be provided by ontology visualization techniques. They help users to inspect, navigate, and verify the ontologies and mappings [10].

Mapping editors like the Karma system constitute a good example of this. Karma is a mapping editor that provides a graphical user interface for visualizing and editing the mappings. Moreover, Karma can automatically suggest mappings and supports R2RML recommendation [9]. The mappings are displayed using a tree layout for the ontology and a table layout for the database schema. Another example is ODEMapster which works in a similar way to Karma. It also offers a graphic visualization of the database schema and of the domain ontology using a tree layout expressing the mappings in a proprietary mapping language called R2O [20]. Furthermore, the RBA (R2RML By Assertion) tool also uses a tree layout to display the mappings and supports R2RML [14].

The limitation of these tools lies in the exploitation of the tree layout for visualization. Such a layout is unable to represent the complete structure of the database schema, ontology and mapping by itself since the structure of an ontology is like a graph. Indeed, even when given a fixed screen space, it is not always possible to visualize the entire tree structure of ontologies with multiple inheritances and with a large number of descendants [7].

Visualizing ontologies and mappings with a graph layout is probably the most natural and the most commonly used form of visualization. It has been found, in a usability test, that graph layouts are more suitable for overviews and their flexibility helped users to hold attention during the mapping tasks [7]. However, graph layouts can become difficult to manage once the nodes visualized exceed a certain number.

A prominent example is the mapping visualization model presented by Lembo et al. in [11]. In this case, the mappings are presented in a graph layout including three views focused on the mapping, the ontol-

ogy, and the source. But a complete overview of the all mappings at once is not provided. RMLEditor is another example of an editor that presents the mappings using a graph layout [8]. The limitation of the mapping representation of RMLEditor is that structure of the relational source is not included in the mapping representation. However the use of graph layout in ontology mapping editors is rare.

Table 1
Summary of tool features

Tools	Mapping language	Graphic Layout	Mapping Overview
OntopPro [22]	Proprietary / R2RML	-	No
Sengupta et al. [24]	R2RML	-	No
Karma [9]	R2RML	Tree	Yes
ODEMapster [20]	Proprietary	Tree	Yes
RBA [14]	R2RML	Tree	Yes
Lembo et al. [11]	R2RML	Graph	No
RMLEditor [8]	RML/R2RML	Graph	Yes
Map-On	R2RML	Graph	Yes

Table 1 summarizes the main features of the different mapping tools and of the Map-On editor described in this paper. The tools are compared with the mapping language used (e.g., proprietary language or the R2RML standard), with the graphic layout to represent mapping elements (e.g., tree and graph), and whether they provide a complete overview of the mappings generated by the users or not.

3. Preliminaries

An R2RML mapping is a set of triples maps that is composed of a logic table which can be defined as a base table, a view (i.e., a result set of a stored query), or a SQL query; a subject map which defines the subject of the RDF triples generated from the logic table; and a set of predicate and object maps which defines the predicates (i.e., roles) and objects (e.g., RDF objects) of the RDF triples. The subject and objects maps describe how the IRIs (Internationalized Resource Identifier) should be generated using the columns specified in the logic table and the elements of a target ontology (i.e., concepts, roles and attributes).

The manual creation of R2RML mappings requires technical skills in SQL query design and in

ontology engineering at the same time. The experts who create the mappings should understand the structure of the database schemas and the target ontology in order to find correspondences between the columns of the relational tables and the ontology entities. Moreover, users have to design SQL queries for the logic tables and the IRI patterns for the subject and object maps.

The triples map illustrated in Figure 1 uses a logic table based on a SQL query for the table *census*. The IRI of the subject map uses the *ID* column of the table and the concept *Building*. The object map is defined with the role *hasAddress* and the column *Addr*. The main limitation of R2RML languages is that in the research community and in the industry, there are certain problems with the adoption of R2RML. The manual creation of R2RML mappings is a time consuming process, the mappings are syntactically heavy in terms of R2RML vocabulary (these processes imply to design valid SQL queries, to use the proper terms of the R2RML language, and to select the ontology concepts and properties), and for the experts using the language, the steep learning curve is mainly caused by gaining expertise of the language [17].

The mapping editor presented in this paper supports the creation of triples maps by providing visual representation of the mappings and a point-and-click interface which enables the user to map the columns of the tables towards the ontology concepts, roles, and attributes. The IRI patterns and the logic tables are automatically generated by the tool.

```
<mapping1> a rr:TriplesMap;
  rr:logicalTable [
    rr:sqlQuery "SELECT ID, Addr FROM census"
  ];
  rr:subjectMap [
    rr:template ".../building/{ID}";
    rr:class ex:Building
  ];
  rr:predicateObjectMap [
    rr:predicate ex:hasAddress;
    rr:objectMap [rr:column "Addr"]
  ].
```

Fig. 1. Example of a R2RML mapping.

4. Map-On features

The main goal of the Map-On tool is to support users to create mappings between a database schema and the existing domain ontology. The tool provides visualizations of mappings based on a graph layout

for the database schema and the ontology, an ontology mapping interface, and the automated generation of IRI patterns and SQL queries for the R2RML statements.

4.1. Mapping visualization

Database schemas and ontologies are usually represented using a graph layout; a recent prominent example is VOWL a visual language for visualizing ontologies as a force-directed graph layout [12]. In graph layout representations of database schemas and ontologies, the node elements (e.g., tables and concepts) are characterized as nodes and the relations (e.g., primary/foreign key constraints and roles) as edges. The mappings between a database schema and an ontology are a set of relations between their elements. That is when, for instance, a column of a relational table is used to define the IRI of a subject map and a concept of the ontology is utilized to define the type of the subject map. In this way, it becomes intuitive to represent the mappings graphically as edges between columns and concepts (Figure 2).

The Map-On tool represents a table and their columns as purple rectangles connected by a solid purple line. The relationships between tables are displayed with a purple dashed line between foreign key and primary key constraints. The ontology concepts are represented as orange ellipses, the roles as directed solid orange lines, and the attributes as green ellipses. The relations between the elements of the database and the ontology are displayed with dashed blue lines. An example of a visual representation of mappings using a graph layout can be found in Figure 2.

After a mapping has been created, the corresponding layout is automatically generated by placing the nodes in a rectangular grid arrangement. As a next step, the user can modify the position of the nodes by dragging them to the desired position.

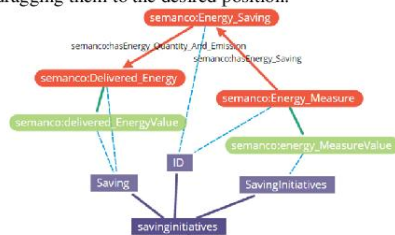


Fig. 2. Visual representation of a mapping

4.2. Ontology mapping interface

The Map-On editor provides the user with a high level of assistance in the ontology-database mapping by means of the four following features: point-and-click interface, ontology-driven mapping approach, top-down visualization, and mapping spaces.

The Map-On graphic user interface is based on a point-and-click paradigm in which most of the user's actions are carried out with the cursor. The main benefits of this kind of interfaces are the high comfort factor and the low getting-started-barrier for those users who lack skills in mapping languages such as R2RML.

Furthermore, the interface provides easy access to the elements to be mapped and fosters productivity, as complex mapping tasks can be carried out with fewer actions by the user.

Map-On implements the ontology-driven approach for editing the mappings. Namely, the user starts with selecting concepts of the ontology and as a second step generates R2RML statements by defining the proper logic tables and IRI patterns. An alternative to the ontology driven approach is the database-driven approach which starts with a selection of database elements followed by the generation R2RML statements through selecting the proper target ontology elements. As stated in [17], none of these approaches (i.e., ontology-driven and database-driven) are better; however users with a background in database can be more familiar with the ontology-driven approach.

The Map-On interface provides top-down mapping visualizations (Figure 3). In particular, the elements of the ontology and database schema (i.e., tables, columns, concepts, roles and attributes) involved in the mapping are visualized in one single representation as a global view which can be related to a set of triples maps in the final R2RML document. This approach helps the user to comprehend both database and ontology structures at the same time, and therefore to reduce mapping errors and to simplify their maintenance. Furthermore, a list of mappings is provided in the left side of the interface using the same colour styles as in the graph representation. When the cursor hovers over one mapping on the list, the node of the graph layout corresponding to that mapping is highlighted, the same as their neighbourhood nodes. This feature works the other way round as well, when the cursor hovers over a node of the graph layout, the corresponding mapping of the list is highlighted.

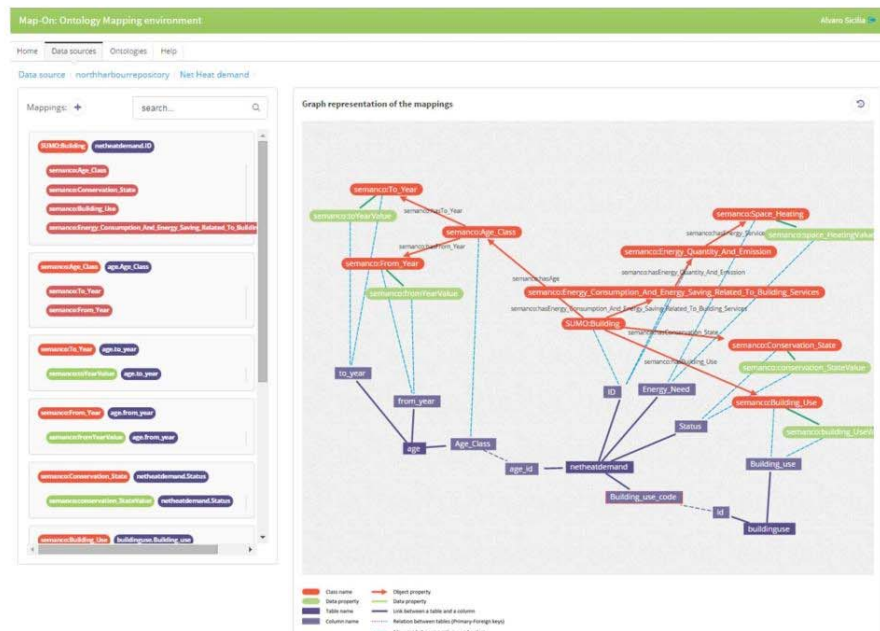


Fig. 3. Map-On main interface. On the left side, a list of mappings is presented on the 'Net Heat demand' mapping space. On the right side, an interactive graphical visualization of the mappings is displayed.

Moreover, Map-On facilitates the definition of mapping spaces. These spaces are partial views of an entire picture of mappings between an ontology and a database. Such spaces contain a limited set of ontology and database entities and serve to divide a complex mapping task into a set of less complex and smaller tasks. This feature is important in scenarios where there are a considerable number of elements involved in the mapping.

4.3. Automated generation of IRI patterns and SQL queries

The Map-On editor automatically generates the IRI patterns and logic tables (i.e., SQL queries) that are required by the R2RML statements. This is based on the concepts and columns involved in the mappings created by the user. For example, in Figure 3, when a user maps the concept *SUMO:Building* to the column *ID* of the table *netheatdemand*, the following

IRI and SQL query are generated for defining the subject map:

```
IRI: <base_iri>/building/{\"netheatdemand.ID\"}
SQL: SELECT buildinguse.Building_use, netheatdemand.Status, age.Age_Class, netheatdemand.ID FROM buildinguse, age, netheatdemand WHERE buildinguse.id = netheatdemand.Building_use_code AND age.Age_Class = netheatdemand.age_id
```

The IRI is generated based on patterned URIs solution [6]. This pattern was selected considering that people are able to read it and that it is easily generated from a database where identifiers (i.e., primary keys) are always present. Furthermore, the name of the concept is added to the base IRI. In this way, the problem of generating different individuals with the same identifier but different concept is mitigated. In the above example, the *<base_iri>* variable is common for all the mappings, the *building* comes from

the concept of the ontology and *netheatdemand.ID* is the column involved in the mapping.

The logic tables of the triples maps statements are defined as SQL queries which are automatically generated by the editor. The editor inspects the mappings created by the user for generating a valid SQL query and takes into account all the possible tables and columns involved in the mapping. In the above example, the SQL query has been generated for the mapping between the concepts *SUMO:Building* and the *ID* column of table *netheatdemand*. In the generation of the SQL query the mappings connected through object properties are taken into account, for instance the mapping between the concept *semanco:Age_Class* and the column *Age_Class* of the table *age*. The query retrieves data from three different tables (i.e., *buildinguse*, *age* and *netheatdemand*) which are connected by constraints defined in the “where” clause. Moreover, the columns involved in the mapping are also included in the “select” clause.

5. Architecture of Map-On

This section presents the general architecture of Map-On including the important aspects of every module (Figure 4).

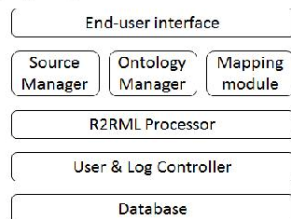


Fig. 4 Map-On basic architecture.

End-user interface. The visualization of the database and the domain ontology by means of a joint graph representation is one of the strong points of the interface together with the creation and modification of the mappings using point-and-click paradigm. Thus, users can change the layout by dragging the graph nodes making the visualization clearer. The interface provides suggestion lists of possible concepts roles, and attributes to be used in the mappings.

Moreover, the interface comprises pop-ups with tips as an integrated help.

Source manager. This module provides the methods for loading schemas of the input databases. The schema is provided as an SQL file. Every database has its own mapping spaces with their mappings for producing a R2RML document.

Ontology manager. The domain ontologies are stored in the database of the Map-On tool for increasing the query response performance. The ontology manager provides functionalities for loading an OWL ontology in RDF/XML format and their related ontologies. The module also implements methods for query specific concepts, roles, and attributes based on a text provided by the user. Finally, the module takes care of the prefixes of the ontology needed for representing the ontology elements by QNames (Qualified names).

Mapping module. This module is responsible for creating and storing the mappings defined by the user. Thus, it manages the mapping spaces. The mapping module implements the methods for the automatic generation of IRI patterns and the SQL queries described in the previous section.

R2RML processor. Based on the mappings created by the user, this module generates a document according to R2RML recommendation. The user can provide custom mappings which are stored in the Map-On database. Furthermore, the custom mappings are attached to the final R2RML document generated by the processor.

User & Log controller. This module tracks the user actions and stores them in the database. The actions are tagged with an identifier depending on where the action takes place. For instance, when a mapping is created between a concept and a column of a table, the action is tagged with the identifier of the mapping space to which the mapping belongs. Moreover, the actions are tagged with the identifier of the user who carries out the action.

Database. The different elements provided or created by the user are stored in the database, specifically the structure of the sources (i.e., table names, column names, column types, and foreign/primary keys), the ontology, and the mappings created. Thus, the graph layout configurations that are personalized by the user are also part of the database.

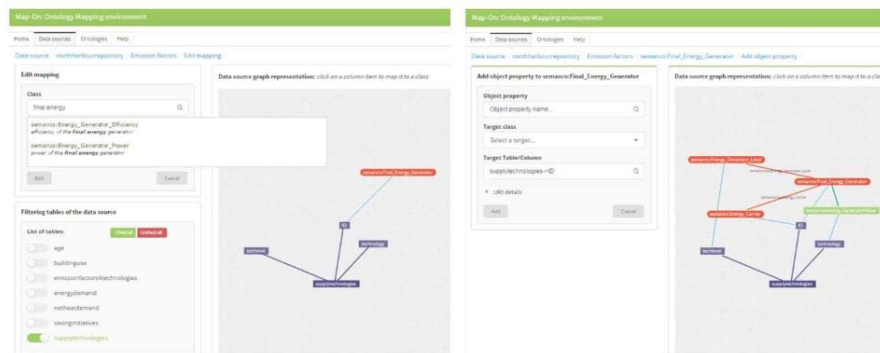


Fig. 5 Map-On interfaces. Left: new mapping creation interface. Right: Object property creation interface.

The modules have been developed in PHP using Code Igniter¹, an Open source PHP web application framework. The graphic visualizations of the source, the ontology and the mappings have been implemented using the VivaGraphJS², a graph drawing library for JavaScript library. The EasyRDF library³ has been used for parsing the ontology files and the Appmosphere RDF classes (ARC) library⁴ for the ontology storage using MySQL engine. The R2RML visualization and text editor use the Codemirror JavaScript library⁵ for the turtle syntax highlighting style.

6. Ontology mapping process with Map-On editor

Users of the Map-On⁶ editor can carry out an ontology mapping process in a user-friendly interface without worrying about dealing with R2RML coding.

The first step of the process is to load the input database and the ontology. The ontology and their imports are stored in the same place in order to reduce the querying time. In order to load the input database, an SQL schema file has to be provided together with the base IRI to be used in the IRIs of the subject and object maps of the R2RML statements.

The second step is to define the mapping spaces. Users are free to create any number of mapping spaces, usually parts of ontology that have something in common are mapped in the same space.

The third step is to create the mappings by clicking the plus button on the top-left side of the interface (Figure 3). In the new mapping page, the graph layout contains an empty node representing an ontology concept and the nodes characterizing the structure of the input database. The user can search for an ontology concept by typing in the input box (Figure 5, left screen). The editor will provide a list of possible concepts based on the input given by the user. Once the user selects a concept from the list, the node in orange takes the name of the concept. After that, the user can click a node of the graph representing a column and a mapping –a blue dashed line– will connect the concept and the column nodes. The tables of the database can be filtered in order to reduce the nodes in the graph visualization by clicking the checkboxes on the bottom-left side of the interface (Figure 5, left screen). After these actions the user has established a mapping between a concept and a column which corresponds to a subject map statement.

The next step is to further elaborate the mappings, particularly with regard to the creation of object map statements. The concepts are linked to other concepts, using roles (i.e., object properties) and related to attributes using data properties. The different interfaces are accessible through a dropdown menu which is shown when the cursor moves over the mapping list on the left-hand side of the interface. For an existing mapping, the user can create a role mapping which will correspond to an object map statement in the

¹ <http://codeigniter.com>

² <http://github.com/anvaka/VivaGraphJS>

³ <http://www.easyrdf.org>

⁴ <http://github.com/semsol/arc2>

⁵ <http://codemirror.net>

⁶ <http://semanco-tools.eu/map-on>

```

#####
# TripleMap for 133: http://semanco-tools.eu/ontology-releases/eu/semanco/ontology/SEMANCO/SEMANCO.owl#Final_Energy_Generator
<mapping1_133> a rr:TripleMap;
  rr:logicalTable [ rr:sqlQuery "SELECT supplytechnologies.technology, supplytechnologies.techlevel, supplytechnologies.ID FROM supplytechnologies" ];
  rr:subjectMap [ rr:template "http://www.semanco-project.eu/copenhagen/final_energy_generator/{\"supplytechnologies.ID\"}";
    rr:class <http://semanco-tools.eu/ontology-releases/eu/semanco/ontology/SEMANCO/SEMANCO.owl#Final_Energy_Generator>
  ];
  rr:predicateObjectMap [
    rr:predicate <http://semanco-tools.eu/ontology-releases/eu/semanco/ontology/SEMANCO/SEMANCO.owl#Energy_GeneratorValue>;
    rr:objectMap [ rr:column "supplytechnologies.technology" ]
  ];
  rr:predicateObjectMap [
    rr:predicate <http://semanco-tools.eu/ontology-releases/eu/semanco/ontology/SEMANCO/SEMANCO.owl#hasEnergy_Generator_Level>;
    rr:objectMap [ rr:template "http://www.semanco-project.eu/copenhagen/energy_generator_level/{\"supplytechnologies.techlevel\"}" ]
  ];
  rr:predicateObjectMap [
    rr:predicate <http://semanco-tools.eu/ontology-releases/eu/semanco/ontology/SEMANCO/SEMANCO.owl#hasEnergy_Carrier>;
    rr:objectMap [ rr:template "http://www.semanco-project.eu/copenhagen/energy_carrier/{\"supplytechnologies.ID\"}" ]
  ];
  .
#####
# TripleMap for 134: http://semanco-tools.eu/ontology-releases/eu/semanco/ontology/SEMANCO/SEMANCO.owl#Energy_Generator_Level
<mapping1_134> a rr:TripleMap;
  rr:logicalTable [ rr:sqlQuery "SELECT supplytechnologies.techlevel FROM supplytechnologies" ];
  rr:subjectMap [ rr:template "http://www.semanco-project.eu/copenhagen/energy_generator_level/{\"supplytechnologies.techlevel\"}";
    rr:class <http://semanco-tools.eu/ontology-releases/eu/semanco/ontology/SEMANCO/SEMANCO.owl#Energy_Generator_Level>
  ];
  rr:predicateObjectMap [
    rr:predicate <http://semanco-tools.eu/ontology-releases/eu/semanco/ontology/SEMANCO/SEMANCO.owl#Energy_Generator_LevelValue>;
    rr:objectMap [ rr:column "supplytechnologies.techlevel" ]
  ];
  .

```

Fig. 6. An excerpt of the R2RML document generated by Map-On

R2RML document. The user can search the role by typing in the input box (Figure 5, right screen). The editor will provide a list of possible roles whose domain is the previously mapped concept and which match the input text provided by the user. Once the role is selected, the graph visualization is updated accordingly and a target concept (i.e., possible ranges of the role) can be selected from the list provided by the editor. Later, the user can click on a column node to map it to the target concept. The process of linking attributes is similar to the one described for the roles.

Finally, when the mappings are created, the R2RML document can be visualized using a turtle syntax highlighting library (Figure 6). Furthermore, users can write their own mappings. In addition, the R2RML document can be downloaded as a text file.

7. User testing

We conducted a user study with five people to identify usability issues of the tool and to demonstrate that it can be used by non-ontology experts. Participants had a similar profile, they were graduates and post-graduates, experts in data base theory who use SQL language in their professional activities. The participants did not have any notions of the Semantic Web concept and nor were they familiar with languages such as OWL, RDF, SPARQL, and R2RML. Two participants were professors of an engineering school, two participants were industry professionals, and one participant was from the research

community. They took part in the test voluntarily, with no payment for their involvement.

7.1. Test design

The test was carried out in a Toshiba Z830 laptop (1.6 GHz, i5, 4GB RAM) with a mouse and a stopwatch. The screen and the voice of the participants were recorded. The participants were informed of this beforehand.

The test has been administered by just one person. All testing sessions were approximately fifty minutes long. The participants were given a brief introduction to the Semantic Web. The concept of ontology was explained as well as the main features of the Resource Description Format. Furthermore, a R2RML mapping was shown to users. Finally, the purpose of the Map-On tool and its main features were introduced to the participants.

A pre-test questionnaire was submitted to assess participant's expertise on relational databases and Semantic Web technologies. After completing the tasks, the participants filled a satisfaction questionnaire aimed at measuring their subjective satisfaction with regard to the Map-on tool interface. Participants answered a five-level Likert scale. Participants were asked to use the tool while continuously thinking out loud. That is, verbalizing their thoughts as they were carrying out the tasks.

The domain of the tasks was research conferences (e.g., authors, committees, papers, abstracts...). This domain is easy to understand for the participants without teaching them basic concepts and their inter-

relations. Both the ontology and the database were taken from the RODI benchmark [18] and were already uploaded to the tool. The user test was composed of three tasks, each involved mapping a class of the domain ontology and an element of the database. The tasks were designed to carry out mappings which ranged from simple to complex. The tasks were to 1) *relate authors*, 2) *relate authors with the submitted papers*, and 3) *relate conferences with their committees*. Tasks were carried out sequentially and the participants had to confirm the task finalization by themselves without the validation of the test administrator.

The usability metrics used to evaluate the test were the effectiveness metric with the accuracy measure – percentage of tasks correctly completed– and the efficiency metric with the completion time which is the time taken to complete the tasks.

7.2. Results

Figure 7 summarizes the overall results for each usability metric obtained in the three tasks. The accuracy was obtained by calculating the sum of the participants that successfully completed the task divided by the number of participants. The completion time was calculated as the time mean in minutes that each participant took to complete the task.

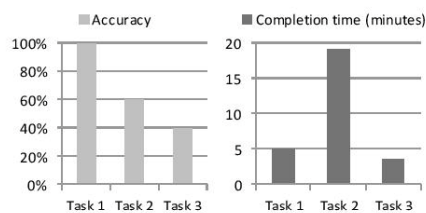


Fig. 7. Accuracy and completion time results of the three tasks

The accuracy for Task 1 was 100%, for Task 2 60% and for Task 3 40%. The difference in the results was related to the complexity of the task. Task 1 involved the creation of a simple mapping while in Task 2 and Task 3 the participant had to create a complex mapping including relating different tables and columns. Additionally, the mappings to create in Task 2 were based on the mappings produced in Task 1.

The completion time for Task 1, was 5 minutes, for Task 2 was 19.1 minutes, and for Task 3 was 3.7 minutes. The difference between the completion time

of Task 1 and 3 lies in the lack of knowledge of how to work with ontologies –in particular when referring to the creation of object properties to connect concepts of an ontology– and how to use the tool since they were not trained beforehand. The completion time of Task 3 was slightly below the time of Task 1 because participants have learned how to use the tool.

The average completion time for Task 2 was much higher than the other two tasks. The main reason was that participants found it difficult to create object properties because some messages and warning displayed in the user interface were not clear enough. Indeed, some participants felt confused and lost. Even so, the participants completed quicker Task 3 which also implied object property creation because they learnt how to use the tool.

The user satisfaction was measured through a post-test questionnaire based on the System Usability Scale [3] where the participants rated some subjective statements with a five-item Likert scale (from 1-completely disagree to 5-completely agree). This was done after finishing all the tasks. Results of the user satisfaction are obtained by calculating the mean of responses per answer and are summarized in Figure 8. Ratings clearly show that Map-On is not perceived as a complex tool. Moreover, it is recognized that the tool can be learnt by most people. However, the participants neither agree nor disagree on how confident they feel using the tool. Finally, participants observed that there is no need to learn a lot of things before using the tool.

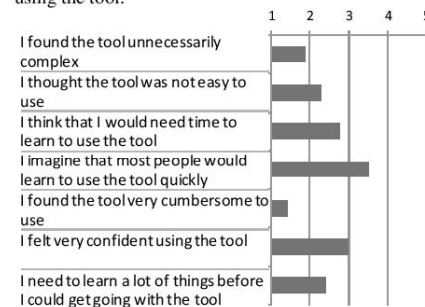


Fig. 8. User satisfaction results

7.3. Observations

A list of observations was detected by the administrator during the test and through an analysis of the voice transcriptions and screen recordings.

Most of the participants stressed that the process of mapping creation should be more guided than it is now. After navigating through the interface, for a while some of them felt a bit lost. This feeling was stronger in the case of a task which could be done in several ways.

The representation of the database schema in a graph basis was a bit confusing for some users since they were used to working with relational views. Two participants highlighted the need of an optional representation of relational tables such as the traditional tools. Moreover, the representation of tables and columns were very similar and therefore, brought confusion to some participants.

All participants changed the layout of the mappings by dragging the graph nodes in a natural way. However, some of them felt that some common features usually offered by drawing tools, such as multi selection of nodes, were missing.

Some participants missed a visual representation of the whole ontology where they could see how the concepts are actually related.

Minor issues related to the interface were raised during the test. Help texts in the pop-ups were not enough clear, error descriptions when the form was not properly completed, and suggestion lists were only displayed when the user typed about it and are not available if the user wishes to browse all the content.

The overall conclusion is that the Map-On editor can be used by non-ontology experts to manually generate mappings between a database and an ontology without acquiring a formal knowledge about several Semantic Web technologies such as OWL, RDF, and R2RML. Although the number of the complex tasks successfully completed were not as good as in the simple ones, the tool is easy to learn. That is, the completion times went down in the last task. Moreover these results can be improved with a previous training session where the tool is presented to the user in a detailed way. This was not done so as not to contaminate the test.

8. Experience on a real scenario

In order to demonstrate the usefulness and the feasibility of the Map-On editor, we outlined a real scenario in which the editor has been used to create R2RML documents.

The SEMANCO Integrated Platform⁷ provides access to energy related data obtained from multiple sources and domains which is generated by different users and applications. In this platform, assessment, simulation and analysis tools interact with the semantically modelled data. The information is visualized in 3D models, diagrams and tables to make it understandable to different user profiles. The platform has been developed within the FP7 SEMANCO⁸ project and has been tested in three case studies: Newcastle in UK, Copenhagen in Denmark, and Manresa in Spain.

Each city has different data sources which comprise two kinds of data: a) data regarding building typologies which provides statistical data about their physical properties and energy performance characteristics; and b) urban data which provides contextual information of the areas at a city and a neighbourhood scale.

The platform makes the data sources and tools interoperable thanks to the semantic energy information system (SEIF) whose main component is the federated query engine ELITE [15] built on top of Quest reasoner [21]. In the core of SEIF lies a global ontology coded in *OWL 2 QL*⁹, which is based on the DL-Lite family (we focus here especially on the variant *DL-Lite_A*) and has been designed for efficient reasoning and query answering tasks in the context of OBDA [21].

Map-On is used in this scenario for mapping the three databases to the global ontology. The R2RML documents generated through the editor are used for accessing the contents of the database through the SEMANCO ontology using Sesame/Quest instance [13]. Table 2 comprises the figures of the mappings generated for these databases. Specifically, a total number of 221 triples maps are generated with a total of 307 object maps. The Map-On editor is also used to update and enhance mappings during the life-cycle of the platform.

Thanks to the Map-On editor it is possible to create all of these mappings with a reduced effort since no SQL query have to be designed manually. As a matter of fact, it is the graphical visualization of the mappings that helps to understand, evaluate, and correct them.

⁷ www.eecities.com

⁸ www.semanco-project.eu

⁹ www.w3.org/TR/owl2-profiles/#OWL_2_QL

Table 2
Overview of the use case features

Number of:	Newcastle	Copenhagen	Manresa
Tables	3	7	23
Columns	33	28	118
Constraints	0	5	7
Subject maps	26	30	165
Object maps	38	45	224

9. Conclusions

In this paper we presented Map-On, a graphical environment for ontology mapping which supports different kinds of users to manually establish relations between the elements of a database and a domain ontology in context of an OBDA scenario. The ontology mapping editor has five distinctive features:

- Point-and-click interface for reducing the mapping activities effort.
- Ontology-driven mapping approach, where the mapping process starts from the ontology instead of working with the database.
- Top-down visualization, for representing the whole database schema, ontology structure, and mappings using an interactive and a joint graph layout.
- Mapping spaces, where the mappings are freely grouped by the users in order to keep mappings tidy.
- R2RML mappings automatically generated from the actions carried out by the users in the graphical interface.

Map-On has been validated through their application in the data integration process of the SEMANCO project. However, the tool is generic enough to be applied to other OBDA scenarios.

We plan to further develop Map-on with the following features:

- Implementing methods to automatically suggest mappings using the database and the domain ontology as inputs. In a later stage, users, thanks to the ontology mapping interface, would correct and extend the proposed mappings.
- Enabling the representation of complex SQL queries to be compatible with the current mapping visualization.
- Integrating an existing OBDA system to evaluate the mappings generated by the editor. In this

way, users can check in real-time whether the mappings are producing the RDF outputs they expect or not.

- To speed up the learning curve and to reduce confusion among the users, the interface should be slightly enhanced following the wizard paradigm to guide users in the mapping creation.

Acknowledgments

This work has been supported by the project of SEMANCO which is being carried out with the support of the Seventh Framework Programme “ICT for Energy Systems” 2011-2014, under the grant agreement no. 287534.

References

- [1] N. Antonioni, F. Castanò, C. Civili, S. Coletta, S. Grossi, D. Lembo, M. Lenzerini, A. Poggi, D.F. Savo, and E. Virardi. Ontology-Based Data Access: The Experience at the Italian Department of Treasury. In V. Pelechano, G. Regev, and Y. Pigneur, editors, *Proceedings of the Industrial Track of the 25th International Conference on Advanced Information Systems Engineering (CAISE'13)*, Valencia, Spain, June 21, 2013, volume 1017 of CEUR-WS, pages 9–16. ceur-ws.org, 2013.
- [2] P. Bellini, M. Benigni, R. Billero, P. Nesi, and N. Rauch. Km4City ontology building vs data harvesting and cleaning for smart-city services. *Journal of Visual Languages & Computing*, 25(6):827–839, 2014. 10.1016/j.jvlc.2014.10.023.
- [3] J. Brooke. SUS-A quick and dirty usability scale. *Usability Evaluation in Industry*, 189(194):4–7, 1996.
- [4] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodríguez-Muro, R. Rosati, M. Ruzzi, and D.F. Savo. The Mastro system for ontology-based data access. *Semantic Web*, 2(1):43–53, 2011. 10.3233/SW-2011-0029.
- [5] C. Civili, M. Console, G. De Giacomo, D. Lembo, M. Lenzerini, L. Lepore, R. Mancini, A. Poggi, R. Rosati, M. Ruzzi, V. Santarelli, and D.F. Savo. MASTRO STUDIO: Managing ontology-based data access applications. *Proceedings of the Very Large Database Endowment (PVLDB)*, 6(12): 1314–1317, 2013. 10.14778/2536274.2536304.
- [6] L. Dodds and I. Davis. Linked Data Patterns, A pattern catalogue for modelling, publishing, and consuming Linked Data. <http://patterns.dataincubator.org/book/linked-data-patterns.pdf> (accessed July 29, 2016).
- [7] B. Fu, N.F. Noy, and M.A. Storey. Indented tree or graph? A usability study of ontology visualization techniques in the context of class mapping evaluation. In H. Alani, L. Kagal, A. Fokoue, P. Groth, C. Biemann, J.X. Parreira, L. Aroyo, N. Noy, C. Welty, and K. Janowicz, editors, *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part I*, volume 8218 of the series Lecture Notes in Computer Science, pages 117–134. Springer, 2013. 10.1007/978-3-642-41335-3_8.
- [8] P. Heyvaert, A. Dimou, A.L. Herregodts, R. Verborgh, D. Schuurman, E. Mannens, and R. Van de Walle. RMLEditor: A

Aquesta Tesi Doctoral ha estat defensada el dia ____ d_____ de 201__
al Centre_____

de la Universitat Ramon Llull, davant el Tribunal format pels Doctors i Doctores
sotasignants, havent obtingut la qualificació:

President/a

Vocal

Vocal *

Vocal *

Secretari/ària

Doctorand/a

(): Només en el cas de tenir un tribunal de 5 membres*