



OMWG D7.1: Requirements for Mapping and MergingTool

DERI OMWG Working Draft 31 August 2005

This version:

<http://www.omwg.org/TR/d7/d7.1/v0.2/20050831/>

Latest version:

<http://www.omwg.org/TR/d7/d7.1/v0.2/>

Previous version:

<http://www.omwg.org/TR/d7/d7.1/v0.1/20041206/>

Authors:

Fancois Scharffe

Editors:

Francois Scharffe, Atanas Kiryakov

Copyright © 2004 [DERI](#)®, All Rights Reserved. [DERI](#) liability, trademark, document use, and software licensing rules apply.

this document is available as a non normative [PDF](#) version

Table of contents

[1 Introduction](#)

[2 Tool Requirements](#)

[1.1 Interoperability and compatibility](#)

[1.2 Genericity](#)

[3 Functional Requirements](#)

[3.1 Ontology Alignment Language Requirements](#)

[3.2 Mapping Patterns](#)

[3.3 Graphical mapping interface](#)

[3.4 Mapping algorithms programming interface](#)

[4 Implementation Priority List](#)

5 Global Architecture

6 Conclusion

7 Acknowledgements

References

1 Introduction

Ontology Mapping is one of the pillars of an ontology management system. The mapping component is used to link different ontologies having an overlap in their domain description. The mapping tool will enable the communication process between these shared representations that ontologies are. This task consists of linking the different entities of the overlapping ontologies and to model these links as axioms in the ontology representation language. We will in the following of this document what a complete ontology mapping tool should provide as functionalities.

2 Tool Requirements

2.1 Interoperability and compatibility

The Ontology management suite is composed of different parts that form a complete ontology management system. The ontology mapping module interacts with these different parts, giving some first requirements to its design.

The ontology mapping module communicates with the ontology editor. The mappings that are realized between ontologies are implying the edition of these ontologies in order to visualize them and decide which entities (concepts, attributes, instances) have to be related. A strong coupling with the ontology editor is then necessary for this visualization to be effective. We may there envisage two possible realization. in the first case, the mappings functionality may be an embedded feature of the ontology editor. In that case, the user may realize mappings by opening different ontologies in different windows. In the second case, mapping editor is a separate component that invoques some features of the ontology editor in order to display the concept tree of the ontologies to be related. For more informations about the ontology editor, please have a look to the working drafts D8.X on [this page](#).

The ontology mapping module communicates with the ontology repository. In order to realize the mappings, the user should first be able to open the ontologies to be mapped. These ontologies are stored in an ontology repository which is part of the Ontology Management System (OMS). The ontology mapping module should then be able to communicate with the ontology repository, to retrieve the stored ontologies that will be used to create the mappings. The same happen when the user open a previously stored mapping (see next point), the mapping module should automatically retrieve the ontologies associated with this module and open them using the editor. For more informations about the ontology editor, please have a look

to the ontology repository fact sheet on this page.

The ontology mapping module communicates with the ontology mapping store. Once the mappings created, they need to be stored. An ontology mapping store should then be introduced as part of the OMS, giving functionalities to store the created mapping and retrieve them in a convenient automatic way. When retrieving these stored mappings the user may want to visualize or modify them. For more informations about the ontology editor, please have a look to the ontology mapping store fact sheet on

The ontology mapping module communicates with the ontology versioning support module. Ontologies are dynamic objects that evolve over time as the domain they conceptualize evolves. The versioning module is taking care of keeping the trace of this evolution. On a conceptual level, it may be considered that two slightly different ontologies and the an ontology on which the user has performed small changes are very similar. In other words the changes made to an ontology may be represented as a mapping between an ontology and itself. For more informations about the ontology versioning support, please have a look to the working drafts D6.X on this page.

2.2 Genericity

An important feature of the DOME mediation component will be the reusability of the mappings generated by the users in other tools (reasonner, semantic web services builder, etc). In that purpose, we add as a requirement the commitment to an abstract mapping representation language, independent with the ontology language. These mappings will of course have to be grounded at the run-time to the language of the ontologies they are dealing with.

The research in the mediation field is mainly concentrated on the automation of the mapping task. This research is active and involve different areas like graph theory or computational linguistics. As a result new algorithm realizing automatic or semi-automatic ontology mappings are continuously proposed. To get the advantage of these new algorithms, the DOME mapping tool needs an architecture that permits to plug the automation algorithms and select an appropriate one for a specific mapping task.

3 Functional Requirements

In this section we elaborate some the requirements of the DOME mapping tool related with the functionalities we will offer. As specified in the last requirement, we impose the use of an abstract mapping language. For this language and a study of the mappings that are recurrently created by the user, we propose a library of mapping patterns. This library helps the user to select the appropriate mapping and then considerably reduce the time spent on the mapping task. The time spent on the mapping task will also be reduced by using automatic or semi-automatic ontology mapping algorithms. We also consider to present to the user a graphical mapping interface with the concepts trees of the ontologies to be mapped. We will now describe this different functionalities in more details.

3.1 Ontology Mapping Language Requirements

The core of the mapping tool is the abstract mapping language. It gives a format to represent the mappings at an abstract level, independent from the ontology language used. We will in the following try to give a list of requirements [De Bruijn and Polleres 2004] that such a language must fulfill to be efficient.

Mapping on the semantic web Our goal is to develop a mapping language for the semantic web. As different ontologies representation language are in use in the semantic web, the W3C standard Ontology Web Language (OWL) [Patel-Schneider et al 2004] or the Web Rule Language (WRL) [de Bruijn et al 2005] to cite only those. The semantic web mapping language must give the possibility to realize mapping between ontologies written in these languages. For this we propose the use of an abstract mapping language ontology-language independent that may be grounded to different ontology languages depending on the needs of the user.

Specify instances transformations The mapping process is specified in [Rahm and Bernstein 2001] as the set of activities required to transform instances of the source ontology into instances of the target ontology. The mapping language must then give the necessary constructs to express this transformation. In instance transformation, we identify two dimensions: structural transformation and value transformation:

A *structural* transformation is a change in the structure of an instance. This means that an instance might be split into two instances, two instances might be merged into one, properties might be transformed to instances, etc. For example an instance of the concept PdD-Student in one ontology might need to be split into two instances, one of Student and one of Researcher, in the target ontology. Another example is the use of aggregate functions (see [functional mappings](#)). An ontology O_s might have a concept Parent with a property hasChild, whereas the ontology O_t might also have a class Parent, but in this case only with the property nrOfChildren. An aggregate function is required to count the number of children in O_s in order to come with a suitable property filler for nrOfChildren.

A *value* transformation is a simple transformation from one value into another. An example of such a value transformation is the transformation from kilograms into pounds.

An example of a transformation, which requires both a structural and two value transformations is the transformation from a full name to separate first and last names. Splitting the full name into both the first and the last names requires structural transformation. After the structural transformation, two value transformations are required; one for the first and one for the last name.

Query Rewriting and ontology merging A query addressed to an ontology O_s has to be rewritten in terms of the ontology O_t in order to get results both from O_s and O_t . This use case indicates the need for the ontology mapping to not only map instances of the ontology but also map concepts and relations in the source and target ontologies. This is necessary for the case when a query written in terms of an ontology O_s must be executed on an instance base, which is described by O_t . The mapping needs to specify exactly how concepts and relations in O_s relate

to concepts and relations in Ot in order to enable the rewriting. The mapping language is then required to provide the possibility of expressing the correspondences between the concepts and relations of the two ontologies.

Versioning support The ontology mapping language must support constructs for the versioning of the mapping and for referring to specific versions of the source and target ontologies. As this task is concerning the Versioning module of DOME we refer you to the Versioning working drafts D6.X [here](#).

treating classes as instances Different ontologies might be modeled within slightly different domains with different granularity. What is seen as a class in one ontology might be seen as an instance of another class in another ontology. In order to support inter-operation between two ontologies with such differences, class need to be mapped to instances and vice-versa. In a more general way, the mapping language should support mapping of any entity in the source ontology to any entity in the target ontology. For example it should be possible to have a relation instance mapping, a class-relation mapping, etc.

Mappings of different cardinalities It might be necessary to map a class in one ontology to a number of different classes in the other ontology. It might also be necessary to map a class in one ontology to a class and a relation in the other ontology. In other words, the language needs to support mappings of arbitrary cardinalities.

Conditionnal mappings In some cases the mapping relation between an entity of the source ontology and another entity of the target ontology may apply only if ceretain condition are satisfied. For example the concept human in the source ontology is mapped to the concept man in the target ontology only if the attribute gender of the concept human is equal to male. This conditional mapping should be expressed in the mapping language.

3.2 Mapping Patterns

Patterns are templates that match the more usual mismatches between two ontologies. The use of predefined patterns considerably reduces the mapping designer task. These patterns should be expressed in the mapping language, they may be seen as building blocks for this language. A pattern is instanciated for a particular pair of ontologies into an ontology mapping, a complex mapping pattern is composed of other (elementary or complex mapping patterns) and a mapping is composed of a number of instancisted mapping patterns.

The ontology mapping language must supports the use of elementary patterns as constructs of the language itself. An elementary pattern specifies the relation between one or more ontology constructs of the source ontology and one or more ontology constructs of the target ontology. The mapping pattern describes both the relation and the necessary instance transformation in a declarative way. When the mapping pattern is instanciated to a mapping relation, the abstract ontology constructs in the patterns are filled-in with the actual ontology constructs from the source and target ontologies. The relation and the transformation can be further refined.

An additionnal benefit that these mapping patterns could bring is when mapping patterns would

be associated with similarity detection methods. A Match operator could then detect mapping patterns based on certain similarities between the different ontologies.

A number of mapping patterns will be proposed, they will be accessible in a library having retrieval functions. For more information on the pattern library, please have a look to the ontology mapping store fact sheet on

3.3 Graphical Mapping Interface

Mapping between two ontologies consists in relating the corresponding entities from one ontology to the other. The abstract mapping language ([section 3.1](#)) gives a format to express the complex relation that may stand between these entities. The user of the DOME mapping tool will have the possibility of creating mappings by directly entering constructs of the language, eventually using patterns. For ergonomics the DOME mapping tool will also propose a graphical interface for this task. An ontology mapping graphical interface is commonly constituted of two representations of the ontology to be mapped (see for example [\[Noy and Musen 2003\]](#)). For more information about the used representation of the ontology, see the ontology editing module of DOME, working drafts D8.X on [this page](#). The user selects the correspondent entities in the two ontologies to realize the mapping. When the relation standing between the two ontologies is complex, the user may select a convenient pattern and apply it to the mapping case.

3.4 Mapping Algorithms Programming Interface

The large ontologies like Proton [\[Terziev et al 2005\]](#) or Wordnet [\[Fellbaum 1998\]](#) contain thousands of concepts and relations. It is obviously a complex and time consuming task to map two ontologies of these sizes, almost each concept and relation in the source ontology having a correspondent that must be determined in the target ontology. In that scope, the assistance of an automated method to (semi-)automatically find the correspondences between the entities of two ontologies would be very welcome. As already mentioned in [section 2.2](#), the research on these methods is very dynamic and different algorithms are trying to cope with this problem.

There are different techniques used to automatically find mappings between ontology schemas. A good classification of these techniques is given in [\[Shvaiko 2004\]](#). The techniques are classified in two categories: Element-level and Structure-Level. The former technique uses the labels and descriptions of the ontological entities and compares them, eventually using some external tools to measure the distances between two elements from two ontologies. An example of a method element-level based method would be to compare the labels of the two ontologies pair by pair using a string comparison method. The latter technique consists in analysing the structure of the ontological schema, by matching parts of the concept tree for example. These techniques are then resulting similarity measures between the instances of two ontologies, these measures are then proposed to the user which validates the effective mapping.



The DOME mediation module will have an interface on which such automation algorithms may

be plugged. This interface takes one or more algorithms as an entry together with the indication about which technique of the classification the algorithm is actually using. The algorithms may then be combined to cover the different techniques and compute better quality mappings, ideally fully automatized.

4 Implementation Priority List

For the implementation we have distinguished three phases:

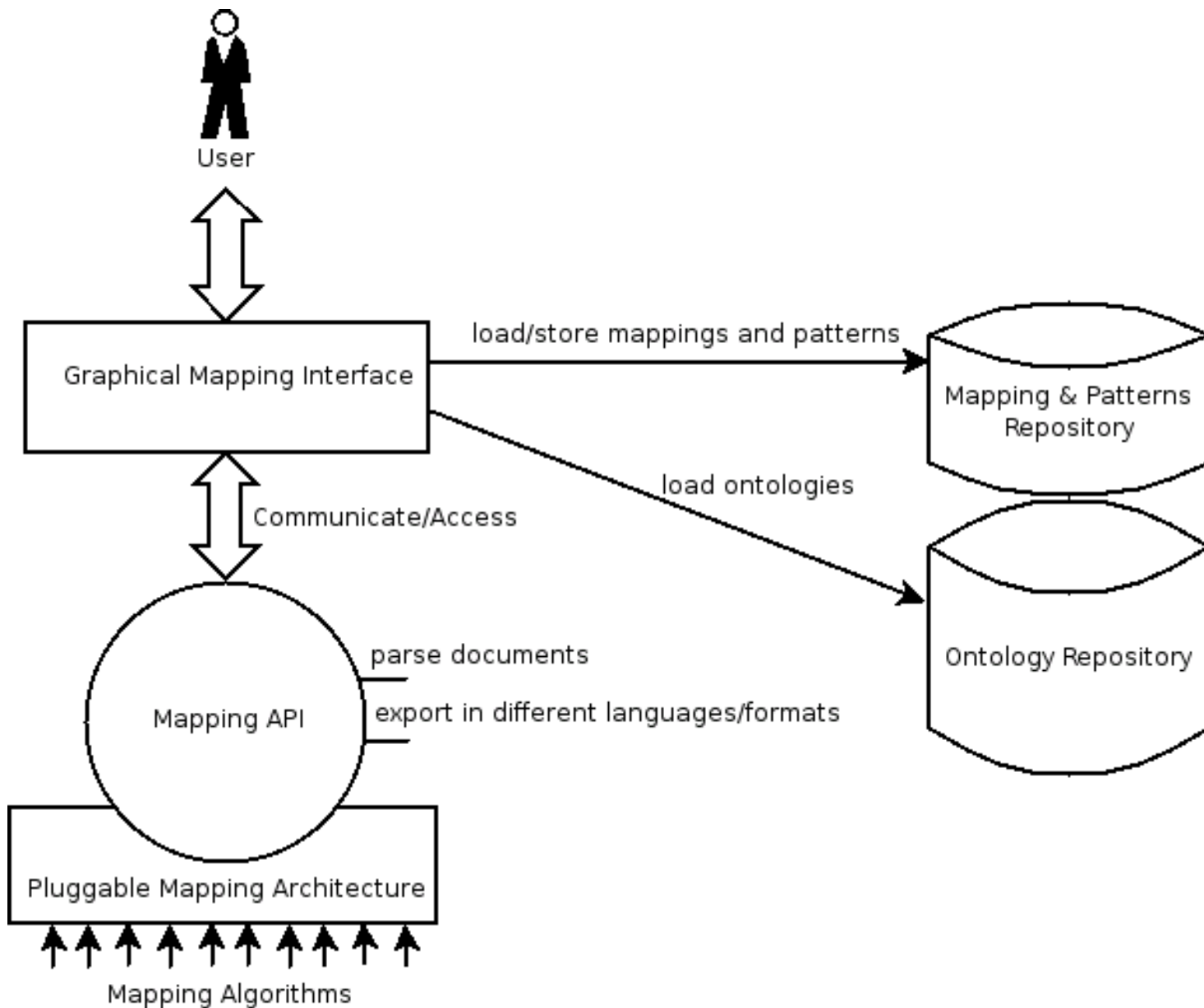
- Version 1: From M13 to M18
- Version 2: M19 to M24
- Version 3: after M24.

The Vs indicate in which phase which requirement is being initially tackled.

Req. ID.	Mapping tool Requirements	Version 1	Version 2	Version 3	Priority
V1	Interoperability/ Compatibility	V			(affects all implementation)
V2	Genericity	V			(affects all implementation)
V3	Mapping Language	V			HIGHEST
V4	Mapping Patterns		V		HIGH
V5	Graphical Mapping Interface	V			HIGHEST
V6	Merging function			V	LOW
V7	Links to the repository		V		HIGH
V8	Mapping Algorithms API		V		HIGH
V10	Mapping Algorithms Research			V	LOW

5 global architecture

Here is the rough architecture diagram of the DOME mapping tool, including the required components presented in this document.



6 Conclusion

The DOME mediation component requirements are positioning it as a top level of the ontology mapping tool. Ontology mediation is a key process that enable interoperability and communication between different semantic bubbles in the semantic web. The Ontology Management Working Group will spend considerable efforts to bring this module to a commercial use maturity as part of the DOME project.

7 Acknowledgement

The work is funded by the European Commission under the projects DIP, Knowledge Web, Ontoweb, SEKT, SWWS, Esperonto and h-TechSight; by Science Foundation Ireland under the DERI-Lion project; and by the Vienna city government under the CoOperate programme.

The authors would like to thank to all the [members of the OMWG working group](#) for their advices and inputs to this document.

References

- [De Bruijn and Polleres 2004]:** Jos de Bruijn, Axel Polleres. Towards an Ontology Mapping Language for the Semantic Web. DERI Technical report, June 2004.
- [De Bruijn et al 2005]:** Jos de Bruijn et al. Web Rule Language (WRL). <http://www.wsmo.org/wsml/wrl/>, 2005
- [Fellbaum 1998]:** Christiane Fellbaum ed. Wordnet: An Electronic Lexical Database. MIT Press, 1998
- [Noy and Musen 2003]:** N.F. Noy and M.A. Musen The PROMPT Suite: Interactive Tools For Ontology Merging And Mapping International Journal of Human-Computer Studies, 59/6 pp. 983-1024. Available as SMI technical report SMI-2003-0973
- [Patel Schneider et al. 2004]:** Peter F. Patel-Shneider, Patrick Hayes, and Ian Horrocks. OWL web ontology language semantics and abstract syntax. Recommendation 10 February 2004, W3C, 2004
- [Rahm and Bernstein 2001]:** Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. VLDB Journal: Very Large Data Bases, 10(4):334-350, 2001
- [Schvaiko 2004]:** A Classification of Schema-Based Matching Approaches. In Proceedings of Meaning Coordination and Negotiation Workshop at ISWC'04, 2004
- [Terzief et al 2005]:** Ivan Terzief and Antanas Kiryakov and Dimitar Manov. Base Upper Level Ontology Guidance. SEKT EU project deliverable D1.8.1, 2005



\$Date: 2004/10/22 16:12:55 \$

[webmaster](#)