Ontology Mediation in WSMX

Adrian Mocan

Digital Enterprise Research Institute (DERI)
National University of Ireland, Galway, Ireland
adrian.mocan@deri.org

1 Introduction

Ontology to ontology mediation is one of the first steps that have to be taken in order to cope with data and information heterogeneity. This paper shortly describes a mediation component [1], part of Web Service Modeling Execution Environment (WSMX) [2], able to provide data mediation by transforming a set of given instances of a source ontology in instances of the target ontology. The input ontologies conform to Web Service Modeling Ontology (WSMO) [3] conceptual model for ontologies, all applied strategies making use of the advantage of having the same meta-level for both source and target ontologies.

We are focusing here on a simple scenario: two enterprises decide to become partners in a business process, which implies the changing of a sequence of messages (e.g. purchase orders and purchase order acknowledgements). Each of these messages is represented in terms of the sender's ontology, containing instances of used ontology, and each of the business partners understands only messages expressed in terms of its own ontology. The role of mediation is to transform, if necessary, the received messages from terms of sender's ontology in terms of receiver's ontology.

2 WSMX Mediation

The WSMX mediation component has three main parts, one used during the design time and the other two used during runtime. A set of mappings are created by domain experts (using a mapping tool), automatically transformed in mapping rules and executed in a proper execution environment. Each of these three functionalities is accomplished by a different subcomponent described in more details in the followings.

Mapping Tool. The mapping tool is a graphical user interface, used at design time for creating a set of mappings between two given ontologies, each mapping identifying a pair of similar entities (concepts or attributes) from the two ontologies. That is, our aim is to minimize the human user efforts, by offering a graphical interface that better illustrates the problem he has to solve, the effects of his actions and the obtained results. Furthermore, a set of strategies are applied for guiding the user through the whole mapping process and for offering suggestions in order to reduce his decisions to simple choices or validations.

The obtained result is a set of mappings between the two ontologies, expressed in a language independent way, which are stored in an external storage for further use.

Mapping Rules Creator. This component is used during runtime for transforming the mappings in mapping rules. The main distinction between mappings and mapping rules is that the first are language independent while the former are expressed in a certain language (in our case Flora2[1]). We can say that the mapping rules express the mappings, in an executable way. As a consequence, another distinction between mappings and mapping rules is that while mappings express the similarities between the two ontologies, the mapping rules describe how these similarities are used in order to transform instances of the first ontology in instances of the second ontology.

This separation in mappings and mapping rules assures a high degree of flexibility: the mapping rules can be expressed in the appropriate language suited for the available execution environment.

Rules Execution Environment. The last step of the mediation process consists of the execution of the mapping rules. During runtime, the mapping rules are received from the mapping rules generator and executed inside the *Rules Execution Environment* – in our case a Flora2 environment. Of course, for assuring caching facilities, the mapping rules could be also stored in an external storage, but this approach would introduce here some of the consistency problems that may appear in the case of evolving/changing ontologies (without storing mapping rules the consistency problems would be addressed by acting only on the generated/stored mappings).

3 Conclusions

The WSMX Mediation Component has a strongly decoupled architecture, offering a high degree of flexibility. Each of these subcomponents may be easily changed or replaced with other components that offer the same functionality, to accommodate for example, the potential language and representation differences. The whole mediation component has a well defined interface, allowing the easy integration in other information systems.

References

1. A. Mocan, E. Cimpian: D13.3v0.1 WSMX Mediation. WSMO Working Draft v0.1, available from http://www.wsmo.org/2004/d13/d13.3/v0.1/20040628
2. E. Oren, M. Zaremba, M. Moran: Overview and Scope of WSMX, WSMO Working Draft v01, 2004, available at http://www.wsmo.org/2004/d13/d13.0/v0.1/20040611/
3. D. Roman, U. Keller, H. Lausen (eds.): Web Service Modeling Ontology - Standard (WSMO - Standard), version 0.2, 2004, available at http://www.wsmo.org/2004/d2/v0.2/

[1] http://flora.sourceforge.net/