TOWARDS SEMANTIC INTEROPERABILTY
In-depth comparison of two approaches to solving Semantic Web Service Challenge mediation tasks

Maciej Zaremba, Tomas Vitvar, Matthew Moran
Digital Enterprise Research Institute National University of Ireland, IDA Industrial Estate, Lower Dangan , Galway, Ireland
{*firstname.lastname@deri.org*}

Marco Brambilla*, Stefano Ceri*, Dario Cerizza†,
Emanuele Della Valle†, Federico M. Facca*, Christina Tziviskou*
**Dipartimento di Elettronica e Informazione, Politecnico di Milano, Milano, Italy*
{*mbrambil, ceri, facca, tzivisko*}*@elet.polimi.it*
†*CEFRIEL, Milano, Italy*
{*cerizza, dellava*}*@cefriel.it*

Keywords: Semantic Web, Web services, B2B integration, business process, workflow

Abstract: This paper overviews and compares the DERI and DEI-Cefriel approaches to the SWS-Challenge workshop mediation scenario in terms of the utilized underlying technologies and delivered solutions. In the mediation scenario one partner uses RosettaNet to define its B2B protocol while the other one operates on a proprietary solution. Goal of the workshop participants was to show how could these partners be semantically integrated.

1 INTRODUCTION

This paper compares two different approaches to semantic integration of a RosettaNet-enabled client with legacy systems in the context of the Semantic Web Services Challenge (SWS-Challange) [1] workshop series. Here we compare the submissions of the Digital Enterprise Research Institute[2] and the joint team DEI[3] and Cefriel[4] to the mediation problem.

The solutions of both groups differ quite substantially in terms of the underlying technologies. The DERI team based its solution on WSMO conceptual framework for Semantic Web services which comes from the quite young Semantic Web research area while DEI-Cefriel followed the path of well-established Software Engineering methods. We compare the similarities and differences of provided solutions mainly with respect to the data and process modeling, execution environments, tool support and changes required in the solutions once the integration requirements change.

The paper is structured as follows. First we overview our approaches, in section 2 the DERI team submission is described while in section 3 that of DEI-Cefriel is described. Section 4 provides in-depth comparison of our submissions. In section 5, we provide reference points to other works in the area of semantic integration. Finally, in section 6 we describe our further plans and conclude this paper.

[1] http://www.sws-challenge.org
[2] http://www.deri.org
[3] http://www.elet.polimi.it
[4] http://www.cefriel.it

2 Solving the Service Mediation Scenario with WSMX

In order to address the SWS-Challenge requirements, DERI based its solution on the specifications of WSMO (Roman et al., 2005), WSML (Roman et al., 2005) and WSMX (Haller et al., 2005) providing a conceptual framework, ontology language and architecture for Semantic Web services.

2.1 Environment

The following artefacts have to be created during the design time phase to apply WSMX middleware to the system integration: ontologies for both involved parties (i.e. service requestors and providers), XML<->WSML adapters and lifting/lowering rules, WSMO Goals and Services, data mediation mapping rules between heterogenous otologies. Each artefact must be registered with WSMX in order to be utilized during the runtime phase.

WSMO is defined on top of existing, well-established Web service standards. In figure 2, the

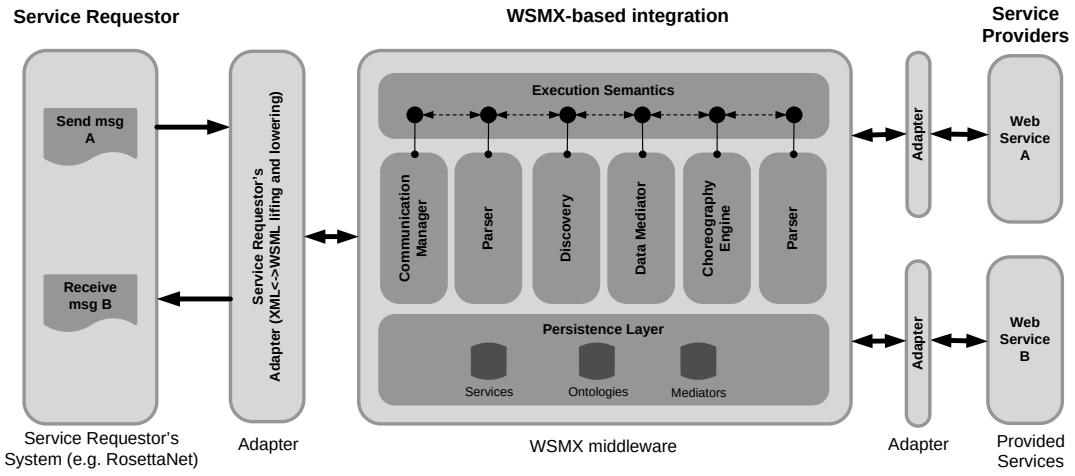

Figure 1: WSMX Architecture

relation between WSMO service definitions, ontologies and WSDL definitions is depicted. DERI modelled and published Semantic Web services using WSMO which defines service semantics along with *non-functional properties*, *functional properties* and *interfaces (behavior definition)* as well as *ontologies* that define the information models on which services operate. A service requestor in WSMO is represented by a Goal which specifies the requirements on the service to be consumed. Service Providers are represented by their Semantic Web services (SWS). WSMO Goals enable goal-based service invocation which is the basis for advanced semantic discovery and mediation provided by the WSMX environment. In addition, grounding from semantic descriptions to underlying WSDL and XML Schema definitions must be defined in order to perform invocation using underlying WSDL and SOAP specifications.

The WSMO functional description (capability) contains the formal specification of functionality that the service can provide. Interfaces describe service behavior, modeled in WSMO as (1) *choreography* describing how service functionality can be consumed by service requester and (2) *orchestration* describing how the same functionality is aggregated out of other services. The interfaces in WSMO are described using ontologized Abstract State Machines (ASM)(Roman and Scicluna, 2006) defining rules modeling interactions performed by the service. Modelling of WSMO elements can be performed using WSMT[5] or WSMO Studio[6].

Once the design time phase is completed and all required artefacts have been defined, the run-time

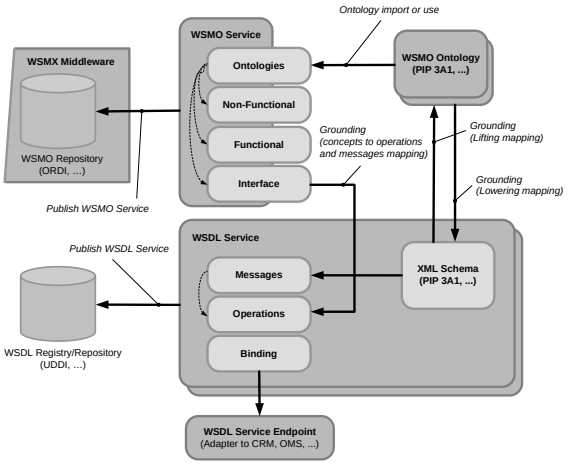

Figure 2: WSMO and WSDL Services

phase can commence. The instance data is sent from the service Requestor in their native data format (e.g. XML) to the dedicated adapter where it is lifted to the semantic level and from there it is sent to the WSMX as a WSMO Goal. WSMX is a component-based middleware following the Service Oriented Architecture (SOA) principles of loose-coupling. Major WSMX components include:

Data Mediation deals with heterogeneity problems between the service requester and service provider that can appear on the ontology level. All messages in WSMX are semantically described in WSML, meaning that the data to be mediated consists of ontology instances. This component applies mappings defined during the design time to the run-time instance data.

[5]http://wsmt.sourceforge.net
[6]http://www.wsmostudio.org

Process Mediation deals with solving the interaction mismatches. In cases where the Goal choreography and the choreography of the Web Service do not match, process mediation is required. The role of the process mediator is to retain, postpone, rebuild or create messages that would allow the communication process to continue.

Discovery determines whether a Goal description matches services descriptions using rich semantic descriptions of services approach.

Choreography Engine is responsible for using the choreography descriptions of both the service requester and provider to drive the conversation between them. It maintains the state of a conversation and takes the correct action when that state is updated.

2.2 Mediation Scenario Solution

The DERI solution to the SWS-Challenge mediation scenario starts with creating ontologies, with existing standards and systems as their basis, namely RosettaNet PIP 3A4 and CRM/OMS schemas. Next, Semantic Web services for the CRM and OMS systems of the Blue legacy system as well as Goal templates, conforming to PIP3A4, for the service requestor are created. In addition, a grounding must be defined from the semantic (WSMO) descriptions to the syntactic (WSDL) descriptions. Lifting and lowering has to be defined between the syntactic and semantic data models. WSDL descriptions are automatically generated by Axis and published on a Jetty server (internal to the WSMX). For the SWS-Challenge we provide two adapters: (1) PIP3A4-WSMX adapter and (2) CRM/OMS-WSMX adapter incorporating lifting and lowering between XML schema and ontologies.

Listing 1 shows a fragment of the choreography for the CRM/OMS service. The choreography is described from the service point of view thus the rule says that in order to send *SearchCustomerResponse* message, the *SearchCustomerRequest* message must be available. By executing the action of the rule ($add(SearchCustomerResponse)$), the underlying operation with corresponding message is invoked according to the grounding definition of the message which in turn results in receiving instance data from the Web service.

```
choreography MoonChoreography
    stateSignature
      in moon#SearchCustomerRequest withGrounding { ... }
      out moon#SearchCustomerResponse withGrounding { ... }

transitionRules MoonChoreographyRules
  forall {?request} with (
      ?request memberOf moon#SearchCustomerRequest
  ) do
    add(_# memberOf moon#SearchCustomerResponse)
  endForall
```

Listing 1: CRM/OMS Choreography

In listing 2, the mapping of *searchString* concept of the CRM/OMS ontology to concepts *cusomterId* of the $PIP3A4$ ontology is shown. The construct $mediated(X,C)$ represents the identifier of the newly created target instance, where X is the source instance that is transformed, and C is the target concept we map to.

```
axiom mapping001 definedBy
  mediated(X, o2#searchString) memberOf o2#searchString :–
  X memberOf o1#customerId.
```

Listing 2: Mapping Rules in WSML

During the runtime phase, first a RosettaNet PIP3A4 PO message is sent from the Blue Company to the entry point of the RosettaNet-WSMX adapter where it is lifted to WSML according to the PIP3A4 ontology and rules for lifting using XSLT. A WSMO Goal is created from this message and it is sent to WSMX where the SWS matching the Goal request can discovered. Next, the Choreography Engine is instantiated with the Goal's and Semantic Web service's choreographes. The Process Mediator is used to decide which data will be added to requester's or provider's choreography – this decision is based on analysis of both choreographies. Once the requester's and provider's choreographies have been updated, the Choreography Engine processes each to evaluate if any transition rules could be fired. During the communication between Blue and Moon, all data heterogeneities between utilized ontologies are handled by the Data Mediator while process level heterogeneity is tackled by Process Mediation. Conversation ends when there are no additional rules to be evaluated from the requester's or the provider's choreography.

3 Solving the Service Mediation Scenario with WebML

The mixed team DEI-Cefriel adopted a solution based on the WebML/Webratio framework (Ceri et al., 2002; Web,) to solve the mediation problem.

3.1 Environment

WebML language is a high-level notation for data- and process- centric Web applications. It allows

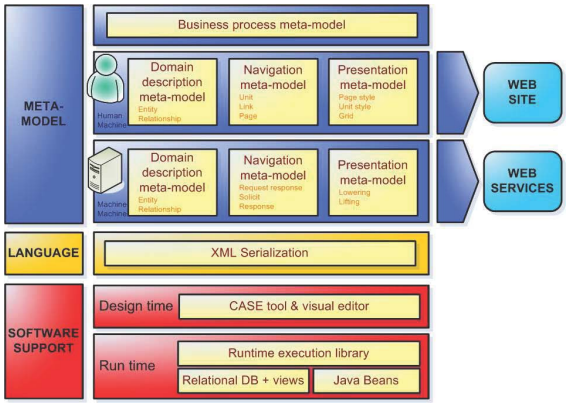

Figure 3: The WebML/Webratio framework.

specifying the conceptual modeling of Web applications built on top of a data schema used to describe the application data, and composed of one or more hypertexts used to publish the underlying data. The WebML data model is the standard Entity-Relationship (E-R) model extended with Object Query Language (OQL) constraints. To describe Web services interactions, WebML has been extended with Web service units (Manolescu et al., 2005). In particular the `Request-Response` and `One-way` operations are used to consume external Web services, while `Solicit` and `Response` unit are used to publish Web services. In (Brambilla et al., 2006b) the language has been extended with operations supporting process specifications, and a further modeling level was added to framework allowing to start workflows/orchestrations using a BPMN model that is later automatically translated to a WebML skeleton to be refined by designers.

Each WebML unit has its own well defined semantic and its execution complies with its semantic. The composition of different units lead to the description of the semantic of hypertext or Web services. The language is extensible, allowing for the definition of customized operations and units. It has been implemented in a prototype that extends the CASE tool WebRatio [7], a development environment for the visual specification of Web applications and the automatic generation of code for the J2EE platform. Among the facilities provided, the design environment has a mapping tool that allows to visually define mappings between incoming soap messages and the WebML data model. The design environment is equipped with a code generator that deploys the specified application and Web services in the J2EE platform, by automatically generating all the necessary pieces of code, including data extraction queries, Web service calls, data mapping logics, page templates, and WSDL service descriptors (see Figure 3 for the overall framework).

[7] http://www.webratio.com

3.2 Mediation Scenario Solution

The solution for the mediation problem starts by designing the data model underlying the RosettaNet messages with an extended E-R model. We identified three main entities: the Pip3APurchaseOrder, the Partner and the ProductLineItem. Each Pip3APurchaseOrder instance is related with one or more ProductLineItem instances, one Partner representing the Buyer, one Partner representing the Seller and one Partner representing the Receiver. Every ProductLineItem instance may have one Partner representing a Receiver for the single line. We modeled only the essential data for the scenario.

After modeling the data structures, an high level BPMN model is created representing the mediator (see on the left-hand side on Figure 4 for the mediation from Blue to Moon); this model formalize the orchestration of the Moon Web services and define states pertaining to the mediation process as by SWS-Challenge specification. Then, the BPMN model is used to automatically generate a WebML skeleton that is manually refined to complete the design of the mediator. The final model for the Blue to Moon mediator is reported on the right-hand side in Figure 4:

1. In the first line, as soon as the order is received (`Solicit` unit), the Pip3APurchaseOrder is converted to the Canonic XML (`Adapter` unit) and stored in the database (`XML-In` unit), the status of the current Pip3APurchaseOrder is set to "To Be Processed" (`Connect` unit) and the Acknowledge message is returned to the service invoker (`Response` unit).
2. Next, the Buyer Partner is selected (`Selector` Unit) and a message to query the CRM service is created (`Adapter` unit) and sent to the Moon Legacy System (`Request-Response` unit). Once a reply has been received, the CustomerId is extracted from the reply message (`Adapter` unit) and stored in the data model (`Modify` unit). The status of the order is set to "CustomerId received" (`Connect` unit).
3. For each Receiver Partner in the order (`Selector` unit) a message for the createNewOrder operation is created (`Adapter` unit) and sent to the Moon Legacy System (`Request-Response` unit). Once a reply has been re-ceived, the OrderId is

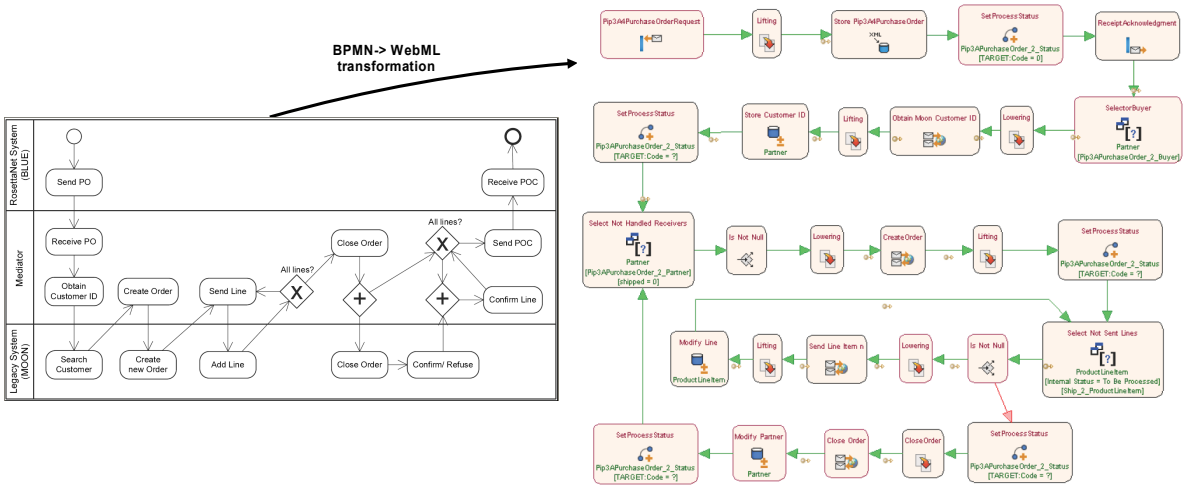

Figure 4: The BPMN and WebML models of the Blue to Moon Mediator.

extracted from the reply message (`Adapter` unit) and stored in the data model (`Modify` unit). The status of the order is set to "OrderId received" (`Connect` unit).

4. Next, the ProductLineItem instances related to current Pip3APurchaseOrder and Receiver Partner are processed by a cycle: at every interaction a message for a single line is created and sent to the Moon Legacy System (`Request-Response` unit), and the received LineId is stored (`Modify` unit).

5. Finally when all the lines have been processed the message for the closeOrder operation is created (`Adapter` unit) and sent to the Moon Legacy System (`Request-Response` unit) and the status of the order is set to "Order closed" (`Connect` unit). If there are still Receiver Partner to be processed, they are processed the loop starts again.

The SOAP messages transformation to and from the WebML data model are performed by proper WebML units (`Adapter` units) that apply XSLT transformations; XSLT stylesheets are designed with the visual mapping tool (a fragment is reported in Listing 3).

```
<xsl:template match="//po:Pip3A4PurchaseOrderRequest">
  <xsl:variable name="fromId" select="./core:fromRole//core:
      businessName/core:FreeFormText"/>
  <xsl:variable name="toId" select="./core:toRole//core:
      businessName/core:FreeFormText"/>
```

Listing 3: A fragment of the XSLT for mapping RosettaNet messages to the WebML data model.

4 Comparison of the Mediation Approaches

Both submissions have successfully addressed the SWS-Challenge Moon mediation scenarios. In this section we elaborate on the details of the similarites and differences in how DERI and DEI-Cefriel approached the mediation tasks. We compare them with respect to the following aspects:

Underlying Technologies. The DERI approach follows the Web Services Modelling Ontology (WSMO) framework reflecting four top elements, viz. explicitly modelling Goals, Web services and Ontologies. Ontology-to-ontology mediation is reflected in the Data Mediation component while goal-to-Web service mediation is handled by the Process Mediation component. On the other hand, WebML focuses more on the modeling of the ww-Mediators and of the internal logics of the services (if they need to be developed), that are defined through visual diagrams representing the execution chains, while gives little emphasis to the design of the Goals and of the other semantic aspects. Indeed, the approach provides semi-automatic extraction of WSMO Goals and Web service behaviour from the designed models, that need to be refined later by the designer.

Data Model. Both teams provided expressive data models reflecting domain specific knowledge and exchanged messages on the data schema and instance level. WebML allows to specify a data model describing the domain data structure as an Entity-Relationship (E-R) or, equivalently, a UML class diagram. WebML E-R diagram pro-

Feature	WebML/WebRatio	WSMO/WSMX
Data Model Design	ER-model manually created from analyzing the RosettaNet messages and adding status information. Used to keep data persistent.	Independent ontologies created both from analyzing the Blue RosettaNet messages and internal data requirements by Moon's legacy services.
Process Mediation Design	WebML model structure with standard units generated from BPMN model. Units are then configured and other units are added from the library manually (no need for any implementation, no code generation, just component configuration).	The process mediation is modeled explicitly using ontologized ASM that represent the orchestration of the mediation service or the choreography of the invoked services. The orchestration and the choreographies are hence decoupled.
Data Mediation Design	XSLT mapping designed within a visual environment to lift SOAP messages to the WebML data model and lower the data selected from the WebML data model to a SOAP message.	Dedicated XML$<-->$WSML adapters handling ontology lifting and lowering. Design time ontology-to-ontology data mediation mappings.
Web service publishing	Generic standard units for receiving SOAP messages.	Generated WSs are internally published on Axis running on the Jetty server hosted on WSMX.
Web service invocation	Generic standard units for calls to Web services that are configured (at design time or at runtime) to invoke the Web services.	Communication Manager based is handling all communication with services using grounding information provided in SWS descriptions.
Process Mediation Execution	The designed mediator represents the process that will be executed. The configuration of the execution environment is automatically obtained from the model.	WSMO Choreography and Orchestrations modelled during the design time are directly executable.
Data Mediation Execution	Incoming and outgoing messages, according to the modeled mediator are lowered to the internal data model calling the preconfigured XSLT mapping.	The data mediator solves heterogeneity problems between the ontologies by applying design time mappings to instance level data .
Execution Monitoring	The Webratio runtime offers a default logging facilities that store all the execution threads.	Simple logging facilities. WSMX execution is presented as components' events flow on the simple Java SWING-based panel.

Table 1: Comparison of the presented solutions.

vides rich notation for specifying structure and relationships between concepts occurring in the given domain and it allows to impose simple constraints over modelled the domain by using WebML-OQL. Logic rules are not explicitly supported, however the authors showed that the expressive power of the model is very close to WSML-Flight. DERI used the WSML-Rule variant, a fully-fledged ontology language with rule support. SWS-Challenge mediation scenario data model did not require to utilize complex rules while modelling the ontology. Thus, despite of using more expressive language by DERI, both underlying data models were quite similar in terms of their expressiveness. Both teams modelled existing concepts, their attributes and relationships between these concepts without imposing additional constraints over the data models taking advantage of the expressiveness provided on the level of UML class diagrams. In both cases, mature tools exists to edit underlying data models, for WSMO there is and WSMT and WSMO Studio, while for WebML one can use Webratio CASE tool.

Process Model. Provided solutions differ quite significantly with respect to the process modelling. The joint team DEI-Cefriel followed a modern Software Engineering approach to model Moon orchestration, while DERI specified orchestration using ontologized Abstract State Machines(Roman and Scicluna, 2006) formalism which falls into process execution based on underlying rich knowledge base formalism. In the utilized Abstract State Machines (ASM), an ontology constitutes the underlying knowledge representation and transition rules are specified in terms of logic formulas. ASM provide precise and executable model for specifying processes allowing simulation (e.g. deadlock, livelock freedom detection) and elaborate reasoning over the model. DERI focused on the executable aspect of the ASM, not utilizing process simulation since

there is currently no tool for WSML supporting ASM simulation. Execution of the ontologized ASM has been carried out by ASM Engine used both in WSMX Choreography and Orchestration. ASM-based modelling allows to model processes in a more flexible way, supporting *strong decoupling* between service requester and service provider where delivery of exchanged messages do not have to be explicitly modelled, instead a Knowledge Base (KB) can be populated with allowed messages and it is up to the state of the KB and transition rules to determine and evaluate the usability of the available information.

In the WebML approach, RosettaNet, the Moon CRM and ORM related messages are modelled as a part of the same process tightly coupling the Moon mediation process with the RosettaNet client. This coupling is embedded within the design of the wwMediatior, specified as a WebML operation chain triggered by a Web service call. The DERI approach is more flexible being more client independent where orchestrated service is not aware of any incoming or outgoing RosettaNet messages. It simply specifies messages in its native ontology and it is up to the Data Mediator to resolve and mediate data heterogeneities between service requester and service provider.

Data Mediation. As mentioned before WSMX, has a strong notion of the mediation what allows decoupling between the interacting participants so that they do not need to directly comply with the requirements of the other party. Due to technical issues with transition of runtime Data Mediation from Flora[8] reasoner to MINS[9] reasoner DERI used a code-based Data Mediator for the last workshop. In the meantime, this issue has been resolved and generic Data Mediation is expected to be utilized for the next phase of the SWS-Challenge. In the DEI-Cefriel submission the notion of data mediation and data mapping from one RosettaNet to Moon specific data model is encoded in XSLT transformations.

In the WebML approach XSLT transformations can be reused but they do not exploit ontological information. They provide one-to-one mapping between XML documents. New transformations need to be devised for new message models. In short-term it is a faster solution; however if number of the clients using different data format grows, then scalability becomes an issue for the WebML approach. For each customer it is required to change and redefine orchestrated business process. For instance, when considering customers using other data formats and following different message exchange patterns, new ooMediators and wwMediators need to be designed. This can be partially avoided when there is no need to process the content of the messages, simply by not checking the format of the incoming message and lifting it to the internal model dynamically according to the incoming message format.

Tool Support. It is also worth overviewing the maturity of both solutions and tools availably for them. Currently, there is a better tool support for WebML modelling, especially on the process modelling level. There is a basic support for editing ontologized ASM and no support for simulation and model testing. On the other hand support for editing WSMO elements is quite good. Tools utilized throughout the lifecycle of development of DERI submission are being actively developed (WSMO editor, data mediation, WSMX, others). Some of them are not yet as mature as the Webratio CASE tool especially in terms of ontologized ASM-based process modelling. However, other aspects of the modelling involved in semantic integration like for instance WSMO ontology editing using WSMT provides already quite mature and user-friendly functionality.

The comparison of the two solution is summarized in the Table 1.

4.1 Coping with the scenario changes

Both solutions were able to comply with the changes required by the second version of the mediation scenario. In particular, as regards the WebML solution, the scenario changes required to update the data model introducing the fact that there may be a receiver for each single item lines. As regard the process mediation, the BPMN model was updated to consider the new loop required to handle different receiver and to invoke the production Web service. Accordingly also data mappings have been updated. The cost of copying with the changes was relatively low and it requires less than one day of work.

For the WSMO based solution minor changes were required in the ontology similarly like in the case of WebML data model. Also the Choreography of the Moon service had to be updated to model the loop required in the changes introduced in the second version of the scenario. Lack of process simulation and graphical support for ontologized ASM modelling requires good understanding of this technology and the DERI team was able to incorporate required changes

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

[9] http://tools.deri.org/mins

also within less than one day. Nevertheless, it is acknowledged that it would take longer for a person unfamiliar with this formalism while the WebML solution is more likely to be grasped and modified reasonably quickly even by a non-expert.

5 Related work

The most obvious related work can be identified among other submissions addressing the SWS-Challenge mediation scenario. There is a significant similarity between jABC approach and WebML since both are based on Software Engineering methods with strong emphasis on graphical process modelling. However, their underlying data model does not support rules and has the expressivity on the level of UML Class Diagrams. On the other hand WSML comes with powerful rule and F-logic support, however the mediation task did not require to utilize its full potential.

Another relation can be drawn to WebML extension(Brambilla et al., 2006a) which allows generation of semantic descriptions of the modeled Web services. The ontological language adopted is WSMO due to his strong separation of the different concepts (i.e., Goals, Mediators, Web Services, and Ontologies) that are at the basis of Semantic Web Services.

6 Future Work and Conclusions

In this paper we compared to different approaches to the mediation scenario proposed in SWS-Challenge. The scenario is very similar to real world scenario and enough complex to stress both the two compared solution and to evidence their advantages and disadvantages. While the WebML based solution exploits well-established, efficient Software Engineering methods that allows some de-coupling and reusing, the WSMO based solution goes beyond standard way of system integration allowing for a better de-coupling and reusability of the modelled elements. The WebML based solution offers a mature and easy to use design environment totally based on a visual paradigm, with a set of automatic facilities for partial generation of semantic descriptions and definitions, while the tool support of the WSMO solution is less mature and is still missing for some parts (especially for ontologized ASM-based process modeling) an easy visual paradigm to facilitate modelling of the elements involved in the semantic integration.

The next edition of the SWS challenge workshop will be held during ESWC 2007; this edition will present a new scenario that requires the combination of discovery and composition. Both the teams, were able to handle the two scenarios separately and hence we are confident that both the technologies employed will be able to propose an effective solution for the new scenario. In particular the WebML based solution will further exploit the integration with the Glue discovery engine by exploiting it for the discovery phase and modeling within WebML a solution to dynamically compose and invoke the services according to the discovery results. On the other side the WSMO based solution will incorporate service discovery via *AchieveGoal* construct into its Orchestration allowing late-binding and service composition.

Acknowledgements

This material is based upon works supported by the Science Foundation Ireland under Grant No. SFI/02/CE1/I131, the EU-funded projects Knowledge Web (FP6 - 507482) and DIP (FP6 - 507483).

REFERENCES

Brambilla, M., Celino, I., Ceri, S., Cerizza, D., Della Valle, E., and Facca, F. M. (2006a). A Software Engineering Approach to Design and Development of Semantic Web Service Applications. In *Proceedings of the 5th International Semantic Web Conference (ISWC 2006)*.

Brambilla, M., Ceri, S., Fraternali, P., and Manolescu, I. (2006b). Process modeling in web applications. *ACM Trans. Softw. Eng. Methodol.*, 15(4):360–409.

Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., and Matera, M. (2002). *Designing Data-Intensive Web Applications*. Morgan Kauffmann.

Haller, A., Cimpian, E., Mocan, A., Oren, E., and Bussler, C. (2005). WSMX – A Semantic Service-Oriented Architecture. In *Proc. of the 3rd Int. Conf. on Web Services*, pages 321 – 328. IEEE Computer Society.

Manolescu, I., Brambilla, M., Ceri, S., Comai, S., and Fraternali, P. (2005). Model-driven design and deployment of service-enabled web applications. *ACM Trans. Internet Techn.*, 5(3):439–479.

Roman, D., Keller, U., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C., and Fensel, D. (2005). Web Service Modeling Ontology. *Applied Ontologies*, 1(1):77 – 106.

Roman, D. and Scicluna, J. (2006). Ontology-based choreography of wsmo services. Wsmo final draft v0.3, DERI. Available at: `http://www.wsmo.org/TR/d14/v0.3/`.