Improvements and Future Perspectives on Web Engineering Methods for Automating Web Services Mediation, Choreography and Discovery: SWS-challenge phase III

Marco Brambilla¹, Irene Celino², Stefano Ceri¹, Dario Cerizza², Emanuele Della Valle², Federico Facca¹, Christina Tziviskou¹

¹ Dipartimento di Elettronica e Informazione, Politecnico, 20133 Milano, Italy {mbrambil, ceri, facca, tzivisko}@elet.polimi.it
² CEFRIEL, 20133 Milano, Italy {celino, cerizza, dellavalle}@cefriel.it

Abstract. In this paper we describe the improvements and the future perspective on our approach to Web services mediation, choreography, and discovery presented at the SWS Challenge 2006. We summarize the main features of our solution, based on the WebML design methodology and Glue WSMO discovery engine. Then, we discuss the current improvements given to the solution for Phase III of the challenge, and the overall future perspectives of our approach.

1 Introduction

We have been involved in the first two phases of the Semantic Web Service challenge 2006 [1], and we completed Phase II of the challenge with the “most complete solution” [2]. We have developed a rather complete and convincing concept of a design support environment for Semantic Web applications [3, 12], but our software solution requires a lot of improvements before encompassing all the needs which are set in the challenge. In this paper, we discuss how we intend to further extend our solution in preparation for the challenge’s next “round”, which is set for November 2006.

The challenge presents two scenarios: the Mediation Scenario integrates a legacy system with a system supporting the RosettaNet protocol, the Discovery Scenario performs order processing by automatically discovering a new supplier together with a shipper. The problem is clearly neither a pure Semantic Web one, nor a pure Software Engineering one. Semantics is clearly needed to address in a flexible way the Discovery scenario, but Software Engineering tools and methods are the right ones to address in a flexible way the Mediation scenario.

For this reason, we adopt an original mix of Semantic Web and Software Engineering techniques: WSMO [4] as Semantic Web Service approach, Glue [5, 6] as Semantic Web Service Discovery engine, WebML [7, 8] as Web engineering model for designing and developing semantically rich Web applications implementing Ser-
vice Oriented Architecture, and WebRatio [9] as WebML CASE tool. These background technologies are reviewed in Section 2.

Section 3 shows the general features of our solution as it was presented in Phase II of the challenge; then, Section 4 illustrates how we are currently extending our solutions so as to incorporate new aspects. This work can be considered as the “short term evolution” of our approach. However, we have decided to take this paper also as an opportunity also for reasoning on a longer-term vision; Section 5 presents such vision and gives hints about our future research directions.

2 Background

WebML/WebRatio WS edition. WebML is a high-level notation for data- and process-centric Web applications. It allows 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 allows one to specify a data model describing the domain data structure as an Entity-Relationship (E-R) or, equivalently, a UML class diagram. Upon the same data model, it is possible to define different hypertexts (e.g., for different types of users or for different publishing devices), called site views. A site view is a graph of pages, allowing users from the corresponding group to perform their specific activities. Pages consist of connected units, representing at a conceptual level atomic pieces of homogeneous information to be published: the content that a unit displays is extracted from an entity, and selected by means of a selector, testing complex logical conditions over the unit’s entity. Units within a Web site are often related to each other thru links carrying data from a unit to another, to allow the computation of the hypertext. WebML allows specifying also update operations on the underlying data (e.g., the creation, modification and deletion of instances of an entity, or the creation and deletion of instances of a relationship) or operations performing other actions (e.g. send an e-mail). To describe Web services interactions, WebML has been extended with Web service units, corresponding to the WSDL classes of Web service operations. Request-response and response operations are triggered when the user navigates one of their input links; from the context transferred by these links, a message is composed, and then sent to a remote service as a request. Solicit and one-way are instead triggered by the reception of a message. Indeed, these units represent the publishing of a Web service, which is exposed and can be invoked by third party applications. In the case of one-way, the WebML specification may dictate the way in which the response is built and sent to the invoker.

The language is extensible, allowing for the definition of customized operations and units. It has been implemented in the CASE tool WebRatio, a development environment for the visual specification of Web applications and the automatic generation of code for the J2EE platform. The design environment is equipped with a code generator that deploys the specified application and Web services.

Glue WSMO discovery Engine Glue is a WSMO compliant discovery engine that provides the basis for introducing discovery in a variety of applications that are easy
to use for requesters, and that provides efficient pre-filtering of relevant services and accurate discovery of services that fulfill a given requester goal.

In conceiving Glue, the model for WSMO Web Service discovery was refined by making the role of mediation more explicit, by (i) using the notions of class of goals and class of Web Service descriptions; (ii) using ggMediators for automatically generating a set of goals semantically equivalent to the one expressed by the requester but expressed with a different form or using different ontologies (iii) making wgMediators the conceptual element responsible for evaluating the matching; (iv) using ooMediators for solving any terminological mismatch that can appear with different polarized ontologies for the domains; and (v) by redefining the discovery mechanism as a composite procedure where the discovery of the appropriate mediators and the discovery of the appropriate services is combined. Moreover in designing Glue the authors refined WSMX Discovery Engine architecture according to their refined WSMO discovery conceptual model both in terms of components and execution semantics.

The Communication Manager exposes Glue functionalities to other services, by offering plain Web Services for publishing semantic Web Service Descriptions (WSD), submitting the goal for searching the repository for WSDs previously published and, finally, getting the results of the matching. The Constructors are responsible for translating the SOAP messages and goals. The Goal Translator translates the user goals into goals more close to the provider perspective. Finally, the Proof Generator gathers all the necessary information from the other components, invoking the internal reasoner and providing the discovered results to the client.

The Glue implementation uses internally F-logic and it is built around an open source F-logic inference engine called Flora-2 that runs over XSB, an open source implementation of tabled-prolog and deductive database system.

3 Summary of the approach

This section briefly outlines the approach we discussed in the previous phases of the challenge [11, 12].

Phase I: Mediator scenario. The modeling of the mediator started from the design of the data model. The RosettaNet message was analyzed and a corresponding WebML E-R diagram was obtained from it. We identified three main entities: the Pip3APurchaseOrder, the Partner and the ProductLineItem, as showed in Figure 1. Once the WebML data model was completed, we started modeling the Web Service providing the mediation feature. The obtained model is shown in Figure 2a.

- First we modeled the operation receiving the RosettaNet message and forwarding the order to Moon (first line in the diagram);
- Next (second line), 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 Cus-
tomerId 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”.

• Then (third line) the Buyer Partner and the Receiver Partner are selected (Selector Units) and 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 received, the OrderId is 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).

• Next (fourth line), the ProductLineItem instances related to current Pip3APurchaseOrder 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).

• Finally, when all the lines have been processed, the message for the closeOrder is sent to the Moon Legacy System (Request-Response Unit) and the status of the order is set to “Order closed” (Connect Unit).

Then we modeled the operation to receive lines confirmation by the Moon Legacy System (Figure 2b).

• For each line confirmation received (Solicit Unit), the status is extracted, the relative order and line are selected (Selector Units), the status of the stored line is modified and the Acknowledge message is returned (Response Unit).

• Finally, if all the lines have been received (Switch Unit), the XML serialization of the data for the current Pip3APurchaseOrder is extracted (XML-Out Unit) and a RosettaNet Purchase Order Confirmation message is sent to the RosettaNet client (Request-Response Unit) and the status of the order is set to “Rosetta PO Confirmation sent” (Connect Unit).

Phase I: Discovery scenario. In modelling ontologies, goals, Web Services and Mediator for the first phase of the challenge, we followed the mediator centric methodology described in [2] and we used F-logic.
First of all, we modelled four ontologies including date-time, location, products and shipments. The development was kept to the minimum necessary for the two scenarios. In particular our date-time ontology is not expressive enough to model the generic notion of "business day".

Secondly, we defined two classes of goals one for the shipment and one for purchasing. In both cases we modelled the capabilities limiting ourselves to post-condition.

Third, we model the classes of Web Services for shipment and purchasing. In both cases we model all the restrictions that must hold in order to invoke the service as assumptions and the result provided by the service as post-conditions. Glue approach was sufficient to model most of the details, but we were not able to model some complex features that require invocation of external Web Services to gather more information regarding a Web Service.

Last, but not least we model the matching rules within the wgMediators. The rules, written in F-logic, can be divided in three groups: those that calculate intermediate
results (such as the price), those that evaluate the restrictions in the assumption part of the description and those that describe the transactions in the post-condition part of the description. The table below shows an example of each type of rule.

<table>
<thead>
<tr>
<th>Rule that calculate intermediate results</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>calculateShipmentPrice(ShipmentPricing, Location, DeliveryModality, Guarantee, GoodWeight, PriceCalculated) :-</code></td>
</tr>
<tr>
<td><code>(Location::ShipmentPricing.location; Location=ShipmentPricing.location),</code></td>
</tr>
<tr>
<td><code>(DeliveryModality::ShipmentPricing.deliveryModality; DeliveryModality=ShipmentPricing.deliveryModality),</code></td>
</tr>
<tr>
<td><code>(Guarantee=ShipmentPricing.guarantee; Guarantee=guaranteeNo),</code></td>
</tr>
<tr>
<td><code>PriceCalculated is (ShipmentPricing.basePrice + (GoodWeight-1)*ShipmentPricing.pricePerWeight)</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rule that evaluate assumptions</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>checkRestrictionOnMaxNumOfDaysBetweenOrderAndPickupInterval(RequestsShipmentService, RestrictionsOnShipmentService) :-</code></td>
</tr>
<tr>
<td><code>RequestsShipmentService[currentDateTime-&gt;OrderDateTime, requestedPickupDateTimeInterval-&gt;_[start-&gt;PickupDateTimeStart, end-&gt;PickupDateTimeEnd]],</code></td>
</tr>
<tr>
<td><code>RestrictionsOnShipmentService[maxNumOfDaysBetweenOrderAndPickupInterval-&gt;MaxDaysForStart, maxNumOfDaysBetweenOrderAndPickupInterval-&gt;MaxDaysForEnd],</code></td>
</tr>
<tr>
<td><code>daysBetween(PickupDateTimeStart, OrderDateTime, X):(X&lt;MaxDaysForStart; X=MaxDaysForStart),</code></td>
</tr>
<tr>
<td><code>daysBetween(PickupDateTimeEnd, OrderDateTime, Y):(Y&lt;MaxDaysForEnd; Y=MaxDaysForEnd)</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rule that encode a necessary condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>checkContainmentOfPickupAndDeliveryLocation(RequestsShipmentService, ProvidesShipmentService) :-</code></td>
</tr>
<tr>
<td><code>RequestsShipmentService[requestedPickupLocation-&gt;X], ProvidesShipmentService[pickupLocations-&gt;Y],</code></td>
</tr>
<tr>
<td><code>(X=Y; X::Y),</code></td>
</tr>
<tr>
<td><code>RequestsShipmentService[requestedDeliveryLocation-&gt;H], ProvidesShipmentService[deliveryLocations-&gt;K],</code></td>
</tr>
<tr>
<td><code>(H=K; H::K)</code></td>
</tr>
</tbody>
</table>

Phase II: Changes in the Mediator scenario. In the second phase of the challenge, first the RosettaNet message was changed introducing also a core:shipTo element for each ProductLineItem. This was achieved by simply updating our data model by introducing a new relationship between the entity ProductLineItem and the entity Partner. Then, a change in the mediation process was introduced: when the Stock Management system is incapable to fulfill request from the customer and it replies that the particular line item cannot be accepted, the Mediator must communicates with the legacy Production Management system to obtain relevant information on date and price to manufacture a new product. If this information meets initial expectations of the customer as specified in the RosettaNet message, the product should be ordered. To fulfill new requirements, we changed the mediator by introducing the set of operation required to query the new the Production Management Web Service.

Phase II: Changes in the Discovery scenario. Most of the proposed changes indeed were achieved by updating the WSDL; this could have been difficult to handle if we had proceed to model choreography for invoking the Web Services after the discovery. Concerning the changes to capabilities, most of them required minor modifications such as: adding some instances to the ontologies (e.g. Oceania as possible location), changing some instances of Web Services (e.g. updating latest pick up time, removing the distinction between letter and worldwidePriorityExpress). Fewer of the changes also required re-modelling of the classes of Web Services (e.g. by changing the accepted multiplicity for a given property or adding the information about the additional price per collection) and, consequently, of the corresponding wgMediators. One proposed change that heavily impacted on the wgMediator was the new rule for computing the price. Such rule, shown in the following table, discover the correct Shipment Pricing in function of the location and the guarantee; then calculates the price taking care of the Dimensional Weight (calculated by the `calculateDimension-`
alWeight rule) and the additional price per collection, if set. The calculated price is used by other rules to filter the results according to the maximal price expressed in the goal.

\[
\text{calculateShipment-Price(ShipmentPricing,Location,DeliveryModality,Guarantee,GoodDimensionalWeight,NumberOfPackages,PriceCalculated) :- (Location::ShipmentPricing..location; Location=ShipmentPricing..location), (DeliveryModality::ShipmentPricing..deliveryModality; DeliveryModality=ShipmentPricing..deliveryModality), (Guarantee::ShipmentPricing..guarantee; Guarantee=guaranteeNo), PriceCalculatedForOnePackage is (ShipmentPricing..basePrice + (GoodDimensionalWeight)*ShipmentPricing..pricePerWeight), \]

\[
\begin{cases}
\text{PriceCalculated is (PriceCalculatedForOnePackage*NumberOfPackages)}; & \text{if } \text{ShipmentPricing..additionalPricePerCollection} \leq 0 \\
\text{PriceCalculated is (PriceCalculatedForOnePackage + ShipmentPricing..additionalPricePerCollection)}; & \text{if } \text{ShipmentPricing..additionalPricePerCollection} > 0
\end{cases}
\]

4 Improvements

The phase III of the challenge do not introduce any new requirement so we aimed to improve and refine our previous submission solving some of the left out issues and the errors. Here we present what we improved in our solutions to the two scenarios.

Mediation. In the previous challenge we did not address correctly some of the changes to the Mediation Scenario: we did not understood that the introduction of the optional shipment address for each line item required also a change in the process organization and not only in the data representation. According to the modified sce-

![Fig. 3. The improved version of the Mediator.](image-url)
nario, line items must be grouped according to their shipment address and for each group an independent new order has to be sent to the Moon Legacy System. We improved our mediator handling this requirements (see Figure 3): i.e., we introduced a loop over the shipment addresses associated to the incoming RosettaNet PO; inside the loop, a new order for the current address is created and each line item pertaining the address is selected and added to the new order; finally the order is closed, and the next address, if available, is processed.

**Discovery.** In phase-II of the challenge, Glue was not able to completely pass level 2 because it cannot invoke any external service in order to gather additional information on the service. In a broader sense, the current implementation of Glue only supports discovery; negotiation and selection are left to the application in which Glue is integrated. Nevertheless, by delegating invocation to an external component the approach of Glue can be extended with limited effort to support negotiation (and selection).

The Glue employed in phase-II assumes that the description of each Web Service is available statically: Glue assumes no dynamic dependencies between Web Service description and goal. However, in many real scenarios a service exposes one or more operations for negotiating the service itself: in the case of the challenge, the calculation of the price for shipping a product. In order to take this new requirement into consideration several changes were introduced in Glue.

> At conceptual level we model both Goal and Web Services making explicit differences among:

---

1 For reason of space we don’t address selection in this paper.
“discovery” capabilities, which are static description of the service in terms of functional properties,

“negotiation” capabilities, which are description of the service that need to be evaluated by invoking one or more operation of the service (for describing the choreography we relay on WebML), and

“selection” capabilities, which are static or dynamic non functional descriptions.

Not surprisingly the different modelling of Goals and Web Services has some implication on Glue, but they are limited to the Communication Manager and the execution semantics. The mediation centric approach adopted for the Discovery is valid also for the Negotiation.
From the Communication Manager point of view, the main difference is in the input; the Discovery requires a Goal of a known Class of Goals, the Negotiation requires such a Goal, possibly resulting from applying any ggMediator to the Goal originally submitted, and the list of Web Services discovered. In order for the Negotiation component to be useful at least one of such Web Services needs to be filtered by acquiring more information. Figure 4 presents in a graphical way the difference among the Glue presented in phase-II (a) and the one presented in phase-III (b). The new version handles negotiation capabilities by delegating their evaluation to an external Invocation component implemented within the WebRatio framework.

The internal components of Glue can be reused for implementing the Negotiation component just by reconfiguring the execution semantics. In Figure 5 we represent the execution semantics of the two components we use for addressing phase-III of the challenge. The execution semantics of the Discovery component is the same of the one presented at phase-II. The execution semantics of the Negotiation component is derived from the discovery by extending it with an activity for evaluating negotiation capabilities and by reusing activities that load WSMO element and evaluate the wgMediator.

5 Future Perspective and Overall Improvement of the Method

As final result of this research activity we aim at convening and implementing a complete extension WebML methodology towards the design of Semantic Web Services applications, supported by a design environment including (within the same framework based upon WebRatio [9]) several new components or the evolution of existing components (see Figure 6).

Our approach extends the design flow supported for conventional Web applications [6] which leads the designer from the process modeling to the running Web application, by producing some intermediate artifacts (BPMN models, WebML skeletons, data models, hypertext models). Such models are enriched by imported ontological descriptions (on top of the figure) and are exploited for semi-automatically generating WSMO-compliant semantic (at the bottom of the figure): the ontology is derived from BP model, data model, and hypertext model; the web services capability

Fig.6. Overall top-down approach for developing Semantic Web Service applications.
description is derived from hypertext model; the choreography information is derived from BP model and hypertext model; the user goals are derived from the BP model.

The resulting running application (at the right of the figure) is a traditional Web application running within WebRatio that relays on a Semantic Execution Environment for delegating the execution of WSMX-empowered tasks (e.g., the discovery scenario of the challenge).

Other ongoing work at Politecnico di Milano and Cefriel can provide useful components to be used in the framework. We already extensively presented Glue, a discovery engine that can be used by semantic web applications with simple customisation. In addition, we are building generic tool support for XML-2-XML mapping based on graphic notation and upon inference, that extends tools such as Clio [13] and makes such technology available as part of the WebRatio environment; transformations are developed by simple graphic drawings and are inferred whenever appropriate [14]. Such transformations are the basis for the semi-automatic development of “syntactic mediators” (i.e., those mediators doing simple format transformations) and are in general very helpful for the design and the implementation of arbitrary mediators.

6 Conclusions

This paper briefly summarized our improvements for Phase III of the SWS Challenge 2006; it also provides an outlook of our future research and development directions. The main message coming with our participation to the challenge is that software and Web engineering methods are a fundamental ingredient for the real success of the Semantic Web concept. Without sound and robust methods, which can guarantee the reusability and evolution of software systems, the wide adoption of Semantic Web solutions will always be at critical danger. The use of such methods and tools, combined with reusable components – such as Glue – for performing specific Semantic Web tasks has proven to be very promising.

7 References

6. Della Valle, E., Cerizza, D.: The mediators centric approach to automatic web service
discovery of Glue. In Hepp, M., Polleres, A., van Harmelen, F., Genesereth, M.R., edi-
tors, Proceedings of the First International Workshop on Mediation in Semantic Web
Services (MEDIATE 2005), Amsterdam, The Netherlands, December 12, 2005, CEUR-
Workshop Proceedings, 8, 35–50.
7. Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., Matera, M.: Designing Da-
Engineering Approach to Design and Development of Semantic Web Service Applica-
tions. In proceedings of 5th International Semantic Web Conference, ISWC 2006, Ath-
ens, GA, USA.
11. Brambilla, M., Ceri, S., Cerizza, D., Della Valle, E., Facca, F. M., Fraternali, P.,
Tziviskou, C.: Web Modeling-based Approach to Automating Web Services Mediation,
Choreography and Discovery. In Semantic Web Service Challenge 2006 Phase I, Stan-
ford University, Palo Alto, CA, March 2006. Available at:
http://swschallenge.org/wiki/index.php/
Workshop_Stanford.
12. Brambilla, M., Ceri, S., Cerizza, D., Della Valle, E., Facca, F. M., Fraternali, P., Tzi-
phase II. In Semantic Web Service Challenge 2006 Phase II, Budva, Montenegro, June
Schemas with CLIO. In System Demonstration, IEEE Data Engineering Conference,
2002.
ware Components Interactions in Service-Oriented Architectures. In IASTED Conf. on