A Software Engineering Approach to Design and Development of Semantic Web Service Applications

Marco Brambilla[1], Irene Celino[2], Stefano Ceri[1], Dario Cerizza[2], Emanuele Della Valle[2], and Federico Michele Facca[1]

[1] Politecnico di Milano, Dipartimento di Elettronica e Informazione, 20133 Milano, Italy
{Marco.Brambilla, Stefano.Ceri, Federico.Facca}@polimi.it
[2] CEFRIEL, 20133 Milano, Italy
{celino, cerizza, dellavalle}@cefriel.it

Abstract. We present a framework for designing and developing Semantic Web Service applications that span over several enterprises by applying techniques, methodologies, and notations offered by Software engineering, Web engineering, and Business Process modeling. In particular, we propose to exploit existing standards for the specification of business processes (e.g., BPMN), for modeling the cross enterprise process, combined with powerful methodologies, tools and notations (e.g., WebML) borrowed from the Web engineering field for designing and developing semantically rich Web applications, with semi-automatic elicitation of semantic descriptions (i.e., WSMO Ontologies, Goals, Web Services and Mediators) from the design of the applications, with huge advantages in terms of efficiency of the design and reduction of the extra work necessary for semantically annotating the information crossing the organization boundaries.

Keywords: Business Process Modeling, Semantic Web Services, Software Engineering, Web Engineering, Model Driven Design, Methodology.

1 Introduction

Taking the e-challenges (e-business, e-government, e-health, etc.) seriously means dealing with business processes that: *(i)* span over several enterprises; *(ii)* involve multiple actors, *(iii)* require asynchronous communication; and *(iv)* are situated in frequently changing scenarios. Current ICT solutions have serious technological and methodological limitations when addressing the abovementioned aspects; the emerging field of Semantic Web Services is offering the most promising approach to overcome such limitations, providing paradigms based on program annotation and self-descriptive implementation, for building cross-enterprise applications which favor flexibility, automatic resource discovery, and dynamic evolution. However, the development of applications based on Semantic Web Services is currently lacking a set of high level software engineering abstractions that may push the spreading of such technology. One of the main problems faced by developers to adopt Semantic Web technologies is the extra cost of semantic annotation of the developed software components. This is mostly because software engineering techniques are seldom used in the context of Semantic Web; hence, no automatic mechanism can be applied for

I. Cruz et al. (Eds.): ISWC 2006, LNCS 4273, pp. 172 – 186, 2006.

extracting semantic descriptions. Therefore, annotations are still added manually, in a very expensive and subjective manner.

In this work, we propose both a method and a toolset for fostering the adoption of Semantic Web Services (i.e., WSMO) in cross-enterprise applications. We exploit Web engineering methods, including visual declarative modeling (i.e., WebML), automatic code generation (locally and globally executable through Semantic Execution Environments such as WSMX), and automatic elicitation of semantic descriptions (i.e., WSMO Ontologies, Goals, Web Services and Mediators) from the design of the application. Global choreography (in W3C sense), front-end, and services implementations are modeled from Business Process models and WebML models, whereas goals, descriptions of Web services (i.e., capability and choreography interface), and descriptions of mediators are automatically generated. The approach also comprises the importing/ exporting of ontologies. The following techniques and notations shall be used for covering the various design aspects:

- *High-level design of the global choreography of the interaction between services:* we adopt BPMN (Business Process Management Notation) to build process models, involving several actors possibly from different enterprises.
- *Design of the underlying data model of the cross-enterprise application:* we use extended E-R (Entity Relationship) diagrams or equivalent subset of object oriented class diagrams (whose expressive power is equivalent to WSML Flight) to model the local ontology of the application and to import existing ontologies; we expose the resulting set of ontologies to the underling WSMX;
- *Design of web services interfaces, of integration platform, and of application front end:* we use visual diagrams representing Web sites and services according to the WebML models [5], including specific hypertext primitives for Web service invocation and publishing [18], and explicit representation of workflows [6].

In this way, instead of coping with textual semantic descriptions of Semantic Web Services, application developers will obtain them from the use of abstractions that are supported by software engineering tools. The use of description generators, sometimes helped by designer's annotations, guarantees the benefits of Semantic Web Services at nearly zero extra-cost, thus positioning the implemented applications within an infrastructure that allows for flexible and dynamic reconfiguration.

The paper is structured as follows: Section 2 presents a running example; Section 3 reviews the background; Section 4 presents the proposed approach to the elicitation of semantic descriptions; Section 5 briefly outlines our implementation experience; Section 6 offers a view of the related work and finally Section 7 concludes.

2 Running Example

We will consider a running example derived by the *Purchase Order Mediation* and the *Shipment Discovery* scenarios proposed at the SWS Challenge 2006 [8], properly extended to represent a classical B2B application. In this scenario, two companies, Blue and Moon, need to integrate their purchase process. In summary (Fig. 1), the architecture includes the two companies, the mediation service, a general-purpose web service built by Blue for interacting with external services, and a discovery engine.Blue

usually handles its purchase orders towards its partners by using a standard RosettaNet PIP 3A4 conversation, while the Moon partner offers a set of legacy Web Services. Blue employees want to use their usual *RosettaNet Purchase Order Interface* to interact with their counterparts in the Moon company, therefore a mediation component is needed. The mediator is in charge of *(i)* transforming the single RosettaNet message (containing all the order details) to the various messages needed by Moon to create and handle a purchase order (data mediation); and *(ii)* of translating the set of confirmation messages by Moon into a whole RosettaNet Purchase Order Confirmation to be sent back to Blue (process mediation). After completing the purchase of a set of products, Blue employees organize the shipment of the products through the *Shipment Organize Interface*. This interface is implemented by a Blue Web Service, whose internal orchestration relies on a WSMX compliant *Discovery Engine* for retrieving available shipment services, and hence needs the shipment goal to be described according to the WSMO standard. The Web Services returned by the Discovery Engine are then invoked to obtain the actual shipment offers. Finally, the system proceeds with the orchestration of the chosen service.

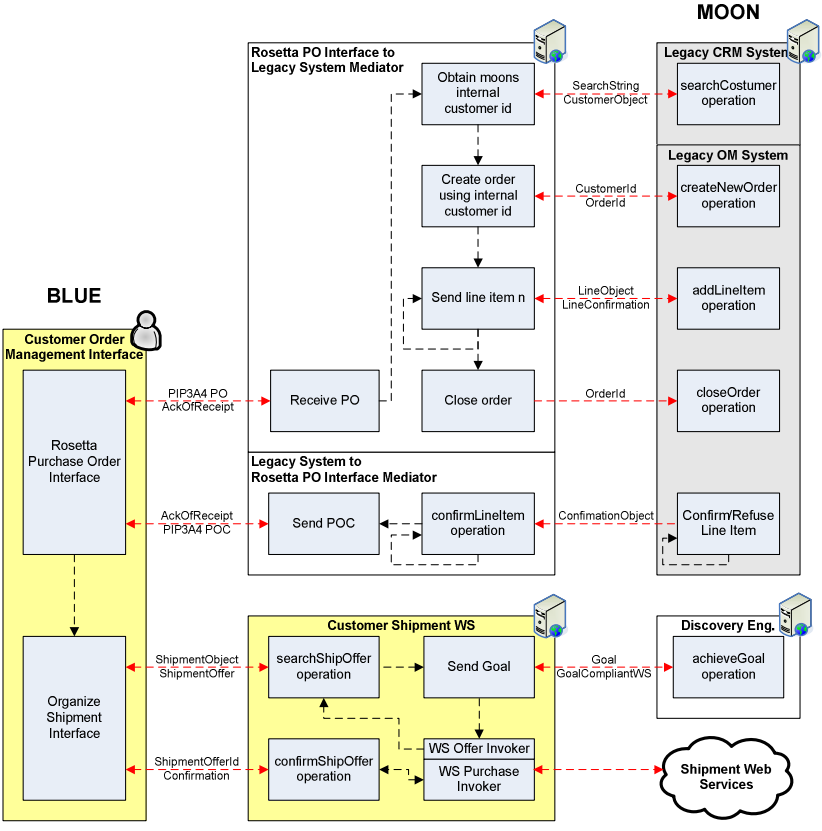

Fig. 1. The B2B scenario derived from the Semantic Web Service Challenge 2006

3 Background

Our approach relies on methodologies, tools and techniques from the fields of Software Engineering, Web Engineering, and Business Process Management.

3.1 Modeling Business Processes Using BPMN

All the B2B Web applications implement a business process, which is represented by using a workflow model. Several notations have been proposed for workflow design. We adopt Business Process Management Notation (http://bpmn.org), which is associated to the BPML standard, issued by the Business Process Management Initiative. The BPMN notation allows one to represent all the basic process concepts defined by the WfMC (http://wfmc.org) model and others, such as data and control flow, activity, actor, conditional/split/join gateways, event and exception management, and others. BPMN activities can be grouped into pools, and one pool contains all activities that are to be enacted by a given process participant. The BPMN formalization of the running case scenario can be seen in Fig. 4.

3.2 Semantic Web Service Modeling Using WSMO

The Web Service Modeling Ontology (WSMO) [23] aims at solving the application integration problem for Web services by defining a coherent technology for Semantic Web services, using four modeling elements: ontologies, Web services, goals, and mediators [13]. *Ontologies* provide the formal semantics to the information used by all other components, by describing concepts, relations, axioms, instances and so on. *Web services* represent the functional and behavioral aspects, which must be semantically described in order to allow semi-automated use. Each Web service represents an atomic piece of functionality that can be reused to build more complex ones. Web services are described in WSMO in terms of non-functional properties, functionality (capabilities), and behavior. The behavior of a Web service is described in its interface from two perspectives: communication and collaboration. A Web service can be described by multiple interfaces, but has one and only one capability. *Goals* specify objectives that a client might have when invoking a Web service. Finally, *mediators* provide interoperability facilities among the other elements, aiming at overcoming structural, semantic or conceptual mismatches between the components of a WSMO description.

3.3 Model-Driven Web Application Design Using WebML

Several Web engineering methodologies provide conceptual models, notations, and tools for the design of Web applications ([20], [14], [12], and others). In this paper, we adopt the WebML methodology [5], envisioning the following steps in the development process: *(i)* design of workflow model of the business process to be implemented; *(ii)* automatic generation of hypertext model and data model skeletons implementing the workflow; *(iii)* refinement of the produced skeletons by designers; *(iv)* automatic generation of the running Web application starting from the specified models.

The specification of a WebML application consists of a set of models: the application *data model* (an extended Entity-Relationship or UML Class Diagram), one or more *hypertext models* (i.e., different site views for different types of users), describing the

Web application structure; the *presentation model*, describing the visual aspects. The hypertext main concept is the site view, which is a graph of pages; pages are composed by units, representing publishing of atomic pieces of information, and operations for modifying data or performing arbitrary business actions. Units are connected by links, to allow navigation, parameter passing, and computation of the hypertext. The WebML service model includes a set of *Web service units* [18], corresponding to the WSDL classes of Web service operations, and components for workflow management and tracking [6].

The Web services units include *request-response* and *one-way* operations, which model services invocation, and *notification* and *solicit-response* operations, which are instead triggered by the reception of a message (thus they represent the publishing of a Web service). The model supports both the *grounding* of Web services to the XML format of Web service messages, and *data-mediation* capabilities.

WebML covers also the development Web applications implementing business processes [6], thereby supporting full-fledged collaborative workflow-based applications, spanning multiple individuals, services, and organizations. The *data model* is extended with the meta-data necessary for tracking the execution of the business process; in particular, *Case* stores information about each instantiation of the process and *Activity* stores the status of each executed activity. The *hypertext model* is extended by specifying activity boundaries and business-dependent navigation links. A*ctivities* are represented by areas tagged with a marker "A"; *workflow links* traverse the boundary of activity areas, starting or ending the activity. *Distributed processes* can be obtained by combining workflow and Web services primitives.

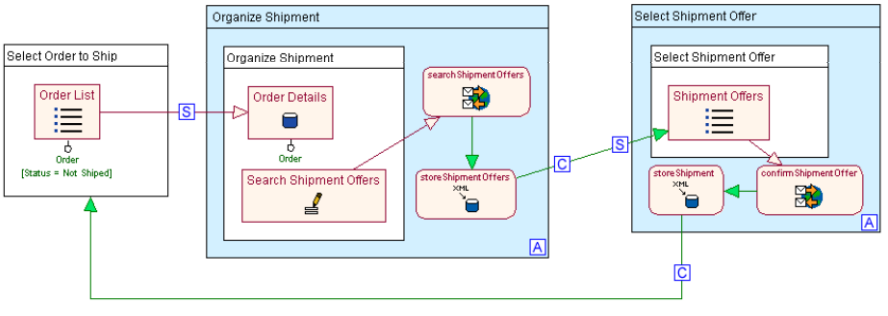

Fig. 2. The Blue Web interface to organize shipments for successful orders

Fig. 2 shows a WebML hypertext model representing a fragment of the Blue Web application: a home page (*Select Order to Ship*) allows the user to choose an *Order* (with *Status* "Not shipped") from the *Order List* index unit. When an order is chosen, the "S" link starts the Organize Shipment activity, showing the Order Details data unit and a form (*Search Shipment Offers*). The data submission triggers the invocation of a remote service (*searchShipmentOffers*), whose results are lifted by *storeShipmentOffer* XML-in. The activity is completed (link "C") and the following one is started. The *Select Shipment Offer* page is shown, containing a list of *Shipment Offers* (the results of the service call). The user chooses an offer and thus triggers the *confirmShipmentOffer*.

4 Design of Semantic Web Service Applications

This section describes our proposal for semi-automatically generating WSMO-compliant semantic specifications of a Web application. Our approach extends the WebML methodology presented in section 3.3 towards the design of semantic Web services and Web applications. Fig. 3 summarizes the envisioned development process. The main design flow, supported on conventional Web technology [6], seamlessly 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) and by delegating part of the execution to a Semantic Execution Environment (e.g. WSMX). Such models are enriched by imported ontological descriptions (on top of the figure) and are exploited for devising the set of WSMO specifications (at the bottom of the figure): the ontology is derived from BP model, data model, and hypertext model; the web services capability 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.

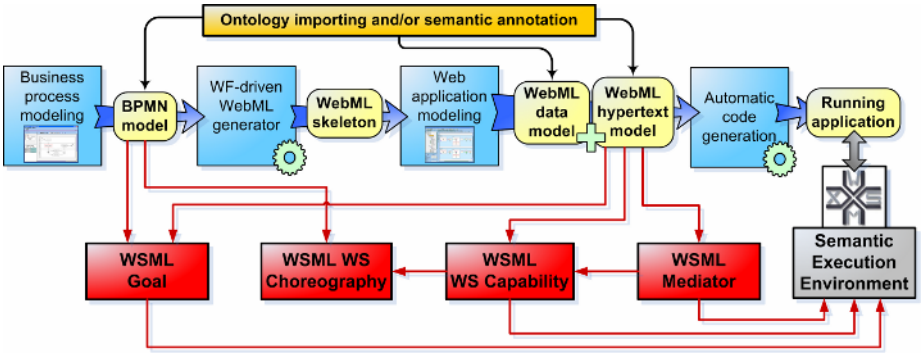

Fig. 3. Overall picture of the approach.

4.1 Design of the Business Process

The business process (BP) design task, focusing on the high-level schematization of the processes underlying the application, results in one or more BP diagrams. The reader may refer to [6] for a methodology for the design of business process-based Web applications. The BP diagram of the running case is represented in Fig. 4, with a well-defined workflow semantics (lacking in Fig. 1): for sake of clarity, the process is split into two sub-processes: part (a) describes the purchase and part (b) describes the shipment management. In the following, we will exemplify the design of the mediator of part (a), and the extraction of ontology, capability and choreography of part (b).

4.2 Design of the Data Model and Extraction of the Ontologies

The elicitation of the ontologies involved in the application is addressed by four steps, each addressing different aspects of the application ontology (see Fig. 3 again):

1. First, existing remote ontologies, possibly provided by third parties, can be imported.
2. Then, the data model is considered as a piece of ontology. This means that an appropriate transformation of the WebML data model transforms it into a WSMO-compliant ontology, which is then registered on the WSMX resource manager [23];
3. Then, the process ontology is extracted from the BPMN specification. The elements of the workflow model (e.g., activity names, lanes) are extracted as semantic concepts and used as additional piece of the ontology that will be useful in defining the state signature of the choreography interfaces of the Web services;
4. Finally, the BPMN model and the WebML data model are annotated with concepts imported from existing ontologies.

This approach is oriented towards T. Berners-Lee vision for Web applications connected by concept annotations [2].

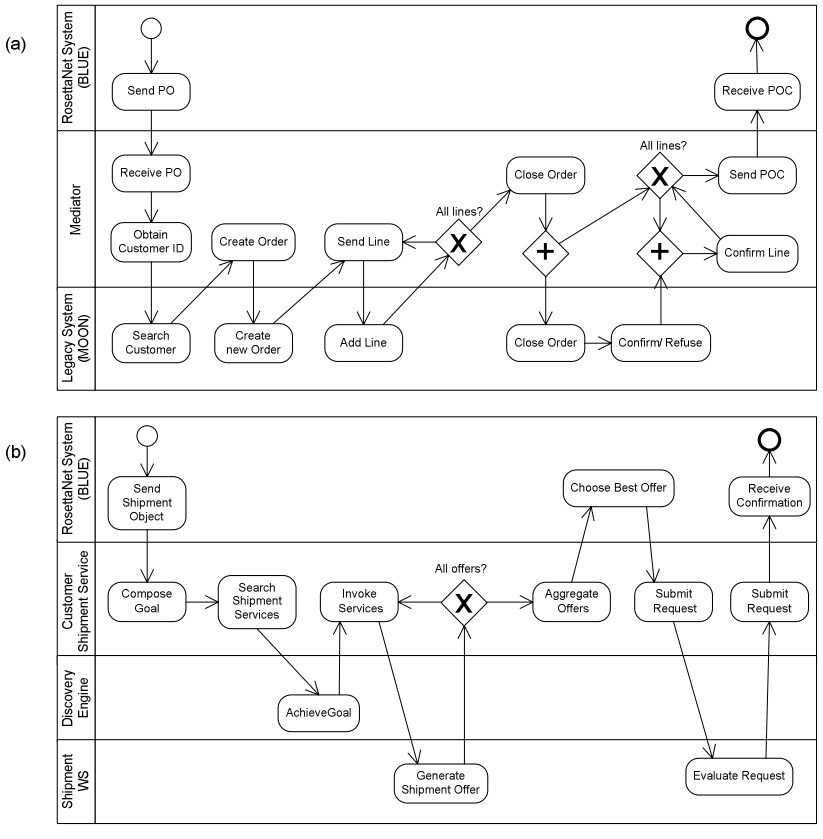

Fig. 4. Workflow representing the interaction of the running example (BPMN notation)

Fig. 5 shows the data model used by the Shipment Web Service. It includes three main domain entities: *Shipment*, *ShipmentService* (shipment partners), and *Location* (geographical places). The diagram includes *Case* and *Activity* entities described in

Section 3.3. Each *Shipment* is related to a *ShipmentService*, to an origin and a destination *Location*, and to an *Activity* indicating its current state. *ShipmentService* is connected to *Location* through the *shipTo* relationship, describing the set of possible shipment locations for each partner; the *hasLocation* relationship specifies the set of valid pick up points for each carrier.

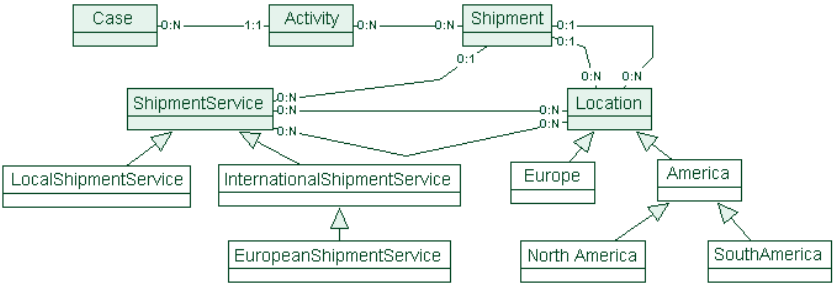

Fig. 5. A portion of the WebML data model used by the Shipment Web Service

WebML data model can be easily converted to a WSML-Flight ontology maintaining all its constraints. E.g., the *EuropeanShipmentService* entity is a sub entity of the *InternationalShipmentService* that is located in *Europe*. This subentity is described in the WebML-OQL syntax as:

```
InternationalShipmentService(as SuperEntity) where
   InternationalShipmentService.hasLocation isa Europe.
```

Its translation to WSML-Flight is:

```
concept EuropeanShipmentService subConceptOf InternationalShipmentService
   nfp dc#relation hasValue { EuShipmentServiceDef } endnfp
axiom EuShipmentServiceDef
   definedBy
?x memberOf InternationlShipmentService
and hasLocation(?x,?nation) and ?nation memberOf Europe
implies ?x memberOf EuropeanShipmentService.
```

The process of WSML ontologies generation starts by importing external ontologies used in the WebML data model to enrich WebML data types definitions. Then, for each entity in the data model, a corresponding WSML concept is generated with its direct super concept, attributes (also relationships are mapped to attributes), and possible axioms.

4.3 Design of the Service and the User Interfaces in WebML

Once the business process has been designed, workflow constraints must be turned into navigation constraints among the pages of the activities of the hypertext and into data queries on the workflow metadata for checking the status of the process. This applies both to the human-consumed pieces of contents (i.e., site interfaces) and to the machine-consumed contents (i.e., Semantic Web Services interactions).

A flexible transformation, depending on several tuning and styling parameters, has been devised for transforming workflow models into skeletons of WebML hypertext diagrams [6]. Since no a-priori semantics is implied by the activity descriptions, the generated skeleton can only implement with the hypertext and queries that are needed for enforcing the workflow constraints. The designer remains in charge of implementing the internals of each activity. Additionally, it is possible to annotate the activities, thus allowing automatic generation of a coarse hypertext that implements the specified behavior, which then needs to be refined by the designer.

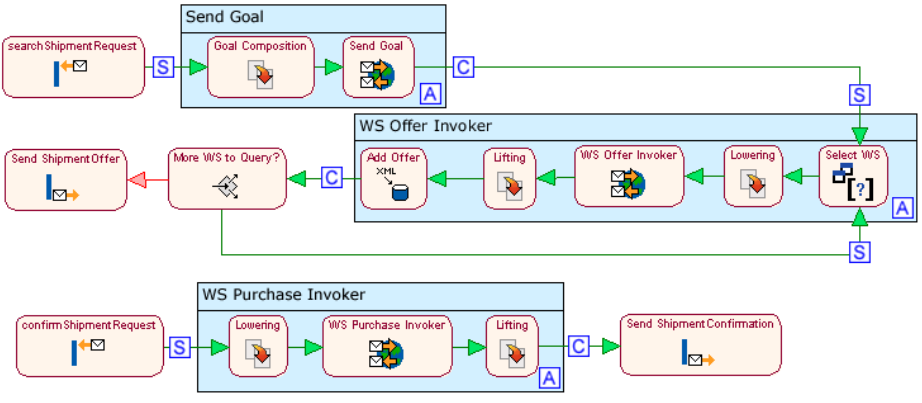

Fig. 6. The Blue Shipment Web Service

For instance, Fig. 6 shows a possible WebML specification of the Blue Shipment service. The upper part in Fig. 6 presents the *searchShipmentRequest* operation: the ShipmentObject, is passed to the *Goal Composition* that transforms it to a Goal description for the WSMX compliant Discovery Engine; the obtained goal description is passed to the *Send Goal*, which sends the goal to Web Service exposed by the Discovery Engine. The Discovery Engine returns a result with a set of Web Services compatible with the original shipment goal. For each Web Service the Lowering and Lifting operations by an appropriate XSLT Stylesheet are applied. Then, for each Web Service returned, a request for a shipment offer is made. The results are combined and converted to the Blue data model and the set of offers is returned the service requester. Once the service requester selects one of the offers and he sends it to the *confirmShipmentRequest* operation (lower part of Fig. 6), the offer is purchased by invoking the appropriate Web Service and the confirmation message is sent back.

4.4 Extraction of the Description of the Web Services

Another important aspect that can be semi-automatically derived from the design specification is the description of Web services. Some information about the services can be directly extracted by the high-level BPMN description of the interactions (in particular, information about possible choreography of the service and basic interface and parameter specification). More details can be elicited from the WebML diagrams, which provide a more refined representation of the specification of the application.

Extraction of Web Services capabilities. The BPMN and WebML models of the Web services provide enough information for describing its behavior. Assuming a BPMN activity as an atomic Web service call, we can exploit the BPMN *data flow* for providing good hints for the extraction of inputs and outputs of the service. Indeed, the data flow specifies the objects that are passed between the various activities. By isolating a single activity, it is possible to automatically extract the WSML pre-conditions (inputs) and post-conditions (outputs). However, designer refinements are then typically required.

WSML pre-conditions are obtained from the first unit of WebML chain describing a Web Service operation (Solicit Unit), while post-conditions are obtained from the last one (Response Unit). These two units contain information about the exact structure of the exchanged message and eventually the mapping of message elements to the domain model and hence to the extracted ontologies (see Section 4.2). Effects are extracted by searching for WebML units that modify or create instances of entities that are related to the activities involved by the process described in WebML Web Service. Shared variables are obtained from the generated conditions by grouping all the variables involved in the operations data flow.

The following WSML description of the Web Service capabilities is automatically generated once the WebML models are fully specified.

```
capability
   sharedVariables (?Req)
   precondition
      definedBy
         (?Req memberOf searchShipmentRequest) or
         (?Req memberOf ConfirmShipmentRequest).
   postcondition
      definedBy
         (?Req[
         pickupdate hasValue ?pkd, deliverydate hasValue ?dd,
         start hasValue ?s, destination hasValue ?dest,
         weight hasValue ?w, maxCost hasValue ?maxc
         ] memberOf searchShipmentRequest)
         implies
         exists ?Res (
            ?Res memberOf ShipmentOfferContainer and
               forall ?offer (
               ?Res [offers hasValue ?offer]
               implies (
                  ?offer [
                     offerID hasValue ?OID, pickupdate hasValue ?pkd,
                     deliverydate hasValue ?dd, start hasValue ?s,
                     destination hasValue ?dest, weight hasValue ?w,
                     cost hasValue ?c] memberOf ShipmentOffer
                  and ?c<=?maxc
         )))) and
         (?Req[ offerID hasValue ?OID] memberOf ConfirmShipmentRequest)
         implies
            exists ?Confirmation (
               ?Confirmation[
                  offerID hasValue ?OID, confirmationID hasValue ?CID
               ] memberOf ShipmentConfirmation
         ))
```

Extraction of the service choreography. The service choreography is a piece of information that typically requires some annotation by the designer, in order to establish all the possible interaction sequences with the service. However, at least one of the choreography sequences can be extracted from the BPMN model, by analyzing the order of invocation of the different operations of the service. Obviously, this does not guarantee that all the possible scenarios are considered, since only one enactment can be analyzed. The extraction of this kind of information is rather simple: provided that a lane describes a single Web service, we can assume that all the control flow links traversing its borders contribute to specifying a possible invocation order of the operations, i.e., a choreography interface of the Web service. The automatically generated WSML description of the Web Service choreography is the following:

```
interface
   choreography
      stateSignature
         in
            searchShipmentRequest withGrounding […]
            ConfirmShipmentRequest withGrounding […]
         out
            ShipmentOfferContainer withGrounding […]
            ShipmentConfirmation withGrounding […]
         controlled oasm#ControlState
         transitionRules
            forall {?x, ?state} with (
               ?state[oasm#value hasValue oasm#InitialState]
                  memberOf oasm#ControlState and
               ?x memberOf ShipmentRequest
            ) do
              add(?state[oasm#value hasValue ShipmentOfferRequested])
              delete(?state[oasm#value hasValue oasm#InitialState])
              add(_# memberOf ShipmentOfferContainer)
            endForall
            forall {?x, ?state} with (
              ?state[oasm#value hasValue ShipmentOfferRequested] and
              ?x memberOf ConfirmShipmentRequest) do
               add(_# memberOf ShipmentConfirmation)
            endForall
```

4.5 Extraction of User's Goal

Extraction of user's goals can be performed by combining information available at the BPMN level with information available at the WebML level. A first level of goal elicitation can be achieved by extracting the sequence of conditions and objects passed to the Web services by the user's lane in the BPMN diagram.

A deeper level of details requires using the WebML hypertext models and analyzing the semantics embedded in the navigation and composition of the pages. Such refined goal is detailed in terms of the tasks performed by the user and of the data manipulated, thus increasing the significance of the WSMO goals that can be generated. In this case we omit the automatically generated code due to space limitation.

4.6 Design of wwMediators with WebML

One of the main strength points of the approach is the ease of design and implementation of complex wwMediators. If a lane is identified as a wwMediator at

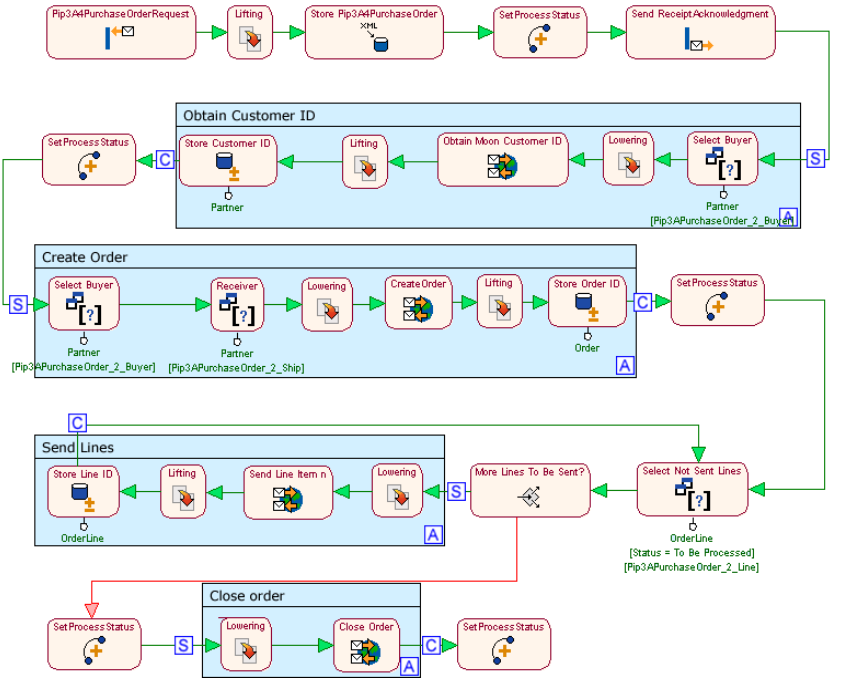

Fig. 7. The WebML model of wwMediator Web Service

the BPMN level, the basic information about the design of the mediation services can be extracted from the high-level BPMN description of the interactions (in particular, information about possible choreography of the service and basic interface and parameter specification). The skeleton model of the mediator is automatically generated and the designer can refine it at a conceptual design level. Then, the WSMO description of the mediator can be derived from the WebML diagrams.

Fig. 7 presents the detailed specification of the wwMediator within WebML. This specification can be used to generate a working Web Service providing mediation between Blue and Moon Web Service. The WebML specification includes some *Lowering* and *Lifting* operations corresponding to WSMO ooMediators and provides mediation between the data model of the source Web Service and the destination one. In WebML this mediation consists in XSLT stylesheets generated by a visual tool.

5 Implementation Experience

The presented approach relies on solid implementation of the background concepts: the WebML methodology is supported by a commercial CASE tool called WebRatio (www.webratio.com), providing visual design interfaces and automatic code generation; the modeling of the business process requirements and their transformation into WebML skeletons are implemented in a prototype tool [3].

A proof of concepts of the integration with the semantic aspects discussed in this paper has been presented at the SWS Challenge 2006 [4, 8]. The first phase of the challenge allowed us to prove the advantages of a Software Engineering approach to Semantic Web Services design. We presented the WebML design and implementation of the wwMediator of the running case addressed in this paper (Fig. 9) and the usage of the CASE tool WebRatio in the context of Semantic Web applications. For validating our approach, we developed several prototypical transformers that generate WSMO-compliant descriptions of Web applications and services starting from WebML models of the applications and BPMN specifications of the processes. The pieces of WSMO specification presented in Sections 4.2 and 4.4 are samples of the generated output of the transformations.

6 Related Work

The Semantic Web is a quite new research area that grew up quickly and in few years produced a great number of publications. However, few of them concern the systematic and methodological development of Semantic Web applications. Some early proposals (e.g., [9]) offered the definition of UML profiles for easily handling ontological definitions; however they haven't been adopted because of the lack of an overall methodology. A number of researches concentrated on the development of tools to support the generation of semantic descriptions for existing Web Services [17, 22, 10]. Most of these tools still require the learning of the annotation language used (e.g., OWL-S or WSMO) and hence do not push enough the adoption of Semantic Web Services towards the standard software development. Furthermore, they do not exploit the advantages of conceptual models of the Web Services to semi-automatically derive any part of the semantic descriptions.

Our research effort is more similar to the recent efforts of the Object Management Group (http://www.omg.org). The OMG proposed the Ontology Definition Metamodel (ODM) [19] to define a suitable language for modeling Semantic Web ontology languages and hence Semantic Web applications in the context of the Model Driven Architecture (http://www.omg.org/mda). In [1] MIDAS, a framework based on MDA to model and develop Semantic Web applications, is introduced. The framework proposed focuses on the creation of Semantic Web Services and associated WSML descriptions using a UML model according to the MDA approach. This proposal inherits the limits of the MDA approach: the use of a UML model is not always fitting the Semantic Web needs, and often the model is too far from the implementation details to provide an effective automatic code generation. Furthermore, MIDAS does not provide a clear overall roadmap to the design of Semantic Web applications.

Other research efforts are converging on the proposal of combining Semantic Web Services (SWS) and Business Process Management (BPM) to create one consolidated technology, which we call Semantic Business Process Management (SBPM) [16]. This is based on the fact that mechanization of BPM can be addressed through machine-accessible semantics, that can be naturally provided by SWS frameworks (e.g., WSMO).

In the last years, realizing the benefits of the Semantic Web platform, some research from the Web Engineering field is spent to design a methodology to develop Semantic Web Information Systems. Traditional Web design methodologies (like OOHDM [20]) and new approaches (like Hera [21]) are now focusing on designing

Semantic Web applications. However, these methodologies are not supported by an effective CASE tool and do not consider the development of Semantic Web Services; instead, they concentrate only on Semantic Web Portals.

7 Conclusions and Future Work

This paper presented an approach for designing Semantic web applications exploiting software engineering techniques. The following results have been shown:

- ontologies can be imported as models of the data necessary for the cross-enterprise application. They can be extended for addressing the specific needs of the application and registered as shared resources in WSMX.
- WSMO Web Services functional capabilities for delegating sub-processes execution from one enterprise to another are automatically provided for each Web Service modelled in WebML. Choreography interfaces can be derived by combining information in the Business Process Model and at application level in the hypertext model of WebML. In particular, service (local) choreography can be derived by taking the point of an external observer of the Web Services that must know the order in which operation can be invoked and the constrains for their successful invocation. In a similar manner we plan to derive an orchestration interface by translating in WSMO the hypertext model of the application.
- WSMO goals can be produced (e.g., goals that triggers the discovery component of WSMX) from gathering data required to perform a given action of the business process, whereas its choreography interface is derived by the explicit representation of workflow primitives within the hypertext.
- mediation services (except for ontology-to-ontology mediation) can be modeled as WebML applications and registered in WSMX according to their roles (e.g., a wwMediator).

At the current stage of development, we propose using existing software engineering abstractions for the semi-automatic extraction of the components of the WSMO architecture. Thus, by means of "conventional design" (although supported by an advanced visual design studio), we build software that can run on conventional Web technology and at the same time is ready to become part of a WSMO execution environment (i.e. WSMX). Our next steps, which we will do in parallel with the wide-spreading and enhancement of WSMO standards, will concentrate upon empowering our design abstractions so as to further improve and simplify the design of native WSMO components.

References

1. Acuña, C. J., Marcos, E.: *Modeling semantic web services: a case study*. In Proceedings of the 6th International Conference on Web Engineering (ICWE 2006), Palo Alto, California, USA, 32-39.
2. Berners-Lee, T.: Web Services - Semantic Web Talk. http://www.w3.org/2003/Talks/08-mitre-tbl

3. Brambilla, M.: Generation of WebML Web Application Models from Business Process Specifications. 6th International Conference on Web Engineering (ICWE) 2006, Palo Alto, ACM press, p. 85-86, 2006.
4. 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 SWS Challenge I , 2006, Palo Alto, CA. (http://sws-challenge.org/wiki/index.php/Workshop_Stanford)
5. Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., Matera, M.: Designing Data-Intensive Web Applications, Morgan-Kaufmann, December 2002.
6. Brambilla, M., Ceri, S., Fraternali, P., Manolescu, I.: Process Modeling in Web Applications. In ACM Transactions on Software Engineering and Methodology (TOSEM), 2006. In print.
7. Della Valle, E. and Cerizza, D.: The mediators centric approach to automatic webservice discovery of Glue. In MEDIATE2005, volume 168 of CEUR. Workshop Proceedings, 35–50.
8. DERI Stanford. Semantic Web Services Challenge 2006. http://sws-challenge.org.
9. Djurić , D., Gašević , D., Devedžić, V. , Damjanović , V.: *UML Profile for OWL*. 4th International Conference on Web Engineering (ICWE 2004), (LNCS 3140, Springer-Verlag), pp. 607-608, 2004.
10. Elenius D., Denker G., Martin D., Gilham F., Khouri J., Sadaati S., Senanayake R.: *The owl-s editor – a development tool for semantic Web services*. In 2nd European Semantic Web Conference, May 2005.
11. Feier, C., Domingue, J.: *WSMO Primer*. http://www.wsmo.org/TR/d3/d3.1/v0.1/
12. Fernandez, M.F., Florescu, D., Levy, A.Y., Suciu, D.: Declarative Specification of Web Sites with Strudel. In VLDB Journal, 9 (1), 38-55.
13. Fensel, D., Bussler, C.: *The Web Service Modeling Framework WSMF*. Electronic Commerce Research and Applications, 1(2), 2002.
14. Fons, J., Pelechano, V., Albert, M. and Pastor, Ó. Development of Web Applications from Web Enhanced Conceptual Schemas. In ER 2003, LNCS, 2813, 232-245.
15. Garrigós, I., Gómez, J. and Cachero, C., Modelling Dynamic Personalization in Web Applications. In ICWE 2003, 472-475.
16. Hepp, M., Leymann, F., Domingue, J., Wahler, A., Fensel, D.: *Semantic Business Process Management: A Vision Towards Using Semantic Web Services for Business Process Management.* In Proceedings of the IEEE ICEBE 2005, October 18-20, Beijing, China, 535-540.
17. Jaeger M., Engel L, Geihs K.: *A methodology for developing owl-s descriptions*. 1st Int. Conf. on Interoperability of Enterprise Software and Applications. Workshop on Web Services and Interoperability. February 2005.
18. Manolescu, I., Brambilla, M., Ceri, S., Comai, S., Fraternali, P.: Model-Driven Design and Deployment of Service-Enabled Web Applications. In ACM TOIT, Vol. 5, number 3 (August 2005).
19. OMG: Ontology Definition Metamodel (ODM). http://www.omg.org/cgi-bin/doc?ad/06-05-01.pdf
20. Schwabe, D. and Rossi, G. The Object-Oriented Hypermedia Design Model. In Communications of the ACM, 38 (8), 45-46.
21. Vdovjak, R., Frasincar, F., Houben, G. J., Barna, P.: *Engineering semantic web information systems in Hera*. Journal of Web Engineering, Rinton Press, 2(1-2), 3 -26, 2003.
22. Web Service Modeling Toolkit. http://sourceforge.net/projects/wsmt
23. WSMO: Web Service Execution Environment (WSMX). http://www.w3.org/Submission/WSMX.