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], 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. This paper presents a framework for designing and developing Semantic Web Service applications that spans over several enterprises by applying techniques, methodologies, and notations offered by other fields, namely 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 the crosses the organizational boundaries.

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

1 Introduction

Taking the e-challenges (i.e., not only e-business, but also 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 scenarios which change frequently. 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 business processes 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.

In this work, we propose both a method and a toolset for fostering the adoption of Semantic Web Services (i.e., WSMO) in developing cross-enterprise applications. We exploit modern Web engineering methods, including visual declarative modeling

(based on the WebML conceptual model), automatic code generation (locally executable using standard Web technologies and globally executable across enterprise boundaries by delegating the execution to a Semantic Execution Environment 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[1], front-end, and services implementations are modeled from Business Process models and WebML models, whereas goals, descriptions of web services (both in terms of capabilities and of their choreography interface), and descriptions of mediators are automatically generated. Concerning ontologies, it's possible to import ontologies as part of the application data model, to refine them and to export the resulting ontologies. In particular, we propose to cover the different aspects of the design by means of the following techniques and notations:

- *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 (whose expressive power is equivalent to WSML Flight) to model the local ontology of the application and to import existing ontologies, whenever they exist; 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 [9], including specific hypertext primitives for Web service invocation and publishing [2] and explicit representation of workflow concepts [3].

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, guarantee the benefits of Semantic Web Services at nearly zero extra-cost, thus positioning their cross-enterprise applications within an infrastructure that allows for flexible and dynamic reconfiguration.

The paper is structured as follows: Section 2 presents the running example that will be discussed throughout the paper; Section 3 offers a view of the related work and on the background of the research; Section 4 presents the proposed approach to the elicitation of semantic description of the application; Section 5 briefly outlines our implementation experience; and finally Section 6 concludes.

2 Running example

For the discussion we will consider a running example derived by the *Purchase Order Mediation* scenario and the *Shipment Discovery* scenario proposed at the Semantic Web Service Challenge 2006 [8], properly extended to show all the components of a classical B2B application. In this scenario, two big companies, Blue and Moon, need to integrate their processes in order to create a new business partnership. In summary, as displayed

[1] The term choreography assumes several meanings in different communities. We refer to W3C definition of choreography with the term "global choreography" (i.e., the choreography of an application made of WS), whereas we refer to WSMO choreography definition with the term "local choreography" (i.e., the choreography interface of a Web Service).

by Fig. 1, the architecture includes the two companies Blue and Moon, the mediation service between them, and a general-purpose web service built by Blue for interacting with external services and an external 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 for products purchase. Blue employees, in the Purchase department, want to "talk" in a transparent way with their counterparts in the Moon partner – using their usual *RosettaNet Purchase Order Interface*, therefore a mediation component is needed between the two. 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; and *(ii)* of translating the set of confirmation messages by Moon into a whole RosettaNet Purchase Order Confirmation to be sent back to Blue. Thus, the mediation requires a data mediation a relevant process mediation between the two totally different purchase processes of RosettaNet and Moon legacy process. After completing the purchase of a set of products, Blue employees organize the shipment of the products through the *Shipment Organize Interface*. This interface relies on a Web Service internally developed by Blue and offered to Blue partners too. The internal orchestration of the Web Service relies on a WSMX compliant *Discovery Engine* for retrieving available shipment services, and hence

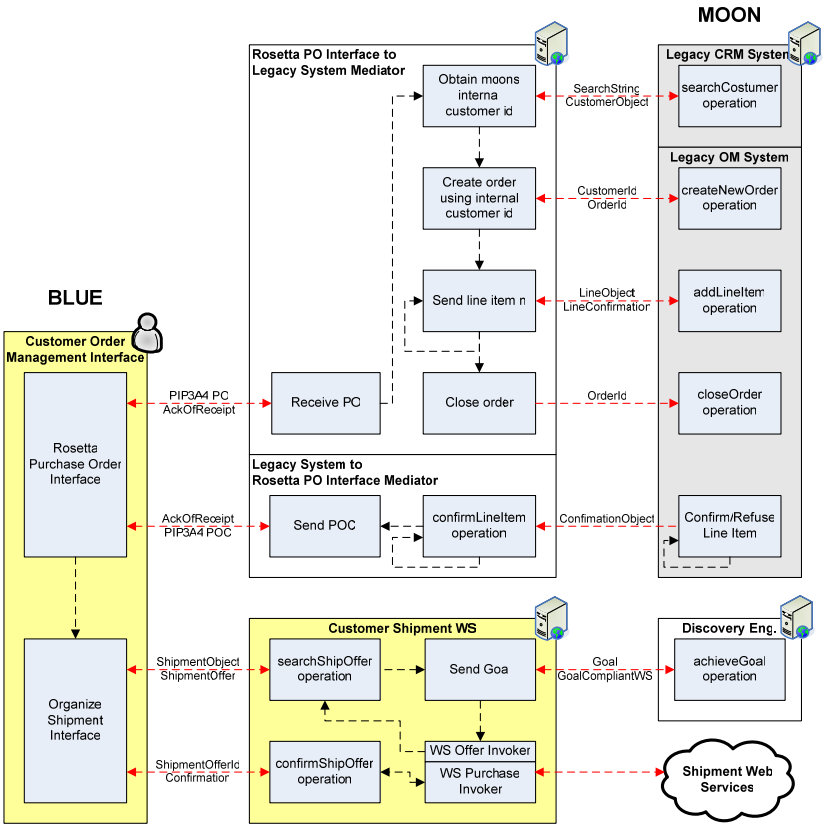

Fig. 1. The B2B scenario derived from the Semantic Web Service Challenge 2006.

needs to describe the shipment goal according to the WSMO standard. When the Discovery Engine returns a list of Web Services offering a shipment service compatible with the original Goal, the Shipment Web Service invokes the returned Web Services to obtain actual shipment offers and proceeds with its orchestration.

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 WSMO

The Web Service Modeling Ontology (WSMO) [17] 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 [12]. *Ontologies* provide the formal semantics to the information used by all other components, thus linking machine and human terminologies. The WSMO constituents of an ontology are *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 from three different points of view: non-functional properties, functionality (described as *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. In WSMO, a goal is characterized in a dual way with respect to Web services: goal's descriptions include the *requested capability* and the *requested interface*. Finally, *mediators* provide interoperability facilities among the other elements, aiming at overcoming structural, semantic or conceptual mismatches that appear between the components that build up a WSMO description.
Current 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) [15]. The claim is based on the observation that the problem of mechanization of BPM, can be

traced back to the lack of machine-accessible semantics, and that the modeling constructs of SWS frameworks, especially WSMO, are a natural fit to creating such a representation.

3.3 Model-driven Web application design

Several approaches in the Web engineering field provide methodologies, conceptual models, notations, and tools for the design of Web applications. Among them, we can cite OO-HDM [16], OO-Method [13], Strudel [11], and WebML [6]. In this paper, we will adopt the WebML methodology, envisioning the following steps in the development process: *(i)*design of workflow model representing the business process model to be implemented; *(ii)*automatic generation of a set of hypertext model and data model skeletons implementing the specifications of 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* [6] consists of a set of models: the application *data model* (an extended Entity-Relationship model), one or more *hypertext models* (i.e., different site views for different types of users), expressing the navigation paths and the page composition of the Web application; the *presentation model*, describing the visual aspects of the pages. 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 (e.g. sending e-mails). Units are connected by links, to allow navigation, parameter passing, and computation of the hypertext. WebML conceptual model has been extended with a service model that includes a set of *Web service units* [2, 9], corresponding to the WSDL classes of Web service operations, and components for workflow management and tracking. The Web services units include *Request-response* and *one-way* operations, which model services invocation and are triggered by the navigation of an input link. *Notification* and *solicit-response* are instead triggered by the reception of a message, thus they represent the publishing of a Web service, which is exposed and can be invoked by third party applications. 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 of B2B Web applications implementing business processes, 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, the *Case* entity stores information about each instantiation of the process and the *Activity* entity stores the status of each activity instance executed. Each Activity is connected to a single Case. Connections to users and application data associate domain information to the process execution. 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 and are associated with workflow logic: every link entering an activity starts or resumes the execution of the activity; every outgoing link ends or suspends the activity. *If* and *switch* units can be used to express navigation conditions. *Distributed processes* can be obtained by combining the workflow extensions and Web services extensions [3].

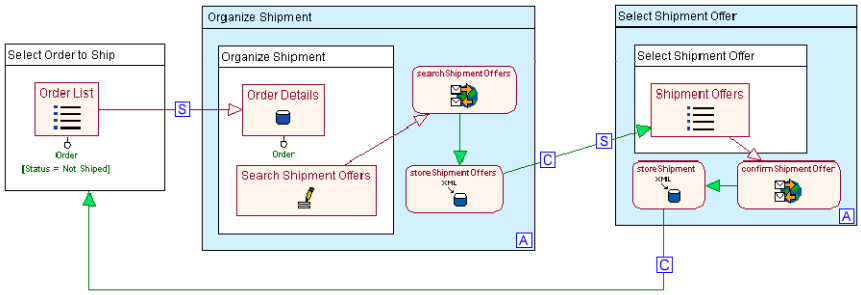

Fig. 2. The Blue Web interface to organize shipments for successful orders.

Fig. 2 shows a WebML hypertext model representing a fragment of the Blue Web application: a home page called *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 in the *Organize Shipment* page, together with a form (*Search Shipment Offers*). The data submission triggers the invocation of a remote service (*searchShipmentOffers* request-response unit), whose results are lifted by the *storeShipmentOffer* XML-in unit. The activity is completed (link "C") and following one is started. The *Select Shipment Offer* page is shown, containing a list of *Shipment Offers* (an index unit displaying the results of the service). The user chooses an offer and thus triggers the *confirmShipmentOffer* request response, whose results are stored locally. Finally, the user is redirected to the home page.

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 [3], 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). Such models are enriched by imported ontological descriptions (on top of the figure) and are exploited for devising

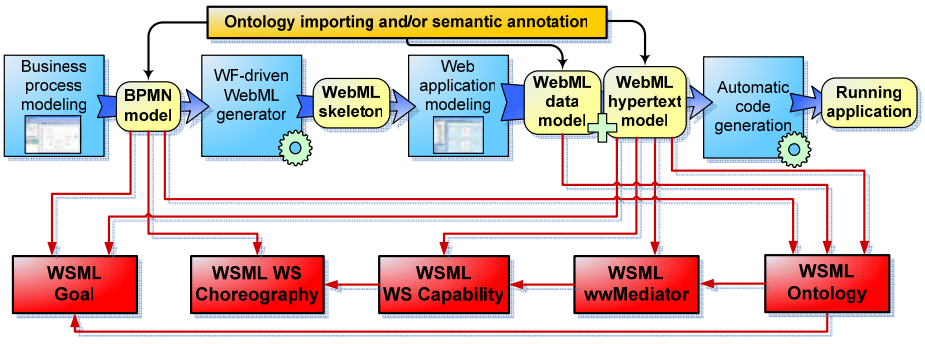

Fig. 3. Overall picture of the approach.

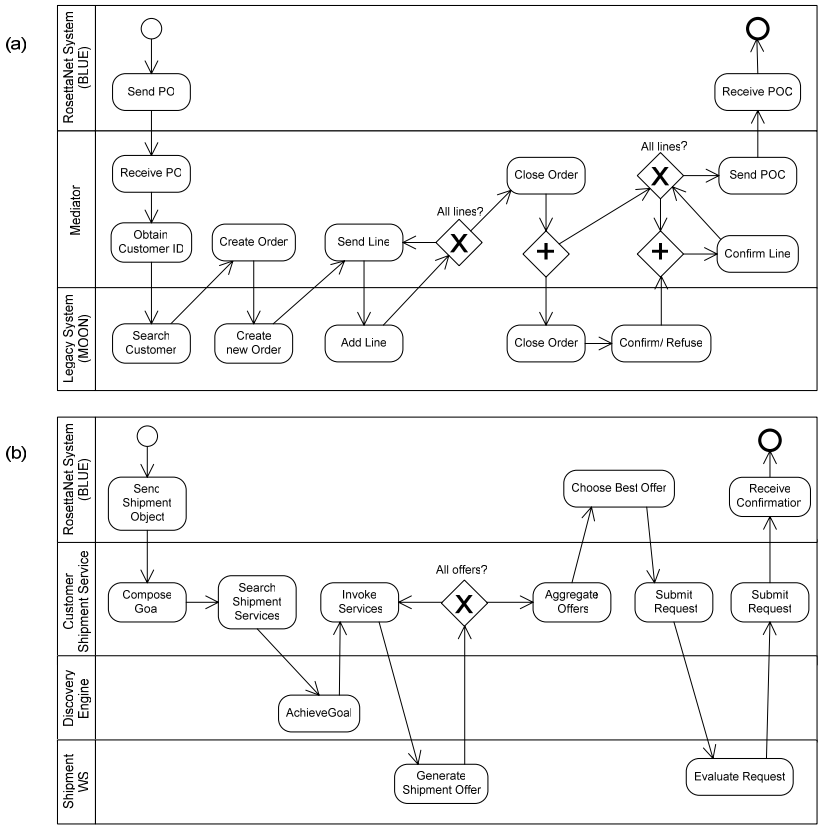

Fig. 4. Workflow representing the interaction of the running example (BPMN notation).

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.

4.2 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 BP diagram of the running case is represented in Fig. 4: 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). Note that the two workflow diagrams have well-defined workflow semantics while the representation of Fig. 1 doesn't.

4.3 Design of the data model and extraction of the ontologies

The elicitation of the ontologies involved in the application is addressed by four distinct steps, each addressing different aspects of the application ontology:

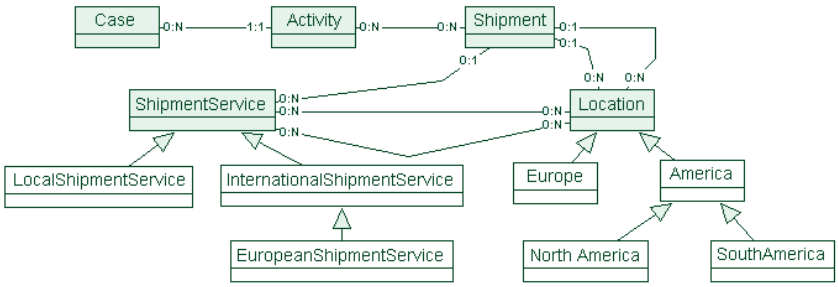

Fig. 7. A portion of the WebML E-R diagram used by the Shipment Web Service.

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 transform the extended E-R internal representation of the data into a WSMO-compliant ontology, which is then registered on the WSMX resource manager [17];
3. Then, the process ontology is extracted from the BPMN specification of the underlying business process of the application. The elements of the workflow model (e.g., activity names, lanes, and so on) are extracted as semantic concepts and used to build an 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 concepts annotations [1].

Fig. 7 shows the E-R diagram used by the Shipment Web Service to describe goals and to invoke external Shipment Web Services. The E-R has tree main domain entities *Shipment*, descpribing each shipping, *ShipmentService*, describing BLUE shipment partners, and *Location*, describing the geographical entities involved in the shipment process. The E-R diagram includes also the two process entities used to describe the status of the process (see Section 3.3). Each *Shipment* instance is related to a *ShipmentService*, to a origin and a destination *Location* and, finally to an *Activity* to indicate its currente state. The *ShipmentService* entity is connected to the *Location* entity through the *shipTo* relationship, i.e. each shipment partner has a set of possible shipment locations, and through the *hasLocation* relationship, each carrier has a set of valid pick up points. Both the *Location* and the *ShipmentService* entities have some sub entities in order to easily specialize their characteristics.
WebML E-R 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 E-R model to enrich WebML data types definitions. Then, for each entity in the E-R, a correspondent WSML concept is generated with its super direct concept, attributes (also E-R relationships are mapped to attributes) and possible axioms.

4.4 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, thus ensuring that the data shown by the application and user navigation respect the constraints described by the process specification. 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. The produced WebML model shall consist of application data model, workflow metadata, and hypertext diagrams. Since no a-priori semantics is implied by the activity descriptions, the generated skeleton can only implement the empty structure of each activity along 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 the designer to map the activity to a coarse hypertext that implement the specified behavior on the specified data. The designer is in charge of refined specification of services and hypertexts. For instance, Fig. 6 shows a possible WebML specification of the services of the Blue Shipment service.

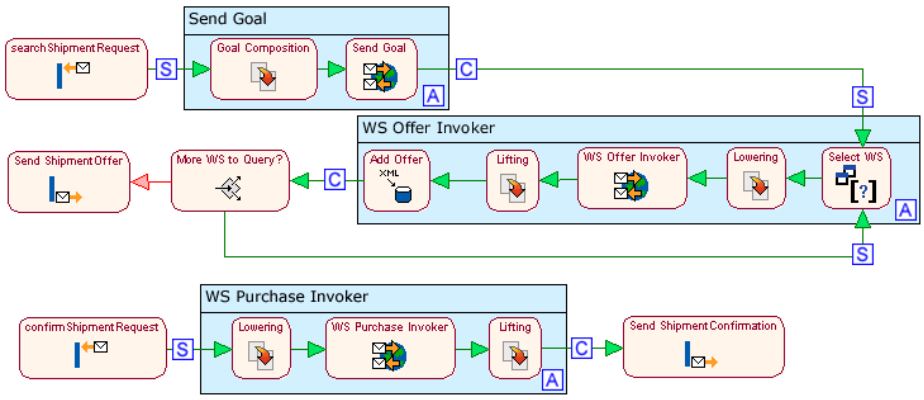

Fig. 6. The Blue Shipment Web Service.

The upper part in Fig. 6 present the *searchShipOffer* operation: the ShipmentObject, is passed to the *Goal Composition* that transform 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 and for each Web Service the description of the Lowering and Lifting operations by an appropriate XSLT Stylesheet. 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 *confirmShipOffer* operation (lower part in the Fig. 6), the offers is purchased by invoking the appropriate Web Service and the confirmation message is sent back to the service requester.

4.5 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, that 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 behaviour. 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 extract the WSML pre-conditions (inputs) and post-conditions (outputs).

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 4.4). Effects are extracted 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 different 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 one the WebML are fully specified.

```
capability
   sharedVariables (?Req)
   precondition
      definedBy
         (?Req memberOf ShipmentRequest) or
         (?Req memberOf ConfirmShipOfferRequest).
   postcondition
      definedBy
         (?Req[
         pickupdate hasValue ?pkd, deliverydate hasValue ?dd,
```

```
        start hasValue ?s, destination hasValue ?dest,
        weight hasValue ?w, maxCost hasValue ?maxc
        ] memberOf ShipmentRequest)
        implies
        exists ?Res (
           ?Res memberOf ShipOfferContainer 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 ConfirmShipOfferRequest)
        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 following is the WSML description of the Web Service choreography interface automatically generated.

```
interface
   choreography
      stateSignature
         in
            ShipmentRequest withGrounding […]
            ConfirmShipOfferRequest withGrounding […]
         out
            ShipOfferContainer 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 ConfirmShipOfferRequest) do
               add(_# memberOf ShipmentConfirmation)
            endForall
```

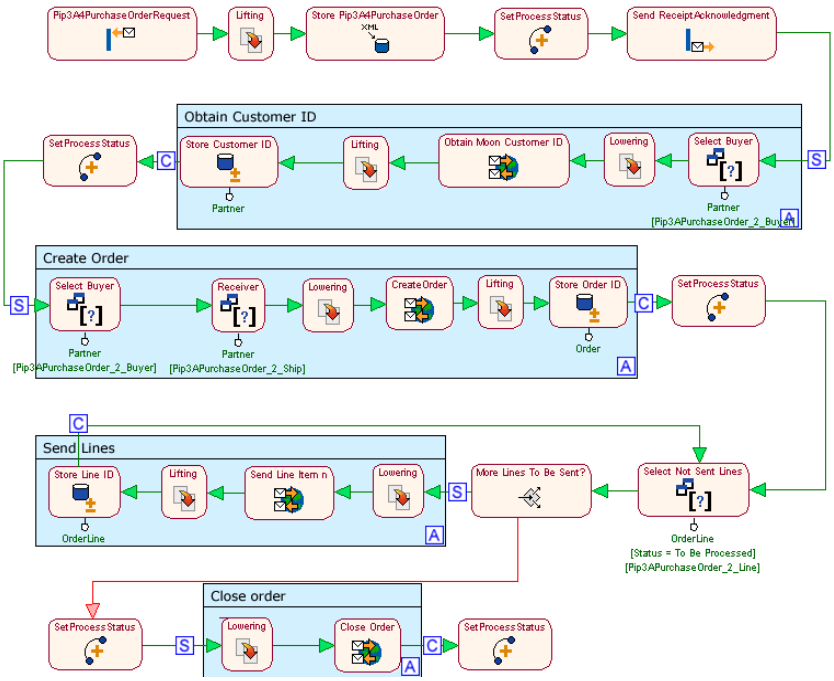

Fig. 9. The WebML model of wwMediator Web Service.

4.6 Extraction of user's goal

Extraction of user's goals can be performed 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.

4.7 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 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. 9 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 graphic 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 [4]. A proof of concepts of the integration with the semantic aspects discussed in this paper has been presented at the SWS Challenge 2006 [5, 8]. The first phase of the challenge allowed us to prove the advantages of a Software Engineering approach to Semantic Web Services design. In particular, 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 4.3, 4.5, and 4.7 are samples of the generated output of the transformations.

6 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, 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. Berners-Lee, T.: Web Services - Semantic Web Talk. http://www.w3.org/2003/Talks/08-mitre-tbl
2. Brambilla, M., Ceri, S., Fraternali, P., Acerbis, R., Bongio, A.: Model-driven Design of Service-enabled Web Applications. In SIGMOD 2005, Industrial Track.
3. Brambilla, M., Ceri, S., Fraternali, P., Manolescu, I.: Process Modeling in Web Applications. In ACM TOSEM, 2006. In print.
4. Brambilla, M., Generation of WebML Web Application Models from Business Process Specifications, in: International Conference on Web Engineering – ICWE 2006, in print.
5. 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 2006 Phase I, Stanford University, Palo Alto, CA.
   Available at: http://sws-challenge.org/wiki/index.php/Workshop_Stanford.
6. Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., Matera, M.: Designing Data-Intensive Web Applications, Morgan-Kaufmann, December 2002.
7. Della Valle, E. and Cerizza, D.: The mediators centric approach to automatic webservice discovery of Glue. In Hepp, M., Polleres, A., van Harmelen, F., Genesereth, M.R., editors, MEDIATE2005, volume 168 of CEURWorkshop Proceedings, 35–50.
8. DERI Stanford. Semantic Web Services Challenge 2006. http://sws-challenge.org.
9. Manolescu, I., Brambilla, M., Ceri, S., Comai, S., Fraternali, P.: Model-Driven Design and Deployment of Service-Enabled Web Applications. In ACM TOIT, Volume 5, number 3 (August 2005).
10. Feier, C., Domingue, J.: *WSMO Primer*. http://www.wsmo.org/TR/d3/d3.1/v0.1/
11. Fernandez, M.F., Florescu, D., Levy, A.Y., Suciu, D.: Declarative Specification of Web Sites with Strudel. In VLDB Journal, 9 (1), 38-55.
12. Fensel, D., Bussler, C.: *The Web Service Modeling Framework WSMF*. Electronic Commerce Research and Applications, 1(2), 2002.
13. 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.
14. Garrigós, I., Gómez, J. and Cachero, C., Modelling Dynamic Personalization in Web Applications. In ICWE 2003, 472-475.
15. 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.
16. Schwabe, D. and Rossi, G. The Object-Oriented Hypermedia Design Model. In Communications of the ACM, 38 (8), 45-46.
17. WSMO: Web Service Execution Environment (WSMX). http://www.w3.org/Submission/WSMX.