Integrating Semantic Web Services and Business Process Management: A Real Use Case*

Christian Drumm, Jens Lemcke, and Kioumars Namiri

SAP Research Center CEC Karlsruhe
SAP AG
firstname.lastname@sap.com

Abstract. In this paper we aim to investigate how semantic Web services can improve standard business process management tools. Using a standard SAP process in the area of logistics as an example, we show the limitations of current approaches, both during design- and runtime. Based on this investigation we will present a new solution enabling the automation of certain task in the process of creating cross organizational business processes using ontologies and Semantic Web Services.

1 Introduction

In this paper we aim to investigate how semantic Web services can improve standard Business Process Management (BPM) software. Based on a standard SAP process in the area of logistics, we show the limitations of current approaches and present a solution based on semantic Web services. The remainder of this paper is organized as follows: After an brief introduction to the SAP Enterprise Services Architecture we will describe the process used throughout the paper as en example and highlight limitations of current implementations. Next we investigate advantages of BPM-based implementations and show where and how this tools can be improved using semantic Web services. We will close with a summary and an outlook on future work.

SAP Enterprise Service Architecture

Vision SAP is a provider of business software supporting enterprises to perform their more or less standardized business tasks as efficient as possible. The main target is to lower the Total Cost of Ownership (TCO) of their SAP business solutions needed to carry out these business activities. The main challenge with respect to business tasks in today's companies is to keep track with increasingly

* This material is based upon work partially supported by the EU funding under the projects DIP (FP6 - 507483) and ATHENA (FP6 - 507312). This paper reflects the author's views and the Communities are not liable for any use that may be made of the information contained therein.

dynamic markets. This changing business environment demands for more and more flexibility of the companies to adapt to changing market requirements. Therefore, SAP solutions aim at equipping its customers with solutions meeting this demand for flexibly performing business.

The demand for flexibility in markets results in the need for high adaptability of internal company IT structures, simplification of interoperation with business partners as well as strong support for integration and outsourcing of business units, respectively. At the same time, prior investments of SAP customers in their existing IT infrastructure should be leveraged.

Description In order to provide maximum support for the market requirements mentioned above, SAP aims at building an integrative platform for individual service providers and requesters. The facilitating SAP technology is called Business Process Platform. It provides a uniform means for companies to represent the interfaces of their business functionality by *Enterprise Services* which are fully based on the Web Service Description Language (WSDL) standard. However, Enterprise Services extend Web Services by providing business level functionality that is also concerned explicitly with aspects like scalability, security, and transactions etc., which are important in an industrial setting. For instance, the task of canceling an order in a real world scenario has usually subsequent consequences, which must be handled as well. An order can usually not be simply deleted in the database; rather it may involve a complex business process, where following questions might occur: Is roll-back of the order in the backend system possible at all and are manual steps in the roll-back process necessary? Which roles are involved and need notification through another Enterprise Services such as Logistic services, Loan services etc.? Must the approval process be stopped as well and is a cancellation fee applicable? With Enterprise Services this related business information is made explicit.

The facilitator of the Business Process Platform is the design of the SAP products in the so-called Enterprise Services Architecture (see Fig. 1). It defines a clear structuring of the SAP software architecture into separate layers. SAP NetWeaver basically acts as application server and technical integration platform for different business partners. The mySAP Business Suite provides the SAP business software in an Enterprise-Service-enabled way while xApps technology allows for simplified composition of new functionality re-using existing applications or parts of those. The Business Solutions layer provides a clear documentation on the different components of the SAP solutions from a manager's perspective.

2 Example Process

The process we want to investigate in this paper is the standard SAP order-to-cash process in the logistics domain. This process, depicted in Fig. 2, involves three parties: i) A customer, ii) a shipper, and iii) a carrier. As illustrated the customer first places a sales order with the shipper, which enters it in its local

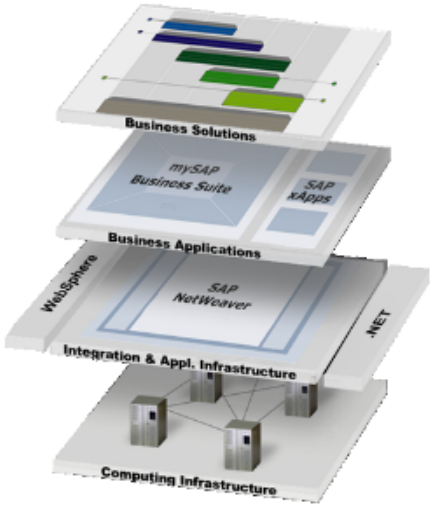

Fig. 1. The SAP Enterprise Service Architecture.

SAP system. After that, the appropriate steps for delivery and picking and packing are taken through. After sending the shipping information to the Express Shipping Interface (XSI), the goods are labeled and the manifest is sent to the carrier. This is the trigger for the actual shipping by the carrier which is tracked by the shipping process step in the SAP system of the shipper. The last activity is the execution of the billing process. Sometimes, carriers change their conditions of service. In this case, they need to notify the shipper of those updates. Also, for a new contractual cooperation between a shipper and a carrier, the new conditions need to be input into the shipper's system.

2.1 Architecture

In the order-to-cash process described in the previous section, different systems are involved. For this use case, we assume that the functionality of the carriers is provided via Web service interfaces. The shipper uses an SAP system. The SAP solution of the shipper is used to integrate the systems of the different parties. However, as connotated in Fig. 2, not all process steps are directly used as they are pre-existent in the SAP system.

Some process steps rather need some configuration adoption in order to fit the needs of the respective SAP customer (here: the shipper) for its specific business case. This is illustrated by the gold coloring in Fig. 2. Also, the process steps of labeling, preparing the manifest, and the shipping itself are highly influenced be the specific requirements of the respective carrier. Therefore, these steps are mainly implemented by the carries. This is illustrated by the blue coloring in Fig. 2.

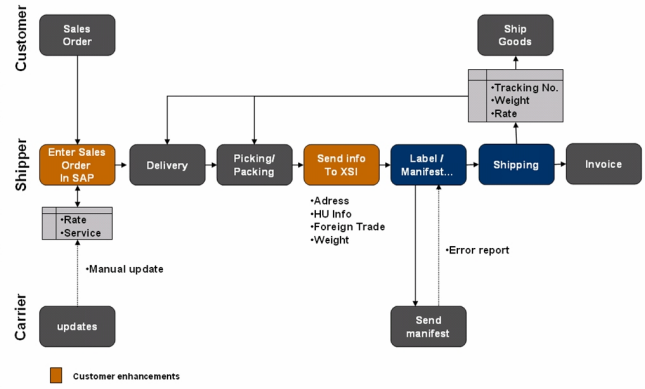

Fig. 2. Current Order-to-Cash Process.

To sum this up, the SAP system is a software that provides some main business functionality that is similar in most companies. Specific adoption to special requirements need to be done by extending the current SAP product via its interfaces in a static manner.

2.2 Pain Points

Due to the more or less static nature of the current solution, the different parties involved face some difficulties. These are explained in the following.

Carriers On the carrier side the following problems occur:

Difficulty to Publish New Services The publication of new services is difficult, because the process with an individual carrier is firmly implemented in the specific SAP system configuration of the shipper. Therefore manual development is necessary if a new service requires to enter this implementation.

Difficulty to Notify Customers of Service Update For this action, the carrier needs to interact with the shipper. If the change of service implies the adoption of the process between both partners, they face the same problem as in the former case.

Difficulty to Connect to SAP Systems In order to connect a third party system to a SAP system in general message mappings are necessary. In addition custom development might be necessary to adapt the business process of the carrier to fit the frame the SAP system sets.

Shippers

Difficulty to Add or Update Services Due to the firm implementation of the processes, this problem also appears to the shipper.

Difficulty to Change Service Provider Since processes are firmly coded into the SAP system, the change from an existing carrier with an already implemented process to a new business partner is difficult. The cost for changing carriers are quite high due to the implementation effort to take.

Different Service Interfaces Provided by Carriers The setup of a process with different carriers requires always new effort, because little knowledge can be reused due to their different interfaces.

Difficulty to Compare Services Provided by Carriers The diversity of interfaces and properties (like pricing model) of the services provided by the carriers makes their comparison difficult.

3 BPM-Based Implementation

In this section we will provide a high-level description of how current BPM tools could be used to implement the scenario described. A BPM-based implementation of the process described above would involve two main components: i) A process modeling tool, and ii) a process execution engine capable of executing the modeled processes. In our case, the process modeling tool is called *Maestro* [1] and the process execution engine is called *Nehemiah* (see Fig. 3).

Focusing on the shipper we will now describe the steps necessary to create an executable process using current BPM tools. Using the Maestro tool, a domain expert would create a graphical representation of the process executed at the shipper whenever a new sales order is processed. Note that this graphical representations is at first not linked to any of the shippers' business systems or any carrier services. Therefore in a second step, the domain expert manually connects the single process steps of the business process to services offered by either the internal or the partner business systems. Connecting different services and systems usually requires a mapping between different message formats. Consequently the domain expert also needs to create the necessary mappings converting between the input and output messages of the different business systems involved.

After the business process has been manually modeled and the involved business systems have been connected to the different process steps, the business process is stored into the process repository. During run-time the BPEL engine retrieves the process from this repository and executes an instance of the process for each incoming sales order.

The main advantage of BPM-based implementations of the scenario described above is the design-time flexibility. After the business process has been modeled using the graphical editor, services implementing different process steps can easily be exchanged during design-time. Integrating a new carrier into the systems

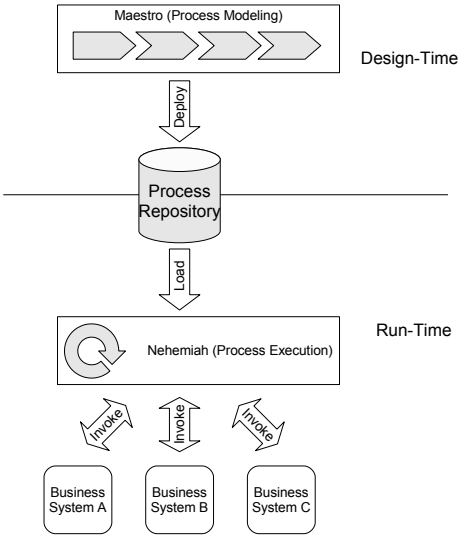

Fig. 3. Current solution using BPM tools.

for example only requires the connections of the carrier services to the appropriate process steps as well as the development of the necessary message transformations. However, BPM-based implementations do not have any additional flexibility during run-time, as the BPEL engine simply executes predefined processes. Therefore, dynamic exchange of carriers during run-time based on the availability of their services is not possible with current BPM-based solutions.

Public vs. Private Processes In the previous paragraph, we have described on a very high abstraction level how the given scenario would be implemented using current BPM tools. However we omitted an important detail during this description. Services available in the business systems of partners usually need to be invoked in a certain order as they are involved in the internal processes of the parter. In our scenario for example it does not make sense to invoke a "track and trace" service of a carrier before a shipping request has be send to that carrier. However, partners interacting during a business process only want to make parts of their internal processes visible to the outside. Therefore the notion of *private* and *public* processes becomes necessary. A private process is the detailed, internal process of a partner. Based on this process, a view—the public process—can be created. This public process hides all the confidential process details and only show the process steps that are required for an interaction. The public process is then used by business partners to create a so-called *Collaborative Business Process* (CPB) involving both internal and external business services (see Fig. 4).

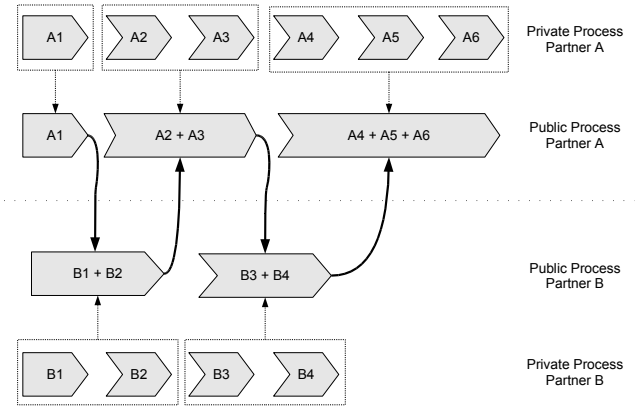

Fig. 4. Relation between public and private processes.

4 Added Value through Semantic Web Services

The previous high-level description of a BPM-based implementation of the order-to-cash process in the logistics domain shows several limitations of current solutions. The most prominent ones are:

- Necessity for manual development of message mappings
- Manual creation of the CPB
- Flexibility limited to design-time.

Using technologies developed in the semantic Web services area, these limitations of current BPM-based implementations can be overcome. In the subsequent sections, we will first describe our overall architecture for integration of semantic Web service technologies into current BPM tools. Following this, we will describe in detail how this architecture enables i) the automatic generation of necessary message mappings, ii) the automatic integration of the public processes of different partners into one Collaborative Business Process (CBP), and iii) the flexible service selection during run-time.

4.1 Definitions

Before we start with the description of our solution we first need to define some elementary terms which we will use during the later discussions.

Definition 1 (Ontology). *An ontology O is a structure*

$$O := (C, R, A, T, \leq_C, \leq_R, \leq_A, \leq_T, \sigma_R, \sigma_A)$$

where:

- C *is a set of concepts aligned in a hierarchy* $\leq_C$
- R *is a set of relations aligned according to* $\leq_R$
- A *is a set of attributes aligned according to* $\leq_A$
- T *is a set of data types aligned according to* $\leq_T$
- $\sigma_R : R \to C \times C$ *is the signature of* R
- $\sigma_A : A \to C \times T$ *is the signature of* A.

In addition to that we define the domain of $r \in R$ *as* $dom(r) := \pi_1(\sigma_R(r))$ *and the range as* $range(r) := \pi_2(\sigma_R(r))$. *This definition of ontologies is based on [2].*

Definition 2 (XML Schema). *An XML schema is a structure*

$$S := (E, A, CT, ST, \delta_e, \alpha_e, \delta_a, \Gamma)$$

where:

- E *is a set of element names*
- A *is a set of attribute names*
- CT *is a set of complex type names*
- ST *is a set of simple type name*
- *a function* $\delta_e : E \to (CT \cup ST)$ *defining the data type of an element* $e \in E$
- *a function* $\delta_a : A \to ST$ *defining the data type of an attribute* $a \in A$
- *a function* $\alpha_e : E \to \mathfrak{P}(A)$ *defining the set of attributes for each element* $e \in E$
- *a regular tree grammar* Γ *specifying the structure of a valid XML schema*

Definition 3 (Mapping). *A mapping map between two structures* S *and* T *is defined as set of mapping elements* me. *Each mapping element relates entities of the source structure* $e_{S,i}, \ldots, e_{S,j}$ *to entities of the target structure* $e_{T,m}, \ldots, e_{T,n}$ *using a mapping expression. The mapping expression specifies how entities of the source structure and the target structure are related.*

$$\begin{aligned} map_{S \to T} &= \{me\} \\ me &= (e_{S,i}, \ldots, e_{S,j}, e_{T,m}, \ldots, e_{T,n}, mapexp) \end{aligned}$$

This definition of mapping has a number of important implications. First a mapping is unidirectional as indicated by the arrow form S to T in our definition and cannot easily be inverted. Second the definition of a mapping given above does not restrict the relation between entities of the source and the target structure to be $1:1$, but rather allows for $m:n$ relations. Third the nature of the involved mapping expressions is not further restricted. This is due to the fact, that it depends strongly on the type of the source and target structure involved.

Definition 4 (Alignment). *An alignment* $A_{S_S \to O_T}$ *from an XML schema* S_S *to an ontology* O_T *is a mapping with:*

$$\begin{aligned} A_{S_S \to O_T} &= \{(e_{S_S,i}, \ldots, e_{S_S,j}, e_{O_T,m}, \ldots, e_{O_T,n}, mapexp)\} \\ e_{S_S,x} &\in E_{S_S} \cup A_{S_S} \\ e_{O_T,y} &\in C_{O_T} \cup R_{O_T} \cup A_{O_T} \end{aligned}$$

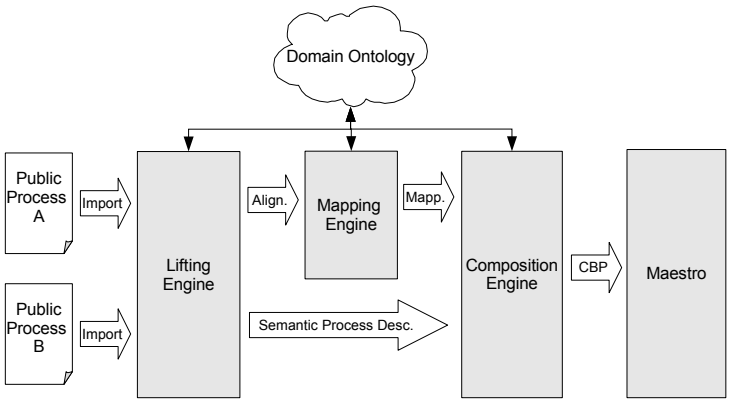

Fig. 5. Design-Time Architecture of the Enhanced Maestro Tool.

4.2 Solution Overview

Our overall architecture consists of two parts: A design-time, and a run-time component. During design-time, we want to simplify the creation of the CBP as much as possible. After loading two public processes, the Maestro tool should generate the CBP automatically (if possible) and present it to the user. Furthermore the tool should generate the message mappings necessary for invoking the involved Web service. Figure 5 shows the design-time architecture of our enhanced Maestro tool. We assume, that the representations of the two public processes not only contain the process flow but also the XSDs of the in- and output messages associated with each process step. After loading the two public processes specifications into our tool, the *lifting engine* generates two things: i) An alignment between the message elements of the XSDs and the domain ontology, and ii) a semantic description of the public processes. In the next step the *mapping engine* uses the alignments between the ontology and the XSDs to generate a list of possible mappings. Now the *composition engine* takes this list of possible mappings and the semantic process descriptions to generate the CPB which is finally presented to the user using the Maestro tool. After a check by the user and possible modifications of the CBP the result is stored into the central process repository.

Details on how the lifting, the mapping generation and the composition are performed will be given in subsequent sections.

During run-time we want to enable the dynamic selection of services based on different criteria. In our scenario we would for example like to select the carrier offering the cheapest price for a given shipment. Therefore we introduce a component called *semantic service selection*. Based on the concrete request, contractual information modeled in the domain ontology and a selection goal, the best process is selected from the process repository, instantiated and executed. Details on the semantic service discovery will be given in section 4.6.

4.3 Lifting Process Descriptions

The first step in our design-time architecture is the lifting of the public processes of the two partners. This lifting consists of two parts, the lifting of the input and output messages associated with the process steps and the lifting of the process descriptions itself.

Lifting Input and Output Messages In order to create an alignment between the domain ontology and the in- and output messages the lifting component executes a set of elementary matching algorithms. We are currently evaluating which elementary matching algorithms perform best in the given setting. An list of possible elementary algorithms can be found in [3] or [4]. These matching algorithms exploit the information available in in the XML schema and the ontology (like, e. g., element and concept names) to create a similarity matrix. This similarity matrix associates each pair of XML schema and ontology entities (e_S, e_O) with $e_S \in E \cup A$ and $e_O \in C \cup R \cup A$ with a similarity value. Based on this similarity matrix an alignment between the XML schema and the domain ontology can be calculated.

Creating Semantic Process Descriptions For the automatic process composition by the composition engine connotated in Fig. 5, the public process description of the shipper as well as the available WSDL descriptions of the carrier services need to be transformed to a format that the composer can work with. This step will be detailed in this section.

Format The composer technology we are going to use bases on the semantic Web services composition approach described in [5] and [6]. This approach will be detailed later on. For now, it is enough to know that for each partner which is to be integrated in the composed process, we need a semantic Web service interface description consisting of the following parts:

- The messages communicated by the semantic Web service given as ontology concepts, and
- Behavioral constraints between the single message exchanges of the semantic Web service.

In other words, the behavioral constraints can be understood as a workflow diagram, like an UML 2.0 Activity Diagram, containing control nodes, like decision, merge, fork and join. The activities in this diagram would be connected to input and output nodes representing the messages communicated. Here, each message is not understood as a technical XML schema description, but an ontology concept for that later on the corresponding XML schema can be nominated.

Shipper The information that is available for the shipper partner is its public process and the liks of the public process steps to specific WSDL operations. From this information, we build the semantic Web service description as follows:

1. Each WSDL operation becomes represented by an input node and an output node connected via a sequential control edge. We refer to this construct as a semantic Web service operation.
2. The data communicated by the input and output node are represented by ontology concepts that are obtained by the alignment step of the Lifting engine from the corresponding WSDL operation's XML schema (cmp. Fig. 5).
3. The semantic Web service operations are connected by semantic Web service behavioral constraints that resenble the workflow of the public process of the shipper.

Carrier For the carrier services, the only information we can get are the WSDL files of their Web servcies. From the lifting component, we again get the alignment of XML schema types communicated to ontology concepts. What is missing however, is a representation of causal interdependencies between the operations that we can use to create the semantic Web service behavioral constraints. Therefore, we build a trivial workflow that is required as input to the composer component. Details follow.

1. Each WSDL operation becomes represented by an input node and an output node connected via a sequential control edge. We refer to this construct as a semantic Web service operation.
2. The data communicated by the input and output node are represented by ontology concepts that are obtained by the alignment step of the Lifting engine from the corresponding WSDL operation's XML schema (cmp. Fig. 5).
3. The semantic Web service behavioral constraints describe a workflow that consists of a fork and a join node. Each branch of this fork-join construct contains exactly one semantic Web service operation.

4.4 Automatic Generation of Message Mappings

The automatic generation of message mappings is performed by the mapping engine. This component takes the alignments created by the lifting engine as input and generates executable mappings between XML schemas. In order to create a mapping between S_1 and S_2, the mapping engine takes the alignments $A_{S_1 \to O}$ and $A_{S_2 \to O}$ as input. For each mapping element in $A_{S_1 \to O}$ the mapping engine searches for a mapping element in $A_{S_2 \to O}$ that relates a schema entity of S_2 to the equivalent ontology entity. If such an entity is found, the mapping expression is used to determine how the schema entities of S_1 and S_2 are related. This in turn creates a new mapping expression that is added to the mapping $map_{S_1 \to S_2}$.

Note that mappings are not generated between each pair of schemas but only between input schemas of one public process and output schemas of the other and vice versa.

4.5 Automatic Integration of Partner Process Steps

As connotated in Fig. 5, the automated process composition by the composition engine is the final step before presenting the suggested CBP to the user via the Maestro tool.

Task For creating the CBP, we use the technology described in [5] and [6]. The main observation behind this technology is that business partners follow their own business processes. These business processes consist of several process steps that on the one hand exchange data amongst themselves, and on the other hand also collaborate with their partners. This however means, that, in order to integrate two business partners with each other, their business processes need to be interconnected. Here, it is important to understand that a business process can not be handled like an atomic transaction. In a process step, the company can have interactions with its partner before proceeding to the next step, which again results in some communication before it enters the next process step, and so forth.

The integration of two business partners in a message-based communication environment, like a service-oriented architecture, can however only work on the basis of atomic transactions. As an additional requirement, this integration must follow the sequential constraints of the message exchanges defined by both parties' business processes. We call these sequential constraints "behavioral constraints" or just the "behavior" of a business partner's systems. Please note, that such information cannot be represented by traditional Web service descriptions using WSDL. For an automatic creation of such an integration (the CBP), a partner's behavior needs to be explicitly specified in a machine-processable manner. This is why the observable part of a business partner's behavior appears in semantic Web service descriptions. The main task of a composer after all is to combine these sets of behavioral constraints to a combined business process of all parties involved.

Realization The inputs for the composer component in general are the semantic Web service descriptions of the participating business partners. In Sect. 4.3, we explained how to create these descriptions for the two parties shipper and carrier. The composition engine basically compares the inputs and outputs that are defined as ontology concepts in the two behavior descriptions and connects them where possible. For the decision whether a connection is possible, the composer relies on the results from the preceding alignment step. The alignment works in a way that it connects those XML schema elements that it later can generate a mapping for to the same ontology element. Therefore, the composer just needs to look for equivalent concepts in the two behavior descriptions that it can connect.

After identifying matching concepts, the composer connects fitting edges by a transformation activity node that defines a conversion which possibly needs to be performed in the real-time execution of the combined process. This conversion

is given by the mapping function $map_{S_1 \to S_2}$. It defines how to transform actual data corresponding to the XML schema of the sender S_1 to a message corresponding to the receiver's format S_2. The result of the composition is a business process that contains the process steps of both parties, their interconnections via mapping activities, and those inputs and outputs that could not be interconnected. The composition therefore is successful, when there are no inputs and outputs left that could not be connected to corresponding communications of the other party.

For the next step of our use case, the composer result needs to be translated into the process representation format of the Maestro tool and will thus be presented to the user. Also in the case of an unsuccessful composer execution, the partly connected bunsiness process will be fed to the Maestro tool as a first suggestion for adaptation by the user.

4.6 Semantic Service Selection

After discussing how semantic Web service technologies can be used to improve the design-time of current business process management solutions, we will now investigate how they can be used to improve their run-time.

As stated in the solution overview, the semantic service selection is responsible for selecting the best fitting carrier for the current shipping request during runtime. The realization of this component is described in detail in [7]. It is based on an approach for semantic Web service discovery introduced by Li and Horrocks [8], and [9]. For applying this approach, an abstract service capability is described based on the domain ontology. The abstract service capability is carrier-independent and covers all possible Web service capabilities within the domain.

Additionally, a successful offline negotiation between a shipper and a carrier is required. The result of this negotiation phase is a contract between that carrier and shipper describing the provided service capabilities by that carrier. Each contract is modeled as a sub-concept of the service capability based on the domain ontology. These semantically described contracts are stored as OWL documents on a separate repository. The concrete shipping request created at run-time is then described either as an instance or as a most specific sub-concept of the abstract Web service capability according to the domain ontology. A shipping request can be fulfilled by a carrier Web service if the the concrete request subsumes a Web service capability. If more than one contract matches the concrete request an additional selection step is required in order to choose between the available carriers. This step usually requires run-time invocation of the carrier Web services in order to get information necessary for the selection according to the goals of the requester. A *selection goal* specifies which criterion defines the best suiting carrier, e. g., best price or shortest delivery. Since these two parameters are subject of frequent change due to the competition on the carrier market, we decided not to design these parameters in the semantically described contracts.

The important parts of the shipment request are: Ship from and ship to addresses, items to be shipped, and the selection goal. The received shipment request by the component is first semantically lifted according to the ontology in order to be processed by a DL-reasoner. In the next step, the repository containing all existing pre-negotiated contracts with carriers is queried. Additionally, the necessary URL to the WSDL of the Carrier Web service associated with the contract can also be obtained from the repository. Each available contract is consequently subsumed by a DL-reasoner to determine the contracts matching the (semantically lifted) shipment request. If the client application has specified, e. g., the best price as its selection goal in the shipment request, all matching carrier Web services are contacted and the carrier with the best price for this shipment is selected.

After the selection is done, the process associated with the selected carrier is loaded from the process repository, instantiated and executed.

5 Summary and Outlook

The proposed carrier/shipper-scenario trys to keep the described system simple in order to be able to concentrate on the important steps first. The important aspect is mainly to examine how semantic technologies can beneficially be applied to real-world business scenarios. We identified the automation of the so far manual business process integration as the main area of contribution for semantic Web technology. The solution extends and therefore bases on standard BPM modeling tools.

Furthermore, we abstain from the requirement of business partners to adhere to exactly the same software component interfaces. Thus, mediation comes into play and its interacting with the semantic Web service composition is a second aspect to focus on using this scenario.

After the successful implementation of this first scenario, the described setting can be extended to a more comprehensive application in a later version. In the current proposal, the composed business process is being created during design-time. When a carrier changes its conditions, the process of composition needs to be executed again in order to compute the potential adoptions to the process instance. In a more dynamic implementation, this step could be executed each time a customer requests a shipment. This way, the system would immediately and automatically incorporate changes to the carrier capabilities.

References

1. Greiner, U., Lippe, S., Kahl, T., Ziemann, J., Jkel, F.W.: Designing and implementing crossorganizational business processes - description and application of a modeling framework. In: Interoperability for Enterprise Software and Applications Conference I-ESA. (2006)
2. Stumme, G., Ehrig, M., Handschuh, S., Hotho, A., Maedche, A., Motik, B., Oberle, D., Schmitz, C., Staab, S., Stojanovic, L., Stojanovic, N., Studer, R., Sure, Y., Volz,

R., Zacharias, V.: The karlsruhe view on ontologies. Technical report, University of Karlsruhe, Institute AIFB (2004)
3. Do, H.H., Rahm, E.: COMA - a system for flexible combination of schema matching approaches. In: Proc. 28th VLDB Conference. (2002)
4. Shvaiko, P., Euzenat, J.: A survey of schema-based matching approaches. Technical report, Informatica e Telecomunicazioni, University of Trento (2005)
5. Albert, P., Henocque, L., Kleiner, M.: Configuration-based workflow composition. In: ICWS, IEEE Computer Society (2005) 285–292
6. Albert, P., Henocque, L., Kleiner, M.: A constrained object model for configuration based workflow composition. In Bussler, C., Haller, A., eds.: Business Process Management Workshops. Volume 3812. (2005) 102–115
7. Friesen, A., Namiri, K.: Towards semantic service selection for B2B integration. In: submitted to Methods, Architectures & Technologies for e-Service Engineering (MATeS) at ICWE. (2006)
8. Li, L., Horrocks, I.: A software framework for matchmaking based on semantic web technology. In: Proc. of the Twelfth World Wide Web Conference. (2003)
9. Preist, C., Cuadrado, J.E., Battle, S., Grimm, S., Williams, S.K.: Automated business-to-business integration of a logistics supply chain using semantic Web services technology. In: International Semantic Web Conference. (2005) 987–1001