Semantic Business Automation*

Jens Lemcke and Christian Drumm

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. Based on a standard SAP process in the area of logistics, we show how current approaches support business flexibility via manual modeling tools. Our application of semantic Web service technologies on top of today's business process management tools enables the automation of major tasks of business process management.

1 Introduction

The main purpose and central challenge of *business process management* (BPM) in today's companies is to help keeping track with increasingly dynamic markets. This changing business environment demands for more and more flexibility of the companies to adapt their own business to changing market requirements and to improve interoperability with potential business partners.

In this paper, we show how *semantic Web service* (SWS) technology can improve standard business process management software. We do this on the basis of the real-life order-to-cash business process in the logistics domain between the two business partners shipper and carrier. Upon a purchase request of a customer, the shipper procures the requested good from its depot, packs and labels it, hands it over to the carrier and possibly provides after-sales services such as package tracking to its customer.

Most of these process steps require heavy interactions between the shipper and the carrier. In order to compete on a quickly changing logistics market, it is essential to a shipper that it can easily switch between different carriers which steadily adopt their conditions and service offers over time. In Sect. 2, we sketch how current business process management software on top of a *service-oriented architecture* (SOA) facilitates this business flexibility. This is mainly achieved by providing manual modeling tools. Section 3 then details how modern semantic Web service technologies can be applied on top of existing business process management software. We show how major parts of business process management tasks can be automated using this technology.

* 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.

2 BPM-Based Implementation

In this section, we will provide a high-level description of how current BPM tools are used to implement the order-to-cash scenario involving a shipper and a carrier as lined out in Sect. 1. In general, a BPM-based implementation of a business process involves 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 *Nehemiah* (see Fig. 1).

Using the Maestro tool, a domain expert would create a graphical representation of the process executed at the shipper side. Note that this graphical representations is not yet linked to any of the shipper's business systems or any services of a carrier. 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's 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 modeled manually 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 process execution engine retrieves a process from this repository and executes an instance of it upon an incoming request.

The main advantage of BPM-based implementations 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. For example, integrating a new business partner into the collaboration only requires to adopt the connections of the partner services to the appropriate process steps as well as the development of the necessary message transformations. However, BPM-based implementations do not provide any additional flexibility during run-time, as the process execution 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.

3 Added Value through Semantic Web Services

The previous high-level description of a BPM-based implementation of a business process shows several limitations of current solutions. The most prominent ones are: i) necessity for manual development of message mappings, ii) manual creation of the *collaborative business process* (CBP, Sect. 3.3), and iii) 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 (Sect. 3.1). Following this, we will describe in detail how this architecture enables i) the automatic generation of necessary message mappings (Sect. 3.2), ii) the automatic integration of the public processes of different partners into one CBP, and iii) the flexible service selection during run-time (Sect. 3.4).

3.1 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 therefore should generate the CBP automatically (if possible) and present this as a proposal to the user. Furthermore, the tool should generate the message mappings necessary for invoking the involved Web services. Figure 2 shows the design-time architecture of our enhanced Maestro tool. We assume that the representations of the two business processes not only contain the process flow but also the *XML schemas* (XSD) of the input 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 these 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 via the Maestro tool. After the user reviewed and applyed possible modifications on the CBP, the result is stored into the central process repository.

During run-time we want to enable the dynamic selection of services based on different criteria. In our scenario, we would, e. g., like to be able to select the carrier offering the cheapest price for a given shipment request. 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 being selected from the process repository, instantiated and executed.

3.2 Automatic Generation of Message Mappings

In order to create an alignment between the domain ontology and the input and output messages as depicted in Fig. 2, the lifting component executes a set of elementary matching algorithms. These matching algorithms exploit the information available in one XML schema and the ontology (like, e. g., element and concept names) to create a similarity matrix. This similarity matrix associates each pair of the XML schema and ontology entities (e_S, e_O) with a similarity value. Based on this similarity matrix, an alignment including a mapping expression between the XML schema and the domain ontology can be calculated $(A_{S \rightarrow O})$.

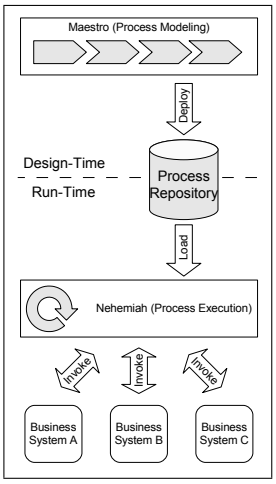

Fig. 1. Current Solution Using BPM Tools.

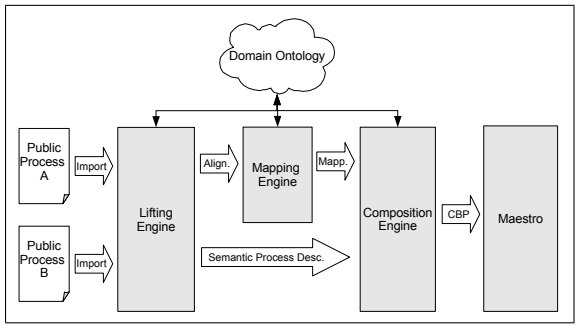

Fig. 2. Design-Time Architecture of the Enhanced Maestro Tool.

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 an 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 $map_{S_1 \to S_2}$. Mappings are not generated between each pair of schemas but only between input schemas of one process and output schemas of the other, and vice versa.

3.3 Automatic Integration of Partner Process Steps

For the automatic process composition by the composition engine connotated in Fig. 2, 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.

The composer technology we are going to use bases on the semantic Web services composition approach described in [2, 3]. For each partner which is to be integrated in the composed process, we therefore need a semantic Web service interface description consisting of the following parts: i) the messages communicated by the semantic Web service given as ontology concepts, and ii) 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 which the corresponding XML schema can be nominated later on.

The main task of a composer is to combine the sets of behavioral constraints to a CBP of all parties involved. The composition engine thus basically compares the inputs and outputs that are defined as ontology concepts in the two behavior descriptions and connects them where possible. In each such connection, a transformation activity node is incorporated 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 $map_{S_1 \rightarrow S_2}$.

The result of the composition is a business process—the CBP—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.

3.4 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 [4]. It is based on an approach for semantic Web service discovery introduced in [5, 6]. 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 a service capability concept of the domain ontology. These semantically described contracts are stored as OWL documents in 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 the criterion to nominate the best suiting carrier, e. g., best price or shortest delivery. Since these two parameters are subject to fre-

quent change due to the competition on the carrier market, we decided not to design these parameters in the semantically described contracts.

After the selection is done, the process associated with the selected carrier is loaded from the process repository, instantiated and executed in the Nehemiah component (see Fig. 2).

4 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. Albert, P., Henocque, L., Kleiner, M.: Configuration-based workflow composition. In: ICWS, IEEE Computer Society (2005) 285–292
3. 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
4. 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)
5. Li, L., Horrocks, I.: A software framework for matchmaking based on semantic web technology. In: Proc. of the Twelfth World Wide Web Conference. (2003)
6. 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