SEMANTIC PROFILE BASED SERVICE DISCOVERY FOR ABSTRACT PROCESS COMPOSITION

Shuying Wang, Miriam A. M. Capretz and Hany El Yamany
Department of Electrical and Computer Engineering, Faculty of Engineering, University of Western Ontario
London, Ontario, N6A 5B9, Canada

Keywords: Business Process Composition, Semantic Web services, Service Oriented Architecture.

Abstract: WSBPEL (Web Services for Business Process Execution Language) is a process modelling language for composing Web Services. Abstract processes can be used in WSBPEL as process templates, which describe abstract activities without specifying the execution details and the service bindings. However, it is a challenge to refine an abstract process into a concrete executable process for the purpose of adapting to different business requirements. In order to discover the potential business partners and Web Services, we propose a profile-based service matching and discovery approach for semi-automatic semantic process representation that occurs during design time. Specifically, our approach utilizes semantic profiles to specify the semantic descriptions of process activities. Consequently, our methodology provides substantial flexibility for abstract process composition while reusing existing processes and services.

1 INTRODUCTION

The service-oriented computing paradigm is transforming traditional workflow management from a close, centralized control system into a dynamic information exchange and business process. In fact, the complexity involved in the establishment of business processes necessitates the reuse of similar processes within organizations. Process reuse minimizes errors and reduces costs, since a process can be constructed from other processes already designed and used by specialists. Therefore, when a process is well-established, it can be shared by service designers or developers for the creation of a similar process.

In order to achieve the reuse of processes, generic templates are often used to create different processes for specific cases. For example, WSBPEL (Web Services for Business Process Execution Language) (WSBPEL, 2007) uses executable processes and abstract processes to ensure that different business processes can understand one another in a Web Services environment. The abstract processes are used to describe the order in which business partners may invoke operations without specifying the execution details and service bindings. Moreover, abstract processes allows for the modelling of common parts and disparate elements of processes in order to enable their configuration in specific cases (Tao and Yang, 2007). Therefore, the abstract process allows designers to create a process template and enables developers to refine the execution details at a later stage.

Incorporating the Semantic Web Services (McIlraith, et al., 2001) with BPM (Business Process Management) (BPM, 2008) provides a semantically enriched environment and a set of tools to help processes respond to the business changes. Some researchers have tried to combine semantic Web Services with WSBPEL in order to reap the benefits of both standards. Liu et al. (2004) present a one-way mapping from the OWL-S (2004) process model to WSBPEL for creating composite and atomic processes. Furthermore, Hepp et al. (2005) present a semantic business process management (SBPM) framework and utilizes semantic Web Services with the existing BPEL processes. More specifically, Sivashanmugam et al., (2004) partially define and then annotate a process with semantic information that may be analyzed at runtime to find missing data and derive a completely executable BPEL process.

Although the combination of BPM and the Semantic Web Services provides significant potential for business flexibility, there is still uncertainty in how to synthesize them. In particular,

Wang S., A. M. Capretz M. and El Yamany H. (2009).
SEMANTIC PROFILE BASED SERVICE DISCOVERY FOR ABSTRACT PROCESS COMPOSITION.
In *Proceedings of the International Conference on e-Business*, pages 258-263
DOI: 10.5220/0002236002580263

there are three major challenges in combining these features:

1. How to formally represent descriptions of potential service providers.
2. How to use such descriptions to discover appropriate service providers.
3. How to integrate discovered services into the WSBPEL engine.

In this paper, we propose a semantic profile-based abstract process composition. Specifically, we use abstract processes, which are independent of the service description and process definition, to instantiate an executable process during process design. Also, in order to dynamically discover the business partners and Web Services before the execution, process requirements are represented as semantic profiles. Specifically, any service that satisfies the semantic requirements of an activity can be potentially used to perform that specific activity in the process. Therefore, the abstract processes can be refined into concrete executable processes based on the activity and the associated Web Services requirements. Finally, our approach works during design time, where the goals are to reuse the existing process and reduce the process redesign cost.

The paper is organized as follows: Section 2 presents a case study in the automotive retail domain. Next, in Section 3, we illustrate our system architecture, which includes mapping from WSBPEL to the OWL-S profile, the semantic representation for adding process profiles to the WSBPEL extension as well as profile matching and process refinement. Finally, Section 4 provides a conclusion and describes future work.

2 A BUSINESS CASE SCENARIO

This section describes a business case scenario in the automotive retail industry. In this industry, there is an increasing need for information exchange and unified service sharing between business partners. STAR (Standards for Technology in Automotive Retail) (STAR, 2009) is a non-profit, unionized organization whose members include dealers, manufacturers, retail system providers and automotive-related industrial organizations. The STAR domain ontology, which is based on STAR metadata, is formalized in OWL DL and serves as a source of background knowledge in the automotive retail business domain.

A DMS (Dealer Management System), created specifically for vehicle dealers, includes the finance, sales, parts, inventory and maintenance components of sales. Moreover, a DMS provides multiple Web Services interfaces so that it can be integrated with potential business partners. These Web Services are composed in a unified manner to provide a configurable business process for the quoting, ordering, delivery, and payment of vehicles and parts.

For example, the parts order process describes the transactional data exchanged between the DMS and other service providers. When a dealer receives a parts order from a customer, he/she searches the service repository for both parts and delivery providers, locates the services and initiates an order process with the Web Services of the factory and delivery company. However, the customer's parts order may be a specific requirement for which the dealer has to find a factory with the capability of producing it. Consequently, the factory's service interface may not match with the dealer's service, necessitating the DMS to generate an alternative service in order to interoperate with the parts providers' services.

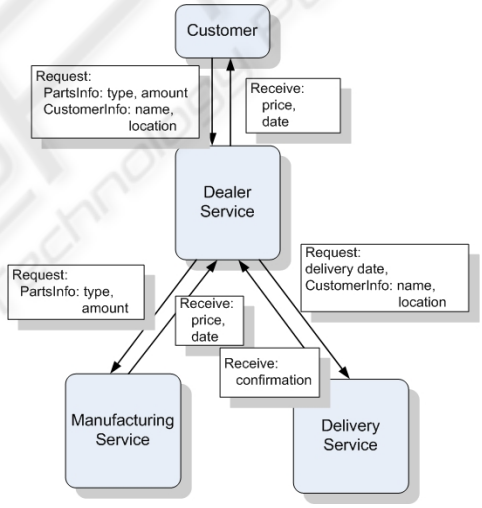

Figure 1: An Example of Parts Order Process.

Figure 1 represents a simple example of an abstract parts order request process. A car parts order process is separated into a customer's order request, the order and the delivery activities. Accordingly, the process can be accomplished by three corresponding Web Services: the Dealer Service, the Manufacturing Service, and the Delivery Service. However, without knowing the manufacture and delivery services in advance, it is impossible to define an executable WSBPEL process, since the WSBPEL process has to define the service participants specified by the port types in the process definition.

3 ABSTRACT PROCESS-BASED SERVICE COMPOSITION

As shown in Figure 2, we are aiming to provide a profile-based service discovery for abstract process composition. During business process modelling, the semantic process profiles, which are compatible with the OWL-S profile model (OWL-S, 2004), are defined for the WSBPEL process activities. These profiles are used to provide semantic descriptions for activities that do not entail service binding. Subsequently, the profiles are sent to the matchmaker to find potential service matches through the domain ontology and the service repository. The matchmaker will then return the candidate service list for further process refinement.

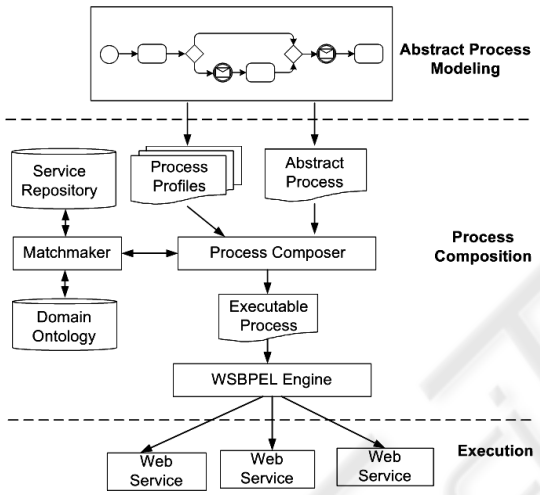

Figure 2: Architecture of the Semantic Profile-Based Abstract Process Composition.

The approach includes the following steps:

- First, the process designer models abstract processes for defining the generic activities that are required by all users regardless of their various contexts.
- Second, the semantic description of a business process is defined in the process profiles. These profiles are based on the mapping from WSBPEL to the OWL-S profile model as well as the user-specified domain ontology.
- Next, the process composer will search for the related service providers in the service repository. In fact, our profile-based service matching algorithm uses the domain ontology to expand the profile and thus increase the mapping precision and the number of candidate services.
- Finally, the executable WSBPEL process and Web Services are generated through process refinement. Specifically, a WSBPEL engine provides a run-time execution environment for a process instance.

Currently, we are in the process of developing a set of components for this procedure. One key component is the process composer, which plays a central role in process composition. When a process requirement originates from the service requester, the composer retrieves the abstract process template and its related process profiles from the repository. Subsequently, the process profiles are sent to the matchmaker for retrieving the relevant services. As a result, the outputs of the process composer entail a concrete process definition that can be executed by the WSBPEL process engine.

3.1 OWL-S based Profile for Abstract WSBPEL Process

A semantic process profile provides a semantic description of the process activities associated with a Web Service. Specifically, it introduces the IOPE descriptions, which are equivalent to the details that the OWL-S profile supplies. Moreover, the semantic process profile can use a domain specific ontology as a parameter for describing the concise activity requirements. In fact, since WSBPEL allows a process to compose one or more services, an abstract process may contain one or more profiles; the activities corresponding to one web service are bundled as one profile. Consequently, the process profile specifies the activity requirements which are used for discovering the potential services in order to refine an abstract process into a concrete executable process.

As shown in Table 1, we map the corresponding WSBPEL variables to the OWL-S profile. The process profile mapping contains the following matches:

- The Input/Output variables of Invoke are related to the Inputs/Outputs of the OWL-S profile.
- The other attributes of Invoke, such as Operation and partnerLink, can be mapped to the Inputs if there are only input variables or they can be mapped to the Outputs if there are only output variables. Lastly, they can be mapped to both the Inputs and the Outputs if there are both input and output variables or if neither variable exists.
- The attributes of Receive, such as Operation and its related variables, are mapped to the Inputs of the OWL-S profile.

- The attributes of Reply, such as Operation and its related variables, are mapped to the Outputs of the OWL-S profile.
- The <onMessage> attributes of Pick are similar to a <receive> activity, since both attributes wait for the receipt of an inbound message.
- The Precondition, Effects, and QoS constraints can be defined by the user, but they cannot be automatically transferred from the WSBPEL process definition to the profile.
- In the process profile, the process name will be defined as the service name.

Table 1: A Reference Mapping of WSBPEL Activities to an OWL-S Profile.

WSBPEL	OWL-S profile
Input variables of <Invoke>	Inputs
Output variables of <Invoke>	Outputs
Other attributes of <Invoke>	Inputs or Outputs or Both
Attributes of <Receive>	Inputs
Attributes of <Reply>	Outputs
<onMessage> attributes of Pick	Inputs
/	Precondition/Effect

In this mapping, there are four activities in the WSBPEL process that provide the inputs and outputs of a process: Invoke, Receive, Reply, and onMessage of Pick. For each activity, the user can define one or more attributes in the corresponding process profile. Moreover, since control structures, such as sequence, flow, fork, and repeat, as well as internal activities, such as the fault handler, have no direct relations with IOPE, we do not provide a mapping for them. The generation of a process profile is a semi-automatic procedure, since the attributes of mapped activities can be automatically retrieved from the defined WSBPEL abstract processes. However, the user is responsible for manually defining the Precondition, Effects and domain-specific ontology, since there are no direct definitions in WSBPEL.

3.2 An Example

A WSBPEL abstract process, shown in Figure 3.a, models the parts order process described in the scenario. After receiving a customer's request, the dealer places the order with the manufacturing service, using PartsOrder_Request, and receives back the details through PartsOrder_Receive. Since the dealer may not have the service information for the manufacturing, the parnterLink, inputVariable and Variable are all marked as ##opaque, which identifies unknown variables in WSBPEL. Similarly, opaqueActivity is another keyword for unknown control activity structures. Additionally, the same process of activities occurs for delivery services.

```
1.  <sequence>
2.    <opaqueActivity name="parts order"/>
3.    <opaqueActivity name="parts delivery"/>
4.    <invoke operation="PartsOrder_Request"
5.         partnerLink="##opaque"
6.         inputVariable="##opaque"
7.         p:profileURI="partsorder_profile"/>
8.    <receive operation="PartsOrder_Receive"
9.       partnerLink="##opaque"
10.      Variable="##opaque"
11.      p:profileURI="partsorder_profile"/>
12.   <invoke operation="PartsDelivery_Request"
13.      partnerLink="##opaque"
14.      inputVariable="##opaque"
15.      p:profileURI="partsdelivery_profile"/>
16.    <receive operation ="PartsDelivery_Receive"
17.      partnerLink=" ##opaque"
18.      Variable="##opaque"
19.      p:profileURI="partsdelivery_profile "/>
20. </sequence>
```

Figure 3.a: An Example of Abstract Process.

```
1.  <profile:serviceName>
2.     PartsOrder</profile:serviceName>
3.  <profile:hasInput>
4.     <process:Input rdf:ID="#PartsOrder_Request">
5.        <process:parameterType>
6.           STAR.owl#PartsOrder
7.        </process:parameterType>
8.     </process:Input>
9.  </profile:hasInput>
10. <profile:hasOutput>
11.    <process:Output rdf:ID="#PartsOrder_Receive">
12.       <process:parameterType>
13.          STAR.owl#PartsOrder
14.       </process:parameterType>
15.    </process:Output>
16. </profile:hasOutput>
```

Figure 3.b: An Example of a Process Profile.

In our approach, we add the semantic process profile to the WSBPEL elements for representing the semantics of a process or an activity. The fragment of the WSBPEL process, in Figure 3.a, specifies the profile as p:profileURI. This is used to define the extension to support the use of process profile. Further, the activities of the same service use the same profile. Therefore, the PartsOrder_Request and PartsOrder_Receive components use the same

profile, partsorder_profile; whereas PartsDelivery_Request and PartsDelivery_Receive use partsdelivery_profile.

As illustrated in Figure 3.b, a process profile is constructed for the PartsOrder_Request and PartsOrder_Receive activities. The profile, which service name is PartsOrder, uses the invoke operation name of PartsOrder_Request as the input and PartsOrder_Receive as the output. Moreover, the STAR.owl#PartsOrder, which specifies the concept of the STAR ontology, is referred to as parameterType.

3.3 Profile based Service Matching

The profile based matching that we are proposing is a semi-automatic approach similar to the OWL-S/UDDI matchmaking algorithm presented in Paolucci et al. (2002). OWL-S/UDDI matchmaking requires service providers to create and register their service profiles for service searching. However, this is very rare in actual applications. Most of services are still registered directly using their service descriptions instead of OWL-S profiles. Therefore, in order to support generic service matching, we propose a profile-based service matching algorithm. The algorithm firstly expends the profile concepts using domain ontology and then searches services with updated profiles. The algorithm can be summarized in four steps:

Step 1: *Profile Extraction.* This step includes extracting the terms defined in the profile. As shown in Figure 4, these terms are categorized as four sets: inputV, outputV, preconditionV, and effectV. These sets correspond to the instances of IOPE, including the input, the output, the precondition and the effect of the profile. Each IOPE instance is extracted as a set, which includes <id, parameterType$_1$, parameter Type$_2$>, where id is the instance name, parameterType$_1$ represents the ontology name and parameter Type$_2$ is the associated ontology concept. For example, according to our defined profile in Figure 3.b, the profile is transformed into an input and output set; the input set includes <"PartsOrder_Request", "STAR.owl", "PartsOrder">, and the output set contains <"PartsOrder_Receive", "STAR.owl", "PartsOrder">.

Step 2: *Profile Expansion.* The second step is to expand the profile by searching relevant concepts in the domain ontology. For example, for the concepts "PartsOrder_Request" and "PartsOrder" in the input set, the search function supported by STAR ontology match engine is used to find the equivalent concepts in "STAR.owl". Subsequently, new input sets are constructed according to the discovered concepts, and these new concepts will be the id of new sets with parameterTypes from the original set.

Step 3: *Profile Reforming.* This step involves reforming the profile using the updated vector sets. A rewrite function writes the four vector sets as corresponding IOPE parts. For example, each vector in the input set will be added as an input instance of hasInput. Consequently, the new profile can be used for service matching.

```
1. Input:SP //semantic profiles
2. Output:SL //candidate service list
3. Begin:
4.   SL = ∅;
5.   if ( SP ≠ ∅ ){
6.    for(∃p ∈ SP){
7.    /*Step1:Profile Extracting*/
8.    inputV=p.inputs;
9.    outputV=p.outputs;
10.   preconditionV=p.preconditions;
11.   effectV=p.effects;
12.   /*Step2:Profile Expansion*/
13.   inputV=inputV∪search(inputV);
14.   outputV=outputV∪search(outputV);
15.   preconditionV=preconditionV∪
16.          search(preconditionV);
17.   effectV=effectV∪search(effectV);
18.   /*Step3:Profile Reforming*/
19.   np=rewrite(inputV,outputV,
20.        preconditionV, effectV);
21.   /*Step4:Profile Matching*/
22.   ServiceSet = match(np);
23.   for(∃s ∈ ServiceSet)
24.    if ( s ≠ ∅ ){
25.    /*Filtering out incompatible
26.     results*/
27.    if(s==“exact“||s==“plugIn“)
28.     SL = SL ∪ s;}
29.   }
30.  }
31. }
32. End.
```

Figure 4: Profile Based Service Matching Algorithm.

Step 4: *Profile Matching.* During profile matching, the expanded profiles are matched with the registered Web Services. The result set is then ranked based on the semantic similarity between the input concepts of the process profile and the returned Web Services. Consequently, the algorithm obtains a result expressing the degree of semantic similarity. We set the similarity value to four possible results:

- `exact`, which means that the profile concept and discovered service concept being compared are identical,
- *`plugIn`*, which indicates that the profile concept subsumes the registered service concept,
- `subsumes`, which occurs when the registered service concept subsumes the profile concept

contained in the request, and

- `fail`, which happens when none of the other relationships occur.

As shown in Figure 4, we consider the results of `exact` and `plug-in` as semantically compatible with the matching request. Consequently, the results of `fail` and `subsume` will be filtered out. The final result will be one selected service from a list of compatible Web Services.

3.4 Process Refinement

The objective of process refinement is to refine the abstract processes and generate the executable process definition according to the discovered services. This is a semi-automatic process since some human decisions are needed to select the proper services and update the processes, such as partnerlink, activities, message correlations, and exception handling. For example, the abstract activity of parts order, presented in Figure 3.a, will be refined as follows.

```
1.  <partners>
2.    <partner name="manufacturer"
3.       serviceLinkType="lns:PartsOrderLinkType"
4.       partnerRole="manufacturer"/>…
5.  </partners>
6.  <sequence>
7.    <while><assign name="parts order">…</assign>
8.    </while>
9.    <flow>
10.     <invoke operation="PartsOrder_Request"
11.        partnerLink="PartsOrder"
12.        inputVariable="PartsOrder_Request" />
13.     <receive operation="PartsOrder_Receive"
14.        partnerLink="PartsOrder"
15.        Variable=" PartsOrder_Receive "/>
16.   </flow>…
17. </sequence>
```

Figure 5: A Fragment of the Refined Parts Order Process.

As shown in Figure 5, at first, partner name, serviceLinkType, and partnerRole are updated. Then, the concealed activities that organize the message sent from or to the manufacturer are replaced by the assigned activities. Next, the invoke activities sending the requests to the manufacturer are embedded into a flow activity. Finally, the keyword opaque is replaced by the concrete variable.

4 CONCLUSIONS

In this paper, we presented a semantic profile-based approach for abstract process composition. By developing semantic profiles for business processes, we propose a semi-automatic method for interpreting these processes with substantial flexibility and exploiting the reusability of existing processes and services with a minimum amount of redesign and redevelopment. In our work, the activities defined in an abstract WSBPEL process use semantic profiles, which contain IOPE descriptions of OWL-S compatible profile model for performing the discovery of services. We are currently developing the process composition tool to enable the semantic profile definition, profile based service discovery, and abstract process refinement. Future work will also focus on extending the utilization of the proposed approach by incorporating business rules.

REFERENCES

BPM, 2008. Business Process Management Initiative, http:// www.bpmi.org.

Liu, S., Khalaf, R., Curbera, F., 2004. From DAML-S Process to BPEL4WS, In *Proc. of the 14th International Workshop on Research Issues on Data Engineering: Web Services for E-Commerce and E-Government Applications (RIDE'04)*.

McIlraith, S.A., Son, T.C., Zeng, H., 2001. Semantic web services, *IEEE Intelligent Systems*, 16, 46–53.

Hepp, M., Leymann, F., Domingue, J., Wahler, A., Fensel, D., 2005. Semantic Business Process Management: A Vision Towards Using Semantic Web services for Business Process Management, *IEEE International Conference on e-Business Engineering*, pp. 535 – 540.

OWL-S, 2004. Web Ontology Language for Web services http://www.w3.org/Submission /OWL-S/.

Paolucci, M., Kawamura, T., Payne, T.R., Sycara, K., 2002. Semantic Matching of Web Services Capabilities, *ISWC 2002, LNCS 2342*, pp. 333-347.

Sivashanmugam, K., Miller, J., Sheth, A., Verma, K., 2004-5. Framework for Semantic Web Process Composition, *International Journal of Electronic Commerce*, Vol. 9(2) pp. 71-106.

STAR, 2009. Standards for Technology in Automotive Retail, http://www.starstandard.org/.

Tao, T. A., Yang, J., 2007. Supporting Differentiated Services With Configurable Business Processes. In *Proc. of IEEE International Conference on Web Services (ICWS 2007)*.

WSBPEL, 2007. Web Services Business Process Execution Language Version 2.0, http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html.