Mappings from BPEL to PMR for Business Process Registration

Jingwei Cheng, Chong Wang, Keqing He, Jinxu Jia, Peng Liang

Mappings from BPEL to PMR for Business Process Registration*

Jingwei Cheng[1], Chong Wang[1+], Keqing He[1], Jinxu Jia[2], Peng Liang[1]

[1] State Key Lab. of Software Engineering, Wuhan University, China
cinfiniter@gmail.com, {cwang, hekeqing}@whu.edu.cn, pliangeng@gmail.com
[2] International School of Software, Wuhan University, China
jiajinxu89@gmail.com

Abstract. In order to facilitate business collaboration and interoperation in virtual enterprises, it is crucial to discover appropriate business processes modeled in different languages and stored in different repositories. For this purpose, it is more efficient to register existent process models into a common process model registry, rather than defining numerous mappings from one modeling language to another. Considering the wide acceptance of BPEL, this paper proposed a common metamodel for process model registration (PMR), and defines the mappings from BPEL to PMR with corresponding mapping rules and algorithms. In this way, BPEL process models can be registered in the process model registry based on PMR automatically, and then the essential data from their registration information can facilitate process discovery across heterogeneous process repositories.

Keywords: BPEL, business process, process model registration, metamodel for process model registration

1 Introduction

With the rapid progress of economic globalization, business collaboration is becoming more and more popular today, demanding solutions for the exploding amount of interoperability problems [1]. Business processes are fundamental in business collaboration for virtual enterprises. To promote the interoperation among business processes from different partners, it's necessary to facilitate business knowledge sharing and business process reuse within/across enterprises.

Existent business processes are modeled by various business process modeling languages, and stored in different repositories. This situation puts obstacles in the way of cross-enterprise discovery and reuse of business processes in virtual enterprises. Therefore, it's needed to provide a uniform manner to register the selected metadata and common semantics of heterogeneous business processes, promote the semantic

* This research project was supported by the Fundamental Research Funds for the Central Universities under Grant No. 3101032.

+ Corresponding author

interoperation between them, and effectively support business process reuse and collaboration. For this purpose, this paper proposes the metamodel for process model registration (PMR) as the common registration facility for business process models in different languages. Furthermore, in order to support the implementation of a PMR-based registration tool, it's necessary to provide mapping rules from process modeling languages to the PMR metamodel when performing automatic registration of these models. Considering the fact that BPEL[2] has been accepted as one of the most widely used business process modeling languages due to its best practice for process implementation in SOA[3], this paper focuses on how to register BPEL models into the common process registry by defining the corresponding mappings rules.

The rest of this paper is organized as follows: section 2 introduces the work on the transformation of BPEL process models and relevant mechanisms for process model registration; section 3 presents the main structure of PMR; section 4 defines the mapping rules from BPEL to PMR and the corresponding mapping algorithms in detail, and shows how to register BPEL models with the mapping rules. Finally, section 5 concludes our work with future work directions.

2 Related Work

Considerable work on mapping BPEL to the other kinds of process modeling languages has been done in the past few years. For example, [4] proposed mapping strategies to transform BPEL to OWL-S, with which the process modeling capabilities and semantic capabilities of OWL-S can be combined, and the process models can be discovered and interacted in a computer understandable way. [5] defined the mappings between BPEL and XPDL (XML Process Definition Language) to support model transformation from one to the other. Meanwhile, mappings from BPEL constructs to graphical BPMN (Business Process Modeling Notation) elements are provided in [6-7] to visualize BPEL process models using BPMN diagrams. It's obvious that those mappings are mainly used to transform BPEL models into the process models in other languages, and can support further interoperation between the corresponding process model repositories. Although one partner of the interoperation can be any repository adapting the modeling languages mentioned above, the other one is limited to BPEL model repository. In this case, it's difficult to achieve interoperation across heterogeneous process repositories.

To relieve this problem, it is common to introduce metamodels or metamodeling techniques to harmonize the differences among various process models. [8] proposed a model transformation framework based on metamodel for model conversion among various kinds of process definition languages, such as BPEL and XPDL. However, the conversion is implemented with the mapping rules from the source process definition language to the target one, which is similar to the approaches mentioned above. Differently, this paper uses PMR as the third part for the interoperation among heterogeneous process models, in which only the mappings from a modeling language to the registry metamodel are needed, rather than that between any two of those languages. Meanwhile, not all the modeling elements of a process modeling language should be mapped to that of PMR since it intends to register the selected metadata of

heterogeneous process models, which is essential for further discovery and reuse of process models.

Moreover, most of the modeling languages provide no semantic information to locate requested process models in a precise and efficient way. By introducing semantic techniques into metamodeling method, PMR allows users to register semantics of process models or add semantic annotations to process components at different levels of granularity. In this way, we can promote semantic-based discovery and interoperation of process models in virtual enterprises to some degree.

3 Introduction of PMR Metamodel

In this paper, PMR is proposed as an extensible and flexible mechanism to register and discover process models described with a specific process modeling language. It focuses on the selected metadata and common semantics representing the function and structure of process models, rather than the information related to the details of process modeling languages or the platform for process execution, such as fault handling mechanisms[9]. The main structure of PMR is shown in Fig. 1.

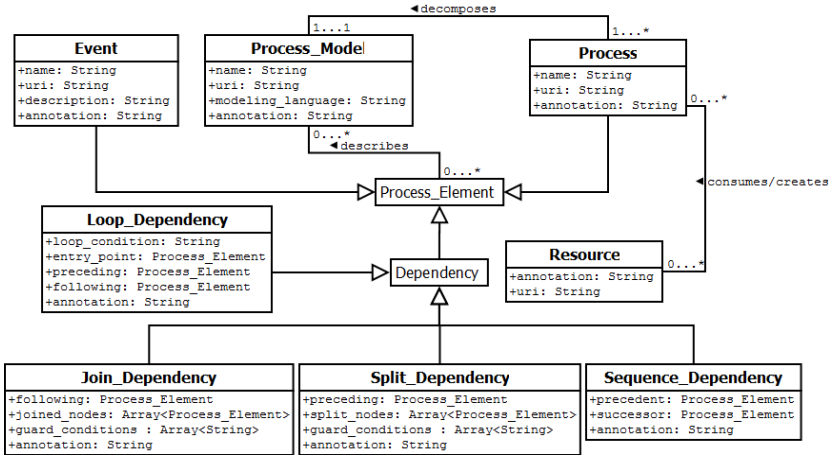


Fig. 1. The structure of PMR metamodel.

In the PMR metamodel, *Process* is the central metaclass representing a collection of related, structured activities or tasks that achieve a particular business goal. *Process* can be related to *Resources* with the association "consumes" or "creates", which is used to indicate inputs or outputs of a process, respectively. A process can be decomposed, and *Process_Model* in a specific modeling language can describe its decomposition with *Process_Elements*. *Process_Element* is an abstract metaclass that can be instantiated as *Processes*, *Events* or *Dependencies*. More specifically, *Event* designates an occurrence or a state at a particular point of time, and *Dependency* represents the control constraints among *Process_Elements* in a *Process_Model*.

The abstract metaclass *Dependency* can be generalized as *Sequence_Dependency*, *Split_Dependency*, *Join_Dependency*, and *Loop_Dependency* in PMR. In detail,

Sequence_Dependency connects two *Process_Elements* that comes sequentially. *Split_Dependency* accepts one *Process_Element* as its predecessor and forks into several branches. Each branch will be registered as a "split_node", and it precedes a *Process_element* and is associated with a "guard_condition". Similarly, *Join_Dependency* merges several branches into one branch. *Loop_Dependency* connects three *Process_Elements*, in which "entry_point" is the start of the loop, "preceding" node refers to the end of the loop, and "following" node is the first *Process_Element* coming after the loop.

All the *Process_Elements*, including *Processes*, *Events* and *Dependencies*, as well as *Resources* and *Process_Models*, may contain an "annotation" attribute. It refers to the URI of a concept of an ontology, and can be manually added when necessary for semantic-based discovery of business processes. Suppose two concepts are used to annotate two separate processes respectively that are stored in different repositories, and the relationship between those two concepts is defined in an ontology. If we query a process in a certain repository using one concept, the process annotated by another concept will be located and retrieved from other repositories in terms of the semantics in the ontology. In this way, the underlying relationship between distributed processes can be specified through semantic annotation provided by ontologies to promote semantic discovery of processes, and preserve recall and precision ratio of process discovery.

4 Mapping BPEL to PMR

This section defines the mappings from BPEL to PMR. When registering a BPEL process, the corresponding BPEL document(.bpel file) will be registered as the instance of *Process_Model* in PMR. The concrete activities and relationships between the activities described in the BPEL document will be registered as the instances of corresponding metaclasses or their attributes in PMR. Note that in this paper, not all the elements or their attributes in BPEL can be mapped to PMR accordingly. Some of the unmapped elements will be discussed in the second paragraph of section 5.

4.1 Mappings of the Main Process

In a BPEL document, the root element <process> is used to describe the main process, which represents the process that is described by the BPEL document as a whole and can be further decomposed. Thus, <process> and its attribute "name" in BPEL will be mapped to metaclass *Process* and its attribute "name" in PMR respectively. <variable> in <variables> in BPEL records the artifacts that are used to perform a process, so it can be mapped to metaclass *Resource* in PMR.

There are many other elements nested directly or recursively in <process> that are used to describe the decomposition of the main process. The mappings of activities are specified in the following sections.

Generally, there are two types of activities in BPEL, i.e. basic activity and structured activity. The mappings of the former will be discussed in section 4.2, and the latter will be presented in section 4.3. Each type of activity has two common

attributes "name" and "suppressJoinFailure". As for the attribute "name", it will be mapped to the attribute "name" of metaclass *Process_Element* in PMR. The attribute "suppressJoinFailure" is related to fault handling, which is out of the scope of PMR and the corresponding mapping will be omitted.

4.2 Mappings of Basic Activities

Basic activities in BPEL describe elemental steps of the process behavior. In this paper, basic activities are mapped to *Events* or *Processes* in PMR, as Table 1 shows. Note that the elements <assign>, <empty> and <exit> are not considered in the mappings since they are used to describe the details of process execution, which is out of the scope of PMR.

The <invoke> element allows a business process to invoke an operation offered by a partner, and it is the only element that can be mapped to *Process*. Its attributes "inputVariable" (or its equivalent element <toPart>) and "outputVariable" (or its equivalent element <fromPart>) are used to designate the inputs and outputs of <invoke> activity. Therefore, "inputVariable" and <toPart> can be mapped to the association "consumes" from *Process* to *Resource* in PMR. The value of "inputVariable" and the attribute "fromVariable" of <toPart> indicate the corresponding instances of *Resources*, which have been registered as specified in section 4.1 when handling <variables> element. The attribute "outputVariable" and <fromPart> element are mapped in a similar way.

Table 1. Mappings of basic activity elements.

Element or attribute in BPEL	Metaclass or association in PMR
<invoke>	Process
inputVariable of <invoke>	consumes (from Process to Resource)
<toPart> in <toParts>	consumes (from Process to Resource)
outputVariable of <invoke>	creates (from Process to Resource)
<fromPart> in <fromParts>	creates (from Process to Resource)
<receive>	Event
<reply>	Event
<wait>	Event
<throw>	Event
<rethrow>	Event

In this section, all the other basic activities in BPEL are mapped to *Event* except for <invoke>. For example, <receive> and <reply> are two essential activities for communication with other processes, such as receiving or sending a message to a specified port. So it's better to map them to *Events* rather than *Processes*. The <wait> element specifies a delay of the execution and can also be mapped to *Event*. Although <throw> and <rethrow> are used to handle faults, this paper maps them to a special kind of *Event* named "fault" to keep the consistency and integrity when registering the structured activities with these elements. Let's take <if> control flow with two branches as an example. If one branch refers to a normal activity while the other is a <throw> activity, then the <throw> activity cannot be omitted. Otherwise, this branch structure is incomplete.

4.3 Mappings of Structured Activities

Structured activities in BPEL prescribe the order in which a collection of activities is executed. They describe how a business process is created by composing the basic activities it performs into structures. There are three common types of structures, i.e. branch structure, loop structure and sequence structure. In this paper, they can be respectively mapped to *Split_Dependency*, *Join_Dependency*, *Loop_Dependency* and *Sequence_Dependency*, but there are still some differences when performing structure and dependency mappings due to the fact that BPEL is a block structured language [7] while PMR is not. Table 2 lists the mappings of those structured activities. Due to page limit, this paper only discusses how to map <if>, <pick> and <sequence> activities, and the mappings of other activities in Table 2 are omitted.

Table 2. Mappings of structured activity elements.

Element in BPEL	Metaclass in PMR
<flow>	Split_Dependency and Join_Dependency
<if>	Split_Dependency and Join_Dependency
<pick>	Split_Dependency and Join_Dependency
<repeatUntil>	Loop_Dependency
<while>	Split_Dependency, Join_Dependency and Loop_Dependency
<forEach>	Split_Dependency, Join_Dependency or Loop_Dependency (Note: The dependency type to be mapped is depended on the attributes of <forEach>)
<sequence>	Sequence_Dependency

(a) Mapping <if> and <pick> activities.

In Table 2, <if> is mapped into a pair of *Split_Dependency* and *Join_Dependency*. Practically, it's easy to convert all the branches of <if> activity into "split_nodes" in *Split_Dependency* and "join_nodes" in *Join_Dependency*. Considering all the branches of <if> activity, one of the branches is expressed as a <condition> element with its corresponding activity, another branch is <else> element, and all the other ones are <elseif> elements. For each branch, its condition should be recorded as the "condition" attribute of the *Split_Dependency* in the pair.

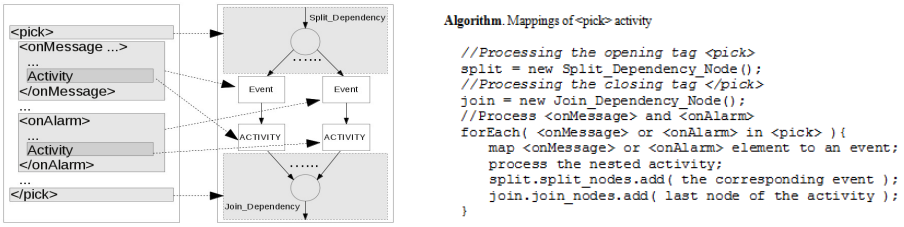


Fig. 2. Mappings and the corresponding algorithm of <pick> activity to PMR

The <pick> element in BPEL is used to wait for the occurrence of exactly one event from a set of events and execute the activity associated with that event. It is

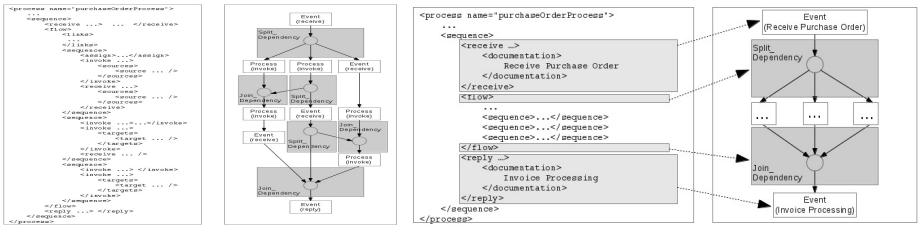
comprised of a set of branches, and each branch contains an event-activity pair. So <pick> can be mapped to a pair of *Split_Dependency* and *Join_Dependency*. Then, as shown in Fig.2, its sub-elements <onMessage> and <onAlarm> can be treated as sequence control flows, and each sequence control flow starts with an event that the <pick> activity waits for.

(b) Mapping of <sequence> activity.

In BPEL, <sequence> activity is the simplest activity, and the contained tasks in <sequence> activity are performed sequentially. In PMR, *Sequence_Dependency* is not a container of several *Process_Elements*, but a connector between two *Process_Elements*. When mapping <sequence> activity to *Sequence_Dependency*, it's needed to attach sequence dependencies to the process elements. For instance, if there are N activities in a <sequence> activity, at most N-1 *Sequence_Dependencies* should be inserted for mapping. However, it's not necessary to attach a sequence dependency to each process element. In general, a sequence dependency should be added only when neither the preceding process element nor the following one is an instance of *Dependency*. For example, if a <receive> activity and an <invoke> activity are executed sequentially, a sequence dependency should be inserted between them. If they are <if> and <invoke> activities, no sequence dependency is required.

4.4 Example of Implementing the Mapping Rules

This section takes the BPEL process "PurchaseOrderProcess" from [2] a registration example to evaluate the effectiveness of PMR metamodel and show how to use the proposed mappings from BPEL to PMR in section 4.3.



(a) PMR-based skeleton of registering process "PurchaseOrderProcess".

(b) Mappings from nested activities in BPEL to the metaclasses in PMR.

Fig. 3. Example of implementing the mapping rules.

If the process in the example above is taken as a whole, its nested activities will be mapped to the PMR-based skeleton in Fig.3(a), with the processes and the dependencies between them. As shown in Fig.3(b), the mapping rules of <receive> and <reply> are used to generate events named "Receive Purchase Order" and "Invoice Processing" respectively when registering the process model, while the mappings of <flow> activity is used to generate the instances of *Split_Dependency*

and *Join_Dependency* as the registration information of the process. The rest of the exemplary process will be handled similarly and the trivial descriptions are omitted in this section.

5 Conclusion and Future Work

In this paper, a registration metamodel called PMR is proposed to register BPEL process models. Then, the mapping rules from BPEL to PMR with mapping algorithms are defined to transform BPEL process models to the corresponding registration information based on PMR. Next, we use an example to show that the mapping rules can facilitate automatic registration of BPEL processes effectively.

However, not all the elements in BPEL specification can be mapped to the corresponding metaclasses and associations of PMR in this paper. For example, <extensionActivity> and <scope> are not taken into account due the complexity of the mappings, and a more comprehensive mapping will be given in the next step. In addition, the mapping rules and the corresponding algorithms are incomplete due to page limit, and considerable work is needed to be done later. Finally, we plan to define the mappings from other process modeling languages to PMR, and develop a PMR-based registration tool to support automatic registration of more process models in practice.

References

1. Jeusfeld, M., Backlund, P., Ralyté, J.: Classifying Interoperability Problems for a Method Chunk Repository. Enterprise Interoperability II, 315-326. (2007)
model registration. Master thesis. Wuhan University.(in Chinese) (2011)
2. OASIS: Web Services Business Process Execution Language Version 2.0 Standard. (2007). http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html. (2007)
3. Khodabakchian, E., Shaffer, D., Gaur, H., Zirn, M.: SOA Best Practices : The BPEL Cookbook. http://www.oracle.com/technetwork/articles/soa/index-095969.html
4. Aslam, M.A., Auer, S., Shen, J.: From BPEL4WS Process Model to Full OWL-S Ontology. Faculty of Informatics-Papers, 433. (2006)
5. Yuan, P., Jin, H., Yuan, S., Cao, W., Jiang, L.: WFTXB: A Tool for Translating between XPDL and BPEL. In: High Performance Computing and Communications, 647- 652. (2008)
6. Schumm, D., Karastoyanova, D., Leymann, F., Nitzsche, J.: On Visualizing and Modeling BPEL with BPMN. In: Grid and Pervasive Computing Conference, 80-87. (2009)
7. Gao, Y.: BPMN-BPEL transformation and round trip engineering. Technical report, eClarus software. (2006).
8. Li, H., Lan, Y., Yang, L., Guo, S.: A Framework for Workflow Process Definition Transformation Based on Meta-Model. In: Computer Science and Software Engineering, 566-569. (2008)
9. Chong W., Keqing H., Wen Z., et al.: Personalized Reuse of Business Process through the Metamodel for Process Model Registration. In: The 4th International Workshop on Personalization in Grid, Service and Cloud Computing (PGSC 2010) at The 9th International Conference on Grid and Cloud Computing (GCC 2010), pp.438-443. IEEE Computer Society, Nanjing. (2010)