A Framework towards Web Service Composition Modeling and Execution

Muhammad Adeel Talib, Zongkai Yang

Electronic & Information Engineering Department,
Huazhong University of Science & Technology,
Wuhan, Peoples Republic of China
matalib@gmail.com; zkyang@public.wh.hb.cn

Abstract. The biggest challenge of business process management is the provision of non-technical tools, based on implementation standards, which swing control of business processes away from technical departments and towards the business process owners themselves. These tools aid business users in designing high level process models using graphical notations which can then be mapped to lower level implementation models for execution. In this paper we propose our framework leading to a tool that aids business user in designing Web service based process (or in other words, Web service compositions) in BPEL4WS. We elaborate what information is required from the user in order to model the composition and how the technological details can be hidden from her. It is our conjecture that such tool will facilitate Web service composition design and development by giving an upper hand to business users – the people who actually conceptualize the processes.

1 Introduction

Used together, business process management and service-oriented architecture can form a dynamic combination that leverages the agility and extends the capabilities of both technologies. In this arena individual Web services are federated into composite services with value added functionality. Organizations can encapsulate their business functions as Web services and create virtual processes that interact with other organizations' processes. The interaction logic is specified as an XML based business process language. One such candidate language that seems to have attracted the most attention at the moment is the Business Process Execution Language for Web Services (BPEL4WS or BPEL in short) [1] that was originally drafted by BEA, IBM and Microsoft and which is now being formalized by a committee at OASIS. Though the language is still going through refinement with its new version under planning phase, it has gained a lot of attraction and support from the industry and has become the *de facto* standard [2].

According to the BPEL specification, BPEL defines a model for describing the behavior of a business process based on interactions between the process and its partners. This statement creates a misconception that BPEL is a business process modeling language. It is rather an execution language [business process modeling and standardization]. Like other XML based languages, it is of textual form and contains complex constructs and not so easy semantics. Business operations people are used to flow diagrams and other graphical notations instead of textual notations. BPEL attempts to offer the best by introducing a flow construct and using links to create 'arbitrary' flow dependencies between the activities contained within the flow construct. However, the semantics relies on a complicated formulation which tests and propagates the status of links. This makes it difficult for a business user, who actually conceptualizes the process, to model the process in it. There is a need to develop a methodology in order to assist the business analyst, who is not a technology expert, to model process compositions. Our aim is to develop a tool that captures the explicitly required information about a composition from the business modeler while at the same time hiding the technological details from her.

In this paper, we have briefly describe our proposed framework that can capture high level process composition requirements in an abstract way and then automatically transforms the high level process design model into low level process execution model i.e. BPEL. We identify various concepts of BPEL metamodel and see how these can be captured with abstraction from the user being a major concern.

Rest of the paper is organized as follows: First of all we present the related work in Section 2 as it gives an idea about the significance of our work. In Section 3 we list various concepts which are required to be captured in order to produce a Web service composition. Then in Section 4 we briefly discuss our proposed framework and describe how it facilitates the business user in capturing the BPEL semantics. Finally we conclude in Section 5 along with future work.

2 Related Work

BPEL as well as other process composition languages are of textual form and the specifications written in them are difficult to understand and visualize [3]. Towards this end various graphical specifications have been proposed to model Web service compositions that sit on top of BPEL in the technology stack. These tend to reduce the design complexity by provided graphical representations. Authors in [4] describe a UML profile and transformation rules that can be used to produce UML models of Web service compositions. The operation signatures are modeled as UML class model and the behavior is modeled in UML activity diagram thus producing a new service model. [5] present a similar UML profile and mapping rules to BPEL. Although these UML based modeling techniques provide a helping hand to developers to model applications, they do not abstract away the syntactic details from the user. The user still has to acquire sound knowledge of the underlying specifications as well as the graphical notations along with their usage scenarios. Besides, as quoted in [6], the gap between UML and BPEL is very large which makes the mapping quite complex.

BPMI has recently proposed a language – Business Process Management Notation (BPMN) [7] which provides graphical constructs with mapping to BPEL. Numerous commercial BPEL implementations are available where the vendors provide visual designing tools such as BPWS4J from IBM and BPEL Server from Oracle. In both cases either the user is bound to learn a new graphical language or should have sound knowledge of BPEL constructs. [8] present an approach to visually model Web services composition using Object-Process Diagrams (OPD). The paper describes a two-way transformation from OPD to BPEL using OPD templates. Again, the approach assumes the user is familiar with BPEL and OPDs. In contrast to UML and other graphical modeling based solutions, we focus on relieving the user from the syntactic details of BPEL along with the modeling notation and providing her a user friendly graphical interface where she can design the composition in step by step manner without any prerequisite modeling or programming skills.

Authors in [9] have proposed a template based web service composition model for automatic code generation of business process languages. The degree to which automatization of service composition is achieved depends on the availability of templates (to the designer) that fit into the desired composition pattern. The availability of templates, in turn, depends on access to public registries for retrieval. Currently there is no such provision and is in the future scope of the project. Our conjecture is that these templates can provide automatic code generation only for a part of code that is repeatable but, cannot be used to generate the whole process flow, for that the designer has to gain knowledge of complex design patterns. Our framework focuses on the whole process flow design instead of partial code generation.

3 Concepts of BPEL metamodel

The full explanation of BPEL semantics is out of the scope of this paper. Here we only discuss the concepts involved. The metamodel of BPEL incorporate the following concepts [10] that represent the operational, behavioral, informational, organizational and transactional aspects of the language:

1. *Task I/O:* Task refer to basic units of work or activity. The input and output (I/O) of these tasks may be modeled using simple or XML complex types.
2. *Task Address:* The address specifies where or how a service can be located to perform a task. The address can be modeled directly via a URI reference of a service or indirectly via a query that identifies a service address.
3. *Control Flow:* The control flow defines the temporal and logical relationships between different tasks. Control flow can be specified via directed graphs or block oriented nesting of control instructions.
4. *Data Handling:* Data handling specifies which variables are used in a process instance and how the actual values of these variables are calculated.
5. *Instance Identity:* This concept addresses how a process instance and related messages are identified. Correlation uses a set of message elements that are unique for a process instance in order to route messages to process instances.
6. *Roles:* Roles provide for an abstraction of participants in a process.

7. *Events:* Events represent real-world changes. Respective event handlers provide the means to respond to them in a predefined way.
8. *Exceptions:* Exceptions or faults describe errors during the execution of a process. In case of exceptions dedicated exception handlers undo unsuccessful tasks or terminate the process instance.
9. *Transactions:* Business transactions represent long-running transactions. In case of failure the effects of a business transaction are erased by a compensation process.

Capturing these concepts of BPEL while at the same time hiding the syntactic details is not an easy task. The higher level metamodel required to do so must be business user friendly that captures information relevant to much lower level BPEL implementation model. This difficultly also implies to the automatic transformation between the two metamodels. We advocate the use of a user friendly interactive graphical interface instead of a graphical modeling language to capture these concepts. The information is stored in a relational model (Fig. 1) which is then transformed into the BPEL model automatically.

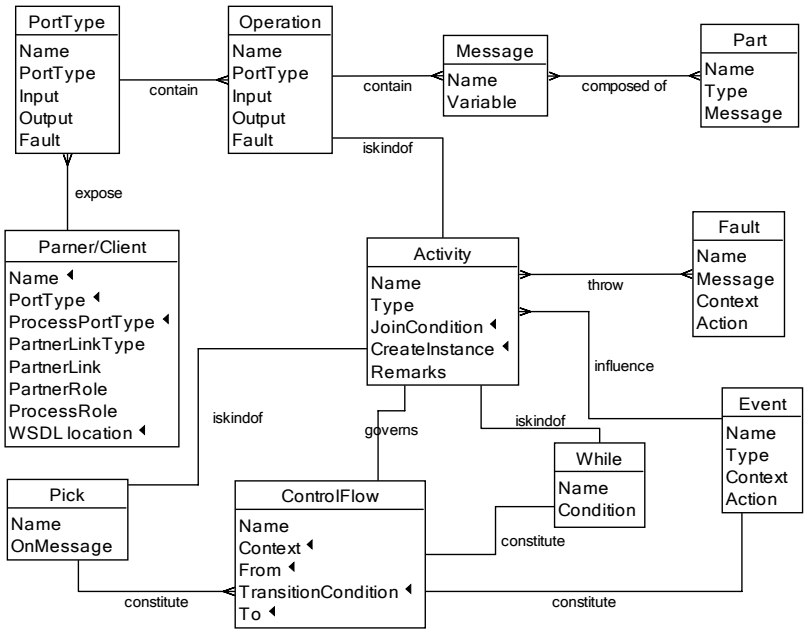

Fig. 1. Relational model which stores the composition information. The marked attributes represent the only information provided by the user

It should be noted that only basic understanding of the concepts of BPEL is required by the user and not its syntactic details. Hiding syntactic details from the business user is the basic theme of our research. Our goal is to provide composition modeler the flexibility to express less than complete detail without prerequisite high specifica-

tion knowledge. Refinement is then a natural process of adding further detail, while still conforming to the laws of composition. The motive here is to facilitate the business user, who does not know much detail about composition language, to construct.

4 Proposed Framework

Through the Graphical User Interface (GUI), the system captures the information required to develop the composition from the Composition Modeler and stores it in a Relational Repository. Knowledge about the parties involved in the collaborations, their interface description file locations, order of activities, instance creation, correlation token, etc. is captured in an incremental fashion in such a way that the modeler remains unaware of the underlying syntax. The Inference Engine automatically infers additional information from the user provided information using inference algorithm (an algorithm that takes values of attributes from the relational model and deduces other attributes). Once all information is captured, transformation rules are applied to map from relational to BPEL metamodel/schema (the word schema is more appropriate to use here as both the relational and BPEL model use schemas as their interchange format). This is done by the Transformation Engine. Once the code is generated, it must be validated for accuracy and verified against deadlocks. This is done using an existing BPEL Validator. The final validated and verified BPEL file and partner interfaces are passed on to an execution engine that executes the composition. The architectural components of our proposed framework are shown in Figure 2.

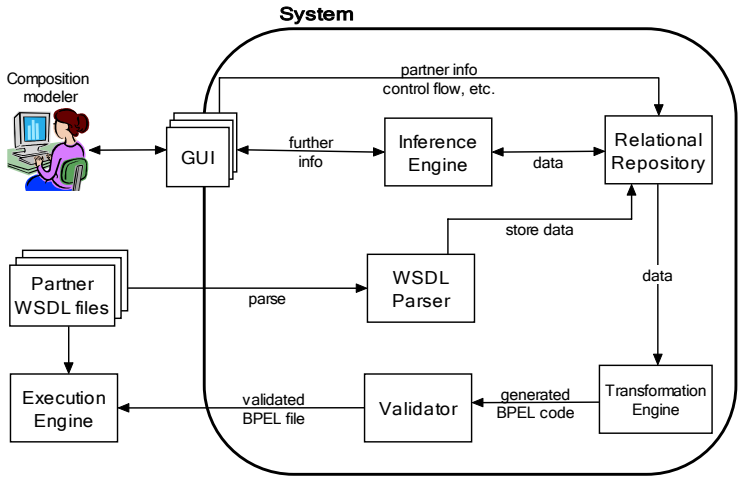

Fig. 2. Conceptual architecture of proposed framework

4.1 Information Capturing

This section describes briefly how our framework captures the BPEL concepts (given in Section 3) keeping the user unaware of the syntactic detail and what information can be inferred from the inference algorithm.

4.1.1 Task I/O. This information is deduced from WSDL Parser. It parses the partner interface files provided by the user and deduces the port types exposed by the collaborating parties, the operations offered, the messages involved and the parts of the messages that correspond to basic or complex data types. Fig. 3 shows mapping between the partner WSDL file and the relational model.

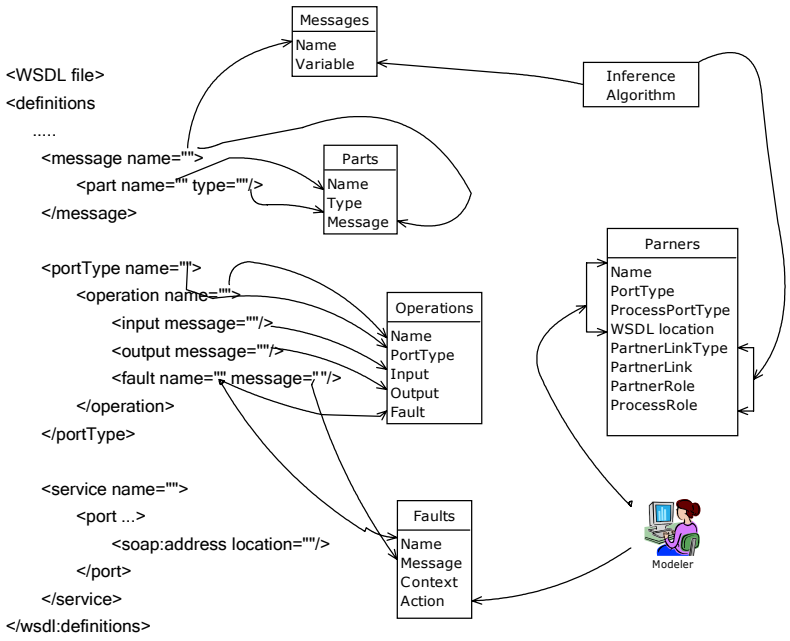

Fig. 3. Mapping from WSDL files to relational model done by WSDL parser. The figure also shows what information is provided by the user and what is inferred from the inference algo.

4.1.2. Task Address. The task address is a URL and is also extracted from the WSDL files from binding element.

4.1.3. Control Flow. The modeler captures the control logic with the help of simple control rules that identify which activity has to be executed after an activity is completed under what conditions. Structure of a control rule is as shown below:

```
ControlRule {

activity (messaging|basic|while|pick|event)

postActivity (messaging|basic|while|pick|event)

transitionCondition (Boolean expression)

}
```

Synchronization (interdependencies) among the activities is handled by BPEL link semantics which mark the parent node as source activity and the child node as target activity. If the target activity is guarded by a condition, the source activity will specify the transition condition. Based on this control link semantics of BPEL, the transformation rules synchronize the activities in a single flow. We believe that this graph based technique allows more abstraction as compared to structured formation of BPEL specification, in that the modeler has to explicitly define the complex structure of composition.

Here inference algorithm is used to decide whether a messaging activities i.e. operation is associated with a receive, reply or invoke construct. The modeler just defines the order in which the activities have to be run without knowing when to use the BPEL messaging constructs. The use of control rules also facilitates the business user in updating changes in the business logic without relying on the developer, providing agility against change.

4.1.4. Data Handling. The message exchange variables are deduced from the description files of the partner interfaces. For each unique message taking part in the collaboration, there is a unique variable. Intermediate variables used to store data during business logic manipulation have to be defined by the user.

4.1.5. Instance Identity. Three instantiation patterns are involved during instance creation: Single start; Multiple start with receive; and Multiple start with pick. The modeler has to select the identified pattern according to the business logic. Correlations are usually context-dependent and thus cannot be derived by general rules. They have to be defined by the user. We are currently working how to abstract the correlation constructs from the user.

4.1.6. Roles. In a business collaboration modeled by BPEL, there is a centralized coordinating authority, the process, which interacts with other parties (i.e. partner and client). It is important to distinguish between a partner and a client. A partner is the party that provides services to the process. Client on the other hand gets service from the process. It is required from the modeler only to provide the port types exposed by the partner or client. The partnerLinkType, partnerLink, partnerRole and myRole constructs can be deduced from the inference algorithm. The detail of the inference algorithm is out of the scope of this paper but a snippet of the algorithm inferring the role semantics of a partner is listed below:

```
for each partner involved {

PartnerLinkType = Partner.Name + 'LinkType'

PartnerLink = Partner.Name + 'Link'

if (Partner.PortType != null then)

PartnerRole = Partner.Name + 'Provider'

if (Partner.ProcessPortType != null then)

MyRole = Partner.Name + 'Requester'

}
```

4.1.7. Events. Events represent real-world changes and their occurrence is captured by the control rules. The activity which is influenced by an event is encapsulated in a scope having an event handler attached that executes the actions to be performed. The actions to be performed are again captured with the help of control rules. The flow thus created will be nested inside the event activity.

4.1.8. Exceptions. Our framework automatically captures the application exceptions from the WSDL files and attaches fault handlers with the affected activities. Handling of system exceptions and compensation is yet to be explored.

5 Conclusions and Future Work

In this paper we have presented our idea for generating BPEL code for static composition semi-automatically. First we described various aspects of BPEL to be captured while modeling the business process choreography. We discussed what information can be abstracted from the user and what cannot. Then we proposed a relational model to store the captured information which is later transformed into BPEL process specification model. We also briefly described how the relational model is populated with information plugged in by the modeler and information derived by inference algorithms. It is our thesis that using this approach a business user with limited BPEL knowledge can have the capability to design complex business to business Web service compositions.

In order to verify our concept a tool is under development. For the sake of simplicity we have abstracted from the namespace issues which are an integral part of BPEL specification. Both relational and process models have strong formal grounds (relational and process algebra). The possibility of formally transforming the relational model into a process model is yet to be explored. Currently our framework only generates abstract composition without considering concrete bindings with the partners, which will be a part of our future work. For now we concentrate on static composi-

tions where the modeler provides the service location information. Our framework can be modified to handle dynamic compositions where the services are selected at runtime. For this functionality we propose the use of a service broker that provides the service URLs upon queries based on functional and non-functional requirements.

References

1. Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., Weerawarana, S.: Business Process Execution Language for Web Services, Version 1.1. Specification. BEA Systems, IBM Corp., Microsoft Corp., SAP AG, Siebel Systems. May, 2003. http://www.ibm.com/developerworks/ webservices/library/ws-bpel/
2. Harmon, P.: BPM Tools. Business Process Trends Newsletter, vol. 2, no. 4, Apr, 2004. http://www.bptrends.com
3. Wil M.P. van der Aalst: Web service composition languages: Old wine in new bottles? In: Proceedings of the 29th IEEE EUROMICRO Conference, pp. 298-307, Belek-Antalya, Turkey, Sep, 2003, 298-307
4. Skogan, D., Gronmo, R., Solheim, I.: Web Service Composition in UML. In: Proceeding of the 8th International IEEE Enterprise Distributed Object Computing Conference (EDOC'04), California, USA, Sep, 2004, 47-57
5. Gardner, T.: UML Modeling of Automated Business Processes with a mapping to BPEL4WS. In: Proceedings of 1st European Workshop on Object Orientation and Web Services (EOOWS'03), Darmstadt, Germany, 2003
6. Bézivin, J., Hammoudi, S., Lopes, D., Joualt, F.: Applying MDA approach to B2B applications: A road map. Workshop on Model Driven Development (WMDD 2004) at ECOOP 2004, Oslo, Norway, Vol. 3344 of Lecture Notes in Computer Science (LNCS), Springer-Verlag, Jun, 2004
7. White, S. A.: Business Process Modeling Notation. BPMN 1.0, Business Process Modeling Initiative. 2004. http://www.bpmn.org
8. Yin, L., Wenyin, L., Changjun, J.: Object-Process Diagrams as Explicit Graphic Tool for Web Service Composition. Journal of Integrated Design & Process Science: Transactions of the SDPS, Vol. 8, No. 1, 2004, 113-127
9. Karastoyanova, D., Buchmann, A.: A Procedure for Development and Execution of Process-based Composite Web Services. In: Proceedings of the IEEE International Conference on Web Engineering (ICWE'04), Munich, Germany, Jul 2004, 593-594
10. Mendling, J., Neumann, G.,Nüttgens, M.: A Comparison of XML Interchange Formats for Business Process Modelling. In: Proceedings of EMISA 2004, Luxembourg, In: Feltz, F., Oberweis, A., Otjacques, B., (eds.), Vol. 56 of Lecture Notes in Informatics (LNI), Oct, 2004, 129-140