ONTOLOGY-BASED AND AUTOMATIC GENERATION OF SERVICE INTERFACES FROM SEVERAL CHOREOGRAPHIES

SAEID KAMARI[1], MOHAMMAD R. KHAYYAMBASHI[2]
[1]Islamic Azad University, Sahneh Branch, Department of Computer Science and Engineering, Kermanshah, IRAN
[2]University of Isfahan, Computer Department, Faculty of Engineering, Isfahan, IRAN

E-mail: S.kamari@shbu.ac.ir , M.R.Khayyambashi@eng.ui.ac.ir

ABSTRACT

Service Oriented Architecture (SOA) is a paradigm for developing distributed and heterogeneous software applications within and across organizational boundaries. Choreography is a coordination model of SOA in which service collaborations to achieve a common goal are described from global point of view. One of the most important issues in SOA is identifying required services and their interfaces. Service interfaces are necessary for searching required organization services or developing them from scratch. Because of involving key information of service interfaces in choreography, it can be used in generation of service interfaces. This paper presents an algorithm for automatic interface generation of service interfaces from several choreographies using ontology. Ontology assists to conceptualize a specific domain of knowledge. This method helps developers to facilitate, automate and speedup a part of development process of SOA-based software systems.

KEYWORDS: *Ontology, Service Oriented Architecture (SOA), Service Interface, Web Service Choreography Description Language (WS-CDL).*

1. INTRODUCTION

Service-Oriented Architecture represents an emerging paradigm to develop flexible and large-scale software systems using the Internet as the main infrastructure. Web services are one realization of this paradigm by using well-established standards to describe and interact with other services [1], [3].

Many organizations build their cross-organizational business processes based on Web services because of their platform-agnostic nature and the ease of integration. Currently available technologies such as composition engines using Web Service Business Process Execution Language (WS-BPEL) [5] can be used to orchestrate business process within organizations.

An engineering method for Web service based business process involving multiple partners requires an agreement on the data that is exchanged which can not be achieved by WS-BPEL. For this purpose, the Web Service Choreography Description Language (WS-CDL) [2], [7] - [9] provides a XML-based language to describe the cross-
organizational message exchange from a global viewpoint.
This different views (local vs. global) are described by the terms orchestration and choreography. Choreography can be
defined as processes involving multiple services where the interactions between these services are seen from a global
perspective [7], [9]. Choreography does not describe any internal actions that occur within a participating service, such as internal computation or data transformation, but rather focuses on the observable

public exchange of messages. In contrast, orchestration is an executable business process that interacts with both internal and external Web services [4]. These interactions occur at the message level. They include business logic and task execution order, and they can span applications and organizations to define a long-lived, transactional, multi-step process model.

Web Service Description Language (WSDL) is a main standard of Web service stack used to describe services and their operations. It includes important information related to service such as service interface, way of binding to service and physical address of service. Service interface, in turn, is composed of service name, method signatures and their input/output parameters [6], [12].

In according to definitions for choreography and service interface, it is possible to use choreography descriptions in automatic interface generation of services participating in several choreographies using ontology. Ontology provides a means for modeling a Universe of Discourse or a special knowledge domain. It can be used to solve semantic inconsistencies between various terms in diverse choreographies and create similar view on them. Nowadays, there is no ontology-based method for automatic interface generation of services from multiple choreographies.

In this paper, a method is given for identifying candidate services participating in choreographies and automatic generation of their interfaces. This approach involves an algorithm in which multiple WS-CDL files are fed as input parameters and behavior interface of services participated in choreographies are returned as output products. In addition to, this algorithm utilizes ontology to match choreographies.

This paper is organized as follows: Section 2 introduces the basic concepts of this paper. Section 3 describes related works. Section 4 describes our main approach and the proposed solution and finally section 5 concludes the paper and states the future works.

2. BASIC CONCEPTS

In this section, the basic concepts and techniques which are used in our approach are introduced.

2.1. An Overview of WS-CDL

The goal of WS-CDL is to define multi-party contracts in which external observable behavior of Web services and their clients is defined by specifying messages exchanged between them. Choreography descriptions are used to generate abstract BPEL processes or Web services code skeleton.

WS-CDL offers a non-executable XML-based specification language which allows each involved part to describe its part in message exchange by specifying details on collaborations, information handling and activities. The UML class diagram in figure 1 provides a high-level overview of WS-CDL's underlying meta-model. It describes the concepts of package and choreography [9]. In the following paragraphs the above concepts are described.

Collaborations. The collaborations of choreography are specified by defining participantTypes, roleTypes, relationshipTypes and channelTypes. These declarations define collaborating participants and their coupling.

A participantType declares an entity playing a particular set of roles in the choreography. Thus, a participantType definition contains one or more roleType definitions.

A roleType defines a role that enumerates the observable behavior a participant can exhibit in order to interact throughout a message exchange. A roleType definition declares a behavior interface which identifies a WSDL interface.

The relations between roles are defined through relationshipType definitions. A relationship type always contains exactly two roleTypes, restricting the relationshipType definition to 1:1 relations.

A channelType definition specifies where and how information between participants is exchanged by defining a reference to a role type which is the target of an information exchange (either the receiver of a message request or the sender of a message reply). This role type reference indicates the behavior interface which is used throughout the information exchange.

Information Handling. The definition and handling of information within choreography is performed by informationTypes and variables.

Information used within choreography is specified by informationTypes which do not directly reference data types but rather reference type definitions. Such

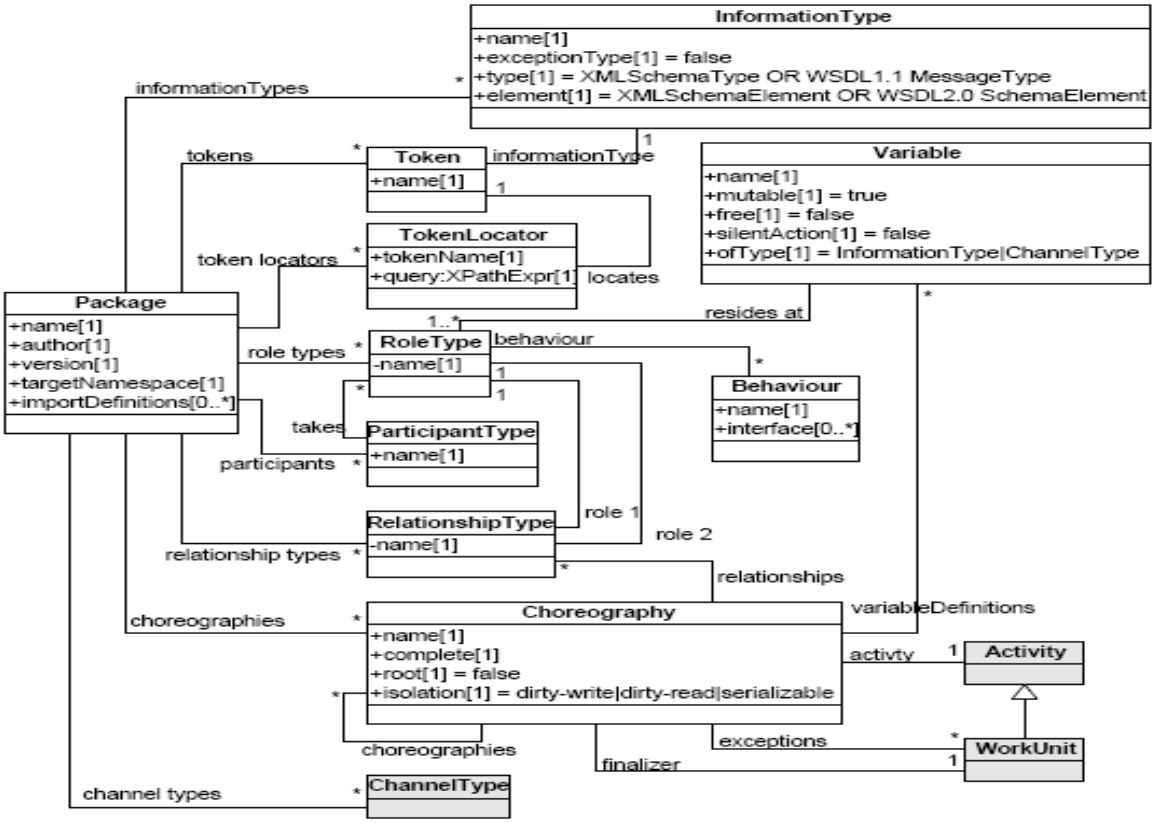

Fig. 1.WS-CDL's meta-model in UML

a referenced type definition can be either a WSDL 1.0 message type, an XML schema type, a WSDL 2.0 schema element or an XML schema element.

Variables capture information about objects in choreography such as exchanged information or the observable information of roleTypes. Variables either bound to informationType or channelType definitions. These information Types can either belong to application or state information.

Tokens are enumerated as aliases for information types. Tokens can be used in correlation of exchanged messages by declaring in channels or defining as the identifiers. Token locators are means for locating and deriving tokens from their carriers using XPath expressions.

Activities. A choreography comprises three different types of activities, namely ordering structures, workunits, and basic activities.

Ordering structures are block structured, enclosing a number of activities or ordering

interface. The attribute operation corresponds to a SOAP operation which is defined throughout this WSDL interface. The element participate defines the requesting and receiving part of the interaction. structures which can be used recursively. Such activities include sequence for handling activities in sequential order, parallel for a parallel execution of activities, and choice for handling data or event-driven conditions.

Workunits prescribe conditional execution of an activity. This conditional execution can either be repetitive (attribute repeat is set to true), competitive (multiple workunits are defined within a choice activity) or blocking (attribute block is set to true). The conditional statement is defined by the attribute guard which specifies a Boolean conditional expression according to XPath 1.0 lexical rules.

Basic activities define interactions, actions or variable assignments of the choreography flow. An interaction activity defines the information to be exchanged and by what means this information exchange will be performed. The attribute channelVariable binds the interaction to a channelType and therefore to a specific WSDL Finally the element exchange defines whether the interaction is a request or response and which variables will be used throughout the message exchange.

The other basic activities include assign, silentAction, perform and noAction. The assign activity enables the creation and manipulation of variables within the choreography. The silentAction activity defines a non-observable behavior which is either performed by one or all the participants in the choreography. A silentAction has to be further defined in the orchestration layer e.g., in the BPEL process of the corresponding participant.

The noAction activity represents a point in choreography in which no special task is performed by a role type, neither manipulation of state information (non-observable behavior) nor interactions (observable behavior). The perform activity is also used to call another choreography to be performed within the context of executing choreography. The called choreography may be defined within the same package as the caller, or it may be from a completely separate package that has been imported [2].

A WS-CDL tool suite from Pi4soa [13] is available to allow the modeling of choreographies without the need to write the XML representation directly.

2.2. Service Behavior Interface

Service behavior interface is a part of Web Service Description Language. WSDL is an XML-based language to describe Web services, their interfaces, and way of binding and using them. Three different versions of WSDL are now available that latest one is WSDL 2.0.

WSDL files define services in a bottom-up model, namely start with data type definitions and end with service physical address. Each WSDL file has three description layers:

- The first layer describes the service interface which contains one or more operations and their input/output parameters.
- The second layer describes how to use a Web service and bind to it. The required protocols to access the provided service operations are also defined at this layer.
- The third and last layer of description defines the physical location and address of the service.

The structure of WSDL 2.0 file and its various parts is shown in figure 2.

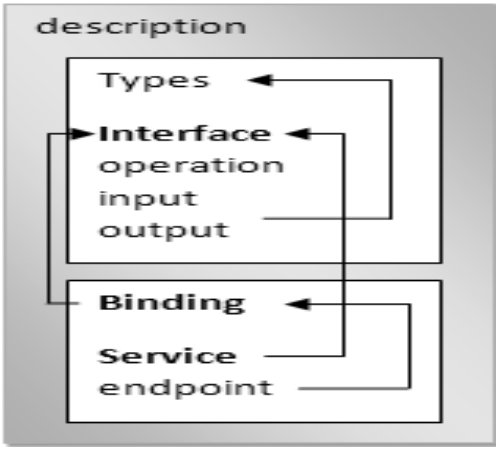

Fig. 2.Structure of WSDL file

2.3. Ontology

Ontology is the formal and explicit specifications of a knowledge domain and includes some terms and their inter-relationships. The terms stand for critical concepts, entities and objects of the domain. There are also assumptions in ontology about meaning of the terms. Furthermore, ontology comprises some rules to infer new knowledge. An ontology defines a common vocabulary for researchers who need to share information in a domain. A typical ontology is composed of basic elements such as individuals (instances), classes (concept), attributes and relations [14].

Ontology language is a formal language for encoding ontology to be used in practice. The most common ontology language is Web Ontology Language (OWL). OWL is intended to provide a language that can be used to describe the classes and relations between them that are inherent in Web documents and applications [15]. The OWL language is used to:

- formalize a domain by defining classes and properties of those classes
- define individuals and assert properties about them
- reason about these classes and individuals to the degree permitted by the formal semantics of the OWL language.

An ontology tool suite called Protégé [16] is now available to allow the semantic modeling of knowledge domains using OWL language. It permits to create ontology graphically without the need to write the OWL representation directly.

3. RELATED WORKS

WS-CDL is used to guarantee collaboration within and across the domain of controls (organizations), so it can be possible to reduce the collaboration problems, and presenting useful solutions within and across organizations using WS-CDL. It guarantees that services are well-behaved to participate in choreography, so it can be used as a means to reduce the overall cost and time of designing and testing the software systems in distributed environments [9].

Because of mentioned advantages, applying and using WS-CDL has increased at different aspects of SOA. Use of choreography in coordination and composing services, applying the choreography in executing business processes and automatic generation of WS-BPEL from WS-CDL [10] are samples of these attempts and researches.

Superimposing over composition layer in Web service stack causes WS-CDL to interoperate with some of Web service standards like WSDL, WS-BPEL. Thus, many researches are focused on mapping WS-CDL to other standards. One of these researches led to present a method for mapping a single WS-CDL to WSDL. The necessary element mapping from WS-CDL to WSDL and WSDL generation pseudocode can be found in [2].

4. PROPOSED SOLUTION

Services are basic elements of SOA. Candidate service interfaces are required in designing and developing SOA-based software systems. Thus, this paper presents an approach for generating the interface of candidate services participating in several choreographies. To achieve this goal, an algorithm is needed that takes WS-CDL files and their corresponding ontologies as input arguments, and returns the service interfaces as the output. Figure 3 shows a general schema of the algorithm.

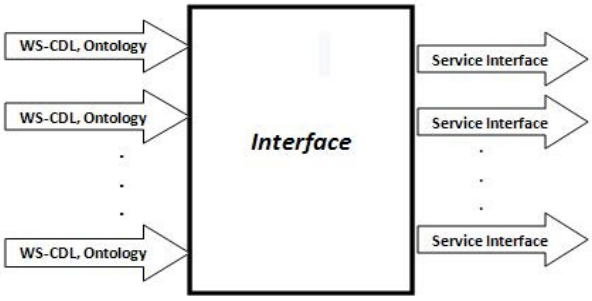

Fig. 3.General Schema of algorithm

This algorithm is comprised of two distinct steps. At first step, the available ontologies are combined by pairs to generate single aggregated ontology. The ultimate produced ontology includes the general choreography containing all the possible elements in all the collaborations. At the second step, candidate service interfaces are generated automatically by a method presented in [2]. In other words, this algorithm implicitly converts multiple choreographies into single aggregated choreography which contains candidate services information.

In matching process of couple ontologies, the algorithm calls a terminology server to find out the similar and synonym terms. The terminology server's rules can be specified by users, organization's expert and stakeholders or be acquired intellectually by searching the web content.

Choreography's meta-model [9] describes its basic elements, each element's attributes and relations between various elements. The meta-model's elements can be easily mapped to ontology's classes, attributes and relations. Thus, this algorithm uses this model to define basic ontologies for input choreographies.

Below assumptions have been held in developing the algorithm:

- Top-down approach is followed in designing and developing the software system in which problem, system or process is divided to smaller parts until reach the service level [6]. In other words, process descriptions are used to derive their composing service interfaces.

- Choreography descriptions and WS-CDL files are pre-defined and belong to algorithm primitives. The designing and generating method of choreography descriptions, e.g., from WS-BPEL [11], does not locate in the scope of the algorithm.

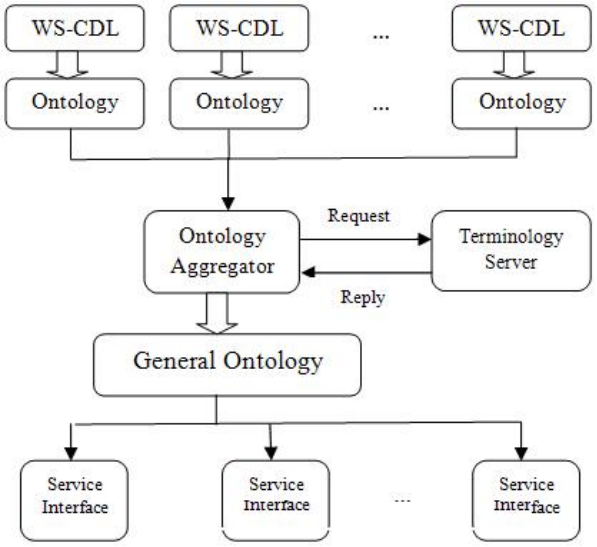

Fig. 4. Algorithm's operational logic

The algorithm details have been shown graphically in above figure (fig. 4). It implies algorithm's operational logic.

The existence of such an algorithm for automatic and ontology-based interface generation of service participating from multiple choreographies seems significant and useful in developing SOA-based software applications. In addition to, this algorithm has many advantages as follows:

- Quick and automatic service generation
- Increasing the speed of developing Service Oriented systems
- Making service interfaces more flexible to participate in several choreographies
- Improving reuse capability of services
- If a service participates in multiple choreographies, this algorithm generates just a single service interface not as the number of choreographies.
- Form business point of view, it is not necessary for clients to adapt with service providers in order to use their services anymore. In other words, this method helps providers to generate services consistent with several coordination protocols

5. CONCLUSION AND FUTURE WORKS

Widespread use of Service Orientation and Web services by industry represents their high acceptance and performance. This acceptance has resulted in increasingly attempts of IT-owners and software engineers to simplify and automate different stages of designing and developing Service Oriented systems.

Focusing on communication and reuse properties of Web services has caused the providers have trend toward generation of services with most reuse capability. It is obvious that such generated services can potentially participate in more collaboration. In this paper, an algorithm was presented for mapping multiple choreography files into service behavior interfaces using choreography model and ontology.

Recent uses of choreography model in different aspects of Service Oriented Architecture and mapping WS-CDL into other Web service standards have resulted in many new research fields. Sample future Works of this paper can be generation of algorithm pseudocode, algorithm implementation and offering a formal semantic-based language for representation of choreography. Collaborations definition using such a semantic-enriched choreography description language helps to generate semantic Web services from choreographies as well.

REFERENCES:

[1] T. Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*, Prentice Hall PTR, August 2005.

[2] F. Rosenberg, C. Enzi, A. Michlmayr, C. Platzer and S. Dustdar, "Integrating Quality of Service Aspects in Top-Down Process Development using WS-CDL and WS-BPEL", IEEE 2007.

[3] M. Papazoglou, "Service-oriented computing: concepts, characteristics and directions", In Proceeding of the Fourth International Conference on Web Information Systems Engineering, December 2003, pages 3-12.

[4] C. Peltz, "Web service orchestration and choreography", Computer, 2003.

[5] OASIS. "Web Service Business Execution Language 2.0".2006.URL:http://www.oasis-open.org/committees/tc_home.php?wg_ abbrev=wsbpe (Last accessed: June. 5, 2009).

[6] N.M. Josuttis, *SOA in practice*, O'REILLY, 2007.

[7] W3C. "Web Service Choreography Description Language (WS-CDL)", Nov. 2005. URL: http://www.W3.org/TR/ws-cdl-10/ (Last accessed: April. 19, 2009).

[8] M. Rani, A. Kumar and S. Batra, "Web Service Choreography Description Language(WS-CDL): Goals and Benefits",

Computer Science and Engineering Department, Thapar University, 2006.

[9] A. Barros, M. Dumas and P. Oaks, “A Critical Overview of the Web Services Choreography Description Language(WS-CDL)”, BPTrends, March 2007.

[10] J. Mendling and M. Hafner, "From WS-CDL Choreography to BPEL Process Orchestration", Vienna University of Economics and Business Administration, 2006.

[11] Kopp O, Leymann F. "Choreography Design UsingWS-BPEL.", Institute of Architecture of Application Systems, University of Stuttgart., IEEE., 2008.

[12] W3C, "Web Services Description Language (WSDL) Version 2.0", June. 2007. URL: *http://www.w3.org/TR/wsdl20-primer* (Last accessed: March. 17, 2009).

[13] Pi4 Technologies Foundation. *pi4soa*, 2007. URL: *sourceforge.net/projects/pi4soa/* (Last accessed: February. 10, 2009).

[14] H.P. Alesso and C.F. Smith, *Thinking On The Web*, John Wiley& Sons,ltd, 2006.

[15] W3C, "OWL Web Ontology Language Guide", February 2004.URL:http://www,w3.org/TR/2004/REC-owl-guide-20040210/ (Last accessed: November 10 2009).

[16] Stanford University, *Protégé, 2000*. URL: http://protege.stanford.edu/ (Last Accessed: November 4 2009).