Computer Standards & Interfaces 48 (2016) 49–70

Contents lists available at ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi

A metamodel-based definition of a conversion mechanism between SOAP and RESTful web services

Antonio Navarro *, Anayansi da Silva

Dpto. Ingeniería del Software e Inteligencia Artificial, Universidad Complutense de Madrid, Spain

ARTICLE INFO

Article history:
Received 10 April 2015
Received in revised form 28 January 2016
Accepted 28 March 2016
Available online 19 April 2016

Keywords:
SOA
WSDL
WADL
MDA
MDD

ABSTRACT

Nowadays there are several frameworks that permit the conversion between SOAP and RESTful web services. However, none of these frameworks defines a high-level characterization of the interchange process, hindering full understanding of this process. This paper provides a metamodel-based approach that formalizes the conversion between SOAP and RESTful web services, clarifying this process. This approach can be used for guiding the ranslation process in forthcoming conversion frameworks and the publication of services in IDEs. In order to characterize the conversion mechanism three MOF metamodels are defined: SOAP, RESTful and intermediate SOA metamodels. This intermediate metamodel is used as a bridge between the other two metamodels. QVT Relations transformation rules between SOAP, RESTful and SOA metamodels are defined for a formal characterization of the transformation process.

1. Introduction

Service-Oriented Architecture (SOA) is a successful architecture nowadays. This architecture focuses its attention on the business services provided by a system [1]. Depending on how the service is implemented and provided to its clients, SOA architecture can have different implementations. Thus, CORBA [2] can be an implementation of SOA. However, nowadays web services are the most common implementation [1].

At present there are two main approaches when implementing web services: SOAP web services [3] and RESTful web services [4].

In both cases a software function (a service) is invoked through the World Wide Web. However there is an important difference: in the case of RESTful web services the client uses a URL to invoke a remote process. Therefore, the client is responsible for the encoding of the information transferred and received (e.g. using XML [5] or JSON [6]), as well as for opening and closing the connection. In the case of SOAP web services a layer of middleware isolates clients from the data encoding/decoding process and from connection management, performing an invocation to a remote object written in the same programming language as the client.

In both cases the client is unaware of the service implementation language, which can be the same as or different from the client language. However, in the case of RESTful services, the client has to make a significant effort to invoke the service, while SOAP service invocation is much simpler. Nevertheless, the invocation of RESTful services is less resource- and time-consuming for both the client and the service provider than in the case of SOAP web services, where the middleware layer has to be deployed on both the client and the service provider sides.

Because SOA is intended for the integration of heterogeneous systems it is not unusual for both SOAP and RESTful web services to coexist in the same organization. In order to unify the approach, there are currently frameworks that make the conversion between SOAP and RESTful web services possible.

The main problem in the use of these frameworks is that they do not provide a high-level model that describes the conversion mechanisms, hindering understanding of the process and thus the maintenance of the converted web service.

This paper describes a high-level modeling for the conversion mechanism between SOAP and RESTful web services. The main idea is to provide an intermediate MOF (Meta Object Facility) metamodel [7] that abstracts the main concepts of an SOA service. Another MOF metamodel is provided for RESTful web services. Finally, a third metamodel that characterizes the Web Services Description Language (WSDL) 1.1 [8] representation of SOAP web services is provided. Then QVT (Query/View/Transformation) Relations transformation rules [9] are defined with their origin in the WSDL and RESTful metamodels and with the SOA intermediate metamodel as target, as well as between the SOA intermediate metamodel and the WSDL and RESTful

* Corresponding author at: Dpto. Ingeniería del Software e Inteligencia Artificial, Universidad Complutense de Madrid, C/ Profesor José García Santesmases, 9, 28040 Madrid, Spain.
E-mail addresses: anavarro@fdi.ucm.es (A. Navarro), adasilva@ucm.es (A. da Silva).

http://dx.doi.org/10.1016/j.csi.2016.03.004
0920-5489/

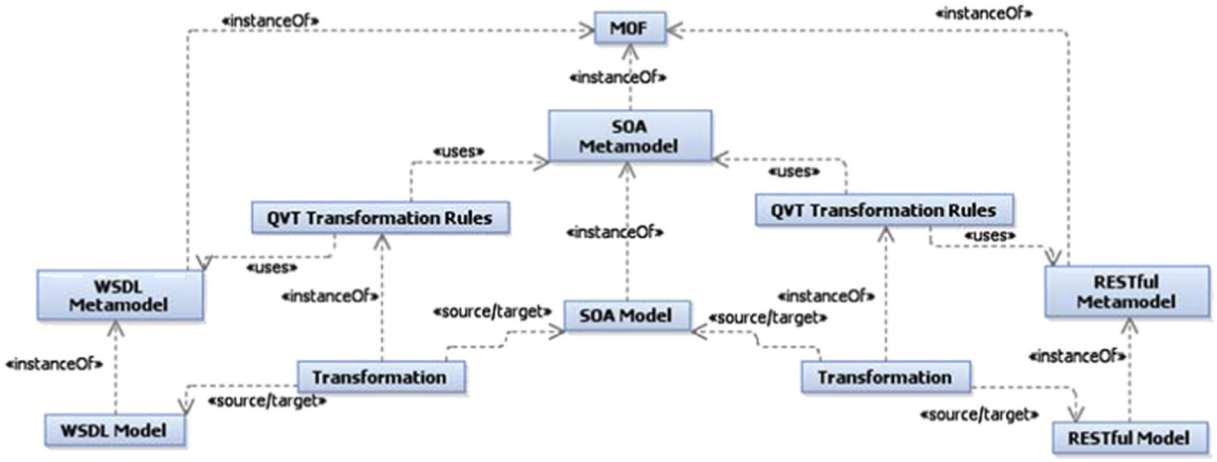

Fig. 1. The conversion mechanism provided in our approach.

metamodels. Thus a conversion mechanism is formally defined. Fig. 1 describes our approach.

Using this approach, conversion frameworks can be defined for the conversion between SOAP and RESTful services. In addition, the same service can be published both as a SOAP or a RESTful service, depending on client preferences. However, the approach obviates the services and publication details specific to each platform and framework.

Regarding compatibility between RESTful and SOAP services, RESTful web services were originally intended for the management of resources. Therefore, using HTTP methods the CRUD (Create, Read, Update and Delete) operations performed on a resource could be implemented: POST for create, GET for read, PUT for update, and DELETE for delete. Thus, from this point of view the translation from RESTful services to WSDL services is straightforward (it is only necessary to provide CRUD methods), but the opposite translation may not be so easily implemented.

For example, we could have a WSDL web service for the management of a cooling system. Thus, the service could have the following operations: start, stop, standby, heat, cool. How can the management of a cooling system be understood from a RESTful point of view? In our opinion this answer can have, at least, two answers, depending on the interpretation of RESTful services. If RESTful services are considered as a paradigm for entity (i.e. resource) management over the Internet, there is no way to translate the WSDL web service for the management of a cooling system. However, if RESTful services are only considered as a way for sending information over the Internet using HTTP methods, anything can be converted to a RESTful web service. Thus, the web service for the management of the cooling system could be implemented as a RESTful service. Only a servlet is needed (supposing a Java-based implementation [10]). This servlet receives a parameter in the request that identifies the operation to be performed on the cooling system (e.g. 1 for start, 2 for stop, 3 for standby, 4 for heat and 5 for cool), as well as the parameters required for each operation (if needed). All this information can be transmitted using the GET or the POST method, and the servlet is only used as a Web Service Broker [11] that provides HTTP access to a Java class. This paper follows the latter philosophy, like most Enterprise Service Buses and SOA gateways that translate WSDL web services into RESTful web services.

The contribution of the paper is twofold: formal metamodels for REST and WSDL web services are defined, and a conversion mechanism using an intermediate SOA metamodel and QVT transformation rules is provided. In this way developers of the middleware responsible for the translation between REST and WSDL web services have a formal tool for guiding its process. Thus, for example, as illustrated in Section 7, platform-independent test cases can be defined using the approach presented in this paper.

This paper gives the technical details of our work. Section 2 provides a brief introduction to MOF and QVT standards. Section 3 analyses related work. Section 4 describes the SOA metamodel. Section 5 describes the RESTful metamodel and the conversion rules between SOA and RESTful metamodels. Section 6 describes the WSDL metamodel and the conversion rules between the SOA and SOAP metamodels. Section 7 provides two real-life cases. Finally, Section 8 presents conclusions and future work.

2. MOF and QVT standards

2.1. MOF standard

OMG Meta-Object Facility, MOF, is a language intended for the definition of metamodels [7]. As their name depicts, metamodels are models of models, and they are necessary for defining models such as UML or the Entity-Relationship (ER) models. OMG defines a four layer hierarchy for dealing with the concepts related to metamodeling [12], but the people that make these models usually work in the M1 modeling layer. Within it lie the UML models that we build in order to characterize the design of software applications, or the ER models for the database schemas that we use. The UML models and the ER models are built according to a set of rules. For example, functions can be included in UML classes, but they cannot be included in ER entities. Why? Because the UML metamodel allows this feature, while the ER metamodel does not allow it. We are used to learn UML and ER metamodels using books or manuals, but a more formal method is needed to comprehensibly define them (and to be able to use them in transformations). MOF metamodels are used to characterize UML, ER or any other formalisms that need to be characterized. Thus, the M2 layer appears, within which the metamodels lie. But now, another question arises: how are metamodels in the M2 layer characterized? We could use natural language, but this is not a very formal method. We still need a higher layer that allows us to describe metamodels in the M2 layer. Thus, the M3 layer provides a formal way of defining metamodels, and the elements that lie within it are the meta-metamodels. Finally, in the M0 layer we have the instances of the M1 models.

Is another M4 layer necessary to characterize M3 meta-metamodels? And an M5 layer for characterizing M4 elements, and so on? The answer is no, because the meta-metamodels that lie within the M3 layer are usually reflective and they are able to describe themselves.