WSMeta: A Meta-Model for Web Services to Compare Service Interfaces

Marios Fokaefs and Eleni Stroulia
Department of Computing Science
University of Alberta
Edmonton, AB, Canada
{fokaefs,stroulia}@ualberta.ca

ABSTRACT

With the increasing adoption of the web-services stack of standards, service-oriented architecture has attracted substantial interest from the research community which has produced several languages and methods for describing and reasoning about services. These languages cover many concepts ranging from individual services and their code generation from specifications, service semantics, service compositions and networks, economics and business aspects around service ecosystems etc. However, this abundance of specification languages has also resulted in communication difficulties between stakeholders and hinders tasks such as service composition, discovery and maintenance. The presented work is a step towards the unification of the specifications and different aspects of service systems using Model-Driven Engineering. We propose a generic and abstract web service meta-model called WSMeta, which has the ability to describe both operation-centric web services (WS-*) and data-centric web services (REST) and can be used in tasks such as service evolution analysis and service systems maintenance.

Categories and Subject Descriptors

H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*web-based services*; D.2.2 [**Software Engineering**]: Design Tools and Techniques—*modules and interfaces, top-down programming*

General Terms

Design

Keywords

model-driven engineering, service-oriented architectures, WADL, WSDL, service comparison

1. INTRODUCTION

PCI 2013 September 19 - 21 2013, Thessaloniki, Greece

http://dx.doi.org/10.1145/2491845.2491860 ...$15.00.

In spite of being a relatively new technology, web services have been extensively studied by academia and industry alike. All this research effort has resulted in the development of a variety of specifications for web services and service systems. Oberle [11] gives a comprehensive overview of these specifications highlighting their purposes and their properties. One of the reasons that contributed to the development of multiple service-related specifications is the dichotomy between operation-centric and data-centric web services. Instances in the former category are usually specified using the Web Service Description Language (WSDL)[1], a W3C standard. WSDL files are XML-based and specify the complete interface of a web service in terms of operations and the data types they manipulate as input and output; essentially a WSDL specification serves (a) as a directory of the operations supported by the service, and (b) as a complete guideline for how clients can invoke the service. A variety of supporting tools have been developed to generate client- and server-side code (in a variety of programming languages) from WSDL specifications, or to reverse engineer WSDL specifications from code. Data-centric web services conform to the Representational State Transfer (REST) [7] architectural style, and they are usually specified in non-standard HTML or XHTML documents. In this informal specifications natural-language text is a dominant element, thus, lacking structure which in turn makes it difficult for the specification to be parsed automatically. For this reason, a new specification has been submitted recently to W3C for standardization called Web Application Description Language (WADL)[2]. WADLs are also XML-based, in the spirit of WSDL, and can be used for client code generation or client configuration. They describe data as resources and all the operations that can be invoked on these resources in the form of HTTP methods (GET, POST, PUT, DELETE).

Services may follow either the REST or the WS-* style and there is often a need for instances in either style to interoperate. To that end, there have been tools and techniques to translate service specification from one format to the other. Furthermore, version 2.0 of the WSDL standard has been designed so that it can be used to natively specify both REST and WS-* services (i.e., there is no need for intermediate transformations). However, in reality, both formats are used independently and this can raise certain challenges. For example, if one wants to create a service composition, one should be able to take full advantage of both REST and WS-* repositories alike. This is why, we propose a web ser-

[1]http://www.w3.org/TR/wsdl20/
[2]http://www.w3.org/Submission/wadl/

vice meta-model called *WSMeta*. This meta-model is simply an abstraction of the WADL and WSDL specifications. It takes into account all the common elements between the two and eventually specifies a service as a collection of operations with inputs and outputs. The meta-model also contains extension points as place-holders for other concepts of service systems such as clients and service compositions. WSMeta is based on Ecore, the Eclipse Modelling Framework (EMF)[3] meta-model. We also provide a set of Epsilon Transformation Language (ETL)[4] scripts to transform WSDL and WADL files into a WSMeta model and vice versa.

WSMeta can be primarily useful to service consumers. The task of service interface comparison is very common for service consumers. This task is performed, for example, when the consumed service has changed and the consumer has to compare the two versions of the interface in order to identify the changes and adapt to the new version. In our previous work [8] we have investigated the evolution of web services and we have identified certain types of changes and how they can affect client applications. The task of differencing web services requires the parsing of the service specification and comparison of its elements between different versions. However, certain properties of the service may be irrelevant to the task at hand, for instance, namespaces, unused operations or types and so on. In this case, a lightweight model to represent the interfaces, like WSMeta, can improve the accuracy and the efficiency of the comparison method. Another scenario where comparison is necessary is when the client decides to migrate the application to the service of another vendor. In this case, the client needs to compare the interfaces of the two services and identify the necessary changes to the client application so that it can be migrated to the new service. A generic web service meta-model, like WSMeta, can allow the client to compare web services, whose interfaces are specified in different standards, for example a WSDL service with a RESTful service. This way, the client's possibilities are greatly expanded. Finally, WSMeta can also be useful to service providers. The meta-model allows the provider to change the format of a service (from WSDL to REST or vice versa) or offer the same service in different formats, which will enable the provider to expand the clientèle and accommodate more applications. The model transformation that accompany WSMeta are specific to service providers for this particular use case.

The rest of the paper is outlined as follows. Section 2 provides an overview of other model-driven approaches that have been applied on Web Services. Section 3 describes WSMeta and provides details on how the meta-model was created and in Section 4 we discuss how the transformation can be applied on a simple service. Finally, Section 5 concludes this work and discusses some of our future plans.

2. RELATED WORK

2.1 Representing service systems

Ortiz and Hernandez [12] propose a model-driven approach to develop web services with extra-functional properties from a platform independent model (PIM). As a PIM, they use a UML profile extended with certain stereotypes for service components and extra-functional properties. Then, using ATL (Atlas Transformation Language) [5] transformation rules this model is transformed into four platform specific models, each serving a certain purpose:
(a) a web service meta-model in JAX-RPC [6], in order to decouple the properties from the service development process;
(b) an aspect-oriented meta-model, in order to decouple the properties from the implementation of the service;
(c) a policy meta-model in WS-Policy, in order to decouple the properties from the description of the service; and finally
(d) a SOAP tag meta-model, to make the services more flexible in the presence of optional extra-functional properties. The amalgamation of model-driven engineering, service component architecture, aspect-orientation and WS-Policy, as presented in the paper, is exemplified in a simple case study.

Jegadeesan and Balasubramaniam [10] present a service meta-model, which extends the UML Infrastructure Library. The motivation behind this work, is the inability of existing modelling techniques to capture all aspects of SOA and the evolution of web service standards. The proposed meta-model has five views each one capturing an aspect of SOA:
The *Service Definition View* captures ownership information as well as the nature of the service (composite, abstract etc.);
The *Service Capability View* captures the description of a service and its operations, QoS properties, constraints and exceptions;
The *Service Policy View* captures the non-functional constraints of a service;
The *Service Realization View* captures information about how a service is realized such as participants, like providers, consumers, aggregators and mediators, and how services can be composed.
The *Service Mediation View*, that handles compositions from a data or process perspective.

Treiber et al. [14] propose SEMF, a service evolution management framework. According to the authors, different stakeholders of a web service system that handle the various aspects or artifacts of the system can trigger various changes. The proposed framework handles all this data and integrates it by employing a Web Service Information Model and then correlates the various changes from the different stakeholders. The information model is implemented as Atom feeds, where the artifacts are linked seamlessly regardless of the underlying data model and the feeds can be queried efficiently using XQuery. The framework is also extendible to accept new data sources and the extensions are in the form of plugins which follow a specified interface.

Fensel and Bussler [6] present a Web Service Modelling Framework (WSMF) that focuses particularly on supporting e-commerce applications with multiple and heterogeneous partners. With respect to the service system implementation, the framework focuses on supporting service composition and mediation between the partners. According to the authors such a modelling framework should satisfy two fundamental properties that need to exist in an e-commerce system: *decoupling* of the system's parts and *mediation* between the various partners of the system. WMSF provides support for discovering, comparing and composing web services and handling heterogeneous data and business logics.

[3]http://www.eclipse.org/modeling/emf/
[4]http://www.eclipse.org/gmt/epsilon/doc/etl/
[5]http://www.eclipse.org/atl/
[6]http://java.net/projects/jax-rpc/

WSMF uses four elements to describe e-commerce systems. Ontologies are employed to provide domain and application-specific terminology needed by the system. A goal repository contains the various objectives of the partners and their specific pre- and post-conditions. The web services used in the system are characterized by their individual descriptions. Finally, mediators are defined for eliminating discrepancies between data structures, business logics, message-exchange protocols, dynamic service invocation and service composition.

Finally, Andrikopoulos et al. [3] follow a model-based approach to facilitate the management of service evolution. They propose three different descriptions for a service system: (a) the Abstract Service Definition (ASD) that describes the general concepts and their relations that are common to all web services; (b) the Service Schema Definition (SSD) that describes a particular web service; and (c) the Instance of Service Schema Definition (ISD) that describes the execution of a web service. In fact, the ASD is the meta-model of the SSD. As such, it contains attributes which are assigned values when the service is instantiated (i.e., in the ISD) and property domains from which a particular property will be selected when a SSD will be generated from the ASD. The ASD comprises of two sections, the public and the private and three layers, the structural, the behavioural and the regulatory. The public section refers to the public interface of the web service (i.e., the one that is exposed to potential clients) and the private section refers to the internal logic of the service. The structural layer contains the interface specific components of the service such as operations, messages and data types, the behavioural layer contains constraints and dependencies between the operations, as well as instructions on how the operations can be combined into a process and, finally, the regulatory layer contains the policies and the business rules of the service.

2.2 Task- or component-specific service models

Cao et al. [5] first recognize the need for a web service meta-model. According to the authors, the meta-model should be a platform-independent model (PIM) in order to be generic). This meta-model can later be transformed into a platform-specific model (PSM). In this work, the proposed tool-independent meta-model is realized using ER diagrams, while the tool-dependent model is specified using Generic Modelling Environment. This approach covers only interface specific elements.

Bordbar and Staikopoulos [4] propose a semi-automatic method to generate meta-models from XSD files. They use hyperModel, an Eclipse plug-in to generate XMI files from XSD. Next, using Poseidon for UML they generate a UML class diagram. According to the authors, the proposed methodology is especially useful for web services since WSDL files are practically XSD based. They demonstrate this ability by generating a WSDL meta-model from the WSDL XSD published by W3C and manually refining it to remove XSD or XML specific elements leaving behind only pure WSDL elements. They also propose a meta-model approach to support integration and interoperability between the different aspects and standards of web-service systems [13], based on UML-related research, where model transformations to allow for integration and interoperability are already established. First, they discuss the mechanisms from both UML and web services and compare them based on five criteria: containers, composite structures, messaging, links and mechanisms. Next, they propose a binding meta-model for web-service integration and interoperability and they explain how this meta-model can be applied to integrate BPEL compositions with WSDL specifications.

Ali and Babar [1] propose an extension for SoaML called AmbientSoaML to incorporate mobility properties, such as boundaries in mobile networks, in the SoaML meta-model, using Ambient Calculus. The problem is motivated and the method is explained in a simple example about voice calling and sms services in mobiles. Later, Ali et al. [2] present an Eclipse plug-in for developing web services in SoaML[7]. They use SoaML as their PIM. The meta-model is specified in Ecore and the users can create instances of it using EMF. The instances can be validated against the meta-model using the EMF Validation Framework. The plug-in also provides a SoaML Graphical Editor to create and edit SoaML instances. Finally, using ATL the SoaML models can be transformed into OSGi Declarative Service models.

Gebhart et al. [9] propose a set of properties for designing service systems based on SoaML. For properties that can be quantified OCL expressions are used to evaluate such properties. Given the values of these properties, a design can be characterized as bad and a better one can be suggested, as demonstrated in the paper.

2.3 Discussion

Although, as we see there have been proposed several model-driven engineering techniques to specify web services and service systems, they don't satisfy the entirety of our needs with respect to the particular task we try to address (i.e., comparison of heterogeneous web service specifications). In this work, the proposed meta-model is not intended to be used as a means to specify new services or service systems, but rather to be used as an intermediate format to be used in tasks that include comparison of service interfaces such as evolution analysis, service discovery and selection, etc.

While holistic approaches may cover many aspects of a service system such as partners, policies, business logic etc., these approaches may not provide sufficient details for these parts. For example, in SoaML, the interface of a web service is implemented as a single component. However, in a task like comparison, one needs to specify all the interface components (operations, messages and types) as individual components and analyse them separately.

An advantage of holistic approaches, especially the ones that are based on established modelling languages and techniques like UML, is that they provide opportunities for extension. Developers can use UML stereotypes, OCL rules and other components to extend a service model and include more details or other aspects. In this sense, one can extend a service model with a more fine-grained model for the service interface. Unfortunately, most of the approaches lack the tool support and detailed instructions on how to extend the models so that the extension communicates harmoniously with the core model.

3. WSMETA

Before embarking into the task of creating a web service meta-model, we first need to identify the elements that the

[7]http://www.omg.org/spec/SoaML/

two main formats that we want to integrate (WADL and WSDL) have in common. Therefore, a pre-processing step would be to analyse the two formats so that we can understand the purpose and the significance of each element. Then, we produce a mapping between the two formats. In some cases, elements in one format may correspond directly to elements in the other. In other cases, the elements of one format might be more complex and correspond to a collection of elements in the other format. Finally, we include in the meta-model the elements that are not common in the two formats, since, when translating from a specific format to WSMeta, we need to keep all the important information so that we can later perform the opposite translation. In this section, we will present these steps of the WSMeta construction in detail.

3.1 Analyzing WSDL

As we have already mentioned, WSDL specifies a service as a collection of operations that require an input and return an output or throw exceptions. A WSDL file consists of three sections: the schema that specifies the data types, the interface that specifies the operations, and the binding that specifies implementation-related aspects of the service. The schema is usually specified as an XML Schema Document (XSD), that can be included, imported or specified as an inline document inside the WSDL. It describes the data that the service handles and organizes it in complex or simple types. The operations are identified by a name and contain an input, an output, a fault corresponding to the input and a fault corresponding to the output. The number and types of these children are dictated by the message-exchange pattern of the operation (request-response, notification etc.). Inputs, outputs and faults have a reference to a type from the schema. The binding of the service specifies how the service is bound to a communication protocol (e.g. RPC/encoded, Document/literal etc.). In the binding, one can also specify what is the communication protocol over which an operation can be invoked. For example, we can have `wsoap:action` and the name of an operation for SOAP or `whttp:method` and an HTTP method for HTTP. Finally, it also contains the service's address.

3.2 Analyzing WADL

WADL describes a service as a set of *resources* on which operations can be applied as HTTP methods (GET, PUT, POST, DELETE). As such, the WADL can be split in two parts: the schema and the resources. The schema serves the same purpose as in WSDL, although in WADL it is referred to as a grammar. There is no restriction on what format the schema is specified, but XSD is a popular choice in WADL as well.

The resources part includes a URL that is the path to the resources. Each resource has a special attribute that defines the path to this particular resource. To find the actual address to a resource one has to concatenate its path attribute with the URL to the resources. Each resource contains the methods that are valid to be applied on it. The method is identified by an id and the name of an HTTP method. A method contains a request and multiple responses depending on the HTTP status (by default, WADL methods follow the request-response message exchange pattern [8]). For example, an HTTP 200 status code means success while 400 means failure. Responses and requests can be specified in two ways: either as representations or as a set of parameters. A representation specifies the type of the media, which can be, for example, XML or JSON, and the structure of the data, which can be specified as an XSD type or it can be a reference to an XSD type declared in the grammars section. A parameter has a name and a reference to an XSD type specified in the grammars. It also has other attributes such as whether it is required or a default value if necessary. Parameters can also be enumerations of options.

3.3 Mapping WADL and WSDL

In order to map the elements and their attributes between WADL and WSDL, we used VTracker, a tree-aligning algorithm, which we have already used to analyse the evolution of WSDL services [8]. VTracker has a special feature that allows us to create a cost function for two different file formats. Essentially, we ran VTracker on the WSDL and WADL schemas and we produced a mapping between the elements of the two standards with their corresponding distances. Because VTracker employs not only the structural properties of the elements but also their references (both incoming and outgoing), even in the case of elements that are different structurally, the algorithm can map them based on what other elements they use and by what other elements they are used. The synthesized cost function helped us map several elements. However, there were cases where not even references helped and for specific elements there was no clear mapping between the two formats. In these cases, we specified a mapping manually by changing the distance between the two elements to 0 in the cost function. VTracker was selected due to its bootstrapping ability to calculate a cost function by comparing a schema to itself or two different schemas. Although this cost function might require some manual tuning, the bootstrapping process performs most of the mapping automatically. In any case, the manual effort is only applied on the schema level and the cost function is used for instance level comparisons.

In order to evaluate the mapping, we created a reduced version of the Amazon EC2[9] service in WSDL, by including only one operation and its corresponding types. Then, we manually translated this service to WADL. The next step was to compare the two formats with VTracker by using various configurations of the algorithm. The first variability point we introduced concerned the employed cost function. We used three configurations: no special cost function, the synthesized cost function, and the manually improved cost function. The second variability point concerned the key elements. VTracker requires from the developer to specify what are the key elements in each file (i.e. the elements for which we are interested in mapping between the two compared files). Then the algorithm reports the mapping between the key elements. If an element is not considered key, it does not participate in the mapping process. First, we included XSD elements, WSDL operations and WADL methods as key elements. Next, we introduced some "noise" in the key elements by adding WSDL messages, WSDL ports and WADL resources. With the first key configuration, all cost functions found the proper mapping (between XSD elements and between WSDL operations and WADL methods). However, the distance reported between the elements (i.e., the confidence of the mapping) was better with the synthe-

[8] http://www.w3.org/2002/ws/cg/2/07/meps.html

[9] http://aws.amazon.com/ec2/

Table 1: Mapping of elements and their attributes between WADL and WSDL.

WADL	WSDL	WSMeta
application	definitions	IService
	@targetnamespace	@targetNamespace
grammars	types	Schema
resource	interface	Interface
resources@base+ resource@path	service::endpoint @address	@address
method	operation	Operation
@id	@name	@name
@name	binding::operation @whttp:method OR binding::operation @wsoap:action	@method
`http://www.w3.org/ns/wsdl/in-out` (fixed)	@pattern	@pattern
@href		@href
	@safe	@safe
	@style	@safe
request (response)	input(output)	Operation::request (response)
::param@type OR ::representation@element	@element	@request(@response)
param	xsd:element	IType
@type	@type	@name
param	xsd:simpleType	SimpleType
::option@value	::restriction::enumeration @value	Option@value

sized cost function than with no special cost function and even better with the manually improved cost function. With the second key configuration and with the default cost function and the synthesized cost function, VTracker got confused and mapped WADL methods with WSDL messages. This happened because WADL methods can reference XSD elements directly (if they use parameters instead of representations). WSDL operations can only access XSD elements only through messages (at least in version 1.1[10]). Therefore, it seems more natural to map such WADL methods with WSDL message since they refer to the same elements and their structure is not very different. However, with the manually improved cost function, where we explicitly specified that WSDL operations are to be mapped with WADL methods, the mapping was correct and with a strong confidence.

Table 1 shows the final mapping between WADL and WSDL. We use the delimiter '::' to denote a *parent-child* relationships, the delimiter '@' to denote an attribute and the delimiter ':' to qualify names with namespaces. The first column of each specification contains their high level elements, while the second column contains the attributes of these elements.

3.4 The Web Service Meta-Model

Figure 1 shows the proposed web service meta-model, WSMeta, produced by merging the WSDL and WADL models as explained previously. WSMeta was implemented using the EMF Ecore. An interesting point about the meta-model concerns the schema. Although a schema is considered a separate file, it is an essential part of a service since it specifies the nature and the structure of the data the service handles. As such it should be included in a web service meta-model and it can be useful in tasks such as service comparison.

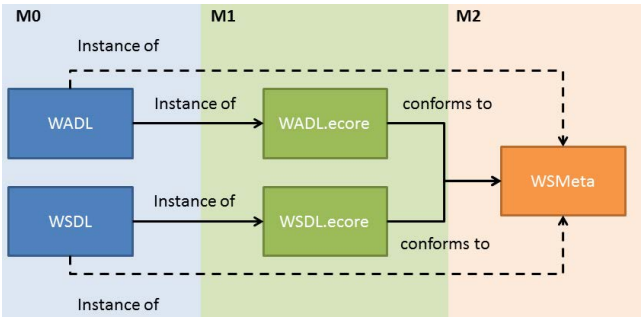


Figure 2: The WSMeta as a meta-model for specific services.

Figure 2 positions WSMeta with respect to specific models. As its name implies, WSMeta is a meta-model for the specific web service models, which have to conform to it. In order to get the model corresponding to a web service file or generate the service from the model, we use the Eclipse Web Standard Tools (WST)[11]. Unfortunately, WST only offers transformations from WSDL to its corresponding model and not from WADL services (as of the time this paper was being written).

Apart from the meta-model, we also contribute a set of transformations, implemented using the Epsilon Transformation Language (ETL). An example of such a transformation rule can be seen in Figure 3. These transformations are

[10]In version 1.2 of WSDL, the operations can directly access data types, therefore the mapping could have been easier in this case

[11]http://www.eclipse.org/webtools/wst/main.php

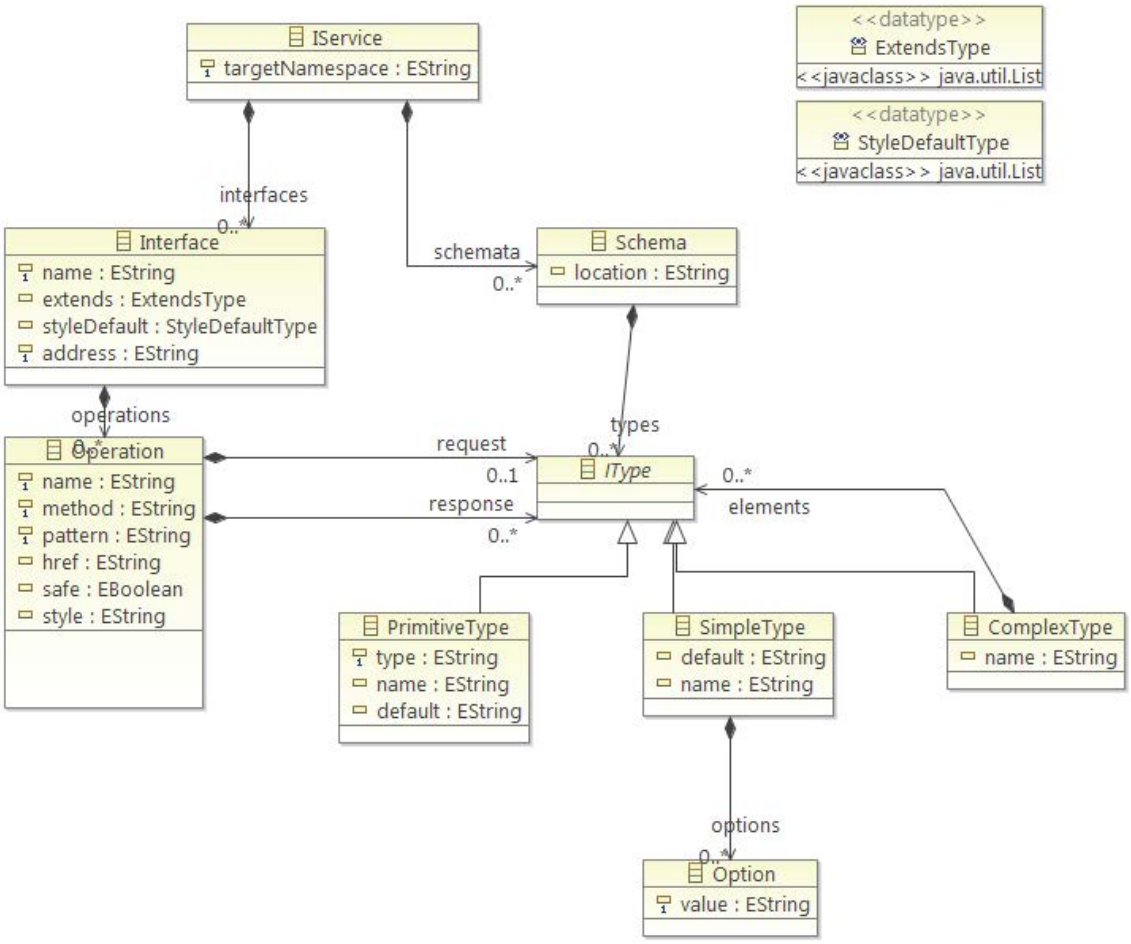


Figure 1: The Web Service meta-model.

```
rule CopyMethodType
    transform s : wadl!MethodType
    to t : wsmeta!Operation {


    t.href := s.href;

    t.name := s.id;

    t.method := s.name;

    t.pattern := "http://www.w3.org/ns/wsdl/in-out";

    t.input ::= s.request;

    t.output ::= s.response;
}
```

Figure 3: ETL rule to transform a WADL method to a WSMeta operation.

used to translate the specific models into the meta-model and vice versa. This way we can easily create meta-model instances from specific web services.

This would also allow us to translate the specification of a service from one standard to the other through WSMeta. The way we have defined our meta-model, i.e., by including not only the common elements between WSDL and WADL but also the unique elements from each standard, we ensure that such a transformation will be lossless. In other words, the produced interface will exactly correspond to the interface of the other standard. Of course, we cannot guarantee that the produced interface will be immediately functional and this is because, for a translated service to work, some low-level requirements need to be fulfilled first. For example, some REST services may require the existence of a persistent resource (e.g., database or file system). If we translate a WSDL interface to a WADL one, the service will not work until such a resource is provided. The purpose of the translation of services using WSMeta is to guide and facilitate the migration of services from one standard to the other. Manual changes to the service interface and the underlying system may still be necessary for the service system to function properly.

One thing that can hinder the transformation of web service interfaces is the schema. Whether the interface includes or imports an XML schema, it is still a different file, not included in the model. This means that for a single interface two models need to be created: one for the functional parts and one for the data types of the service. As a result, the transformation scripts for these two models need to be post-processed in order to resolve the references between the two models.

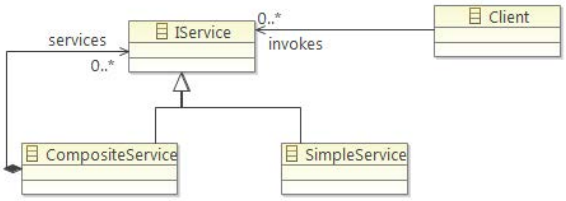


Figure 4: The client/composition extension of WSMeta.

Built with popular and state-of-the-art modelling techniques and tools, WSMeta can be easily extended to include more SOA concepts. For example, Figure 4 shows a WSMeta extension to describe service compositions and to include client applications.

4. AMAZON EC2 CASE STUDY

In this section, we present how we applied the transformation scripts and WSMeta on a case study we created from the Amazon EC2 service. We created a minimal interface with only two operations (`RegisterImage` and `DeregisterImage`) and information only about the abstract part of the interface, without the schema or the binding. We first created a WADL and a WSDL instance of the interface based on the WADL and WSDL schemata respectively. Then using the ETL transformations we generated the WSMeta instances from the other two instances. Finally, using the opposite transformation rules, we attempted to reconstruct the WADL and WSDL instances from the WSMeta instance. The results of this case study are shown in Figures 5 and 6 respectively for the two formats.

As it can be seen from the figures, the transformations from the specific model instance to the meta-model instance and vice versa produced the same model instance as the original in terms of structure. This demonstrates the ability of the meta-model and the transformation scripts to retain the basic structure of the service, which will allow more accurate comparisons. However, there are some discrepancies with respect to specific attributes of the service elements (e.g. names, types etc.), which shows that the transformations are not completely lossless and additional work needs to be done in this aspect. Furthermore, the transformation from both WADL and WSDL to WSMeta produced the same instance in terms of structure, which demonstrates the ability of our method to allow comparisons between heterogeneous service interfaces.

5. CONCLUSION AND FUTURE WORK

In this work, we presented *WSMeta*, a web-service meta-model. We applied VTracker on WSDL and WADL schemas in order to identify the common elements between the two standards. We used Ecore as our modelling language and ETL for the model transformations.

The proposed meta-model is not intended to describe web services but rather to be used as intermediate format to run tasks such as web service interface comparison. Since interfaces are the consumable part of a service, its critical parts are what it can do (operations) and on what data (types). By keeping only these elements, we abstract it and make it more lightweight to facilitate comparison algorithms and improve their efficiency and accuracy. Furthermore, by combining the elements of WSDL and WADL in our meta-model we can compare heterogeneous services specified in these standards. This widens the possibilities in web service discovery and selection. Finally, by using popular and state-of-the-art tools (Ecore, EMF, Epsilon) to create WSMeta and provide tooling support for it, we facilitate its further development and make it easier for other developers to provide custom extensions.

This work is a first step towards a web service meta-model for service interface comparison. Although some of its fundamental aspects have been implemented and presented in this work, we want to further develop and improve WSMeta. First, we plan to systematized the process of creating the meta-model. This can be achieved by model merging techniques. Using such techniques in combination with VTracker, we will no longer have to rely on manual effort to map elements between different formats. Furthermore, we will be able to study more service standards other than WSDL or WADL and automatically integrate them with WSMeta. Finally, Eclipse WST allows us to parse a WSDL service and generate the corresponding Ecore model. However, such a parser doesn't exist for WADL or other standards. It is necessary to create such a tool to allow us to apply the transformations and translate WADL services to WSMeta models.

6. ACKNOWLEDGMENTS

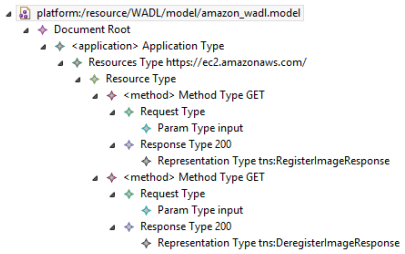
This work has been supported by NSERC, AITF and IBM.

7. REFERENCES

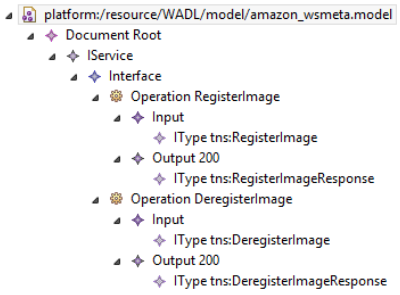
[1] N. Ali and M. A. Babar. Modeling service oriented architectures of mobile applications by extending soaml with ambients. In *Proceedings of the 2009 35th Euromicro Conference on Software Engineering and Advanced Applications (Patras, Greece, August 27-29, 2009)*, SEAA '09, pages 442–449, 2009.

[2] N. Ali, R. Nellipaiappan, R. Chandran, and M. A. Babar. Model driven support for the service oriented architecture modeling language. In *Proceedings of the 2nd International Workshop on Principles of Engineering Service-Oriented Systems (Cape Town, South Africa, May 1-2, 2010)*, PESOS '10, pages 8–14, 2010.

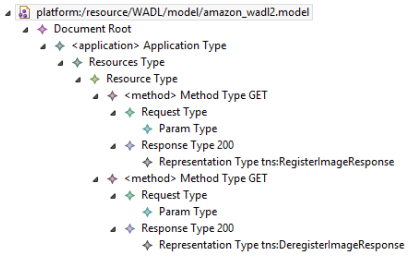
[3] V. Andrikopoulos, S. Benbernou, and M. P. Papazoglou. Managing the evolution of service specifications. In *Proceedings of the 20th international conference on Advanced Information Systems Engineering (Montpellier, France, June 16-20, 2008)*, CAiSE '08, pages 359–374, 2008.



(a) The original instance of the WADL Amazon service.

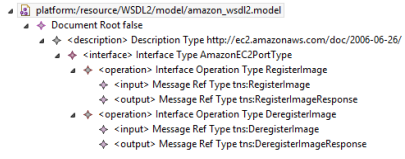


(b) The WSMeta Amazon instance after the WADL2WSMeta transformation.

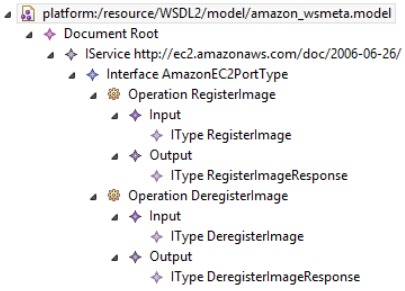


(c) The WADL Amazon instance after the WSMeta2WADL transformation.

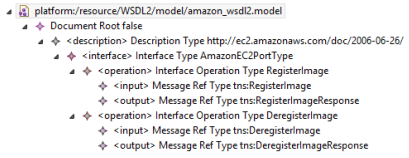
Figure 5: The WADL to WSMeta Amazon case study.



(a) The original instance of the WSDL Amazon service.



(b) The WSMeta Amazon instance after the WSDL2WSMeta transformation.



(c) The WSDL Amazon instance after the WSMeta2WSDL transformation.

Figure 6: The WSDL to WSMeta Amazon case study.

[4] B. Bordbar and A. Staikopoulos. Automated generation of metamodels forweb service languages. In *Proceedings of the 2nd European Workshop on Model Driven Architecture (Canterbury, UK, September 7-8, 2004)*, MDA'04, September 2004.

[5] F. Cao, B. R. Bryant, W. Zhao, C. C. Burt, R. R. Raje, A. M. Olson, and M. Auguston. A meta-modeling approach to web services. In *Proceedings of the IEEE International Conference on Web Services (San Diego, CA, USA, July 6-9, 2004)*, ICWS '04, pages 796–800, 2004.

[6] D. Fensel and C. Bussler. The web service modeling framework wsmf. *Electronic Commerce Research and Applications*, 1(2):113–137, 2002.

[7] R. T. Fielding. *REST: Architectural Styles and the Design of Network-based Software Architectures*. Doctoral dissertation, University of California, Irvine, 2000.

[8] M. Fokaefs, R. Mikhaiel, N. Tsantalis, E. Stroulia, and A. Lau. An empirical study on web service evolution. In *Proceedings of the 2011 IEEE International Conference on Web Services (Washington, DC, USA, July 4-9, 2011)*, pages 49–56, 2011.

[9] M. Gebhart, M. Baumgartner, S. Oehlert, M. Blersch, and S. Abeck. Evaluation of service designs based on soaml. In *Proceedings of the 2010 Fifth International Conference on Software Engineering Advances (Nice, France, August 22-27, 2010)*, ICSEA '10, pages 7–13, 2010.

[10] H. Jegadeesan and S. Balasubramaniam. An MOF2-based Services Metamodel. *Journal of Object Technology*, 7(8):71–96, 2008.

[11] D. Oberle. D1 report on landscapes of existing service description efforts. `http://www.w3.org/2005/Incubator/usdl/wiki/D1`, September 2011.

[12] G. Ortiz and J. Hernandez. A case study on integrating extra-functional properties in web service model-driven development. In *Proceedings of the Second International Conference on Internet and Web Applications and Services (Mauritius, May 13-19, 2007)*, pages 35–40, 2007.

[13] A. Staikopoulos and B. Bordbar. A comparative study of metamodel integration and interoperability in uml and web services. In *Proceedings of the First European conference on Model Driven Architecture: foundations and Applications (Nürnberg, Germany, November 7-10, 2005)*, pages 145–159, 2005.

[14] M. Treiber, H.-L. Truong, and S. Dustdar. Semf - service evolution management framework. In *Proceedings of the 2008 34th Euromicro Conference Software Engineering and Advanced Applications (Parma, Italy, September 3-5, 2008)*, SEAA '08, pages 329–336, 2008.