A Meta-Modeling Approach to Web Services

Fei Cao, Barrett R. Bryant, Wei Zhao, Carol C. Burt
University of Alabama at Birmingham
{caof, bryant, zhaow, cburt} @cis.uab.edu

Rajeev R. Raje, Andrew M. Olson
Indiana University-Purdue University-Indianapolis
{rraje, aolson}@cs.iupui.edu

Mikhail Auguston
Naval Postgraduate School
auguston@cs.nps.navy.mil

Abstract

Web Services (WS) technology is becoming pervasive in the development of distributed systems and is an appealing vehicle for service presentation and horizontal integration. On the other hand, Model Integrated Computing (MIC) offers a means of system integration in the vertical direction by using domain-specific modeling, and then synthesizing the software system from the high-level model using a model-specific generator. This paper presents a meta-modeling approach to WS to explore the application of MIC in WS development and its contribution.

1. Introduction

Web Services (WS) technology emerges as a Service Oriented Computing (SOC) ([8], [9]) paradigm to provide a platform-independent solution for system integration *horizontally*: WS is built upon open standard XML and HTML for service description and transportation, and software systems can be presented as WS so as to be exported and consumed by heterogeneous peers in the distributed environment.

For service description in Web Services Description Language (WSDL), though its XML-based representation is easy for machine processing using widely existent XML parsers, such specification is not straightforward for human comprehension, with service architecture lost in the pure textual form, and hand-crafting service description with WSDL is error-prone. To overcome this problem, there are tools on the horizon such as AXIS[1], and the Microsoft .Net framework that provide the capacity of automatically generating WSDL by parsing implementation code (such as Java and C#), and vice versa. However, WSDL represents the design level knowledge, and the process of generating WSDL from implementation is in contradiction to the general practice of software engineering, for which the design phase precedes the implementation phase. We believe generating WSDL from the design-level model directly offers an appropriate solution.

In this paper we present a meta-modeling approach to WS based on the principles of Model Integrated Computing (MIC) [5]. In MIC, meta-models can be used to define modeling language. Consequently, WS artifacts (WSDL) can be automatically generated from the WS model with generators. The Generic Modeling Environment (GME) [4] is a tool realizing MIC for creation of domain-specific models.

As is shown above, meta-modeling constitutes the corner stone In MIC. This paper is not intended to demonstrate the meta-modeling approach to all aspects of WS (e.g., discovery and orchestration) exhaustively, which is not possible because of the space limitation, but rather to focus on the elicitation of a tool-independent meta-model from WS requirements specifications, and then to map the too-independent meta-model to a tool-specific meta-model (here GME meta-model in particular), which is to be used throughout the main phases of MIC. This paper is organized as follows: Section 2 details the meta-modeling approach. Section 3 describes the related work. Section 4 draws the conclusion.

2. Meta-Modeling of WS

2.1 Meta-modeling WSDL using Entity Relationship (ER) representation

Object-Oriented class diagrams are often used to document software system architecture. The architecture of WSDL elements can be described as in Figure 1. Figure 2 gives the meta-model of WSDL (by removing the extension part enclosed with a dashed line), which is derived by representing the links (*association*, *generalization*, *dependency*) in the class

[1] http://ws.apache.org/axis/

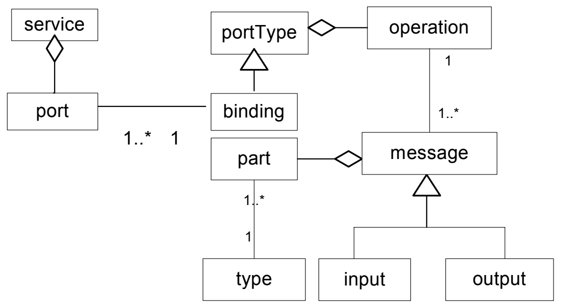

Figure 1: the Architecture of WS Description Elements

diagram in Figure 1 as a *relationship* in Figure 2, as well as representing those classes as an *entity* accordingly.

Note we ignore *type* in the meta-model of Figure 2, because we can put type directly as the attribute of the part element. Also note we will not annotate the attributes to the entities and relationships in the ER representation as the focus here is about the meta-model evolution; the attributes will be annotated in the GME meta-model as shown later.

The meta-model is represented by the ER representation [2] rather than by UML. The justification of using ER representation as an intermediate meta-model is as follows:
Different meta-modeling tools such as GME may adopt its own meta-model paradigm. Thus, there is a need for a tool-independent meta-model representation as an intermediate form for meta-model transformation, so that tool-dependent meta-models can be evolved into each other and used across different meta-modeling tools. This intermediate meta-model representation should be generic enough to describe a meta-meta-model, which resides at the top level (M3 level) of Model Driven Architecture (MDA)[2] metalevel stack [3]. The meta-meta-model used to define UML meta-models is described by the Meta Object Facility (MOF)[3], which is a set of constructs used to define meta-models. The MOF constructs include MOF class, MOF attributes, MOF association. These constructs literally constitute an ER representation (by using an Entity to represent a MOF class). Therefore, we believe ER representation is the right vehicle for representing intermediate meta-model. Moreover, meta-model using ER representation is easy to be described and serialized using XML [10]. This facilitates the meta-model exchange and processing using widely existent XML parsers.

When modeling a WSDL for real business domain services implemented with a specific technology, we use the *generalization* relationship to extend those WSDL elements in Figure 2 rather than embedding the business domain service information as attributes to those WSDL elements. This avoids obfuscation of business and technology domain structure (actually meta-models of business/technology domain applications) with WSDL elements. The business domain information applies a generalization relationship to the operation entity, and technology domain information applies a generalization relationship to the binding entity. To exemplify, Figure 3 is a simple banking domain service specification.

Figure 2 shows the ER-based meta-model of this banking service WSDL. As can be seen from the figure, a typical business domain service represented as WSDL involves the extension of ER elements, which is associated to almost all the elements of WSDL. Nevertheless, by using the ER-based meta-model, such extension still keeps the original WSDL meta-model as shown in Figure 2 without being restructured, which helps generating WSDL from models with consistency.

2.2 The Mapping from ER based Meta-model to Other Forms of Meta-model

In GME, the containment relationship is represented by using a *model* element (tagged with <<*model*>>), which, in contrast to an *atom* element (tagged with <<*atom*>>), can contain other modeling elements. Also the contained elements can be promoted as *ports* of the model to have direct connections with external modeling elements. GME uses a *root model* as an entry point of access to all the modeling elements. Also, the *relationship* of ER is represented in GME as a first-class modeling element, *connection* (tagged with <<*connection*>>), with a *connector* in the form of a dot to associate this relationship with two modeling elements (entities).

The mapping from the ER-based meta-model to the counterpart in GME is based on the relationships in the ER representation. Three cases are involved as is shown in Figure 4. For the sake of limited space, below we only describe the mapping rules for case 3, i.e., B is specialized from A. In this case, A is rendered by an abstract FCO (First Class Object, tagged with <<*FCO*>>, represents an abstract generalization of other modeling constructs), a modeling element to be used as an abstract interface in GME, and B is represented as an inherited class of that FCO. Note there are two special treatments here: firstly, for the input/output elements of Figure 2, they are only used to tag the *connection* (named either "input" or "output") between message entities and its interconnecting entities in GME; secondly, the generalization

[2] http://www.omg.org/mda/
[3] http://www.omg.org/cgi-bin/doc?formal/00-04-03

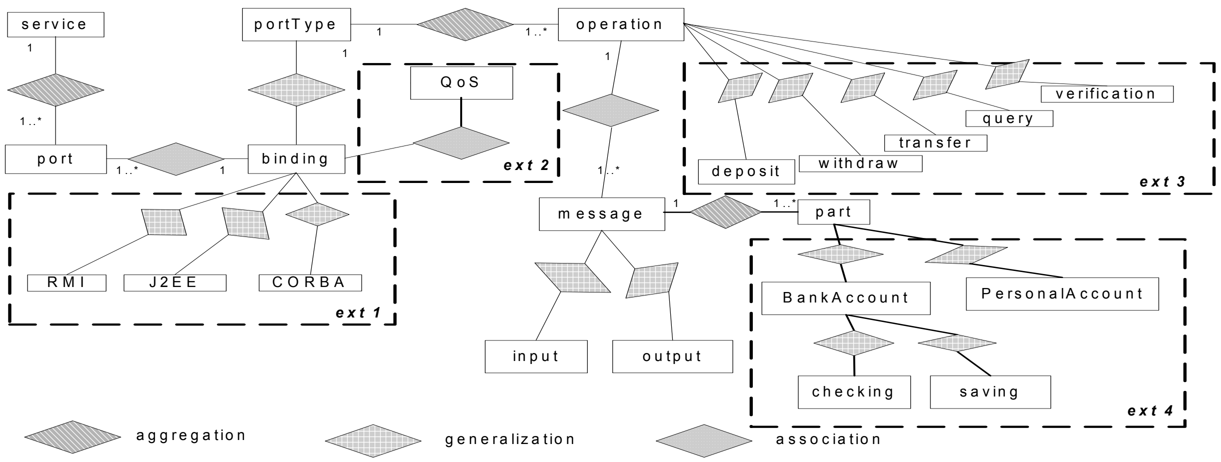

Figure 2: the ER-based Meta-model of Banking Service WSDL: the three parts enclosed with dashed line represent the extended part to the WSDL meta-model

A bank provides the service for users to set up accounts. Account information includes personal data including Name, SSN, phone number, address, and account data including Account Number, PIN, Transaction Record, Balance. There are two types of accounts: checking account and savings account.

For the bank side, it provides such services as: Account Verification, Account Query, Deposit, Withdraw, and Transfer.

The banking service implementation may use such technology as RMI, J2EE, and CORBA. Also it will enforce some Quality of Service (QoS) requirements such as Availability, Dependability, Capacity.

Figure 3: the Banking Domain Service Description

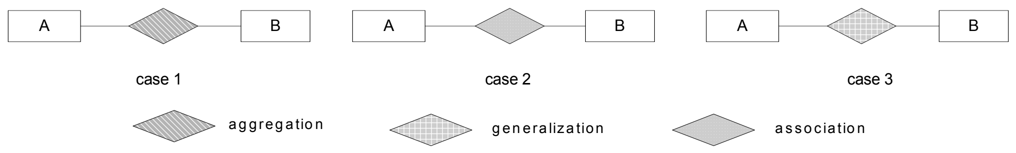

Figure 4: the Cases of Mapping from ER-based Meta-model to GME based Meta-model Based on the Relationship in ER Representation

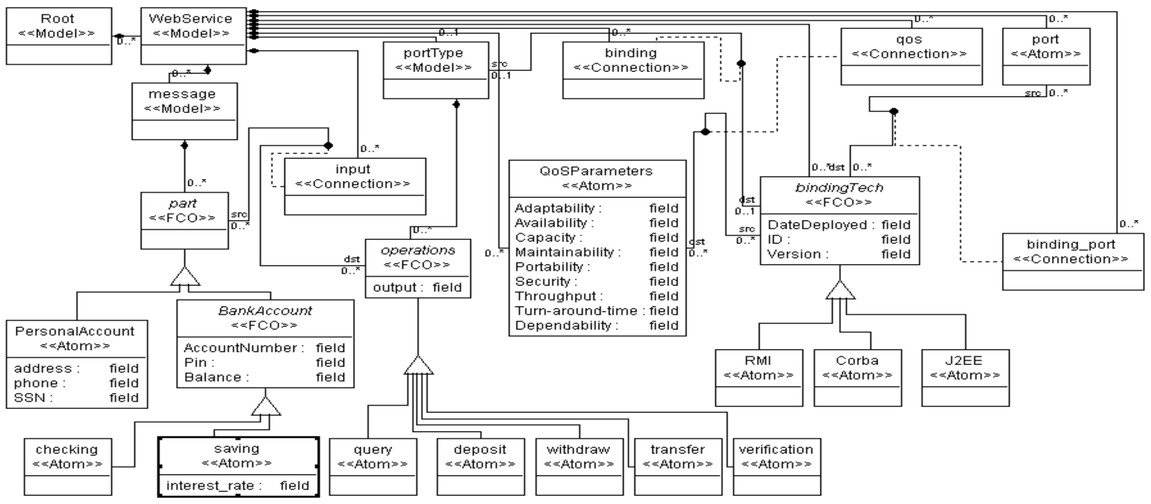

Figure 5: the Meta-model of Banking Domain WSDL in GME

relationship between binding and portType is actually treated as an association when modeling in GME, because the binding entity actually attaches values of the chosen protocol to the portType in WSDL rather than in the real sense of inheritance.

Figure 5 shows the meta-model created by mapping from the WSDL meta-model of the banking domain with ER representation to that in the GME strictly observing the above mapping rules. Note the model *WebService* corresponds to the service entity in Figure 2. The lower part of the models in Figure 5 are attributes for the related models to be instantiated in the modeling phase as described in the next section.

Based on this meta-model, a WS modeling environment can be constructed, and a generator based on this meta-model can be created to interpret WS models to generate WSDL. The WS modeling environment as well as generated WSDL is described in [1].

3. Related Work

In [6], MDA is used together with workflow technology for modeling and composing WS. But the authors do not provide a guideline as to how to create the meta-models. Also the mapping from PIM to PSM is not detailed. In contrast, we focus on meta-modeling WSDL only, while the meta-modeling approach is more complete and general. In [7], an MDA approach is used for BPEL code generation from a UML design. This approach uses XMI[4] processing technology for UML model exchange. Comparatively the XML representation for the ER model is much simplified and easy to process in our approach. Code generation in [7] is based on the UML profile mapping, which is not as flexible as a generator-based approach in our case.

4. Conclusion

WS domain-specific modeling environment provides a user-friendly environment to build WS with underlying WS-specific details abstracted. This paper presents a general meta-modeling approach to WS, which is used for the construction of WS domain-specific modeling environment In particular, we showed the merits of using the ER representation as an intermediate form for deriving and evolving meta-models to avoid the ad-hoc nature of constructing meta-models, which is an problem that is often not addressed (such as in [1]), particularly in constructing large-scale meta-models.

[4] XML Metadata Interchange - http://www.omg.org/technology/documents/formal/xmi.htm

Meta-modeling of WSDL is a static structural modeling. Future work will include meta-modeling of WS behavior such as WS orchestration.

5. Acknowledgements

This research is supported by the U. S. Office of Naval Research under the award number N00014-01-1-0746.

6. References

[1] Cao, F., Bryant, B. R., Burt, C. C., Gray, J. G., Raje, R. R., Olson, A. M., Auguston, M., "Modeling Web Services: Toward System Integration in UniFrame," *Proc. 7th World Conference on Integrated Design and Process Technology (IDPT'03),* December, 2003, pp. 83-91.

[2] Chen, P. P., "The Entity-Relationship Model: toward a Unified View of Data," *ACM Trans. Database Systems*, March, 1976, pp. 9-36.

[3] Frankel, D. S., *Model Driven Architecture: Applying MDA to Enterprise Computing*, Wiley, 2003.

[4] *GME 2000 User's Manual, Version 2.0*, ISIS, Vanderbilt University. 2001.

[5] Lédeczi, Á., Bakay, A., Maroti, M., Volgyesi, P., Nordstrom, G., Sprinkle, J. and. Karsai, G., "Composing Domain-Specific Design Environments," *IEEE Computer*, November, 2001, pp. 44-51.

[6] Lopes, D., Hammoudi, S., "Web Service in the Context of MDA," *Proc. International Conference on Web Services (ICWS'03)*, June, 2003, pp 424-427.

[7] Mantell, K., "From UML to BPEL: Model Driven Architecture in a Web Services World," http://www-106.ibm.com/developerworks/ webservices/library/ws-uml2bpel/.

[8] Olson, A. M., Raje , R. R., Bryant, B. R., Burt, C. C., Auguston, M., "UniFrame-A Unified Framework For Developing Service-oriented, Component-based, Distributed Software Systems," to appear in *Service-Oriented Software System Engineering: Challenges and Practices*, ed. Zoran Stojanovic and Ajantha Dahanayake, 2004.

[9] Papazoglou, M. P., Georgakopoulos, D., "Service-Oriented Computing," *Commun. ACM*, October, 2003, pp. 25-28.

[10] Zhou, S., Xu., C., Wu, H., Zhang, J., Lin, Y., Wang, J., Gray, J. G., Bryant, B. R., "E-R Modeler: A Database Modeling Toolkit for Eclipse," *Proc. Annual ACM Southeast Conference*, April, 2004, pp.160-165.