Marshaling and Unmarshaling Models Using the Entity-Relationship Model*

Fei Cao, Barrett R. Bryant,
Wei Zhao, Carol C. Burt
Department of Computer and Information Sciences
University of Alabama at Birmingham
1300 University Boulevard, Birmingham, AL 35294, USA
{caof, bryant, zhaow, cburt} @cis.uab.edu

Rajeev R. Raje, Andrew M. Olson
Department of Computer and Information Science
Indiana University-Purdue University-Indianapolis
723 W. Michigan Street SL 280, Indianapolis, IN 46202, USA
{rraje, aolson}@cs.iupui.edu

Mikhail Auguston
Computer Science Department
Naval Postgraduate School
1 University Circle, Monterey, CA 93943, USA
maugusto@nps.edu

ABSTRACT
Software systems are usually designed and documented with the aid of visual modeling notations. Visual modeling notations keep evolving over the years in tandem with visual modeling tools, and the tight binding in between impedes the exchanging of modeling assets, which causes a spatial isolation of the models. Another problem with legacy software models is that they are isolated temporally in the early phases of the software engineering life cycle without reaching out to the later phases. This paper presents an approach for breaking both spatial and temporal isolation of software models by marshaling and unmarshaling models using the Entity-Relationship (ER) model, thus providing a promising way for evolving model-driven software development.

Categories and Subject Descriptors
D.2.2 [**Software Engineering**]: Design Tools and Techniques

Keywords
Marshaling and unmarshaling models, Modeling and meta-modeling, Entity-Relationship model

1. INTRODUCTION
Software systems are usually designed and documented with the aid of visual modeling notations. Visual modeling notations keep evolving over the years in tandem with visual modeling tools, and the tight binding in between impedes the exchanging of modeling assets. Above all UML[1] stands out as the *de facto* standard modeling language. But other non-UML based modeling notations abound as evidenced in such publications as JVLC[2]. Meanwhile, a lot of work has been done to converge the diagram notations in the new version of modeling notations, as is mentioned in the recent interview with Keith Short[3]. But to converge all the legacy software modeling assets by reengineering into new generation notations and totally discarding old legacy modeling notations is not only time-consuming, but also not cost-effective. Depending on different usage scenarios, there is a need for marshaling models across different modeling facilities to take advantages of the leverages provided by existent modeling facilities.

The term *Marshaling* comes from the distributed computing area where heterogenous data types are always translated into some common data type over the network so as to be consumed at the other end of the distributed environment, where the common data type is *unmarshaled* again into another environment-specific data type. Here we use the ER model [2] to represent the "common data type", i.e., the intermediate model when exchanging and evolving models. The rationales are as follows:

- **Sufficiency.** Even though UML is widely adopted in software modeling, which seems to justify the use of UML as a common model for exchanging model assets across modeling facilities, UML is not convenient for model serialization, thus not fit for modeling asset exchange and evolution. In fact, the object diagram [1], for which UML is used to capture and store the snapshot of software system state, is represented virtually in an Entity (object) and Relationship (links) model. Moreover, the UML modeling language has its roots in the ER model, and the latter is already widely used as the foundation for CASE tools in software engineering and repository systems in databases[4].
- **Necessity.** Not only models, but also meta-models are in need of exchanging and evolution; the justification for the latter is obviously the same as the former. Therefore, the intermediate model should be

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

SAC'05, March 13-17, 2005, Santa Fe, New Mexico, USA.

[1] Unified Modeling Language-http://www.omg.org/uml
[2] Journal of Visual Languages and Computing-http://www.elsevier.com/locate/jvlc
[3] Interview with Keith Short, http://www.theserverside.net/talks/library.tss#KeithShort.
[4] http://bit.csc.lsu.edu/~chen/chen.html

expressive enough to be at the meta-meta model level in the meta-level stack [3]. The meta-meta-model is described by the Meta Object Facility (MOF)[5], which is a set of constructs used to define meta-models. The MOF constructs are the MOF class, the MOF attributes and the MOF association. These constructs correspond to an ER representation (by using an Entity to represent a MOF class), which indicates that the ER representation is semantically equivalent to MOF fundamentally. Therefore, we believe the ER representation is the right vehicle to play the *dual roles* of marshaling both models and meta-models to break the spatial isolation of software models. Also, other non-UML based languages, even though not as popular, are abundantly present, for which UML is not an omnipotent cure.

Recent years have seen the emergence of the Model Integrated Computing (MIC) [7] paradigm, which moves a step further to break the isolation of models from implementation and the subsequent phases in the software engineering life cycle. In MIC, a meta-model is created to define a model construction language, and a generator is also to be created based on the meta-model to synthesize the constructed models by traversing the model tree. In this way, a model can be more accurately interpreted for code generation than the direct mapping-based approach such as using profiler or stereotype in Rational Rose [3]. Toward that end, this paper presents an approach for marshaling software models to ER models, which, by taking advantage of the dual roles of ER models, are unmarshaled into an environment-specific meta-model to be integrated into MIC. Consequently, not only the spatial isolation, but also the temporal isolation of software models can be broken.

This paper is organized as follows: Section 2 briefly provides an overall picture of this approach. Section 3 uses Web Services (WS) [5] modeling as a proof-of-concept example to illustrate the whole process. Section 4 describes the related work. We conclude in section 5 with a brief description of future work included.

2. OVERVIEW

Figure 1 shows the process of marshaling and unmarshaling models. Generic Modeling Environment (GME) [4] is the tool for MIC paradigm, and we use it as the targeted tool environment for describing destination meta-models, whereupon the domain-specific modeling environment can be constructed. Through the process flow as is directed by the arrows, meta-models can be elicited from models with an automatable process as opposed to traditional practice, for which the meta-model is constructed in an error-prone, ad-hoc way. Consequently, models of legacy systems can be evolved toward the MIC paradigm for model-driven software development.

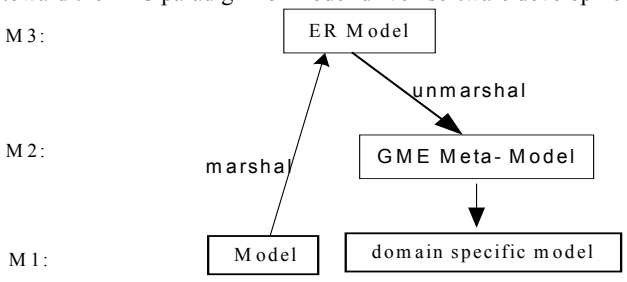

Figure 1. Marshaling and unmarshaling models

[5] Meta-Object Facility - http://www.omg.org/technology/documents/formal/mof.htm

3. THE APPROACH

3.1 A Web Services Modeling Example

Modeling Web Services (WS) is a promising way for service description and orchestration at a higher level. As the scope of this paper is about marshaling and unmarshaling models, the elicitation of models from requirements is skipped here.

One of the characteristics of a meta-model is that it treats not only the models, but also the inter-relationships among models as first-class entities. We derive meta-models by abstracting models and their inter-relationships. Therefore, for the models, even though they are represented as UML diagrams here as the starting point of the marshaling/unmarshaling process, they will not compromise the generality of the approach as is described in the remainder of the paper. To be specific, our approach of marshaling and unmarshaling WS models consists of two steps:

1) Marshal models by converting the OO class diagram to an ER-based meta-model, for which the *relationship* corresponds to *aggregation*, *association*, *generalization*, and *dependency*, while the *entity* corresponds to *class*.
2) Unmarshal models by mapping the ER-based meta-model to the tool-specific (here GME in particular) meta-model to create a WS modeling environment.

The UML class diagram of WSDL elements is shown in Figure 2.

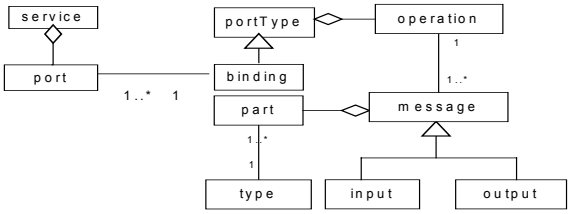

Figure 2. The architecture of WS description elements

The WS ***message***s, which are either ***input*** or ***output*** messages, are composed of ***part***s, each of which corresponds to a specific data ***type***. The ***portType*** is an abstract WS interface definition, where each contained element, i.e., the ***operation***, defines an abstract method signature. The operation uses messages as its parameters. ***Binding*** represents an instantiation to the abstract ***portType*** with concrete protocol and data type. ***Service*** is a collection of ***port***s, denoting a deployment of a binding at a specific network location.

3.2 Marshaling the WSDL Model

Figure 3 gives the meta-model of WSDL in ER form (without considering the extension part enclosed with the dashed lines), which is derived by representing the links (*association*, *generalization*, *dependency*) in the class diagram in Figure 2 as a *relationship* in Figure 3, as well as representing those classes as an *entity* accordingly. Note we ignore *type* in the meta-model of Figure 3, 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 model marshaling and unmarshaling; the attributes will be annotated in the GME meta-model as shown later.

When modeling WSDL for real business domain services implemented with specific technology, we use the *generalization* relationship to extend those WSDL elements in Figure 3 rather than embedding the business domain service information as attributes to those WSDL elements. This avoids obfuscation of business and

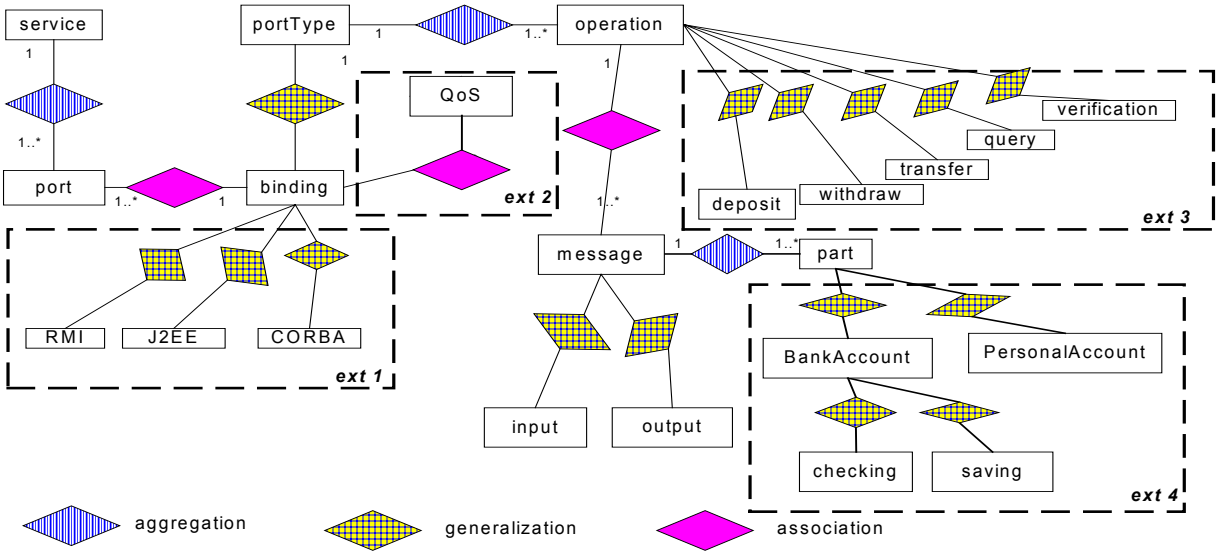

Figure 3. The ER-based meta-model of banking Service WSDL: the three parts enclosed with dashed line represent the extended part to the WSDL meta-model.

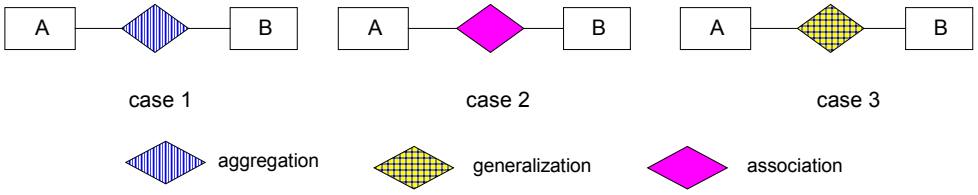

Figure 4. The cases of mapping from ER-based Meta-model to GME-based meta-model based on the relationship in ER representation.

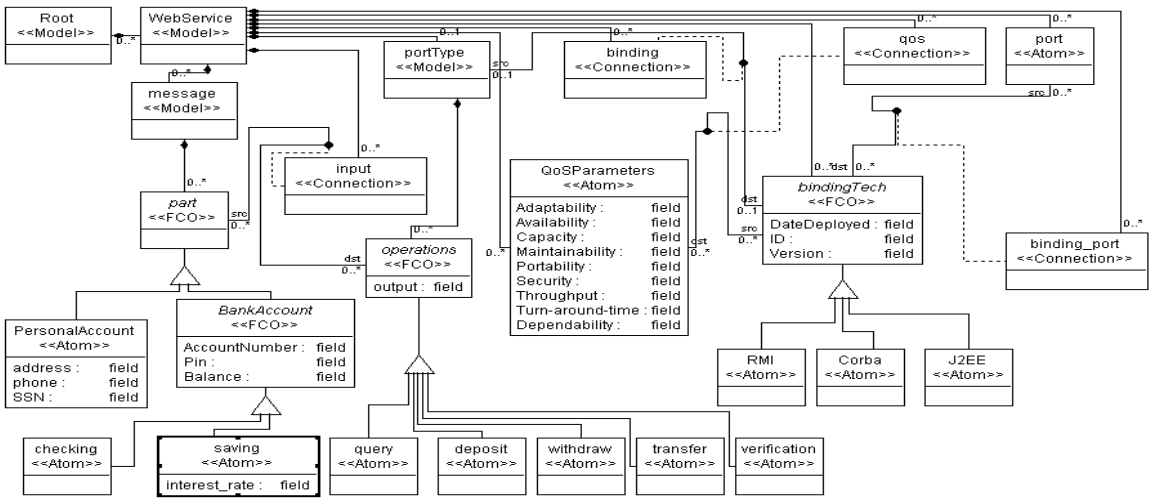

Figure 5. The meta-model of banking domain WSDL in GME

technology domain structure (meta-models of business/technology domain applications) with WSDL elements, and provides a separation of concerns toward domain-specific model refinement. 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, below is a simple banking domain service specification:

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 shows the ER-based meta-model of this banking service WSDL (including those parts enclosed by dashed line). The elicitation of models from natural language requirements is beyond the scope of this paper. 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 3 without being restructured.

3.3 Unmarshaling the WSDL 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:

1) A contains B

In this case, A can be modeled as a *model* element in GME containing B.

2) B is associated to A

In this case, a *connection* can be added to be associated with the A and B representations in GME. The connection element can be named with respect to A's or B's properties as a kind of tag, e.g., the tag can be named as the combination of both A's name and B's name. Note when the situation as described in case 3 applies, then this tag should be named as in case 3.

3) 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 to that FCO. Note there are two special treatments here: firstly, for the input/output elements of Figure 3, 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 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. The model WebService corresponds to the service entity in Figure 3. The boxed part of the models in Figure 5 are attributes for the related models to be instantiated in the modeling phase, described in the next section.

3.4 The Domain Specific Modeling Environment

After a meta-model is derived by marshaling and unmarshaling models, a domain specific modeling environment (which is also a crucial part of MIC) can be created based upon the meta-model. To complete the description of the model evolution process shown in Figure 1, Figure 6 shows the screenshot of the banking-domain WS modeling environment based on the meta-model illustrated in Figure 5. The lower-left corner provides the modeling elements that can be dragged and dropped in the upper-left pane for constructing a banking service model. The names of the models in the lower-left pane represent the meta-model names (*kind names*); when those models are dragged to the above pane, the model name can be changed to reflect the meaning of the model in the domain-specific context, which we call a *context name*. Furthermore, the domain-specific model can be traversed and interpreted in terms of code generation using the GME Builder Object Network (BON) framework [4].

4. RELATED WORK

The ER model, because of its powerful modeling capacity, can be used as an intermediate form for model-to-model and meta-model-to-meta-model exchange. Because of the dual role that the ER model can play, it is treated as an intermediate form for model-to-meta-model elicitation, which is the theme of this paper. This idea is very similar to grammar inference [6], where a grammar can be inferred from language examples. But the two approaches are applied at different abstraction levels. XMI[6] provides a standard mapping from MOF-based nodels to XML, which can be exchanged between software applications and tools. In comparison, ER-based model marshaling and unmarshaling represents a design-level approach for evolving design assets, without being restricted to low-level data representation specifics. Also, note that the XMI-based approach uses top-down mapping, while the ER-based approach uses bottom-up mapping as is illustrated in Figure 1, which offers a means for meta-model recovery for evolving legacy software models into Model Integrated Computing.

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

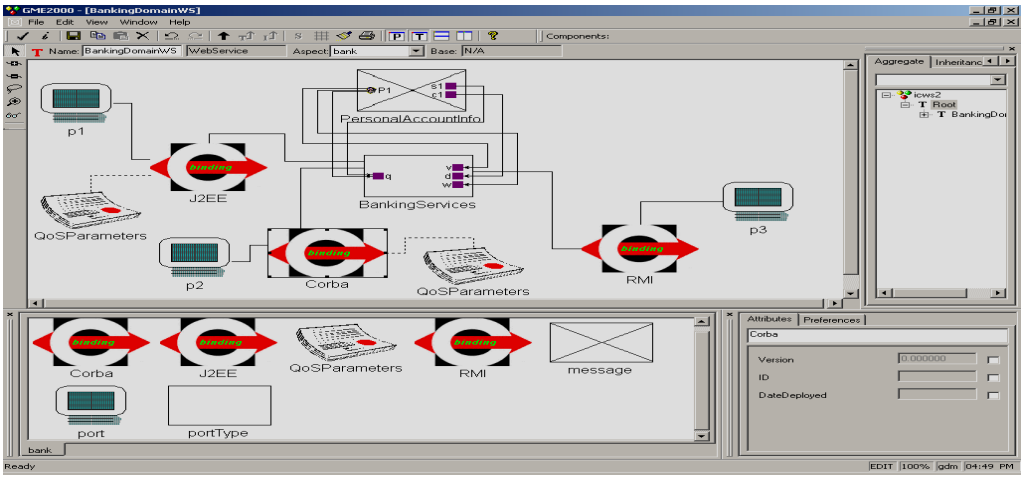

Figure 6. The banking domain-specific WS modeling environment.

Model Driven Architecture (MDA)[7] is about mapping Platform Independent Models (PIM) to Platform Specific Models (PSM) for engineering legacy software systems so as to be integrated into new platform. However, the core part of mapping technology for MDA is either ad-hoc or pre-mature before MDA can be fully adopted in industry. ER-based model marshaling and unmarshaling offers a potential solution to address this problem systematically. It has been observed that ER representation has been adopted in defining Knowledge Discovery Meta-Model (KDM)[8] and Ontology Definition Meta-Model (ODM)[9] in OMG, which underscores the role that ER plays for model marshaling and unmarhaling.

5. CONCLUSION AND FUTURE WORK

Legacy software models are widely existent and heterogeneous in their own graph syntax, and there are two types of isolation in its application: Spatially, models are isolated from being exchangeable over software applications and tools; Temporally, models are isolated in the early phases of the software engineering life cycle. These two types of isolation status of software models restrict their usability and capacity. Toward that end, a model marshaling and unmarshaling approach is presented based on the ER model, a simple, yet powerful modeling notation. This approach offers a promising way to break not only spatial isolations, but also temporal isolation by evolving legacy software models toward MIC for fully exploiting models throughout the software engineering life cycle. In particular, this paper uses a WS modeling example to illustrate an automatable process on how legacy software models can be migrated toward a MIC-oriented environment.

To ultimately automate the marshalling and unmarshaling process, future work will involve representing various models as well as ER models in the form of proper XML specifications, whereupon the automation process can be applied by XML transformation technology such as XSLT[10]. The ER model is easy to be represented in XML because of its simple structure. An Eclipse-based ER modeling tool such as [8] that can generate XML specifications from ER models will be helpful in this regard. The models in GME can be exported and imported as XML. Therefore, an XML specification for an ER-model can be directly transformed to the expected XML specification for destination meta-models and loaded into GME consequently. Note that the simple structure of ER models does not require an XMI–based data representation. Moreover, such existent tool as GME does not use XMI for model serialization and deserialization, for which a simpler and more flexible XML schema is desired for marshaling and unmarshaling models.

6. REFERENCES

[1] G. Booch, J. Rumbaugh, I. Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley, 1999.

[2] P. P. Chen. The Entity-Relationship Model: Toward a Unified View of Data. *ACM Trans. Database Systems*, 1(1), 1976, 9-36.

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

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

[5] S. Graham, S. Simeonov, T. Boubez, D. Davis, G. Daniels, Y. Nakamura, R. Neyam. *Building Web Services with Java*. SAMS, 2002.

[6] C. de la Higuera. Current Trends in Grammatical Inference. In *Proc. Joint IAPR Int. Workshops SSPR & SPR 2000*, 2001, 28-31.

[7] Á. Lédeczi, A. Bakay, M. Maroti, P. Volgyesi, G. Nordstrom, J. Sprinkle, and G. Karsai. Composing Domain-Specific Design Environments. *IEEE Computer*, 34(11), 2001, 44-51.

[8] S. Zhou, C. Xu, H. Wu, J. Zhang, Y. Lin, J. Wang, J. Gray, B. R. Bryant. E-R Modeler: A Database Modeling Toolkit for Eclipse. In *Proc. 42th ACM Southeast Conf.*, 2004, 160-165.

[7] http://www.omg.org/mda/

[8] http://www.omg.org/cgi-bin/doc?lt/2003-11-4

[9] http://codip.grci.com/odm/draft/submission_text/ODMPrelimSubAug04R1.pdf

[10] http://www.w3.org/TR/xslt