A VISUAL DATA MAPPING TOOL FOR SOFTWARE COMPONENTS
INTERACTIONS IN SERVICE-ORIENTED ARCHITECTURES

Marco Brambilla, Stefano Ceri, Sara Comai, Christina Tziviskou
Dipartimento di Elettronica e Informazione, Politecnico di Milano
V. Ponzio 34/5, 20133 Milano
Italy
mbrambil | ceri | comai | tzivisko@elet.polimi.it

ABSTRACT
With the spread of Service-Oriented Architectures, software components are exposed as Web services whose orchestration is gaining more and more importance. Several CASE tools and development frameworks have been put in place for representing conversations and composition without requiring designers to bother about implementation issues. However, as data sources for the exposed software components are arbitrary, data management problems within Web services interactions are often underestimated. This work addresses the topic of data mapping between XML contents of Web service messages and relational databases, which typically stores such information at the various peers. We propose a high-level approach to this problem, by offering (a) a very simple and straightforward visual transformation model as a software layer that enables the communication between peers based on various data formats, and (b) a software tool that allows the designer to draw the transformation diagrams between the message contents and the database structure. We provide the description of an algorithm for the automatic translation of visual mappings to XSLT rules, and the approach application within the WebML modeling framework for Web services design.

KEY WORDS
Web services, Data marshalling, XSL

1. Introduction
Current trends in software development systems move towards the spread of Service-Oriented Architectures for the implementation of inter- and intra-enterprise applications based on the reuse of software components. Web Services are used to expose the functionality of a software component (or a part of it) and make it available on the Web through standard technologies. They allow interactions among software agents independently of the programming language, the implementation platform, and the operating system of the underlying application, based on the exchange of well defined XML-based WSDL messages.

One of the most critical issues in this scenario is the problem of data mapping and marshalling between the various interacting software components called peers. Indeed, they may exploit and expose different data formats and data schemas. Therefore, the documents sent and received through Web service interactions need to be converted between formats and naming conventions of the various local systems.

For Web services interaction implementation, some existing design framework can be adopted by developers. For example, some high-level Web modelling languages (e.g., WebML [1, 2, 3] and others [4, 5, 6, 7]) have been extended to support Web services interaction, thus providing visual design languages for representing Web services interaction. Some CASE tools (WebRatio [8], BEA WebLogic Workshop [9]), at different levels of abstraction, provide several facilities for designing and automatically deploying Web services infrastructures.

However, at our knowledge there is no tool or approach tailored to data marshalling and mapping for Web services. Hence, the designer needs to work at a much lower level for specifying and implementing data mapping between the interacting systems, or to adapt general-purpose and complex high-level approaches.

The contribution of this paper addresses this problem by offering a simple visual transformation model as a software layer, which describes data mapping by means of diagrams. In the transformation model we provide just the primitives necessary in typical transformations among Web service data. The proposed model is supported by a software tool, which allows the designer to draw the transformation diagrams between the message contents and the database structure of the peer (and vice versa). The tool is strictly oriented to Web services interactions, and therefore the message contents can be specified by sample XML documents or by means of a WSDL definition.
We provide also the tool integration into the WebML framework for Web services design, but the XSL mapping approach may result valid in other frameworks too.

2. Related Work

Our approach, aiming at the automatic creation of schema mappings, is based on (i) elements matching and (ii) data mapping among schemas through queries-based translations. A similar approach for schema matching is represented by the tool Protoplasm [10], where algorithms are applied to automatically identify the correspondence between elements belonging to different structures: linguistic, data type, and semantic criteria are applied. Protoplasm produces a generic model of data matching among various structures (database and messages), but it does not generate new instances of the target schema. Our work compares also with data mapping approaches focused on data translation from one format to another. For example, in Clio framework [11] given two schemas in various representations (e.g. XML, relational) and a high-level specification of data mappings provided by the user, the tool generates queries (XQuery, XSLT, SQL) to translate source data into the target format. It is a powerful tool with rich mapping capabilities applied to a wide range of heterogeneous data sources. We got inspired from their approach to come up with a specific query language tailored on the needs of B2B interactions. Other commercial and research tools exist that address the problem of data mapping among different XML formats. They all aim at providing transformation rules as XSLT documents to be applied at XML-based source files in order to produce XML-based destination files of a well-known structure. For example, in Accent XML Transformer [12] the user provides manually all the information of the elements necessary to construct the tree representation of the two XML formats, and he may use a limited number of simple transformation rules through a rather complicate interface. In TagFree X2X Mapper [13], the mapping rules are still simple but enhanced with string and numeric operations like concatenation and subtraction. It uses a DTD as a starting point for the generation of XML examples as the source documents. Although this approach enforces the use of valid XML documents, it limits the transformations range in particular cases since various tree representations may be extracted from the same DTD. In Altova MapForce [14], the approach is focused on direct mappings of an XML document to database objects. It supports complex transformations made available through a powerful but friendly interface. It eliminates ambiguity by using XML examples extracted from the XML scheme, but it is focused on general XML-to-XML transformations rather than exchanged Web services messages. Other visual languages (e.g., XQBE [15]) provide visual facilities for building queries on XML documents, thus allowing generating XQuery (and more recently XSLT) rules for extracting data. Once again, these approaches do not fit perfectly in our scenario, since they provide general query languages, rather than a transformation language for XML messages.

3. The proposed approach

Despite the current trends toward service-oriented architectures and semi-structured data models (i.e., XML), several enterprise applications still rely on relational databases. This aspect cannot be ignored, since Web services are often used just to wrap existing information systems and databases. Thus, to represent the stored and managed data, applications usually rely on the E-R model, while Web service messages follow the XML Schema standard. Invoking a Web service within an application requires the ability to: (a) construct an XML-structured message (e.g., a request message) by extracting E-R data from a relational database, and (b) decompose an XML-structured message (e.g., in a response message) into E-R data to be stored and then manipulated in the database.

3.1 Canonical XML

For dealing with XML to E-R mapping, we decided to preserve the use of E-R as the abstract data model for representing both the content of the data stores and the messages exchanged by the Web services, but to exploit XML as an alternative main memory representation of content. We introduced a so-called “canonical XML format”, whose role is that of an intermediate format between the E-R data representation of the database and the arbitrary data representation assumed by the XML schema of the messages exchanged with external Web services. It is very close to the E-R representation of data with XML elements representing entities, relationships, and their attributes. Figure 1 shows a canonical XML representation of a piece of E-R diagram. Element <entity> includes for each entity the set of instances with respective attributes, and element <relationship> includes all the instances of the E-R relationships, described by source and destination object identifiers.

```
<root>
  <instance>
    <attr id="OID">art1</attr> <attr id="name">Eminem</attr>
  </instance>
  <instance>
    <attr id="OID">art2</attr> <attr id="name">Snoop Dogg</attr>
  </instance>
</root>
```

Figure 1. Canonical XML fragment

The benefits of the canonical XML format are twofold:
• It provides a standard format for describing relational data stored in the native format of the application.

• It facilitates the construction and decoding of the input and output messages necessary to interact with Web services. An incoming XML message, which conforms to its own service-dependent XML Schema, needs only to be converted into the canonical XML format to be automatically inserted into the E-R data repository; dually, the E-R content of the data repository can be automatically extracted as a piece of canonical XML, which simplifies its translation into the XML Schema required by the Web service.

3.2 Transformation Language

Using the XML canonical form, the XML-ER data mapping (and the converse mapping) problem reduces to defining the transformations between the generic Web service XML message and the canonical XML representation (and vice versa). In this section we briefly examine the possible cases necessary for mapping XML data to the canonical XML structure. The converse mapping can be obtained in a symmetrical way.

For mapping XML elements to database objects we present some basic terms that will help us indicating the appropriate correspondence. We call Leader an XML element that is mapped to a database entity. This element may be associated to other XML element(s) in three ways: (i) the associated element is nested within the Leader element, (ii) a nested element, called Code element, is used to univocally identify the referenced element (Reference by Code) in the XML document, (iii) an intermediate XML element called Bridge references by Code the two elements. Transformation rules may be used to indicate the creation of (a) a database entity and of (b) a database relationship. Depending on the kind of association different transformation rules apply.

Creation of a database entity. The creation of an entity requires the specification of the mapping between an XML element and an entity, and of the mapping between the XML elements and the entity’s attributes. The former mapping requires the selection of a Leader and a target database entity. For the latter mapping, three cases may arise, depending on the association of the XML elements to be mapped into attributes with the Leader element:

• The XML element is nested inside the structure of the Leader element: the transformation is achieved by transferring the element value to the corresponding attribute into the database entity. In Figure 2, the CD is the Leader element that fires the creation of the database entity Album. The entity's attributes Title, Release_Date, and Record_Company of the Album entity are filled in from the XML elements Title, Release_Date, Record_Company.Name, all nested inside the internal structure of the CD.

![Figure 2. Database entity creation from nested XML elements](image)

• The XML element is implicitly referenced from the Leader element: the Reference by code association is used. In Figure 3, the attribute Record_Company of the Album entity is filled in with the value of the XML element Record_Company.Name. Differently from Figure 2, the XML element Record_Company is referenced from it through its code (<CODE_RECORD>).

![Figure 3. Database entity creation from XML elements referenced by code](image)

• The XML element is explicitly related to the Leader element through a Bridge element. In Figure 4, the attribute Artist of the Album entity is filled in with the content of the XML element Artist.Name. Differently from Figure 2 and Figure 3, the Bridge XML element <Sings> connects the Codes of the two XML elements (CD and Artist).

![Figure 4. Database entity creation from XML elements associated by a Bridge element](image)

Creation of a database relationship. Also a database (binary) relationship between two entities is obtained through transformation rules, based on the following XML Leaders association:

• The Leader elements are nested one inside the other. In Figure 5, the XML CD and Record_Company elements (a) are the Leaders of the entities Album
and Record_Company respectively, and (b) are nested, therefore the relationship between the database entities is created.

Figure 5. Database relationship creation from nested Leader elements

- The first Leader element references by code the second Leader. In Figure 6, differently from Figure 5, the CD Leader references the Record_Company Leader through its code. Therefore, between the entities Album and Record_Company a relationship is created.

Figure 6. Database relationship creation from Leader elements referenced by code

- The two Leader elements are related through a Bridge element. In Figure 7, differently from Figure 5 and Figure 6, the two Leaders are associated through the Bridge XML element Sings. Therefore, between the entities Album and Artist a new relationship Sings is created.

Figure 7. Database relationship creation from Leader elements associated by a Bridge element

4. The design tool

To facilitate the data mapping from an arbitrary XML message to the canonical XML format and vice versa, a visual XSL generator tool has been developed. The tool accepts as input: (i) an XML example message or a WSDL file, which can be selected by the user to automatically obtain the structure of the input or output Web service message; (ii) an XML description (XML canonical form) of the relational schema of the database that is implemented at the interested peer.

The XSL generator GUI (see Figure 8) is divided in two parts: in case of XML message to E-R mapping, the left-hand side represents the structure of the XML message, while the right-hand side represents the structure of the canonical form of the underlying E-R schema (or a sub-part of the involved entities/relationships); otherwise, the two parts are reversed. By connecting nodes in the two trees, the developer defines the transformations to be supported by the generated XSLT file. The connections are either direct edges between XML elements and entities/relationships, or more complex transformations denoted by means of "mapping boxes". The available mapping boxes are visible in the toolbar at the top of the interface, and will be briefly introduced in this section.

The XSL generator supports the transformation rules described in Section 3.

Creation of new database entities. In order to create a new database entity instance from a particular XML element instance, the XML element Leader of the transformation is selected and connected to the database entity. The attribute values of the generated instance can be filled by choosing the corresponding source XML elements. The three different cases described in Section 3 are implemented as follows:

- The XML element is nested inside the structure of the Leader element: only a direct line is necessary to connect it to the destination database attribute. In Figure 8, a direct line transfers the CD.Title element to the database attribute Album.Title.

Figure 8. "E IDREF" mapping box for database entity creation

- The XML element is referenced by code from the Leader element: an “E IDREF” mapping box associates the Code element in the Leader, with the referenced Code element. The output is the value to transfer at the target relational attribute. In Figure 8, the mapping box associates the CD.Code_Record and the Record_Company.Code elements and transfers the Record_Company.Name value.

- The XML element is related to the Leader element through a Bridge element: an “E TABLE” mapping
box associates the two related Code elements, and selects the value to be transferred. In Figure 9, a mapping box associates the Sings.Code_CD and Sings.Code_Artist elements with the CD.Code and Artist.Code elements respectively, and transfers the Artist.Name element to the attribute Album.Artist.

Figure 9. "E TABLE" mapping box for database entity creation

Creation of new database relationships. The transformation rules for database relationships creation implement the three cases in Section 3:

- The Leader elements are nested one inside the other: the database entities are connected by means of the containment relationship in the XML document.
- The first Leader element references by code the second Leader: an “R IDREF” mapping box receives as input the Code element of the first Leader and the referenced Code element of the second Leader, and indicates as output the relationship that should be created. In Figure 10, the “R IDREF” mapping box associates the CD.Code_Record and Record_Company.Code elements, and the output indicates the new database relationship.
- The two Leader elements are related through a Bridge element: an “R TABLE” mapping box receives as input the Code elements of both Leaders in the Bridge element as well as their Code elements in their internal structure. In output, the box indicates the database relationship to be created. In Figure 11, the “R TABLE” mapping box associates the CD.Code and Artist.Code elements with the Sings.Code_CD and Sings.Code_Artist elements respectively. The output of the mapping box indicates the new database relationship.

Figure 10. "R IDREF" mapping box for database relationship creation

Once all the needed transformation edges and boxes have been specified, the necessary XSL files for the transformation can be automatically generated. The transformation we devise includes two steps:

(i) The first step consists in inserting the needed key attributes (unique identifiers of tags) in the XML file. This is absolutely necessary since XML documents do not guarantee uniqueness properties, but this is fundamental for obtaining correct transformations. The inserted keys will identify the correspondent database objects when the XML source file will be transformed into a canonical structure;

(ii) The second step consists in the actual transformation rules represented in the visual diagram.

To make transformations from an E-R representation to an XML document, the XSL generator can be used in the symmetric way.

5. Integration into existing tools and experiences

The approach to data marshalling has the big advantage of avoiding the designer to manually write the translation rules between the ER and XML data formats: the transformations can be specified at high level, through visual diagrams. However, this solution has a big impact on the design process only if also the other aspects of the design (services) can be achieved with the same level of abstraction. Our purpose consists in offering a full high-level design framework of all the XSL transformations needed in the interaction with remote services.

The XML mapping visual language can be integrated with existing proposals specifying Web applications supporting Web services at a high level. In particular, we have experimented it with the WebML language [1, 2], a high-level notation for data-centric Web applications, recently extended with Web services primitives [3], and supported by the CASE tool WebRatio [8], a development environment for the visual specification of applications in WebML and the automatic generation of code for the J2EE and Microsoft .NET platforms.

Figure 12 shows an example of a typical pattern of a Web service interaction within the user interfaces of a WebML application illustrating the abstraction level in the specification of a WS operation. Suppose that during the
navigation of the user a Web service call is needed to get some data from a remote service (Request-Response operation). This call allows a) to compose a message (Request SOAP message) from input E-R data specified in the XML canonical form; b) to send it as a request to a remote service, c) to decompose the response message into the canonical XML form that represents database objects. Step b) can be automated, so that the designer needs only to specify the target service and port. Steps a) and c) depend on the service and need to be specified by the designer for each service operation. Our tool may be used to facilitate this task to the designer. In WebML two operations have been realized to wrap E-R data into the canonical XM format: the XML-out operation extracts the data from the relational database and exposes them in XML Canonical format, the XML-in operation stores the data received in the XML canonical format into the relational database.

This WebML pattern can be transposed also to other tools supporting Web services.

Figure 12. WebML Web Service interaction pattern

6. Conclusions

In this paper, we have proposed a visual language for representing data mapping between arbitrary XML messages to relational data representation. We proposed a canonical XML format as an intermediate representation between generic XML documents and relational data, and a set of visual primitives for mapping of XML elements to relational concepts. The transformation primitives are specifically tailored to the typical needs of Web services data exchange. No unnecessary complexity has been added to the diagrams, at the purpose of giving an easy and useful tool to the developer.

Our implementation experience includes the development of a XML mapping tool, which allows the designer to draw transformation diagram and to automatically generate the corresponding XSL transformations. This tool has been integrated in the WebML design framework and in the WebRatio CASE tool, but it could be easily integrated in other tools, such as for example BEA WebLogic Workshop [9].

In conclusion, the use of our visual approach to data transformation saves time and effort to the application developer and keeps the design activity at a higher level. This is a killer aspect toward the efficient realization of effective inter-enterprise service based architectures.

Acknowledgements

We would like to thank Stefano Bocconico e Stefano Zoofito for their contribution in the development of the data mapping tool.

References