Automated Data and Service Mapping for Integrated Electronic Markets

Stefan Böttcher, Sven Groppe, Tim Schattkowsky
University of Paderborn, Faculty 5, Cooperative Computing & Communication Laboratory
Fürstenallee 11, D-33102 Paderborn, Germany
email: stb@uni-paderborn.de, sg@uni-paderborn.de, tim.schattkowsky@c-lab.de

ABSTRACT
Currently there are multiple different catalog formats for product descriptions used in electronic markets and enterprise-internal applications. Whenever E-procurement applications rely on the catalog format, but want to integrate product descriptions that are stored in a different catalog format, a key problem is how to map different catalog formats onto each other. A further requirement is the integration of services (offered e.g. by remote market places) into enterprise applications that use a different catalog format. We demonstrate that mappings can be generated to combine services that operate on different catalog formats. Our approach avoids unnecessary manual work for mapping and automatically generates mappings between different services wherever possible. This allows us to run E-procurement applications on different catalogs with a fairly reduced manual work needed for mapping.

Keywords: electronic marketplace, classification, product catalog, data transformation, query reformulation, service mapping.

1. INTRODUCTION
Problem origin and motivation
E-procurement systems (e.g. electronic market places) handle their product data in XML formats like BMEcat [23], GOM [22], cXML [25], xCBL [24]. These XML formats provide for each product data a field for a code of a product group called category. Categories are arranged in a structure called classification (c.f. figure 1).

A variety of standards have been suggested for classifications, e.g. ETIM [16], eCl@ss [19], Edibatec [18], UN/SPSC [17], NACE [21], PRODCOM [20], IEC 61360 [14], UniClass [15].

1 Usually a classification is defined hierarchically, but there is no need for defining the classification in such a way in the context of the paper. In figures here, however, classifications are presented hierarchically, nevertheless.

Additionally, many companies have their own classification for historic reasons. In order to interchange data between different E-procurement systems two tasks have to be solved: a conversion between the XML formats and a conversion of the product data according to the classifications are needed. While XML transformers (e.g. XSLT) usually handle the first task, our work in [7] contributes to save the second task, i.e. to map different classifications for the product data onto each other (c.f. figure 2).

In E-procurement systems, client programs (e.g. PA and PB) are calling services (e.g. service SerA and service SerB) with parameters according to their used catalog that return results according to their used catalog (c.f. figure 3).

A further step to integrate E-procurement systems is to apply service mapping (c.f. figure 4) to service calls of client programs (e.g. PA). Service mapping includes the mapping of service call, its parameters and result. The parameters (and result respectively) can be product data in the format of classification of service SerA (and service SerB respectively).

Because mapping of services and classifications plays such a central role in integrating E-procurement systems and often the
costs prohibits to develop a mapping manually, our goal is to minimize the manual work needed for mapping definitions by reversing and chaining already given mappings for product data, queries and service calls (c.f. figure 5). This contribution is a substantial profit for clients and suppliers participating in E-procurement systems, as clients of integrated E-procurement systems can access more offers of product than in a stand-alone E-procurement system and merchants in integrated E-Procurement systems can access a larger clientele than in a stand-alone E-procurement system. A client using service SerA accesses service SerD by applying a service mapping from SerA to SerD (c.f. figure 5). The service mapping from SerA to SerD is generated by reversing the given service mapping from SerC to SerB in order to get a service mapping from SerB to SerC (c.f. figure 5), then chain this generated service mapping from SerB to SerC and the given service mappings from SerA to SerB to a service mapping from SerA to SerC, at last chain this service mapping from SerA to SerC and the given service mapping from SerC to SerD to the necessary service mapping SerA to SerD.

Reversion of mappings has been discussed in the context of updating XML [27],[28]. Common to the database-XML mapping approaches, we also support data transfer in both directions, however we use a direct mapping from XML data to XML data, based on classifications that are used in catalog systems, without the need to map the data to a database, and we extend the mapping to service mapping. Another system containing a component for mapping of XML data between different XML structures is BizTalk Mapper [26], a part of the BizTalk Server. As BizTalk, we support flexible and complex functions within the mapping definitions, but additionally we consider classifications and reverse mapping definitions. A further contribution [12] reverses rules of logic programming languages. However, this is done in the context of query reformulation and not in order to reverse a mapping definition. Additionally, we examine how to handle functions within the rules, how to add chaining of service mapping, and how to reverse a mapping definition. Finally, service integration and service mapping in distributed electronic market places have been considered e.g. in the context of agent technology (e.g. [1]). In contrast to these approaches, we use data mapping as basis for automated generation of reversed data mapping and service mapping.

We focused in [7] on the automated generation of mappings to integrate different catalog data formats into enterprise applications that require another catalog format and use this as basis for a free exchange of catalog data and queries in a distributed electronic market place. Within this paper, we extend the approach in [7] to the automated generation of chained and reversed service mappings between different electronic market places, where a set of service mappings is given.

2. SERVICE MAPPING

In the following subsections, we describe how to define mapping for service calls and how a set of given mappings can be used in order to generate new mappings for service calls.

Defining service mapping

Whenever a market place MB offers a service that an application of a market place MA likes to use, then mapping the service (i.e. implement the service on MA in terms of the service on MB) may be the preferred alternative to copying (and mapping) all relevant product data of MB to market place MA. For the following examples, we assume that classification ClA has an additional product category vw, representing cars of the manufacturer "VW", and classification ClB has a similar product category car, representing cars of any manufacturer. In the following example, a service on market place MB offers to install a vw_steering-wheel SW in a car of manufacturer vw called VWCar, and returns a modified vw_car called VWCarNew, i.e.

installVW(vw VWCar, vw_steering-wheel SW, vw VWCarNew)

where VWCar and SW are input parameters and VWCarNew is an output parameter, which represents the car with the installed steering-wheel after the service call is completed. In general, we assume each service call to be implemented by a procedure call

\[
s(is_1,\ldots,is_a, os_1,\ldots,os_b)
\]

where \(is_1,\ldots,is_a\) are the input parameters and \(os_1,\ldots,os_b\) are the output parameters of the service.
Assume further, that market place $MB$ offers a service comparable to $installVW$

$$install(car\_manufacturer\ Manuf,\ car\_wheel\ Wheel,\ car\ Car,\ car\ CarNew).$$

which installs a wheel $Wheel$ in a car $Car$ of manufacturer $Manuf$, where $Manuf$, $Wheel$, and $Car$ are input parameters and $CarNew$ is an output parameter, and $car\_manufacturer$, $car\_wheel$, and $car$ are product categories of $ClB$.

The goal of service mapping is to implement the service (or part of the service) offered in market place $MA$ by the service offered in market place $MB$. The problem to be solved is that $MA$ and $MB$ use different classifications, i.e. that although the service offered by $MB$ may fulfill all the requirements needed for the service offered by $MA$, we still have to map the parameters of the service call in order to use the service of $MB$ to contribute to our service on $MA$.

Given a data mapping $M$ from $ClA$ to $ClB$ as declared in [7] and a reverse mapping $M^{-1}$ from $ClB$ to $ClA$, which can be generated from $M$ by the approach presented in [7], $map$ is the function, that applies mapping $M$ to product data of $ClA$ in order to generate a representation of these product data according to $ClB$.

For example, $map$ applied with mapping $M$ to product data $SW$ of the product category $vw\_steering\_wheel$ of catalog $CatA$ generates an instance $Wheel$ of the product category $car\_wheel$ of $ClB$. This is written as

$set\ Wheel=map\ (SW, M^{-1})$.

Analogously, $map$ applied with mapping $M$ to an instance $VWCar$ of category $vw$ generates an instance $Car$ of category $car$ defined in $ClB$. This is written as

$set\ Car=map\ (VWCar, M^{-1})$.

Finally, $map$ applied with the reverse mapping $M^{-1}$ to an object $CarNew$ of category $car$ of $ClB$ generates an instance $VWCarNew$ of the category $vw$ defined in $ClA$. This is written as

$set\ VWCarNew=map\ (CarNew, M^{-1})$.

Then, a service call to

$installVW(VWCar, SW, VWCarNew)$

on market place $MA$ can be mapped to a service call

$install(Manufacturer, Wheel, Car, CarNew)$

on market place $MB$ by following rule (with $install$ and $installVW$ being defined as before):

Example 1:

$installVW(VWCar, SW, VWCarNew):-
set\ Manufacturer=\"VW\",
set\ Wheel=map\ (SW, M),
set\ Car=map\ (VWCar, M),
install(Manufacturer, Wheel, Car, CarNew),
set\ VWCarNew=map\ (CarNew, M^{-1})$.

In general, in order to implement a service $s$ on one market place by a service $t$ on another market place, we need a mapping that defines how input and output parameters are mapped to each other. The mapping rule is:

$s(is_{1},...,is_{n}, os_{1},...,os_{m}):-
\set\ it_{1}=f_{i1}(i1, \ldots ,ik, a),
\set\ it_{2}=f_{i2}(i1, \ldots ,ik, b),
check\ condition, \\ t(it_{1},...,it_{n}, os_{1},...,os_{m}),
\set\ os_{1}=f_{o1}(o1, \ldots ,ok, c),\nonumber
\set\ os_{2}=f_{o2}(o1, \ldots ,ok, d),...,
\set\ os_{m}=f_{om}(ob, \ldots ,oq, e)$.

where $is_{1},...,is_{n}$ are input parameters and $os_{1},...,os_{m}$ are output parameters of the service $s$ to be implemented, and $it_{1},...,it_{n}$ are input parameters and $ot_{1},...,ot_{m}$ are output parameters of the service $t$ that is used to implement $s$, and each $i_{k}$, $o_{k}$, $i_{k}$, $o_{k}$, $i_{k}$, $o_{k}$, $i_{k}$, $o_{k}$, $i_{k}$, $o_{k}$, is one of the input parameters $is_{1},...,is_{n}$ and each $os_{1},...,os_{m}$ is one of the output parameters $ot_{1},...,ot_{m}$. Furthermore, condition restricts the input parameters $it_{1},...,it_{n}$ of the service $t$.

In general, a condition condition of the mapping rule is defined as follows. true and false are conditions. If Attr is an attribute defined for a category source, const is a constant value, and op is a comparison operator ($<,>,\neq,\geq$), then source.Attr op const is a condition. If B1 and B2 are conditions, then B1 and B2, B1 or B2 and not B1 are conditions too.

The assignments to $it_{1},...,it_{n}$ have to be mapped before the service $t$ is called, and the output parameters $os_{1},...,os_{m}$ must be mapped after the call of $t$. Whenever a service has a parameter used for input and output, the rule above can be applied by listing such a parameter in both lists of input parameters and output parameters.

Chained service mapping

As with data mapping, it is possible to chain service mapping, i.e. if market place $MA$ implements a service $s$ that is mapped to a service $t$ of market place $MB$, $MB$ may map this service $t$ to a service $u$ of market place $MC$ on the basis of a given data mapping $M$ between the classifications used in $MB$ and $MC$. Instead of executing the service $t$ of market place $MB$, we can chain the service mapping definitions (see Figure 6) such that market place $MA$ can call directly the service $u$ of market place $MC$. The function query_map(condition, $M$) reformulates the query condition, according to the data mapping $M$ as described in [7].

Reversed service mapping

Assume, we have already a service mapping that implements a service, say $installVW$, on a market place $MA$ by a service, say $Install$, on market place $MB$. With this service mapping, we can use product data of a catalog $CatB$ in $MB$ in the service offered on market place $MA$.

When we now also want to compute a service mapping in the opposite direction, e.g. in order to use the product data of a catalog $CatA$ in $MA$ and the service $installVW$ on market place $MA$ in an application running on market place $MB$, then there are different possibilities how to compute such a reversed service mapping from $installVW$ to $install$.

\footnote{Manuf must be one of "VW" and "Opel".}
(1) \( s(is, \ldots, is_a, os_a, \ldots, os) :\)
\[
\begin{align*}
&\text{set } it_1 = f_{s1}(is_{1s}, \ldots, is_{1s}), \\
&\ldots, \\
&\text{set } it_n = f_{sn}(is_{ns}, \ldots, is_{ns}), \\
&\text{check condition}, \\
&t(it_1, \ldots, it_n, ot_1, \ldots, ot_m), \\
&\text{set } os_1 = f_{os1}(ot_{1s}, \ldots, ot_{1s}), \\
&\ldots, \\
&\text{set } os_n = f_{osn}(ot_{ns}, \ldots, ot_{ns}).
\end{align*}
\]

(2) \( t(it, \ldots, it_n, ot, \ldots, ot_m) :\)
\[
\begin{align*}
&\text{set } iu_1 = f_{iu1}(it_{1s}, \ldots, it_{1s}), \\
&\ldots, \\
&\text{set } iu_j = f_{iu}(it_{js}, \ldots, it_{js}), \\
&\text{check condition}, \\
&t(it_1, \ldots, it_n, ot_1, \ldots, ot_m), \\
&\text{set } ot_1 = f_{ot1}(ou_{1s}, \ldots, ou_{1s}), \\
&\ldots, \\
&\text{set } ot_n = f_{otn}(ou_{ns}, \ldots, ou_{ns}).
\end{align*}
\]

\[
\begin{align*}
(3) \quad s(is, \ldots, is_a, os, \ldots, os_a) :&= \\
&\text{set } iu_1 = f_{iu1}(is_{1s}, \ldots, is_{1s}), \\
&\ldots, \\
&\text{set } iu_j = f_{iu}(is_{js}, \ldots, is_{js}), \\
&\text{check query_map}(condition, M) \text{ and condition}, \\
&t(it_1, \ldots, it_n, ot, \ldots, ot_m), \\
&\text{set } os_1 = f_{os1}(ot_{1s}, \ldots, ot_{1s}), \\
&\ldots, \\
&\text{set } os_a = f_{osn}(ot_{as}, \ldots, ot_{as}).
\end{align*}
\]

Figure 6: Chaining of service mapping definitions

Figure 7: Reversing a service mapping definition

One possibility is to use the data mappings \( M \) and \( M^{-1} \) as before to compute the new service mapping by just exchanging the role of CatA and CatB. The disadvantage of this possibility is that it is necessary to identify corresponding parameters of the services (and to map them by reverse functions) a second time.

In order to avoid this, it is alternatively possible to take the given service mapping as input and to generate a reversed service mapping, similar to as we generate a reversed data mapping.

The reversed service mapping of the service mapping shown in Example 1 is

Example 2:

\[
\begin{align*}
\text{install}(&\text{Manufacturer, Wheel, Car, CarNew}) : \\
&\text{check} \text{ Manufacturer} = \text{"VW"}, \\
&\text{set} \text{ SW} = \text{map(} \text{Wheel, } M^{-1} \text{)}, \\
&\text{set} \text{ VWCar} = \text{map(} \text{Car, } M^{-1} \text{)}, \\
&\text{installVW} (\text{VWCar, SW, VWCarNew}), \\
&\text{set} \text{ CarNew} = \text{map(} \text{VWCarNew, } M \text{)}. \\
\end{align*}
\]

This reversed mapping can be used to implement a service on market place \( MB \) (install) by calling a service of market place \( MA \) (installVW).

The general approach to reverse a service mapping is shown in Figure 7. \( M_{st} \) is the data mapping from the classification used in service \( s \) to the classification used in service \( t \). The reversed mapping \( M_{st}^{-1} \) can be computed by the approach described in [7]. The used functions \( f_s \) must be invertible and in order to retrieve the needed parameters \( is, \ldots, is_a \) and \( ot, \ldots, ot_d \), respectively, we must choose the correct \( pl, \ldots, p_a \) in \( \{1, \ldots, c\} \) (and the correct \( q_1, \ldots, q_d \) in \( \{1, \ldots, b\} \) respectively). The function \( \text{query_map} \) reformulates the condition \( \text{conditions, } M_{st}^{-1} \) given as parameter according to the data mapping \( M_{st}^{-1} \) given as parameter as described in [7].
Product catalogs of engineering companies with a wide product portfolio store products according to classifications, which are also used by enterprise applications that access such a catalog. However, these classifications are not just placed as an island as online catalogue on the companies website but is also integrated into customers’ E-procurement systems or marketplaces, together with the products of other manufacturers. Those company-spanning systems, in general, use different classifications, in most cases less specific than those of the companies or with different structures. Therefore the catalog integration requires a mapping between the classifications. Since the manual map definition between different classifications requires considerable work and is error prone, we suggest instead generating mappings automatically wherever possible. This includes the automated generation of reverse mappings and chained mappings. We presented, how to generate these mappings for service mapping.

This allows a service which has been implemented for a single catalog format to use data of all other catalogs for which a mapping exists, and it allows to use services of other market places with the locally stored catalog data. We consider this exchange of data and services across the boundaries of company specific or market place specific catalog formats to be an important step towards enterprise application integration. Although we developed our approach in the context of electronic market places, it might be promising to investigate how this can be extended to other areas of enterprise application integration.

5. REFERENCE

[19] eCI@ss e.V.: eCI@ss - Standard für Materialklassen und Warengruppen. http://www.eClass.de, 2002. eCI@ss e.V. c/o Institut der deutschen Wirtschaft, Köln.