Translating and Searching Service Descriptions Using Ontologies

Daniela Damm[1], Farshad Hakimpour[2,4], and Andreas Geppert[3,5]

[1]Department of Information Technology, University of Zurich, Switzerland
damm@ifi.unizh.ch

[2]Department of Geography, University of Zurich, Switzerland
farshad@geo.unizh.ch

[3]Credit Suisse, Zurich, Switzerland
geppert@acm.org

Abstract. Web services offer a promising approach to accomplish a loose coupling of processes across organizational boundaries. Several industry standards have been developed to describe process or service related information. Although these standards capture similar concepts, they nonetheless rely on diverging interpretations of processes and their components. In this paper, we present an approach to facilitate the search for services, where services and requests can be described using heterogeneous standards. To enable the mapping of a request to different service description standards, formal ontologies are used. Ontologies are related on the basis of similarity relations in order to translate between elements of different service descriptions. The resulting translation is used to process a request and find an adequate service. We present a top-down and a bottom-up method to implement this kind of request processing.

1 Introduction

Development and evolution of web services allow an enterprise to loosely couple its internal processes with business activities of its suppliers or other partners. Based on common standards, web services enable an interaction between the applications of involved parties without the time-consuming and expensive integration of the involved information systems. Web service-related standards focus mainly on technical aspects of the interaction between two applications [7]. More business process-related aspects of a product or service (like business activities, document exchange, role descriptions) are not addressed.

For this purpose several industry standards (like RosettaNet, ebXML, XPDL) can be applied.[6] Although these standards capture similar concepts, they nonetheless rely on diverging interpretations of processes and their components. Moreover, the B2B-

4. The work of Farshad Hakimpour is funded by Swiss National Science Foundation (Project Number: 2100-053995).

5. Work done while with the University of Zurich

6. We will refer to such standards as service description standards or languages.

J. Eder and M. Missikoff (Eds.): CAiSE 2003, LNCS 2681, pp. 308–323, 2003.

standards often lack unambiguous specifications of the meaning of their constituent terms.

To enhance its processes by integrating external web services, an enterprise must be able to locate the services that meet its requirements. In contrast to traditional process design, the expected number of available web services on the Internet is ever increasing [4]. Manual service detection is no longer possible and efficient service detection mechanisms must be provided.

We present an ontology-based approach to facilitate the resolution of heterogeneity of different service description standards. We use a higher-level ontology that is based on existing service description standards. This ontology is used to match service requests and descriptions formulated in different description languages. While issuing a request, the service consumer can search for services independent of the service description language. We assume that every standard relies on a formalized ontology (called lower-level ontology here) and that a service description specified in a standard is based on its lower-level ontology. We use the lower-level ontologies to detect semantic similarities between different standards that can be applied to translate service descriptions or requests.

The remainder of this paper is organized as following. In section 2, we give a brief overview of web service-related concepts. Section 3 discusses related research approaches. In section 4, we present an overview of our approach. The subsequent sections present different aspects of our work in more detail. In section 5 we discuss the detection of semantic similarities between different standards. Based on the similarity relations, a transformation structure for the translation between two standards is built. Section 6 describes the generation of the transformation structure. In section 7 we show the application of the transformation structure to translate a service description document. In section 8 a top-down and a bottom-up approach to implement the service request processing are introduced and compared. Finally, section 9 concludes the paper and discusses future work.

2 Web Services in a Nutshell

Web services provide concepts and methods to implement dynamic and flexible interaction between companies and their respective business processes. Standards and systems allow an enterprise to offer its products and services via the Internet to its customers and partners. The use of standardized technologies enables the coupling of enterprises' activities without the need for expensive and time-consuming integration with the partner's information systems [7]. Web service technologies are mainly based on WSDL[7], UDDI[8] and SOAP[9].

WSDL is used to describe the interface of a service as an XML document. It contains the minimal necessary information to enable interaction between web service ap-

7. Web Service Description Language (http://www.w3.org/TR/wsdl).
8. Universal Description, Discovery, and Integration (http:// www.uddi.org/).
9. Simple Object Access Protocol (http://www.w3.org/tr/soap/).

plications (such as data formats, messages, port types, etc.). In order to offer his services, a service provider generates a WSDL document and publishes it either directly or to a service registry. UDDI is a standard defining the interface of such service registries. Web services are stored in the form of business registrations which are XML files used to describe business entities and their services. A UDDI service registry entry captures general information (e.g. provider name, contact information), taxonomy references and technical aspects of a service. SOAP is used to implement the interaction between service providers and consumers. Based on a service description, a service consumer generates a SOAP request to invoke the corresponding web service. WSDL focuses on the basic, technical description of service enactment [1]. In order to implement interactions between companies, additional information about a service must be provided (e.g. descriptions of involved business partners, their role, provided activities) [7]. For this purpose several existing industry standards can be used.

We distinguish two different kinds of standardization efforts: *interaction-oriented* and *process-oriented* standards. Interaction-oriented standards focus on the cooperation between business partners and provide concepts to model the information exchange and an abstract description of activities between them. A detailed specification of enterprise internal processes is not in the scope of such standards. An example is *RosettaNet*, which proposes partner interface processes (PIP) in order to describe interaction along the whole supply chain between two processes executing in different enterprises [16]. ebXML offers a framework for the modelling of interactions between business partners and the exchange of business-related information (*business documents*) [8]. The flow of business activities and the involved business document exchange is specified in a process specification. (Other examples of interaction-oriented standards are cXML [6], EDI or OBI).

In contrast to interaction-oriented standards, process-oriented standards support detailed process descriptions, involved activities and their dependencies. Examples of process-oriented standards are XPDL or BPML [19,2]. XPDL relies on the process meta model of the Workflow Management Coalition and is intended to provide a standardized workflow model in order to enable the exchange of workflow specifications between business partners. It provides concepts for a detailed representation of activities, actors and data related to a workflow. Another example of a process-oriented standard is BPML, also intended to support the exchange of process-related specifications between partners.

Although each of these standards contains the same or similar concepts, they often use different names for these concepts (for example: sequence [16] and choreography [8] are semantically equivalent). Likewise, the same term can be used in different standards to denote more or less different concepts. For example, the term “transaction” means business transaction in [8], while in [16] it refers to a database transaction.

3 Related Work

Two projects supporting integration of heterogeneous service description standards are Meteor and PSL. Meteor is using a common taxonomy of service-related terms to resolve semantic heterogeneity which is used to detect similar terms in different service

descriptions in order to enhance the detection of services as well as the mapping of the interfaces of services during their combination [4]. If two service descriptions contain the same attribute, the similarity of their values is quantified by calculating their distance within the taxonomy tree. The detection of similarity is limited to the attribute values and only a mapping of homogenous service requests and descriptions is supported.

The Process Specification Language (PSL) project which defines a specification language for process-related information as a base for the implementation of common gateways to exchange data among different process-related applications [17]. PSL is not concerned with existing process-related standardization efforts, but relies on a proprietary process meta model. PSL provides a formal ontology which defines the semantics of the terminology of PSL [5]. To exchange process data between two applications, the underlying semantics of their specification languages has to be mapped to the overlapping part of the PSL ontology, and the applications' syntax has to be translated into the KIF syntax applied in PSL. Only if the semantics of all three ontologies (that is both ontologies underlying the involved applications and the PSL ontology) have common terms and concepts, information can be exchanged.

Another line of research aims at enhancing existing web service related standards in order to improve their modeling capabilities and to describe complex services [1,3,13]. In [13] a process ontology developed by the MIT Process Handbook project is used to support the discovery of services. Depending on the activities a web service contains, indexes are added to its description based on the process taxonomy defined in the ontology. These indexes can be used to query service repositories more precisely.

The DAML-S project provides a description language for services based on semantic web enabling technologies [1]. DAML-S allows the description of business related aspects of services, and can thus be used to complement existing web service related standards. Two specific elements for the service description are provided. A SERVICE PROFILE allows a high-level description of a service and its provider, while a SERVICE MODEL expresses usage-oriented information. By enhancing a service description with this information (e.g. description, quality aspects, service categories) the service discovery, invocation and combination can be supported. Another approach based on technologies related to the semantic web is the Web Service Modeling Framework (WSMF) [3]. It extends the modeling capabilities of DAML-S in order to provide a flexible framework for the description of various aspects related to web services.

4 The Proposed Solution – An Overview

To avoid semantic heterogeneity, we should guarantee a unique interpretation for every term used in the communication between partners. While relating service descriptions, we face semantic obstacles at two different levels:

1. different service descriptions from different communities presented in the same structural standard
2. different service descriptions prepared in two different standards

The main concern of this work is the second issue. We aim at finding an approach for the automatic or semiautomatic communication between a service consumer and a pro-

vider using different standards while resolving semantic conflicts. A unique and explicit definition for every term in an ontology can serve our purpose to avoid semantic heterogeneity [14]. The major advantage of using ontologies is to avoid misinterpretations by detecting the differences and similarities in the meaning of terms.

In our work, we use a formal higher-level ontology for service description languages (SDL-ontology) to support the detection of services specified in heterogeneous service description standards. We assume that the underlying semantics of each particular service description standard are captured in a formal ontology and service descriptions or requests written in this language are referring to the underlying ontology. The translation between two different service description languages is done by merging their underlying ontologies with respect to the SDL-ontology. Note that the building of ontologies is not within the scope of this paper. Methodologies for this can be found in [9,10,18,11].

The SDL-ontology specifies the meaning of common terms in the involved description service languages. It is built by integrating low-level ontologies of different available service description languages. If another description language has to be added to the solution, in a first step, it has to commit to the SDL-ontology. This process cannot be automated, but needs human intervention. Good knowledge of the low-level language ontology as well as the SDL-ontology is required.

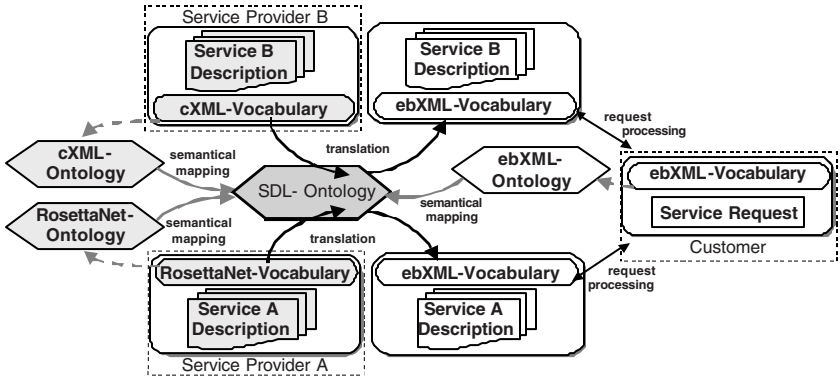

Fig. 1. Ontology-based Service Search (bottom-up)

Once a language is fully added (that is, its low-level ontology is imported and commits to the SDL-ontology), based on the similarity relations introduced in [12] a reasoning system can be used to detect similarities between the low-level ontologies. The result is then used to implement the mapping of a service request with existing service descriptions written in different languages. To allow a mapping of heterogeneous service request and descriptions, similarity relations detected during the merging of the underlying ontologies are used to translate either service descriptions or service requests.

Generally, two approaches for the translation can be distinguished. Applying a top-down procedure, a service request is translated into other available languages. A translated service request is then transferred to the service sources (e.g., a service registry)

in order to search for matching service descriptions. Alternatively, in the bottom-up approach service descriptions are translated into one of the available languages and mapped to a service request that is also specified in one specific language. Figure 1 shows the bottom-up approach.

Scenario: E-Parking Service. In the following we consider an e-parking service as an example to illustrate our approach. It is taken from a current European research project[10] at our department[11] implementing an e-parking platform that offers the possibility for customers to reserve parking lots electronically. It combines services offered by different car park providers. Several devices such as computer, PDAs or mobile phones can be used to access the e-parking services. Figure 2 shows a part of the e-parking service description.

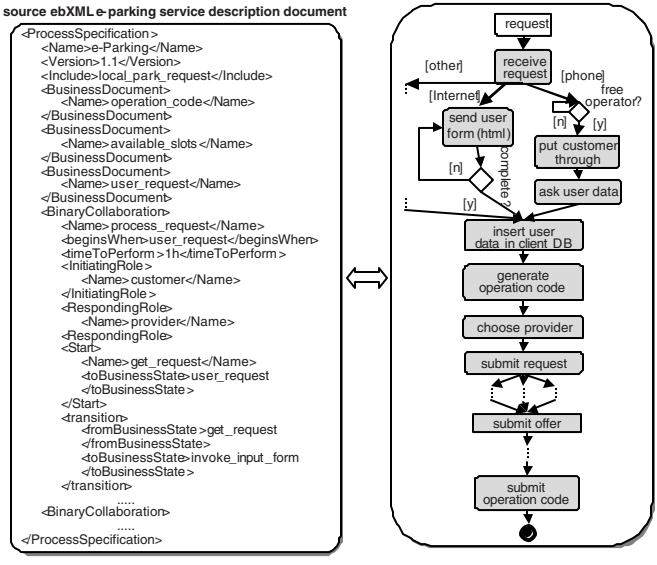

Fig. 2. E-Parking Service

In order to request a parking lot, a registered customer sends a message specifying the desired date, time and location as well as the type of vehicle. In return, he receives a list with available parking slots matching his requirements. The customer selects one of the offers and (if the request can be processed successfully) receives an operation code as parking slot token in return.

We assume that the e-parking service is defined in ebXML and that a service consumer uses XPDL to specify his request (see Figure 3) [8,19].

To process the XPDL request, a translation between ebXML and XPDL has to be provided. In the following we will use the e-parking service to illustrate our approach.

10. E-parking: User-Friendly e-Commerce to Optimise Parking Space (IST-2000-25392)
11. Department of Information Technology, University of Zurich.

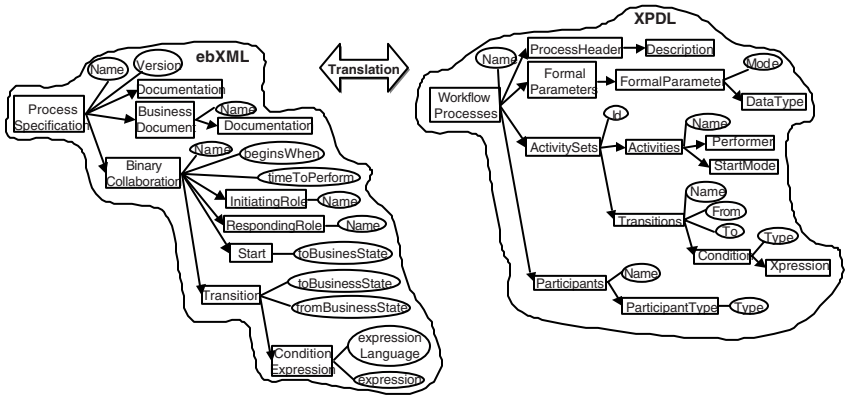

Fig. 3. ebXML and XPDL

5 Semantic Similarities

The translation of a service request or description specified in one standard to another description standard is based on similarity between terms and concepts captured in the underlying ontologies of the different languages. To obtain these similarity relations between service description languages, we use the ontologies available for the service description languages and detect similarities or differences between them. This is done by *matching* their intensional definitions (i.e., definitions of the meaning of terms by logical axioms). For that matter, we use the similarity relations introduced in [12]. There, four levels of similarity between two terms ${}^{p}T_i$ and ${}^{q}T_j$ defined in formal ontologies p and q are proposed. The relations *equal, specializes* (*generalizes), disjoint*, *overlaps* are defined as follows (ι maps a term to its intensional definition):

- ${}^{p}T_i$ is equal to ${}^{q}T_j$ if and only if both intensional definitions are the same:
$$({}^{p}T_i = {}^{q}T_j) \Leftrightarrow (\iota[{}^{p}T_i] \equiv \iota[{}^{q}T_j]) .$$
- ${}^{p}T_i$ specializes ${}^{q}T_j$ if and only if the conjunction of the two intensional definitions is the same as the definition of ${}^{p}T_i$:
$$({}^{p}T_i \leq {}^{q}T_j) \Leftrightarrow (\iota[{}^{p}T_i] \wedge \iota[{}^{q}T_j]) \equiv \iota[{}^{p}T_i] .$$
- ${}^{p}T_i$ overlaps ${}^{q}T_j$ if and only if the conjunction of the two intensional definitions is not false:
$$({}^{p}T_i \sim {}^{q}T_j) \Leftrightarrow (((\iota[{}^{p}T_i] \wedge \iota[{}^{q}T_j]) \equiv \iota[T_k]) \wedge \neg(\iota[T_k] = False)) .$$
- ${}^{p}T_i$ and ${}^{q}T_j$ are disjoint if and only if the conjunction of the two intensional definitions is false:
$$({}^{p}T_i \neq {}^{q}T_j) \Leftrightarrow ((\iota[{}^{p}T_i] \wedge \iota[{}^{q}T_j]) \equiv False) .$$

Matching in this phase requires both ontologies to commit to the SDL-ontology. A reasoning system uses this higher-level ontology as a common reference of the two on-

tologies for matching.The results of the merging is stored and used for the translation between different service description languages.

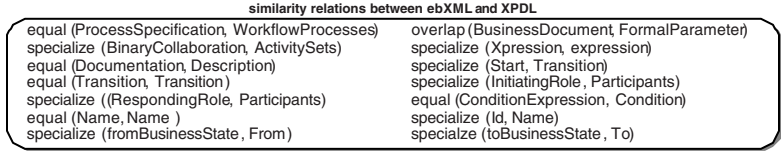

Fig. 4. Similarity Relations between ebXML and XPDL

Figure 4 shows parts of the similarity relations that can be detected between the terms of ebXML and XPDL.

6 Generating a Transformation Structure

In this section we apply the similarity relations between the ontologies to generate a transformation structure between two description standards. Based on the similarities between elements (and their intensional definitions) of description standards, we build a transformation structure. This transformation structure is then used to translate a service description into another service description language. It is not possible to translate the terms of a source (S) and a target service description standard (T) without regarding the structural differences of both. The purpose of a transformation structure is to preserve the structural aspects of T while translating a document. The transformation structure captures not only the semantic similarities of the terms in S and T, but maps the terms in S to the structure of T. It thus offers a way to relate the elements in a particular service document written in S according to their semantics (defined in the underlying ontology of S) to both, the terms and the structure of T. Once a transformation structure is generated, it can be used to translate any document in S to a document in T.

One main objective while generating a transformation structure is to prevent hierarchical relations that are not part of the specification of the elements in S from being included in the translation document. For example, if the specification of an activity contains an application, it can have different interpretations: the activity uses the application, or the activity is enacted by the application. Thus, while relating elements of S and T, we cannot simply change the hierarchical dependencies. Due to the hierarchical structure of XML, we do not have further knowledge about the semantics of a relation between two elements s_1 and s_2 (e.g. s_2 is a subelement of s_1) in S. Thus, the transformation structure is built in a way that the hierarchical dependencies between elements in S are reflected by their counterparts in T (that is, if s_2 is a subelement of s_1 in S, its corresponding element t_2 is a subelement of t_1 in T), but not their inversion (t_1 is a subelement of t_2). Although this can result in empty elements in the translated service description, it avoids the translation of relations between elements in S to different relations in T without knowing for sure whether the intensional definitions of the relations are semantically similar.

If a term s in S generalizes a term t in T, instances of s are not necessarily classified in the extension of t. Therefore, we cannot provide an automatic translation in such a

case. The same applies for a term s in S that overlaps with a term t in T. As the main objective of this work is the matching of service requests with provided services, we aim to keep all the available information of a service during the translation processing. Although such additional information cannot be expressed by the standard T, it can still contain useful information for the decision process of a consumer to select a service or not. In order to generate a translation document that is consistent with T and can be processed by an application relying on T, we add such extensions only as comment to service descriptions. A user can take them into account, but applications do not have to cope with them. In order to avoid information loss, we also keep information about disjoint terms of S and T (that is, a term s in S has a minimum semantic similarity to any term in T) as comments in the translation document.

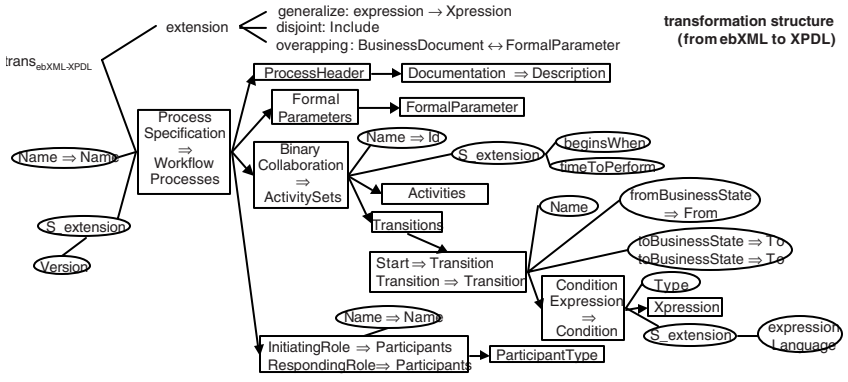

Fig. 5. Transformation structure from ebXML to XPDL

Figure 5 shows the transformation structure for the translation from ebXML to XPDL that is based on the similarity relations between both description standards (see figure 4). In the following we will describe the different steps of generating a transformation structure between S and T. The transformation structure $trans_{S\text{-}T}$ from S to T is generated, starting at the root element of S. First, the similarity relations between elements in S and T, and second, similarity relations between attributes of these elements are processed to generate the transformation structure.

Relating elements of S to those of T

- Generalization, overlapping and disjoint elements:
  If a term *s* in S generalizes or overlaps a term *t* in T, we add *s* as a comment to the translation document. We also include information about the similarity between *s* and *t* (generalize, overlap). During the generation of the transformation structure, we copy the term *s* to a special element in the $trans_{S\text{-}T}$ extension (e.g., $expression_{ebXML}$ is a generalization of $Xpression_{XPDL}$). During the translation, the values of extension are transformed to comments (see section 7.). Disjoint elements in S are included to $trans_{S\text{-}T}$ in the same way (e.g., $Include_{ebXML}$).
- Equal or specialization:
  In case a term *s* in S is equal to or specializes a term *t* in T, we copy the whole subtree $subtree_t$ of T that contains *t* to $trans_{S\text{-}T}$ and include a link from *s* to *t*' (copy of

t) in $subtree_{t'}$. (For example, $ProcessSpecification_{ebXML}$ and $WorkflowProcesses_{XPDL}$ refer to the same concept, thus we include the subtree of *WorkflowProcesses* of the XPDL schema to the translation structure and add an link from *ProcessSpecification* to *WorkflowProcess*.) In the next step we try to resolve semantic relations between the subelements of s and the subelements of t' in the $subtree_{t'}$ in $trans_{S\text{-}T}$:

- Common hierarchical relation between elements:
  If a subelement s_n of s is equal to or specializes a subsequent element t_n' of t' in $subtree_{t'}$, we simply include a link between s_n and t_n' to $subtree_{t'}$. This is done to keep common hierarchical dependencies between corresponding elements of S and T and thus, to transform the hierarchical structure of S to the hierarchical structure of T.
- Avoid inversion of hierarchical relations:
  If a subelement s_n of s cannot be related to any subelement of t' in $subtree_{t'}$, we do not map it to an element that is a superelement of t' (which corresponds to s). This would change the hierarchical dependency between s and s_n during the translation of a document. To avoid the inclusion of an inverted relation, we add a copy $subtree_{tn''}$ of the $subtree_{tn}$ to $trans_{S\text{-}T}$, include a link between s_n and t_n'', and then try to map the subelements of s_n to subtree t_n'' correspondingly.

Relating attributes of elements of S to T

After generating the transformation structure for every element in S, we add the semantic relations between attributes of two equivalent terms s and t. For every element t' in the transformation structure that contains a link to a term s in S (e.g. *ProcessSpecification*, *BinaryCollaboration*, *Start*), we consider the similarity relations between the attributes s.attr and t.attr of s and t.

- If an attribute *s.attr* is equal to *t.attr* and the type of *s.attr* can be converted to the type *t.attr*, we include *t'.attr* to $trans_{S\text{-}T}$ and add a link to *s.attr*. (For example, both $ProcessSpecification_{ebXML}$ and $WorkflowProcesses_{XPDL}$ have an attribute *Name* that addresses the same concept and both are of the same type, thus we add this attribute to the translation structure.) In case an *s.attr* is a specialization of *t.attr*, we proceed in the same way.
- During the translation, we copy information of an attribute *s.attr* that cannot be related to any attribute t.attr to $trans_{S\text{-}T}$ in form of a comment to the translation document. Therefore, we add an additional attribute *S_extension* to every element in the transformation structure that has a link to a term in S. For every attribute *s.attr* that cannot be related (that is equal or specializes) to any attribute *t.attr* in T, we add a reference to *S_extension* (e.g. XPDL does not contain an equivalent to the attribute *Version* of the *ProcessSpecification* in ebXML). While generating the translation service document these references in *S_extension* are transformed into a comment of *t*.

We exemplify the generation of a translation structure with a simple example shown in figure 6. In the first step, all terms in S that generalize or overlap a term in T

or are disjoint to them are copied to *extension* of $trans_{S\text{-}T}$. For example, we include an entry for *c* (generalizing *z*), *b* (overlapping *y*) and *w* which is disjoint to *T* in *extension.* Then, we process terms of *S* that are equal to or are specializations of terms in *T*. Starting at the root *a* (that is equal to *x*), we include to $trans_{S\text{-}T}$ the corresponding $subtree_{x'}$. Then we add the semantic relations of the subelements of *a* that can be mapped directly to the hierarchical structure of *T*. In our example, only *d* and its subelement *l* have the same hierarchical dependencies as their corresponding elements *m* and *k* in *T*.

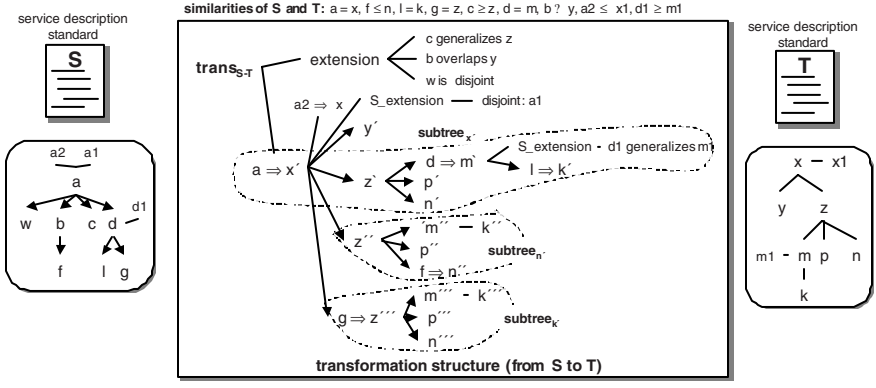

Fig. 6. Generating a transformation structure

Now, the remaining elements *f* and *g* have to be included to $trans_{S\text{-}T}$. For *f* that specializes *n*, we add the copy $subtree_{n'}$ of the subtree in *T* (up to the root *x*) that contains *n* to $trans_{S\text{-}T}$. In the same manner, $subtree_{z'}$ is added for *g* to $trans_{S\text{-}T}$. Finally, the semantic similarities of the attributes of the elements of *S* that are equal or are a specialization of the elements in *T* and therefore are not captured in the extension of $trans_{S\text{-}T}$ are included to the transformation structure.

7 Translation of Service Description Documents

In this section we will show how the transformation structure between the description languages can be used to translate a service request or description from one service description standard to another. In the following, we will focus on the translation of service descriptions (a service request can be translated analogously). The translation of a service description is done in two steps. First, the elements in the service description document are associated with the transformation structure. Then, based on these associations the translated service description document is generated.

To translate the source document we use the semantic links in the transformation structure. Every element in the source document is associated with its corresponding element in the transformation structure. To do this, we process the source document and the elements specified there, starting with the root elements in the document. If a (root) element s_r in the source document has a corresponding element $s_{r'}$ in the transformation structure, we create an instance of the $subtree_{sr'}$ and relate the attributes of s_r to $s_{r'}$ in $subtree_{sr'}$. Attributes, that do not have a semantic link in the transformation structure

are added to the *S_extension* attribute of s_r'. In the same way, we copy subsequent elements *s1*,..., *sn* of s_r to $subtree_{sr}$' and create an instance of the particular $subtree_{sx}$' in $subtree_{s}$'. Figure 7 shows the result of the association of the element in the e-parking service description document (see Figure 2) to the transformation structure $trans_{e\text{-}bXML\text{-}XPDL}$ presented in Figure 5.

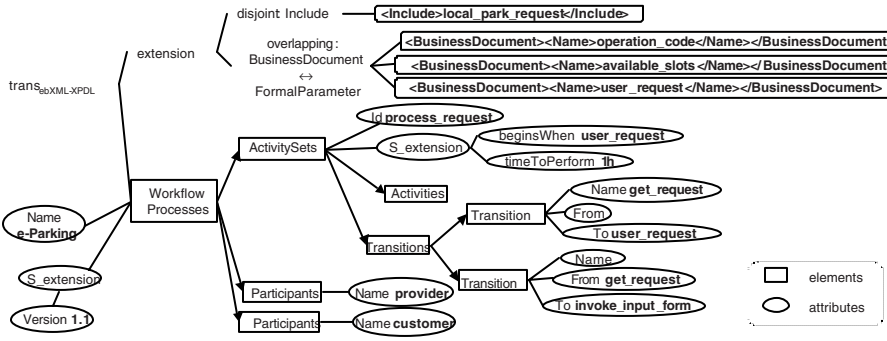

Fig. 7. Association of e-parking service description document

Once every element in the source document has been copied to the transformation structure, the translated service description document can be generated. In a backtracking-oriented method we process the transformation structure and if an element or its sub-elements in the structure have been instantiated, we include the XML specification of the element, its attributes and sub-elements to the translation document (see figure 8). The values of an attributes S_extension of an element s are included as a comment in form of *<!--S extension: <attr*$_s$*> value </attr*$_s$*> -->* to the specification of s. For example, for the attribute *Version* of ebXML the comment *<!-- ebXML extension: <Version>1.1</Version> -->* is added to the specification of *WorkflowProcess.*

Every reference in *extension* (elements of S that cannot be translated automatically) is added as a comment to the translated service document: *<!—ebXML extension: (relation: reference)* -->*. For example, the comment for the element *Include* is *<!—ebXML extension: disjoint: <Include>local_park_request</Include> -->*.

resulting XPDL e-parking service description document

```
<WorkflowProcesses>
    <!– ebXML extension: <Version>1.1</Version> –>
    < Name >e-Parking</ Name >
    < ActivitySets >
       < Id >process_request</ Id >
       < Transitions>
          < Transition ><Name>get_request</Name> <to>user_request</to></ Transition >
          < Transition ><from>get_request</from> <to>invoke _input_form</to></ Transition >
       .....</ Transitions>
    .....</ ActivitySets >.....
    < Participants ><Name> customer</Name></ Participants >
    < Participants ><Name> provider</Name>< Participants >
<!—ebXML extension:
    disjoint: < Include>local_park_request</Include>
    overlapping FormalParameter:<BusinessDocument><Name> operation_code </Name></BusinessDocument>
    overlapping FormalParameter:<BusinessDocument><Name> available_slots</Name></BusinessDocument>
    overlapping FormalParameter:<BusinessDocument><Name> user_request </Name></BusinessDocument> -->
</ WorkflowProcesses >
```

Fig. 8. Translation of the e-parking service document

If elements in the transformation structure have not been instantiated during the association, but have instantiated sub-elements, they are included as blank XML elements. These empty elements serve as containers for their sub-elements in the translation document. Figure 8 illustrates the generation of the translated description document by means of the e-parking example.

8 Request Processing

Two different approaches (see Figure 9) can be devised for mapping service requests and descriptions: *bottom-up* and *top-down* processing. In this section, we give an overview of both and discuss their possible advantages and disadvantages.

In both approaches a *translation layer* contains wrappers for the translation of service description or request documents. A wrapper implements the ontology-based translation between service description standards. In top-down processing, this layer is used to translate a service request issued by a service consumer into other available description languages. In this case the translation is done each time a service request is issued by a consumer. The source registry that is responsible for managing information about service providers transfers the translated services to the particular service sources. In this approach, the actual matching of a service request with the available descriptions is done by the service provider systems. Using the translated service request, available service descriptions are compared to the consumer's requirements. If a service fits the request, the provider transfers it back to the translation layer, where it is then translated into the language used by the consumer. As a result, the consumer receives a list with service descriptions translated into his service description language.

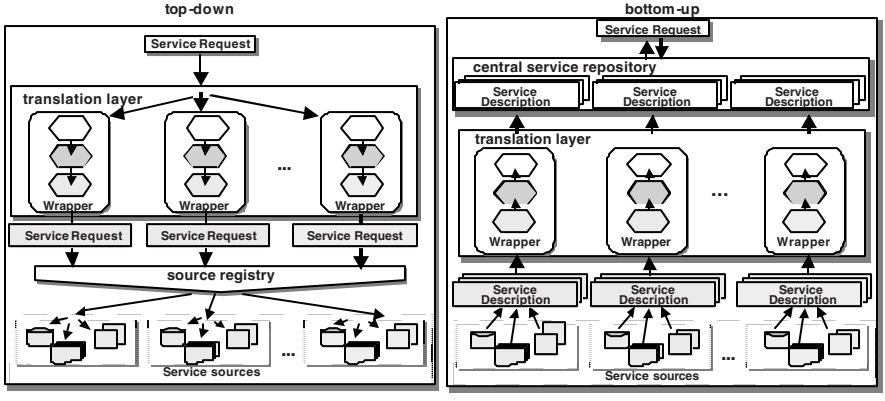

Fig. 9. top-down vs. bottom-up service request processing

In contrast, in the bottom-up approach one central service description language (CSDL) is used to store homogenous service description in a central service repository. This service description language is used to formulate service requests. A consumer can access the central repository with a corresponding interface (e.g. a web form) where he can issue his request.[1 2] The input of the consumer is then transformed into a CSDL-document that is transferred to the central service repository. Afterwards, it is mapped to

the stored service descriptions. Descriptions which are satisfying the request's requirements are then delivered to the consumer.

To publish his services, a provider sends the descriptions of the services he offers to the translation layer, where they are translated into CSDL and stored in the central service repository. Thus, each time a new service is added or an existing one is changed, it has to be translated.

Advantages and disadvantages are summarized in Table 1. Their influence on the application of the two kinds of processing a request are discussed in the following.

One disadvantage of the bottom-up request processing is the necessity of storing translated service descriptions centrally. Validation and consistency checking of service descriptions, support of service migration as well as request processing have to be provided centrally. Thus, bottom-up approach leads to a higher central administrative workload than the top-down approach where service descriptions are maintained locally and requests are processed at the service sources. In addition, in a top-down architecture, a consumer can use his own description language to post a request to the platform, while the bottom-up processing requires a fixed central language and special support for the formulation of requests must be offered.[13]

Table 1. Comparison of bottom-up and top-down request processing

Criteria	Bottom-up	Top-down
workload during build-time	very high	no
workload for processing one request	little	high
translation quality	high	low
support of language variety	medium	high
central maintained functionality	high	little
workload changing service descriptions	high	no
workload for adding new descriptions	high	low

As mentioned, in case of the bottom-up processing every service description must be translated in advance without knowing if it ever will be requested by a consumer. However, the translation of a service description is necessary only when a service is published to the platform or a service change requires an update of its description. Once the translation has been done, a request does not require an additional translation.

In the top-down approach, each request as well as the received service descriptions must be translated on the fly. Note, however, that only the service description satisfying the requirements have to be translated and thus, the overall translation workload will be limited.

Summarizing, a bottom-up approach requires much effort during build-time but offers a more efficient and accurate request processing at runtime. It can be used for ap-

12. To offer an interface which is as much as possible independent of a specific language and therefore understandable for a broad range of consumer, it should be based on the common concepts and terms captured in the SDL-ontology.

13. Beside special interfaces, it is also possible to add another translation layer in order to translate heterogeneous service request, which would increase the work necessary to process one request.

plications where the service offers are rather stable. Furthermore, in order to avoid unnecessary translations, the range of services published should be limited to a specific area. Accordingly, a bottom-up architecture can provide best support for closed, long-term enterprise networks (e.g. to enable inter-organizational cooperation), whereas the top-down architecture can better handle dynamic scenarios where service offerings change frequently. Furthermore, the support of different service request languages is more convenient for applications with a wider, non-limited range of service consumers and providers (e.g. for common service directories).

9 Conclusion

In this paper we have introduced an approach for a semantically correct translation of service descriptions between existing description standards. Our focus is on resolving semantic heterogeneity during the processing of service requests. We advocate formal ontologies to reslove semantic heterogeneity during the translation process.

Two major approaches for an architecture of a system for searching and translating services are presented: bottom-up and top-down. While both of them are generally suited to fit the requirements presented, we argue that each of them has its own advantages and disadvantages for specific application scenarios.

At the moment we focus only a translation of hierarchical relations of elements that are common in the regarded standards. However, XML does not provide semantical information about the nature of the relations between elements of a standard. A model that allows the descriptions of the underlying semantics of such relations (e.g. ER or UML diagrams) has to be used. Therefore, in order to cover also other relations between the elements or concepts (e.g., the inversion of a relation), we will extend our approach and consider conceptual schemas as base for building a translation structure.

References

1. Ankolekar, A.; et al.: DAML-S: Web Service Description for the Semantic Web, in: Proceedings of ISWC 2002, Sardinia, Italy, June 9-12, Springer (2002) 348-363
2. Arkin, A.: Business Process Modeling Language (BPML). BPMI.org Business management initiative. November (2002) http://www.bpmi.org/
3. Bussler, C.; Fensel, D.; Maedche, A.: A Conceptual Architecture for Semantic Web Enabled Web Services, In: SIGMOD Record, Special Issue Semantic Web and Databases (2002)
4. Cardoso, J., Sheth, A.: Semantic e-Workflow Composition. Technical Report LSDIS Lab, Computer Science Department, University of Georgia (2002) http://lsdis.cs.uga.edu/lib/2002.html
5. Ciocoiu, M., Nau D.: "Ontology-Based Semantics". Proceedings of KR'2000, Breckenbridge, Colorado, April 12-17 (2000)
6. cXML, http://www.cxml.org
7. Deitel, H.M., Deitel, P. J., DuWaldt, B., Trees, L. K.: Web Services - A Technical Introduction. Prentice Hall, New Jersey (2002)
8. ebXML, http://www.ebxml.org/specs/index.htm

9. Fernandez, M., Gomez-Perez, A., Juristo, N.: METHONTOLOGY: from ontological art to ontological engineering. In: Workshop on Ontological Engineering AAAI'97, Stanford USA (1997)
10. Guarino, N. and Welty, C.: Identity, unity, and individuality: Towards a formal toolkit for ontological analysis. In: Horn, W. (eds.): Proceedings of ECAI-2000 The European Conference on Artificial Intelligence, Amsterdam. IOS Press (2000)
11. Gruninger, M., Fox, M.: Methodology for the Design and Evaluation of Ontologies. Proceedings of IJCAI'95, Workshop on Basic Ontological Issues in Knowledge Sharing, April 13 (1995)
12. Hakimpour, F., Geppert, A.: Resolving Semantic Heterogeneity in Schema Integration: An Ontology Based Approach. In: Welty, C.; Smith, B. (eds.): Proceedings of International conference on Formal Ontologies in Information Systems FOIS'01. ACM Press (2001) 296-308
13. Klein, M.; Bernstein, A.: Searching for services on the semantic web using process ontologies, in: Proceedings of The First Semantic Web Working Symposium. Stanford (2001)
14. Obrst, L., Wray, R. E., Liu, H.: Ontological Engineering for B2B E-Commerce. In: Welty, C., Smith, C. (eds.): Proceedings of International conference on Formal Ontologies in Information Systems FOIS'01. ACM Press (2001) 117-126
15. Singh M. P., Cannata P. E., Huhns M. N., Jacobs N., Ksiezyk T., Ong K., Sheth A. P., Tomlinson C., Woelk, D.: The Carnot heterogeneous Database Project: Implemented Applications. Distributed and Paralled Databases, Vol. 5, No. 2 (1997) 207-225
16. RosettaNet consortium, http://www.rosettanet.org
17. Schlenoff, C., Ciocoiu, M., Libes, D., Gruninger, M.: Process Specification Language: Results of the First Pilot Implementation. In: Proceedings of the International Mechanical Engineering Congress and Exposition, Nashville, Tennessee, November (1999)
18. Uschold, M., King, M., Moralee, S., Zorgios, Y.: The Enterprise Ontology (1995), http://www.aiai.ed.ac.uk/~entprise/enterprise/ontology.htm
19. Workflow Process Definition Interface - XML Process Definition Language (XPDL). WFMC Document Number WFMC-TC-1025, Hampshire (2002)