periodica polytechnica

Electrical Engineering
52/1-2 (2008) 117–125
doi: 10.3311/pp.ee.2008-1-2.14
web: http://www.pp.bme.hu/ee
© Periodica Polytechnica 2008

RESEARCH ARTICLE

Detecting semantically related concepts in a SOA integration scenario

Péter Martinek / *Béla* Szikora

Received 2008-07-22

Abstract

In this paper, we present an approach to detecting semantically related concepts in a service oriented environment. This method is essential when creating collaborative business processes. Standard enterprise application systems such as enterprise resource planning (ERP), customer relationship management (CRM), supply chain management (SCM) etc. offer a lot of opportunities for application interoperability. System integrators assign a set of services from various application systems to the integration scenario. A well defined discovery process can detect these services. Nevertheless, building an operable business process requires the mapping of these services in the data schema used in the business process. This mapping results in a global understanding of relevant business concepts in the integration scenario. This paper focuses on the identification of semantically relevant concepts in different schemas in the participating services. A short overview of our integration platform and methodology is also included.

Keywords

Integration on Service Oriented Architecture (SOA) · Schema Matching · Enterprise Application Integration (EAI)

Péter Martinek
Béla Szikora
Department of Electronics Technology, BME, H-1111 Budapest Goldmann Gy. t. 3., building V2, Hungary

Introduction

Today, the World Wide Web offers a huge number of online services. It is easy to find services for booking flight tickets, to browse the products of a virtual store or to purchase goods from a Web-shop. Services can be easily integrated into applications providing complex services as a result [15]. This loose coupling of services and applications is also a relevant technology in the integration of enterprise application systems. To increase working efficiency and profits, companies need to focus on business collaboration [12]. As this collaboration is needed at intra- and inter-organizational levels, collaborative applications come from both inside and outside the companies' boundaries [14].

The vendors of enterprise application systems have created services to prepare the applications for participating in service oriented integration scenarios. The participating application system can be a service provider, a service requester or may be both in each integration scenario. In additon, the integration environment may also contain a service registry for storing the services available in different systems. This scenario creates new capabilities on existing services of applications. This architecture is called 'service oriented architecture' (SOA) [3].

In an SOA architecture, there is no need to share the whole databases of each company. Background data and working logic are hidden from service requesters. The SOA private layer secures each of the companies' (secret) data, but makes possible to create collaborative business processes to extract data from the enterprise application systems [6].

Different systems have different conventions for naming, grouping and applying their concepts. Because the services offered also rely on them, the data schema of each reflects the conventions used by the application system designers, programmers and database experts. By modeling real world concepts in the databases of application systems, a lot of additional information is lost in the abstraction process. It merely remains in the minds of the aforementioned IT-experts at the application system vendor company- This makes the identification of semantically related concepts in integration scenarios a hard task [13]. Nevertheless, building a business process requires the common understanding of the data application concepts found in input and output schemas of the underlying services.

Because the processing of input and output data is needed at every service invoke and response in the business process, the identification of semantically related concepts should be completed before undertaking the process.

The rest of the paper is organized as follows: Section 1 compares our work with other related works on the same subject. Section 2 describes our integration architecture and relevant concepts as a background for our current research. Section 3 provides for a detailed description of the methodology and algorithms for detecting semantically related concepts of services. Section 4 presents our experimental results and section 5 draws conclusions and suggests future areas of work and study.

1 Related work

There are numerous researches in literature about interoperable systems in service oriented architecture. For example, see [1, 2, 4, 7, 11, 16] .

Navas-Delgado et. al. [11] present a mediator based approach for interoperable systems. The applied architecture enables dynamic integration but also interoperability between users/applications in the Semantic Web context by means of semantic fields. To overcome the typical design and semantic problems of heterogeneous data integration, the authors rely on given directories including ontology and semantically relevant data sources as well. The wrapper generation process is also improved by applying web service technologies for publishing the web methods of wrappers. The key point of their work is the maintaining of semantic fields. Nevertheless, mapping between the schemas is obtained manually. The authors give neither methods nor calculations to support the identification of semantic relation between schema concepts.

Grossmann et al. [7] proffer a behavior based integration methodology for business processes. By using integration operators, the authors can create, deal and finalize compositions between them. The proposed methodology consists of a number of steps as follows:

1 Observation of the possible relationships between processes.

2 Offering integration operators independent from the identified relationship in the previous step.

3 Combining the results of 1 and 2.

4 Transforming the model based on the identified relationships by using integration operators.

The approach in this work is based only on the observed states of the processes and the behavior of participating services. This may lead to a valid transformation of processes but any integration is hardly realizable without taking into consideration the differences in the input and output data schemas of services and processes. Indeed, processes coming from different companies or application systems may use different semantic conceptualization to describe the same real world concept.

Yi et al. [16] propose XML application schema matching. They start with the semantic modeling of the schemas and then they identify the semantic relations between classes by using a similarity measure of category properties. Compatible constraints are also handled by the introduction of relaxation labeling. Although the method is very reliable (and the degree of confidence in element matching can also be determined), it is highly reliant on well defined XML schemas containing choices and a lot of other XML constructors. Services of standard enterprise systems usually do not provide their input and output data in such well formed XML schemas. In this case, the algorithm can be simplified significantly with no large impact on the confidence of the identification semantic relationships between concepts.

2 Integration architecture and methodology

The aim of our integration architecture is to support the realization of business application integration, defining cross-application processes. Cross-application processes consist of a set of services. The connection between these services is defined by dependencies, which determine the right order when invoking these services. For example, in a process of purchasing a flight ticket, the flight reservation service should be invoked earlier than the service responsible for the on-line payment.

The vendors of enterprise application systems have created services in their systems to prepare the way for participating in such integration scenarios. Unfortunately, the services offered are usually confined to descriptions containing only technical information. This information is enough to invoke the service with some test data, but not enough to understand its behavior. Business process composition also requires that the pre- and post-conditions should be taken into consideration when the given services are invoked. Our architecture offers tools for enterprise application developers to attach such information to service descriptions. Furthermore, it is possible to label a service operation upon the identified functionality of services. A business ontology serves as a common reference to the labeling [8].

Moreover business analysts are able to define templates for business processes. These consist of virtual services and their dependencies. Virtual services are described as the required functionality using the common concepts of our ontology.

Attaching additional information about service behavior to the service descriptions is a precondition and is stored in our service repository. During the service discovery stage, we need to be able to search this registry so that an appropriate service could be assigned to every virtual service's place in the process template.

Although we have already identified existing enterprise application services to participate and defined dependencies between them, the process is not yet ready to 'go live'. Detailed information about the schema of input and output data of the services is also necessary in run-time [9].

Real world concepts are described in the databases of enterprise applications. As explained earlier, the schema of the same real concepts may differ in the applications developed by different vendors. Services applied in composed processes rely on their own schema in their input and output data. To create mappings between them, semantic relationships between their concepts must be known. Only concepts representing the same real world concept can be mapped to each other. For example, if a service requires the name of a customer as an input, the address of the customer provided by another service cannot be used. Complex services often provide a set of concepts as output, so that we first must identify the real world concepts described by them to be able to apply them as an input into another service of the process.

Creation of the mapping of each service input to each service output in an integration scenario requires enormous computational capacity. Furthermore, this knowledge base is also hard to maintain. If a new service is added to the integration scenario, the mapping to every other service has to be defined [10].

Defining a global schema can reduce the complexity of the system. This global schema covers all possible real world concepts in the integration scenario. The services are mapped only to the global schema concepts, and the communication between services in processes is done on the level of global schema concepts. The mapping of (virtual) services on the process level is no longer a complex issue because same real world concepts are represented by the same concepts in the global schema.

Mapping the relationship between services' input and output concepts to the global schema is the focus of our paper. One important step of the mapping is the identification of semantic relationships between the concepts of services and the concepts of the global schema.

The input and output of a service is a set of complex types. These contain attributes to describe their different properties. Some attributes are simple elements, while others connect complex types to attributes. For example, a person complex type may have some simple elements such as name, age or sex and also may have an Address attribute which attaches an address complex type to the person. The address complex type may also have further simple and complex attributes so the structure of complex types can be described by a directed graph.

The global schema is structured as a directed graph too. Consider in our example that the graph of the global schema and the graphs of the services are also acyclic.

By mapping the service's input and output to the global schema, we should be able to determine the semantic relationship between complex types of the service and complex types of the global schema. Semantic relationships can be divided into four categories [5]:

1 Equivalence: two complex types are equivalent if they represent the same real world concepts.

2 Inclusion: complex type A includes complex type B but only if A represents all real world concepts represented by B.

3 Overlap: complex type A and complex type B are overlapping if complex type A represents some real world concepts of B, but A does not include B, and B does not include A.

4 Disjoint: complex type A and complex type B are said to be disjoint if complex type A does not represent any real world concept represented by complex type B.

Complex types of global schema that are disjointed from other complex types of service cannot be assigned to each other during the mapping. Complex types in relationships described in 1-3 above may be contained by the mapping between the global schema and the service.

In the next chapter we introduce an automatic method to detect semantic relationships between the complex types.

3 Detecting Semantic Relations

3.1 Definitions and background

Services provide a uniform interface for the integration. As already mentioned in the previous section, every service provides a set of complex types as an interface for the mapping. In our work these sets of complex types are handled as directed acyclic graphs (DAG). (Note that a schema containing a directed circle can be transformed to an equivalent one which does not contain any circle.) The global schema defined in this paper is also described by a DAG.

To enable the services to participate in real processes they have to be mapped to the global schema. This means, that the definition of data transformations must be presented. There are two transformations to each service:

1 The downcast transformation maps semantically relevant complex types of the global schema to the input of the service. The transformation allows us to invoke the service using concepts of our common reference (Ontology). At every service invocation, the input data described by ontological concepts are transformed into the input schema of the service.

2 The up cast transformation maps semantically relevant complex types of the global schema to the output of the service. The transformation allows us to provide the output of the service using concepts of our Ontology. At every service response, the output data described by the schema of the service output is transformed into the terms which are found in the Ontology.

The precondition of creating these transformations is the identification of semantic related complex types of the schemas. Transformations are well defined sets of rules between the elements of concepts. The elements of concepts are simple data types and complex data types. (See 3.2 for further details.) Due to the type of semantic relationship of complex types:

- transformation rules must be defined between the elements of the global schema complex type A and service's complex type B if A and B are equivalent,

- transformation rules probably should contain also operands from service's complex types C if global complex type A includes C, and
- transformation rules may contain operands from service's complex type D if global complex type A and D are overlapping.

Identifying the above mentioned types of semantic relationships between complex types, one possible set of operands is offered for the creation of transformation. Because the exact type of the semantic relationship between complex types is not relevant from the point of view of the creation of mapping, our approach focuses only on the identification of the corresponding complex types. So instead of determining the type of the semantic relationship we calculate a nominal value called semantic distance (see later in section 3.3) between the complex types.

Please note, that from the point of view of the semantic relationship detection, it is unimportant whether the complex type of the service is from the input or from the output. Hence in the rest of the paper, we will simply regard to them as "complex types of the service".

Let S_1,S_2,$S_3 \ldots S_n$ denote the services in the integration scenario, willing to participate in composed processes. Our proposed methodology is as follows:

1 Identify the semantic related complex types between the service S_1 and the global schema.

2 Create mapping between them.

3 Repeat step 1 and 2 with the service S_{k+1} until k=n.

This methodology ensures that corresponding services of the integration scenario are ready to participate in the business processes. In the rest of this paper we focus on step 1.

3.2 Characterizing of services

Enterprise application services are typically described by Web Service Description Language (WSDL) files. This contains the input and output data schema in the form of XML schemas. These schemas are not sufficient to capture real world semantics. To compare the semantic relevance of complex types of these schemas, they should be enriched with additional information and characterized by given identifiers. The enrichment and addition of these identifiers are processed on the level of complex types. (For the sake of simplicity we express our global schema also as an XML schema.)

Our characterizing identifiers can be seen below:

- The name of the complex type.
- The list of some additional terms. To describe the connection between a complex type and the modelled concepts of the real world, IT experts of enterprise application vendors may attach further terms to complex types. This can be envisaged as an extension of the name of the complex type. It can also help service integrators overcome some semantic relevance detection problems like names in different languages or synonyms.
- The list of attributes. This is expressed by the name of the attribute and the data type connected to it. As already mentioned in the previous section, this data type can be a simple element or another complex type.

Moreover, the complex types are modeled by a tuple: <N, T, A> where N is the name of the complex type, T is the list of associated terms and A is the list of attributes with their names and connected data types. For example, consider a service containing the following complex types:

```
<xs:complexType name="person">
<xs:sequence>
<xs:element type="xs:string" name="surname"> </xs:element>
<xs:element type="xs:string" name="lastname"> </xs:element>
<xs:element type="xs:int" name="age"> </xs:element>
<xs:element type="xs:string" name="idnumber"> </xs:element>
<xs:element type="xs:string" name="mothersname"> </xs:element>
<xs:element type="xs:string" name="placeofbirth"> </xs:element>
< /xs:sequence>
< /xs:complexType>
```

In our model this is described as follows: <person; human, man, woman, employee; surname(string), lastname(string), age(integer), idnumber(string), mothersname(string), placeofbirth(string)>.

In the next section we present the algorithm for detecting the semantic relationships between complex types described in this form.

3.3 Relationship of complex types

Let C_1,$C_2 \ldots C_n$ denote the complex types of the schema of the service and G_1,$G_2 \ldots G_m$ the complex types of the global schema where n is the number of complex types founded in the service's schema and m is the number of complex types in the global schema. For each complex type C_n we try to find a set of complex types G_m which in some way are semantically related to C_n. Semantically related complex types of G_m probably represent the same real world concept as C_n. In other words the semantic distance between C_n and the identified set of G_m is small in this case. As a result of this, the related complex types of G_m should participate in the mapping of C_n to the global schema. This means that the data transformations will be defined between the concepts in C_n and the concepts in the identified set of G_m. Let F be a function that returns a value of the semantic distance between two complex types. The semantic distance between complex types C_i and $G_j$is calculated as follows:

$$F(C_i, G_j) = w_1 \cdot N(C_i, G_j) + w_2 \cdot T(C_i, G_j) + w_3 \cdot A(C_i, G_j)$$

The functions N, T and A are derived from the characterizing identifiers introduced in the previous section. N, T and A are functions calculating the similarity between two classes from the point of view of a given identifier type. There is also a weight w_i in F before the functions N, T and A representing the degree of their contribution to the final result of F.

The proper selection of weights w_i depends on a lot of circumstances (exact type of the application field, granularity of the schemas, complexity of the services, etc.). The specific value of the weights should be determined in every specific integration scenario regarding these. In the paper we introduce the values applied in our experiments. Please note that to achieve acceptable results these should be changed in other integration scenarios.

The detailed calculation method of each function is presented as follows:

$N(C_i, G_j)$ is a function comparing two complex types by their name. To determine the similarity of the names we use a syntactic method. The following definition for $N(C_i, G_j)$ is used:

$N(C_i, G_j)$ = 1 if the name of C_i is the same as the name of G_j,
= 0.5 if the name of C_i is a substring of the name of G_j or vice versa,
= 0 otherwise.

Usually the name of a complex type has not much relevance to the represented real world concept. IT experts of different business application system vendors may use different names for the complex types representing the same real world concept, and different real world concepts may have the same (or similar) names in the complex types of services. In our experiments we set the value of w_1 to 0.15.

$T(C_i, G_j)$ is a function comparing the set of terms connected to the complex types C_i and G_j. $T(C_i, G_j)$ is computed as follows:

$$T(C_i, G_j) = \frac{2 \cdot \|Term(C_i) \cap Term(G_j)\| + \|Term(C_i) \nabla Term(G_j)\|}{\|Term(C_i)\| + \|Term(G_j)\|}$$

where $\|Term(C_i) \cap Term(G_j)\|$ is the number of common terms of C_i and G_j, $\|Term(C_i) \nabla Term(G_j)\|$ is the number of terms which are not the same but one is a substring of the other and $\|Term(C_i)\|$ is the number of terms of C_i.

The addition of terms to each C_i and G_j allows us to express a little bit more about the given complex type than a simple word (the name of the complex type) would do. The IT experts of application system vendors know the real world concept represented by the given complex type of a service so they can attach some effective synonym or keyword to it. Creators of the global schema can do the same for every G_j so, if C_i represents the same real world concept as G_j they will probably have some terms that are the same. If there are no matching terms in C_i and G_j then this probably represents different real world concepts.

In addition, although the degree of relevance of the function T is more than the relevance of N, it is not too relevant in the final result of the function F. For example, granularity differences between the definitions of different data schemas influence the identification of terms: concepts representing the same real world concept but coming from schemas having different granularity may have no common terms.

In our experiments we set the value of w_2 to 0.35.

$A(C_i, G_j)$ is a function comparing the attributes connected to the complex types C_i and G_j.

$A(C_i, G_j)$ is computed as follows:

$$A(C_i, G_j) = \frac{\sum_{i,j} C(a_i, a_j)}{\|Attr(C_i)\| + \|Attr(G_j)\|}$$

where $C(a_i, a_j)$ is a function returning the degree of correlation between attributes (a_i and a_j) of complex types C_i,$G_j$and $\|Attr(C_i)\|$ is the number of attributes in C_i.

Let the function $C(a_i, a_j)$ return the value 1 if attributes a_i and a_j probably correspond, 0.5 if they may correspond and 0 if they do not correspond. Function C is calculated as follows:

a.) If both a_i and a_j have a simple (not complex) data type then:

$C(a_i, a_j)$= 1 if the name of attribute a_i is the same or is a substring of attribute a_j. AND the connected data types of a_i and a_j are the same,
= 0.5 if the name of attribute a_i is the same or is a substring of attribute a_j, but the connected data types of a_i and a_j are not the same,
= 0 otherwise.

b.) If the data type of a_i is a complex type and the data type of a_j is a simple type then:

$C(a_i, a_j)$= 0.5 if $N(a_i, a_j) > 0$ or the name of a_j is the same or is a substring of one or more terms connected to a_i.
= 0 otherwise.

c.) If the data type of a_j is a complex type and the data type of a_i is a simple type then:

$C(a_i, a_j)$= 0.5 if $N(a_i, a_j) > 0$ or the name of a_i is a the same or is a substring of one or more terms connected to a_j.
= 0 otherwise.

d.) If the data types of a_i and a_j are both complex types then:

$C(a_i, a_j)$ = $F(d_i, d_j)$ if $F(d_i, d_j)$ is already calculated, where d_i is the complex type connected to data type a_i, and d_j is the complex type connected to data type a_j
= $w_4 \cdot N(d_i, d_j) + w_5 \cdot T(d_i, d_j)$ otherwise, where $w_4 = 2 \cdot w_2$ and $w_5 = 2 \cdot w_1$ otherwise.

Tab. 1. The schema of the service

Name of the complex type	Name of the attributes: connected data types	List of associated terms
CompanyData		EnterpriseData, OrganizationalData, Public Company, Limited Liability Company
	Name: string	
	ShortName: string	
	Location: **Address**	
	Tax Office: Location string	
	Tax Number: string	
	V.A.T. Number: integer	
	Company Type: string	
	ILN Nr.: integer	
	IBAN Nr.: integer	
	Weekly working hours: real	
	Normal working days: string	
Address		Location, Permanent address, Temporary address
	Street: string	
	Country+PC: string	
	Town: string	
CustomerData		ClientData GuestData, VisitorData, ClientLocation, GuestLocation, Visitor-Location
	Name: string	
	Invoice Addr.: **address**	
	Shipping Addr.: **address**	
	Lang.: string	
	Currency: string	
	Tax Number: integer	
	V.A.T. Number: integer	
	Contact: **contact**	
Contact		CustomerContact, ClientContact, GuestContact, VisitorContact
	Tel.: string	
	Fax.: string	
	E-mail: string	
	Home Page: string	

The relevance of function A is significant within function F. The attributes of a complex type characterize it much more than its name or the connected terms. So that if there are 2 concepts, both having the same set of attributes, then they probably represent the same real world concepts. Complex types with big differences in their attributes probably represent different real world concepts. We set the value of w_3 to 0.5 in our experiment.

As the reader might recognize, functions N, T and A return values between 0 and 1. Because the sum of the weight coefficients w_1, w_2 and w_3 is 1, the return value of the semantic distance function F is between 0 and 1.

In an integration scenario, detecting the semantic relations between the complex types of services and the complex types of the global schema, the above described functions should be calculated recursively. After fixing a threshold value between 0 and 1, we can diagnose complex types that returned values for F greater than the threshold. We define these complex types as semantically related, while complex types with values under the threshold are semantically unrelated. Please note, that similar to the weights w_i, the proper value for the threshold may depend on circumstances of the specific integration scenario.

The expected consequence of our method is the following. Only semantically related complex types should be considered at the transformation of data when mapping services to the global schema.

In the next section we present the application of our method in an experimental environment.

Tab. 2. The applied sample of the global schema

Name of the complex type	Name of the attributes: connected data types	List of associated terms
Organization		Company, Enterprise, Partnership
	Name: string	
	Street: string	
	Country+PC: string	
	Town: string	
	Tax Office: string	
	Tax Number: string	
	V.A.T. Number: string	
	Company Type: string	
	IBAN: integer	
	Weekly hours: real	
	Working days: **days**	
Address		Permanent, Temporary, ContactInfo, Location, Affiliation
	Street: string	
	Country+PC: string	
	Town: string	
	Tel.: string	
	Fax.: string	
	E-mail: string	
	Home Page: string	
Customer		Buyer, Supplier, Client, Guest, CustomerContact
	Name: string	
	Invoice Addr.: **address**	
	Shipping Addr.: **address**	
	Lang.: string	
	Currency: string	
	Tax Number: integer	
	V.A.T. Number: integer	
	Customer Phone: integer	
	Customer Cell Phone: integer	
	Customer Fax: integer	
	Customer E-mail: string	
	Customer Home Page: string	
Days		Week, WorkingDays, Calendar
	Monday: bool	
	Tuesday: bool	
	Wednesday: bool	
	Thursday: bool	
	Friday: bool	
	Saturday: bool	
	Sunday: bool	

4 Experimental results

To demonstrate the workings of our methodology, we give below a short demonstration of our experiments. Table 1. shows the schema of a service participating in our demo example. The name of the contained complex types is shown in column 1. The list of attributes in each complex type with connected data types can be found in column 2. The list of associated terms is placed in column 3, in the same row as the name of the complex type is given.

Our service consists of 4 complex types as follows: *CompanyData*, *Address*, *CustomerData, Contact*. *Contact* and *Address* complex types are also connected to some attributes as data type of the attribute.

The global schema is shown in Table 2. Please note, that in this demo example, only a relevant sample of the global schema is shown. Complex types of the global schema that are not presented in Table 2 are semantically disjointed from the complex types in our service. Table 2 is structured the same way as Table 1. The samples in our global schema consist of 4 complex types: *Organization*, *Address*, *Customer* and *Days*.

First the inner functions, N, T and A are calculated. Table 3. shows the return values of the function N for each pair of the complex types of service and the global schema. Table 4 and Table 5 show the return values of functions T and A.

Tab. 3. Return values of function N

Name: Service/Global	Organization	Address	Customer	Days
CompanyData	0	0	0	0
Address	0	1	0	0
CustomerData	0	0	0.5	0
Contact	0	0	0	0

After the inner functions N, T and A have been calculated, function F can also be calculated. The results are shown in table 5. The reader can see the return value of F for each pair of complex types of the service and the global schema. The threshold in this experiment was set to 0.4. Complex types having a return value in the function F greater than this threshold should probably participate in the mappings. In other words, the transformations that define connections and operations between the attributes should be defined using the attributes of these complex types.

Tab. 4. Return values of function T

Terms: Service/Global	Organization	Address	Customer	Days
CompanyData	0.357	0	0	0
Address	0	0.5	0	0
CustomerData	0	0	0.3	0
Contact	0	0.182	0.444	0

Tab. 5. Return values of function A

List of Attr.: Service/Global	Organization	Address	Customer	Days
CompanyData	0.636	0	0.261	0
Address	0.429	0.6	0	0
CustomerData	0.316	0	0.64	0
Contact	0	0.727	0.25	0

The semantically related complex types in our example are as follows:

The *CompanyData* of the service's schema and the *Organization* of the global schema. Although the name of the two concepts are not the same, and none of them is substring of the other, evaluating the similarities between the list of attributes with function A results in a higher value, also in the function F. Viewing the complex types, it is obvious that these concepts represent the same real world concepts. In spite of the different name of the concepts and differences in the list of associated terms, our method is able to identify the semantic relation between *CompanyData* and *Organization* complex types.

The *Contact* of the services's schema and the *Address* of the global schema are also semantically related. The global schema does not contain any complex type describing the contact of a customer or a client. This information is contained by the address complex type of the global schema. Because the return value of function F is greater than the threshold, the complex type *Address* of the global schema should be attached to the complex type *Contact* by creating the mappings.

Other relationships have also been correctly identified. Obviously, *Address* complex type of the service should be mapped to the *Address* of the global schema and *CustomerData* should be mapped to the *Customer* complex type. The rest of the table shows that no other pairs of complex types have been identified as semantically related.

Tab. 6. Final results

Fin. Res.: Service/Global	Organization	Address	Customer	Days
CompanyData	**0.443**	0	0.13	0
Address	0.214	**0.625**	0	0
CustomerData	0.158	0	**0.5**	0
Contact	0	**0.427**	0.281	0

5 Conclusion and future work

This paper has presented an approach for identifying semantically related concepts in a SOA integration scenario. Detecting of semantically related complex types of services and the global schema is crucial when designing a business process. The mapping of services in the integration scenario should be processed on semantically related complex types exclusively.

The complex types of the service are enriched with some terms before they are analyzed. Our method relies on characterizing identifiers *names*, *terms* and *attributes* which help calculate the semantic distance between complex types of service and the global schema. Our method automatically detects semantically related complex types as a result.

The proposed approach is rather sensitive for the selection of the right values for weights w_i. In our experiments we always checked the results manually as well and made more iteration steps to find the proper values for the weights. This can hardly be done for larger schemas. The circumstances of a specific integrations scenario could be evaluated to determine the right values for the weights. Another solution could be the partitioning of schemas into smaller sets of complex types but this raises a lot of questions as well e.g. how to identify the smaller sets?

Although our method identifies the complex types participating in transformations, it gives no promotion for the creation of transformations. Semantic relationships between the elements of complex types will also be evaluated in future work. This can result some proposals for creating specific transformation rules.

Furthermore we probably can generate proper transformations in specific circumstances.

There is also a business ontology in our integration framework which serves as a common thesauri of service capabilites and states. Attaching a reference to its concepts provides additional information about the service which has not yet been considered in this paper. Furthermore, attached references to pre- and post- conditions of services may also provide useful information when characterizing services. Hence, a future extension of this work should consider identified capabilities, pre- and post-condition of services.

References

1 **Arroyo S, Sicilia M A, Dodero J M**, *Choreography frameworks for business integration: Addressing heterogeneous semantics Computers in Industry* **58** (August 2007), 487-503, DOI 10.1016/j.compind.2006.10.002.

2 **Barbosa M A, Barbosa L S**, *Configurations of Web Services*, Electronic Notes in Theoretical Computer Science **175** (21 June 2007), 39-57, DOI 10.1016/j.entcs.2007.03.004.

3 **Barry D K**, *Web services and service-oriented architectures, Service-Oriented Architectures and Web Services, Morgan. Kaufmann*, posted on 2003, 17-33, DOI 10.1016/B978-155860906-8/50005-6, (to appear in print).

4 **Chen M, Zhang D, Zhou L**, *Empowering collaborative commerce with Web services enabled business process management systems*, Decision Support Systems, Vol. 43, Elsevier, 2007, pp. 530-546.

5 **Chen Y**, *Integrating heterogeneous object oriented schemas*, Journal of Information Science and Engineering **16** (2000), 555–591.

6 **Decker G, Weske M**, *Behavioral Consistency for B2B Process Integration*, CAiSE 2007, LNCS 4495, Springer-Verlag, Berlin Heidelberg, 2007, pp. 81–95.

7 **Grossmann G, Ren Y, Schrefl M, Stumptner M**, *Behavior Based Integration of Composite Business Processes*, BPM 2005, LNCS 3649, Springer-Verlag, Berlin Heidelberg, 2007, pp. 186–204.

8 **Martinek P, Kerekes J, Szikora B**, *Semantically-enriched Service-Oriented Business Applications*, 29th International Spring Seminar on Electronics Technology, 2006.

9 **Martinek P, Szikora B**, *Semantic Execution of BPEL processes*, ISD 2006, 2006, pp. 361-367, DOI 10.1007/978-0-387-70761-7, (to appear in print).

10 **Martinek P, Tóthfalussy B, Szikora B**, *Semantically Described Services in the Enterprise Application Integration*, ISSE 2007, 2007, pp. 335-338.

11 **Navas-Delgado I, Roldán-García M M, Aldana-Montes J F**, *Kreios: Towards Semantic Interoperable Systems*, ADVIS 2004, LNCS 3261, Springer-Verlag, Berlin Heidelberg, 2004, pp. 161–171.

12 **Ni Q, Lu W F, Yarlagadda K D V, Ming X**, *A collaborative engine for enterprise application integration*, Computers in Industry, Vol. 57, Elsevier, 2006, pp. 640–652.

13 **Saltor F, Garcia-Solaco M**, *Diversity with cooperation in database schemata: semantic relativism*, 14th international conference on information systems, ACM, NY, 1993, pp. 247–254.

14 **Sherr A W**, *Business Process Engineering - Reference Models for Industrial Enterprises*, Springer-Verlag, Berlin, 1994.

15 **Szeredi P, Lukácsy G, Benkő T**, *Semantic Web in theory and practise*, Typotex, Budapest, 2005.

16 **Yi S, Huang B, Chan W T**, *XML application schema matching using similarity measure and relaxation labeling*, Information Sciences, 2005, pp. 27–46.