# Towards Path-Based Semantic Annotation for Web Service Discovery

Julius Köpke(✉), Johann Eder, and Dominik Joham

Department of Informatics-Systems, Alpen-Adria-Universität Klagenfurt, Klagenfurt, Austria
{julius.koepke,johann.eder}@aau.at, dominik.joham@gmail.com
http://www.aau.at/isys

**Abstract.** Annotation paths are a new method for semantic annotation, which overcomes the limited expressiveness of concept references as defined in the SAWSDL standard. We introduce annotation paths and show how annotation paths can be applied for service matching capturing the semantics of XML schemas and web service descriptions more precisely. We report some experimental evaluation of the feasibility of annotation paths for web service discovery. The experiments suggest that annotation paths appears as a promising approach for improving web service discovery.

**Keywords:** Web service matching · Service discovery · Semantic annotation · SAWSDL · Annotation paths · XML schema matching · Semantic matching

## 1 Introduction

Web service discovery aims at (semi-)automating the search for suitable web services [14]. A web service discovery systems accepts a service request (a specification of the needed web service) and a set of web service descriptions (advertisements) as input and returns a list of web service descriptions ranked by relevance for the request. There are many different approaches ranging from the structural or lexical comparison of requests and advertisements to approaches based on the explicit definition of the semantics using ontologies [14]. We specifically address the usage of external knowledge provided by semantic annotations with a reference ontology using SAWSDL [7] annotations. The W3C recommendation SWASDL (Semantic Annotations for WSDL and XML Schema) specifies a lightweight approach for the annotation of web services with arbitrary semantic models (e.g. ontologies). SAWSDL introduces additional attributes for XML Schema and WSDL documents: *ModelReferences* and *Lifting-* and *Lowering-Mappings*. *ModelReferences* refer to ontology concepts and *Lifting-* and *Lowering-Mappings* refer to arbitrary scripts that transform the inputs and output XML-data to and from instances of some semantic model (ontology). *ModelReferences* are proposed for service discovery, while *Lifting-* and *Lowering-Mappings* are proposed


S. Nurcan and E. Pimenidis (Eds.): CAiSE Forum 2014, LNBIP 204, pp. 133–147, 2015.
DOI: 10.1007/978-3-319-19270-3_9

for service invocation and only apply to the annotation of inputs and outputs formulated as XML Schemas.

Our previous work focused on the annotation of XML Schemas with reference ontologies in order to automate the generation of executable schema mappings for document transformations [8–11,15]. We could show that the expressiveness of the SAWSDL *ModelReferences* annotation is not sufficient for the generation of schema mappings when as usual in interoperability scenarios general reference ontologies are directly used for the annotation. *Lifting-* and *Lowering-Mappings*, on the other hand, suffer from their procedural rather than declarative semantics. Therefore, we have proposed an extended annotation method based on annotation paths rather than single concept annotations. Since this method already showed its usefulness for XML-document transformations [8] we expect that annotation paths can also improve web service discovery. The general hypothesis for this research is that if the annotation method allows a more precise definition of the semantics, then the precision of service matching for service discovery can be improved. More concretely we state as hypothesis that the annotation path method leads to better results in service discovery.Existing approaches for SAWSDL based service discovery such as [4,6] can partly solve the problem of non precise semantic annotations by using additional dimensions such as structure or textual similarity.

To give a first answer on this hypothesis we discusses the usage of annotation paths for web service discovery and report some experimental results. This paper is an extended version of our previous work [12].

## 2 Annotation Path Method

The SAWSDL [7] standard addresses semantic annotations for both web service descriptions and for XML Schemas. They are related since WSDL descriptions use XML Schema to define the inputs and outputs of operations. Therefore, whenever we refer to an XML Schema, schema annotation or schema matching, this also applies for annotated WSDL documents and requests.

### 2.1 Example and Motivation

In order to motivate the need for a more expressive annotation method we will first discuss some examples. The XML Schema document shown in Fig. 2 is annotated using simple concept references referring to the ontology shown in Fig. 1 using the $sawsdl : ModelReference$ attribute.

The annotated document in Fig. 1 exhibits the following problems:

- The elements *BuyerZipcode* and *BuyerStreet* cannot be annotated because the *zip-code* is modeled in form of a data-type property and not by a concept in the ontology.
- The *BuyerCountry* element is annotated with the concept *country*. This does not fully express the semantics because we do not know that the element should contain the country of the buying-party. In addition, the *SellerCountry* element has exactly the same annotation and can therefore not be distinguished.

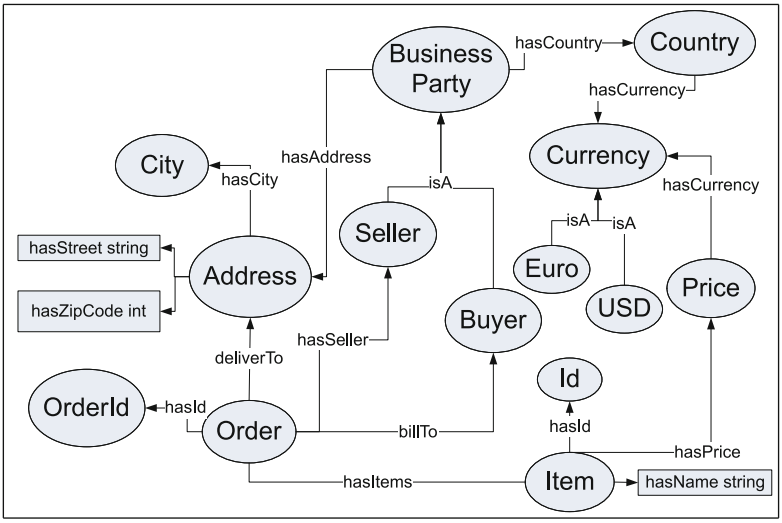


**Fig. 1.** Example reference ontology [9]

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:sawsdl="http://www.w3.org/
  <xs:element name="order" sawsdl:modelReference="/order">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="BuyerZipcode"/>
        <xs:element name="BuyerStreet"/>
        <xs:element name="BuyerCity" sawsdl:modelReference="City"/>
        <xs:element name="BuyerCountry" sawsdl:modelReference="Country"/>
        <xs:element name="SellerCountry" sawsdl:modelReference="Country"/>
        <xs:element name="Item" maxOccurs="unbounded" sawsdl:modelReference="Item">
          <xs:complexType>
            <xs:attribute name="ID" use="required" sawsdl:modelReference="Id"/>
            <xs:attribute name="Name" use="required"/>
            <xs:attribute name="Price" use="required" sawsdl:modelReference="Price"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

**Fig. 2.** Sample XML Schema with model-references [9]

– The attribute *Price* is annotated with the concept *Price*. Unfortunately, this does not capture the semantics. We do not know the subject of the price (an item) and we do not know the currency.

In the example, we have always used exactly one concept for the annotation. However, SAWSDL supports lists of concepts in the *modelReference* attribute but it does not allow specifying the relations between the concepts in this list. Therefore, this does not help to solve the shown problems. In the examples above, we have only annotated data-carrying elements. If we would in addition also annotate the parent elements in this case the *order* element we could add a bit more semantic information as the annotations of the child-elements of the order-element can be seen in the context of an order. Such an approach has limitations and drawbacks, e.g. the ambiguities between the *BuyerCountry*- and the *SellerCountry* element could not be resolved. In addition, such an approach requires a strong structural similarity between the ontology and the annotated XML Schema or service description, which we cannot assume when many different schemas or services are annotated with a single reference ontology. In addition, SAWSDL does not define that there are any relations between the annotations of parent and child elements. A solution for these non-precise annotations is the usage of a more specific reference ontology, which contains concepts that fully match the semantics of each annotated element. For example, it would need to contain the concept *InvoiceBuyerCountry* and *InvoiceBuyerZipCode*. However, enhancing a general reference ontology with all possible combinations of concepts leads to a combinatorial explosion.

### 2.2 Examples for Annotation Paths

Our annotation method [9] that solves the discussed problems is based on so-called annotation paths. An annotation paths consists of a sequence of steps, where one step can refer to a concept, an object-property or a data-type property of the reference ontology. Steps referring to concepts may have additional constraints, which are denoted in square brackets. Each such annotation path expression can automatically be transformed to an OWL concept. Therefore, matchers can exploit the more precise OWL concepts for matching. The example schema document in Fig. 3 is annotated with the proposed path expressions. We will discuss some examples and introduce the annotation paths formally in Sect. 2.3. The element *BuyerZipcode* is annotated with */**Order**/deliverTo/**Address**/hasZipCode*. The annotation of the *BuyerCountry* element is */**Order**/billTo/**Buyer**/has Country/**Country***. The steps that are marked bold refer to concepts. The other steps refer to object-properties or datatype-properties of the reference ontology. Now the *BuyerCountry* element can clearly be distinguished from the *SellerCountry* element and the elements *BuyerZipcode* and *BuyerStreet* can be annotated. The shown paths refer to concepts, object properties and datatype properties. Another requirement is to address instances of the ontology. For example, the path */**Order**/billTo/**Buyer** [Mr_Smith]/hasCountry/**Country*** defines that the Buyer is restricted to one specific buyer with the URI *Mr_Smith*.

In most cases, we assume that a simple annotation path as shown in the examples above is sufficient for an annotation. Nevertheless, there can be cases where additional restrictions are required: When using a simple path expressions as shown above the *Price* attribute of the example schema can be annotated

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:sawsdl="http://www.w3.org/ns/sawsdl" elementFormDefault="qualified" attril
  <xs:element name="order" sawsdl:modelReference="/Order">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="BuyerZipcode" sawsdl:modelReference="/Order/deliverTo/Address/hasZipCode"/>
        <xs:element name="BuyerStreet" sawsdl:modelReference="/Order/deliverTo/Address/hasStreet"/>
        <xs:element name="BuyerCity" sawsdl:modelReference="/Order/deliverTo/Address/hasCity/City"/>
        <xs:element name="BuyerCountry" sawsdl:modelReference="/Order/billTo/Buyer/hasCountry/Country"/>
        <xs:element name="SellerCountry" sawsdl:modelReference="/Order/hasSeller/Seller/hasCountry/Country"/>
        <xs:element name="Item" maxOccurs="unbounded" sawsdl:modelReference="/order/hasItems/Item">
          <xs:complexType>
            <xs:attribute name="ID" use="required" sawsdl:modelReference="/Order/hasItems/Item/hasId/Id"/>
            <xs:attribute name="Name" use="required" sawsdl:modelReference="/Order/hasItems/Item/hasName"/>
            <xs:attribute name="Price" use="required" sawsdl:modelReference="/Order/hasItems/Item/hasPrice/Price[hasCurrency/Euro]"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

**Fig. 3.** Sample XML Schema document with annotation-path method [9]

with */**Order**/hasitems/**Item**/hasPrice/**Price***. Unfortunately, this does not express the currency of the price. Since the ontology in the example has no specialized price-concept for each currency, we need to define the currency within the annotation. The correct currency of a price can be defined by a restriction on the price concept. This restriction is denoted in square brackets and expresses that the price must have a *hasCurrency* property that points to the concept *Euro*. This leads to the full annotation of the *Price* attribute: */**Order**/hasitems/**Item** /hasPrice/ **Price**[hasCurrency/**Euro**]*.

### 2.3 Formal Definition

In order to define the annotation path method we will first define the structure of the reference ontology.

**Definition 1.** ***Ontology:*** An ontology $O$ is a tuple $O = (C, DP, OP, I, A)$, where $C$ is a set of concepts, $DP$ is a set of datatype-properties, $OP$ a set of object-properties, $I$ a set of individuals and $A$ a set of axioms over $C$, $DP$, $OP$, and $I$. Each element in $C$, $DP$, $OP$, and $I$ is a string that represents the URI of the specific element. The sets $C$, $DP$, $I$, and $OP$ are disjoint.

We have already shown the string representation of annotation paths in the examples. The string representation is just a sequence of steps delimited by a slash, where each step refers to a concept or property from the reference ontology. In general, an annotation path is defined as:

**Definition 2.** ***Annotation Path:*** An annotation path $p$ of length $n$ is valid for some specific ontology $O$. It is a tuple $p = (S, t, c)$, where $p.S$ is a sequence of $n$ steps: $p.S = \{s_1, \ldots, s_n\}$, $p.t$ represents the type of the annotation path and $p.c$ refers to the representation of $p$ in form of an ontology concept (see Sect. 2.4).

Each step $s \in p.S$ is a tuple of the form ($uri$, $type$, $res$), where $s.uri$ is an URI defined in the reference ontology: $s.uri \in O.C \bigcup O.OP \bigcup O.DP$. The type of the step $s.type$ can be $cs$ (concept-step), $op$ (object-property) step or $dp$ (datatype-property-step). It is defined by the type of the referenced element in $O$:

$s.uri \in O.C \Rightarrow s.type = cs$;
$s.uri \in O.OP \Rightarrow s.type = op$;
$s.uri \in O.DP \Rightarrow s.type = dp$.

Concept-steps ($s.type = cs$) can have an optional set of restrictions $s.res$. A restriction $r \in s.res$ can restrict the concept to a specific individual ($r \in O.I$) or it can restrict the concept with an annotation path. Such a path must not contain concept-steps with restrictions in form of annotation paths itself. The first step of such a restricting path is always the restricted concept $s.uri$ (This is omitted in the string representation of the examples). If $s.res$ contains multiple restrictions they all apply to the corresponding step $s$ (logical and). The type $p.t$ of an annotation path can be either $ConceptAnnotation$, or $DataTypeProperyAnnotation$. It is defined by the last step of $p.S$:

$p.S_n.type = cs \Rightarrow p.t = ConceptAnnotation$
$p.S_n.type = dp \Rightarrow p.t = DataTypeProperyAnnotation$

**Definition 3.** ***An annotation path p of length*** $n$ ***is structurally valid iff:***

- $p.S_1.type = cs$ - The first step must refer to a concept.
- $p.S_n.type \in \{dp, cs\}$ - The last step must refer to a concept or a datatype-property.
- $\forall i \in \{1..n-1\} : p.S_i.type = cs \Rightarrow p.S_{i+1}.type \in \{dp, op\}$ - The successor of a concept-step must be an object-property or datatype-property-step.
- $\forall i \in \{1..n-1\} : p.S_i.type = op \Rightarrow p.S_{i+1}.type = cs$ - An object-property step must be followed by a concept-step.
- $\forall i \in \{2..n\} : p.S_i.type = op \Rightarrow p.S_{i-1}.type = cs$ - The previous step of an object-property step must be a concept-step.
- $\forall i \in \{1..n-1\} : p.S_i.type = dp \Rightarrow p.S_i = p.S_n$ - Only the last step can refer to a datatype-property.

### 2.4 Representing Annotation Paths as Ontology Concepts

We specify the semantics of annotation paths by representing them in form of ontology concepts that we call annotation concepts. This allows concept-level reasoning over the annotations and can consequently be used to match the annotated elements of different schemas/SAWSDL documents. Given a sequence of steps of an annotation path $p.S$, $p$, we show, how the corresponding annotation concept $p.c$ can be represented in OWL. The URI of the annotation concept is equivalent to the string representation of the sequence of steps $p.S$. Therefore, it can directly be used as a SAWSDL *Model Reference*. We illustrate the representation of annotation paths with one example of a concept annotation.

```
Class: Order/billTo/Buyer[Mr_Smith]/hasCountry/Country
EquivalentClasses(
      ConceptAnnotation and Country and inv
        (hasCountry) some
           (Buyer and {Mr_Smith} and inv (billTo) some (Order)
        )
)
```

**Listing 1.** Representation of a concept annotation path in OWL [9]

In listing 1 the OWL representation of the path */**Order**/billTo/**Buyer** [Mr_Smith]/hasCountry/***Country** is shown. The annotation path has the type *ConceptAnnotation* because the last step refers to a concept. It defines a specialization of a *country* concept. In particular a *country*, that has an inverse *hasCountry* object-property to a *Buyer*. This buyer must be an individual of the enumerated class $\{Mr\_Smith\}$ and must have an inverse *billTo* relation to an *Order*.

Obviously, such a translation can be achieved fully automatically by iterating over the steps of the path. It is required to semantically separate annotations of the type *ConceptAnnotation* from those of the type *DataTypeProperyAnnotation*. This is achieved by defining each annotation concept as a subconcept of the corresponding type.

In general, annotation concepts are constructed using qualified existential restrictions on the inverse of the connecting properties. In contrast, annotation paths that occur in restrictions on concepts are created in the opposite direction without using the inverse of the connecting properties. Qualified existential restrictions are a standard feature of OWL/OWL2 [1] and are consequently supported by current OWL reasoners.

## 3 Service Matching Based on Annotation-Paths

After introducing the annotation method, we can discuss how the annotation paths can be used for service matching. We first provide some preliminaries and then present our service-matching prototype.

### 3.1 Preliminaries

XML Schema allows reusing types and elements. This has also influence on the matching of annotated schemas. We base the matching on Expanded Schemas.

**Definition 4.** ***Expanded Schema:*** An Expanded Schema $ES$ of an annotated XML Schema $S$ is an annotated schema, where all references and type definitions of $S$ have been expanded with their definitions (targets of the references or types) and the annotations are rewritten if elements and referenced types are annotated. The result is a set of nodes that form a tree structure. Since an XML Schema can potentially have multiple root nodes there are possibly multiple Expanded Schemas for one XML Schema.

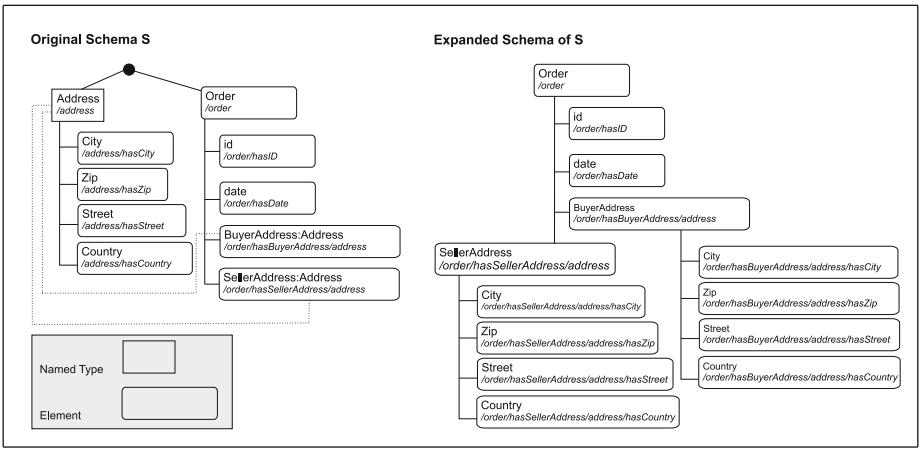


**Fig. 4.** Example of annotated XML Schema and corresponding Expanded Schema

For recursive definitions of elements and types in XML Schema the Expanded Schema would be infinite. In the current prototype we, therefore, do not support recursive definitions.

In Fig. 4, an example of an original XML Schema (left side) and its corresponding Expanded Schema is shown. The original schema has one globally defined type *address* that is referenced by the two elements *SellerAddress* and *BuyerAddress*. This requires the rewriting of the annotation paths for *address*, *SellerAddress* and *BuyerAddress*. In general, if the annotated XML Schema reuses types or elements (via type or ref properties) and both the element and the referenced element or type are annotated then the annotation needs to be constructed from the annotation path of the element and the annotation path of the referenced element. Due to the hierarchical structure of XML, this needs to be applied recursively.

Let $e$ be an element with the annotation $e.annotation$ and the XML-Type $e.type$. Let $s$ be an annotated sub-element of the XML-Type $e.type$, then the complete annotation of $s$ is defined as the path resulting from replacing the last step of $e.annotation$ with $s.annotation$.

In the example, the *country* element of the *sellerAddress* is annotated with */order/hasSellerAddress/address/hasCountry*. This is constructed from the annotation */address/hasCountry* of the *country* element that is a child of the *SellerAddress* element and the annotation */order/hasSellerAddress/address* of the *SellerAddress* element. This path combination adds structural dependencies between the schema and the reference ontology. Therefore, XML-Types should only be reused for semantically related entities, which is in accordance with good modeling practice.

In order to match the SAWSDL advertisements and requests the annotations of the corresponding Expanded Schema are added to the reference ontology. We call the resulting ontology the extended reference ontology:

**Definition 5.** ***Extended Reference Ontology:*** Given two Expanded Schemas $S$ and $T$ and the corresponding reference ontology $O$. The extended reference ontology $O'$ is defined as $O \cup S.A \cup T.A$, where $S.A$ is the set of all annotation concepts of annotations from $S$ and $T.A$ is the set of all annotation concepts from $T$. The annotation concepts are created based on the annotation path expressions as shown in Sect. 2.4. In order to separate source and target annotations different prefixes are used for the URIs of S.A and T.A.

The extended reference ontology must not contain logically invalid annotation concepts (see [11]). Therefore, as a precondition for matching all annotation concepts must be satisfiable.

### 3.2 Path-Based Service Matching Prototype

We have implemented a logics based service matcher [2] to apply the annotation path method to web service discovery. The matcher operates only on path-based annotations of the inputs and outputs of operations. No other dimensions of the service descriptions are used for matching. We assume that matching services/requests have semantically matching input and output parameters. We do not address the annotation of operations themselves. In order to rank the suitability of different web services for a request we automatically generate one XML Schema for the inputs and one XML Schema for the outputs of each operation of the advertisements and the request. These schemas are then matched and an overall confidence value for the service match in the interval [0..1] is computed. The ranking is then based on the confidence values. The matching process of the schemas operates in 4 phases:

- *Annotation Path Extraction:* The input and output schemas of each operation are transformed to an Expanded Schema where no types are reused using the COMA3 [13] library. The annotation paths are rewritten as described in Sect. 3.1.
- *Extended Ontology Generation:* The annotations are extracted from the expanded source and expanded target schema and are transformed to OWL concepts and the extended reference ontology $O'$ is created.
- *Matching and Mapping:* The XML Schemas of the request and of each advertisement are matched based on the annotations using a standard OWL reasoner (pellet). The matching function is shown is Fig. 5. Two schema elements $s$ from the source schema and $t$ from the target schema match if the annotation concept (the corresponding annotation path represented as an OWL concept) of $s$ is equivalent to the annotation concept of $t$ or if there is a subclass or superclass relation between $s$ and $t$. In case of equivalence, the confidence value of the match is defined by $\alpha$. In case of the subclass match the confidence value of the match is defined by $\beta$ weighted by the distance between the annotation concept of $s$ and $t$ in the extended reference ontology. Superclass to subclass matches are defined by $\gamma$ also weighted by the distance in the ontology. For our experiments, we used the following values

$$sm(s,t,O') = \begin{cases} \alpha & \text{if } semEqual(s.a,t.a,O') \\ \beta * / \frac{1}{ConceptDistance(s.a,t.a,O')} & \text{else if } isSubConcept(s.a,t.a,O') \\ \gamma * / \frac{1}{ConceptDistance(s.a,t.a,O')} & \text{else if } isSubConcept(t.a,s.a,O') \\ 0 & \text{else} \end{cases}$$

**Fig. 5.** Semantic matching function [8].

for the parameters: $\alpha = 1$, $\beta = 0.8$, $\gamma = 0.6$. After the confidence values are computed for each combination of elements of the source and target schema, a schema mapping is created based on the best matching elements.

– *Ranking:* Finally, an overall confidence value of each schema mapping is computed by aggregating the confidence values of the mapping elements using either min, max or avg. strategies and the advertisements are ordered descending by the overall confidence values.

## 4 Evaluation

The goal of the evaluation is to check the feasibility of the hypothesis that the annotation path method leads to better results in service discovery. Therefore, we have evaluated [2] our simple service matcher that exploits only path based semantic annotations against existing SAWSDL-based service matchers. We expect that if this simple service matcher can compete with state of the art service matchers that exploit far more aspects of a service and use advanced techniques like machine learning, then the application of annotation paths can improve service discovery.

We have annotated a subset of the SAWSDL-TC3[1] data set with our annotation path method and have evaluated our matcher against service matchers that took part in the *International Semantic Service Selection Contests*[2]. We have evaluated two scenarios. We will first discuss both scenarios and the results of our matcher in Sects. 4.1 and 4.2 and then discuss the results in comparison to state of the art matchers in Sect. 4.3.

### 4.1 Scenario 1

The goal of this scenario was to evaluate how, our simple matcher can compete against current state of the art matchers based on existing requests and advertisements of the SAWSDL-TC-3 data set. Since our approach requires different annotations, we have annotated a subset of the SAWSDL-TC3 data set with our annotation method and have evaluated our matcher using our matching method against other matchers using the original SAWSDL annotations.

[1] http://projects.semwebcentral.org/projects/sawsdl-tc/.
[2] http://www-ags.dfki.uni-sb.de/%7Eklusch/s3/index.html.

```
<xsd:complexType name="BookType" sawsdl:modelReference="/Book">
  <xsd:sequence>
    <xsd:element name="isTitled" type="Title" sawsdl:modelReference="/Book/isTitled/Title"/>
    <xsd:element name="hasType" type="Book-Type" sawsdl:modelReference="/Book/hasType/Book-Type"/>
    <xsd:element name="writtenBy" type="Author" sawsdl:modelReference="/Book/writtenBy/Author"/>
    <xsd:element name="publishedBy" type="Publisher" sawsdl:modelReference="/Book/publishedBy/Publisher
        "/>
    <xsd:element name="datePublished" type="Date" sawsdl:modelReference="/Book/datePublished/Date"/>
    <xsd:element name="timePublished" type="Once" sawsdl:modelReference="/Book/timePublished/Once"/>
  </xsd:sequence>
</xsd:complexType>
```

**Fig. 6.** Annotated Booktype fragment of the request of Scenario 1

*Request:* We have selected the *book_price_service* request from the *SAWSDL-TC3* data set. It describes one operation in its interface. This operation has an input *BookType* and an output *PriceType*. The *BookType* consists of several elements that describe a title, a book type, an author, a publisher and a publishing date. The resulting output *PriceType* consists of an amount and a currency. The book type annotated with our annotation path method as a fragment of the *book_price_service* is shown in Fig. 6.

*Advertisements and Ranking:* We have annotated a subset of the *SAWSDL-TC3* service advertisements with our annotation paths method. The *SAWSDL-TC3* contains relevance grades for each advertisement and each request. The relevance grades are 0 (not relevant), 1 (partially relevant), 2 (relevant) and 3 (highly relevant). We have annotated 5 random advertisements of each relevance grade. The selected advertisements, their defined relevance grades and the ranking computed by our simple matcher (SAPM-WS) are shown in Table 1. The 3 best ranked services of our approach also have the highest relevance grade of 3. The other two services with a relevance grade of 3 are the *monograph_price_service* - rank 6 in our result and the *printedmaterial_price_service* (rank 7). The advertisement *bookpersoncreditcardaccount_price_service* has been ranked to position 4 with a confidence value of 0.9412, while it has an expert rating of 2. The reason for this is that the input of the advertisement consists of a *book*, a *person* and a *creditcardaccount* type. While the *book* type consists of several other subtypes, *person* and *creditcardaccount* are single types and have not as much weight as the book type with its 6 sub elements in our basic aggregation function. Finally, the advertisement *sciencefictionbookuser_price_service*, ranked to rank 14 has a confidence value of 0.5000 with a predefined rating of 2. This low rating results from the *ScienceFictionBook* concept, which is only a sub concept of the *Book* concept. Furthermore, there is additionally a *User* concept in the input of the advertisement, which cannot be found in the request. Overall, we suppose that our simple matcher that operates only on annotation paths with a very simple aggregation method and static parameters already achieves good results. Tuning and more sophisticated aggregation and weighting methods will help achieve even better results.

**Table 1.** Scenario 1: SAPM-WS

| Rank | CV | Advertisement | Relevance grade |
|---|---|---|---|
| 1 | 1.0000 | book_authorprice_Novelservice | 3 |
| 2 | 1.0000 | book_reviewprice_service | 3 |
| 3 | 1.0000 | book_taxedpriceprice_service | 3 |
| 4 | 0.9412 | bookpersoncreditcardaccount_price_service | 2 |
| 5 | 0.9000 | sciencefictionbook_authorprice_service | 2 |
| 6 | 0.8000 | monograph_price_service | 3 |
| 7 | 0.6667 | printedmaterial_price_service | 3 |
| 8 | 0.6500 | novel_authorrecommendedprice_service | 1 |
| 9 | 0.5833 | printedmaterialpersoncreditcardaccount_price_service | 2 |
| 10 | 0.5500 | author_monographmaxprice_service | 1 |
| 11 | 0.5417 | romanticnovel_authorprice_service | 2 |
| 12 | 0.5000 | carbicycle_price_service | 0 |
| 13 | 0.5000 | expensivecar_price_service | 0 |
| 14 | 0.5000 | sciencefictionbookuser_price_service | 2 |
| 15 | 0.5000 | userscience-fiction-novel_price_Bestservice | 1 |
| 16 | 0.5000 | book_person_Publisherservice | 0 |
| 17 | 0.5000 | coconut_price_service | 0 |
| 18 | 0.4000 | sciencefictionbook_author_service | 1 |
| 19 | 0.3750 | SFNovelReview_service | 1 |
| 20 | 0.3333 | autocycle_price_service | 0 |

### 4.2 Scenario 2

The goal of the second scenario was to assess how our matcher competes against other matchers if the semantics cannot be expressed by simple concept annotations. In this case, matchers operating on simple concept annotations can only infer the missing semantics by exploiting other dimensions such as the structure or naming of elements. We have created a new request for this Scenario. We now search for the EURO prices of science fiction comics excluding VAT. The request consists of one operation with an input type *ScienceFictionComic* (annotated with the path */ScienceFictionBook*) and an output type *EuroPriceExcluding-VAT* (annotated with the path */TaxFreePrice* but including a sub element with a path *TaxFreePrice/hasCurrency/Euro*). While *ScienceFictionBook* and *EuroPriceExcludingVAT* can also be expressed with standard concept references, the annotation *TaxFreePrice/hasCurrency/Euro* requires annotation paths.

For the second experiment, we have annotated 15 advertisements of the SAWSDL TC3 data set. Since we now use a request that is not part of the SAWSDL TC3 data set, we asked an independent expert to provide the relevance

**Table 2.** Scenario 2: SAPM-WS

| Rank | CV | Advertisement | Relevance Grade |
|---|---|---|---|
| 1 | 1.000 | sciencefictioncomic_europricetaxfree_request | 3 |
| 2 | 0.667 | author_sciencefictionbooktaxfreeprice_service_20 | 2 |
| 3 | 0.667 | author_monographtaxfreeprice_service_20 | 1 |
| 4 | 0.567 | book_pricereviewbook_service_20 | 2 |
| 5 | 0.567 | book_price_service_20 | 2 |
| 6 | 0.567 | book_Cheapestprice_service_20 | 2 |
| 7 | 0.529 | bookperson_price_service_20 | 2 |
| 8 | 0.300 | book_taxedprice_service_20 | 2 |
| 9 | 0.267 | book_reviewprice_service_20 | 2 |
| 10 | 0.267 | author_bookprice_service_20 | 1 |
| 11 | 0.167 | author_sciencefictionbooktaxedprice_service_20 | 2 |
| 12 | 0.167 | author_sciencefictionbookmaxprice_service_20 | 2 |
| 13 | 0.167 | author_noveltaxedprice_service_20 | 1 |
| 14 | 0.000 | author_sciencefictionbookrecommendedprice_service_20 | 2 |
| 15 | 0.000 | author_bookrecommendedprice_service_20 | 1 |

grades of each advertisement for our new request. The selected advertisements, the ranking according to our matcher and the relevance grades assigned by an independent expert are shown in Table 2. The perfectly matching request was found and the ranking of the non-fully matching services is mostly in accordance to their relevance grades. However, there are some outliers: First *author_monographtaxfreeprice_service_20* with a rank of 3 and with a confidence value of 0.667 but only a relevance grade of 1 and second *author_sciencefiction bookrecommendedprice_service_20* with a rank of 14 but a relevance grade of 2 with a confidence value of 0.0. The low confidence value of *author_sciencefiction bookrecommendedprice_service_20* is the result of a processing failure of our matcher. The high confidence value of *author_monographtaxfreeprice _service_20* is due to a superclass match. Further improvements of our matcher especially tuning the parameters of the matching function should allow achieving even better results.

### 4.3 Results and Comparison

We have executed the evaluation of both scenarios with the Service Matchmaker and Execution Environment (SME2[3]), which is also used for the International Semantic Service Selection Contests. Due to the partial TC3 data set we were not able to execute all matchers. However, we could execute two major representatives iSem [4,5] and SAWSDL-MX [6]. The *iSem* matcher when applied for

[3] http://projects.semwebcentral.org/projects/sme2/.

**Table 3.** Result comparison

| Matcher | NDCG Scenario 1 | NDCG Senario 2 |
|---|---|---|
| Path-based matcher | 0.977 | 0.970 |
| iSem hybrid | 0.990 | 0.886 |
| SAWSDL-MX | 0.937 | 0.867 |

SAWSDL is a hybrid service matcher exploiting inputs and outputs and service names that employs strict and approximated logical matching, text-similarity-based matching and structural matching and automatically adjusts its aggregation and ranking parameters using machine learning. It reached the best binary precision in the contest of 2012. SAWSDL-MX is a typical representative of a hybrid matcher using logics and syntax-based matching. We have assessed the overall performance of each matcher based on the reached Normalized Discounted Cumulative Gain [3] (NDCG) which is also used in the International Semantic Service Selection Contests. The results of both scenarios are shown in Table 3. Our matcher performed more than 4 percent better than the *SAWSDL-MX* matcher. In comparison to the nearly perfect *iSEM iSEM* matcher we have achieved around 1 percent less precise matches. In the second scenario our matcher performed around 9 percent better than *iSem* and around 12 percent better than *SAWSDL-MX*.

While these preliminary results do not yet allow to draw final conclusions the annotation paths approach is promising for improving web service discovery. Our simple path-based matcher could clearly show its advantage in Scenario 2 and in Scenario 1 it could compete well with existing state of the art matchers which use far more advanced matching methods and additional aspects of service descriptions and advertisements.

## 5 Conclusions and Future Work

The annotation path method for semantic annotation was developed to overcome limitations in the expressiveness of simple concept references. We showed in some feasibility tests that already a simple implementation of an annotation path based XML Schema matcher used for comparing web service advertisements with service requests can successfully compete with state-of the art web service discovery systems. We therefore conclude that annotation paths are well suited for capturing the semantics of objects in much finer detail and that the annotation path method and matchers based on it are promising approaches to improve web service discovery. Encouraged by the promising results we plan to evaluate our annotation path based matcher with a larger data set and against additional existing matchers. Other future work is to integrate our matcher into existing state of the art matchers to gain even better results. Another direction of future work is to evaluate not only the matching precision but also the minimum amount of manual work to semi-automatically create annotation path annotations in comparison to simple concept references.

## References


1. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S. (eds.): OWL 2 Web Ontology Language: Primer. W3C Recommendation, 27 October 2009. http://www.w3.org/TR/owl2-primer/
2. Joham, D.: Path-based semantic annotation of web service descriptions for improved web service discovery. Master thesis, Alpen Adria Universitaet Klagenfurt, Universitaetsstrasse 65–67, 9020 Klagenfurt, February 2014
3. Kekäläinen, J.: Binary and graded relevance in IR evaluations-comparison of the effects on ranking of IR systems. Inf. Process. Manage. **41**(5), 1019–1033 (2005)
4. Klusch, M., Kapahnke, P.: iSeM: approximated reasoning for adaptive hybrid selection of semantic services. In: 2010 IEEE Fourth International Conference on Semantic Computing (ICSC), pp. 184–191, September 2010
5. Klusch, M., Kapahnke, P.: The iSeM matchmaker: a flexible approach for adaptive hybrid semantic service selection. Web Semant. Sci. Serv. Agents World Wide Web **15**(3), 1–14 (2012)
6. Klusch, M., Kapahnke, P., Zinnikus, I.: Hybrid adaptive web service selection with SAWSDL-MX and WSDL-analyzer. In: Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., Simperl, E. (eds.) ESWC 2009. LNCS, vol. 5554, pp. 550–564. Springer, Heidelberg (2009)
7. Kopecký, J., Vitvar, T., Bournez, C., Farrell, J.: Sawsdl: semantic annotations for WSDL and XML schema. IEEE Internet Comput. **11**(6), 60–67 (2007)
8. Köpke, J.: Declarative semantic annotations for XML document transformations and their maintenance. Ph.D. thesis, Alpen Adria Universitaet Klagenfurt (2012)
9. Köpke, J., Eder, J.: Semantic annotation of XML-schema for document transformations. In: Meersman, R., Dillon, T., Herrero, P. (eds.) OTM 2010. LNCS, vol. 6428, pp. 219–228. Springer, Heidelberg (2010)
10. Köpke, J., Eder, J.: Semantic invalidation of annotations due to ontology evolution. In: Meersman, R., Dillon, T., Herrero, P., Kumar, A., Reichert, M., Qing, L., Ooi, B.-C., Damiani, E., Schmidt, D.C., White, J., Hauswirth, M., Hitzler, P., Mohania, M. (eds.) OTM 2011, Part II. LNCS, vol. 7045, pp. 763–780. Springer, Heidelberg (2011)
11. Köpke, J., Eder, J.: Logical invalidations of semantic annotations. In: Ralyté, J., Franch, X., Brinkkemper, S., Wrycza, S. (eds.) CAiSE 2012. LNCS, vol. 7328, pp. 144–159. Springer, Heidelberg (2012)
12. Köpke, J., Joham, D., Eder, J.: Path-based semantic annotation for web service discovery. In: Nurcan, S., Pimenidis, E., Pastor, O., Vassiliou, Y. (eds.) CAiSE-Forum-DC 2014, Thessaloniki, June 2014. CEUR Workshop Proceedings, vol. 1164, pp. 81–88. CEUR-WS.org
13. Maßmann, S., Raunich, S., Aumüller, D., Arnold, P., Rahm. E.: Evolution of the coma match system. In: OM (2011)
14. Mukhopadhyay, D., Chougule, A.: A survey on web service discovery approaches. In: Wyld, D.C., Zizka, J., Nagamalai, D. (eds.) ICCSEA 2012. Advances in Intelligent and Soft Computing, vol. 166, pp. 1001–1012. Springer, Heidelberg (2012)
15. Szymczak, M., Koepke, J.: Matching methods for semantic annotation-based XML document transformations. In: New Developments in Fuzzy Sets, Intuitionistic Fuzzy Sets, Generalized Nets and Related Topics: Application, vol. 2, pp. 297–308. SRI-PAS (2012)