97

Finding Categories and Keywords in Web Services

Christian Kop, Doris Gälle and Heinrich C. Mayr

Abstract

Nowadays web services are a common way to integrate functionality in an information system, but most of the time it is very difficult to find an appropriate service. If users of web service engines do not exactly know what they want, they often browse through categories and search with keywords. This, however, depends on the knowledge of the web service owner and his/her willingness to assign such keywords. This chapter gives a proposal to provide the user with candidates for keywords and categories which are derived directly from the web service specification itself.

Keywords Web services · Web service search · Web service categorization · Keywords for web services · Web service user roles

1. Introduction

Web services are an interesting domain not only for the web community but also for enterprise applications. For small companies and private users there exist many web services on several search engines and repositories (e.g., RemoteMethods http://www.remotemethods.com/, StrikeIron http://www.strikeiron.com, and WebserviceX.NET http://www.webservicex.net/). The web service retrieval is realized in different ways [2]. Some provide keyword search or ordered lists. A common way is structuring of web services through categories.

All these approaches and techniques are based on the assumption that adequate metainformation (e.g., categories and keywords) is provided for a certain web service. To our knowledge, the web services are indexed with keywords and categorized in one of the following two ways:

- Manually: The creator or the web service provider assigns keywords and categories for a web service according to his/her knowledge of the service. This approach depends on the knowledge of a person and his/her willingness to assign such keywords.
- Automatically: Based on linguistic tools and approaches (e.g., determination of the frequency of words), the necessary keywords and categories are extracted. This approach helps to relieve the user from manually assigning the service with keywords and categories. However, if documentation is not done, no automatic processing of keywords can be performed. In fact, a large amount of documentation is needed to get useful keywords and categories. Fan et al. [4] analyzed many web services. The authors found out that most of the publicly available web services only have little documentation. As an effect, this also decreases the quality of keywords that can be found with linguistic approaches.

To conclude the most important factor of web service search, categorization and retrieval strongly depend on the willingness and documentation skills of a human. Whereas the documentation skills are

Christian Kop, Doris Gälle and Heinrich C. Mayr • Institute of Applied Informatics, Alpen-Adria-Universität, Klagenfurt, Austria.

G.A. Papadopoulos et al. (eds.), *Information Systems Development*, DOI 10.1007/b137171_97,

available, a lack of willingness to document often is the reason that other people cannot find the right web service. Therefore, we propose a third approach to find keywords for web services and to categorize them:

The keywords and categories are directly derived from the tags of a Web service specification.

This has the advantage that the creator only must concentrate on writing the web service specification. Little attention can be paid on additional documentation or manual categorization. Of course this approach also has a basic assumption–namely, that good and meaningful names for tags are used. Nevertheless, such an approach is promising since a meaningful name for tags is a minimal requirement that can be achieved in most cases. Examining examples for web services we found good and meaningful names.

The chapter is structured as follows: in the next section related work to this topic is presented. In Section 3 existing web service languages, in particular OWL-S, are briefly introduced. In the following section we will argue why web service specifications offer more opportunities for our approach than ordinary documents. Section 5 presents and describes a list of possible categories and keywords that can be derived automatically from OWL-S and WSDL documents, and in the following section we discuss the appropriateness of this data for specific user types. In the last section we will give a summary and an outlook of how this kind of categorization extraction can be optimized in practice.

2. Related Work

There is a lot of research activity in the domains of automated and collaborative categorization of documents, but there is less work to find about categorization of web services. Reading the literature in this domain, we mainly found

- literature for classifying web services,
- articles where categorization and assignment of keywords are done manually combined with clustering to relate a web service within a taxonomy, and
- articles which propose categorization from the additional documentation a web service might have.

In the field of service-oriented architectures (SOA), different ways for classifying services exist. Josuttis [7] defined three categories that are based on different SOA layers: (1) basic services in fundamental SOAs, (2) composed services in federated SOAs, and (3) process services in process-enabled SOAs. This classification shows a good overview about the different categories regarding the SOA stages of extension but we think this feature is not useful for most types of users that want to order their web services.

Josuttis presents also other approaches to classify services. A determination criterion is, e.g., the internal or external users of the service and public enterprise services. In practice also the difference between national and international services is used to group the services. Sometimes it can be reasonable to distinguish between reading and writing services. There exist also different categorizations of business services where, e.g., Allen [1] distinguishes between commodity services, territory services, and value-added services.

Since we concentrate our approach mainly on public available web services, we think that the different classification approaches in literature are not practicable for the requirements of the intended users of this kind of services.

Another approach is the community-based web service site WSFinder.com (http://wsfinder.jot.com). Registered users manually assign services to categories. They even have the possibility to add new categories, change and delete existing categories.

The benefits of Wikis, with Wikipedia on the top, are a matter of common knowledge. The use of a Wiki in this context is ambivalent because beside the benefits there exist also problems. An important challenge is the motivation of the users. They must spend their time and knowledge for adding and categorizing new web services. Every registered user can change the site so it is possible that a user cannot find the added web service because another changed the category or even deleted it. Wikipedia, for example, has some functionalities to avoid such problems (e.g., not everyone can delete an entry and an undo functionality), but these functionalities do not solve the problems completely.

To avoid these inconsistencies during manual assignment of categories, Wu et al. [12, 13] propose the use of personal hierarchies for document repositories which are then merged to a so-called "consensus hierarchy." The users are more motivated to concentrate on a hierarchy that cannot be changed from all other users.

In order to provide automatic extraction of categories and keywords, Fan et al. [4] took a snapshot of public available web services. They present their result in a hierarchy of all found services. For this purpose they used descriptions from the registries and documentation tags from the WSDL (web service description language) files. The services get clustered with the hierarchical agglomerative cluster algorithm to get a browsable service hierarchy. The results were not really encouraging because the descriptions often were very short (over 80% of the text descriptions have less then 50 words). Three years later a look at available web services does not show a better documentation situation so we think that a clustering that only base upon words from the textual descriptions cannot be useful. Hence, in addition to these approaches we propose to extract hints for categories and keywords from the different tags within a web service description itself.

3. Existing Web Service Languages

Actually there are three important web service languages available, namely WSDL, OWL-S, and WSML. WSDL is a quite common standard in practices but it focuses on syntactic features of a web service interaction and more on its technical aspects. Therefore, the W3C has developed the language OWL-S. OWL-S (web ontology language for services) [8] is an ontology for describing semantic and syntactic web service information. It builds on the web ontology language and therefore provides a machine readable and interpretable structure.

OWL-S supports the idea of automatic discovery, invocation, composition, and interoperability of web services. OWL-S descriptions consist of three main parts: profile, model, and grounding. The different parts allow a user to concentrate the search of information on specific questions. The service profile should answer the question "What does the service provide for prospective clients?" This part is used to get an overview about the service and its functionality. The service model documents present more details about the functionality of the service. So it answers the subject "How is it used?" The third component is the service grounding that answers the question "How does one interact with it?" and therefore provides syntactic information. This part also covers WSDL specifications.

Besides the OWL-S standard there exist also other description languages. For the description of syntactic information in practice, the web service definition language (WSDL) [3] is used very often. This standard is integrated in the grounding part of OWL-S. The European counterpart of OWL-S working group is the web service modeling ontology (WSMO) initiative [9]. They focus also on the description of semantic web service information. There exist mapping concepts from the web service modeling language (WSML) to OWL [10].

Since OWL-S is a standard that covers WSDL in its service grounding section and describes more than syntactic aspects of services interaction, we will mainly concentrate on OWL-S.

4. Ordinary Documents Versus Web Service Specifications

Although web services can be seen as documents, the previous section showed that they are special documents, with special features. These features can help to find categories and keywords. Table 97.1 shows some differences between usual documents and web services.

4.1. Structure

The structure of documents depends on the specific system. Generally they do not have any restrictions on the structure and contain natural language. For the publication of a web service standardized document structures, mostly based on XML, are used. Some of the tags also contain natural language, but since it is embedded in tags you can find the needed information easily. Web service documents show among other things machine interpretable data. This knowledge can be used to extract information more accurately.

Table 97.1. Differences between documents and web services.

Distinguishing feature	Document	Web service
Structure	Differs – mostly natural language – no machine interpretable data	Standardized structure in (de facto) standard, e.g., OWL-S documents represent machine interpretable semantic information
Domain	Different domains for different purposes	At least one additional default domain (namely software engineering)
Users	Depends on domain and purpose	Different types of (IT) experts who want to find and use a web service

Ordinary documents normally have no requirements on the structure and so the data are in general neither machine readable nor machine interpretable.

4.2. Domain

Like other documents, web services can belong to several different domains (medicine, bank, etc.). However, a web service in addition also belongs to the domain of "software engineering." This can be seen as a good basis for the next distinguishing feature (the users). The potential users/readers of web services can be reduced to a small set of user roles. Knowing the group of users can help to propose the right category and keyword to the right group of users.

4.3. Users

To support users of web service registries it is important to know who they are and what goals they pursue. In general document repositories it is difficult ascertaining the different user roles. In the domain of web services and software engineering, it is possible to reduce the diverse users on a manageable number of roles.

In [6] possible user groups who have dissimilar requirements on web service documentation were presented. Thus, they are also interested in different parts of the documentation. The human users were distinguished between managers, domain experts, requirements engineers, software engineers, and web service architects.

The different interests can be used to propose different keywords and categories from the web service documentation. A web service architect develops services and so he/she will potentially be interested to categorize in technical details. The software engineers integrate services in their applications so they also focus on technical parameters and use these features to organize their services. Although also requirements engineers are IT experts, their interests will not go too deep into the technical details but rather into the functional parts. So the arrangement in functional groups will be a useful strategy for their selected web services. The experts of a domain mainly concentrate on words of the given domain of the services and also use this as an ordering feature. Managers in their role as decision makers probably are more interested in economic features and also in coarse domain information to arrange their subset of web services.

5. Categories and Keywords Derived from Web Services

As has been mentioned above, one problem of categorization is that it has to be done manually by the user. This is difficult if the user is not the creator of the web service. Therefore, providing him/her with any proposals about the intended purpose and domain of the web service would be very helpful. Even if the service has been already categorized by its creator, it is possible that the intended categorization system of the user does not match with the categories of the creator.

Since web service specifications are semi-structured, it might be obvious to derive such categories from the document structure itself. In this chapter we will explain what can be derived. Therefore, we have studied the examples *CongoBuy*, *BravoAir*, *Amazon* (see e.g., http://www.daml.org/services/owl-s/examples.html, http://www.ai.sri.com/daml/services/owl-s/1.1/examples.html, respectively).

5.1. Useful Tag Information Derived from OWL-S Documents

Within the profile, process, and grounding sections of these examples we looked for categorization candidates which could be interested for users. We found out that there are a lot of them. We therefore picked out the most interested ones which we list is Table 97.2.

Table 97.2. OWL-S tag candidates for categorizations.

Keyword /category candidates	OWL-S section	Derived from pattern	Extracted examples
Service name	Profile	<profile:serviceName> **BravoAir_ReservationAgent** </profile:serviceName>	*BravoAir_ReservationAgent*
Service category	Profile	<profile:serviceCategory>... <profile:value **Airline reservation services** </profile:value> </profile:serviceCategory>	*Airline reservation service*; *Travel Agent*
Service process name	Model	<process:CompositeProcess rdf:ID = "**BookFlight**">	*BravoAir_Process, BookFlight*
Input parameter	Profile, model	<profile:hasInput rdf:resource = "... #"**DepartureAirport**"> <process:hasInput> <process:Input rdf:ID = "**DepartureAirport**"> ... </process:Input> </process:hasInput>	*DepartureAirport, OutboundDate, PreferredFlightItinary_In*
Output parameter	Profile, model	<profile:hasOutput ...> <process:hasOutput ...>	*AcctName_Out* *ReservationID*
Name of contact person/ institution	Profile	<profile:contactInformation><actor:name>**BravoAir Reservation department** </actor:name> </profile: contactInformation>	*BravoAir Reservation department*
Role of contact person	Profile	<profile:contactInformation> <actor:title>**Sale Representative** </actor:title></profile:contactInformation>	*Sale Representative*
Participant	Model	<process:hasParticipant>	*Client*
Datatype of parameters	Model	e.g. <owl:Class rdf:ID = "**CreditCardType**">	*CreditCardType*
Process type	Model	<process:CompositeProcess>	*Atomic*, *Simple*, *Composite*
Port information	Grounding	<http:binding verb = "**POST**" /> <**soap**:binding transport = " ..."> <http:binding verb = "**GET**" />	*soap*, *POST*, *GET*
Quality criteria	Profile	<profile:rating rdf :resource = "&concepts#**GoodRating**"> within <profile:qualityRating>	*GoodRating*
Additional textual information	Profile, model	text description within<profile:textDescription> or<rdf:comment>	
Profile hierarchy	Profile, profile hierachy	<profileHierarchy: **AirlineTicketing**> Classes, Object Properties, subClassOf Specifications in the Profile hierarchy File	*AirlineTicketing Ecommerce, CommercialAirlineTravel, Product*

As can be seen from the examples in Table 97.2, a system that provides the users with categories and keywords from the OWL specification itself has to deal with two main challenges: the information is extracted from a specific OWL-S tag directly or from the values of certain tags or tag parameters. Examples for the first class are *CompositeProcess, soap*: *http*, *and profileHierachy:AirlineTicketing*. Examples for the latter are *Airline Reservation Agent*, *BravoAir_ReservationAgent, and CreditCardType*. Furthermore, it can happen that the extracted candidate for the category has a natural language style (e.g., *Airline Reservation Agent*) or it is artificial (e.g., *CreditCardType*). If it has a natural language style, nothing is left to do. However, if it is artificial, we propose to use the evaluation and verbalization strategy introduced in [5].

Finally, the importance of the examples has to be decided. We therefore propose two kinds of metainformation which were already mentioned (categories and keywords):

- (Domain) categories: help to rank the service within a service taxonomy.
- Keywords: help to index the service according to further "dimensions."

We will briefly now go through the categorization candidates, describe them and relate them to one of the metainformation items. The candidates *Service name* and *Service category* and *Process name* describe the service itself and the context of the service. Thus these values can be classified as categories.

We propose that *input parameter*, *output parameter*, *name of contact person/institution*, *role of contact person*, *participant*, *process type*, *port information*, *quality criteria, and data type* belong to keywords since they do not specify the semantics of the service itself but dimensions of features to which the descriptions of the service can be refined. The port information can be taken from WSDL description(s) referenced in the grounding.

We also propose to use textual description which can be found in tags like "textDescription," "comment" to get further categorization concepts. Linguistic tools like taggers (e.g., QTAG [11]) must be used to categorize the words appropriately and filter out stop words (e.g., *service*, *web*, *string*) Afterward verbs and nouns can be used as candidates for categorization (see also: [4]).

The optional *profile hierarchy* (e.g., "AirlineTicketing" in<profileHierarchy:AirlineTicketing> of Table 97.2) refers to *addition profile hierarchy information* in another document. It was introduced by the W3C consortium in order to relate and position a web service in a broader context of web services (see http://www.daml.org/services/owl-s/1.1/ProfileHierarchy.html). Without any further information "AirlineTicketing" in <profileHierachy> can already be used as a candidate for a category since it provides a more general notion for the service. Stepping into the profile hierarchy document itself gives more *additional information* that can be used to find further categories and keywords. The profile hierarchies used in the BravoAir as well as in the CongoBuy examples describe top ontologies and taxonomies for commercial domains. Figure 97.1 shows an excerpt.

In this excerpt of the ontology *AirlineTicketing* is a sub class of E-commerce. Once again such a sub class specification can be used to extract the next broader context of the service, namely E-commerce. Furthermore, within the top ontology relationships to other classes can be specified. The object property

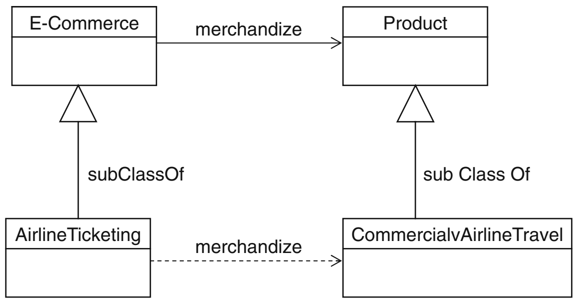

Figure 97.1. Part of the ontology of the profile hierarchy.

specifies a relationship that an E-commerce (service) merchandizes products. Within the sub class airline ticketing, this relationship is bounded to the special product CommercialAirlineTravel. Hence the advantage of looking inside such a profile hierarchy is twofold. The super classes of a given service (e.g., AirlineTicketing and E-commerce) provide a taxonomy of categories. On the other hand, the related classes (*CommercialAirlineTravel*, *Product*) provide candidates for keywords.

5.2. Information Derived from WSDL Specifications

OWL-S actually covers the WSDL specification in its service grounding section and offers much more information of web service aspects. However, WSDL 1.1 [3] is widely used actually. We think also the recent version 2.0 will get common in practice but we concentrated our research in this case on the more popular version 1.1. Thus it is also important to know if we can get at least some of the information when we have only WSDL specifications. Looking inside the WSDL 1.1 definition, we could identify some tags which could be candidates for keywords and categories. Particularly, the following candidates from Table 97.2 can be also derived from WSDL tags: *Service name*, *Service Process name*, *Parameters (Input/Output)*, *Port information*, *Datatype of parameters*, *Additional textual information*. The **Service name** (e.g., "*BravoAirReservation_Agent*") can be derived from the tag parameter *name* within the WSDL tag <service> (e.g., *<service name="BravoAirReservation_Agent">)*. The **Service process name** (e.g., "*BravoAir_Process*") can be extracted from *<operation name="BravoAir_Process">*. The **parameters** (e.g., "*DepartureAirport*") can be found in *<part name="DepartureAirport">* within the tag <message>. However, an explicit distinction between input and output parameters is not supported. **Port information** is extracted from the same patterns which were mentioned in Table 97.2. At least simple **data types** (e.g., string, integer) can be extracted. Finally, information within the tag <documentation> can be used to find further keywords and categories.

6. Categories, Keywords, and Types of Users

The web service profile, model, and grounding focus on a certain kind of intention a reader/user might have. Depending on the placement of a certain category in the profile, model, or grounding sections, specific categories should be offered to specific kinds of users. Therefore, the types of user roles will be discussed first. Afterward the appropriateness of each of the categories, listed and motivated in Section 5.1, will be described for these types of users.

A **WS Architect** develops web services, checks how it works, and customizes them. He/she focuses more on the technical details. Hence categories derived from the grounding and model will help him/her a lot.

The **SW Engineer** integrates web services in the existing software of an enterprise. He/she will be more interested in the profile information than the WS architect but only on information related to the functionality. He/she will (must) be certainly interested in the model itself and how the web service works.

Requirements Engineers mainly focus on the question "what a system should provide." Together with the domain expert, he/she must also know if the service fits. He/she might be interested to trace back to the persons listed in the contact information in order to get more background information. He/she must also know something on "how it works," since the change from "what" to "how" is somewhat blurred. Therefore, he/she might be interested in the service profile and the service model but only a little bit in the service grounding.

Domain experts and managers are the typical end users with a minimal or no technical knowledge. There is no doubt that they focus mainly on information from the profile.

The category candidates *Service name*, *Service category*, *Profile hierarchy*, additional profile hierarchy information, and *Service process* describe the web service or part of it. Thus they implicitly tell something about the purpose and the semantics of a web service. This information is important for all kinds of users. User specific categories appear in those candidates who belong to the keywords.

Input parameter, output parameter, and *data type* information refine the categorization of a web service. Knowing the input and output parameters as well as their data types is mainly important for the requirements engineer, the software engineer, and the web service architect. On the contrary, the next two candidates for categorization (*role of contact person*, *contact person*) are mainly interesting for managers and domain experts. Here, we assume that the role of the contact person can tell something about the specific context in which the web service is used. If, e.g., a web service has a contact person whose role is "sales representative," this could be a hint that this service "supports" sales representatives. Of course in that context the role of a person can have another meaning. The contact person "sales representative" can also be named just to inform who has to be contacted if someone wants to use/buy this service. Therefore, it is necessary that the role of the contact person is only provided as candidate for categorization if

(1) it fits to the meaning "service supports this specific kind of user." This can be assured if other terms also refer into this direction.
(2) the person who has to categorize gets the information that the role of contact person can only be used for the context given in (1).

The contact information itself could be only a hint if an enterprise instead of a person is named. Such information might be interested for a manager who wants to know which enterprise is related to this service. Most often users conclude about the quality of a software from the enterprise/provider who has constructed it (e.g., a web service from enterprise X is much more trustworthy than a similar web service from enterprise Y).

The candidates *process type* and *port information* are typical examples for more technical-oriented persons (software engineer and web service architect).

The quality criterion is a subjective measure to categorize a web service according to its value. It says nothing about the meaning, purpose, or context of the web service. However, together with other categories, it can be a decision support particularly for managers but also for other types of users.

No good examples were given for the moment for *participant*. In the examples examined, the most general notions "TheClient" and "TheServer" were used. Nevertheless according to the OWL-S specification [8], participant seems to be a promising keyword candidate if more meaningful names are used. Also for the additional textual information, it is hard to decide which role profits the most. This strongly depends on the content of the additional textual documentation and where it is located within a web service specification (i.e., at the beginning as an overview description or somewhere in the middle of the specification, in the profile, process, or grounding section).

7. Conclusion and Future Work

Today the search for web services is based on keywords and categories that are assigned manually or extracted automatically from a textual documentation. This implies that someone is willing to either document the web service or to create such metainformation like categories and keywords. Since web services are semi-structured documents, a third approach was presented in this chapter. Web service elements (tags, tag parameters, values of tags, and tag parameters) were used as candidates for categories and keywords. Hence the appearance of a Web service does not strongly depend on its good documentation. Instead, a good specification is a good basis for keywords and categories.

As a next step we will combine our category and keyword extraction with social classification. As described in [13] each user can generate his/her personal categorization hierarchy. Hints for categorization will be extracted from the web service specification and offered to the user. Afterward these personal hierarchies are merged to a so-called consensus hierarchy. Applying our role-specific category and keyword extraction approach to these consensus hierarchies should result in role-specific consensus hierarchies

References

1. Allan, P. (2006) Service Orientation: Winning Strategies and Best Practices. Cambridge University Press, Cambridge.
2. Bachlechner, D., Siorpaes, K., Fensel, D., Toma, I. (2006) Web Service Discovery – Reality Check. Technical Report, DERI – Digital Enterprise Research Institute, January 17.
3. Christensen, E., Curbera, F., Meredith, G., Weerawarana, S. (2001) Web Services Description Language (WSDL) 1.1., March 2001, http://www.w3.org/TR/wsdl.
4. Fan, J., Kambhampati, S. (2005) A snapshot of public web services. *SIGMOD Record*, 34(1): 24–32.
5. Fliedl, G., Kop, C., Vöhringer, J. (2007) From OWL class and property labels to human understandable natural language, NLDB, pp. 156–167.
6. Gälle, D. Kop, C., Mayr, H. C. (2008) A uniform web service description representation for different readers, ICDS, pp. 123–128.
7. Josuttis, N. (2007) SOA in Practice. The Art of Distributed System Design. O'Reilly, Sebastopol, CA
8. Martin, D. et al. (2004): Semantic Markup for Web Services W3C Member Submission 22 November 2004, http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/
9. Roman, D., Keller, U., Lausen, H. de Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, Ch., Fensel, D. (2005) Web Service Modeling Ontology, *Applied Ontology*, 1(1): 77–106.
10. Steinmetz, N., de Bruijn, J., (2008) WSML/OWL Mapping – WSML Working Draft, January 2008, http://www.wsmo.org/TR/d37/v0.1/20080125/d37v0.1_20080125.pdf
11. Tufis, D., Mason, O. (1998) Tagging Romanian texts: a case study for QTAG, a language independent probabilistic tagger. In Proceedings of the First International Conference on Language Resources & Evaluation (LREC), 589–596.
12. Wu, H., Gordon, M. D., DeMaagd, K. (2004) Document co-organization in an online knowledge community. In CHI '04 Extended Abstracts on Human Factors in Computing Systems (Vienna, Austria, April 24–29, 2004). CHI '04. ACM Press, New York, NY, pp. 1211–1214.
13. Wu, H., Gordon, M. (2007) Collaborative structuring: organizing document repositories effectively and efficiently. *Communications of the ACM*, 50(7), July: 86–91.