Adapting SAWSDL for Semantic Annotations of RESTful Services

Maria Maleshkova[1], Jacek Kopecký[2], and Carlos Pedrinaci[1]

[1] Knowledge Media Institute (KMi)
The Open University, Milton Keynes, United Kingdom
m.maleshkova@open.ac.uk, c.pedrinaci@open.ac.uk
[2] STI Innsbruck, Innsbruck, Austria
jacek.kopecky@sti2.at

Abstract. RESTful services are increasingly been adopted as a suitable lightweight solution for creating service-based applications on the Web. However, most often these services lack any machine-processable description and therefore a significant human labour has to be devoted to locating existing services, understanding their documentation, and implementing software that uses them. In order to increase the automation of these tasks, we present an integrated lightweight approach for the creation of semantic RESTful service descriptions. Our work is based on hRESTS, a microformat for including machine-readable descriptions of RESTful service within existing HTML service documentation. We complement hRESTS by the MicroWSMO microformat, which uses SAWSDL-like hooks to add semantic annotations. Finally, we present SWEET–Semantic Web sErvices Editing Tool–which effectively supports users in creating semantic descriptions of RESTful services based on the aforementioned technologies.

1 Introduction

Currently, there is an increased use and popularity of RESTful services [1], which offer a more lightweight alternative to the SOAP- and WSDL-based approach. As a result, more and more Web applications and APIs follow the Representational State Transfer [2] (REST) architecture principles and expose functionalities in the form of RESTful Web services. This trend is supported by the Web 2.0 wave, which drives the creation of user-centered Web applications for communication, information sharing and collaboration. Popular Web 2.0 applications by Yahoo, Google and Facebook offer easy-to-use, resource-oriented APIs, which not only provide simple access to diverse resources but also enable combining heterogeneous data coming from diverse services, in order to create data-oriented service compositions called mashups. Even though RESTful services are already widely accepted, their potential is restricted by the current low lever of automation due to the lack of machine-readable descriptions and the limited applicability of the Semantic Web Services automation technologies.

Web 2.0 principles contributed significantly to the uptake of RESTful services. As a result, the value of Web 2.0 applications is not only for the direct

R. Meersman, P. Herrero, and T. Dillon (Eds.): OTM 2009 Workshops, LNCS 5872, pp. 917–926, 2009.

user, who can receive customized information, but also in exposing functionality through public REST-based APIs. However, the fact that these APIs were inspired by user-centered applications has resulted in the creation of user-oriented descriptions. Even though, the APIs are meant for machine consumption, the descriptions themselves are plain unstructured HTML documents.

The lack of machine-readable descriptions is only one of the challenges, which have to be addressed in order to provide a certain level of automation for RESTful service. The fact that the majority of existing RESTful service descriptions have no semantic annotations, also has to be taken into consideration. Semantic Web Services (SWS) are proposed as means for automating many common tasks involved in using Web services. Discovery, negotiation, composition and invocation can have a higher level of automation, when Web service are supplemented with semantic descriptions of their properties. Similarly to traditional SWS, which improve the automation of WSDL-based solutions, the adding of annotations to RESTful services can bring further automation to the process of creating mashups, in addition to the discovery and invocation tasks.

In this paper we present an integrated lightweight approach for the creation of semantic annotations of RESTful service by adapting SAWSDL [3]. For the creation of machine-readable RESTful service descriptions we use the hRESTS (HTML for RESTful Services) microformat [4]. Microformats [5] offer means for annotating human-oriented Web pages in order to make key information machine-readable, while hRESTS, in particular, enables the creation of machine-processable Web API descriptions based on available HTML documentation.

hRESTS is complemented by the MicroWSMO microformat [6], which enables using SAWSDL-like annotations to RESTful services. MicroWSMO introduces additional HTML classes, in order to enable the specification of a model reference, in addition to lifting and lowering relations for data grounding. Moreover, concrete semantics can be added by applying WSMO-Lite [7] service semantics, which enable the integration of RESTful services with WSDL-based ones. In this way, discovery and composition approaches no longer need to differentiate between WSDL and RESTful services, but rather simply be based on the integrated WSMO-Lite service semantics.

The creation of semantic RESTful services, including both hRESTS tagging and semantic annotation, is supported by SWEET: Semantic Web sErvices Editing Tool[1]. SWEET assists users in injecting hRESTS and MicroWSMO within RESTful HTML service descriptions, thus enabling the effective creation of semantic descriptions. It hides formalism complexities from the user and assists him/her in adding service metadata.

The remainder of this paper is structured as follows: Section 2, provides an analysis of common ways of describing RESTful services, while Section 3 uses this analysis to deduce a lightweight RESTful service model. Our approach for creating machine-readable service descriptions, including the hRESTS microformat and the provided tool support, is described in Section 4. In Section 5, we present our adaptation of SWASDL for RESTful services, by describing the

[1] http://sweet.kmi.open.ac.uk/; http://sweetdemo.kmi.open.ac.uk/

properties of MicroWSMO and the tool support for semantic annotations offered by SWEET. Section 6 presents an overview of related formalisms and approaches. Finally, 7 provides some detail on future work and a conclusion.

2 Common RESTful Service Descriptions

In order to be able to find services and interact with them, RESTful services, and services in general, need to be described in some way. While Web applications and Web APIs contain HTML documentation, which is understandable for human users, it needs to be extended in order to become machine-readable and -processable as well. WSDL [8] is an established standard for Web service descriptions, however, it has not found wide adoption for RESTful services and only a few such services have WSDL descriptions. Similarly, WADL [9] does not seem to be gaining acceptance among API provides and instead Web applications and APIs are usually described in textual documentation. However, in order to support the automation of RESTful services, certain key aspects of the descriptions have to be made machine-readable.

```
1 <h1>Send SMS Service</h1>
2 <p>This is a Short Message Service (SMS).<p>
3 <b>Example usage</b> http://my.test.serv.com/SendSMS.php?recipient=tel:
4 447712345678&message=messagetext&sender=User&title=TheMessage</br>
5 <h2>SendSMS Method</h2>
6 <b>recipient</b><p>List of recipent phone numbers in the format "tel:"
7 followed by an international phone number</p><br/>
8 <b>message</b><p>Content of the message.</p><br/>
9 <h2>The result is a sent SMS.</h2>
```

Listing 1. Example HTML Service Description

Our approach suggests the hRESTS microformat, which can be used to tag service properties and produce a machine-readable service description on top of existing HTML documentation. In order to identify which elements of the service description need to be marked with hRESTS tags, we analyze existing RESTful service descriptions and derive a lightweight RESTful service model, which consists of service properties required for the completion of the discovery, composition and invocations tasks.

Common RESTful service descriptions are usually given in a Web page, which contains a list of the available operations, their URIs and parameters, expected output, error message and an example. The description includes all details necessary for a user to execute the service or use it in a mashup. Based on an existing repository for Web APIs[2], which contains more than 1380 APIs, manually collected over time, we have identified three basic types of descriptions.

Listing 1 shows an example of the first type of HTML descriptions of RESTful services. These descriptions contain only one or a number of operations, which are described with their corresponding parameters and URIs, within the same

[2] `programmableweb.com`

Web page. However, a lot of Web 2.0 applications contain a plenitude of operations. In this case, the second type of descriptions, includes one main page for the service and a number of linked pages, each of which describes one operation. This results in the requirement that the microformat used to annotate the service descriptions should include not only operations, URIs, HTTP methods, inputs and outputs but should also enable the linking of multiple operations from different Web sites to one "root" service description.

activity	blogs	auth
* flickr . activity .userComments	* flickr . blogs . getList	* flickr .auth.checkToken
* flickr . activity .userPhotos	* flickr . blogs . postPhoto	* flickr .auth.getFrob

Listing 2. Example Resource-Oriented Description

Finally, the last type of RESTful service descriptions are the resource-oriented ones. Listing 2 shows parts of the `flickr`[3] API documentation, where operations are not simply listed but they are rather grouped, based on the resources which they manipulate. In the example, there are three resources (activity, blogs and auth), each of which has a set of operations. Similarly, in this case the requirements for the microformat include that separate operation descriptions can be linked to a particular resource and one RESTful service. Based on these three types of RESTful service descriptions, we derive a lightweight RESTful service model that can effectively support the creation of machine-readable service descriptions.

3 Lightweight RESTful Service Model

The service examples given in Section 2 serve as the basis for identifying key service properties present in the textual documentation. These properties are formalized in a model, which specifies the service properties used for creating machine-readable RESTful service descriptions by marking HTML with hRESTS microformat tags.

Generally, a RESTful service description consist of several *operations*, each of which is performed over a *HTTP method* and has a particular *address* (URI or a parametrized URI). Operations have *inputs* and *outputs* with corresponding data formats. In addition, outputs of one operation may link to other operations, creating a resource based "choreography". Also, a number of operations can have distributed descriptions but belong to the same service, or can have different outputs but a common set of input parameters.

In summary, the elements, which have to be identified in an unstructured HTML service description are the service body, the operations, the input and output, the address and the HTTP method. As it can be seen, this list is very similar to the one of the WSDL structure and provides the basis for a machine readable description, which can be extended and annotated with additional information such as semantic descriptions and nonfunctional properties.

[3] `www.flickr.com/services/api/`

4 Machine-Readable Descriptions of RESTful Services

In order to support the creation of machine-readable descriptions, we use the hRESTS microformat. Microformats facilitate the translation of the HTML tag structure into objects and properties. As a result, the visualization of the HTML description remains unchanged, while the microformat uses `class` and `rel` XHTML attributes to mark key service properties. In this way, the hRESTS microformat enables the creation of machine-readable RESTful service descriptions, on top of existing HTML documentation.

hRESTS consists of a number of classes based directly on the properties identified in the previous section. However, even if the hRESTS microformat enables the creation of machine-readable RESTful service descriptions, the manual annotation of an HTML service description with hRESTS is a time-consuming and complex task. In order to support users in adopting hRESTS, we have developed SWEET. SWEET is a JavaScript Web application, which requires no installation and has the form of a vertical widget, which appears on top of the currently browsed Web page. This tool overcomes a number of difficulties associated with the annotation of Web applications and API descriptions, including the fact that the HTML documentation is viewed through a browser and that the user who wants to annotate the service descriptions, usually does not have access to manipulate or change the HTML.

Therefore, we provide a browser-based annotation solution, which can be started on top of the currently viewed RESTful service description. Figure 1 shows a screenshot of SWEET. hRESTS tags can simply be inserted by selecting the relevant part of the HTML and clicking on the corresponding class node

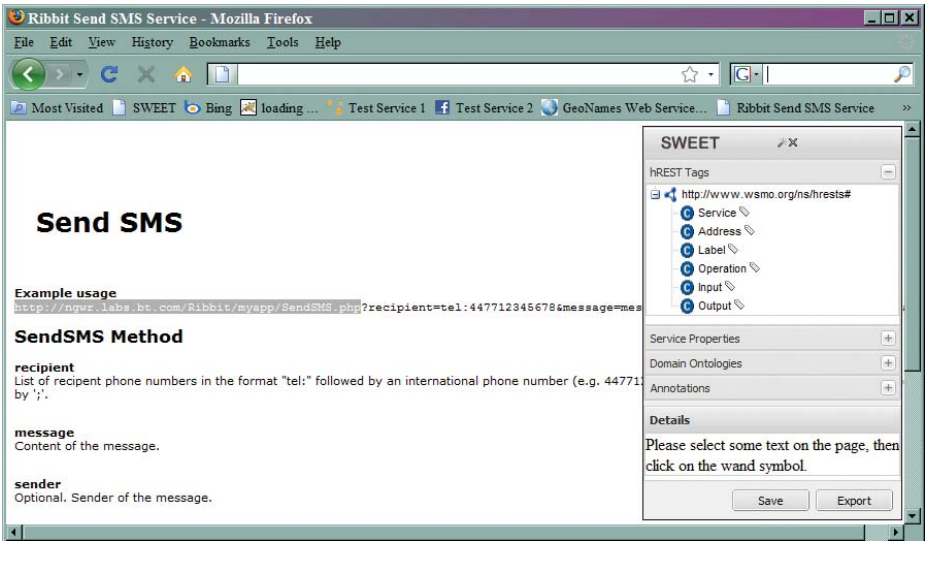

Fig. 1. SWEET: hRESTS Annotation

in the hRESTS panel of SWEET. In this way, the hRESTS annotation process is less error-prone, less time-consuming and much simpler for the user. The result is hRESTS annotated HTML, which can easily be converted into RDF ("Export" button), by using an implemented XSLT stylesheet.

Listing 3 shows the previous service description example, annotated with hRESTS by using SWEET. It visualizes the usage of the microformat annotations, as well as the structure restrictions, which exist for the different classes. The complete Web service or API description is marked by the `service` class. The service may have a `label`, which can be used to mark the human-readable name of the service. A machine readable description can be created, even if there is no service class inserted. It is sufficient that the HTML description contains at least one operation, which is marked with the `operation` class. The operation description itself includes the address where it can be executed and the HTTP method. Input and output details as well as a label can also be part of the operation.

```
1  <div class="service" id="svc">
2  <h1>Send <span class="label">SMS Service</span></h1>
3  <p>This is a Short Message Service (SMS).<p>
4  <b>Example usage</b>
5  <span class="address">http://my.test.serv.com/SendSMS.php?recipient=
6  tel:447712345678&message=messagetext&sender=User&title=TheMessage</span>
7  <div class="operation" id="op1">
8  <h2><code class="label">SendSMS Method</code></h2>
9  <span class="input">
10 <b>recipient</b><p>List of recipent phone numbers in the format "tel:"
11 followed by an international phone number</p><br/>
12 <b>message</b><p>Content of the message.</p></span><br/>
13 <h2><span class="output">The result is a sent SMS.
14 </span></h2></div></div>
```

Listing 3. Example hRESTS Service Description

The hRESTS `address` class is used to specify the URI of the operation. Similarly, the `method` class specifies the HTTP method used for the operation. The final two elements used are `input` and `output`. They are used on block markup and indicate the operation's input and output. These two elements serve as the basis for extensions given by microformats, which provide additional semantic information or details about the particular data type and schema information. Finally, user-oriented labels can be attached to the service, operation, input or output classes.

The current version of SWEET directly supports the annotation only of the first, most common type of RESTful service descriptions, where all details about a service are provided within one webpage. However, the two other types of descriptions can easily by annotated as well, by making some minor manual modifications. First, one API description can have one main page and a number of separate pages, which describe each of the operations. In order not to lose the link between the separate parts of the description, the service page is modified to include `rel="section"` after each of the links pointing to the operation Web pages and each of the operations is modified to include `rel="start"`,

which points to the main service description. As a result the `start` and `section` relations link the pages together. The second requirement, that a number of operations can be grouped based on the resource, to which they apply, is implicitly solved. All of the grouped operations have the same subset of input parameters, which can additionally be emphasized by adding semantic annotations as described in Section 5.

5 Semantic Descriptions of RESTful Services

hRESTS marks the key properties of the RESTful service and provides a machine-readable description based on the available HTML documentation. The result can effectively be used as the basis for adding extensions for supplementary information and annotations. In addition, it enables the adapting of SAWSDL [3] properties for adding semantic annotations to RESTful services because the hRESTS view of services is analogous to the WSDL one. Even though, hRESTS already provides a certain level of automation by enabling the creation of machine-readable descriptions, a higher level of automation of the discovery, composition, ranking, invocation and mediation service tasks can be archived by extending service descriptions with semantic annotations of their properties. As a result, Semantic RESTful Services (SRS) can be developed following and adapting approaches from the Semantic Web Services (SWS).

SAWSDL specifies how to annotate service descriptions, provided in WSDL, with semantic information by defining three XML attributes with equivalent RDF properties. The `modelReference` points to URIs identifying appropriate semantic concepts, while `liftingSchemaMapping` and `loweringSchemaMapping` associate messages with appropriate transformations between the level of technical descriptions (XML) and the level of semantic knowledge (RDF).We adopt these SAWSDL properties as extensions to hRESTS as part of the here described MicroWSMO microformat. Therefore, MicroWSMO represents the SAWSDL layer for RESTful services, based on top of hRESTS, instead of WSDL. Consequently, MicroWSMO has three main elements, which represent links to URIs of semantic concepts and data transformations. `model` indicates that the URI is a link to a model reference, while `lifting` and `lowering` point to links for lifting and lowering transformations.

Since the WSMO-Lite [7] ontology is used for describing the content of SAWSDL annotations in WSDL, we also adapt it for MicroWSMO. WSMO-Lite specifies four aspects of service semantics including *information model*, *functional semantics*, *behavioral semantics* and *nonfunctional descriptions*, instances of which are linked to the MicroWSMO annotations. It is important to point out that since both MicroWSMO and SAWSDL can apply WSMO-Lite service semantics, RESTful services can be integrated with WSDL-based ones. Tasks such as discovery, composition and mediation can be performed based on WSMO-Lite, completely independently from the underlying Web service technology (WSDL/SOAP or REST/HTTP).

The task of associating semantic content with RESTful service properties is even more time- and effort-demanding than the insertion of hRESTS tags.

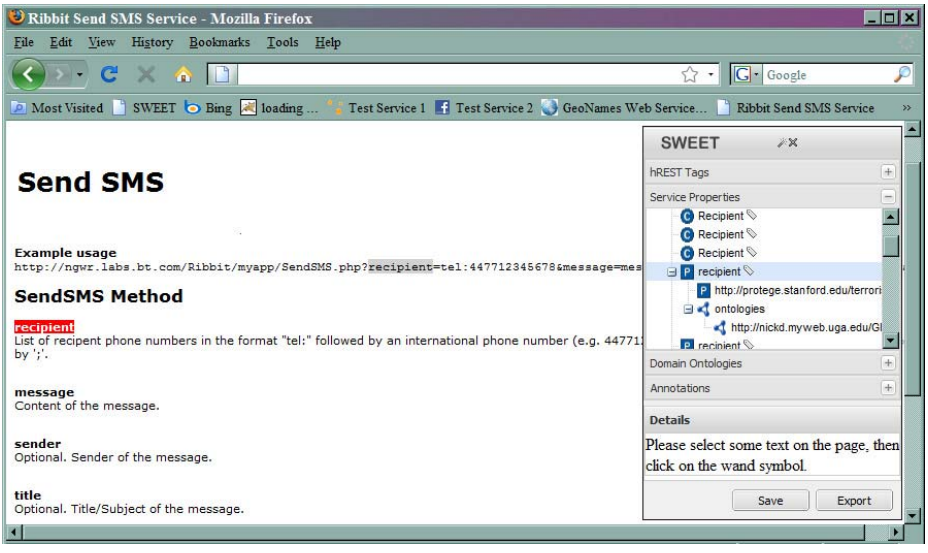

Fig. 2. SWEET: Semantic Annotation

Therefore, SWEET supports users in searching for suitable domain ontologies and in making semantic annotations in MicroWSMO. Whenever, a user wants to add semantics to a particular service property, for example, an input parameter, he/she has to select it and click on the "magic wand" symbol, which send a request to Watson [10]. Waston is a search engine, which retrieves relevant ontologies based on keyword search.

The results are presented in the `service properties` panel visualized in Figure 2. The searched for property is the root of the tree, populated with nodes that represent the found matches. In the example, *recipient* was found to be a property ("P") in an ontology located at `http://protege.stanford.edu`. If the user needs additional information in order to decide whether the particular semantic annotation is suitable or not, he/she can switch to the `domain ontologies` panel, which provides a list of all concepts and the corresponding property matches. The user can make a semantic annotation by simply selecting the property instance and clicking on one of the semantic matches in the `service properties` panel. The result is a MicroWSMO annotated HTML description, which can be saved in a repository (button "Save") or be converted into RDF (button "Export").

Listing 4 shows our example service description annotated with MicroWSMOb using SWEET. Line 4 uses the `model` relation to indicate that the service sends SMS, while line 12 associates the input parameter *recipient* with the class *Recipient*. The lowering schema for the recipient is also provided in line 13.

```
<div class="service" id="svc">
<h1>Send <span class="label">SMS Service</span></h1>
<p>This is a
<a rel="model" href="http://example.com/telecommunications/sendSMS">
Short Message Service (SMS).</a><p>
<b>Example usage</b>
<span class="address">http://my.test.serv.com/SendSMS.php?recipient=
tel:447712345678&message=messagetext&sender=User&title=TheMessage</span>
<div class="operation" id="op1">
<h2><code class="label">SendSMS Method</code></h2>
<span class="input">
<b><a rel="model" href="http://example.com/data/onto.owl#Recipient">
recipient </a>(<a rel="lowering" href="http://example.com/data/sms.xsparql">
lowering</a>)</b><p>List of recipient phone numbers in the format "tel:"
followed by an international phone number</p><br/></div></div>
```

Listing 4. Example MicroWSMO Service Description

6 Related Work

hRESTS is not the only alternative that can be used for the creation of machine-readable descriptions of RESTful services. WADL (Web Application Description Language) [9] and even WSDL 2.0 [8] can be used as description formats. They provide well-structured and detailed forms of descriptions. However, probably due to the user-centered context of Web 2.0 and of the resulting API descriptions, WADL and WSDL seem to add complexity and still the majority of the API descriptions are provided in unstructured text. We use hRESTS, which is relatively simple, easy to use, can be applied directly on the existing HTML descriptions, supports the extraction of RDF and can provide a basis, for the future adoption of dedicated formats such as WADL.

Another description approach if offered by RDFa [11]. RDFa can be effectively used for embedding RDF data in HTML. However, following the simplicity and lightweight principles perused with hRESTS, it needs to be investigated to what extent and in which use cases RDFa can be used for hRESTS. A parallel approach to RDFa would be the use of GRDDL [12] on top of hRESTS. GRDDL is a mechanism for extracting RDF information from Web pages and is particularly suitable for processing microformats.

In the area of tools supporting the semantic annotation of services, ASSAM [13] enables the annotations of services with WSDL-based descriptions. It provides user interface tools as well as some automatic recommendation components, however, it can only be used on WSDL-based descriptions and does not support RESTful services.

7 Conclusion and Future Work

Current RESTful services can be found, interpreted and invoked not without the extensive user involvement and a multitude of manual tasks. This situation can be alleviated through the creation of machine-readable descriptions. Based on such descriptions, crawlers and search engines can better find services, and

developers can better use them. Moreover, extended with semantic annotations, RESTful services can even be discovered, composed and invoked automatically, following the principles of the SWS.

In this paper, we have built on a lightweight RESTful service model, based on the hRESTS microformat that enables the tagging of key service properties and therefore supports the creation of machine-readable service descriptions. We complemented hRESTS by the MicroWSMO microformat, which adapts SAWSDL for the semantic annotation of RESTful services. Finally, we have shown the tool SWEET, which effectively supports users in creating semantic descriptions of RESTful services based on hRESTS and MicroWSMO.

Future work will include the development of additional functionalities of SWEET, which will better support users in the creation of semantic RESTful annotations. Better visualization components, such as structure and properties highlighting are planned. In addition, some work will be devoted to the automatic recognition of service properties such as operations and input parameters, so that the amount of manual work required by the user can be reduced.

The work presented here is partially supported by EU FP7 project SOA4All.

References

1. Richardson, L., Ruby, S.: RESTful Web Services. O'Reilly Media, Sebastopol (2007)
2. Fielding, R.T.: Architectural styles and the design of network-based software architectures. PhD thesis, University of California (2000)
3. Kopecký, J., Vitvar, T., Bournez, C., Farrel, J.: SAWSDL: Semantic Annotations for WSDL and XML Schema. IEEE Internet Computing 11(6), 60–67 (2007)
4. Kopecký, J., Gomadam, K., Vitvar, T.: hRESTS: an HTML Microformat for Describing RESTful Web Services. In: Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2008 (2008)
5. Khare, R., Celik, T.: Microformats: a pragmatic path to the semantic web (Poster). In: Proceedings of the 15th international conference on World Wide Web (2006)
6. Kopecký, J., Vitvar, T., Fensel, D., Gomadam, K.: hRESTS & MicroWSMO. Technical report (2009), `http://cms-wg.sti2.org/TR/d12/`
7. Vitvar, T., Kopecký, J., Viskova, J., Fensel, D.: WSMO-Lite Annotations for Web Services. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 674–689. Springer, Heidelberg (2008)
8. Web Services Description Language (WSDL) Version 2.0. Recommendation, W3C (June 2007), `http://www.w3.org/TR/wsdl20/`
9. Hadley, M.J.: Web Application Description Language (WADL). Technical report, Sun Microsystems (November 2006), `https://wadl.dev.java.net`
10. Watson - The Semantic Web Gateway: Ontology Editor Plugins (November 2008), `http://watson.kmi.open.ac.uk`
11. RDFa in XHTML: Syntax and Processing. Proposed Recommendation, W3C (September 2008), `http://www.w3.org/TR/rdfa-syntax/`
12. Clarke, F., Ekeland, I.: Gleaning Resource Descriptions from Dialects of Languages. Recommendation, W3C (September 2007), `http://www.w3.org/TR/grddl/`
13. Heß, A., Johnston, E., Kushmerick, N.: ASSAM: A tool for semi-automatically annotating semantic web services. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 320–334. Springer, Heidelberg (2004)