Serdica J. Computing **1** (2007), 229–240

Serdica
Journal of Computing

Bulgarian Academy of Sciences
Institute of Mathematics and Informatics

A PROTOTYPE OF AN EXTENSION TO THE UDDI REGISTRY ALLOWING PUBILCATION AND SEARCH BASED ON SUBJECTIVE EVALUATIONS*

Alexander Mintchev

Abstract. The current paper introduces the usage of subjective evaluations by others as a tool that can support consumers' decisions. It summarizes the features of the main UDDI registry providers and presents an extension to any UDDI registry allowing users of the registry to publish subjective evaluations for any artifact found in it and to search for artifacts, based on subjective evaluations. The paper outlines some typical business scenarios, in which the proposed extension would be useful, and introduces some areas for feature work and improvement.

1. Introduction into the subject area. In their paper "The Market for evaluations" Avery, Resnick and Zeckhauser [1] claim that "*Subjective evaluations by others are a valuable tool for consumers who are choosing which products to buy or how to spend their time. Recent developments in computer*

ACM Computing Classification System (1998): D.2.11
Key words: UDDI, web-services, evaluations.

*The paper has been presented at the International Conference Pioneers of Bulgarian Mathematics, Dedicated to Nikola Obreshkoff and Lubomir Tschakaloff, Sofia, July, 2006.

networks have driven the cost of distributing information virtually to zero creating extraordinary opportunities for sharing product evaluations". One of their conclusions concerns electronic goods offered via computer networks, whose purchase cost is zero (i.e. electronic bulletin board messages). The authors conclude that *"a market for evaluations could coordinate decisions about which people should read and evaluate particular messages.... Human effort would only be required to evaluate the messages; the market would be fully automated."*

In this paper, we introduce a high-level design of a software tool, which allows users of UDDI registries to publish their subjective evaluations about artifacts found in a UDDI registry and to discover UDDI artifacts, based on such evaluations. We expect that the conclusions of Avery, Resnick, and Zeckhauser' will also hold for the evaluations of UDDI artifacts (which are also "electronic goods offered via computer networks, with purchase price [close to] zero").

Having started in 2000 as collaboration between Microsoft, Ariba, and IBM, the UDDI project aims at speeding up interoperability and adoption of Web services by enabling enterprises to quickly and dynamically publish, discover and invoke them. This is achieved through the creation of standards-based specifications for service description and discovery. Meanwhile, business process modeling languages have emerged as an important instrument for achieving integration of business applications both within and across organizations. Some of them (i.e. BPEL) represent business processes as interactions of web-services.

While UDDI standard does provide a variety of mechanisms to classify a web-service in terms of standard and custom classification criteria that are objective, it does not provide a means for the users of a UDDI registry to publish their own, subjective evaluations about the perceived "quality" of a web-service found in the UDDI registry, and to search for a web-service based on such subjective criteria. The same holds also for the providers of UDDI registers: of all available implementations of UDDI registries that authors have investigated (e.g. registries from Systinet, Microsoft, SAP, Apache, Oracle) no one provides a possibility for a user of the UDDI registry to publish a subjective evaluation of an artifact from this registry and later to search for artifacts on the basis of such subjective evaluations.

2. Core features of UDDI registries by leading providers.

Table 1. Core features summary of the main UDDI registry providers, Source: see references [2]–[7]

Provider	TAXO CLASS	EVAL CLASSIFY	SDDTAXO	SDDEVAL
Systinet	YES	NO	YES	NO
SAP	YES	NO	YES	NO
Microsoft	YES	NO	YES	NO
Oracle	YES	NO	YES	NO
Apache	YES	NO	YES	NO
BEA	YES	NO	YES	NO

Legend:
TAXO CLASS: Has classification mechanism based on custom and standard taxonomies
EVAL CLASSIFY: Has classification mechanism reflecting subjective evaluation by users
SDDTAXO: Has service description and discovery based on TAXO CLASS
SDDEVAL: Has service description and discovery based on EVAL CLASSIFY

3. Features of the extension. Let us have a running UDDI registry. (i.e. any UDDI registry from the ones presented above). We introduce a design of an extension to it, such that:

1. It is independent of the implementation of the UDDI registry – communication between the extension and the UDDI registry goes through standard UDDI APIs (SOAP messages);

2. It allows a user of the UDDI registry to publish a subjective evaluation about the perceived "quality" of any artifact in the UDDI registry, including a web-service;

3. It uses standard requests to the UDDI registry that allow searching for UDDI artifacts in accordance to the UDDI specification, and, in addition, based on the subjective evaluation criteria mentioned;

4. The existing UDDI application is able to run independently of the extension and is completely unaware of its existence.

4. A typical use case for the extension. Let we have a business process that is implemented as a process-based service composition (i.e. as an interaction of web-services). This business process must dynamically decide which web-service provider to pick for a particular activity. UDDI registry can find appropriate web-services matching only pre-defined, **objective** criteria. What if there are several competing web-services (providers), equally suitable for becoming partners in the business process? We may need to obtain some additional (**subjective**) evaluation about the "quality" of the web- service (providers) that the business process must choose from.

5. Design of the extension. The first question to be answered is if it is possible to represent a user evaluation in terms of UDDI data strictures. UDDI has a limited set of data structures: business entity, business service, tModel, binding template and publisher assertion. A tModel, which represents a reusable abstract concept (i.e. a software artifact, a communication protocol, an address, or a taxonomy) may at first seem an appropriate solution: if a user wants to publish an evaluation regarding, say, a business service, they just publish a tModel, categorizing it as representing an evaluation, referring to the business service being subject to evaluation, and finally giving it a reference to the specific evaluation scheme used, as well as to the concrete value within this specific evaluation system (which is actually the user's own "mark") . Although possible, such a solution has 2 main drawbacks:

1. Such a tModel would not be reusable anymore (i.e. it could not be further used in any identifier or categorization scheme), as it is a concrete mapping between a unique business service and a unique users' mark valid only for this specific business service. This not only contradicts the semantics of a tModel, but also does not prevent other users by mistakenly referring it.

2. Any search of UDDI artifact based on such an evaluation shall involve a couple of independent requests, whose return type could not be automatically sorted by the UDDI registry: A user would first issue a standard UDDI find request, and then, for each item found in the result, perform additional search for tModels representing concrete evaluations of this entity. Finally, of all tModels (i.e all evaluations found in the second request), the user would have to manually sort and interpret the evaluations' semantics.

Therefore, UDDI does not offer appropriate data structure for storing user's evaluations. That is why the need arises that evaluations be stored in

their own data structures, outside the UDDI application. They will be stored, managed and retrieved by the extension to the UDDI application that would be an independent application.

The Evaluation Data Structure stored within the extension

ID	A unique ID of an evaluation
UserID	A userID of an user of an UDDI registry, as defined in the UDDI request get_authInfo
UDDI entity key	A key of an UDDI entity being evaluated, one of the business_entity key, business_service key, tModel key, or bindingTemplate key
Evaluation system key	A tModel key of the evaluation system used. In essence: a tModel key of a custom categorization scheme
Evaluation value	The user's own evaluation or "mark". It must be a valid value within the Evaluation system

Users' evaluations will be based on custom, pre-defined evaluation systems (taxonomies). As shown previously, any leading UDDI registry provider allows for classification based on custom taxonomy. This allows custom evaluation systems to be stored in the UDDI registry as custom taxonomies. Storing the evaluation scheme within the UDDI registry has the following benefits:

1. Any user of the UDDI registry can classify their own entities with the evaluation scheme given. (This is equivalent to a user giving an evaluation about their own entities.)

2. If the UDDI registry does support external value checking (as SAP UDDI registry), the evaluation scheme would be stored only in the UDDI registry, and not in the extension, which will prevent dual maintenance.

If the UDDI registry does not support external value set checking, we will need to store the evaluation scheme in the extension as well.

The extension will be an individual web-application that will be exposed as a web-service. (It will communicate thorough SOAP requests via HTTP/ HTTPS). It will have its own persistence (i.e. own with its own Data Base).

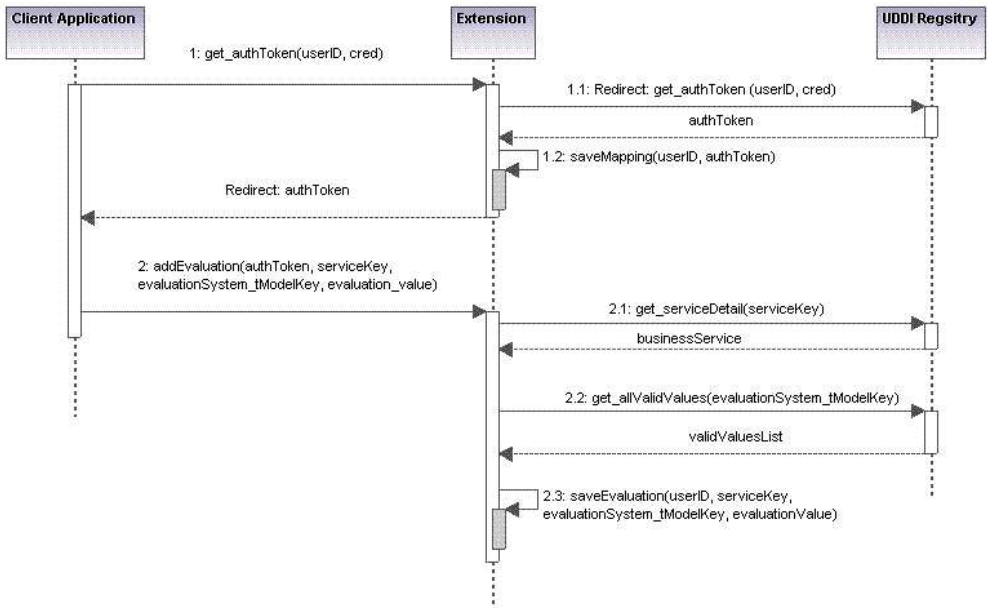
6. Communication between a client, the extension and the UDDI registry. Being a separate application, exposed as a web-service, the extension receives all SOAP requests sent by any client application to the UDDI

registry. With the exception of UDDI **find API** requests, and the **get_authInfo** request, all UDDI requests are directly dispatched to the UDDI application without any processing; the UDDI responses to these requests are, in turn, also dispatched to the client application without any processing. There is one custom API call to the extension **add_evaluation,** which allows users of UDDI registries to publish evaluations about UDDI artifacts. The **add_evaluation** request for the UDDI entity businessservice has the following structure

```
  SAMPLE REQUEST
<add_evaluation>
   <auth_info>XXXX</auth_info>
   <business_service key="uddi:goodle.de:GoogleSevice1">
      <evaluations>
         <evaluation tModelKey="uddi:evaluation:quality" value="6"
name="Excellent Quality"/>
         <evaluation tModelKey="uddi:evaluation:speed" value="5"
name="Very Good Quality"/>
      </evaluations>
   </business_service>
</add_evaluation


   SAMPLE DTD
<!--DTD -->
<!ELEMENT add_evaluation (auth_info, business_service)>
<!ELEMENT auth_info (#PCDATA)>
<!ELEMENT business_service (evaluations)>
<!ATTLIST business_service
      key CDATA #REQUIRED
>
<!ELEMENT evaluation EMPTY>
<!ATTLIST evaluation
      tModelKey CDATA #REQUIRED
      value CDATA #REQUIRED
      name CDATA #REQUIRED
>
<!ELEMENT evaluations (evaluation+)>
```

6.1. Publishing evaluations about a UDDI artifact. Here, we assume that the UDDI registry supports external value set validation via the UDDI API call get_allValidValues. If this is not the case, then the message with number 2.2 is simply not exchanged, but instead the extension cheeks internally the values of the evaluation system referred (in this case, the evaluation system is stored in the extension)



1. Client sends UDDI get_auth_Token request, providing their userID and credentials.

   1.1. The extension redirects this request to the UDDI server, and, receives authToken if userID and credentials were correct

   1.2. The extension maps the authToken received with the userID and returns back to the client the auth_Token. This is required in order for the extension to be able to identify the userID when it receives add_evaluation request.

2. The client sends add_evaluation request. The extension checks that the keys of all UDDI entities being evaluated really exists (i.e. the key of the

business service really represents an existing business service in the UDDI registry) by sending corresponding **get_** requests to the UDDI.

2.1. For each separate evaluation, the extension checks that the value provided is a valid value within the evaluation system referred by sending
get_all_valid_values requests to the UDDI registry;

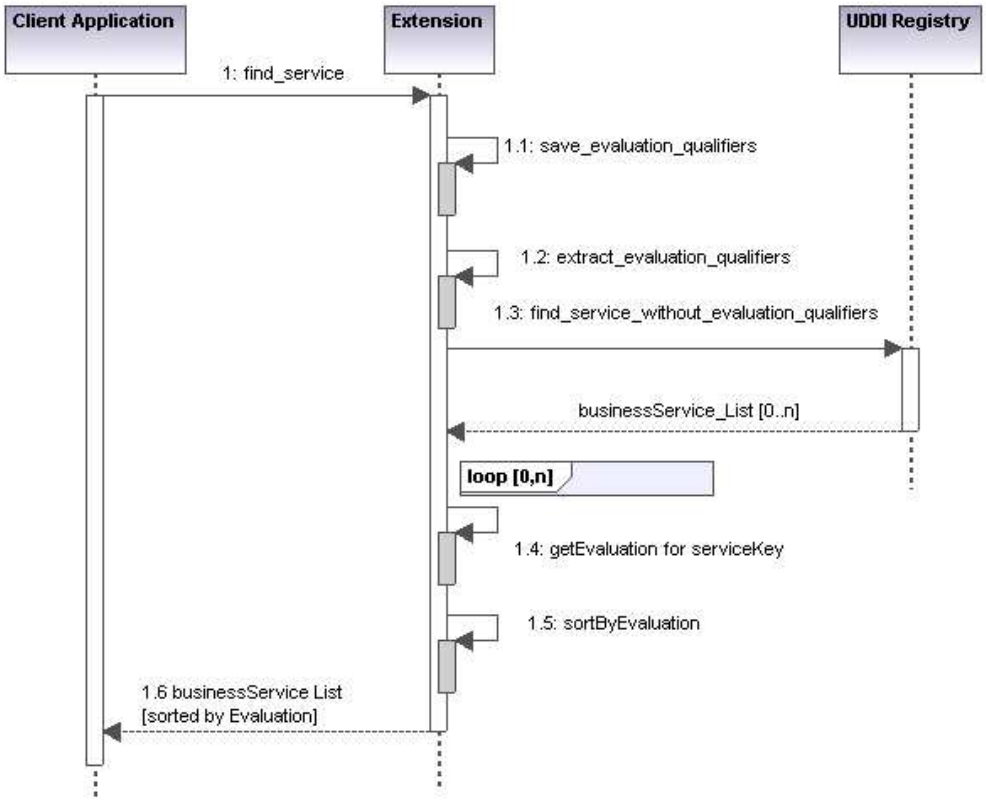
2.2. Having done all checks, the extension saves the corresponding evaluation data structures in its database.

6.2. Finding UDDI artifact based on evaluations. To find a UDDI artifact, based on evaluations (and potentially on other criteria), the client application sends a standard UDDI find request. Let the client wants to find all web-services, which are evaluated according to the evaluation system uddi:evaluation: quality. The UDDI find request look as follows:

```
<find_service xmlns="urn:uddi-org:api_v3">
   <findQualifiers>
      <findQualifier>approximateMatch</findQualifier>
      <findQualifier>uddi:evaluation:quality</findQualifier>
   </findQualifiers>
   <name>%</name>
   <categoryBag>
      <keyedReference tModelKey="uddi:6e090afa-33e5-36eb-81b7-
1ca18373f457" keyName="WSDL type" keyValue="service"/>
   </categoryBag>
</find_service>
```

When the extension receives this find_service request, it ascertains that the findQualifier **uddi:evaluation:quality** is a custom evaluation find qualifier, extracts this qualifier from the request, and composes a new request find_service' that is analogous to the original one, with the exception that all custom evaluation find qualifiers are removed. This second request looks like this:

```
<find_service xmlns="urn:uddi-org:api_v3">
   <findQualifiers>
      <findQualifier>approximateMatch</findQualifier>
   </findQualifiers>
   <name>%</name>
   <categoryBag>
      <keyedReference tModelKey="uddi:6e090afa-33e5-36eb-81b7-
1ca18373f457" keyName="WSDL type" keyValue="service"/>
   </categoryBag>
</find_service>
```



This second request is sent to the UDDI application. The return type is a business_service list containing 0..n web-service details. The extension receives the response, parses it and constructs a new response to the client application as follows: For each service key found in the response from the UDDI registry, the extension queries its own database to see whether a corresponding evaluation is presented. If evaluation is presented, than the corresponding web-service is included in the response of the extension. The extension orders it's newly constructed response by evaluation values (if such order is meaningful) and sends it back to the client.

7. Related work, conclusions and feature work. There exist already tools that offer similar functionality to the extension presented here.

The site Binding Point [9] offers an intuitive GUI for publication and search for web-services and allows users to evaluate a web-service. Evaluations belong to single evaluation scheme, with enumerated values form 1 to 10. The site sorts the results of any search on the basis of the evaluations provided (if any). Evaluations are anonymous; any user can give as many evaluations as they want. Although intuitive and easy to use, this tool does not offer complete UDDI compatibility, neither the possibility to use multiple evaluations schemes. It is not clear whether the tool works with some UDDI registry as a back-end, and whether it is possible to "plug" it to any UDDI registry. On the other hand, it clearly demonstrates the usage of subjective user evaluations when applied to web-services.

The Microsoft UDDI registry already allows users of the UDDI registry to classify their own UDDI artifacts according to subjective criteria [4] *"In addition to geographical location criteria, developers layered standard and custom categorization schemes in UDDI Services including latency,* ***quality of service****, and SLA."* Indeed, classification is still not an evaluation in that only the owner of an entity can classify it (i.e. other users can not classify the same entity with any scheme), but it is a step further that introduces subjective, or quality categorization schemes into the UDDI registry.

The adoption of the extension to the UDDI registry presented in this paper depends to a large extend on the success of the UDDI registry as an enterprise storage of web-services. If companies extensively adopt SOA (service oriented architecture), they will most probably use the UDDI registry for storing and retrieving web-services, and can use the extension described here. In this

case, the following areas of future development can be outlined:

- ✓ Subscription could be introduced to allow clients to receive notifications regarding changes in the "rating" of a given web-service,
- ✓ Web-services hit counters can be introduced: By agreement between a web-service provider and the extension, the extension could count the hits to an end-point of a web-service and, hence, introduce additional, "objective" evaluation for it, namely "the number of times invoked".

REFERENCES

[1] Avery Ch., P. Resnick, R. Zeckhauser. The Market for Evaluations. *American Economic Review* **89** (1999), No 3, 564–584.

[2] `http://www.systinet.com/products/sr/overview`

[3] `http://uddi.sap.com`

[4] `http://www.microsoft.com/windowsserver2003/evaluation/overview/dotnet/uddi.mspx`

[5] `http://www.oracle.com/technology/tech/webservices/htdocs/uddi/index.html`

[6] `http://ws.apache.org/juddi/`

[7] `http://edocs.bea.com/wls/docs81/webserv/uddi.html#1053247`

[8] UDDI Specification, v3: `http://uddi.org/pubs/uddi-v3.0.2-20041019.htm`

[9] Binding Point site: `http://www.bindingpoint.com/default.aspx`

Alexander Mintchev
Kliment Ohridski University of Sofia
Faculty of Mathematics and Informatics
Department of Information Technologies
5, J. Bourchier Blvd
1164 Sofia, Bulgaria
e-mail: alexanderdm@fmi.uni-sofia.bg

Received October 16, 2006
Final Accepted September 13, 2007