Electronic Communications of the EASST
Volume X (2008)

Proceedings of the Workshop on Service-Oriented Computing at KIVS 2009

The Web Service Challenge - A review on Semantic Web Service Composition

Steffen Bleul, Thomas Weise, and Kurt Geihs

12 pages

Guest Editors: Sebastian Abeck, Reinhold Krger, Nicolas Repp, Michael Zapf
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

The Web Service Challenge - A review on Semantic Web Service Composition

Steffen Bleul, Thomas Weise, and Kurt Geihs

Kassel University, Distributed Systems Group, {bleul,weise,geihs}@vs.uni-kassel.de

Abstract: Every year, contesters submit contributions to the Web Service Challenge (WSC) in order to determine which service composition system is the most efficient one. In this challenge, semantic composition tasks must be solved and the results delivered by the composers are checked for correctness. The time needed for the composition process is another important competition criterion.

After we had participated with great success in the 2006 and 2007 WSC, we were asked to manage the Web Service Challenge 2008. In this paper, we present the challenge task, the challenge rules, the document format used, and the results of this competition. We provide a summary over the past challenges and give first previews on the future developments planned for the Web Service Challenges to come.

Keywords: Business Process Management, quality of service, Web Service composition, orchestration, sub-orchestration, BPEL, WSBPEL, WSDL, OWL, WSC, Web Service Challenge

1 Introduction

Since 2005, the annual Web Service Challenge[1] (WSC) provides a platform for researchers in the area of Web Service composition which allows them to compare their systems and to exchange experiences [1]. It is always co-located with the IEEE Joint Conference on E-Commerce Technology (CEC) and Enterprise Computing, E-Commerce and E-Services (EEE) [2, 3, 4].

In the past, the WSC contest scenarios as well as the involved data formats had no resemblance with real-world scenarios but were purely artificial tests for the capability of syntactic and semantic Web Service Composition systems. After being asked to manage the 2008 competition, we decided to develop the WSC to a more practice-oriented event. Therefore, we introduced new rules and based the challenge on standardized data formats such as OWL [5], WSDL [6], and WSBPEL [7]. Additionally, we introduced a new test set generator which produces configurations very similar to those found in real Service Oriented Architectures in the industry.

This led to an increase in the complexity and the quality of the challenge tasks. In this paper, we also introduce our novel and extensible generator for service composition tasks and an additional composition verification utility – both building on WSBPEL, WSDL, and OWL. Combined, the two form an efficient benchmarking system for evaluating Web Service composition engines.

[1] see http://www.ws-challenge.org/ (2007-09-02)

The main contributions of this work are

1. a detailed discussion of the requirements for realistic test scenarios for service composition systems,
2. the introduction of a versatile system able to generate such scenarios *and* to verify the results of composition processes,
3. and a review on the 2008 WSC.

2 Prerequisites

Before discussing the idea of semantic service composition, we define some necessary prerequisites. First, let us assume that all semantic concepts in the knowledge base of the composers are members of the set $\mathbb{M}$ and can be represented as nodes in a wood of taxonomy trees.

Definition 1 (subsumes) Two concepts $A, B \in \mathbb{M}$ can be related in one of four possible ways. We define the predicate *subsumes* : $(\mathbb{M} \times \mathbb{M}) \mapsto \{\texttt{true}, \texttt{false}\}$ to express this relation as follows:

1. *subsumes*(A,B) holds if and only if A is a generalization of B (B is a specialization of A).
2. *subsumes*(B,A) holds if and only if A is a specialization of B (B is a generalization of A).
3. If neither *subsumes*(A,B) nor *subsumes*(B,A) holds, A and B are not related to each other.
4. *subsumes*(A,B) and *subsumes*(B,A) is `true` if and only if $A = B$.

The *subsumes* relation is transitive, and so are generalization and specialization.

If a parameter x of a service is annotated with A and a value y annotated with B is available, we can set $x = y$ and call the service only if *subsumes*(A,B) holds (*contravariance*). This means that x expects less or equal information than given in y.

The set $\mathbb{S}$ contains all the services s known to the registry. Each service $s \in \mathbb{S}$ has a set of required input concepts $s.in \subseteq \mathbb{M}$ and a set of output concepts $s.out \subseteq \mathbb{M}$ which it will deliver on return. We can trigger a service if we can provide all of its input parameters.

Similarly, a composition request R always consists of a set of available input concepts $R.in \subseteq \mathbb{M}$ and a set of requested output concepts $R.out \subseteq \mathbb{M}$. A composition algorithm discovers a (topologically sorted) set of n services $S = \{s_1, s_2, \ldots, s_n\} : s_1, \ldots, s_n \in \mathbb{S}$. As shown in Equation 1, the first service (s_0) of a valid composition can be executed with instances of the input concepts $R.in$. Together with $R.in$, its outputs ($s1.out$) are available for executing the next service (s_2) in S, and so on. The composition provides outputs that are either annotated with exactly the requested concepts $R.out$ or with more specific ones (*covariance*). For each composition solving the request R, $isGoal(S)$ will hold:

$$\begin{aligned} &isGoal(S) \Leftrightarrow \forall A \in s_1.in\ \exists B \in R.in : subsumes(A,B) \wedge \forall A \in s_i.in, i \in \{2..n\} \\ &\exists B \in R.in \cup s_{i-1}.out \cup \ldots \cup s_1.out : subsumes(A,B) \wedge \\ &\forall A \in R.out\ \exists B \in s_1.out \cup \ldots \cup s_n.out \cup R.in : subsumes(A,B) \end{aligned} \tag{1}$$

The search space that needs to be investigated in Web Service composition [8] is the set of all possible permutations of all possible sets of services.

3 The Web Service Challenge 2008

In this paper, we want to give insight into the rules, software, and evaluation criteria of the WSC 2008 [9]. In the past years, we have participated in the Web Service Challenge [10, 11] as contestants. Although these challenges were state of the art at this point in time, some issues arose. Foremost, the WSC is directly related to the Web Service technology but the file formats used were proprietary and did not correlate with real-world formats. In 2008, standard formats were utilized in the challenge tasks and solutions. The services in the SOAs in these tasks were described in a WSDL format annotated with semantic concepts of an ontology stored in an OWL document. The compositions produced by the composers as solutions had to be delivered in WSBPEL.

3.1 The Semantic Web Service Composition Rules

In the competition, we adopt the idea of so-called Semantic Web Services that represent Web Services with a semantic description of the interface and its characteristics as outlined in Section 2. The task is to find a composition of services that produces a set of queried output parameters from a set of given input parameters. The overall challenge procedure is as follows:

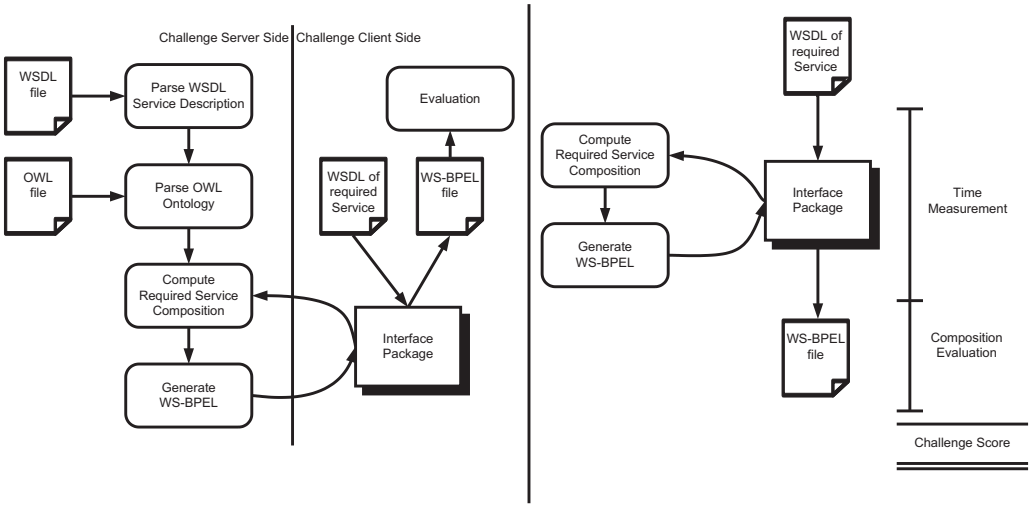

Figure 1: The procedure of the Web Service Challenge 2008.

The challenge environment itself is a distributed system. The composer software of the contestants is placed on the server side and started with a bootstrap procedure. First, it is provided with a path to a WSDL file which contains a set of services along with annotations of their input and output parameters. The number of services will change in each challenge. Every service has an arbitrary number of parameters. Additionally to the WSDL file, we also provide the address of a file containing an OWL document during the bootstrapping process. This document holds the taxonomy of concepts used in the challenge. The bootstrapping process comprises loading all relevant information from these files.

After the bootstrapping on the server side is finished, a client-side GUI queries the composition system with the challenge problem definition and memorizes the current time. The software of the contestants must now compute a solution – one or more service compositions – and answer in the solution format which is a subset of the WSBPEL schema. When the WSBPEL document is received by the GUI, the time is taken again and the compositions inside the document are verified.

This evaluation process is illustrated on the right side in Figure 1. The Web Service Challenge awards both, the most efficient system and also the best system architecture. The best architectural effort will be awarded according to the system features and the contestant's presentation. The evaluation of efficiency consists of two parts as described below:

1. Time Measurement: We take the time after submitting the query and the time when the composition result is fully received. The bootstrap mechanism is excluded from the assessment. There is also a time limit for bootstrapping after which a challenge is considered as failure.

2. Composition Evaluation:
   - Composition Length: The least amount of services which are necessary to produce the required output parameters.
   - Composition Efficiency: The least amount of execution steps inside the orchestration in order to solve the challenge.

3.2 The WSDL service descriptions

In a SOA, services are requested by client applications. Both, the client and service interfaces can be specified with WSDL. Therefore, the challenge request, the challenge response, and the descriptions of all involved services are formulated in valid WSDL documents. Semantics specified in OWL are used to annotate the service descriptions.

In the WSC scenarios, each service has just one unique service binding, portType, request, and response message. For simplification purposes, the elements related to one service adhere to the sequence sketched in Listing 1.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <definitions  xmlns="http://schemas.xmlsoap.org/wsdl/"  ...
3           xmlns:mece="http://www.vs.uni-kassel.de/mece">
4   <service .../> ...
5   <binding .../> ...
6   <portType .../> ...
7   <message .../>  ...
8   <service/> ...
9   <types>
10    <xs:schema/>
11  </types>
12  <!-- WSC-08 Semantic Annotation Section -->
13  <mece:semExtension> ... </mece:semExtension>
14 </definitions>
```

Listing 1: A WSDL Document for the WSC-08

3.3 Semantic Annotation with OWL and MECE

Ontologies are expressed with OWL, an XML format. The semantic evaluation in 2008 was limited to taxonomies consisting of sub and super class relationship between concepts only. In addition to semantic concepts (OWL-Classes), OWL allows to specify instances of classes called individuals. While individuals and classes were distinguished in the competition, the possibility to express equivalence relations between concepts was not used.

In OWL, semantics are defined with statements consisting of subject, predicate, and object, e.g. ISBN−10 is_a ISBN (ISBN subsumes ISBN−10). Such statements can be specified with simple triplets but also with XML-Hierarchies and XML-References. The implementation of an OWL-Parser is hence not trivial. In order to ease the development of the competition contributions, we defined a fixed but valid OWL-Schema.

In the WSC-08 competition, semantic individuals are used to annotate input and output parameters of services. Individuals are instances of classes and can be defined like in the following example. We illustrate the specification of an individual in line 8 with the name Individual1 which is an instance of class Class1

```
1 <?xml version="1.0"?>
2 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" ...>
3   <owl:Ontology rdf:about=""/>
4   <owl:Class rdf:ID="Class1"/>
5   <owl:Class rdf:ID="Class1.1">
6     <rdfs:subClassOf rdf:resource="#Class1"/>
7   </owl:Class>
8   <owl:Thing rdf:ID="Individual1">
9     <rdf:type rdf:resource="#Class1" />
10  </owl:Thing>
11 </rdf:RDF>
```

Listing 2: An example for the specification of semantic individuals.

The semantic annotation of the WSDL-files is done with MECE [12, 13], a valid extension of the WSDL schema.

```
1 <mece:semExtension>
2   <!-- Semantic Extension for a message with ID "getPriceRequest" -->
3   <mece:semMessageExt id="BookShopARequestMessage">
4     <!-- Semantic Annotation for the xsd:element with ID "price" -->
5     <mece:semExt id="price">
6       <!-- Ontology reference to the semantic individual -->
7       <mece:ontologyRef>
8           http://www.ontologies.org/Ontology.owl#Bookprice
9       </mece:ontologyRef>
10    </mece:semExt>
11  </mece:semMessageExt>
12  <!-- Arbitrary amount of message annotations -->
13  <mece:semMessageExt id="BookShopAResponseMessage"/>
14  ...
15  <mece:semMessageExt .../>
16  ...
17 </mece:semExtension>
```

Listing 3: The Semantic Extension

3.4 The WSBPEL solution format

During the WSC 2007, many participants encouraged the usage of a process language like BPEL as output format for the composition solutions. First of all, a process language has more expressiveness than the 2007 solution format. Secondly, a process language can be used to connect the challenge implementation to real world technologies and thus, improves their reusability. Therefore, we decided to use a subset of the common Web Service standard WSBPEL as sketched Listing 4.

In WSBPEL, the concurrent execution of services can (which is a desired feature for the challenge) can be specified. In the 2008 WSC WSBPEL subset, specification details like partnerlinks and copy instructions on message elements were omitted. In the example in 4, two alternative solutions to a composition request are given.

```
<process name="MoreCreditsBP"
  xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
  targetNamespace="http://www.ws-challenge.org/WSC08CompositionSolution/">
 <sequence name="main">
   <receive name="receiveQuery"
     portType="SolutionProcess" variable="query"/>
     <switch name="SolutionAlternatives-SolutionA-SolutionB">
       <case name="Alternative-SolutionA">
         <sequence>
           <invoke name="serviceA"
             portType="seeWSDLFile"
             operation="seeWSDLFile"/>
           <flow>
       <case name="Alternative-SolutionB">...</case>
     </switch>
  </sequence>
</process>
```

Listing 4: The WSBPEL document

4 The Test Set Generation

Besides being not able to express concurrent service invocations, the pre-2008 challenge and solution formats had another limitation: Alternative services inside a composition could only differ in their names but always had exactly the same input and output parameters. This has the effect that the search space shrinks dramatically.

In order to provide challenges which are more realistic, the test set generation had to be rebuilt from the scratch. Doing this revealed several scientific challenges concerning a property which we refer to as the *Test Set Quality*.

There are several features a generator must offer. The basic ability is the generation of a test set consisting of concepts inside taxonomies, a set of services and a challenge with a valid composition solution. The generation must be able to be configured. The desired configuration options include the number of concepts for the ontology and the number Web Services in the knowledge base. The concepts and Web Service have to be ordered and named in a way that no obvious solution appears. This issue becomes even more challenging when we want to be able to

configure the number of solutions, the execution depth of a valid composition, and the inclusion of concurrent execution of Web Services.

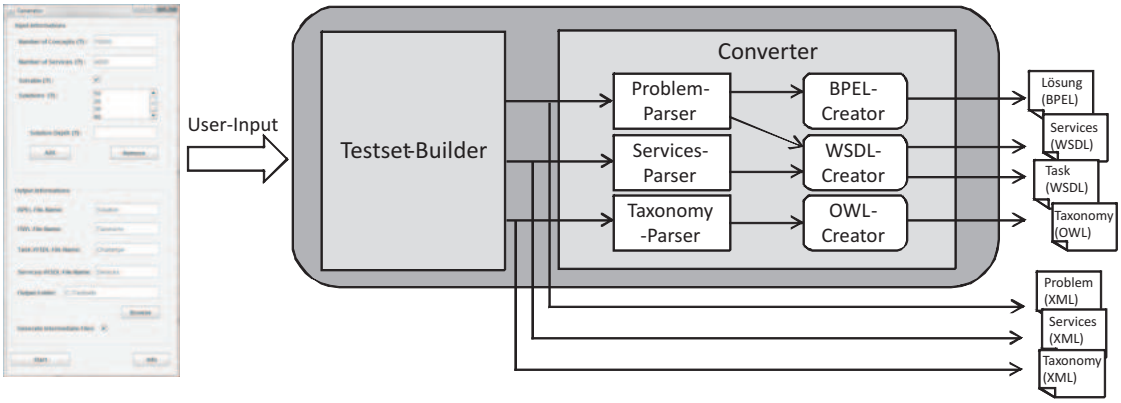

Figure 2: The test set generation process.

Figure 2 illustrates the challenge generation process. We have implemented a user interface for entering the configuration of the test set. Afterwards, the Test Set Builder produces an ontology an ontology which consists of an arbitrary amount of taxonomies with differing size and depth. A set of disjoint solutions is created according to the specified configuration parameters. Then, a set of additional services is added which may (misleadingly) involve some (but not all) of the concepts used in these solutions. The result of the generation process is saved as an intermediate XML format. This format will then be transformed to the discussed WSBPEL, WSDL, and OWL subsets by the Converter component. The intermediate format is still human-readable for manual evaluation of the Test Set Quality. Both, the Test Set Builder and the final challenge formats are independent and can be extended or modified separately, making the Test Set Builder reusable in scenarios different from the WSC.

We define Test Set Quality as a term for generating demanding test sets. The solutions of the generated challenge include complex process patterns, such as multiple levels of sequential and parallel threads of execution. Furthermore, the Test Set Builder can generate multiple valid solutions, so there may be one solution which involves the minimum number of services and another one which has more services but lesser execution steps due to better parallelism. As an example, we used one test set in the WSC 2008 where the minimum amount of services for one solution was 37 with an execution depth of 17, but another solution existed with 46 services and an execution depth of 7 (Table 1).

The evaluation of the composition engines became much more interesting just because of the improved Test Set Quality. In the example just mentioned, for instance, the participating composition engines delivered different optimal results.

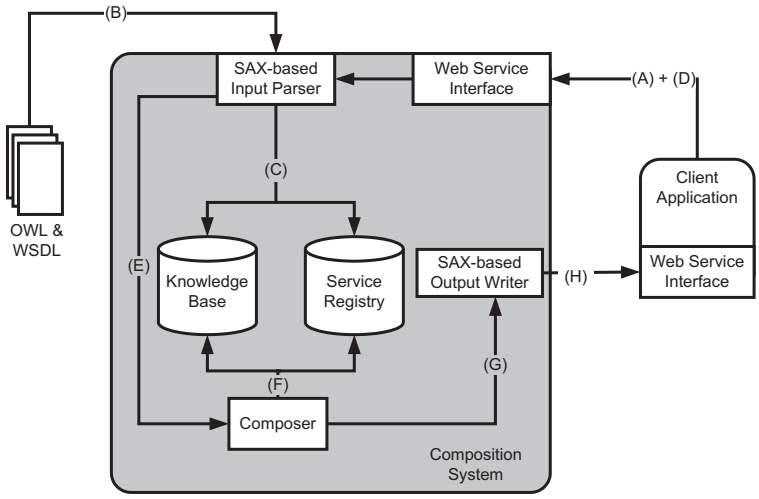

Figure 3: Example Composition System Implementation.

5 The Challenge Software

An important characteristic of the Web Service Challenge is the extent of software which was provided to the contestants. There was not only a Test Set Generator, but also a Composition System client, a Solution Validator, and implementation examples in Java. This bundle is called the interface package. We provided not only the binaries but also the source code of all software prior to the Challenge itself in order to attain full transparency.

Since 2007, the WSC allows heterogeneous implementations in differing programming languages and technologies of composition systems as long as they can be invoked and queried with the Web Service client. In order to make this work, the contestants must implement a server-side Web Service interface defined by a given WSDL description. We present the communication pattern on the basis of our example implementation illustrated in Figure 3.

Firstly, the WSC *Client Application* submits the URL of two local or remote challenge files (A). The first URL locates the OWL taxonomy and the second one locates the WSDL service description (B). The *SAX-based Input Parser* initializes the internal *Knowledge Base* and the *Service Registry* as part of the bootstrap mechanism (C). Secondly, the WSC client submits the URL of the WSDL query document (D). Starting from this point, the parser notifies the *Composer* (E) which computes a solution (F). The solutions are passed to the *SAX-based Output Writer* (G). Thirdly, the client UI offers an internal Web Service as a callback interface. The client-side callback interface is used to avoid communication timeouts. The composition system calls this callback Web Service in order to stream the composition result to the *Client Application* (H).

The evaluation of the result is done with the Test Set Validator software. The validator takes the returned WSBPEL document as its input and produces a readable XML-based analysis. An excerpt of an example in this format is presented in Listing 5. The most prominent feature

is the analysis of the correctness of the solution. A correct solution not only is an executable composition (validity) but also delivers all wanted output parameters of the challenge (solutions). Furthermore, the evaluator determines the minimum set of services and minimum execution steps from all correct solutions in its WSBPEL input.

```
<result>
  <validity numYes="2"...>All proposed BPEL processes can be
      executed.</validity>
  ...
  <solutions numYes="2"...>All proposed BPEL processes are correct solutions
      for the challenge.</solutions>
  <combinations ...>5406912 execution sequences ... were found in
      total.</combinations>
  <minLength ...>The shortest execution sequence that solves the challenge
      found is 14.</minLength>
  <minService ...>The execution sequence involving the least number of
      services ... consists of 20 services.</minService>
  <redundanceStat ...>No redundant services have been
      discovered.</redundanceStat>
</result>
```

Listing 5: The Evaluator XML output.

In this analysis, redundant service invocations are detected as well. A redundant service may be part of a composition but does not provide any necessary output parameter or is executed more than once.

6 Evaluation

The evaluation of the efficiency of the composition systems in the 2008 WSC consisted of two steps as described below. A condition of the Web Service Challenge is that each composition system must be evaluated on the same test system, a Lenovo X61 ThinkPad with an Intel Core2 DUO 1.6 GHz processor, 3 GB memory, and the operating system Windows XP Professional.

Three challenge sets we used in the competition and each composition system can achieve up to 18 points per challenge set. The time limit for solving all challenges has been 15 minutes. The score system for each challenge set was:

- +6 Points for finding the minimum set (Min. Services) of services that solves the challenge.
- +6 Points for finding the composition with the minimum execution length (Min. Execution) that solves the challenge.
- Additional points for:
  1. +6 Points for the composition system which finds the minimum set of services or execution steps that solves the challenge in the fastest time (Time (ms)).
  2. +4 Points for the composition system which solves the challenge in the second fastest time.
  3. +2 Points for the composition system which solves the challenge in the third fastest time.

	Tsinghua University		University of Groningen		Pennsylvania State University	
	Result	Points	Result	Points	Result	Points
Challenge Set 1						
Min. Services	10	+6	10	+6	10	+6
Min. Execution	5	+6	5	+6	5	+6
Time (ms)	312	+4	219	+6	28078	
Challenge Set 2						
Min. Services	20	+6	20	+6	20	+6
Min. Execution	8	+6	10		8	+6
Time (ms)	250	+6	14734	+4	726078	
Challenge Set 3						
Min. Services	46		37	+6	No result.	
Min. Execution	7	+6	17			
Time (ms)	406	+6	241672	+4		
Sum	46 Points		38 Points		24 Points	

Table 1: Web Service Challenge 2008 Score Board

The results of the Web Service Challenge 2008 are listed in Table 1 which is limited to the results of the first three places of the eight participants. The performance winners of the Web Service Challenge 2008 are:

1. Y. Yan, B. Xu, and Z. Gu. Tsinghua University, Beijing, China.
2. M. Aiello, N. van Benthem, and E. el Khoury. University of Groningen, Netherlands.
3. J. Jung-Woo Yoo, S. Kumara, and D. Lee. Pennsylvania State University, and S.-C. Oh of General Motos R&D Center, Michigan, USA.

Finally, we present the winners of the architectural challenge:

1. M. Aiello, N. van Benthem, and E. el Khoury. University of Groningen.
2. P.A. Buhler and R.W. Thomas. College of Charleston, South Carolina, USA.
3. K. Raman, Y. Zhang, M. Panahi, and K.-J. Lin. University of California, Irvine, USA.
   T. Weise, S. Bleul, M. Kirchhoff, and K. Geihs. University of Kassel, Germany.

7 Related Work

The Web Service Challenge is the first competition for (semantic) service composition. During the last years there also was a syntactic challenge, then the introduction of semantics also covered the syntactic tasks and hence, a sole semantic approach was favored. In this section, we give a short overview on other related competitions.

The closest related event is the Semantic Web Service (SWS) Challenge [14]. In contrast to the WSC, the SWS defines test *scenarios* which the participant have to solve. The SWS committees see themselves more as a certification event for frameworks than a competition. Whereas the

WSC generates vast test sets and concentrate on standardized formats, the SWS defines challenge scenarios which must be solved by creating new specification languages, matching algorithms, and execution systems. The scenarios in the SWS comprise state-based services, service mediation, ontology reasoning and service provisioning. The certified frameworks not only solve one or more scenarios, but are also able to execute their results at runtime.

Another closely related contest is the IEEE International Services Computing Contest (SC-Contest 2007–2008) [15]. This is a Web Service centric demonstration event where participants choose both problem definition and solution themselves. The participants are invited to demonstrate methodologies, reference architectures, and tools. The winner of this contest is determined by evaluating importance in Service-oriented Computing and implementation quality.

8 Conclusion and Preview

In this paper we presented a review on the Web Service Challenge 2008 where eight teams from all over the world competed. While the previous challenges only utilized scientific formats, the challenge definition in 2008 adopted the WSBPEL, WSDL, and OWL standards. Additionally, an extensible test set generator for the semantic service composition problems.

The Web Service Challenge created a research community for automated semantic service composition and implementations of fast performing composition engines. The focus of the challenge is currently changing from a solely scientific service indexing competition to a comprehensive and practice-oriented solution for Service-oriented Architectures.

In order to tie up to the result of this year and as a call of participation for the reader, we propose several changes in the WS-Challenge 2009. First, the prescribed test system for evaluation is a restriction for the contestant's system architecture. In 2009, every contestant can use their own (remote) system which will then be invoked with the challenge client. Secondly, the services will be enriched with Quality of Service (QoS) dimensions, e.g. response time and cost. The compositions must not only solve the challenge but must minimize the cost and response time of the overall composition. We wish to thank every contestant of the WSC'08 for their participation. Hopefully, we will meet and also welcome new contestants at the 2009 Web Service Challenge. [2]

Bibliography

[1] M.B. Blake, K.C. Tsui, and A. Wombacher. The eee-05 challenge: a new web service discovery and composition competition. In *Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce, and e-Service, EEE'05*, pages 780–783, 2005.

[2] *Proceedings of 2006 IEEE Joint Conference on E-Commerce Technology (CEC'06) and Enterprise Computing, E-Commerce and E-Services (EEE'06)*. IEEE, 2006.

[3] *Proceedings of IEEE Joint Conference (CEC/EEE 2007) on E-Commerce Technology (9th CEC'07) and Enterprise Computing, E-Commerce and E-Services (4th EEE'07)*. IEEE, 2007.

[2] see http://cec2009.isis.tuwien.ac.at/ (2008-11-03)

[4] *Proceedings of IEEE Joint Conference (CEC/EEE 2008) on E-Commerce Technology (10th CEC'08) and Enterprise Computing, E-Commerce and E-Services (5th EEE'08)*. IEEE, 2006.

[5] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D.L. McGuinness, P.F. Patel-Schneider, and L.A. Stein. *OWL Web Ontology Language Reference*. W3C, 2004. W3C Recommendation. URL: http://www.w3.org/TR/2004/REC-owl-ref-20040210/ [accessed 2007-11-22].

[6] D. Booth and C.K. Liu. *Web Services Description Language (WSDL) Version 2.0 Part 0: Primer*. W3C, 2007. W3C Recommendation. URL: http://www.w3.org/TR/2007/REC-wsdl20-primer-20070626 [accessed 2007-09-02].

[7] D. Jordan, J. Evdemon, A. Alves, A. Arkin, S. Askary, C. Barreto, B. Bloch, F. Curbera, M. Ford, Y. Goland, A. Guízar, N. Kartha, C.K. Liu, R. Khalaf, D. König, M. Marin, V. Mehta, S. Thatte, D. van der Rijn, P. Yendluri, and A. Yiu. *Web Services Business Process Execution Language Version 2.0*. OASIS, April 11, 2007. URL: http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf [accessed 2007-10-25].

[8] T. Weise, S. Bleul, D. Comes, and K. Geihs. Different approaches to semantic web service composition. In A. Mellouk, J. Bi, G. Ortiz, D.K.W. Chiu, and M. Popescu, editors, *Proceedings of The Third International Conference on Internet and Web Applications and Services, ICIW 2008*, pages 90–96. IEEE, 2008.

[9] A. Bansal, M.B. Blake, S. Kona, S. Bleul, T. Weise, and M.C. Jaeger. Wsc-08: Continuing the web service challenge. In *CEC/EEE'08*, pages 351–354, 2008. In [4].

[10] M.B. Blake, W.K.W. Cheung, M.C. Jaeger, and A. Wombacher. Wsc-06: The web service challenge. In *CEC/EEE'06*, pages 505–508, 2006. In [2].

[11] M.B. Blake, W.K.W. Cheung, M.C. Jaeger, and A. Wombacher. Wsc-07: Evolving the web service challenge. In *CEC/EEE'07*, pages 422–423, 2006. In [3].

[12] S. Bleul, D. Comes, M. Kirchhoff, and M. Zapf. Self-integration of web services in bpel processes. In *Proceedings of the Workshop Selbstorganisierence, Adaptive, Kontextsensitive verteilte Systeme (SAKS)*, 2008.

[13] S. Bleul, K. Geihs, D. Comes, and M. Kirchhoff. Automated Management of Dynamic Web Service Integration. In *Proceedings of the 15th Annual Workshop of HP Software University Association (HP-SUA)*, Kassel, Germany, 2008. Infocomics-Consulting, Stuttgart, Germany.

[14] SWS - Challenge on Automating Web Services Mediation, Choreography and Discovery, Challenge Homepage. URL: http://sws-challenge.org/ [accessed 2008-11-03].

[15] IEEE Services Computing Contest (SCC) 2008, Contest Homepage. URL: http://iscc.servicescomputing.org/2008/ [accessed 2008-11-03].