The Web Service Challenge Rules

1   The Web Service Challenge

Since 2005, the annual Web Service Challenge[1] (WS-Challenge, WSC) provides a platform
for researchers in the area of web service composition which allows them to compare their
systems and to exchange experiences. It is co-located with the IEEE Conference on
Electronic Commerce (CEC) and the IEEE International Conference on e-Technology,
e-Commerce, and e-Service (EEE).

During the last years, the web service challenge focused on optimizing the service
discovery and composition process solely, using abstractions from real-world situations.
The taxonomies of semantic concepts as well as the involved data formats were purely
artificial. Starting with this year's competition, the data formats and the contest data
itself will be based on the OWL, WSDL, and WSBPEL schemas for ontologies, services
and service orchestrations.

2   The Challenge

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. The
task is to find a composition of services that produces a set of queried output parameters
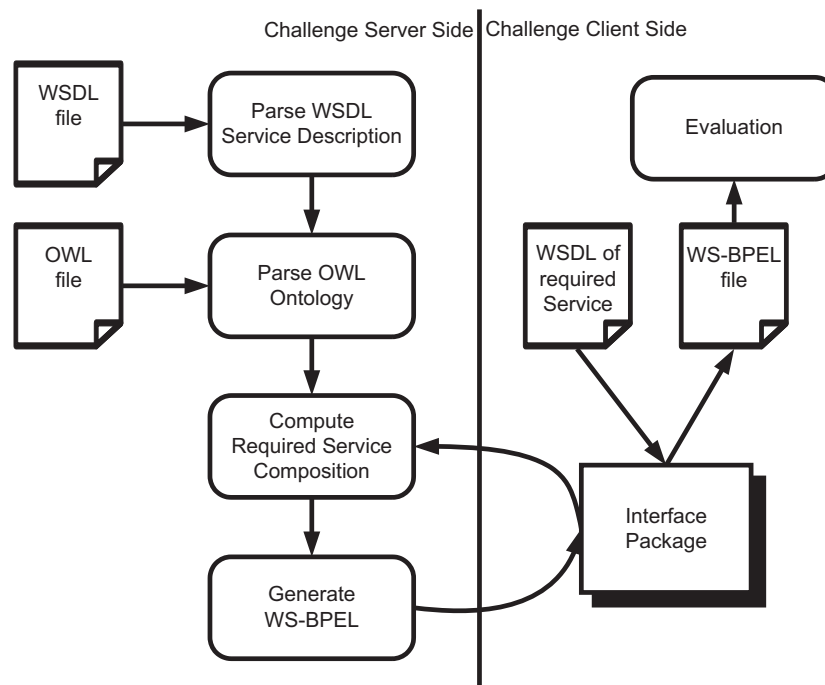from a set of given input parameters. The overall challenge procedure is as follows:

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

The composer software of the contestants is placed on the server side and started
with a bootstrap procedure. First of all, the system is provided with a path to a WSDL
file. The WSDL file contains a set of services along with annotations of their input- and
output parameters. The number of services will change from challenge to challenge. Every
service will have an arbitrary number of parameters. Additional to the WSDL file, we also
provide the address of the OWL file during the bootstrapping process. This file contains

---

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

the taxonomy of concepts used in this challenge in OWL format. The bootstrapping process includes loading the relevant information from these files.

The challenge task will then be sent to the composer via a client-side GUI very similar to last year's challenge. After the bootstrapping on the server side is finished, the GUI queries the composition system with the challenge problem definition. The contestant's software must now compute a solution – one or more service compositions – and answer in the solution format which is a subset of the WS-BPEL schema. When the WS-BPEL document is received by the GUI, we will stop a time measurement and afterwards evaluate the compositions themselves.

3 Evaluation

The Web Service Challenge awards the most efficient system and also the best architectural solution. The best architectural effort will be awarded according to the contestant's presentation and system features. The evaluation of efficiency consists of two parts as described below. The exact formula will be published soon.
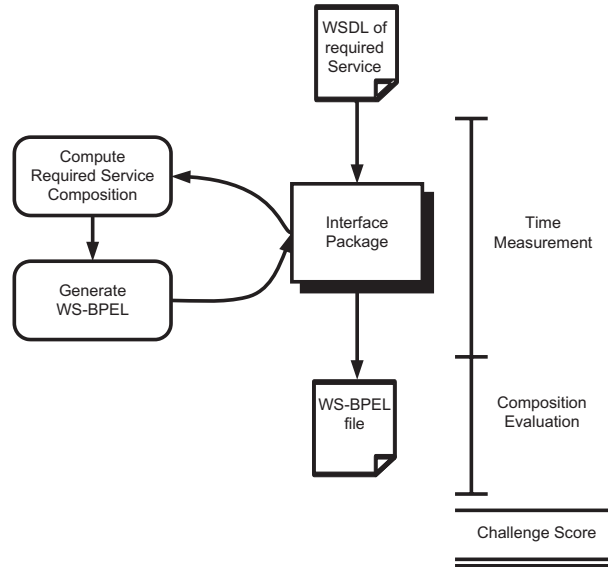

Figure 2: Evaluation in the Web Service Challenge 2008.

1. Time Measurement: The time measurement is done by the interface package. 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 will be a time limit for bootstrapping after which a challenge is considered as failure.

2. Composition Evaluation:

   - Completeness: The amount of compositions discovered by the system.
   - Composition Length: The shortest composition.
   - Composition Efficiency: Parallel versus sequential execution of services.

4 What's New

Document Formats: OWL Ontologies, WSDL-Queries, WS-BPEL solutions schemas. Example files and documentation will follow shortly.
XSD-Types: The challenge will include matching XSD-Type definitions like arrays, simple types, complex types with substructures and enumerations.

Parallel Execution: The WS-BPEL schema supports the specification of parallel execution of services. Valid parallel execution will positively influence the systems challenge score.

No results: The challenge will include sets of services that actually deliver no solution. The system should act accordingly and cancel the discovery in time.

Images: We are currently investigating whether it makes sense to provide images for virtual machines along with the exact system specs of the systems that will be used to test the composers.

5 The OWL Schema

Ontologies are usually expressed with OWL [1], an XML format [2]. We use the OWL format in the 2008 challenge, but like in the previous years, we limit semantic evaluation strictly to taxonomies consisting of sub and super class relationship between semantic concepts only. OWL is quite powerful. In addition to semantic concepts (OWL-Classes), OWL allows to specify instances of classes called individuals. While we also distinguish between individuals and classes in the competition, the possibility to express equivalence relations between concepts is not used.

In OWL, the semantic is 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 will stick to a fixed but valid OWL-Schema.

```
1  <?xml version="1.0"?>
2  <rdf:RDF
3      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4      xmlns:owl="http://www.w3.org/2002/07/owl#"
5      xmlns="http://www.owl-ontologies.com/Ontology.owl#"
6      xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
7      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
8    xml:base="http://www.owl-ontologies.com/Ontology.owl">
9    <owl:Ontology rdf:about=""/>
10 </rdf:RDF>
```

Listing 1: An example of a basic OWL-Document

Listing 1 illustrates an example of a basic OWL document. The single lines have the following meaning:

Line 1:   The XML-Document declaration.

Line 2:   The OWL-Document root. OWL is the ontology schema for the semantic description language RDF [3]. So some language elements are reused by OWL.

Line 3:   The RDF elements namespace.

Line 4:   The OWL elements namespace.

Line 5:   The namespace of this ontology (Syntax: `ontologyname+#`).

Line 6:   The XSD [4] elements namespace.

Line 7:   The RDFS [5] elements namespace. RDF Schema (RDFS) is a language extension of RDF.

Line 8:   The base namespace is also the ontology name in OWL.

Line 9:   The ontology declaration element `owl:Ontology`. The ontology name can be found in the XML-base namespace `xml:base`.

In general, the syntax for ontology namespaces is a valid URI [6, 7] followed directly by `#`. An ontology name must be a valid URI (e.g. `http://www.owl-ontologies.com/Ontology.owl`). Its namespace is then automatically the URI+#. (e.g. `http://www.owl-ontologies.com/Ontology.owl#`). In the WS-Challenge we stick to a fixed schema. We define a fixed URI and namespace for the taxonomy. See Listing 2:

- The WSC-Ontology has the name `http://www.ws-challenge.org/wsc08.owl`.

- The WSC-Ontology namespace therefore is `http://www.ws-challenge.org/wsc08.owl#`.

```
1   <?xml version="1.0"?>
2   <rdf:RDF
3       xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4       xmlns:owl="http://www.w3.org/2002/07/owl#"
5       xmlns="http://www.ws-challenge.org/wsc08.owl#"
6       xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
7       xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
8     xml:base="http://www.ws-challenge.org/wsc08.owl">
9     <owl:Ontology rdf:about=""/>
10  </rdf:RDF>
```

Listing 2: An example of a WSC-08 OWL-based taxonomy document.

```
1   <?xml version="1.0"?>
2   <rdf:RDF
3       xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4       ...>
5     <owl:Ontology rdf:about=""/>
6
7     <owl:Class rdf:ID="Class1"/>
8     <owl:Class rdf:ID="Class2"/>
9
10    <owl:Class rdf:ID="Class1.1">
11      <rdfs:subClassOf rdf:resource="#Class1"/>
12    </owl:Class>
13  </rdf:RDF>
```

Listing 3: An example for the specification of class and subclasses.

Listing 3 outlines how classes and subclasses can be specified like in OWL. The lines have the following meaning:

Line 7:   The declaration of the concept/class with the name `Class1`.

Line 8:   The declaration of the concept/class with the name `Class2`.

Line 10:   The declaration of the concept/class with the name `Class1.1`.

Line 11:   The definition that `Class1.1` is a subclass of `Class1`.

In OWL, classes are defined with the `owl:Class` tag. Subclasses can be specified with `rdfs:subClassOf` tags. The attribute `rdf:ID` declares a new class name, whereas `rdf:resource` is a reference to a class. In OWL, a reference has the syntax `classname+#`, as for instance used in line 11.

In the WSC-08 competition, we use semantic individuals to annotate input and output parameters of services. Individuals are instances of classes and can be defined like in the following example (Listing 4).

Line 13:   Specification of an individual with the name `Individual1` which is an instance of class `Class1`.

Line 17:   Specification of an individual with the name `Individual1.1` which is an instance of class `Class1.1`

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

```
10        <rdfs:subClassOf rdf:resource="#Class1"/>
11    </owl:Class>
12
13    <owl:Thing rdf:ID="Individual1">
14          <rdf:type rdf:ressource="#Class1" />
15    </owl:Thing>
16
17    <owl:Thing rdf:ID="Individual1.1"/>
18          <rdf:type rdf:ressource="#Class1.1" />
19    </owl:Thing>
20 </rdf:RDF>
```

Listing 4: An example for the specification of semantic individuals.

Since Class1.1 is a subclass of Class1, Individual1.1 also is a specialization of Individual1.

In short, the 2008 WSC will use the following basic document structure for the definition of semantic concepts:[2]

```
1  <?xml version="1.0"?>
2  <rdf:RDF
3     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4     ...>
5     <owl:Ontology rdf:about=""/>
6     ...
7     ClassDeclarations
8       SubClassRelationships
9       ...
10    ...
11    IndividualDeclarations
12    ...
13 </rdf:RDF>
```

Listing 5: The basic document structure.

6 Service Descriptions

As already mentioned, we will use semantics specified in OWL to annotate the service descriptions. Furthermore, this year's competition service descriptions offer:

- The service descriptions will be provided in a single WSDL-Document.

- The annotation with semantic individuals will not only be used for message parts, but for whole message structures specified with XSD.

- This message structures can consist of simple elements, SOAP-Arrays [8], Lists, Structures, and Enumerations.

- The matching will be based on documents in an extended WSDL format for semantic annotation.

- Also here we provide a fixed schema for easier parsing of the final document and will provide a schema definition.

6.1 The Basic WSDL-Document

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <definitions
3     xmlns="http://schemas.xmlsoap.org/wsdl/"
4     xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
5     xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
6     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
7     xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
8     xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
9     xmlns:service="http://www.ws-challenge.org/WSC08Services/"
10    targetNamespace="http://www.ws-challenge.org/WSC08Services/">
11
```

---

[2]XSD-definition will follow

```
12    <service name="BookShopA">
13      <port binding="service:searchBookSOAP" name="searchBook">
14        <soap:address location="http://www.unknownexamplehost.ukn/"/>
15      </port>
16    </service>
17    <binding name="searchBookSOAP" type="service:searchBookPortType"/>
18    <portType name="searchBookPortType"/>
19    <message name="BookShopRequestMessage"/>
20    <message name="BookShopResponseMessage"/>
21
22    .....
23
24    <service name="BookShopZ">
25      <port binding="service:searchBookSOAPZ" name="searchBookZ">
26        <soap:address location="http://www.unknownexamplehost.ukn"/>
27      </port>
28    </service>
29    <binding name="searchBookSOAPZ" type="service:searchBookPortTypeZ"/>
30    <portType name="searchBookPortTypeZ"/>
31    <message name="BookShopZRequestMessage"/>
32    <message name="BookShopZResponseMessage"/>
33
34    ......
35
36    <types>
37      <xs:schema/>
38    </types>
39  </definitions>
```

Listing 6: Example of the basic WSDL-Document.

Line 1:	XML-Document declaration.
Line 2-10:	Namespacedeclaration (fixed for any composition challenge)
Line 12-16:	Servicedeclaration for a service named "BookShopA"
Line 17:	The binding for BookShopA.
Line 18:	The `portType` for binding of `BookShopA`.
Line 19:	The request message for the `BookShopA`.
Line 20:	The response message for the `BookShopA`.
Line 22:	Marks an arbitrary amount of services.
Line 24:	Starting the declaration of `BookshopZ`.
Line 34:	Marks an arbitrary amount of services.
Line 36-38:	Here we declare all message structures for all services.
Line 39:	The end of the WSDL document.

In the WSC scenarios, we use the simplification that each service just has one unique service binding, `portType`, request and response message. The declaration of the binding of the service, `portType`, request, and response message is a fixed sequence. The elements related to one service are followed one after the other. Thus, parsing gets a lot easier.

This sequence in the declaration of a service will be used for the whole set of services for the composition challenge. The schema definition of all messages comes at the end. The overall WSDL document structure is valid. Challengers may or may not adjust their parsers for better performance by making use of the restrictions defined above.

6.2 The Binding Section for the Services

```
1  <service name="BookShopA">
2    <port binding="service:searchBookSOAP" name="searchBook">
3      <soap:address location="http://www.unknownexamplehost.ukn/"/>
4    </port>
5  </service>
```

```
6   <binding name="searchBookSOAPA" type="service:searchBookPortTypeA">
7     <soap:binding style="rpc"
        transport="http://schemas.xmlsoap.org/soap/http"/>
8     <operation name="searchBookA">
9       <soap:operation
10        soapAction="http://www.ws-challenge.org/BookShopA/searchBook" />
11      <input>
12        <soap:body use="literal" />
13      </input>
14      <output>
15        <soap:body use="literal" />
16      </output>
17    </operation>
18  </binding>
19  <portType name="searchBookPortType"/>
20  <message name="BookShopARequestMessage"/>
21  <message name="BookShopAResponseMessage"/>
```

Listing 7: An example for the binding section for each service

Line 6:   Declaration of the binding of `BookShopA` (see reference in line 2) and reference to its `portType` "`searchBookPortType`".

Line 7-18: Template generated WSDL-Binding with no specific further semantics.

6.3   The `portType` Section

```
1   <service name="BookShopA"/>
2   <binding name="searchBookSOAPA" type="service:searchBookPortTypeA"/>
3   <portType name="searchBookPortTypeA">
4     <operation name="searchBookAOperation">
5       <input message="service:BookShopARequestMessage"/>
6       <output message="service:BookShopAResponseMessage"/>
7     </operation>
8   </portType>
9   <message name="BookShopARequestMessage"/>
10  <message name="BookShopAResponseMessage"/>
```

Listing 8: An example for the `portType` section for each service

Line 3:   Declaration of the service's BookShopA portType (see reference in line 2).

Line 4-8:   Automatically generated references to the service's input message (see ID in line 9) and response message (see ID in line 10).

6.4   The `message` Section

```
1   <wsdl:message name="BookShopARequestMessage">
2       <wsdl:part element="service:Book" name="BookShopAParameters"/>
3   </wsdl:message>
4   <wsdl:message name="BookShopAResponseMessage">
5       <wsdl:part element="service:Books" name="BookShopAParameters"/>
6   </wsdl:message>
```

Listing 9: An example for the `message` section for each service

Line 2+5:   : Reference to the respective message structures inside the `<types/>`-Section of this WSDL-Document. This reference has no further semantic meaning.

6.5 The Message Structures and Semantic Annotations

```
1   <types >
2     <xsd:schema targetNamespace="http://www.ws-challenge.org/WSC08Services/"
3       xmlns:wsdl-ext="http://www.vs.uni-kassel.de/wsdl_extensions">
4     <!-- Simple message with one element:-->
5       <xsd:element name="BasicBook" name="title" type="xsd:string"
              wsdl-ext:id="Semtitle"/>
6
7       <!-- Message structure of a struct "book":-->
8       <xsd:element name="Book">
9         <xsd:complexType >
10          <xsd:sequence >
11            <xsd:element name="isbn"   type="xsd:string"/>
12            <xsd:element name="title"  type="xsd:string"/>
13            <xsd:element name="author" type="xsd:string"/>
14            <xsd:element name="year"   type="xsd:string"/>
15            <xsd:element name="price"  type="xsd:float"/>
16          </xsd:sequence >
17        <xsd:complexType >
18      </xsd:element >
19
20      <!-- Message structure of an array of books:-->
21      <xsd:element name="Books">
22        <xsd:complexType >
23          <xsd:sequence minOccurs="0" maxOccurs="unbounded">
24            <xsd:element name="isbn"   type="xsd:string"/>
25            <xsd:element name="title"  type="xsd:string"/>
26            <xsd:element name="author" type="xsd:string"/>
27            <xsd:element name="year"   type="xsd:string"/>
28            <xsd:element name="price"  type="xsd:float"/>
29          </xsd:sequence >
30        <xsd:complexType >
31      </xsd:element >
32
33      <!-- Message structure of an array of books -->
34      <xsd:element name="BooksGenre">
35        <xsd:complexType >
36          <xsd:sequence minOccurs="0" maxOccurs="unbounded">
37            <xsd:element name="isbn"   type="xsd:string"/>
38            <xsd:element name="title"  type="xsd:string"/>
39            <xsd:element name="author" type="xsd:string"/>
40            <xsd:element name="year"   type="xsd:string"/>
41            <xsd:element name="price"  type="xsd:float"/>
42          </xsd:sequence >
43        <xsd:complexType >
44      </xsd:element >
45
46    </xsd:schema >
47  </types >
```

Listing 10: Message Examples

Line 1:   Start of the WSDL-Types section. This section specifies the structure of the SOAP messages [8] with XSD.

Line 20:   Start of the XSD section.

Line 4:   Definition of a single message element `BasicBook` consisting of the title of the book as a string.

Line 7-17:   Definition of a book message structure `Book` consisting of single structure elements `isbn`, `title`, `author`, `year` and `price`.

Line 9-15:   Structure is a sequence of elements.

Line 21-31: A definition like the ones above, but this one is an array of books. (see `xsd-attributes minOccurs=""maxOccurs=""`).

Line 34-46: Like the examples before, this one defines an array of books.

Note that:

- There are no references on XSD elements. The message definition is strictly hierarchical. We introduce this simplification to ease parsing.

- Only simple elements like `xsd:element` with types, e.g. `string` are annotated. Complex elements defined by a `xsd:complexType` must be reasoned by its sub-elements. This is the semantic challenge.

7   WSDL Semantic Extension

The semantic annotation of the WSDL-files is done in a valid extension of the WSDL schema. We use references (see Section above) to extensions which are defined in an extra section at the end of the WSDL-file.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <definitions
3    xmlns="http://schemas.xmlsoap.org/wsdl/"
4    ...
5    xmlns:wsdl-ext="http://www.vs.uni-kassel.de/wsdl_extensions">
6
7    <service .../>
8    <binding .../>
9    <portType .../>
10   <message .../>
11   ...
12   <message .../>
13   <service/>
14   ...
15
16   <types>
17     <xs:schema/>
18   </types>
19
20   <!-- WSC-08 Semantic Annotation Section -->
21   <wsdl-ext:semExtension>
22
23     <!-- Semantic Message Annotation -->
24     <wsdl-ext:semMessageExt/>
25     ...
26     </wsdl-ext:semMessageExt>
27
28     <!-- Semantic Message Annotation -->
29     <wsdl-ext:semMessageExt/>
30     ...
31     </wsdl-ext:semMessageExt>
32
33   </wsdl-ext:semExtension>
34  </definitions>
```

Listing 11: A WSDL Document for the WSC-08

Line 2-34: WSDL-document structure.

Line 21-33: WSDL-extension section.

Line 24-26: A annotation element for a whole message.

```
1  <wsdl-ext:semExtension>
2
3    <!-- Semantic Extension for the message with ID "getPriceRequest" -->
4    <wsdl-ext:semMessageExt idref="getPriceRequest">
5
6      <!-- Semantic Annotation for the xsd:element with ID "price" -->
7      <wsdl-ext:semExt id="price">
8
9        <!-- Ontology reference for the semantic individual for this element
              -->
10       <wsdl-ext:ontologyRef>
```

```
11            http://www.owl-ontologies.com/Ontology.owl#Bookprice
12         </wsdl-ext:ontologyRef >
13
14      </wsdl-ext:semExt >
15
16    </wsdl-ext:semMessageExt >
17
18    <!-- Arbitrary amount of message annotations -->
19    <wsdl-ext:semMessageExt idref="getPriceResponse"/>
20    ...
21    <wsdl-ext:semMessageExt .../>
22    ...
23 </wsdl-ext:semExtension >
```

Listing 12: The Semantic Extension

Line 1-23: WSDL-extension section.

Line 4-16: Semantic annotation of a portType-Message.

Line 7-14: Semantic annotation of a single xsd-element with id "price".

Line 10-12: The reference to a semantic individual for the "price" in this message.

Line 19-22: Representing an arbitrary amount of semantic message annotations.

8 The Challenge Format

The challenge itself is also represented by a valid WSDL service description. In a SOA services are requested by client applications and client interfaces are fixed as well as service interfaces. Because client interfaces can also be described by WSDL description it is obvious to specify a challenge request by a WSDL description.

The challenge document schema is illustrated in Listing 13. Here the WSDL document requests a service composition with the given input concepts:

- http://www.ws-challenge.org/wsc08.owl#con728014292

- http://www.ws-challenge.org/wsc08.owl#con2128128646

- http://www.ws-challenge.org/wsc08.owl#con2026950236

- http://www.ws-challenge.org/wsc08.owl#con2129152969

- http://www.ws-challenge.org/wsc08.owl#con995912074

- http://www.ws-challenge.org/wsc08.owl#con1179016734

And the requested output concepts:

- http://www.ws-challenge.org/wsc08.owl#con1406975733

- http://www.ws-challenge.org/wsc08.owl#con1403698683

- http://www.ws-challenge.org/wsc08.owl#con87763306

- http://www.ws-challenge.org/wsc08.owl#con1687981715

- http://www.ws-challenge.org/wsc08.owl#con2034528697

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
       xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
       xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
       xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
       xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
       xmlns:service="http://www.ws-challenge.org/WSC08Services/"
       targetNamespace="http://www.ws-challenge.org/WSC08Services/">
```

```
3    <service name="Task0Service">
4      <port binding="service:Task0SOAP" name="Task0Port">
5        <soap:address location="http://www.unknownexamplehost.ukn/" />
6      </port>
7    </service>
8    <binding name="Task0SOAP" type="service:Task0PortType">
9      <soap:binding style="rpc"
            transport="http://schemas.xmlsoap.org/soap/http" />
10     <operation name="Task0Operation">
11       <soap:operation soapAction="http://www.ws-challenge.org/Task0" />
12       <input>
13         <soap:body use="literal" />
14       </input>
15       <output>
16         <soap:body use="literal" />
17       </output>
18     </operation>
19   </binding>
20   <portType name="Task0PortType">
21     <operation name="Task0Operation">
22       <input message="service:Task0RequestMessage" />
23       <output message="service:Task0ResponseMessage" />
24     </operation>
25   </portType>
26   <message name="Task0RequestMessage">
27     <part element="service:ComplexElement0" name="ComplexElement0Part" />
28     <part element="service:con728014292" name="con728014292Part" />
29     <part element="service:con2128128646" name="con2128128646Part" />
30   </message>
31   <message name="Task0ResponseMessage">
32     <part element="service:con1406975733" name="con1406975733Part" />
33     <part element="service:con1403698683" name="con1403698683Part" />
34     <part element="service:con87763306" name="con87763306Part" />
35     <part element="service:con1687981715" name="con1687981715Part" />
36     <part element="service:con2034528697" name="con2034528697Part" />
37   </message>
38   <types>
39     <xs:schema
            targetNamespace="http://www.ws-challenge.org/WSC08Services/">
40       <xs:element name="ComplexElement0" minOccurs="0"
              maxOccurs="unbounded">
41         <xs:complexType>
42           <xs:sequence>
43             <xs:element name="con2026950236" type="xs:string" />
44             <xs:element name="con2129152969" type="xs:string" />
45             <xs:element name="con995912074" type="xs:string" />
46             <xs:element name="con1179016734" type="xs:string" />
47           </xs:sequence>
48         </xs:complexType>
49       </xs:element>
50       <xs:element name="con728014292" type="xs:string" />
51       <xs:element name="con2128128646" type="xs:string" />
52       <xs:element name="con1406975733" type="xs:string" />
53       <xs:element name="con1403698683" type="xs:string" />
54       <xs:element name="con87763306" type="xs:string" />
55       <xs:element name="con1687981715" type="xs:string" />
56       <xs:element name="con2034528697" type="xs:string" />
57     </xs:schema>
58   </types>
59   <mece:semExtension xmlns:mece="http://www.vs.uni-kassel.de/mece">
60     ...
61   </mece:semExtension>
62 </definitions>
```

Listing 13: The WSDL-Challenge Document

Line 1-62:   The WSDL challenge document.

Line 3-7:   Contains the requested service compositions. A challenge document may
             contain multiple challenges.

Line 8-19: A dummy binding section. In the real world this may describe the client's communication protocol.

Line 20-25: A dummy port type section.

Line 26-30: A dummy message section describing the client's request message respectively the given challenge's input concepts.

Line 31-37: A dummy message section describing the service's response message respectively the requested composition output concepts.

Line 38-58: A dummy types section. This is necessary for the semantic annotation.

Line 59-61: The semantic extension section. See Listing 14

```
1   <?xml version="1.0" encoding="UTF-8"?>
2     ...
3     <mece:semExtension xmlns:mece="http://www.vs.uni-kassel.de/mece">
4       <mece:semMessageExt id="Task0RequestMessage">
5         <mece:semExt id="con728014292">
6           <mece:ontologyRef>
7             http://www.ws-challenge.org/wsc08.owl#con728014292
8           </mece:ontologyRef>
9         </mece:semExt>
10        <mece:semExt id="con2128128646">
11          <mece:ontologyRef>
12            http://www.ws-challenge.org/wsc08.owl#con2128128646
13          </mece:ontologyRef>
14        </mece:semExt>
15        <mece:semExt id="con2026950236">
16          <mece:ontologyRef>
17            http://www.ws-challenge.org/wsc08.owl#con2026950236
18          </mece:ontologyRef>
19        </mece:semExt>
20        <mece:semExt id="con2129152969">
21          <mece:ontologyRef>
22            http://www.ws-challenge.org/wsc08.owl#con2129152969
23          </mece:ontologyRef>
24        </mece:semExt>
25        <mece:semExt id="con995912074">
26          <mece:ontologyRef>
27            http://www.ws-challenge.org/wsc08.owl#con995912074
28          </mece:ontologyRef>
29        </mece:semExt>
30        <mece:semExt id="con1179016734">
31          <mece:ontologyRef>
32            http://www.ws-challenge.org/wsc08.owl#con1179016734
33          </mece:ontologyRef>
34        </mece:semExt>
35      </mece:semMessageExt>
36      <mece:semMessageExt id="Task0ResponseMessage">
37        <mece:semExt id="con1406975733">
38          <mece:ontologyRef>
39            http://www.ws-challenge.org/wsc08.owl#con1406975733
40          </mece:ontologyRef>
41        </mece:semExt>
42        <mece:semExt id="con1403698683">
43          <mece:ontologyRef>
44            http://www.ws-challenge.org/wsc08.owl#con1403698683
45          </mece:ontologyRef>
46        </mece:semExt>
47        <mece:semExt id="con87763306">
48          <mece:ontologyRef>
49            http://www.ws-challenge.org/wsc08.owl#con87763306
50          </mece:ontologyRef>
51        </mece:semExt>
52        <mece:semExt id="con1687981715">
53          <mece:ontologyRef>
54            http://www.ws-challenge.org/wsc08.owl#con1687981715
55          </mece:ontologyRef>
56        </mece:semExt>
57        <mece:semExt id="con2034528697">
58          <mece:ontologyRef>
```

```
59            http://www.ws-challenge.org/wsc08.owl#con2034528697
60          </mece:ontologyRef>
61        </mece:semExt>
62      </mece:semMessageExt>
63    </mece:semExtension>
64  </definitions>
```
Listing 14: The WSDL-Challenge Semantic Extension

Line 3-63: The WSDL challenge document's semantic extension and parameter annotation.

Line 4-35: A message section describing the client's request message respectively the given challenge's input concepts.

Line 5-9: Input concept.

Line 36-62: A message section describing the service's response message respectively the requested composition output concepts.

Line 37-41: Output concept.

9   The Solution Format

During the WS-Challenge 2007, many participants encouraged the usage of a process language like BPEL as the 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, hence, improves the reusability of the challenge implementations. We already use OWL and an extension to the WSDL standard in the 2008 challenge, so we also use a common Web Service standard here. Below we illustrate the subset of WSBPEL as the output format in the 2008 Web Service challenge.

- In WSBPEL we can specify concurrent execution of services which is a desired feature for the next challenge.

- We present a subset of WSBPEL. We omit specification details like partner-links and copy instructions on message elements but apart from that, the solution file is pure WSBPEL.

- There is a close connection between the solution format of the last challenge and the WSBPEL subset, so adapting solutions to this format is rather simple.

We illustrate the current solution format with an example of the last years solution format:

```
1   <!-- Document root -->
2   <WSChallenge type="solutions">
3
4     <!-- First solution -->
5     <case name="SolutionA">
6       <service name="serviceA"/>
7       <serviceSpec name="2.1">
8         <service name="serviceB"/>
9         <serviceSpec name="2.2">
10          <service name="serviceC"/>
11          <service name="serviceD"/>
12        </serviceSpec>
13        <service name="serviceE"/>
14      </serviceSpec>
15    </case>
16
17    <!-- Second alternative solution -->
18    <case name="SolutionB">
19      ...
```

```
20        </case>
21   </WSChallenge>
```

Listing 15: Example Solution

In this example we have two possible solutions.

1. SolutionA: First we invoke the serviceA then invoke serviceB. Afterward we execute
   step 2.2 or service. In step 2.2 we can freely choose between serviceC and serviced.
   Additionally, we can (dependent on the query and test set) execute serviceB and
   service concurrently. But this cannot be expressed with this solution format.

2. SolutionB is omitted here.

```
1    <!-- Document Root -->
2    <process name="MoreCreditsBP"
3      xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
4      targetNamespace="http://www.ws-challenge.org/WSC08CompositionSolution/">
5
6      <!-- Main sequence -->
7      <sequence name="main">
8
9        <!-- Starting BPEL invocation (Input: Challenge Query) -->
10       <receive name="receiveQuery"
11                   portType="SolutionProcess" variable="query"/>
12
13           <!-- Switch/Case operator for alternative solutions -->
14           <switch name="SolutionAlternatives-SolutionA-SolutionB">
15
16             <!-- First solution -->
17             <case name="Alternative-SolutionA">
18
19               <!-- Sequence for serviceSpec 2.0 -->
20               <sequence>
21                 <!-- Firstly invoking Service: servicepA -->
22                 <invoke name="servicepA"
23                            portType="seeWSDLFile"
24                            operation="seeWSDLFile"/>
25
26                     <!-- Parallel execution of serviceSpec 2.1 and
                           serviceE -->
27                     <flow>
28                       <!-- serviceSpec 2.1 -->
29                       <sequence>
30                         <invoke name="servicepB"
31                              portType="seeWSDLFile"
32                              operation="seeWSDLFile"/>
33
34                                             <!-- serviceSpec 2.2 -->
35                         <switch name="Alternative-Services">
36                           <case name="Execute-servicepC">
37                             <sequence>
38
39                               <!-- Trigger Service serviceC -->
40                               <invoke name="serviceC"
41                                       portType="seeWSDLFile"
42                                       operation="seeWSDLFile"/>
43                             </sequence>
44                           </case>
45                           <case name="Execute-servicepD">
46                             <sequence>
47                               <!-- OR Trigger Service serviceD -->
48                               <invoke name="serviceD"
49                                       portType="seeWSDLFile"
50                                       operation="seeWSDLFile"/>
51                             </sequence>
52                           </case>
53                         </switch>
54                       </sequence>
55                       <sequence>
56                         <invoke name="servicepE"
```

```
57                                    portType = " seeWSDLFile "
58                                    operation = " seeWSDLFile "/>
59                             </ sequence >
60                          </ flow >
61                   </ sequence >
62              </ case >
63
64              <!-- Second solution -->
65              < case name = " Alternative - SolutionB " >
66                   ...
67              </ case >
68          </ switch >
69      </ sequence >
70 </ process >
```

Listing 16: The WSBPEL document

Line 2-4:	Document root and namespace declarations for the WSBPEL document.
Line 7-10:	The process starts its "main" after receiving a message. So this is a generic start declaration for every solution file.
Line 14:	As we have two alternative solutions we start with a switch-element. It is a WSPEL flow control element for alternative execution of services.
Line 17:	Each alternative of a switch-element starts with a case-element.
Line 20:	The sequence-element is a flow control element for sequential step execution of the hierarchical lower flow control elements.
Line 22:	First execution of solution A by invoking serviceA.
Line 23:	The execution of a service is defined by the service name along with its `portType`- and operation-Identifier in the related WSDL-document. (See section 3).
Line 27:	The flow-element represents a WSBPEL flow control element for concurrent execution of the hierarchical lower flow control elements.

Note that:

- The mark **seeWSDLFile** must be filled with the appropriate identifiers.

- The value of the **name**-attribute may contain any kind of identifier.

- Solutions may be specified in different ways (with parallel execution or only sequences). We evaluate in comparison to the optimal process solution regardless of the execution time of the composed services.

References

[1] Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. *OWL Web Ontology Language Reference*. World Wide Web Consortium (W3C), February 10, 2004. W3C Recommendation. Online available at `http://www.w3.org/TR/2004/REC-owl-ref-20040210/` [accessed 2007-11-22].

[2] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, and François Yergeau. *Extensible Markup Language (XML) 1.0 (Fourth Edition)*. World Wide Web Consortium (W3C), September 29, 2007. W3C Recommendation. Online available at `http://www.w3.org/TR/2006/REC-xml-20060816` [accessed 2007-09-02].

[3] World Wide Web Consortium (W3C). *RDF Primer*, February 10, 2004. W3C Recommendation. Online available at `http://www.w3.org/TR/2004/REC-rdf-primer-20040210/` [accessed 2007-11-22].

[4] David C. Fallside and Priscilla Walmsley. *XML Schema Part 0: Primer Second Edition.* World Wide Web Consortium (W3C), second edition, October 28, 2004. W3C Recommendation. Online available at `http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/` [accessed 2007-09-02].

[5] World Wide Web Consortium (W3C). *RDF Vocabulary Description Language 1.0: RDF Schema*, February 10, 2004. W3C Recommendation. Online available at `http://www.w3.org/TR/2004/REC-rdf-schema-20040210/` [accessed 2007-11-22].

[6] T. Berners-Lee. *Universal Resource Identifiers in WWW – A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network as used in the World-Wide Web.* Network Working Group, June 1994. Request for Comments (RFC) 1630, Category: Informational. Online available at `http://tools.ietf.org/html/rfc1630` [accessed 2007-11-22]. See also [7].

[7] T. Berners-Lee, R. Fielding, and L. Masinter. *Uniform Resource Identifier (URI): Generic Syntax.* Network Working Group, January 2005. Request for Comments (RFC) 3986, Category: Standards Track, STD: 66, Updates: RFC 1738, Obsoletes: RFC 2732, RFC 2396, RFC 1808. Online available at `http://tools.ietf.org/html/rfc3986` [accessed 2007-11-22]. See also [6].

[8] World Wide Web Consortium (W3C). *SOAP Version 1.2 Part 0: Primer (Second Edition)*, April 27, 2007. W3C Recommendation. Online available at `http://www.w3.org/TR/2007/REC-soap12-part0-20070427/` [accessed 2007-09-02].