Information technology—Metamodel framework for interoperability (MFI) –

Part 7: Metamodel for service registration

Warning

This document is not an ISO International Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an International Standard. Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foreword</td>
<td>VI</td>
</tr>
<tr>
<td>Introduction</td>
<td>VII</td>
</tr>
<tr>
<td>1 Scope</td>
<td>1</td>
</tr>
<tr>
<td>2 Normative references</td>
<td>1</td>
</tr>
<tr>
<td>3 Terms, definitions and abbreviated terms</td>
<td>1</td>
</tr>
<tr>
<td>3.1 Terms and definitions</td>
<td>1</td>
</tr>
<tr>
<td>3.2 Abbreviated terms</td>
<td>4</td>
</tr>
<tr>
<td>4 Conformance</td>
<td>5</td>
</tr>
<tr>
<td>4.1 General</td>
<td>5</td>
</tr>
<tr>
<td>4.2 Degree of conformance</td>
<td>5</td>
</tr>
<tr>
<td>4.3 Implementation Conformance Statement (ICS)</td>
<td>5</td>
</tr>
<tr>
<td>5 Structure of MFI Service registration</td>
<td>6</td>
</tr>
<tr>
<td>5.1 Overview of MFI Service registration</td>
<td>6</td>
</tr>
<tr>
<td>5.2 Associations between MFI Service registration and other parts in MFI</td>
<td>7</td>
</tr>
<tr>
<td>5.3 MFI Service registration</td>
<td>8</td>
</tr>
<tr>
<td>5.3.1 Atomic_Expression</td>
<td>8</td>
</tr>
<tr>
<td>5.3.2 Composite_Expression</td>
<td>8</td>
</tr>
<tr>
<td>5.3.3 Exit_Condition</td>
<td>9</td>
</tr>
<tr>
<td>5.3.4 Expression</td>
<td>9</td>
</tr>
<tr>
<td>5.3.5 Input_Message</td>
<td>10</td>
</tr>
<tr>
<td>5.3.6 Message_Type</td>
<td>11</td>
</tr>
<tr>
<td>5.3.7 Output_Message</td>
<td>12</td>
</tr>
<tr>
<td>5.3.8 Postcondition</td>
<td>12</td>
</tr>
<tr>
<td>5.3.9 Precondition</td>
<td>13</td>
</tr>
<tr>
<td>5.3.10 QoS_Assertion</td>
<td>14</td>
</tr>
<tr>
<td>5.3.11 QoS_Type</td>
<td>14</td>
</tr>
<tr>
<td>5.3.12 Service</td>
<td>15</td>
</tr>
<tr>
<td>5.3.13 Service_Description_Language</td>
<td>16</td>
</tr>
<tr>
<td>5.3.14 Service_Model</td>
<td>16</td>
</tr>
<tr>
<td>5.3.15 Service_Operation</td>
<td>17</td>
</tr>
</tbody>
</table>
5.3.16 Service_Type .................................................................................................................. 18
5.3.17 User_Tag .......................................................................................................................... 18

Annex A ........................................................................................................................................ 19
Annex B ........................................................................................................................................ 20
Bibliography ............................................................................................................................... 35
Figures and Tables

Figure 1 – Scope of MFI Service registration ........................................................................................................ 1  
Figure 2 – Metamodel of MFI service registration .................................................................................................. 6  
Figure 3 – The associations between MFI Service registration and MFI Process model registration and MFI  
Figure 4 – The associations between MFI Service registration and MFI Core and mapping .......................... 8  
Figure B.1 – The service model of Book_Ticket in WSML notation (fragment) .................................................. 20  
Figure B.2 – Registration of the Book_Ticket example (Part 1 of 2) ................................................................. 21  
Figure B.2 – Registration of the Book_Ticket example (Part 2 of 2) ................................................................. 22  
Figure B.3 – The service model of Search_item in WADL notation (fragment) .................................................. 23  
Figure B.4 – Registration of the Search_Item example ....................................................................................... 24  
Figure B.5 – The service model of Congo_BookBuying_Agent in OWL-S notation (fragment) (Part 1 of 4) .. 25  
Figure B.5 – The service model of Congo_BookBuying_Agent in OWL-S notation (fragment) (Part 2 of 4) .. 26  
Figure B.5 – The service model of Congo_BookBuying_Agent in OWL-S notation (fragment) (Part 3 of 4) .. 27  
Figure B.5 – The service model of Congo_BookBuying_Agent in OWL-S notation (fragment) (Part 4 of 4) .. 28  
Figure B.6 – Registration of the Congo_BookBuying_Agent example (Part 1 of 2) .......................................... 29  
Figure B.6 – Registration of the Congo_BookBuying_Agent example (Part 2 of 2) .......................................... 30  
Figure B.7 – The service model of Hotel_Reservation in WSDL 2.0 notation (fragment) (Part 2 of 2) .......... 33  
Figure B.8 – Registration of the Hotel_Reservation example ........................................................................... 34  

Table A.1 – List of existing service description languages ................................................................................. 19
Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75% of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 19763-7 was prepared by Joint Technical Committee ISO/IEC JTC 1, Information Technology, Subcommittee SC 32, Data management and Interchange.

ISO/IEC 19763 consists of the following parts, under the general title Information technology — Metamodel framework for interoperability (MFI):

Part 1: Framework

Part 3: Metamodel for ontology registration

Part 5: Metamodel for process model registration

Part 6: Registry Summary

Part 7: Metamodel for service registration

Part 8: Metamodel for role and goal registration


Part 10: Core model and basic mapping

Part 12: Metamodel for information model registration

Part 13: Metamodel for forms registration
**Introduction**

With the rapid development of SOC (Service Oriented Computing), more and more computing resources are presented in the form of web services. Meanwhile, business integration based on web services is becoming a popular application development method. A web service is a kind of Web based application which encapsulates one or more computing modules and is designed to support interoperable machine-to-machine interaction over a network.

In Web Service registration and management, ebXML RegRep is a standard defining the Service interface, protocols and information model for an integrated registry and repository, which provides basic support for publishing and discovering Web Services within and across enterprises. Nevertheless, keyword matching is the basic service discovery method in ebXML RegRep, thus the discovery results will be inevitably inaccurate, and the discovery process will be time-consuming. When business information interchange and integration becomes increasingly frequent, major work in service discovery should be processed by machine, therefore, it is necessary to semantically describe service information, including functional and non-functional information, and provide corresponding registration and management mechanism.

This part of ISO/IEC 19763 intends to provide a generic framework for registering functional and non-functional information about services in an explicit way.
Information technology – Metamodel framework for interoperability (MFI) – Part 7: Metamodel for service registration

1 Scope

The primary purpose of the multipart standard ISO/IEC 19763 is to specify a metamodel framework for interoperability. This part of ISO/IEC 19763 specifies a metamodel for registering services, facilitating interoperability through the reuse of services.

This part of ISO/IEC 19763 can be applicable only for Web Services whose capabilities are described by some web service description language, see Annex A for examples. Figure 1 shows the scope of this part of ISO/IEC 19763.

NOTE: Not every model needs to exist in a repository before registration

Figure 1 – Scope of MFI Service registration

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 19763-10, Information technology – Metamodel framework for interoperability (MFI) – Part 10: Core model and basic mapping

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

For the purpose of this document, the terms and definitions given in ISO/IEC 19763-3:2010, ISO/IEC 19763-5,
ISO/IEC 19763-8, ISO/IEC 19763-10 and the following apply.

3.1.1
atomic expression
logical expression (3.1.5) that has the unit granularity and similarly throughout

3.1.2
composite expression
logical expression (3.1.5) that comprises multiple atomic expressions (3.1.1) and/or other composite expressions (3.1.2) by using connectives such as conjunction, disjunction and negation

3.1.3
entity service
Web service (3.1.17) that bases its functional boundary on one or more related organization entities, such as customer, employee, invoice and claim

3.1.4
exit condition
constraint that, if true, will cause an operation to finish un成功fully

NOTE The operation can be a process or a service operation

3.1.5
expression
sentence which is expressed using a logical notation to specify either a condition that applies to a service operation (3.1.18) or a quality of service (3.1.17) that applies to a service

3.1.6
goal
descriptive statement of intended outcome of user interaction with a process (3.1.13) or service (3.1.17)

[ISO/IEC 19763-8, 3.1.1]

3.1.7
input message
information contained in the message that the service operation (3.1.18) consumes for its execution

3.1.8
involvement type
statement that indicates the type of involvement of a role with a process (3.1.13) or service (3.1.17)

[ISO/IEC 19763-8, 3.1.4]

NOTE examples are performer, beneficiary, customer

3.1.9
message type
type of the message that is consumed or generated for execution of service operation (3.1.18)

3.1.10
output message
information contained in the message that the service operation (3.1.18) generates after its execution

3.1.11
postcondition
constraint that must be true at the completion of an operation
NOTE The operation can be a process or a service operation.

3.1.12
precondition
constraint that must be true when an operation is invoked

NOTE The operation can be a process or a service operation.

3.1.13
process
collection of related, structured activities or tasks that achieve a particular goal (3.1.6)

3.1.14
QoS assertion
specification of one or more qos types for the service (3.1.17)

3.1.15
QoS type
specified non-functional property for a service (3.1.17), such as availability, response time, etc.

3.1.16
role
expression of an actor playing a part in a social relationship

NOTE adapted from [ISO/IEC 15476-4:2005 6.5]

3.1.17
service
kind of Web based application, which encapsulates one or more computing modules and can be accessed through a specified interface

3.1.18
service operation
execution action of a service (3.1.17)

3.1.19
task service
Web service (3.1.17) with a functional boundary directly associated with a process model

3.1.20
user tag
tag annotated by an individual or organization in order to describe the service (3.1.17) according to the understanding of the creator of the tag

3.1.21
utility service
Web service (3.1.17) that is dedicated to provide reusable, cross-cutting utility functionality, such as event logging, notification, and except handling
3.2 Abbreviated terms

**ebXML RegRep**

ebXML Registry and Repository

[OASIS ebXML RegRep Version 4.0: 2012]

**IRI**

Internationalized Resource Identifier

[W3C RFC 3987: 2005]

**MFI Core and mapping**

ISO/IEC 19763-10, Information technology – Metamodel framework for interoperability (MFI) – Part 10: Core model and basic mapping

**MFI Process model registration**

ISO/IEC 19763-5, Information technology – Metamodel framework for interoperability (MFI) – Part 5: Metamodel for process model registration

**MFI Role and Goal registration**

ISO/IEC 19763-8, Information technology – Metamodel framework for interoperability (MFI) – Part 8: Metamodel for role and goal registration

**MFI Service registration**

ISO/IEC 19763-7, Information technology – Metamodel framework for interoperability (MFI) – Part 7: Metamodel for service registration

**OWL-S**

Web Ontology Language for Services

**QoS**

Quality of Service

**SWRL**

Semantic Web Rule Language

**SWSL**

Semantic Web Service Language

**WADL**

Web Application Description Language
Conformance

General

An implementation claiming conformance with this part of ISO/IEC 19763 shall support the metamodel specified in clause 5, depending on a degree of conformance as described below.

4.2 Degree of conformance

4.2.1 General

The distinction between ‘strictly conforming’ and ‘conforming’ implementations is necessary to address the simultaneous needs for interoperability and extensions. This part of ISO/IEC 19763 describes specifications that promote interoperability. Extensions are motivated by needs of users, vendors, institutions and industries, but are not specified by this part of ISO/IEC 19763.

A strictly conforming implementation may be limited in usefulness but is maximally interoperable with respect to this part of ISO/IEC 19763. A conforming implementation may be more useful, but may be less interoperable with respect to this part of ISO/IEC 19763.

4.2.2 Strictly conforming implementation

A strictly conforming implementation

a) shall support the metamodel specified in clause 5;

b) shall not use, test, access, or probe for any extension features nor extensions to the metamodel specified in clause 5.

4.2.3 Conforming implementation

A conforming implementation

a) shall support the metamodel specified in clause 5;

b) as permitted by the implementation, may use, test, access, or probe for any extension features or extensions to the metamodel specified in clause 5.

NOTE 1 All strictly conforming implementations are also conforming implementations.

NOTE 2 The use of extensions to the metamodel might cause undefined behaviour.

4.3 Implementation Conformance Statement (ICS)

An implementation claiming conformance with this part of ISO/IEC 19763 shall include an Implementation Conformance Statement stating

a) whether it is a strictly conforming implementation in clause 4.2.2 or a conforming implementation in clause 4.2.3;

b) what extensions, if any, are supported or used if it is a conforming implementation.
5 Structure of MFI Service registration

5.1 Overview of MFI Service registration

This part of MFI specifies the metamodel that can be used to register functional and non-functional information about services. Examples of some service description languages that can be registered using this metamodel are listed in Annex A.

Figure 2 shows the metamodel for the registration of services. This metamodel allows the registration of the common functional and non-functional features of services described using a number of service description languages. Each service model, expressed using a specific service description language, may describe one or more services. Each service is comprised of one or more service operations. Each service is described by zero or one QoS assertion. This QoS assertion is used to represent the quantitative or qualitative non-functional features of the service, such as response time, cost, reliability, etc. Each QoS assertion is defined using one and only one expression, which may be a composite expression or an atomic expression. Each QoS assertion is of one and only one expression, which may be a

A service is an independent and modular component and it can be accessed only by interfaces. For this reason the functional capability of a service is expressed using service operations, where each service operation denotes an execution action of the service. Each service is comprised of zero, one or more service operations. Each service operation is described with zero or one pre-condition, with zero or one post condition and with zero or one exit condition. A pre-condition specifies a constraint that must be true when a service operation is invoked. A postcondition
specifies a constraint that must be true at the completion of a service operation, and an exit condition specifies a constraint that, if true, will cause an operation to finish unsuccessfully. Each precondition, each postcondition and each exit condition is defined using one and only one expression, which may be a composite expression or an atomic expression. Each service operation is also described with zero, one or more input messages and with zero, one or more output messages. Each input message specifies information that the operation needs for its execution. Each output message specifies information that the operation generates after its successful execution. Each message type represents data type of the message, including input message and output message. Each input message is constrained by zero, one or more pre-conditions and each output message is constrained by zero, one or more post-conditions. Each service can be annotated by zero, one or more user tags, each of which may be created by all any person using the service.

5.2 Associations between MFI Service registration and other parts in MFI

Figure 3 shows the associations between MFI Service registration (this part) and MFI Role and Goal registration and MFI Process model registration.

Each service achieves zero, one or more goals. Each goal is achieved by zero, one or more services. Each service operation achieves zero, one or more goals. Each goal is achieved by zero, one or more service operations. Each service operation can fully realize zero, one or more processes. Each process is fully realized by zero, one or more service operations. Each service involves zero, one or more service involvements, where each service involvement is the involvement of a role with a service, such as actor or beneficiary. Each service involvement indicates that a role is involved in the execution of one and only one service.

The association between the metaclasses in MFI Service registration and the metaclasses in MFI Core and mapping is shown in Figure 4.
5.3 MFI Service registration

5.3.1 Atomic_Expression

Atomic_Expression is a metaclass each instance of which represents a logical expression that has the unit granularity.

**SuperClass**

Expression

<table>
<thead>
<tr>
<th>Attribute</th>
<th>DataType</th>
<th>Multiplicity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>expression_</td>
<td>string</td>
<td>1..1</td>
<td>The text of logical expression which is described by a kind of notation of text</td>
</tr>
</tbody>
</table>

**Constraints**

[None]

5.3.2 Composite_Expression

Composite_Expression is a metaclass each instance of which represents a logical expression that comprises multiple atomic expressions and/or other composite expressions by using composition types such as conjunction, disjunction and negation.

**SuperClass**
## Expression

<table>
<thead>
<tr>
<th>Attribute</th>
<th>DataType</th>
<th>Multiplicity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>composition_type</td>
<td>string</td>
<td>1..1</td>
<td>The symbol which is used to connect two or more logical expressions</td>
</tr>
</tbody>
</table>

### Reference

<table>
<thead>
<tr>
<th>Reference</th>
<th>Class</th>
<th>Multiplicity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>composing_expression</td>
<td>Expression</td>
<td>1..*</td>
<td>The set of expression, each of which is a sentence that is expressed using a logical notation to specify either a condition that applies to a service operation or a quality of service</td>
</tr>
</tbody>
</table>

### Constraints

The value of composition_type must be one of ‘conjunction’, ‘disjunction’ or ‘negation’.

### 5.3.3 Exit_Condition

Exit_Condition is a metaclass each instance of which specifies the constraint that, if true, will cause the operation to finish unsuccessfully.

### Superclass

Model_Element (from MFI Core and mapping)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>DataType</th>
<th>Multiplicity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>0..1</td>
<td>The name for the exit condition of service</td>
</tr>
</tbody>
</table>

### Reference

<table>
<thead>
<tr>
<th>Reference</th>
<th>Class</th>
<th>Multiplicity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>exit_condition_logical_expression</td>
<td>Expression</td>
<td>1..1</td>
<td>The sentence which is described by a kind of logical notation to express this exit condition</td>
</tr>
<tr>
<td>containing_service_operation</td>
<td>Service_Operation</td>
<td>1..1</td>
<td>The execution action of this service which contains this exit condition</td>
</tr>
</tbody>
</table>

### Constraints

[None]

### 5.3.4 Expression

Expression is an abstract metaclass which represents a sentence that is expressed using a logical notation to specify
either a condition that applies to a service operation or a QoS that applies to a service.

**SuperClass**

Model_Element (from MFI Core and mapping)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>DataType</th>
<th>Multiplicity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>notation</td>
<td>string</td>
<td>1..*</td>
<td>The logic language or notation that is used to declare a quality assertion,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>a precondition, a postcondition or an exit condition of a service</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reference</th>
<th>Class</th>
<th>Multiplicity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>expressed_</td>
<td>Precondition</td>
<td>0..1</td>
<td>The constraint that must be true when an operation is invoked</td>
</tr>
<tr>
<td>precondition</td>
<td></td>
<td></td>
<td>logical_ expression</td>
</tr>
<tr>
<td>expressed_</td>
<td>Postcondition</td>
<td>0..1</td>
<td>The constraint that must be true at the completion of an operation</td>
</tr>
<tr>
<td>postcondition</td>
<td></td>
<td></td>
<td>logical_ expression</td>
</tr>
<tr>
<td>expressed_</td>
<td>Exit_Condition</td>
<td>0..1</td>
<td>The constraint that must be true when a service operation exists abnormally</td>
</tr>
<tr>
<td>exit_condition</td>
<td></td>
<td></td>
<td>logical_ expression</td>
</tr>
<tr>
<td>expressed_</td>
<td>QoS_</td>
<td>0..1</td>
<td>The specification of one or more QoS types for the service</td>
</tr>
<tr>
<td>qos_assertion</td>
<td>Assertion</td>
<td></td>
<td>qos_logical_ expression</td>
</tr>
<tr>
<td>composed_</td>
<td>Composite_</td>
<td>0..1</td>
<td>The logical expression that comprises multiple atomic expressions</td>
</tr>
<tr>
<td>expression</td>
<td>Expression</td>
<td></td>
<td>composing_ Yes</td>
</tr>
</tbody>
</table>

**Constraints**

[None]

**5.3.5 Input_Message**

Input_Message is a metaclass each instance of which specifies information contained in the message that the service operation consumes for its execution.

**SuperClass**
5.3.6 Message_Type

Message_Type is a metaclass each instance of which represents a data type of the message that is consumed or generated for the execution of a service operation.

Superclass

Model_Element (from MFI Core and mapping)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>DataType</th>
<th>Multiplicity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>1..1</td>
<td>The name of a message type</td>
</tr>
<tr>
<td>message_type_description</td>
<td>string</td>
<td>0..1</td>
<td>The description of a message type</td>
</tr>
</tbody>
</table>

Reference

<table>
<thead>
<tr>
<th>Reference</th>
<th>Class</th>
<th>Multiplicity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>involving_input_message</td>
<td>Input_ Message</td>
<td>0..*</td>
<td>The set of input messages, each of which belongs to this message type</td>
</tr>
</tbody>
</table>

Constraints

[None]
message Message messages, each of which message_type belongs to this message type

Constraints

[None]

5.3.7 Output_Message

Output_Message is a metaclass each instance of which specifies information contained in the message that the service operation generates after its execution.

SuperClass

Model_Element (from MFI Core and mapping)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>DataType</th>
<th>Multiplicity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>1..1</td>
<td>The name of the message that is generated after the execution of a service operation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reference</th>
<th>Class</th>
<th>Multiplicity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>containing_service_operation</td>
<td>Service_</td>
<td>1..1</td>
<td>The service operation that generates this output message</td>
</tr>
<tr>
<td></td>
<td>Operation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>constraining_postcondition</td>
<td>Postcondition</td>
<td>0..*</td>
<td>The set of postconditions, each of which constraints this output message when the service operation is invoked</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>involved_output_message_type</td>
<td>Message_</td>
<td>0..1</td>
<td>The type of this output message</td>
</tr>
<tr>
<td></td>
<td>Type</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Constraints

[None]

5.3.8 Postcondition

Postcondition is a metaclass each instance of which specifies the constraint that must be true at the completion of an operation.

Superclass

Model_Element (from MFI Core and mapping)
Table for Postcondition:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>DataType</th>
<th>Multiplicity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>0..1</td>
<td>The name for the postcondition of a service</td>
</tr>
<tr>
<td>referenced_</td>
<td>Class</td>
<td></td>
<td>Description Inverse Precedence</td>
</tr>
<tr>
<td>postcondition_</td>
<td>Expression</td>
<td>1..1</td>
<td>The sentence which is described by logic notation to express this postcondition</td>
</tr>
<tr>
<td>logical_expression</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>constrained_</td>
<td>Output_</td>
<td>0..*</td>
<td>The set of output messages, each of which is constrained by this postcondition</td>
</tr>
<tr>
<td>output</td>
<td>Message</td>
<td></td>
<td></td>
</tr>
<tr>
<td>containing_service_</td>
<td>Service_</td>
<td>1..1</td>
<td>The execution action of the service which constrains this postcondition</td>
</tr>
<tr>
<td>operation</td>
<td>Operation</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Constraints:

[None]

5.3.9 Precondition

Precondition is a metaclass each instance of which specifies the constraint that must be true when an operation is invoked.

Superclass

Model_Element (from MFI Core and mapping)
5.3.10 QoS_Assertion

QoS_Assertion is a metaclass each instance of which represents a specification of one or more QoS types of a service.

SuperClass

Model_Element (from MFI Core and mapping)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>DataType</th>
<th>Multiplicity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>used_qos_type</td>
<td>QoS_Type</td>
<td>1..*</td>
<td>The set of QoS types, each of which is involved in this QoS assertion</td>
</tr>
<tr>
<td>qos_logical_expression</td>
<td>Expression</td>
<td>1..1</td>
<td>The sentence which is described by a kind of logic notation</td>
</tr>
<tr>
<td>asserted_service</td>
<td>Service</td>
<td>1..1</td>
<td>The service whose quality is asserted by this QoS assertion</td>
</tr>
</tbody>
</table>

Constraints

[None]

5.3.11 QoS_Type

QoS_Type is a metaclass each instance of which is used to represent a specified non-functional property for a service, such as availability, response time, etc.

SuperClass

Model_Element (from MFI Core and mapping)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>DataType</th>
<th>Multiplicity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>1..1</td>
<td>The name of the type of a non-functional property</td>
</tr>
</tbody>
</table>

Reference

<table>
<thead>
<tr>
<th>Class</th>
<th>Multiplicity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>describing_qos_assertion</td>
<td>0..*</td>
<td>The set of assertions, each of which is a description about this non-functional property</td>
</tr>
</tbody>
</table>

Constraints

[None]
5.3.12 Service

Service is a metaclass each instance of which represents a kind of web based application which encapsulates one or more computing modules and can be accessed through a specified interface.

Superclass

Model_Element (from MFI Core and mapping)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Data Type</th>
<th>Multiplicity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>1..1</td>
<td>The name of the service</td>
</tr>
<tr>
<td>requested_IRI</td>
<td>string</td>
<td>1..1</td>
<td>The IRI for invoking the service</td>
</tr>
<tr>
<td>domain</td>
<td>string</td>
<td>1..1</td>
<td>The domain that the service belongs to (see bibliography [1])</td>
</tr>
<tr>
<td>service_type</td>
<td>Service_Type</td>
<td>1..1</td>
<td>The type of the service</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reference</th>
<th>Class</th>
<th>Multiplicity</th>
<th>Description</th>
<th>Inverse</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>owned_qos_</td>
<td>QoS_</td>
<td>0..1</td>
<td>The QoS assertion that applies to this service</td>
<td>asserted service</td>
<td>Yes</td>
</tr>
<tr>
<td>assertion</td>
<td>Assertion</td>
<td></td>
<td>The service which contains the assertion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>contained_</td>
<td>Service_</td>
<td>0..*</td>
<td>The set of service operations, each of which denotes an execution action of this service</td>
<td>containing service</td>
<td>Yes</td>
</tr>
<tr>
<td>service_operation</td>
<td>Operation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>service_tag</td>
<td>User_Tag</td>
<td>0..*</td>
<td>The set of tags, each of which is tagged and annotated by an individual or an organization in order to describe this service</td>
<td></td>
<td></td>
</tr>
<tr>
<td>composing_</td>
<td>Service</td>
<td>0..*</td>
<td>The set of services, each of which is a component of this service</td>
<td>composed service</td>
<td>Yes</td>
</tr>
<tr>
<td>service</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>composed_</td>
<td>Service</td>
<td>0..1</td>
<td>The composite service which contains this service</td>
<td>composed service</td>
<td>No</td>
</tr>
<tr>
<td>service</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>containing_</td>
<td>Service_Model</td>
<td>0..*</td>
<td>The set of service models, each of which is used to model this service</td>
<td>containing service</td>
<td>No</td>
</tr>
<tr>
<td>model</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>achieved_goal</td>
<td>Goal(from MFI Role and Goal registration)</td>
<td>0..*</td>
<td>The set of goals, each of which is achieved by this service.</td>
<td>achieving service</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Constraints
The value of attribute "requested_IRI" has to be unique in this metaclass.

### 5.3.13 Service_Description_Language

Service_Description_Language is a metaclass each instance of which represents a language or a notation that is used to model a service.

**SuperClass**

Modelling_Language (from MFI Core and mapping)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>DataType</th>
<th>Multiplicity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>1..1</td>
<td>The name of a service description language</td>
</tr>
</tbody>
</table>

**Reference**

<table>
<thead>
<tr>
<th>Class</th>
<th>Multiplicity</th>
<th>Description</th>
<th>Inverse</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>expressed_language</td>
<td>0..*</td>
<td>The model that is described by this language</td>
<td>describing_language</td>
<td>No</td>
</tr>
</tbody>
</table>

**Constraints**

[None]

### 5.3.14 Service_Model

Service_Model is a metaclass each instance of which represents a model that is used to model services that can be modelled.

**SuperClass**

Model (from MFI Core and mapping)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>DataType</th>
<th>Multiplicity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>1..1</td>
<td>The name of a service model</td>
</tr>
<tr>
<td>described_IRI</td>
<td>string</td>
<td>0..1</td>
<td>The IRI that identifies the corresponding service model</td>
</tr>
</tbody>
</table>

**Reference**

<table>
<thead>
<tr>
<th>Class</th>
<th>Multiplicity</th>
<th>Description</th>
<th>Inverse</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>expressing_language</td>
<td>1..1</td>
<td>The language or notation that is used to model this service</td>
<td>expressing_language</td>
<td>Yes</td>
</tr>
<tr>
<td>language</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>containing_language</td>
<td>1..*</td>
<td>The set of services, each of which is modelled by this service</td>
<td>containing_language</td>
<td>Yes</td>
</tr>
<tr>
<td>service</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Constraints**

The value of attribute 'described_IRI' has to be unique in this metaclass.
5.3.15 Service_Operation

Service_Operation is a metaclass each instance of which denotes the execution actions of a service.

SuperClass

Model_Element (from MFI Core and mapping)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>DataType</th>
<th>Multiplicity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>1..1</td>
<td>The name of a service operation.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reference</th>
<th>Class</th>
<th>Multiplicity</th>
<th>Description</th>
<th>Inverse</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>consumed_</td>
<td>Input_</td>
<td>0..*</td>
<td>The set of information, each of which is contained in the message that is consumed by this service operation for its execution.</td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>consumed_</td>
<td>Message</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>generated_</td>
<td>Output_</td>
<td>0..*</td>
<td>The set of information, each of which is contained in the message that is generated by this service operation after its execution</td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>generated_</td>
<td>Message</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>contained_</td>
<td>Precondition</td>
<td>0..1</td>
<td>The constraint that must be true when this operation is invoked</td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>contained_</td>
<td>Postcondition</td>
<td>0..1</td>
<td>The constraint that must be true at the completion of this operation</td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>contained_</td>
<td>Exit_</td>
<td>0..1</td>
<td>The constraint that must be true when a service operation exists abnormally</td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>contained_</td>
<td>Service</td>
<td>1..1</td>
<td>The service that contains this service operation</td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>achieved_</td>
<td>Goal(from MFI Role and Goal registration)</td>
<td>0..*</td>
<td>The set of goals, each of which is achieved by this service operation.</td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>fully_</td>
<td>Process(from MFI Process model)</td>
<td>0..*</td>
<td>The set of processes, each of which is fully realized by this service operation.</td>
<td></td>
<td>Yes</td>
</tr>
</tbody>
</table>
Constraints

[None]

5.3.16 Service_Type
Service_Type is an enumerated datatype that is comprised of ‘entity_service’, ‘task_service’ and ‘utility_service’. ‘entity_service’
denotes the service that bases its functional boundary on one or more related organization entities, such as customer, employee,
invoice and claim. ‘task_service’ denotes the service with a functional boundary directly associated with a process model.
‘utility_service’ denotes the service that is dedicated to provide reusable and cross-cutting utility functionality, such as event
logging, notification, and except handling.

5.3.17 User_Tag
User_Tag is a metaclass each instance of which represents the tag annotated by an individual or an organization in order to
describe the service according to the understanding of the creator of the tag.

SuperClass

Model_Element (from MFI Core and mapping)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Data Type</th>
<th>Multiplicity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tag_creator</td>
<td>string</td>
<td>1..1</td>
<td>The user who tags the service</td>
</tr>
<tr>
<td>tag_text</td>
<td>string</td>
<td>0..*</td>
<td>The text that a user tags for the service</td>
</tr>
</tbody>
</table>

Reference

<table>
<thead>
<tr>
<th>Class</th>
<th>Multiplicity</th>
<th>Description</th>
<th>Inverse</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>tagged_service</td>
<td>Service</td>
<td>The set of services, each of which is tagged by this user tag</td>
<td>service_tag</td>
<td>No</td>
</tr>
</tbody>
</table>

Constraints

[None]
Annex A
(informative)

List of existing service description languages

This annex provides a list of existing service description languages that can be registered using the metamodel described in this part of ISO/IEC 19763, which are shown in Table A.1.

Table A.1 – List of existing service description languages

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWL-S</td>
<td>A language that conforms to ‘OWL Web Ontology Language for Web Service’, which specifying Semantic Markup for Web Services, 2004-11-02, W3C Member Submission</td>
</tr>
<tr>
<td>WSDL</td>
<td>Web Services Description Language (WSDL) Version 2.0 W3C Recommendation 26 June 2007</td>
</tr>
<tr>
<td>WSML</td>
<td>Web Service Modelling Language, 2005-06-03, W3C Member Submission</td>
</tr>
<tr>
<td>SWSL</td>
<td>Semantic Web Service Language, 2005-09-09, W3C Member Submission</td>
</tr>
<tr>
<td>WADL</td>
<td>Web Application Description Language, 2009-08-31, W3C Member Submission</td>
</tr>
<tr>
<td>SA-Rest</td>
<td>SA-REST: Semantic Annotation of Web Resources, 2010-04-05, W3C Member Submission</td>
</tr>
</tbody>
</table>
Annex B

(informative)

Registration examples

Case 1: Example of WSMO Service Registration

The Service is described in Web Service Modelling Ontology (WSMO). This example is about a scenario from the domain of e-tourism. An agent wants to buy a ticket from Innsbruck to Venice on a certain date. A hypothetical Web Service called the “Book Ticket Web Service” is considered for achieving the goal which specifies the intent of buying a ticket for a trip from Innsbruck to Venice. When its execution is successful, the ‘Book Ticket Web Service’ has a result for reservation that includes the reservation holder and a ticket for the desired trip (postcondition) if there is a reservation request for a trip with its starting point in Austria for a certain person (precondition) and if the credit card intended to be used for paying is a valid one and its type is either PlasticBuy or GoldenCard (assumption). As a consequence of the execution of the Web Service, the price of the ticket will be deduced from the credit card (effect).

The fragmentary code is shown in Figure B.1.

```wsml
webService _"http://example.org/bookTicketWebService"
  importsOntology _"http://example.org/tripReservationOntology"
  capability BookTicketCapability
  interface BookTicketInterface
  ...
  capability BookTicketCapability
  sharedVariables{?creditCard, ?initialBalance, ?trip, ?reservationHolder, ?ticket}
  precondition
    definedBy
      ?reservationRequest[reservationItem hasValue ?trip, reservationHolder hasValue ?reservationHolder] memberOf tr#reservationRequest and
      ?trip memberOf tr#tripFromAustria and
      ?creditCard[balance hasValue ?initialBalance] memberOf po#creditCard.
  assumption
    definedBy
      po#validCreditCard(?creditCard) and (?creditCard[type hasValue "PlasticBuy"] or
        ?creditCard[type hasValue "GoldenCard"]).
  postcondition
    definedBy
      ?reservation memberOf tr#reservation[reservationItem hasValue ?ticket, reservationHolder hasValue ?reservationHolder] and
      ?ticket [trip hasValue ?trip] memberOf tr#ticket.
  effect
    definedBy
      ticketPrice(?ticket, "euro", ?ticketPrice) and
      ?finalBalance= (?initialBalance - ?ticketPrice) and
      ?creditCard[po#balance hasValue ?finalBalance].
```

Figure B.1 – The service model of Book_Ticket in WSML notation (fragment)

The following shows how ‘Book Ticket Web Service’ is registered in terms of MFI Service registration. The detailed Service registration Information is shown as following:
<table>
<thead>
<tr>
<th>Object</th>
<th>Attribute/Reference</th>
<th>Literal/Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>name</td>
<td>&quot;WSML&quot;</td>
</tr>
<tr>
<td></td>
<td>expressed_model</td>
<td>Object102</td>
</tr>
<tr>
<td>102</td>
<td>name</td>
<td>&quot;bookTicketWebService&quot;</td>
</tr>
<tr>
<td></td>
<td>described_IRI</td>
<td>&quot;<a href="http://example.org/bookTicketWebService">http://example.org/bookTicketWebService</a>&quot;</td>
</tr>
<tr>
<td></td>
<td>describing_language</td>
<td>Object101</td>
</tr>
<tr>
<td></td>
<td>contained_service</td>
<td>Object103</td>
</tr>
<tr>
<td>103</td>
<td>name</td>
<td>&quot;Book Ticket&quot;</td>
</tr>
<tr>
<td></td>
<td>requested_IRI</td>
<td>&quot;<a href="http://example.org/bookTicketWebService.wsdl">http://example.org/bookTicketWebService.wsdl</a>&quot;</td>
</tr>
<tr>
<td></td>
<td>domain</td>
<td>&quot;Travel&quot;</td>
</tr>
<tr>
<td></td>
<td>service_type</td>
<td>&quot;Task_Service&quot;</td>
</tr>
<tr>
<td></td>
<td>containing_model</td>
<td>Object102</td>
</tr>
<tr>
<td></td>
<td>contained_service_operation</td>
<td>Object104</td>
</tr>
<tr>
<td>104</td>
<td>name</td>
<td>&quot;BookTicketCapability&quot;</td>
</tr>
<tr>
<td></td>
<td>containing_service</td>
<td>Object103</td>
</tr>
<tr>
<td></td>
<td>consumed_message</td>
<td>Object118, Object119, Object120</td>
</tr>
<tr>
<td></td>
<td>generated_message</td>
<td>Object121, Object122</td>
</tr>
<tr>
<td></td>
<td>contained_precondition</td>
<td>Object105</td>
</tr>
<tr>
<td></td>
<td>contained_postcondition</td>
<td>Object112</td>
</tr>
<tr>
<td>105</td>
<td>name</td>
<td>&quot;BookTicketWebService&quot;</td>
</tr>
<tr>
<td></td>
<td>containing_service_operation</td>
<td>Object104</td>
</tr>
<tr>
<td></td>
<td>precondition_logical_expression</td>
<td>Object11</td>
</tr>
<tr>
<td></td>
<td>constrained_input</td>
<td>Object19, Object20</td>
</tr>
<tr>
<td>106</td>
<td>name</td>
<td>&quot;WSML&quot;</td>
</tr>
<tr>
<td></td>
<td>expression_text</td>
<td>&quot;?trip hasValue ?tripFromAustria&quot;</td>
</tr>
<tr>
<td></td>
<td>composed_expression</td>
<td>Object111</td>
</tr>
<tr>
<td>107</td>
<td>name</td>
<td>&quot;WSML&quot;</td>
</tr>
<tr>
<td></td>
<td>expression_text</td>
<td>&quot;?poValidCreditCard(?creditCard)&quot;</td>
</tr>
<tr>
<td></td>
<td>composed_expression</td>
<td>Object111</td>
</tr>
<tr>
<td>108</td>
<td>name</td>
<td>&quot;WSML&quot;</td>
</tr>
<tr>
<td></td>
<td>expression_text</td>
<td>&quot;?creditCard(type hasValue &quot;GoldenCard&quot;)&quot;</td>
</tr>
<tr>
<td></td>
<td>composed_expression</td>
<td>Object110</td>
</tr>
<tr>
<td>109</td>
<td>name</td>
<td>&quot;WSML&quot;</td>
</tr>
<tr>
<td></td>
<td>expression_text</td>
<td>&quot;?creditCard(type hasValue &quot;PlasticBuy&quot;)&quot;</td>
</tr>
<tr>
<td></td>
<td>composed_expression</td>
<td>Object110</td>
</tr>
</tbody>
</table>

Figure B.2 – Registration of the Book_Ticket example (Part 1 of 2)
**Figure B.2 – Registration of the Book_Ticket example (Part 2 of 2)**
Case2: Example of WADL Service Registration

This service is described in Web Application Description Language (WADL). Presently, WADL only provides functional semantics about service. This service provided by Amazon is to search items in Amazon Internet shop. Browser will return item information after user inputs keywords or index of item. The fragmentary code is shown in Table B.3.

```
<application xmlns=http://wadl.dev.java.net/2009/02 ... >
  <method name="GET" id="ItemSearch">
    <request>
      <param name="Service" style="query" fixed="AWSECommerceService"/>
      <param name="Version" style="query" fixed="2005-07-26"/>
      <param name="Operation" style="query" fixed="ItemSearch"/>
      <param name="SubscriptionId" style="query" type="xsd:string" required="true"/>
      <param name="SearchIndex" style="query" type="aws:SearchIndexType" required="true">
        <option value="Books"/>
        <option value="DVD"/>
        <option value="Music"/>
      </param>
      <param name="Keywords" style="query" type="aws:KeywordList" required="true"/>
      <param name="ResponseGroup" style="query" type="aws:ResponseGroupType" repeating="true">
        <option value="Small"/>
        <option value="Medium"/>
        <option value="Large"/>
      </param>
    </request>
    <response>
      <representation mediaType="text/xml" element="aws:ItemSearchResponse"/>
    </response>
  </method>
</application>
```

Figure B.3 – The service model of Search_item in WADL notation (*fragment*)

The registration information in terms of MFI Service registration is shown as following.
| <Service_Description_Language> Object201 | <input_Message> Object207 |
| Attribute/Reference | Literal/Instance | Attribute/Reference | Literal/Instance |
| name | "WADL"* | name | "Keywords"* |
| expressed_model | Object202 | involved_input_message_type | Object211 |
| containing_service_operation | Object204 | containing_service_operation | Object204 |

| <Service_Model> Object202 | <input_Message> Object208 |
| Attribute/Reference | Literal/Instance | Attribute/Reference | Literal/Instance |
| name | "Search Item" | name | "ResponseGroup" |
| described_IRI | "http://webServices.amazon.com/ onca" | involved_input_message_type | Object212 |
| describing_language | Object201 | containing_service_operation | Object204 |
| contained_service | Object203 |

| <Service> Object203 | <Message_Type> Object209 |
| Attribute/Reference | Literal/Instance | Attribute/Reference | Literal/Instance |
| name | "Search Item" | name | "string" |
| requested_IRI | "http://webServices.amazon.com/onca" | involving_input_message | Object205 |
| domain | "shopping" | |
| service_type | "entity_service" | |
| containing_model | Object202 | |
| contained_service_operation | Object204 | |

| <Service_Operation> Object204 | <Message_Type> Object210 |
| Attribute/Reference | Literal/Instance | Attribute/Reference | Literal/Instance |
| name | "ItemSearch" | name | "SearchIndexType" |
| containing_service | Object203 | involving_input_message | Object206 |
| consumed_message | Object205, Object206, Object207, Object208 | |
| generated_message | Object213 | |

| <Input_Message> Object205 | <Message_Type> Object211 |
| Attribute/Reference | Literal/Instance | Attribute/Reference | Literal/Instance |
| name | "SubscriptionId" | name | "KeywordList" |
| involved_input_message_type | Object209 | involving_input_message | Object207 |
| containing_service_operation | Object204 | |

| <Input_Message> Object206 | <Message_Type> Object212 |
| Attribute/Reference | Literal/Instance | Attribute/Reference | Literal/Instance |
| name | "SearchIndex" | name | "ResponseGroupType" |
| involved_input_message_type | Object210 | involving_input_message | Object208 |
| containing_service_operation | Object204 | |

| <Output_Message> Object213 | <Message_Type> Object214 |
| Attribute/Reference | Literal/Instance | Attribute/Reference | Literal/Instance |
| name | "ItemSearchResponse" | name | "text/xml" |
| involved_output_message_type | Object214 | involving_output_message | Object213 |
| containing_service_operation | Object204 | |

---

Figure B.4 – Registration of the Search_Item example
Case3: Example of OWL-S Service Registration

This Service is described in Ontology Web Language for Service (OWL-S). This Service described here is a fictional book buying Service from www.congo.com. The normal usage scenario for these processes is as follows: they would be published by Congo, but executed by a software agent that wants to buy from Congo. The fragmentary code is shown in Figure B.5.

```xml
<profileHierarchy:BookSelling
    rdf:ID="Profile_Congo_BookBuying_Service">

    <!-- reference to the Service specification -->
    <Service:presentedBy rdf:resource="&congoService;#ExpressCongoBuyService"/>
    <profile:has_process rdf:resource="&congoProcess;#ExpressCongoBuy"/>

    <profile:ServiceName>Congo_BookBuying_Agent</profile:ServiceName>
    <profile:textDescription>
        This agentified Service provides the opportunity to browse a book selling site and buy books there
    </profile:textDescription>

    <profile:contactInformation>
        <actor:Actor rdf:ID="CongoBuy_contacts">
            <actor:name>ExpressCongoBuy</actor:name>
            <actor:title>Service Representative</actor:title>
            <actor:phone>412 268 8780</actor:phone>
            <actor:fax>412 268 5569</actor:fax>
            <actor:email>Bravo@Bravoair.com</actor:email>
            <actor:physicalAddress>
                somewhere 2,
                OnWeb,
                Montana 52321,
                USA
            </actor:physicalAddress>
            <actor:webURL>http://www.daml.org/Services/owl-s/1.1/ExpressCongoBuy.html</actor:webURL>
        </actor:Actor>
    </profile:contactInformation>

    <profileHierarchy:deliveryRegion rdf:resource="&country;#UnitedStates"/>
    <profile:hasInput rdf:resource="&congoProcess;#ExpressCongoBuyBookISBN"/>
    <profile:hasInput rdf:resource="&congoProcess;#ExpressCongoBuySignInInfo"/>
    <profile:hasInput rdf:resource="&congoProcess;#ExpressCongoBuyCreditCardNumber"/>
    <profile:hasInput rdf:resource="&congoProcess;#ExpressCongoBuyCreditCardType"/>
    <profile:hasInput rdf:resource="&congoProcess;#ExpressCongoBuyCreditCardExpirationDate"/>
    <profile:hasPrecondition rdf:resource="&congoProcess;#ExpressCongoBuyAcctExists"/>
    <profile:hasPrecondition rdf:resource="&congoProcess;#ExpressCongoBuyAcctExists"/>
    <profile:hasResult rdf:resource="&congoProcess;#ExpressCongoBuyPositiveResult"/>
    <profile:hasResult rdf:resource="&congoProcess;#ExpressCongoBuyNegativeResult"/>
    <profile:hasOutput rdf:resource="&congoProcess;#ExpressCongoBuyOutput"/>
</profileHierarchy:BookSelling>
```

Figure B.5 – The service model of Congo_BookBuying_Agent in OWL-S notation (fragment) (Part 1 of 4)
Figure B.5 – The service model of Congo_BookBuying_Agent in OWL-S notation (fragment) (Part 2 of 4)
This expression says that the ExpressCongoBuyOutput will be of type OrderShippedAcknowledgment.

This expression says that the selected book is shipped to the account

Figure B.5 – The service model of Congo_BookBuying_Agent in OWL-S notation (fragment) (Part 3 of 4)
...<process:hasResult>
  <process:Result rdf:id="ExpressCongoBuyNegativeResult">
    <rdfs:comment>
      If the book is out of stock, then the result is simply
      that an appropriate acknowledgment is output, indicating
      that the book is out of stock.
    </rdfs:comment>
    <process:inCondition>
      <expr:SWRL-Condition rdf:id="ExpressCongoBuyBookOutOfStock">
        <expr:expressionBody rdf:parseType="Literal">
          <swrl:AtomList>
            <rdf:first>
              <swrl:ClassAtom>
                <swrl:classPredicate>
                  <owl:Restriction>
                    <owl:onProperty rdf:resource="#hasBook"/>
                    <owl:allValueFrom rdf:resource="#OutOfStockBook"/>
                  </owl:Restriction>
                </swrl:classPredicate>
              </swrl:ClassAtom>
            </rdf:first>
            <rdf:rest rdf:resource="&rdf;#nil"/>
          </swrl:AtomList>
        </expr:expressionBody>
      </expr:SWRL-Condition>
    </process:inCondition>
    <process:hasEffect>
      <expr:SWRL-Expression>
        <rdfs:comment>
          This expression just says that the value of ExpressCongoBuyOutput will be
          NotifyBookOutOfStock.
        </rdfs:comment>
        <expr:expressionBody rdf:parseType="Literal">
          <swrl:AtomList>
            <rdf:first>
              <swrl:IndividualPropertyAtom>
                <swrl:propertyPredicate rdf:resource="#type"/>
              </swrl:IndividualPropertyAtom>
            </rdf:first>
            <rdf:rest rdf:resource="&rdf;#nil"/>
          </swrl:AtomList>
        </expr:expressionBody>
      </process:hasEffect>
    </process:Result>
  </process:hasResult>
</swrl:IndividualPropertyAtom>
...<process:hasResult>
  <process:Result rdf:id="ExpressCongoBuyNegativeResult">
    <rdfs:comment>
      If the book is out of stock, then the result is simply
      that an appropriate acknowledgment is output, indicating
      that the book is out of stock.
    </rdfs:comment>
    <process:inCondition>
      <expr:SWRL-Condition rdf:id="ExpressCongoBuyBookOutOfStock">
        <expr:expressionBody rdf:parseType="Literal">
          <swrl:AtomList>
            <rdf:first>
              <swrl:ClassAtom>
                <swrl:classPredicate>
                  <owl:Restriction>
                    <owl:onProperty rdf:resource="#hasBook"/>
                    <owl:allValueFrom rdf:resource="#OutOfStockBook"/>
                  </owl:Restriction>
                </swrl:classPredicate>
              </swrl:ClassAtom>
            </rdf:first>
            <rdf:rest rdf:resource="&rdf;#nil"/>
          </swrl:AtomList>
        </expr:expressionBody>
      </expr:SWRL-Condition>
    </process:inCondition>
    <process:hasEffect>
      <expr:SWRL-Expression>
        <rdfs:comment>
          This expression just says that the value of ExpressCongoBuyOutput will be
          NotifyBookOutOfStock.
        </rdfs:comment>
        <expr:expressionBody rdf:parseType="Literal">
          <swrl:AtomList>
            <rdf:first>
              <swrl:IndividualPropertyAtom>
                <swrl:propertyPredicate rdf:resource="#type"/>
              </swrl:IndividualPropertyAtom>
            </rdf:first>
            <rdf:rest rdf:resource="&rdf;#nil"/>
          </swrl:AtomList>
        </expr:expressionBody>
      </process:hasEffect>
    </process:Result>
  </process:hasResult>
</swrl:IndividualPropertyAtom>
...<process:hasResult>
  <process:Result rdf:id="ExpressCongoBuyNegativeResult">
    <rdfs:comment>
      If the book is out of stock, then the result is simply
      that an appropriate acknowledgment is output, indicating
      that the book is out of stock.
    </rdfs:comment>
    <process:inCondition>
      <expr:SWRL-Condition rdf:id="ExpressCongoBuyBookOutOfStock">
        <expr:expressionBody rdf:parseType="Literal">
          <swrl:AtomList>
            <rdf:first>
              <swrl:ClassAtom>
                <swrl:classPredicate>
                  <owl:Restriction>
                    <owl:onProperty rdf:resource="#hasBook"/>
                    <owl:allValueFrom rdf:resource="#OutOfStockBook"/>
                  </owl:Restriction>
                </swrl:classPredicate>
              </swrl:ClassAtom>
            </rdf:first>
            <rdf:rest rdf:resource="&rdf;#nil"/>
          </swrl:AtomList>
        </expr:expressionBody>
      </expr:SWRL-Condition>
    </process:inCondition>
    <process:hasEffect>
      <expr:SWRL-Expression>
        <rdfs:comment>
          This expression just says that the value of ExpressCongoBuyOutput will be
          NotifyBookOutOfStock.
        </rdfs:comment>
        <expr:expressionBody rdf:parseType="Literal">
          <swrl:AtomList>
            <rdf:first>
              <swrl:IndividualPropertyAtom>
                <swrl:propertyPredicate rdf:resource="#type"/>
              </swrl:IndividualPropertyAtom>
            </rdf:first>
            <rdf:rest rdf:resource="&rdf;#nil"/>
          </swrl:AtomList>
        </expr:expressionBody>
      </process:hasEffect>
    </process:Result>
  </process:hasResult>
</swrl:IndividualPropertyAtom>
...<process:hasResult>
  <process:Result rdf:id="ExpressCongoBuyNegativeResult">
    <rdfs:comment>
      If the book is out of stock, then the result is simply
      that an appropriate acknowledgment is output, indicating
      that the book is out of stock.
    </rdfs:comment>
    <process:inCondition>
      <expr:SWRL-Condition rdf:id="ExpressCongoBuyBookOutOfStock">
        <expr:expressionBody rdf:parseType="Literal">
          <swrl:AtomList>
            <rdf:first>
              <swrl:ClassAtom>
                <swrl:classPredicate>
                  <owl:Restriction>
                    <owl:onProperty rdf:resource="#hasBook"/>
                    <owl:allValueFrom rdf:resource="#OutOfStockBook"/>
                  </owl:Restriction>
                </swrl:classPredicate>
              </swrl:ClassAtom>
            </rdf:first>
            <rdf:rest rdf:resource="&rdf;#nil"/>
          </swrl:AtomList>
        </expr:expressionBody>
      </expr:SWRL-Condition>
    </process:inCondition>
    <process:hasEffect>
      <expr:SWRL-Expression>
        <rdfs:comment>
          This expression just says that the value of ExpressCongoBuyOutput will be
          NotifyBookOutOfStock.
        </rdfs:comment>
        <expr:expressionBody rdf:parseType="Literal">
          <swrl:AtomList>
            <rdf:first>
              <swrl:IndividualPropertyAtom>
                <swrl:propertyPredicate rdf:resource="#type"/>
              </swrl:IndividualPropertyAtom>
            </rdf:first>
            <rdf:rest rdf:resource="&rdf;#nil"/>
          </swrl:AtomList>
        </expr:expressionBody>
      </process:hasEffect>
    </process:Result>
  </process:hasResult>
</swrl:IndividualPropertyAtom>
...<process:hasResult>
  <process:Result rdf:id="ExpressCongoBuyNegativeResult">
    <rdfs:comment>
      If the book is out of stock, then the result is simply
      that an appropriate acknowledgment is output, indicating
      that the book is out of stock.
    </rdfs:comment>
    <process:inCondition>
      <expr:SWRL-Condition rdf:id="ExpressCongoBuyBookOutOfStock">
        <expr:expressionBody rdf:parseType="Literal">
          <swrl:AtomList>
            <rdf:first>
              <swrl:ClassAtom>
                <swrl:classPredicate>
                  <owl:Restriction>
                    <owl:onProperty rdf:resource="#hasBook"/>
                    <owl:allValueFrom rdf:resource="#OutOfStockBook"/>
                  </owl:Restriction>
                </swrl:classPredicate>
              </swrl:ClassAtom>
            </rdf:first>
            <rdf:rest rdf:resource="&rdf;#nil"/>
          </swrl:AtomList>
        </expr:expressionBody>
      </expr:SWRL-Condition>
    </process:inCondition>
    <process:hasEffect>
      <expr:SWRL-Expression>
        <rdfs:comment>
          This expression just says that the value of ExpressCongoBuyOutput will be
          NotifyBookOutOfStock.
        </rdfs:comment>
        <expr:expressionBody rdf:parseType="Literal">
          <swrl:AtomList>
            <rdf:first>
              <swrl:IndividualPropertyAtom>
                <swrl:propertyPredicate rdf:resource="#type"/>
              </swrl:IndividualPropertyAtom>
            </rdf:first>
            <rdf:rest rdf:resource="&rdf;#nil"/>
          </swrl:AtomList>
        </expr:expressionBody>
      </process:hasEffect>
    </process:Result>
  </process:hasResult>
</swrl:IndividualPropertyAtom>

Figure B.5 – The service model of Congo_BookBuying_Agent in OWL-S notation (fragment) (Part 4 of 4)

The following is shown how ‘A B2C book buying Service example of OWL-S’ is registered in terms of MFI Service registration.
Figure B.6 – Registration of the Congo_BookBuying_Agent example (Part 1 of 2)
<table>
<thead>
<tr>
<th>Attribute/Reference</th>
<th>Literal/Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>notation</td>
<td>&quot;SWRL&quot;</td>
</tr>
<tr>
<td>composed_expression</td>
<td>Object316</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute/Reference</th>
<th>Literal/Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>notation</td>
<td>&quot;SWRL&quot;</td>
</tr>
<tr>
<td>composed_expression</td>
<td>Object316</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute/Reference</th>
<th>Literal/Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>notation</td>
<td>&quot;SWRL&quot;</td>
</tr>
<tr>
<td>composed_expression</td>
<td>Object316</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute/Reference</th>
<th>Literal/Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>notation</td>
<td>&quot;SWRL&quot;</td>
</tr>
<tr>
<td>composing_expression</td>
<td>Object315, Object316</td>
</tr>
<tr>
<td>composition_type</td>
<td>&quot;disjunction&quot;</td>
</tr>
<tr>
<td>expressed_postcondition</td>
<td>Object310</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute/Reference</th>
<th>Literal/Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>&quot;ExpressCongoBuyBookISBN&quot;</td>
</tr>
<tr>
<td>containing_service_operation</td>
<td>Object304</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute/Reference</th>
<th>Literal/Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>&quot;ExpressCongoBuySignInInfo&quot;</td>
</tr>
<tr>
<td>containing_service_operation</td>
<td>Object304</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute/Reference</th>
<th>Literal/Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>&quot;ExpressCongoCreditCardNumber&quot;</td>
</tr>
<tr>
<td>containing_service_operation</td>
<td>Object304</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute/Reference</th>
<th>Literal/Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>&quot;ExpressCongoBuyCreditCardType&quot;</td>
</tr>
<tr>
<td>containing_service_operation</td>
<td>Object304</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute/Reference</th>
<th>Literal/Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>&quot;ExpressCongoBuyCreditCardExpirationDate&quot;</td>
</tr>
<tr>
<td>containing_service_operation</td>
<td>Object304</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute/Reference</th>
<th>Literal/Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>&quot;ExpressCongoBuyOutput&quot;</td>
</tr>
<tr>
<td>containing_service_operation</td>
<td>Object304</td>
</tr>
</tbody>
</table>

---

**Figure B.6 – Registration of the Congo_BookBuying_Agent example (Part 2 of 2)**
Case4: Example of WSDL Service Registration

This Service is described in Web Service Description Language (WSDL) Version 2.0. The example scenario is as following. Hotel GreatH (a fictional hotel) is located in a remote island. It has been relying on fax and phone to provide room reservations. Even though the facilities and prices at GreatH are better than what its competitor offers, GreatH notices that its competitor is getting more customers than GreatH. After research, GreatH realizes that this is because the competitor offers a Web Service that permits travel agent reservation systems to reserve rooms directly over the Internet. GreatH then hires somebody to build a reservation Web Service with the following functionality:

- **CheckAvailability.** To check availability, the client must specify a check-in date, a check-out date, and room type. The Web Service will return a room rate (a floating point number in USD) if such a room is available, or a zero room rate if not. If any input data is invalid, the Service should return an error. Thus, the Service will accept a `checkAvailability` message and return a `checkAvailabilityResponse` or `invalidDataFault` message.

- **MakeReservation.** To make a reservation, a client must provide a name, address, and credit card information, and the Service will return a confirmation number if the reservation is successful. The Service will return an error message if the credit card number or any other data field is invalid. Thus, the Service will accept a `makeReservation` message and return a `makeReservationResponse` or `invalidCreditCardFault` message.

To simplify the example, only the **CheckAvailability** operation will be implemented. The fragmentary code is shown in Figure B.7.
This document describes the GreatH Web Service. Additional application-level requirements for use of this Service -- beyond what WSDL 2.0 is able to describe -- are available at http://greath.example.com/2004/reservation-documentation.html

<types>
<xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://greath.example.com/2004/schemas/resSvc">

    <xs:element name="checkAvailability" type="tCheckAvailability"/>
    <xs:complexType name="tCheckAvailability">
        <xs:sequence>
            <xs:element name="checkInDate" type="xs:date"/>
            <xs:element name="checkOutDate" type="xs:date"/>
            <xs:element name="roomType" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
</types>

<interface name = "reservationInterface" >
    <fault name = "invalidDataFault"
            element = "ghns:invalidDataError"/>
</interface>

Figure B.7 – The service model of Hotel_Reservation in WSDL 2.0 notation (fragment) (Part 1 of 2)
Figure B.7 – The service model of Hotel_Reservation in WSDL 2.0 notation (fragment) (Part 2 of 2)

The following is shown how ‘WSDL 2.0 Document for the GreatH Web Service’ is registered in terms of MFI Service registration.
Figure B.8 – Registration of the Hotel_Reservation example
Bibliography

1. ProgrammableWeb Available at: http://www.programmableweb.com/
   Provides the categories that a Service belongs to.

2. Web Service Description Language (WSDL) Version 2.0 Part 0: Primer
   http://www.w3.org/TR/wsdl20-primer/

3. Web Service Modelling Language (WSML), W3C Member Submission 3 June 2005
   http://www.w3.org/Submission/WSML/

4. The Web Service Modelling Language WSML
   http://www.wsmo.org/wsml/wsml-syntax

5. Web Application Description Language W3C Member Submission 31 August 2009
   http://www.w3.org/Submission/wadl/

6. OWL-S: Semantic Markup for Web Services, W3C Member Submission 22 November 2004
   http://www.w3.org/Submission/OWL-S/


   http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/

9. SWRL: A Semantic Web Rule Language Combining OWL and RuleML, W3C Member Submission 21 May 2004
   http://www.w3.org/Submission/SWRL/

10. SWSL: Semantic Web Service Language. W3C Member Submission 9 September 2005
    http://www.w3.org/Submission/SWSF-SWSL/