INTERNATIONAL STANDARD

ISO/IEC 19763-7

First edition
2015-10-15

Information technology — Metamodel framework for interoperability (MFI) —

Part 7: Metamodel for service model registration

Technologies de l'information — Cadre du métamodèle pour l'interopérabilité (MFI) —

Partie 7: Métamodèle pour l'enregistrement du modèle de service

Reference number
ISO/IEC 19763-7:2015(E)

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

Contents

Page

Foreword v

Introduction vi

1 Scope 1

2 Normative references 1

3 Terms, definitions and abbreviated terms 2
3.1 Terms and definitions 2
3.2 Abbreviated terms 4

4 Conformance 5
4.1 General 5
4.2 Degree of conformance 5
4.2.1 General 5
4.2.2 Strictly conforming implementation 5
4.2.3 Conforming implementation 5
4.3 Implementation Conformance Statement (ICS) 5

5 MFI Service model registration 6
5.1 Overview of MFI Service model registration 6
5.2 Associations between MFI Service model registration and other parts in MFI 7
5.3 Structure of MFI Service model registration 8
5.3.1 Atomic_Expression 8
5.3.2 Composite_Expression 9
5.3.3 Exit_Condition 9
5.3.4 Expression 10
5.3.5 Input_Message 11
5.3.6 Message_Type 11
5.3.7 Output_Message 12
5.3.8 Postcondition 13
5.3.9 Precondition 13
5.3.10 QoS_Assertion 14
5.3.11 QoS_Type 14
5.3.12 Service 15
5.3.13 Service_Description_Language 16
5.3.14 Service_Model 16
5.3.15 Service_Operation 17
5.3.16 Service_Type 18
5.3.17 User_Tag 18

Annex A (informative) List of existing service description languages 19

Annex B (informative) Examples 20

Bibliography 35

Figures

Figure 1 — Scope of MFI Service model registration 1

Figure 2 —Metamodel of MFI service model registration 6

Figure 3 —The associations between MFI Service model registration and MFI Process model registration and MFI Role and Goal model registration 7

Figure 4 —The associations between MFI Service model registration and MFI Core and mapping 8

Figure B.1 –The service model of Hotel_Reservation in WSDL 2.0 notation (*fragment*) (Part 1 of 2) 21

Figure B.1 –The service model of Hotel_Reservation in WSDL 2.0 notation (*fragment*) (Part 2 of 2) 22

Figure B.2 –Registration of the Hotel_Reservation example 23

Figure B.3 –The service model of Book_Ticket in WSML notation (fragment) 24

Figure B.4 –Registration of the Book_Ticket example (Part 1 of 2) 25

Figure B.4 –Registration of the Book_Ticket example (Part 2 of 2) 26

Figure B.5 –The service model of Search_item in WADL notation (*fragment*) 27

Figure B.6 –Registration of the Search_Item example 28

Figure B.7 –The service model of Congo_BookBuying_Agent in OWL-S notation (*fragment*) (Part 1 of 4) 29

Figure B.7 –The service model of Congo_BookBuying_Agent in OWL-S notation (*fragment*) (Part 2 of 4) 30

Figure B.7 – The service model of Congo_BookBuying_Agent in OWL-S notation (fragment) (Part 3 of 4) 31

Figure B.7 – The service model of Congo_BookBuying_Agent in OWL-S notation (fragment) (Part 4 of 4) 32

Figure B.8 –Registration of the Congo_BookBuying_Agent example (Part 1 of 2) 33

Figure B.8 –Registration of the Congo_BookBuying_Agent example (Part 2 of 2) 34

Tables

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.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

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. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT), see the following URL: Foreword — Supplementary information.

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 model registration*

— *Part 8: Metamodel for role and goal model registration*

— *Part 9: On demand model selection [Technical Report]*

— *Part 10: Core model and basic mapping*

— *Part 12: Metamodel for information model registration*

— *Part 13: Metamodel for form design 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 web service discovery should be processed by machines. It is, therefore, necessary to semantically describe service information, including functional and non-functional information, and provide corresponding registration and management mechanism.

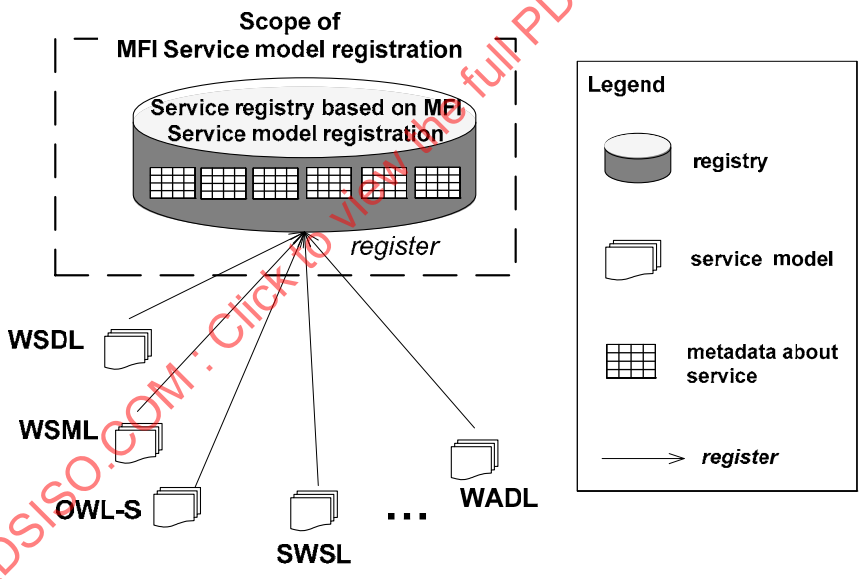
This part of ISO/IEC 19763 provides a framework for registering generic functional and non-functional information about web services.

Information technology — Metamodel framework for interoperability (MFI) — Part 7: Metamodel for service model 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 models of services, facilitating interoperability through the reuse of services.

This part of ISO/IEC 19763 is only applicable to web services whose capabilities are described by some web service description language (see Annex A for examples of such languages). 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 model registration

2 Normative references

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

NOTE One or more terms and definitions of the referenced International Standards listed below are used in Clause 3 Terms and Definitions.

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

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

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 purposes of this document, the following terms and definitions apply.

3.1.1
atomic expression
logical **expression** (3.1.5) that is not decomposed

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

3.1.3
entity service
web service (3.1.22) 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 unsuccessfully

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
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 a **service operation** (3.1.18) consumes for its execution

3.1.8
involvement type
statement that indicates the type of involvement of a **role**(3.1.16) with a **process** (3.1.13) or **service** (3.1.17)

NOTE Examples are performer, beneficiary, customer

[ISO/IEC 19763-8, 3.1.4]

3.1.9
message type
classification of the message or a set of messages that is/are consumed or generated during execution of a **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

[ISO/IEC 14813-5:2010 B.1.116]

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

[ISO/IEC 14813-5:2010 B.1.117]

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

[ISO/IEC 19763-5, 3.1.12]

3.1.14
QoS assertion
specification of one or more **QoS type**s (3.1.15) 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
named specific behaviour of an entity participating in a particular context

[ISO/IEC 19763-8, 3.1.7]

3.1.17
service
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 (3.1.22) 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 (3.1.22) that is dedicated to provide reusable, cross-cutting utility functionality, such as event logging, notification, and except handling

3.1.22
web service
service (3.1.17) that is designed to support interoperable machine-to-machine interaction over a network

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

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

WSDL
Web Service Description Language

WSML
Web Service Modelling Language

4 Conformance

4.1 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

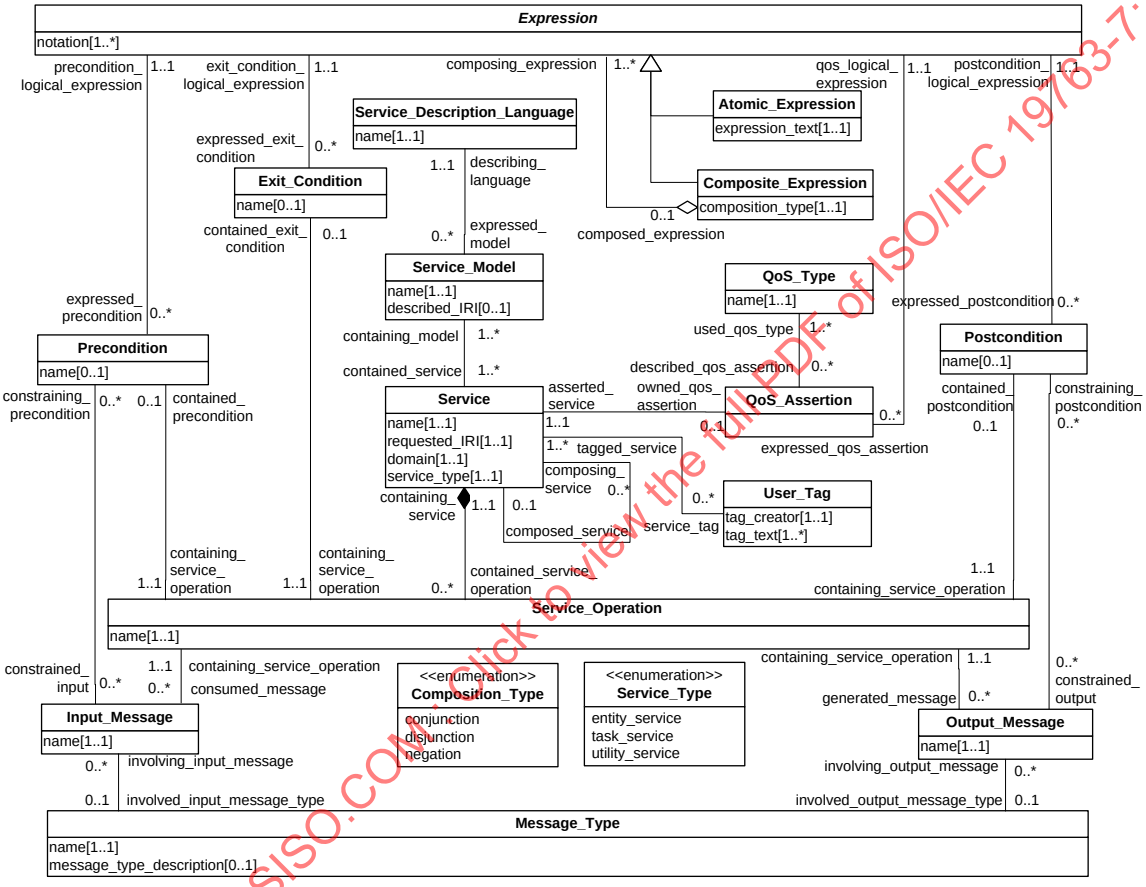
a) whether it is a strictly conforming implementation in clause 4.2.2 or a conforming implementation in clause 4.2.3;

b) what extensions, if any, are supported or used if it is a conforming implementation.

5 MFI Service model registration

5.1 Overview of MFI Service model registration

This part of MFI specifies the metamodel that can be used to register functional and non-functional information about web services (hereafter referred to simply as “services”). Examples of some service description languages that can be registered using this metamodel are listed in Annex A.



NOTE Metaclasses whose names are italicized are abstract metaclasses

Figure 2 —Metamodel of MFI service model registration

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 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 or more QoS types.

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 precondition, with zero or one post condition and with zero or one exit condition. A precondition 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. 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 provides a description of a message or a set of messages, where each of these messages is consumed or generated during execution of a service operation. Each input message is constrained by zero, one or more preconditions 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 any person using the service.

5.2 Associations between MFI Service model registration and other parts in MFI

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

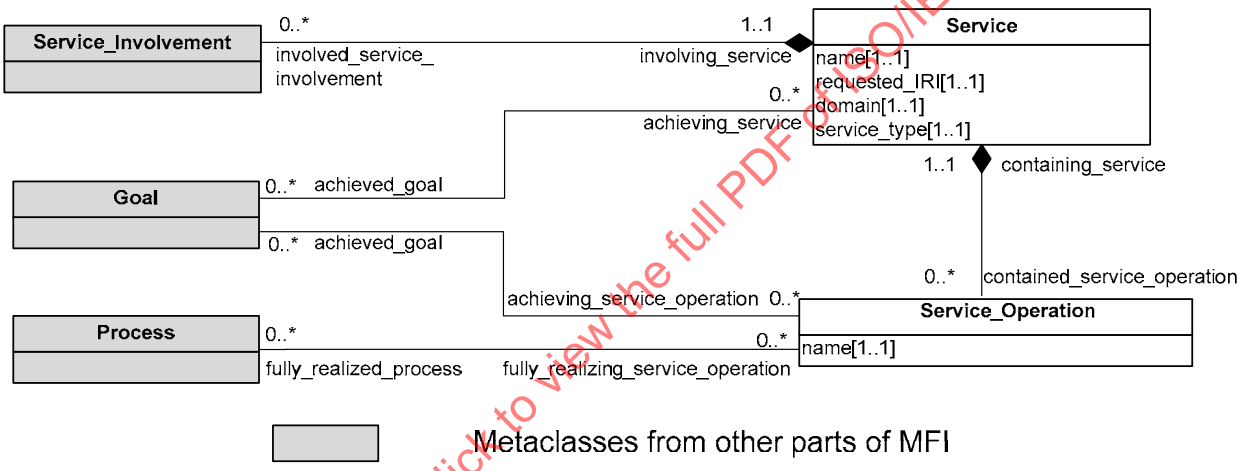


Figure 3 —The associations between MFI Service model registration and MFI Process model registration and MFI Role and Goal model registration

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 model registration and the metaclasses in MFI Core and mapping is shown in Figure 4.

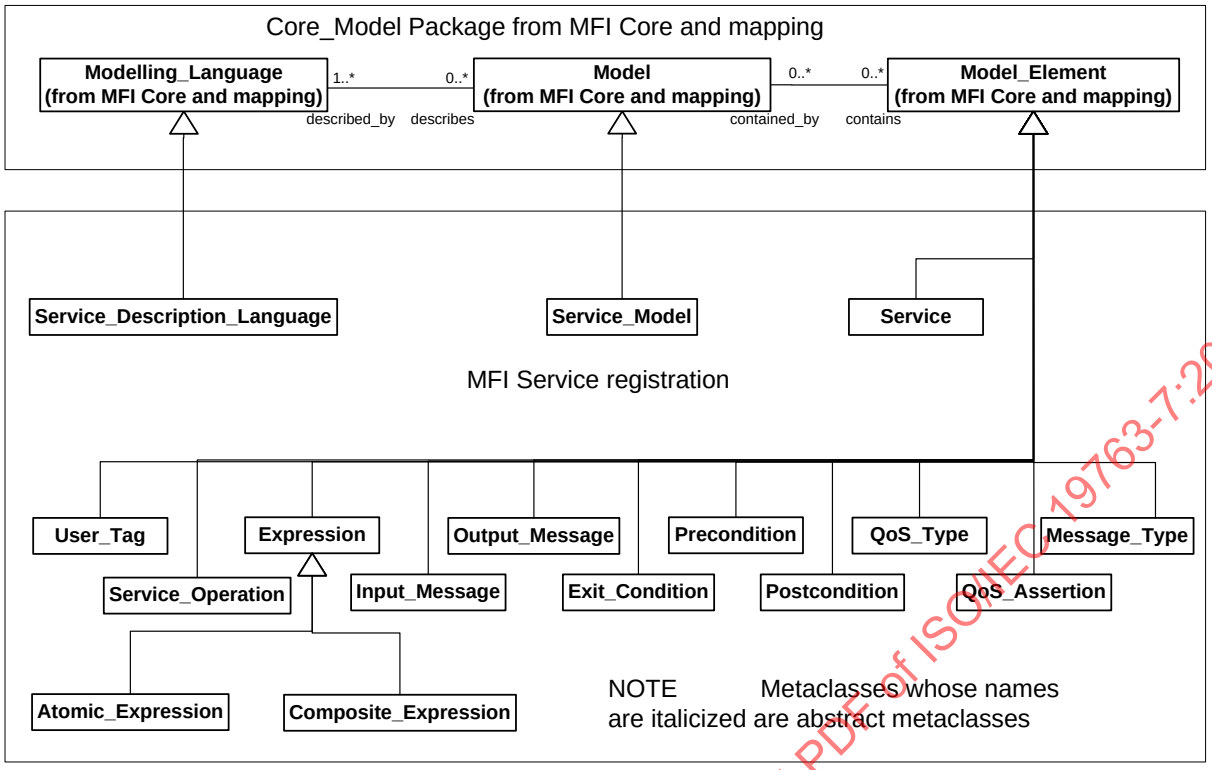


Figure 4 —The associations between MFI Service model registration and MFI Core and mapping

Service_Description_Language in MFI Service model registration is a subclass of Modelling_Language in MFI Core and mapping. Service_Model in MFI Service model registration is a subclass of Model in MFI Core and mapping. All the remaining metaclasses are subclasses of Model_Element in MFI Core and mapping.

All subclasses inherit the associations from their superclass in MFI Core and mapping. Some of these inherited associations are further specialized in this part. The details of these specializations are defined in Clause 5.3.

5.3 Structure of MFI Service model registration

5.3.1 Atomic_Expression

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

Superclass

Expression

Attribute	**Datatype**	**Multiplicity**	**Description**
expression_text	string	1..1	The text, using a logical notation, of this atomic expression.

Reference	**Class**	**Multiplicity**	**Description**	**Inverse**	**Precedence**

[None]

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

Attribute	Datatype	Multiplicity	Description
composition_ type	Composition_Type	1..1	The symbol which is used to connect two or more logical expressions.

Reference	Class	Multiplicity	Description	Inverse	Precedence
composing_ expression	Expression	1..*	The set of expressions that comprise this composite expression.	composed_ expression	Yes

Constraints

[None]

5.3.3 Composition_Type

Composition_Type is an enumerated datatype with the following values:

Value	Description
conjunction	An indication that the composite expression is true only in the case when all of the expressions that comprise this composite expression are true.
disjunction	An indication that the composite expression is false only in the case when all of the expressions that comprise this composite expression are false.
negation	An indication that whether the composite expression is true or false is the opposite of that of the single expression that comprises this composite expression.

5.3.4 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)

Attribute	DataType	Multiplicity	Description
name	string	0..1	The name of this exit condition.

Reference	Class	Multiplicity	Description	Inverse	Precedence
exit_condition_ logical_	Expression	1..1	The sentence which is described by a kind of logic	expressed_ exit_	Yes

expression			notation to express this exit condition.	condition	
containing_ service_ operation	Service_Operation	1..1	The service operation that may be finished unsuccessfully if this exit condition is true.	contained_ exit_ condition	No

Constraints

[None]

5.3.5 Expression

Expression is an abstract metaclass each instance of 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)

Attribute	DataType	Multiplicity	Description
notation	string	1..*	The set of logic languages or notations, each of which is used to declare a quality assertion, a precondition, a postcondition or an exit condition of a service.

Reference	Class	Multiplicity	Description	Inverse	Precedence
expressed_ precondition	Precondition	0..*	The constraint that must be true when an operation is invoked.	precondition_ logical_ expression	No
expressed_ postcondition	Postcondition	0..*	The constraint that must be true at the completion of an operation.	postcondition_ logical_ expression	No
expressed_ exit_condition	Exit_Condition	0..*	The constraint that must be true to cause an operation to complete unsuccessfully.	exit_condition_ logical_ expression	No
expressed_ qos_assertion	QoS_Assertion	0..*	The specification of one or more QoS types for the service.	qos_logical_ expression	No
composed_ expression	Composite_ Expression	0..1	The composite expression of which this expression forms one of the elements of that composite expression.	composing_ expression	Yes

Constraints

[None]

5.3.6 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

Model_Element (from MFI Core and mapping)

Attribute	Datatype	Multiplicity	Description
name	string	1..1	The name of the message that is consumed for the execution of a service operation.

Reference	Class	Multiplicity	Description	Inverse	Precedence
containing_ service_ operation	Service_Operation	1..1	The service operation that consumes this input message.	consumed_ message	No
constraining_ precondition	Precondition	0..*	The set of preconditions, each of which constrains this input message when the service operation is invoked.	constrained_input	Yes
involved_ input_message_ type	Message_Type	0..1	The type of this input message.	involving_ input_ message	Yes

Constraints

[None]

5.3.7 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)

Attribute	DataType	Multiplicity	Description
name	string	1..1	The name of this message type.
message_type_ description	string	0..1	The description of this message type.

Reference	Class	Multiplicity	Description	Inverse	Precedence
involving_input_ message	Input_Message	0..*	The set of input messages, each of which is described by this message type.	involved_input_ message_type	No

involving_output_message	Output_Message	0..*	The set of output messages, each of which is described by this message type.	involved_output_message_type	No

Constraints

[None]

5.3.8 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)

Attribute	DataType	Multiplicity	Description
name	string	1..1	The name of the message that is generated after the execution of a service operation.

Reference	Class	Multiplicity	Description	Inverse	Precedence
containing_service_operation	Service_Operation	1..1	The service operation that generates this output message.	generated_message	No
constraining_postcondition	Postcondition	0..*	The set of postconditions, each of which constrains this output message when the service operation is invoked.	constrained_output	Yes
involved_output_message_type	Message_Type	0..1	The type of this output message.	involving_output_message	Yes

Constraints

[None]

5.3.9 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)

Attribute	Datatype	Multiplicity	Description
name	string	0..1	The name for this postcondition.

Reference	Class	Multiplicity	Description	Inverse	Precedence
postcondition_ logcal_ expression	Expression	1..1	The expression that expresses this postcondition as a logical sentence.	expressed_ postcondition	Yes
constrained_ output	Output_Message	0..*	The set of output messages, each of which is constrained by this postcondition.	constraining_ postcondition	No
containing_ service_ operation	Service_Operation	1..1	The execution action of the service which contains this postcondition.	contained_ postcondition	No

Constraints

[None]

5.3.10 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)

Attribute	Datatype	Multiplicity	Description
name	string	0..1	The name for this precondition.

Reference	Class	Multiplicity	Description	Inverse	Precedence
precondition_ logical_ expression	Expression	1..1	The expression that expresses this precondition as a logical sentence.	expressed_ precondition	Yes
constrained_ input	Input_ Message	0..*	The set of input messages, each of which is constrained by this precondition.	constraining_ precondition	No
containing_ service_ operation	Service_ Operation	1..1	The execution action of a service which contains this precondition.	contained_ precondition	No

Constraints

[None]

5.3.11 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)

Attribute	DataType	Multiplicity	Description
[None]			

Reference	Class	Multiplicity	Description	Inverse	Precedence
used_qos_type	QoS_Type	1..*	The set of QoS types, each of which is involved in this QoS assertion.	described_ qos_assertion	Yes
qos_logical_ expression	Expression	1..1	The expression that expresses this QoS assertion as a logical sentence.	expressed_ qos_assertion	Yes
asserted_ service	Service	1..1	The service whose quality is asserted by this QoS assertion.	owned_qos_ assertion	No

Constraints

[None]

5.3.12 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)

Attribute	DataType	Multiplicity	Description
name	string	1..1	The name of this type of non-functional property.

Reference	Class	Multiplicity	Description	Inverse	Precedence
described_ qos_assertion	QoS_ Assertion	0..*	The set of QoS assertions, each of which is described by this QoS assertion type.	used_qos_ type	No

Constraints

[None]

5.3.13 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)

Attribute	DataType	Multiplicity	Description
name	string	1..1	The name of this service.
requested_IRI	string	1..1	The IRI for invoking this service.
domain	string	1..1	The domain that this service belongs to (see bibliography [1]).
service_type	Service_Type	1..1	The type of this service.

Reference	Class	Multiplicity	Description	Inverse	Precedence
owned_qos_ assertion	QoS_ Assertion	0..1	The QoS assertion that applies to this service.	asserted_ service	Yes
contained_ service_operation	Service_ Operation	0..*	The set of service operations, each of which denotes an execution action of this service.	containing_ service	Yes
service_tag	User_Tag	0..*	The set of tags, each of which is annotated by an individual or an organization in order to describe this service.	tagged_ service	Yes
composing_ service	Service	0..*	The set of services, each of which is a component of this service.	composed_ service	Yes
composed_ service	Service	0..1	The composite service which contains this service.	composing_ service	No
containing_model	Service_Model	1..*	The set of service models, each of which is used to model this service. This reference specializes the ‘contained_by’ reference which is inherited from the superclass.	contained_ service	No
achieved_goal	Goal (from MFI Role and Goal model registration)	0..*	The set of goals, each of which is achieved by this service.	achieving_ service	Yes
involved_ service_ involvement	Service_ Involvement (from MFI Role and Goal model registration)	0..*	The set of service involvements, each of which represents a statement that specifies how a particular role is involved in this service.	involving_ service	Yes

Constraints

The value of attribute “requested_IRI” has to be unique in this metaclass.

5.3.14 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)

Attribute	DataType	Multiplicity	Description
name	string	1..1	The name of this service description language.

Reference	Class	Multiplicity	Description	Inverse	Precedence
expressed_model	Service_Model	0..*	The set of service models that are described by this service description language. This reference specializes the 'describes' reference which is inherited from the superclass.	describing_language	No

Constraints

[None]

5.3.15 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)

Attribute	DataType	Multiplicity	Description
name	string	1..1	The name of this service model.
described_IRI	string	0..1	The IRI that identifies the corresponding service model.

Reference	Class	Multiplicity	Description	Inverse	Precedence
describing_language	Service_Description_Language	1..1	The language or notation that is used to model this service. This reference specializes the 'described_by' reference which is inherited from the superclass.	expressed_model	Yes
contained_service	Service	1..*	The set of services, each of which is modelled by this service model. This reference specializes the 'contains' reference which is inherited from the superclass.	containing_model	No

Constraints

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

5.3.16 Service_Operation

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

Superclass

Model_Element (from MFI Core and mapping)

Attribute	DataType	Multiplicity	Description
name	string	1..1	The name of this service operation.

Reference	Class	Multiplicity	Description	Inverse	Precedence
consumed_ message	Input_ Message	0..*	The set of messages, each of which is consumed by this service operation for its execution.	containing_ service_ operation	Yes
generated_ message	Output_ Message	0..*	The set of messages, each of which is generated by this service operation after its execution.	containing_ service_ operation	Yes
contained_ precondition	Precondition	0..1	The constraint that must be true when this operation is invoked.	containing_ service_ operation	Yes
contained_ postcondition	Postcondition	0..1	The constraint that must be true at the completion of this operation.	containing_ service_ operation	Yes
contained_ exit_condition	Exit_ Condition	0..1	The constraint that must be true to cause an operation to complete unsuccessfully.	containing_ service_ operation	Yes
containing_ service	Service	1..1	The service that contains this service operation.	contained_ service_ operation	No
achieved_ goal	Goal(from MFI Role and Goal model registration)	0..*	The set of goals, each of which is achieved by this service operation.	achieving_ service_ operation	Yes
fully_realized_ process	Process(from MFI Process model registration)	0..*	The set of processes, each of which is fully realized by this service operation.	fully_ realizing_ service_ operation	Yes

Constraints

[None]

5.3.17 Service_Type

Service_Type is an enumerated datatype with the following values:

Value	Description
entity_service	Indicates that this service bases its functional boundary on one or more related organization entities, such as customer, employee, invoice and claim.
task_service	Indicates that this service has a functional boundary directly associated with a process model.
utility_service	Indicates that this service provides reusable and cross-cutting utility functionality, such as event logging, notification, and except handling.

5.3.18 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)

Attribute	DataType	Multiplicity	Description
tag_creator	string	1..1	The user who tags the service.
tag_text	string	1..*	The text that a user tags for the service.

Reference	Class	Multiplicity	Description	Inverse	Precedence
tagged_service	Service	1..*	The set of services, each of which is tagged by this user tag.	service_tag	No

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. These are shown in Table A.1.

Table A.1 - List of existing service description languages

Name	Description
OWL-S	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
WSDL	Web Services Description Language (WSDL) Version 2.0 W3C Recommendation 26 June 2007
WSML	Web Service Modelling Language, 2005-06-03, W3C Member Submission
SA-WSDL	Semantic Annotations for WSDL and XML Schema, 2007-08-28, W3C Recommendation
SWSL	Semantic Web Service Language, 2005-09-09, W3C Member Submission
WADL	Web Application Description Language, 2009-08-31, W3C Member Submission
SA-REST	SA-REST: Semantic Annotation of Web Resources, 2010-04-05, W3C Member Submission

Annex B
(informative)

Examples

B.1 Example 1 –WSDL Service Registration

This Service is described using Web Service Description Language (WSDL) Version 2.0. The example concerns Hotel GreatH (a fictional hotel) located on 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. As a result GreatH issues a contract 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.1.

```
<?xml version="1.0" encoding="utf-8" ?>
<description
   xmlns="http://www.w3.org/ns/wsdl"
   targetNamespace= "http://greath.example.com/2004/wsdl/resSvc"
   xmlns:tns= "http://greath.example.com/2004/wsdl/resSvc"
   xmlns:ghns = "http://greath.example.com/2004/schemas/resSvc"
   xmlns:wsoap= "http://www.w3.org/ns/wsdl/soap"
   xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
   xmlns:wsdlx= "http://www.w3.org/ns/wsdl-extensions">
 <documentation>
   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
 </documentation>
 <types>
   <xs:schema
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://greath.example.com/2004/schemas/resSvc"
      xmlns="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:element name="checkAvailabilityResponse" type="xs:double"/>
     <xs:element name="invalidDataError" type="xs:string"/>
   </xs:schema>
 </types>

 <interface  name = "reservationInterface" >

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

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

```
  <operation name="opCheckAvailability"
        pattern="http://www.w3.org/ns/wsdl/in-out"
        style="http://www.w3.org/ns/wsdl/style/iri"
        wsdlx:safe = "true">
    <input messageLabel="In"
        element="ghns:checkAvailability" />
    <output messageLabel="Out"
        element="ghns:checkAvailabilityResponse" />
    <outfault ref="tns:invalidDataFault" messageLabel="Out"/>
  </operation>

</interface>

<binding name="reservationSOAPBinding"
      interface="tns:reservationInterface"
      type="http://www.w3.org/ns/wsdl/soap"
      wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/">

  <fault ref="tns:invalidDataFault"
   wsoap:code="soap:Sender"/>

  <operation ref="tns:opCheckAvailability"
   wsoap:mep="http://www.w3.org/2003/05/soap/mep/soap-response"/>

</binding>

<Service name="reservationService"
    interface="tns:reservationInterface">

   <endpoint name="reservationEndpoint"
          binding="tns:reservationSOAPBinding"
          address ="http://greath.example.com/2004/reservation"/>

 </Service>
</description>
```

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

Figure B.2 provides the object instances to illustrate the registration of the ‘GreatH Web Service’.

<Service_Description_Language>
Object101

Attribute/Reference	Literal/Instance
name	"WSDL"
expressed_model	Object102

<Service_Model>
Object102

Attribute/Reference	Literal/Instance
name	"reservationService_Model"
described_IRI	"http://greath.example.com/2004/reservation"
describing_language	Object101
contained_service	Object103

<Service>
Object103

Attribute/Reference	Literal/Instance
name	"reservationService"
requested_IRI	"http://greath.example.com/2004/reservation?wsdl"
domain	"travel"
service_type	'entity_service'
containing_model	Object102
contained_operation	Object104

<Service_Operation>
Object104

Attribute/Reference	Literal/Instance
name	"opCheckAvailability"
containing_service	Object103
consumed_message	Object105
generated_message	Object106

<Input_Message>
Object105

Attribute/Reference	Literal/Instance
name	"In"
involved_input_message_type	Object107
containing_service_operation	Object104

<Output_Message>
Object106

Attribute/Reference	Literal/Instance
name	"Out"
involved_output_message_type	Object108
containing_service_operation	Object104

<Message_Type>
Object107

Attribute/Reference	Literal/Instance
name	"tCheckAvailability"
message_type_description	"<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>"
involving_input_message	Object105

<Message_Type>
Object108

Attribute/Reference	Literal/Instance
name	"double"
involving_output_message	Object106

Figure B.2 –Registration of the Hotel_Reservation example

B.2 Example 2 –Example of WSML Service Registration

The Service in this example is described using Web Service Modelling Language (WSML). 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 deducted from the credit card (effect). The fragmentary code is shown in Figure B.3.

```
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 ?intialBalance] 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.3 –The service model of Book_Ticket in WSML notation (fragment)

Figure B.4 provides the object instances to illustrate the registration of the ‘Book Ticket Web Service’.

<Service_Description_Language>
Object201

Attribute/Reference	Literal/Instance
name	"WSML"
expressed_model	Object202

<Service_Model>
Object202

Attribute/Reference	Literal/Instance
name	"bookTicketWebService_Model"
described_IRI	"http://example.org/bookTicketWebService"
describing_language	Object201
contained_service	Object203

<Service>
Object203

Attribute/Reference	Literal/Instance
name	"BookTicketWebService"
requested_IRI	"http://example.org/bookTicketWebService?wsdl"
domain	"Travel"
service_type	'task_service'
containing_model	Object202
contained_service_operation	Object204

<Service_Operation>
Object204

Attribute/Reference	Literal/Instance
name	"BookTicketCapability"
containing_service	Object203
consumed_message	Object218,Object219,Object220
generated_message	Object221,Object222
contained_precondition	Object205
contained_postcondition	Object212

<Precondition>
Object205

Attribute/Reference	Literal/Instance
containing_service_operation	Object204
precondition_logical_expression	Object211
constrained_input	Object219,Object220

<Atomic_Expression>
Object206

Attribute/Reference	Literal/Instance
notation	"WSML"
expression_text	"?trip memberOftr#tripFromAustria"
composed_expression	Object211

<Atomic_Expression>
Object207

Attribute/Reference	Literal/Instance
notation	"WSML"
expression_text	"po#validCreditCard(?creditCard)"
composed_expression	Object211

<Atomic_Expression>
Object208

Attribute/Reference	Literal/Instance
notation	"WSML"
expression_text	"?creditCard[type hasValue "PlasticBuy"]"
composed_expression	Object210

<Atomic_Expression>
Object209

Attribute/Reference	Literal/Instance
notation	"WSML"
expression_text	"?creditCard[type hasValue "GoldenCard"]"
composed_expression	Object210

<Composite_Expression>
Object210

Attribute/Reference	Literal/Instance
notation	"WSML"
composing_expression	Object208, Object209
composition_type	'disjunction'
composed_expression	Object211

<Composite_Expression>
Object211

Attribute/Reference	Literal/Instance
notation	"WSML"
composing_expression	Object206, Object207,Object210
composition_type	'conjunction'
expressed_precondition	Object205

<Postcondition>
Object212

Attribute/Reference	Literal/Instance
containing_service_operation	Object204
postcondition_logical_expression	Object217
constrained_output	Object221, Object222

<Atomic_Expression>
Object213

Attribute/Reference	Literal/Instance
notation	"WSML"
expression_text	"?ticket [trip hasValue ?trip] memberOf tr#ticket"
composed_expression	Object217

<Atomic_Expression>
Object214

Attribute/Reference	Literal/Instance
notation	"WSML"
expression_text	"ticketPrice(?ticket, "euro", ?ticketPrice)"
composed_expression	Object217

<Atomic_Expression>
Object215

Attribute/Reference	Literal/Instance
notation	"WSML"
expression_text	"?finalBalance=(?initialBalance - ?ticketPrice)"
composed_expression	Object217

<Atomic_Expression>
Object216

Attribute/Reference	Literal/Instance
notation	"WSML"
expression_text	"?creditCard[po#balance hasValue ?finalBalance]"
composed_expression	Object217

Figure B.4 –Registration of the Book_Ticket example (Part 1 of 2)

<Composite_Expression>
Object217

Attribute/Reference	Literal/Instance
notation	“WSML”
composing_expression	Object213, Object214, Object215, Object216
composition_type	‘conjunction’
expressed_postcondition	Object212

<Input_Message>
Object218

Attribute/Reference	Literal/Instance
name	“reservationRequest”
involved_input_message_type	Object223
containing_service_operation	Object204

<Input_Message>
Object219

Attribute/Reference	Literal/Instance
name	“trip”
containing_service_operation	Object204
constraining_precondition	Object205

<Input_Message>
Object220

Attribute/Reference	Literal/Instance
name	“creditCard”
involved_input_message_type	Object224
containing_service_operation	Object204
constraining_precondition	Object205

<Output_Message>
Object221

Attribute/Reference	Literal/Instance
name	“reservation”
involved_output_message_type	Object225
containing_service_operation	Object204
constraining_postcondition	Object212

<Output_Message>
Object222

Attribute/Reference	Literal/Instance
name	“ticket”
involved_output_message_type	Object226
containing_service_operation	Object204
constraining_postcondition	Object212

<Message_Type>
Object223

Attribute/Reference	Literal/Instance
name	“reservationRequest”
message_type_description	“[reservationItem hasValue ?trip, reservationHolder hasValue ?reservationHolder]”
involving_input_message	Object218

<Message_Type>
Object224

Attribute/Reference	Literal/Instance
name	"creditCard“
message_type_description	“[balance hasValue ?intialBalance]”
involving_input_message	Object220

<Message_Type>
Object225

Attribute/Reference	Literal/Instance
name	“reservation”
message_type_description	"[reservationItem hasValue ?ticket, reservationHolder hasValue ?reservationHolder]"
involving_output_message	Object221

<Message_Type>
Object226

Attribute/Reference	Literal/Instance
name	“ticket"
message_type_description	“[trip hasValue ?trip]"
involving_output_message	Object222

Figure B.4 –Registration of the Book_Ticket example (Part 2 of 2)

B.3 Example 3 –Example of WADL Service Registration

This service in this example is described using the Web Application Description Language (WADL). Presently, WADL only provides functional semantics about services. This service, provided by Amazon, is to search for items in Amazon's Internet shop. The browser will return information about an item after the user inputs keywords or the index of that item. The fragmentary code is shown in Figure B.5.

```
<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"/>
        <option value="Images"/>
       </param>
   </request>
   <response>
      <representation mediaType="text/xml" element="aws:ItemSearchResponse"/>
   </response>
 </method>
</application>
```

Figure B.5 –The service model of Search_item in WADL notation (*fragment*)

Figure B.6 provides the object instances to illustrate the registration of the 'Item Search Web Service'.