
ISO/IEC JTC 1/SC 32 N xxxx

Date: 2009-11-08

ISO/IEC CD2 19763-5

ISO/IEC JTC 1/SC 32/WG 2

Secretariat: ANSI

**Information technology—Metamodel framework for interoperability (MFI) –
Part 5: Metamodel for process model 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.

Document type: International Standard

Document subtype:

Document stage: (50) Committee Draft

Document language: E

Copyright notice

This ISO document is a Draft International Standard and is copyright-protected by ISO. Except as permitted under the applicable laws of the user's country, neither this ISO draft nor any extract from it may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, photocopying, recording or otherwise, without prior written permission being secured.

Requests for permission to reproduce should be addressed to either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office

Case postale 56 • CH-1211 Geneva 20

Tel. + 41 22 749 01 11

Fax + 41 22 749 09 47

E-mail copyright@iso.ch

Web www.iso.ch

Reproduction may be subject to royalty payments or a licensing agreement.

Violators may be prosecuted.

Contents

Foreword	v
Introduction	vi
1 Scope	1
2 Conformance	1
2.1 General	1
2.2 Degree of conformance	1
2.2.1 General	1
2.2.2 Strictly conforming implementation	2
2.2.3 Conforming implementation	2
2.3 Implementation Conformance Statement (ICS)	2
3 Normative references	2
4 Terms, definitions and abbreviated terms	3
4.1 Terms and definitions	3
4.2 Abbreviated terms	3
5 Structure of MFI Process	4
5.1 Overview of MFI Process	4
5.2 Relationship between MFI Process and other parts in MFI	5
5.3 MFI Process	6
5.3.1 Process	7
5.3.2 Process_Model	8
5.3.3 Process_Modeling_Language	8
5.3.4 Composite_Process	9
5.3.5 Atomic_Process	9
5.3.6 Event	10
5.3.7 Input	10
5.3.8 Output	11
5.3.9 Resource	12
5.3.10 Control_Construct	12
5.3.11 Precondition	13
5.3.12 Postcondition	13
Annex A (informative) Examples of MFI Process registration	14
Annex B (informative) Collaboration between MFI members	18
Annex C (informative) List of process modeling languages	19
Annex D (informative) Code type lists	20

Figures and tables

Figure 1 – The scope of MFI process 1

Figure 2 – The metamodel for process model registration 5

Figure 3 – Relationship between MFI Process and other parts in MFI 6

Figure A.1 – Registration information of BravoAir Reservation Service 15

Figure A.2 – Registration Information of top level process for manufacturing GT350 **Error! Bookmark not defined.**

Figure A.3 – Registration information of some sub-processes of manufacturing GT350 17

Figure B.1 – Semantic interoperation based on MFI Process and MFI Ontology registration 18

Table C.1 – List of Process_Modeling_Languages 19

Table D.1 – Code type list of state of a process 20

Table D.2 – Code type list of state of a resource 20

Table D.3 – Code type list of control construct 21

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 part of ISO/IECWD 19763 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 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*:

Part 1: Reference Model

Part 2: Core Model

Part 3: Metamodel for ontology registration

Part 4: Metamodel for model mapping

Part 5: Metamodel for process model registration

Part 6: Registration procedures

Part 7: Metamodel for service registration

Part 8: Metamodel for role and goal registration

Part 9: On Demand Model Selection (ODMS) [Technical Report]

Introduction

Across-organizational collaboration and integration are blooming to provide better service for discriminating users. Discovery and reuse of process models registered in different repositories becomes the key issue to promote interoperation between them.

Many industrial consortia have contributed to standardization of domain specific process models using various representation notations and description languages for different purpose. The great differences in the syntax and semantic of process models will hamper sharing of them. It is necessary to provide a generic metamodel to support registration of administrative information of process models, having no reference to details of languages it adopts and platform it executes.

This part of ISO/IEC 19763 intends to provide a metamodel to register administrative information of process models.

Information Technology–Metamodel Framework for Interoperability –Part 5: Metamodel for process 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 the metamodel that provides a facility to register administrative information of process models.

The metamodel specified in this part is intended to promote discovery and reuse of process models within/across process model repositories. It provides administrative information of process models which have been created with a specific process modeling language, including Process Specification Language(PSL), Business Process Execution Language(BPEL), Web Ontology Language for Web Service(OWL-S), etc. Figure 1 shows the scope of this part.

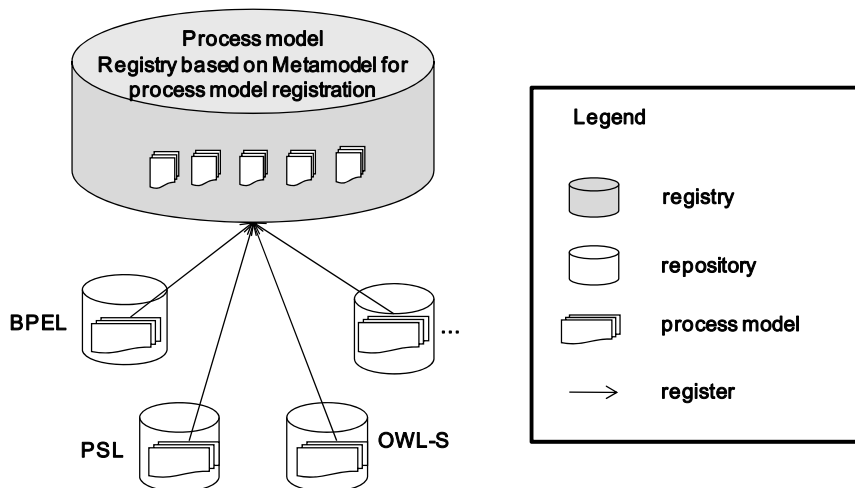


Figure 1 – The scope of MFI process

The followings are outside the scope of this part of ISO/IEC 19763:

- details related to modeling notations or descriptive languages of process models;
- runtime environment or implementation platforms for executing processes.

2 Conformance

2.1 General

An implementation claiming conformance with this part of ISO/IEC 19763 shall support the metamodel specified in 5.3, depending on a degree of conformance as described below.

2.2 Degree of conformance

2.2.1 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.

2.2.2 Strictly conforming implementation

A strictly conforming implementation

- a) shall support the metamodel specified in 5.3;
- b) shall not support any extensions to the metamodel specified in 5.3.

2.2.3 Conforming implementation

A conforming implementation

- a) shall support the metamodel specified in 5.3;
- b) may support extensions to the metamodel specified in 5.3 that are consistent with the metamodel specified in 5.3.

2.3 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 or a conforming implementation (2.2);
- b) what extensions are supported if it is a conforming implementation.

3 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 19763-1:2007, Information technology – Metamodel framework for interoperability (MFI) – Part 1: Reference model

ISO/IEC 19763-2, Information technology – Metamodel framework for interoperability (MFI) – Part 2: Core model

ISO/IEC 19763-3:2007, Information technology – Metamodel framework for interoperability (MFI) – Part 3: Metamodel for ontology registration

ISO/IEC 11179-3:2003, Information technology – Metadata registries (MDR) – Part 3: Registry metamodel and basic attributes

ISO/IEC 19501:2005, Information technology - Open Distributed Processing - Unified Modeling Language (UML) Version 1.4.2

ISO 18629-1:2004, Industrial automation systems and integration -- Process specification language -- Part 1: Overview and basic principles

4 Terms, definitions and abbreviated terms

4.1 Terms and definitions

The definitions provided in ISO/IEC 19763-1:2007, ISO/IEC 19763-2, ISO/IEC 19763-3:2007, ISO/IEC 11179-3:2003 and ISO/IEC 19501:2005 shall apply to this part of ISO/IEC 19763.

4.1.1

Process

a set of activities and resources, organized according to constraints, which all participate in fulfilling a given purpose.

4.1.2

Process model

a specification that is the result of modeling one or more processes, adopting a specific process modeling language to describe features of a process. It shows what the process does and how it is done.

4.1.3

Sub-process

a process that is contained in another process.

NOTE a sub-process may be an atomic process, or a composite process.

4.1.4

Atomic process

a process that does not have a sub-process.

4.1.5

Composite process

a process that consists of other processes and only one type of control construct.

4.1.6

Precondition

a kind of condition that must always be true just prior to the execution of a process in a formal specification.

4.1.7

Postcondition

a kind of condition that must always be true just after the execution of a process in a formal specification.

4.1.8

Event

<UML> a notable occurrence at a particular point in time.

4.1.9

Resource

anything participating in a process to help its performance.

NOTE Resource can be either physical or virtual things.

4.2 Abbreviated terms

MDR

Metadata Registry

[ISO/IEC 11179-3:2003, 3.4.5]

MFI

Metamodel framework for interoperability

[ISO/IEC 19763-1:2007, 4.2]

MFI Ontology registration

ISO/IEC 19763-3:2007, Information technology – Metamodel framework for interoperability (MFI) – Part 3: Metamodel for ontology registration

[ISO/IEC 19763-3:2007, 4.2]

MFI Process

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

MFI Core

ISO/IEC 19763-2, Information technology –Metamodel framework for interoperability(MFI) – Part 2: Core model

MFI Goal&Role

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

MFI Service

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

PSL

Process Specification Language (see ISO 18629-1:2004)

UML

Unified Modeling Language (see ISO/IEC 19501:2005)

5 Structure of MFI Process

5.1 Overview of MFI Process

MFI Process provides a generic facility to register administrative information of process models described by specific modeling languages. Figure 2 shows the metamodel for process model registration.

Process_Model is a specification that is the result of modeling **Process**, describing what the Process does and how it is done. **Process_Modeling_Language** specifies the modeling language that **Process_Model** uses to represent processes. **Process** can be triggered by **Event**, which is expressed as a notable occurrence of a process.

Atomic_Process and **Composite_Process** are two kinds of **Process**. **Atomic_Process** is a process that has no sub-process, while **Composie_Process** consists of other processes that can be either atomic processes or composite processes. In MFI Process, **Composite_Process** is defined to have only one kind of **Control_Construct** to connect its sub-processes.

A process can transform **Input** to **Output** to achieve the given purpose of **Process**. Either **Input** or **Output** may refer to **Resource**, which can be any virtual or physical thing participating in a process to help its performance. Particularly, the same instance of **Resource** can be referred to by **Input** or **Output** of any processes in the registry. Moreover,

binding constraints between **Input/Output** of a process and its sub-processes are recorded by different roles that **Input/Output** plays in different processes and the relation from **Output** to **Input** as well.

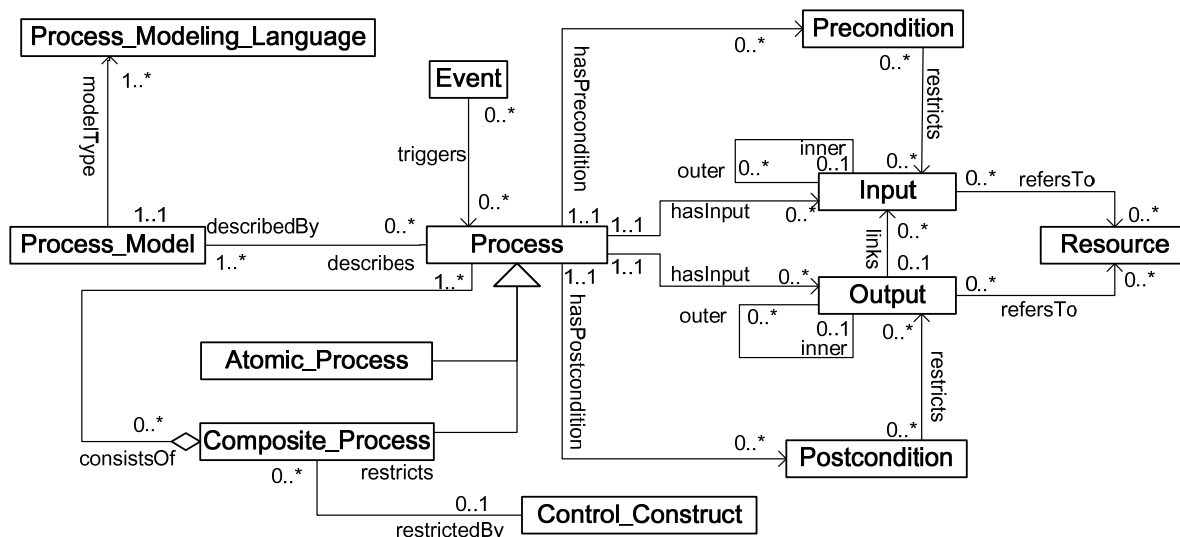


Figure 2 – The metamodel for process model registration

Precondition defines a kind of condition that must always be true just prior to the execution of a process in a formal specification, while **Postcondition** specifies the condition that must always be true just after the execution of a process in a formal specification. They can be used to restrict **Input** and **Output** respectively.

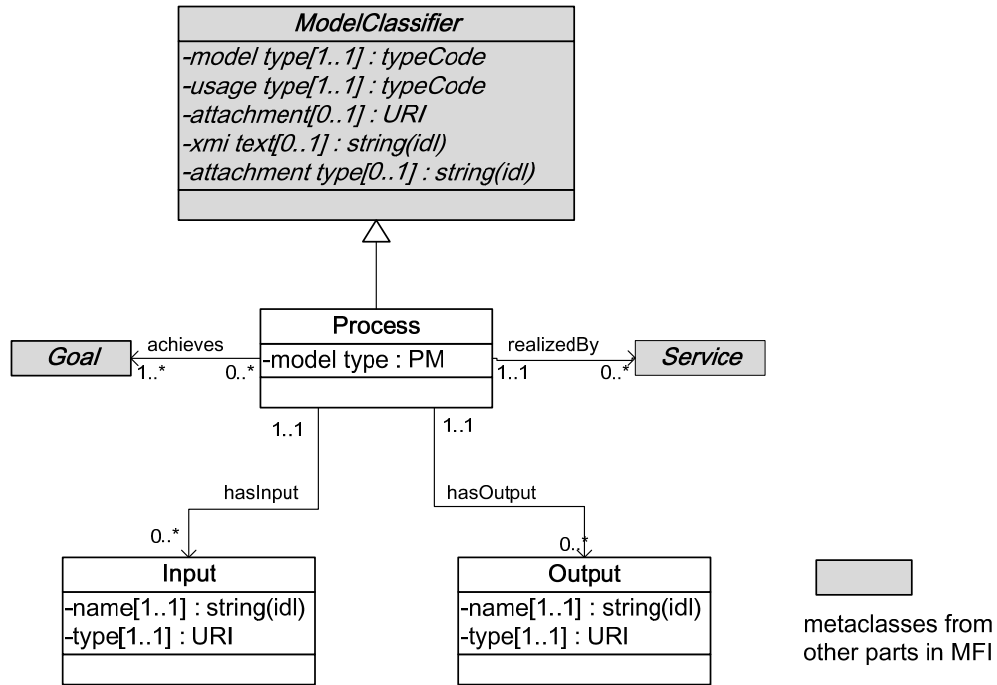
5.2 Relationship between MFI Process and other parts in MFI

Figure 2 shows the relationship between MFI Process and other parts in MFI.

Process model is one kind of registered target in MFI Core. So **Process** inherits **ModelClassifier** from MFI Core and declare the code of registered process model by fixing the value of model type as PM.

A process can achieve a set of goals, which are instances of **Goal** in MFI Role&Goal. The relationship between MFI Process and MFI Role&Goal means that a goal can be achieved by zero to many instances of **Process** and a process can achieve one to many instances of **Goal**. On the other hand, **Service** in MFI Service can be used to realize **Process**. The relationship between MFI Process and MFI Service means that a process can be realized by zero to many instances of **Service** and a service can achieve only one **Process**.

The attribute “type” of **Input** and **Output** can be declared as the URI of registered **Ontology_Atomic_Construct** based on MFI Ontology Registration, which means that ontology and its constructs can be used annotated inputs and outputs of a process.



NOTE PM is declared as the type code Of Process Model that can be registered as a kind of ModelClassifier. The instance of Process Model is a modeled process with a specific process modeling language.

Figure 3 – Relationship between MFI Process and other parts in MFI

5.3 MFI Process

The MFI Process is shown as a UML Class diagrams, with each Class being described as follows.

(1) Superclasses

immediate inherited classes

(2) Attributes

n. attribute name: datatype and multiplicity

-Use: Mandatory or Optional condition for attribute

-Description: description for content and purpose of attribute

(3) References

n. reference name: datatype and multiplicity

-Use: Mandatory or Optional condition for attribute

-Description: description for content and purpose of attribute

(4) Constraints

- reference name: Class name and multiplicity if exposed reference exists

- binding constraint: description about enforcement derived from a reference

-constraints specified if necessary, in natural language

5.3.1 Process

Process is an abstract metaclass representing the process described by a process model to be registered, which is the superclass of Atomic_Process and Composite_Process.

(1) Superclasses

ModelClassifier, Administered Item(from MDR)

(2) Attributes

1. **URI**: *String* [1..1]

-**Use**: Mandatory

-**Description**: URI where a process exists

2. **name**: *String* [1..1]

-**Use**: Mandatory

-**Description**: Name of a process.

3. **orderNumber**: *Integer* [1..1]

-**Use**: Optional

-**Description**: A number allocating to a process to identify a process

4. **type**: *String* [1..1]

-**Use**: Mandatory

-**Description**: The type of a registered process. "A" denotes Atomic_Process and "C" denotes Composite_Proces.

5. **stateType**: *typeCode* [1..1]

-**Use**: Optional

-**Description**: A type code specifying the current state of a process.

NOTE The code set of the state of a process should be defined by each MFI Process registry, seeing Table D.1 in Annex D.

(3) References

1. **containedIn**: *Composite_Process* [0..*]

-**Use**: Mandatory

-**Description**: The processes that are contained in a registered process.

2. **decribedBy**: *Process_Model* [1..*]

-**Use**: Mandatory

-**Description**: Process_Model specifying a specification of the registered process.

3. **triggeredBy**: *Event* [0..*]

-**Use**: Optional

-**Description**: Event that triggers execution of a process.

4. **hasInput**: *Input* [0..*]

-**Use**: Optional

-**Description**: Input that will be transferred by a process.

5. **hasOutput**: *Output* [0..*]

-**Use**: Optional

-**Description**: Output that is generated as the results after executing the process.

6. hasPrecondition: *Precondition [0..*]*

-Use: Optional

-Description: The condition that should be satisfied after executing a process.

7. hasPostcondition : *Postcondition [0..*]*

-Use: Optional

-Description: The condition that should be satisfied after executing a process.

(4) Constraints

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

5.3.2 Process_Model

Process_Model is a metaclass designating a specification that is the result of modeling a process.

(1) Superclasses

Administered Item(from MDR)

(2) Attributes

1. name: *String [1..1]*

-Use: Mandatory

-Description: Name of a process.

2. URI: *String [1..1]*

-Use: Mandatory

-Description: URI where a process exists.

(3) References

1. describes: *Process [0..*]*

-Use: Optional

-Description: The process that may be described by a process model.

2. modelType: *Process_Modeling_Language [1..*]*

-Use: Mandatory

-Description: The modeling language used to describe a process model.

(4) Constraints

None

5.3.3 Process_Modeling_Language

Process_Modeling_Language is a metaclass representing the modeling language of a process model.

(1) Superclasses

Administered Item(from MDR)

(2) Attributes

1. name: *String [1..1]*

-Use: Mandatory

-Description: Name of the process modeling language that is used to describe a process. Its value can be one of the values in column “name” of Table 1 in Annex C.

2. version: *String [1..1]*

-Use: Optional

-Description: A string identifies the version number about a process modeling language.

(3) References

1. **usedBy:** *Process_Model [1..1]*

-Use: Mandatory

-Description: The process model that adopts the modeling language.

(4) Constraints

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

5.3.4 Composite_Process

Composite_Process is a metaclass designating the process that contains other sub-processes, which might be either atomic process or composite process. And the contained sub-processes are connected by only one kind of control construct.

(1) Superclasses

Process

(2) Attributes

None

(3) References

1. **consistsOf:** *Process [1..*]*

-Use: Mandatory

-Description: The sub-processes contained in a process.

2. **restrictedBy:** *Control_Construct [0..1]*

-Use: Optional

-Description: The specified type of control construct that is used to connect processes contained in a composite process.

(4) Constraints

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

Exists at least one Process whose “containedIn” is this Composite_Process.

5.3.5 Atomic_Process

Atomic_Process is a metaclass designating the process that has no sub-process.

(1) Superclasses

Process

(2) Attributes

None

(3) References

None

(4) Constraints

None.

5.3.6 Event

Event is a metaclass designating a notable occurrence that triggers a process.

(1) Superclasses

Administered Item(from MDR)

(2) Attributes

1. **description:** *String [1..1]*

-**Use:** Mandatory

-**Description:** The description addressing the possible occurrence to be happened to a process.

2. **type:** *String [1..1]*

-**Use:** Mandatory

-**Description:** The type of an event. Its value could be "internal", "external" or "conditional".

(3) References

1. **triggers:** *Process [0..1]*

-**Use:** Optional

-**Description:** The process that is triggered by the event.

(4) Constraints

None

5.3.7 Input

Input is a metaclass specifying the resources that will be transferred by the process model.

(1) Superclasses

Administered Item(from MDR)

(2) Attributes

1. **name:** *String[1..1]*

-**Use:** Mandatory

-**Description:**Name of the input resource.

2. **type:** *URI[1..1]*

-**Use:** Mandatory

-**Description:**URI of the concept from a specified ontology.

(3) References

1. **restrictedBy:** *Precondition [0..*]*

-**Use:** Optional

-**Description:**The conditions attached to the input of a process. It should be satisfied before executing a process.

2. **refersTo:** *Resource [0..*]*

-**Use:** Optional

-**Description:**The resources participating in a process.

3. **inner:** *Input [0..*]*

-Use: Optional

-Description:The input of a composite process can be recorded as that of its sub processes.

4. outer: Input [0..1]

-Use: Optional

-Description:The input of a sub-process can be recorded as that of its parent processes.

(4) Constraints

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

5.3.8 Output

Output is a metaclass specifying the resources that are generated by a process.

(1) Superclasses

Administered Item(from MDR)

(2) Attributes

1. name: String[1..1]

-Use: Mandatory

-Description:Name of the output resource.

2. type: URI[1..1]

-Use: Mandatory

-Description:URI of the concept from a specified ontology.

(3) References

1. restrictedBy: Postcondition [0..*]

-Use: Optional

-Description:The conditions attached to the output of a process. It should be satisfied after executing a process.

2. refersTo: Resource [0..*]

-Use: Optional

-Description:The resources participating in a process.

3. inner: Output [0..*]

-Use: Optional

-Description:The output of a composite process can be recorded as that of its sub-process.

4. outer: Output [0..1]

-Use: Optional

-Description: The output of a sub-process can be recorded as that of its parent processes.

5. links: Input [0..*]

-Use: Optional

-Description:The output of a process can be the input of other processes.

(4) Constraints

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

5.3.9 Resource

Resource is a metaclass designating the resources participating in a process.

(1) Superclasses

Administered Item(from MDR)

(2) Attributes

1. **name:** *String* [1..1]

-**Use:** Mandatory

-**Description:** Name of a thing that participates in performing a process.

2. **URI:** *String* [1..1]

-**Use:** Mandatory

-**Description:**URI where a resource exists.

3. **stateType:** *stateTypeCodeOfResource* [1..1]

-**Use:** Optional

-**Description:** A type code specifying the state type of resources participating in the process.

NOTE The code set of state of Resource should be defined by each MFI Process registry, seeing Table D.2 in Annex D.

(3) References

1. **referredTo:** *Input* [0..*]

-**Use:** Optional

-**Description:** Input that the resource is referred to.

2. **referredTo:** *Output* [0..*]

-**Use:** Optional

-**Description:** Output that the resource is referred to.

(4) Constraints

The value of attribute "URI" should be unique in this metaclass.

5.3.10 Control_Construct

Control_Construct is a metaclass designating constraint relationship between sub-processes.

(1) Superclasses

Administered Item(from MDR)

(2) Attributes

1. **componentType:** *typeCodeOfControlConstruct* [0..1]

-**Use:** Mandatory

-**Description:** A type code specifying the type of control construct that is used in a composite.

NOTE The code set of control construct should be defined by each MFI Process registry, seeing Table D.3 in Annex D.

(3) References

1. **connects:** *Composite_Process* [0..1]

-**Use:** Mandatory

-**Description:** The composite processes connected by the specified type of control construct.

(4) Constraints

None

5.3.11 Precondition

Precondition is a metaclass designating a kind of condition that must always be true just prior to the execution of a process.

(1) Superclasses

Administered Item(from MDR)

(2) Attributes

1. description: *string* [1..1]

-Use: Mandatory

-Description: The formal description of a precondition.

(3) References

1. restricts: *Input* [0..*]

-Use: Optional

-Description: The inputs that the precondition restricts.

(4) Constraints

The precondition of a composite process should be specified by that of the involved processes and the corresponding control constructs

5.3.12 Postcondition

Postcondition is a metaclass designating a kind of condition that must always be true after the execution of a process.

(1) Superclasses

Administered Item(from MDR)

(2) Attributes

1. description: *String* [1..1]

-Use: Mandatory

-Description: The formal description of a postcondition.

(3) References

1. restricts: *Output* [0..*]

-Use: Optional

-Description: The outputs that the postcondition restricts.

(4) Constraints

The postcondition of a composite process should be specified by that of the involved processes and the corresponding control constructs

Annex A

(informative)

Examples of MFI Process registration

In this section, two cases will be studied to illustrate how to register various kinds of process models based on MFI-5 and enable semantic interoperability between them. It sounds that MFI-5 can harmonize with existing specifications related to process/process model.

Case 1: BravoAir reservation service (<http://www.daml.org/services/owl-s/1.0/examples.html>)

BravoAir reservation service is expressed in OWL-s to designate the processes of online flight booking. More specifically, BravoAir process consists of a sequence of sub-processes, involving two atomic processes respectively called GetDesiredFlightDetails and SelectAvailableFlight, and a composite process named BookFlight.

```

-->
- <process:CompositeProcess rdf:ID="BravoAir_Process">
  <rdfs:label>This is the top level process for BravoAir</rdfs:label>
- <process:composedOf>
  - <process:Sequence>
    - <process:components rdf:parseType="Collection">
      <process:AtomicProcess rdf:about="#GetDesiredFlightDetails" />
      <process:AtomicProcess rdf:about="#SelectAvailableFlight" />
      <process:CompositeProcess rdf:about="#BookFlight" />
    </process:components>
  </process:Sequence>
</process:composedOf>
</process:CompositeProcess>

```

BookFlight comprise two atomic processes Login and ConfirmReservation.

```

- <process:CompositeProcess rdf:ID="BookFlight">
  - <process:composedOf>
    - <process:Sequence>
      - <process:components rdf:parseType="Collection">
        <process:AtomicProcess rdf:about="#Login" />
        <process:AtomicProcess rdf:about="#ConfirmReservation" />
      </process:components>
    </process:Sequence>
  </process:composedOf>
</process:CompositeProcess>

```

Each instance of Process has several Input and Output, carrying Artifacts to fulfill a common purpose. Meanwhile, corresponding control constraints will also be added to Input and Output respectively to restrict the execution of the process model and obtain the desirable results.

```

- <process:AtomicProcess rdf:ID="ConfirmReservation">
  <process:hasInput rdf:resource="#ReservationID_In" />
  <process:hasInput rdf:resource="#Confirm_In" />
  <process:hasOutput rdf:resource="#PreferredFlightItinerary_Out" />
  <process:hasOutput rdf:resource="#AcctName_Out" />
  <process:hasOutput rdf:resource="#ReservationID_Out" />
  <process:hasEffect rdf:resource="#HaveSeat" />
</process:AtomicProcess>
- <process:Input rdf:ID="ReservationID_In">
  <process:parameterType rdf:resource="http://www.daml.org/services/owl-s/1.0/Concepts.owl#ReservationNumber" />
</process:Input>
- <process:Input rdf:ID="Confirm_In">
  <process:parameterType rdf:resource="http://www.daml.org/services/owl-s/1.0/Concepts.owl#Confirmation" />
</process:Input>
- <process:UnConditionalOutput rdf:ID="PreferredFlightItinerary_Out">
  <process:parameterType rdf:resource="http://www.daml.org/services/owl-s/1.0/Concepts.owl#FlightItinerary" />
</process:UnConditionalOutput>
- <process:UnConditionalOutput rdf:ID="AcctName_Out">
  <process:parameterType rdf:resource="http://www.daml.org/services/owl-s/1.0/Concepts.owl#AcctName" />
</process:UnConditionalOutput>
- <process:UnConditionalOutput rdf:ID="ReservationID_Out">
  <process:parameterType rdf:resource="http://www.daml.org/services/owl-s/1.0/Concepts.owl#ReservationNumber" />
</process:UnConditionalOutput>
- <process:UnConditionalEffect rdf:ID="HaveSeat">
  <process:ceEffect rdf:resource="http://www.daml.org/services/owl-s/1.0/Concepts.owl#HaveFlightSeat" />
</process:UnConditionalEffect>
</rdf:RDF>

```

Then the registration information of Composite_Process, Atomic_Process, Process_Model, Control_Construct and Resource will be illustrated in Figure A.1.

Process	
name	BravoAir_Process
URI	URI_BravoAir_Process
administration_Record	#
type	C
describedBy	BravoAir_ProcessModel
hasInput	Input: DepartureAirport_In
	Input: ArrivalAirport_In
	Input: RoundTrip_In

hasOutput	Output: ReservationID_Out
	Output: PreferredFlightItinerary_Out

consistsOf	Process01: GetDesiredFlightDetails
	Process02: SelectAvailableFlight
	Process03: BookFlight

Process	
name	Login
URI	URI_Login
administration_Record	#
type	C
describedBy	Login_ProcessModel
hasInput	Input: AcctName_In
	Input: Password_In

Control_Construct	
componentType	SEQ
restricts	BravoAir_Process

Resource	
name	AcctName
referredTo	Input:AcctName_In
URI	http://www.daml.org/services/owl-s/1.0/Concepts.owl#AcctName

Process_Model	
name	BravoAir_ProcessModel
URI	URI_BravoAir_ProcessModel
describes	BravoAir_Process
modelType	OWL-S

Figure A.1 – Registration information of BravoAir Reservation Service

Case 2: process for manufacturing a GT350 (from Annex C of ISO 18629-12)

The GT-350 manufacturing process is divided into six main areas. They are “make interior”, “make drive”, “make trim”, “make engine”, “make chassis” and “final assembly”. Particularly, the first five tasks are all unordered with respect to each other but they must all be complete before final assembly takes place.

The following is the fragment of manufacturing sub-process “make_engine”.

```

(subactivity make_block make_engine)
(subactivity make-harness make_engine)
(subactivity make-wires make_engine)
(subactivity assemble_engine make_engine)

(forall (?occ)
  (⊕(occurrence_of ?occ make_engine)
    (exists (?occ1 ?occ2 ?occ3 ?occ4)
      (and (occurrence_of ?occ1 make_block)
           (occurrence_of ?occ2 make_harness)
           (occurrence_of ?occ3 make_wires)
           (occurrence_of ?occ4 assemble_engine)
           (subactivity_occurrence ?occ1 ?occ)
           (subactivity_occurrence ?occ2 ?occ)
           (subactivity_occurrence ?occ3 ?occ)
           (subactivity_occurrence ?occ4 ?occ)
           (forall (?s1 ?s2 ?s3 ?s4)
             (implies (and (leaf_occ ?s1 ?occ1)
                          (leaf_occ ?s2 ?occ2)
                          (leaf_occ ?s3 ?occ3)
                          (root_occ ?s4 ?occ4))
                       (min_precedes ?s1 ?s4 make_engine)
                       (min_precedes ?s2 ?s4 make_engine)
                       (min_precedes ?s3 ?s4 make_engine))))))

```

Figure A.2 shows the registration information of “make_engine”.

Process	
name	<i>Process010</i>
URI	<i>URI_Process_010</i>
administration_Record	#
type	<i>C</i>
describedBy	<i>ProcessModel010</i>
consistsOf	<i>Process011: making_harness</i>
	<i>Process012: making_wires</i>
	<i>Process013: making_block</i>

Process_Model	
name	<i>ProcessModel010</i>
URI	<i>URI_ProcessModel010</i>
describes	<i>Process010</i>
modelType	<i>PSL</i>

Control_Construct	
type	<i>Join</i>
restricts	<i>Process010</i>

Process	
name	<i>making_engine</i>
URI	<i>URI_making_engine</i>
administration_Record	#
type	<i>C</i>
describedBy	<i>PM_making_engine</i>
consistsOf	<i>Process010</i>
	<i>Process015: assemble_engine</i>

Process_Model	
name	<i>ProcessModel_making_engine</i>
URI	<i>URI_ProcessModel_making_engine</i>
describes	<i>making_engine</i>
modelType	<i>PSL</i>

Control_Construct	
componentType	<i>Sequence</i>
restricts	<i>making_engine</i>

Figure A.2 – Registration information of sub-process "make_engine"

Annex B (informative)

Collaboration between MFI members

Annex B shows the exemplary collaboration between MFI members, especially the semantic interoperation based on the cooperation between MFI Process and MFI Ontology registration.

There are two process models existing on the web. One is named Withdraw Deposit and described with BPEL. If we input valid AccountID and password, the output will be an amount of RMB. The other is Sell Goods expressed in OWL-s, which is used to get expected Goods by Currency. Now users intend to exchange information between these two process models and enable interoperation between them.

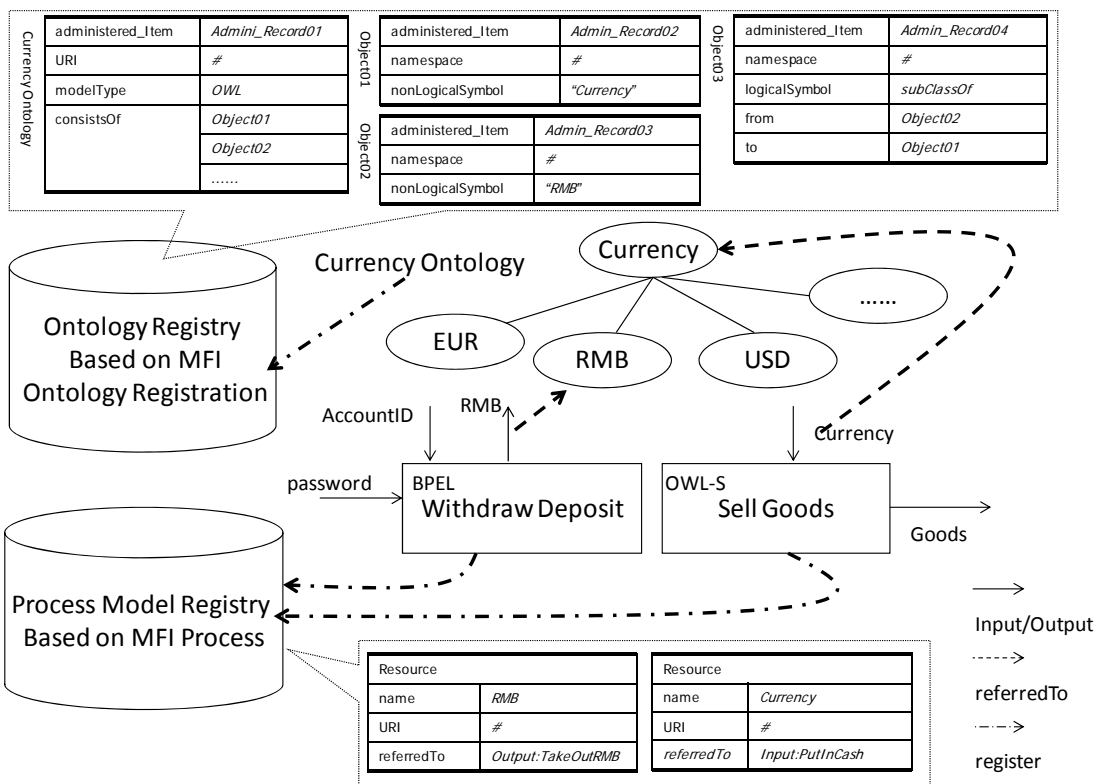


Figure B.1 – Semantic interoperation based on MFI Process and MFI Ontology registration

For this purpose, it is required to match the output message of Withdraw Deposit process to the input message of Sell Goods process. First of all, we will search the process model registry based on MFI Process registration for semantic registration of both Withdraw Deposit and Sell Goods, as the bottom of Figure 8 suggests. Obviously, input/output of these two models are referred to designated artifacts which are interrelated semantically because RMB is a kind of Currency in Currency Ontology. Then registration information of Currency Ontology on the top of Figure 8 explains the inherent relation between these two concepts in a precise way. Therefore, it is feasible for agent to fill in the Artifact_Constraint within process models and establish semantic relation between them. That is, there is semantic relation between the output message of Withdraw Deposit process and the input of Sell Goods process, which will promote interoperation between these two process models.

Annex C

(informative)

List of process modeling languages

It is advisable that the value of attribute “name” of “Process_Modeling_Language” can be one of the values in column “name” of Table 1.

Table C.1 – List of Process_Modeling_Languages

Name	Description
OWL-S	A language that conforms to “OWL Web Ontology Language for Web Service”, which specifying Semantic Markup for Web Services. (see bibliography item [1])
BPMN	Business Process Modeling Notation, Object Management Group, 2008. (see bibliography item [2])
BPEL	Business Process Execution Language for Web Service (BPEL/BPEL4WS), 2003-05-03, Version 1.1. (see bibliography item [3])
UML	A language that conforms to ISO/IEC 19501 Information technology – Open Distributed Processing – Unified Modeling Language (UML) Version 1.4.2.
PSL	A language that conforms to ISO/IEC 18629 Process Specification Language.
IDEF0	IDEF0(Integration Definition for Function Modeling) is a function modeling methodology for describing manufacturing functions, which offers a functional modeling language for the analysis, development, reengineering, and integration of information systems; business processes; or software engineering analysis. (see bibliography item [4])
IDEF3	IDEF3(Integrated DEFinition for Process Description Capture Method) is a business process modeling method complementary to IDEF0. It is a scenario-driven process flow description capture method intended to capture the knowledge about how a particular system works. (see bibliography item [5])
DFD	DFD(Data Flow Diagram) is a graphical representation of the flow of data through an information system.
Other	

Annex D
(informative)
Code type lists

MFI Process provides a generic and flexible facility to register process models described by different modeling languages. MFI Process registries established by specific applications are quite different. The annex of this document provides three kinds of typed code lists, i.e. code type list of state of processes, code type list of state of resources and code type list of control constructs, as examples.

Table D.1 shows the code set of state of a process. Any revision on the listed codes is allowed for specific MFI Process registries.

Table D.1 – Code type list of state of a process

Code	Name	Description
NOR	Normal	Normal indicates that the process executes well.
EXP	Exception	Exception states that the process cannot execute as designed and has been halted.
EXT	Exit	Exit states that the process has been executed successfully. The condition for exiting a process should be specified.
PAU	Pause	Pause states that the process stops temporarily. The condition for pausing and restarting the paused process should be specified respectively.
Others		

Table D.2 shows the code set of state of a resource. Any revision on the listed codes is allowed for specific MFI Process registries.

Table D.2 – Code type list of state of a resource

Code	Name	Description
OCU	Occupied	Occupied indicates that the resource has been used in a specified process and cannot participate in another one.
VAC	Vacant	Occupied indicates that the resource can be used in a specified process.
Others		

Table D.3 shows the code set of control construct. Any revision on the listed codes is allowed for specific MFI Process registries.

Table D.3 – Code type list of control construct

Code	Description
AnyOrder	AnyOrder is defined to allow execution of processes in an unspecified order. It can be expressed as a set of processes.
Sequence	Sequence is defined to execute processes in order. It can be expressed as an array of processes.
Choice	Choice is defined for the case that only one of the successor processes can be executed after performing its predecessors. Its representation must include the guard condition that specifies which successor process is allowed to execute.
Split	Split is a defined for the case that the successors of a process might be performed in parallel. It should explicitly indicate whether the divided processes must execute synchronously or not.
Join	Join is defined for the case that predecessors of a process must be executed before it is performed. It should explicitly indicate whether the processes before join node must complete synchronously or not.
Loop	Loop is defined for the case that some predecessors of a process may be executed iteratively until it is performed. It should record the condition of stopping the iterating process. Meanwhile, the process designating the starting point of Loop and that designating the end of Loop must be specified explicitly.
Others	

Bibliography

- [1] OWL Web Ontology Language for Web Service, W3C. Available at: <http://www.daml.org/services/owl-s/1.1/>.
- [2] Business Process Modeling Notation(BPMN 1.1), OMG Document Number: formal/2008-01-17, February, 2008. Available at: <http://www.omg.org/spec/BPMN/1.1/PDF>.
- [3] Business Process Execution Language for Web Services(BPEL 1.1), 2003-5-5. Available at: <http://xml.coverpages.org/BPELv11-May052003Final.pdf>.
- [4] IDEF0 Integration Definition for Function Modeling, 1993-12-31. Available at: <http://www.idef.com/pdf/idef0.pdf>.
- [5] IDEF3 Process Description Capture Method Report, September 1995. Available at: http://www.idef.com/pdf/Idef3_fn.pdf.