

~~ISO/IEC JTC 1/SC 32 N xxxx~~

Date: 2010-05-07

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.

Deleted: ISO/IEC J

Formatted: Font color: Blue

Deleted: 1

Formatted: Font color: Blue

Formatted: Font color: Blue

Deleted: 32

Field Code Changed

Deleted: 2

Formatted: Font: 10 pt, Not Bold, Font color: Blue

Deleted: ISO/IEC J

Formatted: Font: 10 pt, Not Bold, Font color: Blue

Deleted: 1

Formatted: Font: 10 pt, Not Bold, Font color: Blue

Deleted: 32

Formatted: Font: 10 pt, Not Bold, Font color: Blue

Formatted: Font: 10 pt, Not Bold, Font color: Blue

Deleted: ANSI

Field Code Changed

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	vi
Introduction	vii
1 Scope	1
2 Conformance	1
2.1 General	1
2.2 Degree of conformance	1
2.3 Implementation Conformance Statement (ICS)	2
3 Normative references	2
4 Terms, definitions and abbreviated terms	2
4.1 Terms and definitions	2
4.2 Abbreviated terms	3
5 Structure of MFI PMR	4
5.1 Overview of MFI PMR	4
5.2 Relationship between MFI PMR and other parts in MFI	5
5.3 MFI PMR	6
5.3.1 Process	6
5.3.2 Process_Model	7
5.3.3 Process_Modeling_Language	8
5.3.4 Composite_Process	8
5.3.5 Atomic_Process	9
5.3.6 Resource	9
5.3.7 Event	10
5.3.8 Dependency_Construct	10
5.3.9 AnyOrder	11
5.3.10 Sequence	11
5.3.11 AND_Split	11
5.3.12 AND_Join	12
5.3.13 OR_Split	12

Deleted: vii

Deleted: viii

Deleted: 8

Deleted: 9

Deleted: 10

5.3.14 OR_Join..... 13

5.3.15 XOR_Split..... 13

5.3.16 XOR_Join..... 14

5.3.17 Conditional Process..... 14

Annex A (informative) Examples of MFI PMR registration..... 16

Annex B (informative) Collaboration between MFI members..... Error! Bookmark not defined.

Annex C (informative) Metaclasses inherited form MDR 31

Annex D (informative) List of process modeling languages 32

Annex E (informative) Code type lists..... 35

Deleted: Condition_Result

Deleted: 15

Deleted: 30

Figures and tables

Figure 1 – The scope of MFI PMR 1

Figure 2 – The metamodel for process model registration..... 5

Figure 3 – Relationship between MFI PMR and some parts in MFI 6

Figure A.1.1– Registration information of composite process “BravoAir_Process”, atomic processes
 “GetDesiredFlightDetails” and “SelectAvailableFlight” 17

Figure A.1.2– Registration information of composite processes “BookFlight” and “Login” 18

Figure A.1.3 – Registration information of atomic process “ConfirmReservation”. 19

Figure A.1.4 – Registration information of all resources..... 20

Figure A.2.1 – Registration information of process ” HandleOrder” 22

Figure A.2.2 – Registration information of two branches of “HandleOrder” 23

Figure A.2.3 – Registration information of sub-process “DeliverOrder” 24

Figure A.3.1– General registration information of process “TravelBooking” 26

Figure A.3.2– Registration information of three branches if no error exists after checking credit card 27

Figure A.3.3– Registration information of one branch about reserving cars 28

Figure A.3.4 Registration information of the loop process named Process00 30

Figure B.1 – Semantic interoperation based on MFI PMR and MFI Ontology registration **Error! Bookmark
 not defined.**

Table D.1 – List of Process_Modeling_Languages 32

Table D.2 – Mapping table of Process_Modeling_Languages 33

Table E.1 – Code type list of state of a process 35

Deleted: 16

Deleted: 17

Deleted: 18

Deleted: 19

Deleted: 21

Deleted: 22

Deleted: 23

Deleted: 25

Deleted: 26

Deleted: 27

Deleted: 29

Deleted: 30

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 satisfy discriminating users. Discovery and reuse of process models stored in different repositories become the key issues when the interoperation between them is available.

Many industrial consortia have contributed to the standardization of domain specific process models using various representation notations and description languages for different purpose. The 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 the languages it adopts and the platform it executes on.

This part of ISO/IEC 19763 intends to provide a metamodel to register administrative information of process models. On one hand, the registration information based on this part provides hints for discovering appropriate process models through the purpose they achieve to promote large-grain process reuse. On the other hand, a process expresses the semantics of a service to support semantic discovery of a service. So this part could specify how to orchestrate Web services to realize the specified process.

Deleted: provide better service for

Deleted: registered

Deleted: s

Deleted: to promote

Deleted: y

Deleted: it

Deleted: also

Deleted: ies

Deleted: w

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. For the purpose, it provides administrative information and common semantics of process models created with a specific process modeling language, including Process Specification Language (PSL), Business Process Modeling Notation (BPMN), and Web Ontology Language for Web Service (OWL-S), etc. In that case, the metamodel can help discover exposed functions of a process and reuse its components at different levels of granularity, rather than all of them. Figure 1 shows the scope of this part.

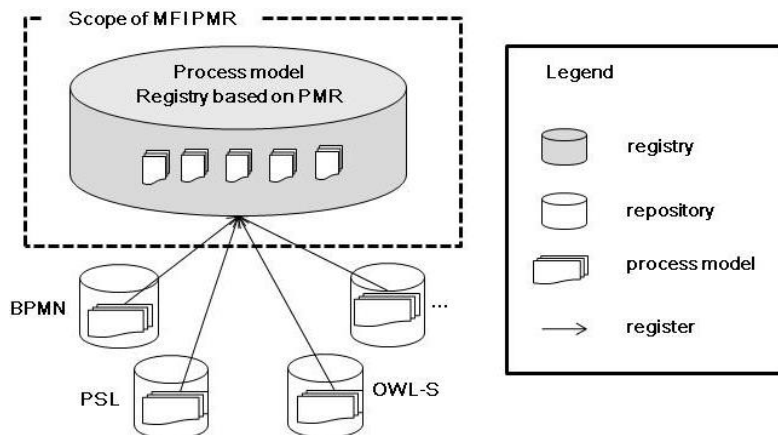


Figure 1 – The scope of MFI PMR

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 19763-3:2007, 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

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 by a dependency construct, which all participate in fulfilling a given purpose.

Deleted: according to constraints

4.1.2**Process model**

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

Deleted: NOTE If the process model has not been registered into the registry, it describes no process.

4.1.3**Process modelling language**

a kind of modeling language that is used by the process model to describe processes.

NOTE PSL, BPMN, OWL-S etc. are all process modeling languages.

4.1.4**Sub-process**

process that is contained in another process.

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

4.1.5**Atomic process**

process that does not have a sub-process.

4.1.6**Composite process**

process that consists of other processes and is organized by one and only one type of dependency construct.

4.1.7**Dependency construct**

uniformed connector to handle the execution order of processes.

Deleted: a control with

Deleted: in which the connected processes are executed

4.1.8**Resource**

anything participating in a process to help its performance.

NOTE Resource can be either physical or virtual things.

4.1.9**Event**

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

4.1.10**Conditional process**

split node containing a guard condition and the according process to be executed if the guard condition is satisfied.

Deleted: result

4.2 Abbreviated terms**MDR**

Metadata Registry

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

MFI

Metamodel framework for interoperability

ISO/IEC CD2 19763-5:2010(E)

[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 PMR

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

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 PMR

5.1 Overview of MFI PMR

MFI PMR provides a generic facility to register administrative information of process models described by specific modeling languages.

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** has no sub-process, while **Composite_Process** consists of sub-processes that can be either atomic or composite. In MFI PMR, **Composite_Process** has only one kind of **Dependency_Construct** to specify the order that its sub-processes should follow. More specifically, **Dependency_Construct** can be generalized as **AnyOrder**, **Sequence**, **AND_Split**, **AND_Join**, **OR_Split**, **OR_Join**, **XOR_split** and **XOR_Join**. **AnyOrder** means that all the sub-processes are performed in an unspecified order. **Sequence** implies that all the sub-processes have to be executed in order. **And_Split** is a metaclass designating that after the preceding process has been completed all processes in a given collection will be started. **And_Join** designates that the execution of processes in a given collection must have been all completed before the successor starts. **OR_Split** specifies that after the predecessor has been completed one or more processes from a given collection will be started if the according specified guide condition is satisfied. **OR_Join** is used to designate that one or more processes whose according specified guide condition is satisfied from a given collection must have been completed before the successor starts. **XOR_Split** describes that after the preceding process has been executed only one process whose according specified guide condition is satisfied from a given collection will be started. **XOR_Join** is a metaclass designating that only one process whose according specified guide condition is

satisfied from a given collection must have been completed before the successor starts.

Conditional Process is a metaclass designating a split node of “OR_Split” or “XOR_Split” which contains a guard condition and the according process to be executed if the guard condition is satisfied.

A process may consume some **Resource**, create other **Resource** or just refer to some kind of **Resource**, which can be any virtual or physical thing participating in a process to help its performance, or to achieve the given purpose of **Process**. consume Besides, a resource staying in a specific kind of state or just the existing of a resource may initiate an event and then trigger some processes.

Figure 2 shows the metamodel for process model registration.

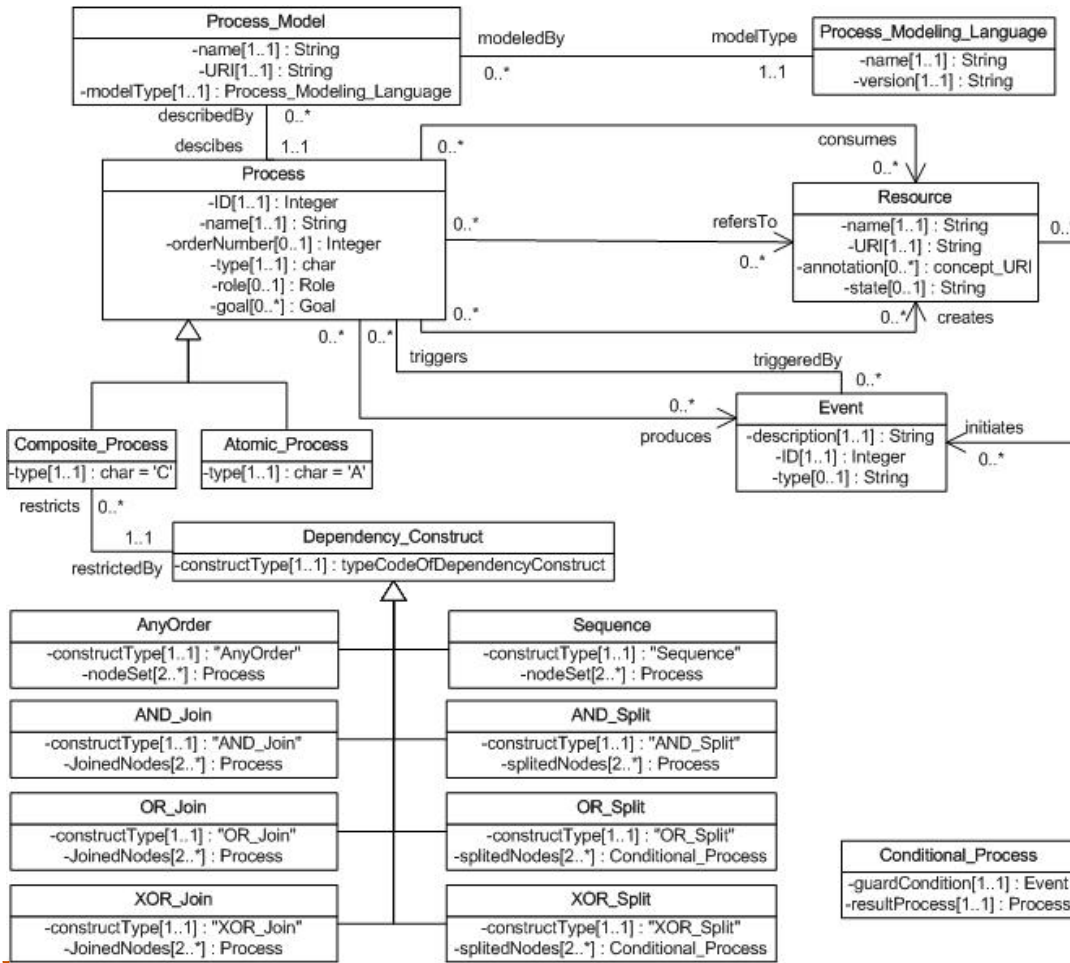
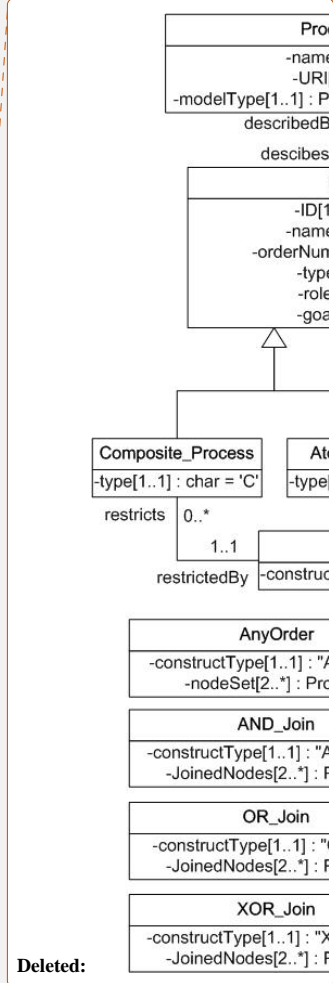


Figure 2 – The metamodel for process model registration

5.2 Relationship between MFI PMR and other parts in MFI

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

- Deleted: Condition_Result
- Deleted: split
- Deleted: ed
- Deleted: and
- Deleted: Particularly, the same instance of Resource can be
- Deleted: resume
- Deleted: d or created by any processes in different cases.



A process can achieve a set of goals, which are instances of **Goal** in MFI Role&Goal. The relationship between MFI PMR and MFI Role&Goal means that on one hand, a goal can be achieved by zero to many instances of **Process** and a process can achieve zero to one instances of **Goal**. On the other hand, a **Role** can involve zero to many instances of **Process**, and a process can be performed by zero to many instances of actors playing specific Roles. Similarly, **Service** in MFI Service can be used to realize **Process**. The relationship between MFI PMR and MFI Service means that a process can be realized by zero to many instances of **Service** and a service can also achieve zero to many **Process**.

The attribute “annotation” of **Resource** can be declared as the URI of the registered **Ontology_Atomic_Construct** based on MFI Ontology Registration. It means that the concepts in ontology can be used to annotate resources participating in a process.

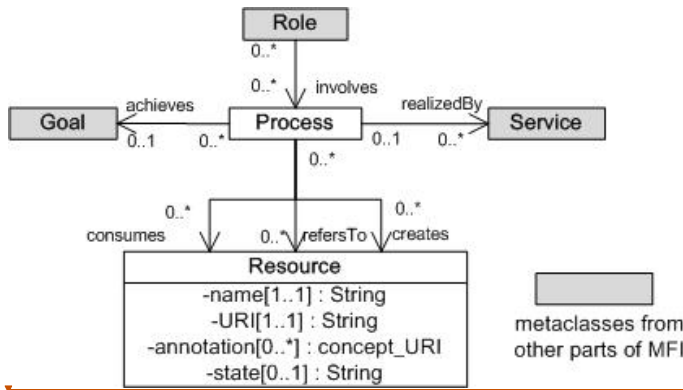


Figure 3 – Relationship between MFI PMR and some parts in MFI

5.3 MFI PMR

The MFI PMR 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

-Description: description for content and purpose of attribute

(3) References

n. reference name: datatype and multiplicity

-Description: description for content and purpose of attribute

(4) Constraints

-constraints specified if necessary, in natural language

5.3.1 Process

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

Deleted: one

Deleted: many

Deleted: playing a specific

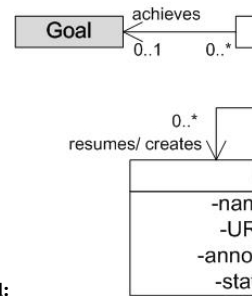
Deleted: only one

Deleted: Type

Deleted: Ontology_Whole, Ontology_Component or

Deleted:

Deleted: and its ingredients



Deleted:

(1) Superclasses

Administered Item(from MDR)

(2) Attributes

1. **ID:** Integer[1..1]

-Description: The unique identifier of a process.

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

-Description: Name of a process.

3. **orderNumber:** Integer[0..1]

-Description: A number allocating to a process. It's an optional attribute to identify a process.

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

-Description: Type of a registered process. 'A' denotes Atomic_Process and 'C' denotes Composite_Proces.

5. **goal:** Goal[0..1]

-Description:The common purpose that a process should achieve. It implies the main function of the process..

6. **role:** Role[0..*]

-Description: The role that involves a process.

(3) References

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

-Description: processes involved in another specified process.

2. **decribedBy:** Process_Model[0..*]

-Description: Process_Model specifying a specification of the process.

3. **consumes:** Resource[0..*]

-Description: Input resource that must be consumed in the execution of a process.

4. **creates:** Resource[0..*]

-Description: Output resource that is created as the result after executing a process.

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

-Description: The resource that is referred to in the execution of a process.

6. **produces:** Event[0..*]

-Description: The event that a process produces, namely the condition that should be satisfied after execution of a process.

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

-Description: The event that triggers a process, namely the condition that should be satisfied before execution of a process.

(4) Constraints

The value of attributes "ID" of a process 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.

Deleted: 1

Deleted: resume

Deleted: Input that must be transferred to the execution of a process.

Deleted: resume

Deleted: Output that is generated as the results after executing a process.

Deleted: 5. **changesState:**
Resource [0..*]¶
-Description: The resource whose state is changed to another after executing a process.¶

Deleted: its parent process

(1) Superclasses

Administered Item(from MDR)

(2) Attributes

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

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

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

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

(3) References

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

-**Description:** The process described by a process model.

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

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

(4) Constraints

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

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]*

-**Description:** Name of the process modeling language that is used by a process model to describe a process. Its value can be one of the values in column "name" of Table D.1 in Annex D.

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

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

(3) References

1. **modeledBy:** *Process_Model[0..*]*

-**Description:** The process model that adopts the process 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 or composite. The involved sub-processes are connected by zero or one kind of control construct.

(1) Superclasses

Process

(2) Attributes

1. **type:** 'C'

Deleted: 0

Deleted: NOTE: If the process model has not been registered into the metamodel, the cardinality is 0..1

-Description: The process is a composite process.

(3) References

1. **restrictedBy:** *Dependency_Construct[1..1]*

-Description: The specified type of Control_Construct that is used to connect subprocesses involved in the composite process.

(4) Constraints

The value of attribute “name” has to be unique in this metaclass.
Exists at least one Process in this Composite_Processs.

5.3.5 Atomic_Process

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

(1) Superclasses

Process

(2) Attributes

1. **type:** ‘A’

-Description: The process is an atomic process.

NOTE: An empty process which does nothing is also an atomic process, described as N/A.

(3) References

None

(4) Constraints

None.

5.3.6 Resource

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

(1) Superclasses

Registered Item(from MDR)

(2) Attributes

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

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

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

-Description: URI where a resource exists.

3. **annotation:** *concept_URI[0..*]*

-Description: URI of the registered Ontology_Atomic_Construct based on MFI Ontology Registration. The concepts in ontology can be used to annotate resources participating in a process.

4. **state:** *String[0..1]*

-Description: The current state that a resource remains to describe a state transition.

(3) References

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

-Description: The process that consumes a kind of resource.

Deleted: **Type**

Deleted: **Ontology_Whole, Ontology_Component or**

Deleted: **O**

Deleted: **and its ingredients**

Deleted: **resume**

Deleted: **resume**

2. createdBy: *Process[0..*]*

-Description: The process that creates a kind of resource.

3. stateChangedBy: *Process[0..*]*

-Description: The process that changes the state of a kind of resource to another.

4. initiates: *Event[0..*]*

-Description: A resource staying in a specific kind of state, or just the existing of a resource may initiate an event and then trigger some process.

(4) Constraints

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

5.3.7 Event

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

(1) Superclasses

Registered Item(from MDR)

(2) Attributes

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

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

2. ID: *Integer[1..1]*

-Description: The unique identifier of an event.

3. type: *String[0..1]*

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

(3) References

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

-Description: The process triggered by the event.

(4) Constraints

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

5.3.8 Dependency_Construct

Dependency_Construct is an abstract metaclass designating the order in which processes are executed.

(1) Superclasses

Administered Item(from MDR)

(2) Attributes

1. constructType: *typeCodeOfDependencyConstruct [0..1]*

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

NOTE *TypeCodeOfDependencyConstruct* is an enumeration type including "AnyOrder", "Sequence", "AND_Split", "AND_Join", "OR_Split", "OR_Join", "XOR_Split" and "XOR_Join" as 8 elements. It is also the code set of *Dependency_Construct* in this document, which can be extended by each MFI PMR registry.

(3) References

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

-Description: The composite processes using the specified type of control construct to connect its sub-processes.

(4) Constraints

None

5.3.9 AnyOrder

AnyOrder is a metaclass designating that processes are allowed to execute in an unspecified order.

(1) Superclasses

Dependency_Construct

(2) Attributes

1. **constructType:** *AnyOrder*

-Description: Sub-processes of the composite process are connected by AnyOrder.

2. **nodeSet:** *Process[2..*]*

-Description: Sub processes in the node set are connected by AnyOrder.

(3) References

None

(4) Constraints

None

5.3.10 Sequence

Sequence is a metaclass designating that the processes have to execute in order.

(1) Superclasses

Dependency_Construct

(2) Attributes

1. **constructType:** *Sequence*

-Description: Sub-processes of the composite process are connected by Sequence.

2. **nodeSet:** *Process[2..*]*

-Description: Processes in the set are executed in order. It can be expressed as an array of Processes.

(3) References

None

(4) Constraints

None

5.3.11 AND_Split

AND_Split is a metaclass designating that after the preceding process has been completed all processes in a given collection will be started.

(1) Superclasses

Dependency_Construct

(2) Attributes

1. constructType: AND_Split

-Description: Sub processes of the composite process are connected by AND_Split.

2. splitNodes: Process[2..*]

-Description: The processes to be executed in parallel after its preceding process have been completed.

3. synchronismOfSplitNodes: Boolean[1..1]

-Description: The processes to be divided have to complete synchronously or not. The default is true.

(3) References

None

(4) Constraints

The split and join construct must appear in pair. If the process containing AND_Split also contains a join Dependency_Construct, the type must be AND_Join, but not OR_Split or XOR_Split.

5.3.12 AND_Join

AND_Join is a metaclass designating that execution of processes in a given collection must have been all completed before the successor starts.

(1) Superclasses

Dependency_Construct

(2) Attributes

1. constructType: AND_Join

-Description: Sub processes of the composite process are connected by AND_Join.

2. joinedNodes: Process[2..*]

-Description: The processes that must have been all completed before the successor starts.

3. synchronismOfJoinedNodes: Boolean[1..1]

-Description: The processes to be merged have to complete synchronously or not. The default is true.

(3) References

None

(4) Constraints

None

5.3.13 OR_Split

OR_Split is a metaclass designating that after the predecessor has been completed one or more processes from a given collection will be started if the according specified guard condition is satisfied.

(1) Superclasses

Dependency_Construct

(2) Attributes

1. constructType: OR_Split

Deleted: split

Deleted: ed

Deleted: Split

Deleted: ed

-Description: Sub processes of the composite process are connected by OR_Split.

2.splitNode: [Conditional Process](#)[2..*]

-Description: The divided processes to be performed in parallel if the according guard conditions are satisfied.

3.synchronismOfSplitNodes: [Boolean](#)[1..1]

-Description: The processes to be divided have to complete synchronously or not. The default is true.

(3) References

None

(4) Constraints

- i. The split and join construct must appears in pair. If the process containing OR_Split also contains a join Dependency_Construct, the type must be OR_Join, but not AND_Split or XOR_Split.
- ii. The condition in each guard condition of [split](#) node may be satisfied in the same situation.

Deleted: *split*

Deleted: *ed*

Deleted: *Condition_Result*

Deleted: *Split*

Deleted: *ed*

Deleted: *split*

Deleted: *ed*

5.3.14 OR_Join

OR_Join is a metaclass designating that one or more processes whose according specified guard condition is satisfied from a given collection must have been completed before the successor starts.

(1) Superclasses

Dependency_Construct

(2) Attributes

1. componentType: *OR_Join*

-Description: Sub processes of the composite process are connected by OR_Join.

2. joinedNode: [Process](#)[2..*]

-Description: The merged processes to be performed in parallel.

3. synchronismOfJoinedNodes: [Boolean](#)[1..1]

-Description: The processes to be merged have to complete synchronously or not. The default is true.

(3) References

None

(4) Constraints

None

5.3.15 XOR_Split

XOR_Split is a metaclass designating that after the preceding process has been executed one and only one process from a given collection will be started if the according specified guard condition is satisfied.

(1) Superclasses

Dependency_Construct

(2) Attributes

1. constructType: XOR_Split

-Description: Sub processes of the composite process are connected by XOR_Split.

2. splitNode: Conditional_Process[2..*]

-Description: The divided processes to be performed in parallel.

3. synchronismOfSplitNodes: Boolean[1..1]

-Description: The processes to be divided have to complete synchronously or not. The default is true.

(3) References

None

(4) Constraints

- i. The split and join construct must appears in pair. If the process containing XOR_Split also contains a join Dependency_Construct, the type must be XOR_Join, but not AND_Split or OR_Split.
- ii. The condition in each guard condition of split node should be unique with no overlap and mutually exclusive.

5.3.16 XOR_Join

XOR_Join is a metaclass designating that one and only one process whose according specified guard condition is satisfied from a given collection must have been completed before the successor starts.

(1) Superclasses

Dependency_Construct

(2) Attributes

1. componentType: XOR_Join

-Description: Sub processes of the composite process are connected by XOR_Join.

2. joinedNode: Process[2..*]

-Description: The merged processes to be performed in parallel.

3. synchronismOfJoinedNodes: Boolean[1..1]

-Description: The processes to be merged have to complete synchronously or not. The default is true.

(3) References

None

(4) Constraints

None

5.3.17 Conditional_Process

Conditional_Process is a metaclass designating a split node of "OR_Split" or "XOR_Split" which contains a guard condition and the according process to be executed if the guard condition is satisfied.

(1) Superclasses

Registered Item(from MDR)

(2) Attributes

Deleted: split

Deleted: ed

Deleted: Condition_Result

Deleted: Split

Deleted: ed

Deleted: split

Deleted: ed

Deleted: Condition_Result

Deleted: Condition_Result

Deleted: split

Deleted: ed

1. guardCondition: *Event[1..1]*

-Description: The guard condition in the "OR_Split" or "XOR_Split" to decide which process to be executed

2. resultProcess: *Process[1..1]*

-Description: The process to be performed if the guard condition is satisfied.

(3) References

None

(4) Constraints

None

Annex A (informative)

Examples of MFI PMR 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.

Registration 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>

- <process:AtomicProcess rdf:ID="GetDesiredFlightDetails">
  <process:hasInput rdf:resource="#DepartureAirport_In" />
  <process:hasInput rdf:resource="#ArrivalAirport_In" />
  <process:hasInput rdf:resource="#OutboundDate_In" />
  <process:hasInput rdf:resource="#InboundDate_In" />
  <process:hasInput rdf:resource="#RoundTrip_In" />
</process:AtomicProcess>

- <process:Input rdf:ID="DepartureAirport_In">
  <process:parameterType rdf:resource="http://www.daml.org/services/owl-s/1.0/Concepts.owl#Airport" />
</process:Input>

- <process:Input rdf:ID="ArrivalAirport_In">
  <process:parameterType rdf:resource="http://www.daml.org/services/owl-s/1.0/Concepts.owl#Airport" />
</process:Input>

- <process:Input rdf:ID="OutboundDate_In">
  <process:parameterType rdf:resource="http://www.daml.org/services/owl-s/1.0/Concepts.owl#FlightDate" />
</process:Input>

- <process:Input rdf:ID="InboundDate_In">
  <process:parameterType rdf:resource="http://www.daml.org/services/owl-s/1.0/Concepts.owl#FlightDate" />
</process:Input>

- <process:Input rdf:ID="RoundTrip_In">
  <process:parameterType rdf:resource="http://www.daml.org/services/owl-s/1.0/Concepts.owl#RoundTrip" />
</process:Input>
```

```

- <process:AtomicProcess rdf:ID="SelectAvailableFlight">
  <process:hasInput rdf:resource="#PreferredFlightItinerary_In" />
  <process:hasOutput rdf:resource="#AvailableFlightItineraryList_Out" />
</process:AtomicProcess>
- <process:Input rdf:ID="PreferredFlightItinerary_In">
  <process:parameterType rdf:resource="http://www.daml.org/services/owl-s/1.0/Concepts.owl#FlightItinerary" />
</process:Input>
- <process:UnconditionalOutput rdf:ID="AvailableFlightItineraryList_Out">
  <process:parameterType rdf:resource="http://www.daml.org/services/owl-s/1.0/Concepts.owl#FlightItineraryList" />
</process:UnconditionalOutput>

```

Figure A.1.1 shows the registration information of composite process “BravoAir_Process”, atomic processes “GetDesiredFlightDetails” and “SelectAvailableFlight”.

Process00	
ID	0
name	BravoAir_Process
administration_Record	#
type	C
describedBy	Process_Model00:BravoAir_ProcessModel
consumes	Resource00: DepartureAirport_In Resource01: ArrivalAirport_In Resource02: RoundTrip_In
creates	Resource: ReservationID_Out Resource: PreferredFlightItinerary_Out
restrictedBy	Sequence00

Process01	
ID	1
name	GetDesiredFlightDetails
administration_Record	#
type	A
describedBy	Process_Model01: GetDesiredFlightDetails_ProcessModel
consumes	Resource00: DepartureAirport_In Resource01: ArrivalAirport_In Resource02: OutboundDate_In Resource03: InboundDate_In Resource04: RoundTrip_In

Process_Model00	
name	BravoAir_ProcessModel
URI	URL_BravoAir_ProcessModel
Describes	Process00:BravoAir_Process
modelType	OWL-S

Sequence00	
constructType	Sequence
nodeSet	Process01: GetDesiredFlightDetails Process02: SelectAvailableFlight Process03: BookFlight

Process_Model01	
name	GetDesiredFlightDetails_ProcessModel
URI	URL_GetDesiredFlightDetails_ProcessModel
Describes	Process01: GetDesiredFlightDetails_Process
modelType	OWL-S

Process_Model02	
name	SelectAvailableFlight_ProcessModel
URI	URL_SelectAvailableFlight_ProcessModel
Describes	Process02: SelectAvailableFlight_Process
modelType	OWL-S

Process02	
ID	2
name	SelectAvailableFlight
type	A
administration_Record	#
consumes	Resource05: PreferredFlightItinerary_In
creates	Resource06: AvailableFlightItineraryList_Out

Figure A.1.1– Registration information of composite process “BravoAir_Process”, atomic processes “GetDesiredFlightDetails” and “SelectAvailableFlight”

BookFlight coPMRise 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>

```

The following is the fragment of an atomic processes “Login” and its registration information in Figure A.1.2.

```

- <process:AtomicProcess rdf:ID="LogIn">
  <process:hasInput rdf:resource="#AcctName_In" />
  <process:hasInput rdf:resource="#Password_In" />
</process:AtomicProcess>
- <process:Input rdf:ID="AcctName_In">
  <process:parameterType rdf:resource="http://www.daml.org/services/owl-s/1.0/Concepts.owl#AcctName" />
</process:Input>
- <process:Input rdf:ID="Password_In">
  <process:parameterType rdf:resource="http://www.daml.org/services/owl-s/1.0/Concepts.owl#Password" />
</process:Input>

```

Process03	
ID	3
name	BookFlight
administration_Record	#
type	C
describedBy	Process_Model03:BookFlight_ProcessModel
consumes	Resource07: AcctName_In
	Resource08: Password_In
	Resource09: ReservationID_In
	Resource10: Confirm_In
creates	Resource11: PreferredFlightItinerary_Out
	Resource12: AcctName_Out
	Resource13: ReservationID_Out
restrictedBy	Sequence01

Process_Model03	
name	BookFlight_ProcessModel
URI	URI_BookFlight_ProcessModel
describes	Process03:BookFlight
modelType	OWL-S

Process04	
ID	4
name	Login
administration_Record	#
type	A
describedBy	Process_Model04:Login_ProcessModel
consumes	Resource07: AcctName_In
	Resource08: Password_In

Process_Model04	
name	Login_ProcessModel
URI	URI_Login_ProcessModel
describes	Process04:Login_Process
modelType	OWL-S

Sequence01	
constructType	Sequence
nodeSet	Process04: Login
	Process05: ConfirmReservation

Figure A.1.2– Registration information of composite processes “BookFlight” and “Login”

The following is the fragment of a composite process “ConfirmReservation” and its registration information in Figure A.3.

```

- <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>

```

Figure A.1.3 shows the registration information of atomic process “ConfirmReservation”.

Process05	
ID	4
name	ConfirmReservation
administration_Record	#
achieves	Confirmreservation
type	A
describedBy	Process_Model05: ConfirmReservation_ProcessModel
consumes	Resource09:ReservationID_In Resource10:Confirm_In
creates	Resource11:PreferredFlightItinerary_Out Resource12:AcctName_Out Resource13:ReservationID_Out
produces	Event00:HaveSeat

Process_Model05	
name	ConfirmReservation_ProcessModel
URI	URI_ConfirmReservation_ProcessModel
describes	Process05:ConfirmReservation
modelType	OWL-S

Event00	
Description	HaveSeat
ID	0

Resource14	
Name	FlightSeat
URI	http://www.daml.org/services/owl-s/1.0/Concepts.owl#HaveFlightSeat
initiates	Event00:HaveSeat

Figure A.1.3 – Registration information of atomic process “ConfirmReservation”.

Figure A.1.4 shows the registration information of all resources as input or output.

Resource00		Resource07	
name	<i>DepartureAirport_In</i>	name	<i>AcctName_In</i>
URI	<i>http://www.daml.org/services/owl-s/1.0/Concepts.owl#Airport</i>	URI	<i>http://www.daml.org/services/owl-s/1.0/Concepts.owl#AcctName</i>
Resource01		Resource08	
name	<i>rrivalAirport_In</i>	name	<i>Password_In</i>
URI	<i>http://www.daml.org/services/owl-s/1.0/Concepts.owl#Airport</i>	URI	<i>http://www.daml.org/services/owl-s/1.0/Concepts.owl#Password</i>
Resource02		Resource09	
name	<i>OutboundDate_In</i>	name	<i>ReservationID_In</i>
URI	<i>http://www.daml.org/services/owl-s/1.0/Concepts.owl#FlightDate</i>	URI	<i>http://www.daml.org/services/owl-s/1.0/Concepts.owl#ReservationNumber</i>
Resource03		Resource10	
name	<i>InboundDate_In</i>	name	<i>Confirm_In</i>
URI	<i>http://www.daml.org/services/owl-s/1.0/Concepts.owl#FlightDate</i>	URI	<i>http://www.daml.org/services/owl-s/1.0/Concepts.owl#Confirmation</i>
Resource04		Resource11	
name	<i>RoundTrip_In</i>	name	<i>PreferredFlightItinerary_Out</i>
URI	<i>http://www.daml.org/services/owl-s/1.0/Concepts.owl#RoundTrip</i>	URI	<i>http://www.daml.org/services/owl-s/1.0/Concepts.owl#FlightItinerary</i>
Resource05		Resource12	
name	<i>PreferredFlightItinerary_In</i>	name	<i>AcctName_Out</i>
URI	<i>http://www.daml.org/services/owl-s/1.0/Concepts.owl#FlightItinerary</i>	URI	<i>http://www.daml.org/services/owl-s/1.0/Concepts.owl#AcctName</i>
Resource06		Resource13	
name	<i>AvailableFlightItineraryList_Out</i>	name	<i>ReservationID_Out</i>
URI	<i>http://www.daml.org/services/owl-s/1.0/Concepts.owl#FlightItineraryList</i>	URI	<i>http://www.daml.org/services/owl-s/1.0/Concepts.owl#ReservationNumber</i>

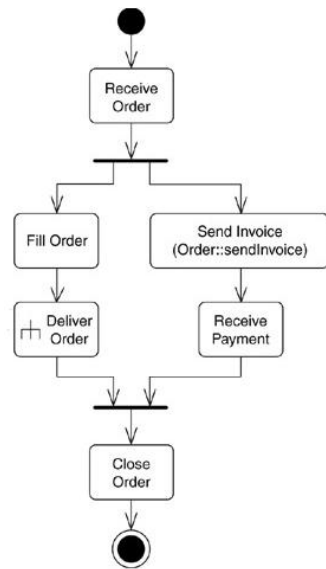
Figure A.1.4 – Registration information of all resources.

Registration Case 2: process for handle an order.

HandleOrder activity is expressed in UML to designate the processes of online order dealing with. More specifically, *HandleOrder* consists of a sequence of sub-processes, involving *ReceiveOrder*, the handle details and *CloseOrder*. The handle details include a split and then a join with branches *FillOrder*, *SendInvoice*, and the subsequent actions occurring in parallel. And one of its sub-activities named *DeliverOrder* has its own sub-activity.

The following is the UML activity diagram of *HandleOrder*.

Deleted: activity



And the following is the UML activity diagram of *DeliverOrder* , sub-activity of *HandleOrder*

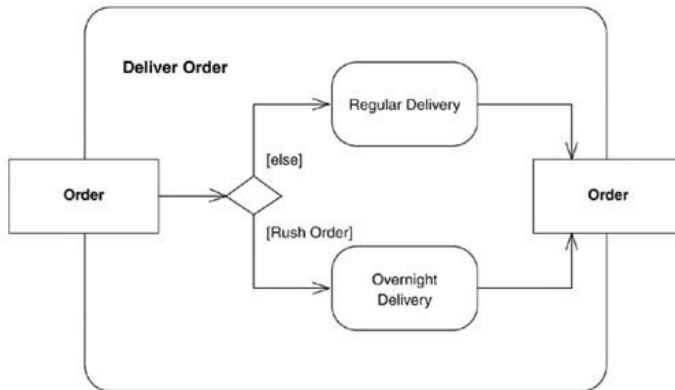


Figure A.2.1 shows the registration information of *HandleOrder* and its 4 direct sub-processes .

Process00	
ID	0
name	HandleOrder
administration_Record	#
type	C
describedBy	Process_Model00: HandleOrder_ProcessModel
restrictedBy	Sequence00

Process01	
ID	1
name	ReceiveOrder
administration_Record	#
type	A

Process02	
ID	2
name	AND_Split00_Process
administration_Record	#
type	C
restrictedBy	AND_Split00

Process03	
ID	3
name	AND_Join00_Process
administration_Record	#
type	C
restrictedBy	AND_Join00

Process_Model00	
name	HandleOrder_ProcessModel
URI	URL_HandleOrder_ProcessModel
Describes	Process00::HandleOrder
modelType	UML

Sequence00	
constructType	Sequence
nodeSet	Process01:ReceiveOrder Process02:AND_Split00_Process Process03:AND_Join00_Process Process04:CloseOrder

AND_Split00	
constructType	AND_Split
nodeSet	Process05:Sequence01_Process Process06:Sequence02_Process

AND_Join00	
constructType	AND_Join
nodeSet	Process05:Sequence01_Process Process06:Sequence02_Process

Process04	
ID	4
name	CloseOrder
administration_Record	#
type	A
consumes	Resource00:Order

Figure A.2.1 – Registration information of process ” HandleOrder”

Figure A.2.2 shows the registration detail information of the two branches to deal with an order excluding ReceiveOrder and CloseOrder .

Process05	
ID	5
name	<i>sequence01_Process</i>
type	C
administration_Record	#
creates	<i>Resource00:Order</i>
restrictedBy	<i>Sequence01</i>

Process06	
ID	6
name	<i>sequence02_Process</i>
type	C
administration_Record	#
restrictedBy	<i>Sequence02</i>

Process07	
ID	7
name	<i>FillOrder</i>
administration_Record	#
type	A
creates	<i>Resource00:Order</i>

Process08	
ID	8
name	<i>DeliverOrder</i>
administration_Record	#
type	C
describedBy	<i>Process_Model01:DeliverOrder_ProcessModel</i>
consumes	<i>Resource00:Order</i>
creates	<i>Resource00:Order</i>
restrictedBy	<i>Sequece03</i>

Sequence01	
constructType	<i>Sequence</i>
nodeSet	<i>Process07:FillOrder</i> <i>Process08:DeliverOrder</i>

Sequence02	
constructType	<i>Sequence</i>
nodeSet	<i>Process09:SendInvoice</i> <i>Process10:ReceivePayment</i>

Process09	
ID	9
name	<i>SendInvoice</i>
administration_Record	#
type	A
produces	<i>Event00:sendInvoice</i>

Process10	
ID	10
name	<i>ReceivePayment</i>
administration_Record	#
type	A

Process_Model01	
name	<i>DeliverOrder_ProcessModel</i>
URI	<i>URI_DeliverOrder_ProcessModel</i>
Describes	<i>Process08:DeliverOrder</i>
modelType	<i>UML</i>

Sequence03	
constructType	<i>Sequence</i>
nodeSet	<i>Process11:XOR_Split00_Process</i> <i>Process12:XOR_Join00_Process</i>

Figure A.2.2 – Registration information of two branches of “HandleOrder”

Figure A.2.3 shows the registration information of details about one sub-activity named *DeliverOrder*.

Process11	
ID	11
name	XOR_Split00_Process
administration_Record	#
type	C
restrictedBy	XOR_Split00

Process12	
ID	12
name	XOR_Join00_Process
administration_Record	#
type	C
restrictedBy	XOR_Join00

Process13	
ID	7
name	RegularDelivery
administration_Record	#
type	A
creates	Resource00:Order

Process14	
ID	7
name	OvernightDelivery
administration_Record	#
type	A
creates	Resource00:Order

Event00	
description	SendInvoice
ID	0

XOR_Split00	
constructType	XOR_Split
nodeSet	Conditional_Process00 Conditional_Process01

XOR_Join00	
constructType	XOR_Split
nodeSet	Process13:RegularDelivery Process14:OvernightDelivery

Resource00	
name	Order
URI	URL_Order
state	rush/notRush
initiates	Event01:RushOrder / Event02:NotRushOrder

Event01	
description	RushOrder
ID	1

Event02	
description	NotRushOrder
ID	2

Conditional_Process00	
guardCondition	Event01:RushOrder
resultProcess	Process13:RegularDelivery

Conditional_Process01	
guardCondition	Event02:NotRushOrder
resultProcess	Process14:OvernightDelivery

Process11	
ID	11
name	XOF
administration_Record	#
type	C
restrictedBy	XOF

Process12	
ID	12
name	XOF
administration_Record	#
type	C
restrictedBy	XOF

Process13	
ID	7
name	Reg
administration_Record	#
type	A
creates	Res

Process14	
ID	7
name	Ove
administration_Record	#
type	A
creates	Res

Event00	
description	SendInv
ID	0

Deleted:

Figure A.2.3 – Registration information of sub-process “DeliverOrder”

Registration Case 3: process for Travel_Booking.

In this section, the case will be studied to illustrate how to register various kinds of process models based on BPMN.

The Process begins with the receipt of a request for a travel booking. After a check on the credit card, reservations are made for a flight, a hotel, and a car. The car reservation may take more than one attempt before it is successful. After all three reservations are confirmed, a reply is sent.

The following is the description of Travel_Booking using BPMN.

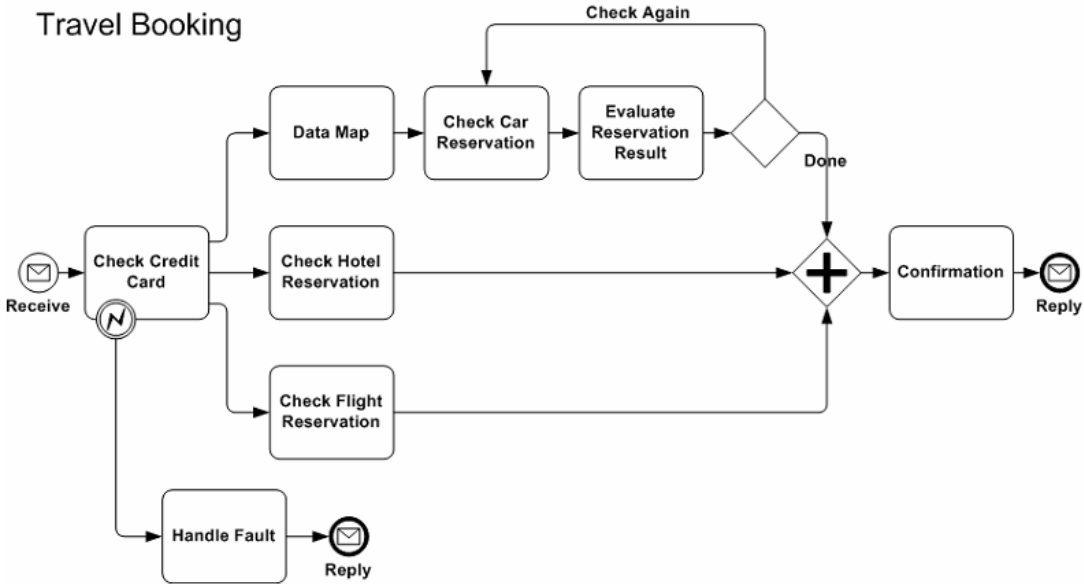


Figure A.3.1 shows the general registration information of process “TravelBooking”

Process_Model00	
name	<i>TravelBooking_ProcessModel</i>
URI	<i>URI_TravelBooking_FM</i>
Describes	<i>TravelBooking</i>
modelType	<i>BPMN</i>
Process00	
ID	0
name	<i>TravelBooking</i>
administration_Record	#
type	C
describedBy	<i>Process_Model00</i>
Restrictedby	<i>Sequence01</i>
tirggeredBy	<i>Event01</i>
Process01	
ID	1
name	<i>Check_Credit_Card</i>
administration_Record	#
Type	A
produces	<i>Event02/Event03</i>
Process02	
ID	2
name	<i>XOR01_Process</i>
administration_Record	#
type	C
restrictedby	<i>XOR01</i>
Process03	
ID	3
name	<i>Sequence02_Process</i>
administration_Record	#
type	C
restrictedby	<i>Sequence02</i>
XOR_Split01	
ConstructType	<i>XOR_Split</i>
splitedNodes	<i>Conditional_Process01</i> <i>Conditional_Process02</i>
Conditional_Process01	
condition	<i>Event01</i>
resultProcess	<i>Process03</i>
Conditional_Process02	
condition	<i>Event02</i>
resultProcess	<i>Process04</i>
Process04	
ID	5
name	<i>Handel_Default</i>
administration_Record	#
type	A
produces	<i>Event04</i>
Event01	
description	<i>Recieve</i>
ID	1
Event02	
Description	<i>Error</i>
ID	2
Event03	
description	<i>NoError</i>
ID	3
Event04	
description	<i>Reply</i>
ID	4

Figure A.3.1– General registration information of process “TravelBooking”

Figure A.3.2 shows the registration information of three branches if no error exists after checking credit card.

Sequence01	
ConstructType	Sequence
nodeSet	Process01 Process03
Sequence02	
ConstructType	Sequence
nodeSet	Process05 Process06 Process07
Process05	
ID	5
name	AND_Split01_Process
administration_Record	#
Type	C
RestrictedBy	AND_Split01
Process06	
ID	6
name	AND_Join01_Process
administration_Record	#
Type	C
RestrictedBy	AND_Join01
Process07	
ID	7
name	Confirmation
administration_Record	#
Type	A
produces	Event04

AND_Split01	
ConstructType	AND_Split
splitedNodes	Process08 Process09 Process10
AND_Join01	
ConstructType	AND_Join
JoinedNodes	Process08 Process09 Process10
Process08	
ID	8
name	Sequence03_Process
administration_Record	#
Type	C
RestrictedBy	Sequence03
Process09	
ID	9
name	Check_Hotel_Reservation
administration_Record	#
Type	A
Process10	
ID	10
name	Check_Flight_Reservation
administration_Record	#
Type	A

Figure A.3.2– Registration information of three branches if no error exists after checking credit card

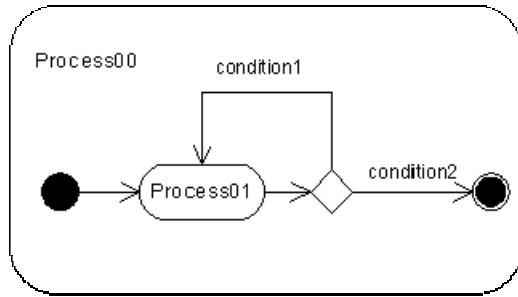
Figure A.3.3 shows the registration information of one branch about reserving cars as follows:

Sequence03		Process15	
ConstructType	<i>Sequence</i>	ID	<i>15</i>
nodeSet	<i>Process11 Process12</i>	name	<i>XOR_Join02_Process</i>
Process11		administration_Record	<i>#</i>
ID	<i>11</i>	Type	<i>C</i>
name	<i>Data_Map</i>	restrictedby	<i>XOR_Join02</i>
administration_Record	<i>#</i>	Sequence04	
Type	<i>A</i>	ConstructType	<i>Sequence</i>
Process12		nodeSet	<i>Process13 Process14 Process15</i>
ID	<i>12</i>	Sequence05	
name	<i>Sequence04_Process</i>	ConstructType	<i>Sequence</i>
administration_Record	<i>#</i>	nodeSet	<i>Process16 Process17</i>
Type	<i>C</i>	Process16	
restrictedby	<i>Sequence04</i>	ID	<i>16</i>
Process13		name	<i>Check_Car_Reservation</i>
ID	<i>13</i>	administration_Record	<i>#</i>
name	<i>Sequence05_Process</i>	Type	<i>A</i>
administration_Record	<i>#</i>	Process17	
Type	<i>C</i>	ID	<i>17</i>
restrictedby	<i>Sequence05</i>	name	<i>Evaluate_Reservation_Result</i>
Process14		administration_Record	<i>#</i>
ID	<i>14</i>	Type	<i>A</i>
name	<i>XOR_Split02_Process</i>	XOR_Split02	
administration_Record	<i>#</i>	ConstructType	<i>XOR_Split</i>
Type	<i>C</i>	splitNodes	<i>Conditional_Process03 Conditional_Process04</i>
restrictedby	<i>XOR_Split02</i>	Event05	
XOR_Join02		Description	<i>CheckAgain</i>
ConstructType	<i>XOR_Join</i>	ID	<i>5</i>
splitNodes	<i>Process12 N/A</i>	Event06	
Conditional_Process3		description	<i>Done</i>
condition	<i>Event05</i>	ID	<i>6</i>
resultProcess	<i>Process12</i>	Conditional_Process4	
Conditional_Process4		condition	<i>Event06</i>
condition	<i>Event06</i>	resultProcess	<i>N/A</i>
resultProcess	<i>N/A</i>		

Figure A.3.3– Registration information of one branch about reserving cars

NOTE: In case 3, there is a Loop Structure. PMR doesn't define one kind of Dependency_Construct to describe it, for the existing constructs can describe loop construct and perform well.

For example, Process00 is restricted by a loop construct as follows:



Then it can be translated into this:

Process00 has a sequence construct which consists of Process01 and another sequence construct containing an XOR_Split and XOR_Join construct. The splitNodes of XOR_Split are two branches. One of them is to execute Process00 again if condition1 is satisfied, and the other is to execute no process. Then the two branches are joined to execute latter processes.

Deleted: split

Deleted: ed

Figure A.3.4 shows the registration information of the loop process named *Process00*.

Process00		Process_Model00	
ID	0	name	Process00_ProcessModel
name	Process00	URI	URL_Process00_ProcessModel
administration_Record	#	Describes	Process00
type	C	modelType	UML
describedBy	Process_Model00: Process00_ProcessModel	Sequence00	
restrictedBy	Sequence00	constructType	Sequence
Process01		nodeSet	Process01:Process01 Process02:Sequence01_Process
ID	1	Sequence01	
name	Process01	constructType	Sequence
administration_Record	#	nodeSet	Process03:XOR_Split00_Process Process04:XOR_Join00_Process
type	A	Process04	
Process02		ID	4
ID	2	name	XOR_Join00_Process
name	Sequence01_Process	administration_Record	#
administration_Record	#	type	C
type	C	restrictedBy	XOR_Join00
restrictedBy	Sequence01	XOR_Split00	
Process03		ConstructType	XOR_Split
ID	3	splitedNodes	Conditional_Process00 Conditional_Process01
name	XOR_Split00_Process	XOR_Join00	
administration_Record	#	ConstructType	XOR_Join
type	C	splitedNodes	Process00 N/A
restrictedBy	XOR_Split00		

Conditional_Process00	
guardCondition	Event00
resultProcess	Process00

Event00	
description	condition1
ID	0
type	conditional

Conditional_Process01	
guardCondition	Event01
resultProcess	N/A

Event01	
description	condition2
ID	1
type	conditional

Figure A.3.4– Registration information of the loop process named Process00

Annex B
(informative)

Metaclasses inherited from MDR

Figure C.1 shows all metaclasses that inherit from Registered_Item and Administered_Item in MDR.

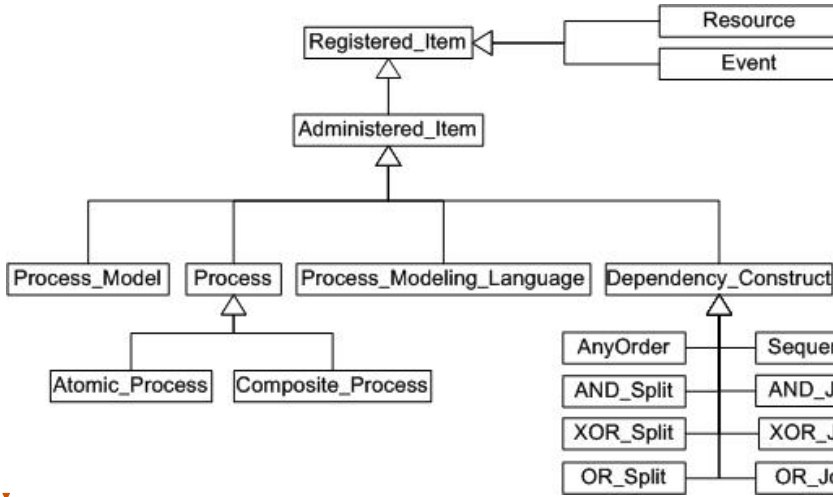


Figure C.1 –All metaclasses that inherit from Registered_Item and Administered_Item.

Deleted: C

Deleted:

Annex C

(informative)

Deleted: D

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 D.1.

Table C.1 – List of Process Modeling Languages

Deleted: D

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 (see bibliography item [4]).
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 [5])
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 [6])
DFD	DFD(Data Flow Diagram) is a graphical representation of the flow of data through an information system.
Other	

Mapping table of process modeling languages

If you are ready to register a process modeling language into the metamodel, you can map its metaclasses or attributes of metaclasses to MFI-5 according to Table D.2 .

Table C.2 – Mapping table of Process Modeling Languages

Deleted: D

PMR	BPMN	IDEF0	IDEF3	UML Activity Diagram	UML State Diagram	PSL	EPC
Process	Activity	Box	UOB	Activity	Action	Activity , Activity occurrence	Function
Resource		Input/ output	Object	Object node/ pin	Object	Object	
Attribute of Resource: state			Object State	State	State	State	
Event	Event			condition/ Signal/ Time	Event/ Condition		Event
Dependency_ Construct		Arrow	Links	Control flow	Transition		
AnyOrder							
Sequence	Sequenc e Flow		Simple precedenc e Link				
AND_Split	Fork	Forking Arrow	AND (Junction)	Fork	Fork		AND_Split
AND_Join	Join	Joining Arrow	AND (Junction)	Join	Join		AND_Join
OR_Split	Fork	Forking Arrow	OR (Junction)	Fork	Fork		OR_Split
OR_Join	Merging	Joining Arrow	OR (Junction)	Join	Join		AND_Join
X OR_split	Exclusive	Forking Arrow	XOR (Junction)	Fork	Fork		XOR_Split
X OR_Join	Exclusive	Joining Arrow	XOR (Junction)	Join	Join		XOR_Join
Condi tional	Guar dCon	gateway	Control Arrow	GuardCon dition	GuardCo ndition	Guard	logical connector/

Deleted: Condition_Result

<u>Proc</u> <u>ess</u>	dition							Event
	Result Process	Activity	Box	UOB	Activity	Action		Function

Annex D

(informative)

Code type lists

MFI PMR provides a generic and flexible facility to register process models described by different modeling languages. MFI PMR 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 E.1 shows the code set of state of a process. Any revision on the listed codes is allowed for specific MFI PMR 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		

Deleted: E

Deleted: E

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] ISO 18629-1:2004, Industrial automation systems and integration -- Process specification language -- Part 1: Overview and basic principles[5] IDEF0 Integration Definition for Function Modeling, 1993-12-31. Available at: <http://www.idef.com/pdf/idef0.pdf>.
- [6] IDEF3 Process Description Capture Method Report, September 1995. Available at: http://www.idef.com/pdf/idef3_fn.pdf.
- [7] Martin Fowler, UML Distilled: A Brief Guide to the Standard Object Modeling Language, Third Edition, 0-321-19368-7, Addison Wesley, 2003-9-15, Chapter 11. Available at: <http://www.ebookee.net/ebooks-UML-Distilled-A-Brief-Guide-to-the-Standard-Object-Modeling-Language-3rd-Edition-d/>