

ISO/IEC JTC1/SC32 Nxxxx

Date: 2010-12-24

ISO/IEC CD2 19763-5

ISO/IEC JTC1/SC32/WG2 **N1493**

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: (24) 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
3 Normative references	2
4 Terms, definitions and abbreviated terms	2
4.1 Terms and definitions	2
4.2 Abbreviated terms	4
5 Structure of MFI PMR	4
5.1 Overview of MFI PMR	4
5.2 Relationship between MFI PMR and other parts in MFI	6
5.3 MFI PMR	7
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 Resource.....	10
5.3.7 Event	10
5.3.8 Linear_Process	11
5.3.9 Split_Process	11
5.3.10 Join_Process	12
5.3.11 Split_Join_Process	12
5.3.12 Loop_Process	13
Annex A (informative) Examples of MFI PMR registration	14
Annex B (informative) Metaclasses inherited form MDR	21
Annex C (informative) List of process modelling languages	22

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 process“HandleOrder”	15
Figure A.1.2 – Registration information of sub-process “DeliverOrder”	16
Figure A.2.1– Registration information of process “TravelBooking”	17
Figure A.2.2– Registration information of three branches in TravelBooking	18
Figure A.2.3– Registration information of Loop_Process	19
Figure A.3.1– Registration information of process “Book Borrowing”(1).....	20
Figure A.3.2– Registration information of process “Book Borrowing”(2).....	20
Table C.1 – List of Process_Modeling_Languages.....	22
Table C.2 – Mappings among MFI MPR and other process modeling specifications	23

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. Especially for on-demand service selection and composition, business process is widely used to represent the execution order within a service or orchestration of a set of services. Then, how to discover and reuse existing process models 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. However, the differences in the syntax and semantic of process models will hamper sharing and reusing them among enterprises. Therefore, it is necessary to provide a generic mechanism to support registration of administrative information of process models, standardize the presentation formats of their functions, and improve discovery and reuse of them.

This part of ISO/IEC 19763 intends to provide a metamodel to register administrative information of process models, especially the function and decomposition of a process. Any information related to the details of process modeling languages or the platform for process execution is not taken into account. In this way, semantic discovery and reuse of both process and services can be supported through registration of heterogeneous 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 semantic 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 Business Process Modeling Notation (BPMN), UML(Unified Modeling Language) Activity Diagram, and EPC(Event-driven Process Chain), etc. In that case, the metamodel can help discover function and composition 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.

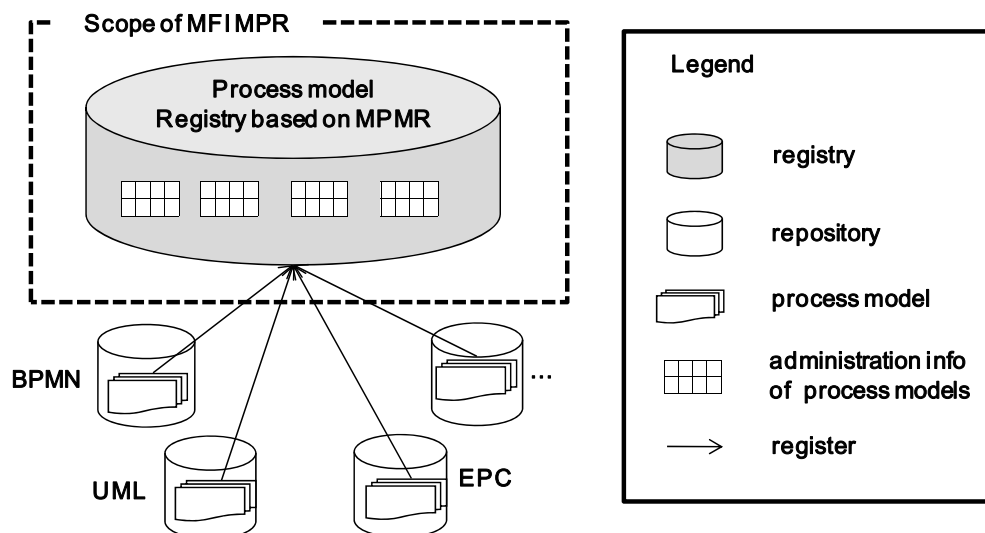


Figure 1 – The scope of MFIMPR

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.

2.1.2 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.

2.1.3 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 19763-3:2007, ISO/IEC 11179-3:2003 and ISO/IEC

19501:2005 shall apply to this part of ISO/IEC 19763.

4.1.1

Process

system of activities, which use resources to transform inputs into outputs.

[ISO/IEC 25000:2005 4.441]

NOTE Activity is defined as a set of cohesive tasks of a process. [ISO/IEC 12207:2008 4.3]

4.1.2

Process model

specification that is the result of modeling one or more process, adopting a specific process modeling language to describe features of a 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, UML Activity Diagram 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 in the same level of granularity.

4.1.6

Composite process

process that consists of other processes.

4.1.7

Resource

asset that is utilized or consumed during the execution of a process.

[ISO/IEC 12207:2008 4.37]

NOTE Resource can be either physical or virtual things.

4.1.8

Event

half-way between real events, as occurrences with no “duration”, and states, as steady conditions of the process or of some entities manipulated by it.

[Bibliography 7]

NOTE The event can be certain or uncertain. It also can be a single occurrence or a series of occurrences.

4.1.9

Condition process

branch process containing a guard condition and the corresponding process to be executed if the guard condition is satisfied.

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 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

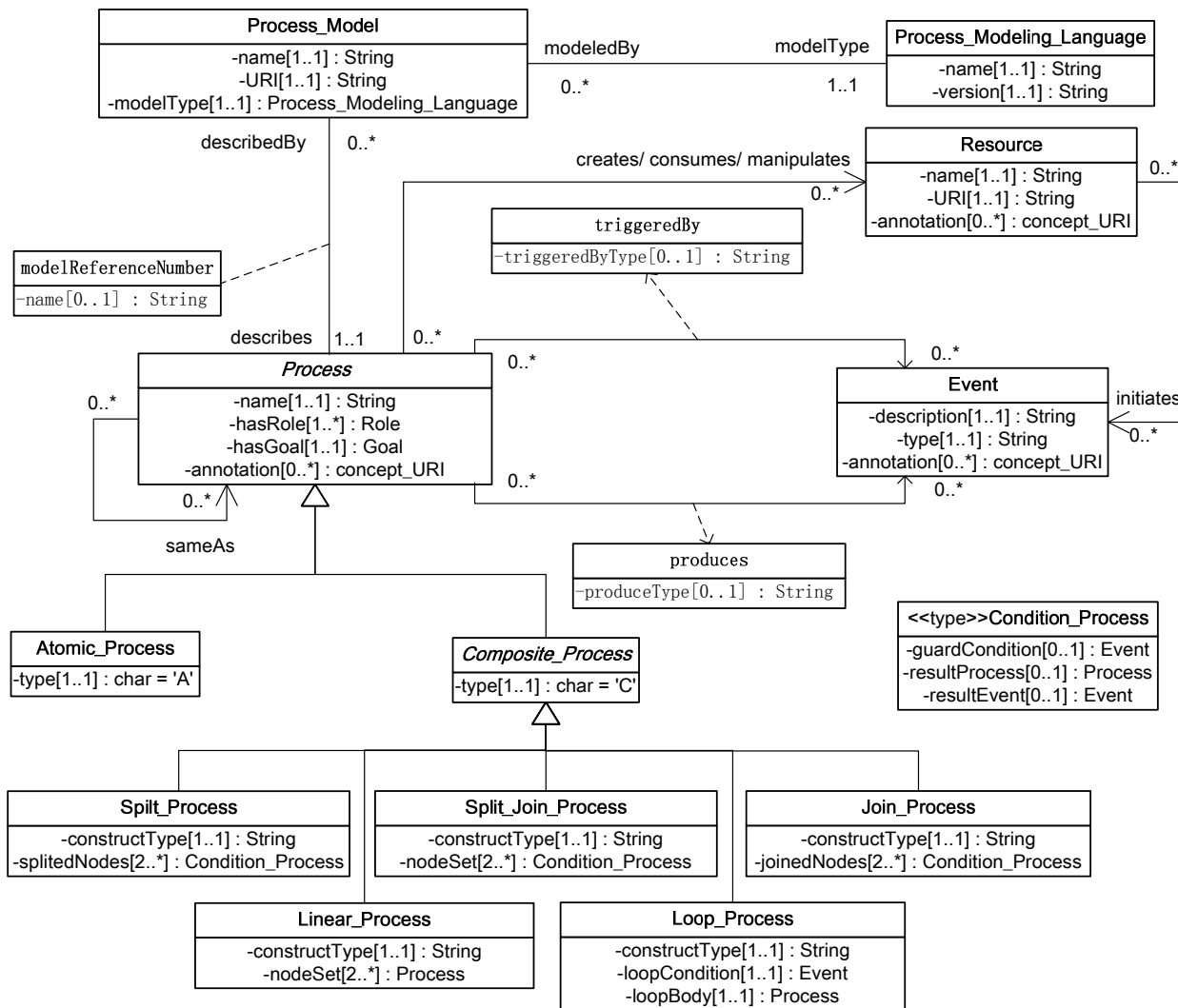
5.1 Overview of MFI PMR

MFI PMR provides a generic metamodel 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**. **Process_Modeling_Language** specifies the modeling language that **Process_Model** uses to represent processes. **Process** can be triggered by **Event**, which is expressed as half-way between real events, as occurrences with no “duration”, and states, as steady conditions of the process or of some entities manipulated by it. In that case, an event may act as the postcondition or precondition of a process. The association class “triggeredBy” from **Process** to **Event** means that a process may be triggered by a series of events, and the connector among them is specified by the attribute “triggeredByType” of “triggeredBy”. Similarly, the association class “produces” from **Process** to **Event** means that a process may produces a series of events, and the corresponding connector among them is specified by the attribute “produceType” of “produces”. The value of both “triggeredByType” and “produceType” can be “AND”, “OR” or “XOR”.

Atomic_Process and **Composite_Process** are two kinds of **Process**. **Atomic_Process** has no sub-process, while **Composie_Process** consists of sub-processes that can be either atomic or composite. In MFI PMR,

Composite_Process can be generalized as five kinds of processes. They are **Linear_Process**, **Split_Process**, **Split_Join_Process**, **Loop_Process** and **Join_Process**. **Linear_Process** means that its sub-processes will be executed in sequence or in any order. **Split_Process** means that when the precedent process is completed, one or more processes in a given collection will be executed. **Join_Process** designates that the successor process will start till the processes in a given collection are completed. **Split_Join_Process** means that the processes involved in a given collection should execute in parallel after their common precedent process is completed and before their successor starts. **Loop_Process** means that the involved processes will be executed circularly when the corresponding loop condition is satisfied. **Condition_Process** is used as a composite data type, which is used to express the split node of **Split_Process**, **Join_Process** and **Split_Join_Process** as a pair of a guard condition and a process or event. Especially, the corresponding process or event will be triggered only if its guard condition is satisfied.



NOTE Metaclasses whose names are italicized are abstract metaclasses

Figure 2 – The metamodel for process model registration

The association “sameAs” from **Process** to **Process** specifies that the two processes can be the same even they are respectively described by two process models in two kinds of modeling languages. This kind of association can also work in the case that as a sub-process of a process at upper level, an atomic process can be further decomposed by another composite process at lower level. In that case, the atomic process at upper level is same as the composite one in lower level.

Resource represents any virtual or physical things participating in **Process** to help its performance and achieve the given purpose of **Process**. In general, a process may consume some resources, create other resources or just manipulates some resources. A resource in a specific state or in existence may initiate an event, which then may trigger specified processes.

5.2 Relationship between MFI PMR and other parts in MFI

Figure 3 shows the relationship between MFI PMR and other parts in MFI.

Process can achieve goals, which are instances of **Goal** in MFI Role&Goal. The relationship between MFI PMR and MFI Role&Goal means that on one hand, **Goal** can be achieved by zero to many instances of **Process** and **Process** can achieve only one instance of **Goal**. Meanwhile, **Role** can involve zero to many instances of **Process**, and **Process** can be involved by one to many instances of actors playing specific **Roles**. Similarly, **Service** in MFI Service can be used to perform **Process**. The relationship between MFI PMR and MFI Service means that **Process** can be performed by zero to many instances of **Service** and **Service** can perform only one instance of **Process**.

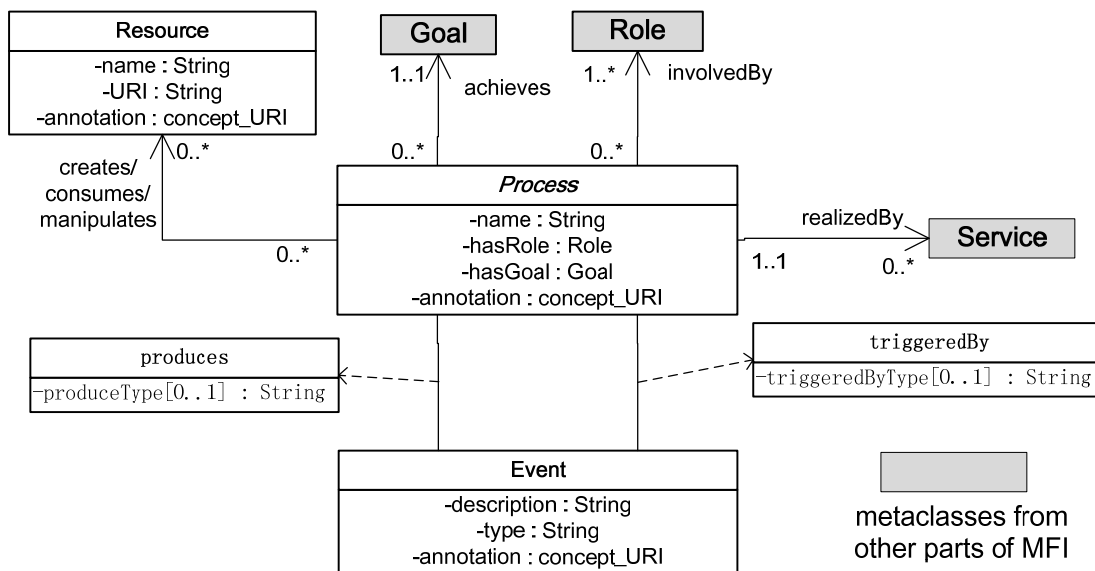


Figure 3 – Relationship between MFI PMR and some parts in MFI

The attribute “annotation” of **Process**, **Event** and **Resource** can be declared as the URI of the registered **Ontology_Atomic_Construct** based on MFI Ontology Registration. It means that the concepts in an ontology can be used to annotate a process, the resource participating in a process, or the event initiated by a resource.

5.3 MFI PMR

The MFI PMR is shown as 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.

(1) Superclasses

Administered_Item(from MDR)

(2) Attributes

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

-Description: Name of a process.

2. *hasGoal: Goal[1..1]*

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

3. *hasRole: Role[1..*]*

-Description: The role that involves in a process.

4. *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 a process.

(3) References

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

-Description: The process model as the specification of a process.

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

-Description: the process whose function is same to another process described by different process models or in different levels of granularity.

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

-Description: A number allocating to a process to identify it in a specific process model.

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

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

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

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

6. manipulates: *Resource[0..*]*

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

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

-Description: The events that a process produces. The relations among events can be specified by the attribute “*produceType*”, which can be “AND”, “OR” or “XOR”. In some cases, it may act as the condition that should be satisfied after the execution of the process.

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

-Description: The event that triggers a process. The relations among events can be specified by the attribute “*triggeredType*”, which can be “AND”, “OR” or “XOR”. In some cases, it may act as the condition that should be satisfied before the execution of the process.

(4) Constraints

The value of attribute “name” 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.

(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[0..*]*

-Description: The process described by a process model.

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

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

(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 to model a process. Its value can be one of the values in column “name” of Table C.1 in Annex C.

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 using 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 an abstract metaclass designating the process that contains other sub-processes, which might be either atomic or composite. It is the superclass of Linear_Process, Join_Process, Split_Process, Split_Join_Process and Loop_Process because only kind of control construct is allowed in this part to connect the involved sub-processes.

(1) Superclasses

Process

(2) Attributes

1. **type:** ‘C’

-Description: The registered process is a composite process.

(3) References

None.

(4) Constraints

Exists at least one process in this Composite_Processes.

5.3.5 Atomic_Process

Atomic_Process is a metaclass designating the process that has no sub-process in the same level of granularity.

(1) Superclasses

Process

(2) Attributes

1. **type:** ‘A’

-Description: The registered process is an atomic process.

(3) References

None.

(4) Constraints

None.

5.3.6 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]*

-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.

(3) References

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

-Description: The process that consumes the resource.

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

-Description: The process that creates the resource.

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

-Description: The process that manipulates the resource.

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

-Description: The event initiated by the resource in a specific state or in existence.

(4) Constraints

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

5.3.7 Event

Event is a metaclass designating the occurrences with no "duration" or the steady conditions of the process.

(1) Superclasses

Administered_Item(from MDR)

(2) Attributes

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

-Description: The description addressing the occurrence or the conditions of a process.

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

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

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 the event produced by a process or triggering a process.

(3) References

None.

(4) Constraints

None.

5.3.8 Linear_Process

Linear_Process is a metaclass designating a composite process whose sub-processes are allowed to execute linearly.

(1) Superclasses

Composite_Process

(2) Attributes**1. constructType:** *String[1..1]*

-Description: Type of a composite process, specifying the execution logic of its sub-processes. Its value could be “Sequence” or “AnyOrder”.

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

-Description: Sub processes connected by Sequence or AnyOrder.

(3) References

None.

(4) Constraints

None.

5.3.9 Split_Process

Split_Process is a metaclass designating a composite process whose sub-processes will execute in parallel when the precedent process is completed.

(1) Superclasses

Composite_Process

(2) Attributes**1. constructType:** *String[1..1]*

-Description: Type of a composite process, specifying the execution logic of its sub-processes. Its value could be “AND_Split”, “XOR_Split” or “OR_Split”.

2. splitNodes: *Condition_Process[2..*]*

-Description: The condition processes to be executed in parallel when its precedent process is completed.

(3) References

None.

(4) Constraints

None.

5.3.10 Join_Process

Join_Process is a metaclass designating a composite process whose sub-processes should be completed before successor process starts.

(1) Superclasses

Composite_Process

(2) Attributes

1. **constructType:** *String[1..1]*

-Description: Type of a composite process, specifying the execution logic of its sub-processes. Its value could be “AND_Join”, “XOR_Join” or “OR_Join”

2. **joinedNodes:** *Condition_Process[2..*]*

-Description: The condition processes to be executed in parallel before the successor starts.

(3) References

None.

(4) Constraints

None.

5.3.11 Split_Join_Process

Split_Join_Process is a metaclass designating a composite process whose sub-processes should be executed after its precedent is completed and before its successor starts.

(1) Superclasses

Composite_Process

(2) Attributes

1. **constructType:** *String[1..1]*

-Description: Type of a composite process, specifying the execution logic of its sub-processes. Its value could be “AND_Split_Join”, “XOR_Split_Join” or “OR_Split_Join”

2. **nodeSet:** *Condition_Process[2..*]*

-Description: The condition processes that have to be executed in parallel when the precedent process is completed and before the successor starts.

(3) References

None.

(4) Constraints

None.

5.3.12 Loop_Process

Loop_Process is a metaclass designating a composite process whose sub-processes will be executed circularly when the loop condition is satisfied.

(1) Superclasses

Composite_Process

(2) Attributes

1. **constructType:** *String[1..1]*

-Description: Type of a composite process, specifying the execution logic of its sub-processes. Its value could be "Condition_First_Loop" or "Process_First_Loop". "Condition_First_Loop" specifies that the processes in the loop body will execute till the loop condition is satisfied. "Process_First_Loop" means that the process in the loop body will execute before checking the loop condition is satisfied or not.

2. **loopCondition:** *Event[1..1]*

-Description: The event addressing the conditions that should be satisfied to execute some processes circularly.

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

-Description: The process to be executed circularly when the loop condition is satisfied.

(3) References

None

(4) Constraints

None

Annex A
(informative)

Examples of MFI PMR registration

In this section, three cases are provided to illustrate how to register various kinds of process models based on MFI PMR

In detail, three kinds of process models described with UML Activity Diagram, BPMN and EPC are registered respectively based on MFI PMR. The corresponding registration information is listed to show that MFI PMR can be used to register heterogeneous process models. It also means that MFI PMR can harmonize with existing specifications related to process modeling.

Case 1: HandleOrder process in UML Activity Diagram.

HandleOrder activity is expressed in UML to designate the process of how to deal with online order. As the following figure at the left side shows, *HandleOrder* consists of a sequence of sub-processes, involving *ReceiveOrder*, the processes describing handle details and *CloseOrder*. The handle details are represented by two split-join branches by *FillOrder*, *SendInvoice*, and their respective subsequent actions. Especially, one of its sub activities named *DeliverOrder* can be further decomposed, as the following figure at the right side.

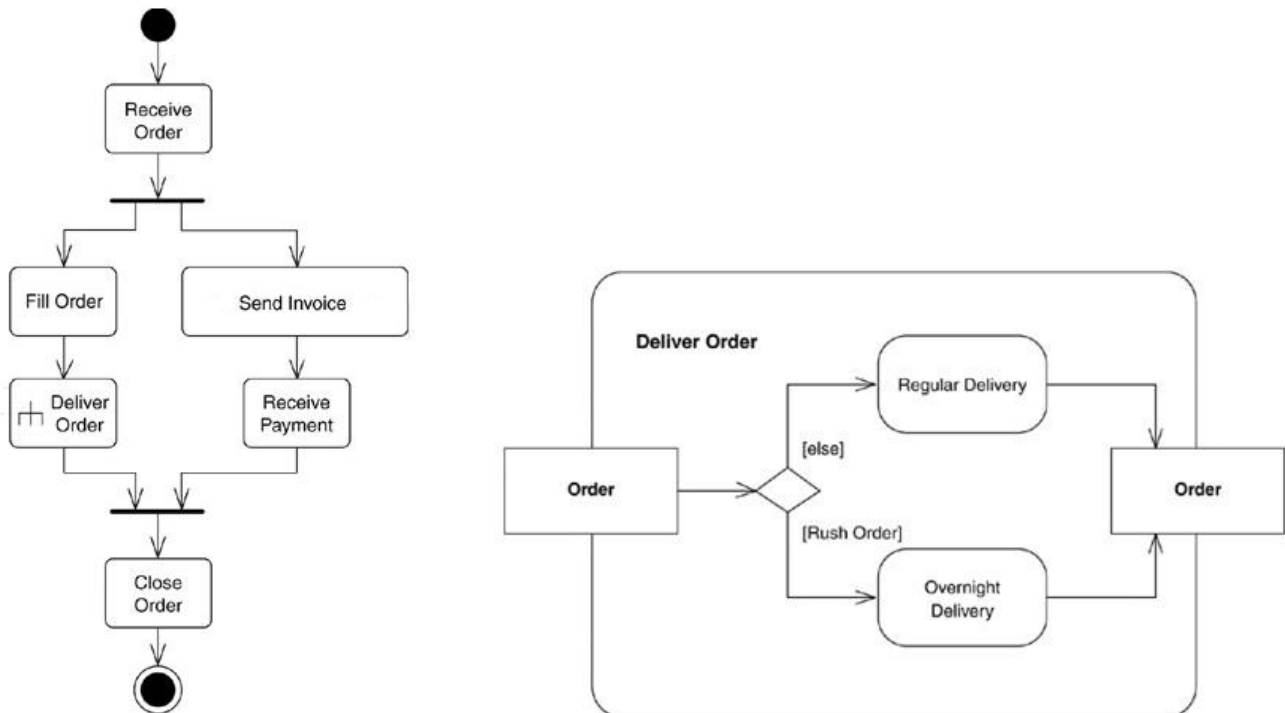


Figure A.1.1 shows the registration information of *HandleOrder* and its 4 sub-processes .

Process00	
name	<i>HandleOrder</i>
administration_Record	#
type	C
describedBy	<i>Process_Model00: HandleOrder_ProcessModel</i>
constructType	<i>Sequence</i>
nodeSet	<i>Process01: ReceiveOrder</i> <i>Process02: AND_Split_Join00</i> <i>Process03: CloseOrder</i>

Process_Model00	
name	<i>HandleOrder_ProcessModel</i>
URI	<i>URI_HandleOrder_ProcessModel</i>
Describes	<i>Process00::HandleOrder</i>
modelType	<i>UML</i>

Process01	
name	<i>ReceiveOrder</i>
administration_Record	#
type	A

Process02	
name	<i>AND_Split_Join00</i>
administration_Record	#
type	C
constructType	<i>AND_Split_Join</i>
nodeSet	<i>Process04: Sequence01</i> <i>Process05: Sequence02</i>

Process03	
name	<i>CloseOrder</i>
administration_Record	#
type	A

Process04	
name	<i>Sequence01</i>
administration_Record	#
type	C
constructType	<i>Sequence</i>
nodeSet	<i>Process06: FillOrder</i> <i>Process07: DeliverOrder</i>

Process05	
name	<i>Sequence02</i>
administration_Record	#
type	C
constructType	<i>Sequence</i>
nodeSet	<i>Process08: SendInvoice</i> <i>Process09: ReceivePayment</i>

Process06	
name	<i>FillOrder</i>
administration_Record	#
type	A

Process07	
name	<i>DeliverOrder</i>
administration_Record	#
type	A
sameAs	<i>Process10: DeliverOrder</i>

Process08	
name	<i>SendInvoice</i>
administration_Record	#
type	A

Process09	
name	<i>ReceivePayment</i>
administration_Record	#
type	A

Figure A.1.1 – Registration information of process “HandleOrder”

Figure A.1.2 shows the registration information of the sub-activity named *DeliverOrder*.

Process10	
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>
sameAs	<i>Process07:DeliverOrder</i>
constructType	<i>XOR_Split_Join</i>
nodeSet	<i>Conditional_Process00</i> <i>Conditional_Process01</i>

Conditional_Process00	
guardCondition	<i>Event01:RushOrder</i>
resultProcess	<i>Process11:RegularDelivery</i>

Conditional_Process01	
guardCondition	<i>Event02:NotRushOrder</i>
resultProcess	<i>Process12:OvernightDelivery</i>

Resource00	
name	<i>Order</i>
URI	<i>URI_Order</i>

Process_Model01	
name	<i>DeliverOrder_ProcessModel</i>
URI	<i>URI_DeliverOrder_ProcessModel</i>
Describes	<i>Process10:DeliverOrder</i>
modelType	UML

Event01	
description	<i>RushOrder</i>
type	<i>Internal_conditional</i>

Event02	
description	<i>NotRushOrder</i>
type	<i>Internal_conditional</i>

Process11	
name	<i>RegularDelivery</i>
administration_Record	#
type	A
manipulates	<i>Resource00:Order</i>

Process12	
name	<i>OvernightDelivery</i>
administration_Record	#
type	A
manipulates	<i>Resource00:Order</i>

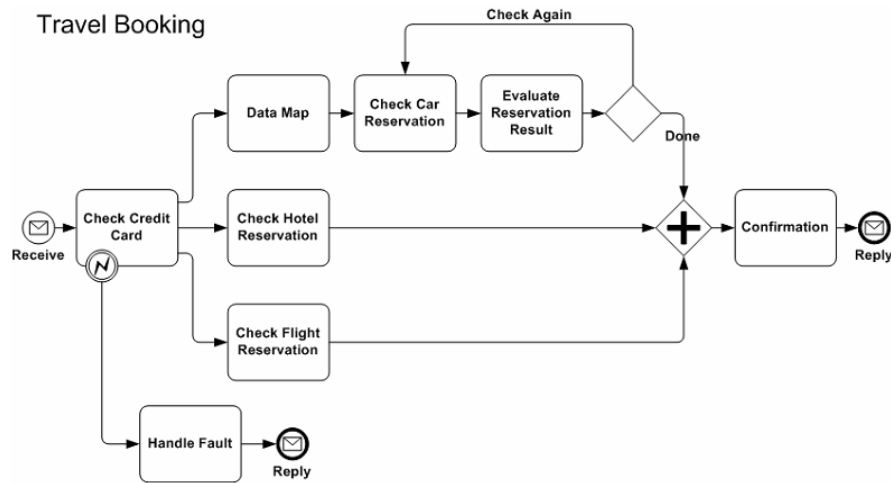
Figure A.1.2 – Registration information of sub-process “DeliverOrder”

Case 2: TravelBooking process in BPMN.

This case illustrates how to register process models expressed in BPMN.

The TravelBooking process begins with the receipt of a request for a travel booking. After checking the credit card, we can book a flight, a hotel, and a car. Particularly, the car reservation may execute several times till the process “Evaluate Reservation Result” meets the given requests. After all reservations are confirmed, a reply will be returned.

The following is the description of TravelBooking using BPMN.



Process_Model01	
Name	TravelBooking_ProcessModel_Level1
URI	URI_TravelBooking_FM_Level1
Describes	Process00
modelType	BPMN
Process00	
name	TravelBooking
administration_Record	#
Type	C
triggeredBy	Event01
constructType	Sequence
nodeSet	Process01 Process02
Process01	
name	Check_Credit_Card
administration_Record	#
Type	A
produceType	XOR
produces	Event02 Event03
Process02	
name	Split01_Process
administration_Record	#
type	C
constructType	XOR_Split
splitedNoedes	Conditional_Process01 Conditional_Process02
Event04	
Description	Reply
Type	External

Conditional_Process01	
condition	Event01
resultProcess	Process03
Conditional_Process02	
condition	Event02
resultProcess	Process04
Process03	
name	Linearity01_Process
administration_Record	#
Type	C
constructType	Sequence
nodeSet	Process05 Process14
Process04	
name	Handel_Default
administration_Record	#
type	A
produces	Event04
Event01	
description	Recieve
type	external
Event02	
Description	Error
type	internal_conditional
Event03	
description	NoError
type	internal_conditional

Figure A.2.1– Registration information of process “TravelBooking”

Figure A.2.2 shows the registration information of three branches if no error exists after checking credit card.

Process05	
name	Split_Join_Process
administration_Record	#
type	C
constructType	AND_Split_Join
nodeSet	Conditional_Process03 Conditional_Process04 Conditional_Process05

Process14	
name	Confirmation
administration_Record	#
Type	A
produces	Event04

Process06	
name	Linearity02_Process
administration_Record	#
Type	C
constructType	Sequence
nodeSet	Process09 Process10

Event05	
description	Nodone
type	external

Conditional_Process03	
guardCondition	
resultProcess	Process06

Conditional_Process04	
guardCondition	
resultProcess	Process07

Conditional_Process05	
guardCondition	
resultProcess	Process07

Process07	
name	Check_Hotel_Reservation
administration_Record	#
Type	A

Process08	
name	Check_Hotel_Reservation
administration_Record	#
Type	A

Process09	
name	Data_Map
administration_Record	#
Type	A

Figure A.2.2– Registration information of three branches in TravelBooking

Figure A.2.3 shows the registration information of a loop process as follows.

Process10	
Name	Check_Again
administration_Record	#
Type	C
constuctType	Process_First_Loop
loopCoditon	Event05
loopBody	Process11

Process11	
Name	Linearity03_Process
administration_Record	#
Type	C
constuctType	Sequence
nodeSet	Process12 Process13

Process12	
name	Check_Car_Reservation
administration_Record	#
Type	A

Process13	
name	Evaluate_Reservation_Result
administration_Record	#
Type	A

Figure A.2.3– Registration information of Loop_Process

Case 3: “Book Borrowing” process in EPC.

This case provides how to register the “*Book Borrowing*” process using EPC.

As the following figure shows, the process is triggered by an event “User needs a book”. Then we will search the library index for titles. If the book title does not exist, then go on searching; if yes, we will check the borrower eligibility and book availability. After that, the requested book can be provided to the borrower, followed by the update of library index.

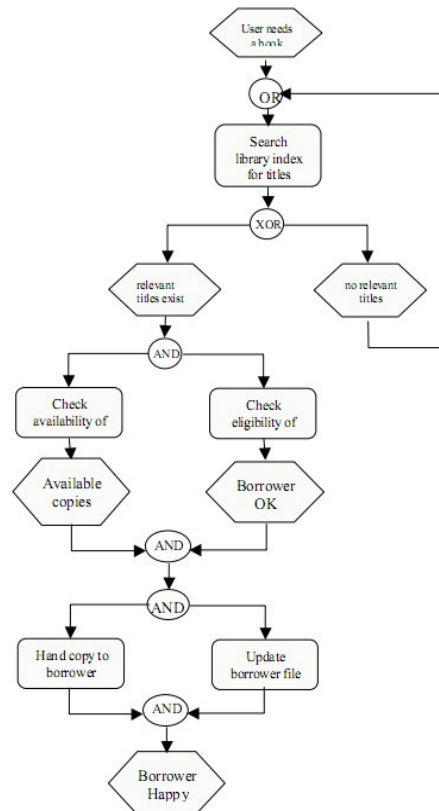


Figure A.3.1 shows the general registration information of process “Book Borrowing”

Process0	
name	Book Borrowing
administration_Record	#
type	C
describedBy	Process_Model0: Book Borrowing_ProcessModel
constructType	Sequence
nodeSet	Process1: Search library index for titles Process2: XOR_Split1
triggeredBy	Event1: User needs a book Event3: no relevant titles
triggeredByType	OR
Produces	Event6: Borrower Happy
Process1	
name	Search library index for titles
administration_Record	#
Type	A
Produces	Event2: relevant titles exist Event3: no relevant titles
produceType	XOR
Event 1	
Description	User needs a book
type	Internal_conditional

Process_Model0	
name	Book Borrowing_processModel
URI	URI_Book Borrowing_ProcessModel
Describes	Process0: Book Borrowing
modelType	EPC
Process2	
name	Process2: XOR_Split1
type	C
administration_Record	#
constructType	XOR_Split
nodeSet	Condition_Process1 Condition_Process2
Condition_Process1	
guardCondition	Event1: relevant titles exist
resultProcess	Process3: Sequence1
Condition_Process2	
guardCondition	Event2: no relevant titles
resultProcess	Process0
Event2	
Description	relevant titles exist
type	Internal_conditional

Figure A.3.1– Registration information of process “Book Borrowing”(1)

Figure A.3.2 shows the detail registration information of process “Book Borrowing” if book title exists in library.

Process3	
name	Sequence1
administration_Record	#
type	C
constructType	Sequence
nodeSet	Process4: And_Split_Join_1 Process5: And_Split_Join_2
Process4	
name	And_Split_Join_1
administration_Record	#
Type	C
constructType	And_Split_Join
nodeSet	Process6: Check availability of copies Process7: Check eligibility of borrower
Process5	
name	And_Split_Join_2
administration_Record	#
Type	C
constructType	And_Split_Join
nodeSet	Process8: Hand copy to borrower Process9: Update borrower file
triggeredBy	Event4: Available copies Event5: Borrower OK
triggeredByType	AND

Process6	
name	Check availability of copies
administration_Record	#
type	A
produces	Event4: Available copies
Process7	
name	Check eligibility of borrower
administration_Record	#
type	A
produces	Event5: Borrower OK
Process8	
name	Hand copy to borrower
administration_Record	#
type	A
Process9	
name	Update borrower file
administration_Record	#
type	A
Event4	
Description	Available copies
type	Internal_conditional
Event5	
Description	Borrower OK
type	Internal_conditional

Figure A.3.2– Registration information of process “Book Borrowing”(2)

Annex B

(informative)

Metaclasses inherited from MDR

Figure B.1 shows all metaclasses that inherit from Administered_Item in MDR.

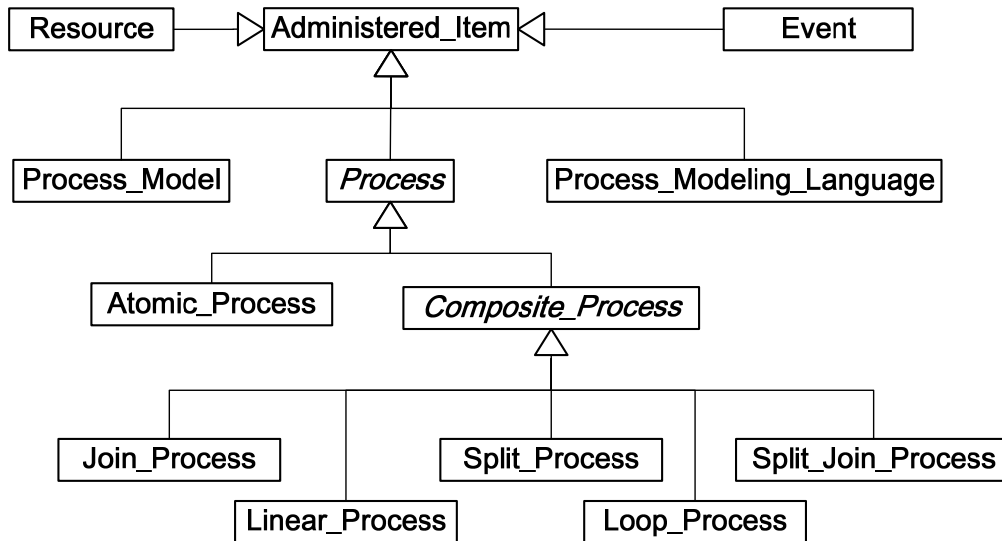


Figure B.1 –All metaclasses that inherit from Administered_Item.

Annex C

(informative)

List of process modelling 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 C.1.

Table C.1 – List of Process_Modeling_Languages

Name	Description
BPMN	Business Process Modeling Notation, Object Management Group, 2008. (see bibliography item [1])
BPEL	Business Process Execution Language for Web Service (BPEL/BPEL4WS), 2003-05-03, Version 1.1. (see bibliography item [2])
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 [3]).
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.
EPC	Event-driven Process Chain(EPC) is a type of flowchart used for business process modelling. It was developed in the early nineties in a joint effort between researchers at the University of Saarland and SAP(see bibliography item [7]). It is used to describe business processes at the informal business level. (see bibliography item [8])
Other	

Table C.2 lists the mappings between elements in MFI PMR and the corresponding ones in other specifications. On one hand, those mappings illustrate how to map modeling constructs of a process to the registration facility based on MFI PMR. On the other hand, it can promote reusing and remodeling a business process by transformation.

Table C.2 – Mappings among MFI MPR and other process modeling specifications

PMR		BPMN	IDEF0	IDEF3	UML Activity Diagram	PSL	EPC
Process		Activity	Box	UOB	Activity	Activity , Activity occurrence	Function
Resource		N/A	Input/ output	Object	Object node/ pin	Object	N/A
Attribute of Resource: state		N/A	N/A	Object State	State	State	N/A
Event		Event	N/A	N/A	condition/ Signal/ Time	N/A	Event
Linear_Process		Flow	N/A	Simple preceden ce Link	Flow	N/A	N/A
Split_Process		Fork/ Exclusive	Forking Arrow	AND/OR/ XOR (Junction)	Fork	N/A	forking AND connector forking OR connector forking XOR connector
Join_Process		Join/ Merging/ Exclusive	Joining Arrow	AND/OR/ XOR (Junction)	Join	N/A	joining AND connector , joining OR connector joining XOR connector
Split_Join_Process		N/A	N/A	N/A	N/A	N/A	N/A
Loop_Process		N/A	N/A	N/A	N/A	N/A	N/A
Conditio nal_Pro cess	GuardCo ndition	gateway	Control Arrow	GuardCo ndition	GuardCon dition		Event
	ResultPr ocess	Activity	Box	UOB	Activity		Function

Bibliography

- [1] 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>.
- [2] Business Process Execution Language for Web Services(BPEL 1.1), 2003-5-5. Available at: <http://xml.coverpages.org/BPELv11-May052003Final.pdf>.
- [3] ISO 18629-1:2004, Industrial automation systems and integration -- Process specification language -- Part 1: Overview and basic principles
- [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.
- [6] Unified Modeling Language(UML 1.4), OMG Document Number: formal/2001-09-67, 2001. Available at: <http://www.omg.org/spec/UML/1.4/PDF>.
- [7] Keller, G; Nüttgens, M, Scheer, A.-W, "Semantische Prozeßmodellierung auf der Grundlage, publication of the Institut für Wirtschaftsinformatik, paper 89, Saarbrücken, 1992.
- [8] Hafedh Mili, Guitta Bou Jaoude, Éric Lefebvre, et al., "Business Process Modeling Languages: Sorting Through the Alphabet Soup", ACM Computing Surveys, Springer, Volume 43 Issue 1, 2010.