

OMG Business Process Specification MetaModel (BPDM)

(Revised submission)

June 5, 2006

. Submitted by:

- *Axway*
- *Borland*
- *Data Access*
- *EDS*
- *Lombardi Software*
- *MEGA International*
- *Unisys*

Supported by:

- *U.S. National Institute of Standards and Technology (NIST)*

OMG document: bmi/2006-06-02

Copyright 2006

Adaptive
Axway Software
Borland
Data Access
EDS
Lombardi Software
MEGA International
Unisys

The OMG mark, 'UML,' is a trademark of
Object Management Group, Inc. (OMG).

Table of Contents

| | | |
|---|--|----|
| 1 | Introduction | 6 |
| | Submitting Companies..... | 6 |
| | Status of the Document..... | 6 |
| | Guide to the Submission..... | 6 |
| | Proof of Concept..... | 6 |
| | Submission Contacts..... | 6 |
| | Change History..... | 8 |
| 2 | Problem to be solved..... | 9 |
| | Analytic Decomposition and Synthetic Composition..... | 9 |
| | Internal processes versus collaborations..... | 10 |
| | Collaboration indirectly visible from within an Orchestration..... | 10 |
| | Flow..... | 11 |
| | Events..... | 11 |
| 3 | The Business Modeling Context..... | 12 |
| 4 | BPDM and UML 2.0..... | 13 |
| 5 | BPDM and BPMN..... | 14 |
| 6 | Design Principles..... | 15 |
| 7 | Specification..... | 16 |
| | Overview..... | 16 |
| | Composition Model..... | 18 |
| | Composition..... | 18 |
| | Selection..... | 18 |
| | Generalization..... | 18 |
| | Individual..... | 18 |
| | Part..... | 19 |
| | Part Connection..... | 19 |
| | Part Connection as Part..... | 19 |
| | Selection List..... | 19 |
| | Selector..... | 20 |
| | Typable Part..... | 20 |
| | Type..... | 20 |
| | TypedElement..... | 20 |
| | Course Model..... | 22 |
| | Course Model..... | 22 |
| | Abstract Step..... | 22 |
| | And Join..... | 22 |
| | And Split..... | 23 |
| | Course..... | 23 |
| | Or Join..... | 23 |
| | Process Protocol..... | 23 |
| | Step..... | 23 |
| | Step Path..... | 24 |
| | Succession..... | 24 |

| | |
|---------------------------------|----|
| Succession Condition..... | 24 |
| Succession Constraint..... | 24 |
| Xor Split | 25 |
| Process Interaction Modell..... | 25 |
| Process Step | 25 |
| Message Flows..... | 26 |
| Interaction | 26 |
| Emerging Behavior..... | 27 |
| Composite..... | 27 |
| Happening | 27 |
| Interaction | 27 |
| Interaction Connection | 28 |
| Interaction Flow..... | 28 |
| Interactive Flow Binding | 28 |
| Interactive Part..... | 29 |
| Message Flow | 29 |
| Participant | 29 |
| Process..... | 29 |
| Process Step | 30 |
| Processing Behavior | 30 |
| Protocol Participant | 30 |
| Notation samples | 31 |
| Process Activity Model | 33 |
| Core..... | 33 |
| Action Flows | 33 |
| ActionNodes | 34 |
| Control Nodes..... | 34 |
| Activity..... | 34 |
| Control Flow | 35 |
| Data Flow | 35 |
| Event Notification..... | 35 |
| Event Publication | 35 |
| Holder | 36 |
| InitialNode | 36 |
| Input Flow | 36 |
| Managed Process | 36 |
| MessageEvent Flow | 37 |
| MessageResult Flow | 37 |
| Monitor..... | 37 |
| Monitored Context..... | 37 |
| Output Flow | 37 |
| PerformerRole | 38 |
| Publisher..... | 38 |
| Rule | 38 |
| Simple Activity | 38 |
| Structured Activity | 38 |
| Sub Process Activity | 39 |
| Terminate | 39 |
| ValueSpecification..... | 39 |
| Notation samples | 39 |
| Class Hierarchies | 40 |
| Process Hierarchy | 40 |
| Flow Hierarchy..... | 41 |
| Participant hierarchie..... | 41 |
| Step Hierarchie..... | 42 |
| 8 Language Mappings..... | 43 |

| | | |
|----|---|----|
| | BPEL mapping | 43 |
| | W3C Choreography mapping | 43 |
| 9 | Responses to RFP Requirements | 44 |
| | 1.5 Mandatory requirements | 44 |
| | Required Metamodel..... | 44 |
| | Metamodel Compatibility..... | 44 |
| | MOF Compliance | 44 |
| | Procedural and rule-based flow of control..... | 44 |
| | Specification of Activity Performers | 44 |
| | Asynchronous and Concurrent Execution..... | 45 |
| | Specification of initiation and termination..... | 45 |
| | Choreography..... | 45 |
| | Audit Log Generation..... | 45 |
| | Distributed Execution | 46 |
| | Process Definition import and export | 46 |
| | Non-Normative Notation Mappings | 46 |
| | Compatible Versions of Existing Specifications | 46 |
| | Optional requirements | 46 |
| | Additional Non-Normative Mappings | 46 |
| | Additional Execution Constraints | 46 |
| | Discussion topics | 47 |
| | Relationship to existing UML metamodel | 47 |
| | Relationship to Related UML Profiles, Metamodels and Notations..... | 47 |
| | Mapping to Existing Business Process notations and UML Notation | 47 |
| | Resource Model | 47 |
| | Relationships with related OMG specification activities. | 47 |
| | Consistency checks | 47 |
| | Access Control..... | 47 |
| | Web services and collaboration support | 47 |
| 10 | Compliance Levels | 49 |
| | PIM Compliance | 49 |
| | Other? | 49 |
| | Appendix A: Glossary | 50 |

Submitting Companies

- Axway Software
- Borland Software
- Data Access
- EDS
- Lombardi Software
- MEGA International
- Unisys

Status of the Document

This document is the third revised submission.

Guide to the Submission

- Section 2 describes the problem to be solved
- Section 3 discusses the design rationale.
- Section 4 specifies the metamodel
- Section 5 describes mappings of the metamodel to example languages
- Section 6 responds to the specific RFP requirements
- Section 7 specifies compliance points

Proof of Concept

These specifications are based on understanding of the metamodels of existing business process modeling solutions and business process execution languages such as BPEL (Business Process Execution Language). Collaboration Languages such as BPSS have also been taken under consideration as well as business process notations such as BPMN.

Submission Contacts

Antoine Lonjon
MEGA International
email: antoine.lonjon@mega.com

Bernard Debauche
Axway
email: bdebauche@axway.com

Karl Frank
Borland Software
email: Karl.Frank@borland.com

Fred Cummins
EDS
email: fred.cummins@eds.com

Cory Casanave
Data Access
email: cbc@enterprisecomponent.com

Petko Chobantonov
Lombardi Software
email: Petko.Chobantonov@lombardisoftware.com

Vitaly Khusidman
Unisys
email: Vitaly.Khusidman@unisys.com

Ed Barkmeyer
NIST
email : edbark@nist.gov

Change History

| | |
|---------------|---|
| August 2003 | Initial submission |
| January 2004 | Revised submission <ul style="list-style-type: none">• Clarification of the business modeling context• Alignment with UML 2.0 information flow model• Clarification of the role based approach for process interaction. |
| March 2005 | Revised submission 2 <ul style="list-style-type: none">• Clarification of the business modeling context• Remove UML 2.0 alignment• Flow Model• Process Model |
| November 2005 | Revised submission 3 <ul style="list-style-type: none">• Introduction of the composition model• Introduction of the Process Behavior model• Introduction of the Community Process model |
| May 2006 | Initial Submission 2 : <ul style="list-style-type: none">• Introduction of the course model• Introduction of the happening model |

2 Problem to be solved

This introduction explains the core structure and motivation of the proposed Business Process Definition (BPDM) meta-model, sticking as far as possible to business language.

The first problem to be solved by any proposal for a BPDM is the disambiguation of the key phrase business process definition. Process definitions occur at different levels of detail and in different contexts in the analysis and design of business processes, activities, functions and workflows. Furthermore, different industries and consulting practices use different terms for what is, at a conceptual level, often the same. This is the familiar problem of alternative lexicalizations for the same concept. Consequently, the phrase business process is controversial and may not have the same meaning depending on the context. In defining the problem to be solved here, we first attend to this task of disambiguation.

There are different understandings of business processes for a number of reasons. One is that there are different kinds of business. A first distinction is the separation of business process concepts in the general sense, from process concepts specific to particular industries. For example, manufacturing is essential to many businesses, but in the context of this metamodel for business process definition, we restrict the scope to process behavior common to a broad range of industries, industries as different from one another as are the manufacturing, financial, healthcare, entertainment and petroleum industries.

There was once a fashion for time and motion studies specific to manual assembly-line work. Such studies are an example of process engineering that is outside the scope of BPDM. For another example showing where the boundary of our domain lies, we observe medical treatment protocols important to the healthcare business; these too are ruled out of scope for BPDM. If our metamodel can cover them, so much the better, but if it cannot, we won't care. In contrast with industry-specific examples of process, we find processes for internal management and for interaction with suppliers and customers, which involve the same concepts across all industries we have examined. These are in scope for BPDM.

A second distinction can be made between management processes and value chain processes.

Management processes are ongoing processes performed by a company to control its value chain processes, while value chain processes achieve distinct end-goals, which create value for their customers and profits for themselves. Ongoing control processes are in scope, but we shall begin with a study of value chain processes. Amongst these, one finds mainstream processes and support processes: support processes provide resources to mainstream processes. There are also different kinds of mainstream sub-processes: action-oriented processes (internal) and communication-oriented processes (collaborations among peers). These will be our focus. An explanation of what we mean by action-oriented and communication-oriented will come in Section 3 below. All these kinds of process may overlap with regard to a common core, but the common core is too thin to be of any practical use. We therefore focus on two different metamodels, one for each of the kinds of business process we consider most important.

- Action-oriented processes, also known as internal processes or orchestrations.
- Communication-oriented processes, also known as collaborations or choreographies.

The goal is two arrive at two concrete metamodels that can be related to one another.

Analytic Decomposition and Synthetic Composition

When modeling a business process, we may want to do hierarchical decomposition. The process is analyzed as a structure of sub-processes. Alternatively, we may be concerned to reuse one process **P1** in defining another process, **P2**. The metamodel should support both approaches.

We assume that analytic decomposition is familiar. SADT and IDEF/0 provide concrete examples of this approach to process modeling. We contrast that approach with *Synthetic Composition*. Synthetic composition is not just the inverse of analytic decomposition, because it is not hierarchical, and because it presupposes the availability of process definitions for reuse.

Frequently, processes **P1** and **P2** are defined by reference to what we consider to be one and the same action or activity, **A**. In this case, we distinguish the usage of **A** in process **P1** from its usage in process **P2**. This we consider to be different from the hierarchical decomposition of **P1** and **P2**, in that the decomposition of any process leads to elements owned by that process, considered as different from the elements of any other process, even though their names might be the same. In order to represent the definition of processes by this synthetic approach, we introduce the composition – usage pattern.

This will be explained in more detail later, but those familiar with UML 2 Actions, in particular the InvokeBehavior Action, will have a sense of what we are doing here. Another insight into the difference between the two approaches may be provided by considering the difference between recursive and iterative definitions of equivalent behaviors.

Internal processes versus collaborations

One of the biggest challenges of BPDM is to clarify the relationship between *internal processes*, *collaborations*, *choreographies*, and *orchestrations*, within the broader domain of *process behavior*.

We understand *orchestrated* processes as those defined within the control scope of a single business entity or organization. Here, the *control scope* of an entity is the range of all behavior over which that entity is considered to exercise control. The controlling entity can say: “my way or the highway”. The process definition can mandate the ways and means for doing everything that gets done. This is in contrast to situations where the process definition can at best say: send out requests for something be done in whatever way the recipient chooses. In our view, orchestrated processes are *action oriented*.

Actions perform transformations on resources. For example, a business which has not outsourced the transportation function is considered to include transportation within its control. In this case, any actions related to delivery of goods to customers are represented as performed by instruments of the controlling business. Performance of these actions change the location of the goods, and end with transferal of those goods to the custody of the customer. An orchestrated process is by definition considered to be internal to some real or hypothetical entity.

Definition: an *orchestrated* process (also known as an *internal* process or *orchestration*) is processing behavior conceived as being under the exclusive control of a single entity. This entity may be a real or hypothetical business, enterprise or organization at any level of size or complexity, provided it is considered to be unitary. Because of the assumption of a unitary process owner, the size and complexity of the controlling entity is irrelevant.

In contrast with orchestrations, we propose that there are also business processes that exceed the control any single entity. These we refer to as *collaborative* processes. These are communication oriented, not action oriented.

If business **B** has outsourced the transportation function, but must move around raw materials or deliver finished goods, it will have to collaborate with another business, **TB**, in performing its business processes. In this context, **TB** is an independent business entity, controlling its own internal processes, even doing its best to conceal from **B** how those internal processes are defined.

The two business entities are collaborating, and the collaboration proceeds through communications, whose effectiveness depends on an agreed language. The collaboration does not depend on the presence of any controlling overlord. Each party to the collaboration will be an external participant in relation to the other business entity’s internal process.

For example, business **B** may send a shipment order, and wait for **TB** to send back an acknowledgement. If there is no agreement on the language and delivery mechanisms of communication, the sending of some data, electronically, by telephone, or carrier pigeon, cannot result in an appropriate response. When these are agreed, then a flow of messages can proceed between the two collaborating business entities. To define such a process, one must define the meanings of the communications, the allowed sequences, all of which we sum up in the concept of a protocol.

It is in our view important that no one of the business entities has a privileged relationship to the protocol. This should make it clear why we characterize collaborative processes as *communication* oriented, which in a simple case comes down to the exchange of messages according to a protocol.

Definition: a *collaborative* process (also known as a peer-to-peer process or choreography) is a business process which proceeds thru communications between two or more independent parties. The definition of a collaborative process, termed a *protocol*, is a definition of the communications and communicator roles it allows. Stated otherwise, a collaborative process is one defined in terms of the interaction of peers thru mutually understood communication. Because mutual understanding is essential for communication to succeed, these processes may alternatively be described as involving *joint actions*. The parties to a collaborative process may be real or hypothetical businesses, enterprises, or organizations at any level of complexity, so long as they are considered to be external to one another and are ascribed the ability to understand the communications, which amounts to being able to follow an agreed protocol.

Collaboration indirectly visible from within an Orchestration

A *participant* is a business entity as seen by and from another business entity. In the example of outsourced shipping, business **B** sees business **TB** as a participant in **B**’s process, and business **TB** sees **B** as a participant in its process. The definition of **B**’s *internal* process may orchestrate the creation and dispatch of messages meant for **TB**, but cannot stipulate **B**’s response. The *creation and dispatch* of a message is an action, not a joint action.

In other words, when the owner of an orchestrated process is also a collaborator with peers, those collaborators may appear at orchestrated process boundaries as opaque entities, typically sources or sinks. These are called

participants, in the context of defining an orchestrated process. They indicate interaction with entities whose internal processes are unknown. The protocols for these interactions may or may not be documented in a choreography.

To return to our ongoing example, **B's** orchestrated process model will specify the dispatch of messages intended for **TB** and will define what to do upon receipt of messages from **TB**.

The protocol agreed by **B** and **TB** is not an orchestrated process. The protocol defines all allowed sequences of message flows between collaborators and their agreed semantics. If the language of communication is English or some other natural language, it will not be possible to document the protocol fully, for the simple reason that linguistic science has been unable to provide an adequate theory of natural language syntax and semantics.

Flow

We are used to concepts like “control flow” and “data flow” when modeling “business processes”. This paragraph aims at showing how to express the dependencies between the (behavioral) parts of a business process (behavioral composite).

When looking at the data and their relation with the process model, it is key to differentiate the data required for the process control flow and the data which are resources produced, consumed or transformed by process actions. It is also important to distinguish the data from their conveyors (a message may convey data : but the message and the data that it conveys are not the same “thing”).

Regarding action-oriented processes, the orchestration of actions may depend upon the usage of resources: for example, one action that consumes a resource expects another action that updates that resource to take place earlier. This is a first kind of dependency, through “resource usage”: two actions make use of the same resource, one uses it to transform it, the other one uses it to consume it for example.

On a “process boundary” (interaction of the process with other participants), and in collaboration protocols, the control flow is message flow oriented: we mean that the control flow is directly dependent upon the messages that are exchanged between participants. In the context of a process, message flows link actions (and possibly special actions referred to as control actions) and participants: a process action can send a message to a participant (in B process, an action sends a “shipment order” message flow to participant GTB), and a participant can send a message to a control action (in B process, participant GTB sends a “shipment order acknowledgement” to a control action). In the context of a collaboration protocol, messages flows are send and received by participants.

Now that we have these two ways of “synchronizing” business process parts, we can link them: a message flow can make use of resources. For example, the “shipment order” message flow from participant B or from an action within B process to participant GTB can convey the order related information, which is a resource used by the two business process parts.

The other kind of dependency between two business process parts is the control flow itself (for example, the sequencing of two process actions).

Events

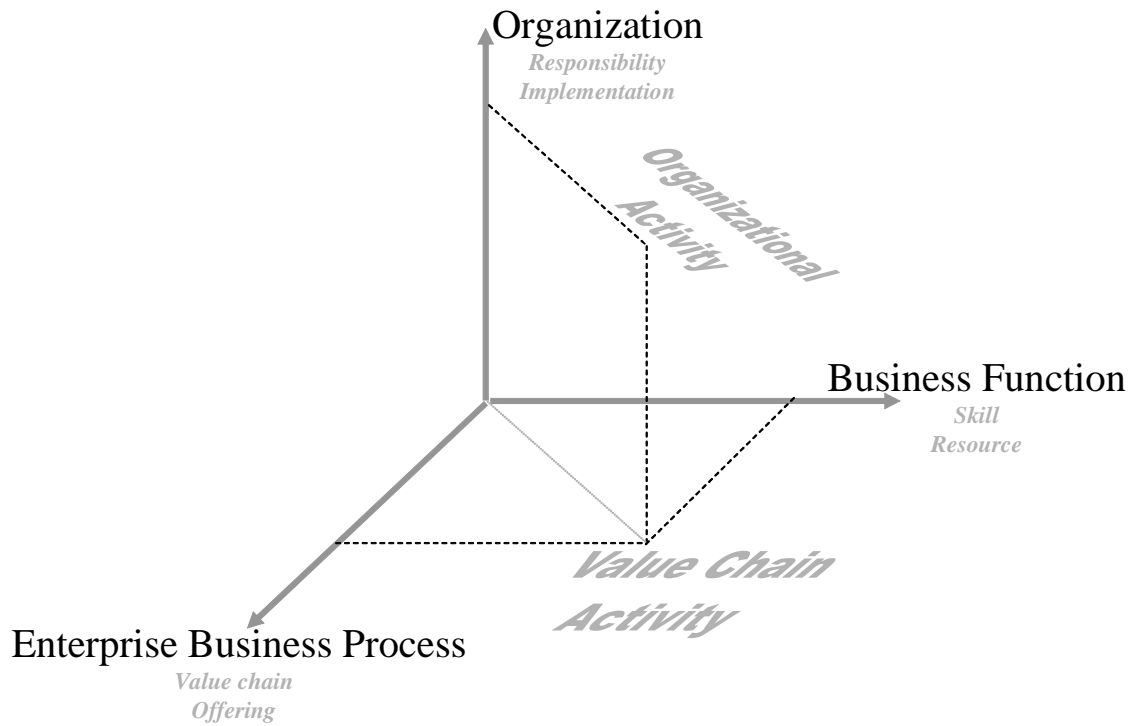
In BPMN, event is a “first class” concept: so should it be in BPDm. In BPMN, events are used as “control flow” elements: to capture a message flow in the process control flow, an “intermediate message event” is used; to interrupt a task or a process, an “intermediate event” of any kind is used. In BPMN, events are much like gateways: they are used when the control flow has to be influenced.

In BPDm, event flows are either input or output flows to process actions. Event flows can for example be the arrival of a message from another participant or the occurrence of a change in the business (such as the reaching of an inventory level threshold – this event generates a change event flow).

3 The Business Modeling Context

The Business Process Definition Metamodel is part of a more global business modeling approach. The BPDM proposal provides the framework to more specialized business models such as value chain models or organizational models.

The global business model must be able to articulate the different views of processes as shown in the following figure. This global picture is not in the scope of BPDM but will be covered by additional BMI Business Modeling RFPs among which the Organizational Structure Metamodel (OSM).



4 BPDM and UML 2.0

UML 2.0 provides a powerful set of modeling artifacts to build the backbone of a business process framework. UML 2.0 concepts are reuse wherever they match the corresponding BPDM business concepts. MOF based concepts are created where UML 2.0 equivalent concepts do not exist or do not fit to common business semantic. A very careful attention will be provided in order to avoid any duplication between BPDM and UML 2.0.

5 BPDM and BPMN

BPMN is intended to be the notation that comes along with BPDM. BPDM is compatible with the adopted BPMN 1.0 version and provides an extension to BPMN to fulfill the requirements of BPDM concepts. The version of BPMN aligned with BPDM is BPMN 2.0.

6 Design Principles

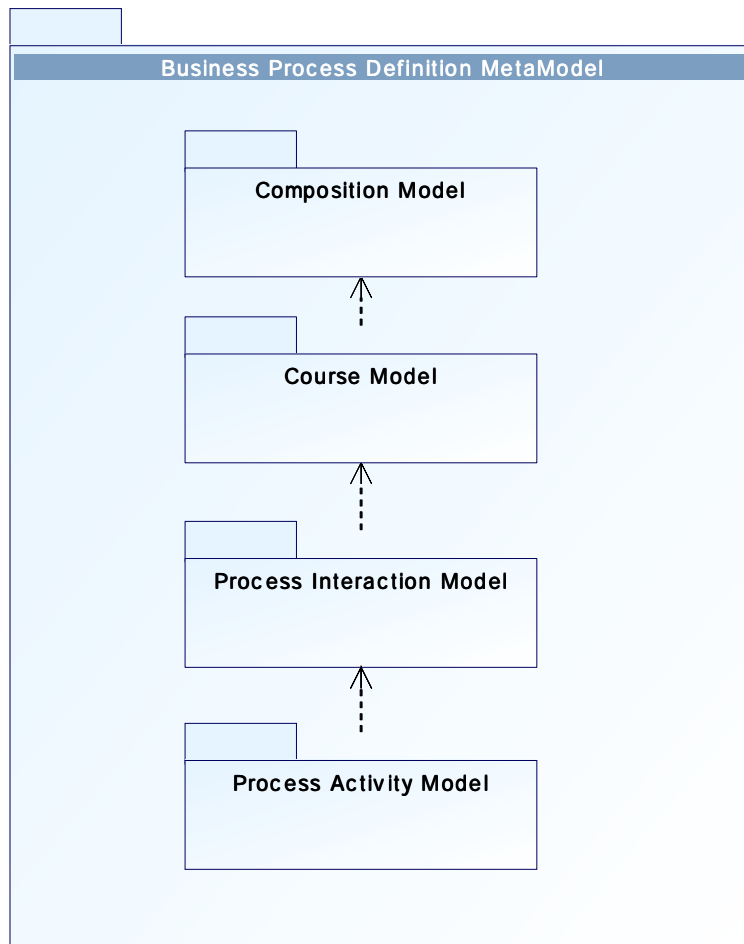
BPDM uses a flow centric approach to describe the structural architecture of processes. Flows are communication channels between units of behavior that act as a mechanism of coordination and are used as the basis for behavior encapsulation.

7 Specification

This section presents the normative specification for business process definition metamodel. It begins with an overview of the BPDM metamodel structure followed by a description of each sub-package.

Overview

BPDM holds all definitions common to process oriented models. It also establishes relationships between collaborative processes, also called choreographies and action based processes, also called orchestrations.

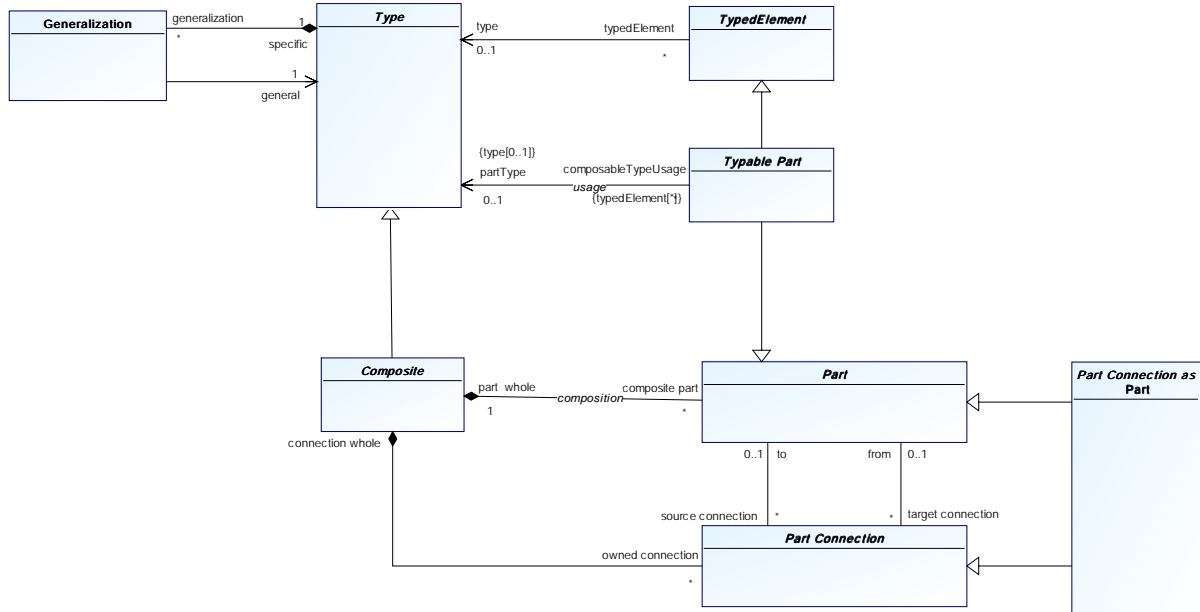


| Package | Comment |
|---------------------------------------|---|
| Business Process Definition MetaModel | BPDM holds all definitions common to process oriented models. It also establishes relationships between collaborative processes, also called choreographies and action based processes, also called orchestrations. |
| Composition Model | The composition model provides the principles and mechanisms for reuse and decomposition. It is based on the part/usage pattern where reusable element called Typable Parts are reused through intermediate objects called Parts. |
| Course Model | The Course Model describes common concepts used for all process oriented models. It describes the kinds of Parts that make a process object and its associated Part Connections. |
| Process Activity Model | The Process Activity Model focuses on transformation processes. |

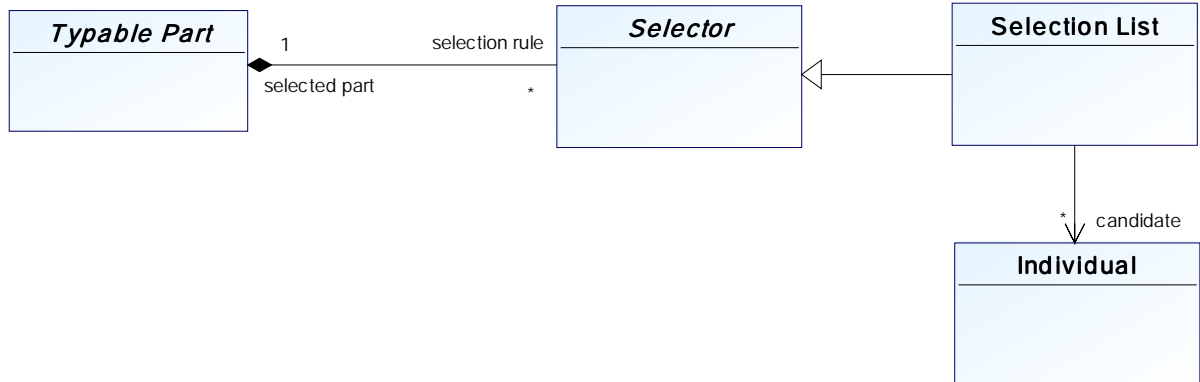
Composition Model

The composition model provides the principles and mechanisms for reuse and decomposition. It is based on the part/usage pattern where reusable element called **Typable Parts** are reused through intermediate objects called **Parts**.

Composition



Selection



Generalization

Namespace: Composition Model

isAbstract:

Description:

Associations

| | | |
|---------------------|--|--|
| general : Type [1] | | |
| specific : Type [1] | | |

Individual

Namespace: Composition Model

isAbstract:

Description:

Part

Namespace: Composition Model

isAbstract: Yes

Description:

A **Part** is an element of the structure of a **Composite**. A **Part** is typed by at most one **Typable Part**. The kinds of suitable **Typable Parts** depend on the nature of the **Part**. For instance, the type of an action in a process composite may be a process but can't be a partner. The type of a participant may be a partner but not a process. The range of possible **Typable Parts** for a **Part** may include more than one type. For instance, in a **Collaborative Process**, a **PerformerRole** may be typed either by a **Partner**, an **Processing Behavior** or even a **Collaborative Process**.

A **Part** without a **Typable Part** is a leaf **Part**.

A **Part** with a **Typable Part** is a complex **Part**.

Associations

| | | |
|---|--|--|
| part whole : Composite [1] | | |
| source connection : Part Connection [*] | | |
| target connection : Part Connection [*] | | |

Part Connection

Namespace: Composition Model

isAbstract: Yes

Description:

A **Part Connection** is used to connect two **Parts** of a **Composite**.

Part Connection is an abstract metaclass.

Associations

| | | |
|---------------------------------|--|--|
| connection whole : Composite [] | | |
| from : Part [0..1] | | |
| to : Part [0..1] | | |

Part Connection as Part

Namespace: Composition Model

isAbstract: Yes

Generalization: "Part Connection" "Part"

Description:

A **Part Connection as Part** is a **Part Connection** that is also a **Part**.

This means it can be typed and be connected with other kind of **Part Connection**

Selection List

Namespace: Composition Model

isAbstract:

Generalization: "Selector"

Description:**Associations**

| | | |
|----------------------------|--|--|
| candidate : Individual [*] | | |
|----------------------------|--|--|

*Selector***Namespace:** Composition Model**isAbstract:** Yes**Description:****Associations**

| | | |
|----------------------------------|--|--|
| selected part : Typable Part [1] | | |
|----------------------------------|--|--|

*Typable Part***Namespace:** Composition Model**isAbstract:** Yes**Generalization:** "Part" "TypedElement"**Description:**

A **Typable Part** is a type that is used for typing parts of a **Composite**. A **Typable Part** cannot be used directly as a part of a **Composite**. It must be "embedded" as a part in the context of a **Composite**.

Associations

| | | |
|-------------------------------|----------|--|
| partType : Type [0..1] | { type } | |
| selection rule : Selector [*] | | |

*Type***Namespace:** Composition Model**isAbstract:** Yes**Description:**

from UML 2.0

Associations

| | | |
|-------------------------------------|--|--|
| generalization : Generalization [*] | | |
|-------------------------------------|--|--|

*TypedElement***Namespace:** Composition Model**isAbstract:** Yes**Description:**

A typed element has a type.
from UML 2.0

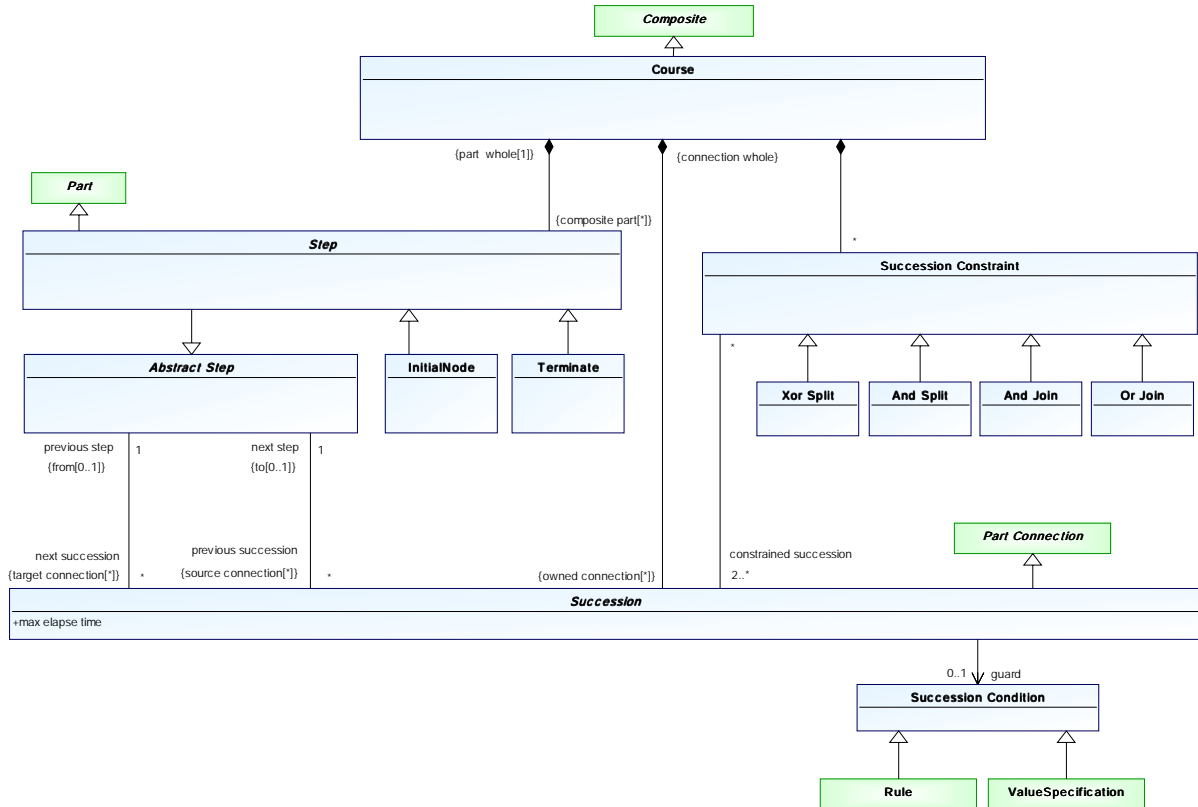
Associations

| | | |
|--------------------|--|--|
| type : Type [0..1] | | |
|--------------------|--|--|

Course Model

The **Course Model** describes common concepts used for all process oriented models. It describes the kinds of **Parts** that make a process object and its associated **Part Connections**.

Course Model



Abstract Step

Namespace: Course Model

isAbstract: Yes

Description:

Associations

| | | |
|--------------------------------------|---------------------|--|
| next succession : Succession [*] | {target connection} | |
| previous succession : Succession [*] | {source connection} | |

And Join

Namespace: Course Model

isAbstract:

Generalization: "Succession Constraint"

Description:

And Split

Namespace: Course Model

isAbstract:

Generalization: “Succession Constraint”

Description:

act of selecting a path from one place to another

Course

Namespace: Course Model

isAbstract:

Generalization: “Composite” “Monitored Context”

Description:

A **Course** is an ordered **Succession** of **Steps**

Associations

| | | |
|---|----------------------|--|
| Step : Step [] | { composite part } | |
| Succession Constraint : Succession Constraint [*] | | |
| Succession : Succession [] | { owned connection } | |

Or Join

Namespace: Course Model

isAbstract:

Generalization: “Succession Constraint”

Description:

Process Protocol

Namespace: Course Model

isAbstract:

Generalization: “Processing Behavior”

Description:

A **Process Protocol** is a concrete **Processing Behavior** that defines a set of prescribed **Interaction**

Step

Namespace: Course Model

isAbstract: Yes

Generalization: “Abstract Step” “Part”

Description:

A **Step** is a stage or interval in a development or **Course**

Associations

| | | |
|--------------------------|--------------|--|
| Course : Course [] | {part whole} | |
| Step Path : Step Path [] | | |

Step Path

Namespace: Course Model

isAbstract:

Generalization: “Abstract Step”

Description:

Associations

| | | |
|---------------------------------|--|--|
| path context : Step [1] | | |
| pathed step : Abstract Step [1] | | |

Succession

Namespace: Course Model

isAbstract: Yes

Generalization: “Part Connection”

Description:

A **Succession** is **Part Connection** that organizes **Steps** in series in the context of a **Course**

Merriam-Webster:

the act or process of following in order of time or place : a repeated following up of one by another

Associations

| | | |
|---|--------------------|--|
| Course : Course [] | {connection whole} | |
| next step : Abstract Step [1] | {to} | |
| previous step : Abstract Step [1] | {from} | |
| Succession Constraint : Succession Constraint [*] | | |

Succession Condition

Namespace: Course Model

isAbstract:

Description:

Succession Constraint

Namespace: Course Model

isAbstract:

Description:

Associations

| | | |
|--|--|--|
| constrained succession : Succession [2..*] | | |
| Course : Course [] | | |

Xor Split

Namespace: Course Model

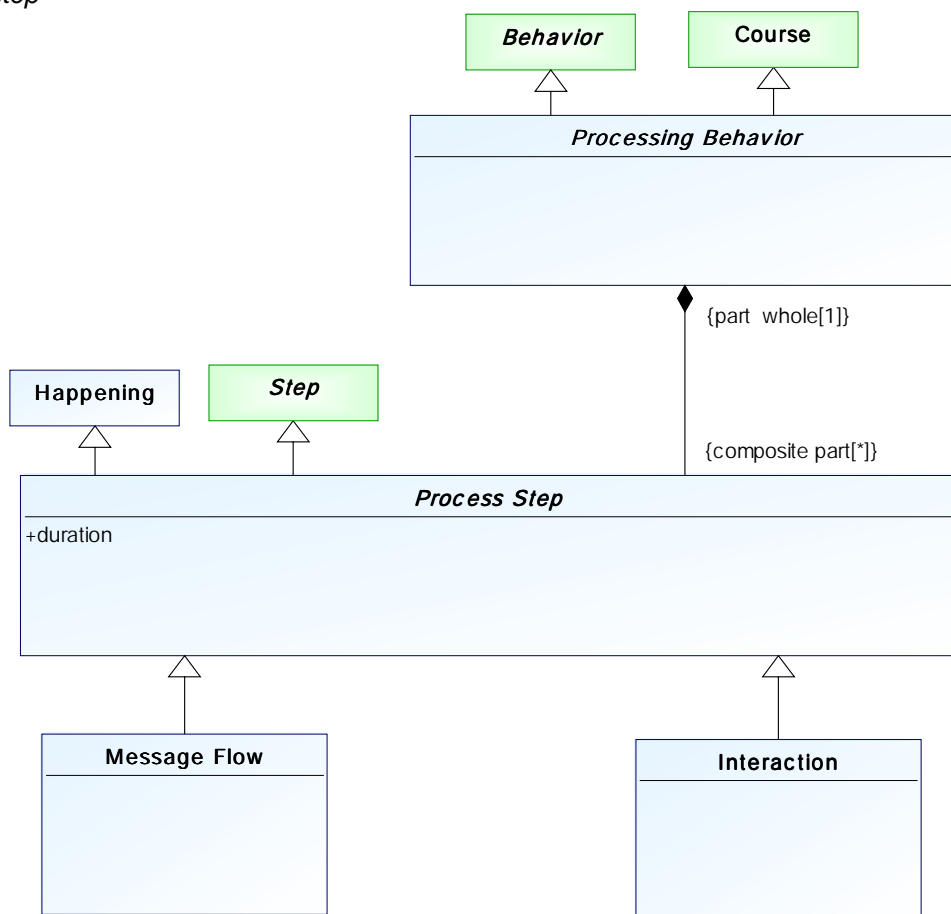
isAbstract:

Generalization: "Succession Constraint"

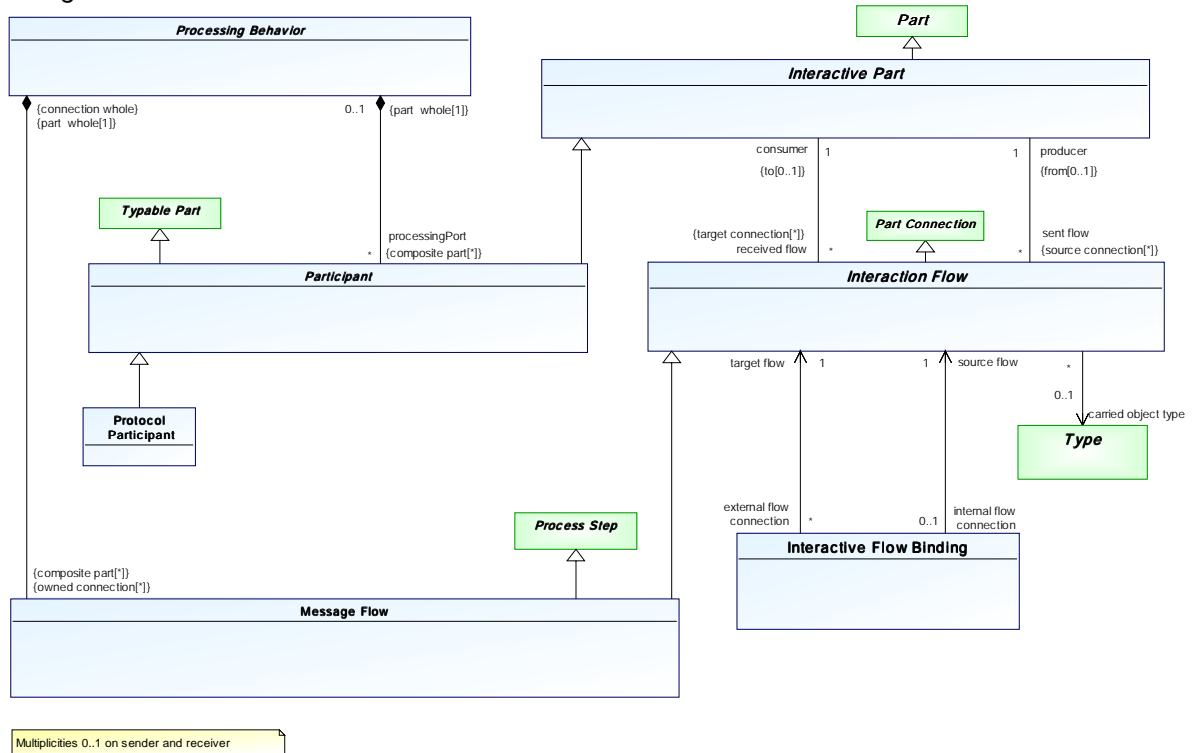
Description:

Process Interaction Modell

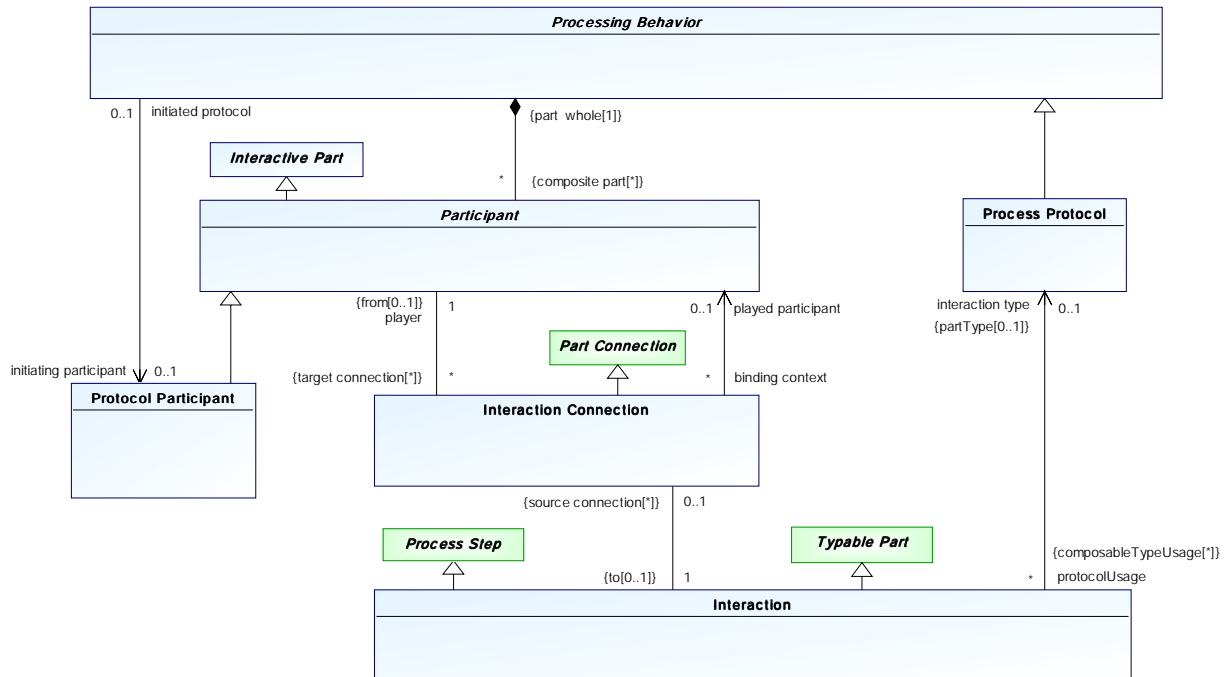
Process Step



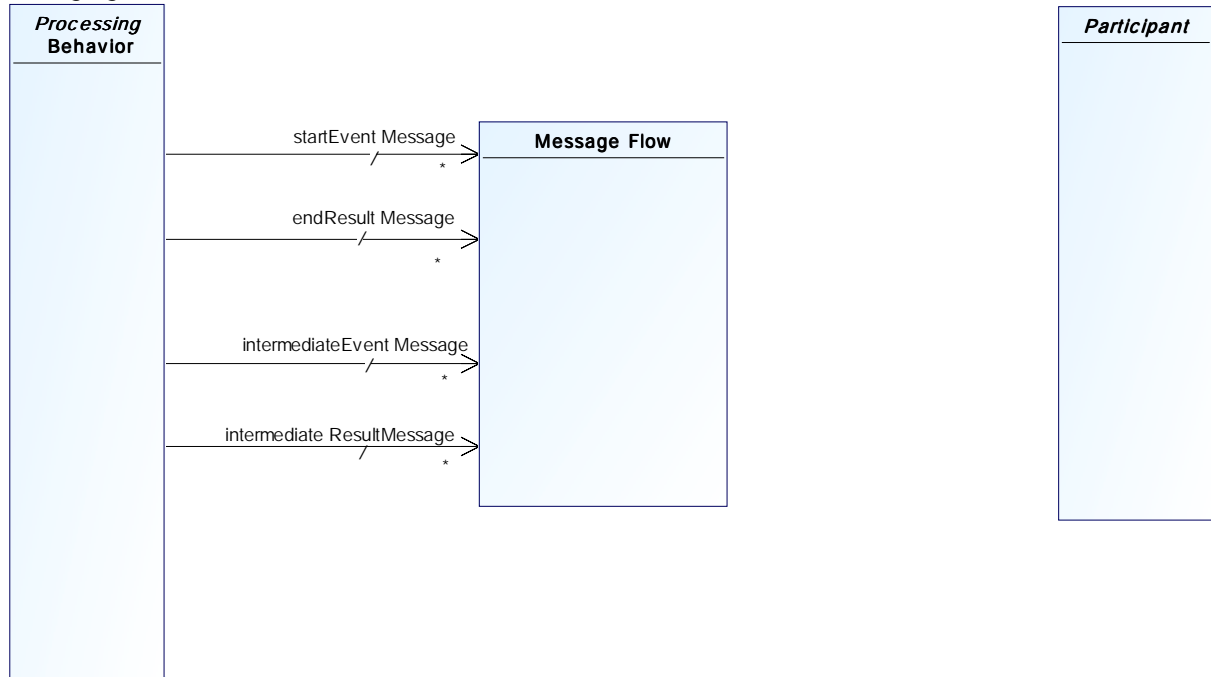
Message Flows



Interaction



Emerging Behavior



Composite

Namespace: Process Interaction Model
isAbstract: Yes

Generalization: “Type”

Description:

A **Composite** is an object that has a complex structure. This structure is made of "**Parts**". The complexity comes from the fact that a **Part** can also be the usage of a **Composite**.

Happening

Namespace: Process Interaction Model
isAbstract:

Description:

A **Happening** is anything that happens. May be natural or caused by an internal or external process

Interaction

Namespace: Process Interaction Model
isAbstract:

Generalization: “Process Step” “Typable Part”

Description:

A **Interaction** is a kind of **Process Step** that is typed by a **Processing Behavior** protocol.

For each **Interaction**, there is one **interaction trigger** and one **interaction fulfillment** that connect the **Interaction** to the corresponding **Participant**.

Interactions define shared agreements between **Participants**. They manifest the agreement for achieving some common transformations by means of coordinated communications.

Associations

| | | |
|--|---------------------|--|
| Interaction Connection : Interaction Connection [0..1] | {source connection} | |
| interaction type : Process Protocol [0..1] | {partType} | |

Interaction Connection

Namespace: Process Interaction Model

isAbstract: No

Generalization: “Part Connection”

Description:

A **Interaction Connection** is a kind of **Part Connection**.

Interaction Connections are used to connect **Interactions** to **Participants**.

A **Interaction Connection** also provides a binding between its **Participant** and the **played participant** within the **Process** that types the connected **Interaction**.

Associations

| | | |
|---|--------|---|
| Interaction : Interaction [1] | {to} | |
| played participant : Participant [0..1] | | |
| player : Participant [1] | {from} | Participant that benefits from the results of a Interaction that it has initiated though a joint action trigger |

Interaction Flow

Namespace: Process Interaction Model

isAbstract: Yes

Generalization: “Part Connection”

Description:

A **Interaction Flow** defines a communication channel between a **Interactive Part** and a **target**.

The kind of communication handled by a **Interaction Flow** is given by its subtypes.

A **Interaction Flow** always has one and only one **Interactive Part** and one and only one **target**.

Interaction Flow is an abstract metaclass.

Associations

| | | |
|-----------------------------------|--------|--|
| carried object type : Type [0..1] | | |
| consumer : Interactive Part [1] | {to} | |
| producer : Interactive Part [1] | {from} | |

Interactive Flow Binding

Namespace: Process Interaction Model

isAbstract:

Description:

A **Interactive Flow Binding** provides an explicit connection between a **Interaction Flow** and the implementation of its sender or receiver.

In many cases, a **Interactive Flow Binding** can be dynamically inferred from the **carried object type** of its **source flow**.

Associations

| | | |
|------------------------------------|--|--|
| source flow : Interaction Flow [1] | | |
| target flow : Interaction Flow [1] | | |

Interactive Part

Namespace: Process Interaction Model

isAbstract: Yes

Generalization: “Part”

Description:

Associations

| | | |
|--------------------------------------|-----------------------|--|
| received flow : Interaction Flow [*] | { target connection } | |
| sent flow : Interaction Flow [*] | { source connection } | |

Message Flow

Namespace: Process Interaction Model

isAbstract:

Generalization: “Interaction Flow” “Process Step”

Description:

A **Message Flow** is a **Interaction Flow** that occurs from or to a **Participant**. **Message Flows** are used to describes communications that occur at the boundaries of **Processing Behaviors**.

A **Message Flow** is also a **Process Step** that represent an effect used to synchronized **Processing Behaviors**.

Associations

| | | | |
|--|----------------------|----------------|--|
| Processing Behavior : Processing Behavior [] | { connection whole } | { part whole } | |
|--|----------------------|----------------|--|

Participant

Namespace: Process Interaction Model

isAbstract: Yes

Generalization: “Interactive Part” “Typable Part”

Description:

Participants are used to define the range of metaclasses that can participates to **Interactions**.

There are two kinds of **Participant**: **Protocol Participant** and **PerformerRole**.

Associations

| | | |
|---|-----------------------|--|
| Interaction Connection : Interaction Connection [*] | { target connection } | |
| Processing Behavior : Processing Behavior [0..1] | { part whole } | |
| Processing Behavior : Processing Behavior [] | { part whole } | |

Process

Namespace: Process Interaction Model

isAbstract:

Generalization: “Processing Behavior”

Description:

A **Process** is a kind of **Processing Behavior** that describes prescribed communications between participants. **Processes** are used to define agreed coordination between **Protocol Participants** based on **Interactions** and **Message Flows**.

Process Step

Namespace: Process Interaction Model

isAbstract: Yes

Generalization: "Happening" "Step"

Description:

A **Process Step** represents a **Happening** that occurs in the context of a **Processing Behavior**. A **Process Step** can be either the effect of a doing or a doing itself.

Process Steps are **Steps** organized as ordered series that manifest the course of the **Processing Behavior** in which they occur. As such, **Process Steps** can be connected by **Successions**.

A **Process Step** has a **duration** and is started by the occurrence of a **Succession**.

Associations

| | | |
|--|--------------|--|
| Processing Behavior : Processing Behavior [] | {part whole} | |
|--|--------------|--|

Processing Behavior

Namespace: Process Interaction Model

isAbstract: Yes

Generalization: "Behavior" "Course"

Description:

A **Processing Behavior** is an abstract metaclass that is the basis for any process metaclass.

Processing Behaviors are kind of **Behavior** that have the following characteristics that distinguish them from other kind of behavior

- They aim at producing a specific result and therefore do not present themselves as cycles
- They are organized as a serie of steps
- They use communications through message flows for coordination

Associations

| | | |
|----------------------------------|-------------------------------------|--|
| Message Flow : Message Flow [] | {composite part} {owned connection} | |
| Participant : Participant [*] | {composite part} | |
| Process Step : Process Step [] | {composite part} | |
| processingPort : Participant [*] | {composite part} | |

Protocol Participant

Namespace: Process Interaction Model

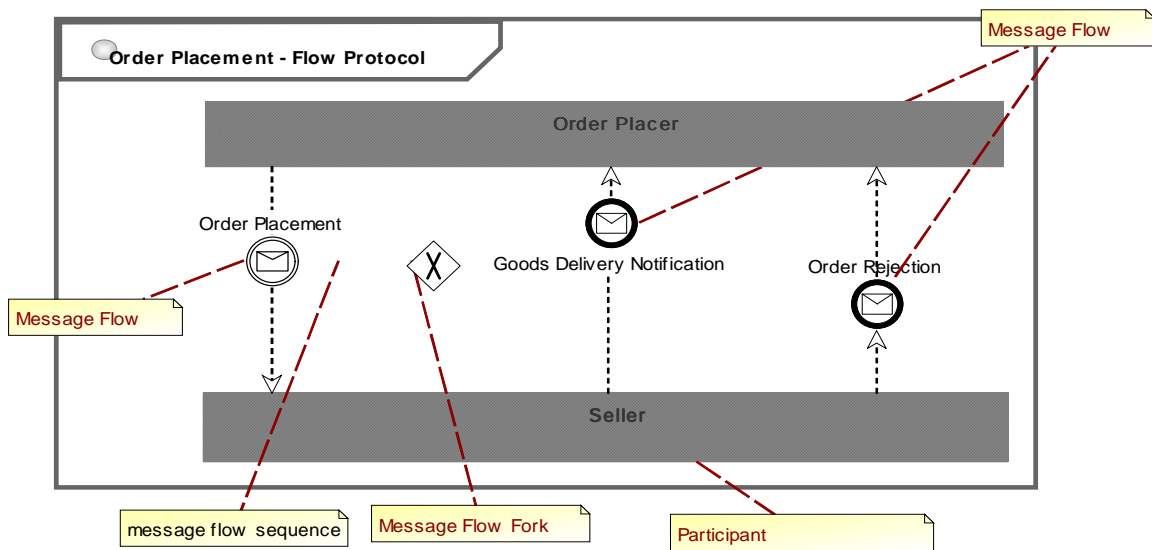
isAbstract:

Generalization: "Participant"

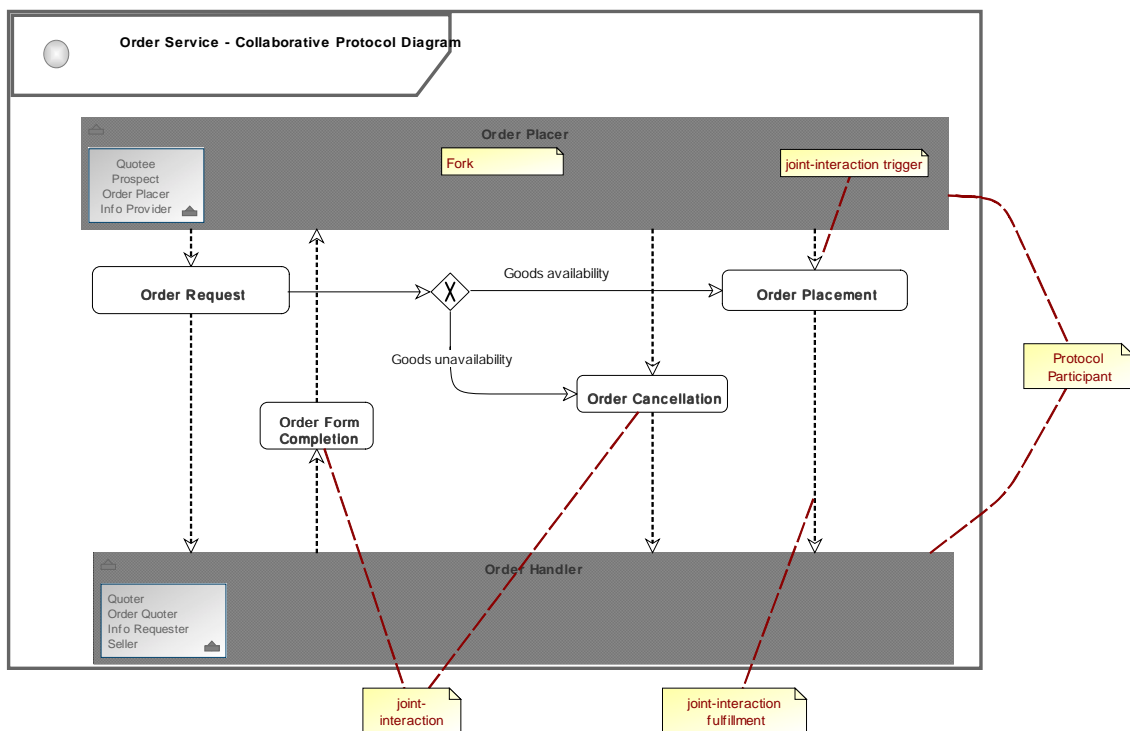
Description:

A **Protocol Participant** is a concrete **Participant** that define a boundary of its **Processing Behavior**.

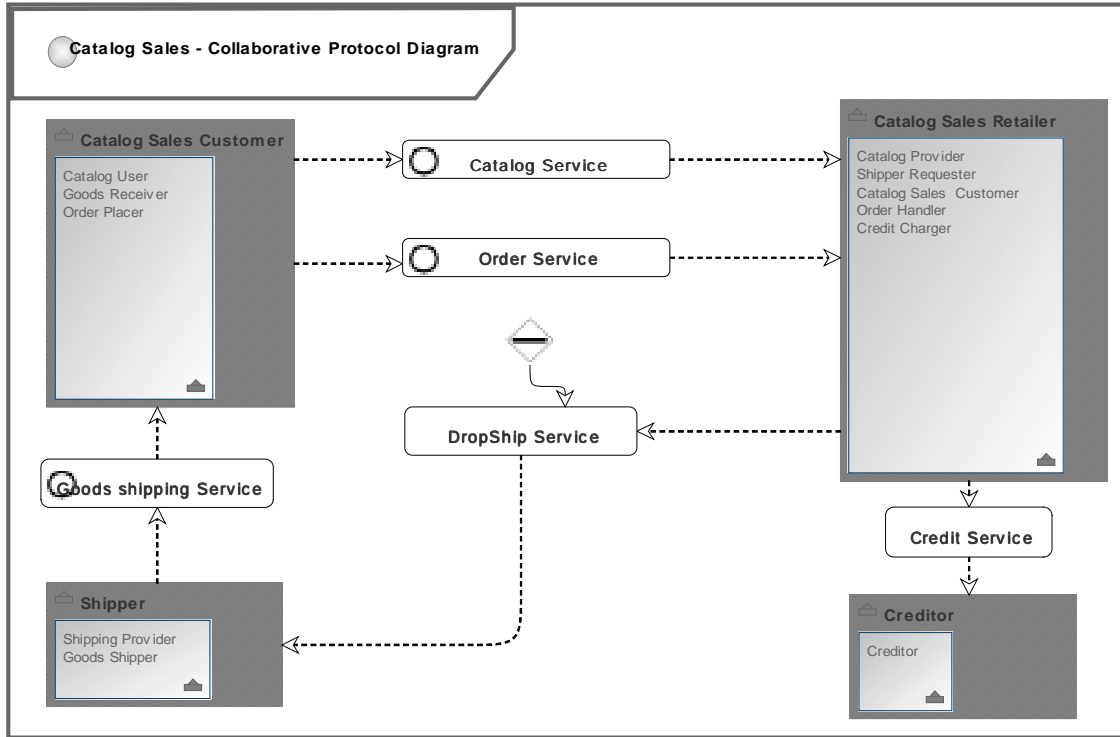
Notation samples
Flow Protocol



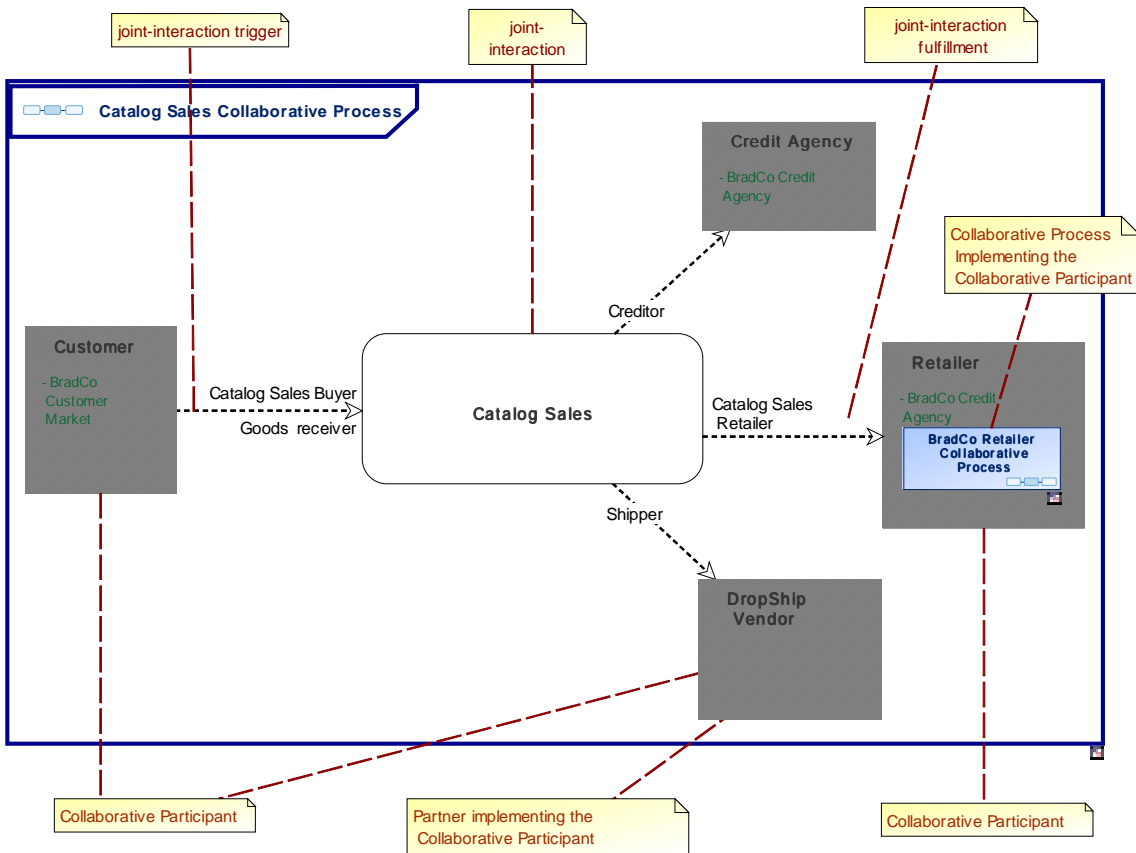
Collaborative Protocol Diagram



Collaborative Protocol Diagram



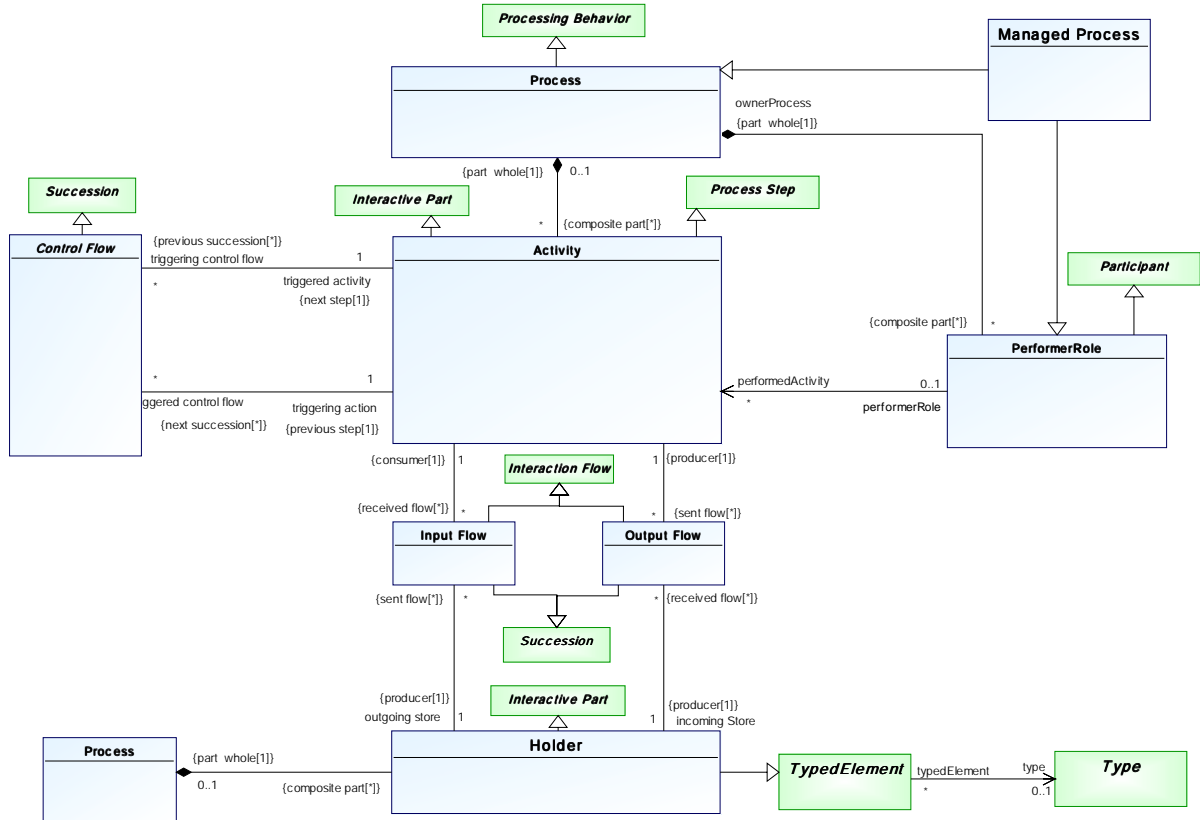
Collaborative Process Diagram



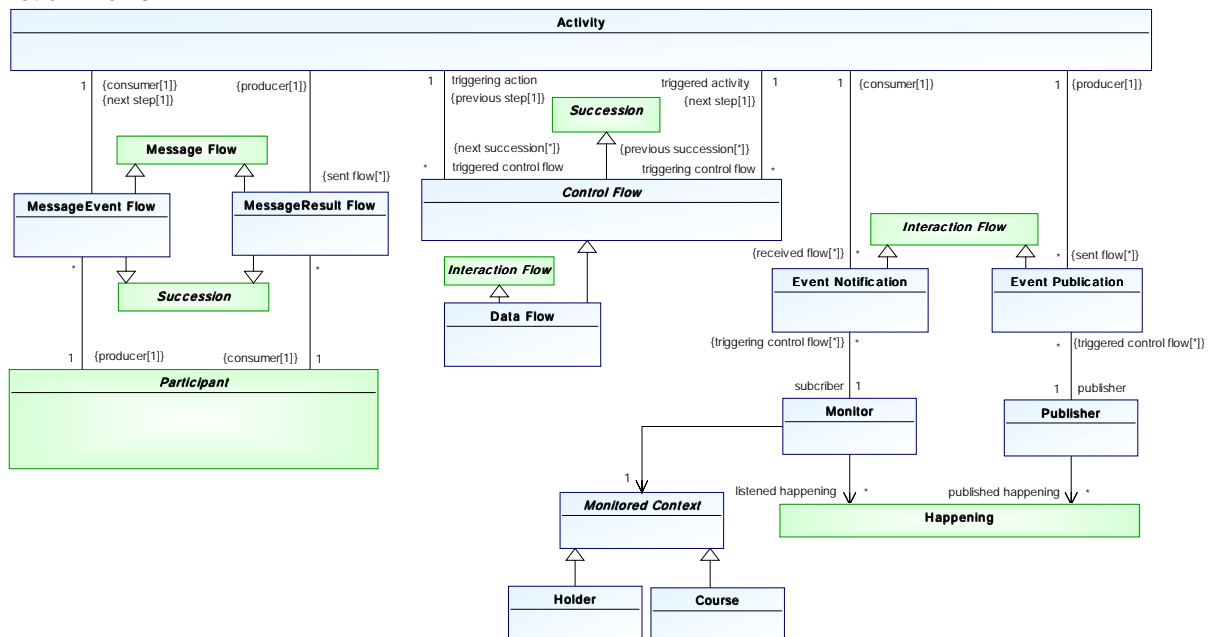
Process Activity Model

The **Process Activity Model** focuses on transformation processes.

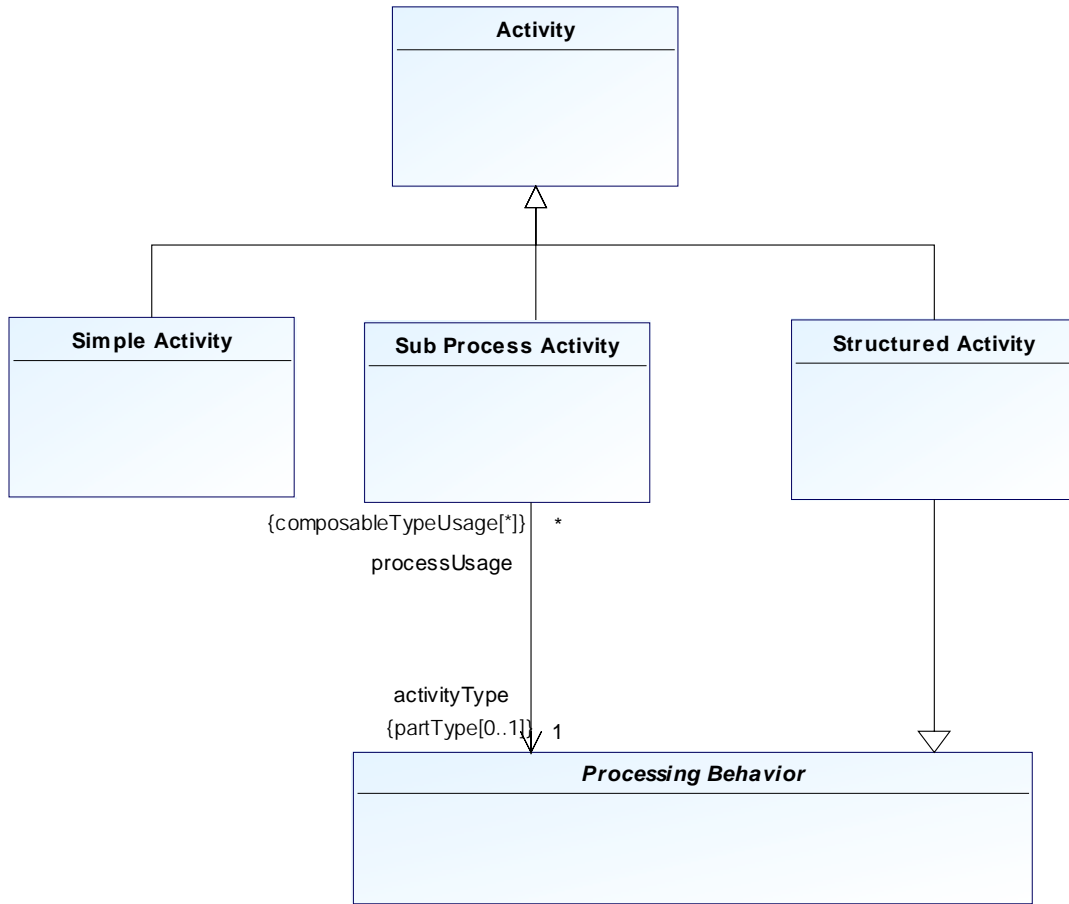
Core



Action Flows



ActionNodes



Control Nodes

Activity

Namespace: Process Activity Model

isAbstract: No

Generalization: “Process Step” “Interactive Part”

Description:

An **Activity** is a transformation step in a **Process**.

There are three kinds of **Activity**: **Simple Activity** , **Sub Process Activity**, **Structured Activity**

Associations

| | | |
|---|-----------------------|--|
| Event Notification : Event Notification [*] | {received flow} | |
| Event Publication : Event Publication [*] | {sent flow} | |
| Input Flow : Input Flow [*] | {received flow} | |
| MessageEvent Flow : MessageEvent Flow [] | | |
| MessageResult Flow : MessageResult Flow [] | {sent flow} | |
| Output Flow : Output Flow [*] | {sent flow} | |
| Process : Process [0..1] | {part whole} | |
| triggered control flow : Control Flow [*] | {next succession} | |
| triggering control flow : Control Flow [*] | {previous succession} | |

Control Flow

Namespace: Process Activity Model

isAbstract: Yes

Generalization: “Succession”

Description:

An **Control Flow** is a kind of **Interaction Flow** that acts as a **Succession** between **Activitys** within an **Processing Behavior**. An **Control Flow** is connected to at least one **Activity**, either as a **consumer** or a **producer**.

The different subtypes of **Control Flow** specify which other type can participate to the **Control Flow**.

Control Flow is a redefinition UML 2.0 ActivityEdge

Associations

| | | |
|-----------------------------------|-----------------|--|
| triggered activity : Activity [1] | {next step} | |
| triggering action : Activity [1] | {previous step} | |

Data Flow

Namespace: Process Activity Model

isAbstract:

Generalization: “Control Flow” “Interaction Flow”

Description:

Event Notification

Namespace: Process Activity Model

isAbstract:

Generalization: “Interaction Flow”

Description:

An **Event Notification** is a kind of **Input ActionFlow** that connects an **Monitor** to an **Activity**

Associations

| | | |
|--------------------------|------------|--|
| Activity : Activity [1] | {consumer} | |
| subscriber : Monitor [1] | | |

Event Publication

Namespace: Process Activity Model

isAbstract:

Generalization: “Interaction Flow”

Description:

An **Event Publication** is a kind of **Output Flow** that connects an **Activity** to an an **Publisher**

Associations

| | | |
|---------------------------|------------|--|
| Activity : Activity [1] | {producer} | |
| publisher : Publisher [1] | | |

Holder

Namespace: Process Activity Model

isAbstract:

Generalization: “Interactive Part” “Monitored Context” “TypedElement”

Description:

Associations

| | | |
|-------------------------------|-----------------|--|
| Input Flow : Input Flow [*] | {sent flow} | |
| Output Flow : Output Flow [*] | {received flow} | |
| Process : Process [0..1] | {part whole} | |

InitialNode

Namespace: Process Activity Model

isAbstract:

Generalization: “Step”

Description:

Input Flow

Namespace: Process Activity Model

isAbstract:

Generalization: “Interaction Flow” “Succession”

Description:

A **Input Flow** is a kind of **Control Flow** that carries a **Resource**. The carried **Resource** is coming out the previous **Activity** and going into the next **Activity**.

Associations

| | | |
|-----------------------------|------------|--|
| Activity : Activity [1] | {consumer} | |
| outgoing store : Holder [1] | {producer} | |

Managed Process

Namespace: Process Activity Model

isAbstract:

Generalization: “PerformerRole” “Process”

Description:

MessageEvent Flow

Namespace: Process Activity Model

isAbstract:

Generalization: “Message Flow” “Succession”

Description:

A **MessageEvent Flow** is a kind of **Message Flow** that triggers an **Activity** in an **Processing Behavior**

Associations

| | | |
|-------------------------------|-----------------------|--|
| Activity : Activity [1] | {next step}{consumer} | |
| Participant : Participant [1] | {producer} | |

MessageResult Flow

Namespace: Process Activity Model

isAbstract:

Generalization: “” “Message Flow” “Succession”

Description:

Associations

| | | |
|-------------------------------|------------|--|
| Activity : Activity [] | {producer} | |
| Participant : Participant [1] | {consumer} | |

Monitor

Namespace: Process Activity Model

isAbstract:

Description:

An **Monitor** is the declaration of the intent of a **Processing Behavior** to react to the occurrences of an **Event** by the triggering of an **Activity** or is the result of an **Activity**

Associations

| | | |
|---|---------------------------|--|
| Event Notification : Event Notification [*] | {triggering control flow} | |
| listened happening : Happening [*] | | |
| Monitored Context : Monitored Context [1] | | |

Monitored Context

Namespace: Process Activity Model

isAbstract: Yes

Description:

Output Flow

Namespace: Process Activity Model

isAbstract:

Generalization: “Interaction Flow” “Succession”

Description:

Associations

| | | |
|-----------------------------|------------|--|
| Activity : Activity [1] | {producer} | |
| incoming Store : Holder [1] | {producer} | |

PerformerRole

Namespace: Process Activity Model

isAbstract: No

Generalization: “Participant”

Description:

A **PerformerRole** is a kind of **Participant** in a **Process** that can also perform **Activity**.

Associations

| | | |
|----------------------------------|--------------|--|
| ownerProcess : Process [] | {part whole} | |
| performedActivity : Activity [*] | | |

Publisher

Namespace: Process Activity Model

isAbstract:

Description:

An **Publisher** is the declaration of the intent of a **Processing Behavior** to publish the result of an **Activity** as an **Event** occurrence.

Associations

| | | |
|---|--------------------------|--|
| Event Publication : Event Publication [*] | {triggered control flow} | |
| published happening : Happening [*] | | |

Rule

Namespace: Process Activity Model

isAbstract:

Generalization: “Succession Condition”

Description:

Simple Activity

Namespace: Process Activity Model

isAbstract:

Generalization: “Activity”

Description:

A **Simple Activity** is an **Activity** that is its own behavior. **Simple Activities** are final **Activities** that are performed directly by a **PerformerRole**.

Structured Activity

Namespace: Process Activity Model

isAbstract:

Generalization: “Activity” “Processing Behavior”

Description:

Structured Activities are similar to UML 2.0 structured activities. They are used for process hierarchical decomposition.

Sub Process Activity

Namespace: Process Activity Model

isAbstract:

Generalization: “Activity”

Description:

A **Sub Process Activity** is an **Activity** that trigger a process.

Associations

| | | |
|--|------------|--|
| activityType : Processing Behavior [1] | {partType} | |
|--|------------|--|

Terminate

Namespace: Process Activity Model

isAbstract:

Generalization: “Step”

Description:

A **Terminate** is a **ConControlAction** that that stops all flows in an **Processing Behavior**.

ValueSpecification

Namespace: Process Activity Model

isAbstract:

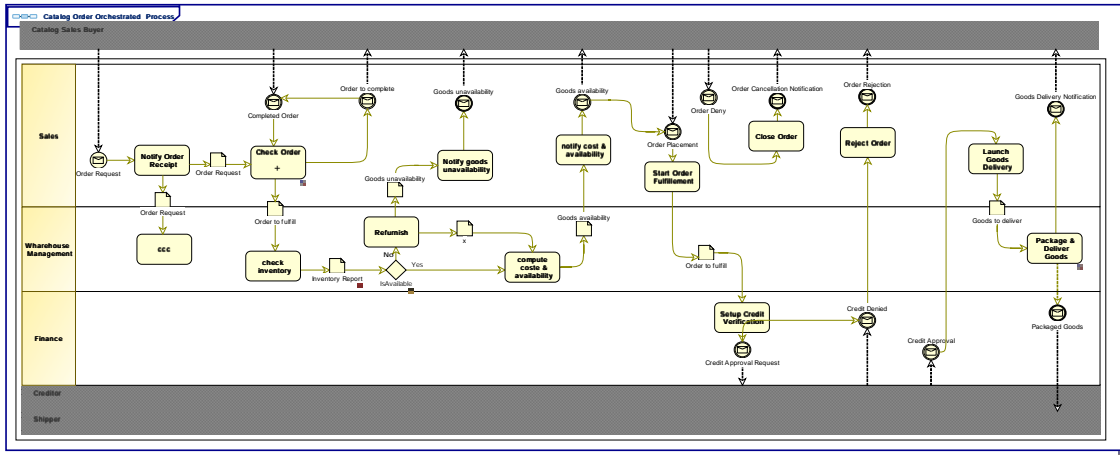
Generalization: “Succession Condition”

Description:

From UML 2.0 intra

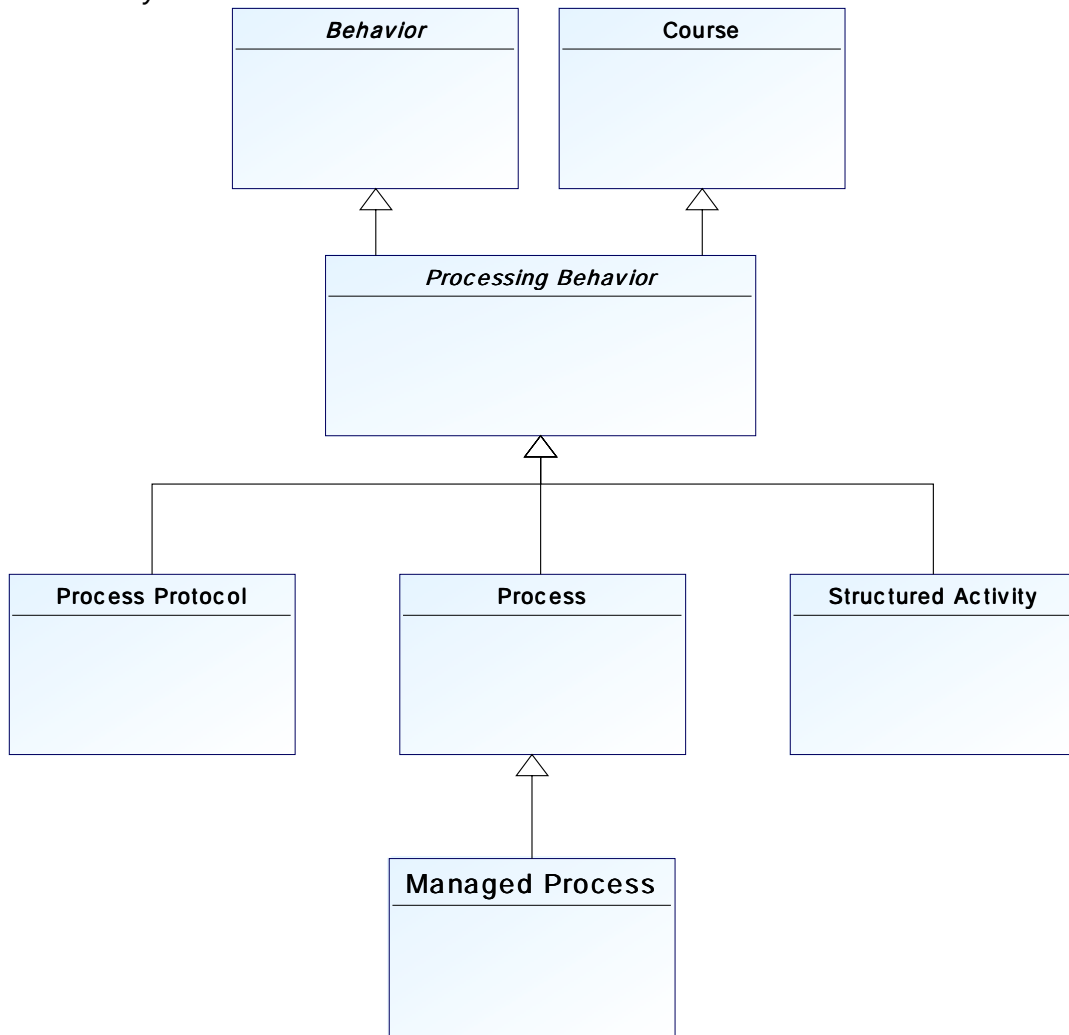
Notation samples

Orchestrated Process Diagram (BPMN)

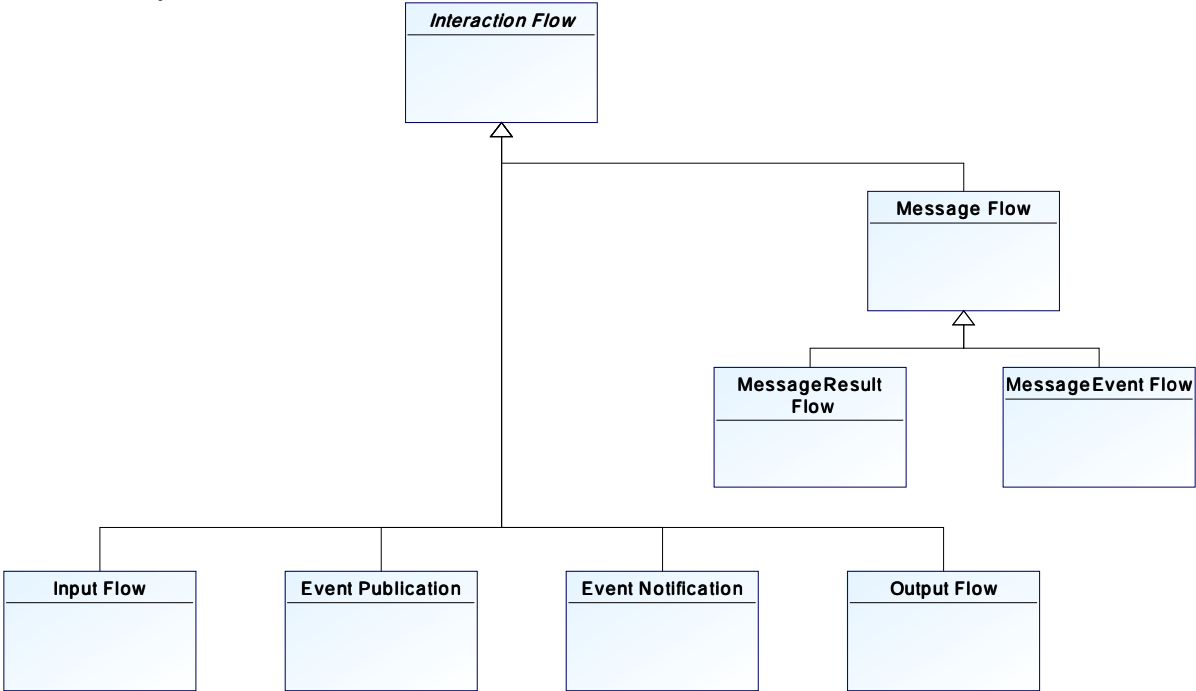


Class Hierarchies

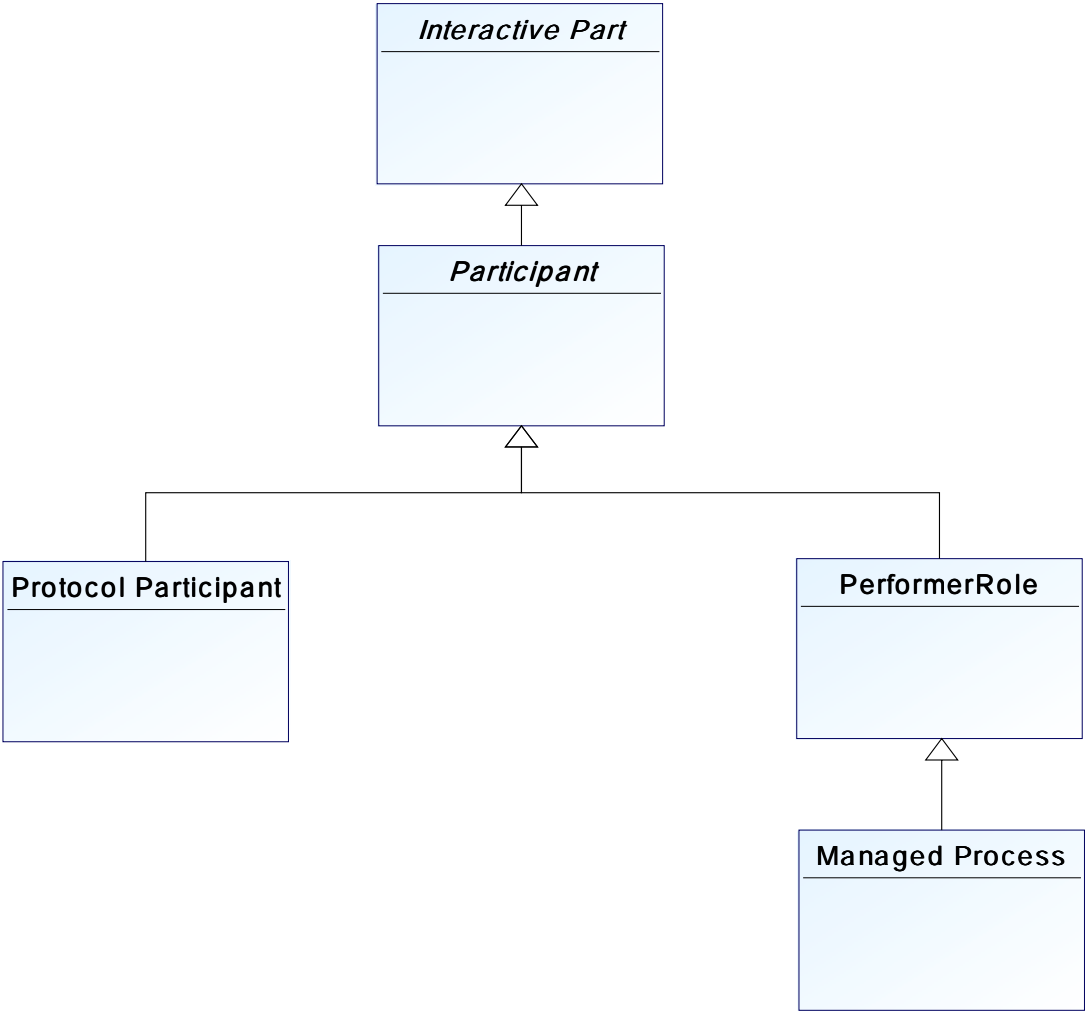
Process Hierarchy



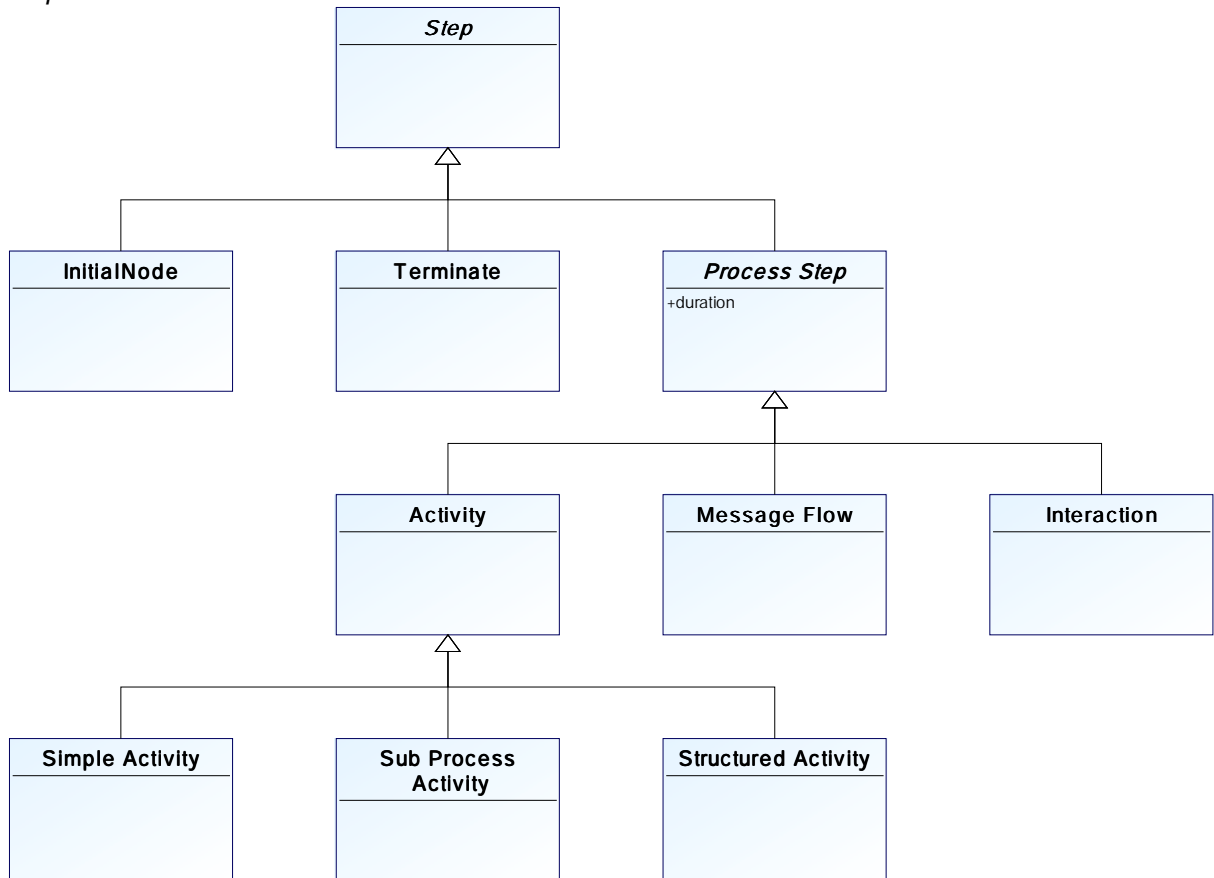
Flow Hierarchy



Participant hierarchie



Step Hierarchie



8 Language Mappings

The following sub-sections describe mappings to specific languages as proofs of concept.

BPEL mapping

To be completed in a later version of this specification.

To be completed in a later version of this specification.

W3C Choreography mapping

To be completed in a later version of this specification.

The following paragraphs discuss the submission relative to the RFP requirements. Paragraph numbering starts at 6.5 to correspond to the RFP numbering.

6.5 Mandatory requirements

Responses to the issues to be discussed are presented below. The RFP requirements are expressed in italics.

Required Metamodel

Responses to this RFP shall provide a metamodel forming an abstract language for the expression of business process definitions.

Proposals shall depict the solicited metamodel using UML.

Metamodel Compatibility

Proposals shall use the appropriate elements of existing metamodels including, UML, EDOC, MOF and OCL.

The metamodel is based on UML and uses OCL and a MOF compatible sub-set of UML.

MOF Compliance

The resulting metamodel shall be MOF-compliant.

Will be addressed in final specification.

Procedural and rule-based flow of control

Proposals shall provide for specification of process flow based on control flow from completed activities to activities to be initiated as well as initiation of activities based on rules or pre-conditions.

Will be addressed in final specification.

Specification of Activity Performers

Proposals shall provide for the specification of selection criteria for performers and resources, including human performers, applications, passive resources and sub-processes.

The basis for selection shall include

- 1. Specific resource identity*
- 2. Resource attributes*
- 3. Relationships with other assigned resources*
- 4. Relationships to subject matter*
- 5. Combinations of the above.*

Will be addressed in final specification.

Asynchronous and Concurrent Execution

Proposals shall provide mechanisms for specifying concurrent execution of activities:

- a) A process model shall be able to define when multiple, concurrent activities are initiated.*
- b) A process model shall be able to define when an activity or completion of a process depend on the completion of multiple, concurrent activities. This shall include initiation of an activity based on completion of n of m concurrent activities (where $n \leq m$).*
- c) Business processes shall be able to invoke processes that execute asynchronously.*
- d) The model shall support specification of the publication of events and messages for asynchronous delivery.*
- e) The model shall support receipt of messages from a collaborator and subscription to events. Messages and events shall be received as asynchronous inputs to a receiving activity executing concurrently with other activities in the process.*

Interaction Models and Activity Models handle the requirements

Specification of initiation and termination

Proposals shall provide modeling constructs for specifying when and how activities and processes can be initiated and terminated:

- a) Pre or post conditions*
- b) Actions for initialization and termination with consideration of actions required for abnormal termination, including the initiation of compensating processes.*
- c) Propagation of termination to active activities and sub-processes.*

Collaborative processes handle interaction fault. Issue will be addressed in final specification for Internal Processes.

Choreography

Proposals shall provide for the specification of the signals and exchanges performed between processes to achieve collaboration as described in 6.2.5.

Addressed by the Collaborative Model

Audit Log Generation

Proposals shall include provisions in the metamodel to allow the specification of business logic related audit log records. This part of the metamodel shall provide for the specification of:

- a) The content of the log record in relation to the process definition*

b) *The timing of the log record emission*

Will be addressed in final specification.

Distributed Execution

Proposals shall ensure that the form of definition does not preclude distributed execution.

The metamodel does not preclude distributed execution.

Process Definition import and export

Proposals shall support XMI export and import for exchange of process definitions.

As a MOF metamodel, it is XMI compatible. XMI will be addressed in the final specification.

Non-Normative Notation Mappings

As a proof of concept, proposals shall provide non-normative mappings to two recognized business process modeling languages, e.g.:

- *BPELAWS*
- *XPDL*

Will be addressed in final specification.

Compatible Versions of Existing Specifications

The final, revised submission shall be based on the most recently adopted version of related specifications (e.g., UML and MOF) that is adopted three months prior to the final revised submission deadline for this RFP.

Will be addressed in final specification.

Optional requirements

Responses to the optional requirements are presented below. The RFP requirements are in italics.

Additional Non-Normative Mappings

Proposals may provide additional mappings to recognized languages for business process definition, complementing the list of mandatory mappings requested in 6.5.12.

Will be addressed in final specification.

Additional Execution Constraints

Proposals may provide for the ability to model additional execution constraints, like maximum duration of a process or activity execution. For these additional constraints the behavior of constraint violation should be modeled and its affect on the process enactment described.

Will be addressed in final specification.

Discussion topics

Responses to the issues to be discussed are presented below. The RFP requirements are expressed in italics.

Relationship to existing UML metamodel

Proposals shall discuss the relationship of model elements used for business process modeling to the existing UML metamodel to demonstrate consistency with the UML metamodel.

Final mapping will be addressed in final specification.

Relationship to Related UML Profiles, Metamodels and Notations

Proposals shall discuss how business process definitions may be incorporated with or complement other UML profiles, metamodels and notations for specification of business systems, particularly the UML Profile for EDOC.

Will be addressed in final specification.

Mapping to Existing Business Process notations and UML Notation

Proposals shall discuss how the proposed metamodel may be mapped to existing process definition notations as a demonstration of completeness and compatibility.

Will be addressed in final specification.

Resource Model

Proposals shall describe assumptions regarding an associated resource model.

Will be addressed in final specification.

Relationships with related OMG specification activities.

Proposals shall discuss how the specifications relate to the specification development efforts currently under way as noted in Section 6.4.3.

This will be discussed when we are closer to a final specification.

Consistency checks

Proposals shall discuss how the specification supports consistency checks, particularly between choreography specifications and a business process that participates in the choreography.

Will be addressed in final specification.

Access Control

Proposals shall discuss how access authorization for process data, artifacts, activities in a process, and process enactment may be based on process roles of individuals associated with a specific process instance.

Will be addressed in final specification.

Web services and collaboration support

Proposals shall discuss how the specification supports the definition of business processes and choreography for web services and other collaborations including the relationships with messages, documents, interface specifications, participant roles, signatures and message exchanges.

Collaborative Processes are used to define WSDL:PortType for each Participant

10 Compliance Levels

The following levels of compliance are defined for this specification.

PIM Compliance

An implementation is PIM compliant if it implements the complete PIM model in a specific technology.

Other?

Appendix A: Glossary