BPEL vs. BPMN 2.0:
Should You Care?

(2nd Intl. Workshop BPMN 2010, Potsdam, Germany; October 13, 2010)

Prof. Dr. Frank Leymann
Institute of Architecture of Application Systems

University of Stuttgart
Universitätsstr. 38
70569 Stuttgart
Germany

Phone   +49-711-7816 470
Fax   +49-711-7816 472
e-mail   Leymann@iaas.
            uni-stuttgart.de

Agenda

Historical Remarks

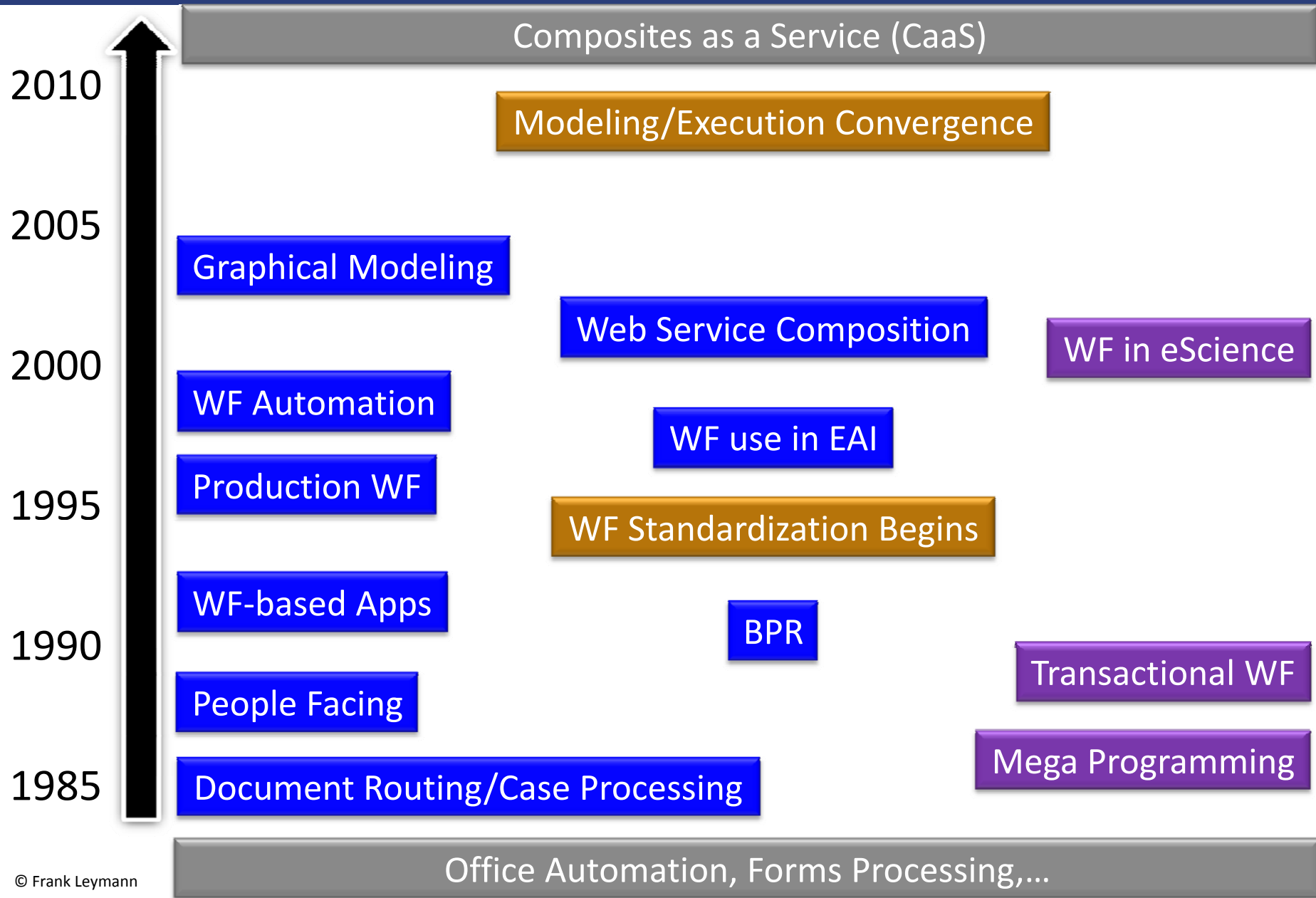BPMN Origin and Positioning

The Notion of "Native Metamodel"

View on Process Engines
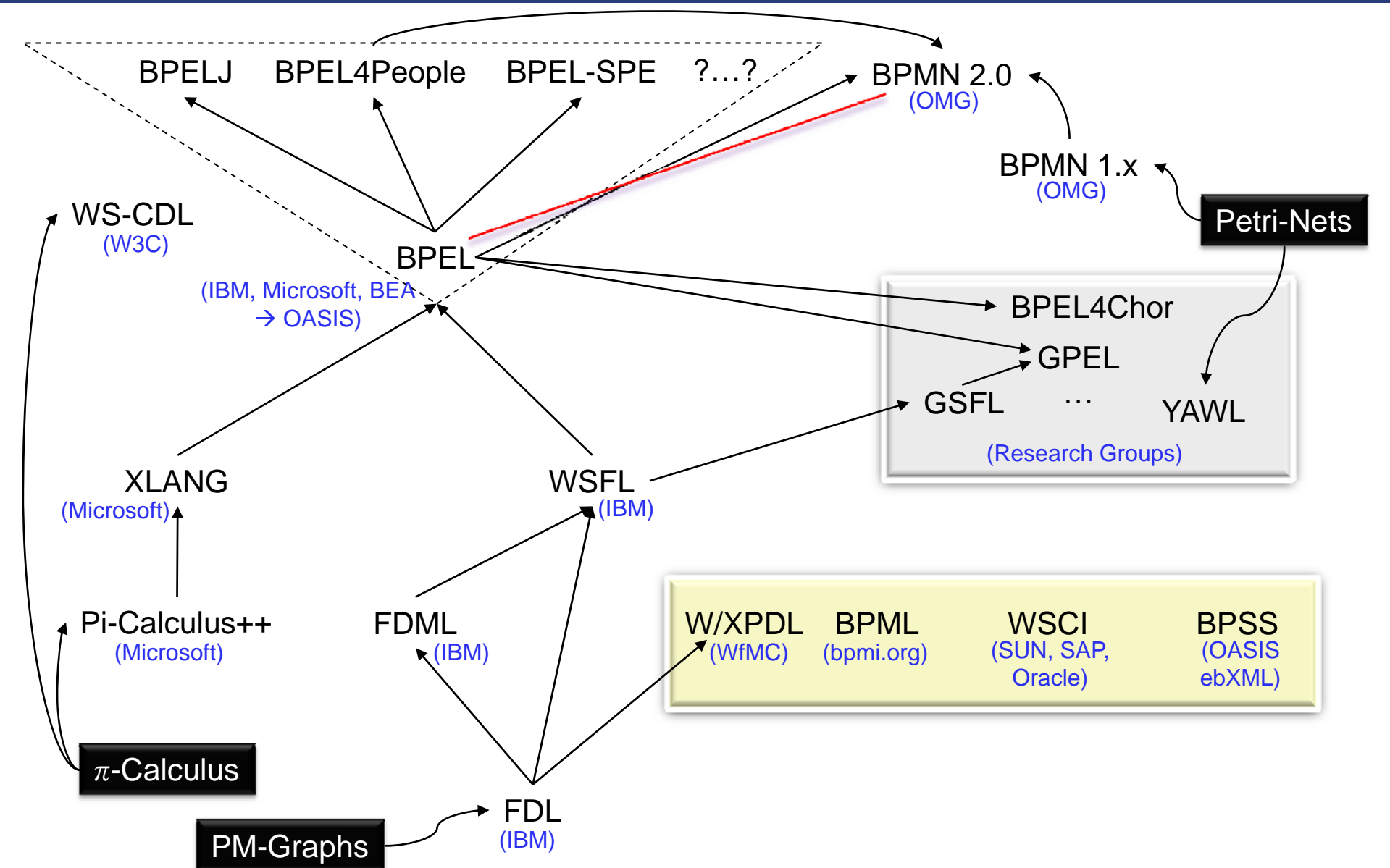
Selective BPMN 2.0 Features
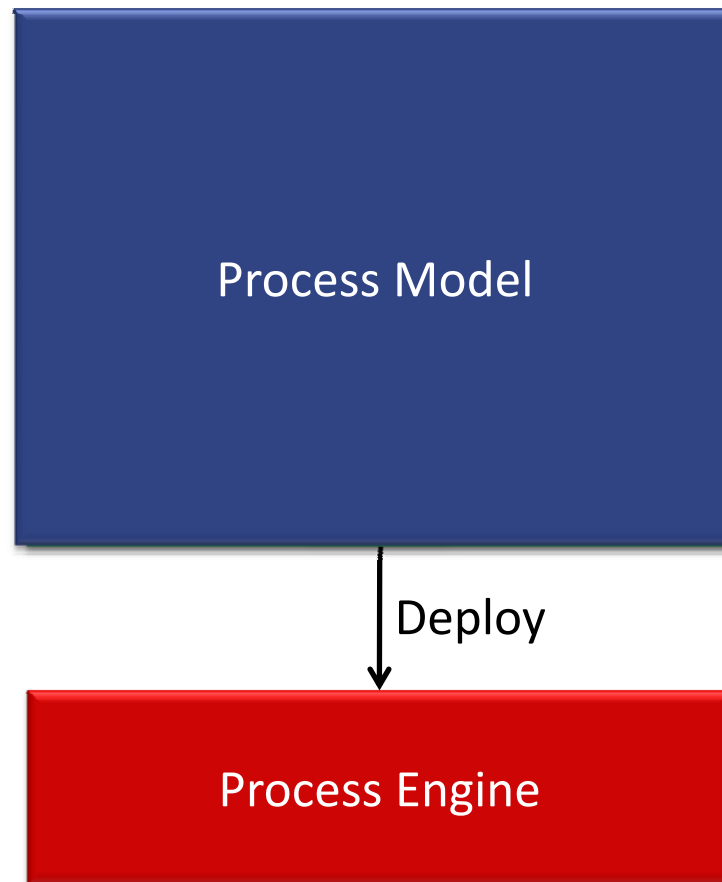
Executable BPMN - BPEL Mapping

Summary

© Frank Leymann

Brief History of Workflow Technology (no scientific rigor intended)

Composites as a Service (CaaS)

2010

Modeling/Execution Convergence

2005

Graphical Modeling

Web Service Composition

WF in eScience

2000

WF Automation

WF use in EAI

Production WF

1995

WF Standardization Begins

WF-based Apps

BPR

1990

Transactional WF

People Facing

1985

Document Routing/Case Processing

Mega Programming

Office Automation, Forms Processing,…

Workflow Language Genealogy

BPELJ BPEL4People BPEL-SPE ?...? BPMN 2.0
(OMG)

BPMN 1.x
(OMG)

Petri-Nets

WS-CDL
(W3C)

BPEL
(IBM, Microsoft, BEA
→ OASIS)

BPEL4Chor

GPEL

GSFL ... YAWL

(Research Groups)

XLANG
(Microsoft)

WSFL
(IBM)

Pi-Calculus++
(Microsoft)

FDML
(IBM)

W/XPDL   BPML   WSCI   BPSS
(WfMC)   (bpmi.org) (SUN, SAP, (OASIS
                      Oracle)   ebXML)

π-Calculus

FDL
(IBM)

PM-Graphs

Process Modeling is (has been?) Layered

Process Model

Deploy

Process Engine

Process Modeling is (has been?) Layered

Conceptual Process Model

BPMN

↓ Transform

Logical Process Model

BPEL

↓ Deploy

Process Engine
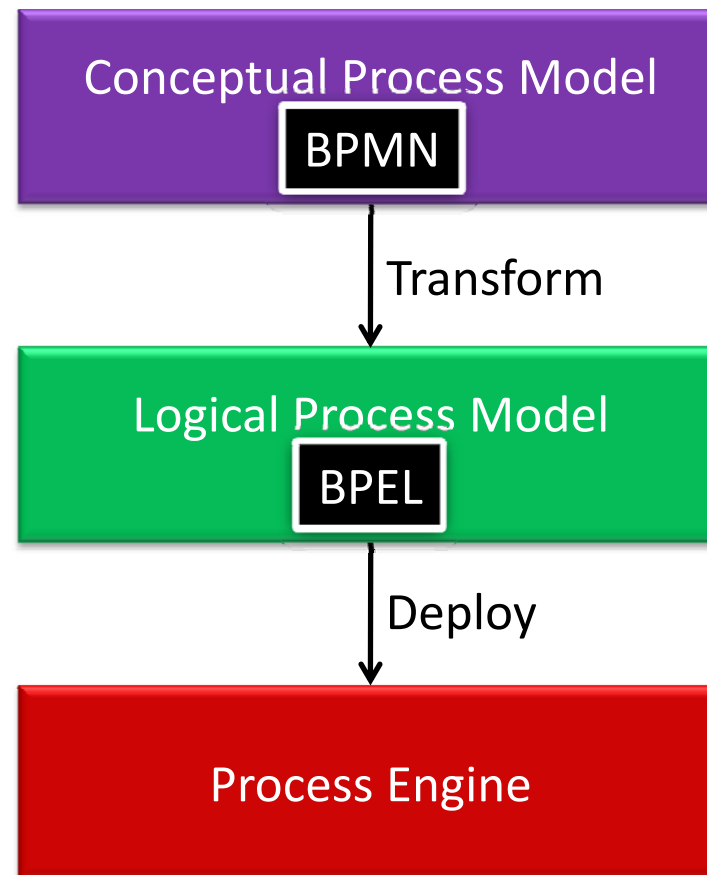
BPM Lifecycle: Where BPMN Fits

KPIs

Modeling

Analysis

IT Refinement

Monitoring

Deployment

New in 2.0!
(but this is *optional*)

Execution
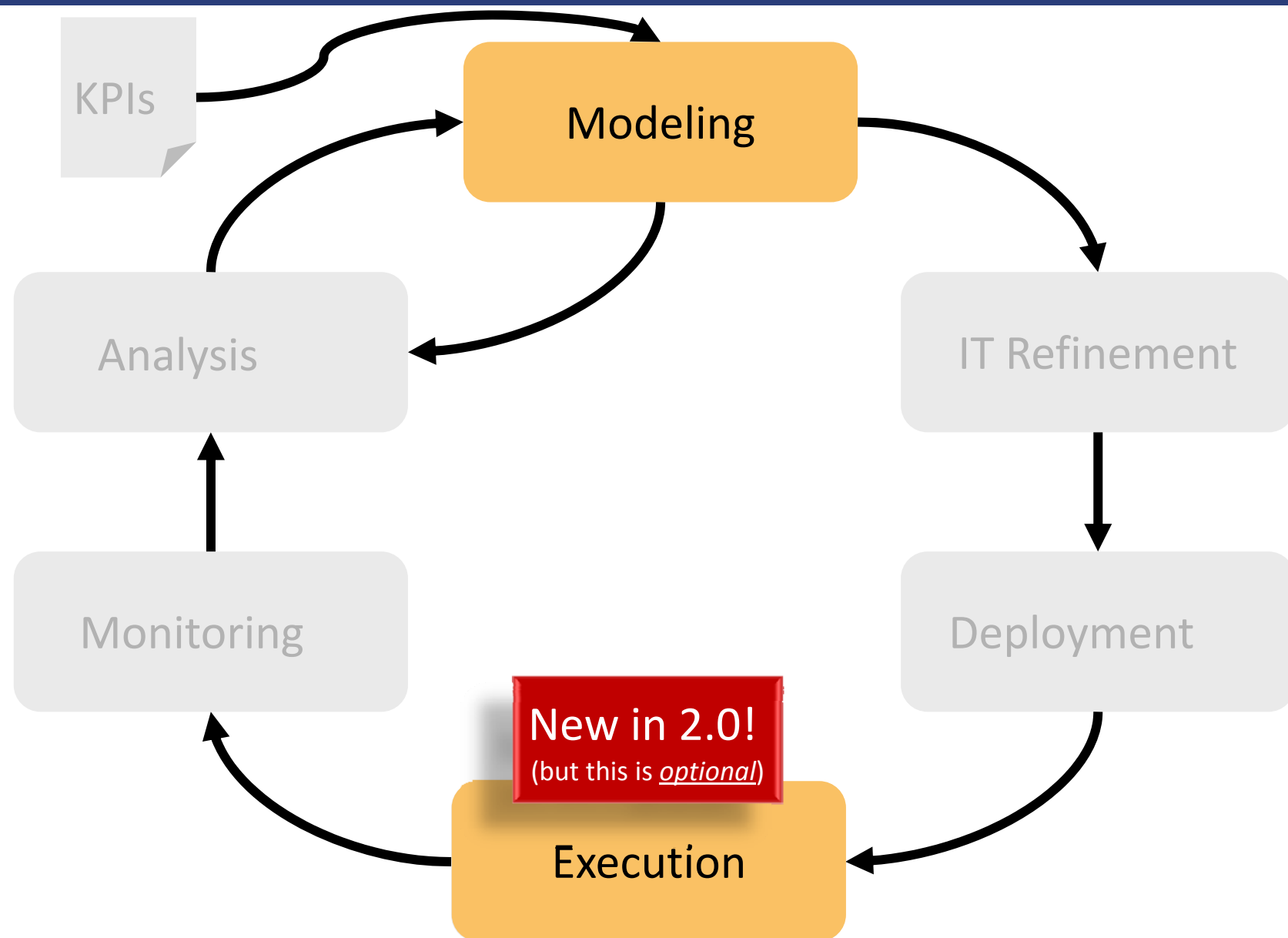
BPMN 2.0 vs. BPMN 1.1

- BPMN 2.0 is based on BPMN 1.1 but additionally contains:
    - An XML-based *interchange format* for BPMN models
    - Clean operational semantics and metamodel (connecting BPMN to OMGs MDA efforts)
        - The metamodel is one of the most important changes, hence the new name (see below)
- The graphical notation is mostly unchanged
    - some new event types, choreography, collaboration,…
- New BPMN "Level": executable BPMN

> BPMN in 1.1: "**B**usiness **P**rocess **M**odeling **N**otation"
> BPMN in 2.0: "**B**usiness **P**rocess **Model and Notation**"
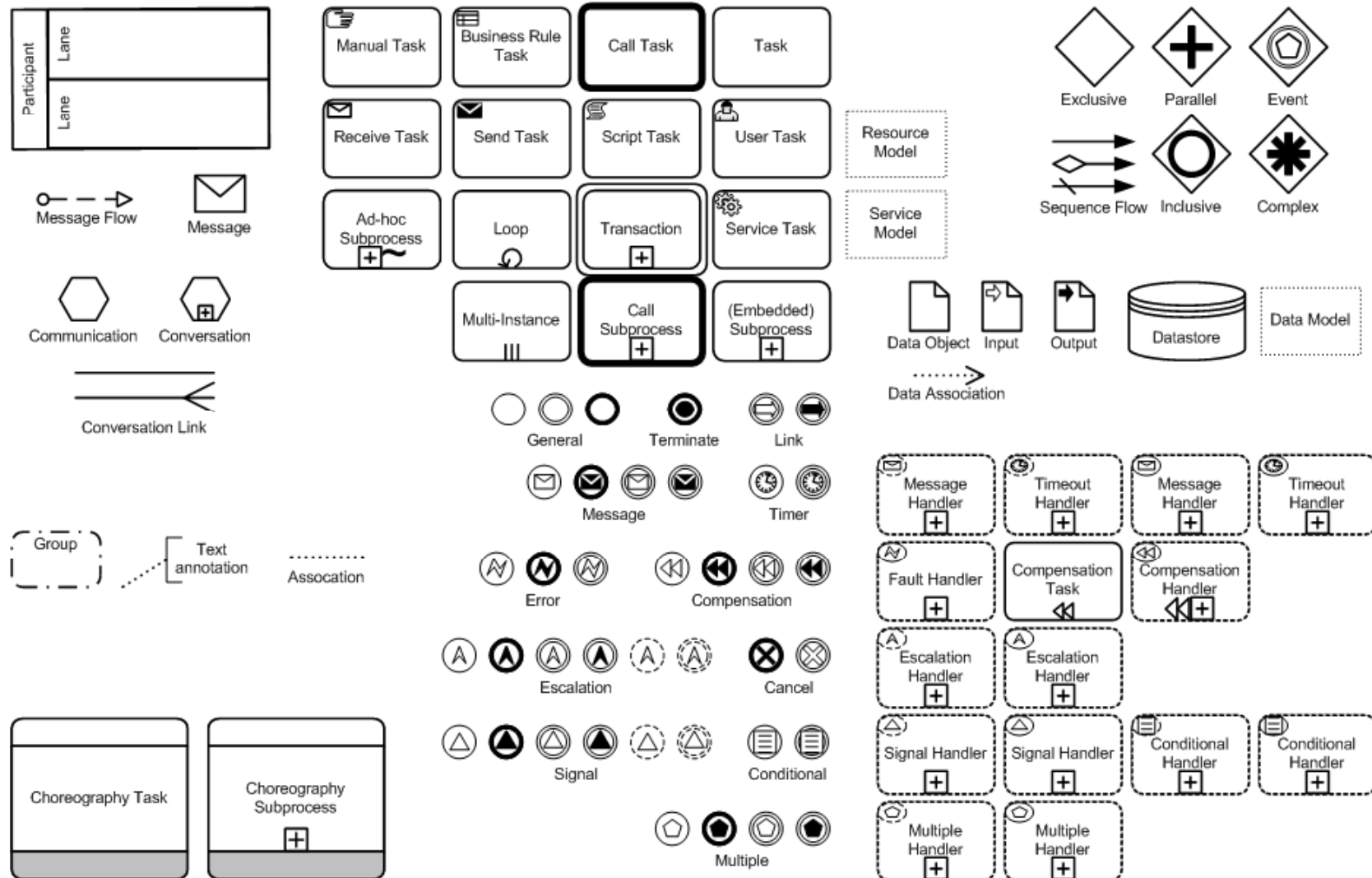
Agenda

Historical Remarks

BPMN Origin and Positioning

The Notion of "Native Metamodel"

View on Process Engines

Selective BPMN 2.0 Features

Executable BPMN - BPEL Mapping

Summary

The Key Aspect of BPEL

BPEL exists and is supported by most middleware vendors!

- Combination of two modeling paradigms
- Avoided split of WF product market/industries
- <u>The</u> runtime standard
- Well defined syntax & operational semantics
- (Certain degree of) portability across vendors
- (High degree of) tool portability
- Portability of skills

- And, yes, even original authors think it's a chimera, i.e. they don't think it's a beautiful language 🙂

BPMN 1.x Filled a Gap

- BPEL focused on language syntax and operational semantics

- It sacrificed time-to-market and ignored visual representations by will

- When looking back this was a BIG mistake!

- BPEL turned out to be too difficult for mere mortals

- BPMN 1.x filled this gap

- Non-IT experts could communicate about business processes

- Big progress in BPM technology

Chasm between BPMN 1.x and BPEL

- BPEL processes are modeled with their future execution in mind

- This has not been often the case in the past for BPMN

- But more and more, BPMN users wanted their processes to be executed

- Wide-spread support of BPEL in workflow engines required to map BPMN to BPEL

- But BPMN and BPEL metamodels are quite different

- Thus, mapping is not straightforward, even complex

- Many of you understand this much better than me 😳

- Problem: How to bridge this chasm?

Options to Bridge the Chasm

- Standardize a visual representation for BPEL

    - Can be done as proven by the many graphical tools for BPEL

- But BPMN contains a bunch of constructs without BPEL counterparts

    - Corresponding BPEL extensions are required

    - Very time consuming! (see BPEL4People…)

    - But users ask for a solution asap!

- This was not practical

- Thus, way chosen is to make (significant subset of) BPMN 2.0 more compatible with BPEL

Most Important Additions of BPMN 2.0

- Clear operational semantics

- "Native" XML exchange format

- Event & exception handlers (BPEL compliant)

- Choreography extensions

… and:

- Pattern-based mapping to BPEL (subset of BPMN)

- Effectively, BPMN 2.0 contains a significant subset isomorphic to BPEL ≈ Visual rendering of BPEL!

The Price to Pay

- Complexity of BPMN 2.0 has increased significantly!
    - Well, ensuring executability is always complex!

- Only a <u>subset</u> of BPMN 2.0 can be mapped to BPEL in a straightforward manner
    - It is still possible to model BPMN processes that have no canonical representation in BPEL

- Some features have no runtime environment yet
    - E.g. corresponding domain is out of scope of BPEL at all
        - See BPEL4Chor
    - New middleware has to be developed

Resulting Viewpoints

Viewpoint 1:

- Subset of BPMN 2.0 is isomorphic to BPEL
  - This subset is canonically transformed to BPEL and can be executed in BPEL engines
- Corresponding subset is a visual layer on top of BPEL

Viewpoint 2:

- BPMN 2.0 is a process language with well-defined operational semantics imprinted by BPEL
- It is possible to build a BPMN 2.0 engine without the need to use a separate BPEL engine

Confusion Ahead!

- Now we have two process languages in the market supported by major vendors

- Very likely, we will be faced with the situation that we will have both, BPEL engines and BPMN engines in the market

- But what does it mean: *BPEL engine*? *BPMN engine*?

Major Building Blocks of a Process *Engine*

Buildtime

Modeling Tool

Process Model

Process Engine
(aka Workflow System)

Workflow/
Process
Database

Runtime

Applications,...

© Frank Leymann
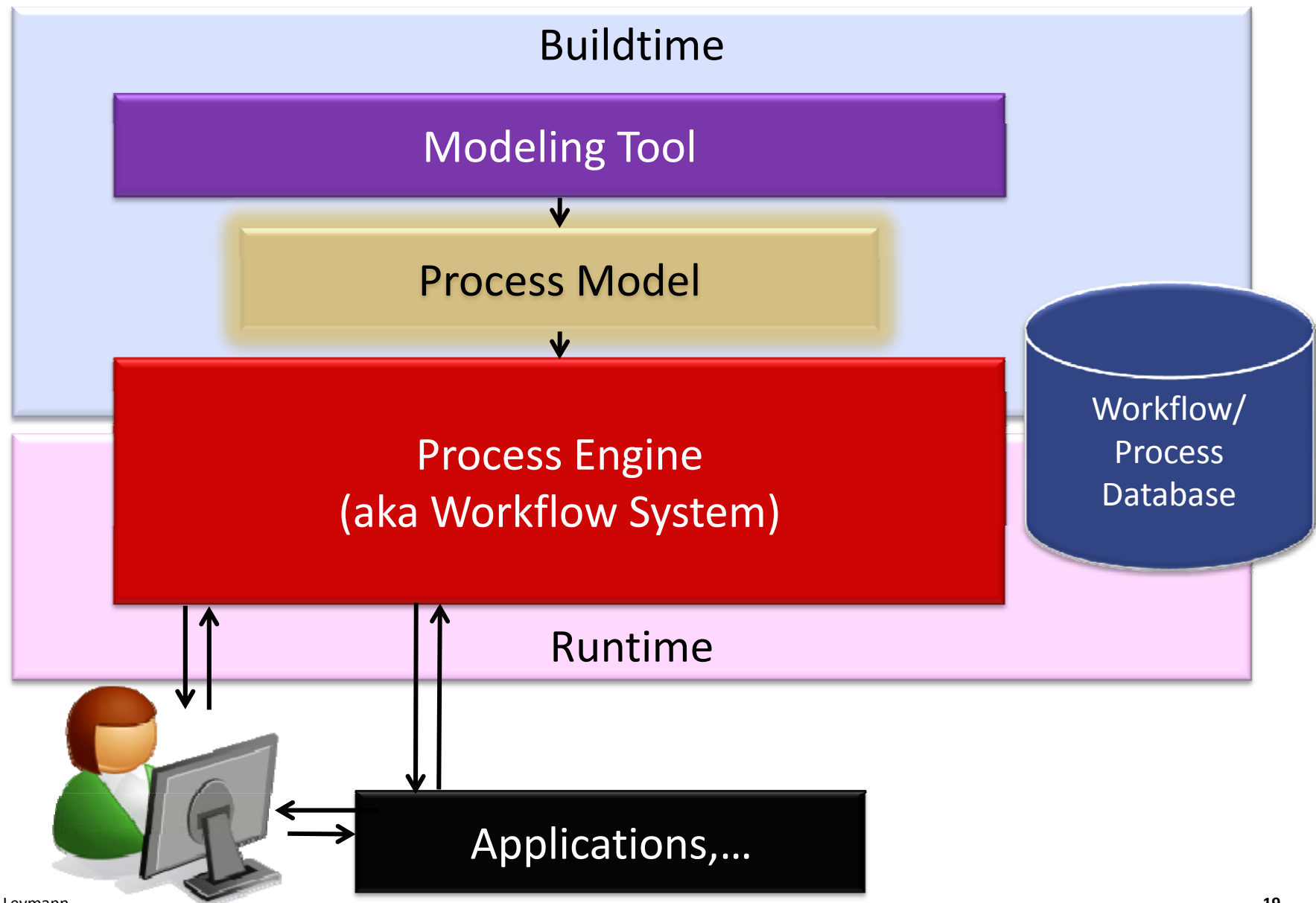
More Detailed View: The WfMC Reference Model

© Frank Leymann

Modern Terminology

© Frank Leymann
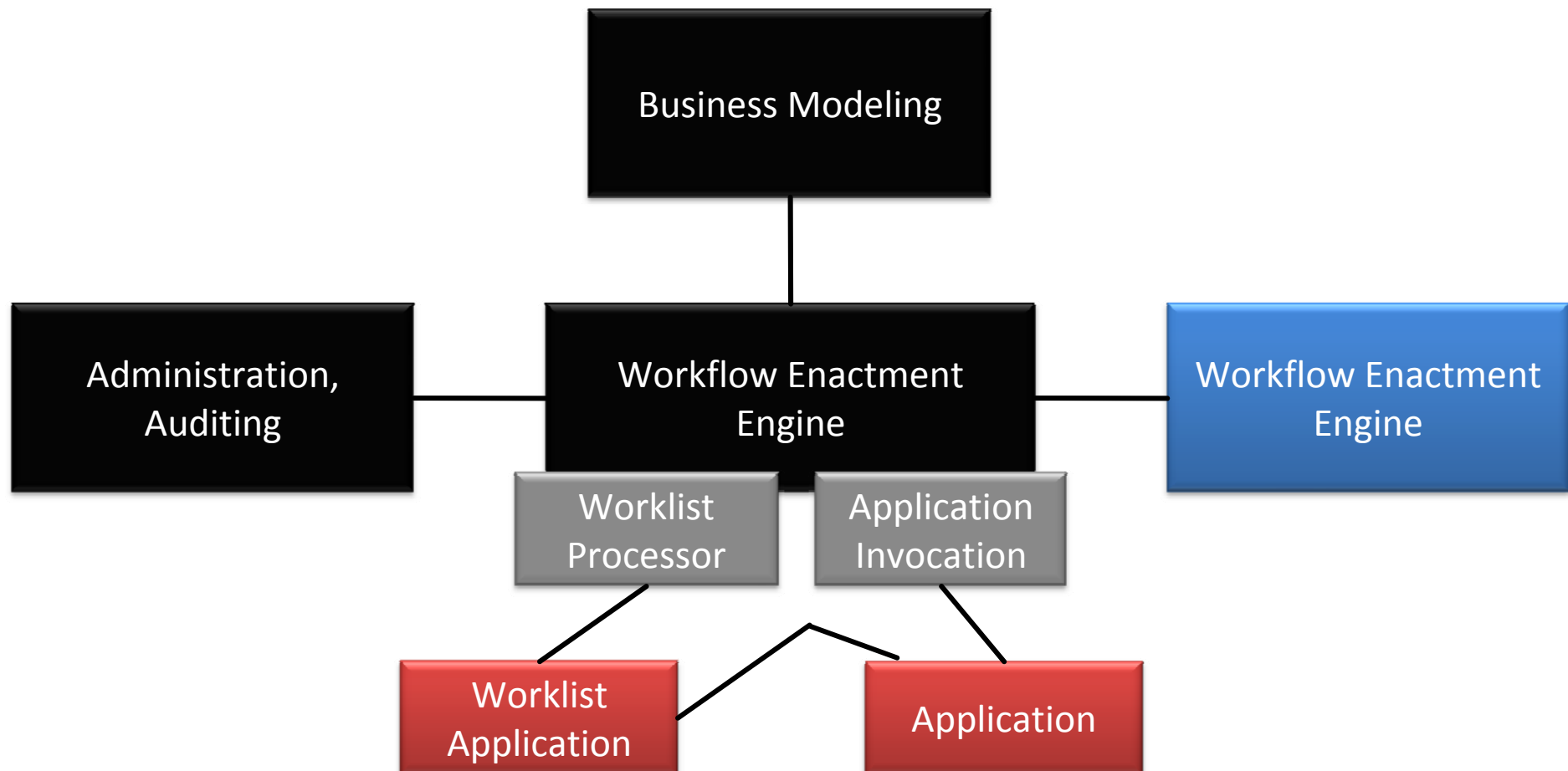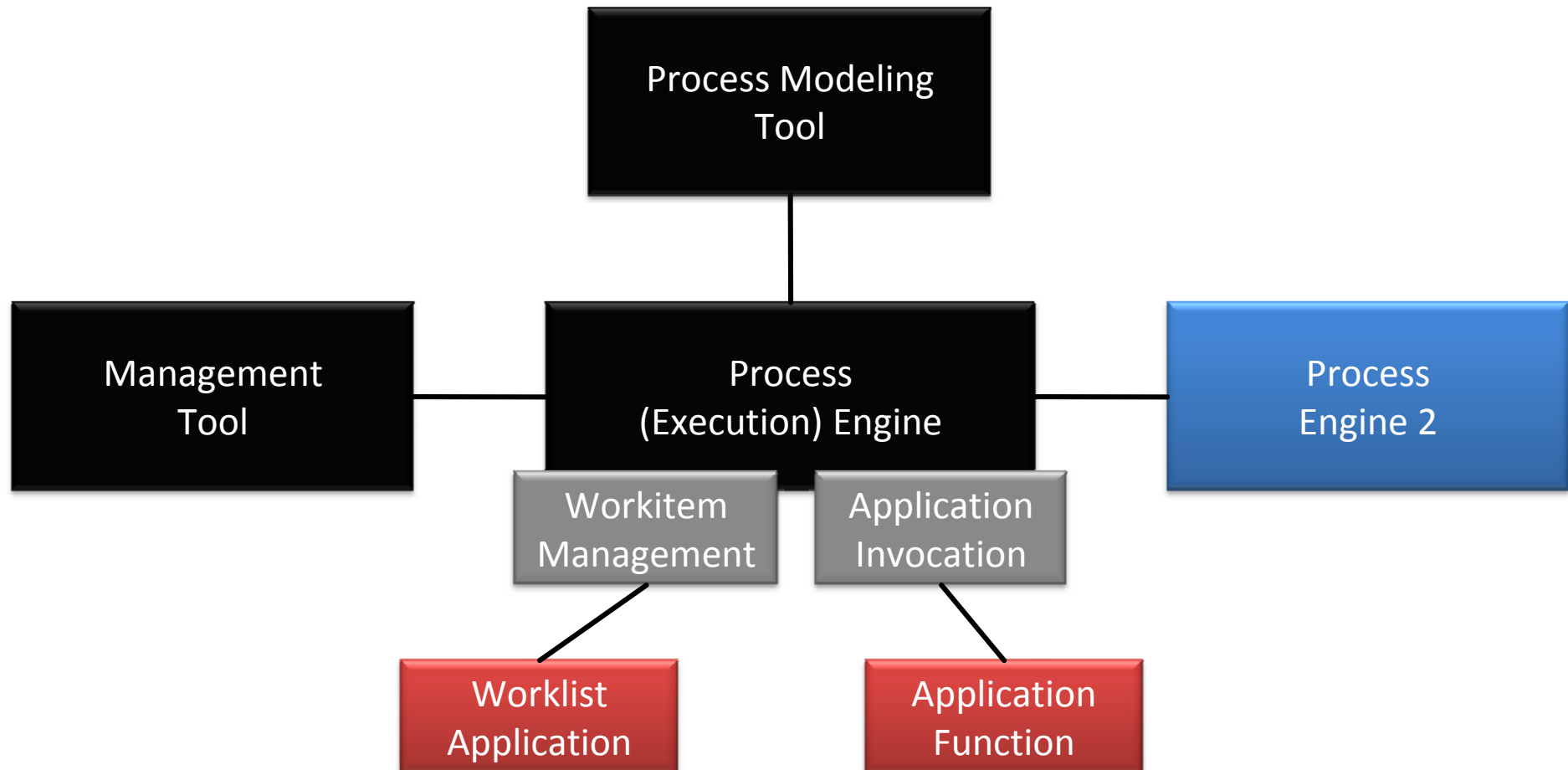
Yet More Modern Terminology

© Frank Leymann
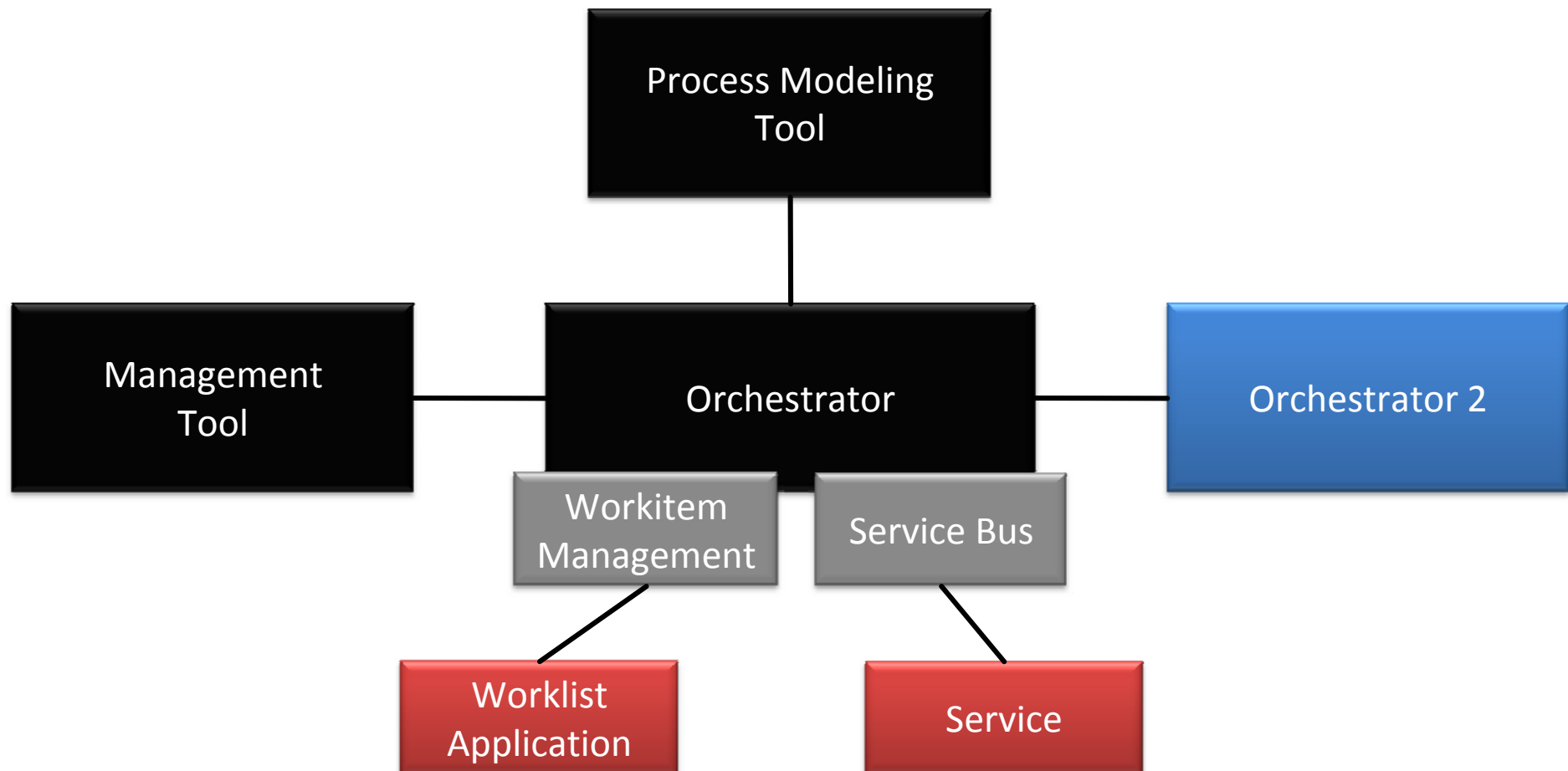
Architectural Impact of BPEL4People

WFMS

Navigator

Workitem Manager

Past Architecture

WFMS

Navigator

Task Manager
(aka Task Processor
aka
Human Task
Infrastructure)

Today's Architecture

© Frank Leymann
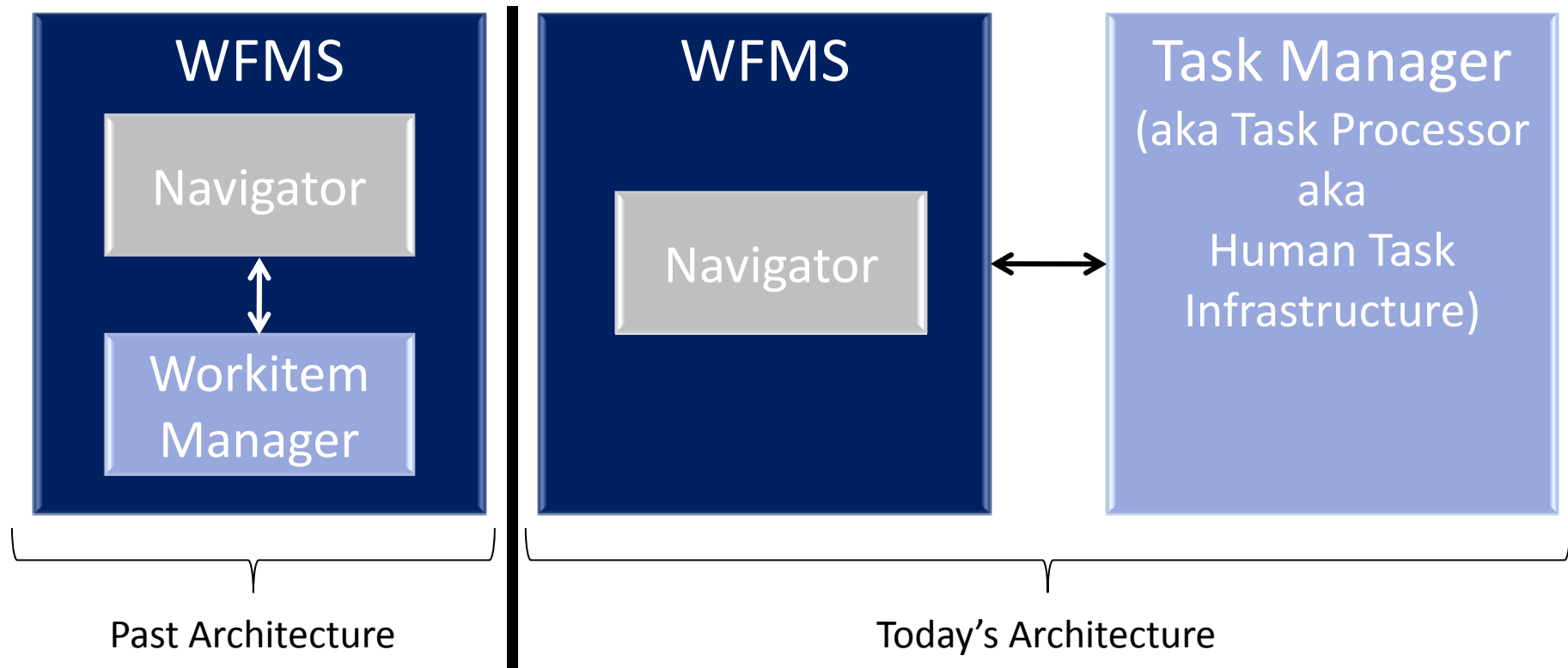
Architectural Impact: Refined View
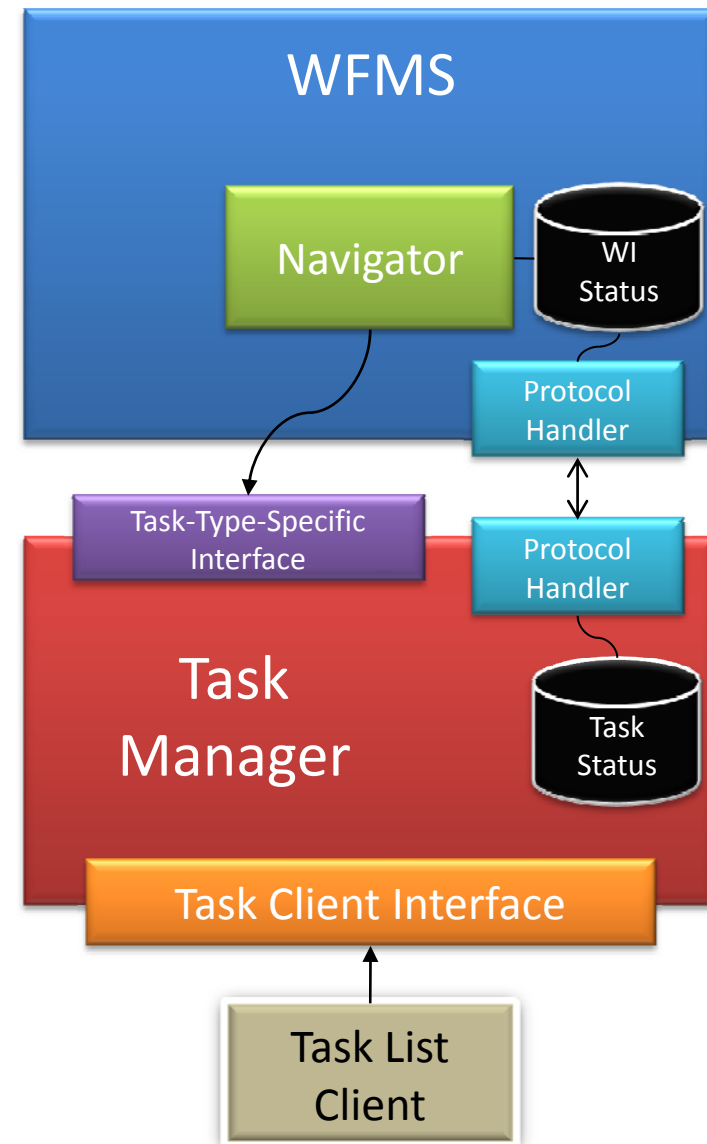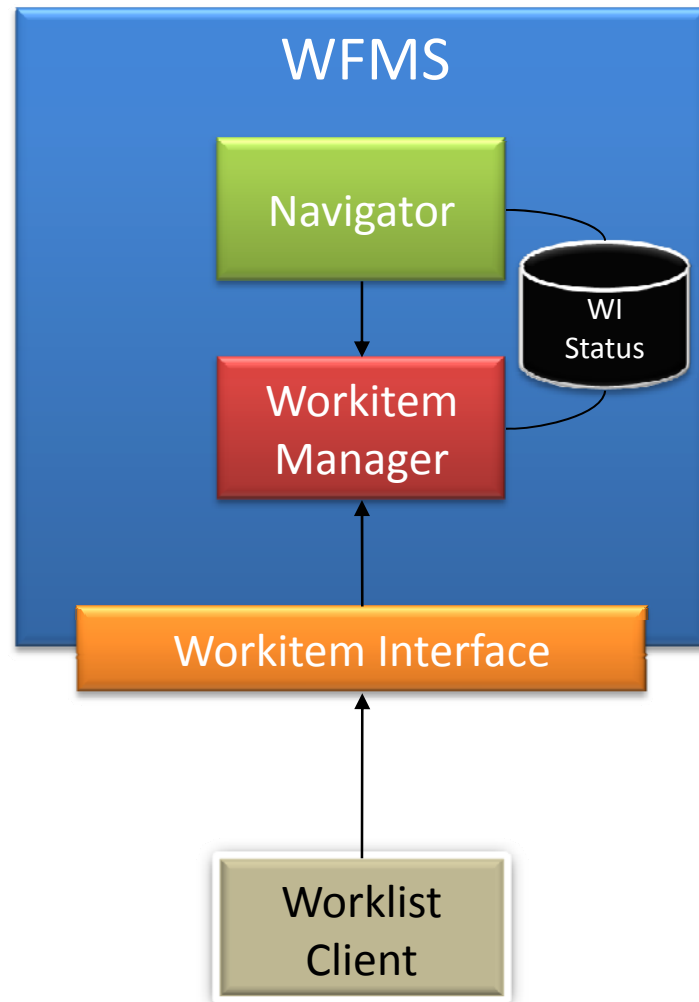
Resulting SOA-View on WfMC Reference Model

© Frank Leymann
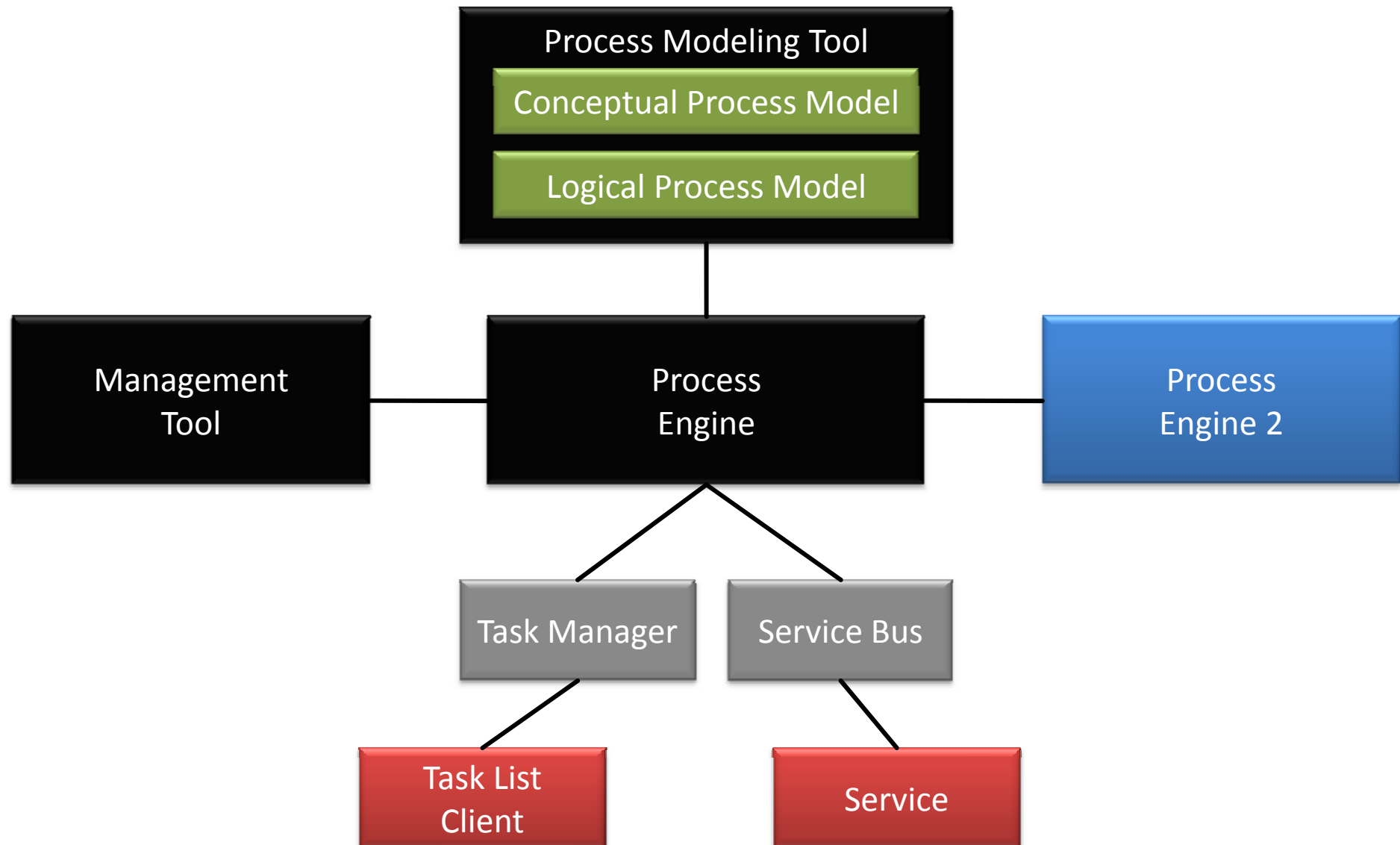
Process Modeling is Layered

Process Management Stack

Conceptual Process Model (BPMN)

↓ Transform

Logical Process Model (BPEL)

↓ Deploy

Process Engine (Internal Metamodel)

Data Management Stack

Conceptual Data Model (Entity-Relationship)

↓ Transform

Logical Data Model (SQL)

↓ Deploy

DBMS (Relational Datamodel)
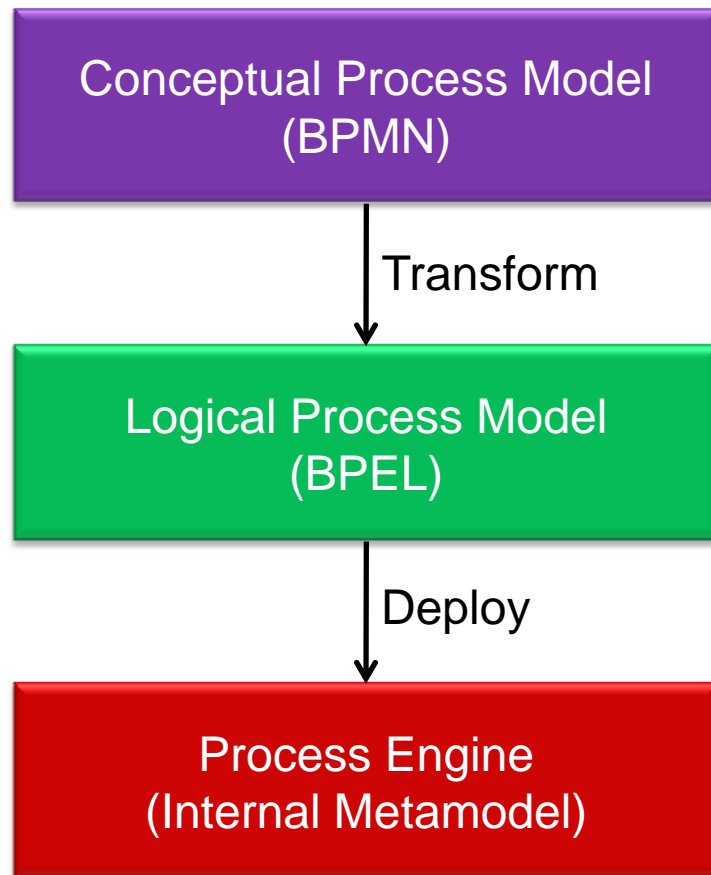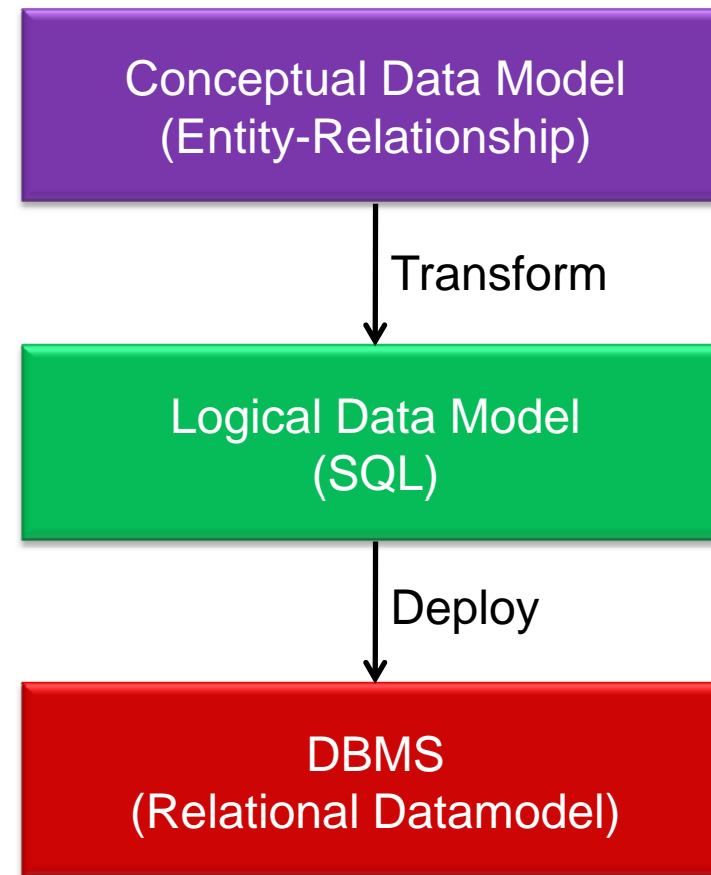
DBMS: What Happened in the Past

Conceptual Data Model (Object Model)

↓

Logical Data Model (SQL)

↓

DBMS (Relational Datamodel)

Data Model (Object Model)

↓

OODBMS (Object Model)

Conceptual Data Model (XSD)

↓

Logical Data Model (SQL)

↓

DBMS (Relational Datamodel)

Data Model (XSD)

↓

XML-DBMS (XSD)

Data Model (Object/XSD Model)

↓

Extended RDBMS (OR-XML Model)
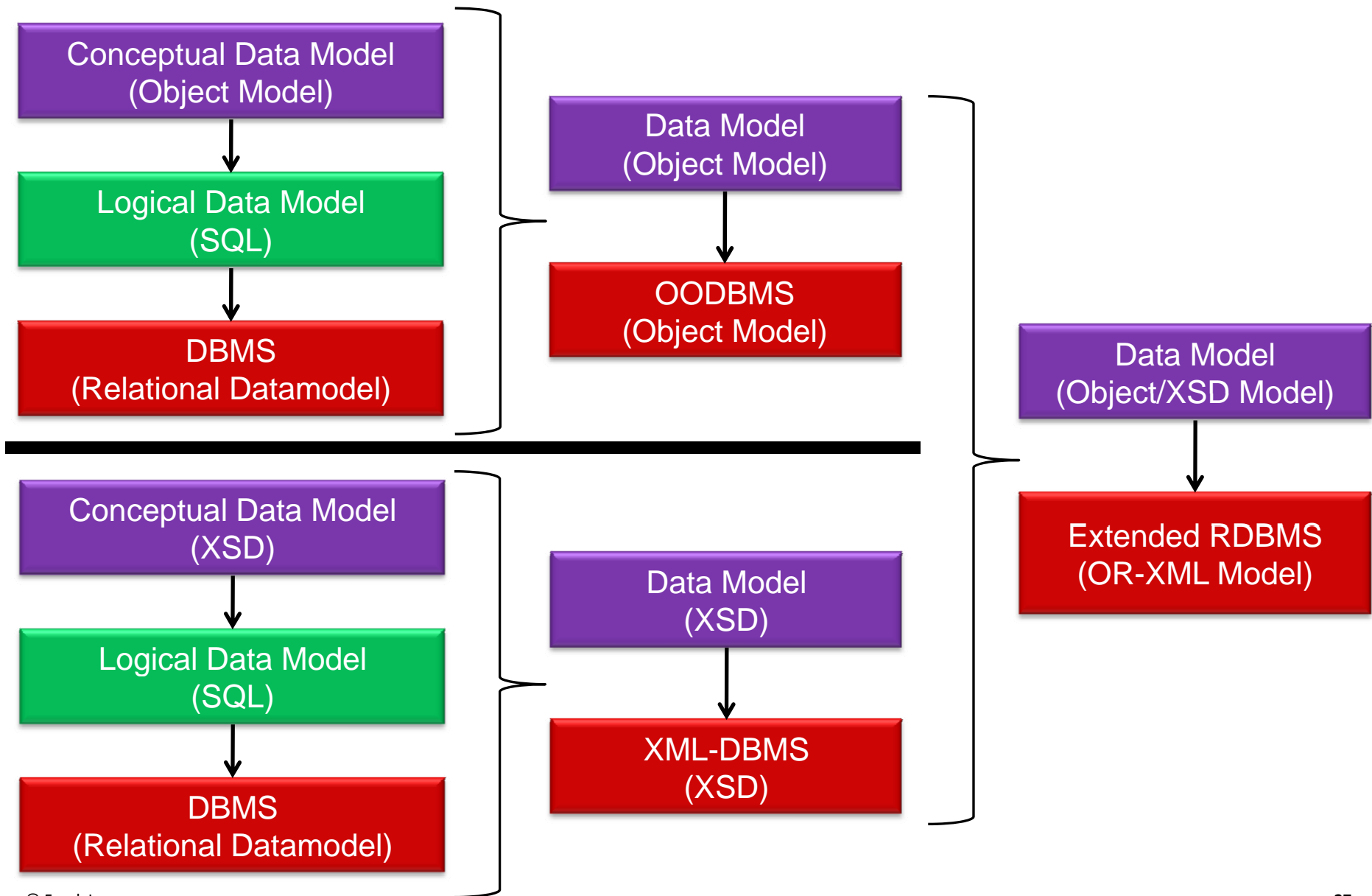
The Open Question

Will history repeat in the "Process World"?

Agenda

Historical Remarks

BPMN Origin and Positioning

The Notion of "Native Metamodel"

View on Process Engines

Selective BPMN 2.0 Features

Executable BPMN - BPEL Mapping

Summary

The Notion of a Native Metamodel

- What does it mean to be a BPEL (BPMN) engine?
- Well, the engine can import process models specified in BPEL (BPMN), instantiate, run, monitor,… them *according to the operational semantics of BPEL (BPMN)*
- Most often, during import BPEL (BPMN) artifacts will be mapped onto internal artifacts of the engine anyway:

> Most engines implement their own proprietary process metamodel
>
> (Native Metamodel)

- "Native" BPEL (BPMN) engines are rare exceptions!
  - *Native BPEL (BPMN)*:= BPEL (BPMN) is the internal metamodel

What Determines an Engine's Native Metamodel?

- What the engineers building the engine chose to ensure efficiency, scalability etc of the execution!

- Modeling language(s) to be supported at time of creation of first release of process engine has deep impact on its native metamodel

- Good engine engineers take care about generality and extensibility of native metamodel striving towards "straightforward" support of extensions of original modeling language(s) and even new modeling languages

Typically, users of a BPEL engine are not aware of the engine's native metamodel and how different it is from BPEL as a metamodel itself

The Role of Deployment
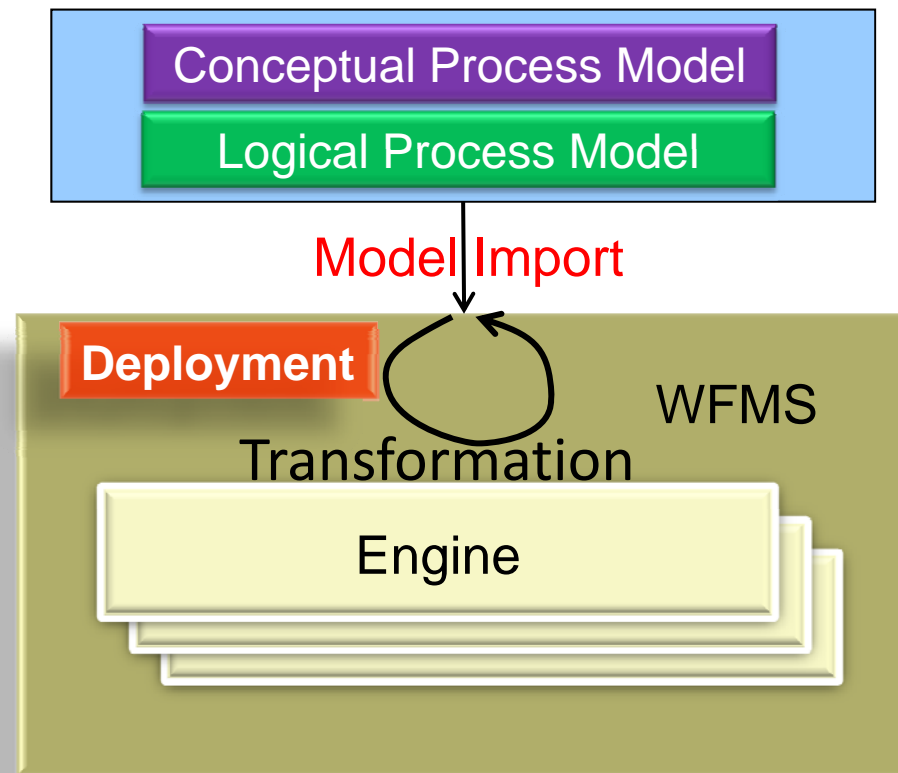
Deployment

- **Deployment** = Putting process model into production
  - I.e. make it ready for execution

- Deployment typically translates the corresponding model data into a different format, e.g.
  - Imported model might be specified in a metamodel different from what the WFMS supports directly
    - See next!

- ...plus many other things happen

Deployment: Key Aspect

```
<process …>
  <sequence>
    …
    <flow>
    …
    </flow>
  </sequence>
</process>
```

Deployment

Navigator

Model Cache

Navigation Logic

Instance Cache

. . .

Process Engine Database

Activity			
Name	**AID**	**JCond**	**...**

Control Link			
Source	**Target**	**TCond**	**...**

. . .

Activity Instance			
PIID	**AIID**	**AID**	**...**
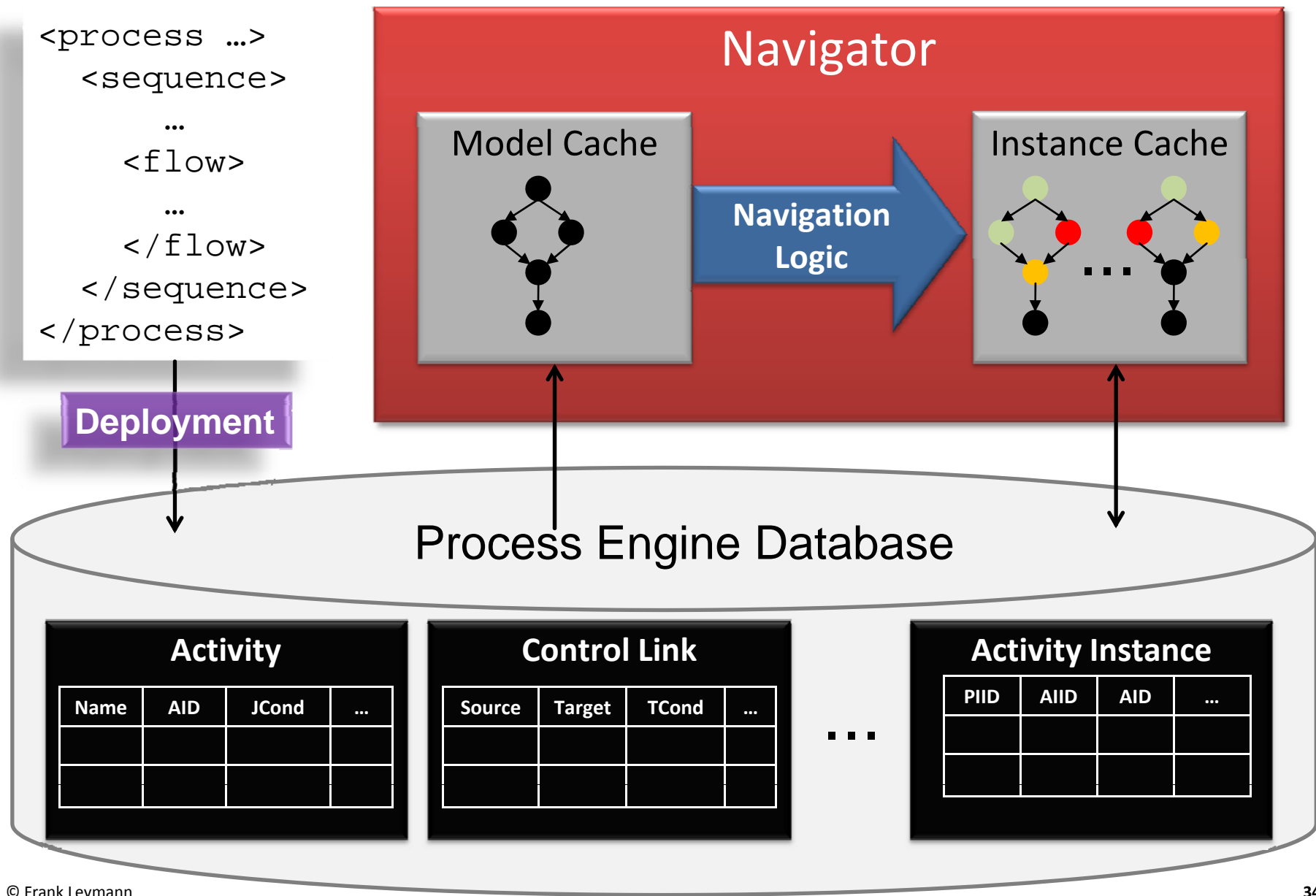
Native Support of a Metamodel

- Each workflow engine supports a particular metamodel **natively**

- Typically, the *native metamodel* supported is…

  - reflected in the database of the WFMS
    - database schema holds instances of metamodel constructs immediately
    - database schema is tuned towards supporting the navigator optimally

  - reflected in the state model of the WFMS
    - all metamodel constructs have a set of states and transitions between them
    - state model is reflected in monitoring model and audit trail

  - directly implemented in the navigator of the WFMS
    - navigator understands each metamodel construct directly, their states and valid transitions, the relation between the states of different artifacts
    - this implementation is "optimal"
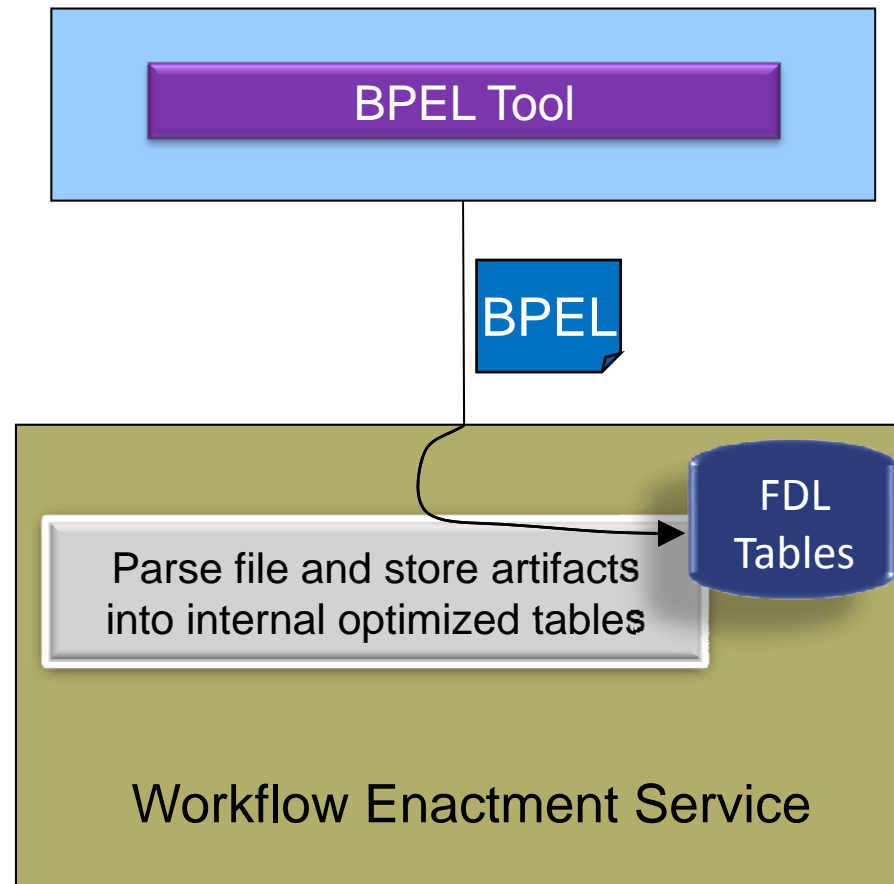
Transformation During Deployment: Example
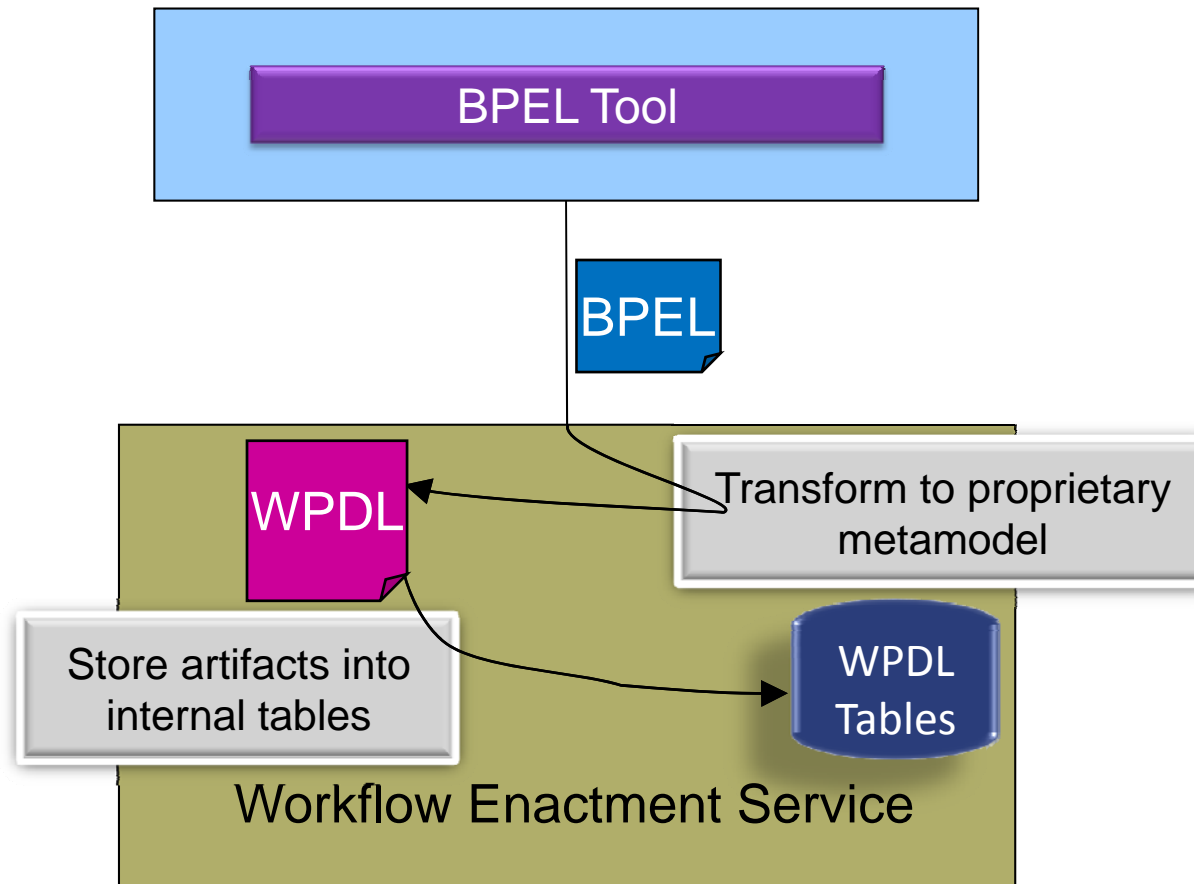
Transformation During Deployment: Example
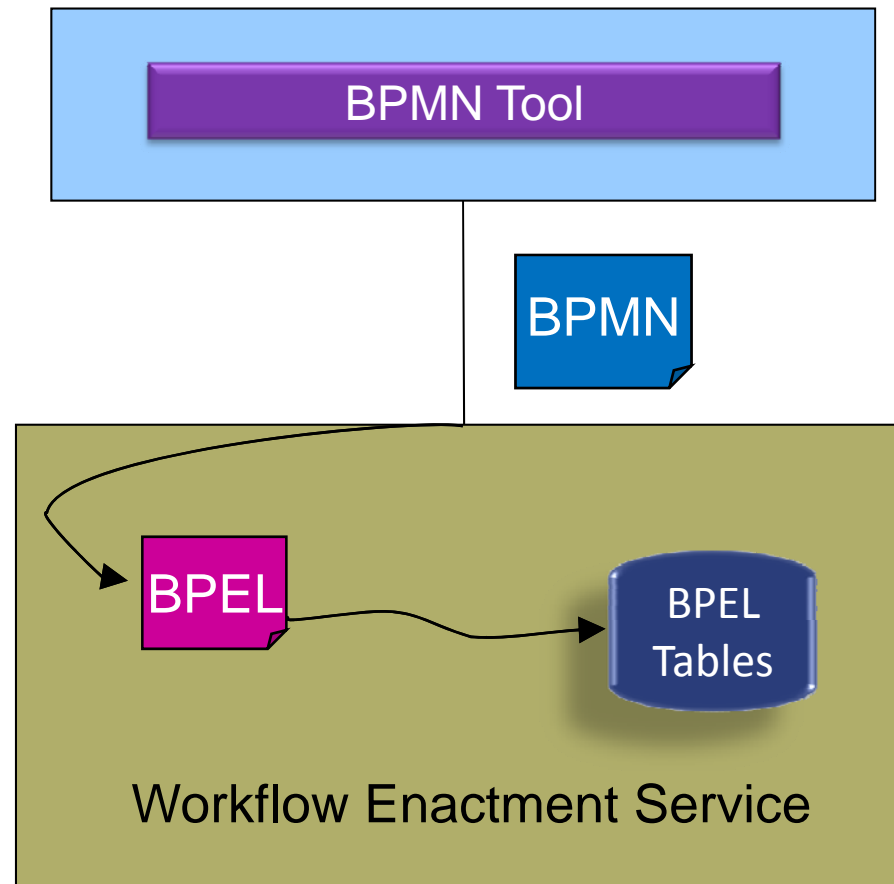
Model Transformations

- Process models specified in a different metamodel than the engine's native metamodel may be transformed into this native metamodel supported by a particular WFMS

- This way, a WFMS supporting WPDL natively may execute a BPEL model after corresponding transformation, for example

- This way, a WFMS supporting BPEL natively may execute a BPMN model after corresponding transformation, for example

Problems of Model Transformations

- Such transformations are <u>not</u> always <u>faithful</u>, i.e. the transformed model has different…

- …semantic behavior than the original model
    - E.g. behavior of the source model has to be "emulated" in the target model – if possible at all
        - E.g. BPEL exception behavior is hard to emulate

- …operational behavior than the original model
    - E.g. the navigator performs the transformed model less efficient than a WFMS supporting the metamodel of the original model
        - E.g. supporting FDL data flow in BPEL is cumbersome

Supporting One Metamodel Within Another

Metamodel $\mathfrak{M}^1$
(BPEL)

Metamodel $\mathfrak{M}^{1++}$
(BPEL^{++})

Metamodel $\mathfrak{M}^2$
(BPMN 2.0)

Activities
Control Flow
Data Flow
Choreos

Activities
Link
Flow

Activities
Link
Flow

Assign
Link

Inefficient!

dataLink

BPEL-D

???

Impossible!

Topology

BPEL4Chor
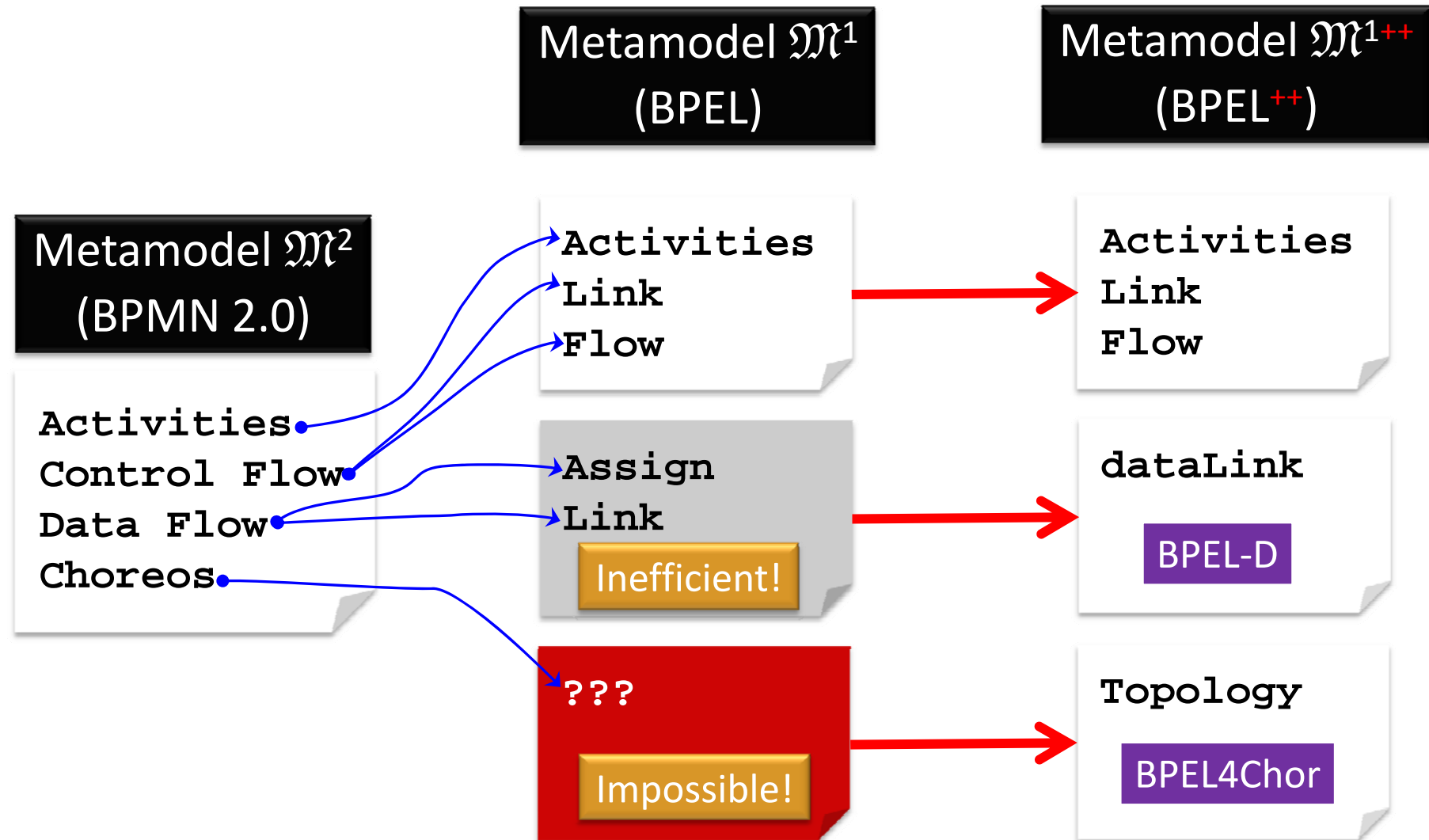
Extending a Given Metamodel

- Assume that models specified in a particular metamodel $\mathfrak{M}^2$ must be "best" supported in an engine with a different metamodel $\mathfrak{M}^1$

- Then, constructs of $\mathfrak{M}^2$ that are hard to emulate in $\mathfrak{M}^1$ must be identified and corresponding constructs must be added to $\mathfrak{M}^1$

- This is why BPEL is designed to be extensible, to allow new constructs to be added to BPEL to facilitate optimal mappings from different metamodels

- That way, an extended variant of a given engine (an $\mathfrak{M}^{1++}$ engine) may actually support process models specified in different metamodels $\mathfrak{M}^2, \mathfrak{M}^3, \ldots, \mathfrak{M}^n$

The Dawn of BPMN Engines

BPMN Tool

BPMN 2.0

BPMN 2.0 → BPMN Tables

Workflow Enactment Service
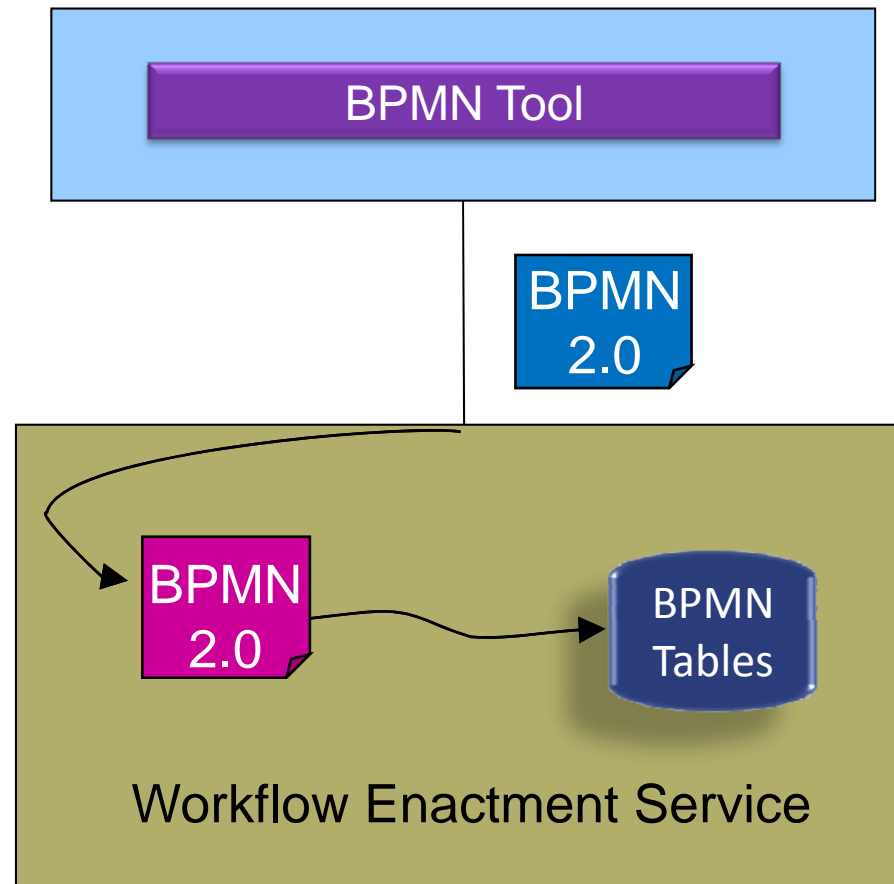
...and it Might Get Surprising ☺
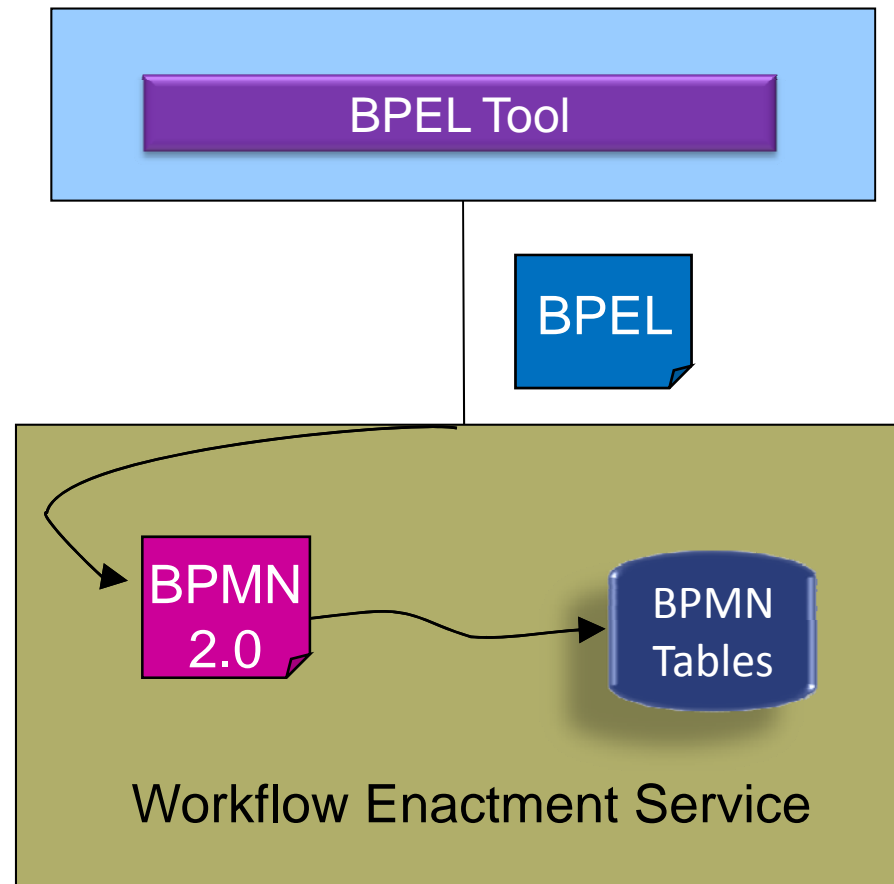
Agenda

Historical Remarks

BPMN Origin and Positioning

The Notion of "Native Metamodel"

View on Process Engines

Selective BPMN 2.0 Features

Executable BPMN - BPEL Mapping

Summary

Consequence For BPMN 2.0 Engines

- Whether the BPMN 2.0 process model is executed by a BPEL engine or by a *new BPMN 2.0 engine is not important*

- Native metamodel of a BPMN 2.0 engine is likely to be different from BPMN 2.0 as a metamodel anyhow

- Native metamodel of a BPMN 2.0 engine might be one that supports also BPEL today

- I.e. vendor of a BPEL engine that provides import capabilities for BPMN 2.0 process models might become a vendor of a BPMN 2.0 engine based on the very same execution engine

- It is absolutely valid that an intermediate BPEL process model from the BPMN 2.0 process model is generated before the actual import takes place
    - ...users might not even be aware of this intermediate BPEL format at all

What is Important About a Process Engine?

Far more important than the native metamodel of a process engine (BPMN 2.0 engine too) is its robustness, efficiency, scalability etc

- Providers of today's BPEL engines typically have invested a lot into these non-functional properties of their engines

- Thus, such vendors have the capabilities to also offer BPMN 2.0 engines with the very same non-functional properties

A DBMS Analogy

- In the past, new upcoming vendors implemented dedicated object oriented database systems OODBMS
  - The object model was natively supported by these DBMS

- But robustness, efficiency, scale,... was often lacking by such systems

- Over time established RDBMS vendors supported key constructs of the object paradigm while providing robustness, efficiency, scale,...

- OODBMS are no mainstream anymore

Extrapolation to Process Engines

- BPMN 2.0 is significantly more complex than BPEL
- Thus, vendors will subset BPMN 2.0 based on their customers' requirements

It is likely that no vendor will support all of BPMN 2.0 in its product

- BPEL engines will get extended over the time to support key features of BPMN 2.0 that are missing in BPEL and BPEL engines today
- BPEL engines will evolve to support relevant BPMN 2.0 features (those missing in BPEL today)
- So, relax and wait until today's process engines support missing BPMN features (but robust, efficient, scaling,…)

Agenda

Historical Remarks

BPMN Origin and Positioning

The Notion of "Native Metamodel"

View on Process Engines

Selective BPMN 2.0 Features

Executable BPMN - BPEL Mapping

Summary

How Execution is Enabled

- Many execution relevant information is only available via the BPMN 2.0 schema

- I.e. the graphical notation by far does not contain the information required to instantiate a BPMN process model and execute it

- The corresponding syntactical details are given in the BPMN 2.0 spec as UML Class Diagrams and XML Schema Definitions

    - But the given operational semantics does <u>not</u> match this level of detail
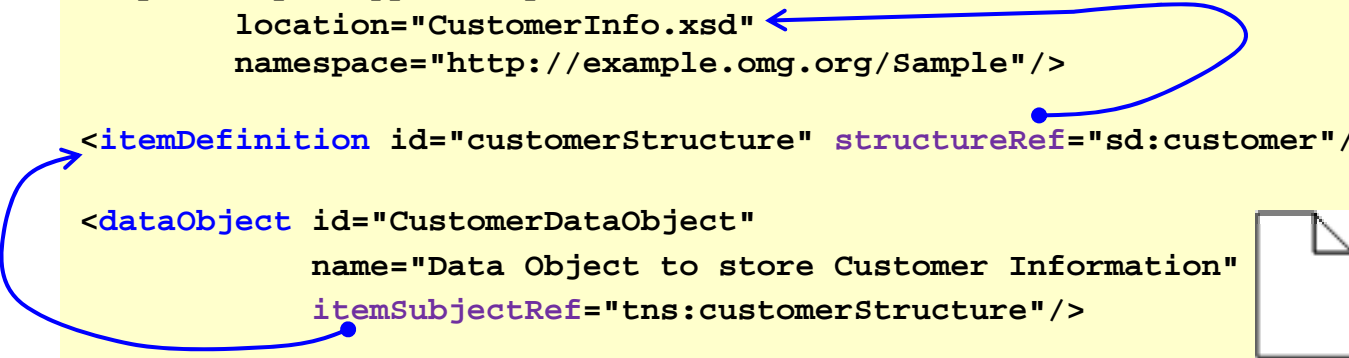
Data Context In Processes

```
<import importType="http://www.w3c.com/XMLschema"
        location="CustomerInfo.xsd"
        namespace="http://example.omg.org/Sample"/>

<itemDefinition id="customerStructure" structureRef="sd:customer"/>

<dataObject id="CustomerDataObject"
            name="Data Object to store Customer Information"
            itemSubjectRef="tns:customerStructure"/>
```

Messages In Processes

```
<itemDefinition id="tasksvcCustomerMessageStructure"
                structureRef="tasksvc:MyFirstTaskServiceRequestMessage"/>


<message id="myServiceCustomerMessage"
         name="MyFirstTaskService Request Message"
         structureRef="tns:tasksvcCustomerMessageStructure"/>
```

Mental Model About Task Input/Output


```
<serviceTask name="Task" id="Task" ... >
   ...
   <ioSpecification>
      <dataInput id="dataInput" .../>
      <dataOutput id="dataOutput" .../>
   </ioSpecification>

   <dataInputAssociation>
      <sourceRef>InputDataObject</sourceRef>
      <targetRef>dataInput</targetRef>
   </dataInputAssociation>

   <dataOutputAssociation>
      <sourceRef>dataOutput</sourceRef>
      <targetRef>OutputDataObject</targetRef>
   </dataOutputAssociation>

   ...
</serviceTask>
```
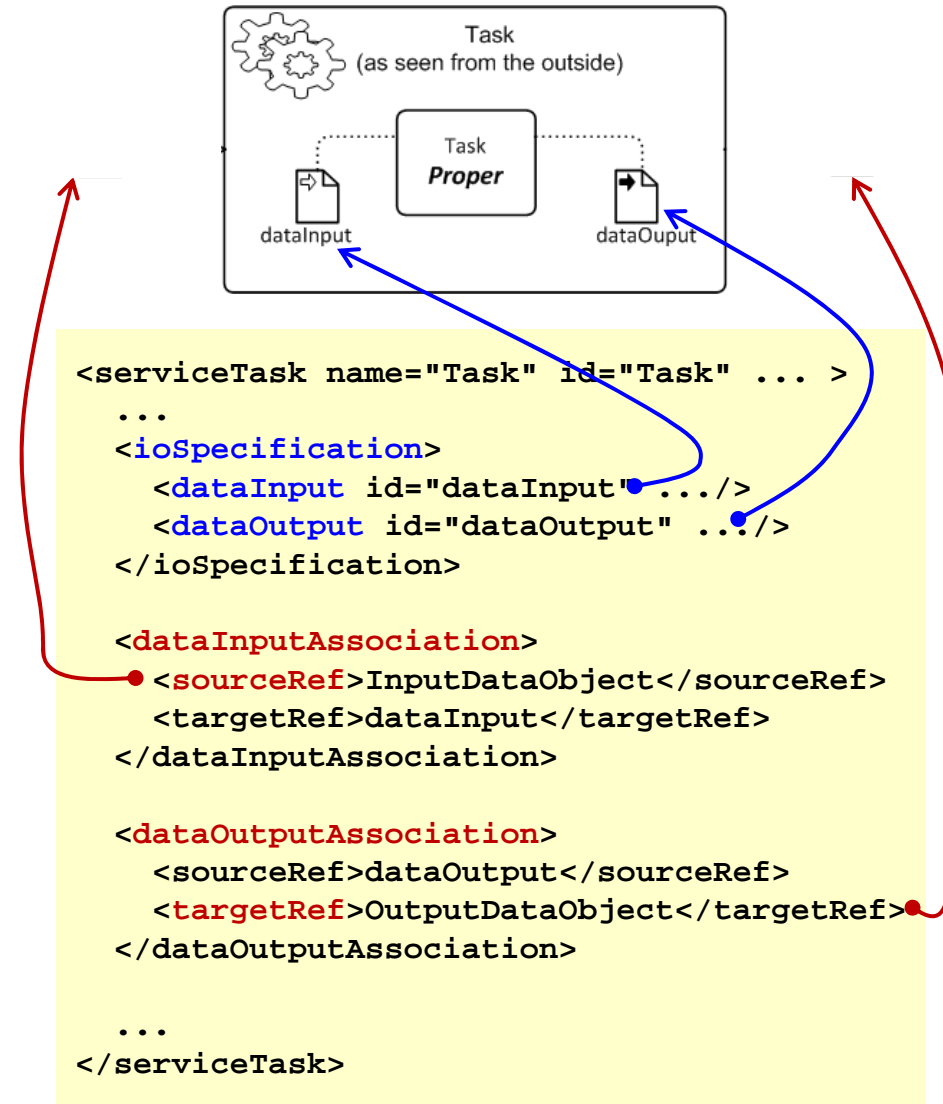
Transforming Data Objects to Task Data

```
<serviceTask name="myTask" id="myTask" ... >
  ...
  <ioSpecification>
    <dataInput id="myTaskInput" .../>
    <dataOutput id="myTaskOutput" .../>
  </ioSpecification>

  <dataInputAssociation>
    <assignment>
      <from>bpmn:getDataObject('InputDataObject')/XPath</from>
      <to>bpmn:getDataInput('myTaskInput')/XPath</to>
    </assignment>
    <sourceRef>InputDataObject</sourceRef>
    <targetRef>myTaskInput</targetRef>
  </dataInputAssociation>

  <dataOutputAssociation>
    <transformation> expression </transformation>
    <sourceRef>myTaskOutput</sourceRef>
    <targetRef>OutputDataObject</targetRef>
  </dataOutputAssociation>
  ...
</serviceTask>
```
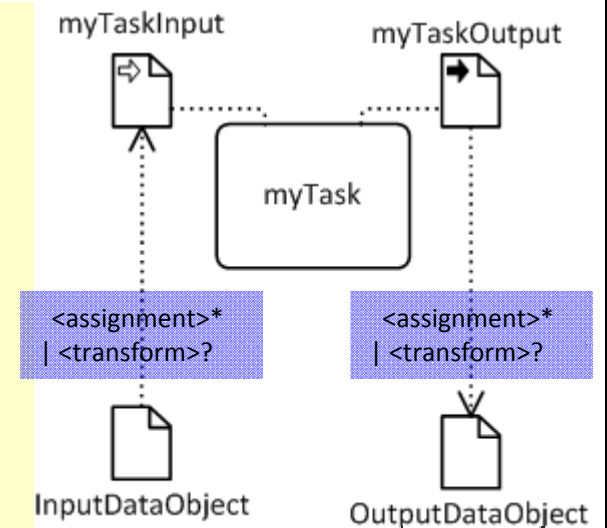
Summary: Important Relations For Input/Output


```
<serviceTask name="Task" id="Task" ... >
   ...
   <ioSpecification>
      <dataInput .../>
      <dataOutput .../>
   </ioSpecification>

   <dataInputAssociation>
      <sourceRef>...</sourceRef>
      <targetRef>...</targetRef>
   </dataInputAssociation>
   ...
</serviceTask>
```

.wsdl/msg   via itemDefinition

InputDataObject   Task (as seen from the outside)   OutputDataObject

Task Proper

dataInput   dataOuput
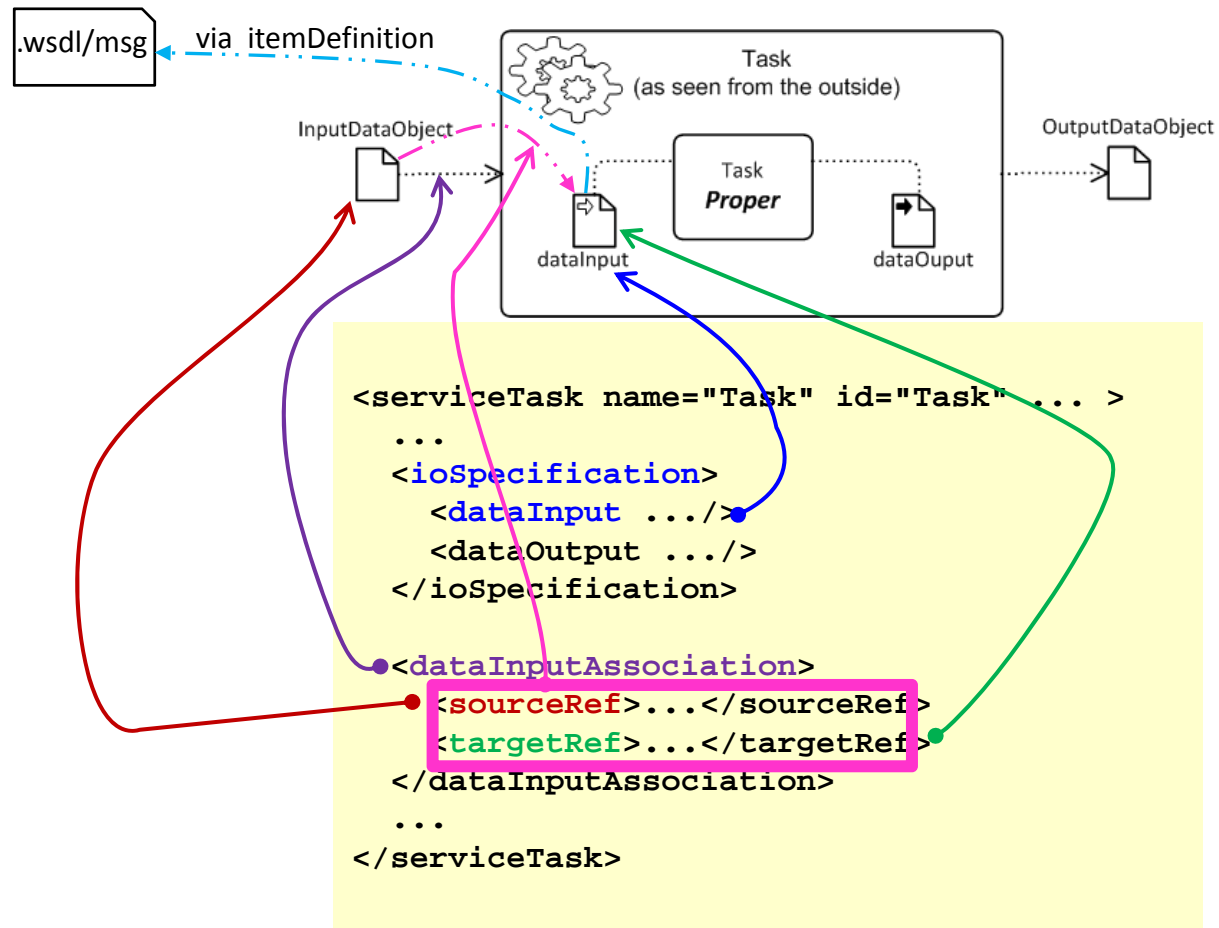
Defining Interfaces

Imported .wsdl (or .xsd or…)

3.
```
<itemDefinition id="ServiceInputMessageStructure"
                structureRef="fl:MyFirstTaskServiceRequestMessage"/>
<itemDefinition id="tasksvcResultMessageStructure"
                structureRef="fl:MyFirstTaskServiceResponseMessage"/>



<message id="myServiceInputMessage"
         name="My Request Message"
         structureRef="tns:ServiceInputMessageStructure"/>
<message id="myServiceResultMessage".../>

<interface id="MyInterface"
           name="MyInterface">
  <operation id="MyOperation"
             name="MyOperation">
    <inMessageRef>tns:myServiceInputMessage</inMessageRef>
     <outMessageRef>tns:myServiceResultMessage</outMessageRef>
  </operation>
</interface>
```

2.

1.

Service Tasks

```
<interface id="MyInterface"
            name="MyInterface">
  <operation id="MyOperation"
             name="MyOperation">
    <inMessageRef>tns:myServiceInputMessage</inMessageRef>
    <outMessageRef>tns:myServiceResultMessage</outMessageRef>
  </operation>
</interface>

<serviceTask id="MyTask" name="MyTask"
             operationRef="tns:MyOperation">
 <ioSpecification>
    <dataInput id="DataInput" name="Data to be send to service" .../>
    <dataOutput id="DataOutput" name="Result Output" .../>
    ...
  </ioSpecification>
  ...
</serviceTask>
```
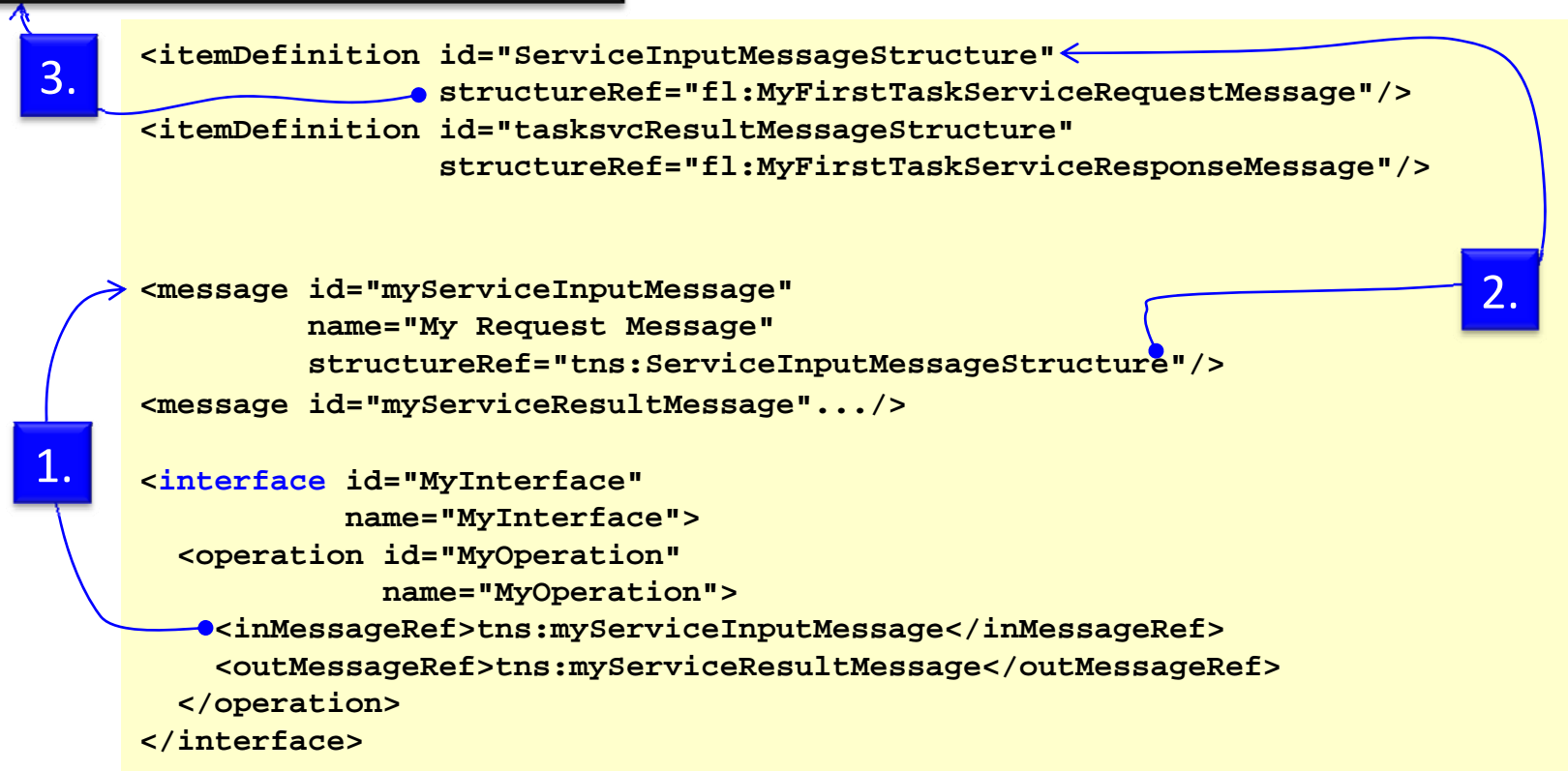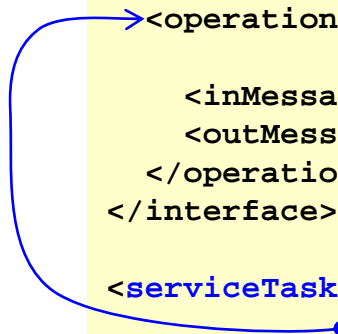
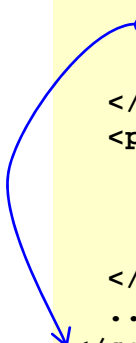# Early Binding of Interfaces

```
<process id="ServiceTaskProecss" name="My Process">

<partnerRole id="SvcProviderRole" name="Provider of Service Implementation"/>
    ...
<collaboration id="MyCollab" name="Service Task Collaboration">
    <participant id="ServiceProvidingPartner"
                name="Partner Providing Service Implementations">
    <partnerRolRef>tns:SvcProviderRole</partnerRoleRef>
    <interfaceRef>tns:BillingInterface</interfaceRef>
    <interfaceRef>tns:PaymentInterface</interfaceRef>
    <endPointRef>tns:MyBillingEndpoint</endPointRef>
    <endPointRef>tns:MyPaymentEndpoint</endPointRef>
  </participant>
  <participant id="OrderProcess"
               name="Process using services">
    <processRef>ServiceTaskProcess</processRef>
      ...
  </participant>
  ...
</collaboration>
<endPoint id="MyBillingEndpoint"> e.g. an EPR </endPoint>
<endPoint id="MyPaymentEndpoint"> e.g. an EPR </endPoint>
```
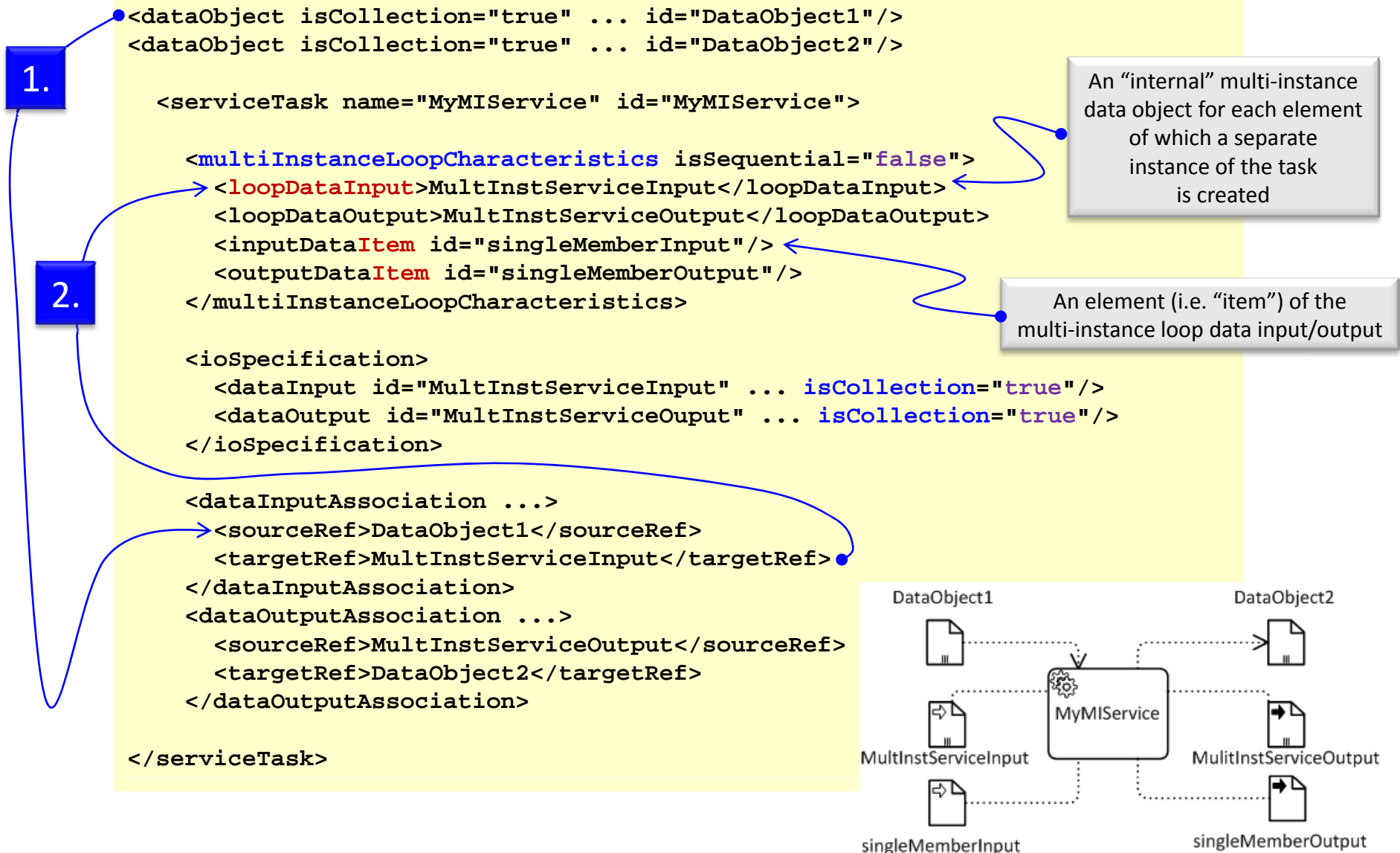
Endpoint of the implementation of the interface

Multi-Instance Task

```
<dataObject isCollection="true" ... id="DataObject1"/>
<dataObject isCollection="true" ... id="DataObject2"/>

  <serviceTask name="MyMIService" id="MyMIService">

    <multiInstanceLoopCharacteristics isSequential="false">
     <loopDataInput>MultInstServiceInput</loopDataInput>
      <loopDataOutput>MultInstServiceOutput</loopDataOutput>
      <inputDataItem id="singleMemberInput"/>
      <outputDataItem id="singleMemberOutput"/>
    </multiInstanceLoopCharacteristics>

    <ioSpecification>
      <dataInput id="MultInstServiceInput" ... isCollection="true"/>
      <dataOutput id="MultInstServiceOuput" ... isCollection="true"/>
    </ioSpecification>

    <dataInputAssociation ...>
     <sourceRef>DataObject1</sourceRef>
      <targetRef>MultInstServiceInput</targetRef>
    </dataInputAssociation>
    <dataOutputAssociation ...>
      <sourceRef>MultInstServiceOutput</sourceRef>
      <targetRef>DataObject2</targetRef>
    </dataOutputAssociation>

</serviceTask>
```

1.

2.

An "internal" multi-instance data object for each element of which a separate instance of the task is created

An element (i.e. "item") of the multi-instance loop data input/output

DataObject1   DataObject2

MultInstServiceInput   MultInstServiceOutput   MyMIService

singleMemberInput   singleMemberOutput

Agenda

Historical Remarks

BPMN Origin and Positioning

The Notion of "Native Metamodel"

View on Process Engines

Selective BPMN 2.0 Features

Executable BPMN – BPEL Mapping

Summary

Basic Principle

- A (recursive) function **[..]** is specified that allows to map a <u>subset</u> of BPMN to BPEL

- This map is specified by describing:
  1. [t] for all elementary BPMN tasks t that can be mapped to BPEL
  2. [e] for all elementary BPMN events e that can be mapped to BPEL
  3. [s] for all BPMN structures s that have a straight-forward mapping to BPEL

- This constructively defines the BPMN subset that can be mapped, as well as the associated map [..] itself

Mapping a Process
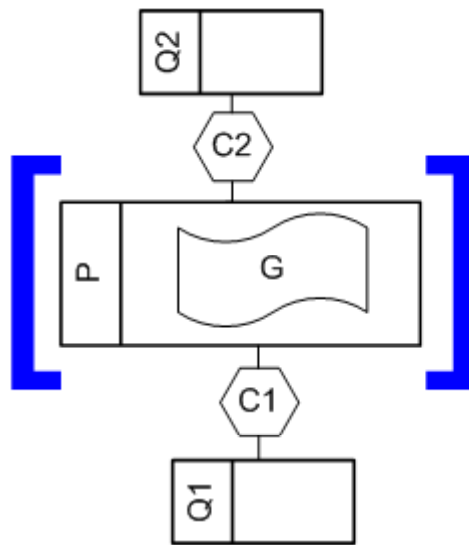

```
<process name="[P-name]"
    targetNamespace="[targetNamespace]"
    expressionLanguage="[expressionLanguage]"
    suppressJoinFailure="yes"
    xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/executable">
  <partnerLinks>
    [ {P-Interfaces} UNION {Qi-Interfaces} ]
  </partnerLinks>
  <variables>
    [ {dataObjects} UNION {properties} ]
  </variables>
  <correlationSets>
    [ {Ci-CorrelationKeys} ]
  </correlationSets>
  [G]
</process>
```

Mapping Data


```
[  D1  ]  =  <variable name="[D1-name]" type="[D1-structureDefinition]"/>
```

```
[ <inputSet name="iset">          ]
    <dataInput name="input1">
       <structureDefinition structure="type1"/>
    </dataInput>
    ...
  </inputSet>
```
=
```
<wsdl:message name="[iset-name]">
  <part name="[input1-name]" type="[type1]"/>
  ...
</wsdl:message>
```

```
[ <outputSet name="oset">          ]
    <dataOutput name="output1">
       <structureDefinition structure="type3"/>
    </dataOutput>
    ...
  </outputSet>
```
=
```
<wsdl:message name="[oset-name]">
  <part name="[output1-name]" type="[type3]"/>
  ...
</wsdl:message>
```

```
[ <property id="P1-id"          ]
    structureRef="P1-structureDefinition"/>
```
=
```
<variable name="[{container-name}.P1-id]"
    type="[P1-structureDefinition]"/>
```

Mapping Messages and Interfaces

```
[ <Message name="msg-name">
   <StructureDefinition typeLanguage=
       "http://www.w3.org/2001/XMLSchema">
    xmlSchema
   </StructureDefinition>
  </Message> ]
```

```
=   <wsdl:message name="[msg-name]">
      [xmlSchema]
    </wsdl:message>
```

```
[ <Interface name="if-name">
   <Operations>
    <Operation name="op1-name">
     <inMessageRef ref="msg1i-name"/>
     <outMessageRef ref="msg1o-name"/>
     <errorRef ref="error1a-name"/>

     ...
    </Operation>

    ...
   </Operations>
  </Interface> ]
```

```
=   <wsdl:portType name="[if-name]">
     <operation name="[op1-name]">
       <wsdl:input message="[msg1i-name]" />
       <wsdl:output message="[msg1o-name]" />
       <wsdl:fault name="[error1a-faultname]"
          message="[error1a-name]" />

       ...
     </operation>

     ...
    </wsdl:portType>
```

Mapping Correlations

```
<KeyBasedCorrelationSet name="c-set">
  <Key name="k-name1" type="k-type1"
       messageRef="msg-name1">
    <MessageKeyExpression
       expressionLanguage="lang1">
      expr1
    </MessageKeyExpression>
  </Key>

  ...
  <Key name="k-nameN" />

    ...
</KeyBasedCorrelationSet>
```

=

```
<vprop:property name="[k-name1]" type="[k-type1]"/>
...
<vprop:property name="[k-nameN]" />

<vprop:propertyAlias propertyName="[kName1]"
    messageType="[msg-name1]"
    part="[expr1-part]">
  <vprop:query queryLanguage="[lang1]">
    [expr1]
  </vprop:query>
</vprop:propertyAlias>
...
<vprop:propertyAlias propertyName="[kNameN]" />

<correlationSets>
  <correlationSet name="[c-set]"
      properties="[k-name1] ... [k-nameN]"/>
  ...
</correlationSets>
```
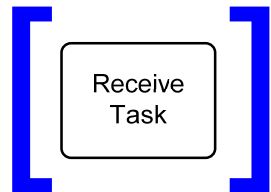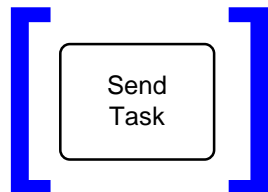
Mapping Tasks

Service Task

=

```
<invoke name="[Task-name]"
  partnerLink="[Q, Task-operation-interface]"
  portType="[Task-operation-interface]"
  operation="[Task-operation]">
  <correlations>
    <correlation set="[Task-messageFlow-conversation-correlationKey]"
        initiate="[initialInConversation? 'join':'no']"/>
  </correlations>
</invoke>
```
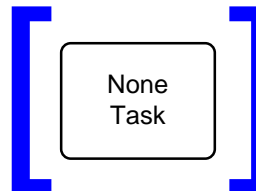
Receive Task

=

```
<receive name="[Task-name]"
  createInstance="[instantiate? 'yes':'no']"
  partnerLink="[Task-serviceRef]"
  portType="[Task-operation-interface]"
  operation="[Task-operation]">
</receive>
```
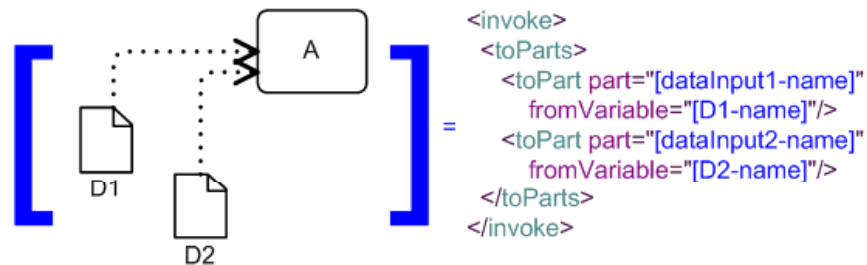
Send Task

=

```
<invoke name="[Task-name]"
  partnerLink="[Task-serviceRef]"
  portType="[Task-operation-interface]"
  operation−"[Task-operation]">
</invoke>
```

None Task

=

```
<empty name="[Task-name]">
</empty>
```

Mapping Task Input/Output

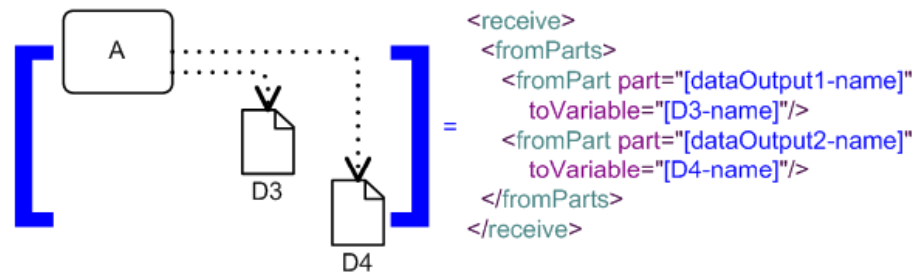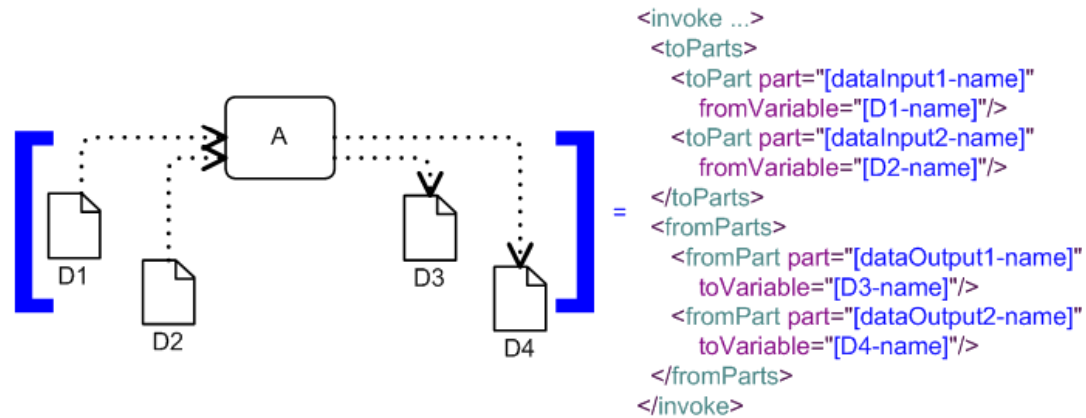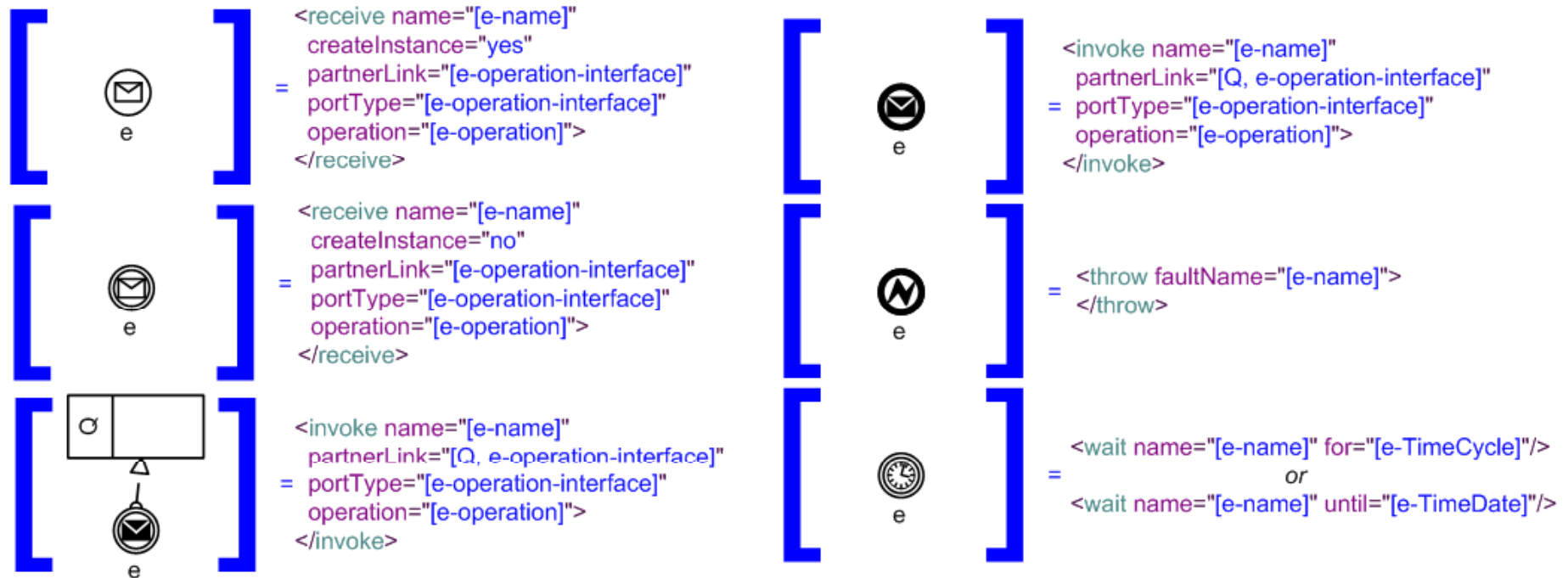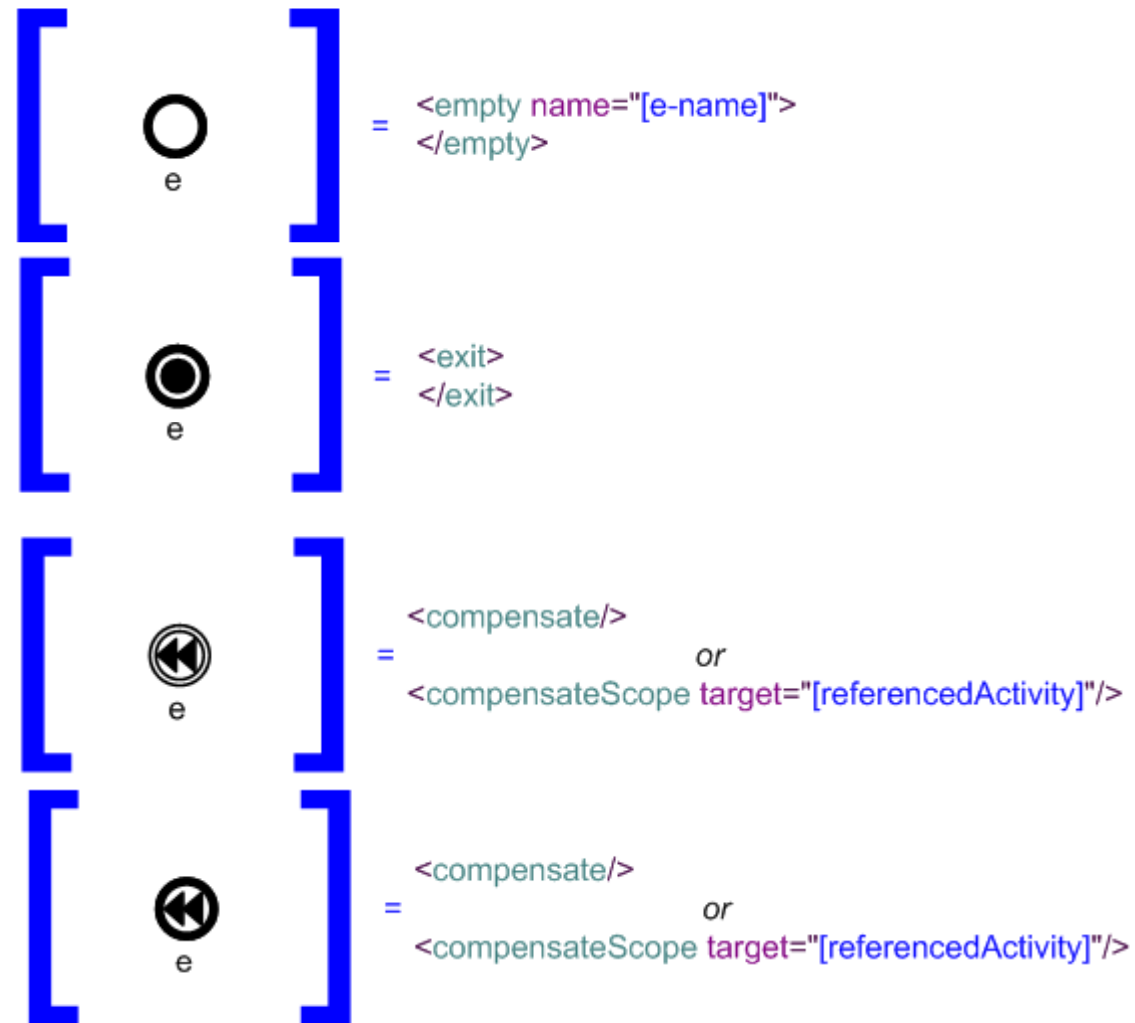

```
<invoke ...>
  <toParts>
    <toPart part="[dataInput1-name]"
      fromVariable="[D1-name]"/>
    <toPart part="[dataInput2-name]"
      fromVariable="[D2-name]"/>
  </toParts>
  <fromParts>
    <fromPart part="[dataOutput1-name]"
      toVariable="[D3-name]"/>
    <fromPart part="[dataOutput2-name]"
      toVariable="[D4-name]"/>
  </fromParts>
</invoke>
```

```
<receive>
  <fromParts>
    <fromPart part="[dataOutput1-name]"
      toVariable="[D3-name]"/>
    <fromPart part="[dataOutput2-name]"
      toVariable="[D4-name]"/>
  </fromParts>
</receive>
```

```
<invoke>
  <toParts>
    <toPart part="[dataInput1-name]"
      fromVariable="[D1-name]"/>
    <toPart part="[dataInput2-name]"
      fromVariable="[D2-name]"/>
  </toParts>
</invoke>
```
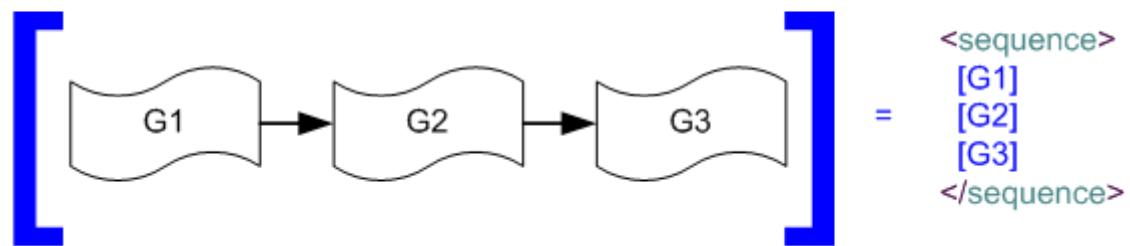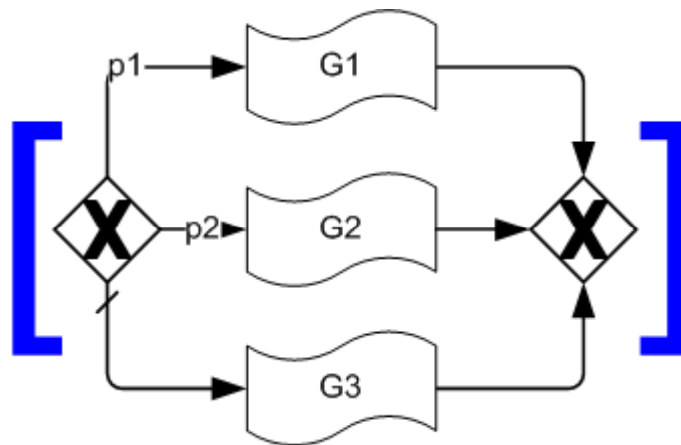
[✉ e] = ```
<receive name="[e-name]"
  createInstance="yes"
  partnerLink="[e-operation-interface]"
  portType="[e-operation-interface]"
  operation="[e-operation]">
</receive>
```

[ ✉ e ] = ```
<receive name="[e-name]"
  createInstance="no"
  partnerLink="[e-operation-interface]"
  portType="[e-operation-interface]"
  operation="[e-operation]">
</receive>
```

[↻ / ✉ e] = ```
<invoke name="[e-name]"
  partnerLink="[Q, e-operation-interface]"
  portType="[e-operation-interface]"
  operation="[e-operation]">
```

[ ✉ e ] = ```
<invoke name="[e-name]"
  partnerLink="[Q, e-operation-interface]"
  portType="[e-operation-interface]"
  operation="[e-operation]">
</invoke>
```

[Ⓝ e] = ```
<throw faultName="[e-name]">
</throw>
```

[ 🕐 e ] = ```
<wait name="[e-name]" for="[e-TimeCycle]"/>
            or
<wait name="[e-name]" until="[e-TimeDate]"/>
```

[○ e] = `<empty name="[e-name]">`
`</empty>`

[◉ e] = `<exit>`
`</exit>`

[◉ e] = `<compensate/>`
or
`<compensateScope target="[referencedActivity]"/>`

[◉ e] = `<compensate/>`
or
`<compensateScope target="[referencedActivity]"/>`

Mapping Sequences


```
<sequence>
  [G1]
  [G2]
  [G3]
</sequence>
```

Mapping If-Then-Else


```
<if><condition>[p1]</condition>
  [G1]
<elseif><condition>[p2]</condition>
  [G2]
</elseif>
<else>
  [G3]
</else>
</if>
```
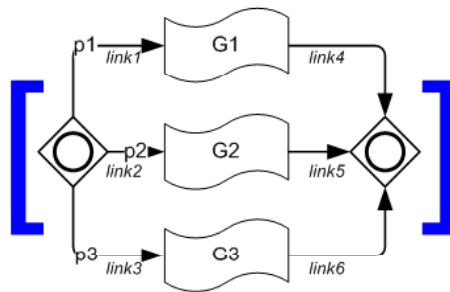
Mapping Pick
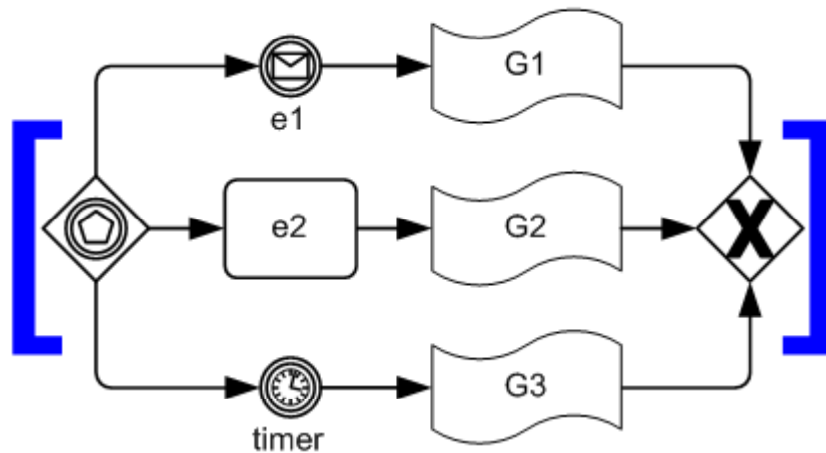

```
<pick createInstance="[instantiate? 'yes':'no']">
 <onMessage partnerLink="[e1-operation-interface]"
    operation="[e1-operation]">
   [G1]
 </onMessage>
 <onMessage partnerLink="[e2-operation-interface]"
    operation="[e2-operation]">
   [G2]
 </onMessage>
 <onAlarm>
   [timer-spec]
   [G3]
 </onAlarm>
</pick>
```

Mapping Flow


```
<flow>
 <links>
  <link name="[link1]"/>
   ...
  <link name="[link6]"/>
 </links>

 <empty>
  <sources>
   <source linkName="[link1]">
    <transitionCondition>[p1])</transitionCondition>
   </source>
   <source linkName="[link2]">
    <transitionCondition>[p2])</transitionCondition>
   </source>
   <source linkName="[link3]">
    <transitionCondition>[p3])</transitionCondition>
   </source>
  </sources>
 </empty>

 <flow>
  <targets><target linkName="[link1]"/></targets>
  <sources><source linkName="[link4]"/></sources>
  [G1]
 </flow>

 <flow>
  <targets><target linkName="[link2]"/></targets>
  <sources><source linkName="[link5]"/></sources>
  [G2]
 </flow>

 <flow>
  <targets><target linkName="[link3]"/></targets>
  <sources><source linkName="[link6]"/></sources>
  [G3]
 </flow>

 <empty>
  <targets>
   <target linkName="[link4]"/>
   <target linkName="[link5]"/>
   <target linkName="[link6]"/>
  </targets>
 </empty>
</flow>
```
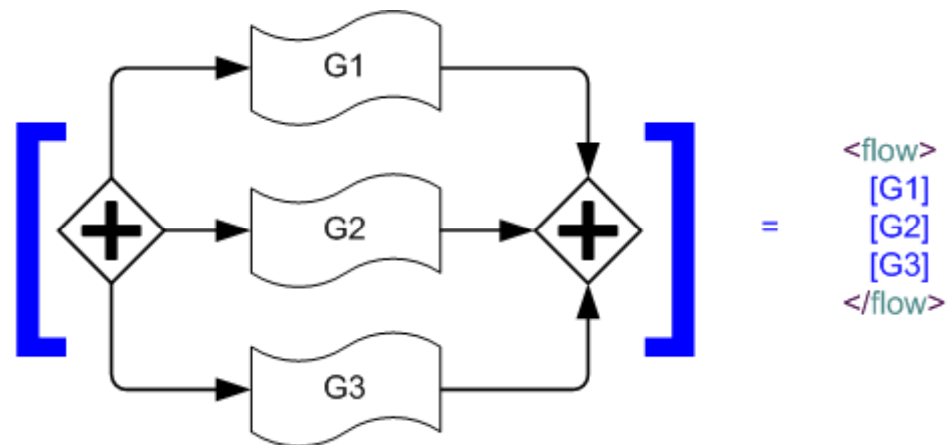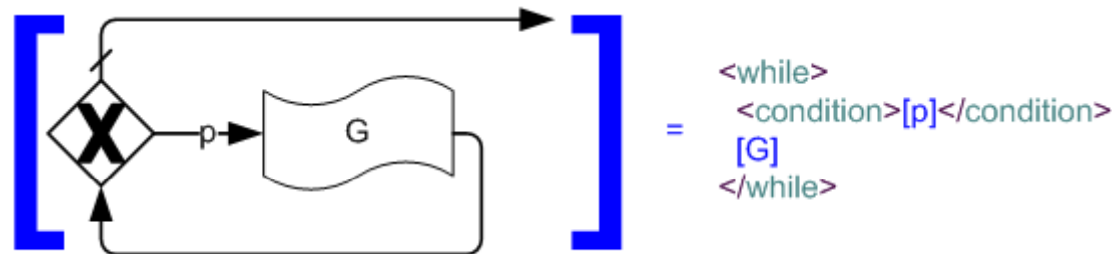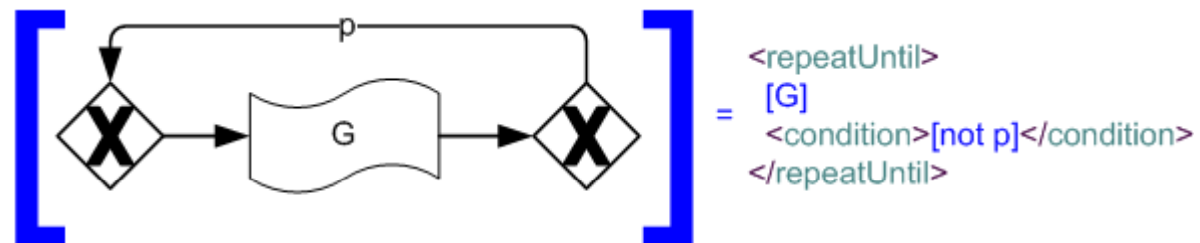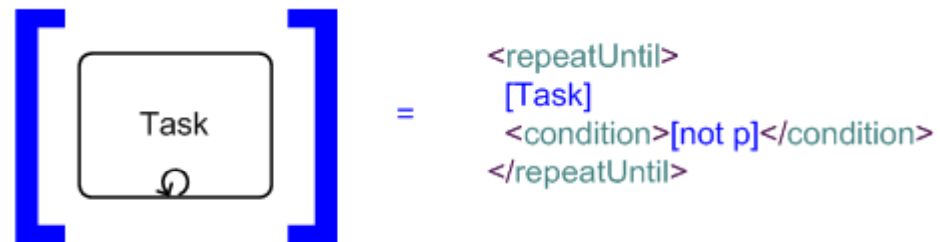
Mapping Parallel

$$= \begin{array}{l} \texttt{<flow>} \\ \quad [\text{G1}] \\ \quad [\text{G2}] \\ \quad [\text{G3}] \\ \texttt{</flow>} \end{array}$$

Mapping While-Loops


```
<while>
 <condition>[p]</condition>
 [Task]
</while>
```



```
<while>
 <condition>[p]</condition>
 [G]
</while>
```

Mapping Until-Loops


```
            [  [ Task ↺ ]  ]  =    <repeatUntil>
                                     [Task]
                                      <condition>[not p]</condition>
                                   </repeatUntil>
```



```
      ┌────── p ──────┐
    [ X ──→ G ──→ X ]  =    <repeatUntil>
                              [G]
                               <condition>[not p]</condition>
                            </repeatUntil>
```
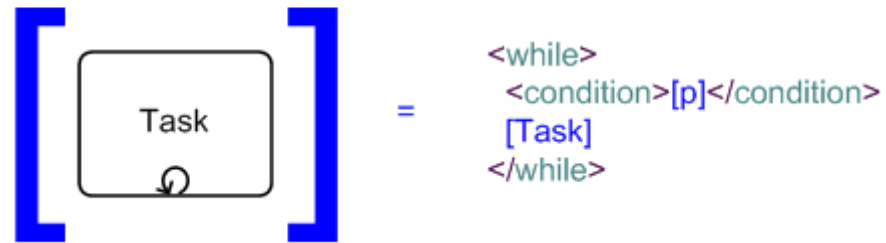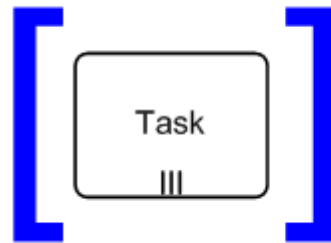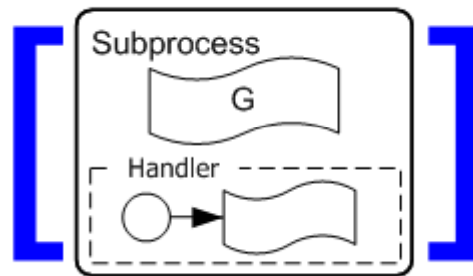
Mapping Multi-Instance-Loop-Characteristics


```
                              <variable name="[counter]" type="xsd:integer"/>
                              ...
  ┌─┐ ┌─────────┐ ┌─┐        <forEach counterName="[counter]" parallel="[isSequential? 'no':'yes']">
  │ │ │  Task   │ │ │           <startCounterValue>1</startCounterValue>
  │ │ │         │ │ │  =        <finalCounterValue>[condition]</finalCounterValue>
  │ │ │   III   │ │ │           <scope>
  └─┘ └─────────┘ └─┘              [Task]
                                 </scope>
                              </forEach>
```

Mapping Subprocesses


```
<scope name="[Subprocess-name]">
 <partnerLinks>
   [ {serviceRefs} ]
 </partnerLinks>
 <variables>
   [ {dataObjects} UNION {properties} ]
 </variables>
 <correlationSets>
   [ {correlations} ]
 </correlationSets>
 [Handler]
 [G]
</scope>
```
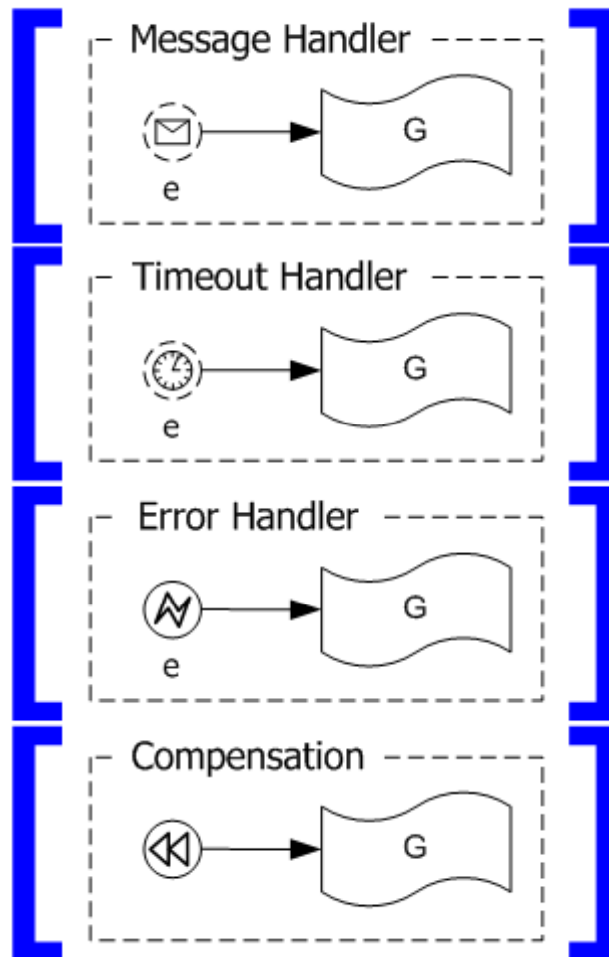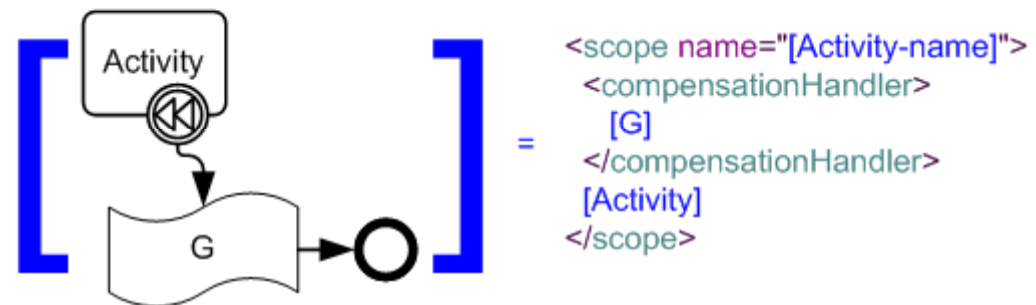
Mapping Handlers


```
<eventHandlers>
  <onEvent partnerLink="[e-operation-interface]"
       operation="[e-operation]">
    <scope>[G]</scope>
  </onEvent>
</eventHandlers>
```

```
<eventHandlers>
  <onAlarm>[timer-spec]
    <scope>[G]</scope>
  </onAlarm>
</eventHandlers>
```

```
<faultHandlers>
  <catch faultName="[e-error]">
    [G]
  </catch>
</faultHandlers>
```

```
<compensationHandler>
  [G]
</compensationHandler>
```

Mapping Boundary Events (1/4)

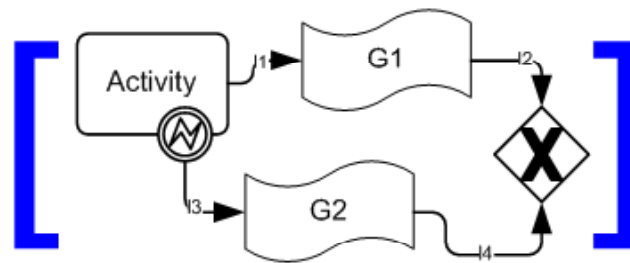```
<scope>
  <faultHandlers>
    <faultHandler>
      <catch faultName="...">
        <empty>
          <sources><source linkName="faultLink"/></sources>
        </empty>
      </catch>
    </faultHandler>
  </faultHandlers>
  [S]
</scope>
```
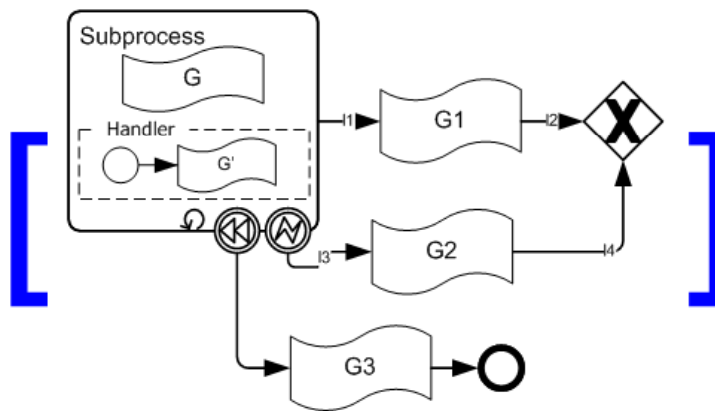
# Mapping Boundary Events (3/4)



```
<flow>
 <links>
  <link name="[l1]"/>
  ...
  <link name="[l4]"/>
 </links>
 <scope>
  <sources><source linkName="[l1]"/></sources>
  <faultHandlers>
   <catch faultName="[e-error]">
    <empty>
     <sources><source linkName="[l3]"/></sources>
    </empty>
   </catch>
  </faultHandlers>
  [Activity]
 </scope>
 <flow>
  <targets><target linkName="[l1]"/></targets>
  <sources><source linkName="[l2]"/></sources>
  [G1]
 </flow>
 <flow>
  <targets><target linkName="[l3]"/></targets>
  <sources><source linkName="[l4]"/></sources>
  [G2]
 </flow>
 <empty>
  <sources><source linkName="[l2]"/>
  <source linkName="[l4]"/></sources>
 </empty>
</flow>
```

Mapping Boundary Events (1/4)


```
<flow>
 <links>
   <link name="[l1]"/>
   ...
   <link name="[l4]"/>
 </links>
 <scope>
   <sources><source linkName="[l1]"/></sources>
   <faultHandlers>
     <catch faultName="[e-error]">
       <empty>
         <sources><source linkName="[l3]"/></sources>
       </empty>
     </catch>
   </faultHandlers>
   <compensationHandler>
     [G3]
   </compensationHandler>
   <while>
     <condition>[p]</condition>
     <scope>
       [Handler]
       [G]
     </scope>
   </while>
 </scope>
 <flow>
   <targets><target linkName="[l1]"/></targets>
   <sources><source linkName="[l2]"/></sources>
   [G1]
 </flow>
 <flow>
   <targets><target linkName="[l3]"/></targets>
   <sources><source linkName="[l4]"/></sources>
   [G2]
 </flow>
 <empty>
   <sources><source linkName="[l2]"/>
   <source linkName="[l4]"/></sources>
 </empty>
</flow>
```

Agenda

Historical Remarks

BPMN Origin and Positioning

The Notion of "Native Metamodel"

View on Process Engines

Selective BPMN 2.0 Features

Executable BPMN - BPEL Mapping

Summary

Takeaways

- **BPMN 2.0…**
  - …is a significant extension of BPMN 1.x
    - Choreographies, Collaborations, Conversations,
    - New event types and modes,
    - Event subprocesses,…
  - …allows to specify <u>executable</u> process models
  - …contains a subset nearly isomorphic to BPEL
- **Especially, "BPMN 2.0 engines" can be build**
- **Process engines implement their own internal metamodel**
  - Native BPEL engines are very rare!
  - So will be native BPMN engines!
- **Based on BPMN 2.0 complexity, 100% compliant implementations will take far more than a decade (if they will hit the market at all)**
  - Do you know a 100% BPEL compliant engine?
- **In practice, most process engines will support both, BPEL and (a subset of) BPMN 2.0**

End of Document