



+

The logo for Apache Camel, featuring a grey silhouette of a camel in profile to the left of the text "Apache" in a large, bold, black sans-serif font, and "Camel" in a large, bold, black sans-serif font with a purple "C".

**Apache  
Camel**



Consulting  
Developing  
Speaking  
Coaching  
Writing



## Main Tasks

**Evaluation of Technologies and Products**

**Requirements Engineering**

**Enterprise Architecture Management**

**Business Process Management**

**Architecture and Development of  
Applications**

**Planning and Introduction of SOA**

**Integration of Legacy Applications**

**Cloud Computing**

## Contact

**Email: [kai.waehner@mwea.de](mailto:kai.waehner@mwea.de)**

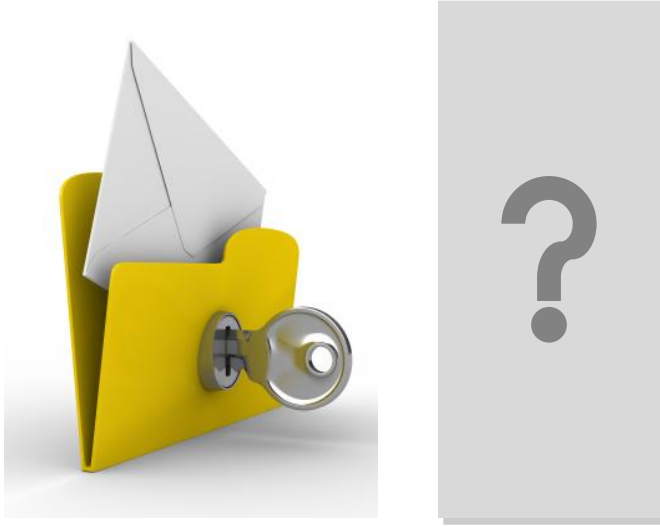
**Blog: [www.kai-waehner.de/blog](http://www.kai-waehner.de/blog)**

**Twitter: [@KaiWaehner](https://twitter.com/KaiWaehner)**

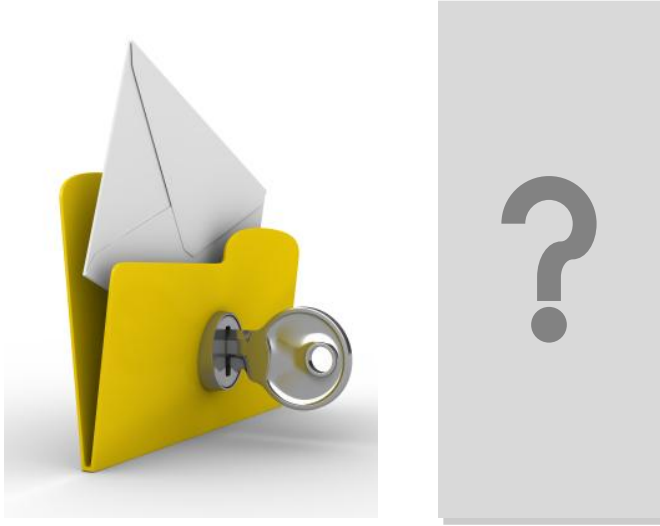
**Social Networks: Xing, LinkedIn**

# What is the Key Message?



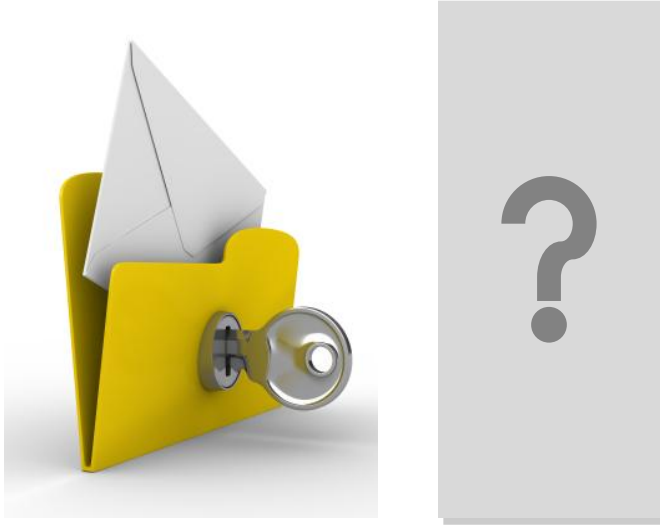


**BPM should be used for optimizing business processes!**



BPM should be used for optimizing business processes!

**BPM should NOT be used for systems integration!**



BPM should be used for optimizing business processes!

BPM should NOT be used for systems integration!

**Activiti and Apache Camel are a perfect combination!**

- 1) Business Process Management (BPM)
- 1) Activiti
- 1) Apache Camel
- 2) Combination of Activiti and Apache Camel

## **1) Business Process Management (BPM)**

1) Activiti

1) Apache Camel

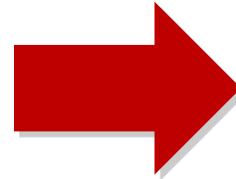
2) Combination of Activiti and Apache Camel



*BPM attempts to **improve processes continuously**.  
It can therefore be described as a "process  
optimization process."*

Wikipedia

## Business-IT-Alignment



increase efficiency  
better quality  
reduce costs  
enable new business models



BPEL

jPDL

BPMN

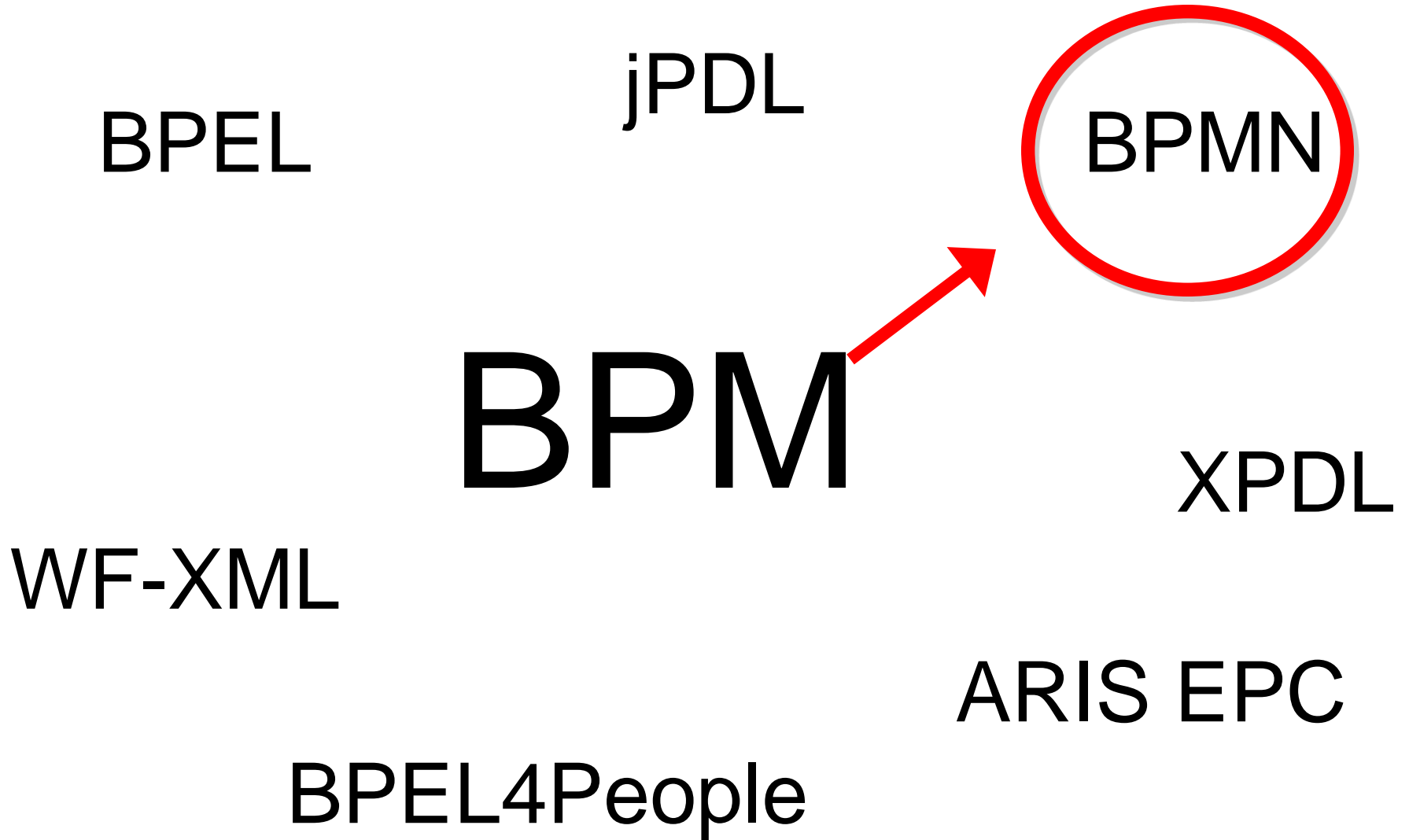
**BPM**

XPDL

WF-XML

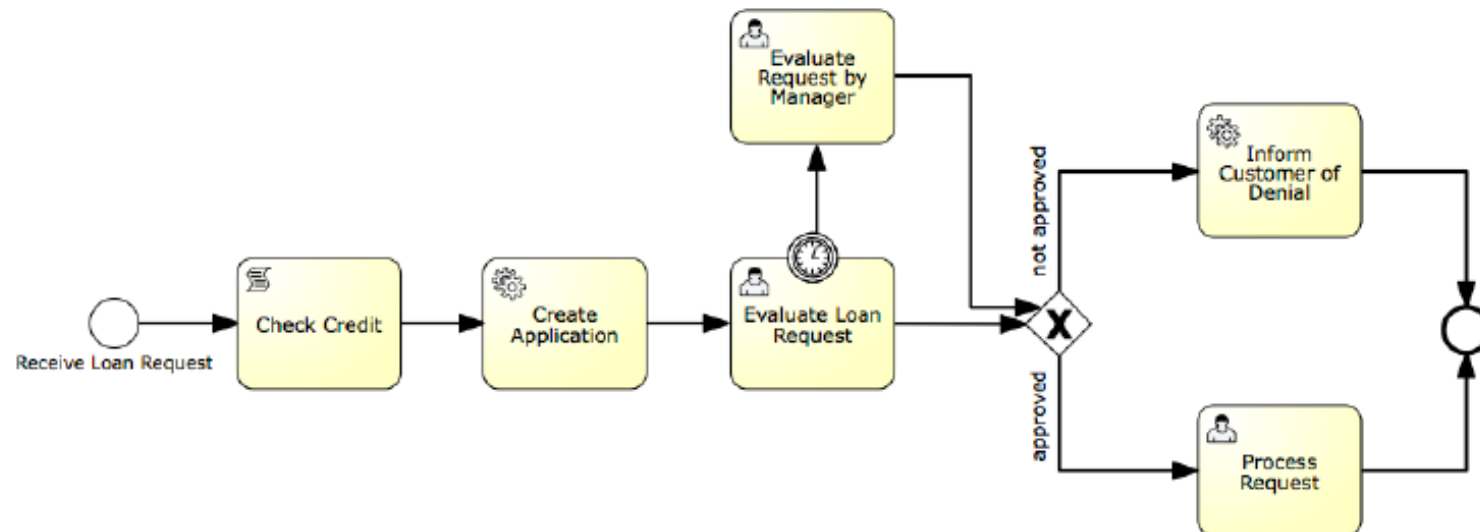
ARIS EPC

BPEL4People



*Business Process Model and Notation (BPMN) is a **graphical representation** for specifying business processes in a business process model.*

Wikipedia



- BPMN is a standard notation for **designing business processes**  
(versus: UML is a standard modeling language best suited for designing and implementing software)
- not just flow charts! sufficient restrictions / constraints => **executable!**
- standardized **XML** format
- **orchestration** and **choreography**
- **extension points** => add specific needs without breaking interoperability
- optional mapping of a BPMN subset to BPEL  
(restricted to block-structured flows without cycles)

## BPMN 2.0 - Business Process Model and Notation

<http://bpmb.de/poster>

### Activities

- Task**: A Task is a unit of work, the job to be performed. When marked with a symbol it indicates a Sub-Process, an activity that can be reused.
- Transaction**: A Transaction is a unit of activities that logically belong together; it might follow a specified transaction protocol.
- Event Sub-Process**: An Event Sub-Process is placed into a Process or Sub-Process. It is activated when its start event gets triggered and can leverage the higher level process context or run in parallel from interrupting/depending on the start event.
- Call Activity**: A Call Activity is a wrapper for a globally defined Task or Process nested in this current Process. A call to a Process is marked with a symbol.

**Activity Markers**  
Markers indicate execution indicator of activities

- Sub-Process marker
- Loop marker
- Parallel marker
- Sequential marker
- Call-For-Work marker
- Companion marker

**Task Types**  
Types specify the nature of the activity to be performed

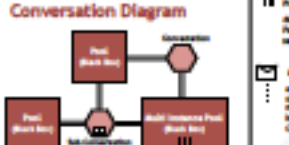
- Send Task
- Receive Task
- Manual Task
- Service Task
- Event Task
- Complex Task
- Call Activity

**Sequence Flow**: define the execution order of activities.  
**Default Flow**: to the default branch in a choice if all other conditions evaluate to false.  
**Conditional Flow**: has a condition assigned that defines whether or not the flow is used.

### Conversations

- A Conversation defines a set of logically related message exchanges. When marked with a symbol it indicates a Sub-Conversation, a composed conversation element.
- A Call Conversation is a wrapper for a globally defined Conversation or Sub-Conversation. A call to a Sub-conversation is marked with a symbol.
- A Conversation Link connects Conversations and Participants.

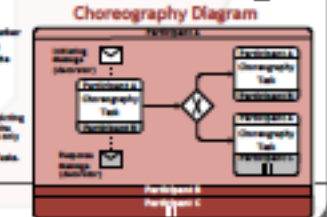
#### Conversation Diagram



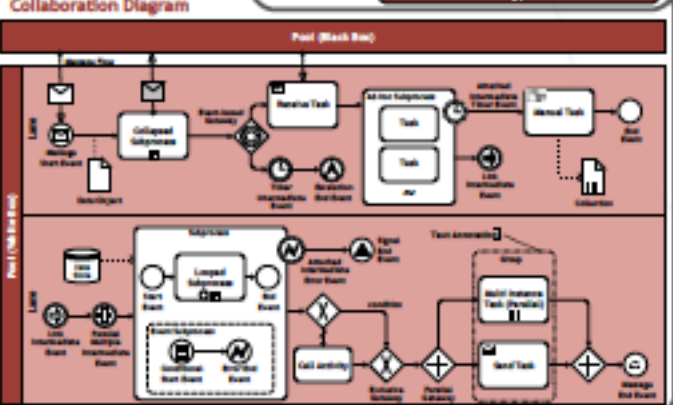
### Choreographies

- A Choreography Task represents an interaction (Message Exchange) between two Participants.
- A Sub-Choreography contains a defined choreography with several interactions.
- A Call Choreography is a wrapper for a globally defined Choreography Task or Sub-Choreography. A call to a Sub-Choreography is marked with a symbol.

#### Choreography Diagram



### Collaboration Diagram



**Swimlanes**  
Swimlanes represent organizational boundaries. Message flows can be attributed to participants, activities, or message events. The message flow can be determined with an arrowhead depicting the direction of the message.

### Events


	Start	Intermediate	End
None (Start)			
Message Receiving and Sending			
Timer (Cycle timer events, points in time, time span or duration)			
Escalation (Escalation to an higher level of responsibility)			
Conditional (Starting to change business conditions or triggering business rules)			
Error (Error page, error events, error messages)			
Event Catching or throwing manual events			
Cancel (Starting to cancel transactions or triggering compensation)			
Compensation (Handling or triggering compensation)			
Signal (Signaling across different processes, a signal event can be caught multiple times)			
Multiple (Catching instance of a set of events, throwing all events defined)			
Parallel Multiple (Catching all out of a set of parallel events)			
Termination (Triggering the iterative termination of a process)			

### Gateways

- When splitting, it routes the sequence flow to exactly one of the outgoing branches. When merging, it awaits one incoming branch to complete before engaging the outgoing flow.
- It is always followed by waiting events or receive tasks. Sequence flow is initiated in the subsequent activity which triggers the flow.
- When used to split the sequence flow, all outgoing branches are activated simultaneously. When merging parallel branches it waits for all incoming branches to complete before engaging the outgoing flow.
- When splitting, one or more branches are activated. All active outgoing branches must complete before merging.
- Each occurrence of a subsequent event starts a new process instance.
- Complex merging and branching behavior that is not supported by other gateways.
- The occurrence of all subsequent events starts a new process instance.

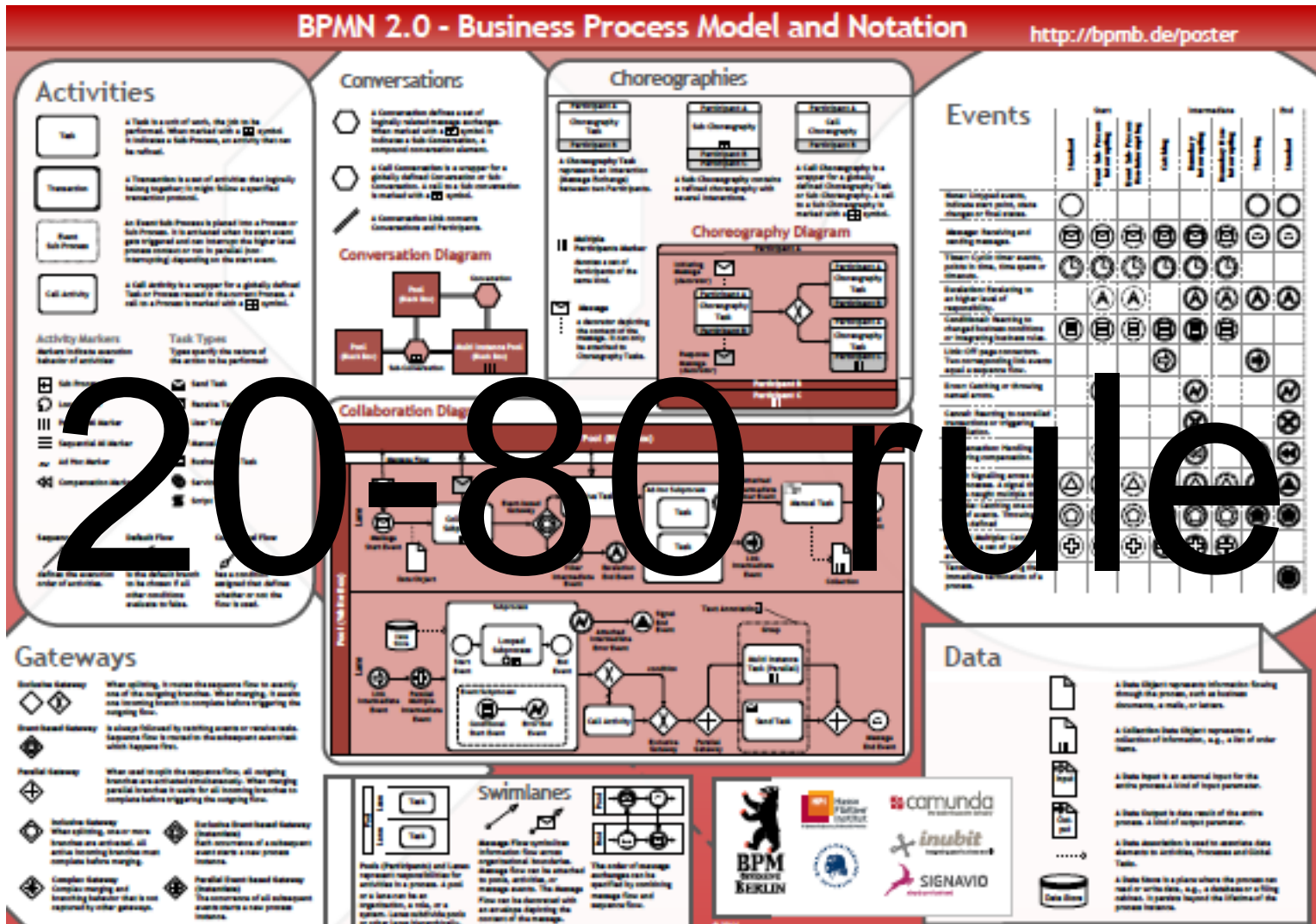
### Data

- A Data Object represents information flowing through the process, such as business documents, a mail, or a letter.
- A Collection Data Object represents a collection of information, e.g., a list of order items.
- A Data Input is an external input for the entire process or a list of input parameters.
- A Data Output is data results of the entire process, a list of output parameters.
- A Data Association is used to associate data elements in activities, Processes and Global Tasks.
- A Data Store is a place where the process can read or write data, e.g., a database or a filing cabinet. It persists beyond the lifetime of the process instance.





**BPMN 2.0 - Business Process Model and Notation** <http://bpmb.de/poster>



**Activities**

- Task**: A Task is a unit of work, the job to be performed. When marked with a symbol it indicates a Sub-Process, an activity that can be reused.
- Transaction**: A Transaction is a set of activities that logically belong together; it might follow a specified transaction protocol.
- Event Sub-Process**: An Event Sub-Process is placed into a Process or Sub-Process. It is activated when its start event gets triggered and can overwrite the higher level process context or run in parallel from, interrupting/depending on the start event.
- Call Activity**: A Call Activity is a wrapper for a globally defined Task or Process element in this current Process. A call in a Process is marked with a symbol.

**Activity Markers**

- Sub-Process marker: indicates a sub-process.
- Event marker: indicates an event.
- Start marker: indicates a start event.
- End marker: indicates an end event.
- Intermediate marker: indicates an intermediate event.
- Call marker: indicates a call activity.
- Complex marker: indicates a complex activity.
- Transaction marker: indicates a transaction.
- Event Sub-Process marker: indicates an event sub-process.
- Call Sub-Process marker: indicates a call sub-process.
- Complex Sub-Process marker: indicates a complex sub-process.
- Transaction Sub-Process marker: indicates a transaction sub-process.
- Event Sub-Process Call marker: indicates an event sub-process call.
- Call Sub-Process Call marker: indicates a call sub-process call.
- Complex Sub-Process Call marker: indicates a complex sub-process call.
- Transaction Sub-Process Call marker: indicates a transaction sub-process call.

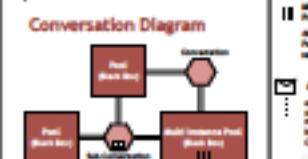
**Task Types**

- Start Task**: Type specifies the nature of the action to be performed.
- Intermediate Task**: Type specifies the nature of the action to be performed.
- End Task**: Type specifies the nature of the action to be performed.
- Complex Task**: Type specifies the nature of the action to be performed.
- Event Task**: Type specifies the nature of the action to be performed.
- Call Task**: Type specifies the nature of the action to be performed.
- Transaction Task**: Type specifies the nature of the action to be performed.
- Event Sub-Process Task**: Type specifies the nature of the action to be performed.
- Call Sub-Process Task**: Type specifies the nature of the action to be performed.
- Complex Sub-Process Task**: Type specifies the nature of the action to be performed.
- Transaction Sub-Process Task**: Type specifies the nature of the action to be performed.
- Event Sub-Process Call Task**: Type specifies the nature of the action to be performed.
- Call Sub-Process Call Task**: Type specifies the nature of the action to be performed.
- Complex Sub-Process Call Task**: Type specifies the nature of the action to be performed.
- Transaction Sub-Process Call Task**: Type specifies the nature of the action to be performed.

**Conversations**

- A Conversation defines a set of logically related message exchanges. When marked with a symbol it indicates a Sub-Conversation, a composed conversation element.
- A Conversation is a wrapper for a globally defined Conversation or Sub-Conversation. A call to a Sub-Conversation is marked with a symbol.
- A Conversation Link connects Conversations and Participants.

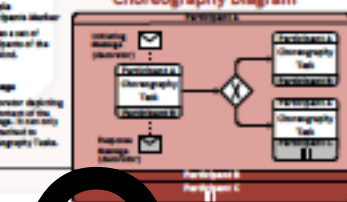
**Conversation Diagram**



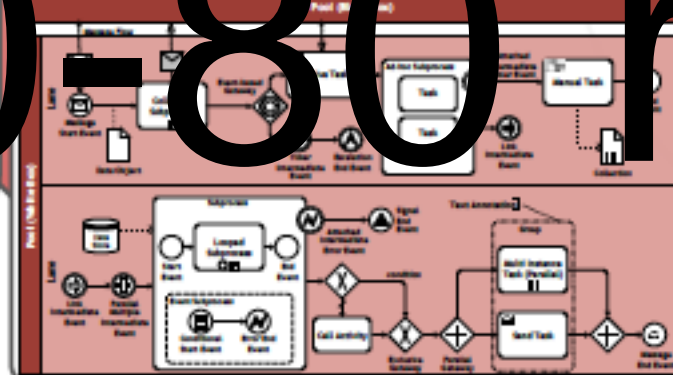
**Choreographies**

- Choreography Task**: A Choreography Task represents an interaction (Message Exchange) between two Participants.
- Sub-Choreography**: A Sub-Choreography consists of a refined choreography with several interactions.
- Call Choreography**: A Call Choreography is a wrapper for a globally defined Choreography Task or Sub-Choreography. A call to a Sub-Choreography is marked with a symbol.

**Choreography Diagram**



**Collaboration Diagram**



**Events**

	Start	Intermediate	End
<b>Timer</b>			
<b>Message</b>			
<b>Exception</b>			
<b>Complex</b>			
<b>Conditional</b>			
<b>Parallel</b>			
<b>Exclusive</b>			
<b>OR</b>			
<b>XOR</b>			
<b>AND</b>			
<b>OR-AND</b>			
<b>AND-OR</b>			
<b>AND-AND</b>			
<b>OR-OR</b>			
<b>XOR-AND</b>			
<b>AND-XOR</b>			
<b>AND-AND-OR</b>			
<b>OR-AND-OR</b>			
<b>AND-AND-AND</b>			
<b>OR-AND-AND</b>			
<b>AND-AND-XOR</b>			
<b>OR-AND-XOR</b>			
<b>XOR-AND-XOR</b>			
<b>AND-AND-XOR-AND</b>			
<b>OR-AND-XOR-AND</b>			
<b>XOR-AND-XOR-AND</b>			
<b>AND-AND-XOR-AND-OR</b>			
<b>OR-AND-XOR-AND-OR</b>			
<b>XOR-AND-XOR-AND-OR</b>			
<b>AND-AND-XOR-AND-AND</b>			
<b>OR-AND-XOR-AND-AND</b>			
<b>XOR-AND-XOR-AND-AND</b>			
<b>AND-AND-XOR-AND-AND-OR</b>			
<b>OR-AND-XOR-AND-AND-OR</b>			
<b>XOR-AND-XOR-AND-AND-OR</b>			

**Gateways**

- Exclusive Gateway**: When splitting, it routes the outgoing flow to exactly one of the outgoing branches. When merging, it waits until one incoming branch is complete before engaging the outgoing flow.
- Event-based Gateway**: It is always followed by waiting events or receive tasks. Splits flow to one or more of the outgoing activities which trigger the flow.
- Parallel Gateway**: When used to split the outgoing flow, all outgoing branches are activated simultaneously. When merging parallel branches it waits for all incoming branches to complete before engaging the outgoing flow.
- Inclusive Gateway**: When splitting, one or more branches are activated. All active outgoing branches must complete before merging.
- Event-based Inclusive Gateway**: Each occurrence of a subsequent event starts a new process instance.
- Complex Gateway**: Complex merging and branching behavior that is not expressed by other gateways.
- Parallel Event-based Gateway**: The occurrence of all subsequent events starts a new process instance.

**Swimlanes**

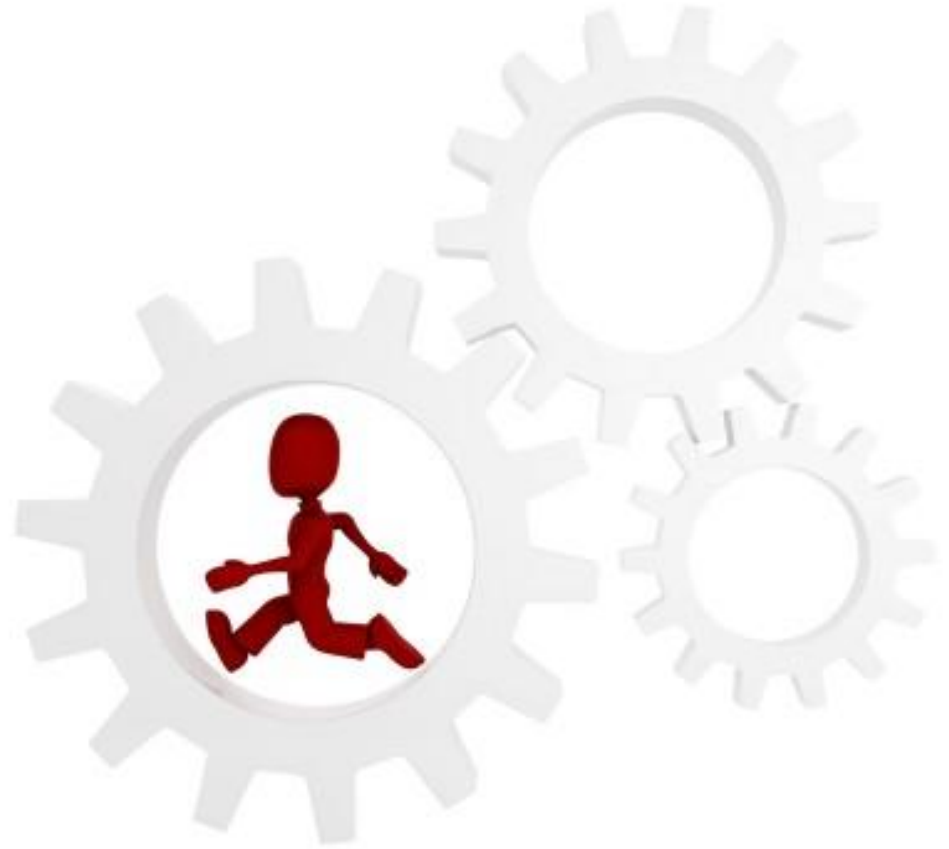
- Pool (Participant) and Lane**: represent organizational boundaries. A lane represents responsibilities for activities in a process. A pool or a lane can be an organization, a role, or a system. Lanes within a pool or other lanes (hierarchically).
- Message Flow**: swimlanes information flow across organizational boundaries. Message flow can be initiated by a task, activity, or message event. The message flow can be terminated with an activity depicting the content of the message.
- Order of message exchanges**: can be verified by combining message flow and sequence flow.

**Data**

- Data Object**: represents information flowing through the process, such as business document, a mail, or a letter.
- Collection Data Object**: represents a collection of information, e.g., a list of order items.
- Data Input**: is an external input for the entire process; a list of input parameters.
- Data Output**: is data results of the entire process; a list of output parameters.
- Data Association**: is used to associate data elements in activities, Processes and Global Tasks.
- Data Store**: is a place where the process can read or write data, e.g., a database or a filing cabinet. It persists beyond the lifetime of the process instance.

20-80 rule

- **long-running stateful** workflows
- frequently changing processes
- **human interaction**





- 1) Business analysts **will** create executable process models
- 2) Business analysts **can** create executable process models
- 3) Business analysts **want** to create executable process models
- 4) **IT wants** business analysts to create executable process models

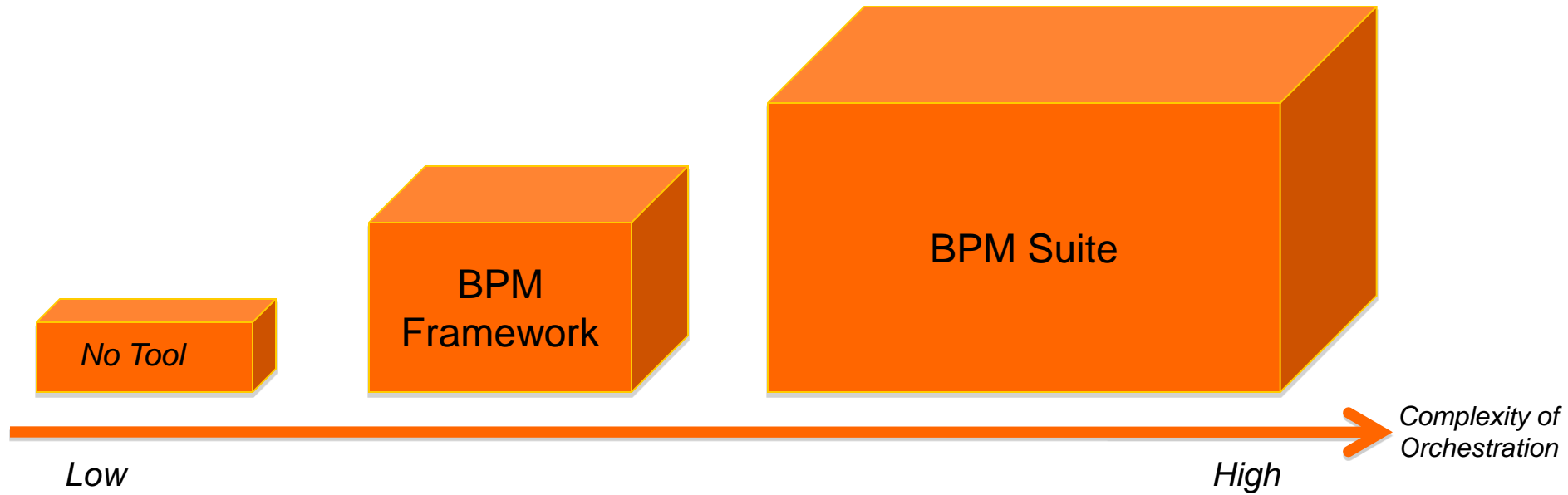
<http://www.activevos.com/blog/soa/the-four-myths-of-bpm-projects-what-it-project-teams-need-to-know/2011/01/18/>

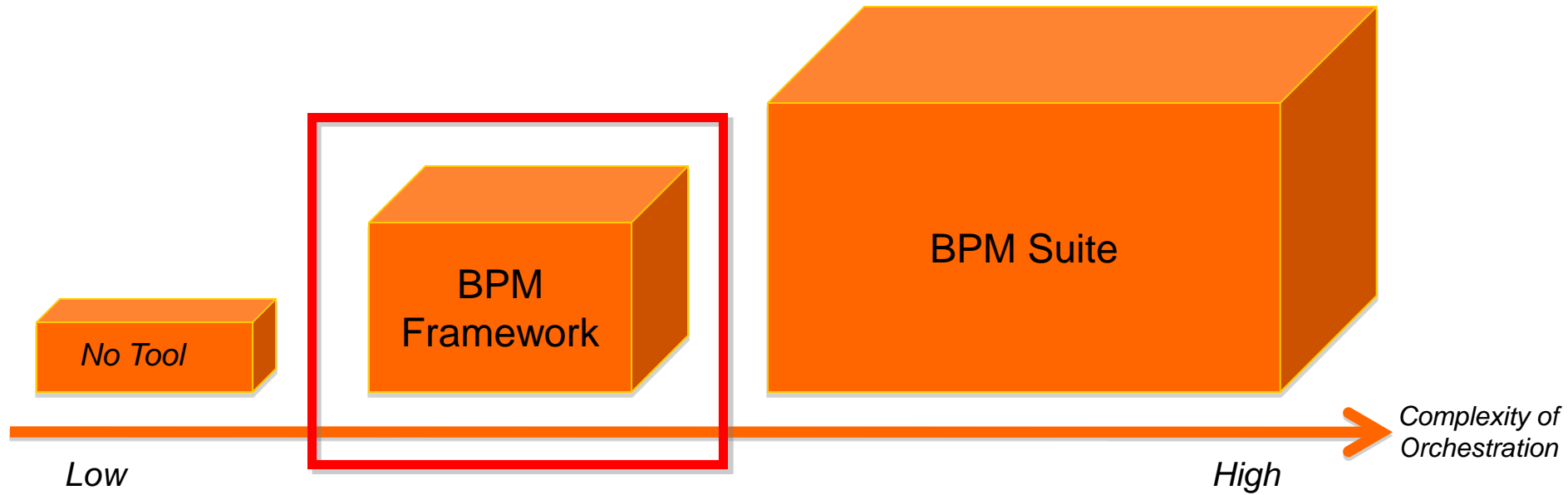
1) Business Process Management (BPM)

**1) Activiti**

1) Apache Camel

2) Combination of Activiti and Apache Camel





**Activiti** vs. JBoss jBPM vs. Bonita vs. ProcessMaker

- open source
- BPMN 2.0 process engine
- lightweight
- easy to use
- Java API
- developer-focused
- embeddable



Activiti Modeler

Activiti Designer

# Process Engine

camunda fox style

Activiti Explorer





- BPMN 2.0 process engine
- state machine with one active state
- execution progresses via transitions
- most BPMN 2.0 elements are implemented as a state
- states are connected with leaving and arriving transitions (called sequence flows)
- every state (i.e. its corresponding BPMN 2.0 element) can have a piece of logic attached (executed when the process instance enters the state)

## Services

RuntimeService

TaskService

FormService

HistoryService

IdentityService

ManagementService

RepositoryService

## Service Tasks (BPMN Standard)

Web Service Task

Script Task (e.g. Groovy or JavaScript)

User Task

Business Rule Task

## Service Tasks (Activiti Extension)

Java Tasks

Spring Service Task

... more

```
<process id="bookorder" name="Order book">
  <startEvent id="startevent1" name="Start"/>
  <sequenceFlow id="sequenceflow1" name="Validate order"
    sourceRef="startevent1" targetRef="scripttask1"/>
  <scriptTask id="scripttask1"
    name="Validate order" scriptFormat="groovy">
    <script>out.println "validating order for isbn " + isbn;</script>
  </scriptTask>
  <sequenceFlow id="sequenceflow2" name="Ending process"
    sourceRef="usertask1" targetRef="endevent1"/>
  <endEvent id="endevent1" name="End"/>
</process>
```

```
ProcessEngine processEngine = ProcessEngineConfiguration
    .createStandaloneInMemProcessEngineConfiguration()
    .buildProcessEngine();

RuntimeService runtimeService =
    processEngine.getRuntimeService();

RepositoryService repositoryService =
    processEngine.getRepositoryService();

repositoryService.createDeployment()
    .addClasspathResource("bookorder.simple.bpmn20.xml")
    .deploy();

ProcessInstance processInstance =
runtimeService.startProcessInstanceByKey(
    "simplebookorder");

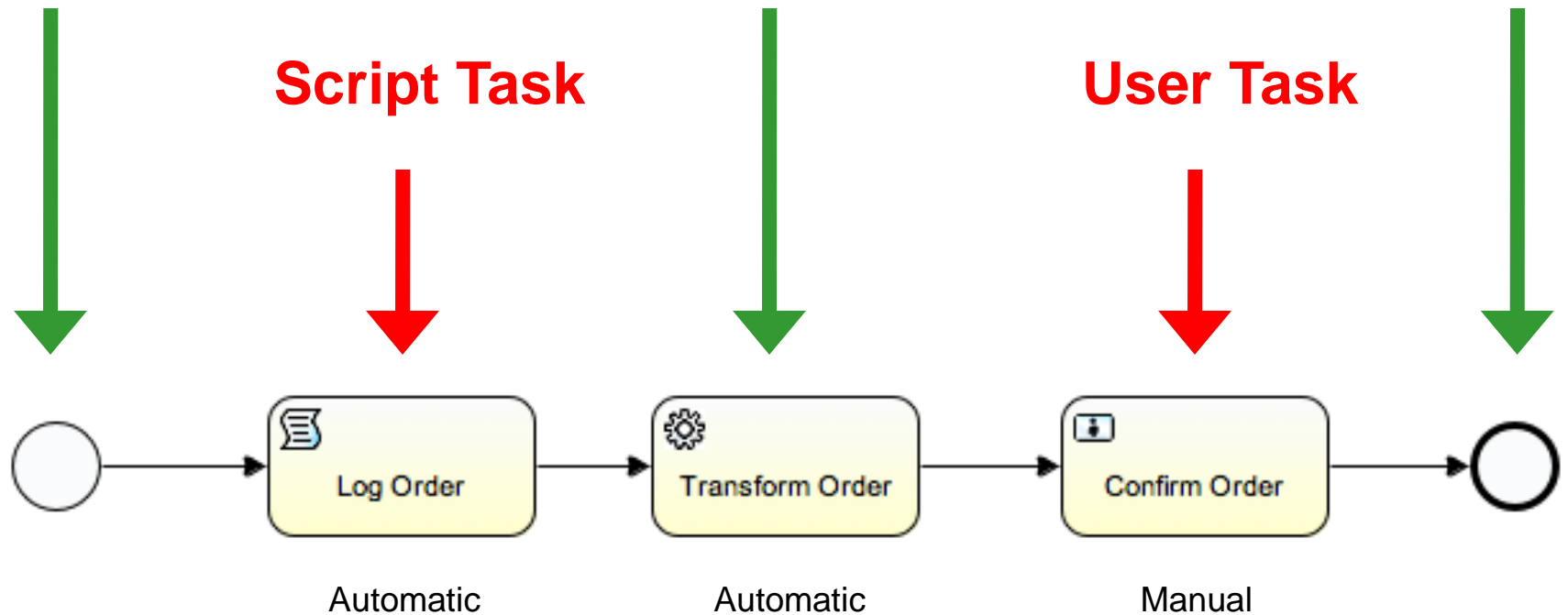
assertNotNull(processInstance.getId());

System.out.println("id " + processInstance.getId() + " " +
    processInstance.getProcessDefinitionId());
```

Start Event

Service Task

End Event





## Activiti in Action

- 1) Business Process Management (BPM)
- 1) Activiti
- 1) Apache Camel**
- 2) Combination of Activiti and Apache Camel

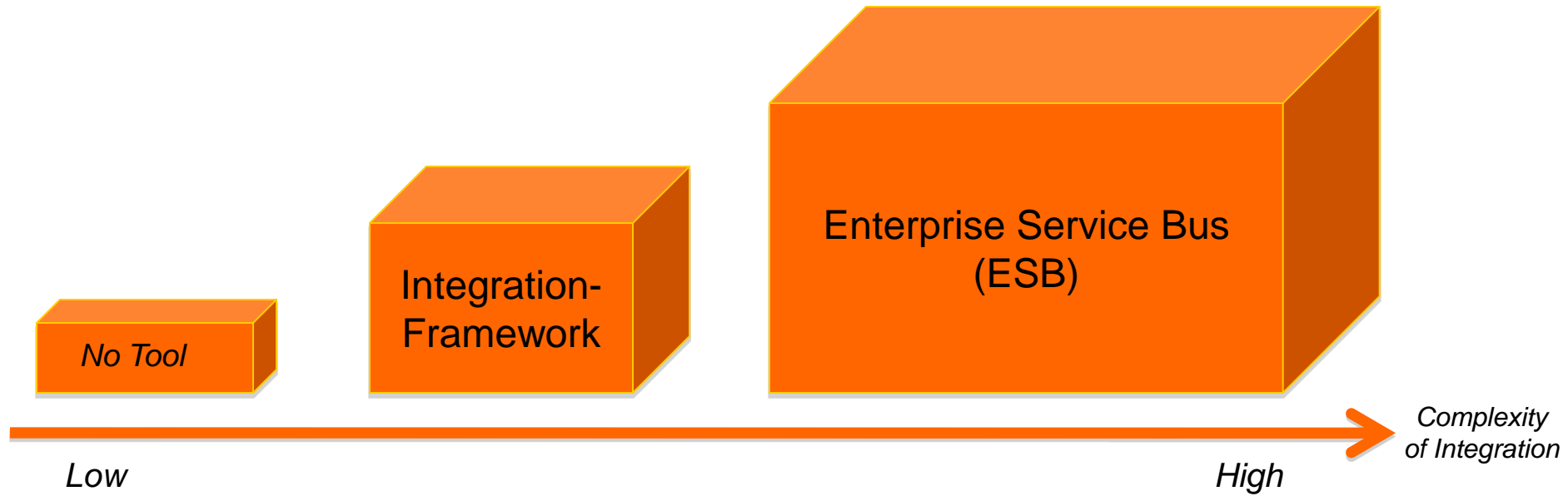


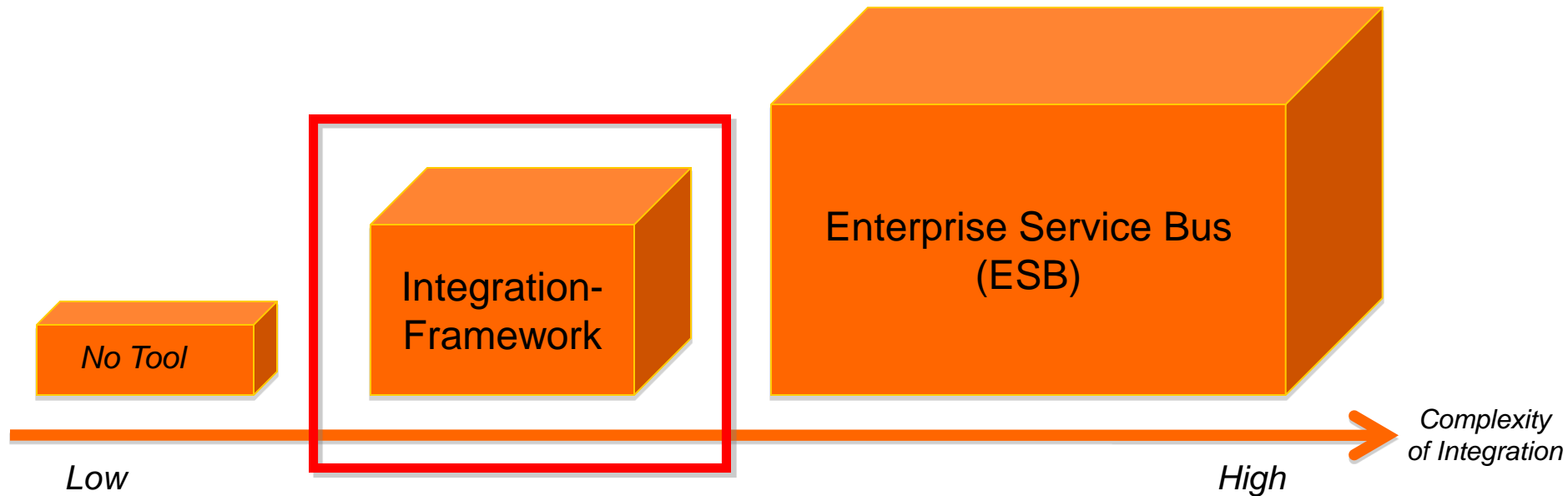




Come on, guys! We are a







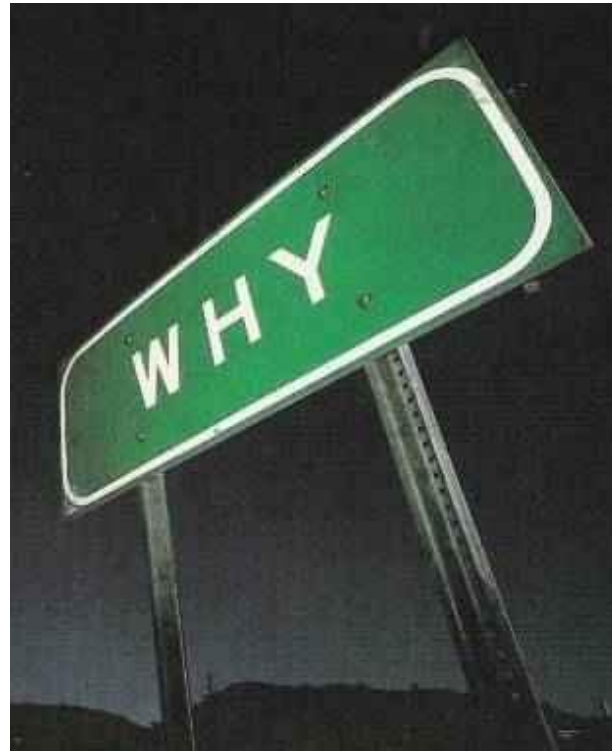
**Apache Camel vs. Spring Integration vs. Mule ESB**

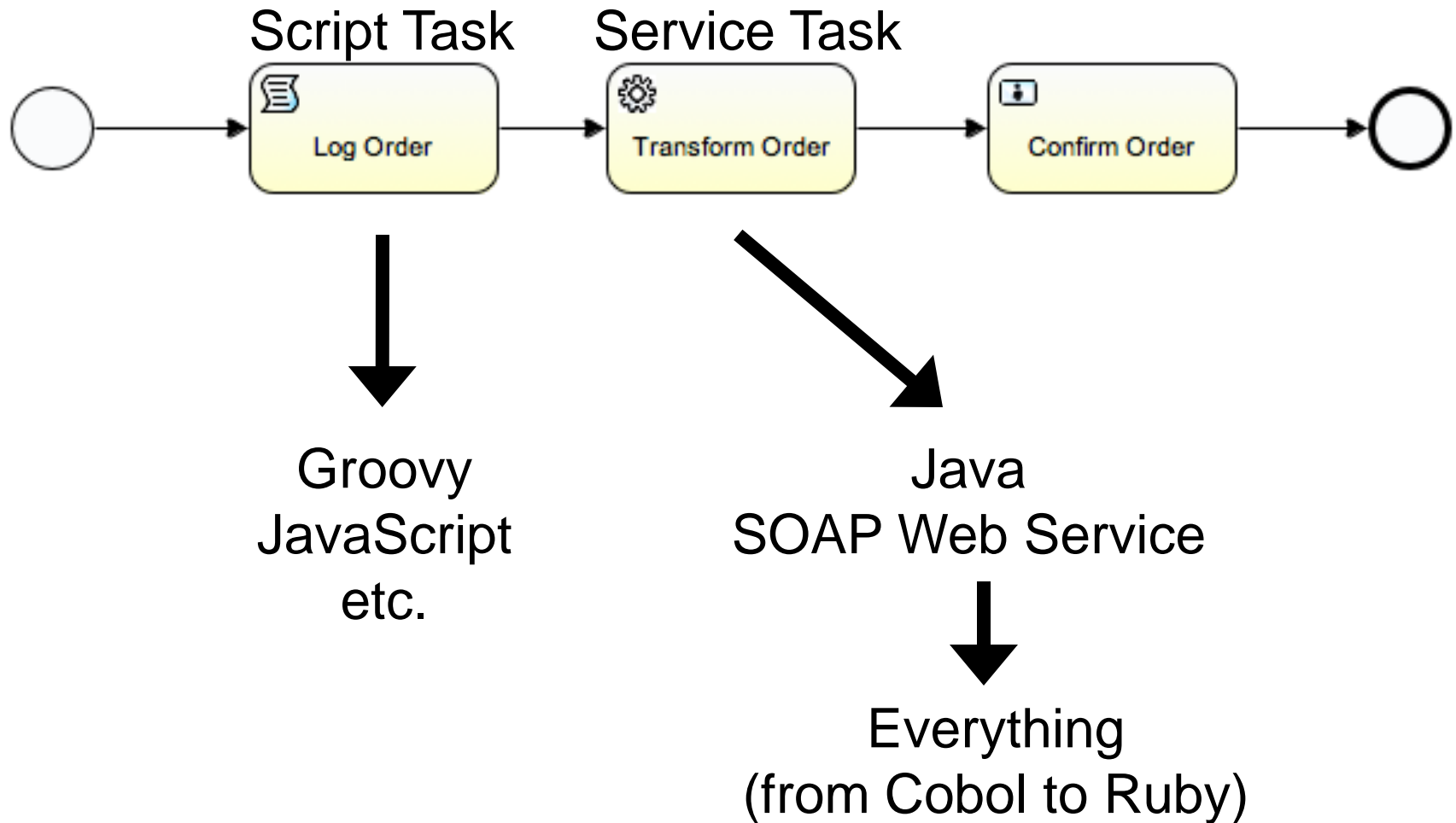
<http://www.kai-waehner.de/blog/2012/01/10/spoilt-for-choice-which-integration-framework-to-use-spring-integration-mule-esb-or-apache-camel>



- Standardized Modeling
- Efficient Realization
- Developer-focused
- Automatic Testing
- **Many Components**
- **Several DSLs**
- **Awesome Community**

- 1) Business Process Management (BPM)
- 1) Activiti
- 1) Apache Camel
- 2) Combination of Activiti and Apache Camel**







All Roads  
lead to Rome!







Both can realize processes. Both can integrate services.

Both can realize processes. Both can integrate services.



- Support for **long running stateful** processes
- **Human workflow** integration

Both can realize processes. Both can integrate services.

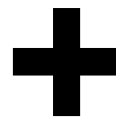


- Support for **long running stateful** processes
- **Human workflow** integration



- **Connectivity / Adaptors** to connect to external systems using a variety of different protocols
- Predefined **EIP** for message routing

<http://www.pleus.net/blog/?p=1028>



Both ...

- ... are lightweight
- ... are open source
- ... are developer-focused
- ... offer combination out-of-the-box

## BPMN - Java Service Task

```
<serviceTask id="myServiceTask"
  activiti:delegateExpression="\${camelBehaviour}" />
<sequenceFlow sourceRef="myServiceTask" targetRef="myUserTask" />
```

BPMN Process Instance

## Spring Bean

```
<bean id="camelBehaviour" class="org.activiti.camel.CamelBehaviour">
  <constructor-arg index="0">
    <list>
      <bean class="org.activiti.camel.SimpleContextProvider">
        <constructor-arg index="0" value="activitiCamelProcess" />
        <constructor-arg index="1" ref="camelContext" />
      </bean>
    </list>
  </constructor-arg>
</bean>
```

CamelContext

**// Producer => Call Activiti process from Camel**

```
from("direct:start")  
    .to("activiti:myProcess");
```

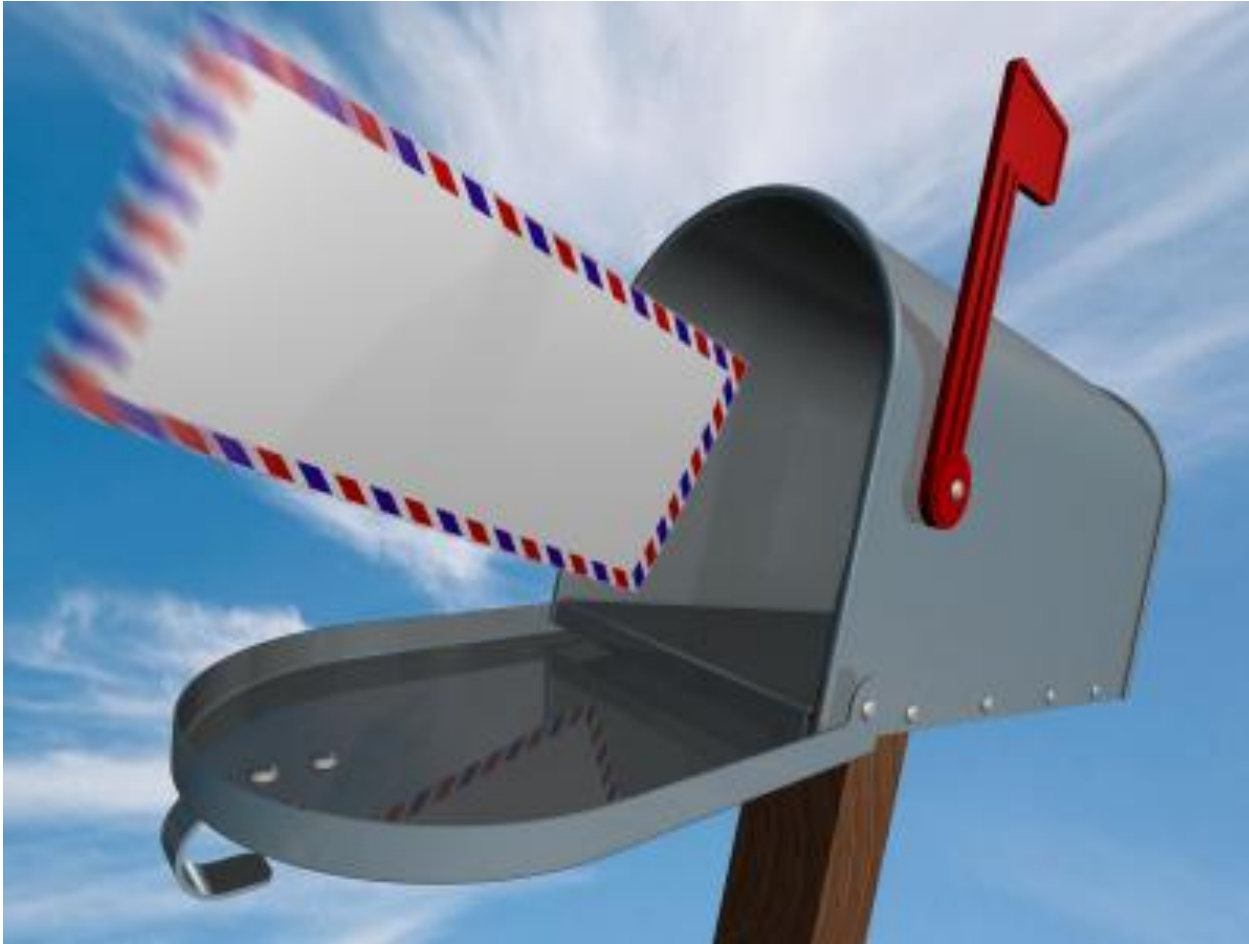
**// Consumer => Get called from Activiti process**

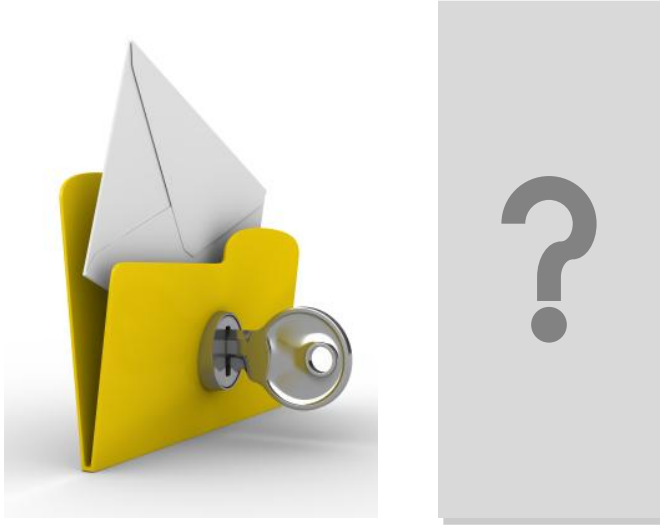
```
from("activiti:myProcess:myServiceTask")  
    .log(LoggingLevel.INFO, "Received message on service task ${property.var1}")  
    .setProperty("var2").constant("world")  
    .setBody().properties();
```





## Activiti and Apache Camel combined ...



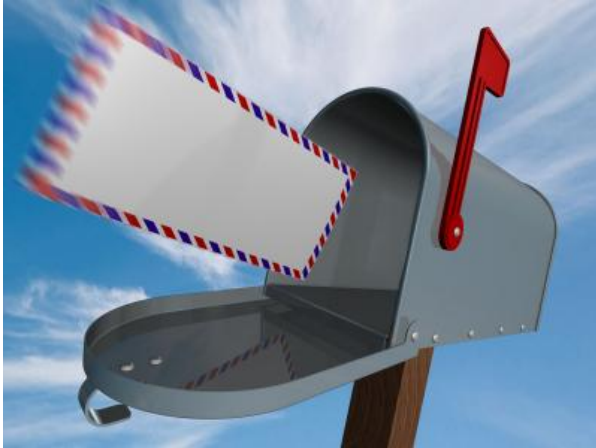


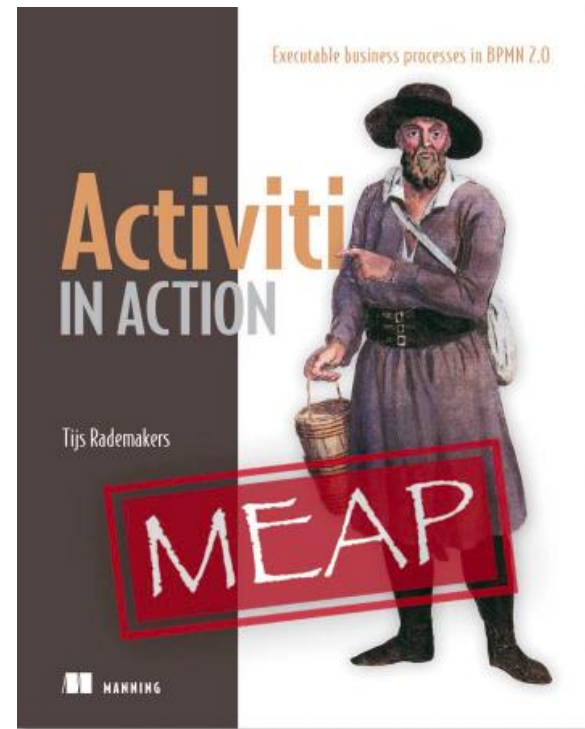
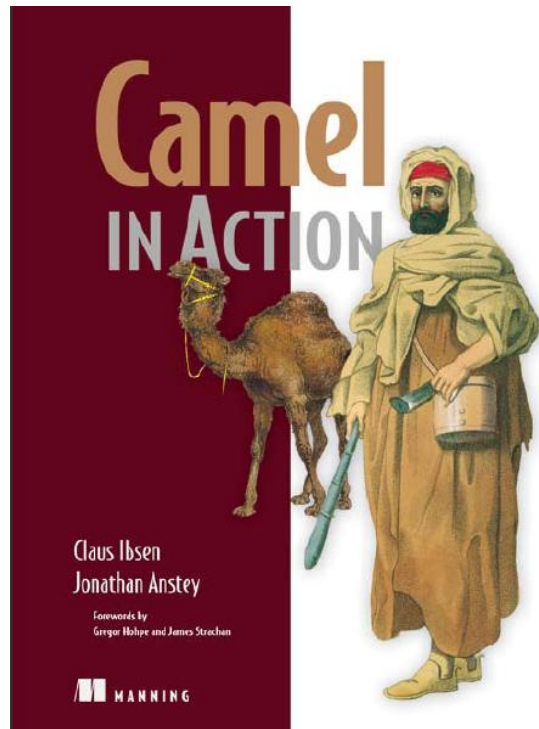
BPM should be used for optimizing business process 

BPM should NOT be used for systems integrati 

Activiti and Apache Camel are a perfect combinatic 

# Did you get the Key Message?









Thank you for your Attention. Any Questions?



Kai Wähler

MaibornWolff et al:  
[www.mwea.de](http://www.mwea.de)

Email: [kai.waehner@mwea.de](mailto:kai.waehner@mwea.de)

Twitter: @KaiWaehner

Blog: [www.kai-waehner.de/blog](http://www.kai-waehner.de/blog)

