University of Vienna, Sommersemester 2015

INTEROPERABILITY
- Service-oriented Architectures: Process-Orientation-

Univ. Prof. Dr. Stefanie Rinderle-Ma
Workflow Systems and Technology Group
Faculty of Computer Science
University of Vienna
stefanie.rinderle-ma@univie.ac.at

Outline

4.1 Motivation

4.2 Business Process Modeling Notation (BPMN)

4.3 Modeling Process Choreographies

4.4 Process Choreography Design

4.5 Process Choreography Implementation

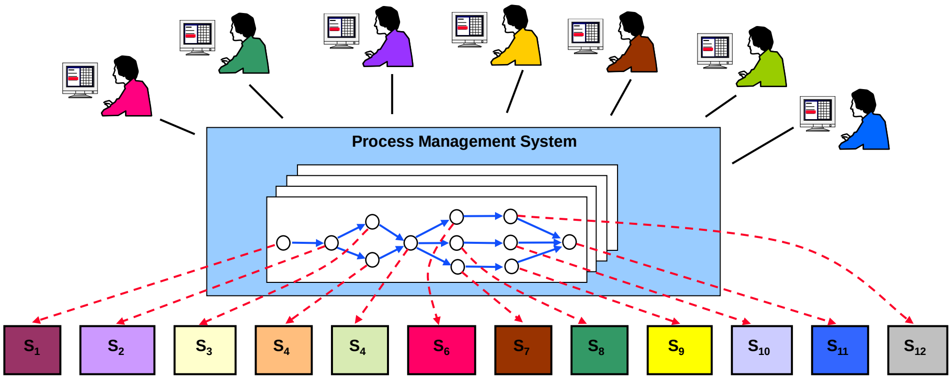
4.6 Business Process Execution Language (BPEL)

4.7 Summary and Current Research

References

4.1 Motivation

- ❐ **Again: Vision of SOA (*Service-Oriented Architecture*)**
  - ❍ Application functions (services) that can be individually invoked
  - ❍ Explicitely defined processes, managed by a process management system



Chapter based on : M. Reichert. R. Pryss: DBIS-Softwarelabor: Service-Oriented Computing, University of Ulm (2009)

4.1 Motivation

Very important to distinguish:

- ❐ **Process Orchestration**

  → One (executable) business process from the perspective and under the control of one particular partner (end point)

- ❐ **Process Choreography**

  → Exchange of messages between partners according to defined interaction rules between two or more partners (endpoints); also called global process!

Chapter based on : M. Reichert. R. Pryss: DBIS-Softwarelabor: Service-Oriented Computing, University of Ulm (2009)

Outline

4.1 Motivation

4.2 Business Process Modeling Notation (BPMN)

4.3 Modeling Process Choreographies

4.4 Process Choreography Design

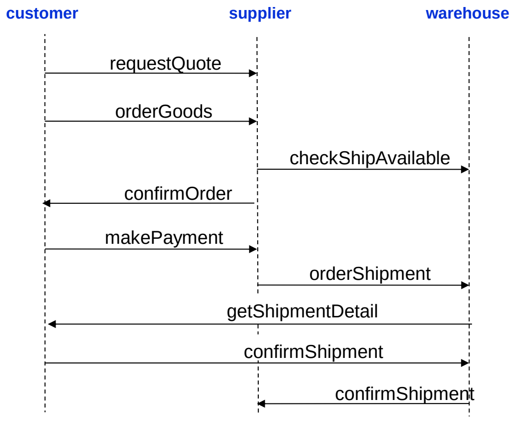
4.4 Process Choreography Implementation

4.6 Business Process Execution Language (BPEL)

4.7 Summary and Current Research

References

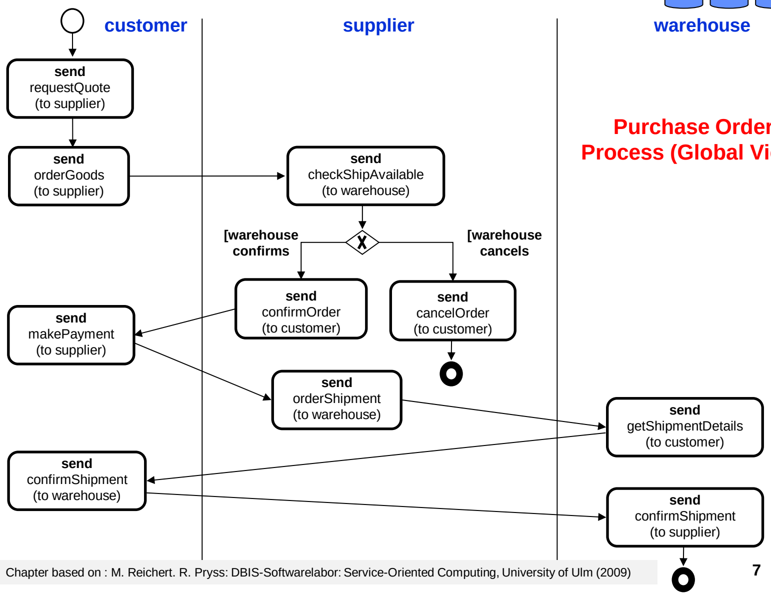
4.2 Business Process Modeling Notation



Purchase Order Process

Chapter based on : M. Reichert. R. Pryss: DBIS-Softwarelabor: Service-Oriented Computing, University of Ulm (2009)

4.2 Business Process Modeling Notation



customer

supplier

warehouse

Purchase Order Process (Global View)

send requestQuote (to supplier)

send orderGoods (to supplier)

send checkShipAvailable (to warehouse)

[warehouse confirms

[warehouse cancels

send confirmOrder (to customer)

send cancelOrder (to customer)

send makePayment (to supplier)

send orderShipment (to warehouse)

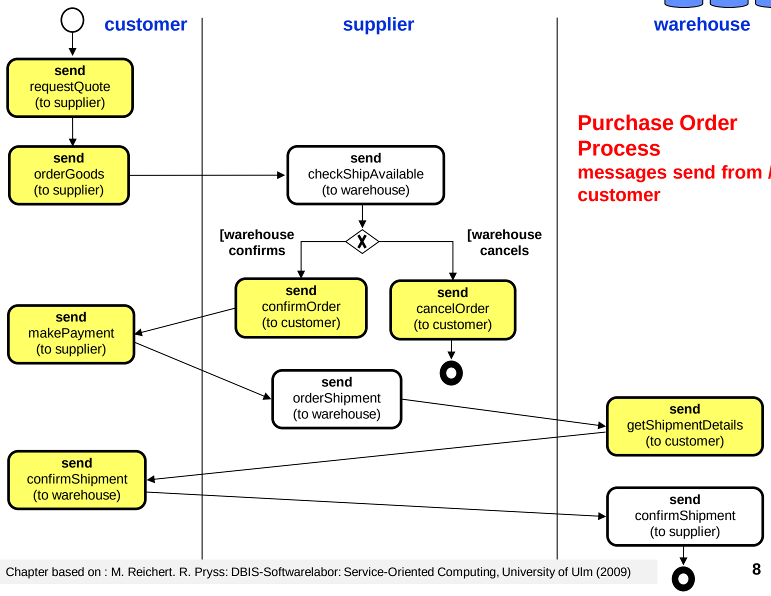
send getShipmentDetails (to customer)

send confirmShipment (to warehouse)

send confirmShipment (to supplier)

Chapter based on : M. Reichert. R. Pryss: DBIS-Softwarelabor: Service-Oriented Computing, University of Ulm (2009)

4.2 Business Process Modeling Notation



customer

supplier

warehouse

Purchase Order Process
messages send from / to customer

send requestQuote (to supplier)

send orderGoods (to supplier)

send checkShipAvailable (to warehouse)

[warehouse confirms

[warehouse cancels

send confirmOrder (to customer)

send cancelOrder (to customer)

send makePayment (to supplier)

send orderShipment (to warehouse)

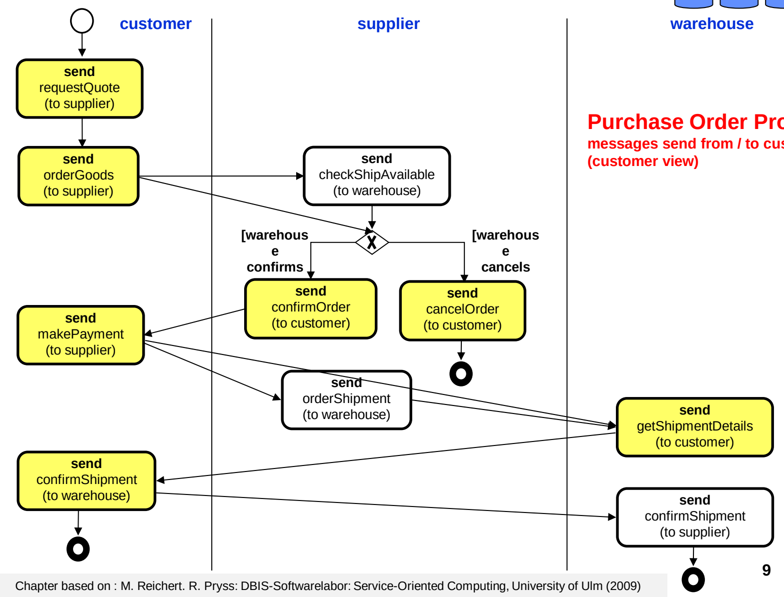
send getShipmentDetails (to customer)

send confirmShipment (to warehouse)

send confirmShipment (to supplier)

Chapter based on : M. Reichert. R. Pryss: DBIS-Softwarelabor: Service-Oriented Computing, University of Ulm (2009)

4.2 Business Process Modeling Notation



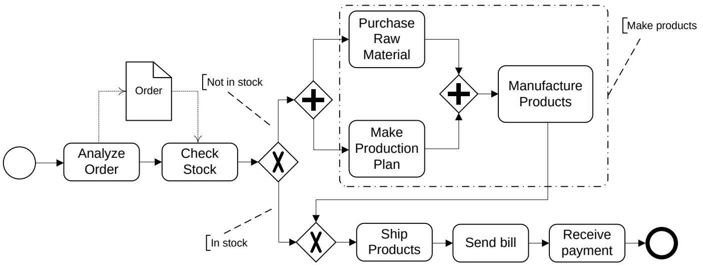
Purchase Order Process

messages send from / to customer (customer view)

Chapter based on : M. Reichert. R. Pryss: DBIS-Softwarelabor: Service-Oriented Computing, University of Ulm (2009)

4.2 Business Process Modeling Notation

Example of a BPMN Diagram



M. Weske: Business Process Management, © Springer-Verlag Berlin Heidelberg 2007

Fig 4.77. Business process diagram expressed in BPMN

4.2 Business Process Modeling Notation

Categories of Elements

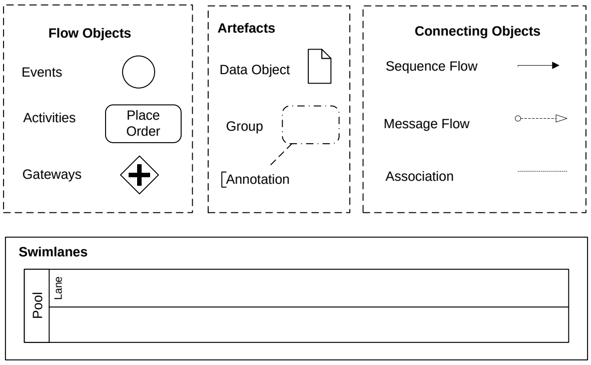


Fig 4.78. Business Process Modeling Notation: categories of elements

M. Weske: Business Process Management,
© Springer-Verlag Berlin Heidelberg 2007

4.2 Business Process Modeling Notation

Example: Scientific conference review process

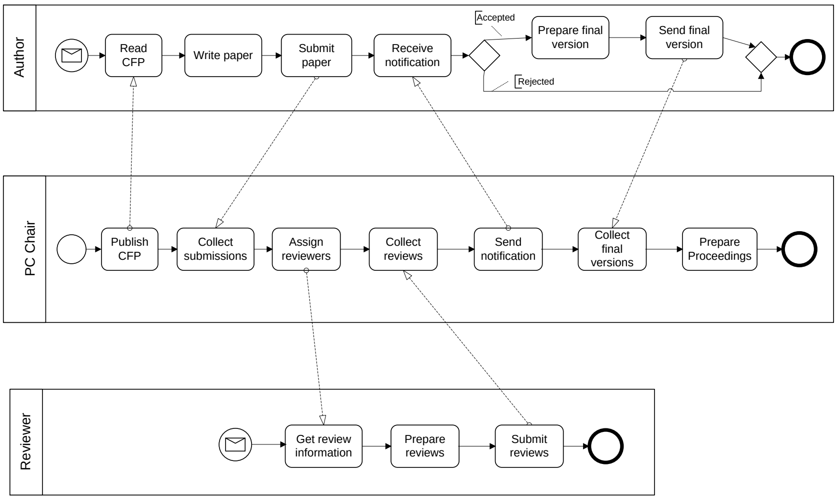


Fig 4.79. Business process diagram of a scientific conference review process

M. Weske: Business Process Management,
© Springer-Verlag Berlin Heidelberg 2007

4.2 Business Process Modeling Notation

Events

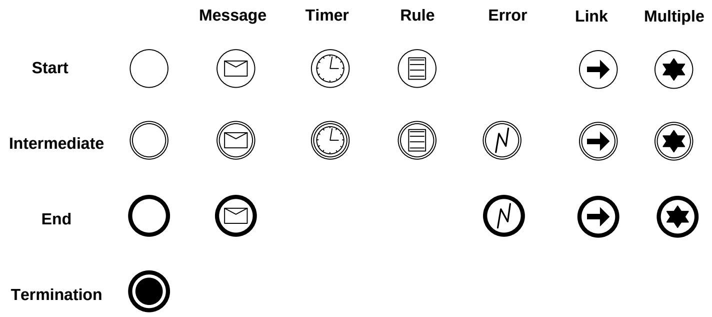


Fig 4.80. Event types in the BPMN, Object Management Group (2006)

M. Weske: Business Process Management,
© Springer-Verlag Berlin Heidelberg 2007

4.2 Business Process Modeling Notation

Activities and Subprocesses

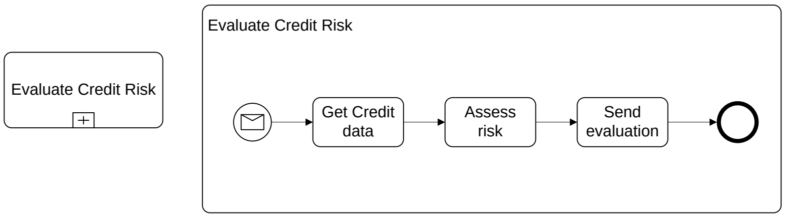


Fig 4.81. Collapsed and expanded subprocess

M. Weske: Business Process Management,
© Springer-Verlag Berlin Heidelberg 2007

4.2 Business Process Modeling Notation

Ad-hoc Process

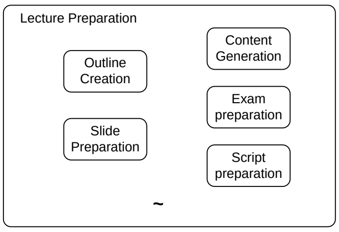


Fig 4.82. Sample adhoc process

M. Weske: Business Process Management,
© Springer-Verlag Berlin Heidelberg 2007

4.2 Business Process Modeling Notation

Sequence Flow and Exception Flows

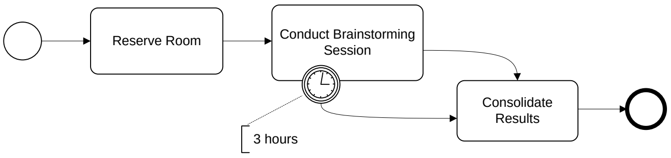
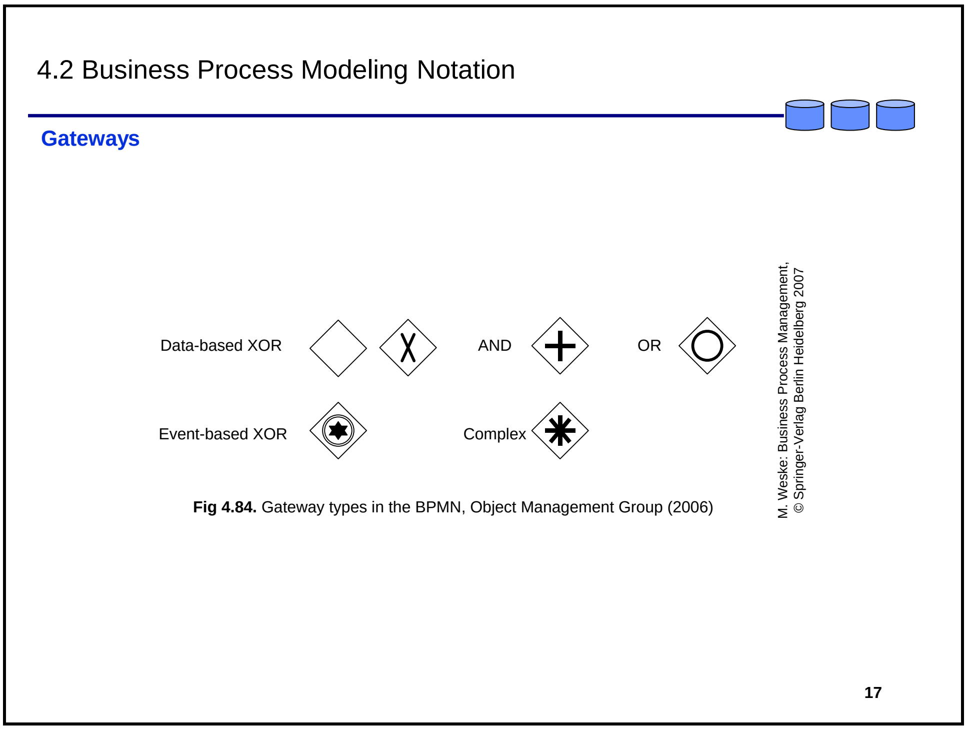


Fig 4.83. Exception flow, triggered by intermediate timer event

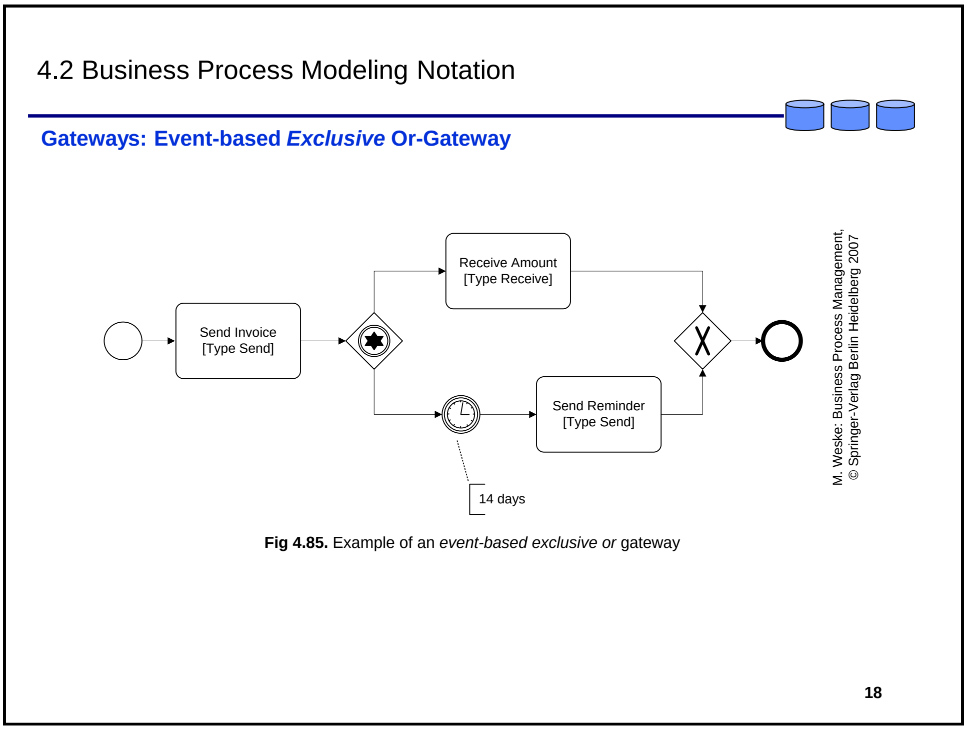
M. Weske: Business Process Management,
© Springer-Verlag Berlin Heidelberg 2007



4.2 Business Process Modeling Notation

Gateways

Fig 4.84. Gateway types in the BPMN, Object Management Group (2006)



4.2 Business Process Modeling Notation

Gateways: Event-based *Exclusive* Or-Gateway

Fig 4.85. Example of an *event-based exclusive or* gateway

4.2 Business Process Modeling Notation

Gateways: *Inclusive* Or-Gateway

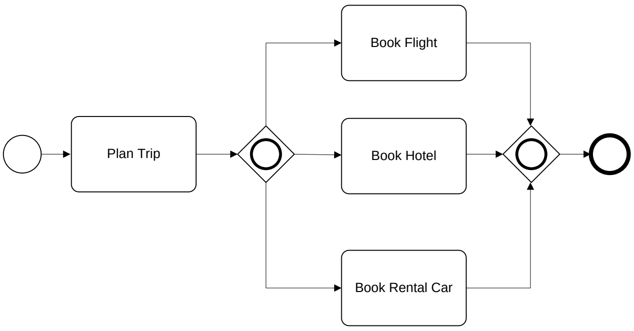


Fig 4.86. Example of an *inclusive or* gateway

4.2 Business Process Modeling Notation

Gateways: AND-Split and AND-Join Gateway

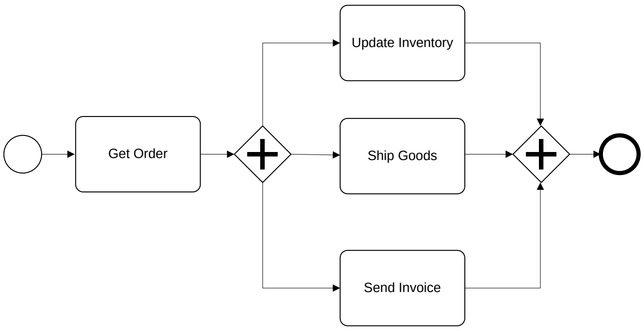


Fig 4.87. Example of an *and split* and *and join* gateway

4.2 Business Process Modeling Notation

Gateways: Default Paths

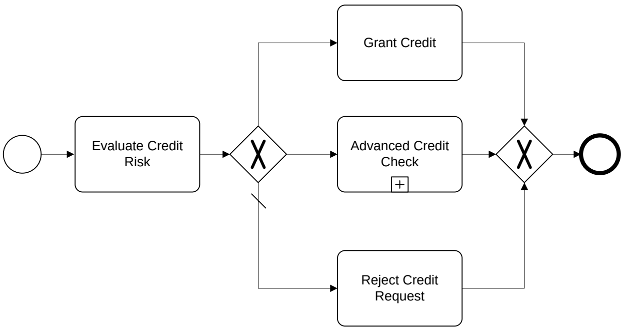


Fig 4.88. Sample business process with sequence flow and default sequence flow

4.2 Business Process Modeling Notation

Interacting Processes

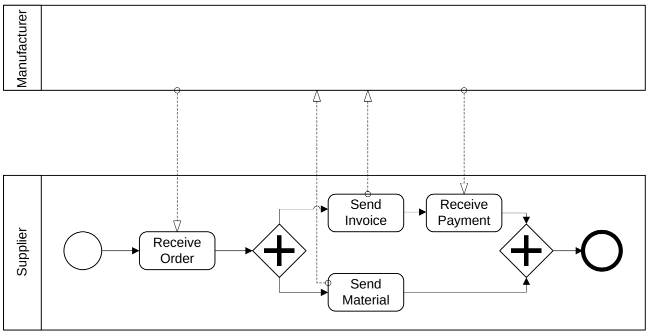


Fig 4.89. Business processes interacting by message flow

4.2 Business Process Modeling Notation

Interacting Processes

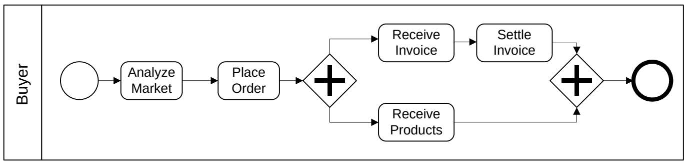


Fig 4.90. Private business process

M. Weske: Business Process Management,
© Springer-Verlag Berlin Heidelberg 2007

4.2 Business Process Modeling Notation

Interacting Processes

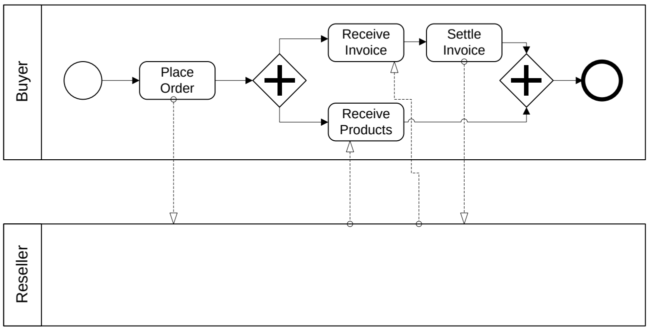
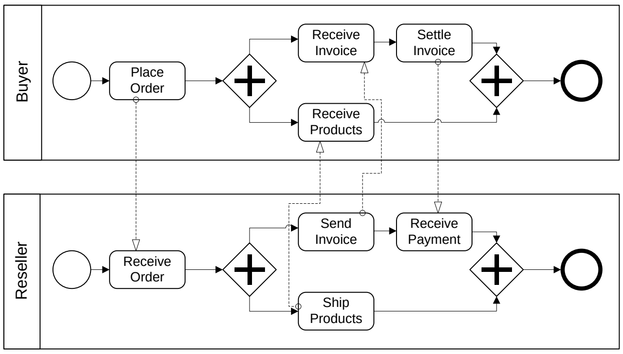


Fig 4.91. Public business process of buyer and corresponding abstract business process of reseller

M. Weske: Business Process Management,
© Springer-Verlag Berlin Heidelberg 2007

4.2 Business Process Modeling Notation

Interacting Processes

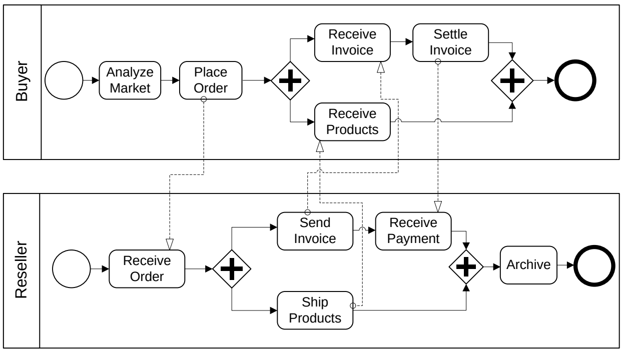


M. Weske: Business Process Management,
© Springer-Verlag Berlin Heidelberg 2007

Fig 4.92. Collaborative business process, representing the combined public business processes

4.2 Business Process Modeling Notation

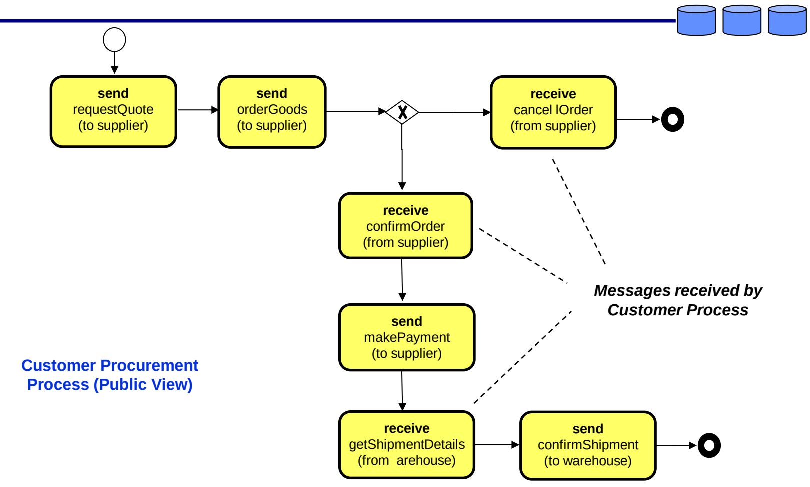
Interacting Processes



M. Weske: Business Process Management,
© Springer-Verlag Berlin Heidelberg 2007

Fig 4.93. Global business process, enriching collaborative business process with activities that do not expose communication behaviour

4.2 Business Process Modeling Notation



send requestQuote (to supplier)

send orderGoods (to supplier)

receive cancel lOrder (from supplier)

receive confirmOrder (from supplier)

send makePayment (to supplier)

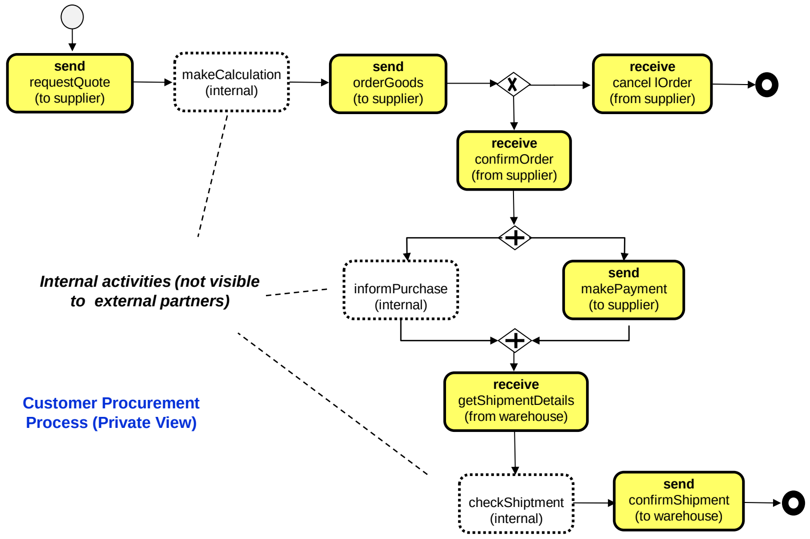
receive getShipmentDetails (from arehouse)

send confirmShipment (to warehouse)

Messages received by Customer Process

Customer Procurement Process (Public View)

4.2 Business Process Modeling Notation



send requestQuote (to supplier)

makeCalculation (internal)

send orderGoods (to supplier)

receive cancel lOrder (from supplier)

receive confirmOrder (from supplier)

informPurchase (internal)

send makePayment (to supplier)

receive getShipmentDetails (from warehouse)

checkShiptment (internal)

send confirmShipment (to warehouse)

Internal activities (not visible to external partners)

Customer Procurement Process (Private View)

Outline

4.1 Motivation

4.2 Business Process Modeling Notation (BPMN)

4.3 Modeling Process Choreographies

4.4 Process Choreography Design

4.5 Process Choreography Implementation

4.6 Business Process Execution Language (BPEL)

4.7 Summary and Current Research

References

4.3 Modeling Process Choreographies

- □ BPMN suitable for modeling process choreographies
- □ Example:

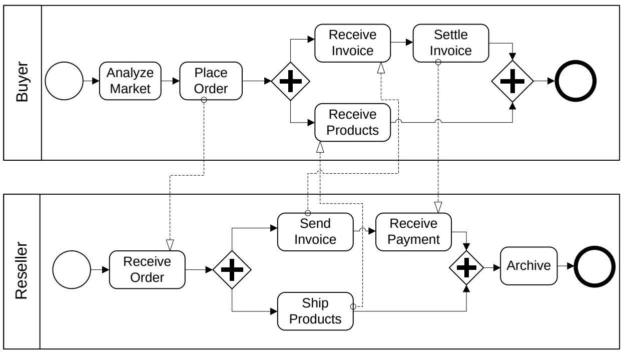


Fig 4.93. Global business process, enriching collaborative business process with activities that do not expose communication behaviour

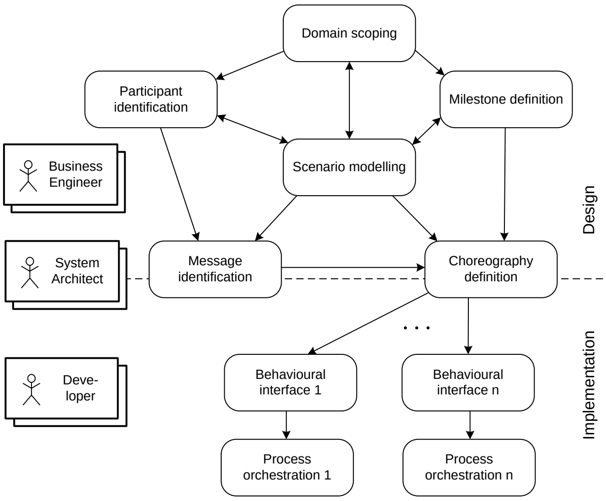
M. Weske: Business Process Management,
© Springer-Verlag Berlin Heidelberg 2007

4.3 Modeling Process Choreographies

- One basic challenge: Ensuring correctness of a modeled choreography; i.e., compatibility of the interacting process orchestrations
- Example of a choreography model „containing“ a deadlock

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing, University of Ulm (2009) and
M. Weske: Business Process Management. © Springer-Verlag Berlin Heidelberg (2007)

4.3 Modeling Process Choreographies



M. Weske: Business Process Management,
© Springer-Verlag Berlin Heidelberg 2007

Fig 5.4. Phases during choreography design and implementation

Outline

4.1 Motivation

4.2 Business Process Modeling Notation (BPMN)

4.3 Modeling Process Choreographies

4.4 Process Choreography Design

4.5 Process Choreography Implementation

4.6 Business Process Execution Language (BPEL)

4.7 Summary and Current Research

References

4.4 Process Choreography Design

1. **High-level Structure Design**
   - Identifying participant roles and their communication structure
   - Conducted during the *participant identification phase*
2. **High-level Behavioral Design**
   - Specifying the milestones of the collaboration and the order in which they are reached
   - Conducted during the *milestone definition phase*
3. **Collaboration Scenarios**
   - Refining high-level choreographies by introducing dedicated collaboration scenarios that relate the reaching of milestones to the communication between participating roles
   - Developed in the *choreography definition phase*, based on scenarios informally specified during *scenario modelling*
4. **Behavioral Interfaces**
   - Deriving a behavioral interface for each participant role from the collaboration scenarios

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing, University of Ulm (2009) and
M. Weske: Business Process Management. © Springer-Verlag Berlin Heidelberg (2007)

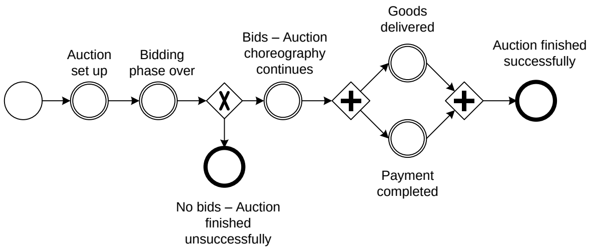
4.4.1 High-Level Design

- **High-level structure diagram for participants in a *bidding scenario***
- **High-level behavioral model for *bidding scenario* represented by milestones**

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing, University of Ulm (2009) and
M. Weske: Business Process Management. © Springer-Verlag Berlin Heidelberg (2007)

4.4.1 High-Level Design

- A certain milestone might be not reached in a certain conversation
- This situation occurs in the bidding scenario, for example, if no single bid is placed during the acution!
- High-level behavioral model for *bidding scenario* with different outcomes

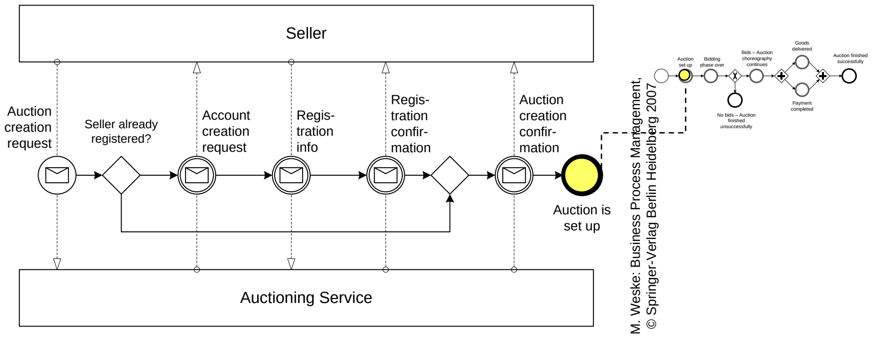


M. Weske: Business Process Management,
© Springer-Verlag Berlin Heidelberg 2007

Fig 5.7. High-level behavioural model for bidding scenario, with different outcomes

4.4.2 Collaboration Scenarios

- **Collaboration scenario**: reaching milestones through interactions (i.e., message exchanges)

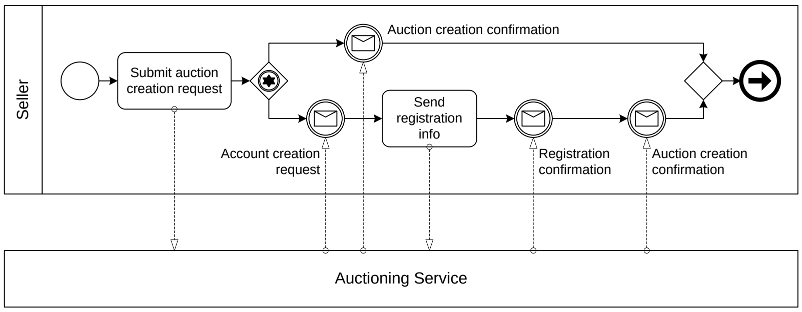

Seller
Auction creation request
Seller already registered?
Account creation request
Regis-tration info
Regis-tration confir-mation
Auction creation confir-mation
Auction is set up
Auctioning Service
Auction set up
Bidding phase over
Bids – Auction choreography continues
Goods delivered
Payment completed
Auction finished successfully
No bids – Auction finished unsuccessfully

M. Weske: Business Process Management,
© Springer-Verlag Berlin Heidelberg 2007

Fig 5.8. Collaboration scenario: reaching milestones through interactions

4.4.2 Collaboration Scenarios

- Behavioral interface for participant role `Seller` (partial view)


Seller
Submit auction creation request
Auction creation confirmation
Account creation request
Send registration info
Registration confirmation
Auction creation confirmation
Auctioning Service

M. Weske: Business Process Management,
© Springer-Verlag Berlin Heidelberg 2007

Fig 5.9. Behavioural interface for seller

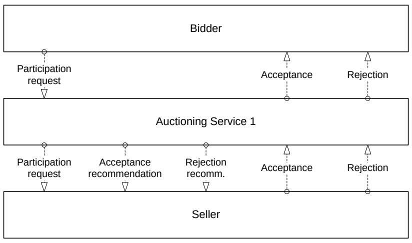
4.4.3. Compatibility

- □ The design of a process choreography needs to ensure that the process orchestrations of the participants play together well in the overall collaboration
- □ **Compatibility** → Ability of a set of participants to interact successfully according to a given process choreography!
- □ Sources of incompatibility:
  - ❍ Different messages are used in a collaboration and one participant does not understand the content of a message sent by another participant
  - ❍ Wrong or misaligned interactions; e.g., if a participant expects a notification at some point in its process before it can proceed and none of the other participants sends such notification message → **DEADLOCK**
- □ Compatibility of interacting processes aims at avoiding such undesired behavior; i.e., to exclude errorneous interactions between orchestrations

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing, University of Ulm (2009) and
M. Weske: Business Process Management. © Springer-Verlag Berlin Heidelberg (2007)

4.4.3.1 Example

- □ **Interactions between participants in auctioning scenario**
  - ❍ **A potential bidder must be accepted for participation before she can place her bid.**
  - ❍ **The bidder needs to send a *Participation request* to the auctioning service**
  - ❍ **As response the latter can send an *Acceptance* notification or a *Rejection* notification**
  - ❍ **In some cases the seller is requested to make the final decision on whether or not a bidder shall be accepted.**
    - – **To perform this interaction, the auctioning service forwards the request of the bidder to the seller**
    - – **It might also give a recommendation for accepting / rejecting the bidder**
    - – **The seller can send a notification about his decision back to the auctioning service**



M. Weske: Business Process Management.
© Springer-Verlag Berlin Heidelberg 2007

- □ ***Participants represented by pools that interact by sending and receiving messages***
- □ ***Above figure does not show any behavioral dependencies between the different message exchanges***

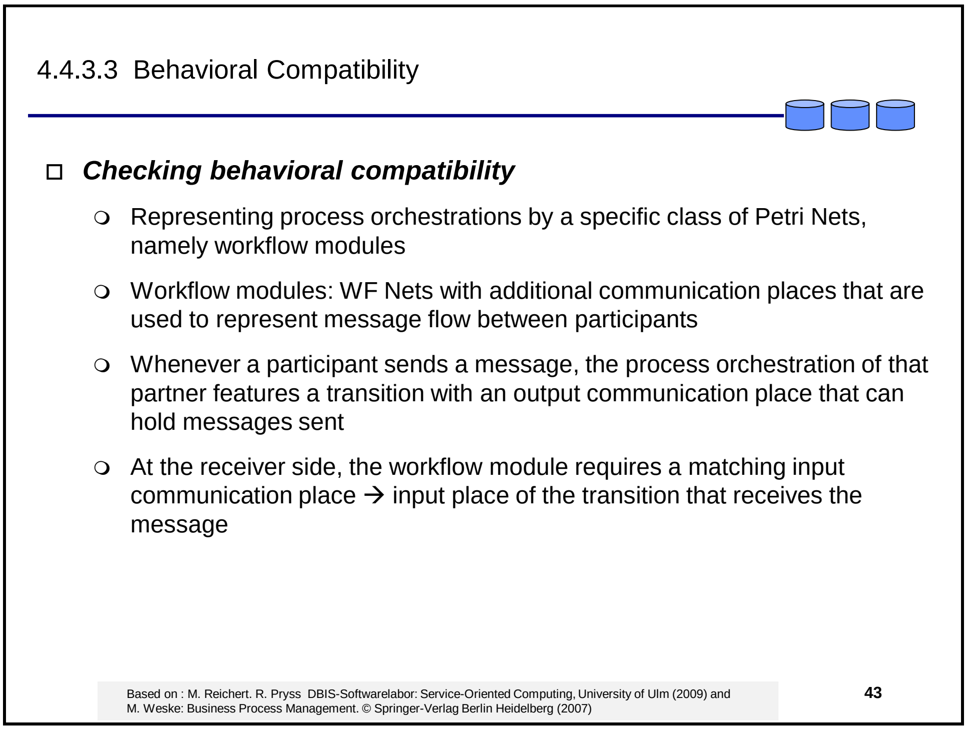
Chapter based on : M. Reichert. R. Pryss: DBIS-Softwarelabor: Service-Oriented Computing, University of Ulm (2009)

4.4.3.2 Structural Compatibility

- □ ***Weak structural compatibility***
  - ❍ Messages that can be sent by a participant correspond to messages that other participants can receive!
  - ❍ Ensures that all messages sent can actually be received by participants
  - ❍ Does not forbid that participants may receive additional messages not sent by any of the other participants (in the given choreography)
- □ ***Strong structural compatibility***
  - ❍ For every message that can be sent there is a participant that can receive it, and
  - ❍ for every message that can be received there is a participant that can send it

4.4.3.3 Behavioral Compatibility

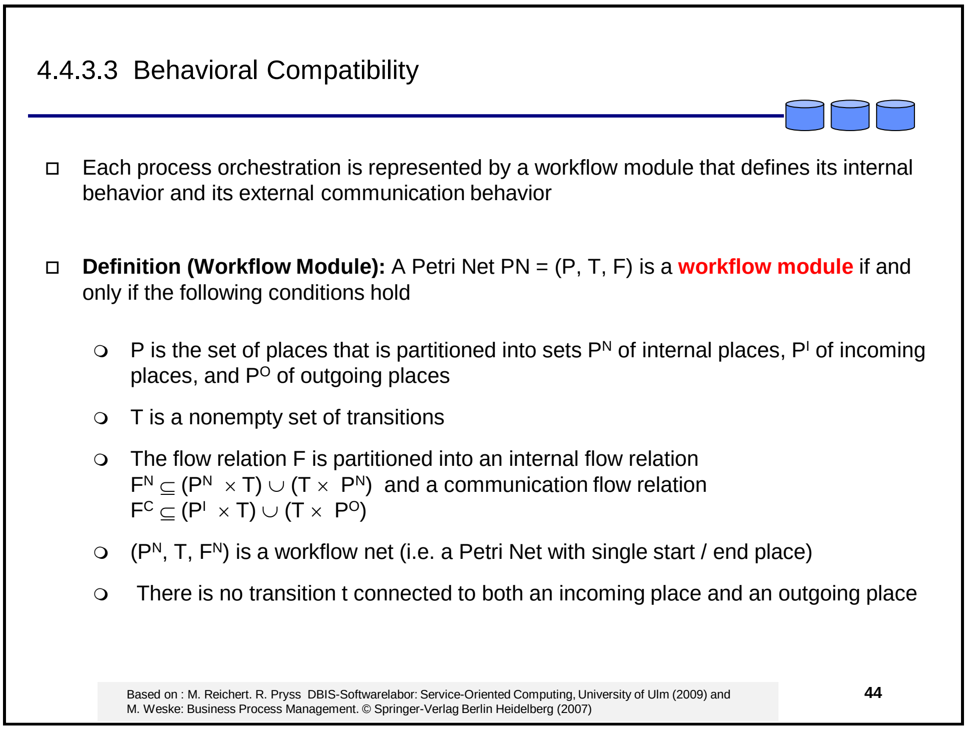
- □ ***Behavioral compatibility***
  - ❍ Considers behavioral dependencies (i.e., control flow) between interaction instances of a conversation as well
  - ❍ The process orchestrations of the interacting partners are interconnected, and the resulting process structure is analyzed
  - ❍ Such analysis of the dynamic behavior requires a formal, unambiguos representation



4.4.3.3 Behavioral Compatibility

- ***Checking behavioral compatibility***
  - Representing process orchestrations by a specific class of Petri Nets, namely workflow modules
  - Workflow modules: WF Nets with additional communication places that are used to represent message flow between participants
  - Whenever a participant sends a message, the process orchestration of that partner features a transition with an output communication place that can hold messages sent
  - At the receiver side, the workflow module requires a matching input communication place → input place of the transition that receives the message

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing, University of Ulm (2009) and M. Weske: Business Process Management. © Springer-Verlag Berlin Heidelberg (2007)



4.4.3.3 Behavioral Compatibility

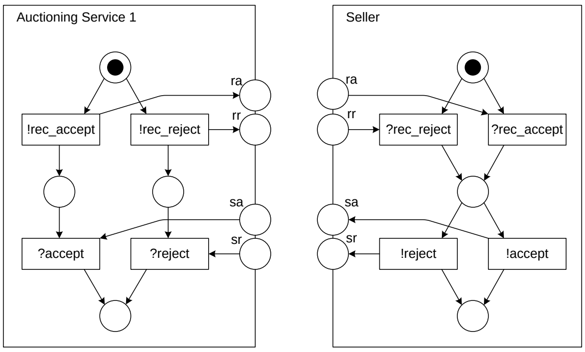
- Each process orchestration is represented by a workflow module that defines its internal behavior and its external communication behavior

- **Definition (Workflow Module):** A Petri Net PN = (P, T, F) is a **workflow module** if and only if the following conditions hold
  - P is the set of places that is partitioned into sets P^N of internal places, P^I of incoming places, and P^O of outgoing places
  - T is a nonempty set of transitions
  - The flow relation F is partitioned into an internal flow relation $F^N \subseteq (P^N \times T) \cup (T \times P^N)$ and a communication flow relation $F^C \subseteq (P^I \times T) \cup (T \times P^O)$
  - (P^N, T, F^N) is a workflow net (i.e. a Petri Net with single start / end place)
  - There is no transition t connected to both an incoming place and an outgoing place

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing, University of Ulm (2009) and M. Weske: Business Process Management. © Springer-Verlag Berlin Heidelberg (2007)

4.4.3.3 Behavioral Compatibility

- □ **The following figure shows workflow modules for participants `Auctioning Service 1` and `Seller`**
- □ **For the sake of readability, the workflow modules only represent a small part of the auctioning and seller process orchestrations**

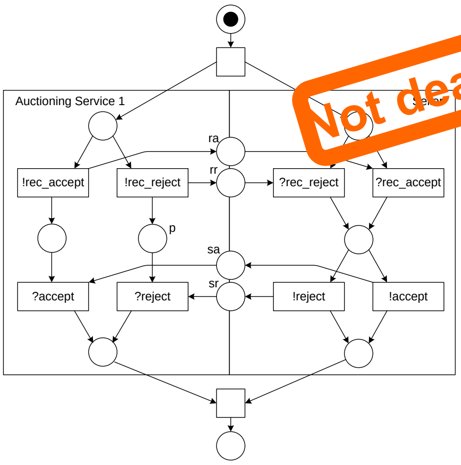


M. Weske: Business Process Management.
© Springer-Verlag Berlin Heidelberg 2007

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing, University of Ulm (2009) and
M. Weske: Business Process Management. © Springer-Verlag Berlin Heidelberg (2007)

4.4.3.3 Behavioral Compatibility

- □ **Workflow net as composition of the workflow modules (requires strong structural compatibility of the workflow modules)**

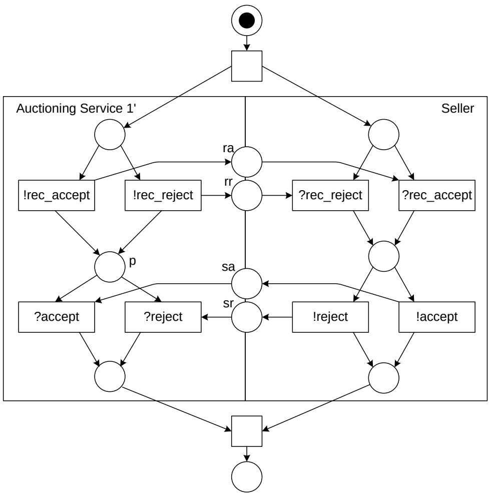


Not deadlock-free

M. Weske: Business Process Management.
© Springer-Verlag Berlin Heidelberg 2007

4.4.3.3 Behavioral Compatibility

- □ **Workflow modules that are compatible**

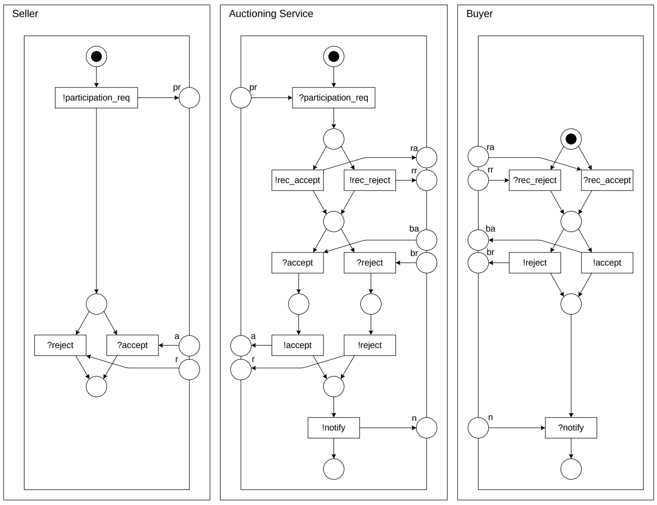


M. Weske: Business Process Management,
© Springer-Verlag Berlin Heidelberg 2007

Fig 5.14. Workflow modules that are compatible

4.4.3.3 Behavioral Compatibility

- □ **A larger part of the collaboration is depicted in the following figure:**



M. Weske: Business Process Management,
© Springer-Verlag Berlin Heidelberg 2007

Fig 5.15. Behavioural interfaces: getting a participation permission

Outline

4.1 Motivation

4.2 Business Process Modeling Notation (BPMN)

4.3 Modeling Process Choreographies

4.4 Process Choreography Design

4.5 Process Choreography Implementation

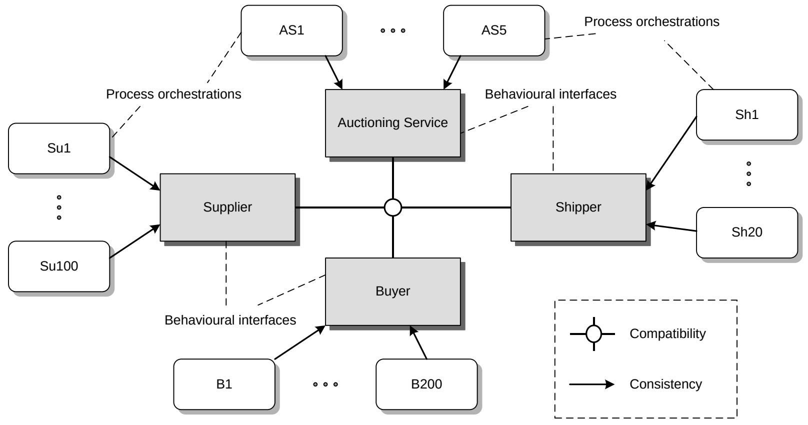
4.6 Business Process Execution Language (BPEL)

4.7 Summary and Current Research

References

4.5 Process Choreography Implementation

□ **Participants and roles in an auctioning scenario**



M. Weske: Business Process Management,
© Springer-Verlag Berlin Heidelberg 2007

Fig 5.16. Participants and roles in a reverse auctioning scenario

4.5 Process Choreography Implementation

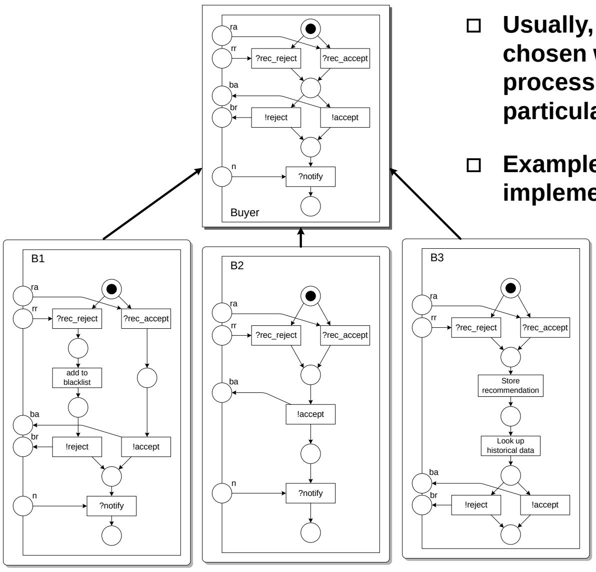


Fig 5.17. Alternative implementations for buyer role

M. Weske: Business Process Management,
© Springer-Verlag Berlin Heidelberg 2007

- □ **Usually, different variants may be chosen when implementing the process orchestration of a particular role**
- □ **Example: Alternative implementations for buyer role**

51

Outline

4.1 Motivation

4.2 Business Process Modeling Notation (BPMN)

4.3 Modeling Process Choreographies

4.4 Process Choreography Design

4.5 Process Choreography Implementation

4.6 Business Process Execution Language (BPEL)

4.7 Summary and Current Research

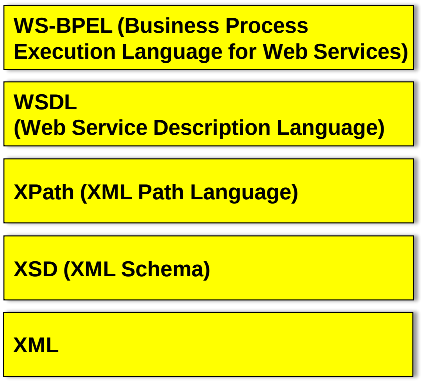
References

52

4.6 Business Process Execution Languages (BPEL)

WS-BPEL is...

- ❐ A standardized language to define executable processes
- ❐ Not bound to a certain implementation
- ❐ Based on WSDL and other XML standards
  - ❍ WSDL → interface description of involved Web Services
  - ❍ Data context and business rules are defined by XML schemata and Xpath expressions


WS-BPEL (Business Process Execution Language for Web Services)
WSDL (Web Service Description Language)
XPath (XML Path Language)
XSD (XML Schema)
XML

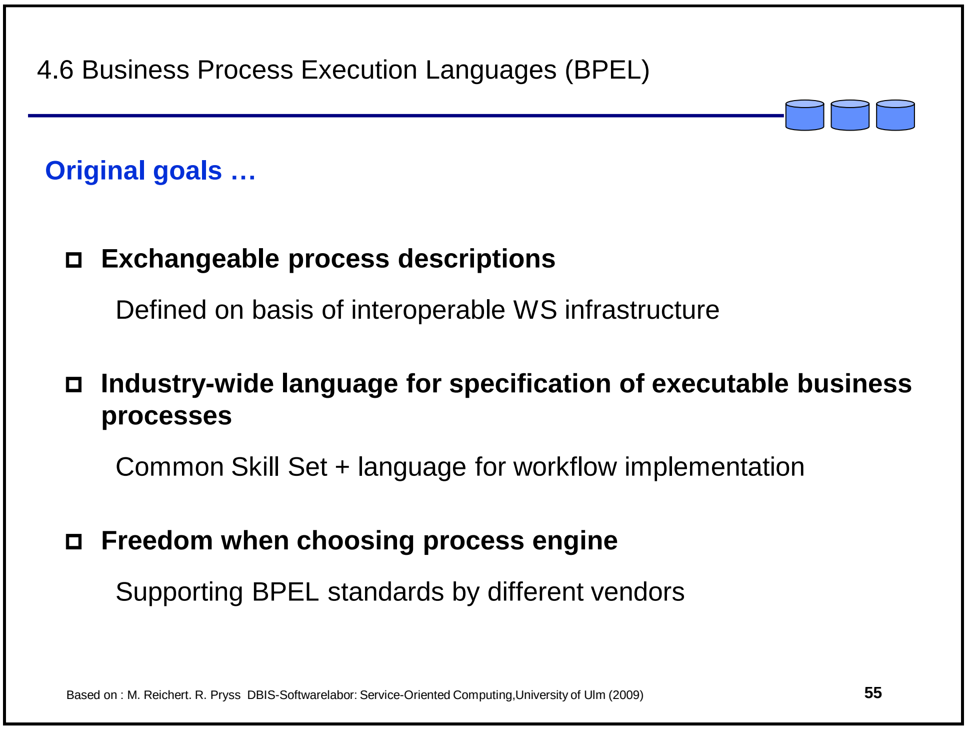
Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

4.6 Business Process Execution Languages (BPEL)

History

- ◻ WS-BPEL (also: BPEL; BPEL4WS):
  ***B**usiness **P**rocess **E**xecution **L**anguage **for** **W**eb **S**ervices*
- ◻ XML based description language for executable processes
  - ❍ Standard for Web Service Composition
  - ❍ Based on WSDL and SOAP
- ◻ Current status:
  - ❍ 2002: first specification (IBM, Microsoft, BEA)
  - ❍ Merging two languages: IBM WSFL + Microsoft XLANG
  - ❍ 2003: submission of version V 1.1 to OASIS;
  - ❍ 2007: Version 2.0
  - ❍ Platt forms: WebSphere Process Server, BizTalk, Oracle BPEL Process Manager, Intalio, …
  - ❍ Extensions: BPEL4People, BPEL4J(ava), …

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)



4.6 Business Process Execution Languages (BPEL)

Original goals …

- **Exchangeable process descriptions**

  Defined on basis of interoperable WS infrastructure

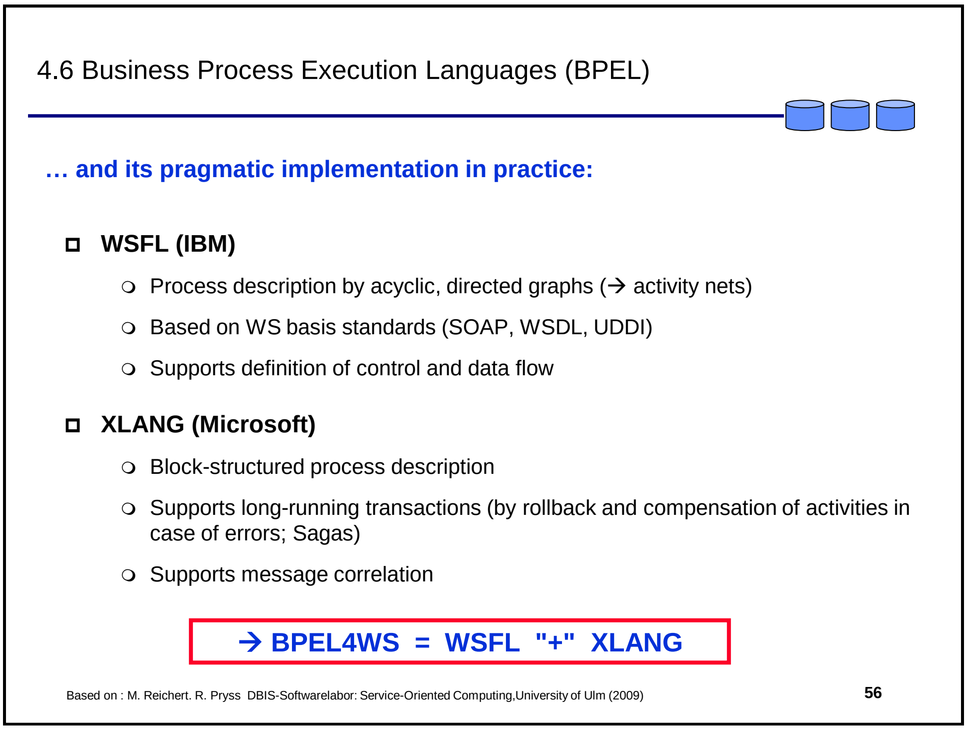
- **Industry-wide language for specification of executable business processes**

  Common Skill Set + language for workflow implementation

- **Freedom when choosing process engine**

  Supporting BPEL standards by different vendors

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)



4.6 Business Process Execution Languages (BPEL)

… and its pragmatic implementation in practice:

- **WSFL (IBM)**
  - Process description by acyclic, directed graphs (→ activity nets)
  - Based on WS basis standards (SOAP, WSDL, UDDI)
  - Supports definition of control and data flow
- **XLANG (Microsoft)**
  - Block-structured process description
  - Supports long-running transactions (by rollback and compensation of activities in case of errors; Sagas)
  - Supports message correlation

→ BPEL4WS = WSFL "+" XLANG

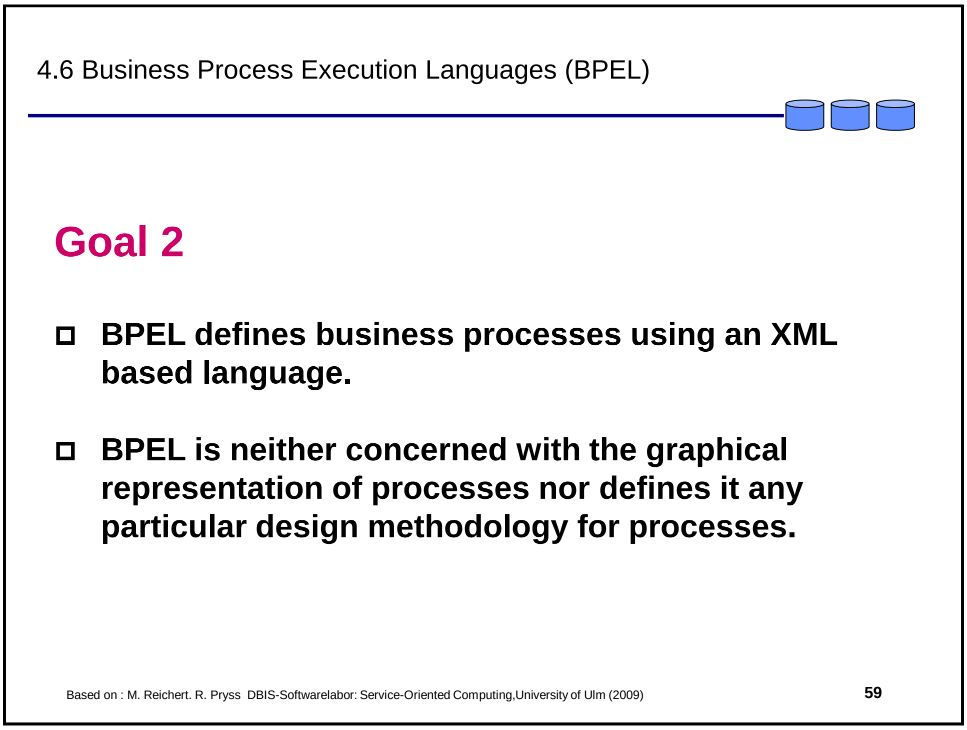
Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

Which were the technical goals when designing WS-BPEL?

Goal 1

- □ **Define business processes that interact with external entities through web service operations defined using WSDL**
- □ **Interactions are abstract in the sense that the dependence is on `portType` definitions, not on `port` definitions.**
- □ **BPEL has to stay compatible with WSDL.**

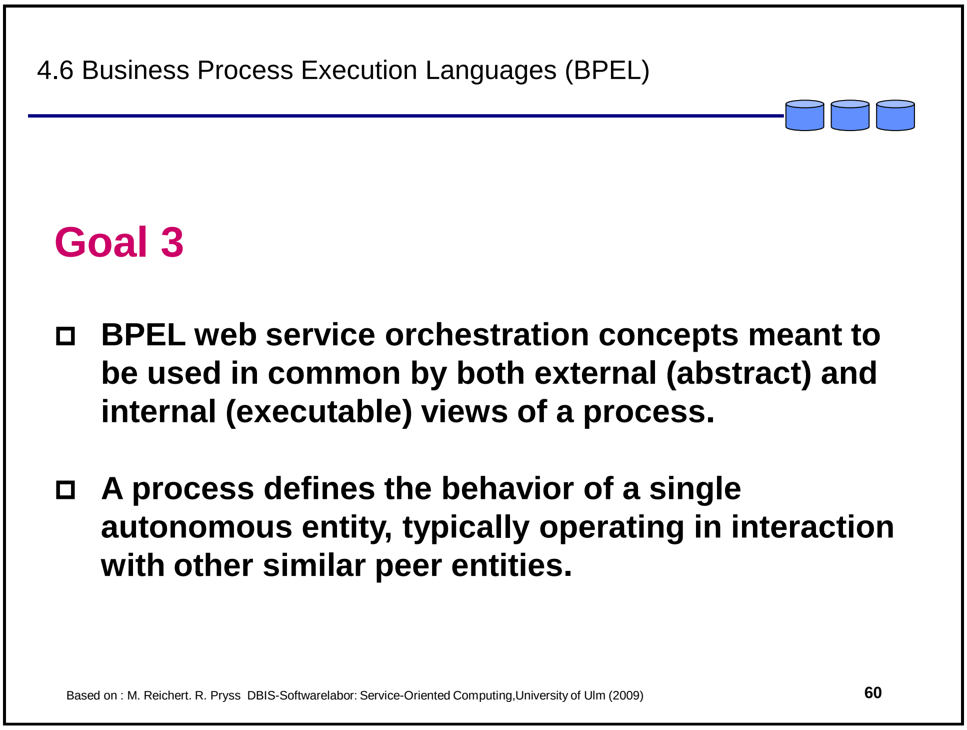
Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)



4.6 Business Process Execution Languages (BPEL)

Goal 2

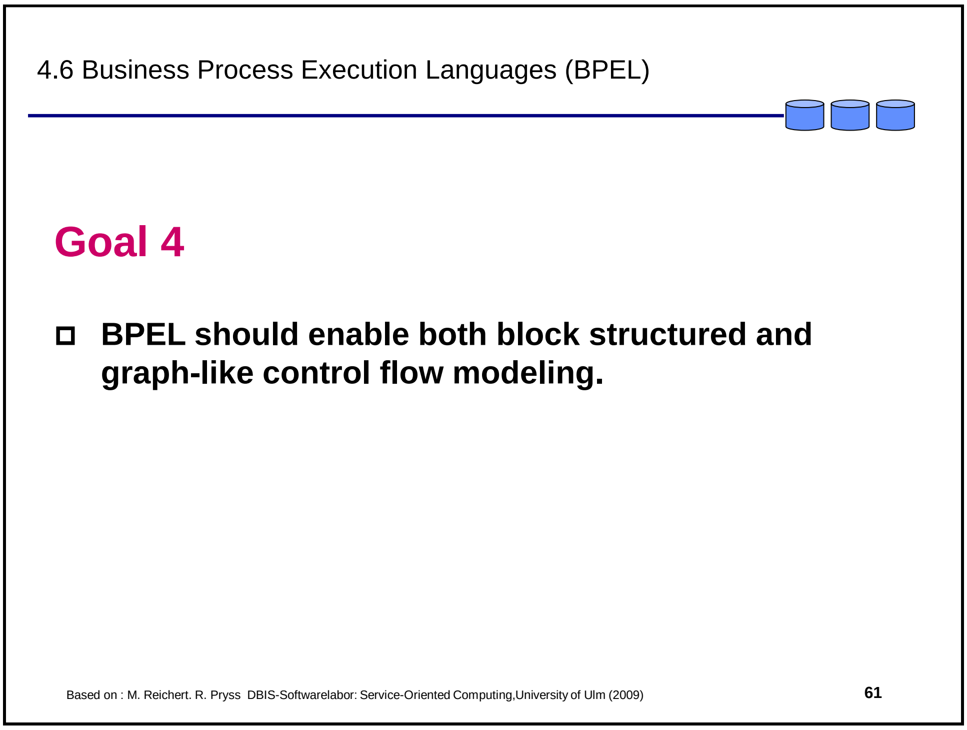
- □ **BPEL defines business processes using an XML based language.**
- □ **BPEL is neither concerned with the graphical representation of processes nor defines it any particular design methodology for processes.**



4.6 Business Process Execution Languages (BPEL)

Goal 3

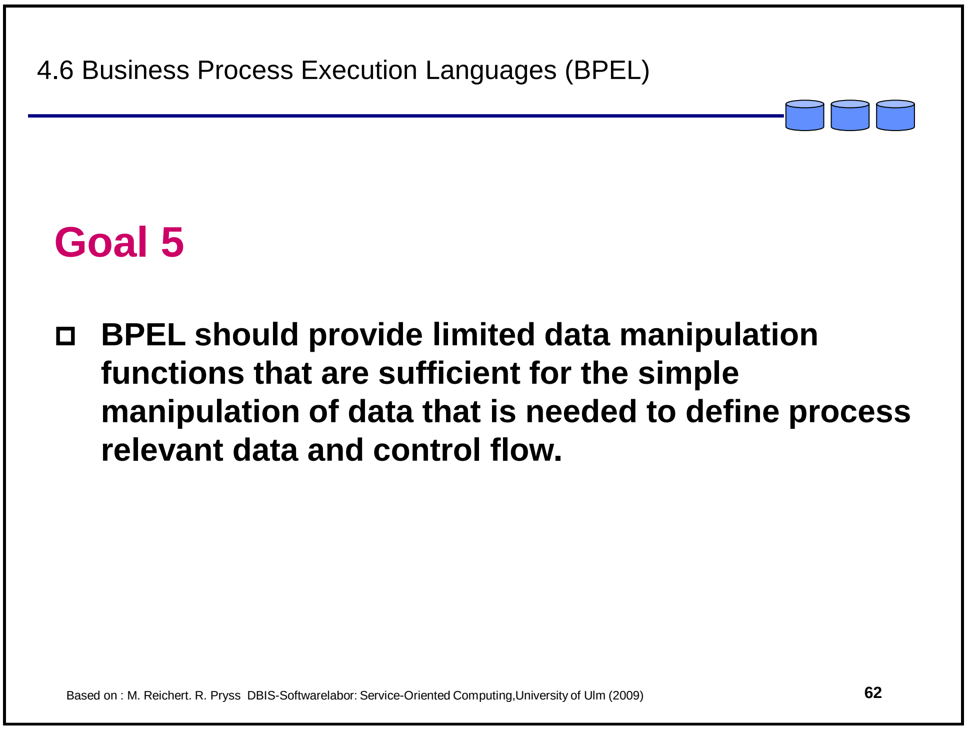
- □ **BPEL web service orchestration concepts meant to be used in common by both external (abstract) and internal (executable) views of a process.**
- □ **A process defines the behavior of a single autonomous entity, typically operating in interaction with other similar peer entities.**



Goal 4

- BPEL should enable both block structured and graph-like control flow modeling.

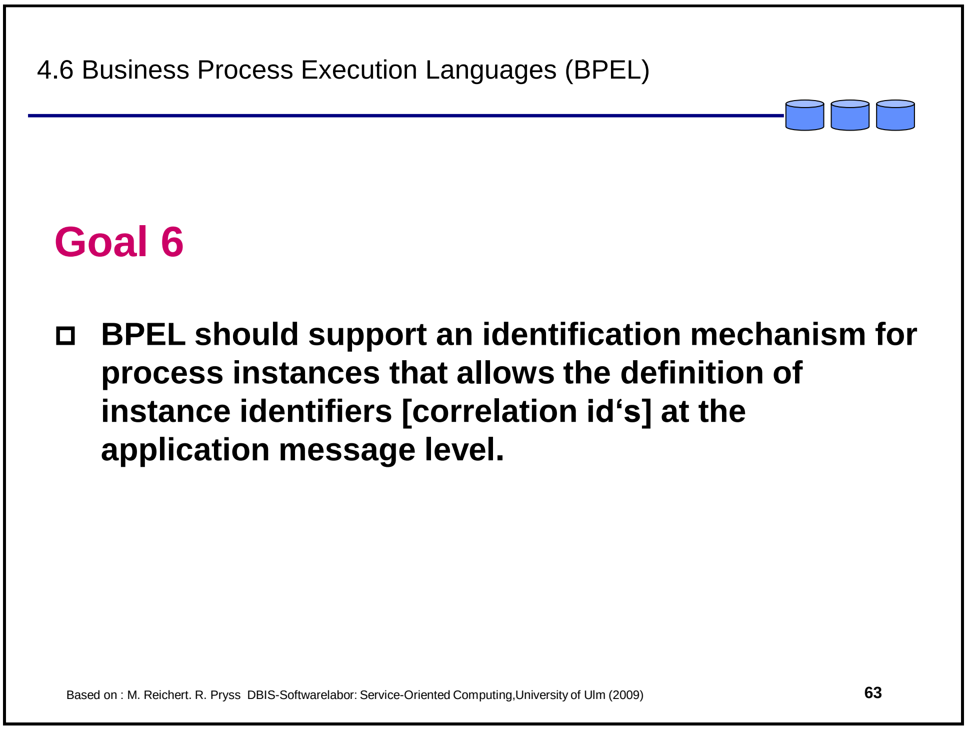
Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)



Goal 5

- BPEL should provide limited data manipulation functions that are sufficient for the simple manipulation of data that is needed to define process relevant data and control flow.

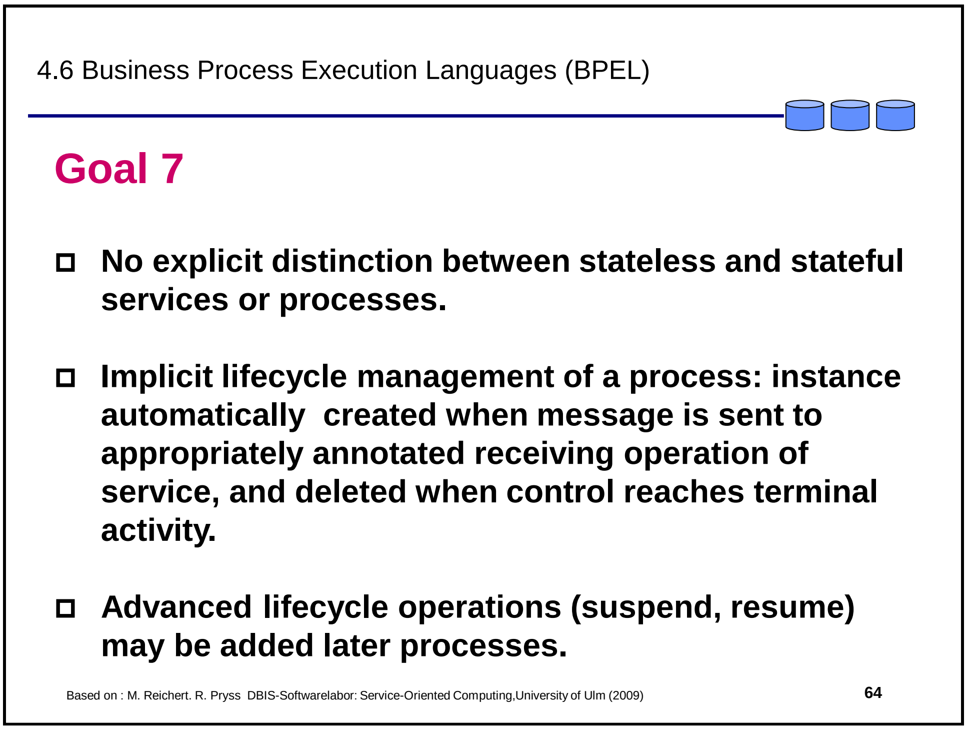
Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)



4.6 Business Process Execution Languages (BPEL)

Goal 6

- BPEL should support an identification mechanism for process instances that allows the definition of instance identifiers [correlation id‘s] at the application message level.



4.6 Business Process Execution Languages (BPEL)

Goal 7

- No explicit distinction between stateless and stateful services or processes.
- Implicit lifecycle management of a process: instance automatically created when message is sent to appropriately annotated receiving operation of service, and deleted when control reaches terminal activity.
- Advanced lifecycle operations (suspend, resume) may be added later processes.

Goal 8

- BPEL should define a long-running transaction model that is based on practically proven techniques like compensation actions and scoping to support failure recovery for parts of long-running business processes.

Goal 9

- BPEL should use web services as the model for process decomposition and assembly.

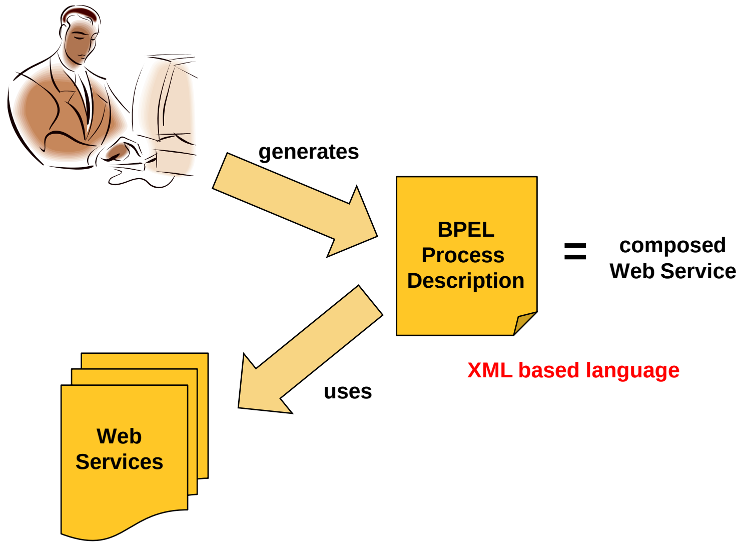
4.6 Business Process Execution Languages (BPEL)

Goal 10

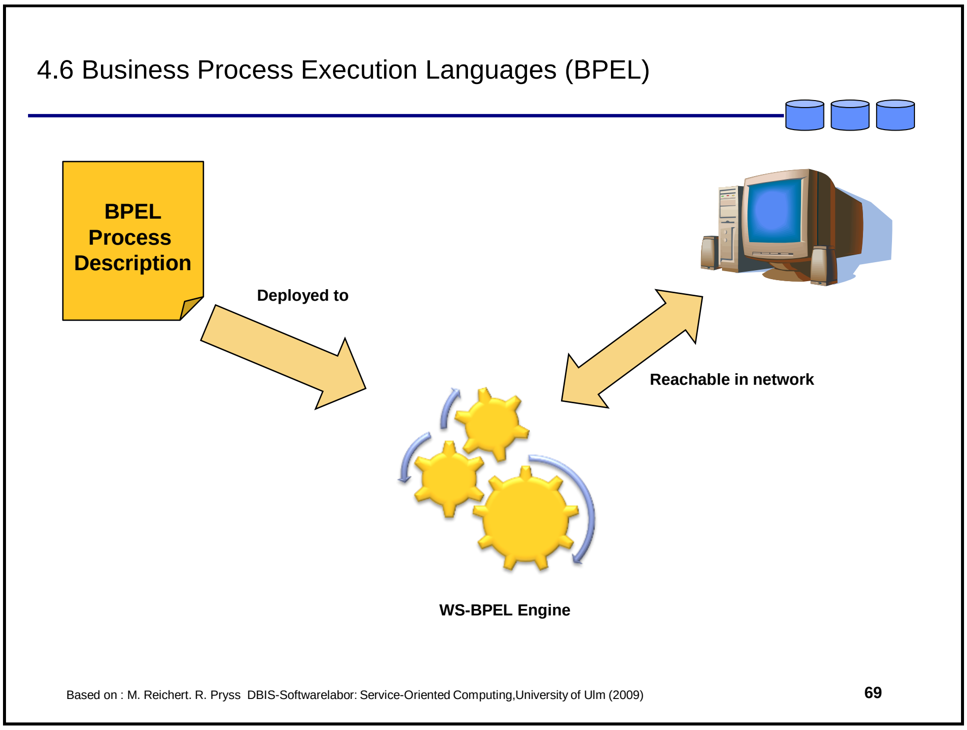
- BPEL should build on compatible Web services standards and standards proposals as much as possible in a composable and modular manner.
- Only if no appropriate standard or standards proposal is available for a particular requirement, an appropriate specification should [be] developed within the BPEL specification or as a separate Web services standards proposal.

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

4.6 Business Process Execution Languages (BPEL)



Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)



4.6 Business Process Execution Languages (BPEL)

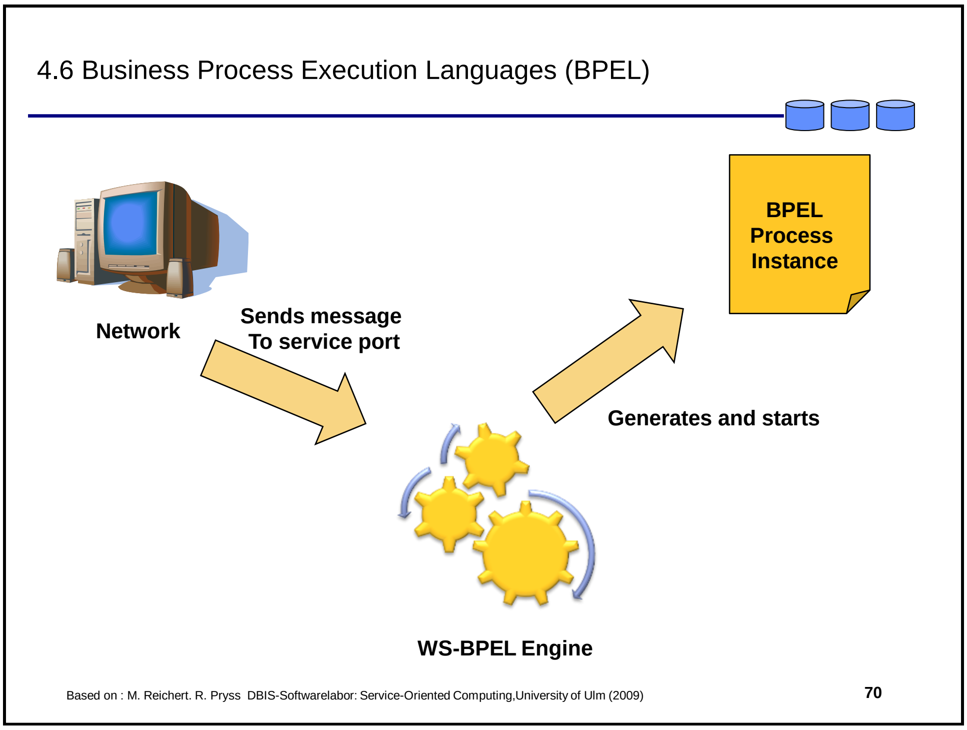
BPEL Process Description

Deployed to

Reachable in network

WS-BPEL Engine

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)



4.6 Business Process Execution Languages (BPEL)

Network

Sends message
To service port

BPEL Process Instance

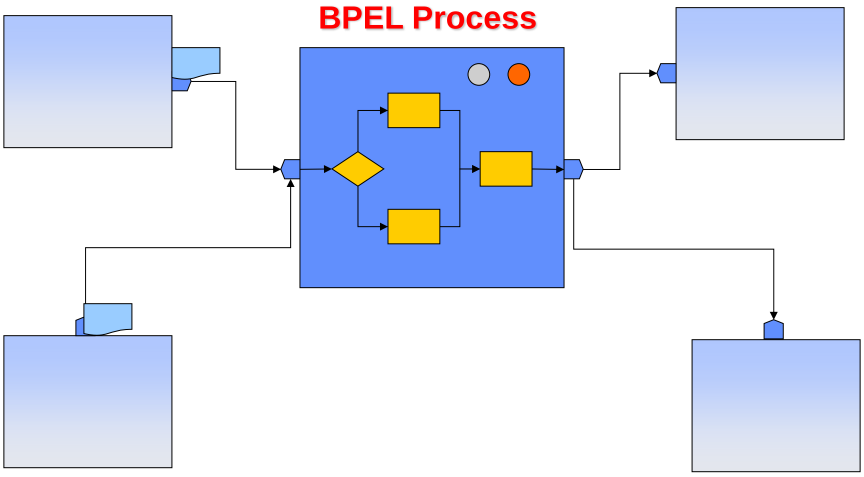
Generates and starts

WS-BPEL Engine

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

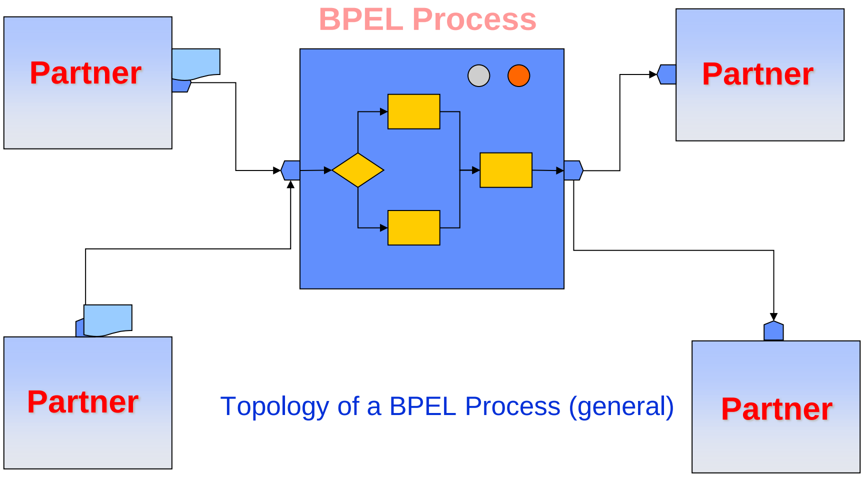
4.6 Business Process Execution Languages (BPEL)

Topology of a BPEL Process (general)



Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

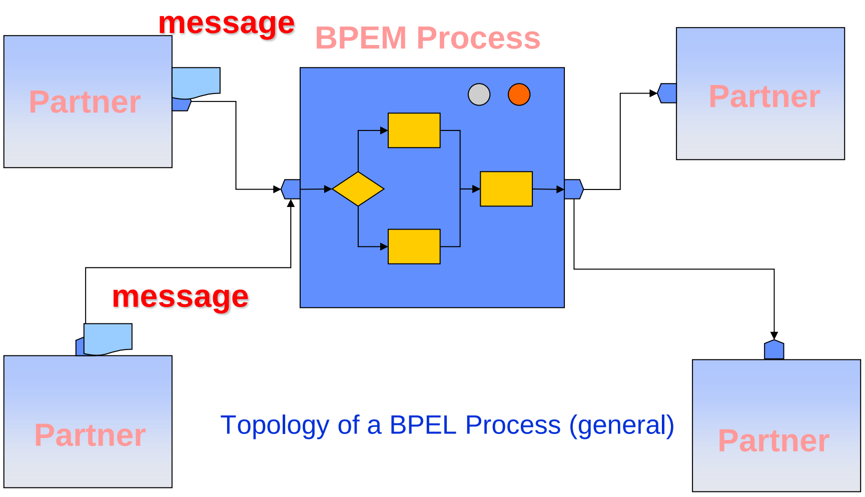
4.6 Business Process Execution Languages (BPEL)



Topology of a BPEL Process (general)

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

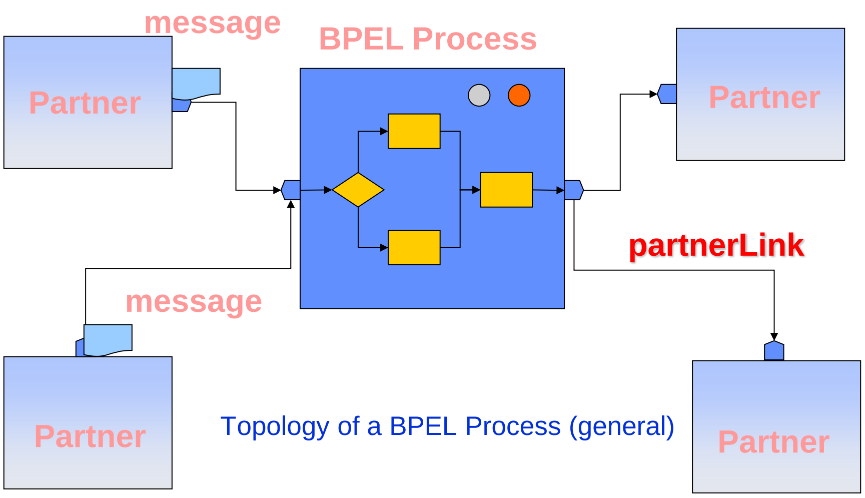
4.6 Business Process Execution Languages (BPEL)



Topology of a BPEL Process (general)

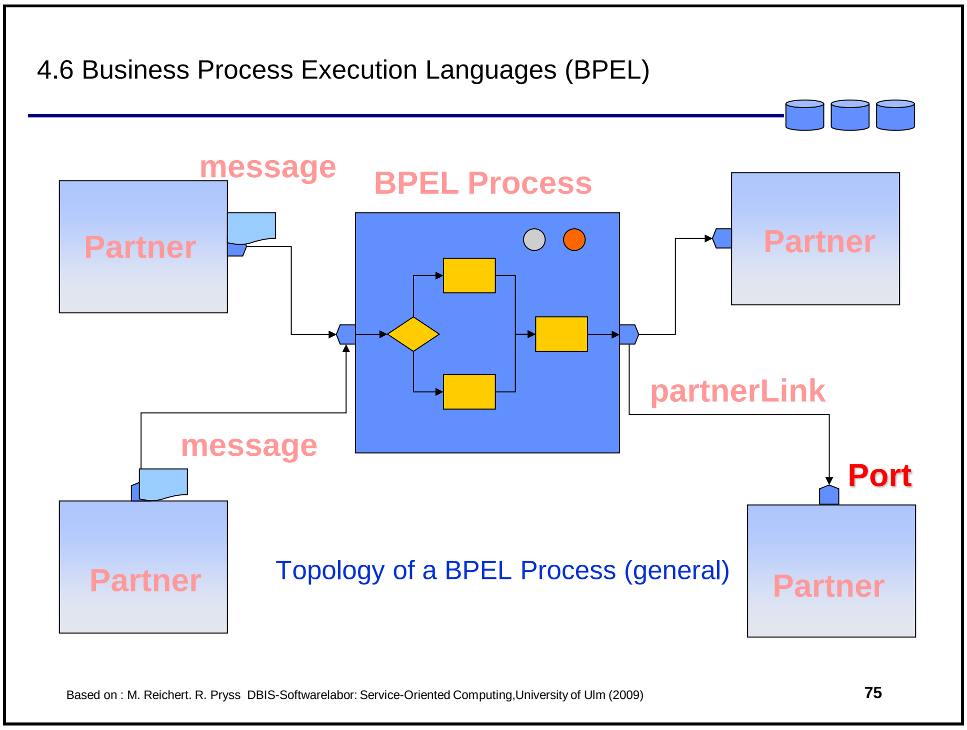
Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

4.6 Business Process Execution Languages (BPEL)



Topology of a BPEL Process (general)

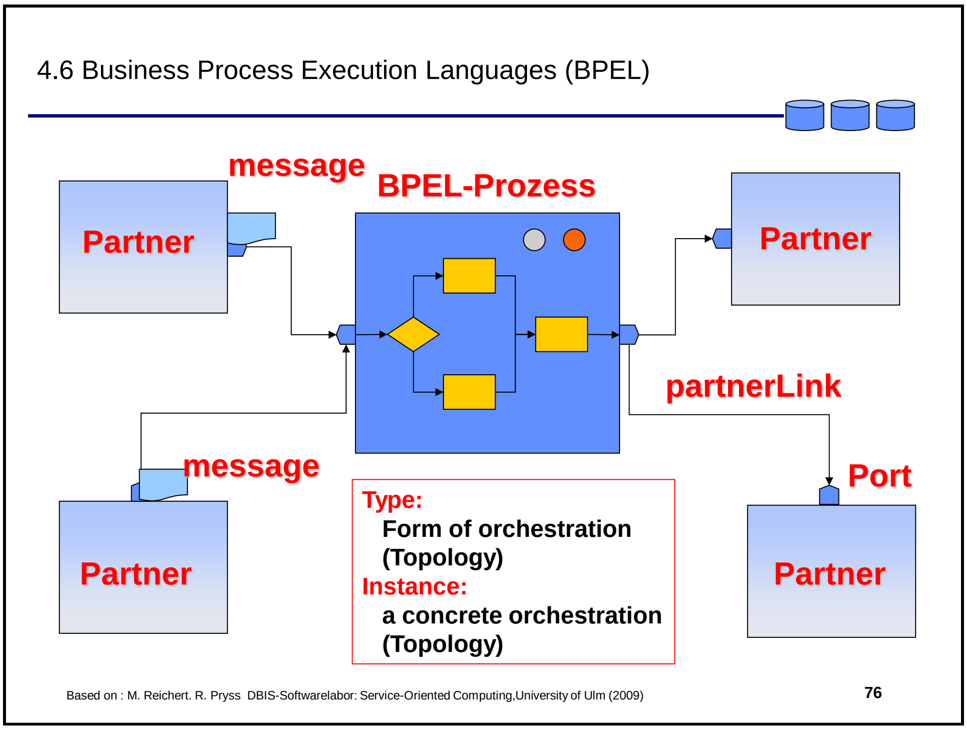
Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)



4.6 Business Process Execution Languages (BPEL)

Topology of a BPEL Process (general)

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)



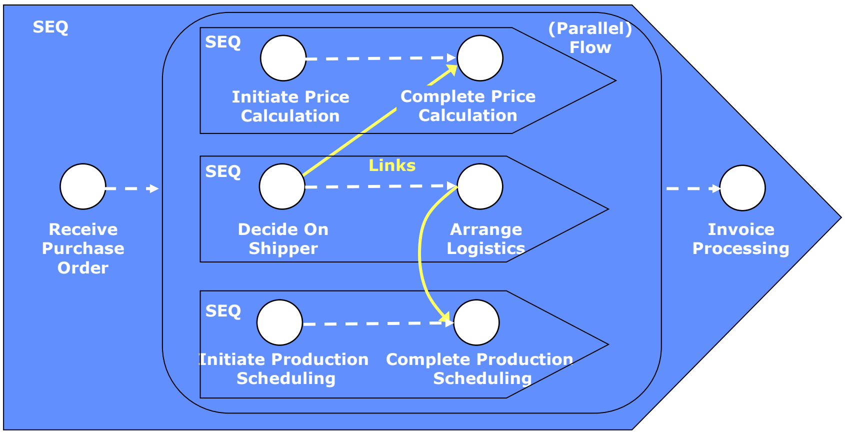
4.6 Business Process Execution Languages (BPEL)

Type:
Form of orchestration (Topology)
Instance:
a concrete orchestration (Topology)

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

4.6 Business Process Execution Languages (BPEL)

Basic description elements:


SEQ
SEQ
(Parallel) Flow
Initiate Price Calculation
Complete Price Calculation
Links
SEQ
Receive Purchase Order
Decide On Shipper
Arrange Logistics
Invoice Processing
SEQ
Initiate Production Scheduling
Complete Production Scheduling

Application Example

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

4.6 Business Process Execution Languages (BPEL)

Structure and aspects of a process description with WS-BPEL

```
<process>
process header
<partners>
</partners>
<variables>
</variables>
<correlationSets>
</correlationSets>
<faultHandlers>
</faultHandlers>
<compensationHandler>
</compensationHandler>
       <eventHandlers>
</eventHandlers>
       <sequence>
</sequence>
</process>
```

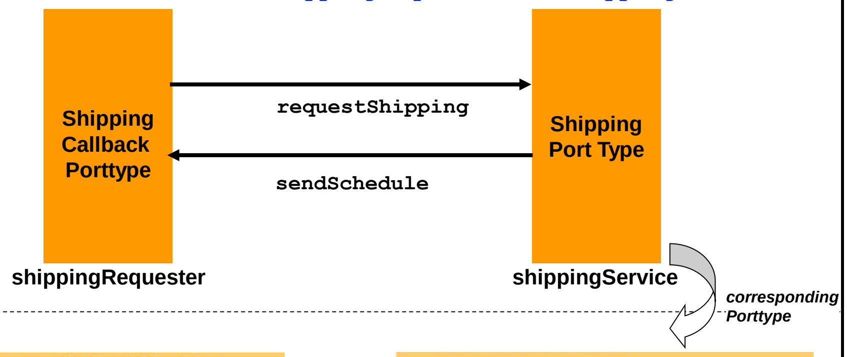
Description of Service Interactions:

With which partners does the BPEL Process interact and which roles does it take within these interactions?

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

4.6 Business Process Execution Languages (BPEL)

Example: Interaction between `shippingRequester` and `Shipping Service`



WSDL

```
<portType name="shippingCallbackPT">
   <operation name="sendSchedule">
      <input message="pos:scheduleMessage"/>
   </operation>
</portType>
```

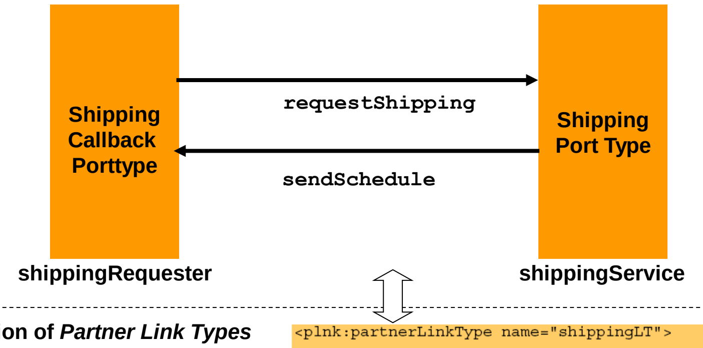
```
<portType name="shippingPT">
   <operation name="requestShipping">
      <input message="pos:shippingRequestMessage"/>
      <output message="pos:shippingInfoMessage"/>
      <fault name="cannotCompleteOrder"
             message="pos:orderFaultType"/>
   </operation>
</portType>
```

Callback Service **to be supported by requester**

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

4.6 Business Process Execution Languages (BPEL)

Example: Interaction between `shippingRequester` and `Shipping Service`



WSDL

Definition of *Partner Link Types* (in WSDL file):

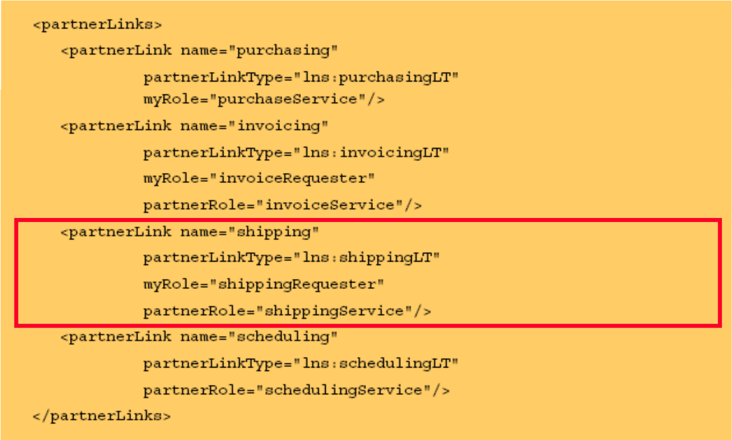
- **Definition of 2 roles of an interaction**
- **Listing of port types that support each role / each partner such that interaction can take place**

```
<plnk:partnerLinkType name="shippingLT">
   <plnk:role name="shippingService">
      <plnk:portType name="pos:shippingPT"/>
   </plnk:role>
   <plnk:role name="shippingRequester">
      <plnk:portType name="pos:shippingCallbackPT"/>
   </plnk:role>
</plnk:partnerLinkType>
```

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

4.6 Business Process Execution Languages (BPEL)

Linking a WS-BPEL Process with partners + determining the role that the process takes



```
<partnerLinks>
   <partnerLink name="purchasing"
            partnerLinkType="lns:purchasingLT"
            myRole="purchaseService"/>
   <partnerLink name="invoicing"
            partnerLinkType="lns:invoicingLT"
            myRole="invoiceRequester"
            partnerRole="invoiceService"/>
   <partnerLink name="shipping"
            partnerLinkType="lns:shippingLT"
            myRole="shippingRequester"
            partnerRole="shippingService"/>
   <partnerLink name="scheduling"
            partnerLinkType="lns:schedulingLT"
            partnerRole="schedulingService"/>
</partnerLinks>
```

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

4.6 Business Process Execution Languages (BPEL)

Process Variables and Data Flows: Structure and aspects of a process description with WSBPEL

```
<process>
process header
<partners>
</partners>
<variables>
</variables>
<correlationSets>
</correlationSets>
       <faultHandlers>
</faultHandlers>
<compensationHandler>
</compensationHandler>
       <eventHandlers>
</eventHandlers>
       <sequence>
</sequence>
</process>
```

Description of data and message flow between services

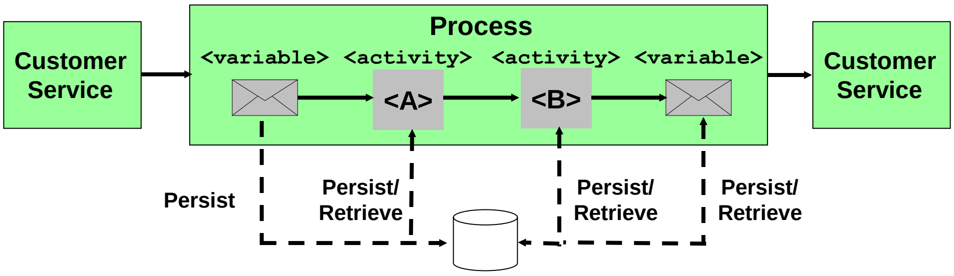
Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

4.6 Business Process Execution Languages (BPEL)

Process Variables and Data Flows:

Message sent / received to / from partner

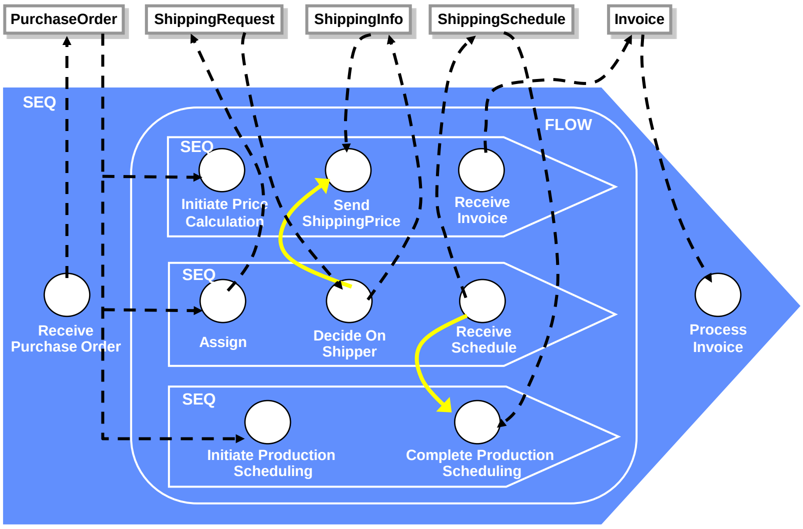
- Message types are defined with WSDL
- Have to be made persistent at runtime (→ long-running interactions)



Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

4.6 Business Process Execution Languages (BPEL)

Messages flows and process variables in our example:



Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

4.6 Business Process Execution Languages (BPEL)

Determining process variables in WS-BPEL:

Part of BPEL specification

```
<variables>
   <variable name="PO" messageType="lns:POMessage"/>
   <variable name="Invoice"
              messageType="lns:InvMessage"/>
   <variable name="POFault"
              messageType="lns:orderFaultType"/>
   <variable name="shippingRequest"
              messageType="lns:shippingRequestMessage"/>
   <variable name="shippingInfo"
              messageType="lns:shippingInfoMessage"/>
   <variable name="shippingSchedule"
              messageType="lns:scheduleMessage"/>
</variables>
```

Message types are defined based on WSDL

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

4.6 Business Process Execution Languages (BPEL)

Read / write access on process variables within service interactions (i.e. invoking Web Services)

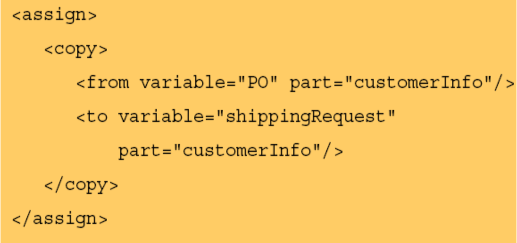
```
<invoke  partnerLink="shipping"
         portType="lns:shippingPT"
         operation="requestShipping"
         inputVariable="shippingRequest"
         outputVariable="shippingInfo">
   <source linkName="ship-to-invoice"/>
</invoke>
```

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

4.6 Business Process Execution Languages (BPEL)

Problem: „Mismatches“ between exchanged messages → Mapping!

- **By using BPEL statememts <assign> and <copy> data fields can be copied and transformed (from messages)**
- **<copy> supports Xpath expressions**



```
<assign>
   <copy>
      <from variable="PO" part="customerInfo"/>
      <to variable="shippingRequest"
          part="customerInfo"/>
   </copy>
</assign>
```

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

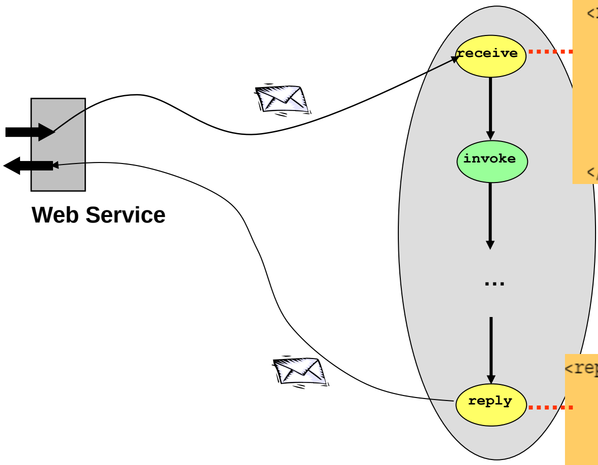
Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

4.6 Business Process Execution Languages (BPEL)

Elementary activities in WS-BPEL

- **receive**
  - Activity waits for incoming message and is then finished
  - BPEL processes often start with `receive`-activity → receiving of message leads to (implicitely) generating a new process instance
- **reply**
  - Reply message to precedent `receive`-activity
- **invoke**
  - Synchronous our asynchronous invocation of a WS operation offered by one of the partners
- **pick**
  - Defines a set of possibly incoming messages
  - Activity is finished as soon as one of the messages is coming in
  - Can (like `receive`) lead to generating new process instances

Example for using of `<receive>` / `<reply>`:



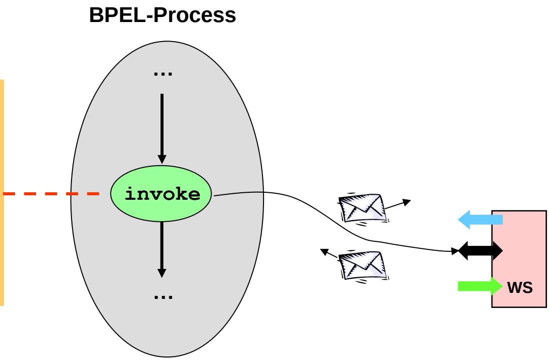
```
<receive partnerLink="purchasing"
         portType="lns:purchaseOrderPT"
         operation="sendPurchaseOrder"
         variable="PO">
</receive>
```

```
<reply partnerLink="purchasing"
       portType="lns:purchaseOrderPT"
       operation="sendPurchaseOrder"
       variable="Invoice"/>
```

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

Example for using `<invoke>`:

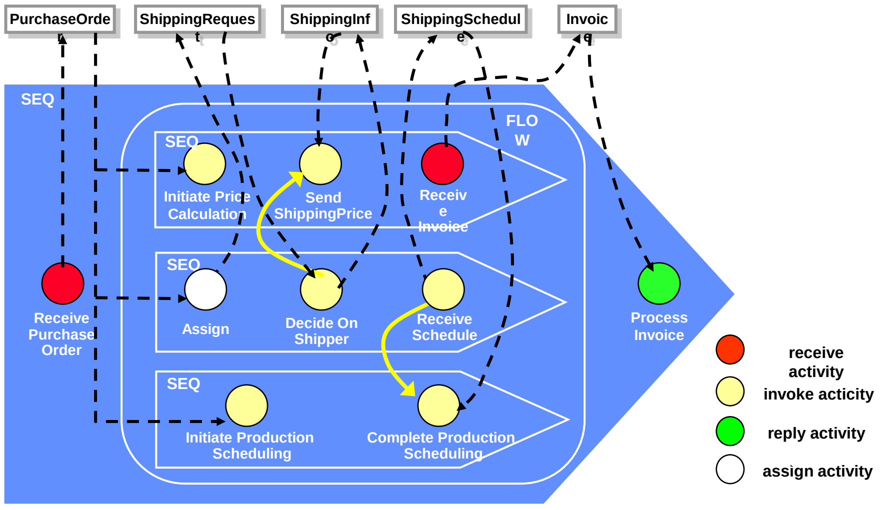
```
<invoke  partnerLink="shipping"
         portType="lns:shippingPT"
         operation="requestShipping"
         inputVariable="shippingRequest"
         outputVariable="shippingInfo">

</invoke>
```



Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

4.6 Business Process Execution Languages (BPEL)

Elementary activities in our example:



Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

4.6 Business Process Execution Languages (BPEL)

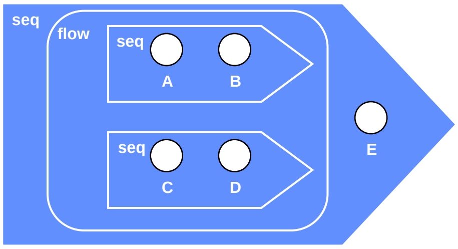
- **Structured activities…**
  - Follow block-based approach
  - Enable the composition of elementary and complex activities
  - Can be arbitrarily nested
- **Structured activities in WS-BEPL:**
  - `<sequence>` (→ Sequence)
  - `<switch>` (→ Alternative paths)
  - `<flow>` (→ parallel execution)
  - `<while>` (→ loops)
  - `<scope>` (→ spheres of control)
- **Further element: `links`**
  - Similar to *control connector* as used in Activity Nets
  - Can be used for describing control dependencies between parallel ordered activities
  - Can be connected with a transition condition

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

4.6 Business Process Execution Languages (BPEL)

Example: well-structured BPEL process

```
…
<sequence>
  <flow>
      <sequence>
          activity A
          activity B
      </sequence>
      <sequence>
          activity C
          activity D
      </sequence>
  </flow>
  activity E
</sequence>
```



graphical visualization

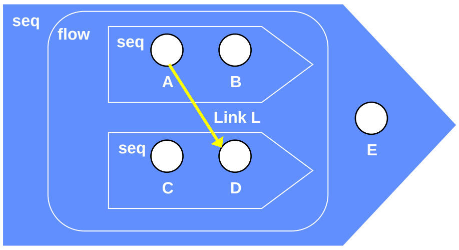
BPEL Process

→ Parallel structured modeled using `Flow`

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

4.6 Business Process Execution Languages (BPEL)

Example: BPEL Process by using links



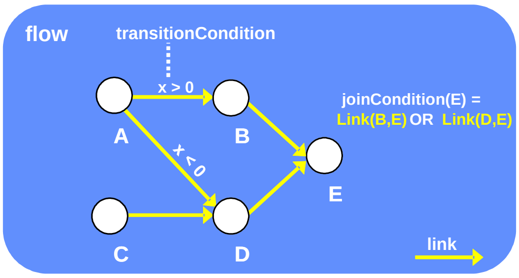
graphical visualization

```
...
<sequence>
  <flow>
    <links>
        <link name = "L"/>
    </links>
     <sequence>
         activity A
            <source linkName = "L"/>
         activity B
     </sequence>
     <sequence>
         activity C
         activity D
            <target linkName = "L"/>
     </sequence>
  </flow>
     activity E
</sequence>
...
```

BPEL Process

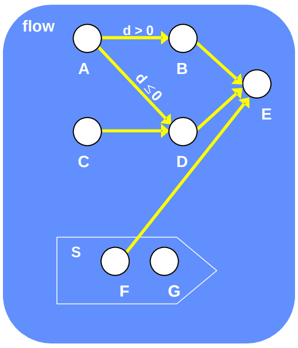
Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

Example: Link-based BPEL Process



Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

example: mixed usage of structured activities and links

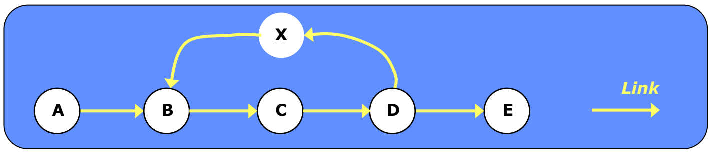


```
…
<flow>
    <links>
        <link name = "L_AB"/>
        <link name = "L_AD"/>
        <link name = "L_FE"/>
        …
    </links>
    activity A
        <source linkName = "L_AB" transitionCondition = "d < 0"/>
        <source linkName = "L_AD" transitionCondition = "d ≤ 0"/>
    activity B
        <target linkName = "L_AB"/>
        <source linkName = "L_BE"/>
    …
    activity E
        joinCondition = "(L_BE OR L_DE) AND L_FE"
        <target linkName = "L_BE"/>
        <target linkName = "L_DE"/>
        <target linkName = "L_EB"/>
    <sequence name = "S"
        activity F
            <source linkName = "L_FE"/>
        activity G
    </sequence>
</flow>
…
```

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

Example: rules for using links

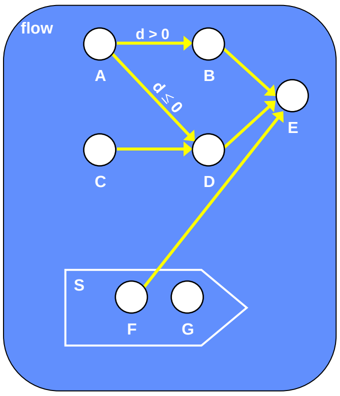
→ Links must not lead to cycles in execution graph

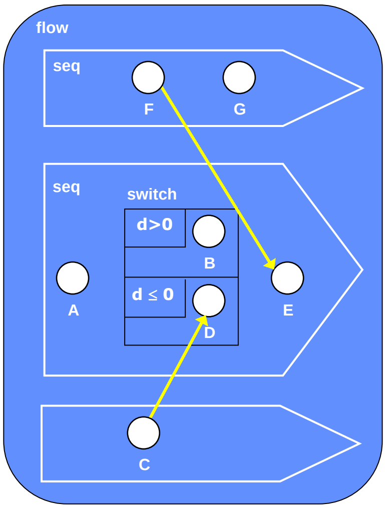


Cycle → **Deadlock**

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

WS-BPEL offers many possibilities to describe the same behavior

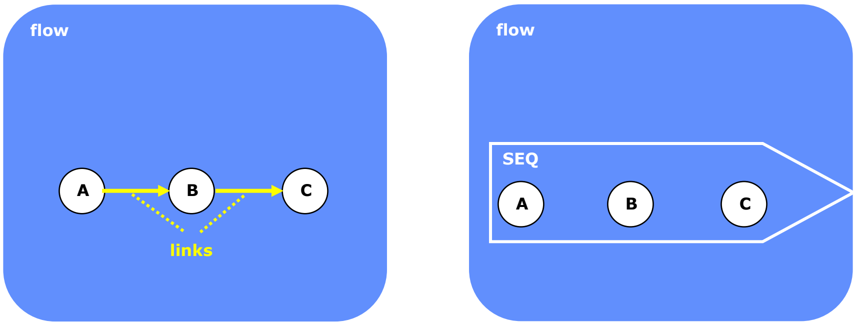




Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

4.6 Business Process Execution Languages (BPEL)

However: many undesired side-effects when changing the execution graph due to the high expressiveness and parametrization of the language

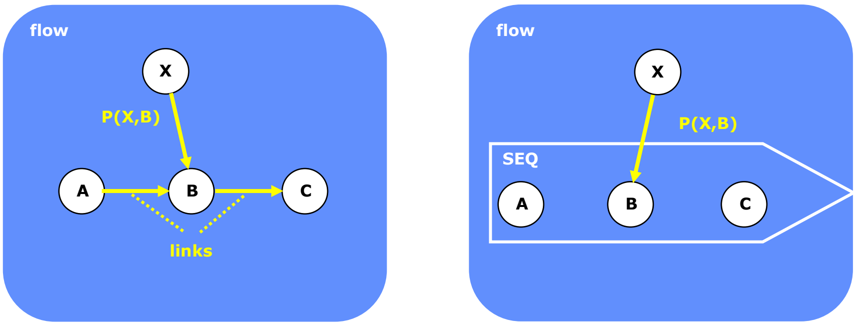


→ equivalent execution behavior

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

4.6 Business Process Execution Languages (BPEL)

However: many undesired side-effects when changing the execution graph due to the high expressiveness and parametrization of the language

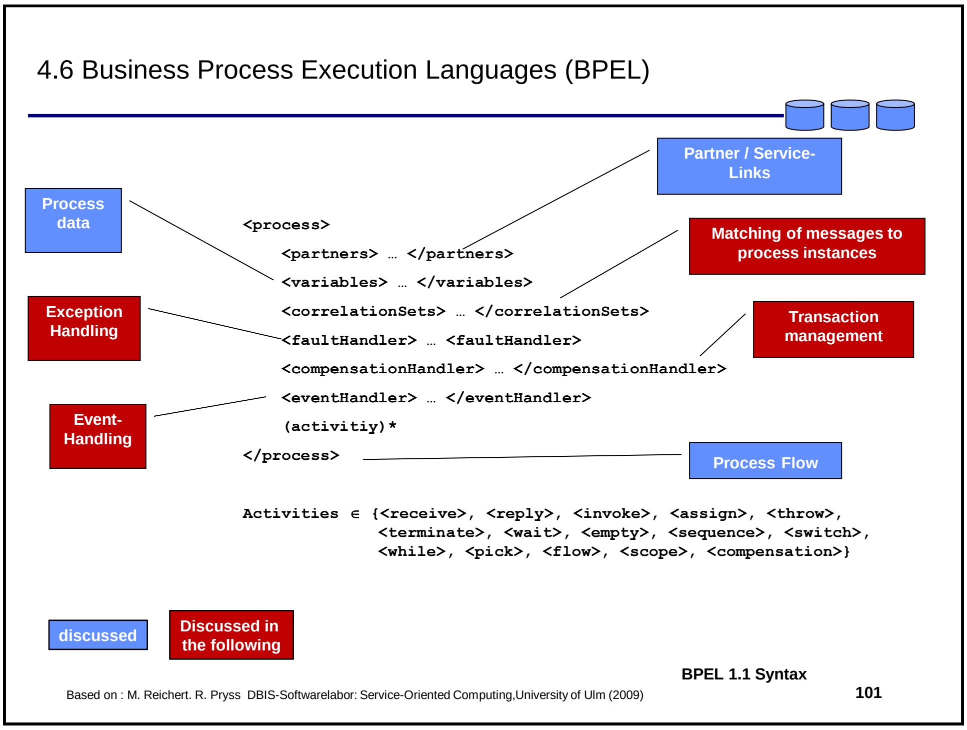


joinCond(B) ≡ P(X,B) ∨ *True* ≡ *True* (Default)

joinCond(B) ≡ P(X,B) (Default)

→ different execution behavior

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)



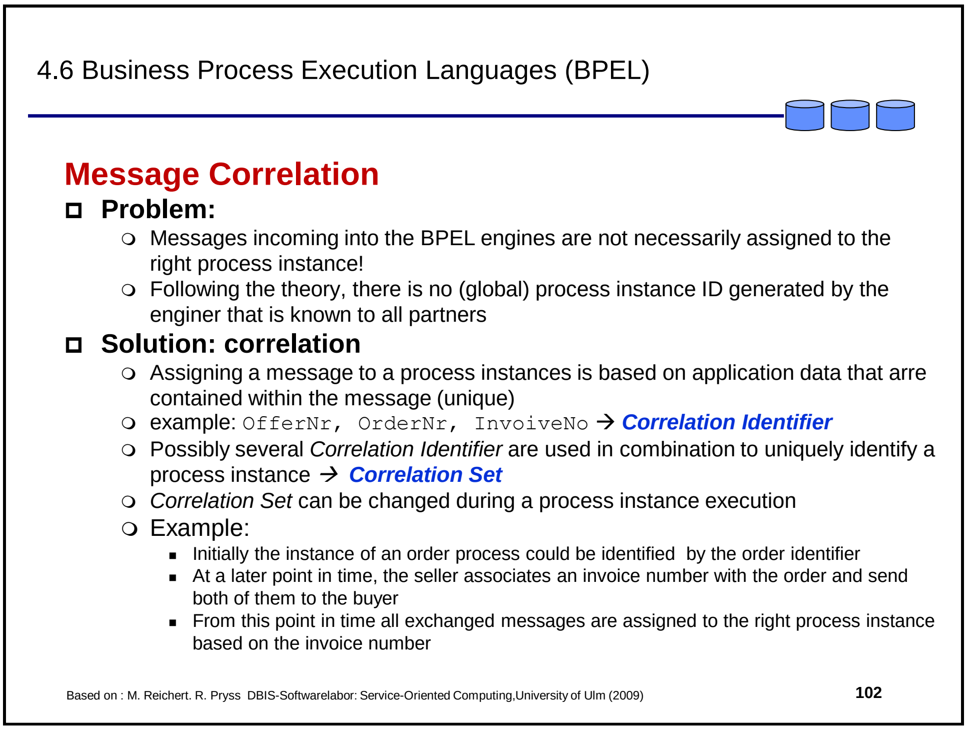
4.6 Business Process Execution Languages (BPEL)

```
<process>
    <partners> … </partners>
    <variables> … </variables>
    <correlationSets> … </correlationSets>
    <faultHandler> … <faultHandler>
    <compensationHandler> … </compensationHandler>
    <eventHandler> … </eventHandler>
    (activitiy)*
</process>

Activities ∈ {<receive>, <reply>, <invoke>, <assign>, <throw>,
              <terminate>, <wait>, <empty>, <sequence>, <switch>,
              <while>, <pick>, <flow>, <scope>, <compensation>}
```

BPEL 1.1 Syntax

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

101



4.6 Business Process Execution Languages (BPEL)

Message Correlation

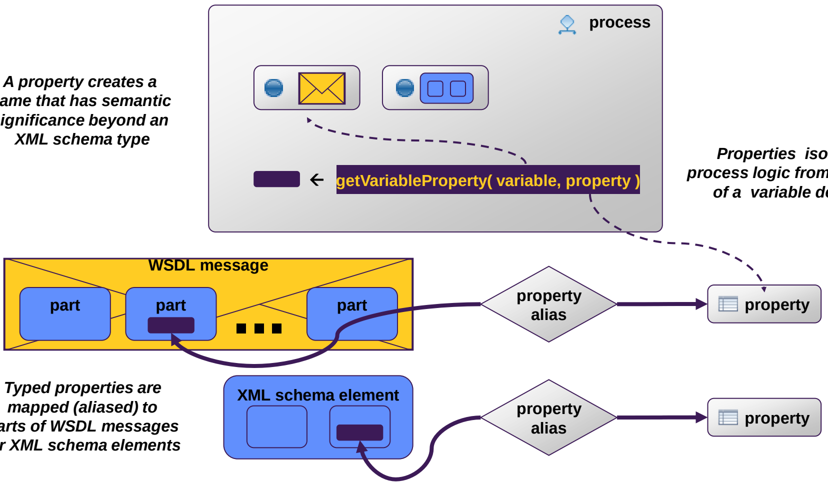
- **Problem:**
  - Messages incoming into the BPEL engines are not necessarily assigned to the right process instance!
  - Following the theory, there is no (global) process instance ID generated by the enginer that is known to all partners
- **Solution: correlation**
  - Assigning a message to a process instances is based on application data that arre contained within the message (unique)
  - example: `OfferNr, OrderNr, InvoiceNo` → ***Correlation Identifier***
  - Possibly several *Correlation Identifier* are used in combination to uniquely identify a process instance → ***Correlation Set***
  - *Correlation Set* can be changed during a process instance execution
  - Example:
    - Initially the instance of an order process could be identified by the order identifier
    - At a later point in time, the seller associates an invoice number with the order and send both of them to the buyer
    - From this point in time all exchanged messages are assigned to the right process instance based on the invoice number

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

102

4.6 Business Process Execution Languages (BPEL)

A property creates a name that has semantic significance beyond an XML schema type



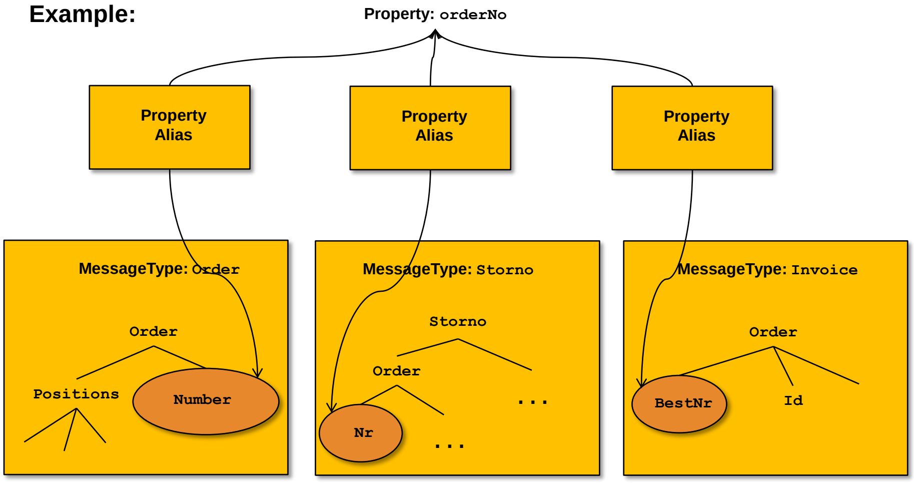
Properties isolate the process logic from the details of a variable definition

Typed properties are mapped (aliased) to parts of WSDL messages or XML schema elements

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

4.6 Business Process Execution Languages (BPEL)

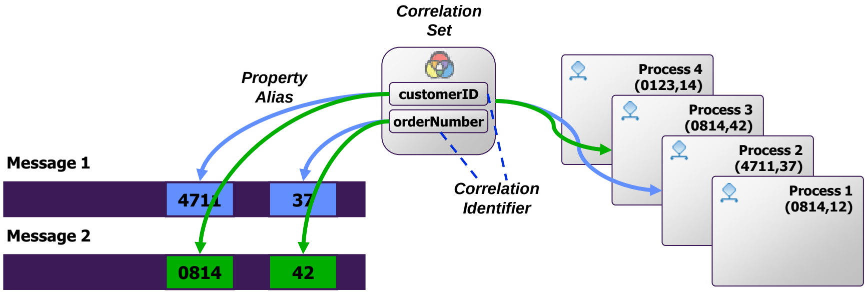
Example:



Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

4.6 Business Process Execution Languages (BPEL)

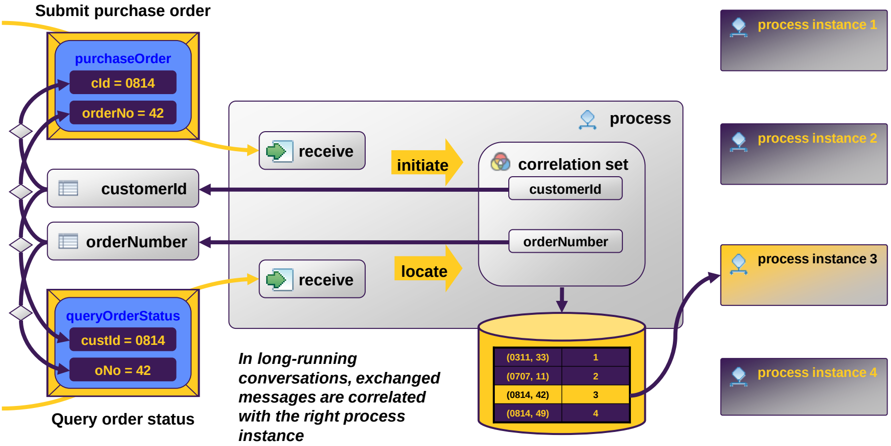
Functioning



Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

4.6 Business Process Execution Languages (BPEL)

Example:

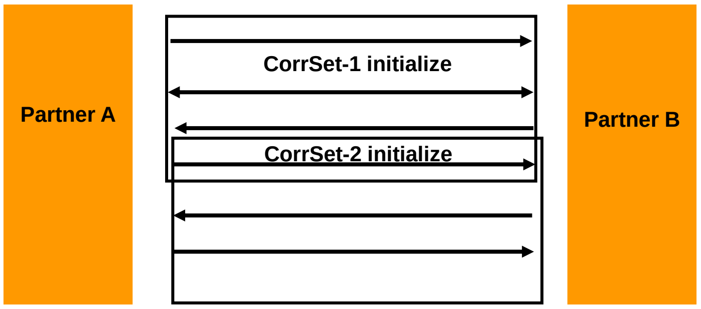


In long-running conversations, exchanged messages are correlated with the right process instance

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

4.6 Business Process Execution Languages (BPEL)

Multiple Correlation Sets



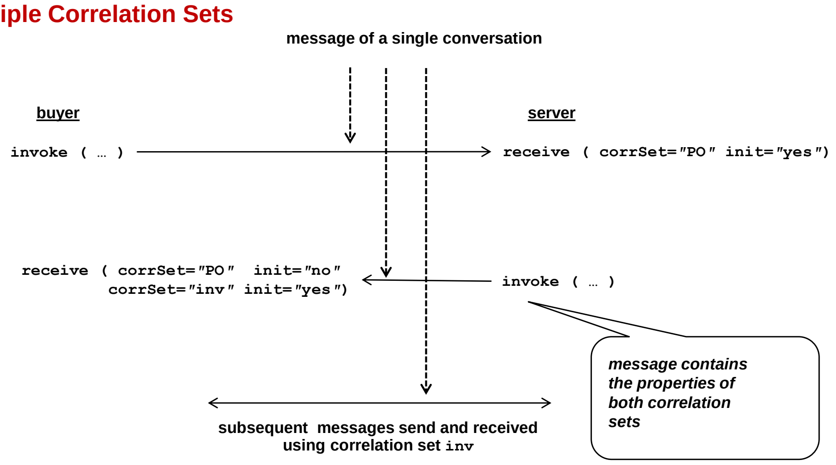
→ ***Correlation Set* can change at runtime!**

→ **Change must be exchanged between partners!**

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

4.6 Business Process Execution Languages (BPEL)

Multiple Correlation Sets



message of a single conversation

buyer

server

`invoke ( … )`

`receive ( corrSet=″PO″ init=″yes″)`

`receive ( corrSet=″PO″ init=″no″`
`corrSet=″inv″ init=″yes″)`

`invoke ( … )`

message contains the properties of both correlation sets

subsequent messages send and received using correlation set `inv`

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

4.6 Business Process Execution Languages (BPEL)

Multiple Correlation Sets

A Correlation Set is logically connected with a message. In principle, for Input-/Output-Message of a synchronous service call (invoke) different Correlation Sets can be used (→ example)

```
<invoke partnerLink=“Buyer” interface=“SP:BuyerIf”
        operation=“synchPurchaseResp”
        inputVariable=“PO” outputVariable=“invoice”>
        <correlations>
            <correlation set=“PurchaseOrder” initiate=“yes” pattern=“out”/>
                                   (initiated in outbound msg)
            <correlation set=“InvoiceResp”   initiate=“yes”  pattern=“in”/>
                                   (initiated in inbound msg,
                                    which contains both sets)
        </correlations>
</invoke>
```

4.6 Business Process Execution Languages (BPEL)

Scopes, Compensation, Event Handling

- **WS-BPEL offers a number of further useful implementation concepts, e.g., for exception and event handling**
- **In the following, we discuss the following BPEL elements:**
  - **Scopes**
  - **Compensation**
  - **Event Handling**

4.6 Business Process Execution Languages (BPEL)

Scopes

- Offer common context for a sub set of activities
- A Scope either contains …
  - Fault handler
  - Event handler
  - Compensation handler
  - Correlation sets
- By using scopes, exception and error handling can be defined specifically for certain parts of the process
- Can serialize parallel accesses on process variables

```
<scope
variableAccessSerializable="yes|no"
 ...>

<variables>
</variables>

<correlationSets>? ...
        </correlationSets>

<faultHandlers>
</faultHandlers>

<compensationHandler>? ...
</compensationHandler>

<eventHandlers>
</eventHandlers>
(activities)*

</scope>
```

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

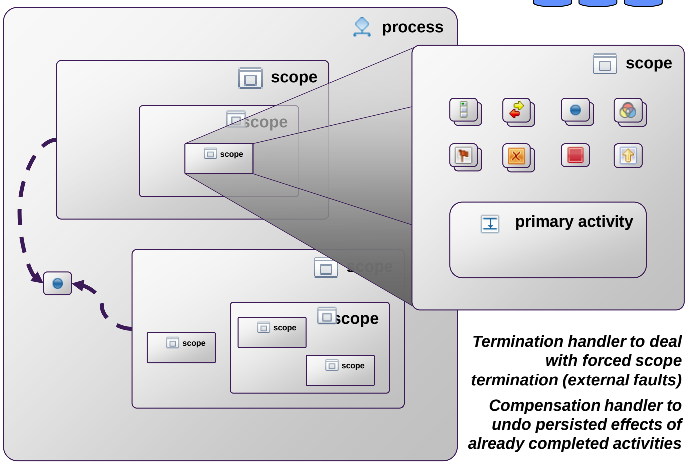
4.6 Business Process Execution Languages (BPEL)

Scopes provide a context which influences the execution behavior of its enclosed activities

Local declarations: partner links, message exchanges, variables, correlation sets

Local handlers: event handlers, fault handlers, a termination handler, and a compensation handler

Isolated scopes provide control of concurrent access to shared resources



Termination handler to deal with forced scope termination (external faults)

Compensation handler to undo persisted effects of already completed activities

Scopes in WSBPEL

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

4.6 Business Process Execution Languages (BPEL)

Compensation in WS-BPEL

- During execution of a single process activity an error might occur, for example, if a web service that is invoked by this activity cannot be reached or is terminated with an error message
- When such error situations occur, the process cannot be continued as planned, but an exception handling becomes necessary:
  - Controlled termination of process instance
  - Backward / Forward Recovery
- In the following we discuss possibilites that WS-BPEL offers in this context!

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

4.6 Business Process Execution Languages (BPEL)

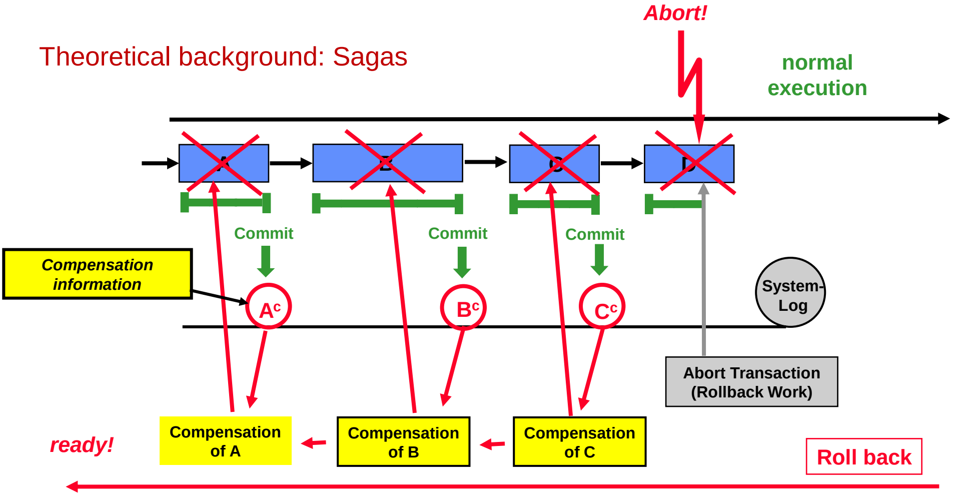
Theoretical background: Sagas

- Principle of ACID transactions („All-or-nothing“) in connection with long-running processes is usually not applicable
- Atomicity or isolation is too restrictive:
  - Long-running transactions → locks have to be hold very long
  - Not all applications support Two-Phase-Commit
    → „intermediate results“ are visible to the outside
- Basic idea: → Sagas [Garcia-Molina, ACM-Sigmod Conf. 1987]
  - Linking part-transactions (with ACID properties)
  - Classical Rollback for aborted (i.e., incompletely executed) part transactions
  - Semantic compensation of already finished transactions

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

4.6 Business Process Execution Languages (BPEL)

Theoretical background: Sagas

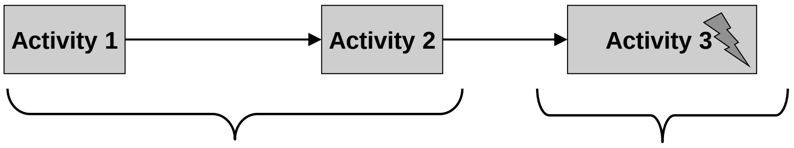


Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

4.6 Business Process Execution Languages (BPEL)

Compensation Handler and Fault Handler

- **BPEL offers extended transactional support similar to Sagas**
- **Important parts: Compensation Handler + Fault Handler**
- **Example: Sequence of 3 activities and error when executing the third activity**



Potentially taking back precedent (and actually successful) operations

Treating current error

CompensationHandler

FaultHandler

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

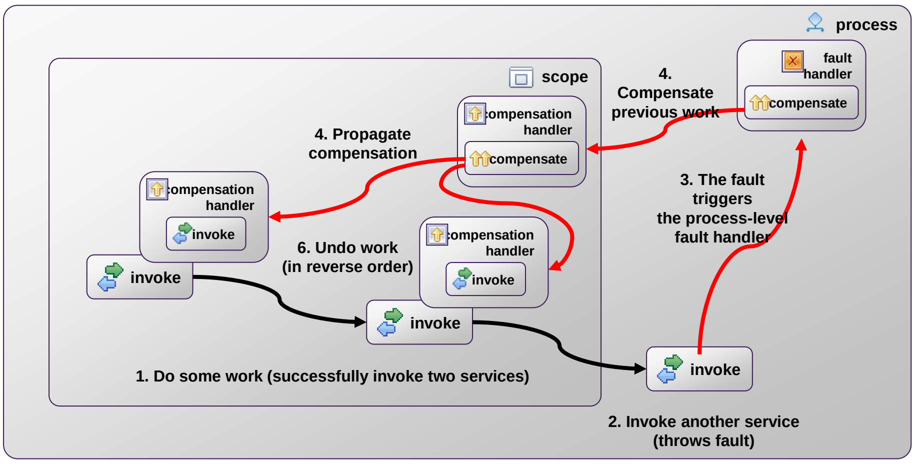
4.6 Business Process Execution Languages (BPEL)

Scopes and Compensation

- Compensation behavior of a process can be defined based on *scopes* (i.e., spheres)
- In case of an error during scope execution, the running activities of the scope are aborted and already finished activities are compensated (default behavior)
- Depending on the scope at time of the errors, a more specific compensation can take place
- For this purpose, a separate compensation handler can be assigned to each scope
- High expressiveness when nesting scopes!

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

4.6 Business Process Execution Languages (BPEL)



4. Compensate previous work

4. Propagate compensation

3. The fault triggers the process-level fault handler

6. Undo work (in reverse order)

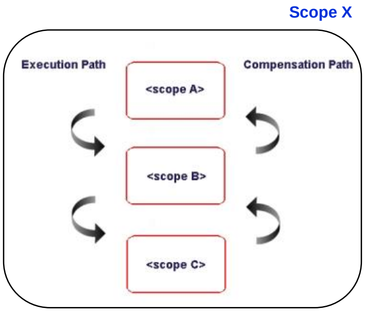
1. Do some work (successfully invoke two services)

2. Invoke another service (throws fault)

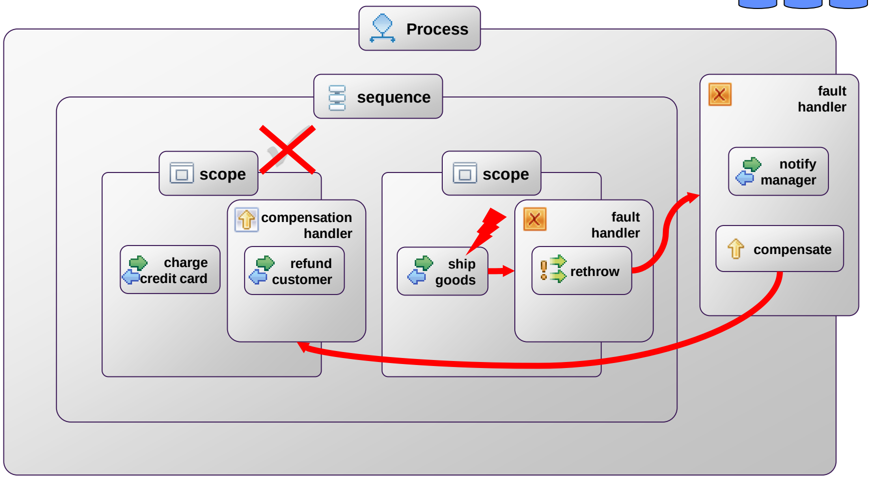
Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

Nested Compensation

Compensation semantics can be flexibly defined when nesting scopes



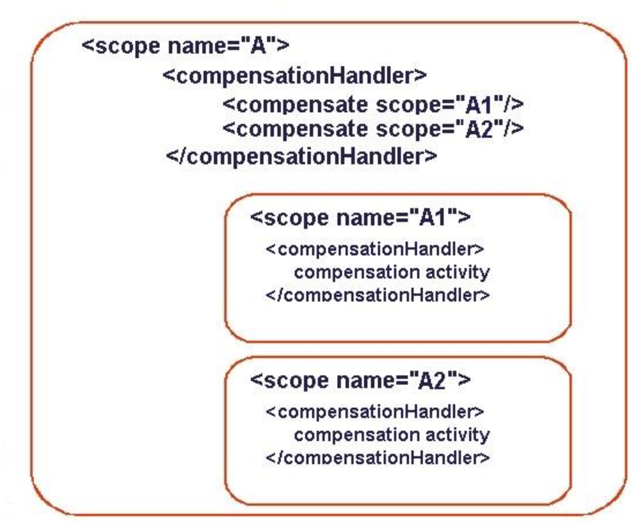
Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)



Nested Compensation

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

4.6 Business Process Execution Languages (BPEL)



```
<scope name="A">
        <compensationHandler>
              <compensate scope="A1"/>
              <compensate scope="A2"/>
        </compensationHandler>

              <scope name="A1">
                <compensationHandler>
                   compensation activity
                </compensationHandler>

              <scope name="A2">
                <compensationHandler>
                   compensation activity
                </compensationHandler>
```

Nested Compensation

Quelle: activebpel.org

4.6 Business Process Execution Languages (BPEL)

Faults and Fault Handler

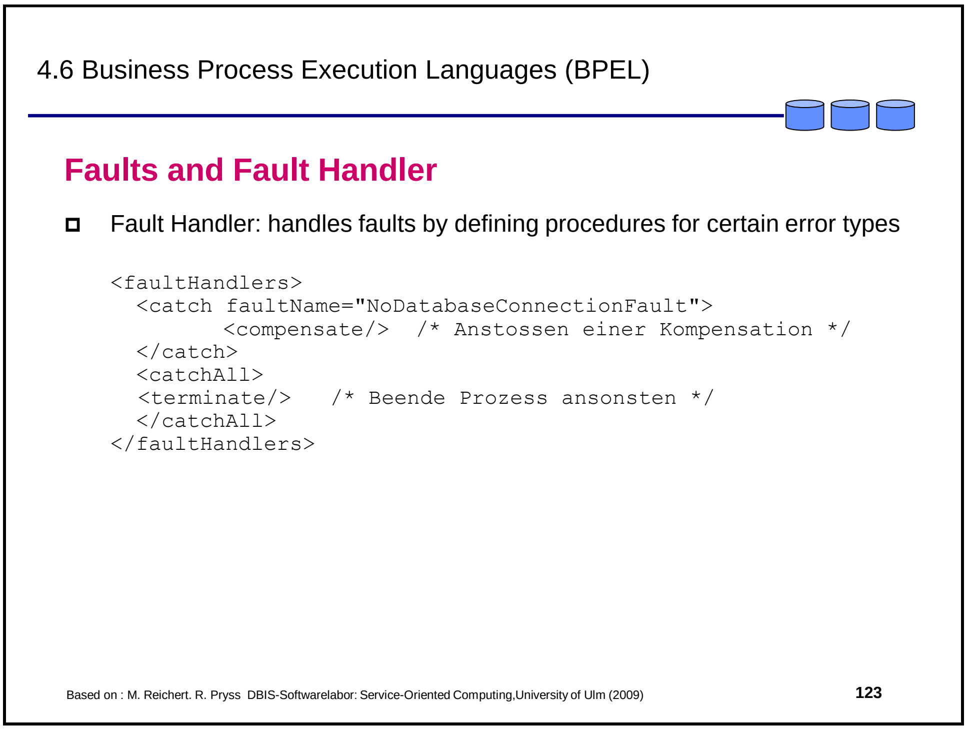
- Fault messages of a Web Service Operation can be explicitely defined in WSDL
- Errors can be caused during process instance runtime by:

1. Failure of a synchronous WS call, i.e., an `invoke` activity of a process instance does not yield a standard output message, but an error message
2. A partner process or service sends a `Reply` with error message as reaction on a received message (`receive` from process instance perspective).

```
<reply partnerLink="selling" portType="SellerPT"
    operation="buy" faultName=“OutOfStockFault"/>
```

3. Internally with `throw`

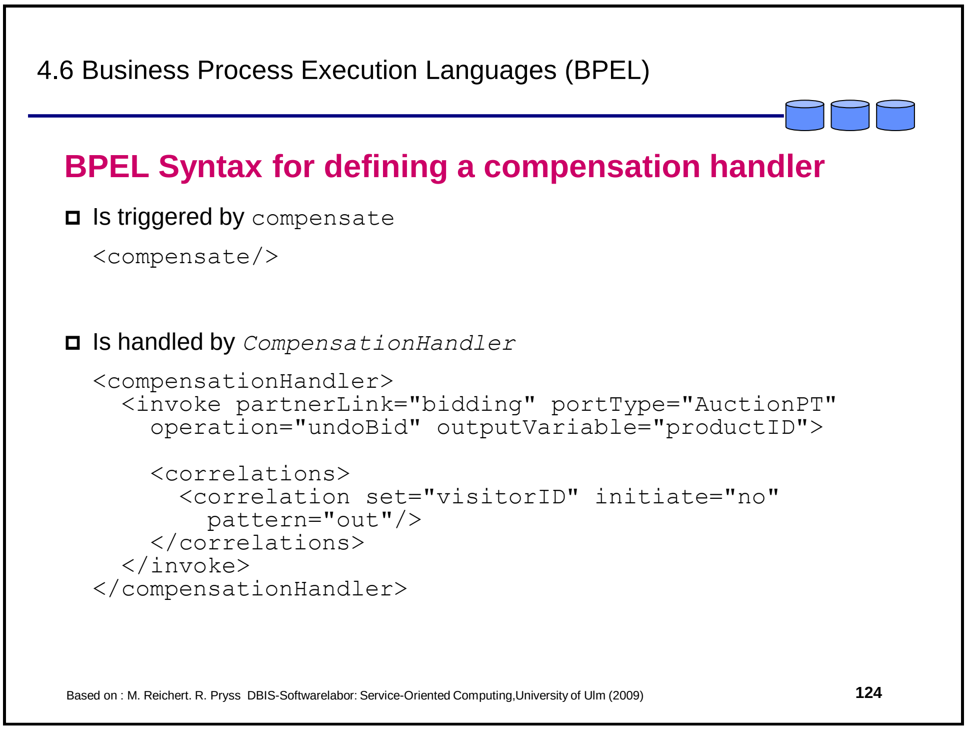
```
<throw faultName="NoDatabaseConnectionFault"/>
```



Faults and Fault Handler

- Fault Handler: handles faults by defining procedures for certain error types

```
<faultHandlers>
  <catch faultName="NoDatabaseConnectionFault">
         <compensate/>  /* Anstossen einer Kompensation */
  </catch>
  <catchAll>
  <terminate/>   /* Beende Prozess ansonsten */
  </catchAll>
</faultHandlers>
```



BPEL Syntax for defining a compensation handler

- Is triggered by `compensate`

```
<compensate/>
```

- Is handled by *`CompensationHandler`*

```
<compensationHandler>
  <invoke partnerLink="bidding" portType="AuctionPT"
    operation="undoBid" outputVariable="productID">

    <correlations>
      <correlation set="visitorID" initiate="no"
        pattern="out"/>
    </correlations>
  </invoke>
</compensationHandler>
```

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

4.6 Business Process Execution Languages (BPEL)

Event Handling

- *eventHandler*: can handle incoming calls and messages in parallel to normal process
- `onMessage`

```
<onMessage partnerLink=“buyer”
                    portType=“car”
                    operation=”cancel”
                    variable=“cancelDetails”>
               <terminate/>
</onMessage>
```

- Event that is invoked by external partner

```
<onAlarm (for=“duration-expr” | until=“deadline-expr”)>*
     <! –- activity -- >
</onAlarm>
```

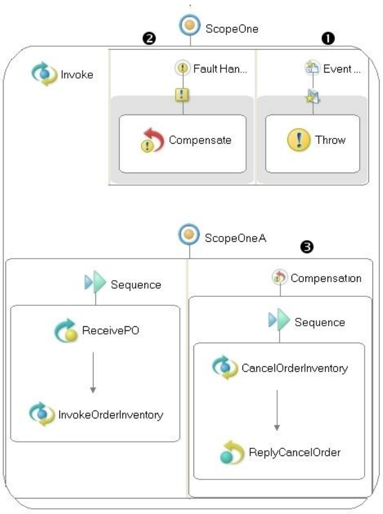
4.6 Business Process Execution Languages (BPEL)

Event Handling

Example: Events in ActiveBPEL

With an EventHandler the termination of a process instance and the compensation activities required in this context can be triggered!

1. **When the event handler receives an order cancellation message, it throws a fault to the fault handler.**
2. **The fault handler executes a compensate activity for the previously completed scope to which it is linked.**
3. **The compensation handler for the completed scope rolls back the work of the InvokeOrderInventory service.**



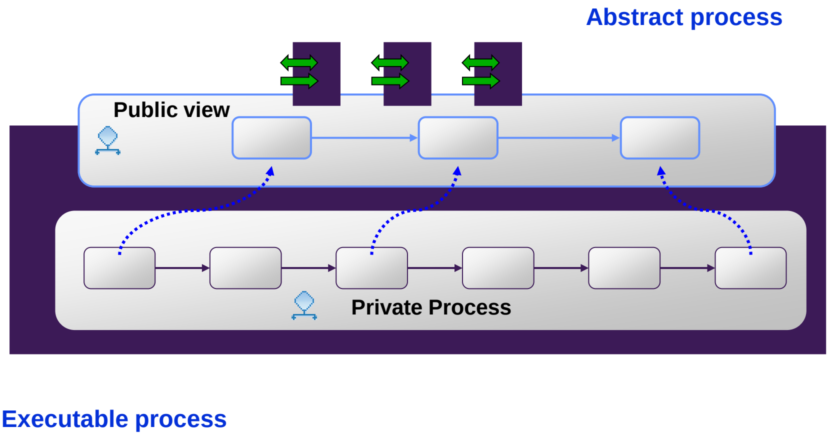
Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

Quelle: activebpel.org

Abstract Processes In WS-BPEL

- BPEL-Syntax does not not only allow for specification of executable processes but also for definition of abstract processes
- In the following:
  - Hiding process details
  - Basic principle of abstract processes
  - Typical applications in detail

Hiding process details



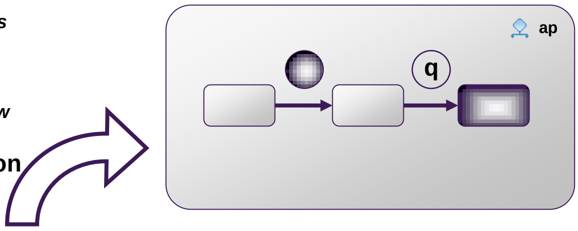
4.6 Business Process Execution Languages (BPEL)

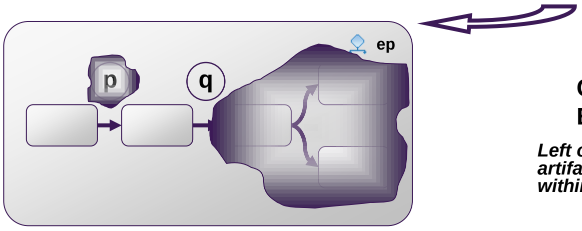
Basic principle of abstract processes:

An abstract process describes "behavior", but is not executable itself!

Abstract process builds a view on the concrete process!

Abstraction





Concretization / Executable process

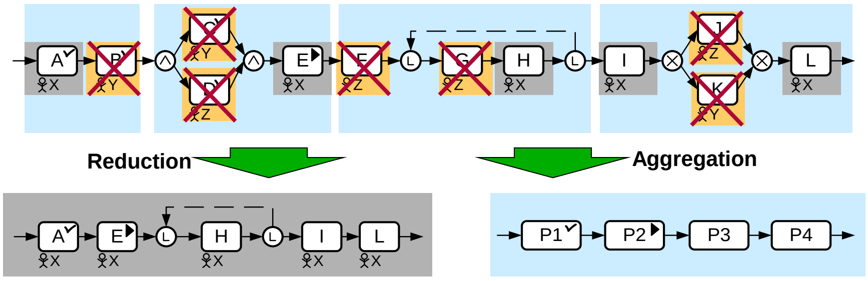
Left out information represents artifacts that are later defined within a top-down-design

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

4.6 Business Process Execution Languages (BPEL)

Basic Principle of abstract processes:

Background: Techniques for building abstract processes (i.e., views) on executable process models



Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

Typical use cases:

- □ **Use Case 1: View on internal process**
  - ❍ Only a projection of an internal (executable) process is made visible to the outside
    …to protect process model as corporate asset
    …to hide non-optimal parts of a process model
- □ **Use Case 2: Template as "best practice"**
  - ❍ Specification of common activities, major data structures, and main control flow
  - ❍ Must be refined into an executable processes on a case-by-case basis
- □ **Use Case 3: Constraints on Message Exchanges / Service Operations**
  - ❍ Specification about the order in which messages are consumed or produced
    → Business functionality is implemented as a (set of) port types, and operations must be used in a certain order to achieve intended business goal

View on Internal Processes

- □ **Use Case 1**: **An abstract process as view on an internal one**
  An abstract process is derived from an executable process by abstracting away parts that are not part of the behavior one wishes to expose
- □ Examples:
  - ❍ Show a particular business partner the interactions that the partner must follow
    → Interactions with all other partners are dropped
  - ❍ Use an abstract process to represent common behavior in a set of executables, and drop any non-repeated behavior

Template as “Best Practice”

- **Use Case 2**: **Template as Best Practice**
  An abstract process is basis to create one or more executables, or more detailed abstract processes
- Example:
  - One needs to create an implementation of an abstract process provided as a behavioral prescription for complying with a known, domain-specific business function
  - One wants to implement “best practices” while maintaining some company specifics
    → The abstract process may have been purchased from a consulting firm, as a model of an optimized approach to a problem

Constraints on Service Operations

- **Use Case 3**: **Constraints on the orders of service operations**
- Typically, the operations of a service may not be used in arbitrary order
  - Example: It doesn’t make sense to use the `Cancel` operation of an `Order` service before using its `Buy` operation.
- To describe such ordering constraints an abstract process can be used that only refers to operations of a single port type
- The port type of a service may be associated with a process which describes the order in which the operations of the port type can be used

4.6 Business Process Execution Languages (BPEL)

Innovations in WS-BPEL 2.0 when compared to BPEL4WS 1.1

- **Data access and data manipulation**
  - XML schema complex-typed variables
  - Simplified XPath expressions
  - Simplified WSDL message access
  - Elaborated **<copy>** operation behavior in **<assign>**
  - Attribute *keepSrcElementName* in **<copy>**
  - Attribute *ignoreMissingFromData* in **<copy>**
  - Extension operations in **<assign>**
  - XSLT 1.0 function for use within XPath expressions
  - XML data validation
  - New **<validate>** activity
  - Inline variable initialization within variable declarations

4.6 Business Process Execution Languages (BPEL)

Do a blocking wait for a matching message to arrive / send a message in reply

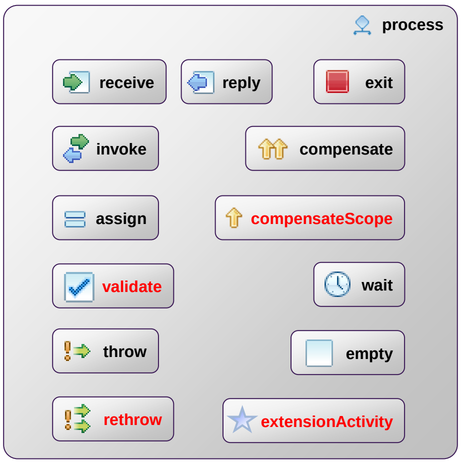
Invoke a one-way or request-response operation

Update the values of variables or partner links with new data

Validate XML data stored in variables

Generate a fault from inside the business process

Forward a fault from inside a fault handler



Immediately terminate execution of a business process instance

Invoke compensation on all completed child scopes in default order

Invoke compensation on one completed child scope

Wait for a given time period or until a certain time has passed

No-op instruction for a business process

Wrapper for language extensions

New in WS-BPEL 2.0

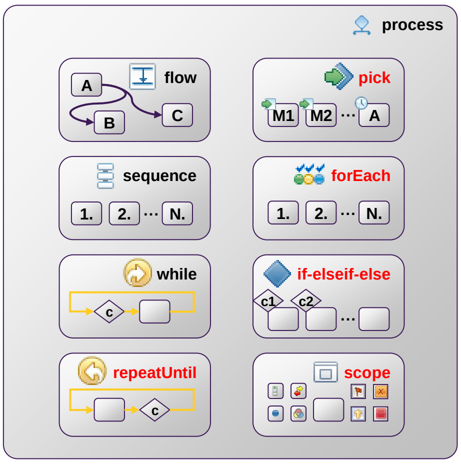
4.6 Business Process Execution Languages (BPEL)

Contained activities are executed in parallel, partially ordered through control links

Contained activities are performed sequentially in lexical order

Contained activity is repeated while a predicate holds

Contained activity is repeated until a predicate holds



Block and wait for a suitable message to arrive (or time out)

Contained activity is performed sequentially or in parallel, controlled by a specified counter variable

Select exactly one branch of activity from a set of choices

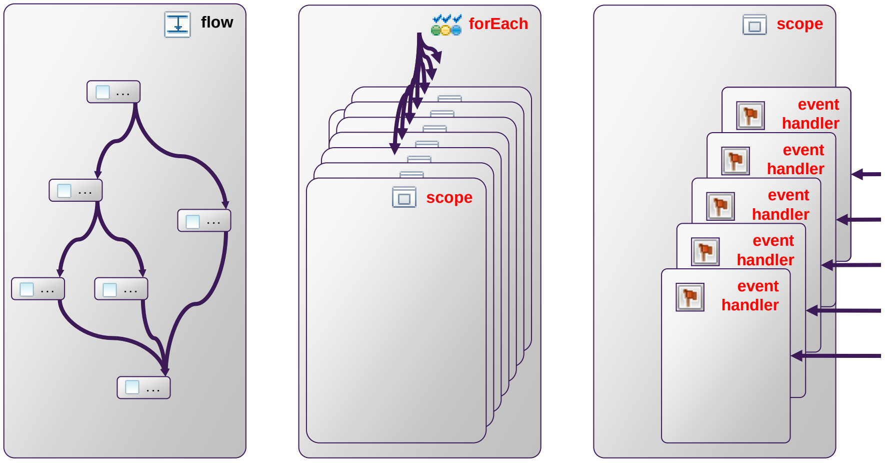
Associate contained activity with its own local variables, partner links, etc., and handlers

New in WS-BPEL

Structured Activities in BPEL4WS 1.1 and WS-BPEL 2.0

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

4.6 Business Process Execution Languages (BPEL)



New in WS-BPEL

Parallel execution BPEL4WS 1.1 and WS-BPEL 2.0

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

Outline

4.1 Motivation

4.2 Business Process Modeling Notation (BPMN)

4.3 Modeling Process Choreographies

4.4 Process Choreography Design

4.5 Process Choreography Implementation

4.6 Business Process Execution Language (BPEL)

4.7 Summary and Current Research

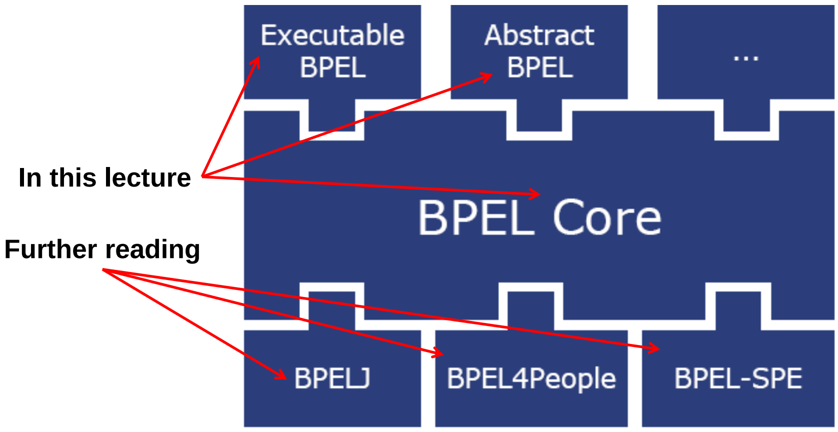
References

4.7 Summary and Outlook

- WS-BPEL accepted as de facto standard for process-oriented composition and orchestration of services
- Support by many vendors
- Many possibilities for configuration and execution behavior (e.g., for exception handling)
- However: expressiveness also causes high complexity and errors
- Partly unclear or imprecise (formal) semantics
- Vendor-independence more a dream than reality
  - Vendors often implement subsets of BPEL on the one side
  - And do a lot of extension on the other side!
- Currently: support of transforming BPMN into BPEL

Based on : M. Reichert. R. Pryss DBIS-Softwarelabor: Service-Oriented Computing,University of Ulm (2009)

4.7 Summary and Outlook



References

Internet:

- WS-BPEL 2.0 – Approved Standard (04/11/2007):
  http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html
- OASIS Technical Committee
  http://www.oasis-open.org
- BPEL4WS on IBM:
  http://www.redbooks.ibm.com/abstracts/sg246381.html?Open

Books:

- Buch Michael Papazoglou: *Web Services: Principles and Technology* (Pearson (Prentice Hall) , **ISBN-10:** 0321144446)
- Buch Fabio Casati: *Web Services. Concepts, Architectures and Applications* (Springer-Verlag, **ISBN-10:** 3440440089)

Commercial Products

- Active Endpoints ActiveBPEL
  http://www.activevos.com/bpel.php
- Apache Orchestration Director Engine (Ode)
  http://ode.apache.org/ws-bpel-20.html
- IBM WebSphere Process Server
  http://www.ibm.com/developerworks/websphere/library/techarticles/0608_kagan/0608_kagan.html
- Microsoft BizTalk Integration Platform
  http://www.microsoft.com/germany/biztalk/interop/prozessmodellierung.mspx
- OpenLink Virtuoso Universal Server
  http://www.openlinksw.com/virtuoso/overview/VOSRelease/index.htm
- Oracle BPEL Process Manager
  http://www.oracle.com/technetwork/middleware/bpel/overview/index.html
- Parasoft BPEL Maestro
  http://parasoft-bpel-maestro.software.informer.com/5.0/
- SAP NetWeaver
  http://scn.sap.com/community/netweaver

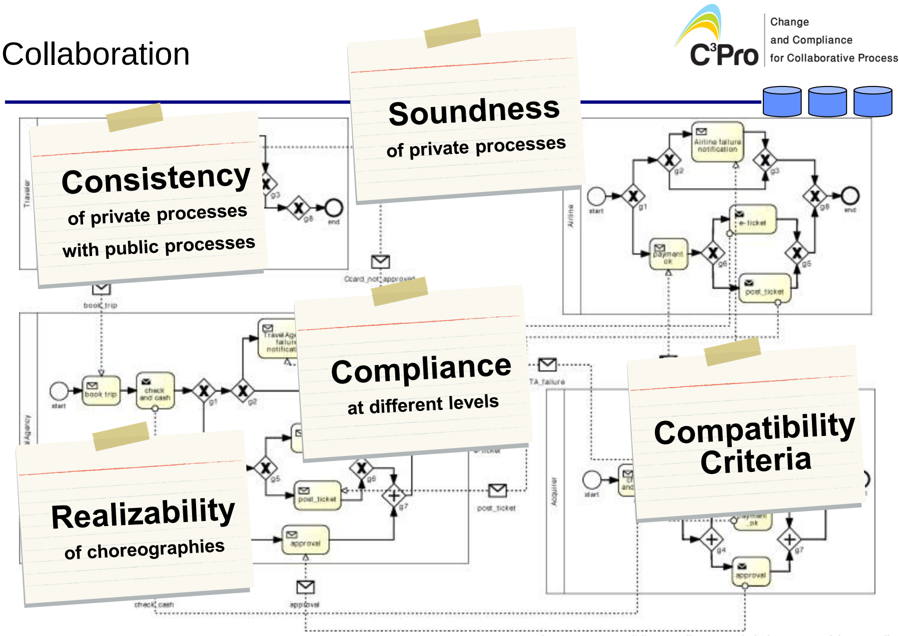
Current Research

C^{3}Pro: Change and Compliance for Collaborative Processes

- http://www.wst.univie.ac.at/communities/c3pro/
- Funded by FWF (lead agency) and DFG

C^{3}Pro
Change
and Compliance
for Collaborative Processes

Collaboration



Consistency of private processes with public processes

Soundness of private processes

Compliance at different levels

Compatibility Criteria

Realizability of choreographies

Book trip process: public models.

© Elsever 2015, W. Fdhila, C. Indiono, S. Rinderle-Ma, M. Reichert: Dealing with change in process choreographies: Design and implementation of propagation algorithms. Information Systems, Volume 49, 2015, 1 - 24

Collaborative Processes

C^{3}Pro project:

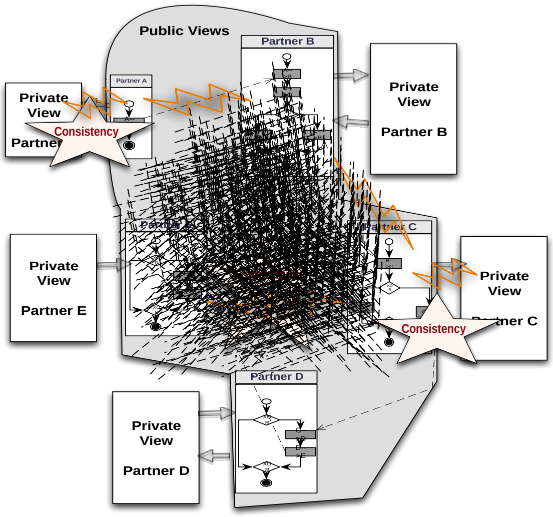
funded by FWF (lead agency) and DFG under D-A-CH principles

Cooperation University of Vienna (lead) and University of Ulm

Objectives

- Change and change propagation in collaborative processes
- Compliance in collaborative processes
- Interplay between different aspects

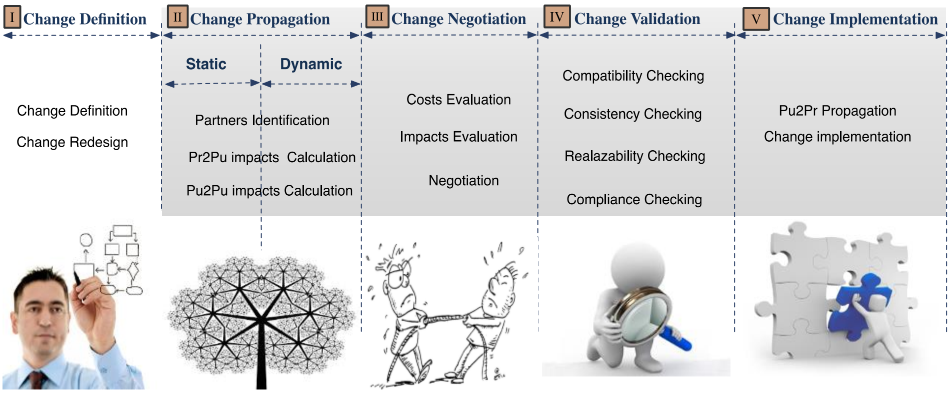
2 Change in Collaborations



Challenges:

- ❑ **Consistency**
- ❑ **Behavioral Compatibility**
  - **Waiting for a message which would never arrive**
  - **Sending message which would not be consumed**
- ❑ **Structural compatibility**
- ❑ **Transitivity effects**
- ❑ **Negotiation**
- ❑ **Compliance!**

Change Propagation: Overall Picture



I Change Definition

- Change Definition
- Change Redesign

II Change Propagation

- Static
- Dynamic
- Partners Identification
- Pr2Pu impacts Calculation
- Pu2Pu impacts Calculation

III Change Negotiation

- Costs Evaluation
- Impacts Evaluation
- Negotiation

IV Change Validation

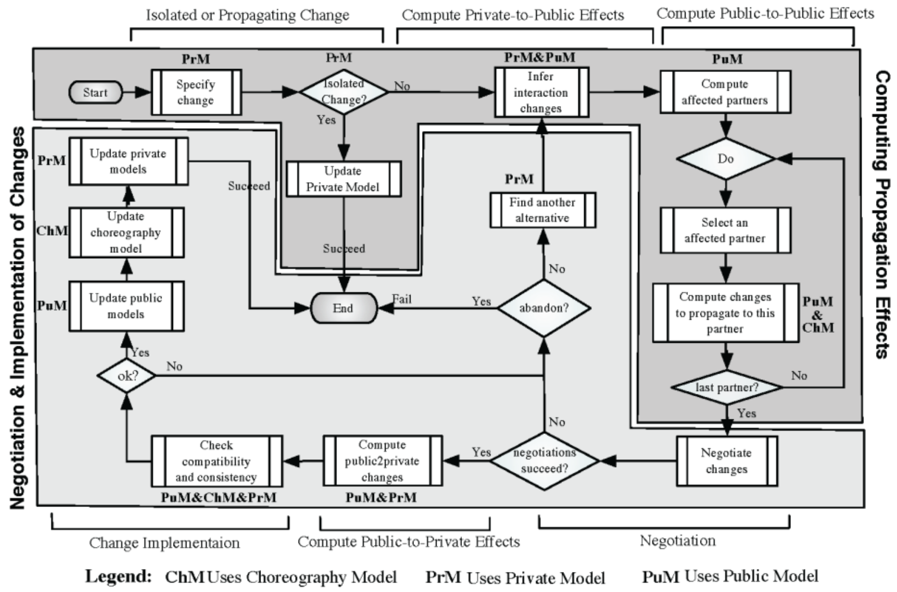
- Compatibility Checking
- Consistency Checking
- Realazability Checking
- Compliance Checking

V Change Implementation

- Pu2Pr Propagation
- Change implementation

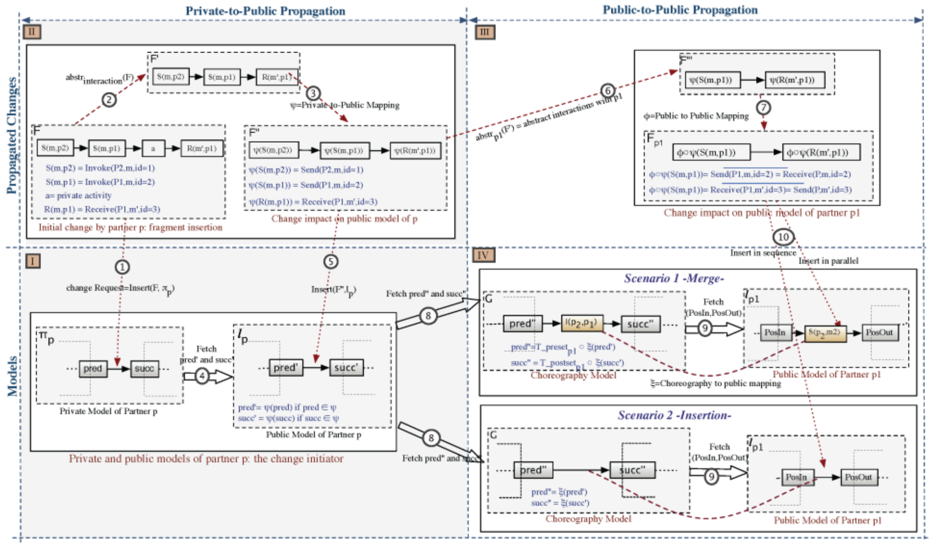
W. Fdhila, S. Rinderle-Ma, C. Indiono: Memetic Algorithms for Mining Change Logs in Process Choreographies, ICSOC 2014

Change Propagation: Overall Picture


Isolated or Propagating Change
Compute Private-to-Public Effects
Compute Public-to-Public Effects
PrM
PrM
PrM&PuM
PuM
Start
Specify change
Isolated Change?
No
Yes
Infer interaction changes
Compute affected partners
Do
Select an affected partner
Compute changes to propagate to this partner
PuM & ChM
last partner?
No
Yes
Computing Propagation Effects
Negotiation & Implementation of Changes
PrM
Update private models
ChM
Update choreography model
PuM
Update public models
Update Private Model
Succeed
Succeed
PrM
Find another alternative
No
End
Fail
Yes
abandon?
Yes
ok?
No
No
Check compatibility and consistency
Compute public2private changes
Yes
negotiations succeed?
Negotiate changes
PuM&ChM&PrM
PuM&PrM
Change Implementaion
Compute Public-to-Private Effects
Negotiation
Legend: ChM Uses Choreography Model PrM Uses Private Model PuM Uses Public Model

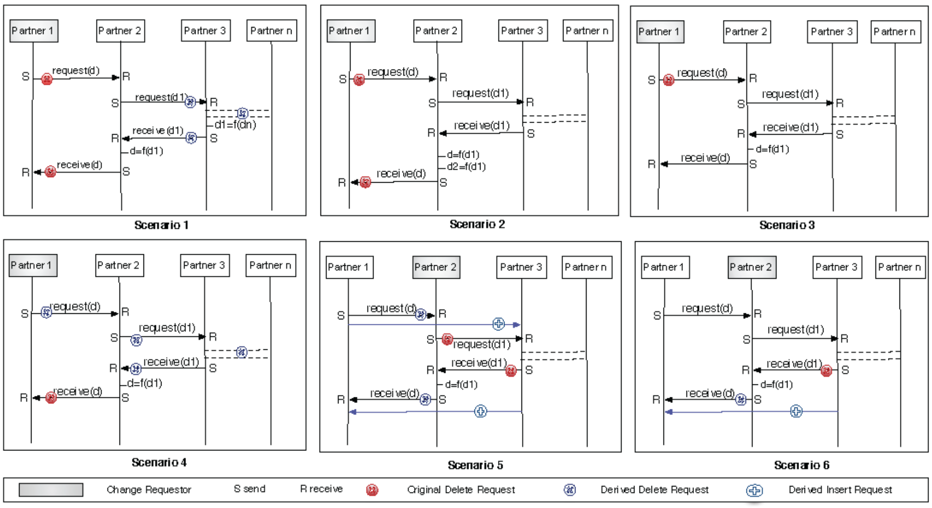
© Elsevier, 2015 W. Fdhila, C. Indiono, S. Rinderle-Ma, M. Reichert: Dealing with change in process choreographies: Design and implementation of propagation algorithms. Information Syst. 49:1 -24 (2015)

Example of propagating an INSERT


Private-to-Public Propagation
Public-to-Public Propagation
Propagated Changes
Models
II
III
I
IV
F'
S(m,p2)
S(m,p1)
R(m',p1)
abstr_interaction(F)
2
3
ψ=Private-to-Public Mapping
F
S(m,p2)
S(m,p1)
a
R(m',p1)
S(m,p2) = Invoke(P2,m,id=1)
S(m,p1) = Invoke(P1,m,id=2)
a= private activity
R(m,p1) = Receive(P1,m',id=3)
Initial change by partner p: fragment insertion
F''
ψ(S(m,p2))
ψ(S(m,p1))
ψ(R(m',p1))
ψ(S(m,p2)) = Send(P2,m,id=1)
ψ(S(m,p1)) = Send(P1,m,id=2)
ψ(R(m,p1)) = Receive(P1,m',id=3)
Change impact on public model of p
abstr_p1(F') = abstract interactions with p1
6
F'''
ψ(S(m,p1))
ψ(R(m',p1))
7
φ=Public to Public Mapping
F_p1
φ○ψ(S(m,p1))
φ○ψ(R(m',p1))
φ○ψ(S(m,p1))= Send(P1,m,id=2) = Receive(P,m,id=2)
φ○ψ(S(m,p1))= Receive(P1,m',id=3)= Send(P,m',id=3)
Change impact on public model of partner p1
10
Insert in sequence
Insert in parallel
1
change Request=Insert(F, π_p)
5
Insert(F'',l_p)
Fetch pred'' and succ''
8
π_p
pred
succ
Private Model of Partner p
Fetch pred' and succ'
4
l_p
pred'
succ'
pred'= ψ(pred) if pred ∈ ψ
succ' = ψ(succ) if succ ∈ ψ
Public Model of Partner p
Private and public models of partner p: the change initiator
8
Fetch pred'' and succ''
Scenario 1 -Merge-
G
pred''
I(p2,p1)
succ''
pred''=T_preset_p1 ○ ξ(pred')
succ'' = T_postset_p1 ○ ξ(succ')
Choreography Model
Fetch (PosIn,PosOut)
9
l_p1
PosIn
S(p2,m2)
PosOut
Public Model of Partner p1
ξ=Choreography to public mapping
Scenario 2 -Insertion-
G
pred''
succ''
pred''= ξ(pred')
succ'' = ξ(succ')
Choreography Model
Fetch (PosIn,PosOut)
9
l_p1
PosIn
PosOut
Public Model of Partner p1

© Elsevier, 2015 W. Fdhila, C. Indiono, S. Rinderle-Ma, M. Reichert: Dealing with change in process choreographies: Design and implementation of propagation algorithms. Information Syst. 49:1 -24 (2015)

Transitivity in Change Propagation



© Elsevier, 2015 W. Fdhila, C. Indiono, S. Rinderle-Ma, M. Reichert: Dealing with change in process choreographies: Design and implementation of propagation algorithms. Information Syst. 49:1 -24 (2015)

Change Propagation Prototype

	Components	GUI Tools	Technologies Used
Process Change	Change Management Service Change Validation Versioning Static Adaptation Negotiation Impact Analysis Dynamic Adaptation	C3Pro Editor	IR RPST (jBPT) REST Clojure Java
Process Definition	Private Models ↔ consistency ↔ Public Models compatibility Public Models ↔ compatibility ↔ Choreography Model	JBPM Signavio	XML JSON BPMN 2.0
Process Execution	Execution Engine Event Bus Monitoring Logging Messaging Clients	Cloud Process Execution Engine (CPEE)	XML JSON Ruby BPEL

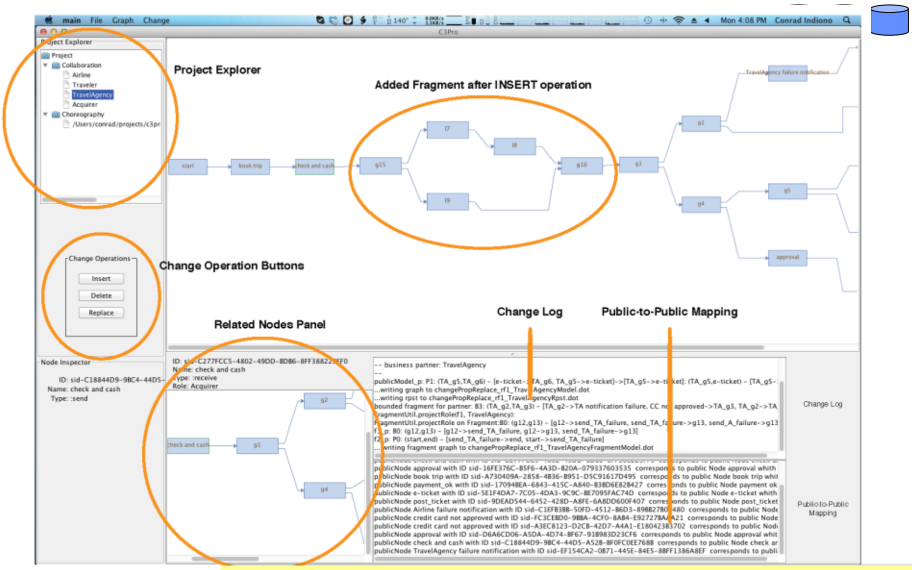
© Elsevier, 2015 W. Fdhila, C. Indiono, S. Rinderle-Ma, M. Reichert: Dealing with change in process choreographies: Design and implementation of propagation algorithms. Information Syst. 49:1 -24 (2015)

Cloud Process Execution Engine

Check also out: Cloud Process Execution Engine:cpee.org

Change Propagation Prototype

Change
and Compliance
for Collaborative Processes



```
ID: sid-C18844D9-9BC4-44D5-
Name: check and cash
Type: :send
```

```
ID: sid-C277FCC5-4802-49DD-8D86-8FF388223FF0
Name: check and cash
Type: :receive
Role: Acquirer
```

```
-- business partner: TravelAgency
--
publicModel_p: P1: (TA_g5,TA_g6) - [e-ticket->TA_g6, TA_g5->e-ticket]->[TA_g5->e-ticket]: (TA_g5,e-ticket) - [TA_g5-
...writing graph to changePropReplace_rf1_TravelAgencyModel.dot
...writing rpst to changePropReplace_rf1_TravelAgencyRpst.dot
bounded fragment for partner: B3: (TA_g2,TA_g3) - [TA_g2->TA notification failure, CC not approved->TA_g3, TA_g2->TA
FragmentUtil.projectRole(f1, TravelAgency):
FragmentUtil.projectRole on Fragment:B0: (g12,g13) - [g12->send_TA_failure, send_TA_failure->g13, send_A_failure->g13
f1_p: B0: (g12,g13) - [g12->send_TA_failure, g12->g13, send_TA_failure->g13]
f2_p: P0: (start,end) - [send_TA_failure->end, start->send_TA_failure]
...writing fragment graph to changePropReplace_rf1_TravelAgencyFragmentModel.dot
```

```
publicNode approval with ID sid-16FE376C-B5F6-4A3D-820A-079337603535  corresponds to public Node approval whith
publicNode book trip with ID sid-A730409A-2858-4B36-B951-D5C91617D495  corresponds to public Node book trip whit
publicNode payment_ok with ID sid-170948EA-6843-415C-A840-83BD6E828427  corresponds to public Node payment ok
publicNode e-ticket with ID sid-5E1F4DA7-7C05-4DA3-9C9C-BE7095FAC74D  corresponds to public Node e-ticket whith
publicNode post_ticket with ID sid-9DEAD544-6452-428D-A8FE-6A8DD600F407  corresponds to public Node post_ticket
publicNode Airline failure notification with ID sid-C1EFB38B-50FD-4512-B6D3-898B27B0480  corresponds to public Node
publicNode credit card not approved with ID sid-FC3CEBD0-9BBA-4CF0-8AB4-E92727BAAA21  corresponds to public Node
publicNode credit card not approved with ID sid-A3EC8123-D2CB-42D7-A4A1-E180423B3702  corresponds to public Node
publicNode approval with ID sid-D6A6CD06-A5DA-4D74-BF67-91B983D23CF6  corresponds to public Node approval whit
publicNode check and cash with ID sid-C18844D9-98C4-44D5-A528-8F0FC0EE7688  corresponds to public Node check ar
publicNode TravelAgency failure notification with ID sid-EF154CA2-0B71-445E-84E5-8BFF1386A8EF  corresponds to publi
```

http://www.wst.univie.ac.at/communities/c3pro/index.php?t=downloads

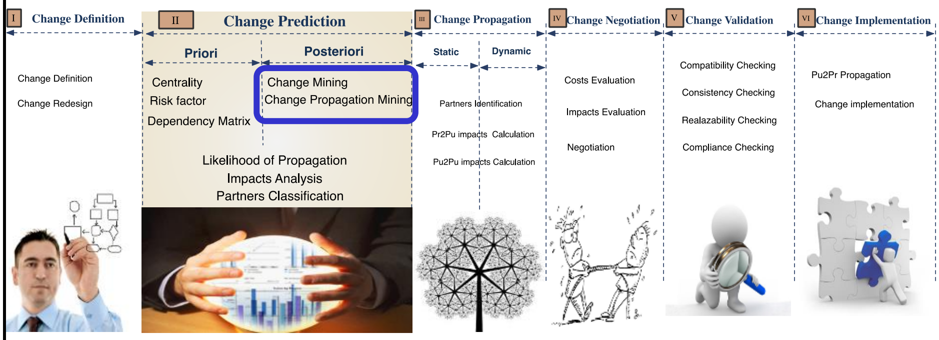
© Elsevier, 2015 W. Fdhila, C. Indiono, S. Rinderle-Ma, M. Reichert: Dealing with change in process choreographies: Design and implementation of propagation algorithms. Information Syst. 49:1 -24 (2015)

Change Impact Analysis and Prediction

Change
and Compliance
for Collaborative Processes

- Change can become expensive
- Particularly in case of propagation and negotiations

2 Change Impact Analysis and Prediction



I Change Definition

- Change Definition
- Change Redesign

II Change Prediction

Priori

- Centrality
- Risk factor
- Dependency Matrix

Posteriori

- Change Mining
- Change Propagation Mining

Likelihood of Propagation
Impacts Analysis
Partners Classification

III Change Propagation

Static | Dynamic

- Partners Identification
- Pr2Pu impacts Calculation
- Pu2Pu impacts Calculation

IV Change Negotiation

- Costs Evaluation
- Impacts Evaluation
- Negotiation

V Change Validation

- Compatibility Checking
- Consistency Checking
- Realazability Checking
- Compliance Checking

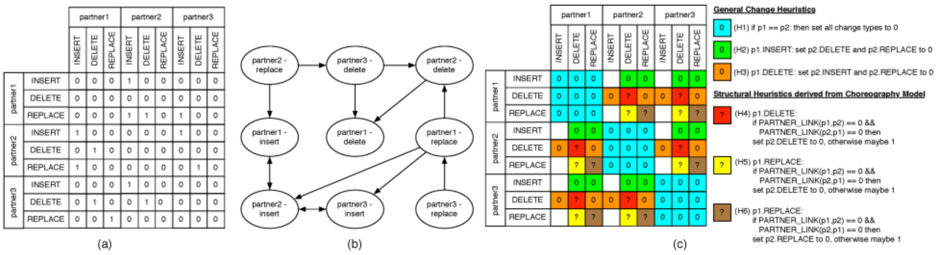
VI Change Implementation

- Pu2Pr Propagation
- Change implementation

- Change propagation mining: input data change propagation logs (CPL)
- Change mining: input data change event logs (CEL)

More challenging, but also more realistic (privacy!)

Change Impact Analysis and Prediction



		partner1			partner2			partner3		
		INSERT	DELETE	REPLACE	INSERT	DELETE	REPLACE	INSERT	DELETE	REPLACE
partner1	INSERT	0	0	0	1	0	0	0	0	0
	DELETE	0	0	0	0	0	0	0	0	0
	REPLACE	0	0	0	1	1	0	1	0	0
partner2	INSERT	1	0	0	0	0	0	1	0	0
	DELETE	0	1	0	0	0	0	0	0	0
	REPLACE	1	0	0	0	0	0	0	1	0
partner3	INSERT	0	0	0	1	0	0	0	0	0
	DELETE	0	1	0	0	1	0	0	0	0
	REPLACE	0	0	1	0	0	0	0	0	0

(a)

(b)

		partner1			partner2			partner3		
		INSERT	DELETE	REPLACE	INSERT	DELETE	REPLACE	INSERT	DELETE	REPLACE
partner1	INSERT	0	0	0		0	0		0	0
	DELETE	0	0	0	0	?	0	0	?	0
	REPLACE	0	0	0		?	?		?	?
partner2	INSERT		0	0	0	0	0		0	0
	DELETE	0	?	0	0	0	0	0	?	0
	REPLACE		?	?	0	0	0		?	?
partner3	INSERT		0	0		0	0	0	0	0
	DELETE	0	?	0	0	?	0	0	0	0
	REPLACE		?	?		?	?	0	0	0

(c)

General Change Heuristics

0 (H1) if p1 == p2: then set all change types to 0

0 (H2) p1.INSERT: set p2.DELETE and p2.REPLACE to 0

0 (H3) p1.DELETE: set p2.INSERT and p2.REPLACE to 0

Structural Heuristics derived from Choreography Model

? (H4) p1.DELETE:
if PARTNER_LINK(p1,p2) == 0 &&
PARTNER_LINK(p2,p1) == 0 then
set p2.DELETE to 0, otherwise maybe 1

? (H5) p1.REPLACE:
if PARTNER_LINK(p1,p2) == 0 &&
PARTNER_LINK(p2,p1) == 0 then
set p2.DELETE to 0, otherwise maybe 1

? (H6) p1.REPLACE:
if PARTNER_LINK(p1,p2) == 0 &&
PARTNER_LINK(p2,p1) == 0 then
set p2.REPLACE to 0, otherwise maybe 1

© Springer 2014, W. Fdhila, S. Rinderle-Ma, C. Indiono: Memetic Algorithms for Mining Change Logs in Process Choreographies, ICSOC 2015

Collaboration

C^{3}Pro
Change
and Compliance
for Collaborative Processes

Achieved so far:

- ❒ Generic approach (Refined Process Structure Tree RPST)
- ❒ Four change patterns
- ❒ Transitive effects
- ❒ Implementation of C^{3}Pro Editor

Still many open challenges

- ❒ **Compliance and Security**
- ❒ Change impact analysis
- ❒ Behavorial correctness
- ❒ Concurrent changes

C^{3}Pro (Selected Publications)

Linh Thao Ly, Fabrizio Maria Maggi, Marco Montali, Stefanie Rinderle-Ma, Wil M. P. van der Aalst: A Framework for the Systematic Comparison and Evaluation of Compliance Monitoring Approaches. EDOC 2013: 7-1

David Knuplesch, Manfred Reichert, Walid Fdhila, Stefanie Rinderle-Ma: On Enabling Compliance of Cross-Organizational Business Processes. BPM 2013: 146-154

David Knuplesch, Manfred Reichert, Rüdiger Pryss, Walid Fdhila, Stefanie Rinderle-Ma: Ensuring compliance of distributed and collaborative workflows. CollaborateCom 2013: 133-142

David Knuplesch, Manfred Reichert, Linh Thao Ly, Akhil Kumar, Stefanie Rinderle-Ma: Visual Modeling of Business Process Compliance Rules with the Support of Multiple Perspectives. ER 2013: 106-120

Walid Fdhila, Stefanie Rinderle-Ma, Manfred Reichert: Change propagation in collaborative processes scenarios. CollaborateCom 2012: 452-461

Walid Fdhila, Stefanie Rinderle-Ma, Aymen Baouab, Olivier Perrin, Claude Godart: On evolving partitioned Web Service orchestrations. SOCA 2012: 1-6