# Business Process Technologies and Management
# 4. Interorganizational Processes

Prof. Dr. Stefanie Rinderle-Ma

Dr. Jürgen Mangler

Technical University of Munich

Department of Informatics

Chair of Information Systems

and Business Process Management

stefanie.rinderle-ma@tum.de

juergen.mangler@tum.de

# Literature and links

- Weske, M. (2019) Business Process Management - Concepts, Languages, Architectures, Third Edition. Springer 2019, ISBN 978-3-662-59431-5, pp. 1-417 2019
- https://www.signavio.com/bpm-academic-initiative/
- https://cpee.org/

# Teaching Objectives

In this chapter we want to

- introduce the notion of process choreographies
- distinguish process choreographies from process orchestrations
- show how to model process choreographies
- introduce notions of correctness for process choreographies
- introduce verification techniques for process choreographies

# Introduction

- *Process Orchestration*: One (executable) business process from the perspective and under the control of one particular partner (endpoint)
- *Process Choreography*: Exchange of messages between partners according to defined interaction rules between two or more partners (endpoints); also called global process.
- Examples for process choreographies:
    - healthcare
    - blockchain-based processes
    - logistics scenarios
    - supply chains

# Process Choreography Design

1. **High-level Structure Design**
   - Identifying participant roles and their communication structure
   - Conducted during the *participant identification phase*

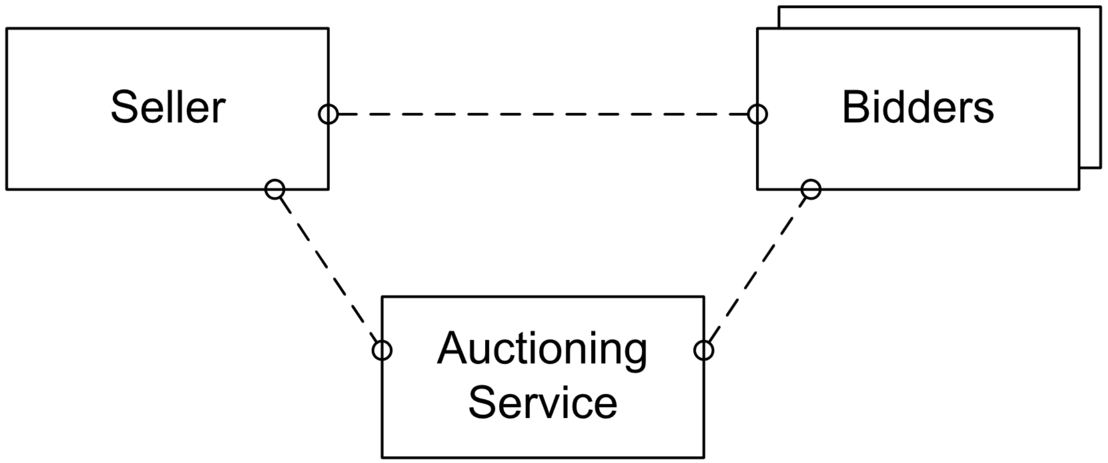


**Fig. 6.5.** High-level structural model of participants in bidding scenario

M. Weske: Business Process Management, Third Edition
© Springer-Verlag 2019, 2012, 2007

# Process Choreography Design

**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*

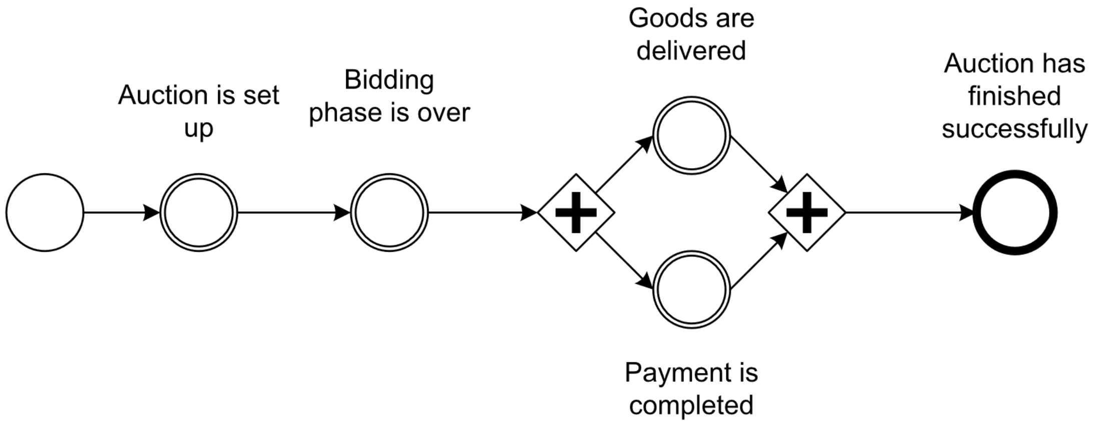
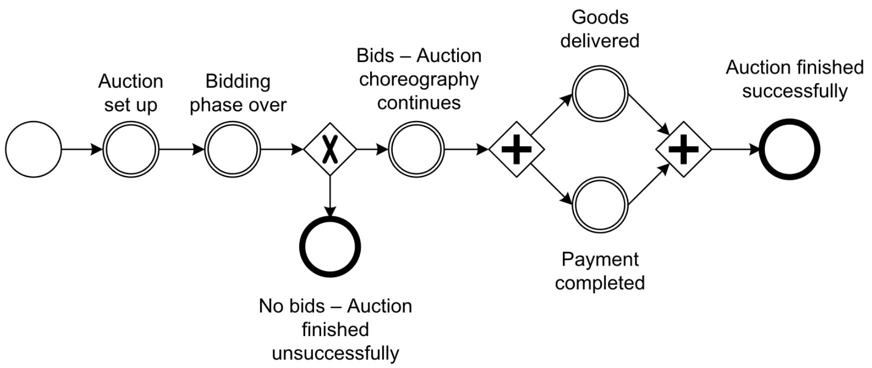


**Fig. 6.6.** High-level behavioural model for bidding scenario, represented by milestones

M. Weske: Business Process Management, Third Edition
© Springer-Verlag 2019, 2012, 2007

# Process Choreography 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



Auction set up
Bidding phase over
Bids – Auction choreography continues
Goods delivered
Auction finished successfully
Payment completed
No bids – Auction finished unsuccessfully

**Fig. 6.7.** High-level behavioural model for bidding scenario, with different outcomes

M. Weske: Business Process Management, Third Edition
© Springer-Verlag 2019, 2012, 2007

# Process Choreography Design

**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*

# Process Choreography Design

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

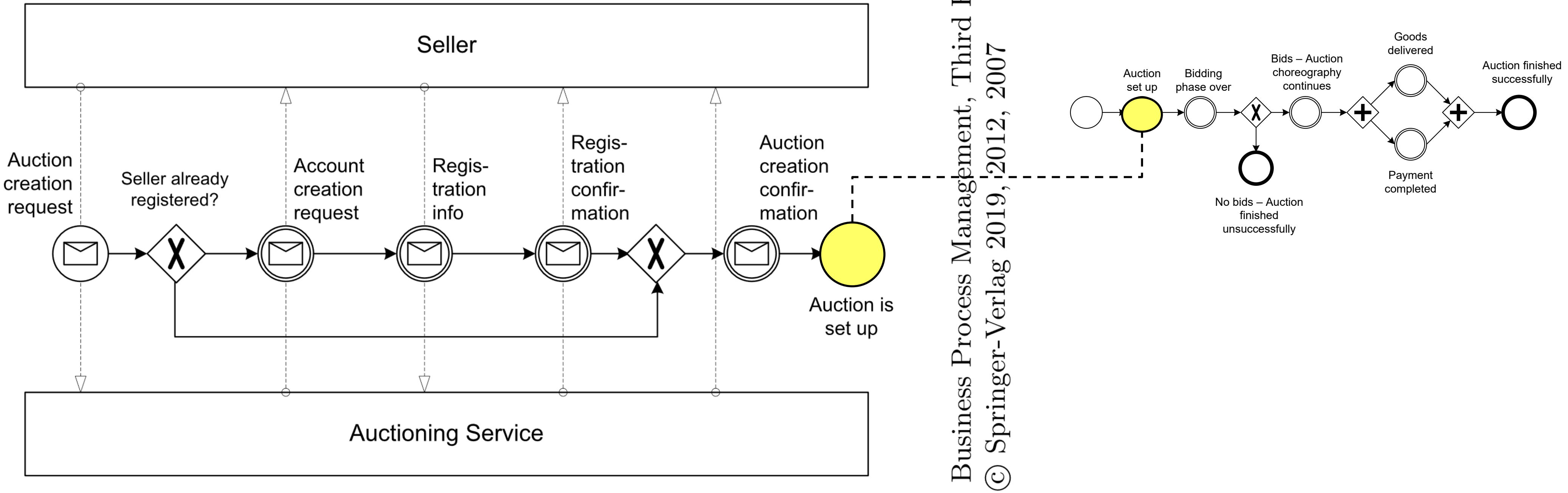

Seller
Auctioning Service
Auction creation request
Seller already registered?
Account creation request
Regis-tration info
Regis-tration confir-mation
Auction creation confir-mation
Auction is set up
Auction set up
Bidding phase over
Bids – Auction choreography continues
Goods delivered
Payment completed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

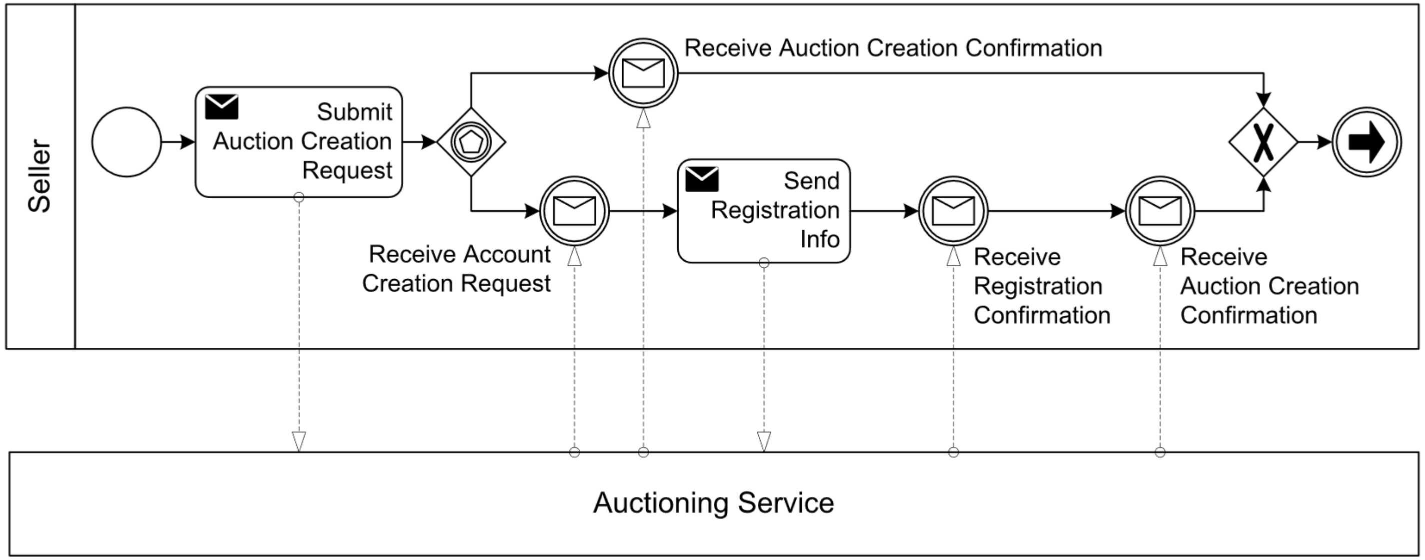
**Fig. 6.8.** Collaboration scenario: reaching milestones through interactions

M. Weske: Business Process Management, Third Edition
© Springer-Verlag 2019, 2012, 2007

# Process Choreography Design

**4. Behavioral Interfaces**

- Deriving a behavioral interface for each participant role from the collaboration scenarios



Seller
Submit Auction Creation Request
Receive Auction Creation Confirmation
Receive Account Creation Request
Send Registration Info
Receive Registration Confirmation
Receive Auction Creation Confirmation
Auctioning Service

**Fig. 6.9.** Behavioural interface for seller

M. Weske: Business Process Management, Third Edition
© Springer-Verlag 2019, 2012, 2007

# Process Choreography Design

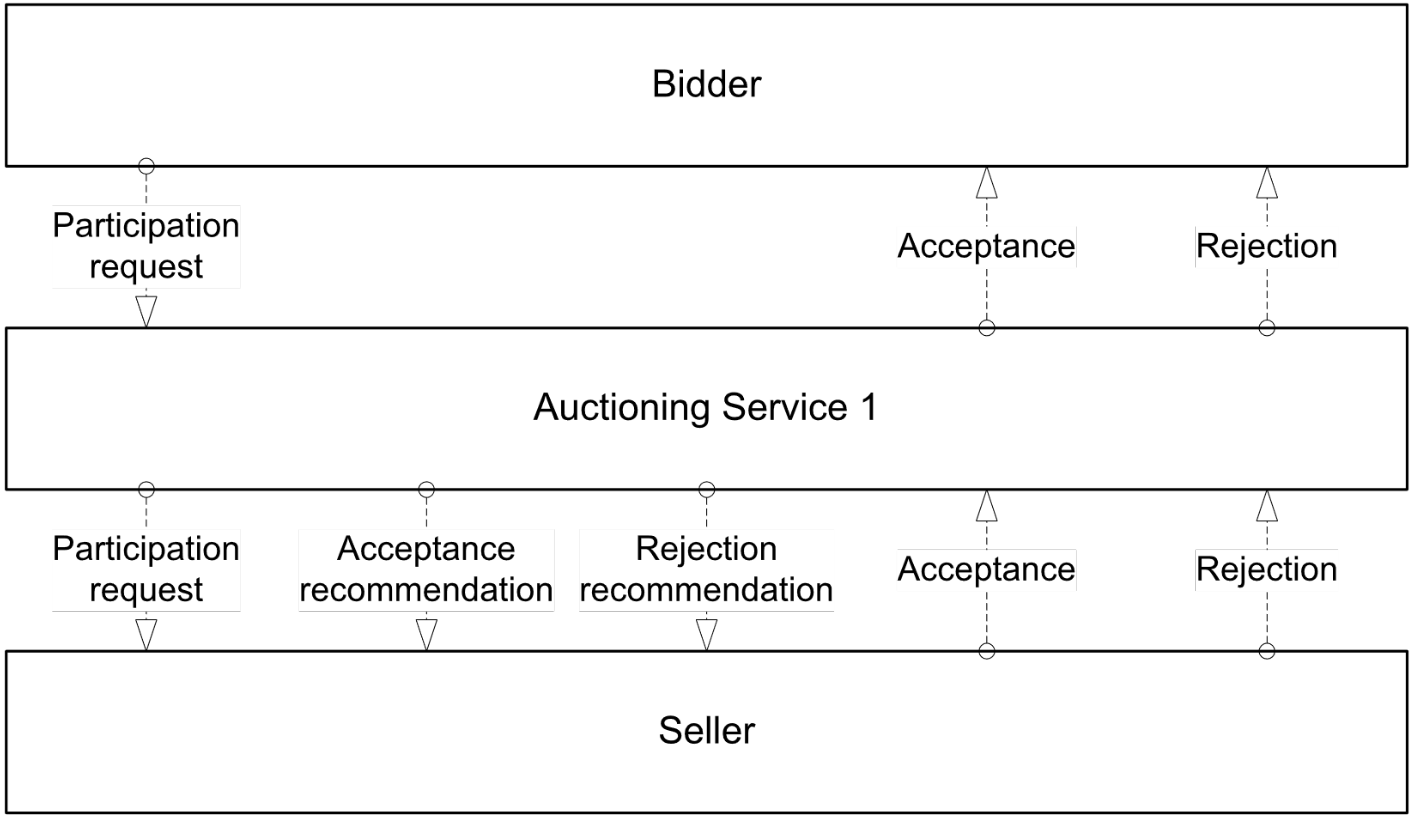
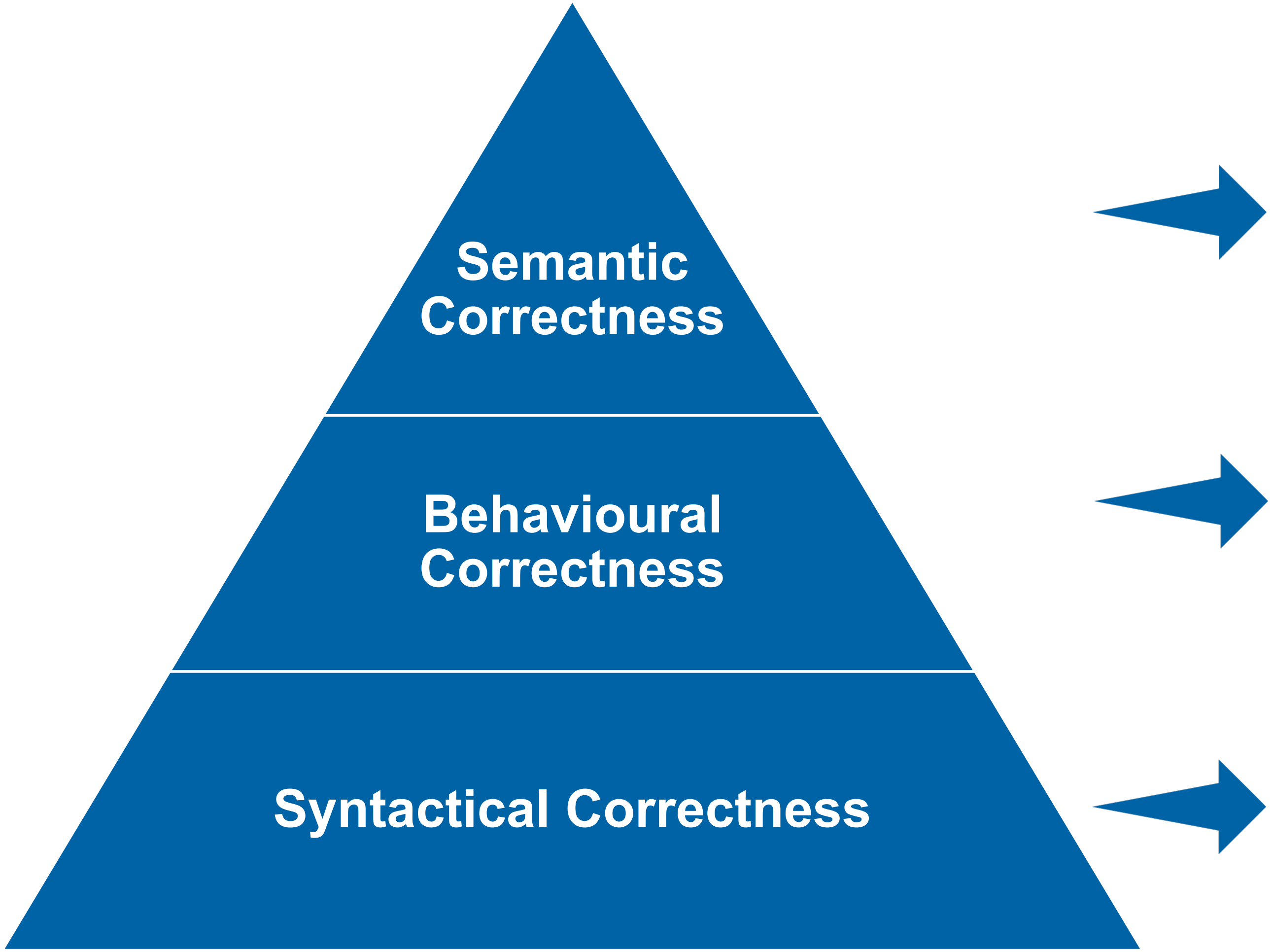


**Fig. 6.10.** Interactions of participants in auctioning scenario

M. Weske: Business Process Management, Third Edition

# Process Choreography Correctness



**The process models…**

…must **comply** with imposed local/global **compliance rules**.

…must be **executable** and the composition of the **public models** is **compatible** as well as that the **private models** are **consistent** with the **corresponding public model**.

…must ensures the **correct use** and **composition** of the corresponding **model elements**. This is defined by the underlying **BPMN meta model.**

# 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

# 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

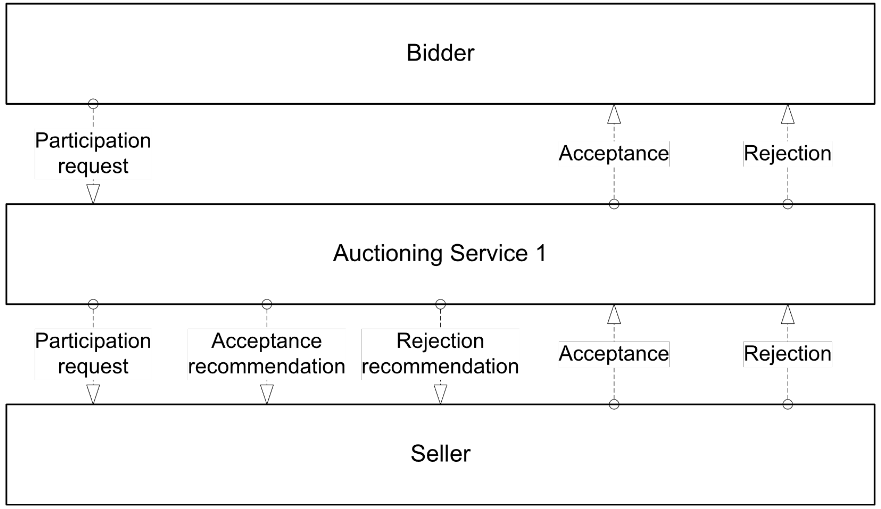


**Fig. 6.10.** Interactions of participants in auctioning scenario

M. Weske: Business Process Management, Third Edition
© Springer-Verlag 2019, 2012, 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*

# 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

# 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

# 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

# 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 non-empty 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

# 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

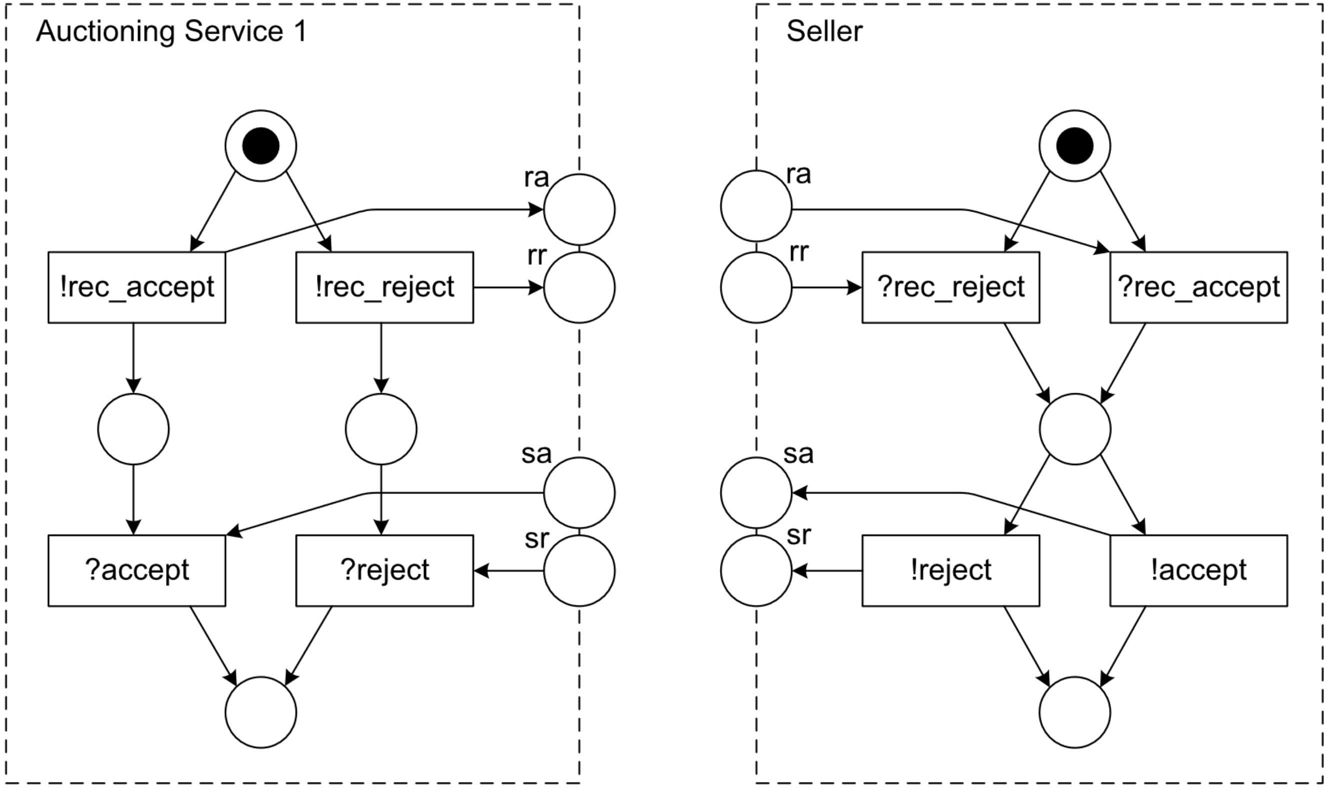


**Fig. 6.12.** Workflow modules as basis for checking compatibility

M. Weske: Business Process Management, Third Edition
© Springer-Verlag 2019, 2012, 2007

# Behavioral Compatibility

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

Not deadlock-free

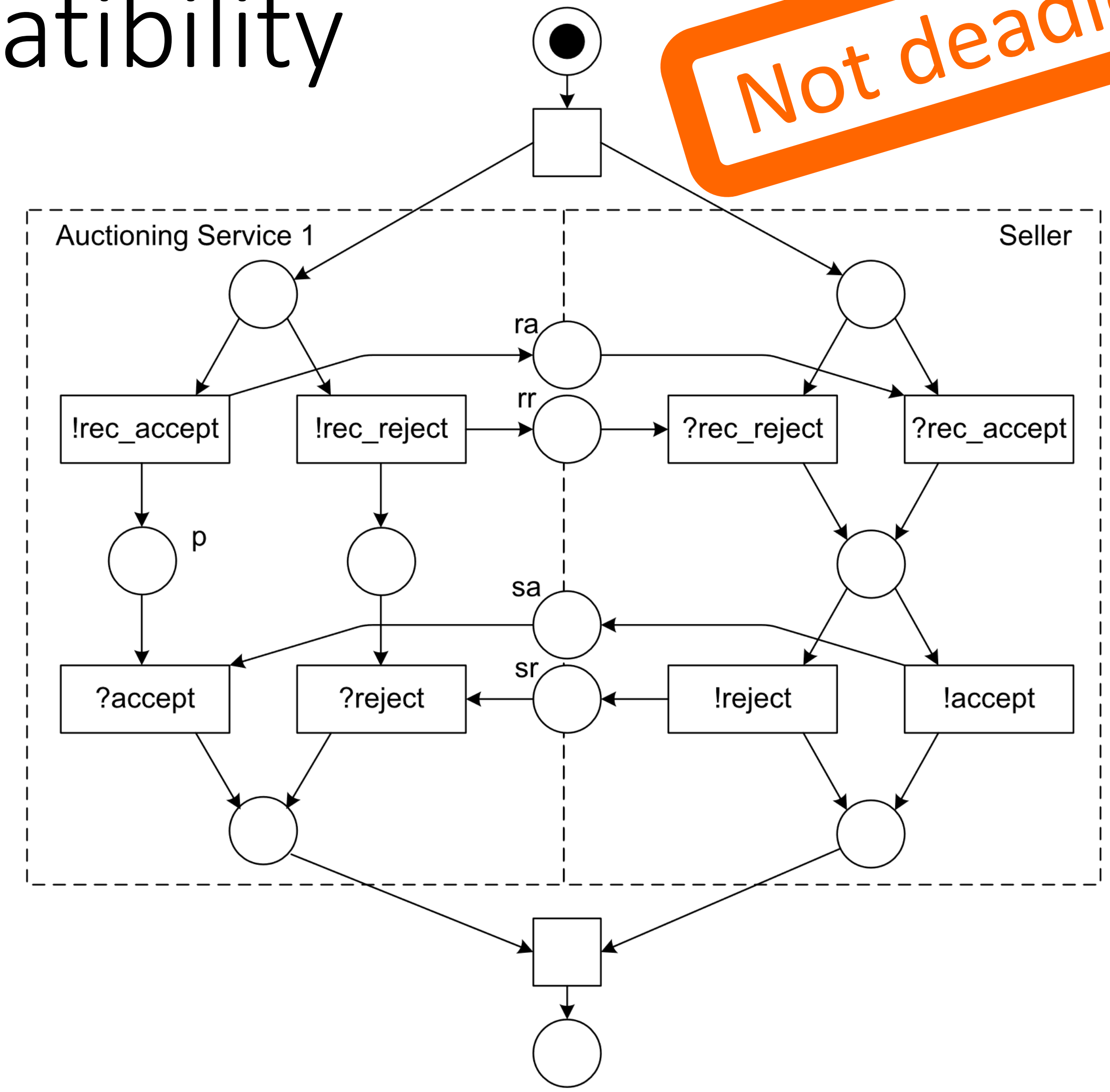


**Fig. 6.13.** Workflow net as composition of workflow modules

M. Weske: Business Process Management, Third Edition
© Springer-Verlag 2019, 2012, 2007

# Behavioral Compatibility

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

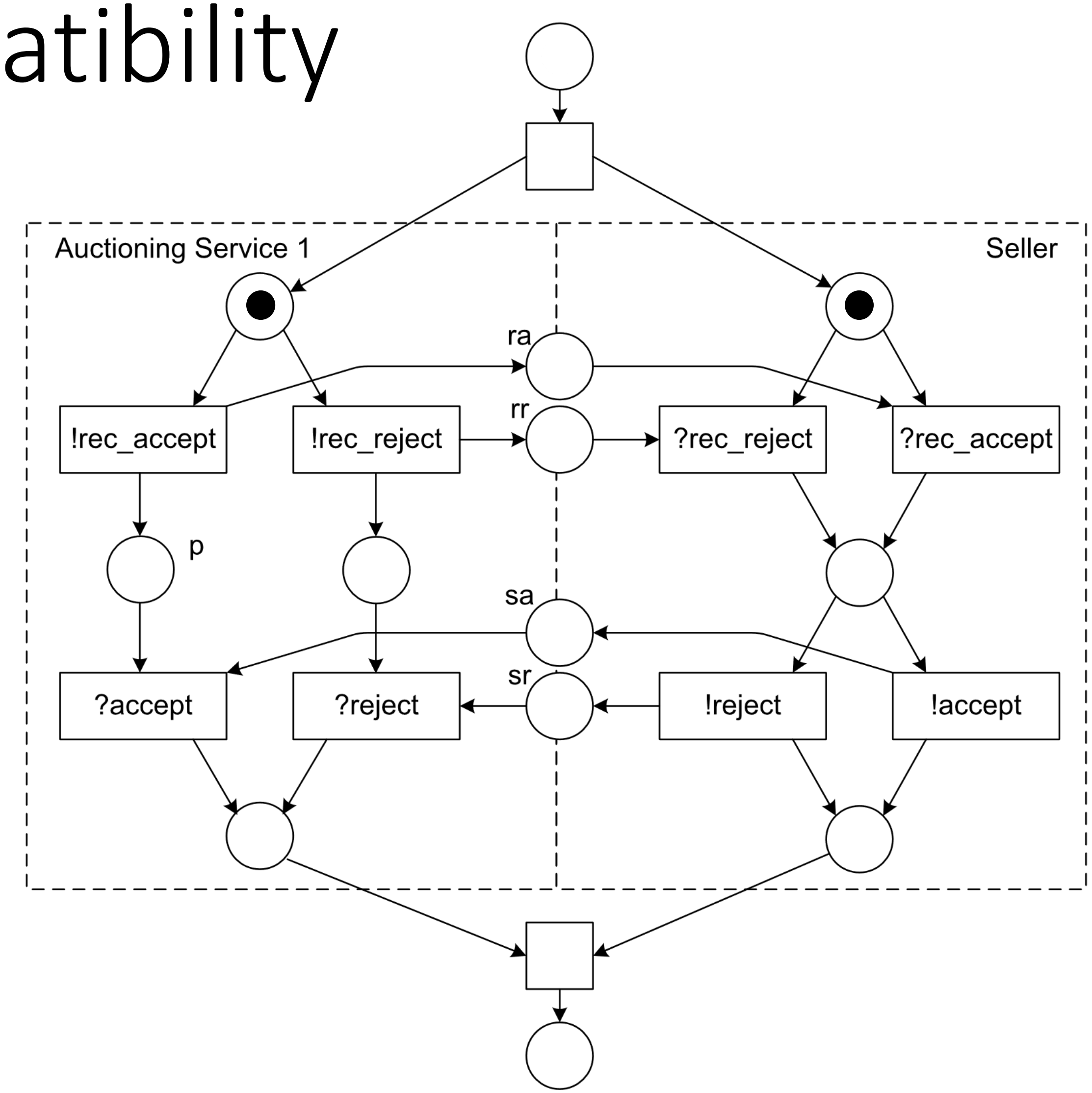
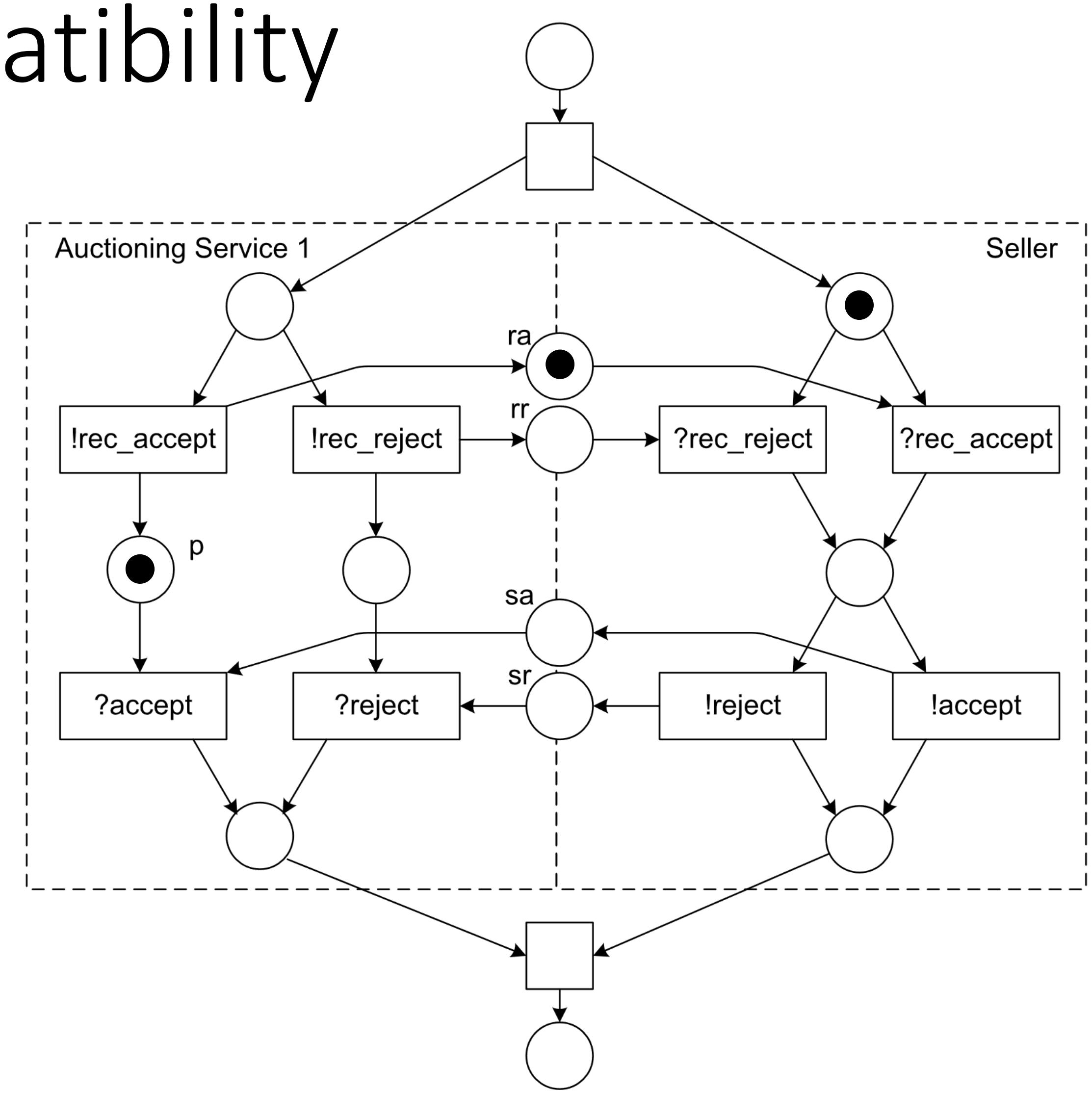


**Fig. 6.13.** Workflow net as composition of workflow modules

M. Weske: Business Process Management, Third Edition
© Springer-Verlag 2019, 2012, 2007

# Behavioral Compatibility

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

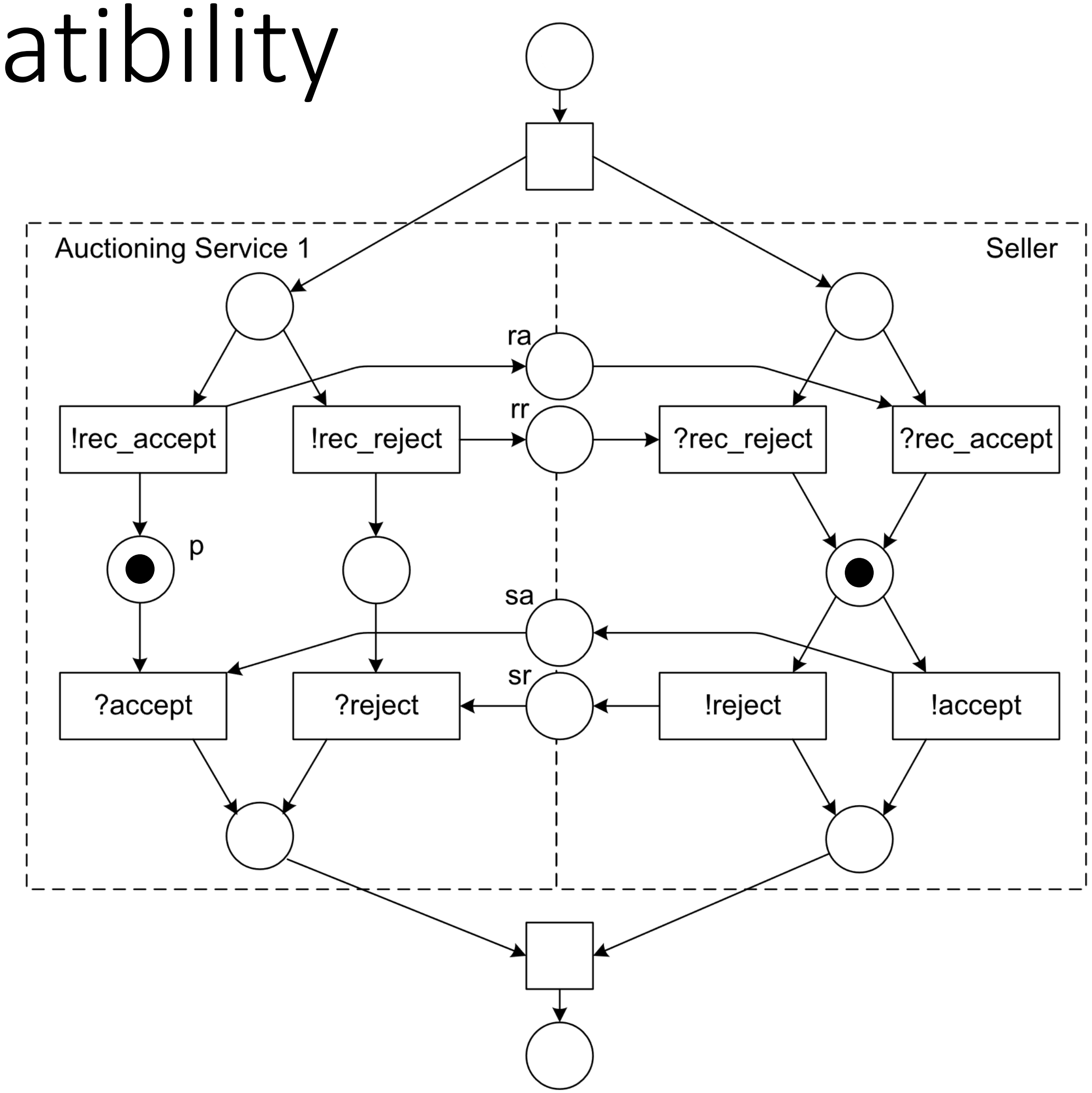


M. Weske: Business Process Management, Third Edition
© Springer-Verlag 2019, 2012, 2007

**Fig. 6.13.** Workflow net as composition of workflow modules

# Behavioral Compatibility

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



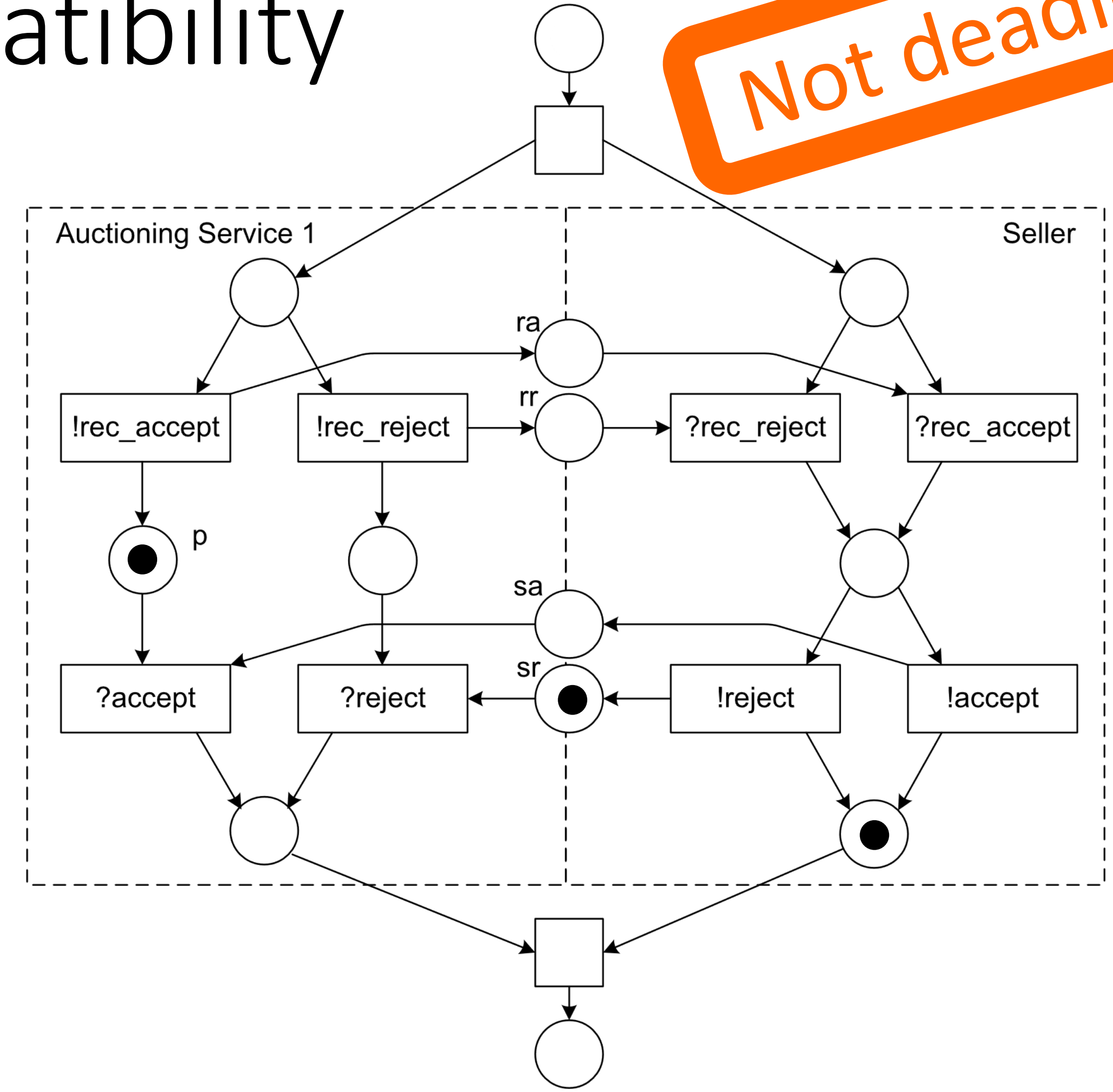
M. Weske: Business Process Management, Third Edition
© Springer-Verlag 2019, 2012, 2007

**Fig. 6.13.** Workflow net as composition of workflow modules

# 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, Third Edition
© Springer-Verlag 2019, 2012, 2007

**Fig. 6.13.** Workflow net as composition of workflow modules

# Behavioral Compatibility

- Deadlock free?
- Reachability analysis on the „combined“ workflow modules

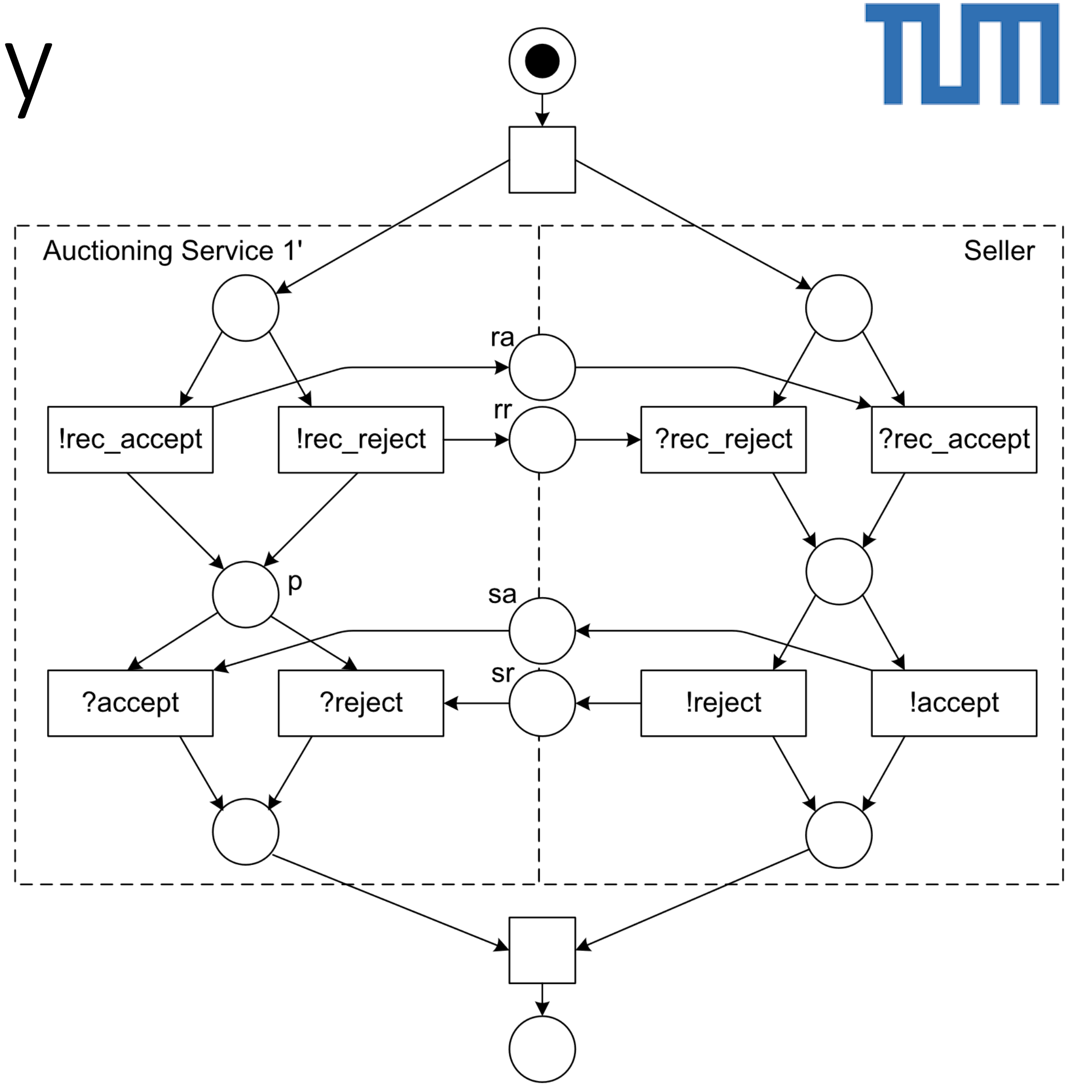
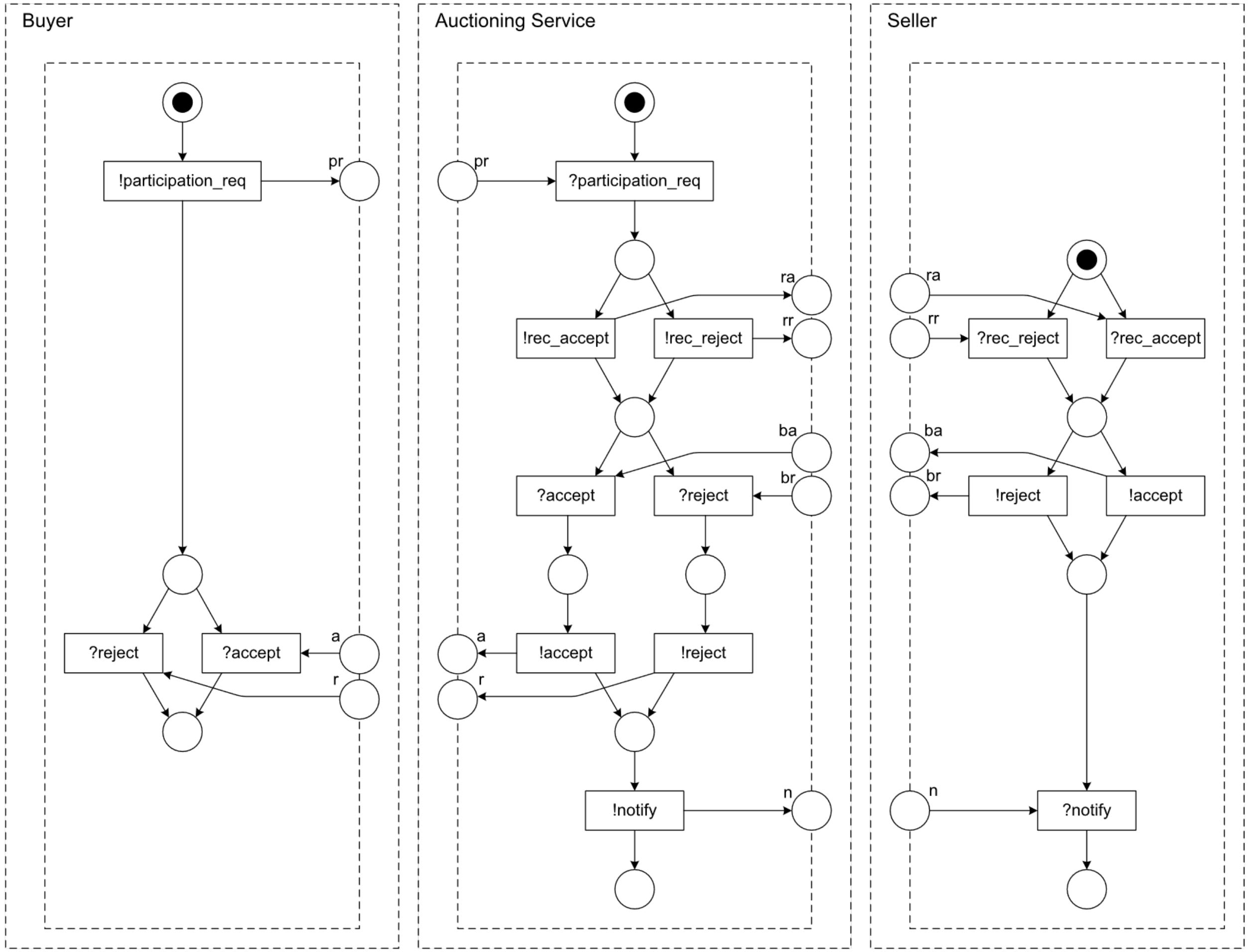


**Fig. 6.14.** Workflow modules that are compatible

M. Weske: Business Process Management, Third Edition
© Springer-Verlag 2019, 2012, 2007

# Behavioral Compatibility

- A larger part of the collaboration is depicted in the figure on the right
- Deadlock?



M. Weske: Business Process Management, Third Edition
© Springer-Verlag 2019, 2012, 2007

**Fig. 6.15.** Behavioural interfaces: getting a participation permission

# Choreography Patterns

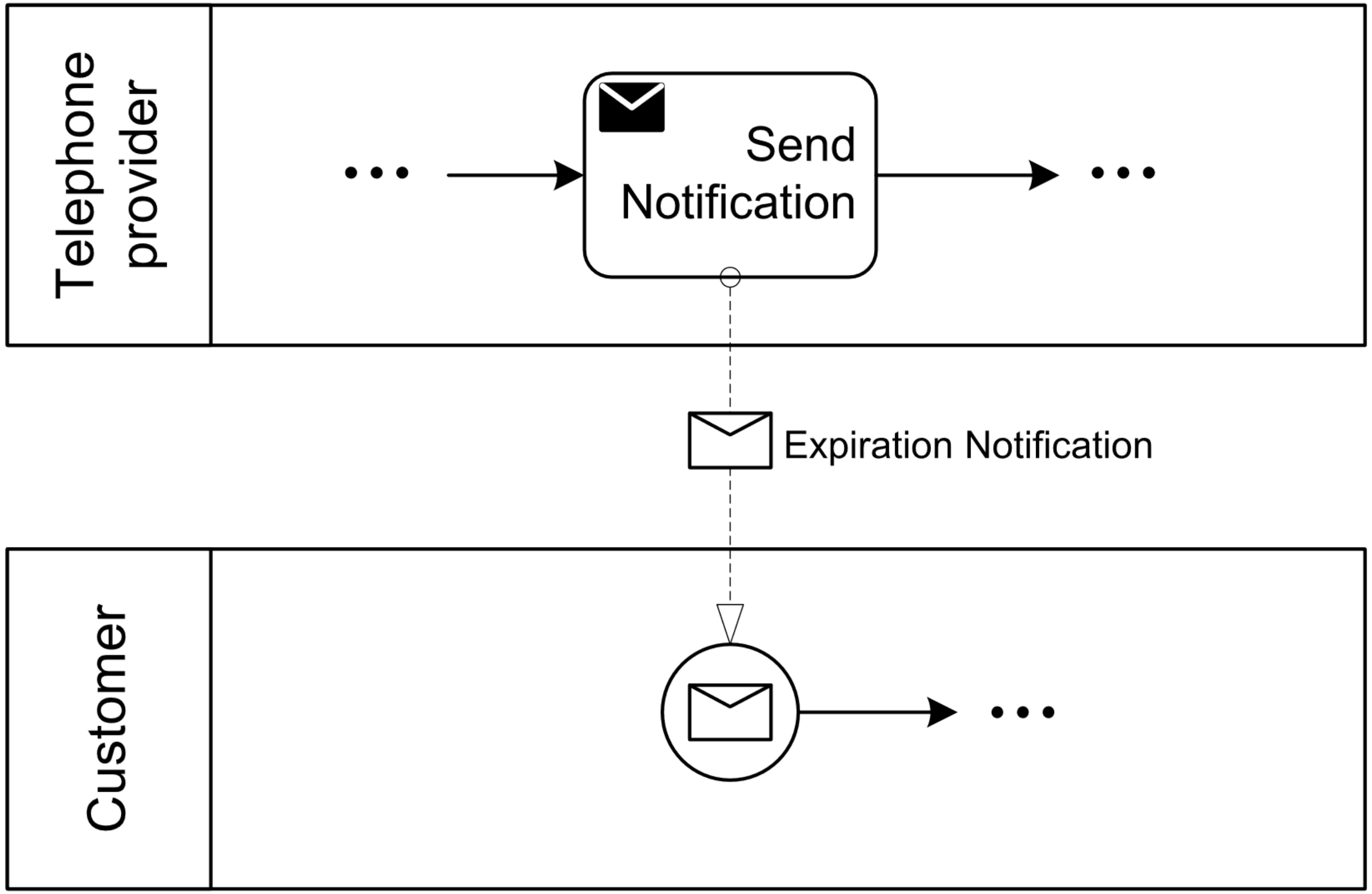


**Fig. 6.22.** Send pattern

M. Weske: Business Process Management, Third Edition
© Springer-Verlag 2019, 2012, 2007

# Choreography Patterns

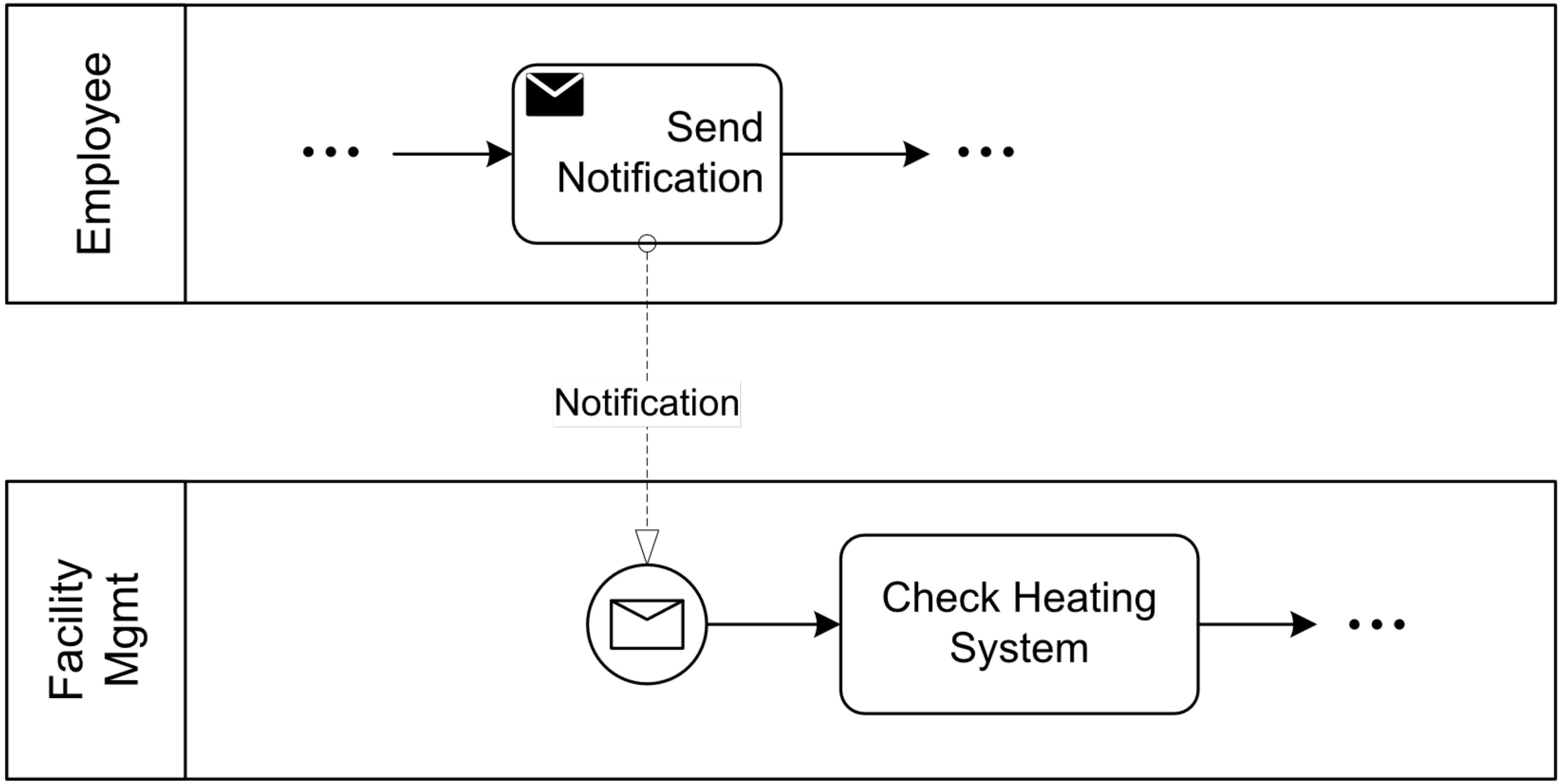


**Fig. 6.23.** Receive pattern

M. Weske: Business Process Management, Third Edition
© Springer-Verlag 2019, 2012, 2007

# Choreography Patterns

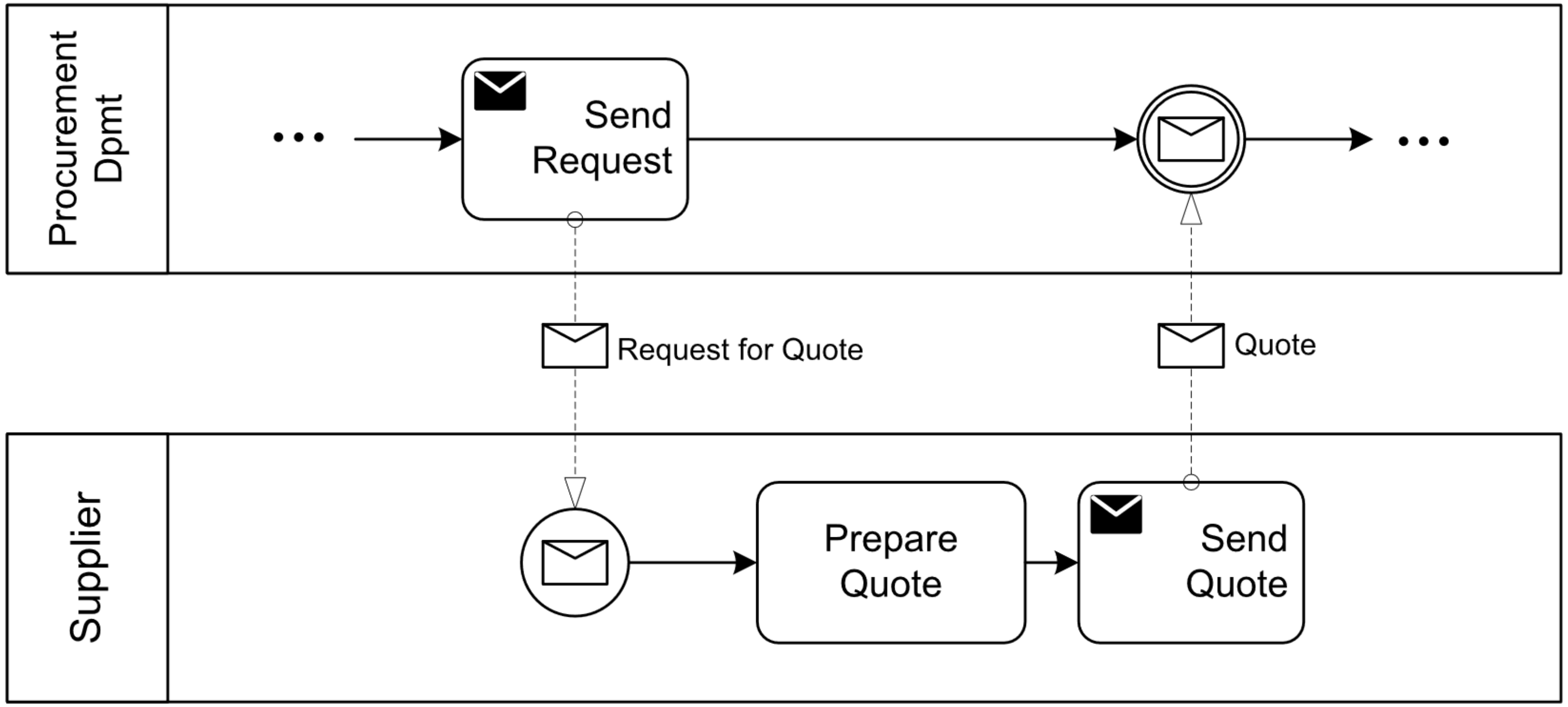


**Fig. 6.24.** Send/receive pattern

M. Weske: Business Process Managament, Third Edition
© Springer-Verlag 2019, 2012, 2007

# Choreography Patterns

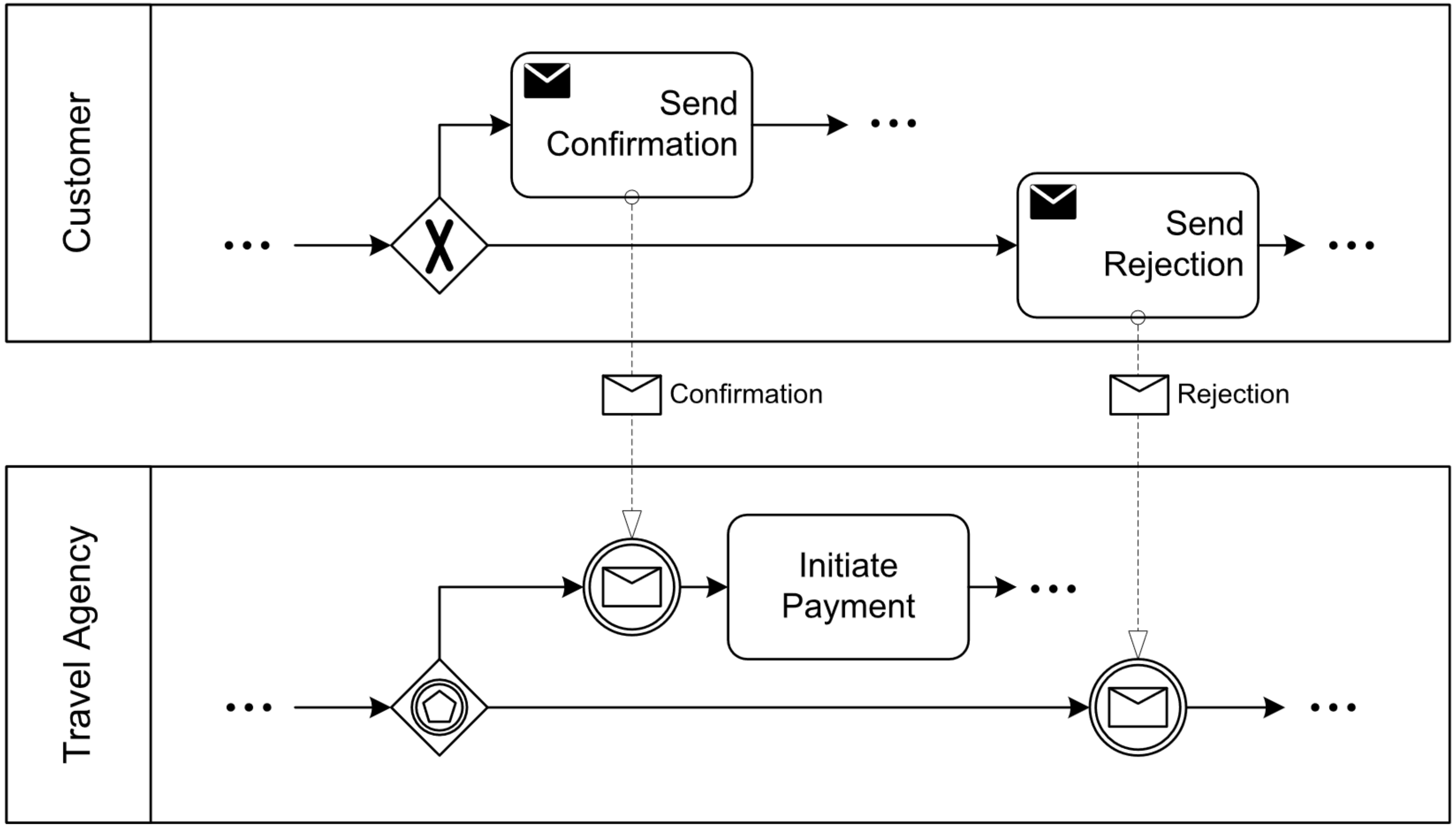


**Fig. 6.25.** Racing incoming messages pattern

M. Weske: Business Process Management, Third Edition
© Springer-Verlag 2019, 2012, 2007

# Choreography Patterns

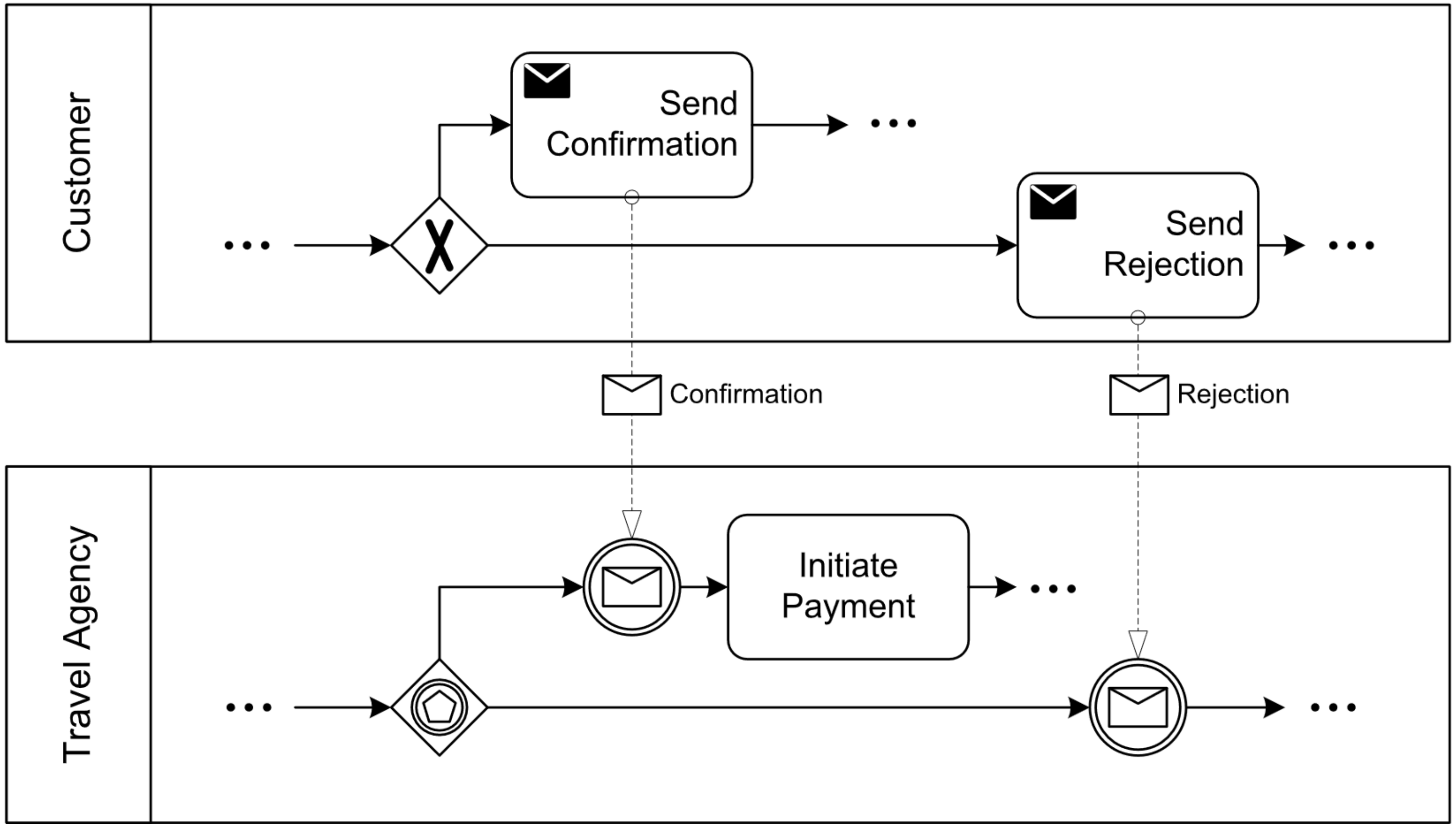


**Fig. 6.25.** Racing incoming messages pattern

M. Weske: Business Process Management, Third Edition
© Springer-Verlag 2019, 2012, 2007

# Process Choreographies

- Represented by four model types.
- They differ in terms of perspectives and level of detail:
  - Private Model
  - Public Model
  - Collaboration Model
  - Choreography Model

# Choreography Diagrams

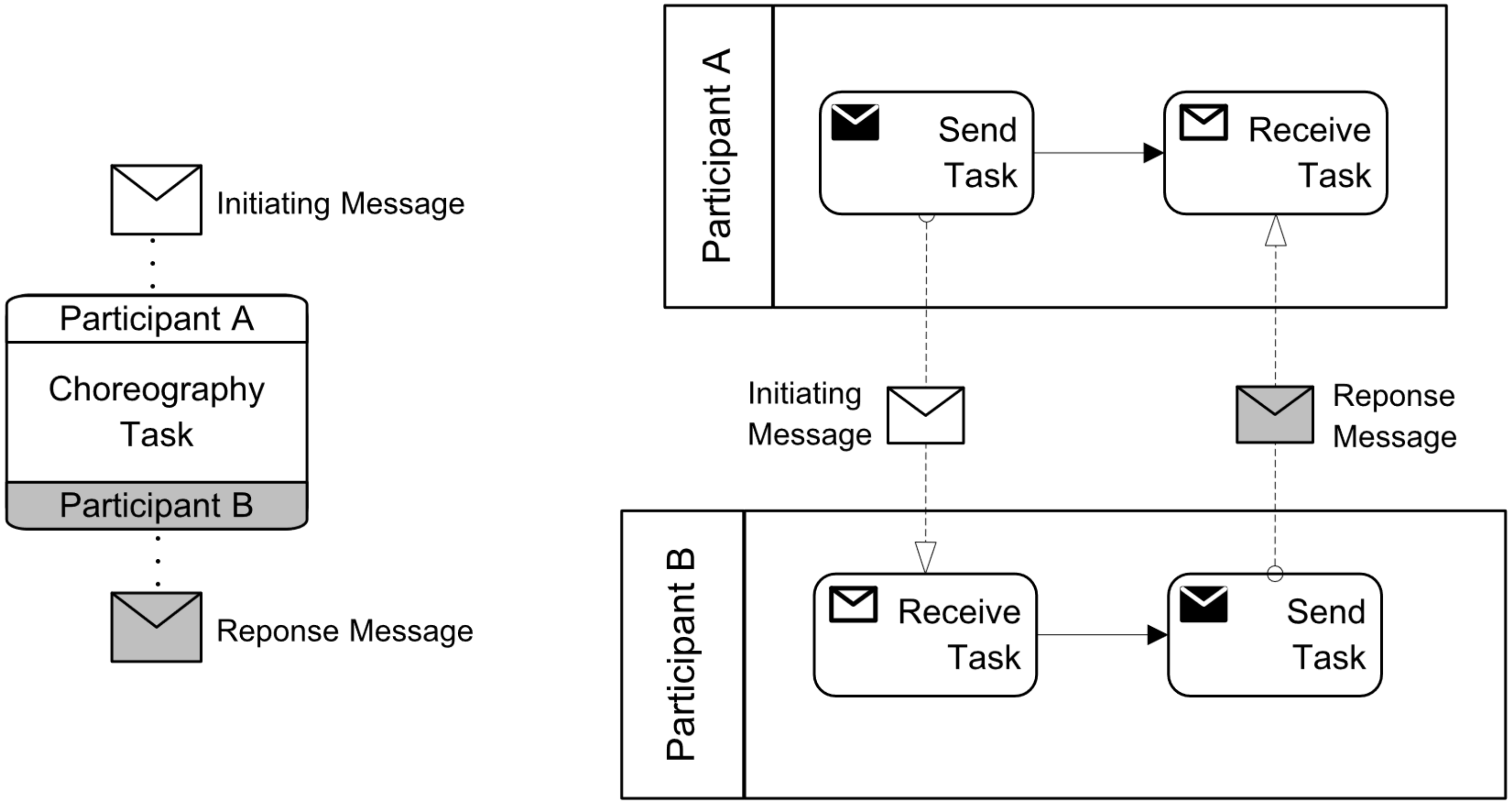
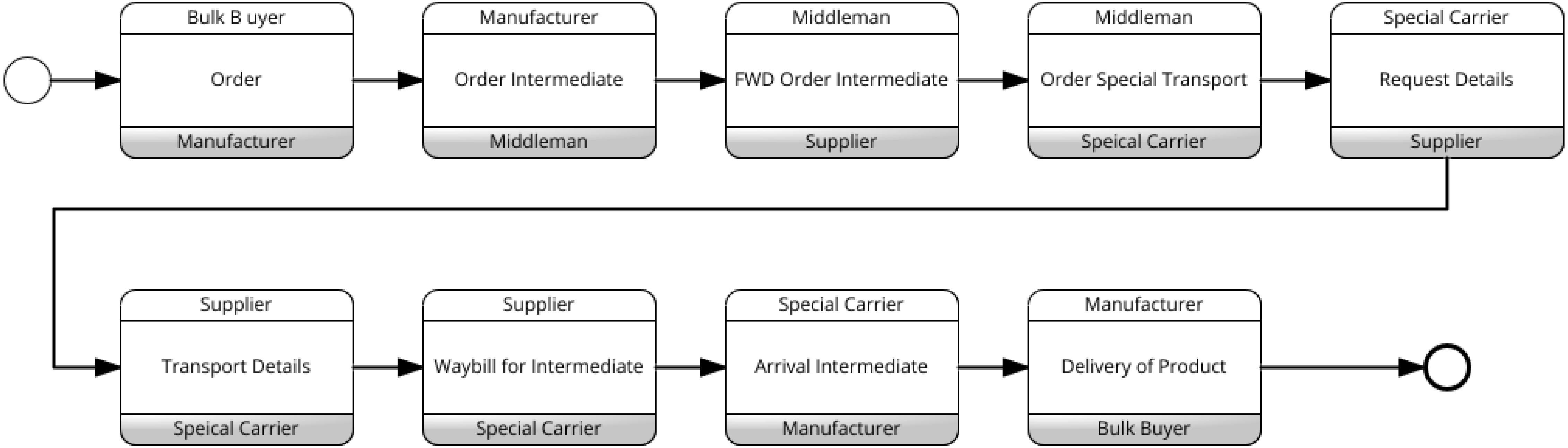


**Fig. 6.37.** Choreography task and corresponding collaboration diagram

M. Weske: Business Process Management, Third Edition
© Springer-Verlag 2019, 2012, 2007

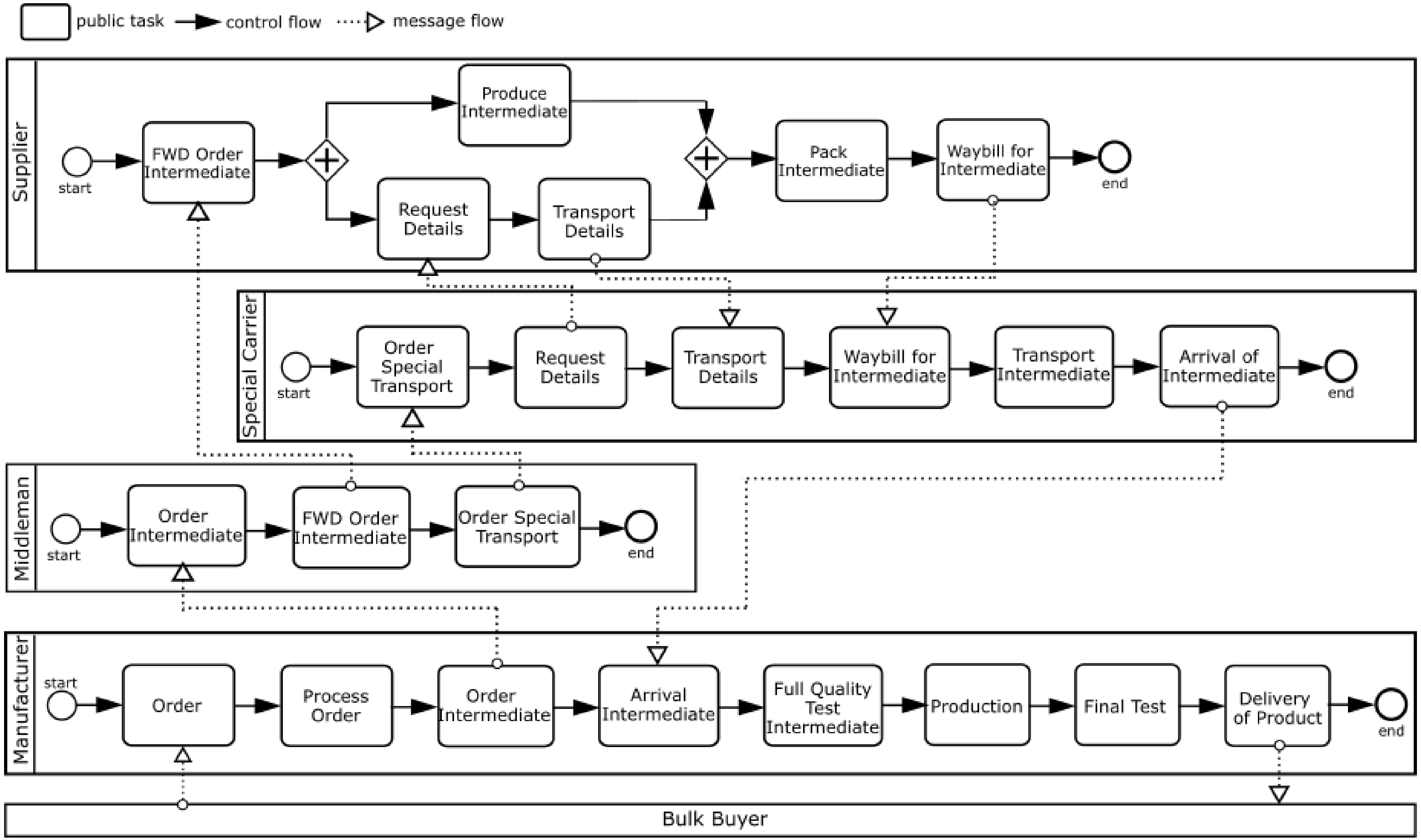
W. Fdhila, D. Knuplesch, S. Rinderle-Ma, M. Reichert: Verifying Compliance in Process Choreographies: Foundations, Algorithms, and Implementation. Information Systems 2022, https://doi.org/10.1016/j.is.2022.101983

# Supply Chain: Choreography



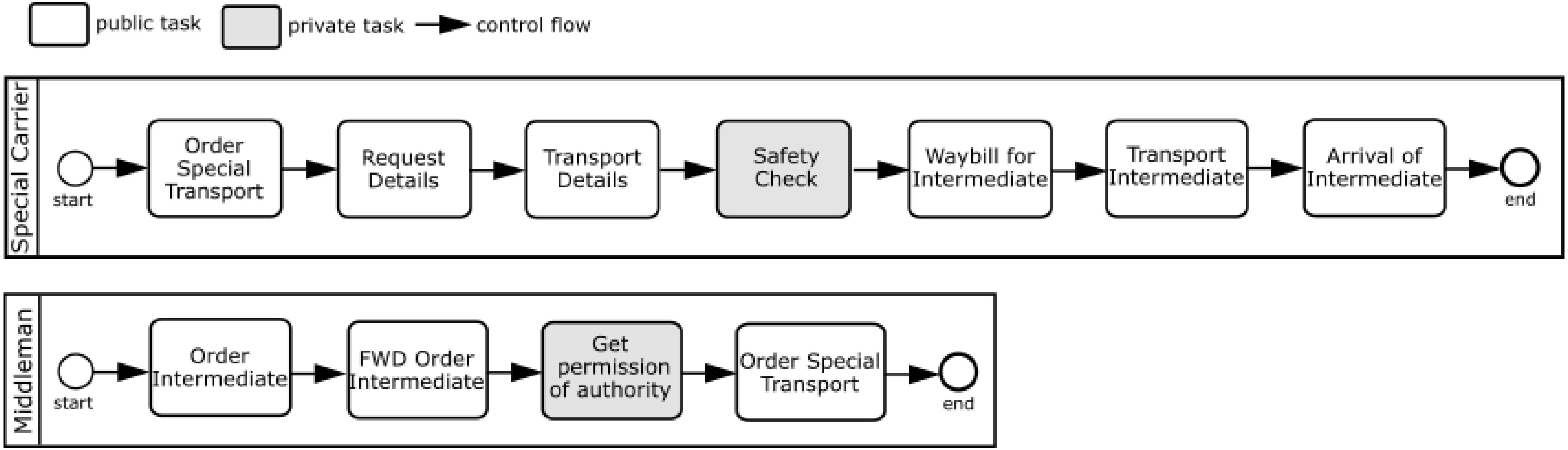
W. Fdhila, D. Knuplesch, S. Rinderle-Ma, M. Reichert: Verifying Compliance in Process Choreographies: Foundations, Algorithms, and Implementation. Information Systems 2022 (accepted for publication)

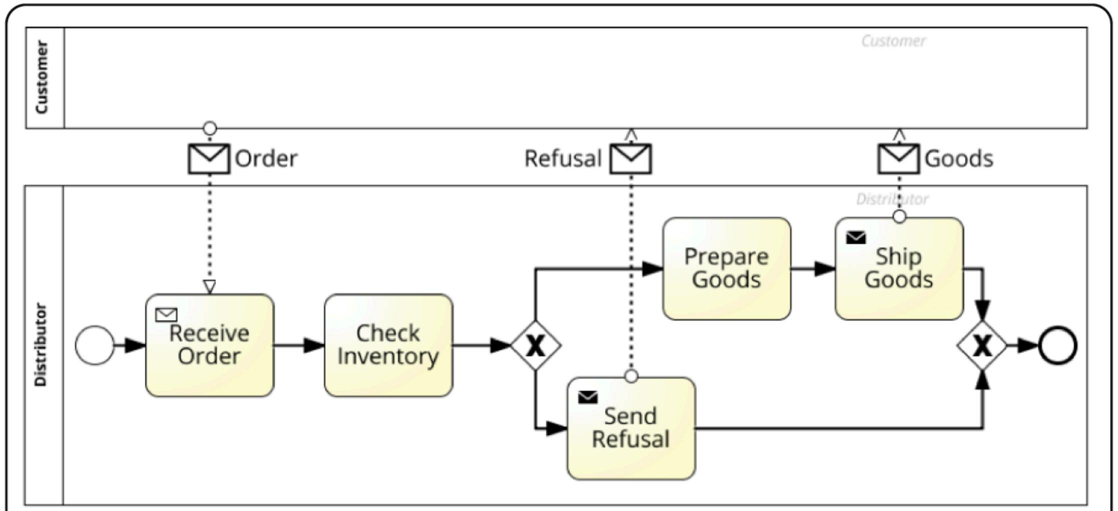
# Supply Chain: Collaboration



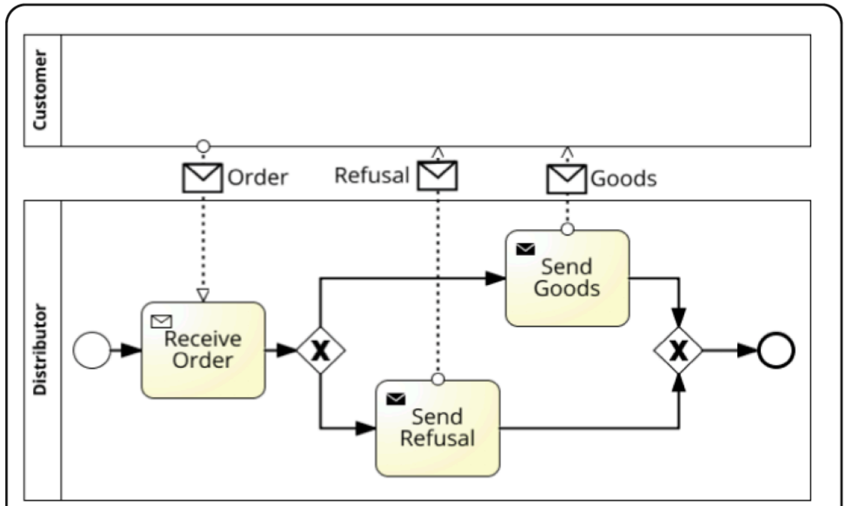
W. Fdhila, D. Knuplesch, S. Rinderle-Ma, M. Reichert: Verifying Compliance in Process Choreographies: Foundations, Algorithms, and Implementation. Information Systems 2022 (accepted for publication)

# Supply Chain: Selected Private Processes

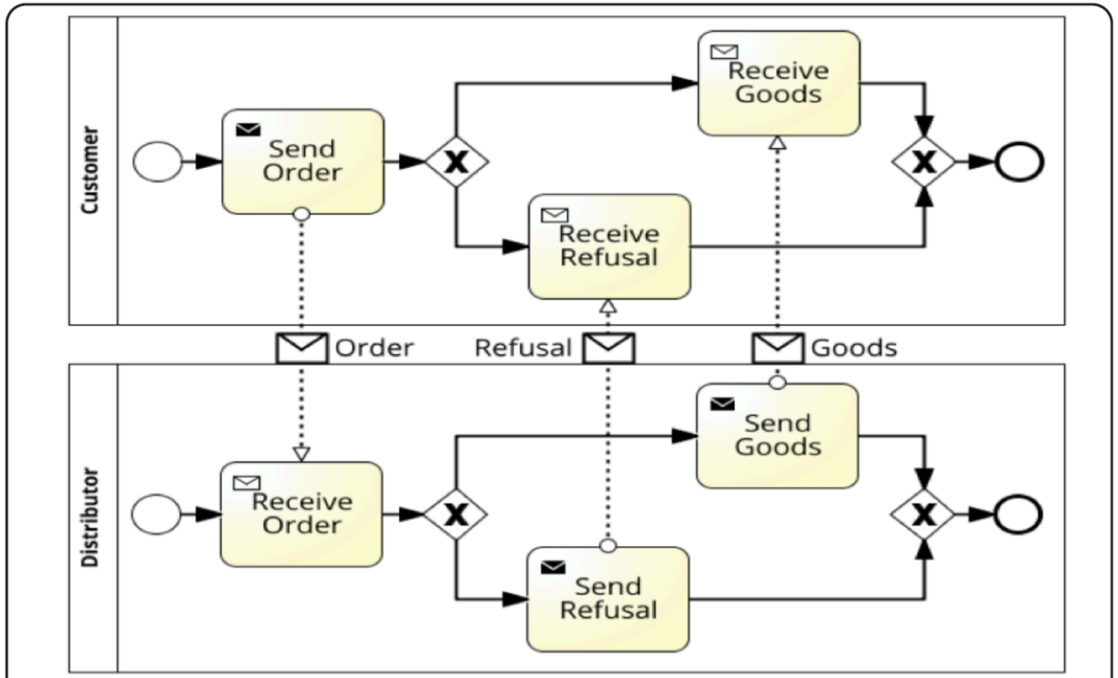




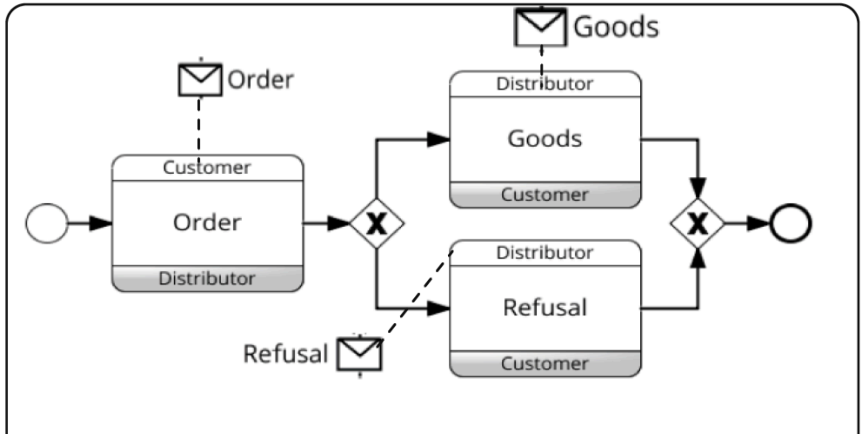
(a) Private Model



(b) Public Model

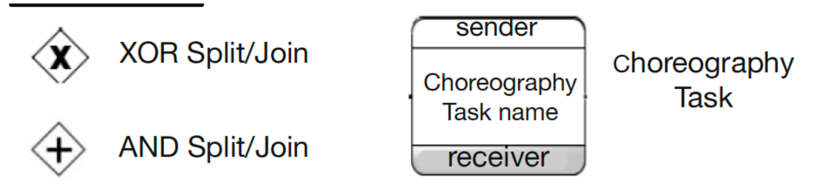


(c) Collaboration Model

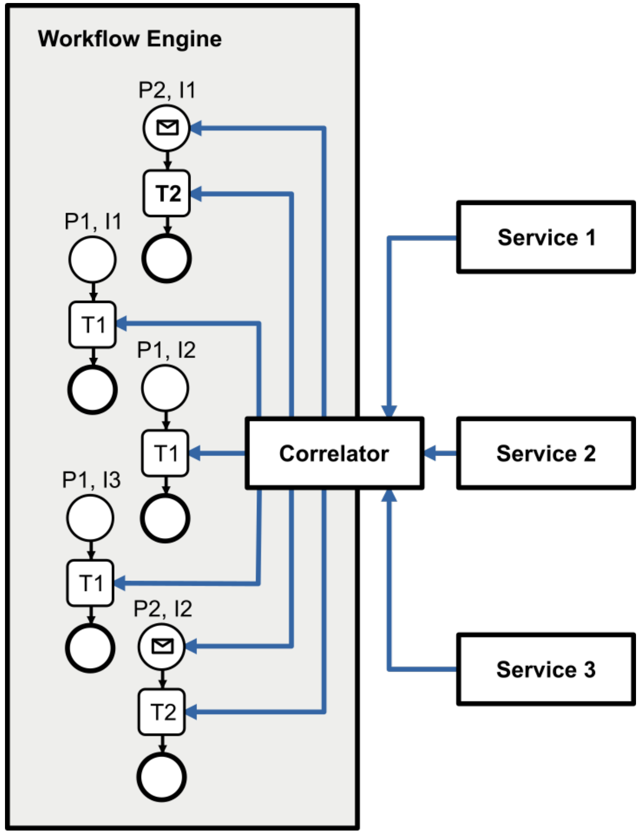


(d) Choreography Model

Legend



# Correlation

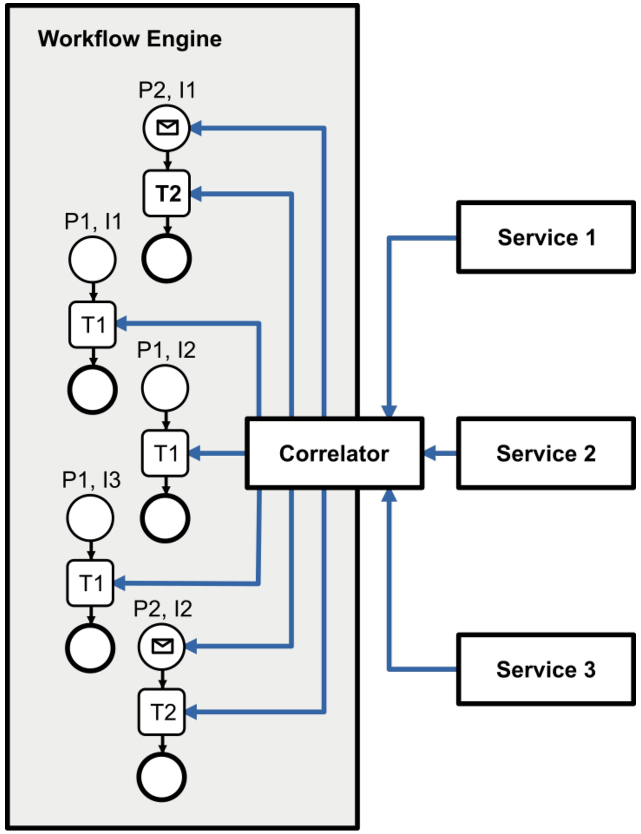


| | | |
|---|---|---|
| Px | ... | Process |
| Ix | ... | Instance |
| Tx | ... | Task |
| Service | ... | Implementation or other instance/engine |
| Correlator | ... | Match and distribute based on message content |

Patterns - Point of View:

- Ask for information: the process engine asks an external service for some results.
- React on some external event:
  - A new process instance has to be started.
  - Waiting for some external message.
    - External source may be unknown.
    - External source may send the message long before an instance is needing the value.
    - External source may send the value long after an instance is needing the value.
    - The contents of a message contain the key to decide what to do

# Correlation - How



| | | |
|---|---|---|
| Px | ... | Process |
| Ix | ... | Instance |
| Tx | ... | Task |
| Service | ... | Implementation or other instance/engine |
| Correlator | ... | Match and distribute based on message content |

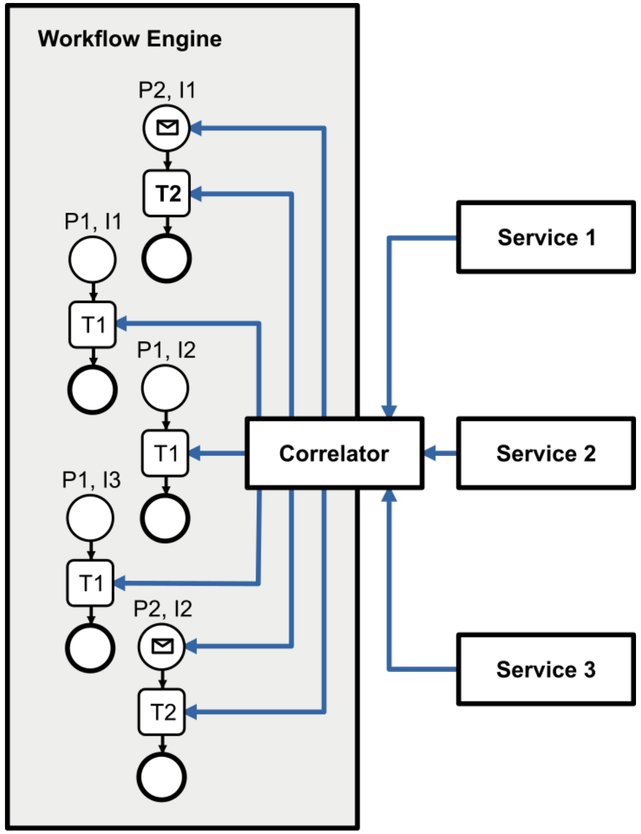
Correlator has to:

- Receive messages from arbitray sources
- Analyse the contents of the message. A message might be a mail received via IMAP, a word document received via HTML Form upload, or mor generic: any fileformat submitted through any means/protocol.
- Forward the message to a potential instance that is waiting for the message, OR
- Store the message for later use
  OR
- Create a new instance of a process, with the message as input

Correlator requires:

- A set of rules to analyse incoming messages
  - Message type, e.g. email
  - Where to look in message, e.g. subject contains ticket number
  - What to do with the message: forward or instantiate

# Correlation - Message Analysis



| | | |
|---|---|---|
| Px | ... | Process |
| Ix | ... | Instance |
| Tx | ... | Task |
| Service | ... | Implementation or other instance/engine |
| Correlator | ... | Match and distribute based on message content |

A correlator will support a set of sources: Mailbox, HTTP/REST file upload, ...

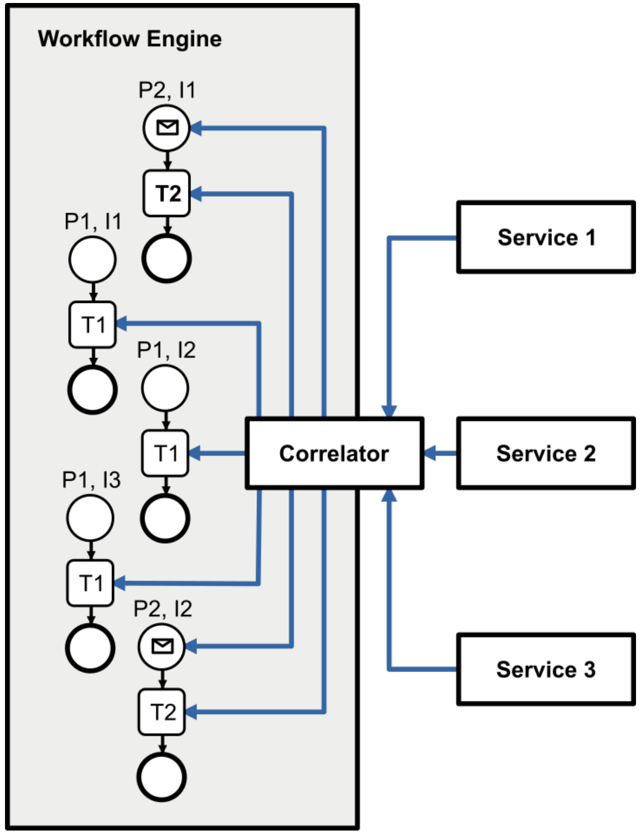
A correlator will support a set of tile types that can be analysed:

- A means to read and analyse the contents of a word file
- A means to read and analyse the contents of a excel file
- A means to read and analyse the contents of a CAD file
- ...

A correlator will support a rule language to describe how to extract correlation information

- A word files contanins the signature of a particular person at the end of page 7, contains a ticket number in the form /#\w{12}/ on page 2, and has a heading “Customer Contract” on page 1
- An excel sheet has three worksheets, and on the first worksheet cell A1 contains the identifier of the customer
- ...

# Correlation - Target Identification



| | | |
|---|---|---|
| Px | ... | Process |
| Ix | ... | Instance |
| Tx | ... | Task |
| Service | ... | Implementation or other instance/engine |
| Correlator | ... | Match and distribute based on message content |

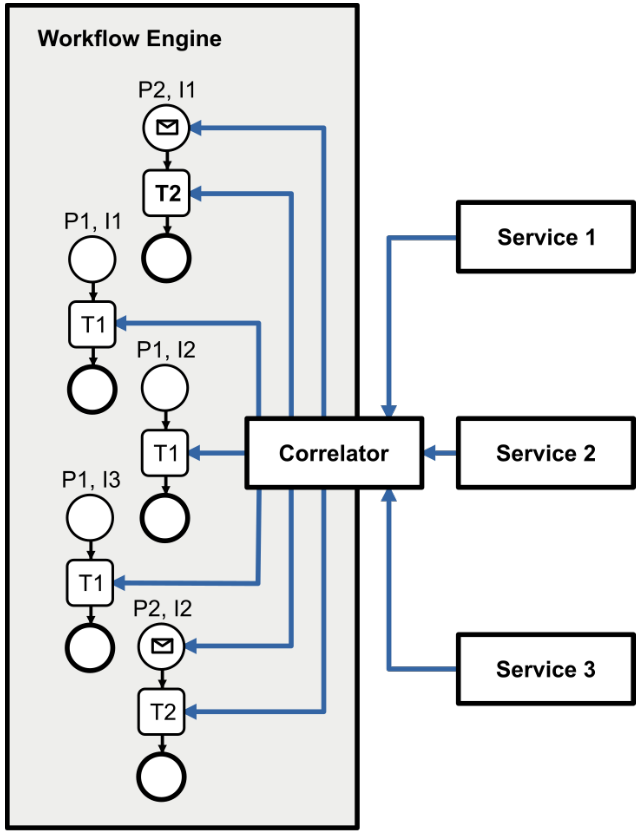
Processes that are to be instantiated have to contain **correlation rules** when to do so:

- A word files contanins the signature of a particular person at the end of page 7, contains a ticket number in the form /#\w{12}/ on page 2, and has a heading “Customer Contract” on page 1
- The rule is generic, any match leads to an instantiation
- The message is an input to the resulting process instance, see P2 on the left
- The correlator has to analyse each process model to extract the correlation rules

Instances that require certain message (but no direct means to contact a source) have to make a **correlation request** to get the required information from the correlator:

- I need a word files contanins the signature of John Boss at the end of page 7, contains a ticket number in the form #QWER12tzui34 on page 2, and has a heading “Customer Contract” on page 1
- The request is very special, this one message matches
- The correlator can store a list of requests on the fly, and delete entries whenever a request has fullfilled

# Correlation - Message Received



| | | |
|---|---|---|
| Px | ... | Process |
| Ix | ... | Instance |
| Tx | ... | Task |
| Service | ... | Implementation or other instance/engine |
| Correlator | ... | Match and distribute based on message content |

A matching correlation rule exists:

- instantiate process

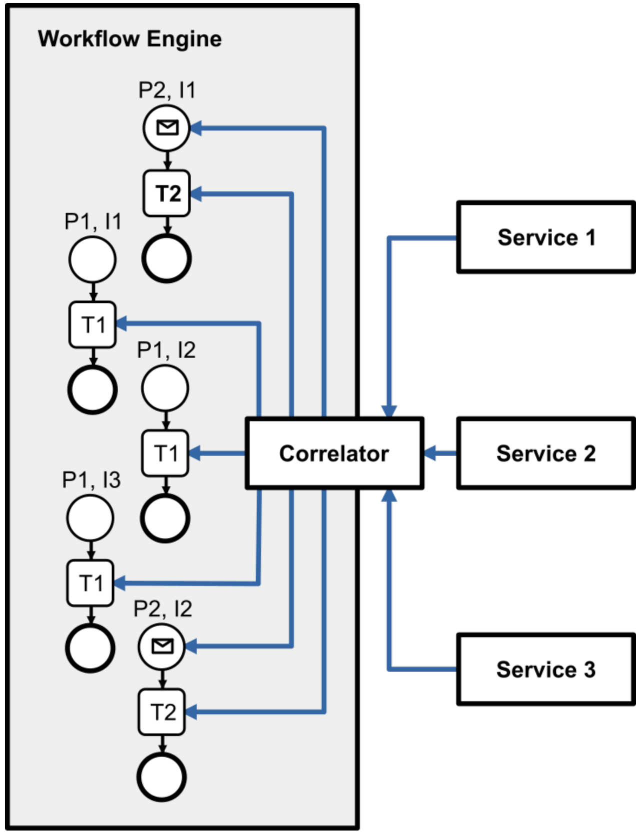
A matching correlation request exists:

- forward message
- Data Retention (next slide)

No matches exists:

- Data Retention (next slide)

# Correlation - Data Retention



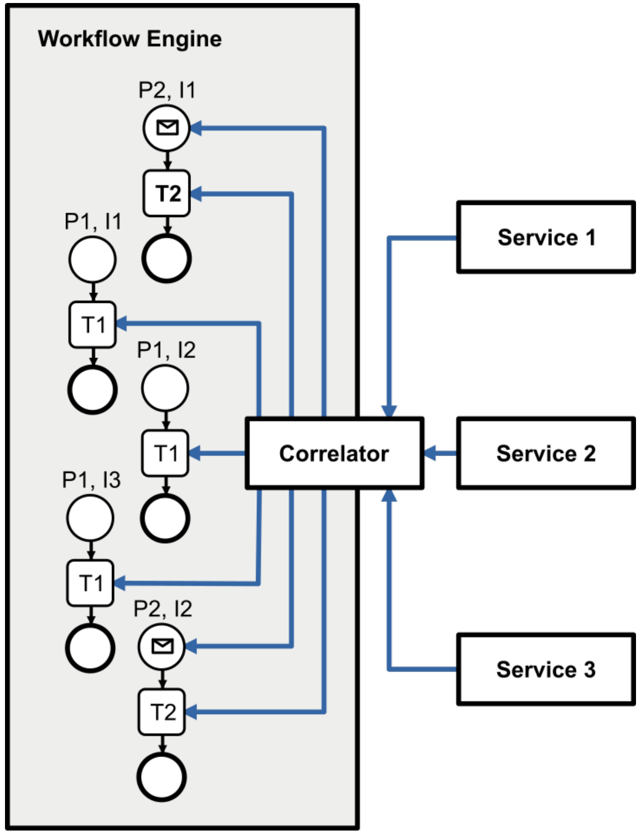
| | | |
|---|---|---|
| Px | ... | Process |
| Ix | ... | Instance |
| Tx | ... | Task |
| Service | ... | Implementation or other instance/engine |
| Correlator | ... | Match and distribute based on message content |

A message that instantiates a process: drop message after instantiation

A message that matches a correlation request - a retention policy has to exists:

- No policy: drop the message after forward
- Store indefinitely
  - Other requests can be answered
- Store for a limited time
  - Other requests can be answered
- Store with additional request rules
  - Additional parameters have to be fulfilled in order to reuse the message to fulfill request
  - E.g. a requestor has to submit a code to get the info
- Queue vs. Slot
  - Slot - Messages can replace other existing messages - only the newest message fitting a correlation request is stored
  - Queue - All messages are stored - the newest message fitting a correlation request is delivered

# Correlation - Application 1



| | | |
|---|---|---|
| Px | ... | Process |
| Ix | ... | Instance |
| Tx | ... | Task |
| Service | ... | Implementation or other instance/engine |
| Correlator | ... | Match and distribute based on message content |

**Security:** correlators connect external systems to internal process instances without exposing the internal structure.

**Loose Coupling:** external systems have a single point where to send information.

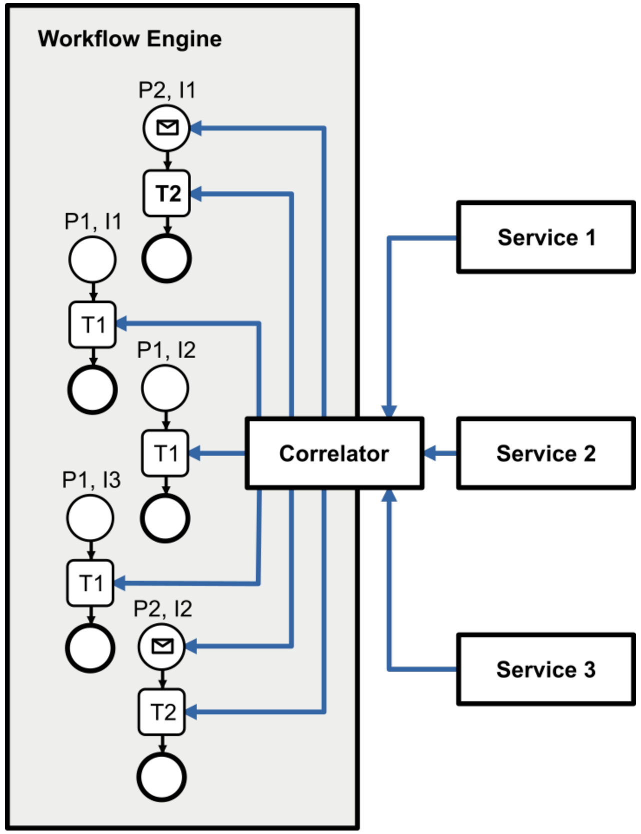
**Policy Enforcing:** how to deal with message retention - loose coupling & maintainability.

**Compliance Checking:** incoming data can be checked before it enters internal infrastructure

**Maintainabilty:** single place to deal with incoming messages. No need to implement anything in the engine.

The alternative would be much more complicated and relying on tighly coupled system.

# Correlation - Application 2



| | | |
|---|---|---|
| Px | ... | Process |
| Ix | ... | Instance |
| Tx | ... | Task |
| Service | ... | Implementation or other instance/engine |
| Correlator | ... | Match and distribute based on message content |

Services (see left) can be other Worflow Engines or BPM systems:

**Correlation is imperative for Choreographies**

# Summary

- Interorganizational processes are prevalent in many application domains, e.g., logistics, health care.
- In this chapter, we looked at top-down design of process choreographies, i.e., designing them on from scratch.
- In reality, often there are already processes running at the partners‘ side, requiring often lengthy and costly negotiations.
- Further challenges:
  - Change → W. Fdhila, C. Indiono, S. Rinderle-Ma, M. Reichert: Dealing with change in process choreographies: Design and implementation of propagation algorithms. Inf. Syst. 49: 1-24 (2015), https://doi.org/10.1016/j.is.2014.10.004
  - Compliance → W. Fdhila, D. Knuplesch, S. Rinderle-Ma, M. Reichert: Verifying Compliance in Process Choreographies: Foundations, Algorithms, and Implementation. Information Systems 2022, https://doi.org/10.1016/j.is.2022.101983