Process Choreographies

Presenter: Ayça Avcı

Content

- Motivation & Terminology
- Development Phases
- Process Choreography Design
  - High-Level Design
  - Collaboration scenarios
  - Compatibility
- Process Choreography Implementation
  - Consistency criterion: Public-to-Private Approach
- Service Interaction Patterns
- Let’s Dance

Motivation & Terminology

- Business-to-business collaborations are quite complex.
- The cooperation between companies should be designed very carefully.
- Process choreographies can be used for this purpose.

The requirements of process choreography development depend on:

1. The number of interacting partners.
2. The desired level of automation.

Bidding Scenario

Consider a bidding scenario:

The owner of a car uses an auctioning service to sell his car to the highest bidder.
Potentially, thousands of people can participate in the auction and place their bids.

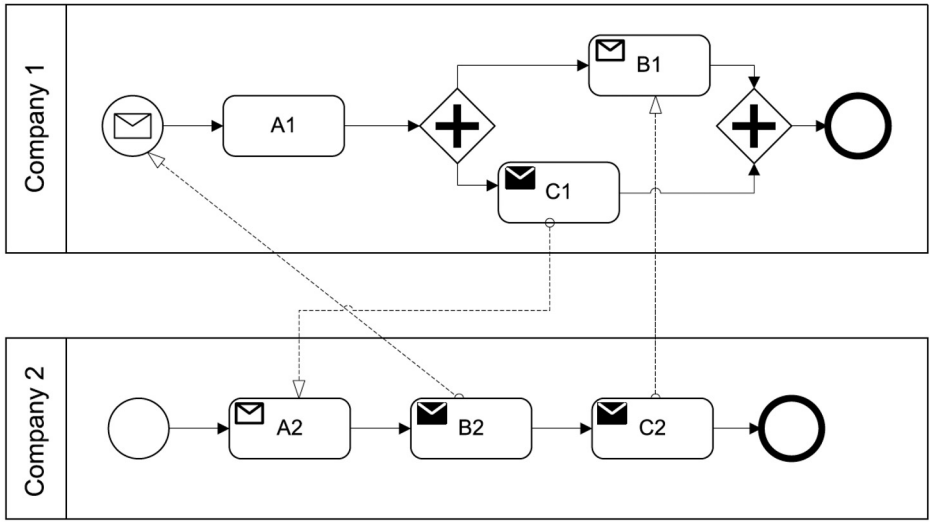
Such scenarios require agreement on **how the participants need to interact with each other to avoid problems that could appear as the result of wrong interaction.**

Deadlock Situation [1]

- Process of Company 1 waits for B2 from Company 2 to be initiated.
- B2 can be only performed when A2 is completed.
- However, A2 waits for C1 from Company 1.

As a result, both process orchestrations cannot proceed.

DEADLOCK!



Development Phases

Goal:

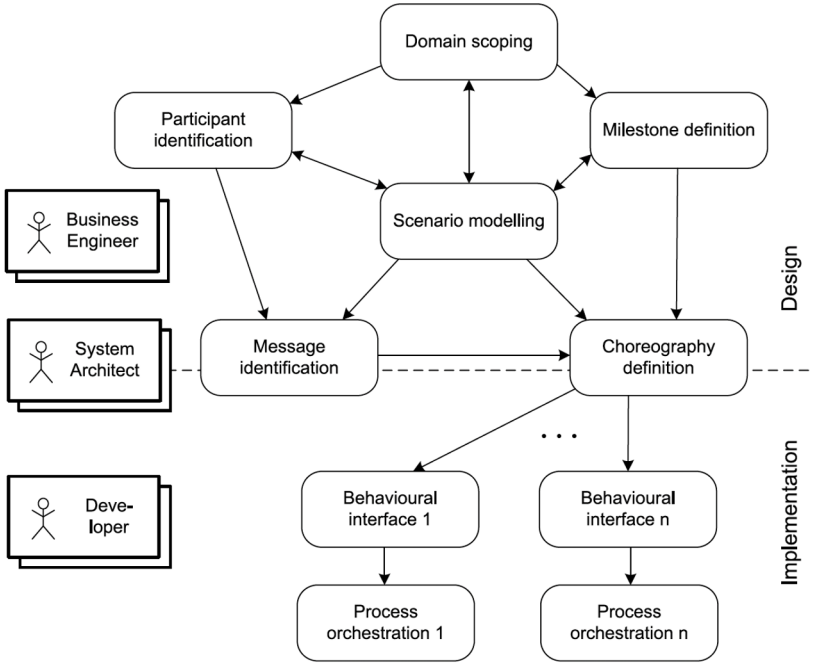
To provide an understanding of the concepts and artefacts involved in the design of process choreographies.

Phases during choreography design & implementation [1]

Business Engineer: Involved in choreography design phases, scenario modelling, domain scoping, milestone definition, and participant identification (business related aspects).

System Architects: Responsible for the architectural aspects of the implemented process choreography. In particular, involved in specification of the behavioural interfaces.

Developer: Responsible for realizing the process orchestrations.



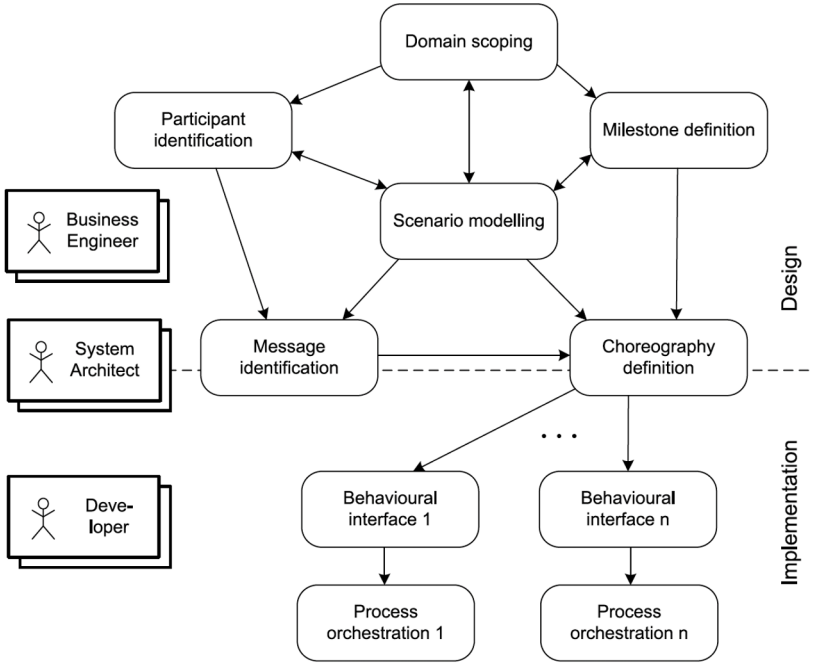
Phases during choreography design & implementation [1]

Scenario modelling: Describes the overall setting and goals of the process choreography.

Domain scoping: A domain in which the cooperation will take place needs to be specified.

Participant identification: Defining different roles of choreography participants (refers to organizations, not individuals).

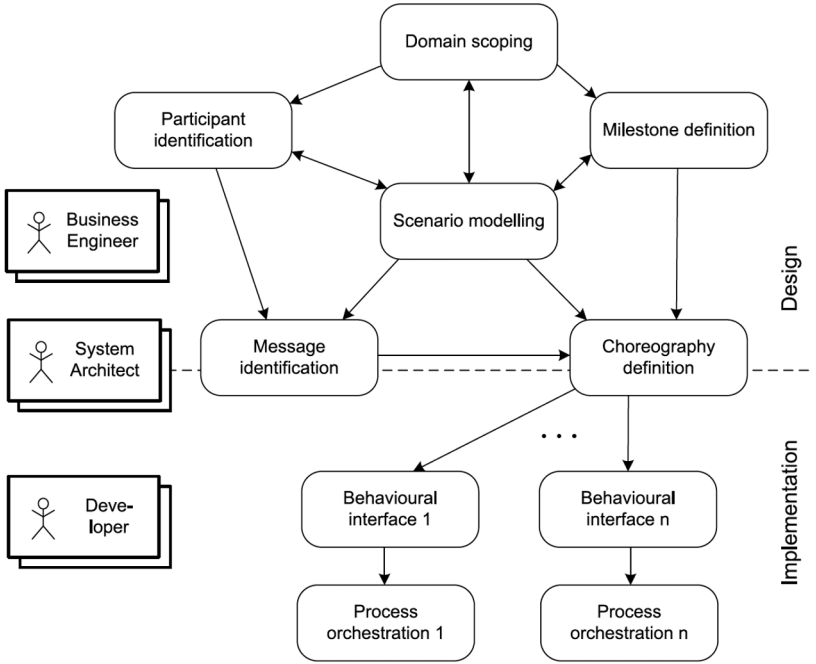
Milestone definition: Certain states of the choreography in which the cooperation has achieved certain results.



Phases during choreography design & implementation [1]

Message identification: Design messages that realize the various interactions.

Choreography definition: Detailed specification of the interactions between the participants, the messages to realize the interactions, and the milestones that are reached during the resulting conversation.



Process Choreography Design

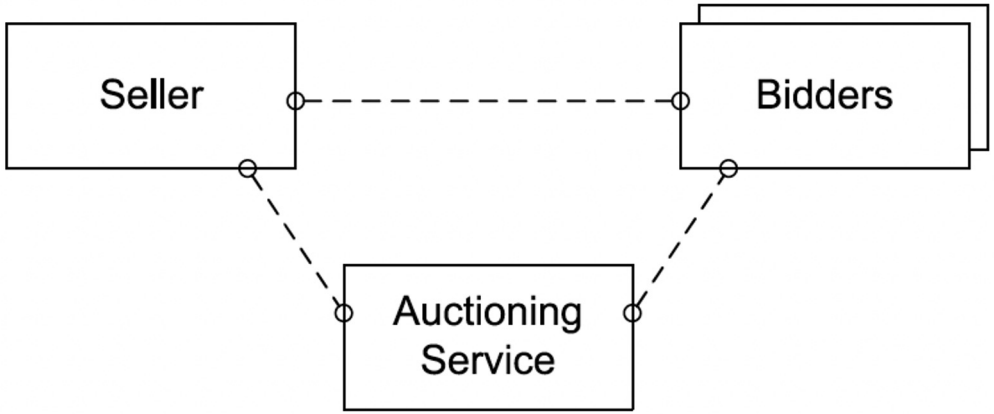
1. **High-level Structure Design:** The participant roles and their communication structures are identified.

1. **High-level Behavioural Design:** Specify the milestones of the collaboration and the order in which the milestones are reached.

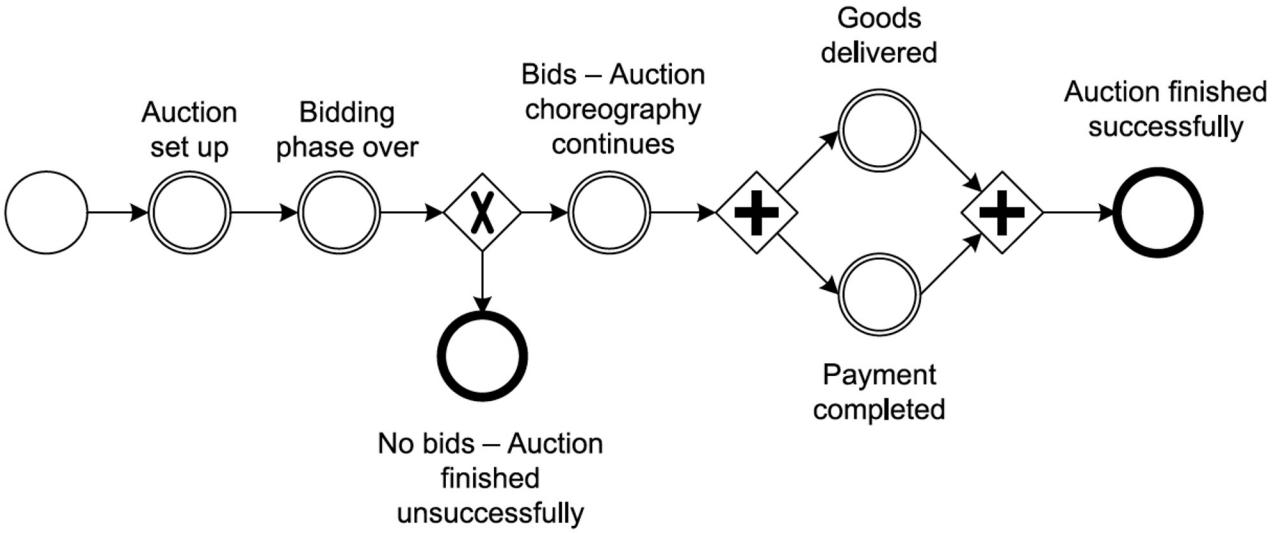
1. **Collaboration Scenarios:** High-level choreographies are introduced with dedicated collaboration scenarios that relate the reaching of milestones to the communication between process participants.

1. **Behavioural Interfaces:** From the collaboration scenarios, for each participant role, a behavioural interface is derived.

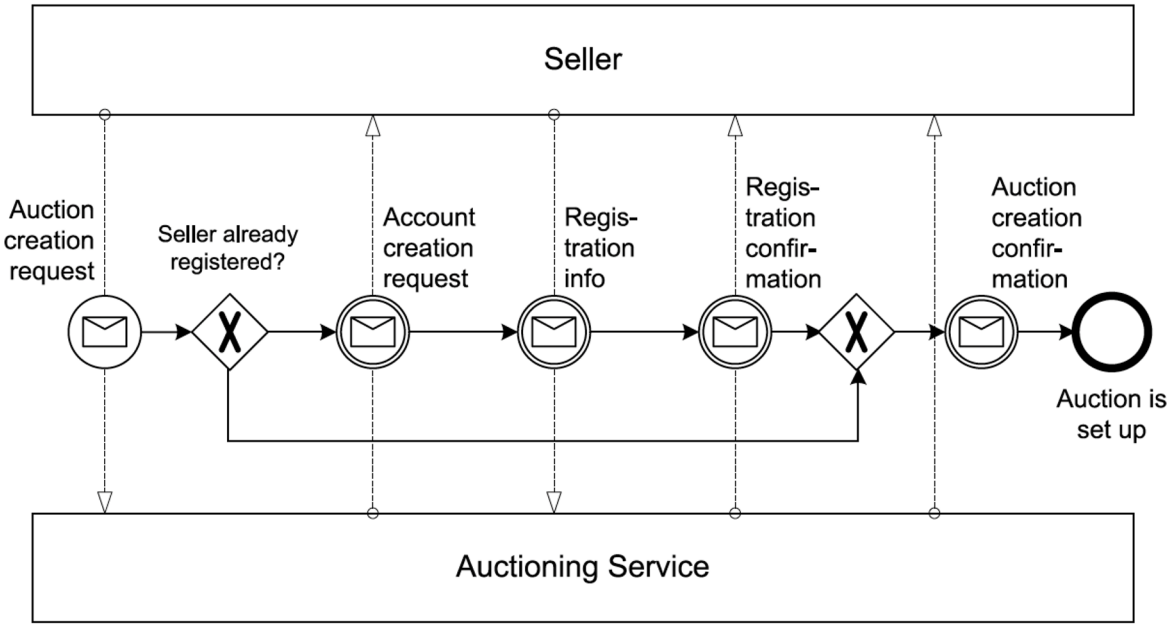
High-Level Structure Design [1]



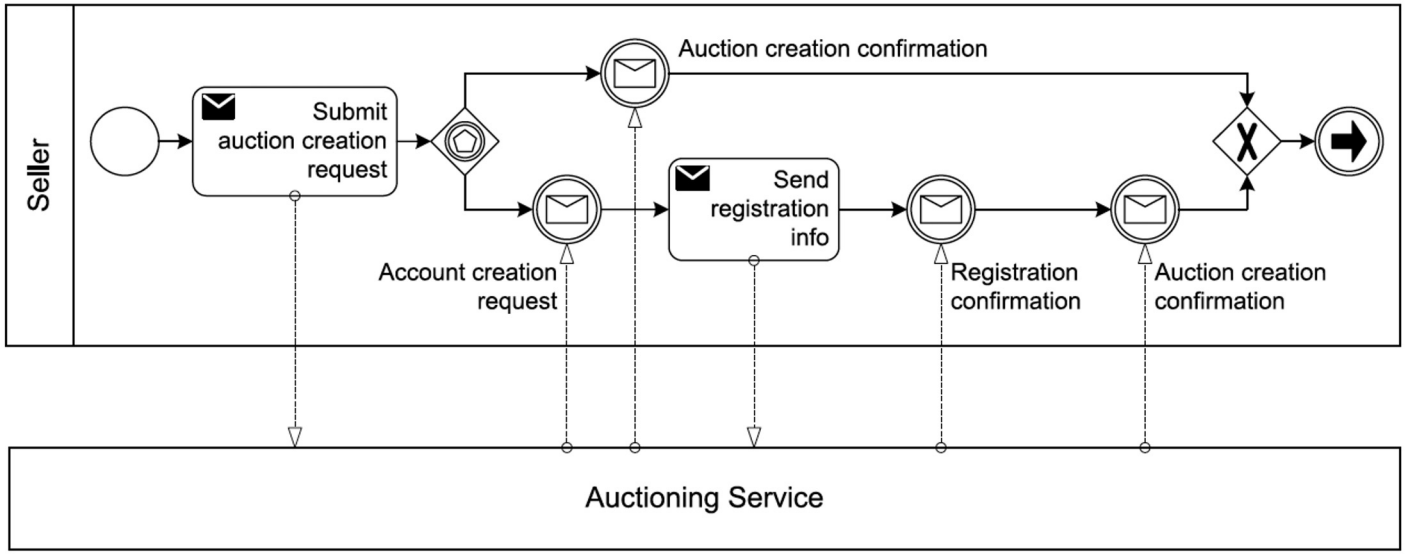
High-Level Behavioral Design [1]



Collaboration Scenario [1]



Behavioural Interface [1]



Compatibility

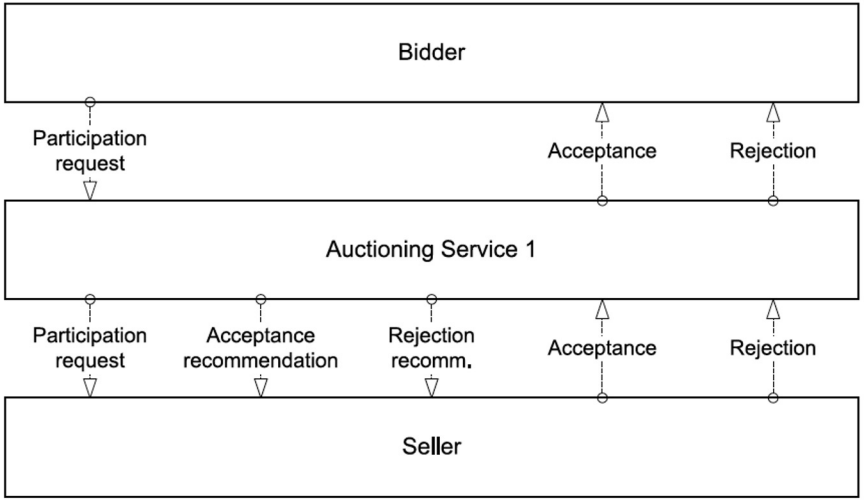
Process choreography design needs to ensure that the process orchestrations of the participants play together well in the overall collaboration.

Compatibility is the ability of a **set of participants to interact successfully according to a given process choreography.**

- **Structural compatibility**
  - **Strong structural compatibility**
  - **Weak structural compatibility**
- **Behavioural compatibility**

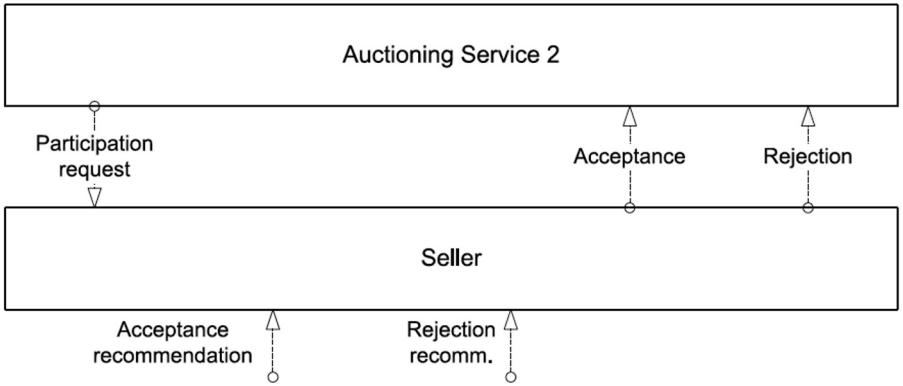
Structural compatibility [1]

Strong structural compatibility: For every message that can be sent there is a participant who can receive it, and if for every message that can be received, there is a participant who can send it.



Structural compatibility [1]

Weak structural compatibility: If all messages sent by participants can be received by other participants.

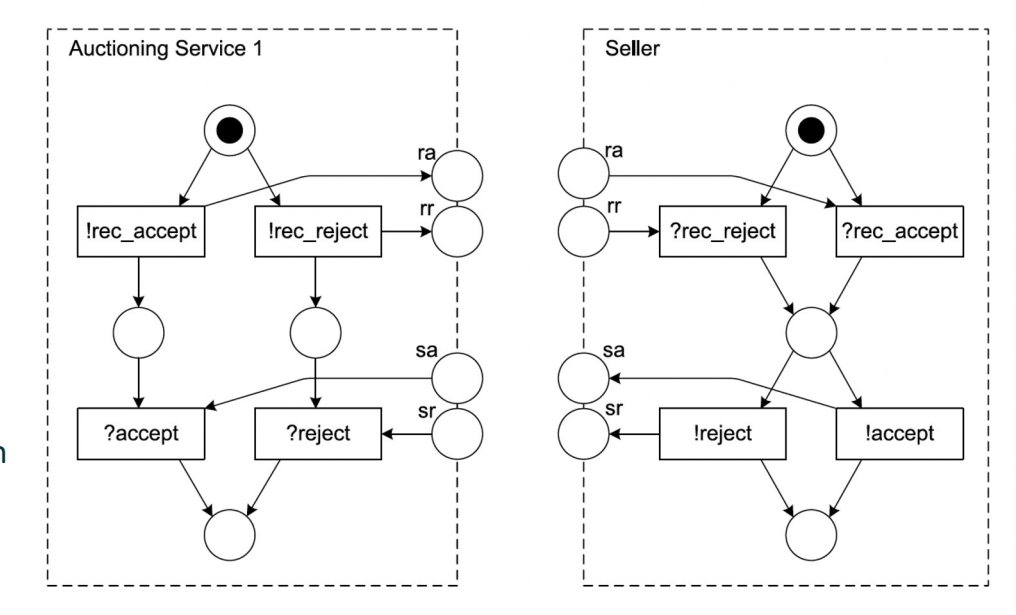


Behavioural compatibility [1]

Workflow modules are not workflow nets!

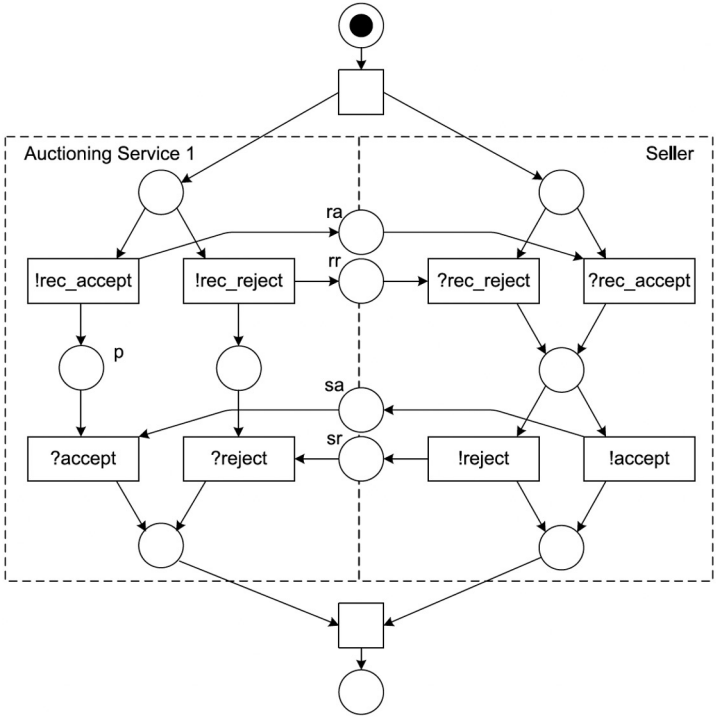
In workflow nets, each place and each transition is on a path from the initial node to the final node. Thus, only the initial place can have no incoming edge and only the final place can have no outgoing edge.

In workflow modules this is not true, because communication places by definition have either **no incoming edges (places for receiving messages)** or **no outgoing edges (places for sending messages).**

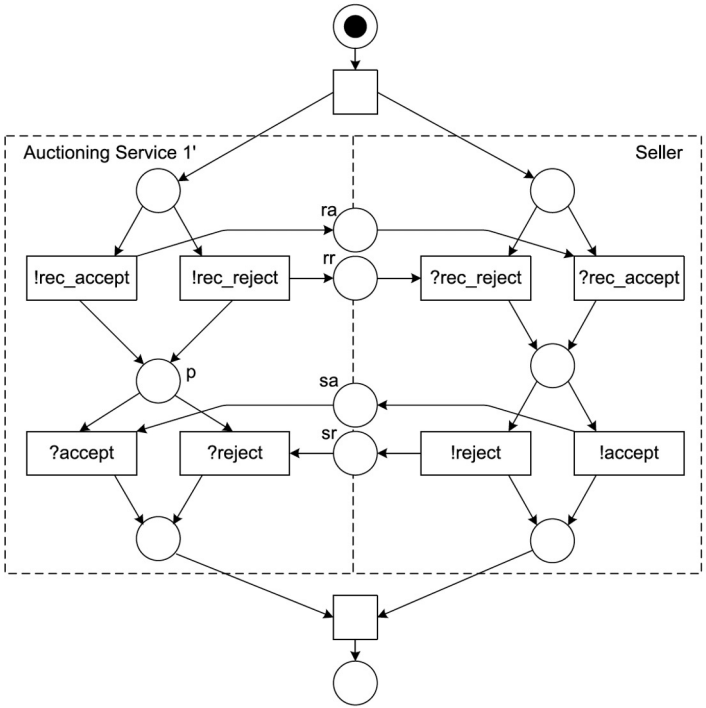


Behavioural compatibility [1]

- This Petri net is a workflow net since there is **one dedicated initial place and one final place, and each node is on a path from the initial place to the final place.**
- But, the composition of workflow modules is not satisfactory.
- Consider a process instance in which the auctioning service recommends accepting the bidder, while the seller decides to reject the bidder.

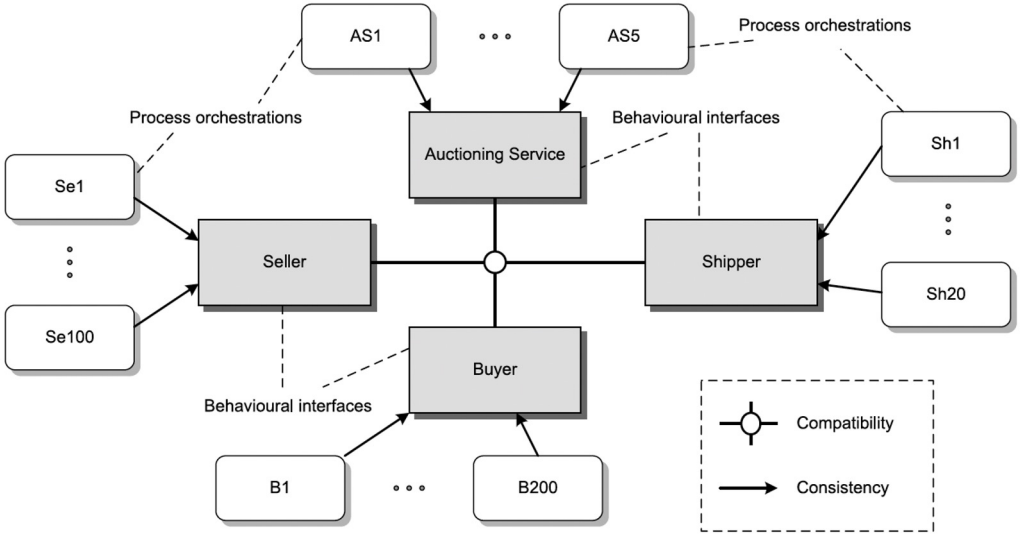


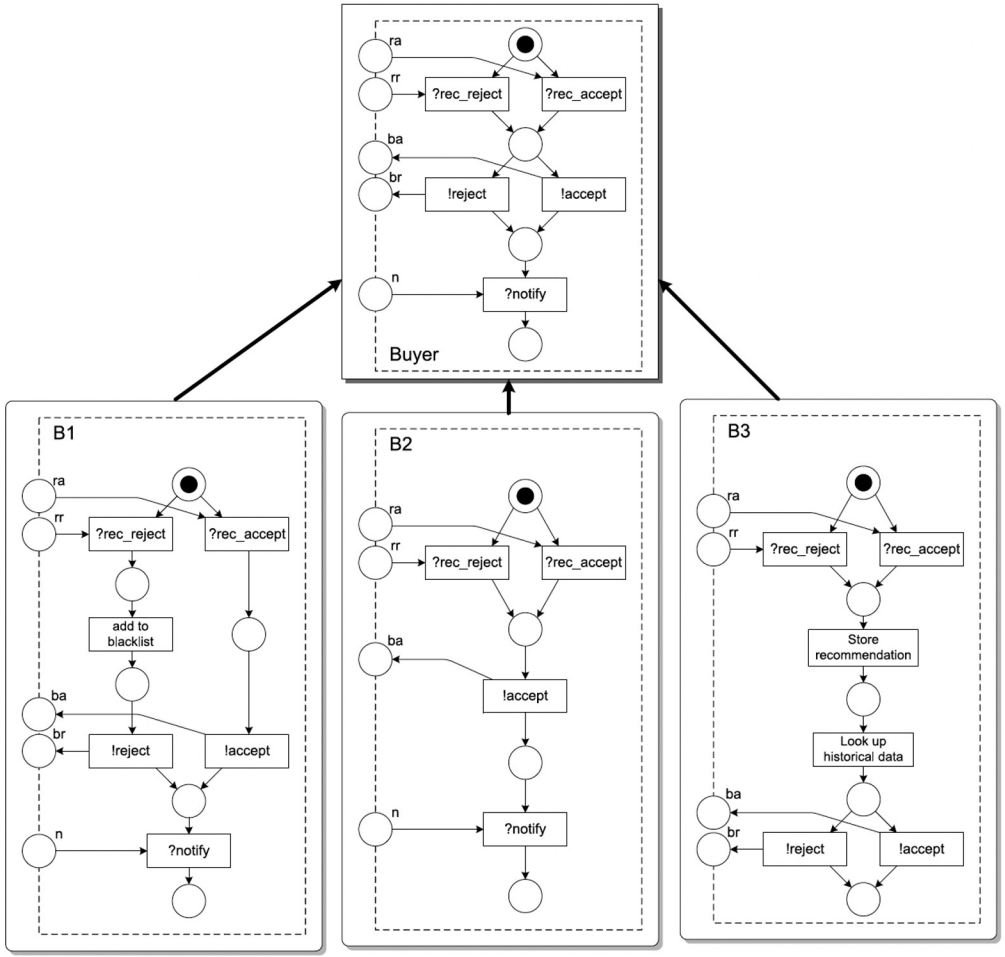
Behavioural compatibility [1]



Process Choreography Implementation [1]

- Each participant role is specified by a set of its behavioural interfaces.
- They need to be compatible with each other, so that the collaboration can be successful.




Buyer
ra
rr
?rec_reject
?rec_accept
ba
br
!reject
!accept
n
?notify
B1
add to blacklist
B2
B3
Store recommendation
Look up historical data

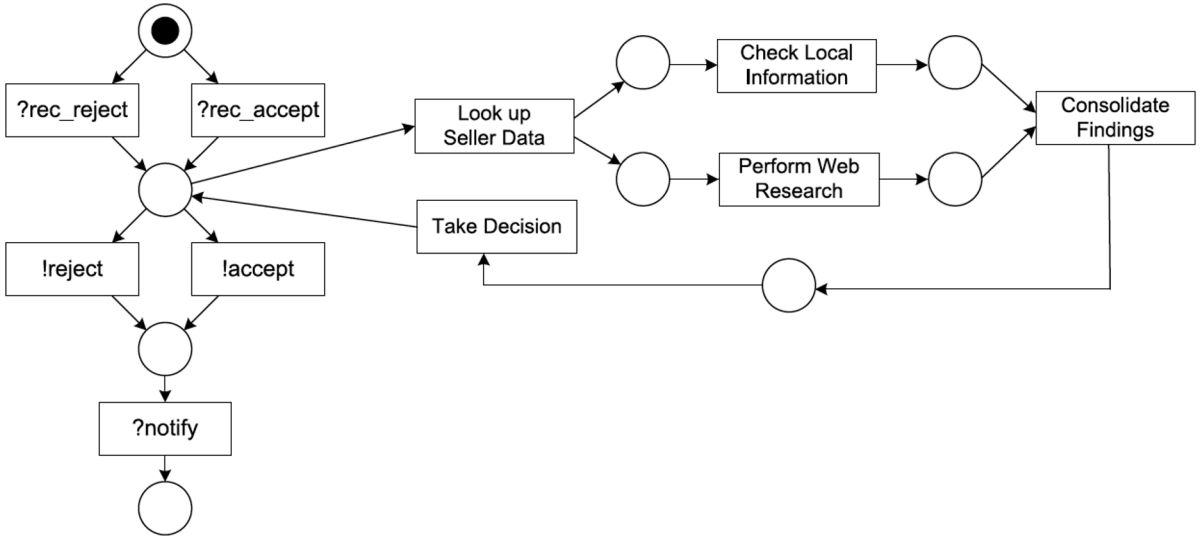
Consistency Criterion: Public-to-Private Approach

Loop: By adding a loop with start place and end place of the loop being exactly one place in the public process, the process can be transformed.

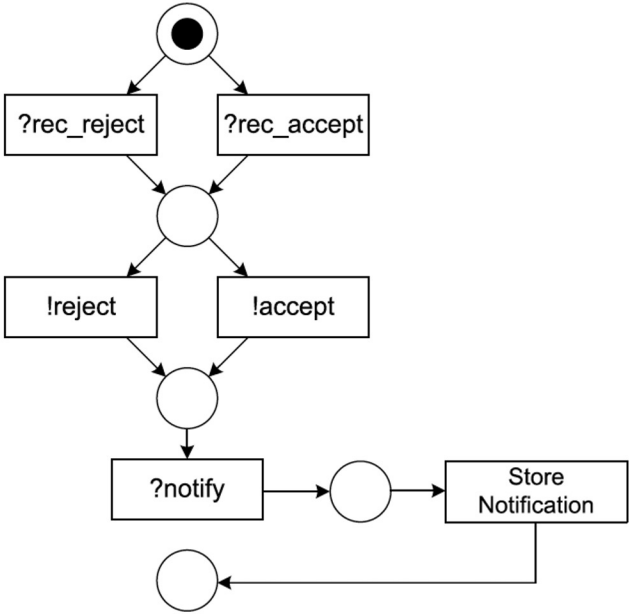
Detour: An edge in the Petri net can be substituted by a subnet, which implements a detour of the original flow, defined in the public process.

Concur: A concurrent branch can be added by designing a subnet which is spawned concurrently to the original flow, later to be synchronized with the flow.

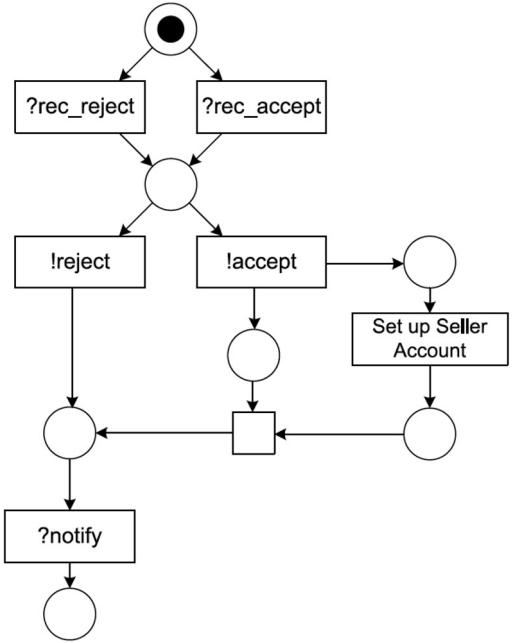
Loop transformation operation [1]



Detour transformation operation [1]

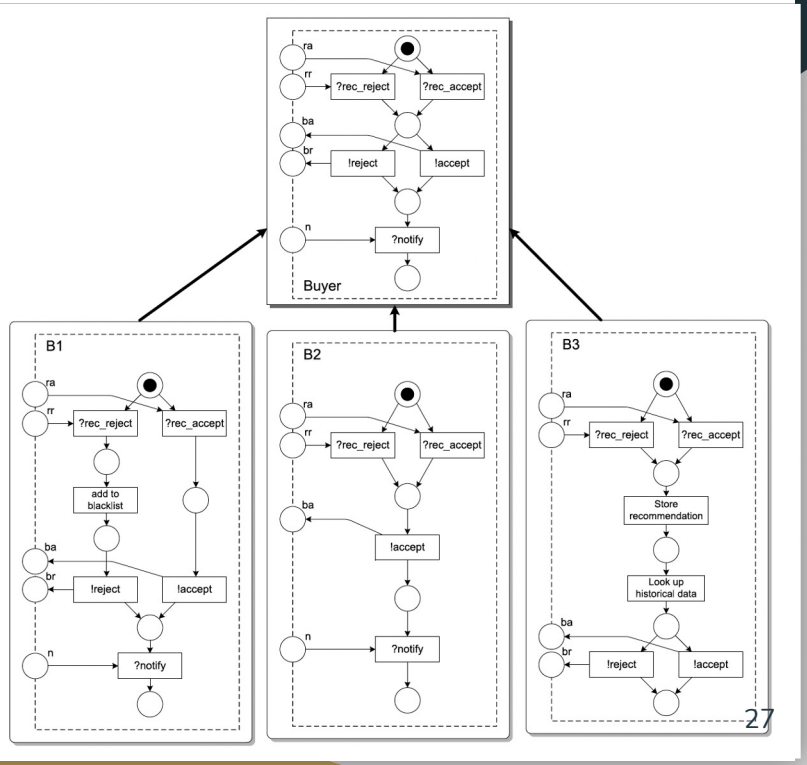


Concur transformation operation [1]

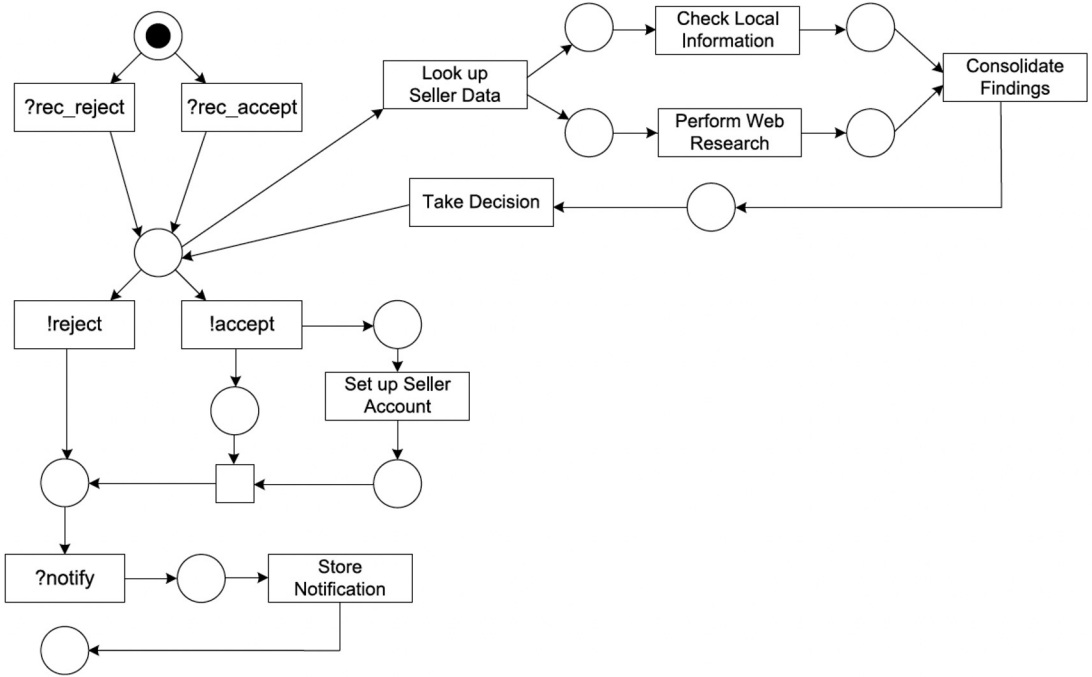


Is private process B1, B2 or B3 consistent with public process Buyer (B)?

- **B1:** No. B1 cannot send an acceptance message after receiving a rejection recommendation, although this is a legal behaviour of B.
- **B2:** No. B2 does not even have the same set of transitions that B has (B2 can never send a rejection message).
- **B3:** No. B3 cannot receive a notification message as B does.



Combined transformation operations [1]



Service Interaction Patterns

- Proposes small granular types of interactions that can be combined to process choreographies.
- Can also be used to benchmark languages for their ability to express advanced conversations as with control flow patterns for process orchestrations.

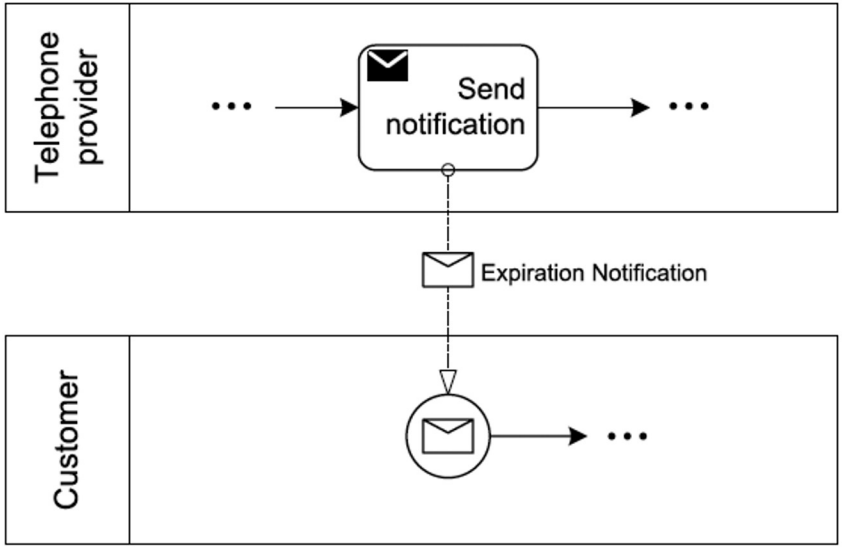
Includes following schemes:

- Number of participants involved
- Number of messages exchanged
- Variations in message receiver

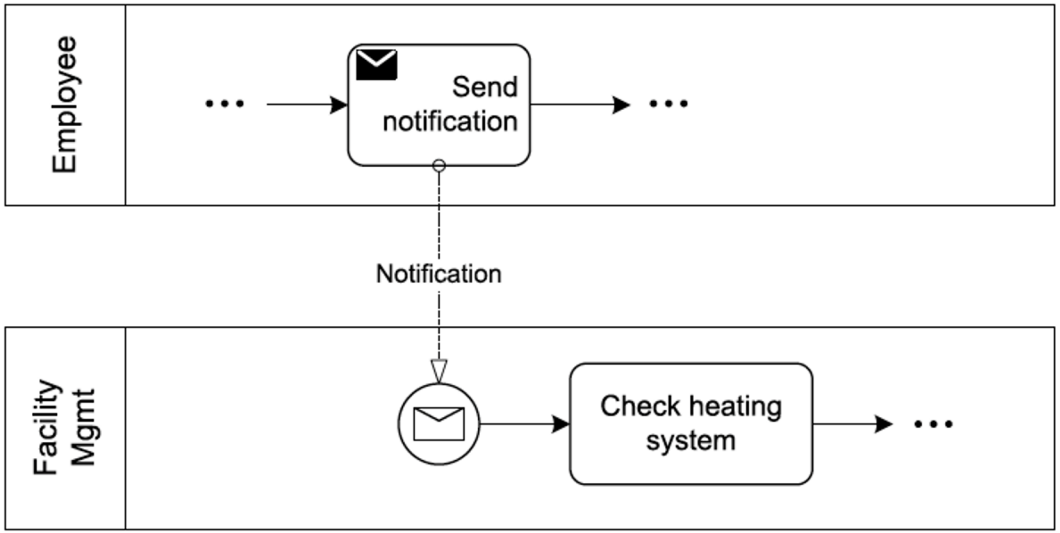
Service Interaction Patterns

- Send
- Receive
- Send/Receive
- One-to-many send
- Request with referral

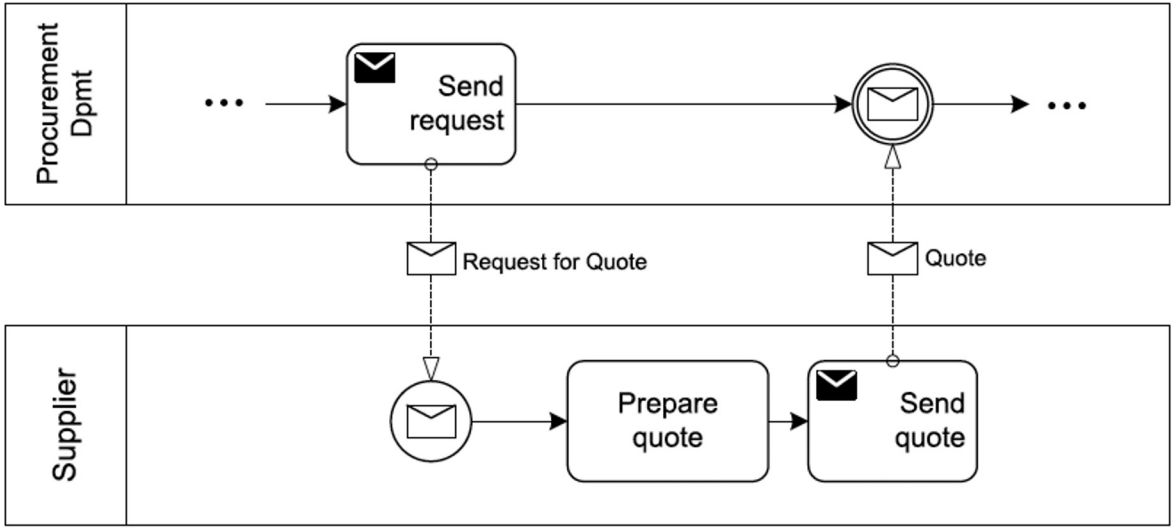
Send [1]


Telephone provider
Send notification
Expiration Notification
Customer

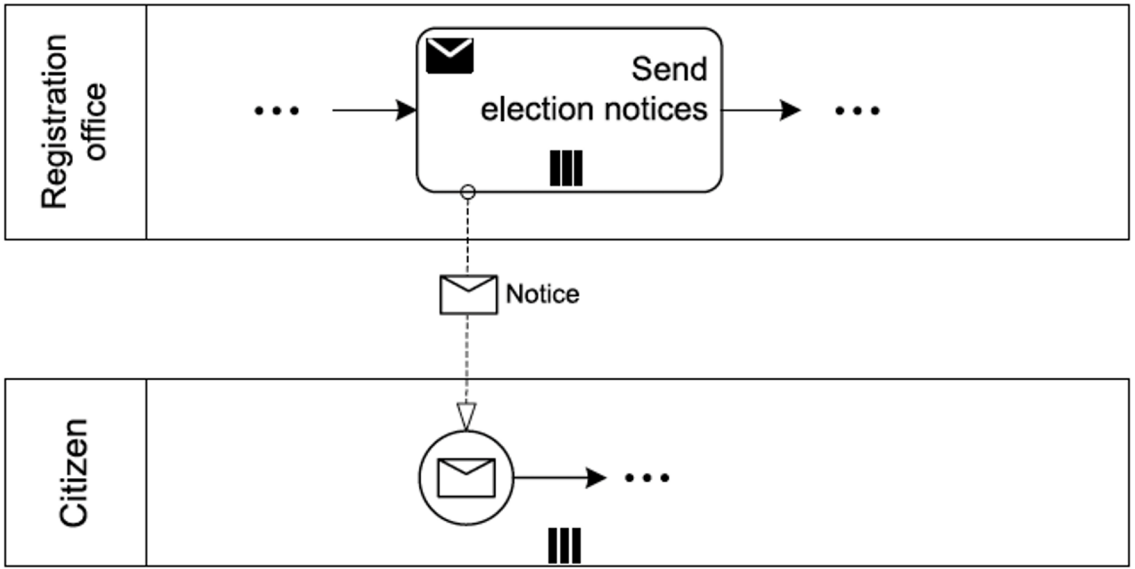
Receive [1]


Employee
Send notification
Notification
Facility Mgmt
Check heating system

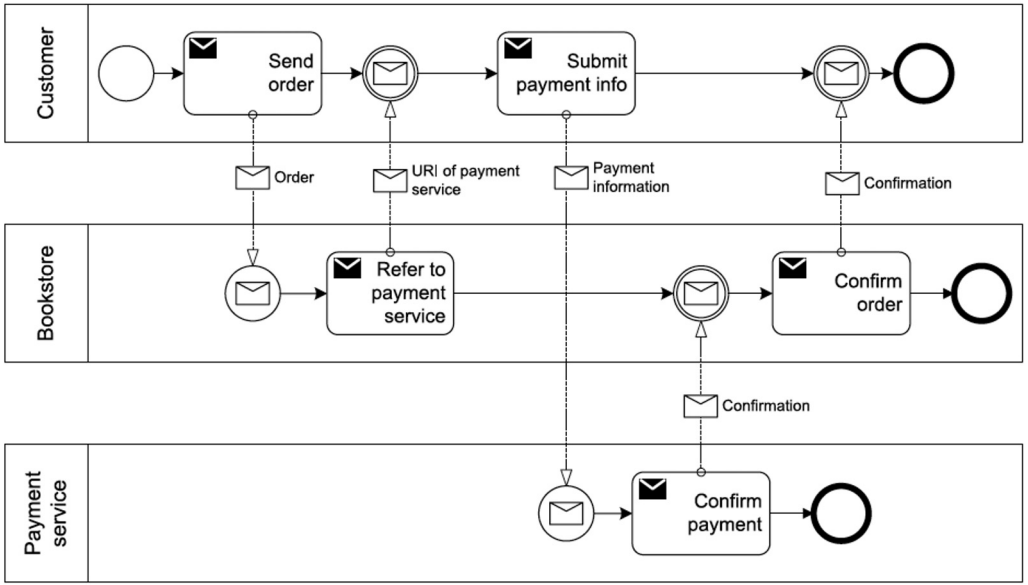
Send/Receive [1]



One-to-many send [1]

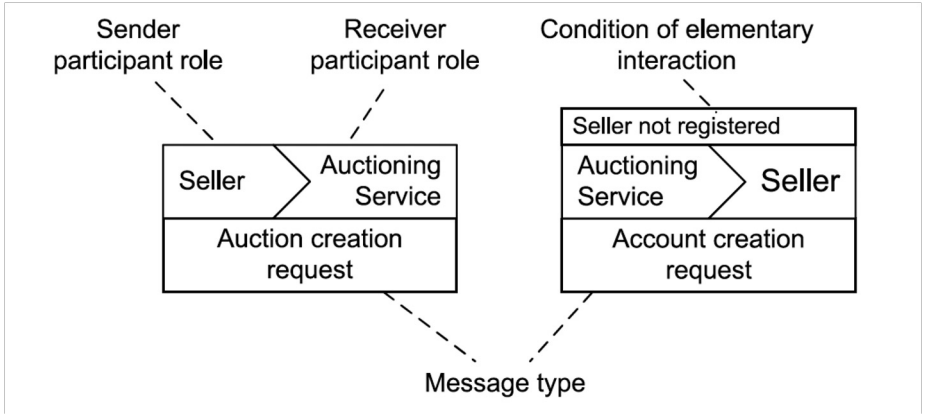


Request with referral [1]



Let’s Dance [1]

- Choreography language following interaction-centric approach.
- Based on control flow patterns and service interaction patterns.
- Control flow specification is the main focus, so the language abstracts from concrete message formats.



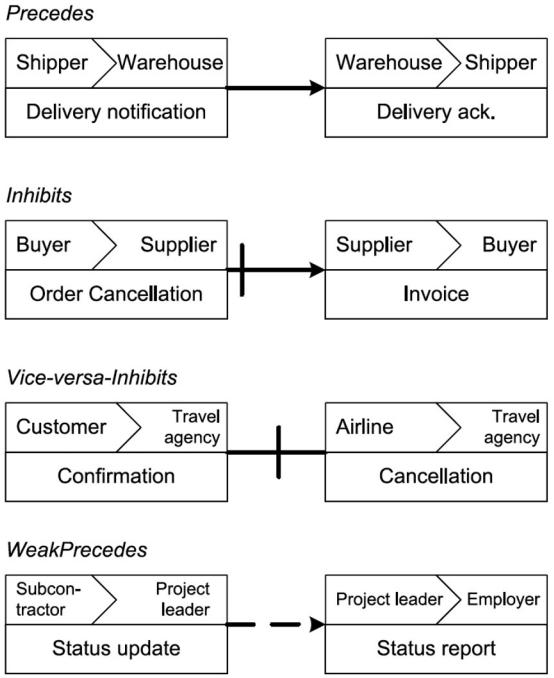
Basic control flow structures relating interactions [1]

Precedes: An instance of the target interaction can occur only if the instance of the source interaction has already occurred.

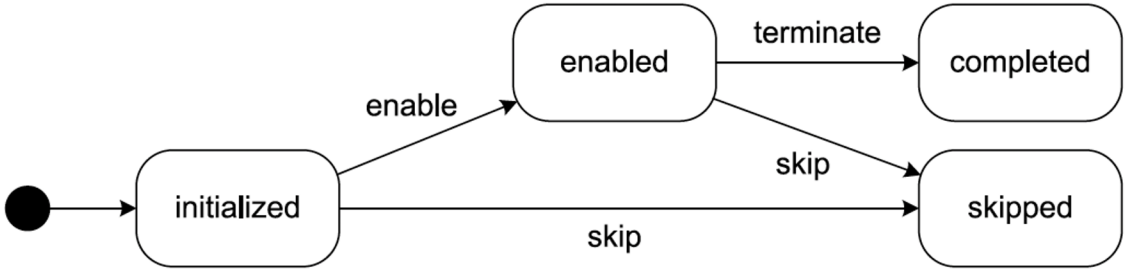
Inhibits: An instance of the target interaction can occur only if no instance of the source interaction has occurred yet.

Vice-versa-Inhibits: An instance where either one or the other interaction can complete.

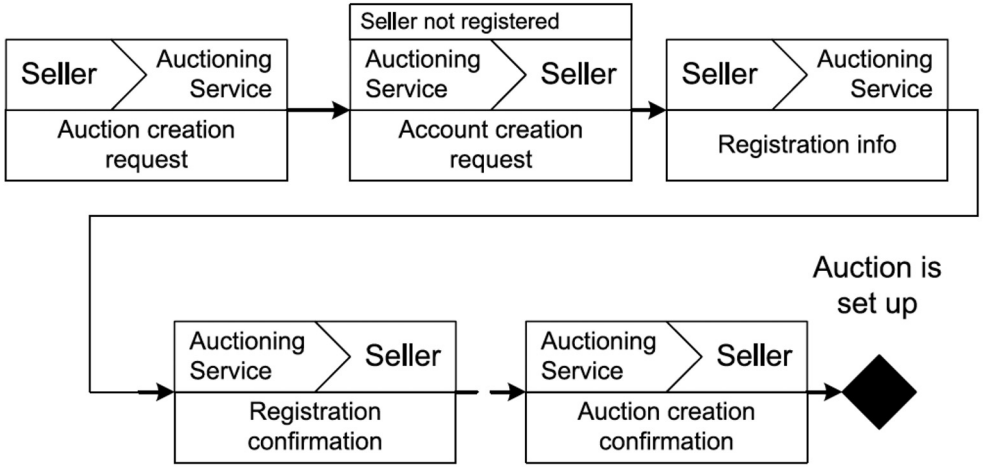
WeakPrecedes: An instance of the target interaction can occur only after the instance of the source interaction has already completed or was skipped.



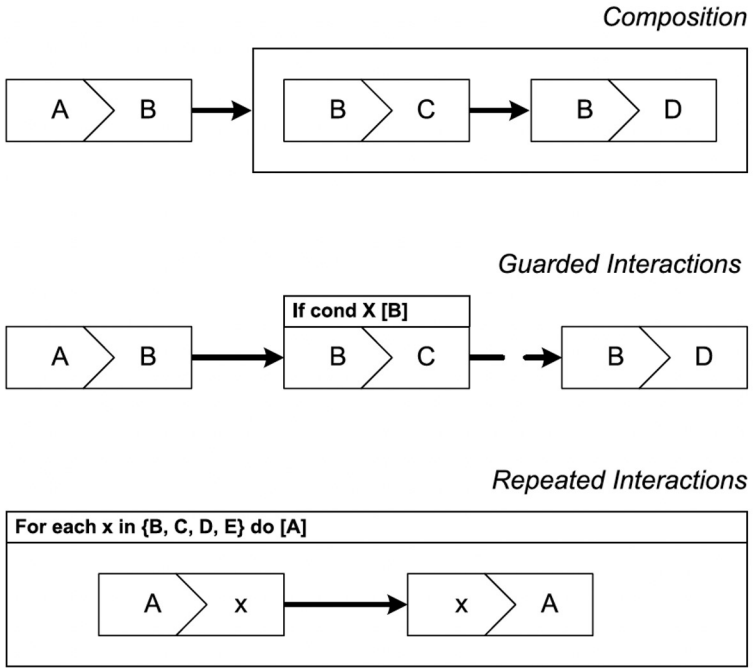
Lifecycle of interaction instances [1]



Interaction modelling [1]



Advanced control flow constructs [1]



References

[1] Weske, Mathias. ‘Process Choreographies’. Business Process Management: Concepts, Languages, Architectures. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. 243–291. Web.

**All the images that are used in the slides are taken from the above reference (Chapter 5).

Thank you!

Any questions?