A Methodology to Define QoS and SLA Requirements in Service Choreographies

Authors
Victoriano Alfonso Phocco Diaz
Daniel Macedo Batista

Institute of Mathematics and Statistics
Departament of Computer Science
University of Sao Paulo

alfonso7@ime.usp.br, batista@ime.usp.br

September 17, 2012

Agenda

SOC

SOC (Service Oriented Computing)

It is a new computing paradigm that utilizes services as the basic constructs to support the development of rapid, low-cost and easy composition of distributed applications even in heterogeneous environments. [Papazoglou et al., 2006].

Key elements:

- Services (mainly Web services).

SOC

SOC (Service Oriented Computing)

It is a new computing paradigm that utilizes services as the basic constructs to support the development of rapid, low-cost and easy composition of distributed applications even in heterogeneous environments. [Papazoglou et al., 2006].

Key elements:

- Services (mainly Web services).
- SOA.

SOC

SOC (Service Oriented Computing)

It is a new computing paradigm that utilizes services as the basic constructs to support the development of rapid, low-cost and easy composition of distributed applications even in heterogeneous environments. [Papazoglou et al., 2006].

Key elements:

- Services (mainly Web services).
- SOA.
- Service Composition.

SOC

SOC (Service Oriented Computing)

It is a new computing paradigm that utilizes services as the basic constructs to support the development of rapid, low-cost and easy composition of distributed applications even in heterogeneous environments. [Papazoglou et al., 2006].

Key elements:

- Services (mainly Web services).
- SOA.
- Service Composition.
  - **Service Orchestration**.

SOC

SOC (Service Oriented Computing)

It is a new computing paradigm that utilizes services as the basic constructs to support the development of rapid, low-cost and easy composition of distributed applications even in heterogeneous environments. [Papazoglou et al., 2006].

Key elements:

- Services (mainly Web services).
- SOA.
- Service Composition.
  - **Service Orchestration**.
  - **Service Choreography**.

SOC

SOC (Service Oriented Computing)

It is a new computing paradigm that utilizes services as the basic constructs to support the development of rapid, low-cost and easy composition of distributed applications even in heterogeneous environments. [Papazoglou et al., 2006].

Key elements:

- Services (mainly Web services).
- SOA.
- Service Composition.
  - **Service Orchestration**.
  - **Service Choreography**.
- ...

Web Services and SOA

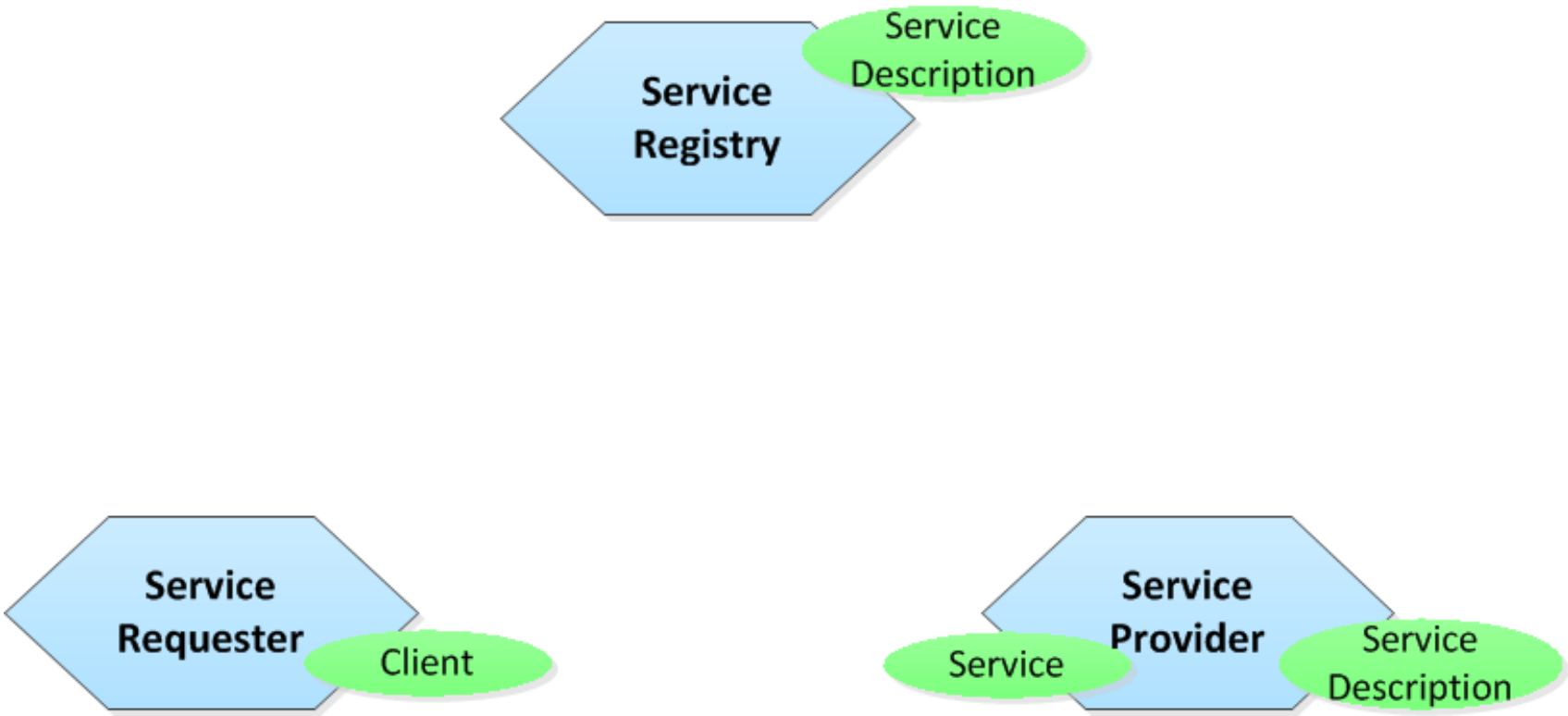


Figure: SOA triangle (based on [W3C, 2002])

Web Services and SOA

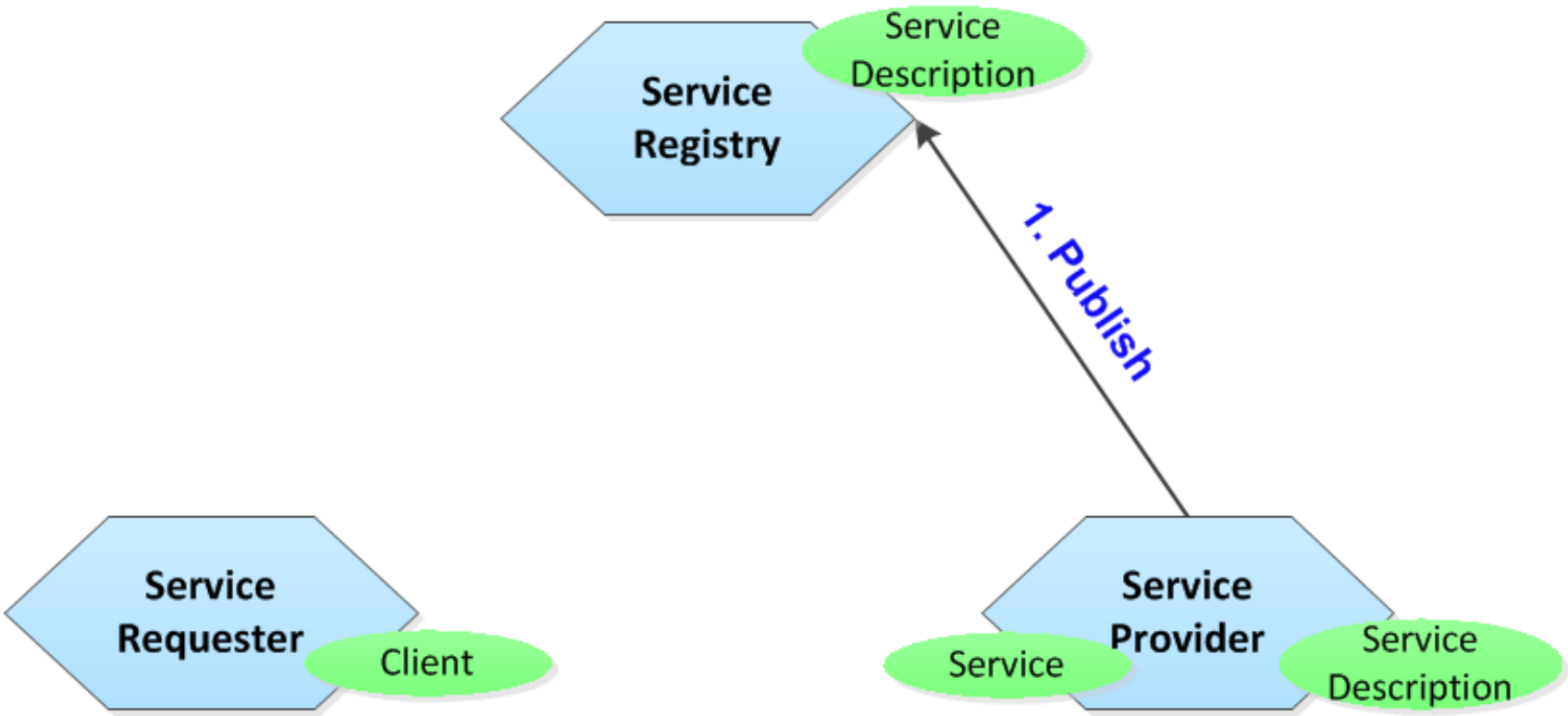


Figure: SOA triangle (based on [W3C, 2002])

Web Services and SOA

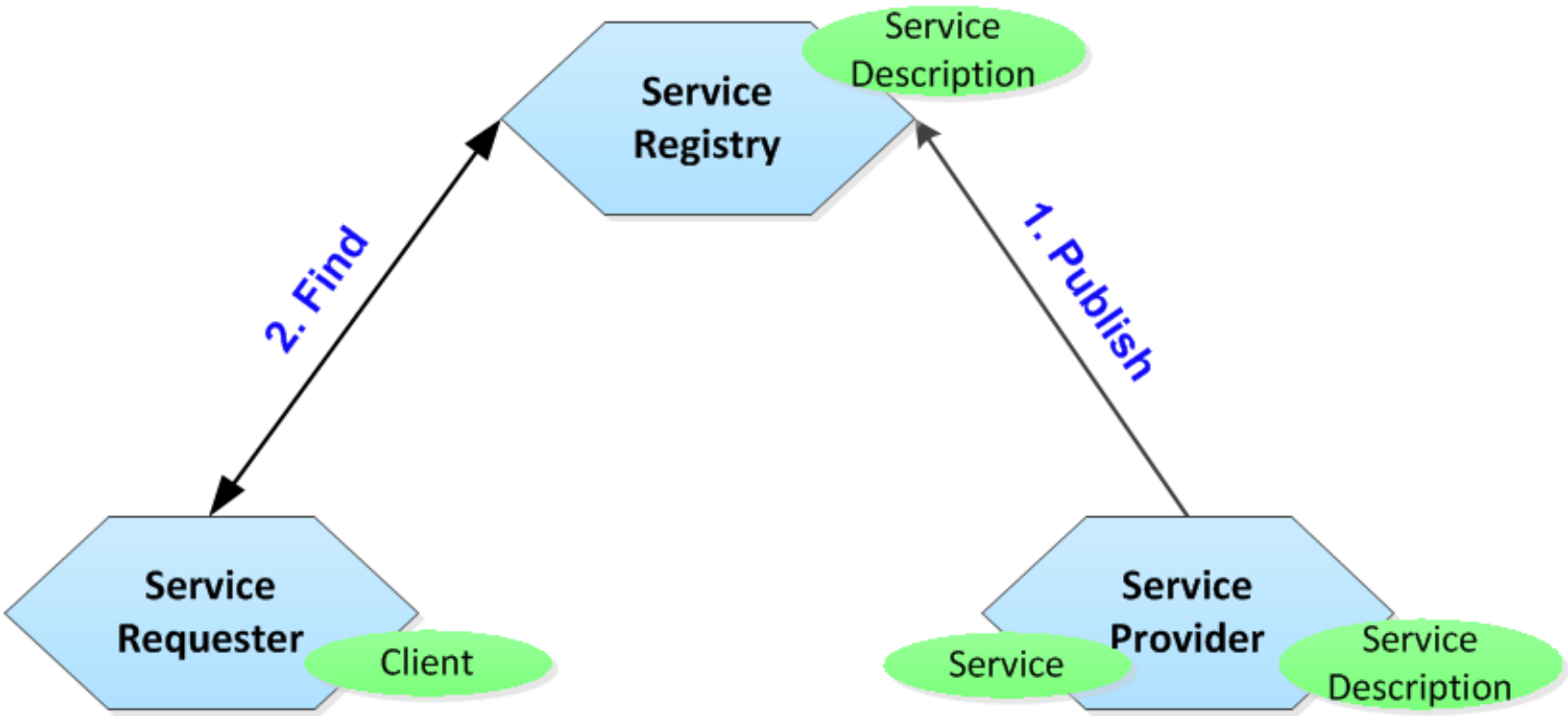


Figure: SOA triangle (based on [W3C, 2002])

Web Services and SOA

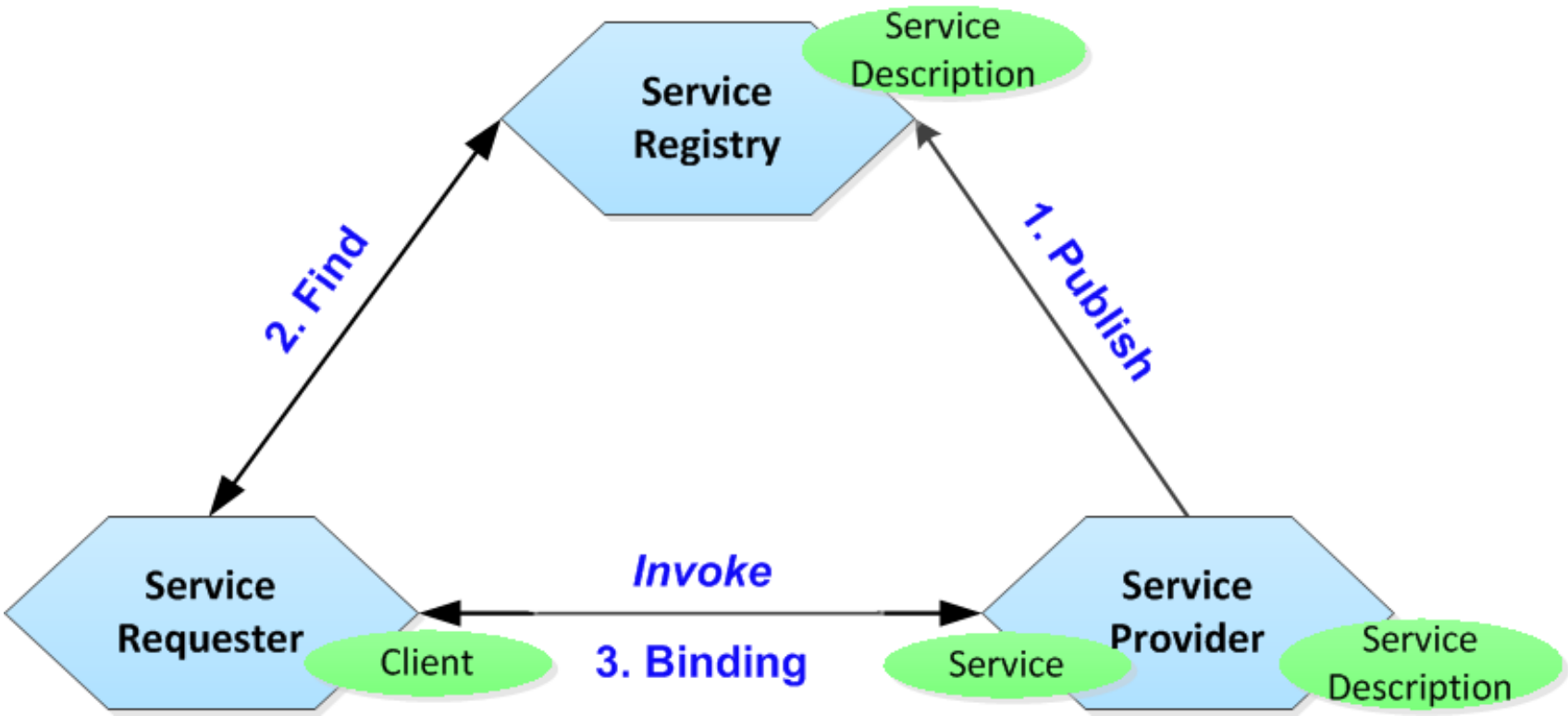


Figure: SOA triangle (based on [W3C, 2002])

Web Services and SOA

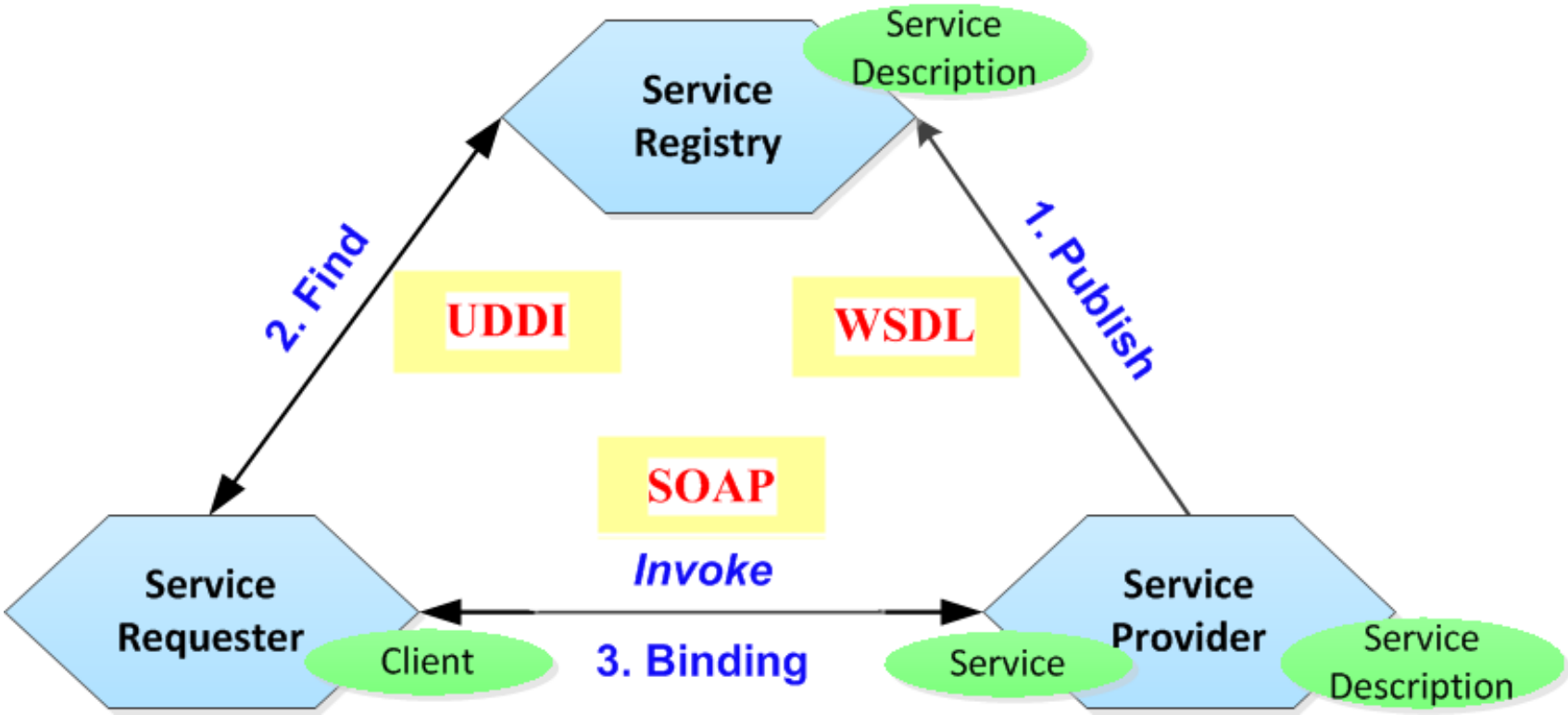


Figure: SOA triangle (based on [W3C, 2002])

Service Orchestration

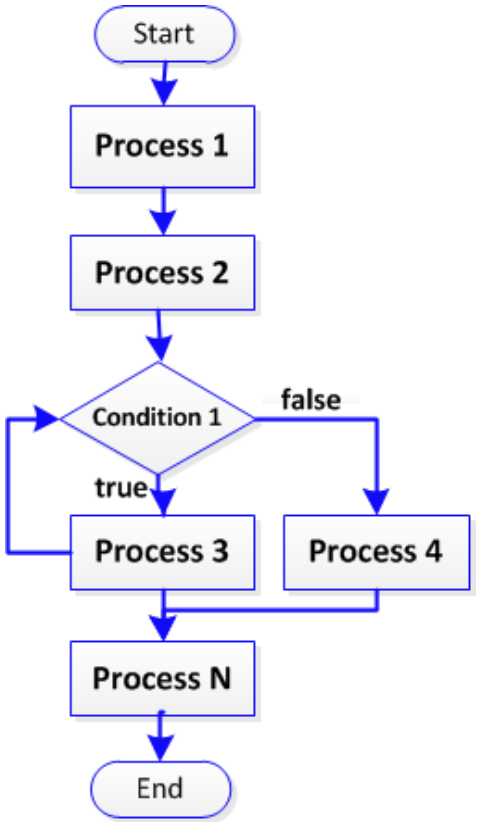


Figure: Service Orchestration

Service Orchestration

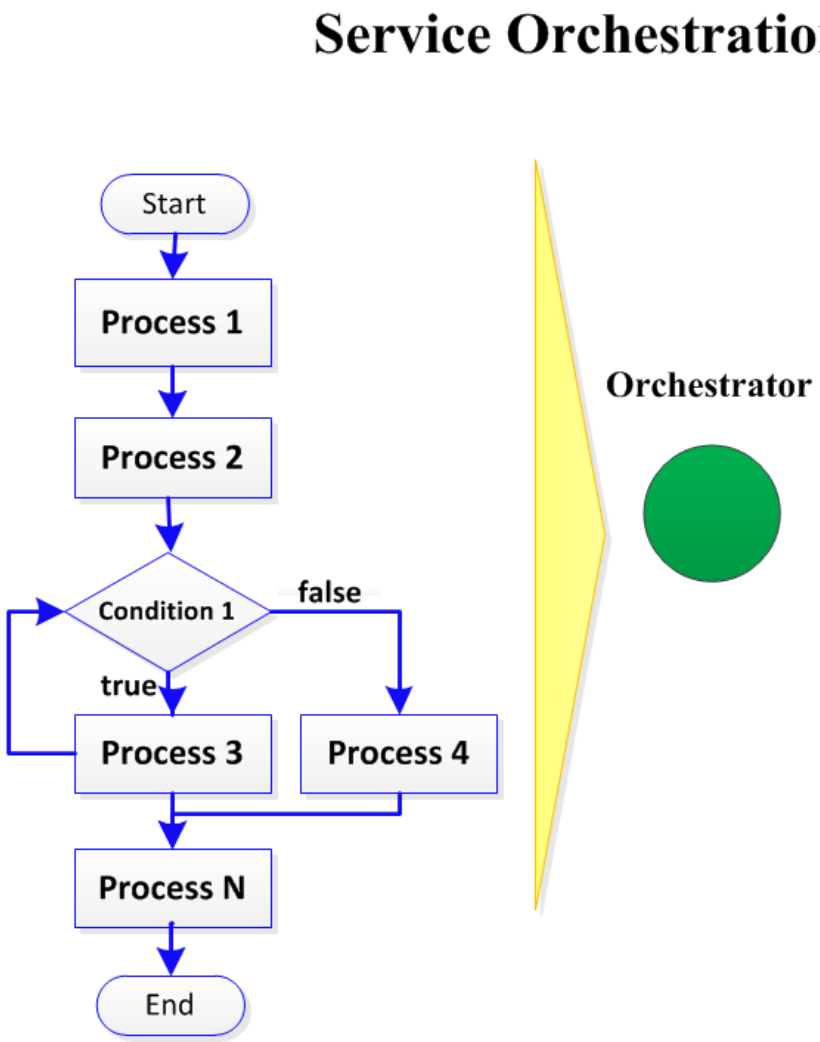


Figure: Service Orchestration

Service Orchestration

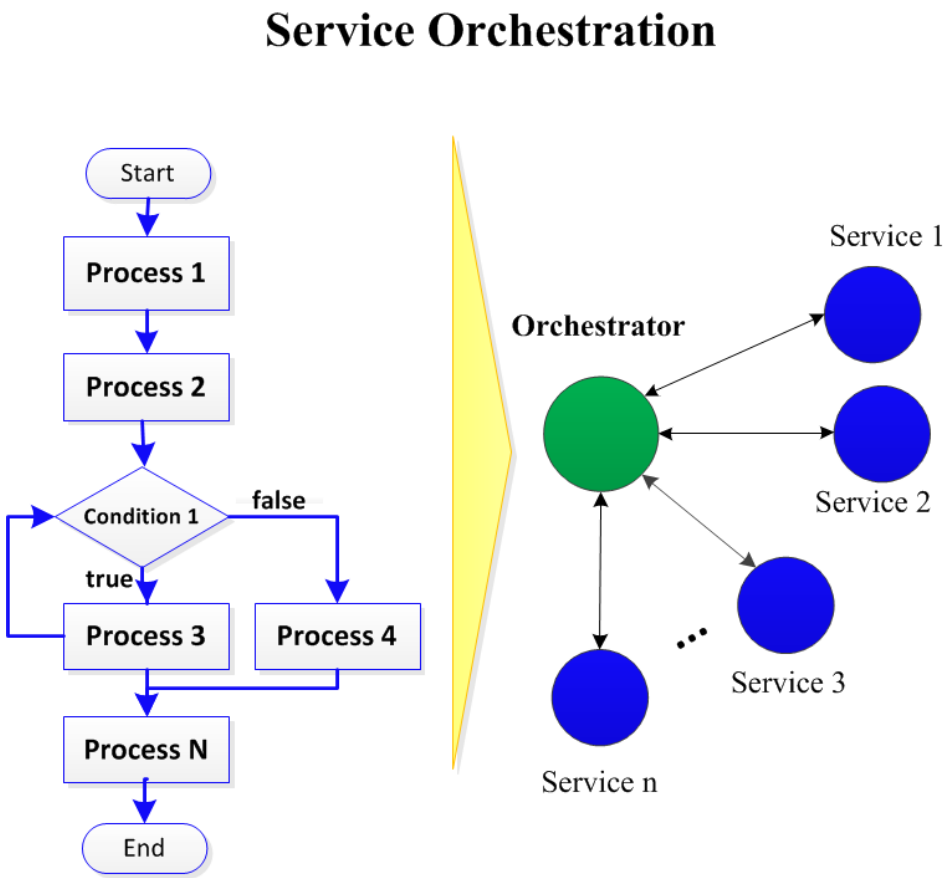


Figure: Service Orchestration

Service Choreography

- Allows service composition in a **collaborative** manner.
- Describes the **P2P interactions** of the externally **observable behavior of its participants**.
- Don't have a single point of control or coordination.

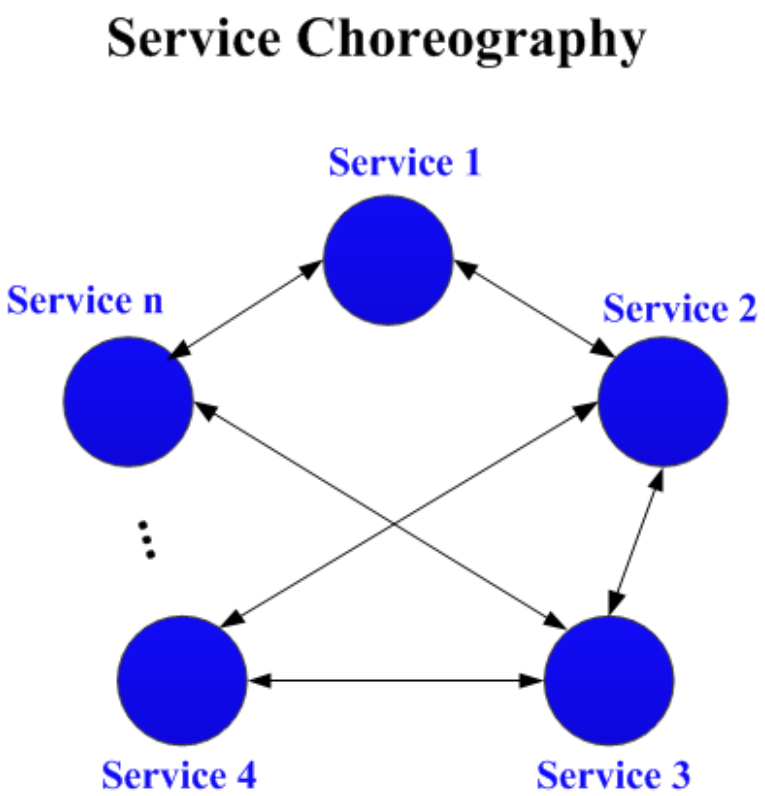


Figure: Service Choreography

Service Choreography

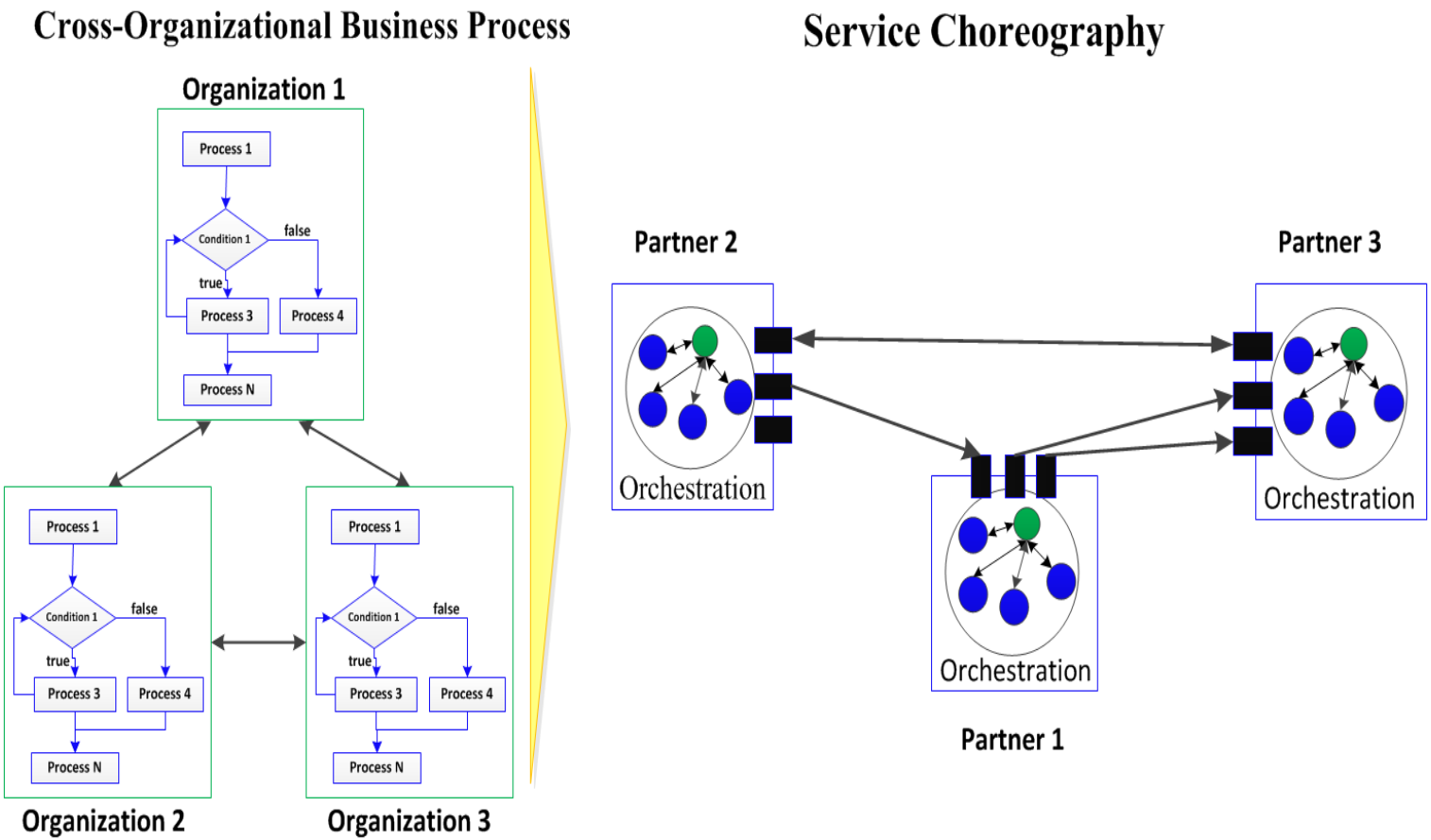


Figure: Service Choreography

Service Choreography

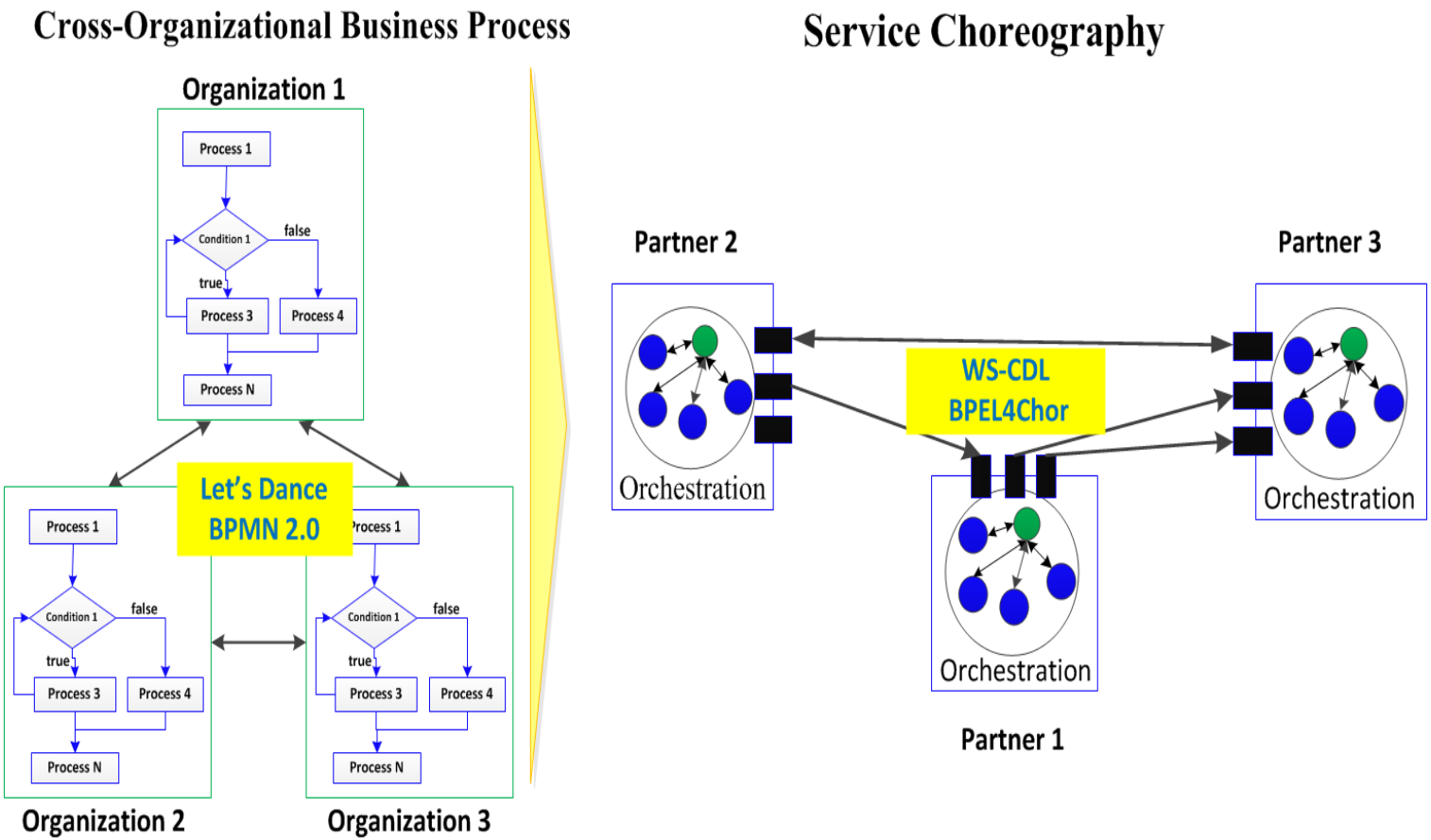


Figure: Service Choreography

BPMN and Choreography

- A Choreography is a type of process.
  - Differs in purpose and behavior from a standard BPMN Process (Process Orchestration).
  - Formalizes the way business Participants **coordinate** their interactions.

BPMN and Choreography

- A Choreography is a type of process.
  - Differs in purpose and behavior from a standard BPMN Process (Process Orchestration).
  - Formalizes the way business Participants **coordinate** their interactions.
- Focus on the exchange of information (Messages) between these Participants.

BPMN and Choreography

- A Choreography is a type of process.
  - Differs in purpose and behavior from a standard BPMN Process (Process Orchestration).
  - Formalizes the way business Participants **coordinate** their interactions.
- Focus on the exchange of information (Messages) between these Participants.
- Two approaches:

BPMN and Choreography

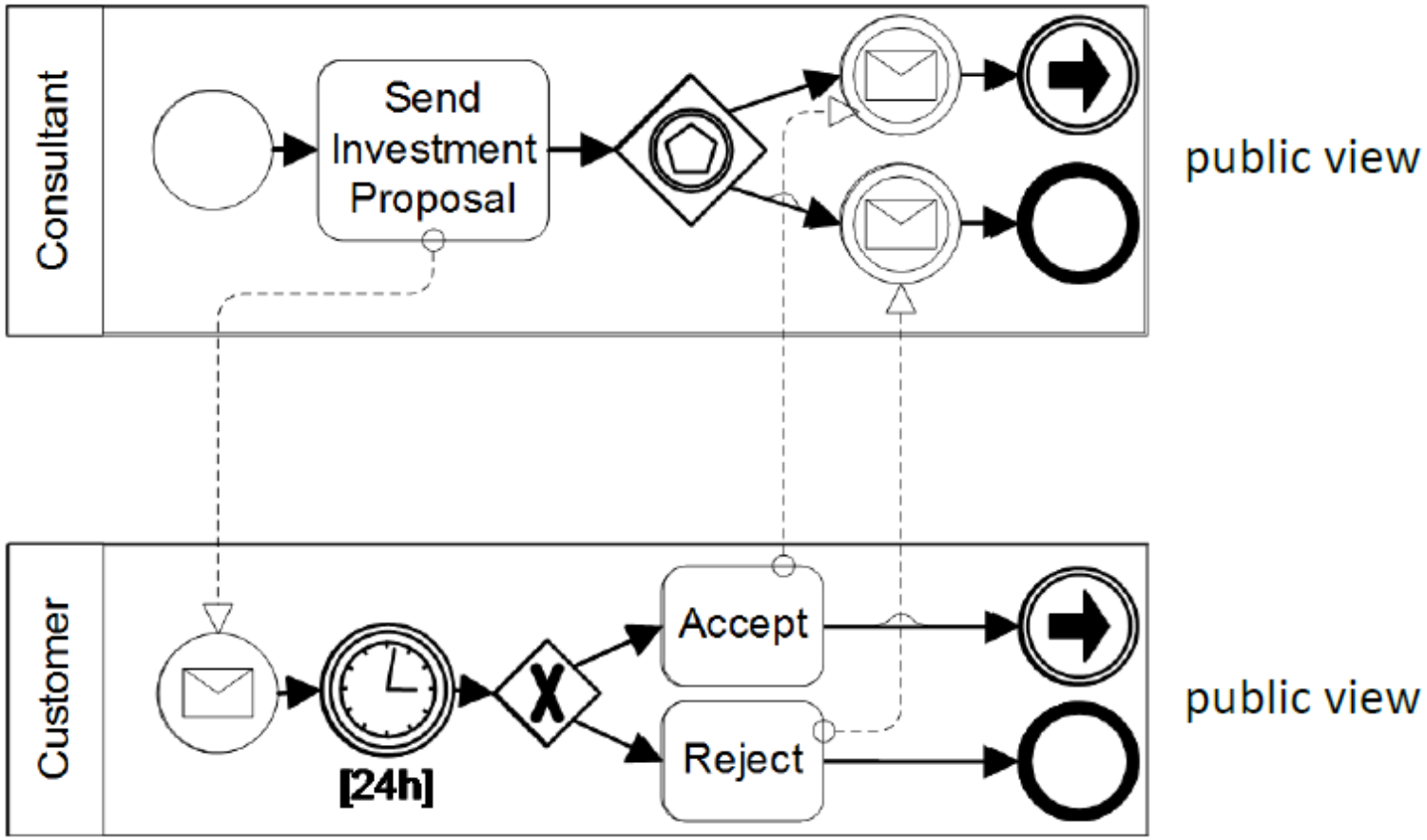
- A Choreography is a type of process.
  - Differs in purpose and behavior from a standard BPMN Process (Process Orchestration).
  - Formalizes the way business Participants **coordinate** their interactions.
- Focus on the exchange of information (Messages) between these Participants.
- Two approaches:
  - Interconnection Model : With collaborations diagrams.

BPMN and Choreography

- A Choreography is a type of process.
  - Differs in purpose and behavior from a standard BPMN Process (Process Orchestration).
  - Formalizes the way business Participants **coordinate** their interactions.
- Focus on the exchange of information (Messages) between these Participants.
- Two approaches:
  - Interconnection Model : With collaborations diagrams.
  - Interaction Model : BPMN Choreographies. using special activities (*Choreography Activity*).

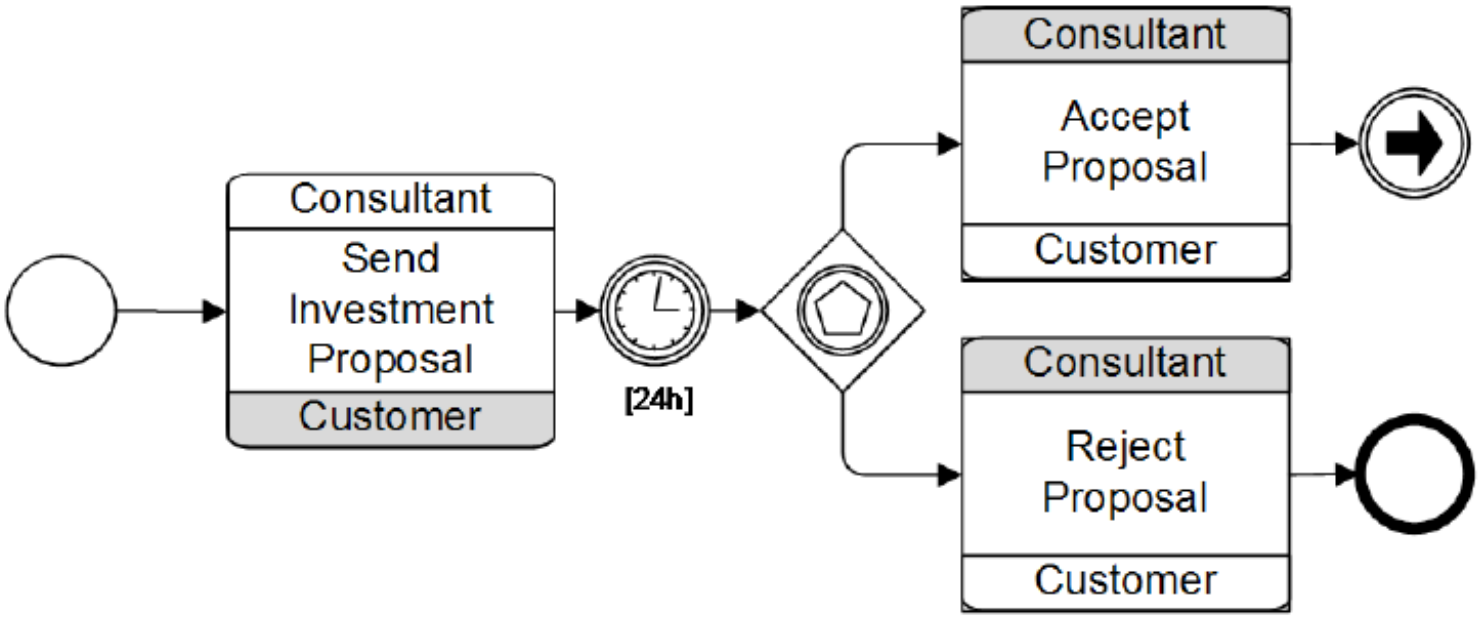
Interconnection Model

- Interconnected public views.
- Use of standard activities.
- "Collaboration" in BPMN 2.0.



Interaction Model

- Interactions **globally captured**.
- Basic building block: **atomic interaction** between two parties.
- "Choreography" in BPMN 2.0.



BPMN Choreography

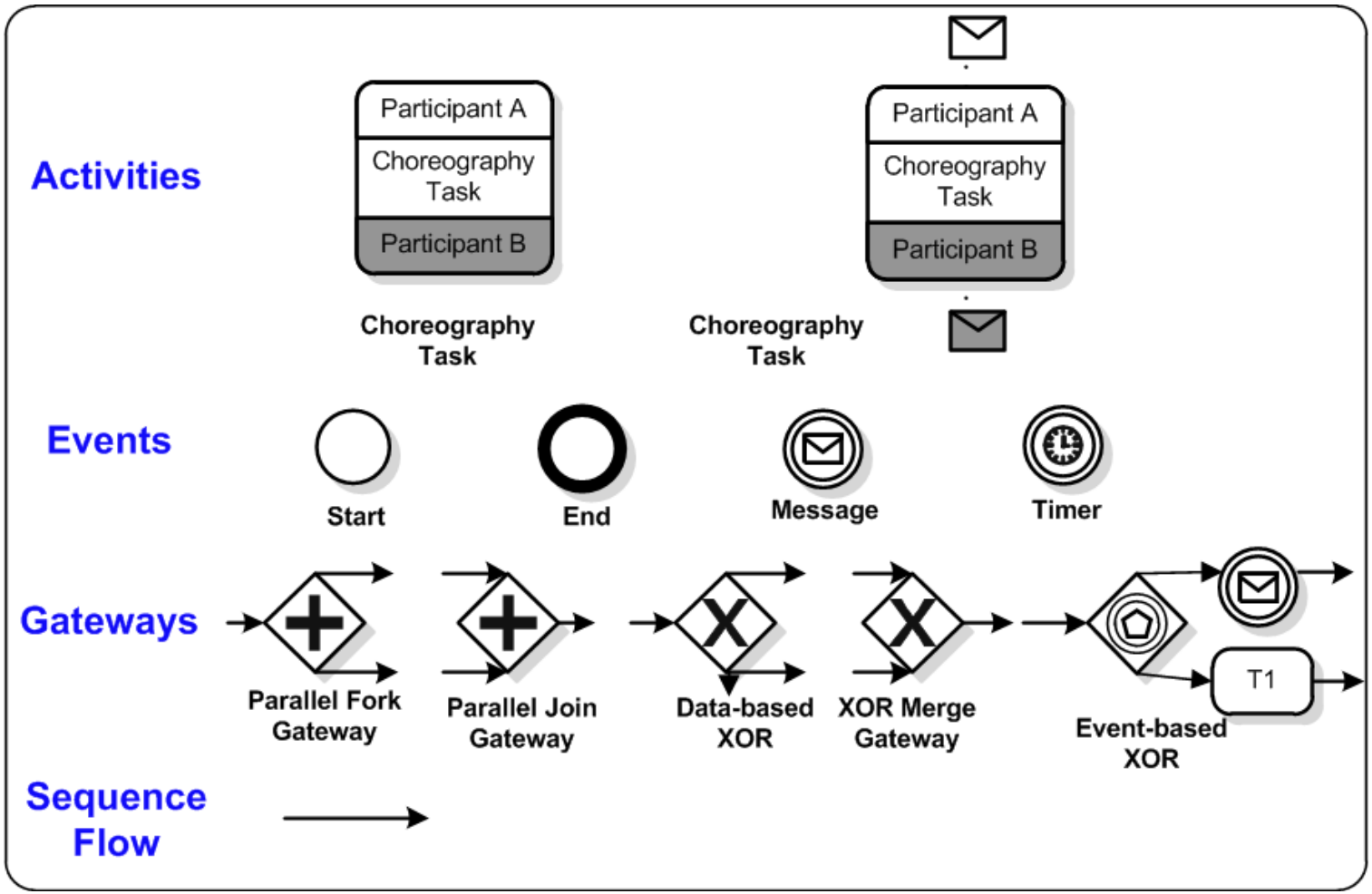


Figure: BPMN elements for modeling choreographies (BPMN 2.0).

Generalized Stochastic Petri Net (GSPN) (I)

Generalized Stochastic Petri Net (GSPN) (II)

Problem to Solve

- Planning of resources before/during development of choreography.

Problem to Solve

- Planning of resources before/during development of choreography.
- Little approaches don't evaluate choreographies:
  - focusing on **QoS** or
  - in earlier stages of development.

Problem to Solve

- Planning of resources before/during development of choreography.
- Little approaches don't evaluate choreographies:
  - focusing on **QoS** or
  - in earlier stages of development.
- To guarantee QoS about communications (network) is important.

Objectives

- To assess the **impact of QoS** attributes in a **choreography interaction model**.
- To propose a novel methodology to establish **requirements for QoS and SLA** in **early stages of development**.
- To plan the capacity of the network elements in choreographies.
- To convert a interaction model to a GSPN (Generalized Stochastic Petri Net) with QoS.

Description

1. Mapping of a choreography to a GSPN.
   - The choreography is specified according "interaction model".
   - The choreography is specified in BPMN 2.0.
   - The resulting GSPN include a QoS model.

Description

1. Mapping of a choreography to a GSPN.
   - The choreography is specified according "interaction model".
   - The choreography is specified in BPMN 2.0.
   - The resulting GSPN include a QoS model.
2. Configurations of resulting GSPN.

Description

1. Mapping of a choreography to a GSPN.
   - The choreography is specified according "interaction model".
   - The choreography is specified in BPMN 2.0.
   - The resulting GSPN include a QoS model.
2. Configurations of resulting GSPN.
3. Simulations of scenarios.

Choreography Formalization

Definition: Process Choreography

A process choreography is a tuple:
$PC = (\mathcal{O}, \mathcal{A}, \mathcal{E}, \mathcal{G}, \mathcal{T}, \{e^S\}, \mathcal{E}^I, \{e^E\}, \mathcal{E}^{I_M}, \mathcal{E}^{I_T}, \mathcal{G}^F, \mathcal{G}^J, \mathcal{G}^X, \mathcal{G}^M, \mathcal{G}^V, \mathcal{F})$
where:

- $\mathcal{O}$ is a set of objects and it's partitioned in **activities** $\mathcal{A}$, **events** $\mathcal{E}$ and **gateways** $\mathcal{G}$.
- $\mathcal{A}$, is the set of **choreography tasks** $\mathcal{T}$.
- $\mathcal{E}$ is the set of **events** and it's is partitioned in **Start event** $e^{\mathcal{S}}$, **Intermediate events** $\mathcal{E}^{\mathcal{I}}$ and **End event** $e^{\mathcal{E}}$.
- $\mathcal{G}$ é the set of **gateways** and is partitioned in **parallel fork gateways** $\mathcal{G}^F$, **parallel join gateways** $\mathcal{G}^J$, **data-based XOR gateways** $\mathcal{G}^X$, **XOR merge gateways** $\mathcal{G}^V$ and **event-based XOR gateways** $\mathcal{G}^M$.
- $\mathcal{F} \subseteq \mathcal{O}x\mathcal{O}$ is the control flow relation, i.e. a **set of sequence flows connecting objects**.

QoS Model

- Defining the QoS attributes involved in **service**, **network** and **message** aspects.
- QoS attributes:
  - In service operation: time to complete the service.
  - In network: delay and communication errors.
  - In message: format of message.

Mapping BPMN to Petri Net (I)

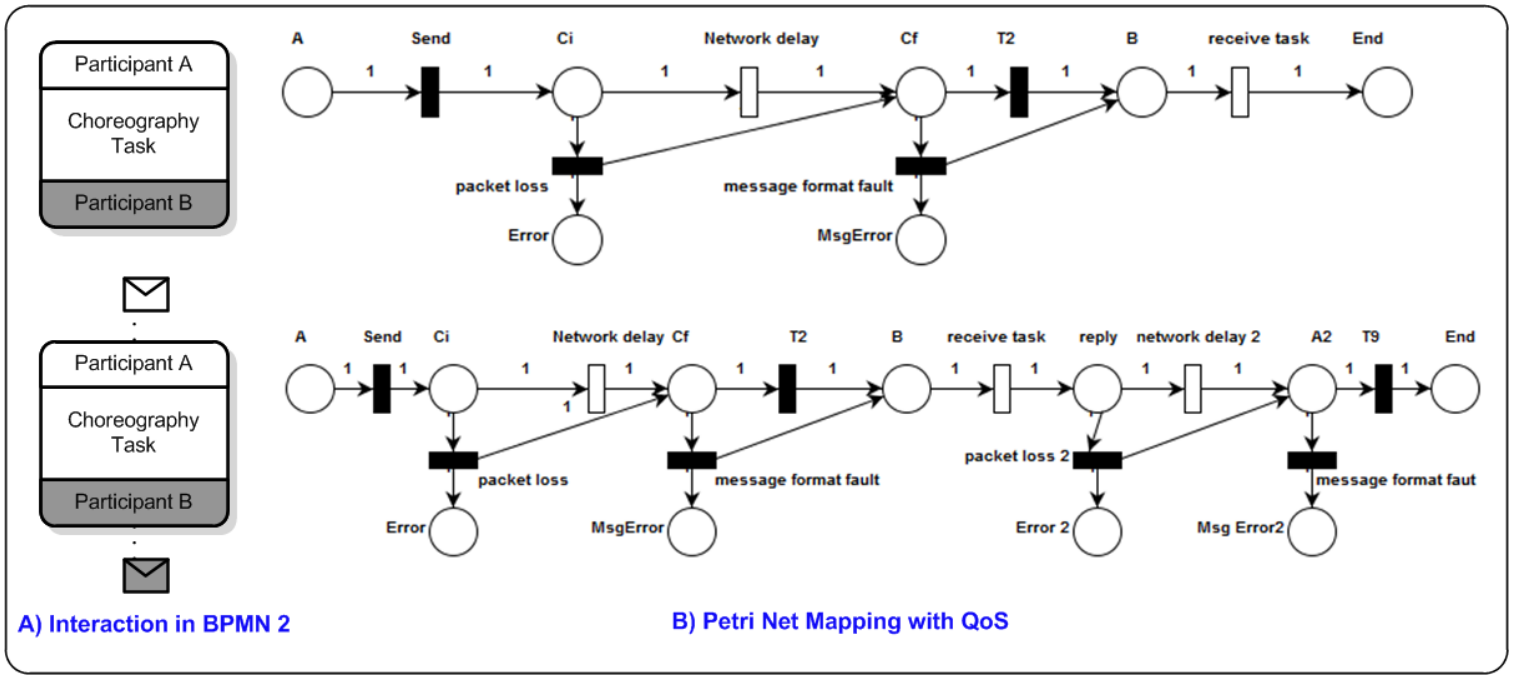


Figure: Mapping of two different choreography tasks with the QoS model

Mapping BPMN to Petri Net (II)

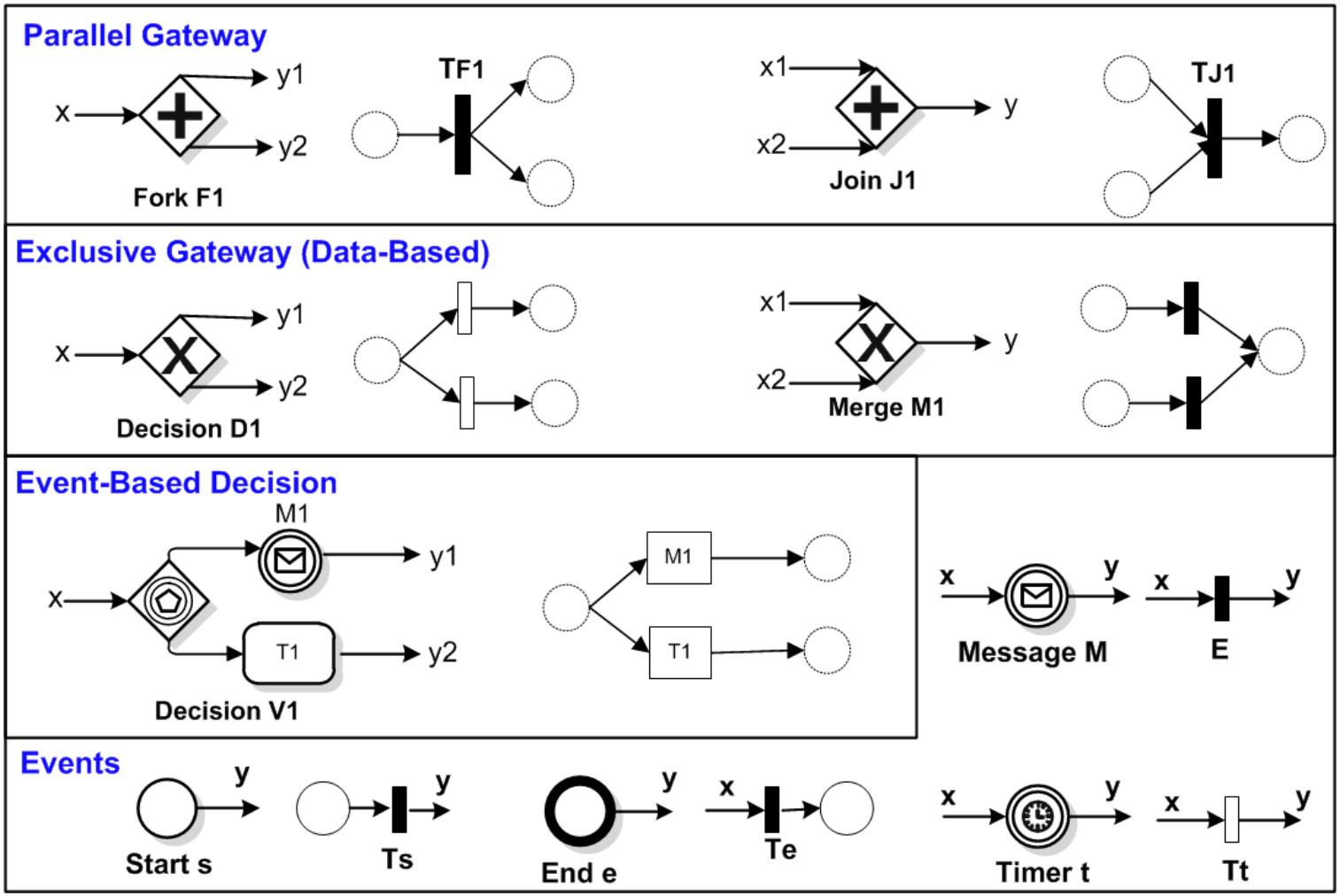


Figure: Mapping of events and gateways elements to modules of Petri nets

Mapping Algorithm (I)

Algorithm 1 Mapping of choreography specified in BPMN 2.0 to a GSPN with QoS model

Input: **Process Choreography** $PC = (\mathcal{O}, \mathcal{A}, \mathcal{E}, \mathcal{G}, \mathcal{T}, \{e^S\}, \mathcal{E}^I, \{e^E\}, \mathcal{E}^{I_M}, \mathcal{E}^{I_T}, \mathcal{G}^F, \mathcal{G}^J, \mathcal{G}^X, \mathcal{G}^M, \mathcal{G}^V, \mathcal{F})$ in BPMN 2.0.

Output: Generalized Stochastic Petri Net $GSPN_{QoS}$.

Mapping Algorithm (I)

Algorithm 2 Mapping of choreography specified in BPMN 2.0 to a GSPN with QoS model

Input: **Process Choreography** $PC = (\mathcal{O}, \mathcal{A}, \mathcal{E}, \mathcal{G}, \mathcal{T}, \{e^S\}, \mathcal{E}^I, \{e^E\}, \mathcal{E}^{I_M}, \mathcal{E}^{I_T}, \mathcal{G}^F, \mathcal{G}^J, \mathcal{G}^X, \mathcal{G}^M, \mathcal{G}^V, \mathcal{F})$ in BPMN 2.0.

Output: Generalized Stochastic Petri Net $GSPN_{QoS}$.

Consider $CT_i \in \mathcal{T}$, $G_j \in \mathcal{G}$ and $E_k \in \mathcal{E}$. $i, j, k \in \mathbb{N}$.

Mapping Algorithm (I)

Algorithm 3 Mapping of choreography specified in BPMN 2.0 to a GSPN with QoS model

Input: **Process Choreography** $PC = (\mathcal{O}, \mathcal{A}, \mathcal{E}, \mathcal{G}, \mathcal{T}, \{e^S\}, \mathcal{E}^I, \{e^E\}, \mathcal{E}^{I_M}, \mathcal{E}^{I_T}, \mathcal{G}^F, \mathcal{G}^J, \mathcal{G}^X, \mathcal{G}^M, \mathcal{G}^V, \mathcal{F})$ in BPMN 2.0.

Output: Generalized Stochastic Petri Net $GSPN_{QoS}$.

Consider $CT_i \in \mathcal{T}$, $G_j \in \mathcal{G}$ and $E_k \in \mathcal{E}$. $i, j, k \in \mathbb{N}$.

Consider $PNQoS(CT_i)$ the respective GSPN including QoS as a function of the type of CT_i according to mapping rules.

Mapping Algorithm (I)

Algorithm 4 Mapping of choreography specified in BPMN 2.0 to a GSPN with QoS model

Input: **Process Choreography** $PC = (\mathcal{O}, \mathcal{A}, \mathcal{E}, \mathcal{G}, \mathcal{T}, \{e^S\}, \mathcal{E}^I, \{e^E\}, \mathcal{E}^{I_M}, \mathcal{E}^{I_T}, \mathcal{G}^F, \mathcal{G}^J, \mathcal{G}^X, \mathcal{G}^M, \mathcal{G}^V, \mathcal{F})$ in BPMN 2.0.

Output: Generalized Stochastic Petri Net $GSPN_{QoS}$.

Consider $CT_i \in \mathcal{T}$, $G_j \in \mathcal{G}$ and $E_k \in \mathcal{E}$. $i, j, k \in \mathbb{N}$.

Consider $PNQoS(CT_i)$ the respective GSPN including QoS as a function of the type of CT_i according to mapping rules.
Consider $PNQoS(G_j)$ the respective GSPN including QoS as a function of the type of G_j according to mapping rules.

Mapping Algorithm (I)

Algorithm 5 Mapping of choreography specified in BPMN 2.0 to a GSPN with QoS model

Input: **Process Choreography** $PC = (\mathcal{O}, \mathcal{A}, \mathcal{E}, \mathcal{G}, \mathcal{T}, \{e^S\}, \mathcal{E}^I, \{e^E\}, \mathcal{E}^{I_M}, \mathcal{E}^{I_T}, \mathcal{G}^F, \mathcal{G}^J, \mathcal{G}^X, \mathcal{G}^M, \mathcal{G}^V, \mathcal{F})$ in BPMN 2.0.

Output: Generalized Stochastic Petri Net $GSPN_{QoS}$.

Consider $CT_i \in \mathcal{T}$, $G_j \in \mathcal{G}$ and $E_k \in \mathcal{E}$. $i, j, k \in \mathbb{N}$.

Consider $PNQoS(CT_i)$ the respective GSPN including QoS as a function of the type of CT_i according to mapping rules.
Consider $PNQoS(G_j)$ the respective GSPN including QoS as a function of the type of G_j according to mapping rules.
Consider $PNQoS(E_k)$ the respective GSPN including QoS as a function of the type of E_k according to mapping rules.

Mapping Algorithm (I)

Algorithm 6 Mapping of choreography specified in BPMN 2.0 to a GSPN with QoS model

Input: **Process Choreography** $PC = (\mathcal{O}, \mathcal{A}, \mathcal{E}, \mathcal{G}, \mathcal{T}, \{e^S\}, \mathcal{E}^I, \{e^E\}, \mathcal{E}^{I_M}, \mathcal{E}^{I_T}, \mathcal{G}^F, \mathcal{G}^J, \mathcal{G}^X, \mathcal{G}^M, \mathcal{G}^V, \mathcal{F})$ in BPMN 2.0.

Output: Generalized Stochastic Petri Net $GSPN_{QoS}$.

Consider $CT_i \in \mathcal{T}$, $G_j \in \mathcal{G}$ and $E_k \in \mathcal{E}$. $i, j, k \in \mathbb{N}$.

Consider $PNQoS(CT_i)$ the respective GSPN including QoS as a function of the type of CT_i according to mapping rules.
Consider $PNQoS(G_j)$ the respective GSPN including QoS as a function of the type of G_j according to mapping rules.
Consider $PNQoS(E_k)$ the respective GSPN including QoS as a function of the type of E_k according to mapping rules.

Consider $\oplus$ the binary operator of composition of two GSPNs that returns other GSPN.

Mapping Algorithm (II)

$GSPN_{QoS} \leftarrow$ *Empty Petri Net*

Mapping Algorithm (II)

$GSPN_{QoS} \leftarrow$ *Empty Petri Net*
For $CT_i \in \mathcal{T}$ **Do**

End

Mapping Algorithm (II)

$GSPN_{QoS} \leftarrow$ *Empty Petri Net*

For $CT_i \in \mathcal{T}$ **Do**

$GSPN_{QoS} \leftarrow GSPN_{QoS} \oplus PNQoS(CT_i)$

End

Mapping Algorithm (II)

$GSPN_{QoS} \leftarrow$ *Empty Petri Net*

For $CT_i \in \mathcal{T}$ **Do**

$GSPN_{QoS} \leftarrow GSPN_{QoS} \oplus PNQoS(CT_i)$

Add a arrival **timed Transition** at beginning of the $GSPN_{QoS}$.

End

Mapping Algorithm (II)

$GSPN_{QoS} \leftarrow$ *Empty Petri Net*

For $CT_i \in \mathcal{T}$ **Do**

$GSPN_{QoS} \leftarrow GSPN_{QoS} \oplus PNQoS(CT_i)$

Add a arrival **timed Transition** at beginning of the $GSPN_{QoS}$.

End

For $G_j \in \mathcal{G}$ **Do**

End

Mapping Algorithm (II)

$GSPN_{QoS} \leftarrow$ *Empty Petri Net*
For $CT_i \in \mathcal{T}$ **Do**
  $GSPN_{QoS} \leftarrow GSPN_{QoS} \oplus PNQoS(CT_i)$
  Add a arrival **timed Transition** at beginning of the $GSPN_{QoS}$.
End

For $G_j \in \mathcal{G}$ **Do**
  $GSPN_{QoS} \leftarrow GSPN_{QoS} \oplus PN(G_j)$
End

Mapping Algorithm (II)

$GSPN_{QoS} \leftarrow$ *Empty Petri Net*
For $CT_i \in \mathcal{T}$ **Do**
  $GSPN_{QoS} \leftarrow GSPN_{QoS} \oplus PNQoS(CT_i)$
  Add a arrival **timed Transition** at beginning of the $GSPN_{QoS}$.
End

For $G_j \in \mathcal{G}$ **Do**
  $GSPN_{QoS} \leftarrow GSPN_{QoS} \oplus PN(G_j)$
End

For $E_k \in \mathcal{E}$ **Do**

End

Mapping Algorithm (II)

$GSPN_{QoS} \leftarrow$ *Empty Petri Net*
For $CT_i \in \mathcal{T}$ **Do**
  $GSPN_{QoS} \leftarrow GSPN_{QoS} \oplus PNQoS(CT_i)$
  Add a arrival **timed Transition** at beginning of the $GSPN_{QoS}$.
End

For $G_j \in \mathcal{G}$ **Do**
  $GSPN_{QoS} \leftarrow GSPN_{QoS} \oplus PN(G_j)$
End

For $E_k \in \mathcal{E}$ **Do**
  $GSPN_{QoS} \leftarrow GSPN_{QoS} \oplus PN(E_k)$
End

Mapping Algorithm (II)

$GSPN_{QoS} \leftarrow$ *Empty Petri Net*
For $CT_i \in \mathcal{T}$ **Do**
  $GSPN_{QoS} \leftarrow GSPN_{QoS} \oplus PNQoS(CT_i)$
  Add a arrival **timed Transition** at beginning of the $GSPN_{QoS}$.
End

For $G_j \in \mathcal{G}$ **Do**
  $GSPN_{QoS} \leftarrow GSPN_{QoS} \oplus PN(G_j)$
End

For $E_k \in \mathcal{E}$ **Do**
  $GSPN_{QoS} \leftarrow GSPN_{QoS} \oplus PN(E_k)$
End

Add a starting Place and **immediate Transition** at the beginning of the $GSPN_{QoS}$.

Mapping Algorithm (II)

$GSPN_{QoS} \leftarrow$ *Empty Petri Net*
For $CT_i \in \mathcal{T}$ **Do**
  $GSPN_{QoS} \leftarrow GSPN_{QoS} \oplus PNQoS(CT_i)$
  Add a arrival **timed Transition** at beginning of the $GSPN_{QoS}$.
End

For $G_j \in \mathcal{G}$ **Do**
  $GSPN_{QoS} \leftarrow GSPN_{QoS} \oplus PN(G_j)$
End

For $E_k \in \mathcal{E}$ **Do**
  $GSPN_{QoS} \leftarrow GSPN_{QoS} \oplus PN(E_k)$
End

Add a starting Place and **immediate Transition** at the beginning of the $GSPN_{QoS}$.
Add a ending Place and **immediate Transition** at the end of the $GSPN_{QoS}$.

Mapping Algorithm (II)

$GSPN_{QoS} \leftarrow$ *Empty Petri Net*
For $CT_i \in \mathcal{T}$ **Do**
  $GSPN_{QoS} \leftarrow GSPN_{QoS} \oplus PNQoS(CT_i)$
  Add a arrival **timed Transition** at beginning of the $GSPN_{QoS}$.
End

For $G_j \in \mathcal{G}$ **Do**
  $GSPN_{QoS} \leftarrow GSPN_{QoS} \oplus PN(G_j)$
End

For $E_k \in \mathcal{E}$ **Do**
  $GSPN_{QoS} \leftarrow GSPN_{QoS} \oplus PN(E_k)$
End

Add a starting Place and **immediate Transition** at the beginning of the $GSPN_{QoS}$.
Add a ending Place and **immediate Transition** at the end of the $GSPN_{QoS}$.

Return $GSPN_{QoS}$

Scenario

Mapping

Configuration

Simulation

Results

Conclusions

- We have prosposed a Nobel methodology to aid define QoS and SLA requirements in service Choreography.

Conclusions

- We have prosposed a Nobel methodology to aid define QoS and SLA requirements in service Choreography.

Future Works

- .

Future Works

- .

Thanks so much!