

A Methodology to Define QoS and SLA Requirements in Service Choreographies

Authors

Victoriano Alfonso Phocco Diaz

Daniel Macedo Batista

Institute of Mathematics and Statistics

Department of Computer Science

University of Sao Paulo

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

September 17, 2012

Agenda

- 1 Introduction
- 2 Problem
- 3 Methodology
- 4 Performance Evaluation
- 5 Conclusions and Future Works

- 1 Introduction
- 2 Problem
- 3 Methodology
- 4 Performance Evaluation
- 5 Conclusions and Future Works

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 (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 (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 (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 (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 (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

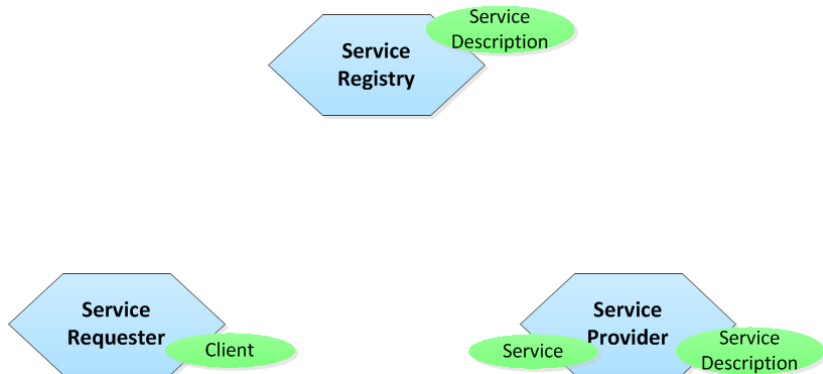


Figure: SOA triangle (based on [W3C, 2002])

Web Services and SOA

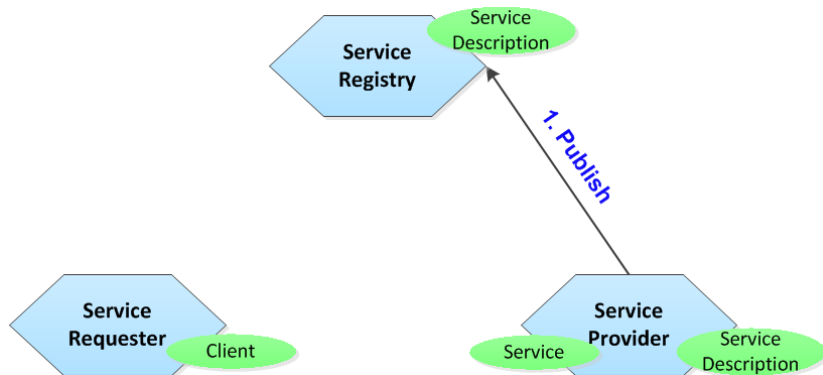


Figure: SOA triangle (based on [W3C, 2002])

Web Services and SOA

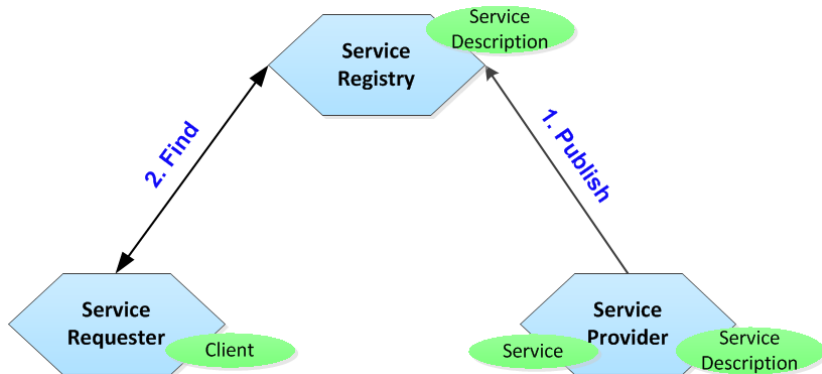


Figure: SOA triangle (based on [W3C, 2002])

Web Services and SOA

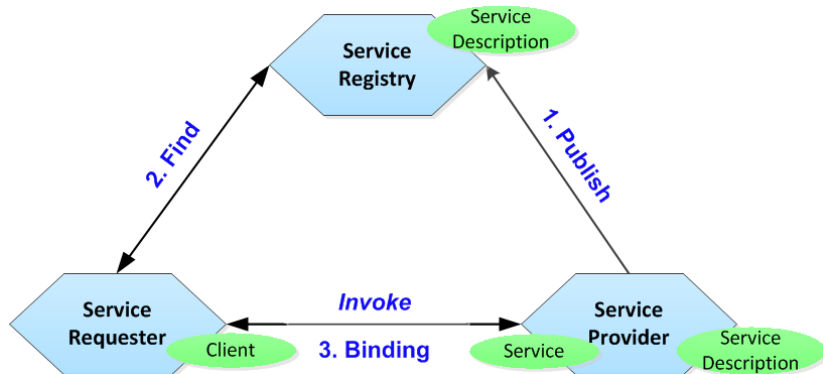


Figure: SOA triangle (based on [W3C, 2002])

Web Services and SOA

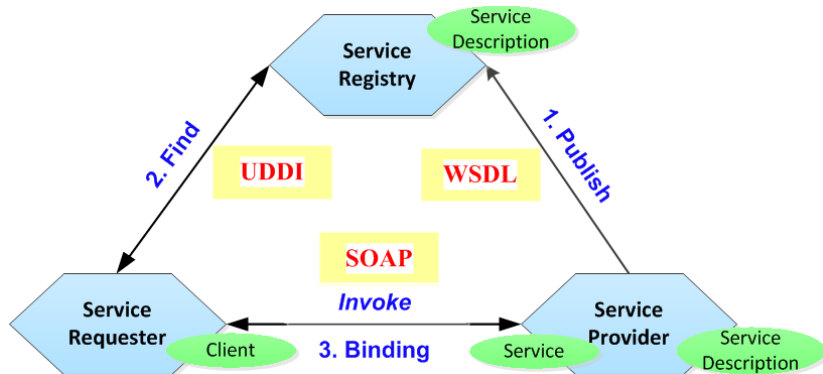


Figure: SOA triangle (based on [W3C, 2002])

Service Orchestration

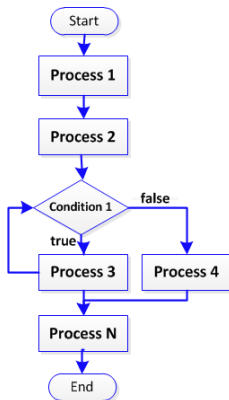


Figure: Service Orchestration

Service Orchestration

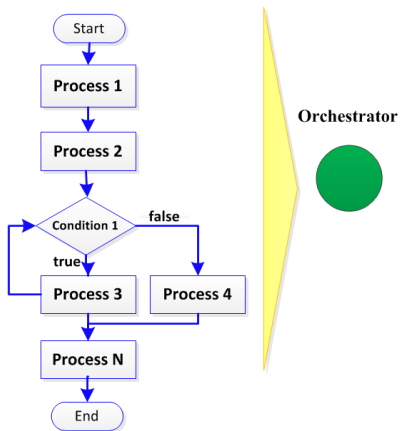


Figure: Service Orchestration

Service Orchestration

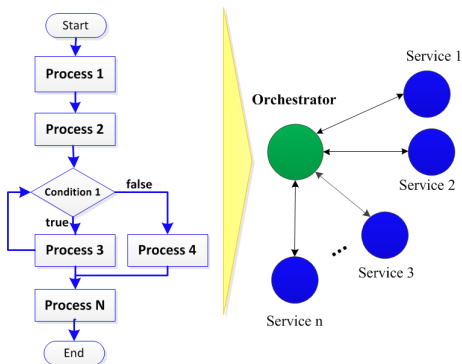


Figure: Service Orchestration

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.

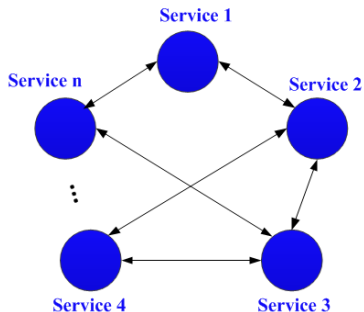
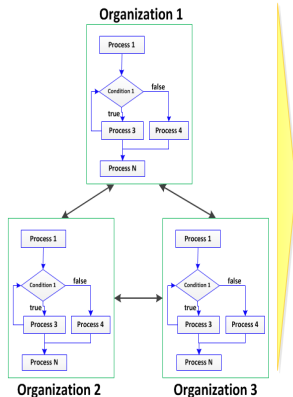


Figure: Service Choreography

Service Choreography

Cross-Organizational Business Process



Service Choreography

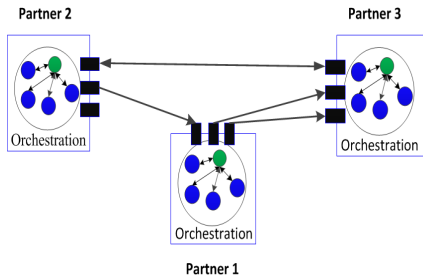
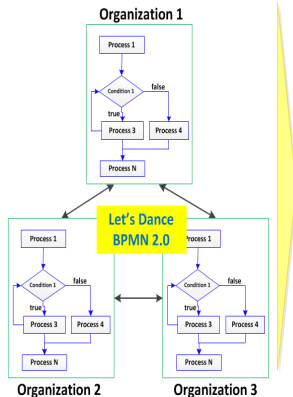


Figure: Service Choreography

Service Choreography

Cross-Organizational Business Process



Service Choreography

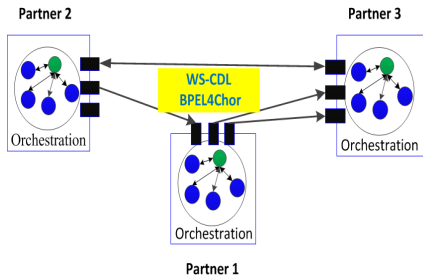


Figure: Service Choreography

- 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**.

- 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.

- 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:

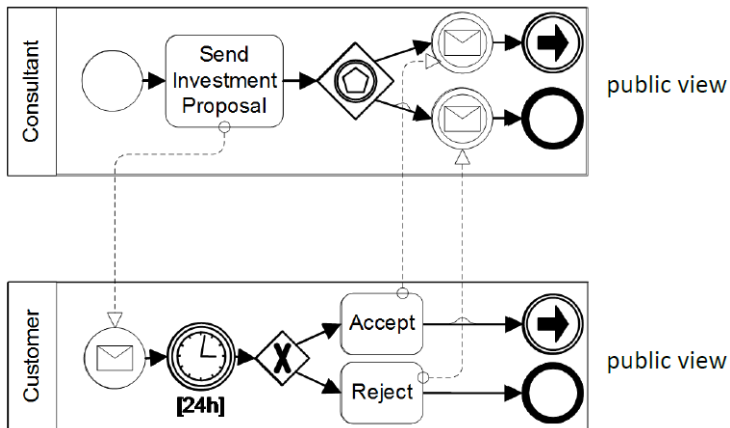
- A Choreography is a type of process.
 - ▶ Differs in purpose and behavior from a standard BPMN Process (Process Orchestration).
 - ▶ Formalizes the way business **Participants** **coordinate** their **interactions**.
- Focus on the exchange of information (**Messages**) between these Participants.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

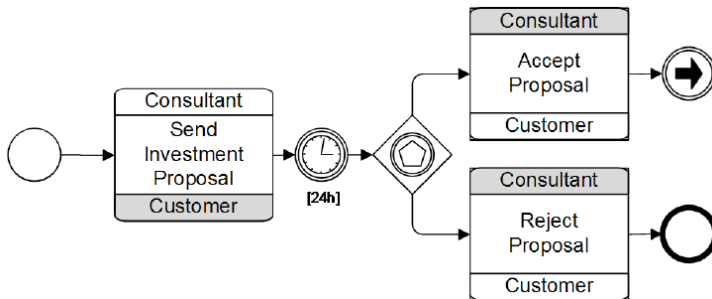
Interconnection Model

- 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

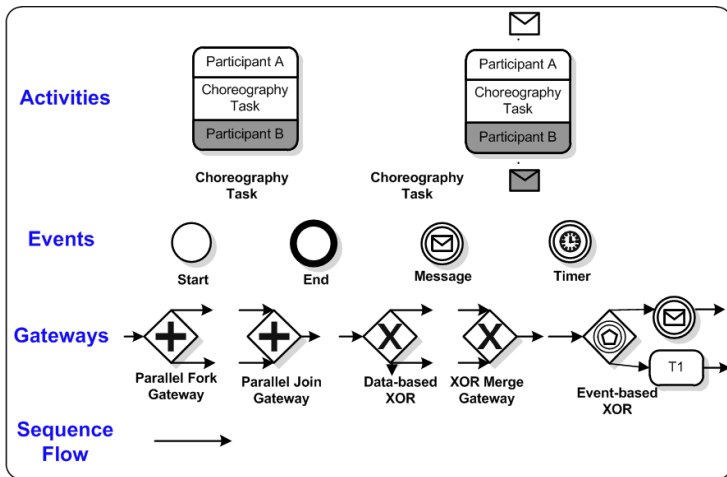


Figure: BPMN elements for modeling choreographies (BPMN 2.0).

Petri Net (PN)

- **PN** is a formalism for modeling **concurrency**, **causality** and **conflict**.

Petri Net (PN)

- **PN** is a formalism for modeling **concurrency**, **causality** and **conflict**.
- PN is a **bipartite graph** with the following elements:

Petri Net (PN)

- **PN** is a formalism for modeling **concurrency**, **causality** and **conflict**.
- PN is a **bipartite graph** with the following elements:

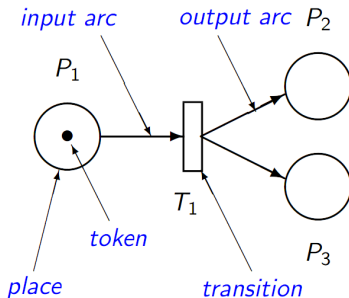


Figure: Elements of a petri net.

Generalized Stochastic Petri Net (GSPN)

- **GSPN** is an extension of PN that associate an **exponentially distributed delay** with the firing of each transition (**time transition**).

Generalized Stochastic Petri Net (GSPN)

- **GSPN** is an extension of PN that associate an **exponentially distributed delay** with the firing of each transition (**time transition**).
- New elements: **immediate transitions**(no time) and **inhibitor arcs**.

Generalized Stochastic Petri Net (GSPN)

- **GSPN** is an extension of PN that associate an **exponentially distributed delay** with the firing of each transition (**time transition**).
- New elements: **immediate transitions**(no time) and **inhibitor arcs**.

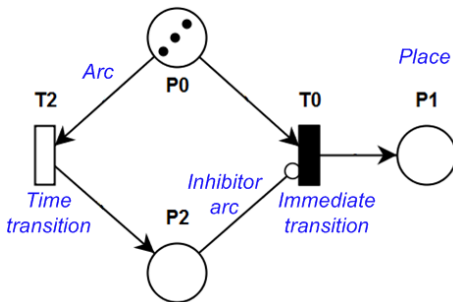


Figure: Elements of a GSPN.

- 1 Introduction
- 2 Problem
- 3 Methodology
- 4 Performance Evaluation
- 5 Conclusions and Future Works

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 including a QoS model.

- 1 Introduction
- 2 Problem
- 3 Methodology**
- 4 Performance Evaluation
- 5 Conclusions and Future Works

- 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.

- 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 the resulting GSPN.

- 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 the resulting GSPN.
- ③ **Simulations** of scenarios.

Choreography Formalization

Definition: Process Choreography

A process choreography is a tuple:

Choreography Formalization

Definition: Process Choreography

A process choreography is a tuple:

$$PC = (\mathcal{O}, \mathcal{A}, \mathcal{E}, \mathcal{G}, \mathcal{T}, \{e^S\}, \mathcal{E}^I, \{e^E\}, \mathcal{E}^{IM}, \mathcal{E}^{IT}, \mathcal{G}^F, \mathcal{G}^J, \mathcal{G}^X, \mathcal{G}^M, \mathcal{G}^V, \mathcal{F})$$

where:

- \mathcal{O} is a set of objects and it's partitioned in **activities** \mathcal{A} , **events** \mathcal{E} and **gateways** \mathcal{G} .

Choreography Formalization

Definition: Process Choreography

A process choreography is a tuple:

$$PC = (\mathcal{O}, \mathcal{A}, \mathcal{E}, \mathcal{G}, \mathcal{T}, \{e^S\}, \mathcal{E}^I, \{e^E\}, \mathcal{E}^{IM}, \mathcal{E}^{IT}, \mathcal{G}^F, \mathcal{G}^J, \mathcal{G}^X, \mathcal{G}^M, \mathcal{G}^V, \mathcal{F})$$

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} .

Choreography Formalization

Definition: Process Choreography

A process choreography is a tuple:

$$PC = (\mathcal{O}, \mathcal{A}, \mathcal{E}, \mathcal{G}, \mathcal{T}, \{e^S\}, \mathcal{E}^I, \{e^E\}, \mathcal{E}^{IM}, \mathcal{E}^{IT}, \mathcal{G}^F, \mathcal{G}^J, \mathcal{G}^X, \mathcal{G}^M, \mathcal{G}^V, \mathcal{F})$$

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 partitioned in **Start event** e^S , **Intermediate events** \mathcal{E}^I and **End event** e^E .

Choreography Formalization

Definition: Process Choreography

A process choreography is a tuple:

$$PC = (\mathcal{O}, \mathcal{A}, \mathcal{E}, \mathcal{G}, \mathcal{T}, \{e^S\}, \mathcal{E}^I, \{e^E\}, \mathcal{E}^{IM}, \mathcal{E}^{IT}, \mathcal{G}^F, \mathcal{G}^J, \mathcal{G}^X, \mathcal{G}^M, \mathcal{G}^V, \mathcal{F})$$

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 partitioned in **Start event** e^S , **Intermediate events** \mathcal{E}^I and **End event** e^E .
- \mathcal{G} is 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 .

Choreography Formalization

Definition: Process Choreography

A process choreography is a tuple:

$$PC = (\mathcal{O}, \mathcal{A}, \mathcal{E}, \mathcal{G}, \mathcal{T}, \{e^S\}, \mathcal{E}^I, \{e^E\}, \mathcal{E}^{IM}, \mathcal{E}^{IT}, \mathcal{G}^F, \mathcal{G}^J, \mathcal{G}^X, \mathcal{G}^M, \mathcal{G}^V, \mathcal{F})$$

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 partitioned in **Start event** e^S , **Intermediate events** \mathcal{E}^I and **End event** e^E .
- \mathcal{G} is 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 .
- $\mathcal{F} \subseteq \mathcal{O} \times \mathcal{O}$ is the control flow relation, i.e. a **set of sequence flows connecting objects**.

- Defining the QoS attributes involved in **service**, **network** and **message** aspects.
- QoS attributes:

- Defining the QoS attributes involved in **service**, **network** and **message** aspects.
- QoS attributes:
 - ▶ In service operation: **time to complete the service**.

- 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**.

- 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 : **message format**.

Mapping BPMN to Petri Net (I)

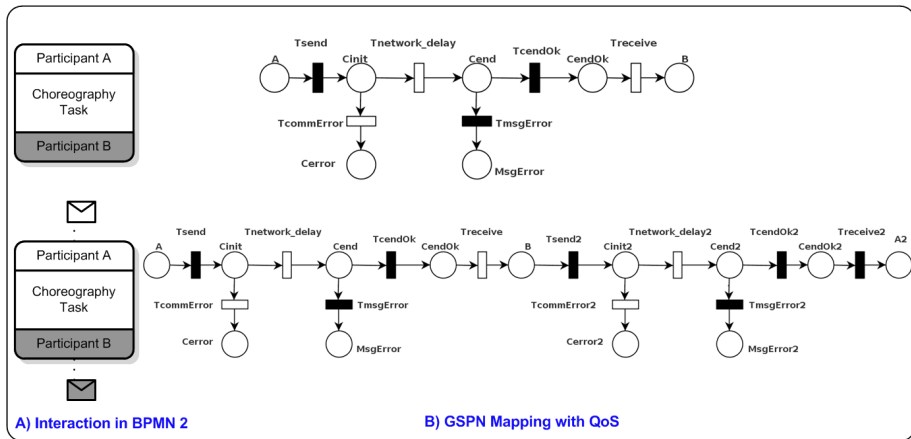
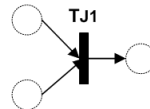
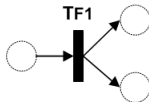
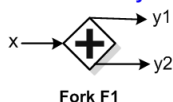


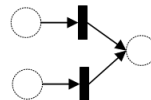
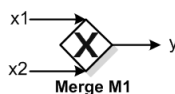
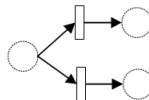
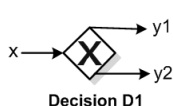
Figure: Mapping of two different choreography tasks with the QoS model

Mapping BPMN to Petri Net (II)

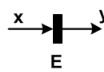
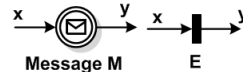
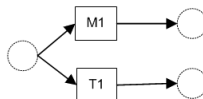
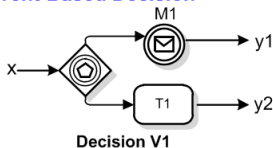
Parallel Gateway



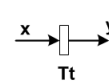
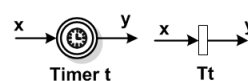
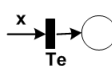
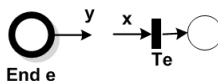
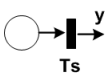
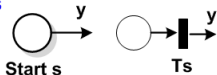
Exclusive Gateway (Data-Based)

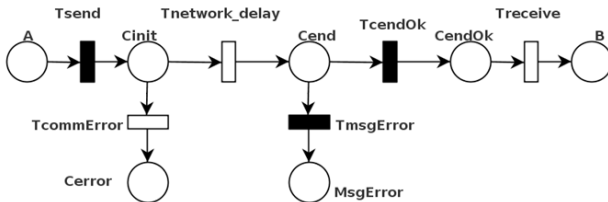
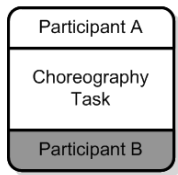


Event-Based Decision



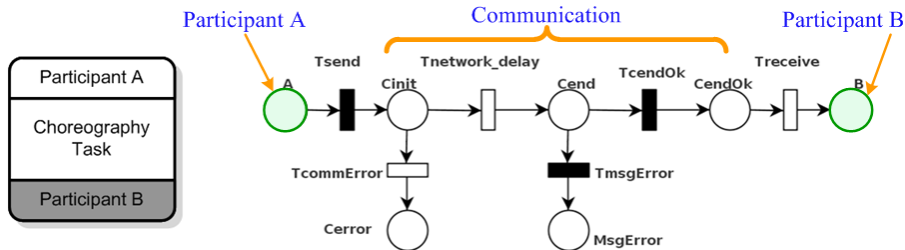
Events





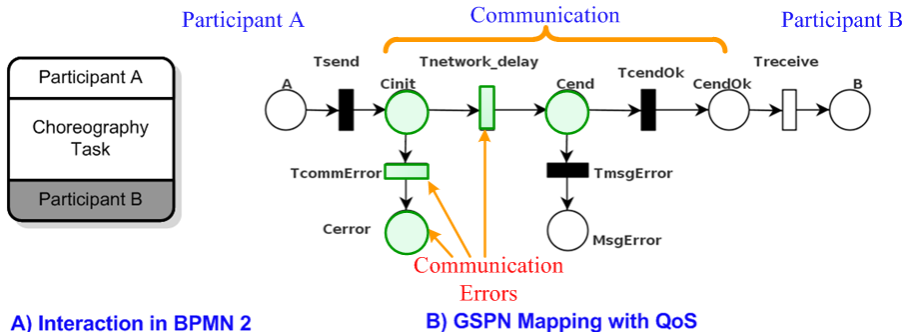
A) Interaction in BPMN 2

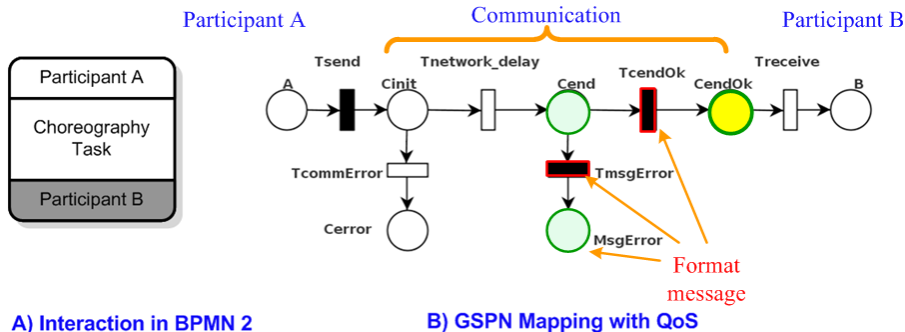
B) GSPN Mapping with QoS

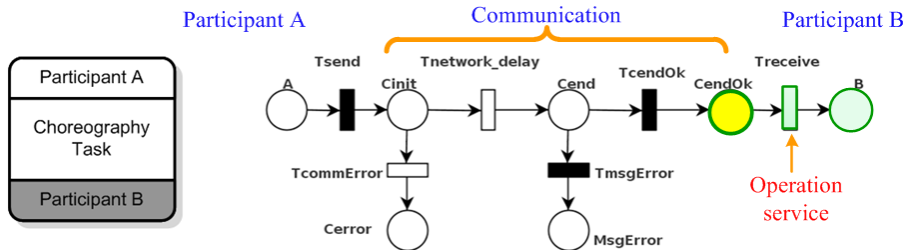


A) Interaction in BPMN 2

B) GSPN Mapping with QoS

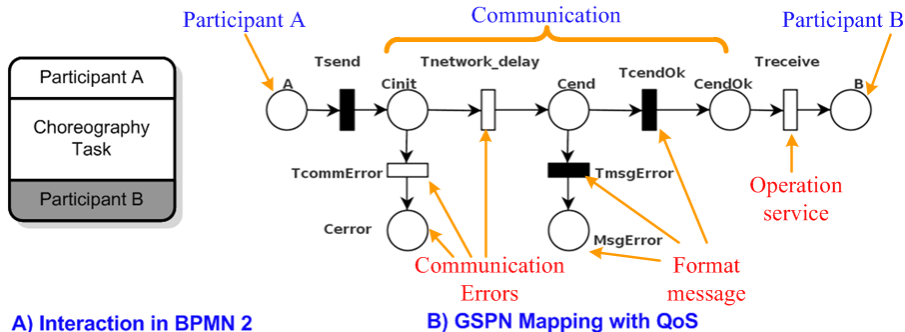






A) Interaction in BPMN 2

B) GSPN Mapping with QoS



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.

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 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 $GSPN_{QoS}$.

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

Mapping Algorithm (I)

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

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

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

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

Consider $PNQoS(CT_i)$ is a function of the type of CT_i that returns a GSPN according to mapping rules.

Mapping Algorithm (I)

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

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

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

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

Consider $PNQoS(CT_i)$ is a function of the type of CT_i that returns a GSPN according to mapping rules.

Consider $PNQoS(G_j)$ is a function of the type of G_j that returns a GSPN according to mapping rules.

Mapping Algorithm (I)

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

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

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

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

Consider $PNQoS(CT_i)$ is a function of the type of CT_i that returns a GSPN according to mapping rules.

Consider $PNQoS(G_j)$ is a function of the type of G_j that returns a GSPN according to mapping rules.

Consider $PNQoS(E_k)$ is a function of the type of E_k that returns a GSPN according to mapping rules.

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 $GSPN_{QoS}$.

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

Consider $PNQoS(CT_i)$ is a function of the type of CT_i that returns a GSPN according to mapping rules.

Consider $PNQoS(G_j)$ is a function of the type of G_j that returns a GSPN according to mapping rules.

Consider $PNQoS(E_k)$ is a function of the type of E_k that returns a GSPN according to mapping rules.

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

Mapping Algorithm (II)

Algorithm 1: (continued)

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

Mapping Algorithm (II)

Algorithm 1: (continued)

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

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

End

Mapping Algorithm (II)

Algorithm 1: (continued)

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

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

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

End

Mapping Algorithm (II)

Algorithm 1: (continued)

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

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)

Algorithm 1: (continued)

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

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)

Algorithm 1: (continued)

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

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

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

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)

Algorithm 1: (continued)

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

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

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

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)

Algorithm 1: (continued)

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

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

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

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)

Algorithm 1: (continued)

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

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

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

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)

Algorithm 1: (continued)

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

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

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

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)

Algorithm 1: (continued)

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

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

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

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 $GSPN_{QoS}$

- 1 Introduction
- 2 Problem
- 3 Methodology
- 4 Performance Evaluation**
- 5 Conclusions and Future Works

Scenario

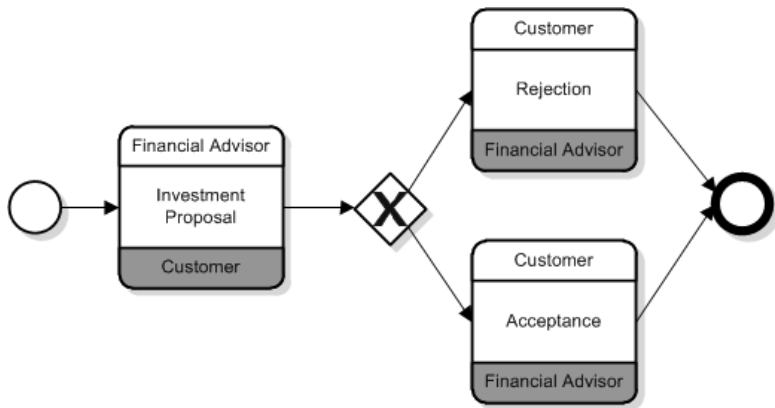


Figure: Choreography example using BPMN2 elements.

Scenario

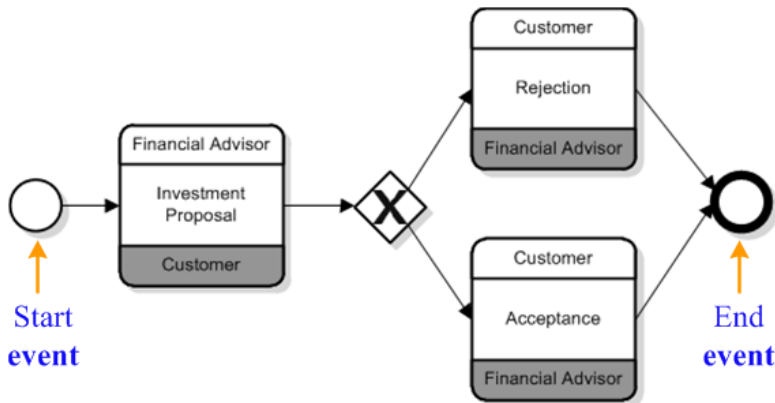


Figure: Choreography example using BPMN2 elements.

Scenario

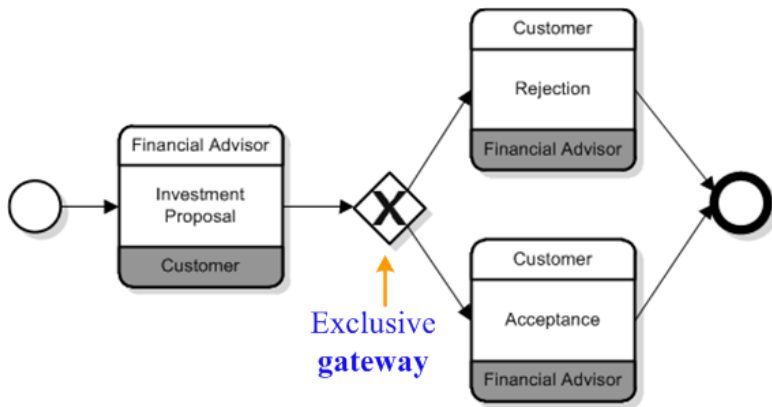


Figure: Choreography example using BPMN2 elements.

Scenario

Participant A: Financial Advisor

Participant B: Customer

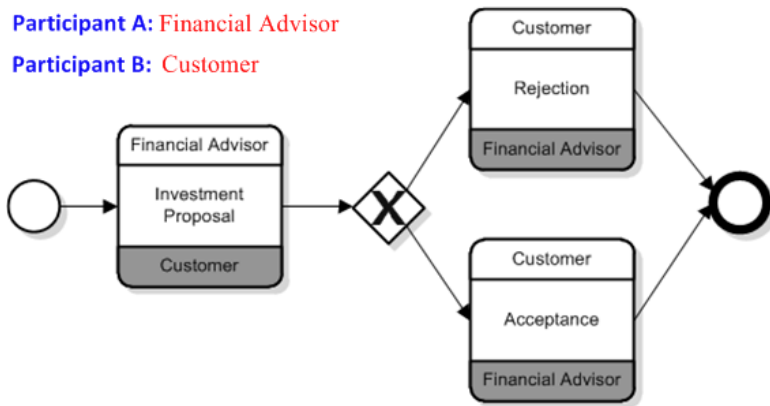


Figure: Choreography example using BPMN2 elements.

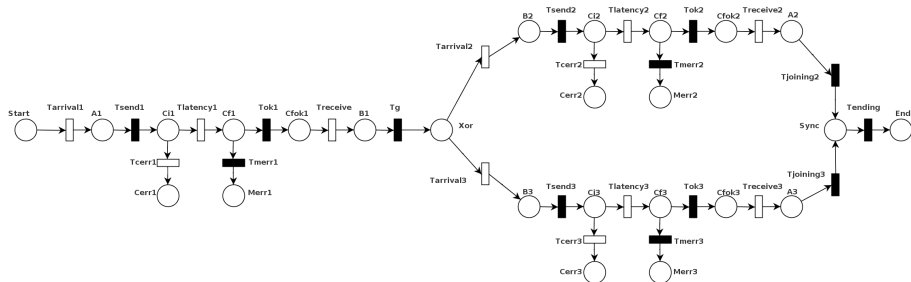


Figure: GSPN obtained from the choreography.

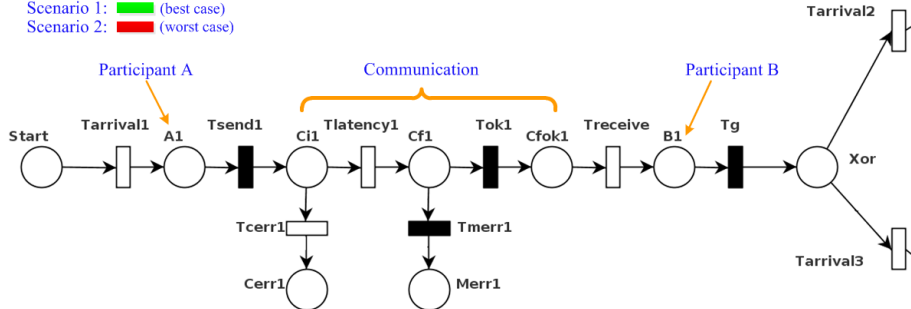
Configuration (I)

Table: Weights of Scenario 1 and Scenario 2

Transition	Weights	
	Scenario 1	Scenario 2
$T_{latency1}, T_{latency2}, T_{latency3}$	0.99	0.94
$T_{cerr1}, T_{cerr2}, T_{cerr3}$	0.01	0.06
$T_{receive}, T_{receive2}, T_{receive3}$	99	97
$T_{merr1}, T_{merr2}, T_{merr3}$	1	3
$T_{arrival2}, T_{arrival3}$	0.5	0.5

Configuration (II)

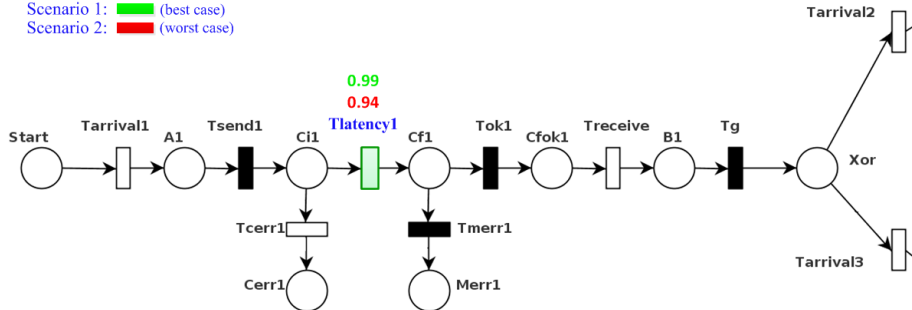
Scenario 1:  (best case)
Scenario 2:  (worst case)



Configuration (II)

Scenario 1:  (best case)

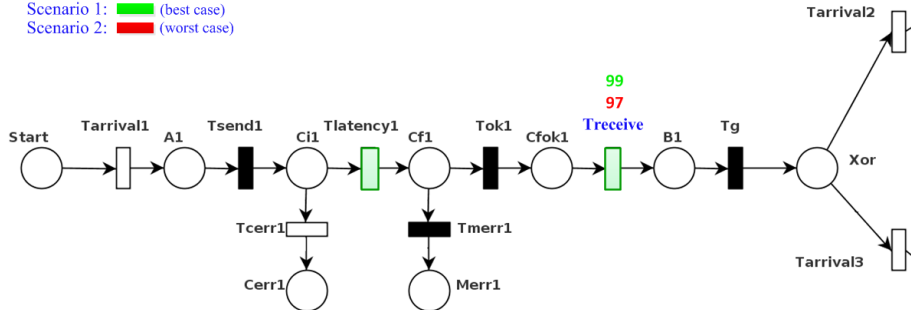
Scenario 2:  (worst case)



Configuration (II)

Scenario 1:  (best case)

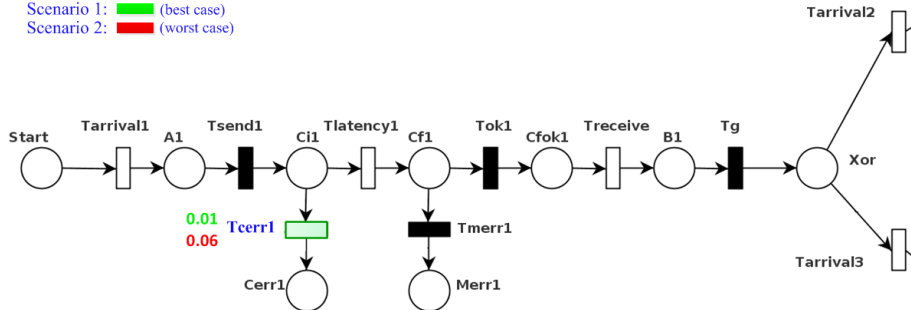
Scenario 2:  (worst case)



Configuration (II)

Scenario 1: █ (best case)

Scenario 2: █ (worst case)



- 1 **token** = 1 **choreography instance**.

- 1 **token** = 1 **choreography instance**.
- 100 **tokens** are considered to each scenario at the **place Start**.
- 100 **concurrent instances** were executed (**multiple-server semantic**).

- 1 **token** = 1 **choreography instance**.
- 100 **tokens** are considered to each scenario at the **place Start**.
- 100 **concurrent instances** were executed (**multiple-server semantic**).
- The **Pipe2** tool was used to **model** and **simulate** the **GSPN**.

- 1 **token** = 1 **choreography instance**.
- 100 **tokens** are considered to each scenario at the **place Start**.
- 100 **concurrent instances** were executed (**multiple-server semantic**).
- The **Pipe2** tool was used to **model** and **simulate** the **GSPN**.
- 1500 fires and 10 replications.
- Confidence level of 95%.

Results (I)

Table: Simulation results

Place	Average number of tokens (%)		95% Confidence interval (+/- %)	
	Scenario 1	Scenario 2	Scenario 1	Scenario 2
<i>Start</i>	35.28	40.15	5.83	6.23
<i>End</i>	41.95	38.78	2.53	3.82
M_{err1}	0.39	0.91	0.95	1.92
M_{err2}	0.00	0.93	0.63	0.64
M_{err3}	0.00	0.66	0.87	0.74
C_{err1}	0.74	2.94	0.82	2.02
C_{err2}	0.00	0.00	0.67	1.75
C_{err3}	0.78	0.16	0.92	1.52
C_{i1}	8.32	8.90	5.33	7.48
C_{i2}	0.63	0.69	0.23	0.52
C_{i3}	0.75	8.90	0.39	0.21

Results (II)

- **Communication errors:** An average of $C_{err1} + C_{err2} + C_{err3}$ of instances **didn't finish the process**.
 - ▶ Scenario 1: 1.52%.
 - ▶ Scenario 2: 3.10% (more errors).

Results (II)

- **Communication errors:** An average of $C_{err1} + C_{err2} + C_{err3}$ of instances **didn't finish the process**.
 - ▶ Scenario 1: 1.52%.
 - ▶ Scenario 2: 3.10% (more errors).
- **Invalid format message:** An average of $M_{err1} + M_{err2} + M_{err3}$ of instances **didn't finish the process**.
 - ▶ Scenario 1: 0.39%.
 - ▶ Scenario 2: 2.50% (more invalid messages).

Results (II)

- **Communication errors:** An average of $C_{err1} + C_{err2} + C_{err3}$ of instances **didn't finish the process**.
 - ▶ Scenario 1: 1.52%.
 - ▶ Scenario 2: 3.10% (more errors).
- **Invalid format message:** An average of $M_{err1} + M_{err2} + M_{err3}$ of instances **didn't finish the process**.
 - ▶ Scenario 1: 0.39%.
 - ▶ Scenario 2: 2.50% (more invalid messages).
- **Bottleneck:** It was found a communication bottleneck in the first interaction (C_i place).
 - ▶ Scenario 1: 8.32%.
 - ▶ Scenario 2: 8.90%.

- 1 Introduction
- 2 Problem
- 3 Methodology
- 4 Performance Evaluation
- 5 Conclusions and Future Works

Conclusions

- We have proposed a Novel methodology to define QoS and SLA requirements in service Choreography.

Conclusions

- We have proposed a Novel methodology to define QoS and SLA requirements in service Choreography.
- It's a initial approach using the “interaction model” (supported by BPMN 2.0).

Conclusions

- We have proposed a Novel methodology to define QoS and SLA requirements in service Choreography.
- It's a initial approach using the “interaction model” (supported by BPMN 2.0).
- The GSPN is good to model and analyze several aspects involved into service choreography.

Conclusions

- We have proposed a Novel methodology to define QoS and SLA requirements in service Choreography.
- It's a initial approach using the “interaction model” (supported by BPMN 2.0).
- The GSPN is good to model and analyze several aspects involved into service choreography.
- The simulation is needed for supporting analysis of complex process (e.g. process choreography).

Conclusions

- We have proposed a Novel methodology to define QoS and SLA requirements in service Choreography.
- It's a initial approach using the “interaction model” (supported by BPMN 2.0).
- The GSPN is good to model and analyze several aspects involved into service choreography.
- The simulation is needed for supporting analysis of complex process (e.g. process choreography).
- The simulation results can be used to establish early QoS and SLA constraints.
 - ▶ Integration is expensive, then early detections are needed.
 - ▶ Establishing SLAs according to resources.
 - ▶ Planning in order to reduce failures.
 - ▶ For example: the detected bottleneck can be solved by changing the interaction (modeling issues) or by employing QoS mechanisms in the network to prioritize the traffic affected (resource planning).

- To extend the mapping to support more choreography BPMN elements.

- To extend the mapping to support more choreography BPMN elements.
- To expand our methodology to support generic probability distributions in the decision points. Use of Colored Petri Nets (CPNs) can be a alternative.

- To extend the mapping to support more choreography BPMN elements.
- To expand our methodology to support generic probability distributions in the decision points. Use of Colored Petri Nets (CPNs) can be a alternative.
- To make more analysis and to use complex scenarios, where correlation problems could happen.

- To extend the mapping to support more choreography BPMN elements.
- To expand our methodology to support generic probability distributions in the decision points. Use of Colored Petri Nets (CPNs) can be a alternative.
- To make more analysis and to use complex scenarios, where correlation problems could happen.
- To include more QoS attributes.

Thanks!