Analysis, Optimisation and Debugging of BPMN Processes

PhD Defended by Quentin NIVON before a jury composed of:

- Pr. Olivier BARAIS, Examiner
- Pr. Remco DIJKMAN, Examiner
- Pr. Massimo MECELLA, Reviewer
- Pr. Pascal POIZAT, Reviewer
- Pr. Claudia RONCANCIO, Examiner
- Pr. Gwen SALAÜN, Supervisor

A Little Bit of History: BPMN

BPMN stands for **Business Process Model and Notation**.
But what is a business process?

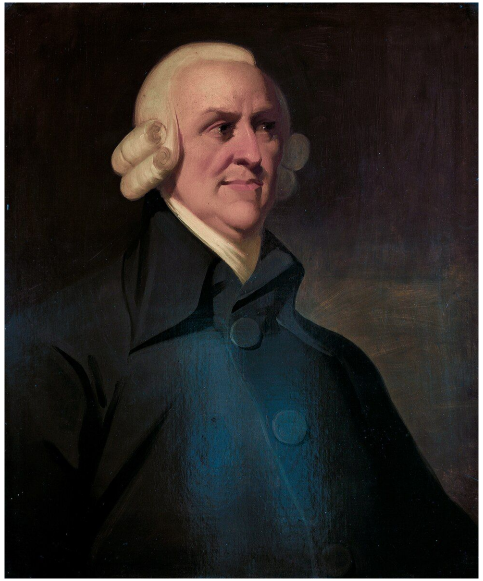
A Little Bit of History: BPMN

BPMN stands for **Business Process Model and Notation**.
But what is a business process?

“*A business process [...] is a collection of related, structured activities or tasks performed by people or equipment in which a specific sequence produces a service or product (that serves a particular business goal) for a particular customer or customers*”

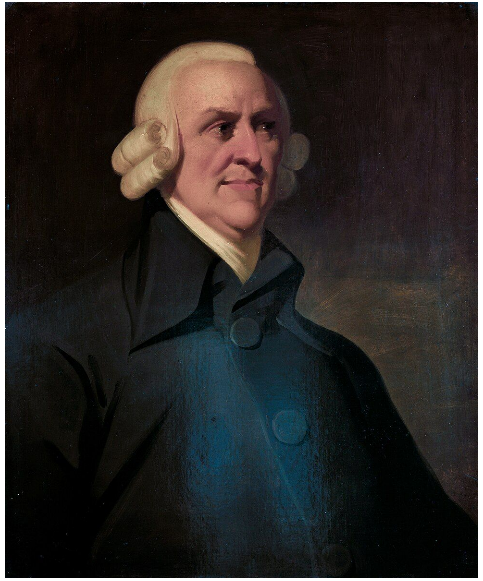
A Little Bit of History: Adam Smith

According to history, the **first** man to have ever evokated the term “business process” is the scottish economist Adam Smith in 1776.


Adam Smith

A Little Bit of History: Adam Smith

According to history, the **first** man to have ever evokated the term "business process" is the scottish economist Adam Smith in 1776.


Adam Smith

In [Smith1776], he described the production of a pin as follows:

"One man draws out the wire; another straights it; a third cuts it; a fourth points it; a fifth grinds it at the top for receiving the head; to make the head requires two or three distinct operations; to put it on is a peculiar business; to whiten the pins is another ... and the important business of making a pin is, in this manner, divided into about eighteen distinct operations, which, in some manufactories, are all performed by distinct hands, though in others the same man will sometimes perform two or three of them."

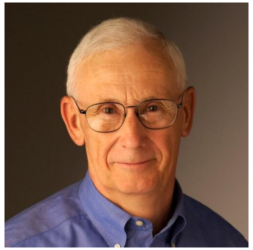
A Little Bit of History: …and His Successors

Frederick Winslow Taylor

- ➢ standardization of processes
- ➢ systematic training
- ➢ clear definition of the roles of management and employees

A Little Bit of History: …and His Successors

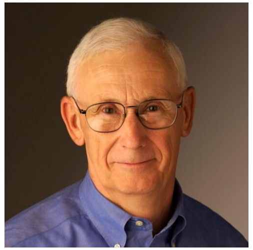
Frederick Winslow Taylor


Geary A. Rummler

- ➢ standardization of processes
- ➢ systematic training
- ➢ clear definition of the roles of management and employees

A Little Bit of History: …and His Successors

Frederick Winslow Taylor


Geary A. Rummler

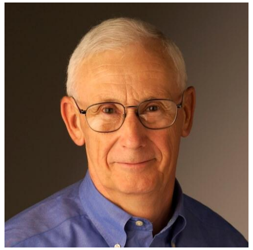
Michael Hammer

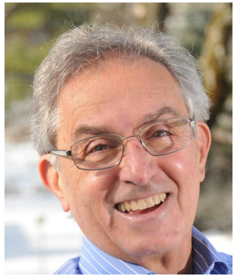
- ➢ standardization of processes
- ➢ systematic training
- ➢ clear definition of the roles of management and employees

A Little Bit of History: …and His Successors

Frederick Winslow Taylor

- ➢ standardization of processes
- ➢ systematic training
- ➢ clear definition of the roles of management and employees


Geary A. Rummler

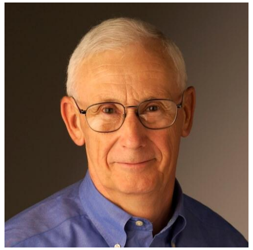

James Champy

Michael Hammer

A Little Bit of History: …and His Successors

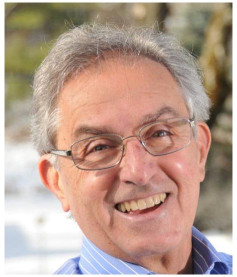
Frederick Winslow Taylor

- ➢ standardization of processes
- ➢ systematic training
- ➢ clear definition of the roles of management and employees


Geary A. Rummler

Thomas H. Davenport

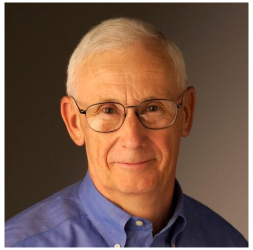
Michael Hammer


James Champy

A Little Bit of History: …and His Successors

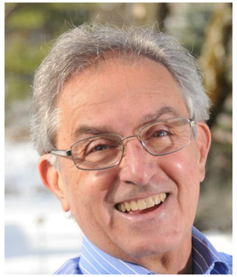
Frederick Winslow Taylor

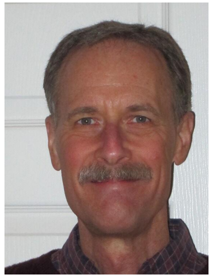
- ➢ standardization of processes
- ➢ systematic training
- ➢ clear definition of the roles of management and employees


Geary A. Rummler

Thomas H. Davenport

Michael Hammer


James Champy


Alan P. Brache

and others

A Little Bit of History: Business Process Management

This desire to provide a **rigorous**, **unified** definition of business processes paved the way to the creation of a new discipline: the **business process management**.

A Little Bit of History: Business Process Management

This desire to provide a **rigorous**, **unified** definition of business processes paved the way to the creation of a new discipline: the **business process management**.

This holistic discipline encompasses all the fields related to business processes, such as:

A Little Bit of History: Business Process Management

This desire to provide a **rigorous**, **unified** definition of business processes paved the way to the creation of a new discipline: the **business process management**.

This holistic discipline encompasses all the fields related to business processes, such as:

Process optimisation

A Little Bit of History: Business Process Management

This desire to provide a **rigorous**, **unified** definition of business processes paved the way to the creation of a new discipline: the **business process management**.

This holistic discipline encompasses all the fields related to business processes, such as:

Process discovery

Process optimisation

A Little Bit of History: Business Process Management

This desire to provide a **rigorous**, **unified** definition of business processes paved the way to the creation of a new discipline: the **business process management**.

This holistic discipline encompasses all the fields related to business processes, such as:

Process discovery

Process measurement

Process optimisation

A Little Bit of History: Business Process Management

This desire to provide a **rigorous**, **unified** definition of business processes paved the way to the creation of a new discipline: the **business process management**.

This holistic discipline encompasses all the fields related to business processes, such as:

Process discovery

Process measurement

Process optimisation

Process automation

A Little Bit of History: Business Process Management

This desire to provide a **rigorous**, **unified** definition of business processes paved the way to the creation of a new discipline: the **business process management**.

This holistic discipline encompasses all the fields related to business processes, such as:

Process discovery

Process analysis

Process measurement

Process optimisation

Process automation

A Little Bit of History: Business Process Management

This desire to provide a **rigorous**, **unified** definition of business processes paved the way to the creation of a new discipline: the **business process management**.

This holistic discipline encompasses all the fields related to business processes, such as:

Process discovery

Process analysis

Process measurement

Process modelling

Process optimisation

Process automation

A Little Bit of History: Business Process Management

This desire to provide a **rigorous**, **unified** definition of business processes paved the way to the creation of a new discipline: the **business process management**.

This holistic discipline encompasses all the fields related to business processes, such as:

- Process discovery
- Process analysis
- Process improvement
- Process measurement
- Process modelling
- Process optimisation
- Process automation

A Little Bit of History: Business Process Management

This desire to provide a **rigorous**, **unified** definition of business processes paved the way to the creation of a new discipline: the **business process management**.

This holistic discipline encompasses all the fields related to business processes, such as:

Process discovery

Process analysis

Process improvement

Process measurement

Process modelling

Process optimisation

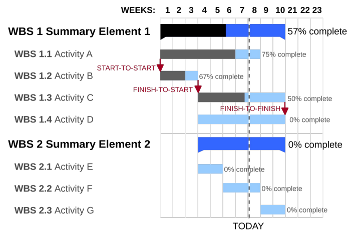
Process automation

A Little Bit of History: How to Model a Process?

The term **business process modelling** was coined in the 1960s by Stanley Williams, but people were interested in modelling processes **years before**.

A Little Bit of History: How to Model a Process?

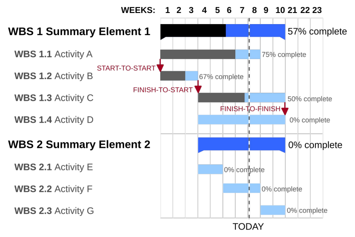
The term **business process modelling** was coined in the 1960s by Stanley Williams, but people were interested in modelling processes **years before**.



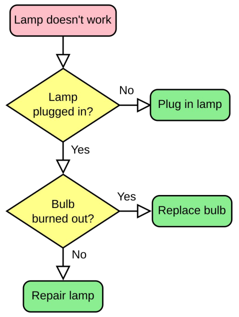
Gantt chart, 1910-15

A Little Bit of History: How to Model a Process?

The term **business process modelling** was coined in the 1960s by Stanley Williams, but people were interested in modelling processes **years before**.



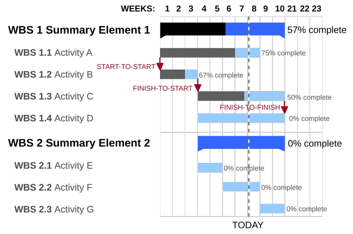
Gantt chart, 1910-15



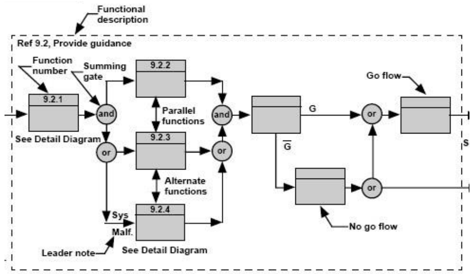
Flowchart, 1921

A Little Bit of History: How to Model a Process?

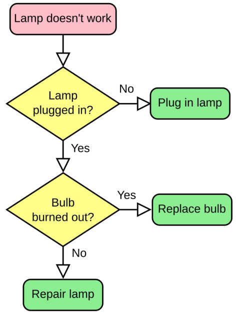
The term **business process modelling** was coined in the 1960s by Stanley Williams, but people were interested in modelling processes **years before**.



Gantt chart, 1910-15



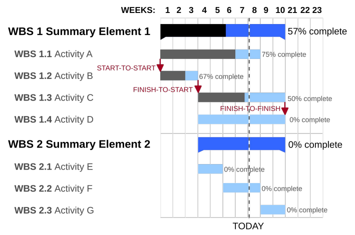
Functional flow block diagram (FFBD), 195X



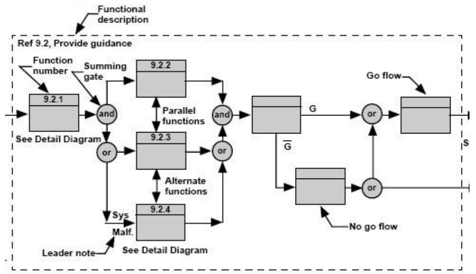
Flowchart, 1921

A Little Bit of History: How to Model a Process?

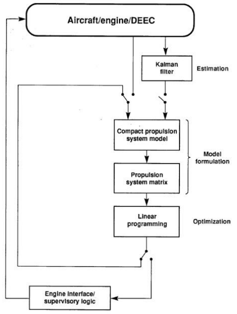
The term **business process modelling** was coined in the 1960s by Stanley Williams, but people were interested in modelling processes **years before**.



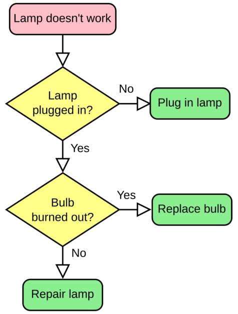
Gantt chart, 1910-15



Functional flow block diagram (FFBD), 195X



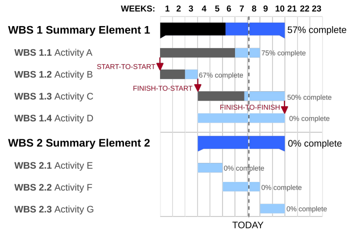
Control-flow diagram (CFD), 195X



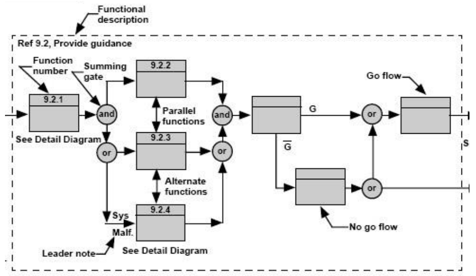
Flowchart, 1921

A Little Bit of History: How to Model a Process?

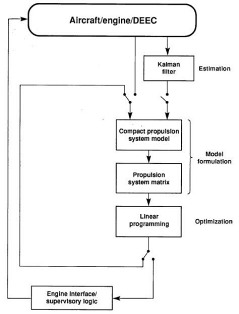
The term **business process modelling** was coined in the 1960s by Stanley Williams, but people were interested in modelling processes **years before**.



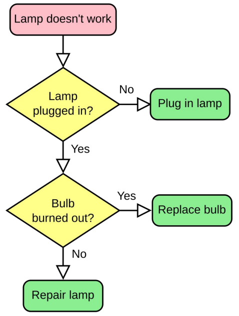
Gantt chart, 1910-15



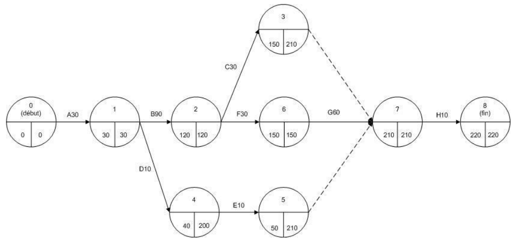
Functional flow block diagram (FFBD), 195X



Control-flow diagram (CFD), 195X



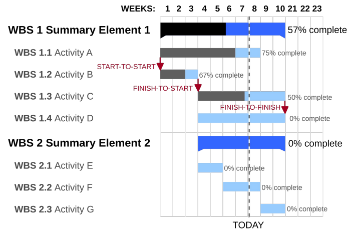
Flowchart, 1921



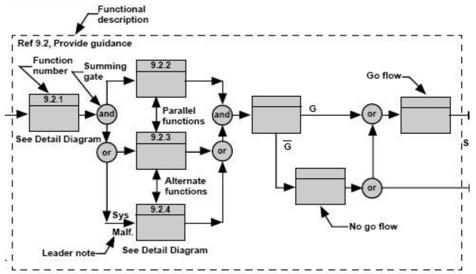
PERT diagram, 195X

A Little Bit of History: How to Model a Process?

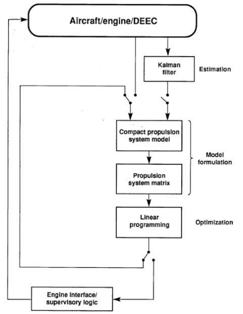
The term **business process modelling** was coined in the 1960s by Stanley Williams, but people were interested in modelling processes **years before**.



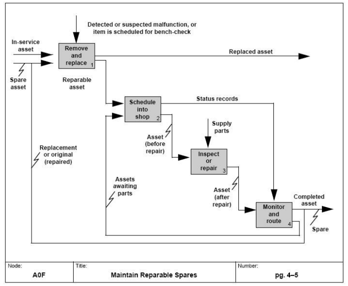
Gantt chart, 1910-15



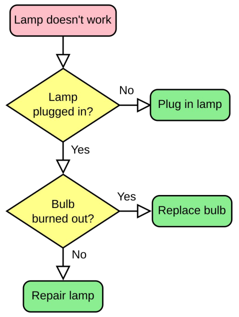
Functional flow block diagram (FFBD), 195X



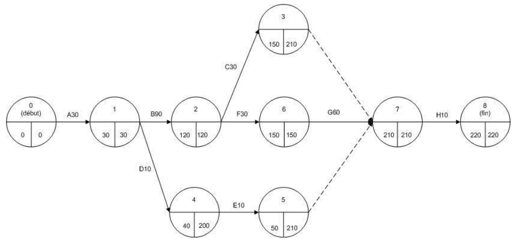
Control-flow diagram (CFD), 195X



IDEF diagram, 197X



Flowchart, 1921



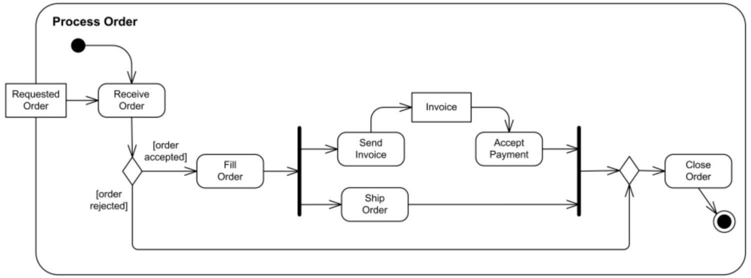
PERT diagram, 195X

A Little Bit of History: UML or BPMN?

More recently, other notations emerged, such as the **Unified Modelling Language (UML)** [BRJ2000] and the **Business Process Management Notation (BPMN)** [OMG2011].

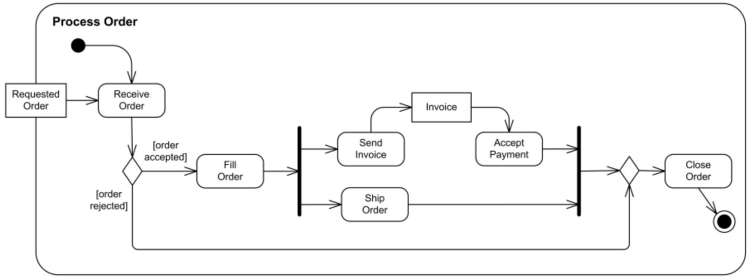
A Little Bit of History: UML or BPMN?

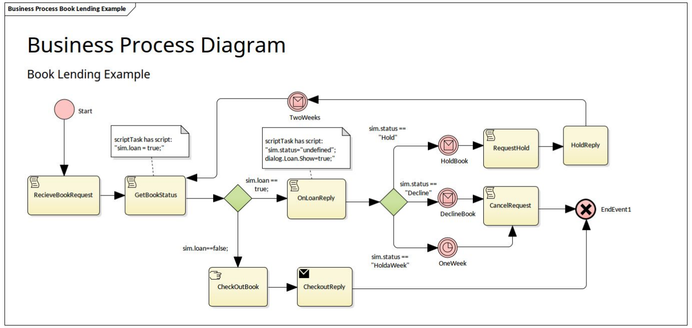
More recently, other notations emerged, such as the **Unified Modelling Language (UML)** [BRJ2000] and the **Business Process Management Notation (BPMN)** [OMG2011].



A Little Bit of History: UML or BPMN?

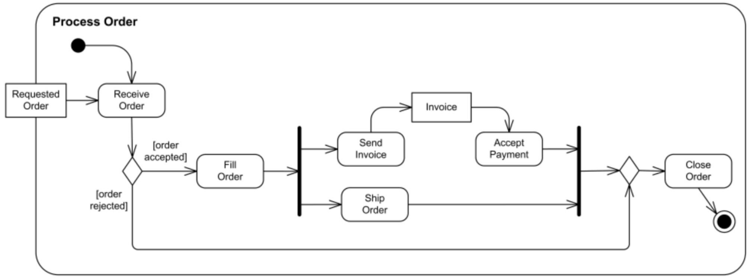
More recently, other notations emerged, such as the **Unified Modelling Language (UML)** [BRJ2000] and the **Business Process Management Notation (BPMN)** [OMG2011].

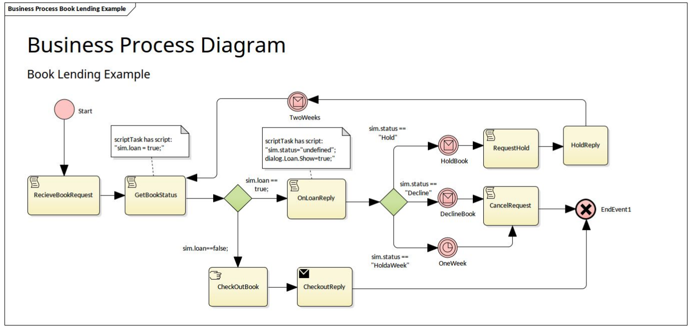




A Little Bit of History: UML or BPMN?

More recently, other notations emerged, such as the **Unified Modelling Language (UML)** [BRJ2000] and the **Business Process Management Notation (BPMN)** [OMG2011].

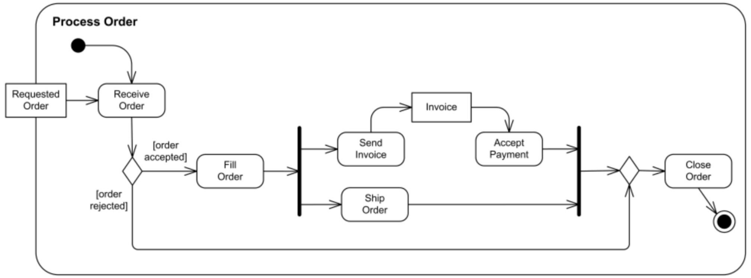


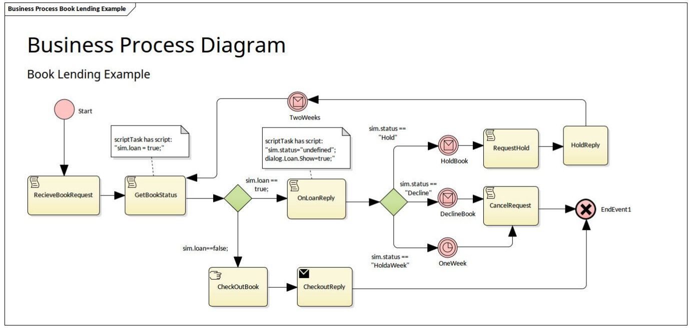


Due to their **completeness** and **understandability**, both notations rapidly became widely used **worldwide standards**.

A Little Bit of History: UML or BPMN?

More recently, other notations emerged, such as the **Unified Modelling Language (UML)** [BRJ2000] and the **Business Process Management Notation (BPMN)** [OMG2011].

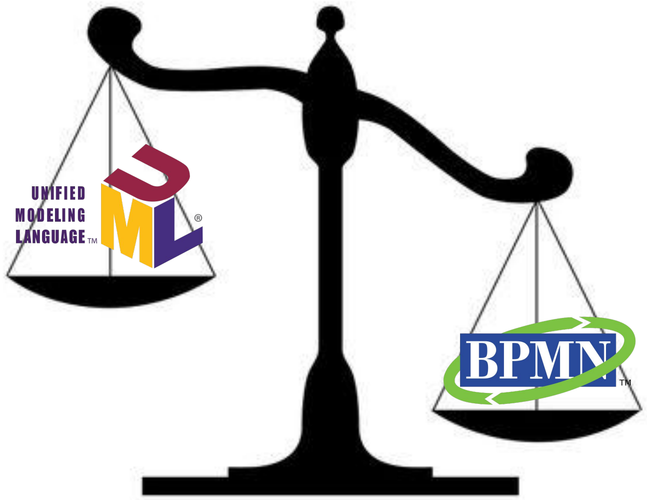




Due to their **completeness** and **understandability**, both notations rapidly became widely used **worldwide standards**.

A Little Bit of History: UML or BPMN?

Some research suggested that **BPMN was more suitable** than UML to represent business processes [White2004, NK2006, Weske2007].



A Little Bit of History: UML or BPMN?

Although being **refuted** afterwards [BKO2010, Geambasu2012], the seed was planted, and **many companies** and institutions started making **use** of the **BPMN** notation to represent their business processes.

A Little Bit of History: UML or BPMN?

Although being **refuted** afterwards [BKO2010, Geambasu2012], the seed was planted, and **many companies** and institutions started making **use** of the **BPMN** notation to represent their business processes.

But what is BPMN?

➢ A **workflow-based notation** created in 2004 by the Business Process Management Initiative (BPMI) and the Object Management Group (OMG).

But what is BPMN?

➢ A **workflow-based notation** created in 2004 by the Business Process Management Initiative (BPMI) and the Object Management Group (OMG).

➢ It aims at **representing business processes** in a way that is **understandable for both experienced and novice users.**

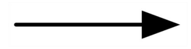
But what is BPMN?

➢ A **workflow-based notation** created in 2004 by the Business Process Management Initiative (BPMI) and the Object Management Group (OMG).

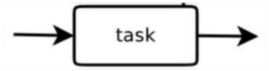
➢ It aims at **representing business processes** in a way that is **understandable for both experienced and novice users.**

➢ An **ISO/IEC standard** since version 2.0 in 2013.

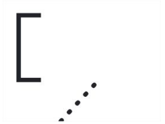
Excerpt of the BPMN Syntax



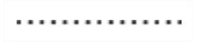
Sequence Flow



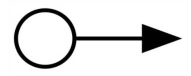
Task



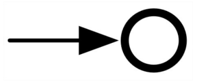
Annotation



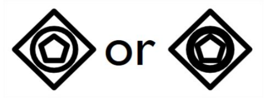
Association



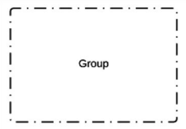
Initial Event



End Event

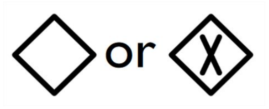


Event-based Gateway

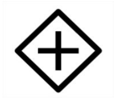


Group

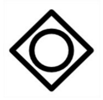
etc.



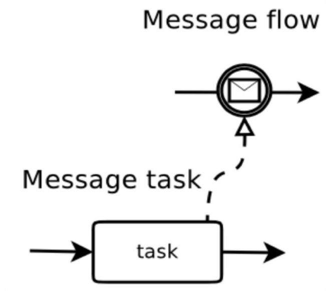
Exclusive Gateway



Parallel Gateway

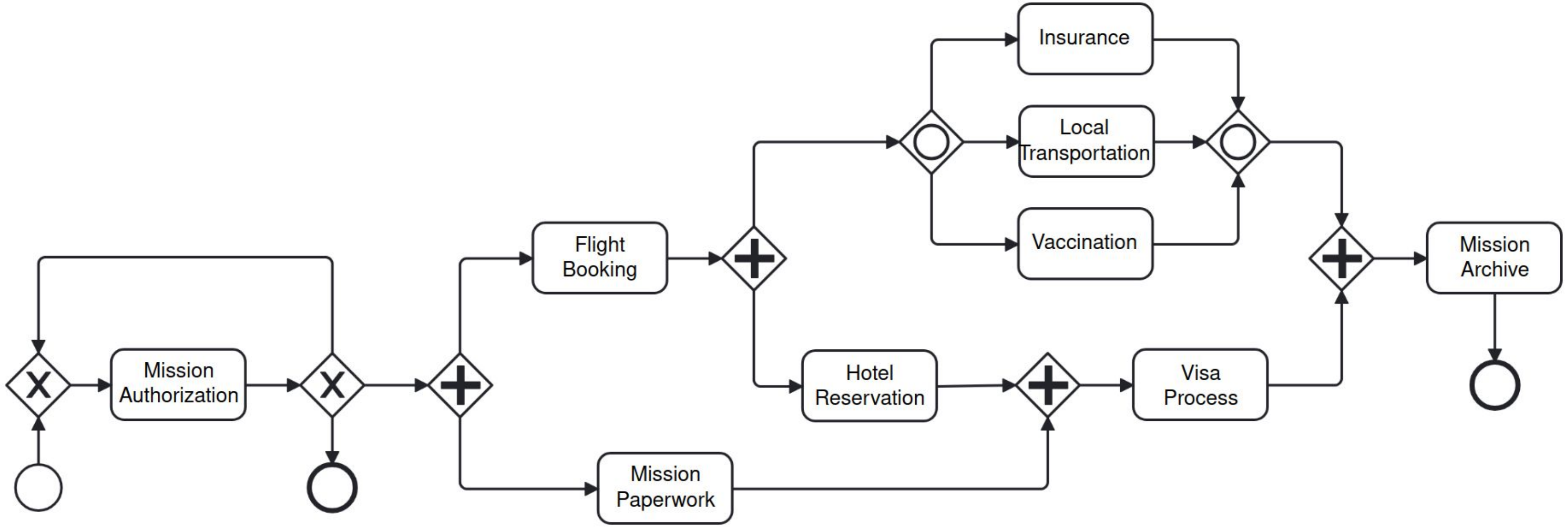


Inclusive Gateway

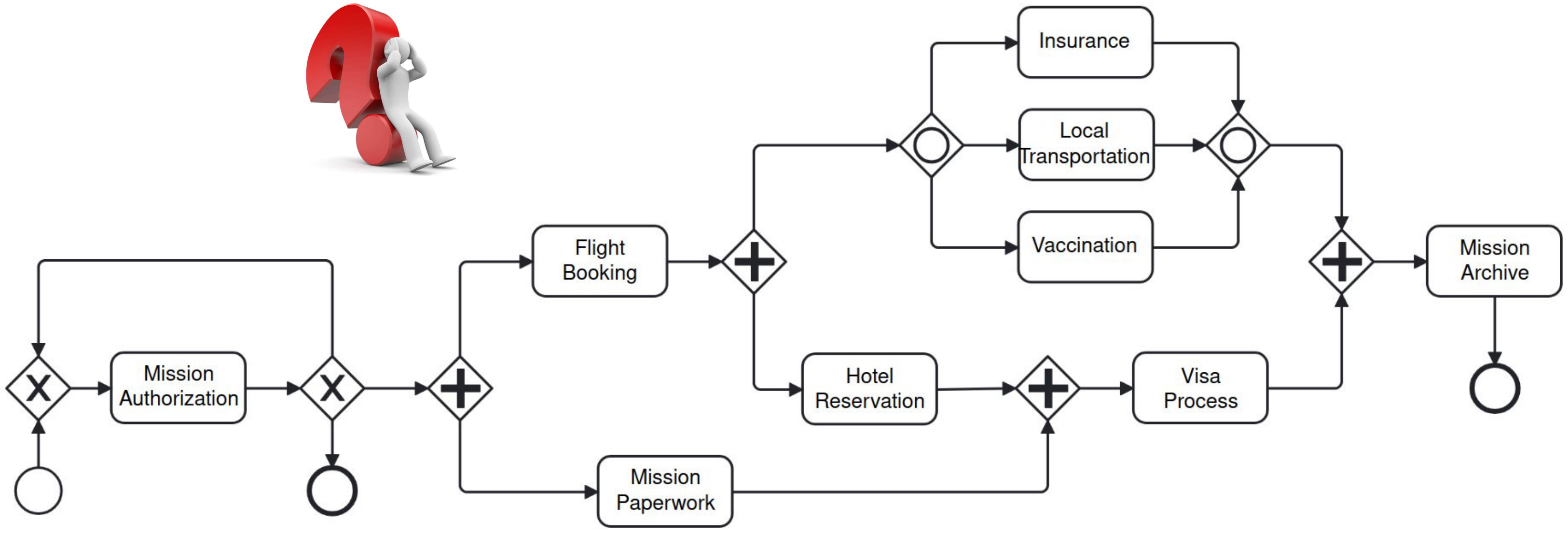


Example of BPMN Process

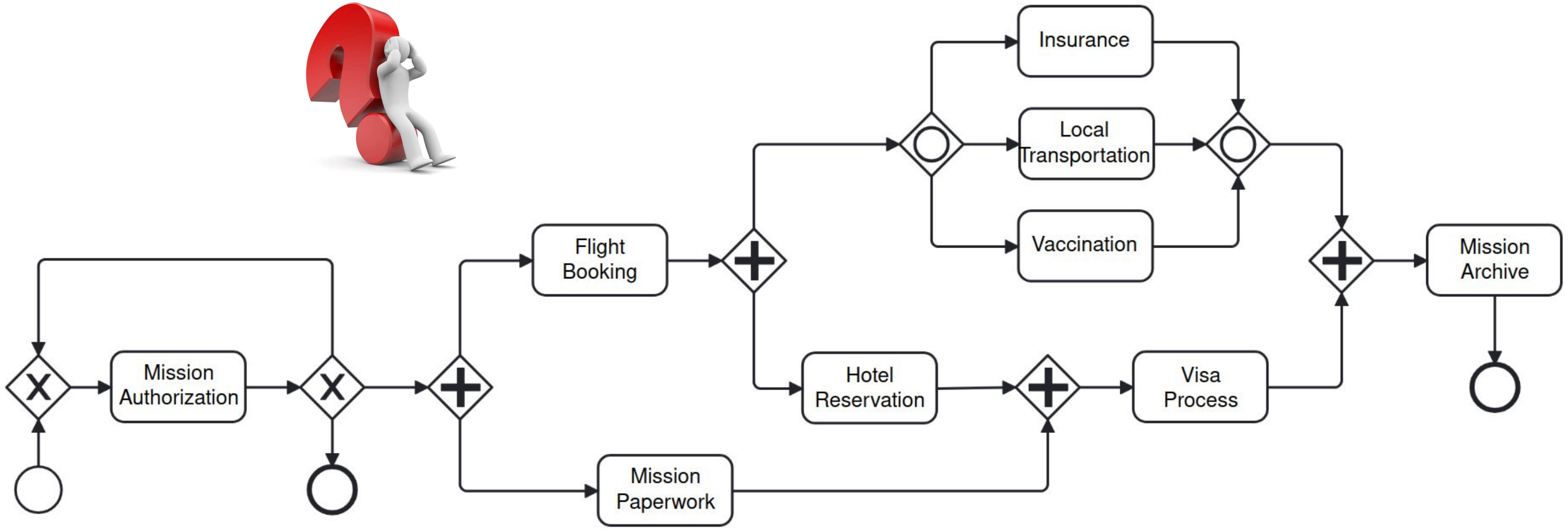
Given the BPMN syntax, one can, for instance, write a **business trip organization** process as follows:



First Research Question

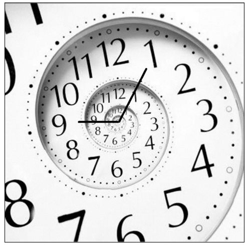


First Research Question

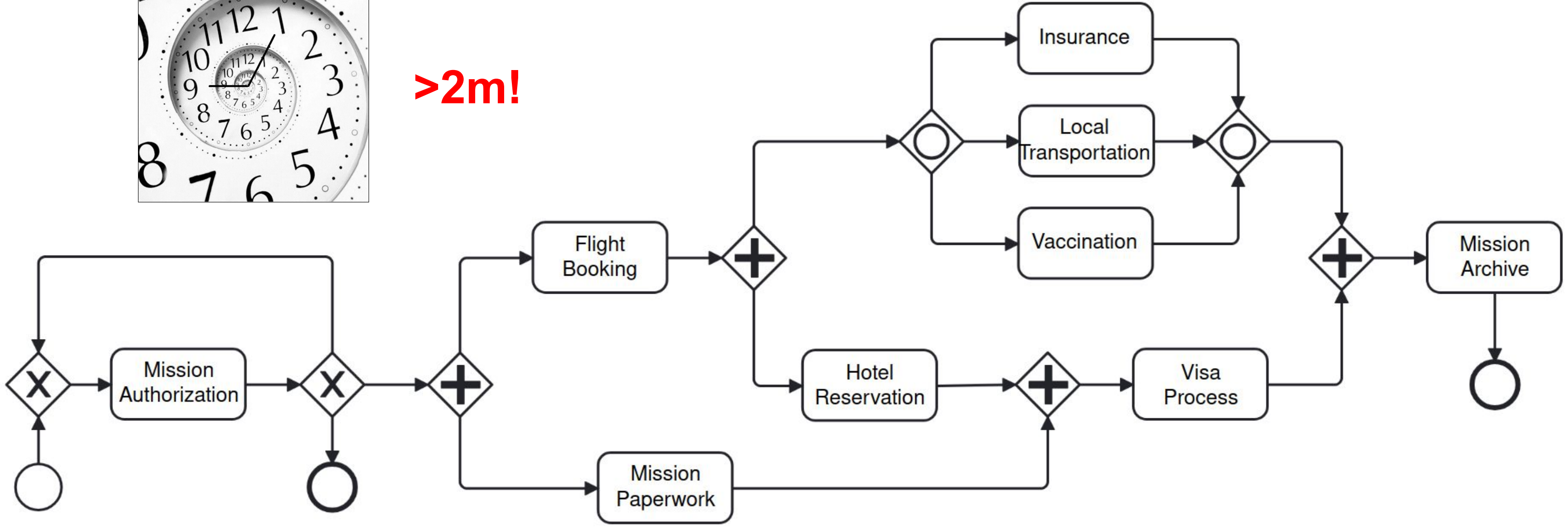


➢ What if you do not know **how to write BPMN** processes?

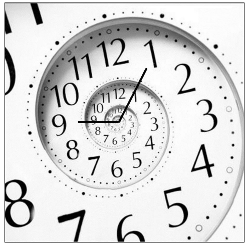
Second Research Question



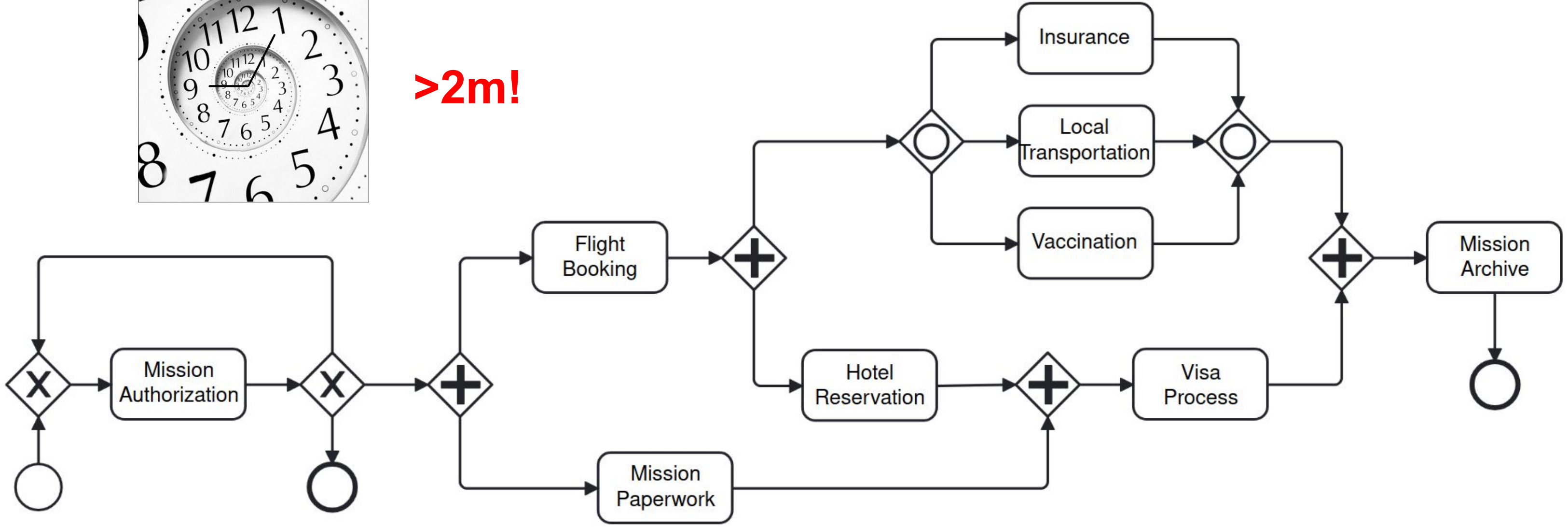
>2m!



Second Research Question



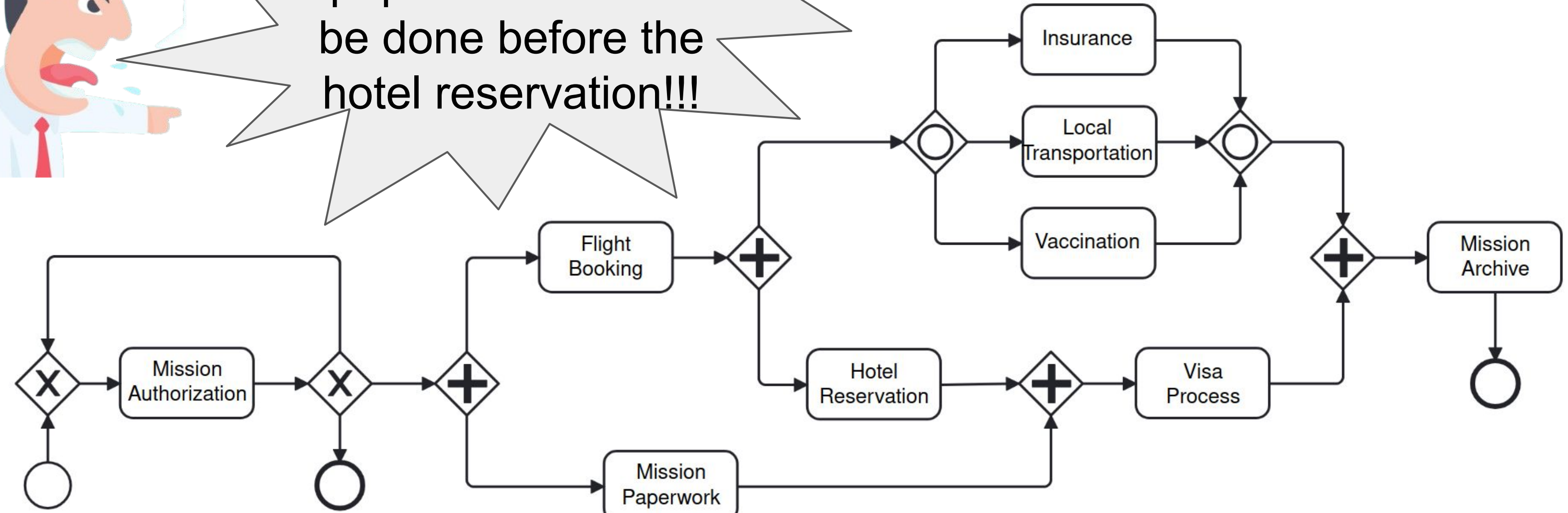
>2m!



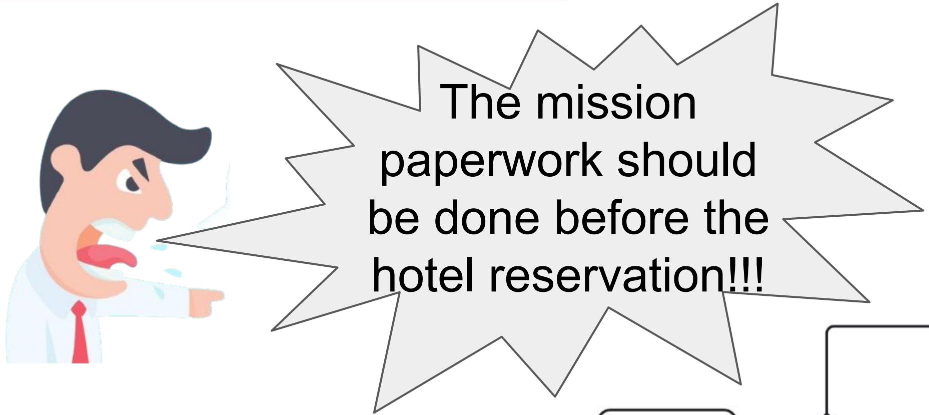
➢ What if you do not want to **spend time designing** your process graphically?

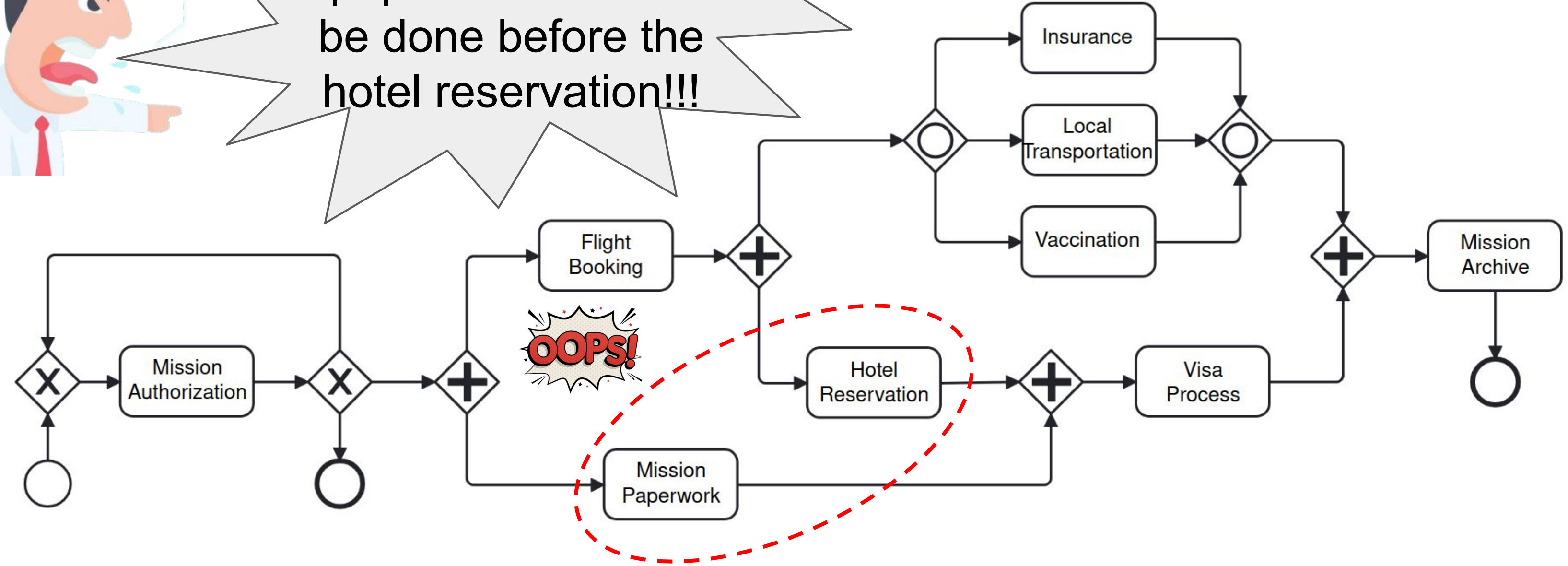
Third Research Question

The mission paperwork should be done before the hotel reservation!!!

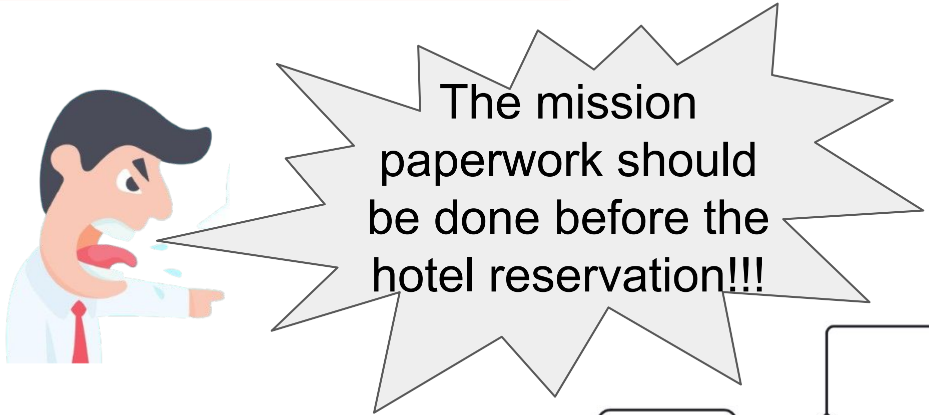


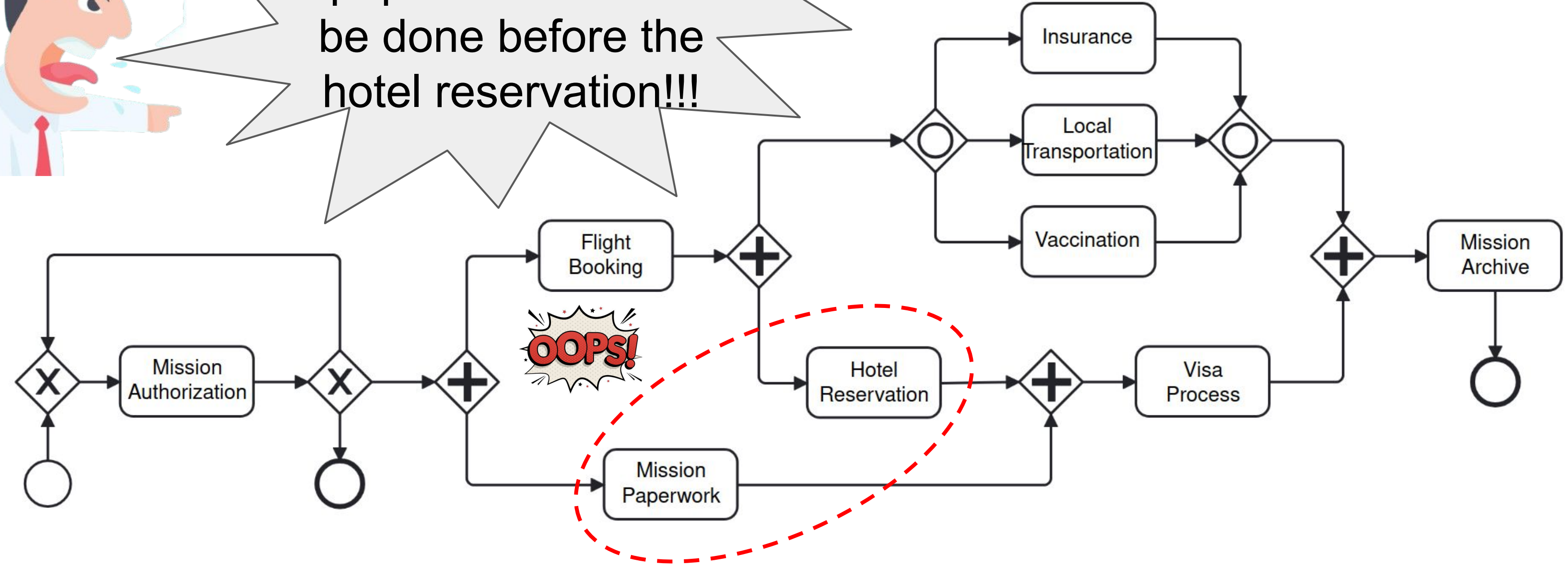
Third Research Question


The mission paperwork should be done before the hotel reservation!!!



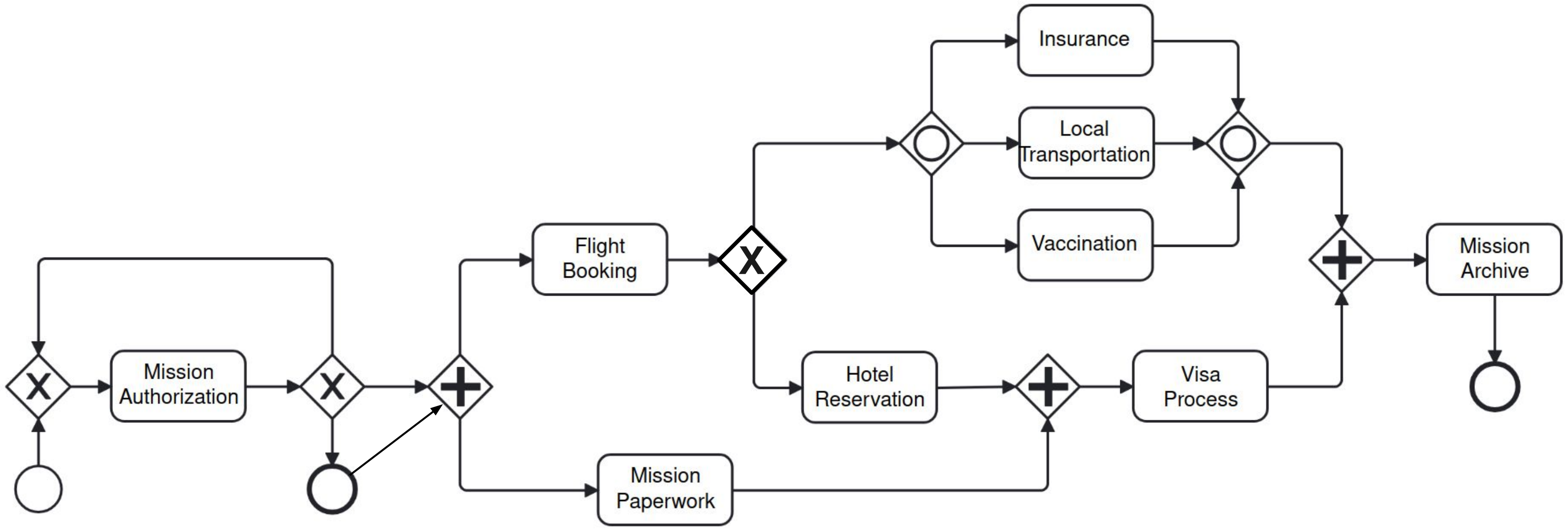
Third Research Question


The mission paperwork should be done before the hotel reservation!!!

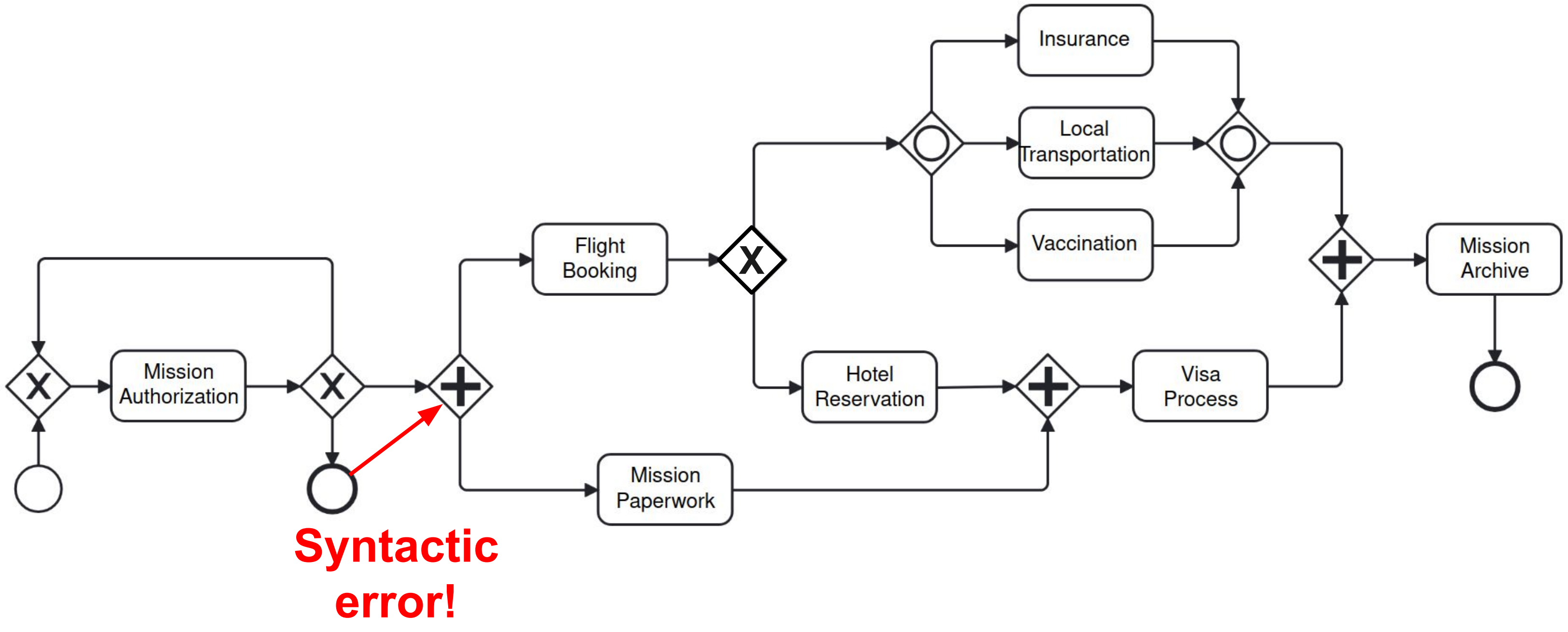


➢ How can you be sure that your BPMN process **matches its expected behaviour**?

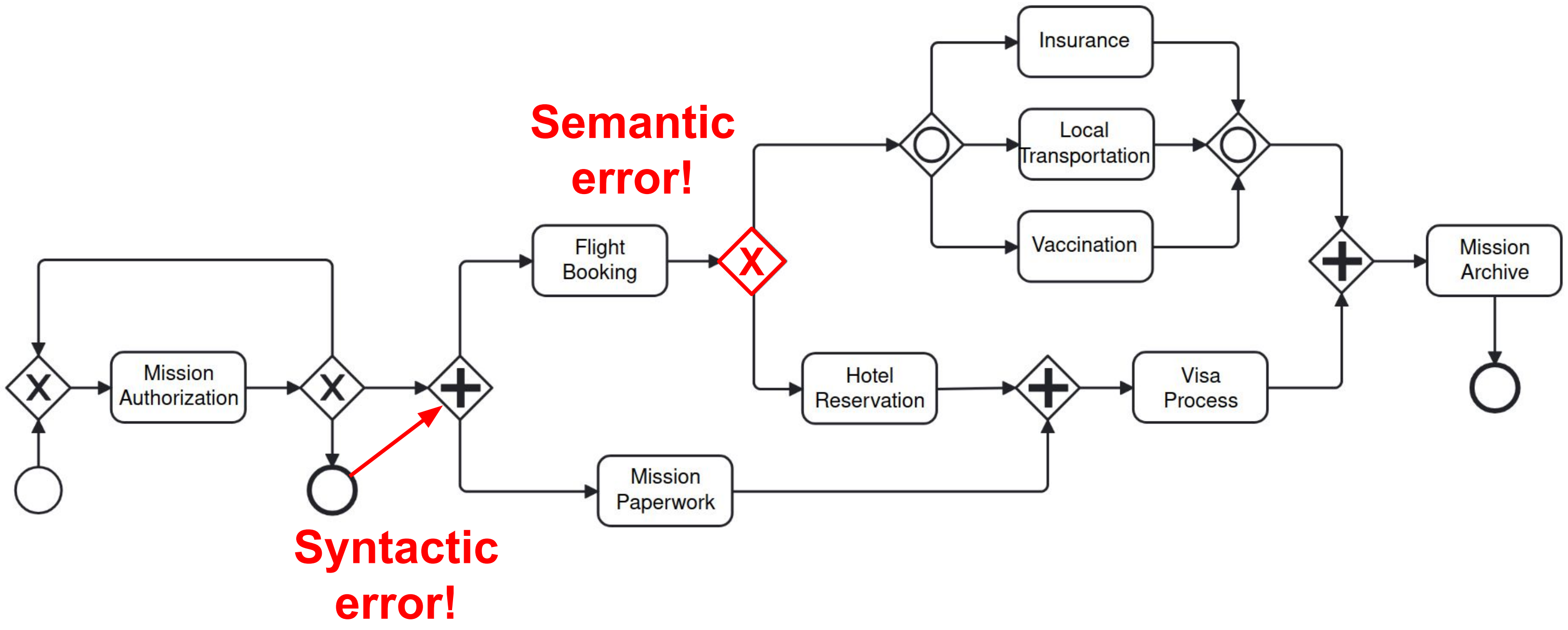
Fourth Research Question



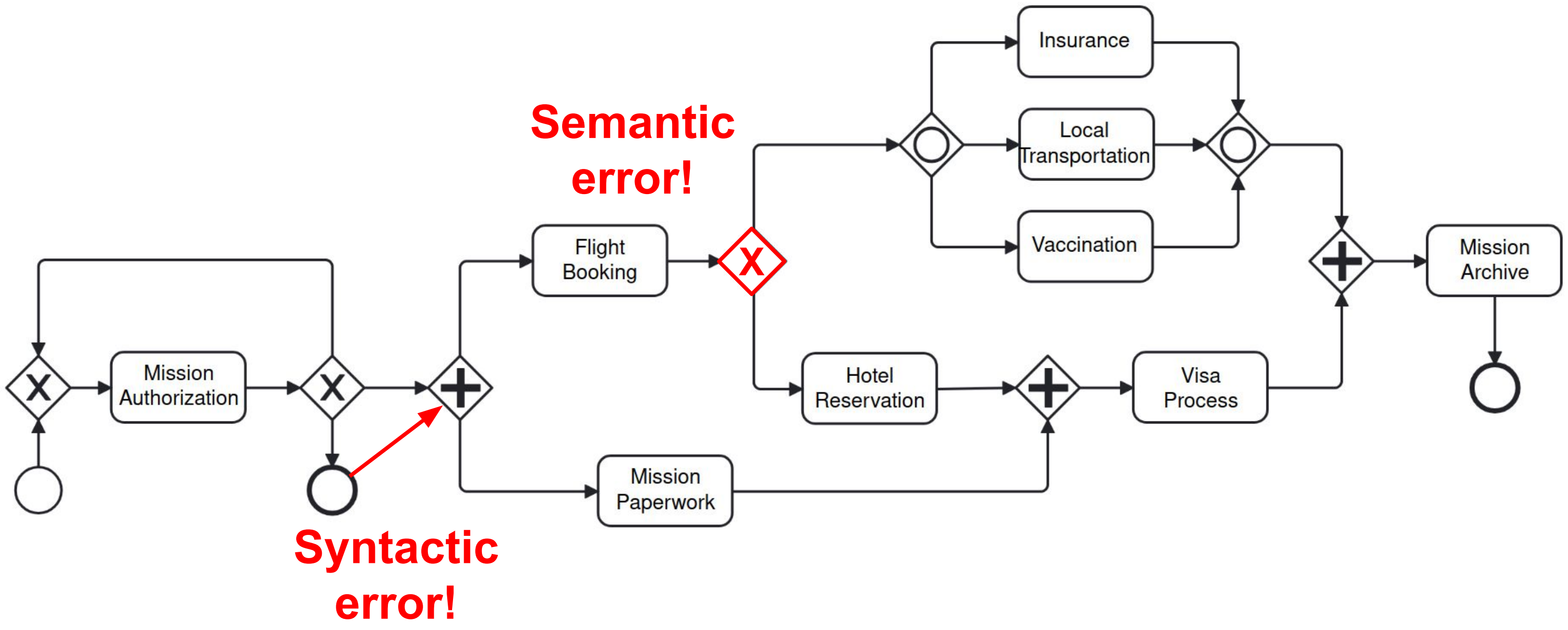
Fourth Research Question



Fourth Research Question



Fourth Research Question

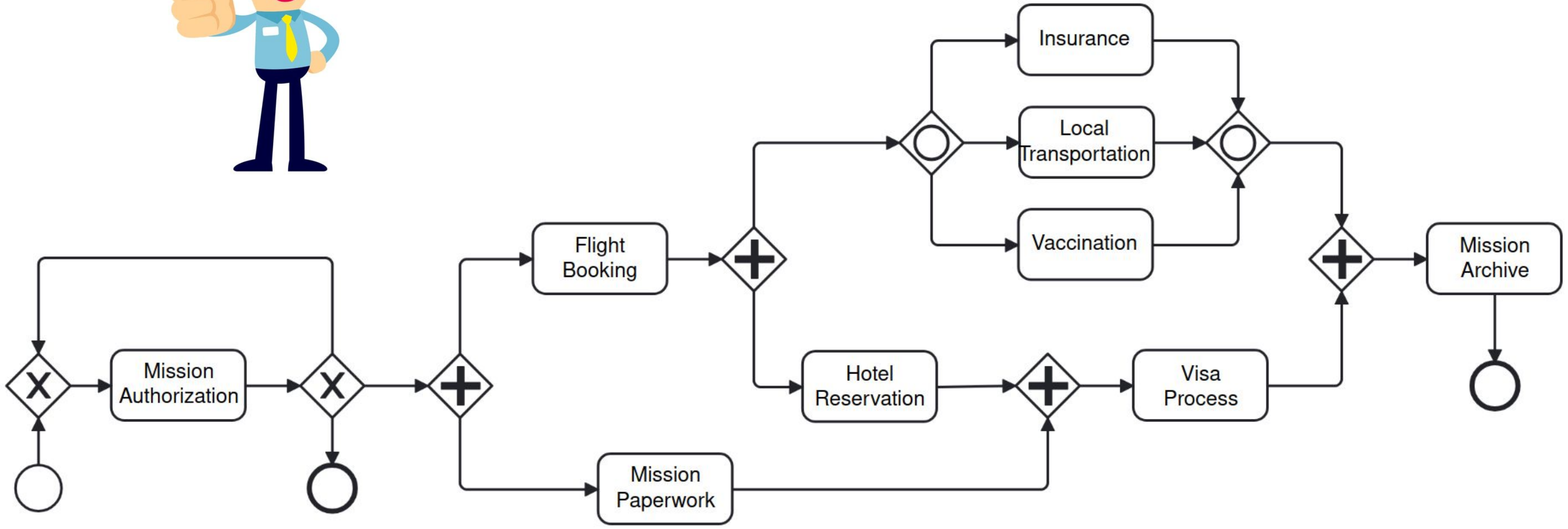


➢ How can you be sure that your BPMN process is syntactically and semantically correct?

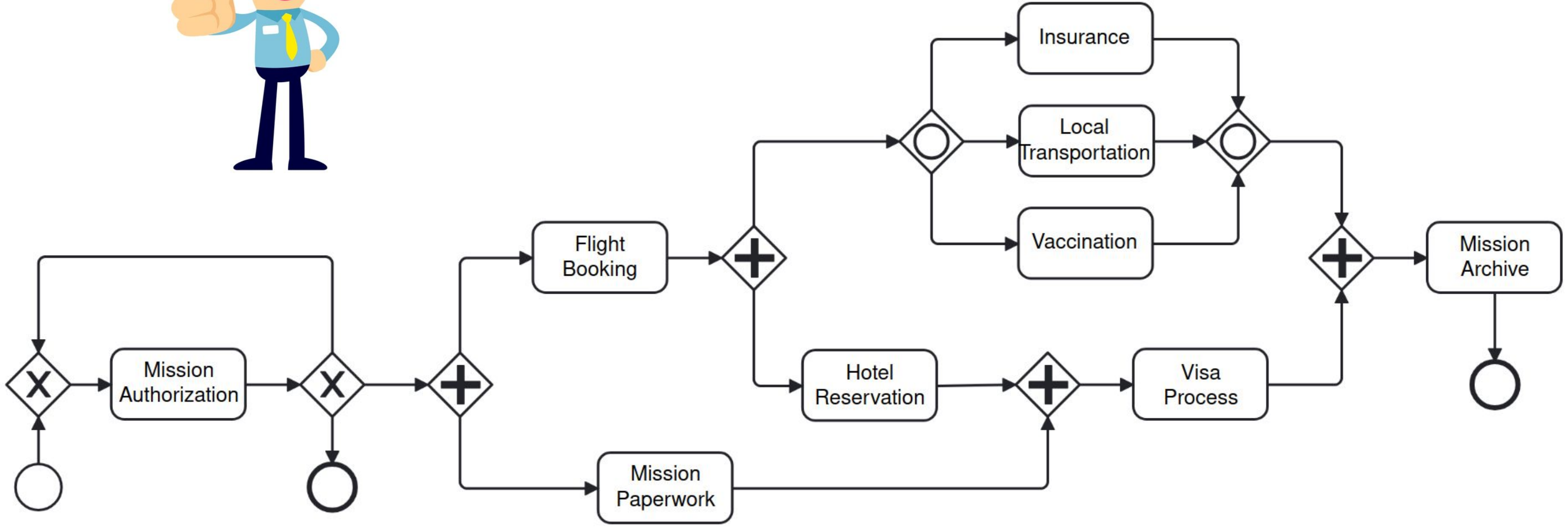
PhD Research Axes

- ➢ What if you do not know **how to write BPMN**?
- ➢ What if you do not want to **spend time designing** your process graphically?
- ➢ How can you be sure that your BPMN process **matches its expected behaviour**?
- ➢ How can you be sure that your BPMN process is **syntactically and semantically correct**?

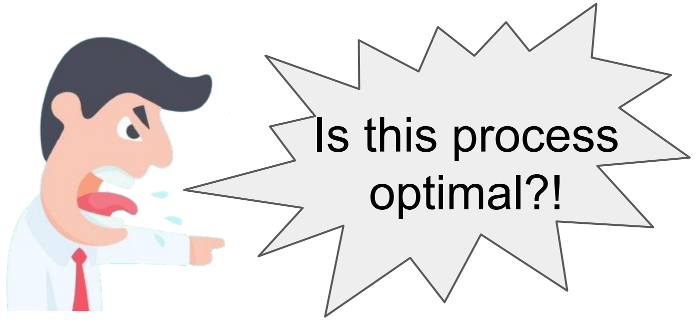
Fifth Research Question

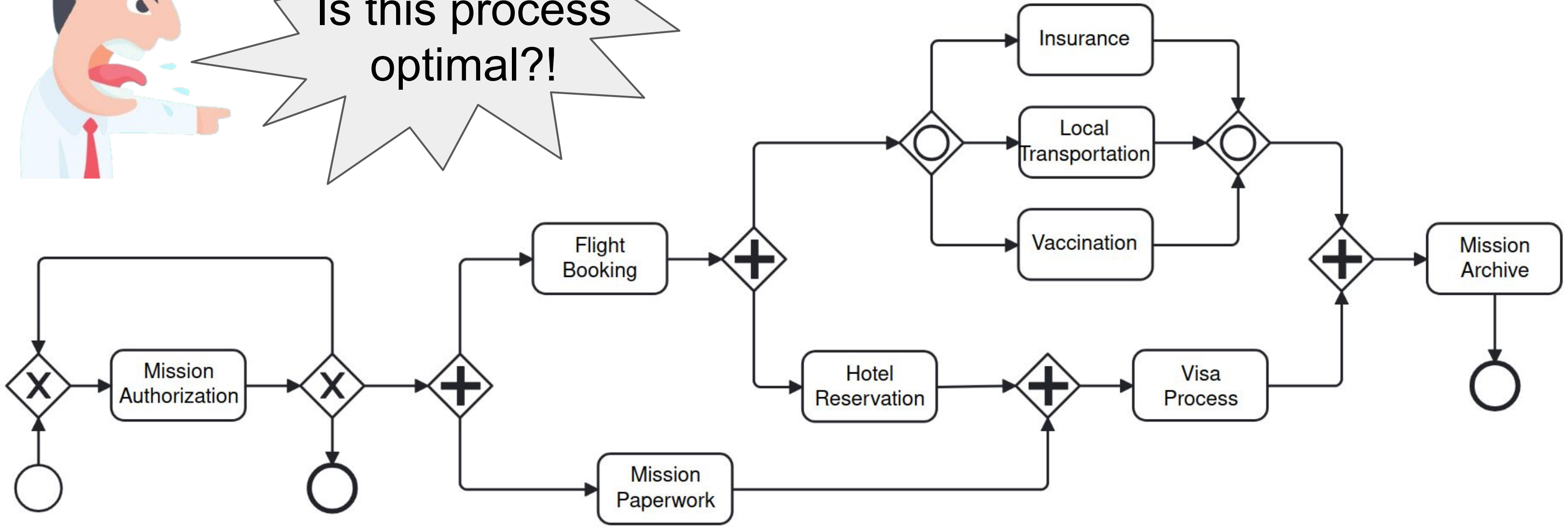


Fifth Research Question

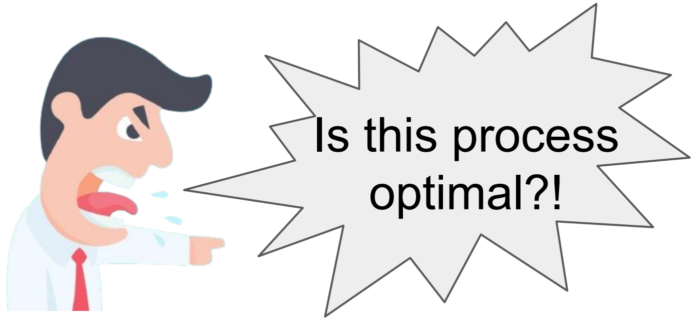


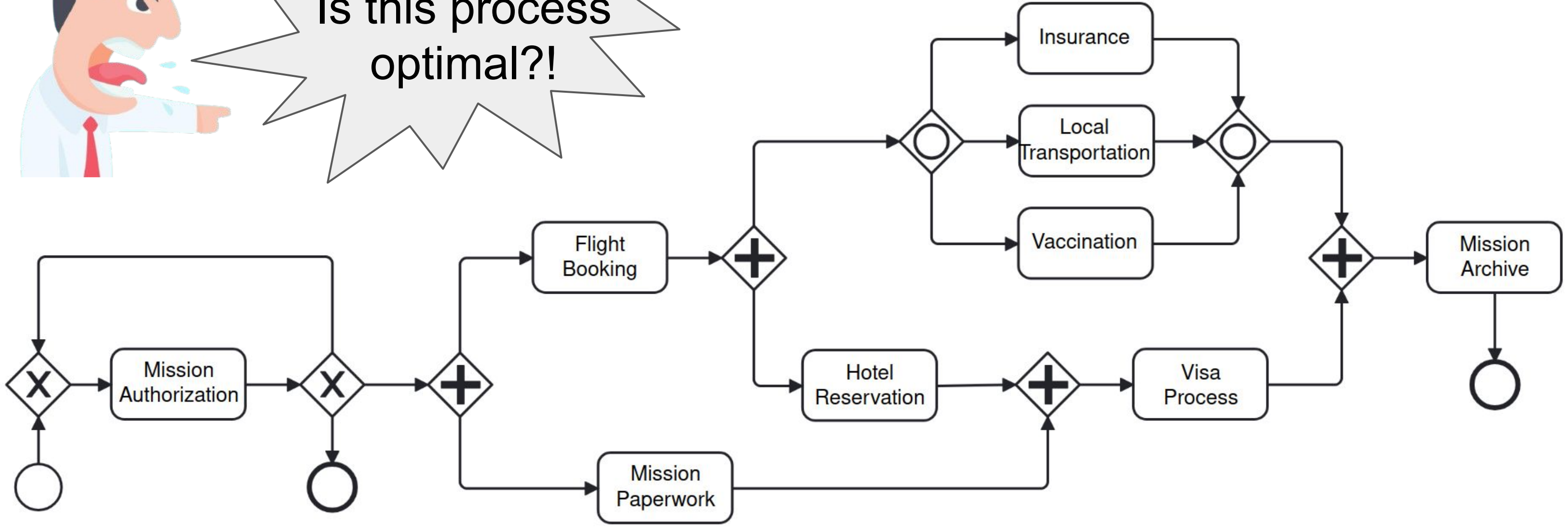
Fifth Research Question





Fifth Research Question

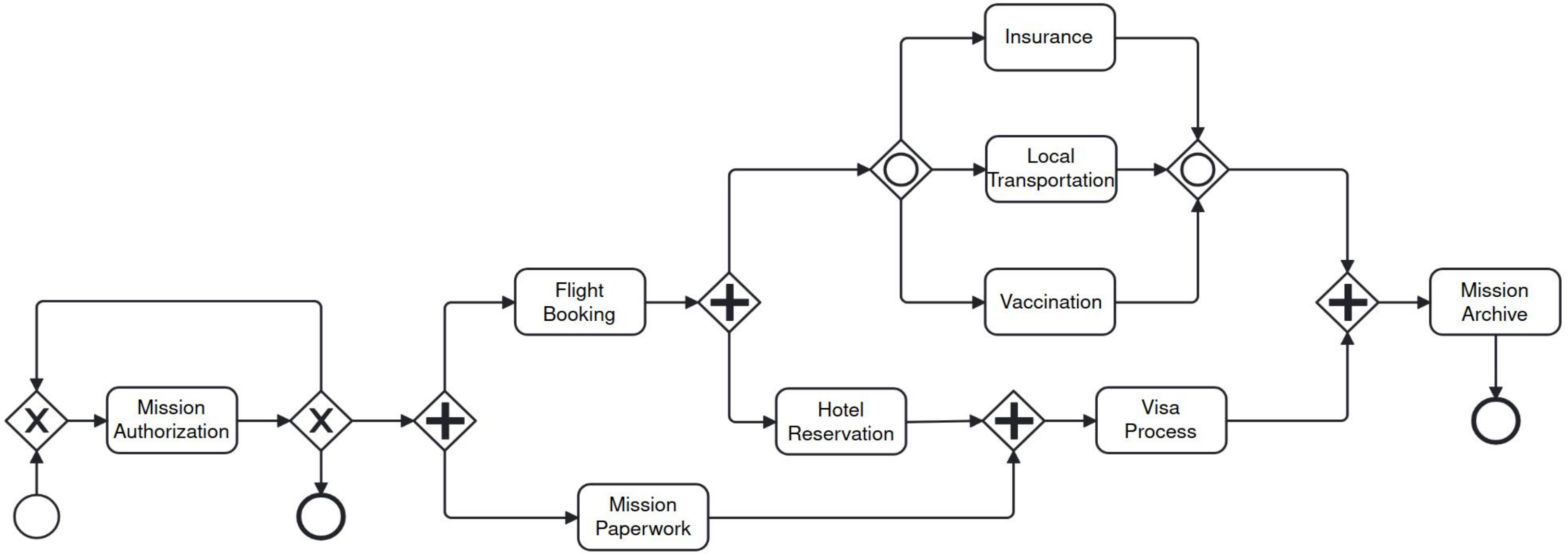




➢ In the **resource-free**, **durations-free**, **single instance** context, yes!

Fifth Research Question

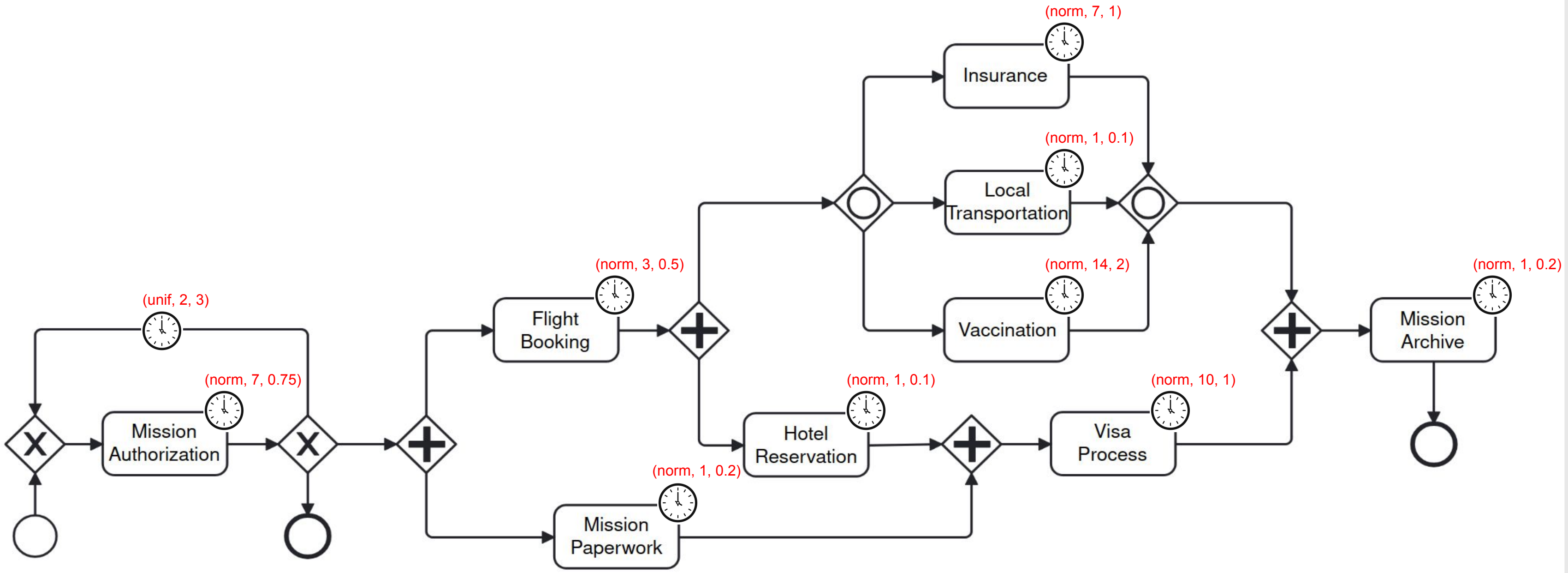
But what if we enrich the process with:



Fifth Research Question

But what if we enrich the process with:

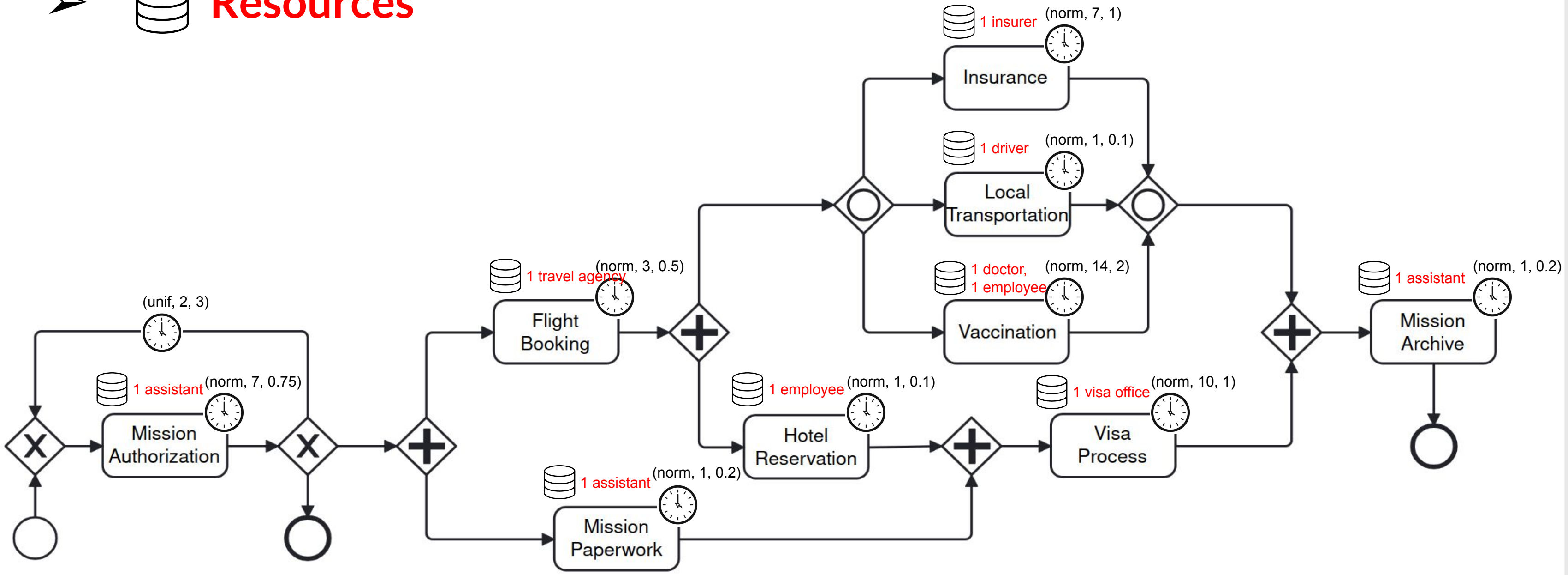
- ➢ 🕓 **Durations** (following probabilistic distributions)



Fifth Research Question

But what if we enrich the process with:

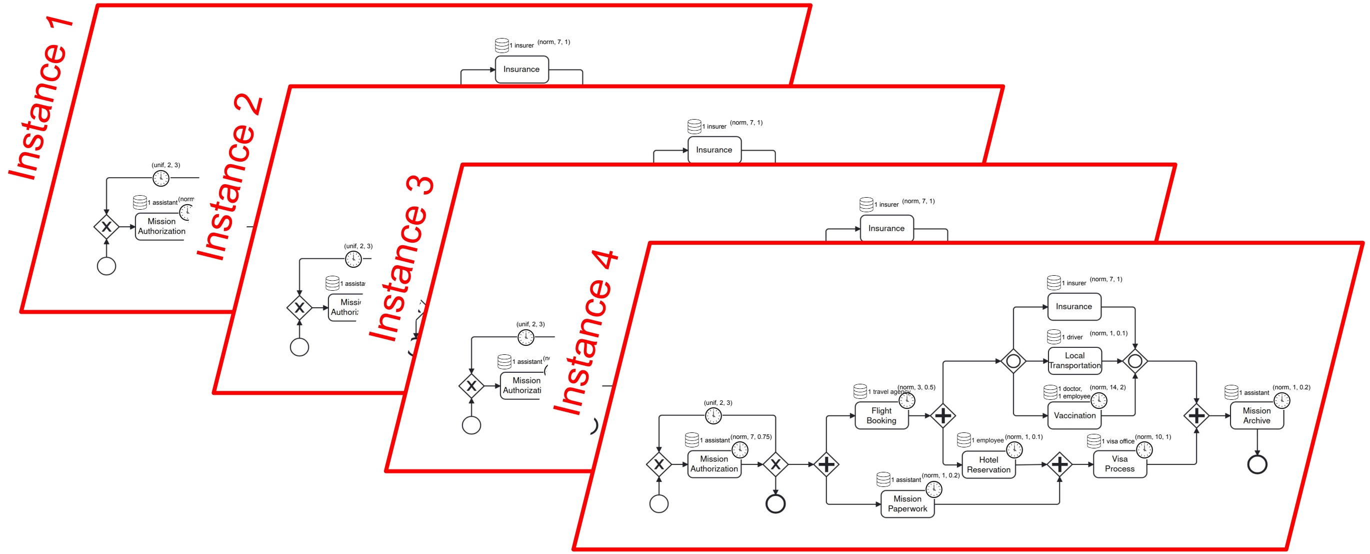
- ➢ 🕓 Durations (following probabilistic distributions)
- ➢ 🛢 Resources



Fifth Research Question

But what if we enrich the process with:

- ➢ Durations (following probabilistic distributions)
- ➢ Resources
- ➢ **Multiple Simultaneous Executions**



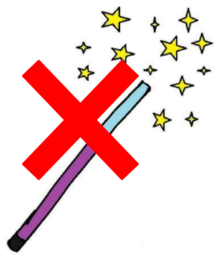
Fifth Research Question

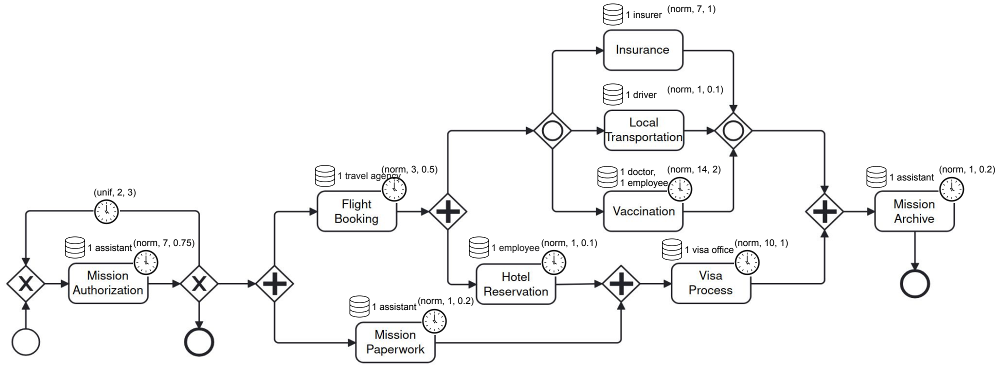
The problem becomes **much more complex** with **resources**, **durations**, and **multiple executions** of the process, despite being common when dealing with business processes.

Fifth Research Question

The problem becomes **much more complex** with **resources**, **durations**, and **multiple executions** of the process, despite being common when dealing with business processes.

And unfortunately, there is **no magic wand** performing this task.

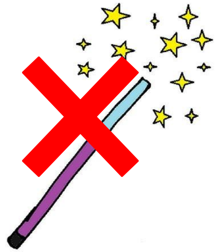


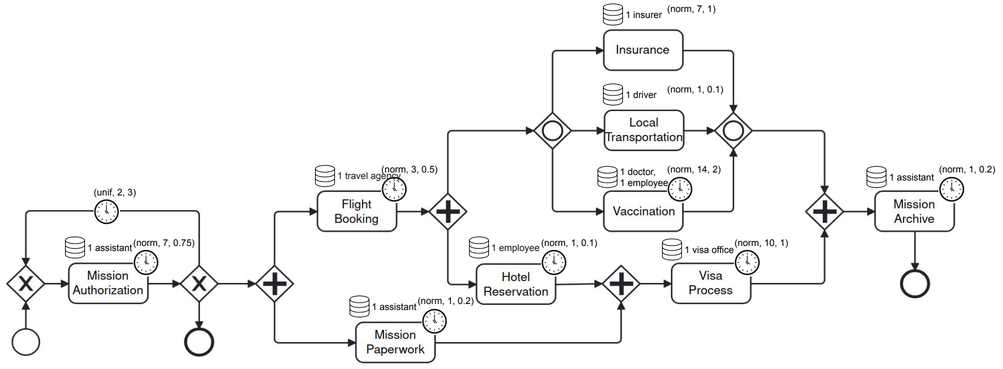


Fifth Research Question

The problem becomes **much more complex** with **resources**, **durations**, and **multiple executions** of the process, despite being common when dealing with business processes.

And unfortunately, there is **no magic wand** performing this task.





1 insurer (norm, 7, 1)
Insurance
1 driver (norm, 1, 0.1)
Local Transportation
1 doctor, 1 employee (norm, 14, 2)
Vaccination
1 travel agency (norm, 3, 0.5)
Flight Booking
(unif, 2, 3)
1 assistant (norm, 7, 0.75)
Mission Authorization
1 employee (norm, 1, 0.1)
Hotel Reservation
1 visa office (norm, 10, 1)
Visa Process
1 assistant (norm, 1, 0.2)
Mission Paperwork
1 assistant (norm, 1, 0.2)
Mission Archive

➢ How can you **optimise** a BPMN process in real-world conditions?

PhD Research Axes

- ➢ What if you do not know how to write BPMN?
- ➢ What if you do not want to spend time designing your process graphically?
- ➢ How can you be sure that your BPMN process matches its expected behaviour?
- ➢ How can you be sure that your BPMN process is syntactically and semantically correct?

- ➢ How can you **optimise** a BPMN process in real-world conditions?

Published Results

I

- ➢ What if you do not know how to write BPMN?
- ➢ What if you do not want to spend time designing your process graphically?
- ➢ How can you be sure that your BPMN process matches its expected behaviour?
- ➢ How can you be sure that your BPMN process is syntactically and semantically correct?

→ [ICSOC’24]

→ [FSE’25]

→ [TSE’25]*

II

- ➢ How can you optimise a BPMN process in real-world conditions?

→ [SEFM’23]

→ [QRS’24]

→ [SoSyM’25]*

Focus of this Presentation

Ⅰ

- ➢ What if you do not know how to write BPMN?
- ➢ What if you do not want to spend time designing your process graphically?
- ➢ How can you be sure that your BPMN process matches its expected behaviour?
- ➢ How can you be sure that your BPMN process is syntactically and semantically correct?

[ICSOC'24]

[FSE'25]

[TSE'25]*

Ⅱ

- ➢ How can you optimise a BPMN process in real-world conditions?

[SEFM'23]

[QRS'24]

[SoSyM'25]*

Plan

Global Picture of the Approach

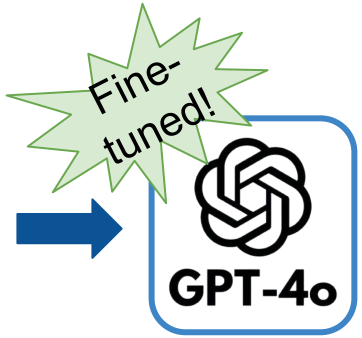
First of all, an employee CollectGoods. Then, the client PayForDelivery while the employee PrepareParcel. Finally, the company can either DeliverByCar or DeliverByDrone (depending on the distance for example)

Textual Representation of the Process

Global Picture of the Approach

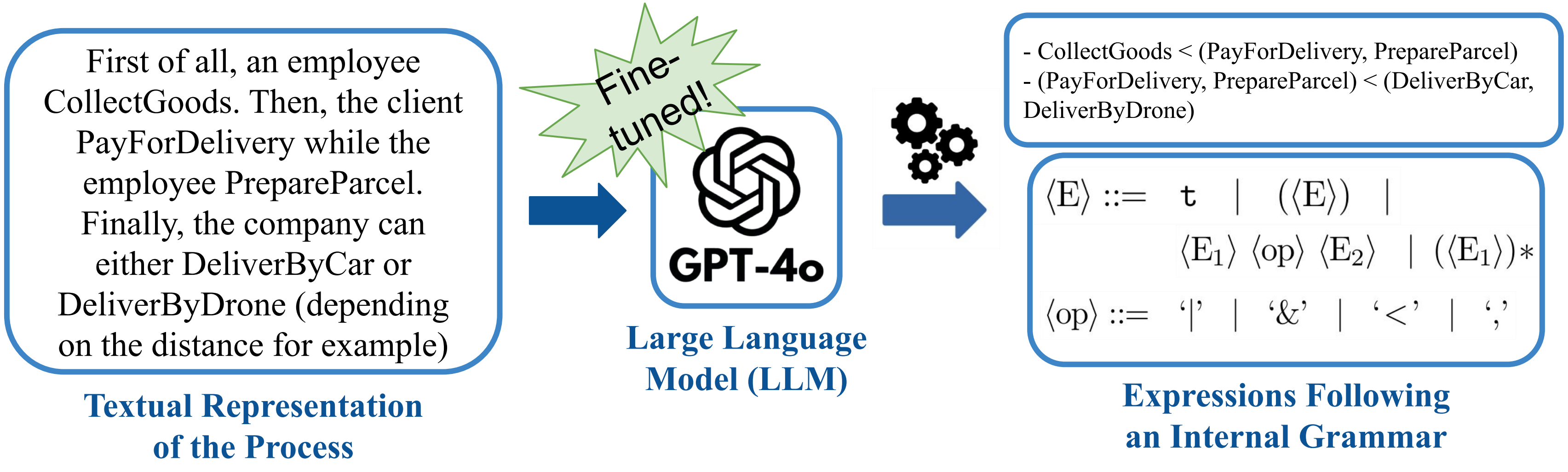
First of all, an employee CollectGoods. Then, the client PayForDelivery while the employee PrepareParcel. Finally, the company can either DeliverByCar or DeliverByDrone (depending on the distance for example)

Textual Representation of the Process



Large Language Model (LLM)

Global Picture of the Approach



First of all, an employee CollectGoods. Then, the client PayForDelivery while the employee PrepareParcel. Finally, the company can either DeliverByCar or DeliverByDrone (depending on the distance for example)

Textual Representation of the Process

Large Language Model (LLM)

- CollectGoods < (PayForDelivery, PrepareParcel)
- (PayForDelivery, PrepareParcel) < (DeliverByCar, DeliverByDrone)

$$\begin{aligned} \langle \mathrm{E} \rangle &::= \texttt{t} \mid (\langle \mathrm{E} \rangle) \mid \langle \mathrm{E}_1 \rangle\, \langle \mathrm{op} \rangle\, \langle \mathrm{E}_2 \rangle \mid (\langle \mathrm{E}_1 \rangle)* \\ \langle \mathrm{op} \rangle &::= \text{‘}|\text{’} \mid \text{‘}\&\text{’} \mid \text{‘}<\text{’} \mid \text{‘};\text{’} \end{aligned}$$

Expressions Following an Internal Grammar

Global Picture of the Approach

First of all, an employee CollectGoods. Then, the client PayForDelivery while the employee PrepareParcel. Finally, the company can either DeliverByCar or DeliverByDrone (depending on the distance for example)

Textual Representation of the Process

Fine-tuned!

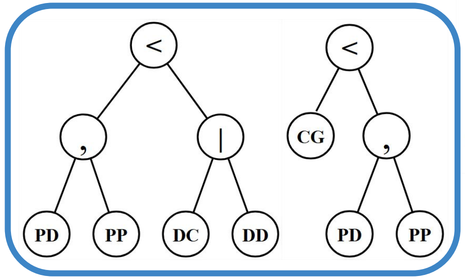
GPT-4o

Large Language Model (LLM)

- CollectGoods < (PayForDelivery, PrepareParcel)
- (PayForDelivery, PrepareParcel) < (DeliverByCar, DeliverByDrone)

$$
\begin{aligned}
\langle E\rangle &::= \texttt{t} \mid (\langle E\rangle) \mid \\
&\quad \langle E_1\rangle\,\langle op\rangle\,\langle E_2\rangle \mid (\langle E_1\rangle)* \\
\langle op\rangle &::= \text{‘}|\text{’} \mid \text{‘}\&\text{’} \mid \text{‘}<\text{’} \mid \text{‘},\text{’}
\end{aligned}
$$

Expressions Following an Internal Grammar



Abstract Syntax Trees

Global Picture of the Approach

First of all, an employee CollectGoods. Then, the client PayForDelivery while the employee PrepareParcel. Finally, the company can either DeliverByCar or DeliverByDrone (depending on the distance for example)

Textual Representation of the Process

Fine-tuned!

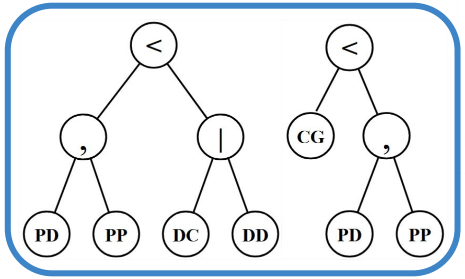
GPT-4o

Large Language Model (LLM)

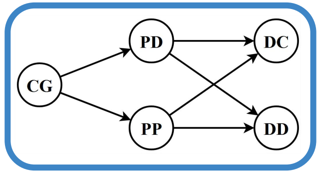
- CollectGoods < (PayForDelivery, PrepareParcel)
- (PayForDelivery, PrepareParcel) < (DeliverByCar, DeliverByDrone)

$$\begin{aligned}\langle E\rangle &::= \mathtt{t} \mid (\langle E\rangle) \mid \\ &\quad\ \langle E_1\rangle\,\langle op\rangle\,\langle E_2\rangle \mid (\langle E_1\rangle)* \\ \langle op\rangle &::= \text{`}|\text{'} \mid \text{`}\&\text{'} \mid \text{`}<\text{'} \mid \text{`},\text{'}\end{aligned}$$

Expressions Following an Internal Grammar



Abstract Syntax Trees



Dependency Graph (Skeleton of the Process)

Global Picture of the Approach

First of all, an employee CollectGoods. Then, the client PayForDelivery while the employee PrepareParcel. Finally, the company can either DeliverByCar or DeliverByDrone (depending on the distance for example)

Textual Representation of the Process

Fine-tuned!

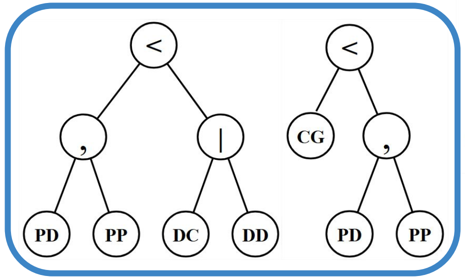
GPT-4o

Large Language Model (LLM)

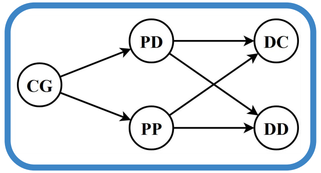
- CollectGoods < (PayForDelivery, PrepareParcel)
- (PayForDelivery, PrepareParcel) < (DeliverByCar, DeliverByDrone)

$$
\begin{aligned}
\langle E \rangle &::= \texttt{t} \mid (\langle E \rangle) \mid \\
&\quad \langle E_1 \rangle \langle op \rangle \langle E_2 \rangle \mid (\langle E_1 \rangle)* \\
\langle op \rangle &::= \text{`}|\text{'} \mid \text{`}\&\text{'} \mid \text{`}<\text{'} \mid \text{`},\text{'}
\end{aligned}
$$

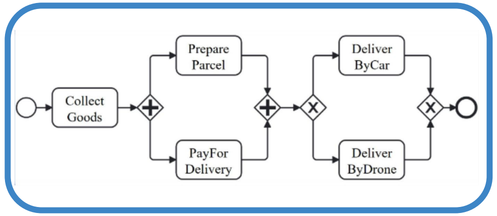
Expressions Following an Internal Grammar



Abstract Syntax Trees



Dependency Graph (Skeleton of the Process)



BPMN Process

Global Picture of the Approach

First of all, an employee CollectGoods. Then, the client PayForDelivery while the employee PrepareParcel. Finally, the company can either DeliverByCar or DeliverByDrone (depending on the distance for example)

Textual Representation of the Process

Fine-tuned!

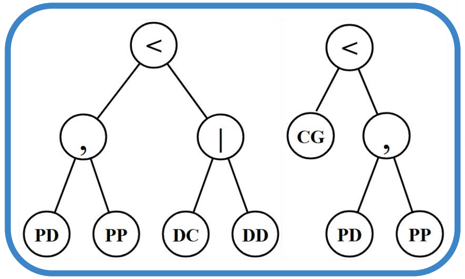
GPT-4o

Large Language Model (LLM)

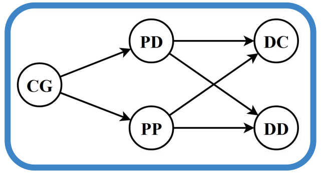
- CollectGoods < (PayForDelivery, PrepareParcel)
- (PayForDelivery, PrepareParcel) < (DeliverByCar, DeliverByDrone)

$$
\begin{aligned}
\langle E \rangle &::= \texttt{t} \mid (\langle E \rangle) \mid \langle E_1 \rangle \langle op \rangle \langle E_2 \rangle \mid (\langle E_1 \rangle)* \\
\langle op \rangle &::= \text{‘}|\text{’} \mid \text{‘}\&\text{’} \mid \text{‘}<\text{’} \mid \text{‘},\text{’}
\end{aligned}
$$

Expressions Following an Internal Grammar

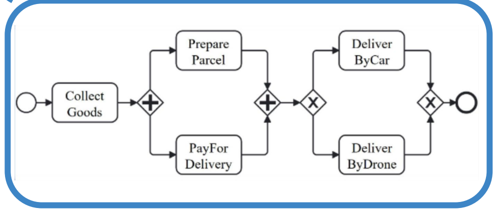


Abstract Syntax Trees



Dependency Graph (Skeleton of the Process)

Refinement



BPMN Process

Detailed Approach – Step 1 – Description

The user first has to write a **textual description** of the process-to-be.

First, the developer StartFeatureManagementSoftware (**StFMS**).
Then, he DescribeNewFeatureRequirements (**DNFR**). After that, the staff ValidateInternally (**VI**), and the client ValidateExternally (**VE**). Once the feature has been validated internally, the developer can CreateNewFeatureBranch (**CNFB**). Once the feature is completely validated (internally and externally), the staff can StartTechnicalDesign (**STD**). Instead of describing a new feature, validate it, create a new branch and start technical design, the developer can also LoadCurrentlyDevelopedFeature (**LCDF**). The FeatureDevelopment (**FD**) then eventually starts, followed by a DebuggingPhase (**DP**) useful to chase possible bugs before releasing the feature. This phase leads either to a BugCaseOpening (**BCO**), or to ReleaseFeature (**RF**) if no bug was found. If a bug case is opened, three different operations may start: either the first support level initiates a FirstStageDebugPhase (**FSDP**), which eventually leads to ClosingFirstLevelRequest (**CFLR**), or the second support level initiates a SecondStageDebugPhase (**SSDP**), which eventually leads to ClosingSecondLevelRequest (**CSLR**), or the third support level initiates a ThirdStageDebugPhase (**TSDP**), which eventually leads to ClosingThirdLevelRequest (**CTLR**). Once these phases are closed, either there is no bug anymore to correct, and the ReleaseFeature task (**RF**) occurs, or a new bug is found, leading to DebuggingPhase (**DP**) again. Also, the FirstStageDebugPhase (**FSDP**), SecondStageDebugPhase (**SSDP**) and ThirdStageDebugPhase (**TSDP**) and their closing can be repeated until a bug is properly corrected. Once ReleaseFeature (**RF**) occurred, the developer can either ShutdownFeatureManagementSoftware (**ShFMS**), or start again with the task DescribeNewFeatureRequirements (**DNFR**).

Detailed Approach – Step 2 – LLM Prompting

The textual description is then **given to a (fine-tuned) LLM** (GPT-4o atm).

First, the developer StartFeatureManagementSoftware (**StFMS**).
Then, he DescribeNewFeatureRequirements (**DNFR**). After that, the staff ValidateInternally (**VI**), and the client ValidateExternally (**VE**). Once the feature has been validated internally, the developer can CreateNewFeatureBranch (**CNFB**). Once the feature is completely validated (internally and externally), the staff can StartTechnicalDesign (**STD**). Instead of describing a new feature, validate it, create a new branch and start technical design, the developer can also LoadCurrentlyDevelopedFeature (**LCDF**). The FeatureDevelopment (**FD**) then eventually starts, followed by a DebuggingPhase (**DP**) useful to chase possible bugs before releasing the feature. This phase leads either to a BugCaseOpening (**BCO**), or to ReleaseFeature (**RF**) if no bug was found. If a bug case is opened, three different operations may start: either the first support level initiates a FirstStageDebugPhase (**FSDP**), which eventually leads to ClosingFirstLevelRequest (**CFLR**), or the second support level initiates a SecondStageDebugPhase (**SSDP**), which eventually leads to ClosingSecondLevelRequest (**CSLR**), or the third support level initiates a ThirdStageDebugPhase (**TSDP**), which eventually leads to ClosingThirdLevelRequest (**CTLR**). Once these phases are closed, either there is no bug anymore to correct, and the ReleaseFeature task (**RF**) occurs, or a new bug is found, leading to DebuggingPhase (**DP**) again. Also, the FirstStageDebugPhase (**FSDP**), SecondStageDebugPhase (**SSDP**) and ThirdStageDebugPhase (**TSDP**) and their closing can be repeated until a bug is properly corrected. Once ReleaseFeature (**RF**) occurred, the developer can either ShutdownFeatureManagementSoftware (**ShFMS**), or start again with the task DescribeNewFeatureRequirements (**DNFR**).

Fine-tuned!

GPT-4o

Detailed Approach – Step 2 – LLM Prompting

The textual description is then **given to a (fine-tuned) LLM** (GPT-4o atm).

First, the developer StartFeatureManagementSoftware (**StFMS**).
Then, he DescribeNewFeatureRequirements (**DNFR**). After that, the staff ValidateInternally (**VI**), and the client ValidateExternally (**VE**). Once the feature has been validated internally, the developer can CreateNewFeatureBranch (**CNFB**). Once the feature is completely validated (internally and externally), the staff can StartTechnicalDesign (**STD**). Instead of describing a new feature, validate it, create a new branch and start technical design, the developer can also LoadCurrentlyDevelopedFeature (**LCDF**). The FeatureDevelopment (**FD**) then eventually starts, followed by a DebuggingPhase (**DP**) useful to chase possible bugs before releasing the feature. This phase leads either to a BugCaseOpening (**BCO**), or to ReleaseFeature (**RF**) if no bug was found. If a bug case is opened, three different operations may start: either the first support level initiates a FirstStageDebugPhase (**FSDP**), which eventually leads to ClosingFirstLevelRequest (**CFLR**), or the second support level initiates a SecondStageDebugPhase (**SSDP**), which eventually leads to ClosingSecondLevelRequest (**CSLR**), or the third support level initiates a ThirdStageDebugPhase (**TSDP**), which eventually leads to ClosingThirdLevelRequest (**CTLR**). Once these phases are closed, either there is no bug anymore to correct, and the ReleaseFeature task (**RF**) occurs, or a new bug is found, leading to DebuggingPhase (**DP**) again. Also, the FirstStageDebugPhase (**FSDP**), SecondStageDebugPhase (**SSDP**) and ThirdStageDebugPhase (**TSDP**) and their closing can be repeated until a bug is properly corrected. Once ReleaseFeature (**RF**) occurred, the developer can either ShutdownFeatureManagementSoftware (**ShFMS**), or start again with the task DescribeNewFeatureRequirements (**DNFR**).

Fine-tuned!

GPT-4o

The LLM processes the description and returns a **set of expressions** following an **internal grammar**.

$$\langle \mathrm{E} \rangle ::= \mathtt{t} \;\mid\; (\langle \mathrm{E} \rangle) \;\mid\; \langle \mathrm{E}_1 \rangle \; \langle \mathrm{op} \rangle \; \langle \mathrm{E}_2 \rangle \;\mid\; (\langle \mathrm{E}_1 \rangle)*$$

$$\langle \mathrm{op} \rangle ::= \text{‘}|\text{’} \;\mid\; \text{‘}\&\text{’} \;\mid\; \text{‘}<\text{’} \;\mid\; \text{‘};\text{’}$$

Detailed Approach – Step 3 – Expressions

Given our description, the LLM returns **ten expressions**:

Detailed Approach – Step 3 – Expressions

Given our description, the LLM returns **ten expressions**:

StFMS < DNFR < (VI, VE)

VI < CNFB

(VI, VE) < STD

(STD, CNFB) < (FD < DP)

(DNFR, VI, VE, CNFB, STD) | LCDF

DP < (BCO | RF)

BCO < ((FSDP < CFLR) | (SSDP < CSLR) | (TSDP < CTLR))

(CFLR, CSLR, CTLR) < (RF | DP)

(FSDP, SSDP, TSDP, CFLR, CSLR, CTLR)*

RF < (ShFMS | DNFR)

Detailed Approach – Step 3 – Expressions

Given our description, the LLM returns **ten expressions**:

StFMS < DNFR < (VI, VE)

VI < CNFB

(VI, VE) < STD

(STD, CNFB) < (FD < DP)

(DNFR, VI, VE, CNFB, STD) | LCDF

DP < (BCO | RF)

BCO < ((FSDP < CFLR) | (SSDP < CSLR) | (TSDP < CTLR))

(CFLR, CSLR, CTLR) < (RF | DP)

(FSDP, SSDP, TSDP, CFLR, CSLR, CTLR)*

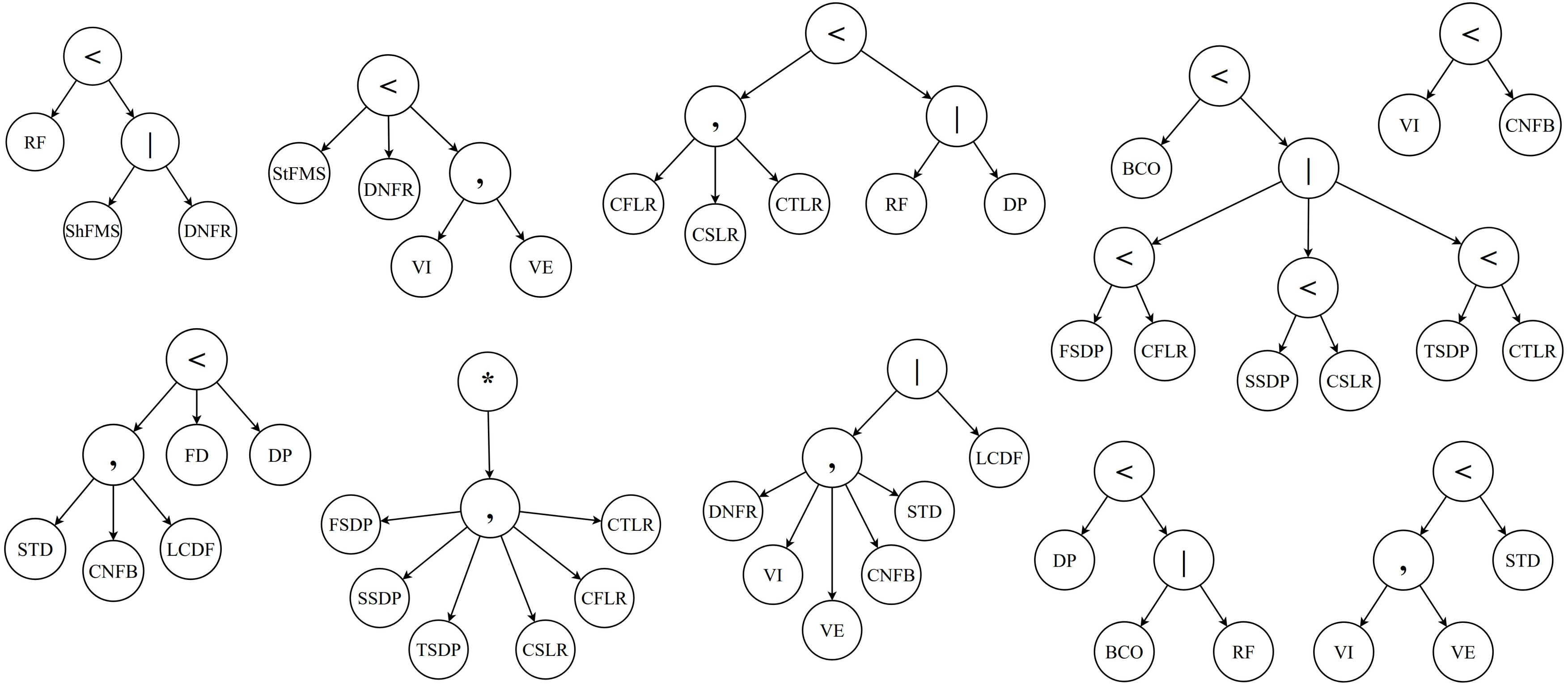
RF < (ShFMS | DNFR)

Detailed Approach – Step 4 – ASTs Mapping

These expressions are then **mapped to** their corresponding (reduced) **abstract syntax trees (ASTs)**.

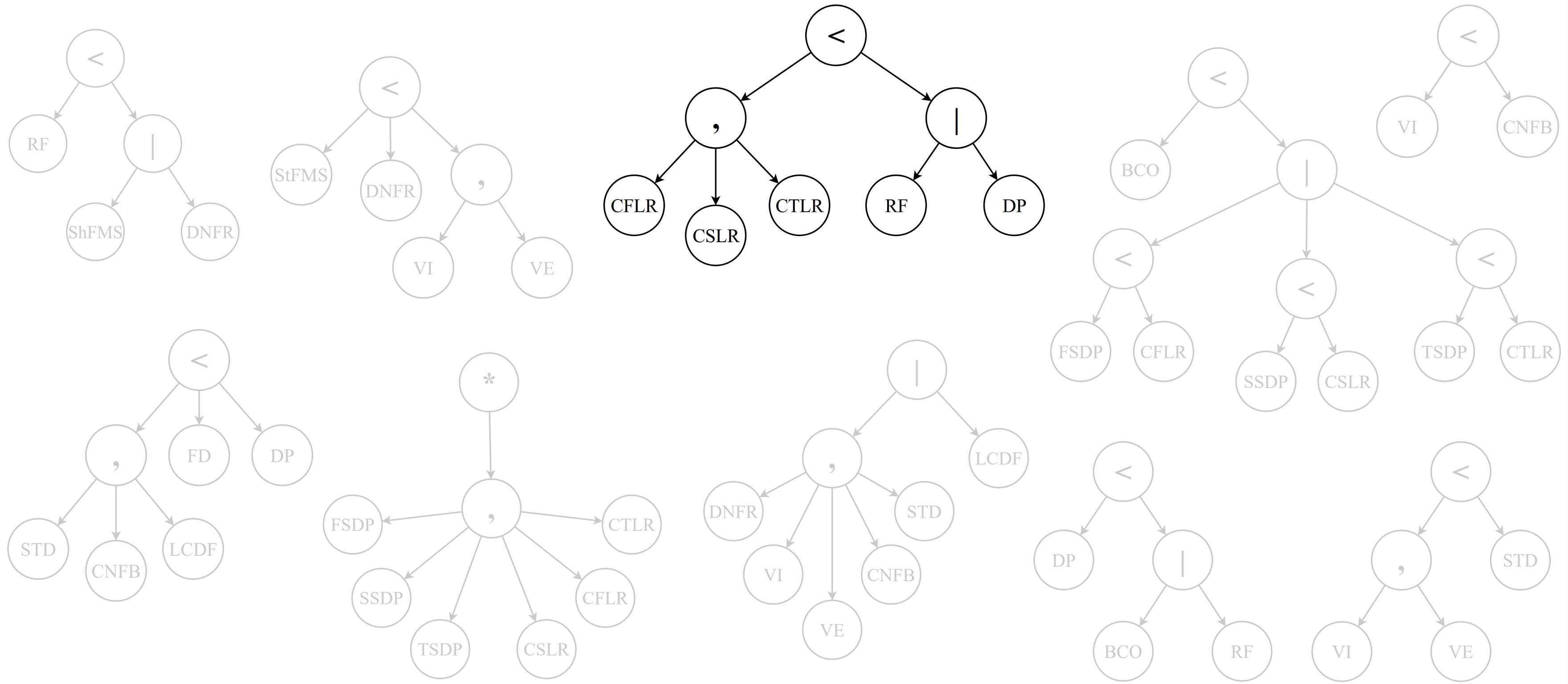
Detailed Approach – Step 4 – ASTs Mapping

These expressions are then **mapped to** their corresponding (reduced) **abstract syntax trees (ASTs)**.



Detailed Approach – Step 4 – ASTs Mapping

These expressions are then **mapped to** their corresponding (reduced) **abstract syntax trees (ASTs)**.

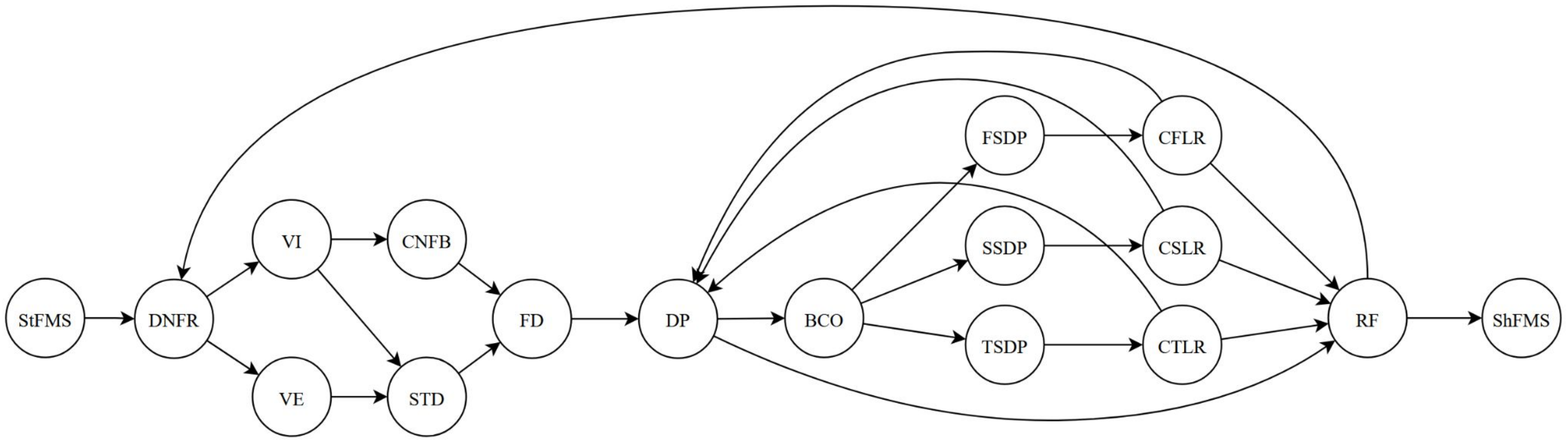


Detailed Approach – Step 5 – Dependency Graph

The **sequential information** contained in the multiple ASTs is then gathered to obtain a **cleaner** and **more compact** representation of it, called **dependency graph**.

Detailed Approach – Step 5 – Dependency Graph

The **sequential information** contained in the multiple ASTs is then gathered to obtain a **cleaner** and **more compact** representation of it, called **dependency graph**.

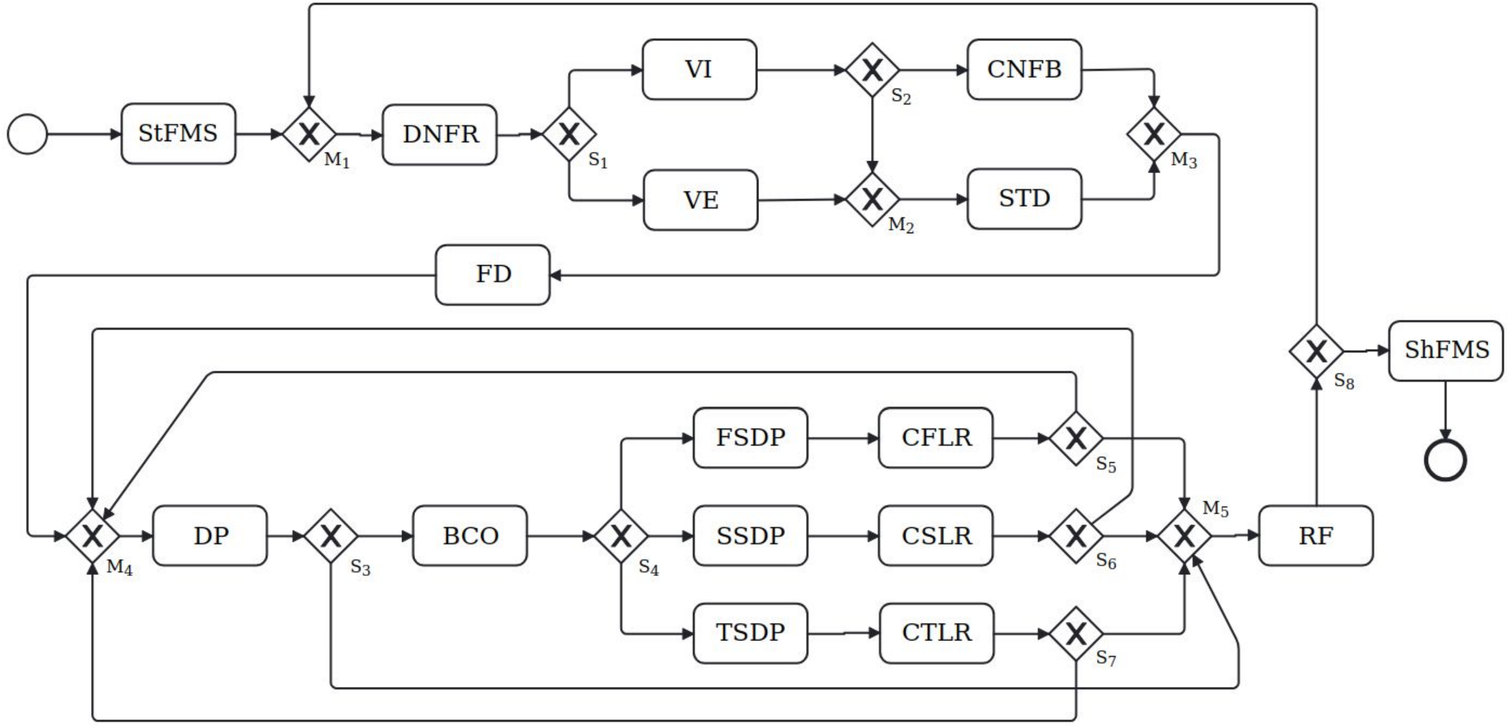


Detailed Approach – Step 6 – BPMN Generation

This graph is then transformed into the corresponding BPMN process by adding a **start event**, one or several **end events**, and **exclusive gateways**.

Detailed Approach – Step 6 – BPMN Generation

This graph is then transformed into the corresponding BPMN process by adding a **start event**, one or several **end events**, and **exclusive gateways**.



Detailed Approach – Step 6.1 – Mutual Exclusions

However, this BPMN process may be **incomplete** with regards to the **generated expressions**.

Detailed Approach – Step 6.1 – Mutual Exclusions

However, this BPMN process may be **incomplete** with regards to the **generated expressions**.

For instance, expression $\boxed{\text{(DNFR, VI, VE, CNFB, STD) | LCDF}}$ introduces task **LCDF** which is **not yet in the BPMN** process!

Detailed Approach – Step 6.1 – Mutual Exclusions

However, this BPMN process may be **incomplete** with regards to the **generated expressions**.

For instance, expression `(DNFR, VI, VE, CNFB, STD) | LCDF` introduces task **LCDF** which is **not yet in the BPMN** process!

This expression states that task **LCDF** must be **mutually exclusive** of the tasks **DNFR**, **VI**, **VE**, **CNFB**, and **STD**.

Detailed Approach – Step 6.1 – Mutual Exclusions

However, this BPMN process may be **incomplete** with regards to the **generated expressions**.

For instance, expression `(DNFR, VI, VE, CNFB, STD) | LCDF` introduces task **LCDF** which is **not yet in the BPMN** process!

This expression states that task **LCDF** must be **mutually exclusive** of the tasks **DNFR**, **VI**, **VE**, **CNFB**, and **STD**.

Let us recall what mutual exclusion is.

Detailed Approach – Step 6.1 – Mutual Exclusions

In BPMN, a mutual exclusion between two tasks can be defined as the **impossibility to perform both tasks** during the execution of the process.

Detailed Approach – Step 6.1 – Mutual Exclusions

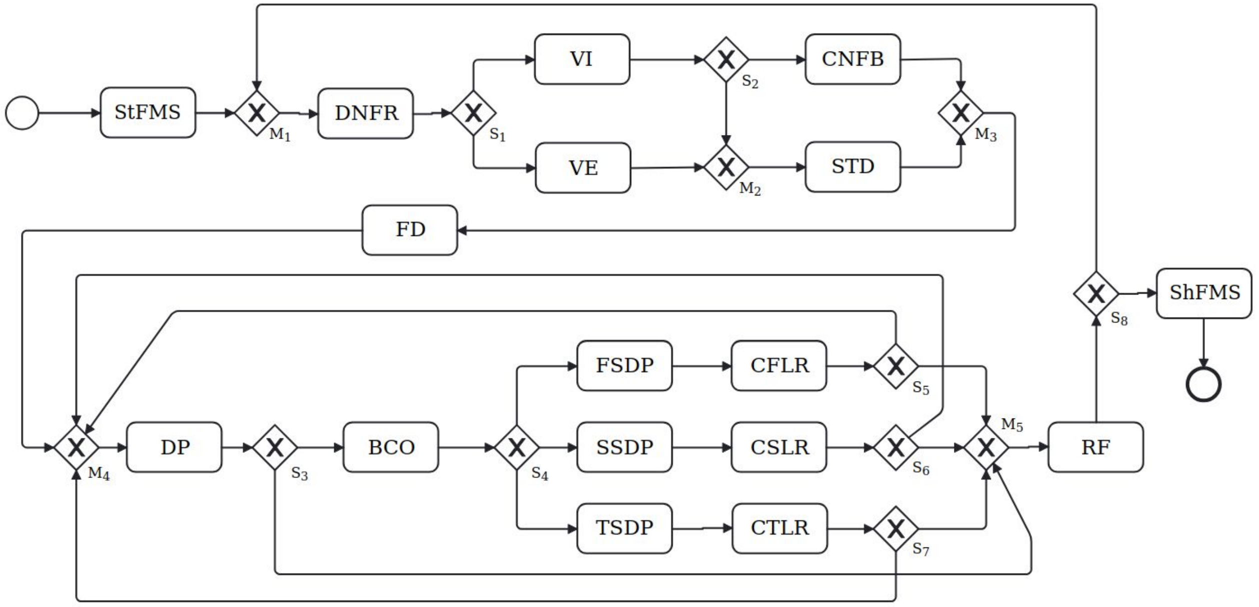
In BPMN, a mutual exclusion between two tasks can be defined as the **impossibility to perform both tasks** during the execution of the process.

This can be verified by ensuring that **no path** of the process **contains these two tasks**. However, this is **quite restrictive**.

Detailed Approach – Step 6.1 – Mutual Exclusions

In BPMN, a mutual exclusion between two tasks can be defined as the **impossibility to perform both tasks** during the execution of the process.

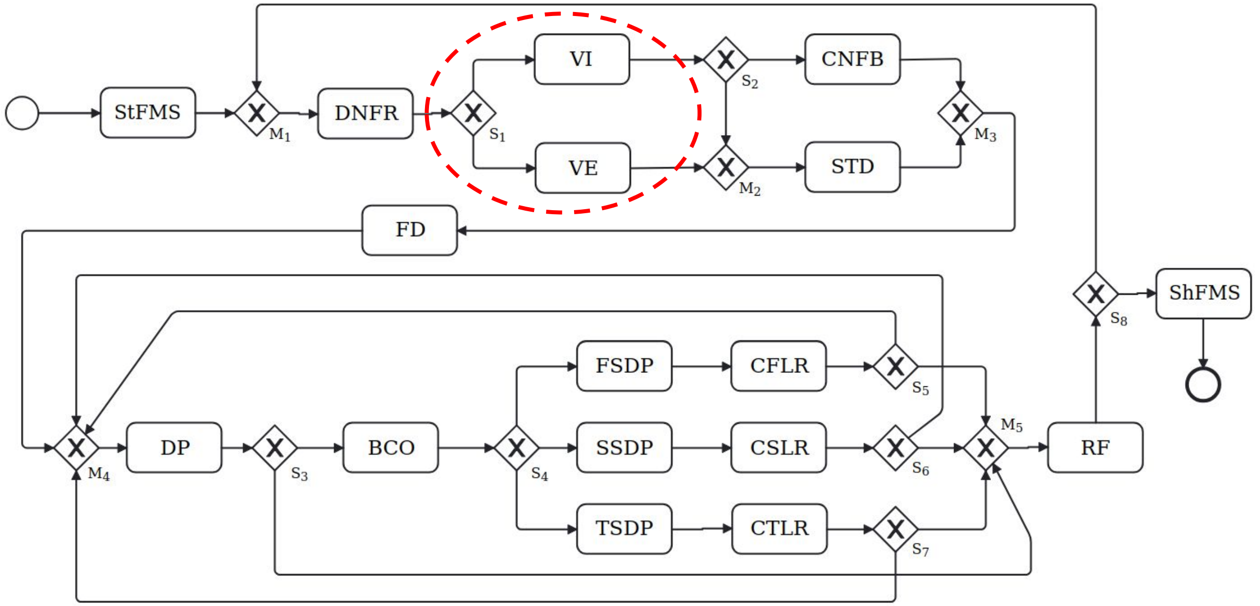
This can be verified by ensuring that **no path** of the process **contains these two tasks**. However, this is **quite restrictive**.



Detailed Approach – Step 6.1 – Mutual Exclusions

In BPMN, a mutual exclusion between two tasks can be defined as the **impossibility to perform both tasks** during the execution of the process.

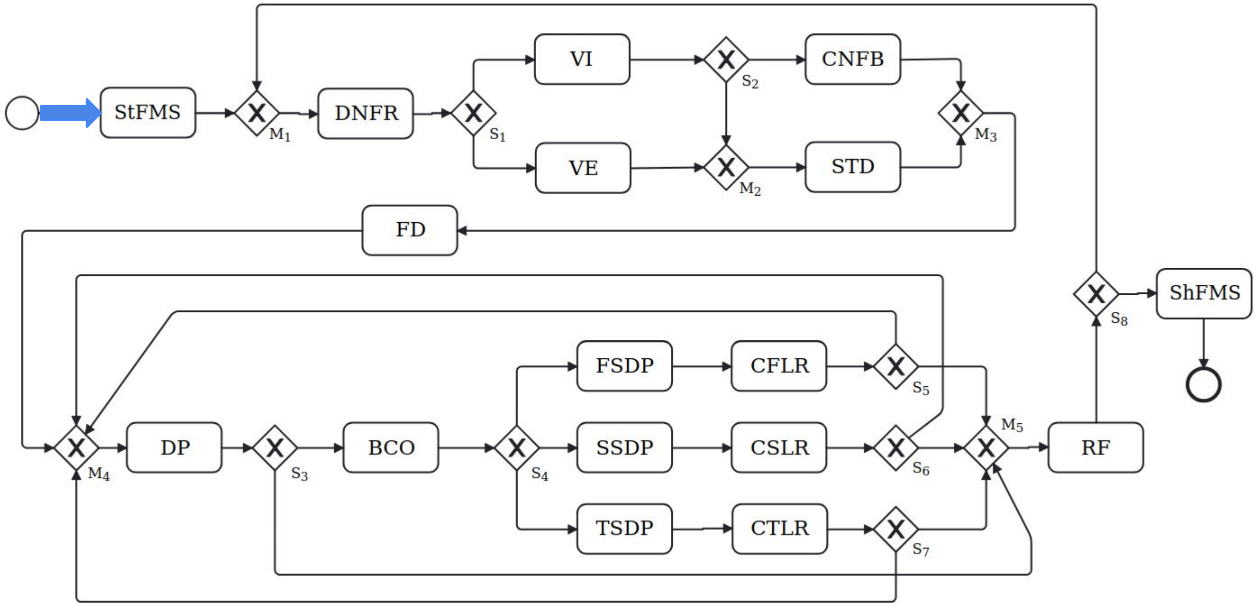
This can be verified by ensuring that **no path** of the process **contains these two tasks**. However, this is **quite restrictive**.



Detailed Approach – Step 6.1 – Mutual Exclusions

In BPMN, a mutual exclusion between two tasks can be defined as the **impossibility to perform both tasks** during the execution of the process.

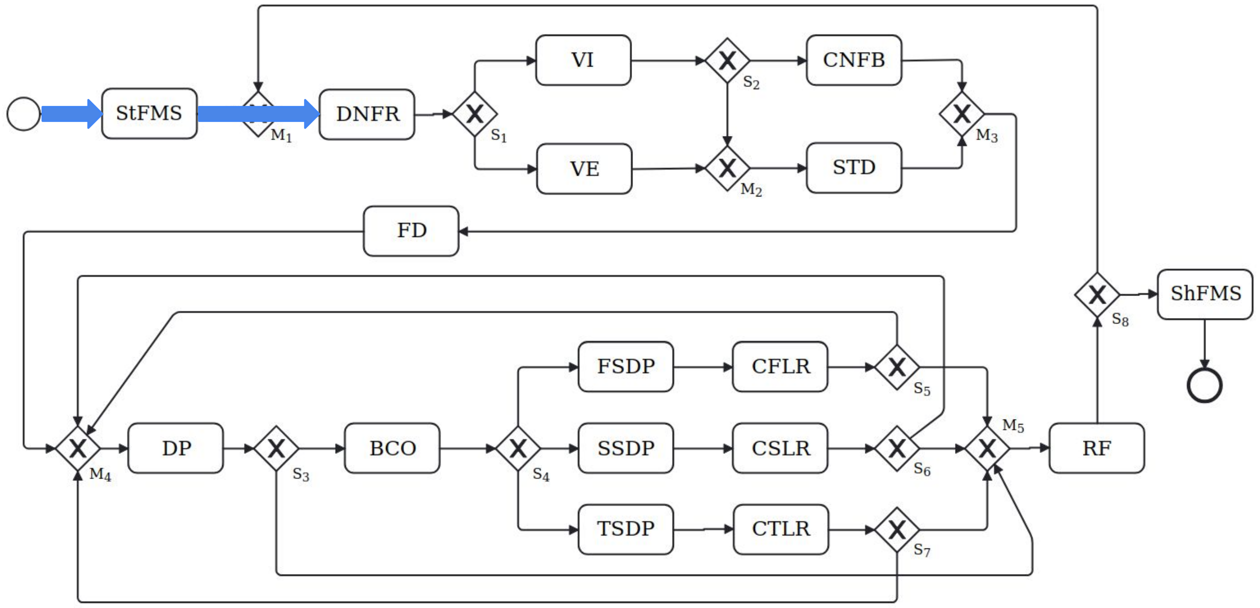
This can be verified by ensuring that **no path** of the process **contains these two tasks**. However, this is **quite restrictive**.



Detailed Approach – Step 6.1 – Mutual Exclusions

In BPMN, a mutual exclusion between two tasks can be defined as the **impossibility to perform both tasks** during the execution of the process.

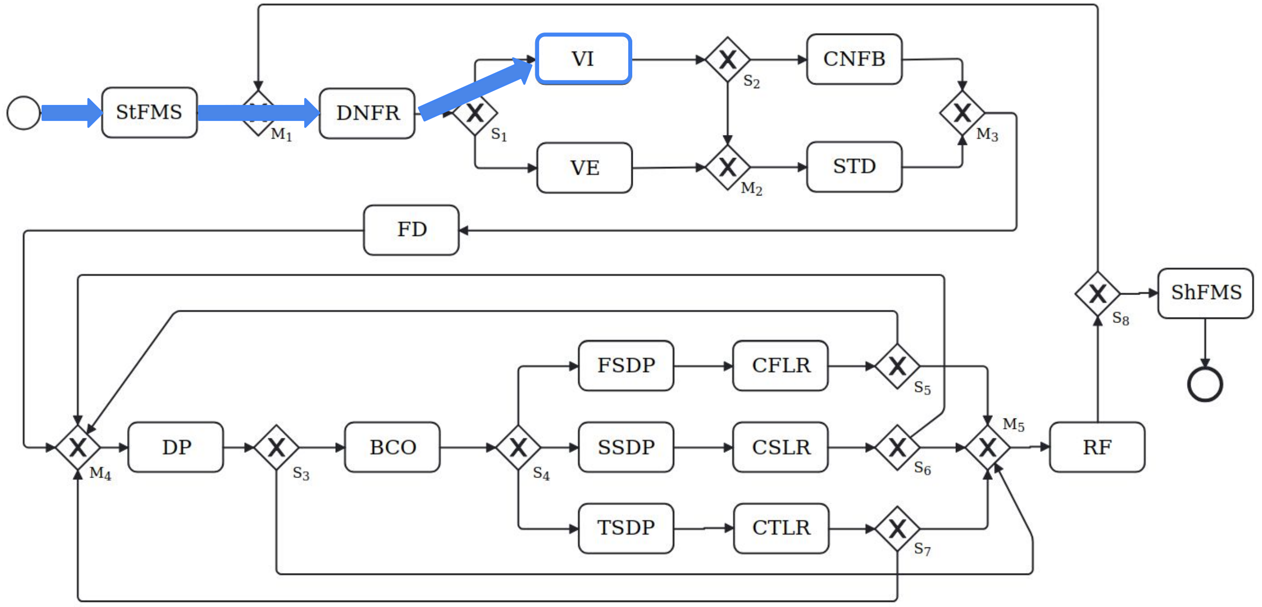
This can be verified by ensuring that **no path** of the process **contains these two tasks**. However, this is **quite restrictive**.



Detailed Approach – Step 6.1 – Mutual Exclusions

In BPMN, a mutual exclusion between two tasks can be defined as the **impossibility to perform both tasks** during the execution of the process.

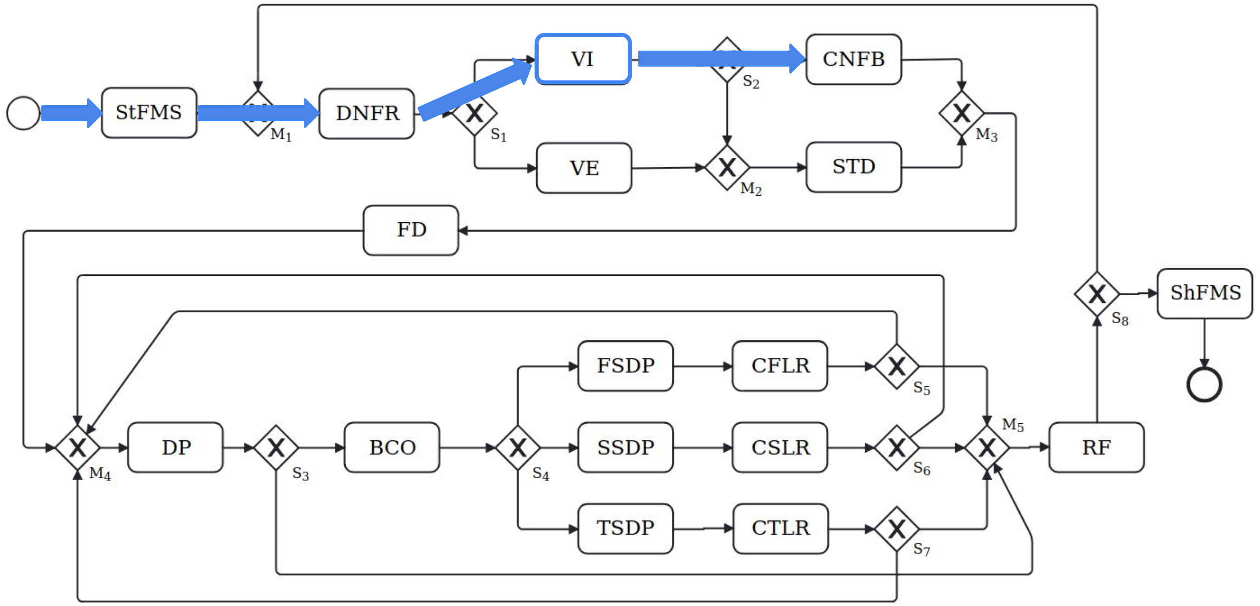
This can be verified by ensuring that **no path** of the process **contains these two tasks**. However, this is **quite restrictive**.



Detailed Approach – Step 6.1 – Mutual Exclusions

In BPMN, a mutual exclusion between two tasks can be defined as the **impossibility to perform both tasks** during the execution of the process.

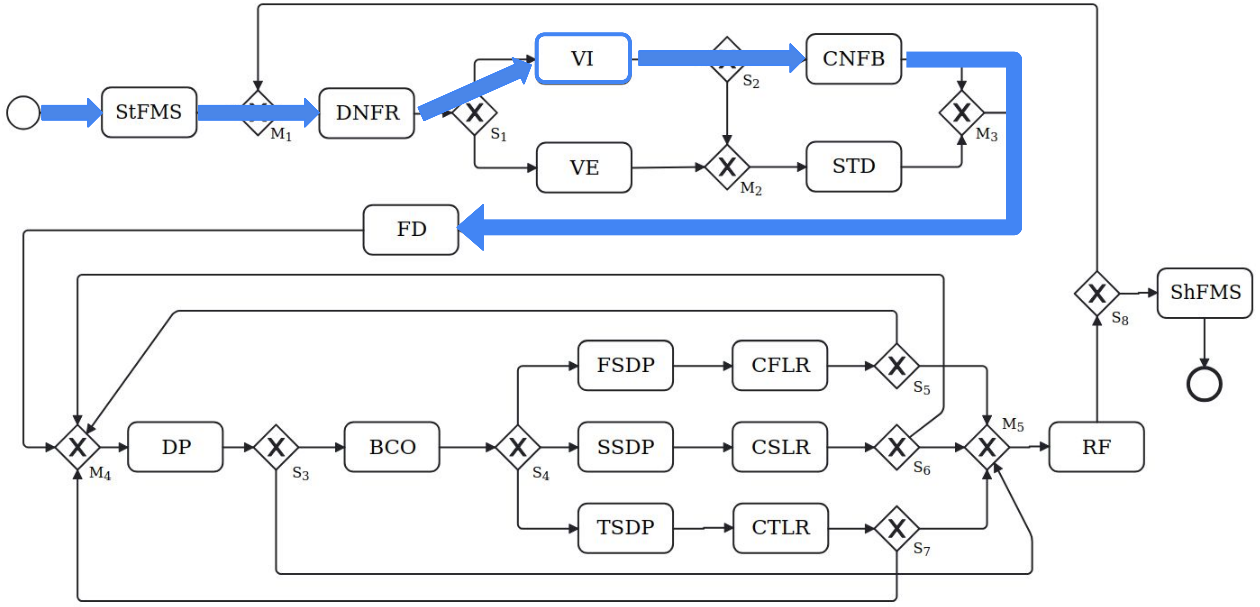
This can be verified by ensuring that **no path** of the process **contains these two tasks**. However, this is **quite restrictive**.



Detailed Approach – Step 6.1 – Mutual Exclusions

In BPMN, a mutual exclusion between two tasks can be defined as the **impossibility to perform both tasks** during the execution of the process.

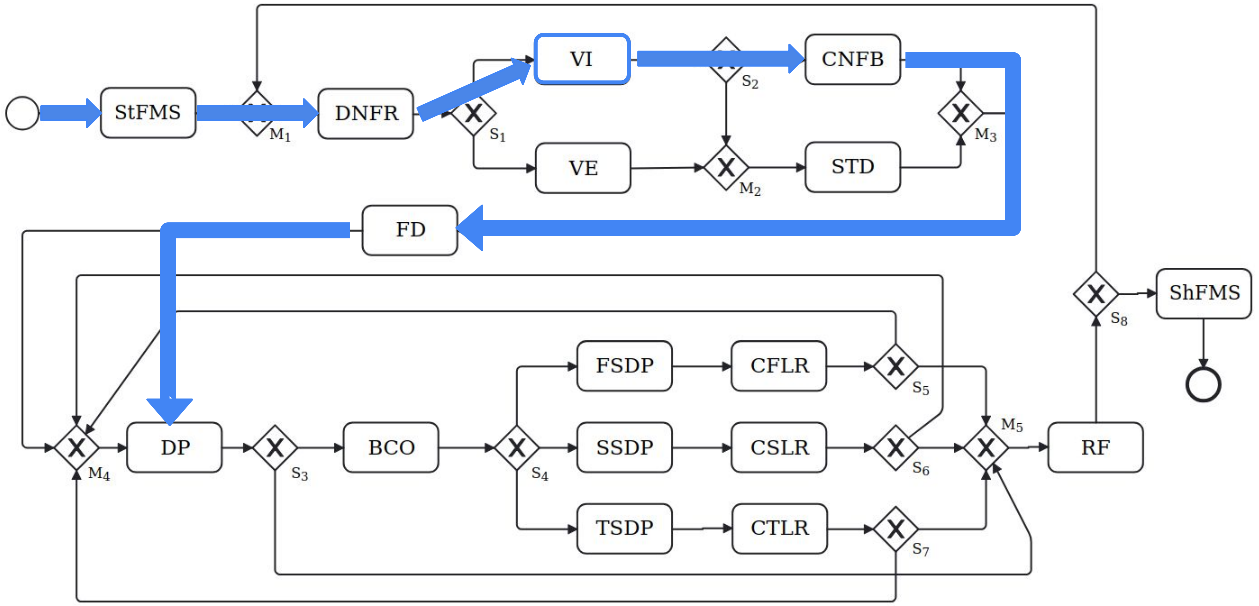
This can be verified by ensuring that **no path** of the process **contains these two tasks**. However, this is **quite restrictive**.



Detailed Approach – Step 6.1 – Mutual Exclusions

In BPMN, a mutual exclusion between two tasks can be defined as the **impossibility to perform both tasks** during the execution of the process.

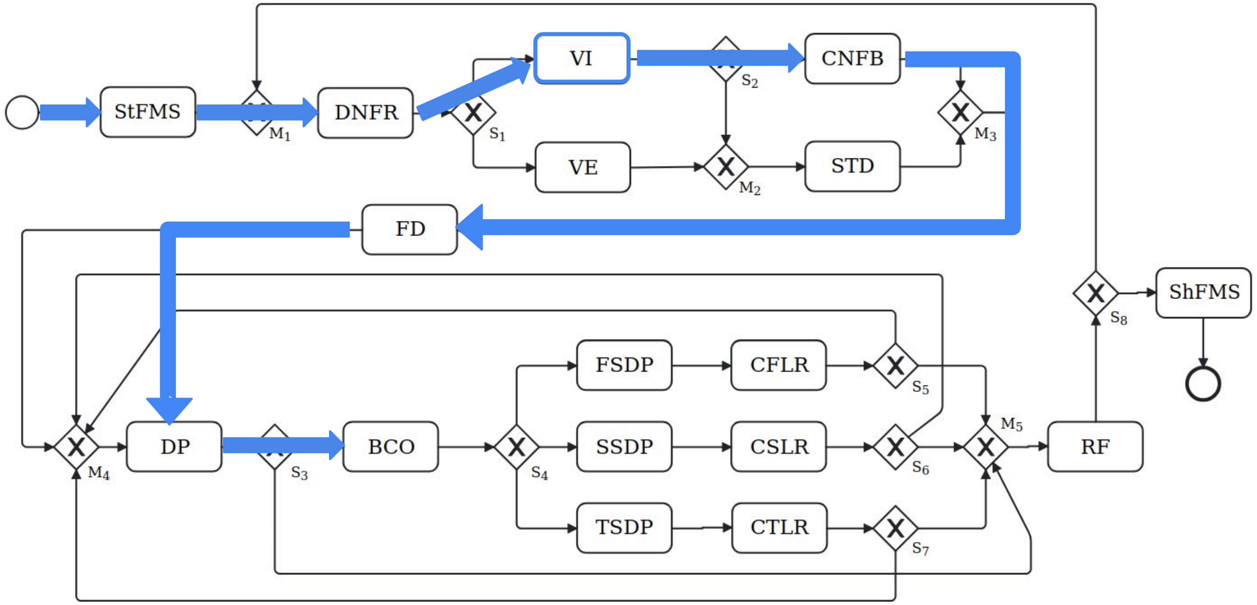
This can be verified by ensuring that **no path** of the process **contains these two tasks**. However, this is **quite restrictive**.



Detailed Approach – Step 6.1 – Mutual Exclusions

In BPMN, a mutual exclusion between two tasks can be defined as the **impossibility to perform both tasks** during the execution of the process.

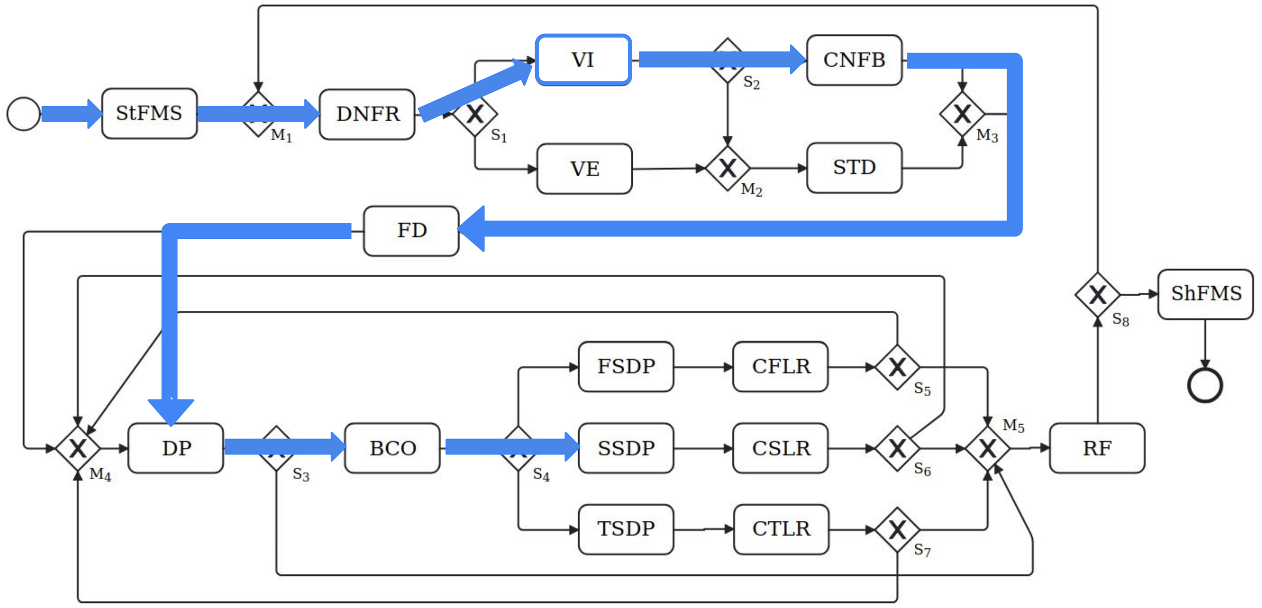
This can be verified by ensuring that **no path** of the process **contains these two tasks**. However, this is **quite restrictive**.



Detailed Approach – Step 6.1 – Mutual Exclusions

In BPMN, a mutual exclusion between two tasks can be defined as the **impossibility to perform both tasks** during the execution of the process.

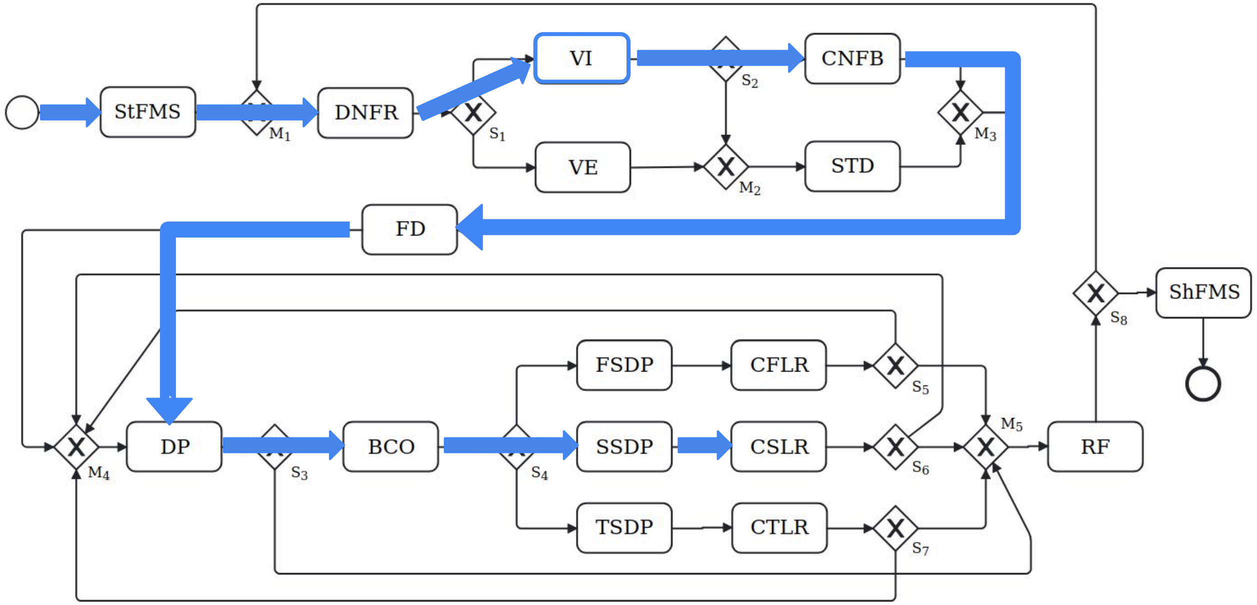
This can be verified by ensuring that **no path** of the process **contains these two tasks**. However, this is **quite restrictive**.



Detailed Approach – Step 6.1 – Mutual Exclusions

In BPMN, a mutual exclusion between two tasks can be defined as the **impossibility to perform both tasks** during the execution of the process.

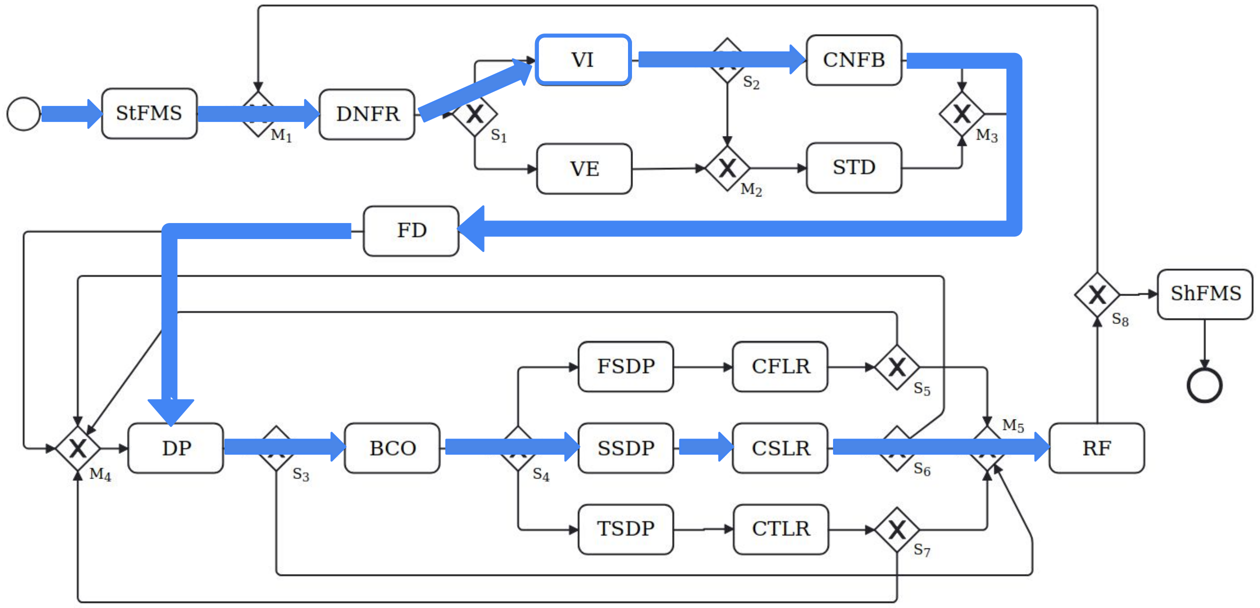
This can be verified by ensuring that **no path** of the process **contains these two tasks**. However, this is **quite restrictive**.



Detailed Approach – Step 6.1 – Mutual Exclusions

In BPMN, a mutual exclusion between two tasks can be defined as the **impossibility to perform both tasks** during the execution of the process.

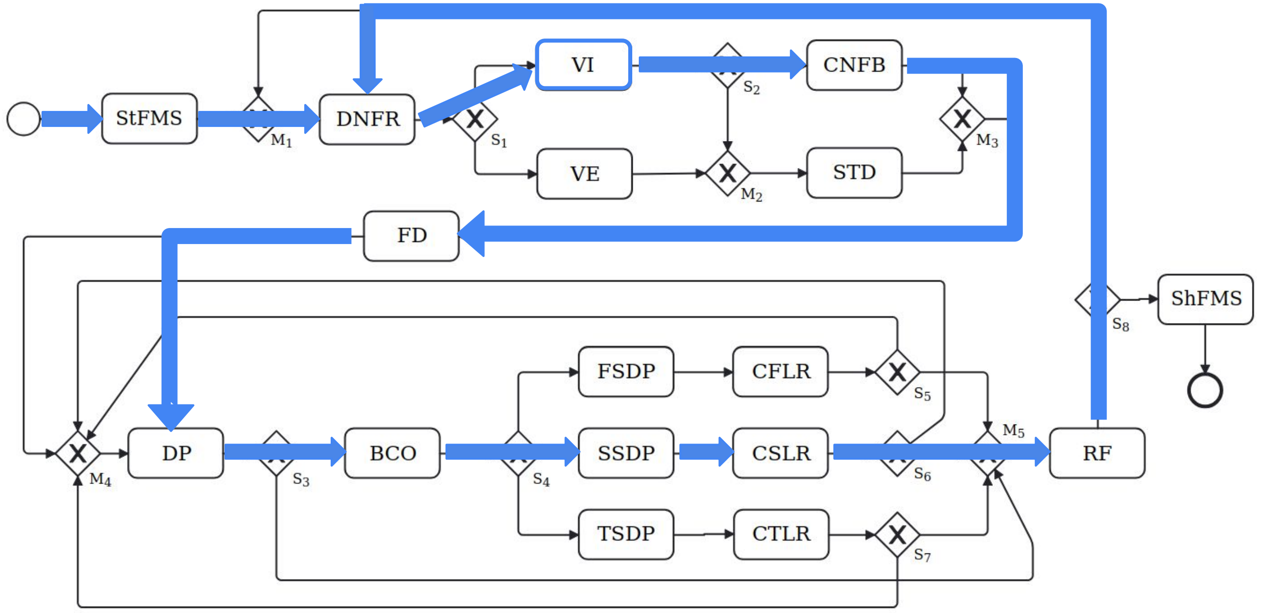
This can be verified by ensuring that **no path** of the process **contains these two tasks**. However, this is **quite restrictive**.



Detailed Approach – Step 6.1 – Mutual Exclusions

In BPMN, a mutual exclusion between two tasks can be defined as the **impossibility to perform both tasks** during the execution of the process.

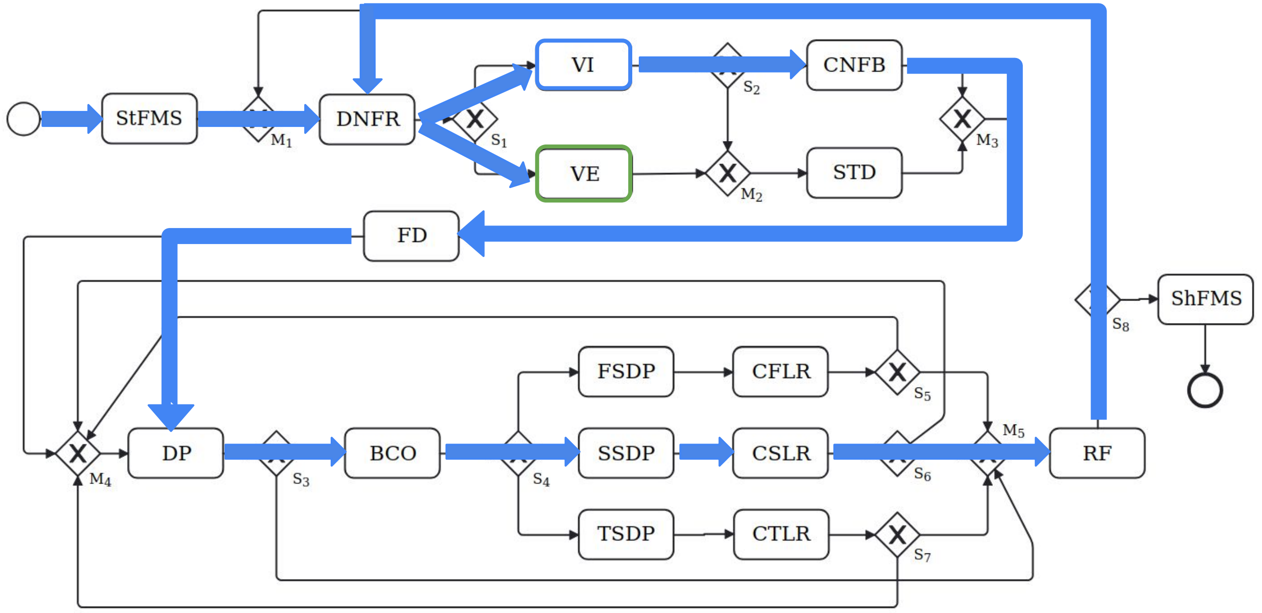
This can be verified by ensuring that **no path** of the process **contains these two tasks**. However, this is **quite restrictive**.



Detailed Approach – Step 6.1 – Mutual Exclusions

In BPMN, a mutual exclusion between two tasks can be defined as the **impossibility to perform both tasks** during the execution of the process.

This can be verified by ensuring that **no path** of the process **contains these two tasks**. However, this is **quite restrictive**.

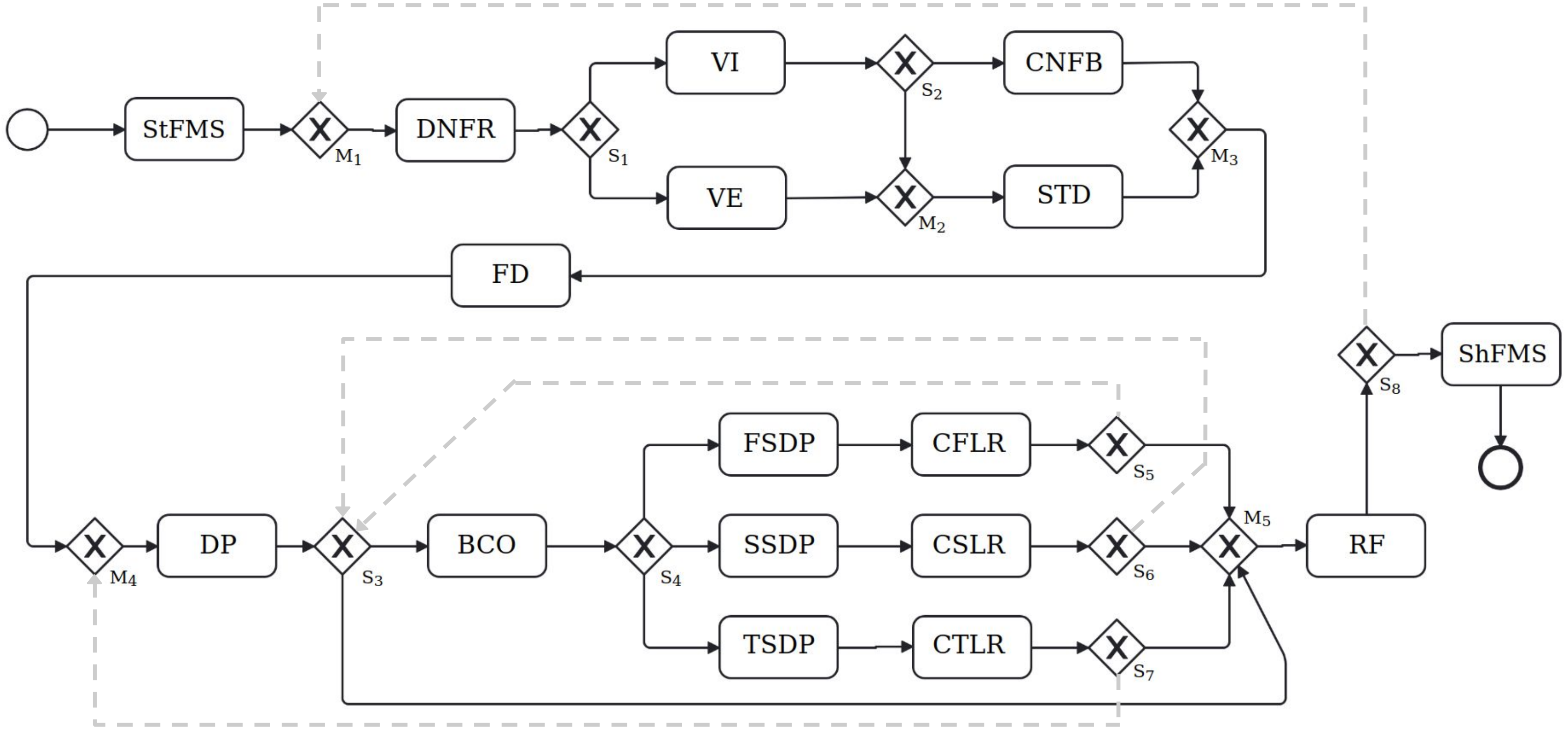


Detailed Approach – Step 6.1 – Mutual Exclusions

A **weaker** definition of **mutual exclusion** consists in considering only the **acyclic paths** of the process, instead of all the paths.

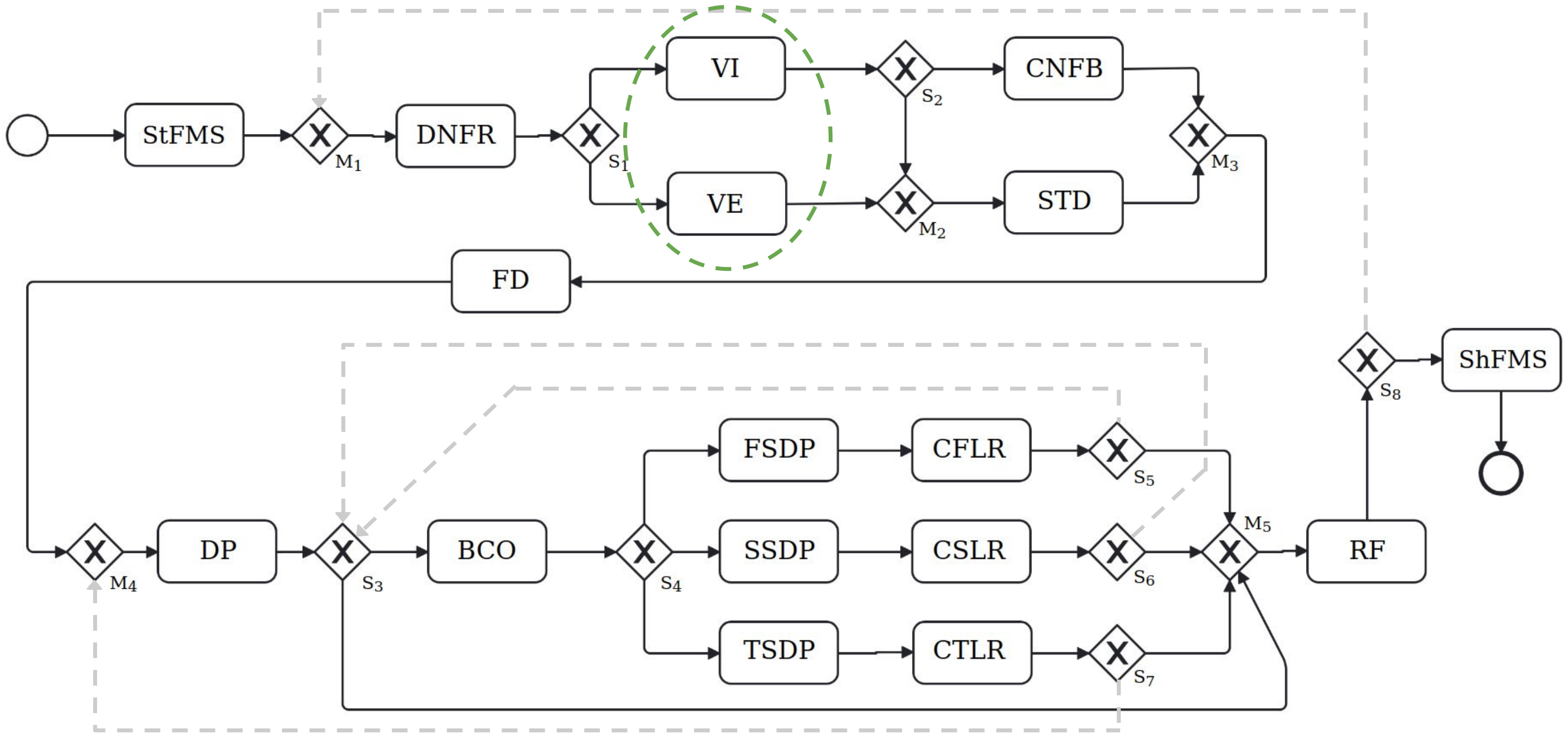
Detailed Approach – Step 6.1 – Mutual Exclusions

A **weaker** definition of **mutual exclusion** consists in considering only the **acyclic paths** of the process, instead of all the paths.



Detailed Approach – Step 6.1 – Mutual Exclusions

A **weaker** definition of **mutual exclusion** consists in considering only the **acyclic paths** of the process, instead of all the paths.



Detailed Approach – Step 6.1 – Mutual Exclusions

Inserting a **mutually exclusive task** to the process thus consist in ensuring that this **definition holds** for that task.

Detailed Approach – Step 6.1 – Mutual Exclusions

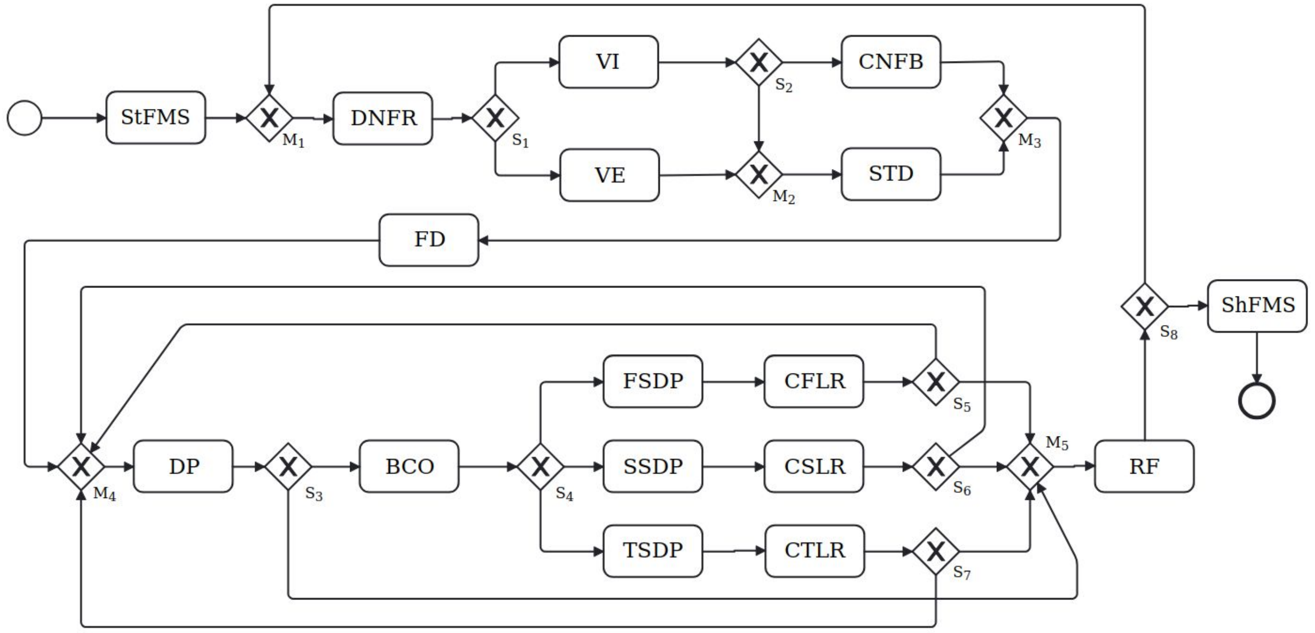
Inserting a **mutually exclusive task** to the process thus consist in ensuring that this **definition holds** for that task.

To do so, we **connect the task** to insert to one of the **closest inevitable common ancestor** of the mutually exclusive tasks.

Detailed Approach – Step 6.1 – Mutual Exclusions

Inserting a **mutually exclusive task** to the process thus consist in ensuring that this **definition holds** for that task.

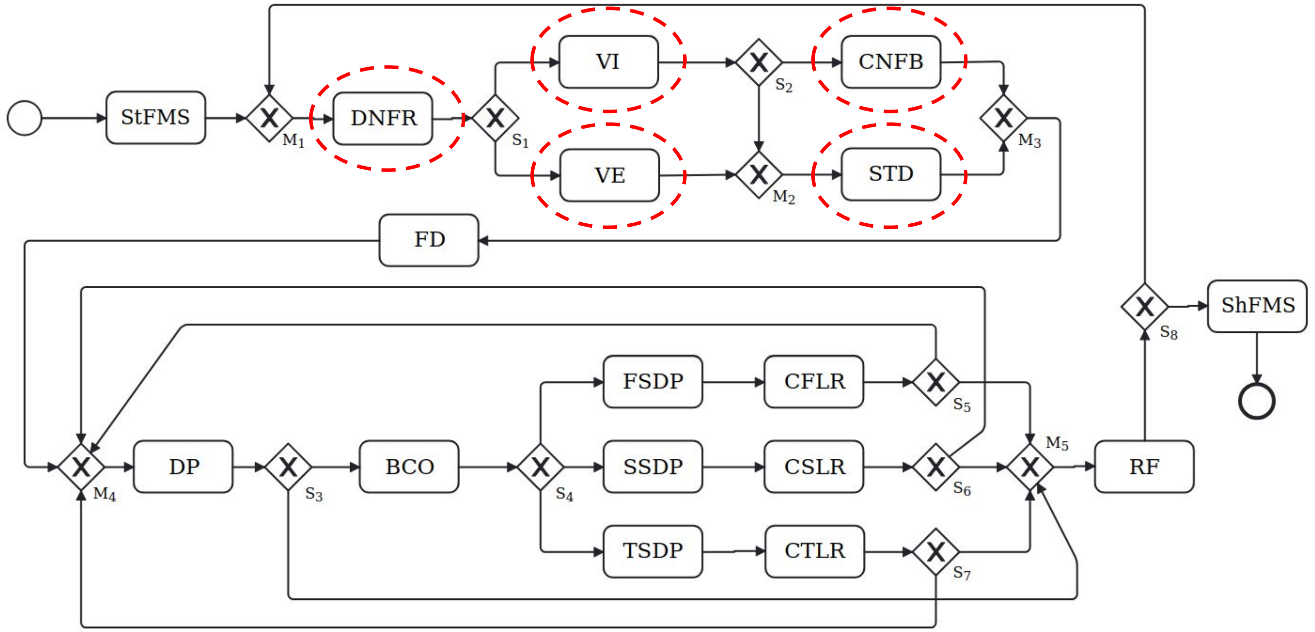
To do so, we **connect the task** to insert to one of the **closest inevitable common ancestor** of the mutually exclusive tasks.



Detailed Approach – Step 6.1 – Mutual Exclusions

Inserting a **mutually exclusive task** to the process thus consist in ensuring that this **definition holds** for that task.

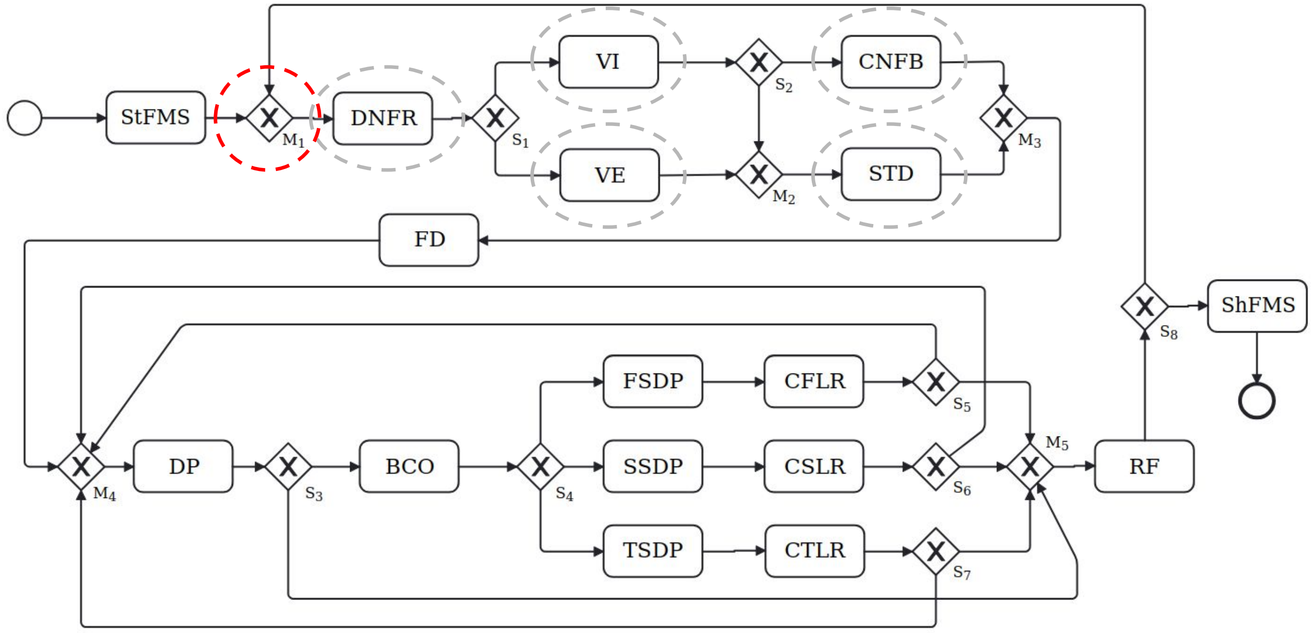
To do so, we **connect the task** to insert to one of the **closest inevitable common ancestor** of the mutually exclusive tasks.



Detailed Approach – Step 6.1 – Mutual Exclusions

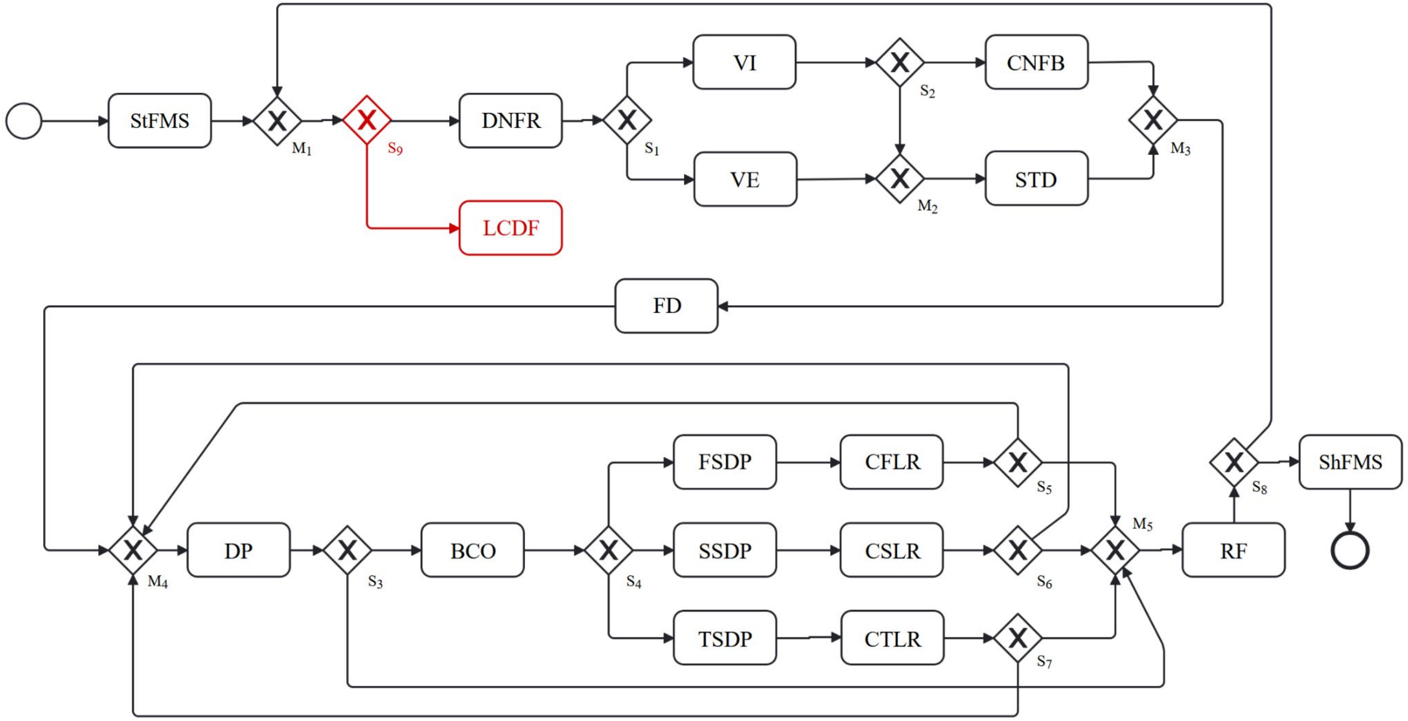
Inserting a **mutually exclusive task** to the process thus consist in ensuring that this **definition holds** for that task.

To do so, we **connect the task** to insert to one of the **closest inevitable common ancestor** of the mutually exclusive tasks.



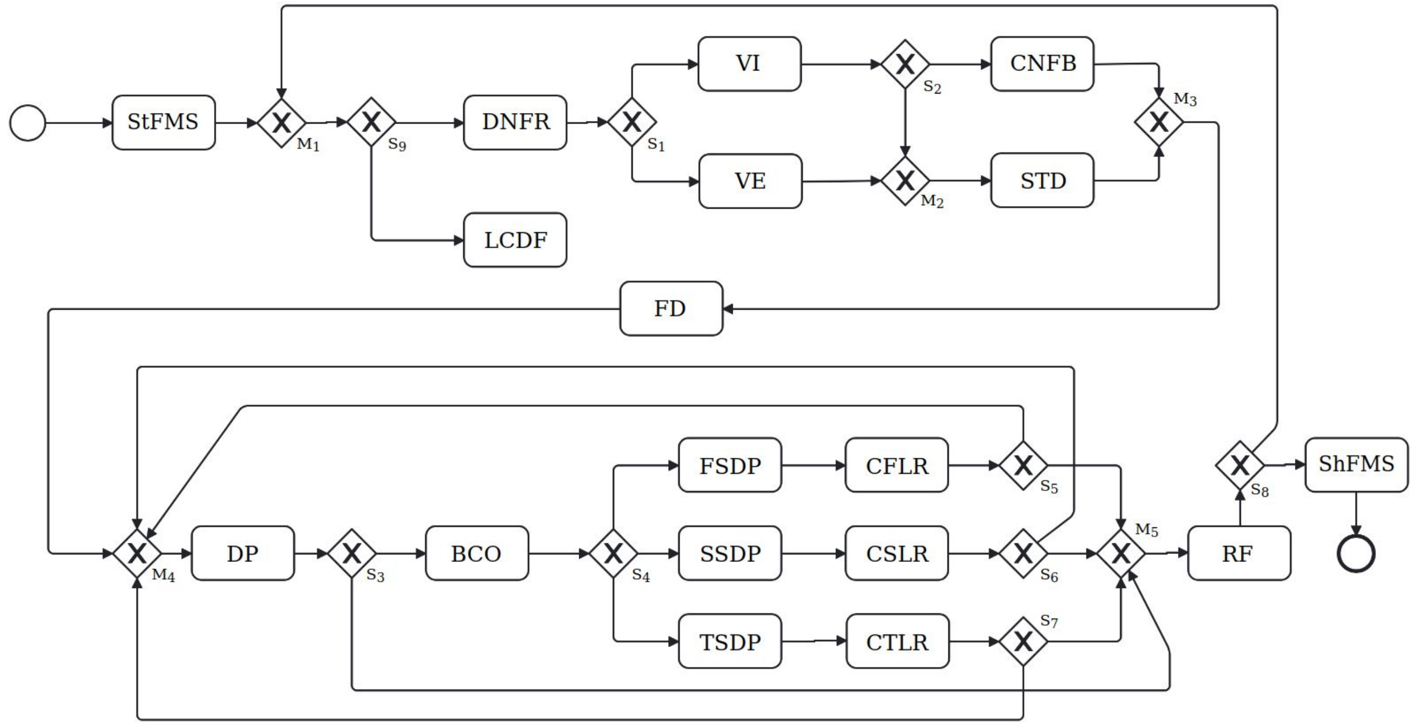
Detailed Approach – Step 6.1 – Mutual Exclusions

Adding the task **LCDF** as child of this gateway modifies the process so as to make tasks **DNFR**, **VI**, **VE**, **CNFB** and **STD** mutually exclusive of task **LCDF**.



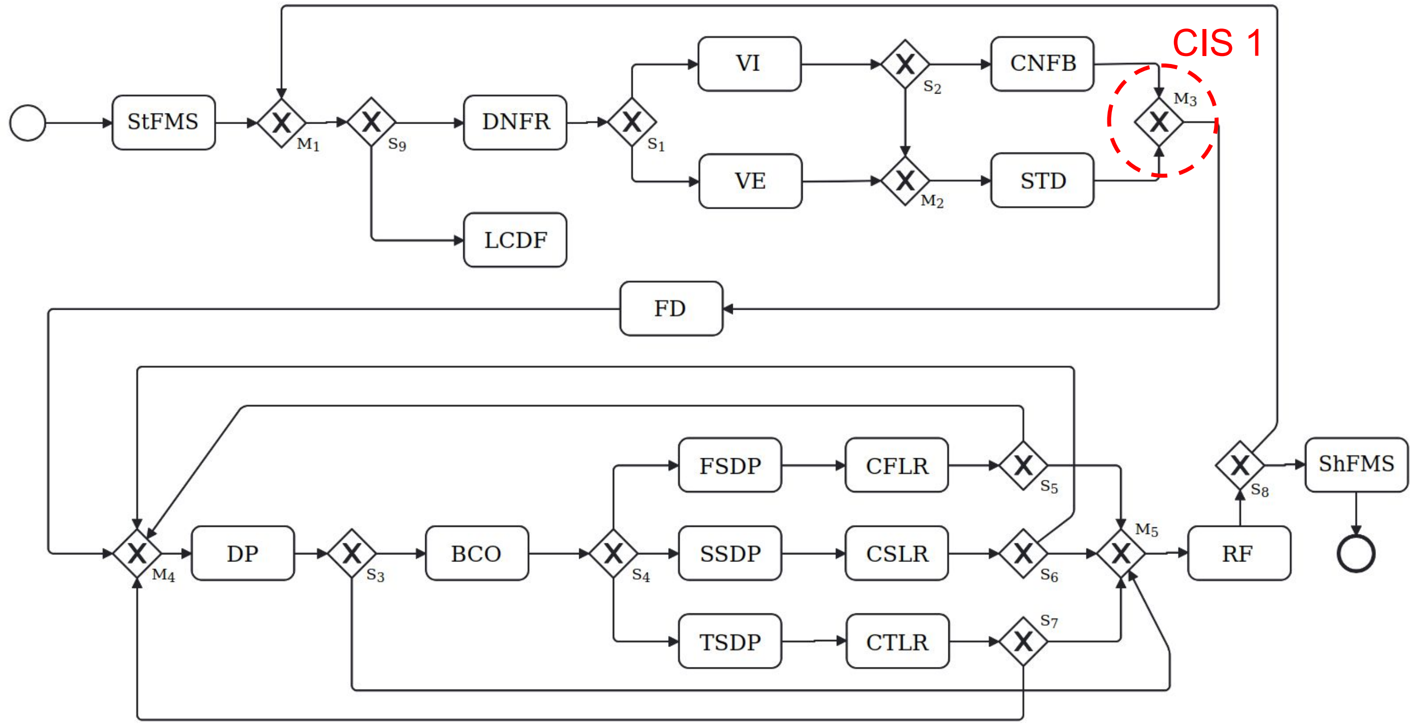
Detailed Approach – Step 6.1 – Mutual Exclusions

In a manner similar to the insertion of the task, the **common inevitable successors (CIS)** of the mutually exclusive tasks are **analysed successively**.



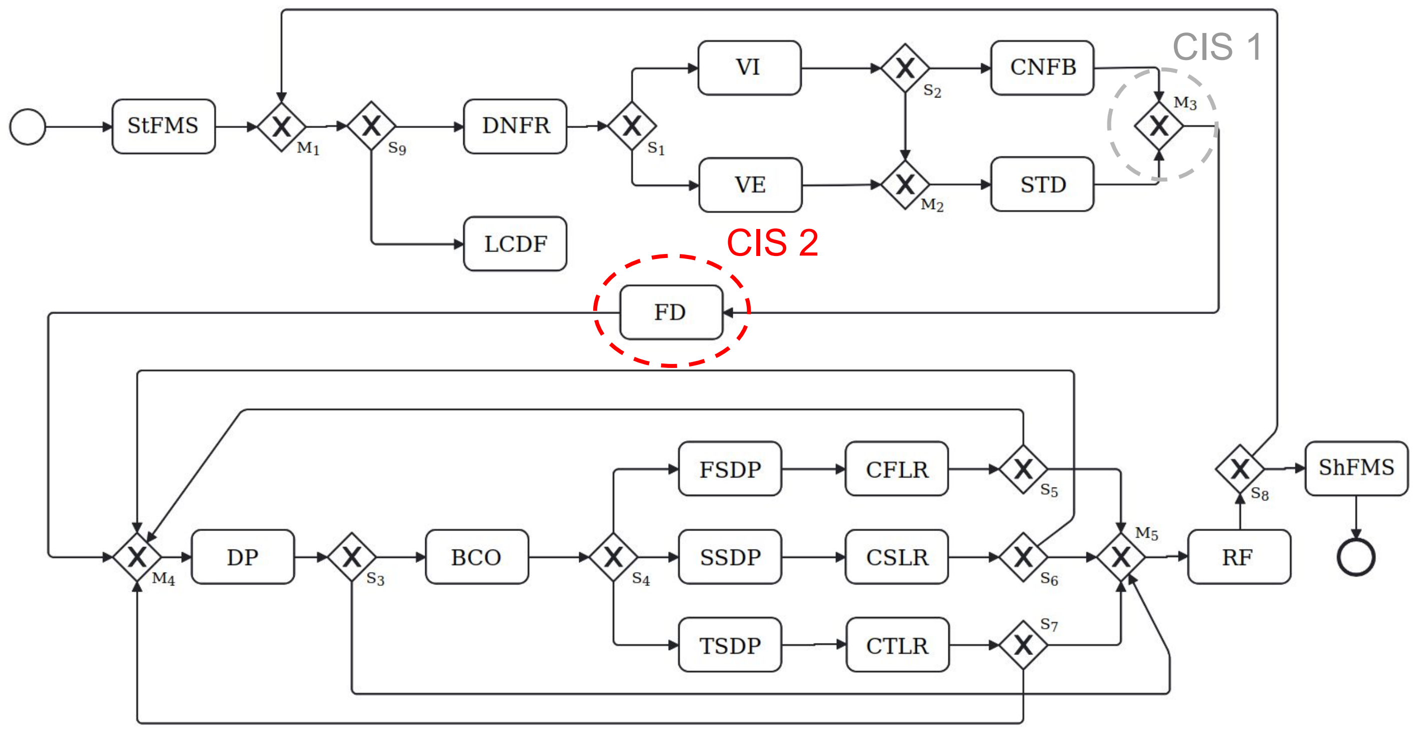
Detailed Approach – Step 6.1 – Mutual Exclusions

In a manner similar to the insertion of the task, the **common inevitable successors (CIS)** of the mutually exclusive tasks are **analysed successively**.



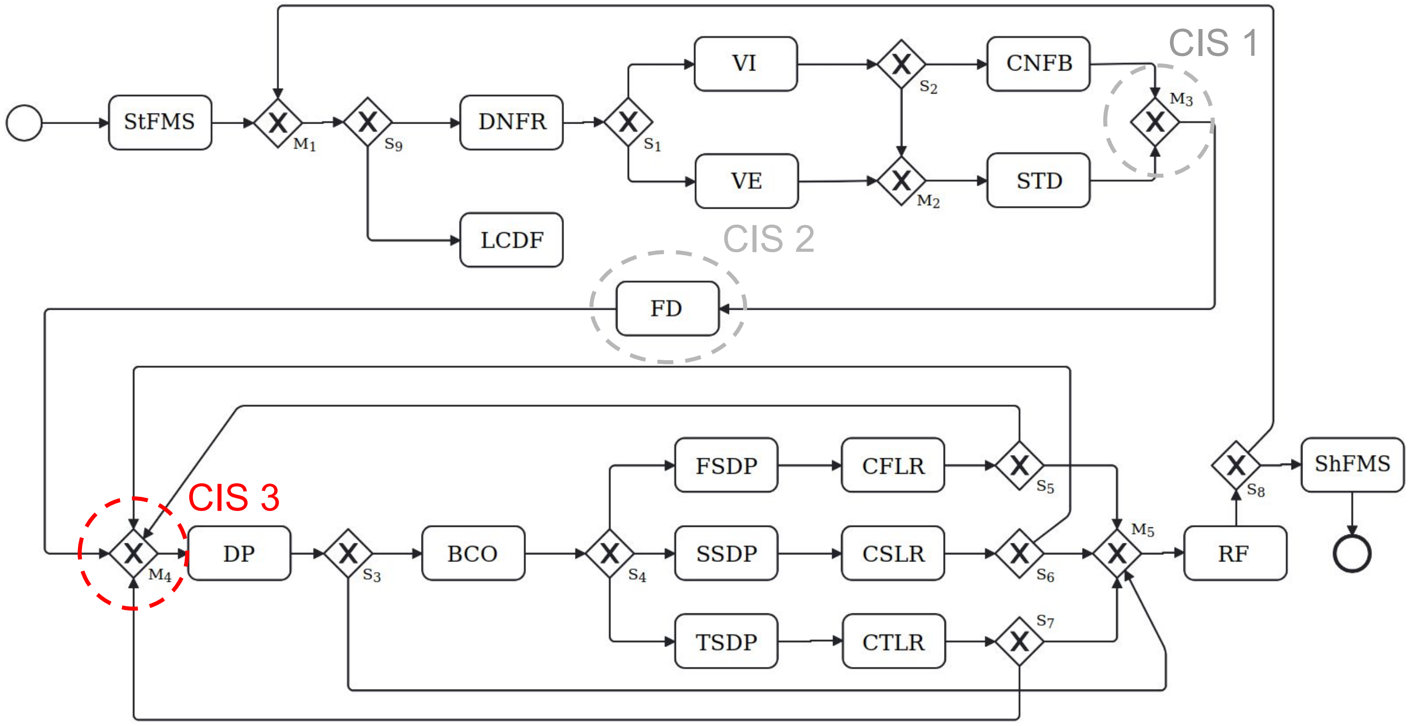
Detailed Approach – Step 6.1 – Mutual Exclusions

In a manner similar to the insertion of the task, the **common inevitable successors (CIS)** of the mutually exclusive tasks are **analysed successively**.



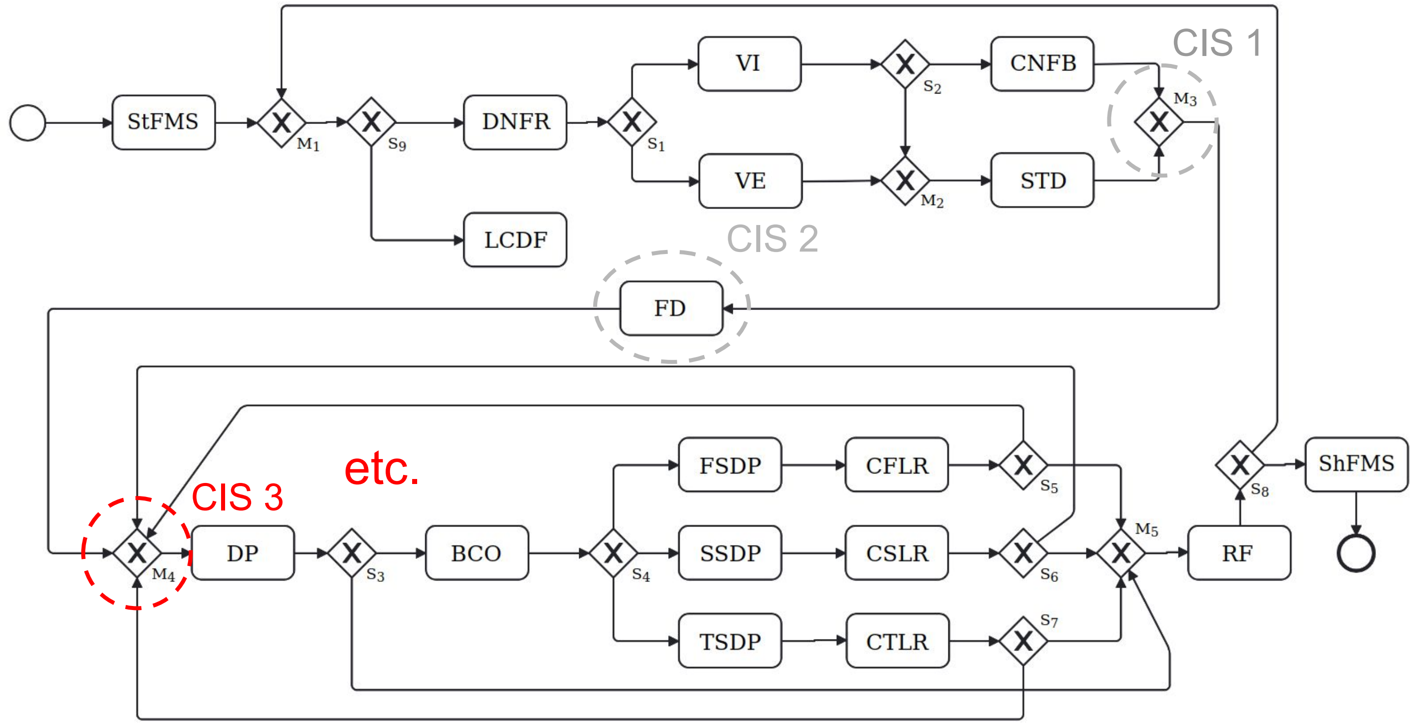
Detailed Approach – Step 6.1 – Mutual Exclusions

In a manner similar to the insertion of the task, the **common inevitable successors (CIS)** of the mutually exclusive tasks are **analysed successively**.



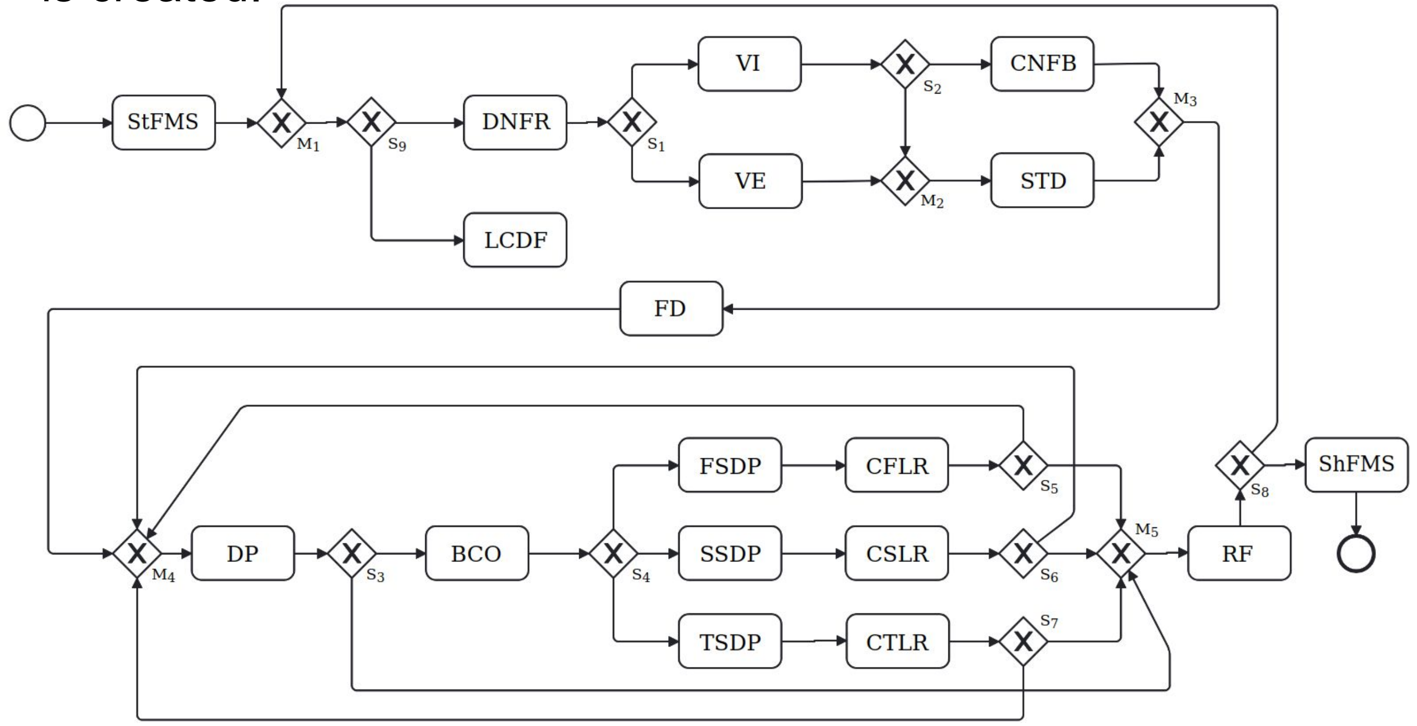
Detailed Approach – Step 6.1 – Mutual Exclusions

In a manner similar to the insertion of the task, the **common inevitable successors (CIS)** of the mutually exclusive tasks are **analysed successively**.



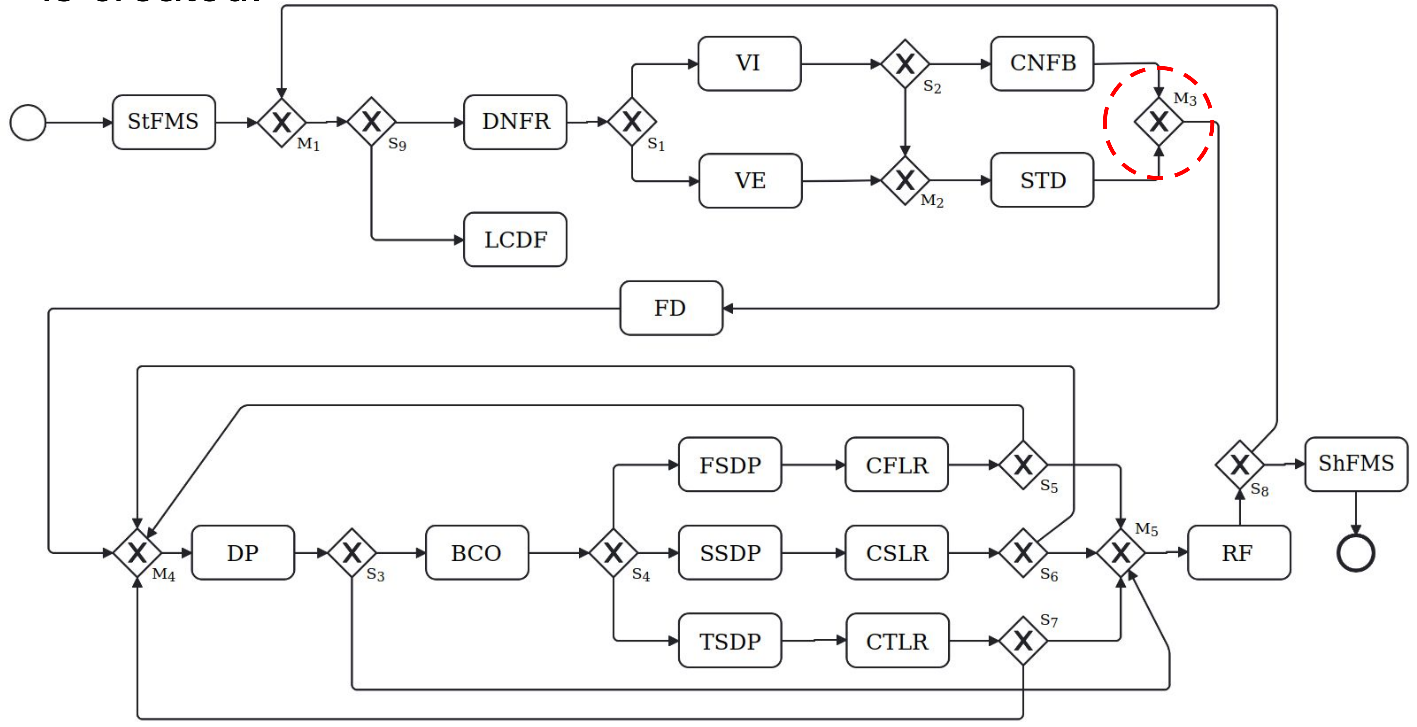
Detailed Approach – Step 6.1 – Mutual Exclusions

The first such successor whose connection **does not break the mutual exclusion** (if any) is selected, and an edge between it and the inserted task is created.



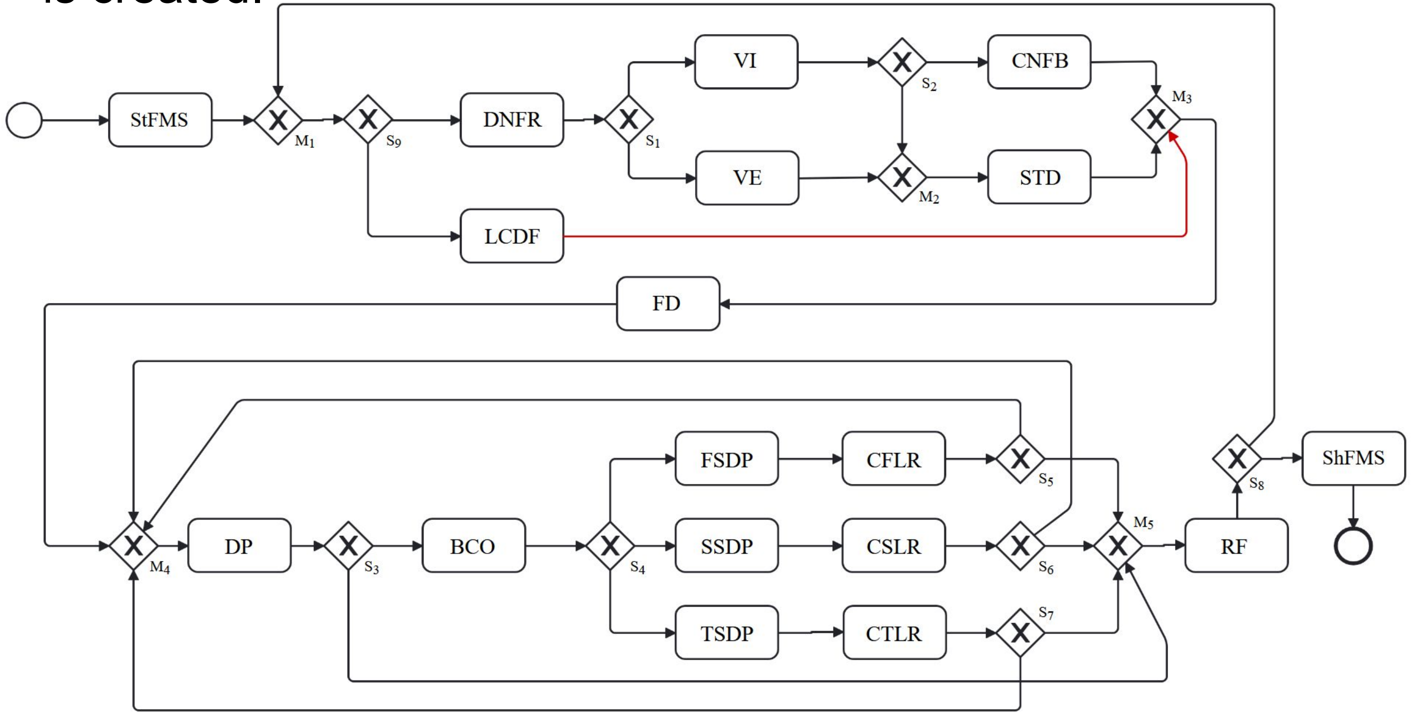
Detailed Approach – Step 6.1 – Mutual Exclusions

The first such successor whose connection **does not break the mutual exclusion** (if any) is selected, and an edge between it and the inserted task is created.



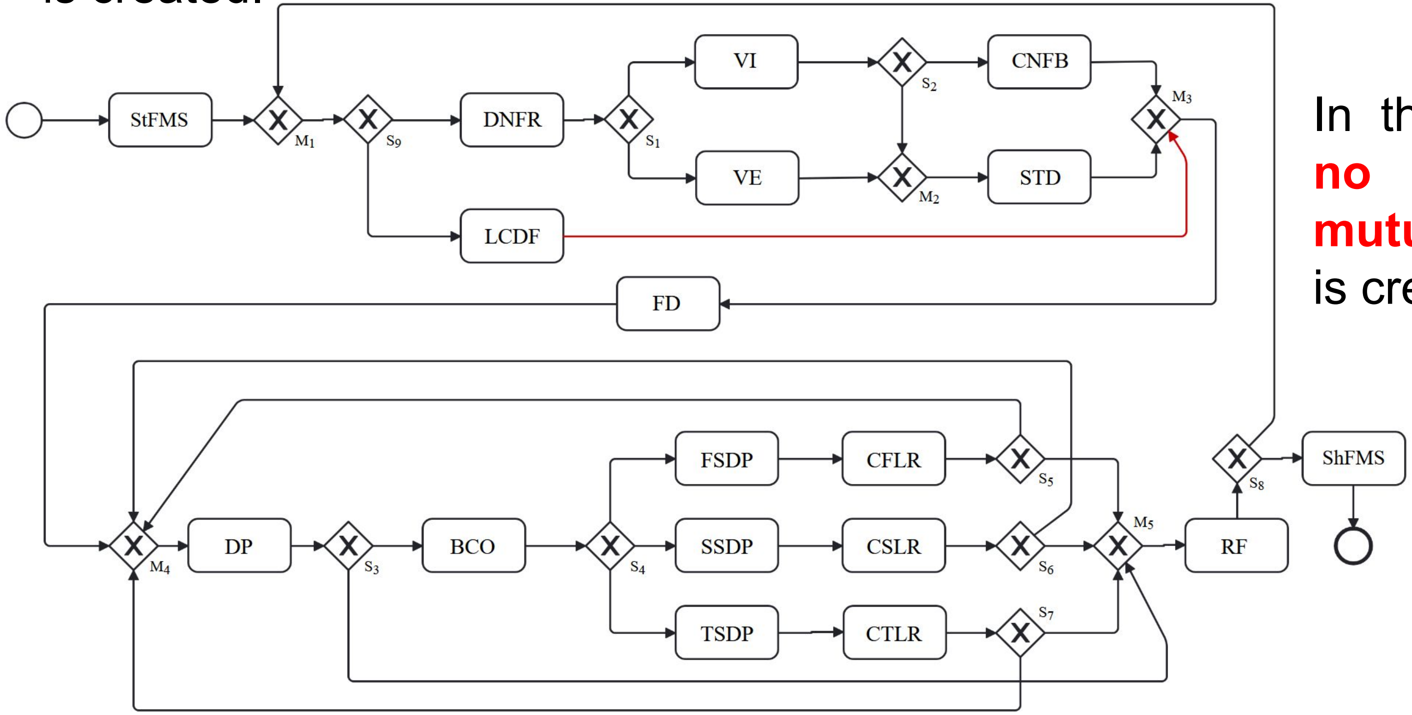
Detailed Approach – Step 6.1 – Mutual Exclusions

The first such successor whose connection **does not break the mutual exclusion** (if any) is selected, and an edge between it and the inserted task is created.



Detailed Approach – Step 6.1 – Mutual Exclusions

The first such successor whose connection **does not break the mutual exclusion** (if any) is selected, and an edge between it and the inserted task is created.



In the best case, **no unnecessary mutual exclusion** is created.

Detailed Approach – Step 6.2 – Explicit Loops

The second refinement step consists in inserting the **loops explicitly stated** by the user to the graph.

Detailed Approach – Step 6.2 – Explicit Loops

The second refinement step consists in inserting the **loops explicitly stated** by the user to the graph. This is for instance the case of expression

```
(FSDP, SSDP, TSDP, CFLR, CSLR, CTLR)*
```

which states that tasks **FSDP**, **SSDP**, **TSDP**, **CFLR**, **CSLR**, and **CTLR** must **appear in the same loop**.

Detailed Approach – Step 6.2 – Explicit Loops

The second refinement step consists in inserting the **loops explicitly stated** by the user to the graph. This is for instance the case of expression

(FSDP, SSDP, TSDP, CFLR, CSLR, CTLR)∗

which states that tasks **FSDP**, **SSDP**, **TSDP**, **CFLR**, **CSLR**, and **CTLR** must **appear in the same loop**.

Our proposal thus consists in connecting these tasks in the **restriction of the BPMN process** to the subset of its vertices corresponding to the loop.

Detailed Approach – Step 6.2 – Explicit Loops

The second refinement step consists in inserting the **loops explicitly stated** by the user to the graph. This is for instance the case of expression

(FSDP, SSDP, TSDP, CFLR, CSLR, CTLR)*

which states that tasks **FSDP**, **SSDP**, **TSDP**, **CFLR**, **CSLR**, and **CTLR** must **appear in the same loop**.

Our proposal thus consists in connecting these tasks in the **restriction of the BPMN process** to the subset of its vertices corresponding to the loop.

$$G \restriction_{\{v_1,\ldots,v_n\}} \overset{\text{def}}{=} (V^{\restriction}, E^{\restriction}, \Sigma^{\restriction}) \ \textit{where}$$

- $V^{\restriction} = \{v_1, \ldots, v_n\} \subseteq V$
- $E^{\restriction} = \{v \rightarrow v' \in E \mid v, v' \in V^{\restriction}\}$
- $\Sigma^{\restriction} = \{l \in \Sigma \mid \exists v^{\restriction} \in V^{\restriction} \ s.t. \ \sigma(v^{\restriction}) = l\}$

is the restriction of G *to the subset* $\{v_1, \ldots, v_n\}$ *of its vertices;*

Detailed Approach – Step 6.2 – Explicit Loops

Given our BPMN process, its restriction to the tasks belonging to expression

$$\boxed{(\mathrm{FSDP}, \mathrm{SSDP}, \mathrm{TSDP}, \mathrm{CFLR}, \mathrm{CSLR}, \mathrm{CTLR})*}$$

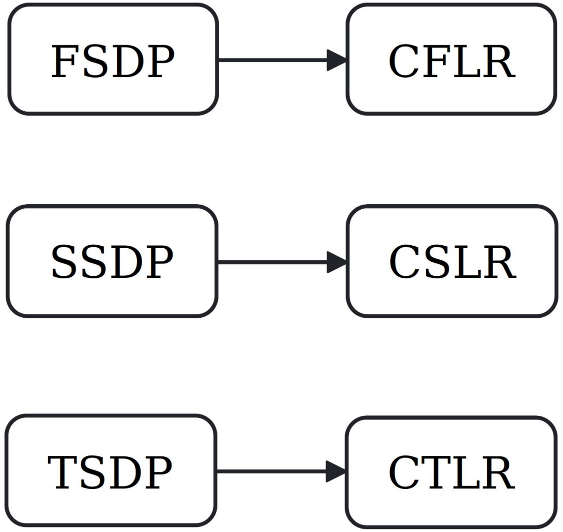
is:

Detailed Approach – Step 6.2 – Explicit Loops

Given our BPMN process, its restriction to the tasks belonging to expression

(FSDP, SSDP, TSDP, CFLR, CSLR, CTLR)∗

is:



Detailed Approach – Step 6.2 – Explicit Loops

These **disconnected portions** of the process have to be **connected** to make the **loop appear** in the process.

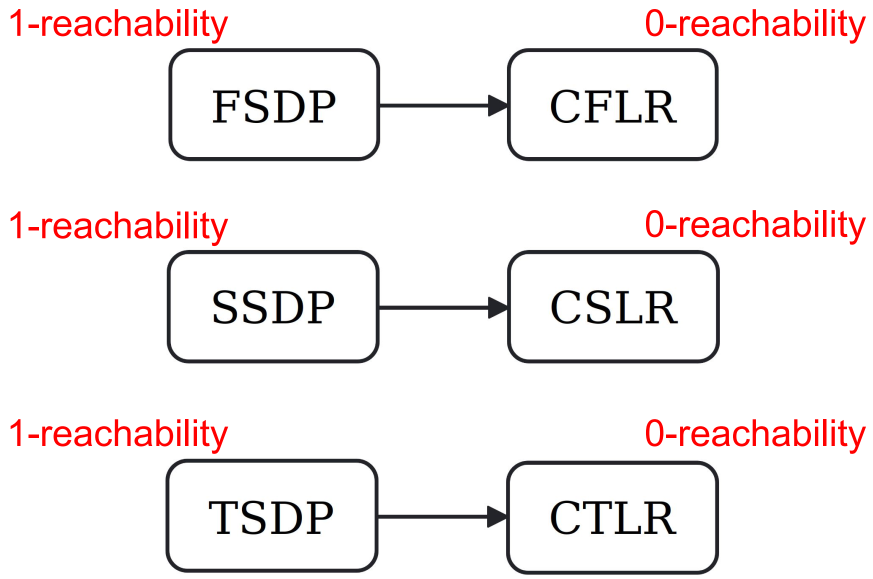
Detailed Approach – Step 6.2 – Explicit Loops

These **disconnected portions** of the process have to be **connected** to make the **loop appear** in the process.

This connection is based on the ***n*-reachability** of each node, i.e., the number of nodes that they can reach.

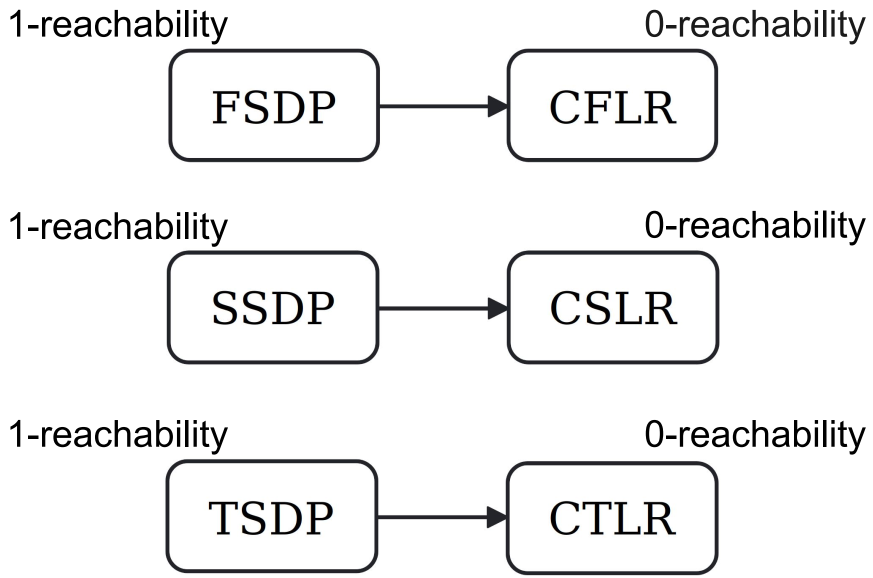
Detailed Approach – Step 6.2 – Explicit Loops

These **disconnected portions** of the process have to be **connected** to make the **loop appear** in the process.
This connection is based on the ***n*-reachability** of each node, i.e., the number of nodes that they can reach.



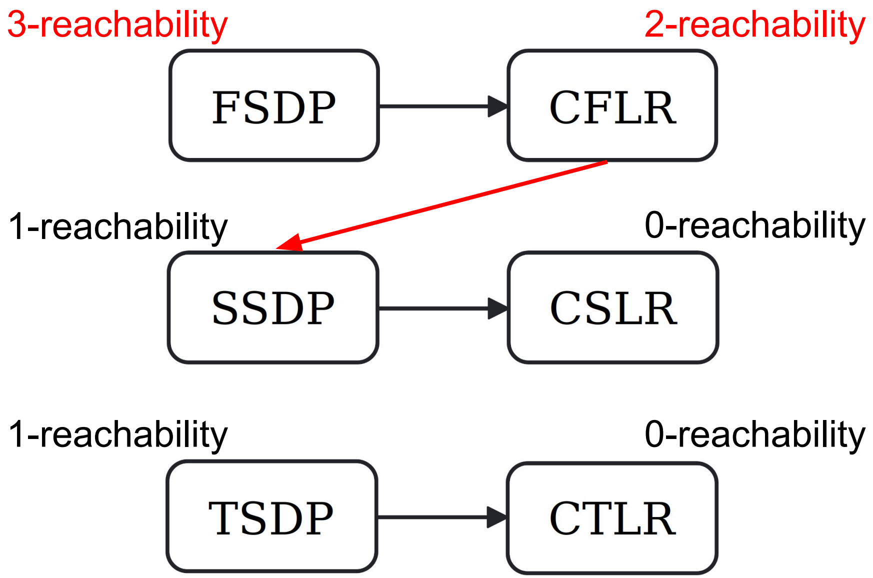
Detailed Approach – Step 6.2 – Explicit Loops

The connection consists in adding **an edge** between **each node** of a sub-graph having a **0-reachability**, and the **smallest subset of nodes** of another sub-graph ensuring a **total reachability**.



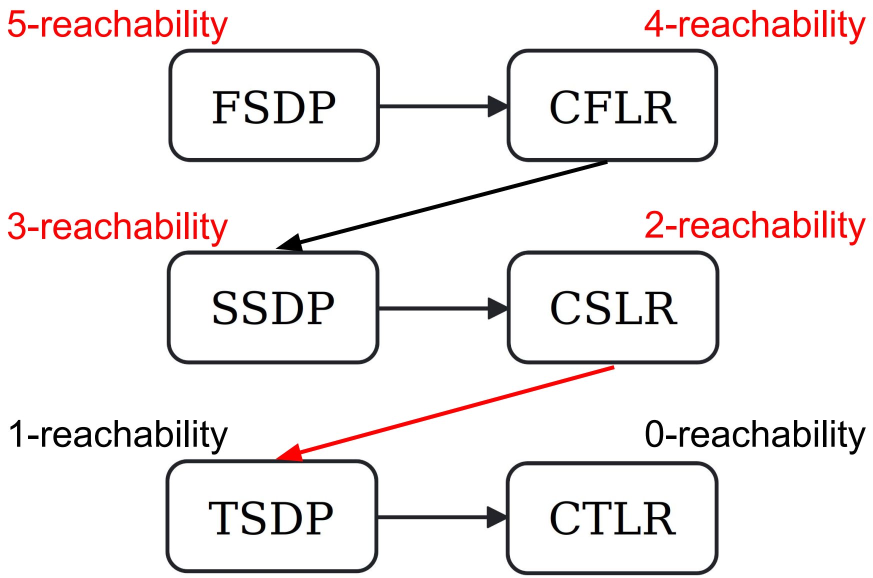
Detailed Approach – Step 6.2 – Explicit Loops

The connection consists in adding **an edge** between **each node** of a sub-graph having a **0-reachability**, and the **smallest subset of nodes** of another sub-graph ensuring a **total reachability**.



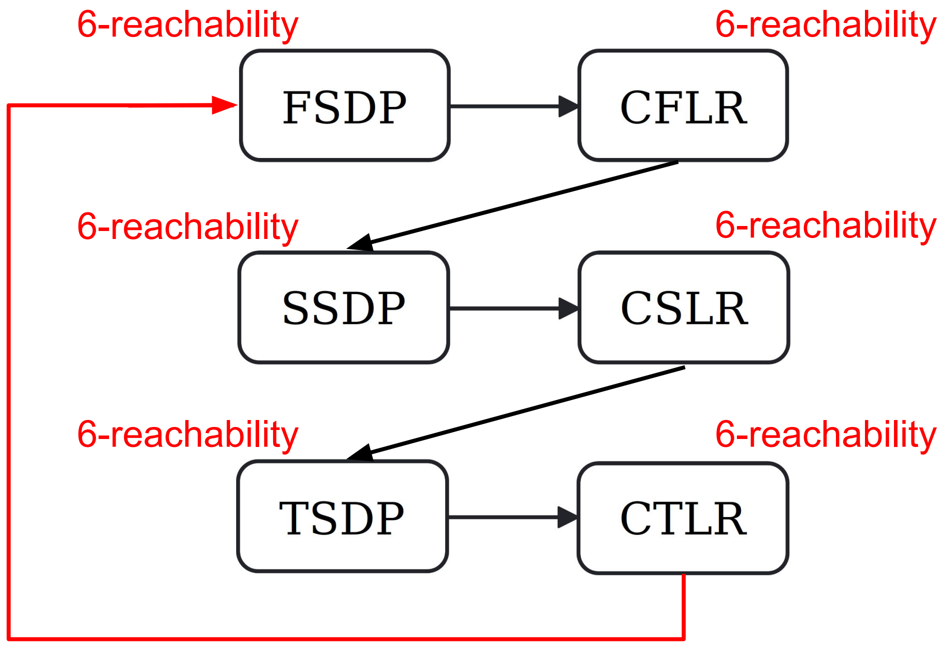
Detailed Approach – Step 6.2 – Explicit Loops

The connection consists in adding **an edge** between **each node** of a sub-graph having a **0-reachability**, and the **smallest subset of nodes** of another sub-graph ensuring a **total reachability**.



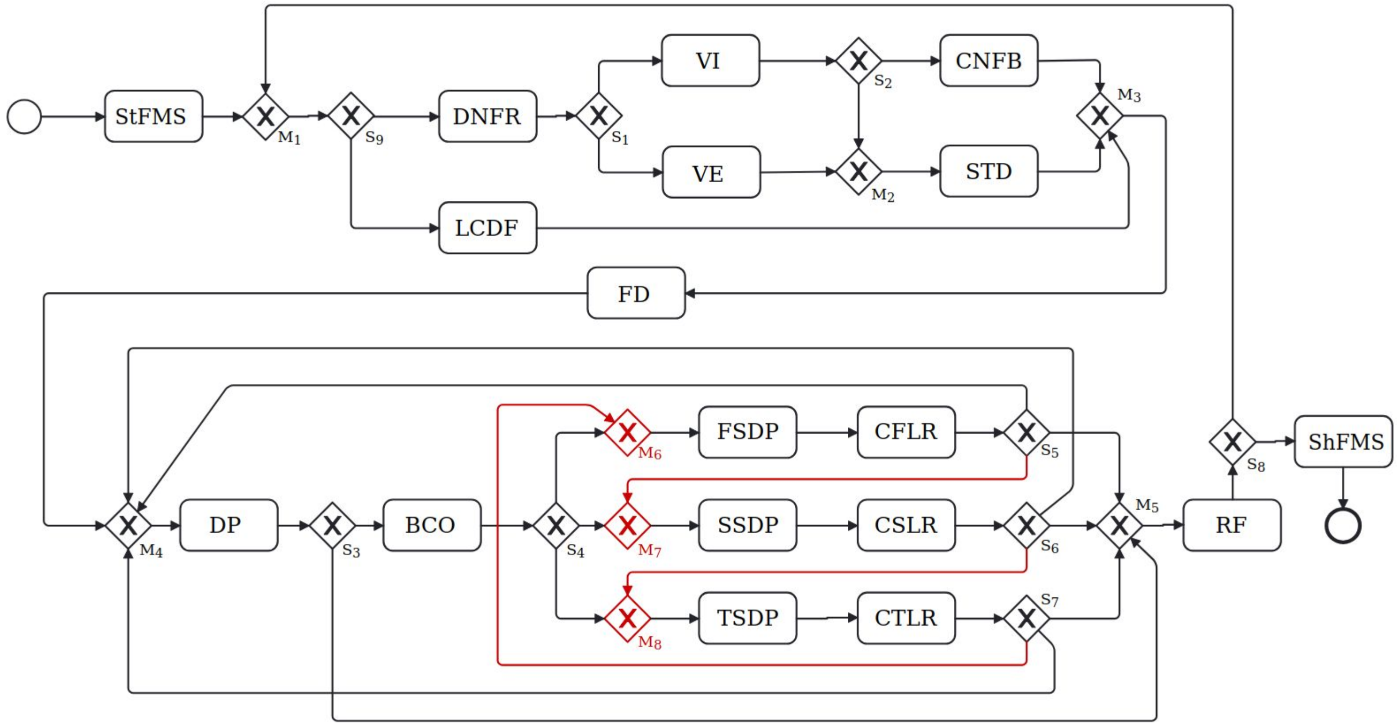
Detailed Approach – Step 6.2 – Explicit Loops

The connection consists in adding **an edge** between **each node** of a sub-graph having a **0-reachability**, and the **smallest subset of nodes** of another sub-graph ensuring a **total reachability**.



Detailed Approach – Step 6.2 – Explicit Loops

These **new edges** are then eventually **added to the BPMN** process to make the loop appear in it:



Detailed Approach – Step 6.3 – Parallelism

The third and last refinement step consists in **introducing parallelism** in the process.

Detailed Approach – Step 6.3 – Parallelism

The third and last refinement step consists in **introducing parallelism** in the process.

We consider **parallelism as the default relationship** between tasks, unless otherwise specified.

Detailed Approach – Step 6.3 – Parallelism

The third and last refinement step consists in **introducing parallelism** in the process.

We consider **parallelism as the default relationship** between tasks, unless otherwise specified.

Its introduction thus consists in **switching** some **exclusive** gateways to **parallel** gateways, while **preserving** the desired **mutual exclusions**.

Detailed Approach – Step 6.3 – Parallelism

The third and last refinement step consists in **introducing parallelism** in the process.

We consider **parallelism as the default relationship** between tasks, unless otherwise specified.

Its introduction thus consists in **switching** some **exclusive** gateways to **parallel** gateways, while **preserving** the desired **mutual exclusions**.

However, this must be done **carefully** to avoid **two major issues** induced by parallelism: (execution) **deadlocks** and **livelocks**.

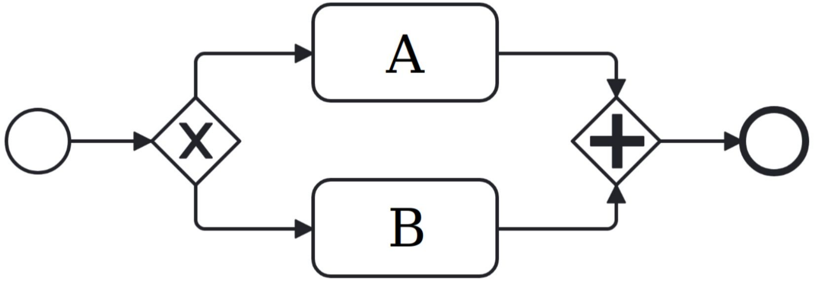
Detailed Approach – Step 6.3 – Parallelism

The third and last refinement step consists in **introducing parallelism** in the process.

We consider **parallelism as the default relationship** between tasks, unless otherwise specified.

Its introduction thus consists in **switching** some **exclusive** gateways to **parallel** gateways, while **preserving** the desired **mutual exclusions**.

However, this must be done **carefully** to avoid **two major issues** induced by parallelism: (execution) **deadlocks** and **livelocks**.



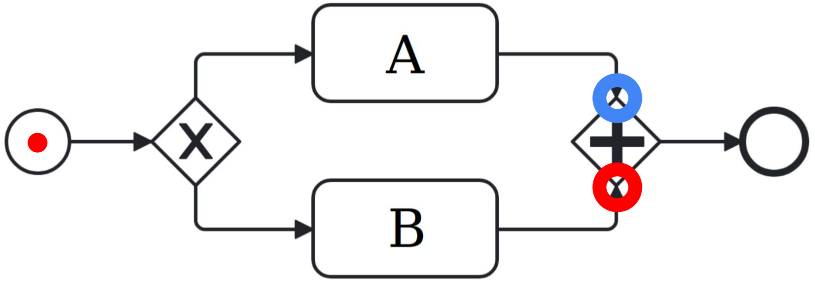
Detailed Approach – Step 6.3 – Parallelism

The third and last refinement step consists in **introducing parallelism** in the process.

We consider **parallelism as the default relationship** between tasks, unless otherwise specified.

Its introduction thus consists in **switching** some **exclusive** gateways to **parallel** gateways, while **preserving** the desired **mutual exclusions**.

However, this must be done **carefully** to avoid **two major issues** induced by parallelism: (execution) **deadlocks** and **livelocks**.



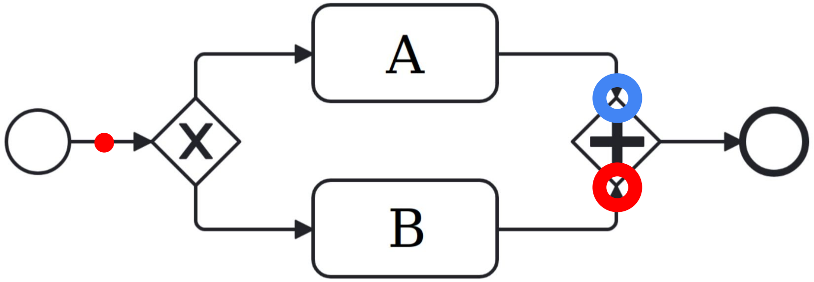
Detailed Approach – Step 6.3 – Parallelism

The third and last refinement step consists in **introducing parallelism** in the process.

We consider **parallelism as the default relationship** between tasks, unless otherwise specified.

Its introduction thus consists in **switching** some **exclusive** gateways to **parallel** gateways, while **preserving** the desired **mutual exclusions**.

However, this must be done **carefully** to avoid **two major issues** induced by parallelism: (execution) **deadlocks** and **livelocks**.



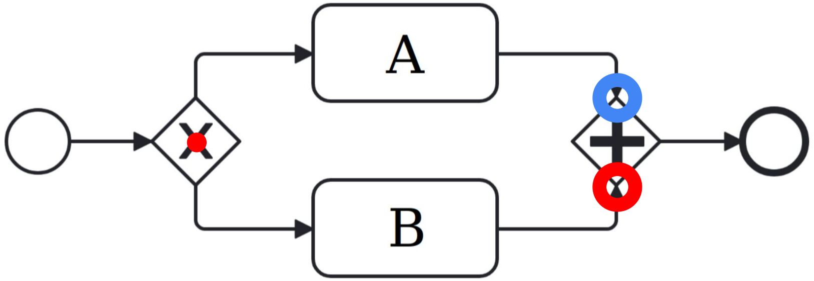
Detailed Approach – Step 6.3 – Parallelism

The third and last refinement step consists in **introducing parallelism** in the process.

We consider **parallelism as the default relationship** between tasks, unless otherwise specified.

Its introduction thus consists in **switching** some **exclusive** gateways to **parallel** gateways, while **preserving** the desired **mutual exclusions**.

However, this must be done **carefully** to avoid **two major issues** induced by parallelism: (execution) **deadlocks** and **livelocks**.



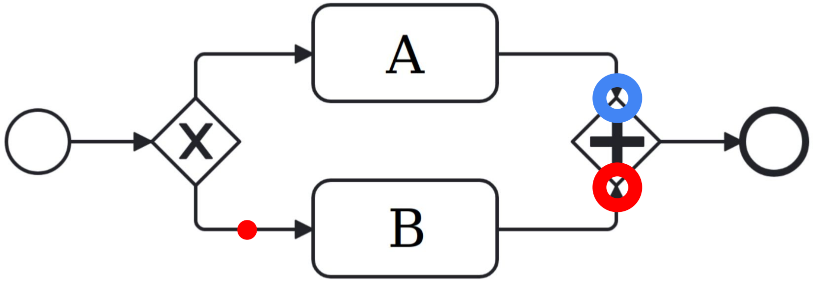
Detailed Approach – Step 6.3 – Parallelism

The third and last refinement step consists in **introducing parallelism** in the process.

We consider **parallelism as the default relationship** between tasks, unless otherwise specified.

Its introduction thus consists in **switching** some **exclusive** gateways to **parallel** gateways, while **preserving** the desired **mutual exclusions**.

However, this must be done **carefully** to avoid **two major issues** induced by parallelism: (execution) **deadlocks** and **livelocks**.



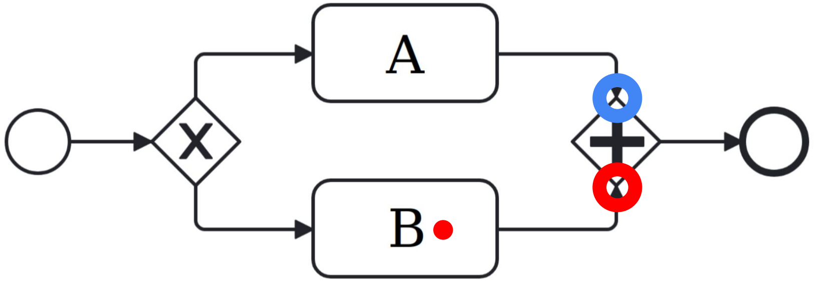
Detailed Approach – Step 6.3 – Parallelism

The third and last refinement step consists in **introducing parallelism** in the process.

We consider **parallelism as the default relationship** between tasks, unless otherwise specified.

Its introduction thus consists in **switching** some **exclusive** gateways to **parallel** gateways, while **preserving** the desired **mutual exclusions**.

However, this must be done **carefully** to avoid **two major issues** induced by parallelism: (execution) **deadlocks** and **livelocks**.



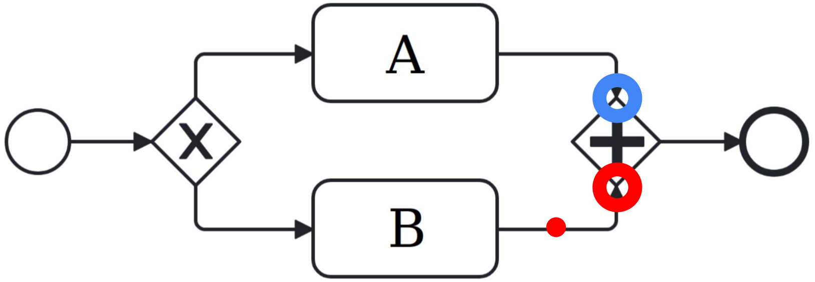
Detailed Approach – Step 6.3 – Parallelism

The third and last refinement step consists in **introducing parallelism** in the process.

We consider **parallelism as the default relationship** between tasks, unless otherwise specified.

Its introduction thus consists in **switching** some **exclusive** gateways to **parallel** gateways, while **preserving** the desired **mutual exclusions**.

However, this must be done **carefully** to avoid **two major issues** induced by parallelism: (execution) **deadlocks** and **livelocks**.



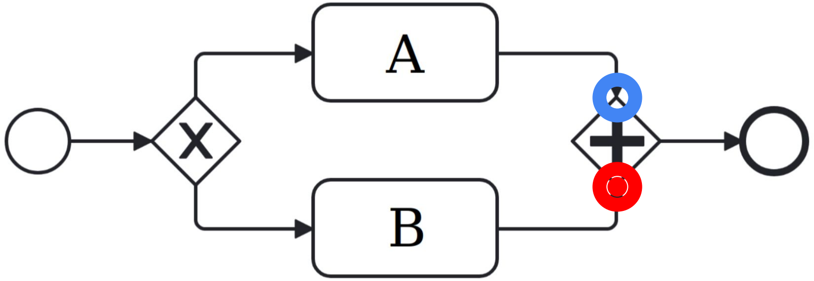
Detailed Approach – Step 6.3 – Parallelism

The third and last refinement step consists in **introducing parallelism** in the process.

We consider **parallelism as the default relationship** between tasks, unless otherwise specified.

Its introduction thus consists in **switching** some **exclusive** gateways to **parallel** gateways, while **preserving** the desired **mutual exclusions**.

However, this must be done **carefully** to avoid **two major issues** induced by parallelism: (execution) **deadlocks** and **livelocks**.



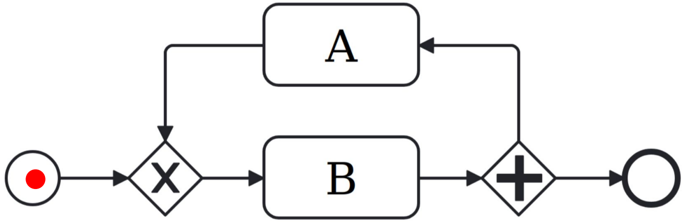
Detailed Approach – Step 6.3 – Parallelism

The third and last refinement step consists in **introducing parallelism** in the process.

We consider **parallelism as the default relationship** between tasks, unless otherwise specified.

Its introduction thus consists in **switching** some **exclusive** gateways to **parallel** gateways, while **preserving** the desired **mutual exclusions**.

However, this must be done **carefully** to avoid **two major issues** induced by parallelism: (execution) **deadlocks** and **livelocks**.



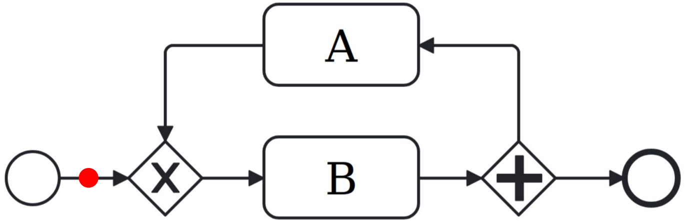
Detailed Approach – Step 6.3 – Parallelism

The third and last refinement step consists in **introducing parallelism** in the process.

We consider **parallelism as the default relationship** between tasks, unless otherwise specified.

Its introduction thus consists in **switching** some **exclusive** gateways to **parallel** gateways, while **preserving** the desired **mutual exclusions**.

However, this must be done **carefully** to avoid **two major issues** induced by parallelism: (execution) **deadlocks** and **livelocks**.



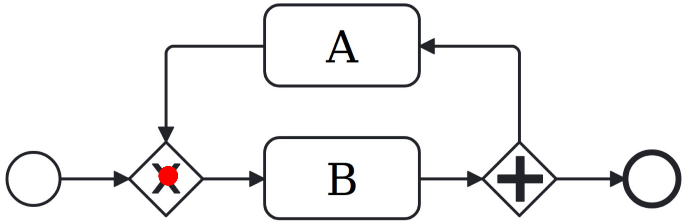
Detailed Approach – Step 6.3 – Parallelism

The third and last refinement step consists in **introducing parallelism** in the process.

We consider **parallelism as the default relationship** between tasks, unless otherwise specified.

Its introduction thus consists in **switching** some **exclusive** gateways to **parallel** gateways, while **preserving** the desired **mutual exclusions**.

However, this must be done **carefully** to avoid **two major issues** induced by parallelism: (execution) **deadlocks** and **livelocks**.



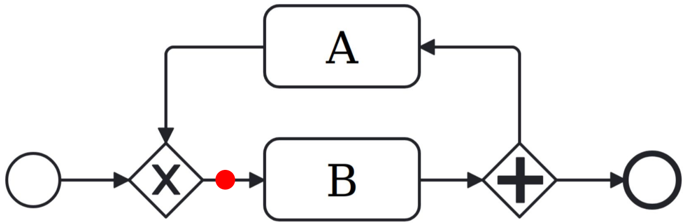
Detailed Approach – Step 6.3 – Parallelism

The third and last refinement step consists in **introducing parallelism** in the process.

We consider **parallelism as the default relationship** between tasks, unless otherwise specified.

Its introduction thus consists in **switching** some **exclusive** gateways to **parallel** gateways, while **preserving** the desired **mutual exclusions**.

However, this must be done **carefully** to avoid **two major issues** induced by parallelism: (execution) **deadlocks** and **livelocks**.



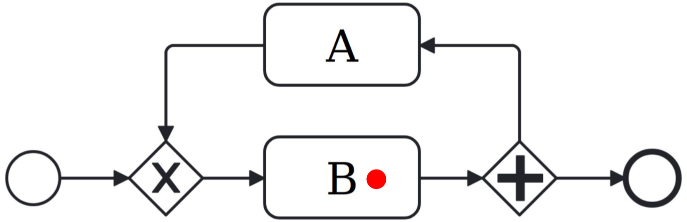
Detailed Approach – Step 6.3 – Parallelism

The third and last refinement step consists in **introducing parallelism** in the process.

We consider **parallelism as the default relationship** between tasks, unless otherwise specified.

Its introduction thus consists in **switching** some **exclusive** gateways to **parallel** gateways, while **preserving** the desired **mutual exclusions**.

However, this must be done **carefully** to avoid **two major issues** induced by parallelism: (execution) **deadlocks** and **livelocks**.



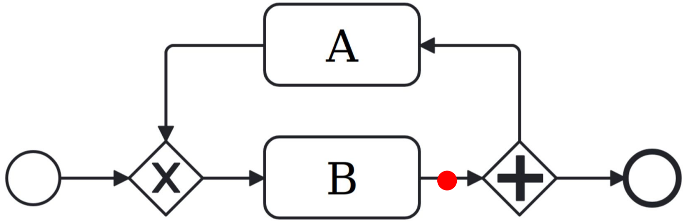
Detailed Approach – Step 6.3 – Parallelism

The third and last refinement step consists in **introducing parallelism** in the process.

We consider **parallelism as the default relationship** between tasks, unless otherwise specified.

Its introduction thus consists in **switching** some **exclusive** gateways to **parallel** gateways, while **preserving** the desired **mutual exclusions**.

However, this must be done **carefully** to avoid **two major issues** induced by parallelism: (execution) **deadlocks** and **livelocks**.



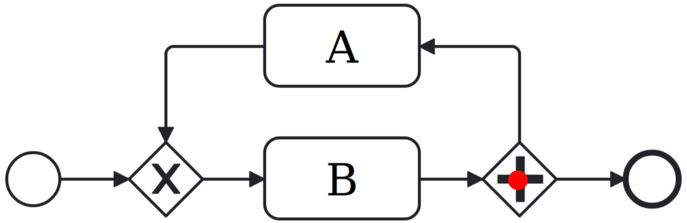
Detailed Approach – Step 6.3 – Parallelism

The third and last refinement step consists in **introducing parallelism** in the process.

We consider **parallelism as the default relationship** between tasks, unless otherwise specified.

Its introduction thus consists in **switching** some **exclusive** gateways to **parallel** gateways, while **preserving** the desired **mutual exclusions**.

However, this must be done **carefully** to avoid **two major issues** induced by parallelism: (execution) **deadlocks** and **livelocks**.



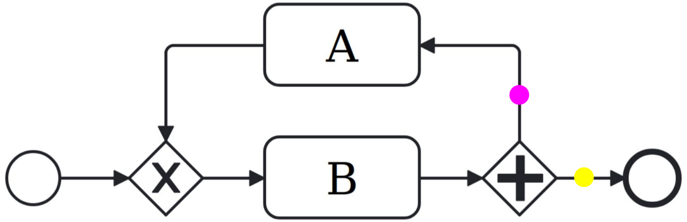
Detailed Approach – Step 6.3 – Parallelism

The third and last refinement step consists in **introducing parallelism** in the process.

We consider **parallelism as the default relationship** between tasks, unless otherwise specified.

Its introduction thus consists in **switching** some **exclusive** gateways to **parallel** gateways, while **preserving** the desired **mutual exclusions**.

However, this must be done **carefully** to avoid **two major issues** induced by parallelism: (execution) **deadlocks** and **livelocks**.



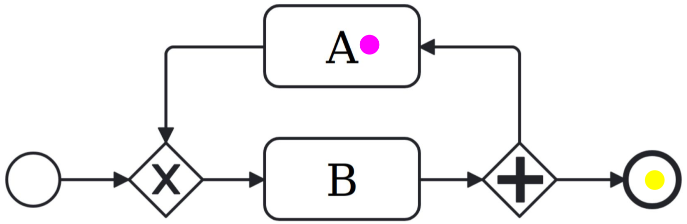
Detailed Approach – Step 6.3 – Parallelism

The third and last refinement step consists in **introducing parallelism** in the process.

We consider **parallelism as the default relationship** between tasks, unless otherwise specified.

Its introduction thus consists in **switching** some **exclusive** gateways to **parallel** gateways, while **preserving** the desired **mutual exclusions**.

However, this must be done **carefully** to avoid **two major issues** induced by parallelism: (execution) **deadlocks** and **livelocks**.



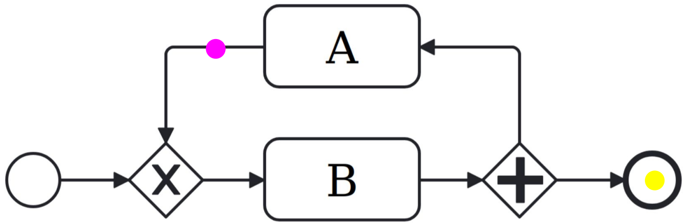
Detailed Approach – Step 6.3 – Parallelism

The third and last refinement step consists in **introducing parallelism** in the process.

We consider **parallelism as the default relationship** between tasks, unless otherwise specified.

Its introduction thus consists in **switching** some **exclusive** gateways to **parallel** gateways, while **preserving** the desired **mutual exclusions**.

However, this must be done **carefully** to avoid **two major issues** induced by parallelism: (execution) **deadlocks** and **livelocks**.



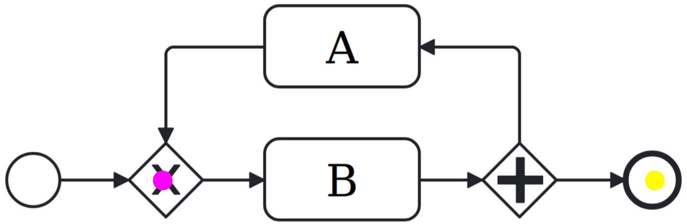
Detailed Approach – Step 6.3 – Parallelism

The third and last refinement step consists in **introducing parallelism** in the process.

We consider **parallelism as the default relationship** between tasks, unless otherwise specified.

Its introduction thus consists in **switching** some **exclusive** gateways to **parallel** gateways, while **preserving** the desired **mutual exclusions**.

However, this must be done **carefully** to avoid **two major issues** induced by parallelism: (execution) **deadlocks** and **livelocks**.



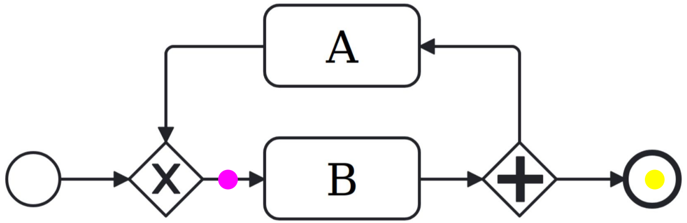
Detailed Approach – Step 6.3 – Parallelism

The third and last refinement step consists in **introducing parallelism** in the process.

We consider **parallelism as the default relationship** between tasks, unless otherwise specified.

Its introduction thus consists in **switching** some **exclusive** gateways to **parallel** gateways, while **preserving** the desired **mutual exclusions**.

However, this must be done **carefully** to avoid **two major issues** induced by parallelism: (execution) **deadlocks** and **livelocks**.



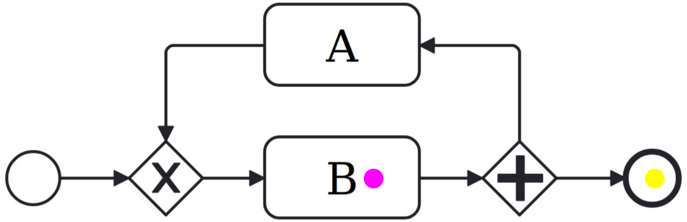
Detailed Approach – Step 6.3 – Parallelism

The third and last refinement step consists in **introducing parallelism** in the process.

We consider **parallelism as the default relationship** between tasks, unless otherwise specified.

Its introduction thus consists in **switching** some **exclusive** gateways to **parallel** gateways, while **preserving** the desired **mutual exclusions**.

However, this must be done **carefully** to avoid **two major issues** induced by parallelism: (execution) **deadlocks** and **livelocks**.



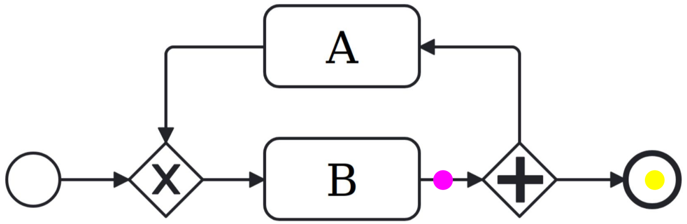
Detailed Approach – Step 6.3 – Parallelism

The third and last refinement step consists in **introducing parallelism** in the process.

We consider **parallelism as the default relationship** between tasks, unless otherwise specified.

Its introduction thus consists in **switching** some **exclusive** gateways to **parallel** gateways, while **preserving** the desired **mutual exclusions**.

However, this must be done **carefully** to avoid **two major issues** induced by parallelism: (execution) **deadlocks** and **livelocks**.



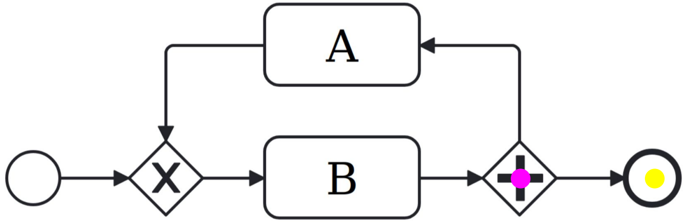
Detailed Approach – Step 6.3 – Parallelism

The third and last refinement step consists in **introducing parallelism** in the process.

We consider **parallelism as the default relationship** between tasks, unless otherwise specified.

Its introduction thus consists in **switching** some **exclusive** gateways to **parallel** gateways, while **preserving** the desired **mutual exclusions**.

However, this must be done **carefully** to avoid **two major issues** induced by parallelism: (execution) **deadlocks** and **livelocks**.



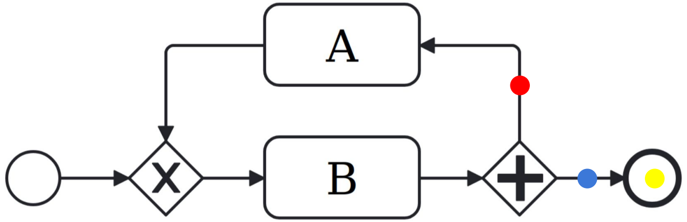
Detailed Approach – Step 6.3 – Parallelism

The third and last refinement step consists in **introducing parallelism** in the process.

We consider **parallelism as the default relationship** between tasks, unless otherwise specified.

Its introduction thus consists in **switching** some **exclusive** gateways to **parallel** gateways, while **preserving** the desired **mutual exclusions**.

However, this must be done **carefully** to avoid **two major issues** induced by parallelism: (execution) **deadlocks** and **livelocks**.



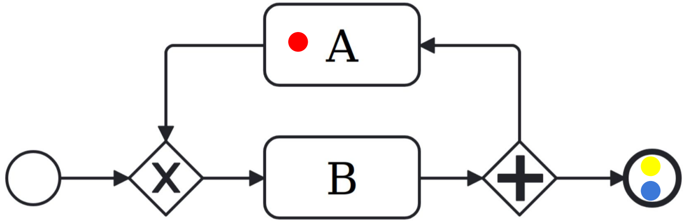
Detailed Approach – Step 6.3 – Parallelism

The third and last refinement step consists in **introducing parallelism** in the process.

We consider **parallelism as the default relationship** between tasks, unless otherwise specified.

Its introduction thus consists in **switching** some **exclusive** gateways to **parallel** gateways, while **preserving** the desired **mutual exclusions**.

However, this must be done **carefully** to avoid **two major issues** induced by parallelism: (execution) **deadlocks** and **livelocks**.

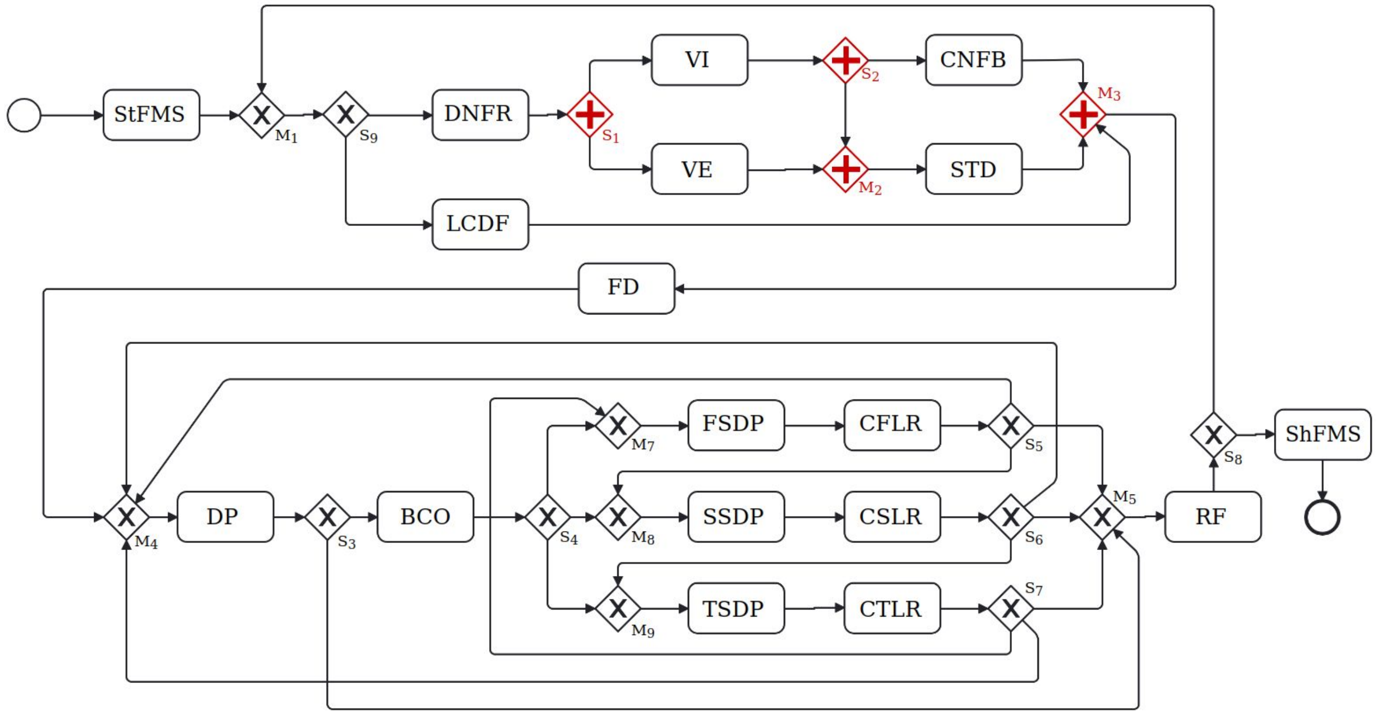


Detailed Approach – Step 6.3 – Parallelism

In our case, **switching exclusive** gateways **to parallel** ones while **preserving the constraints** leads to the following process:

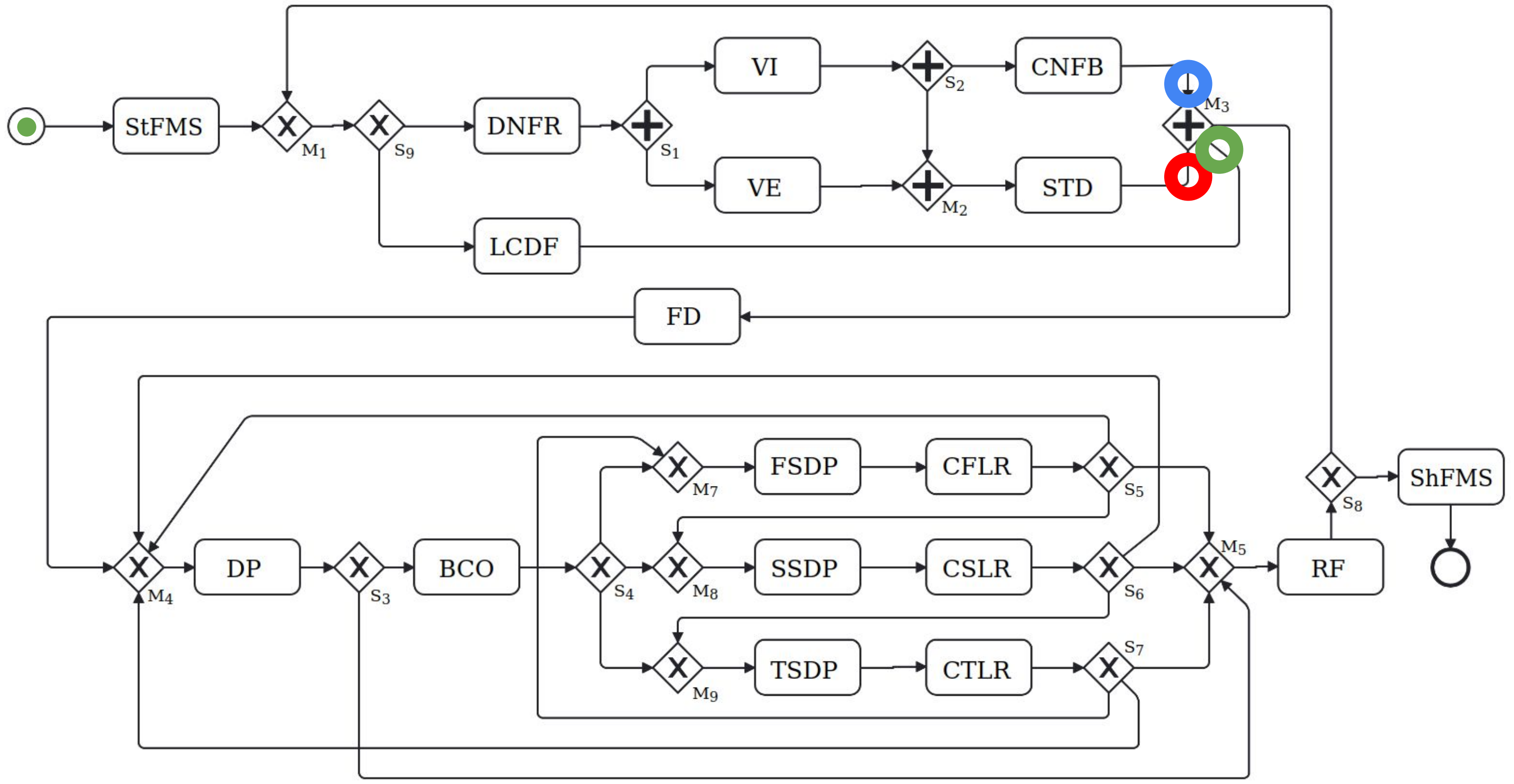
Detailed Approach – Step 6.3 – Parallelism

In our case, **switching exclusive** gateways **to parallel** ones while **preserving the constraints** leads to the following process:



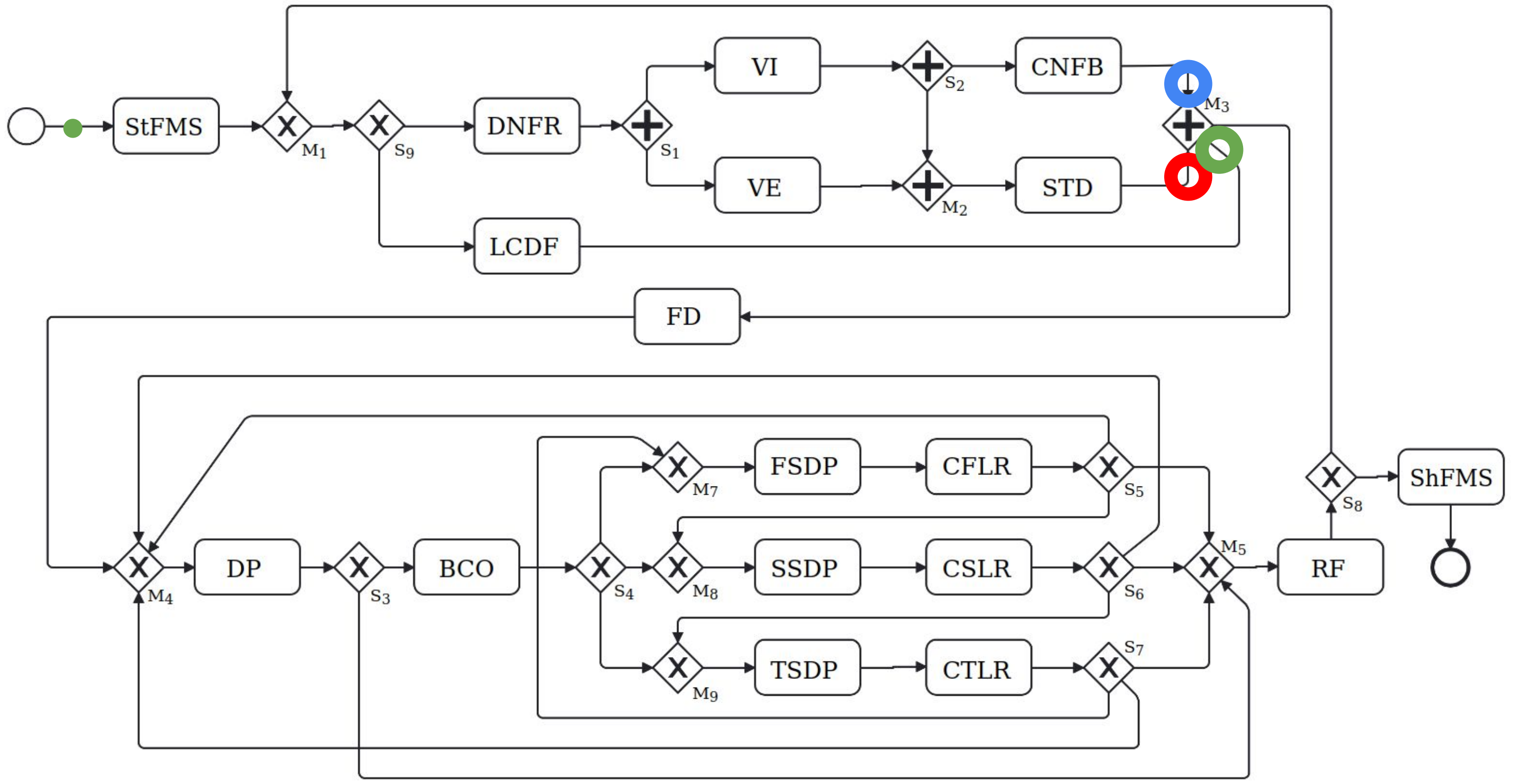
Detailed Approach – Step 6.3 – Parallelism

However, this process contains a **deadlock**:



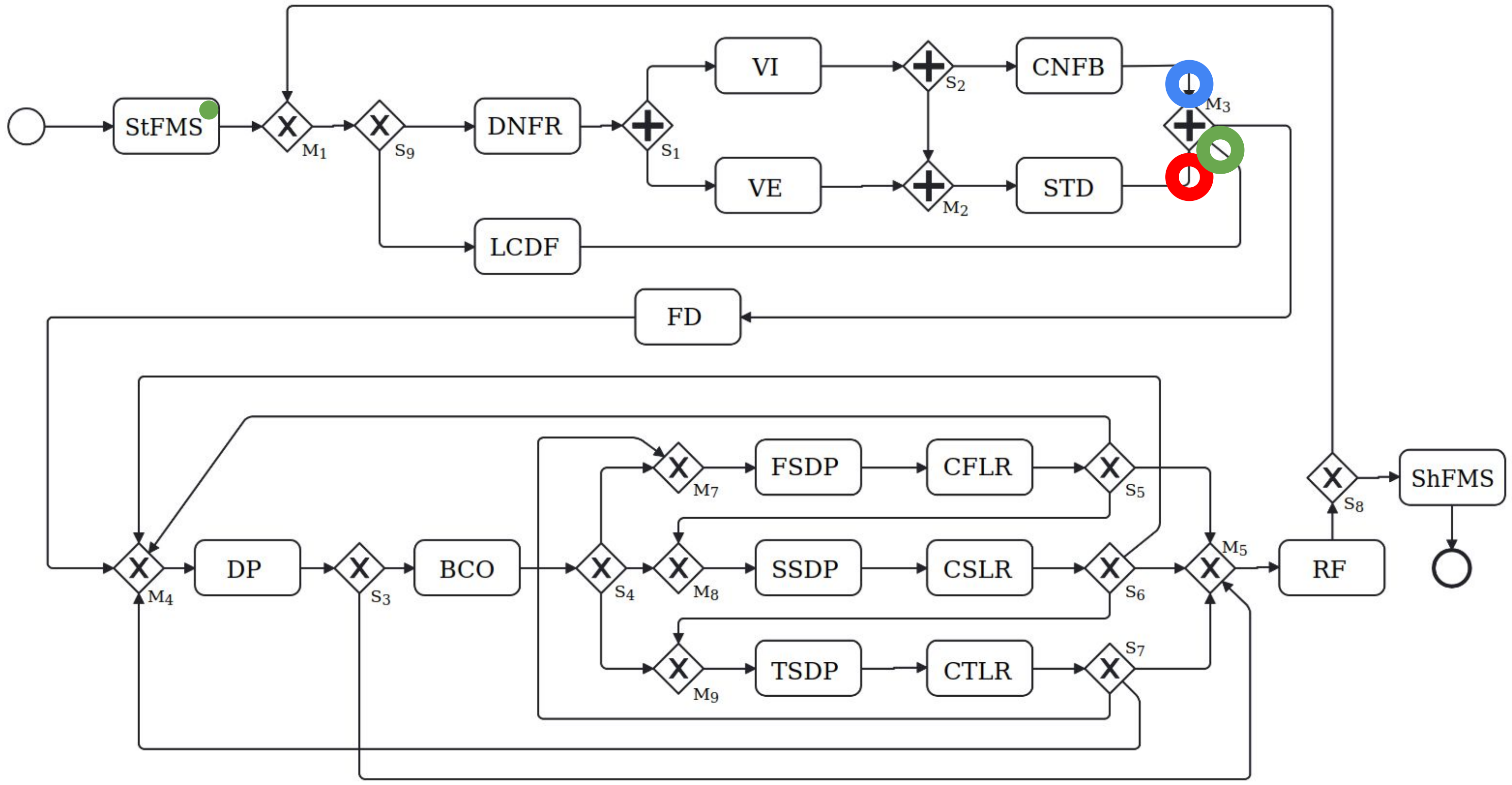
Detailed Approach – Step 6.3 – Parallelism

However, this process contains a **deadlock**:



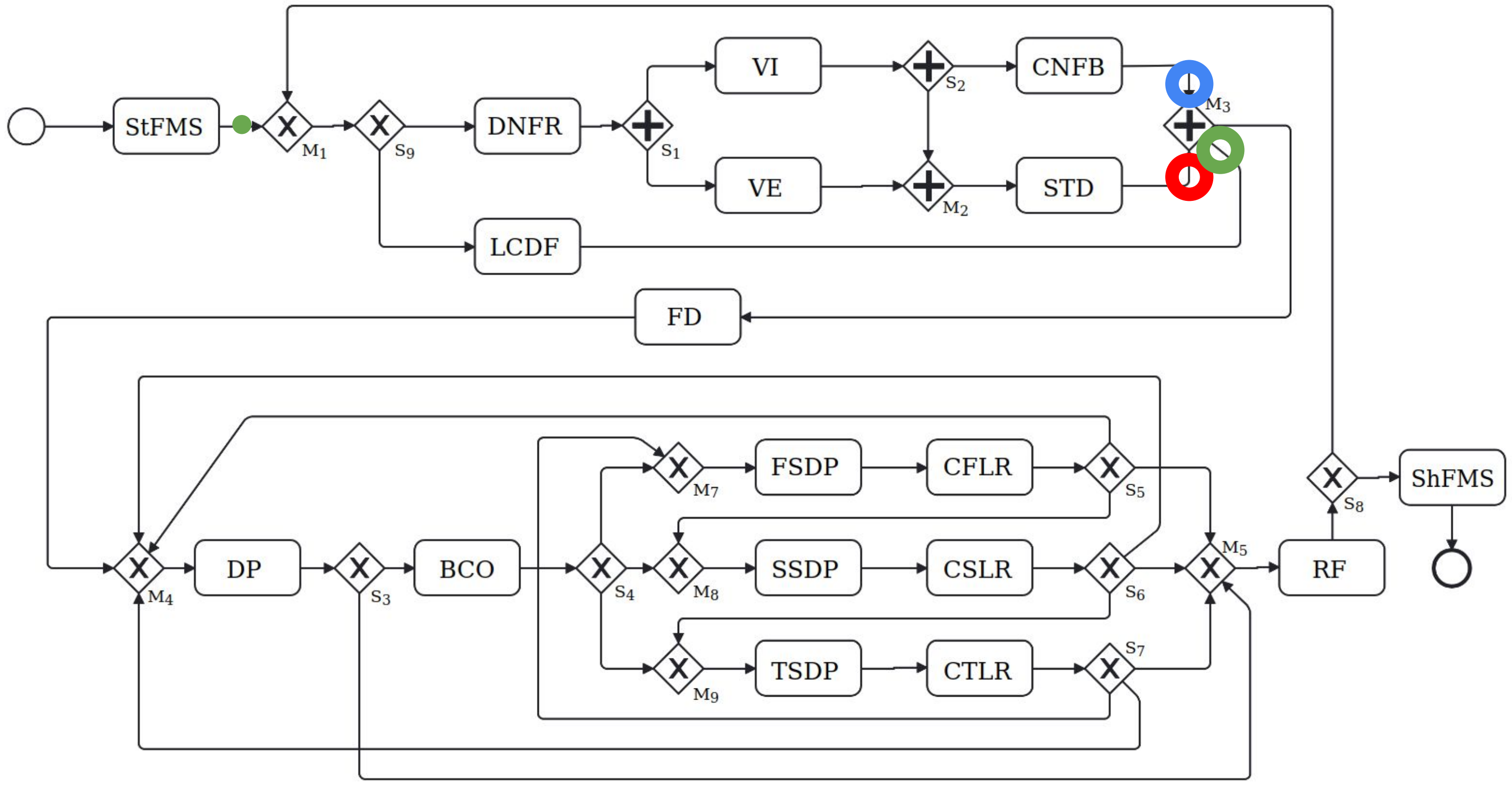
Detailed Approach – Step 6.3 – Parallelism

However, this process contains a **deadlock**:



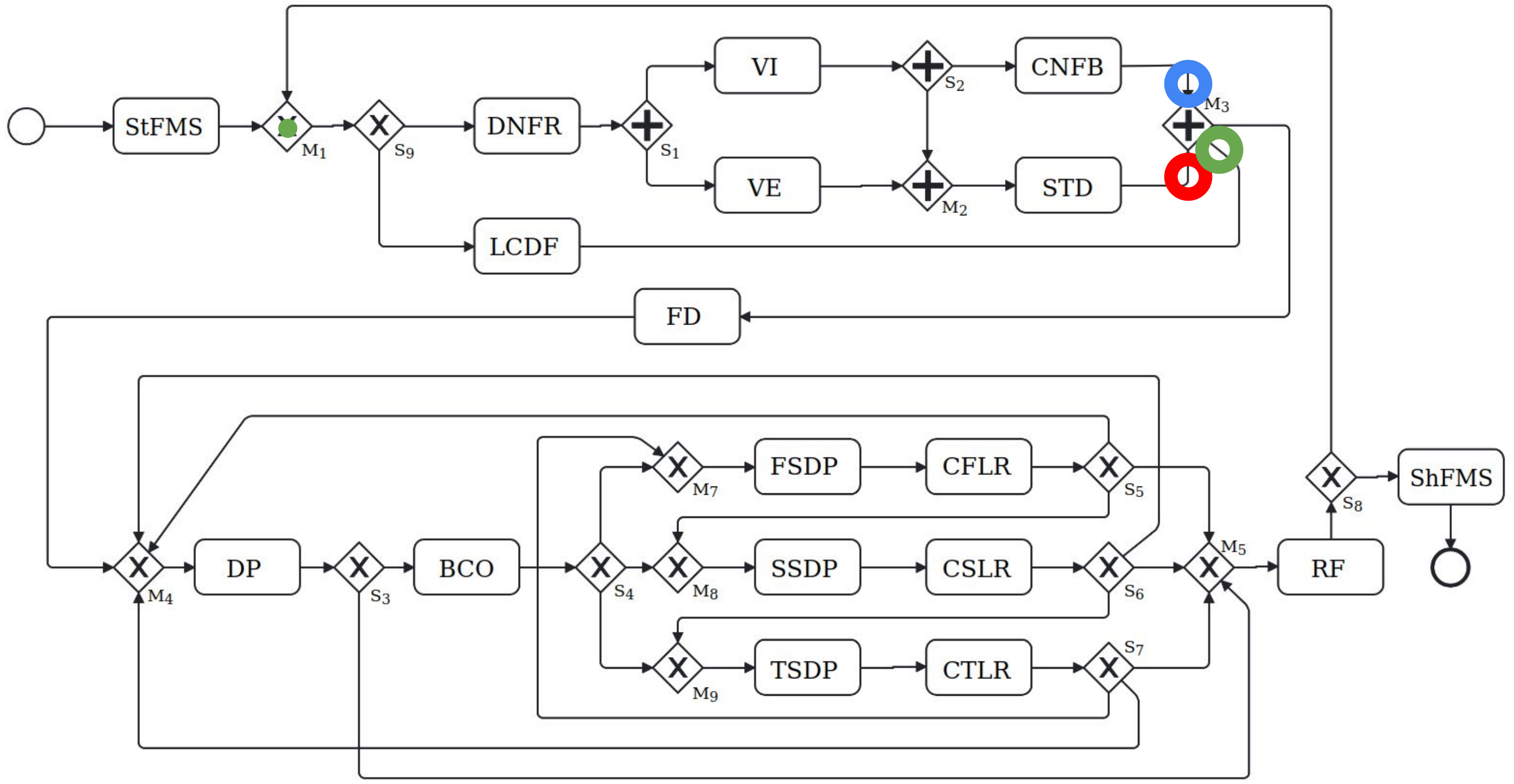
Detailed Approach – Step 6.3 – Parallelism

However, this process contains a **deadlock**:



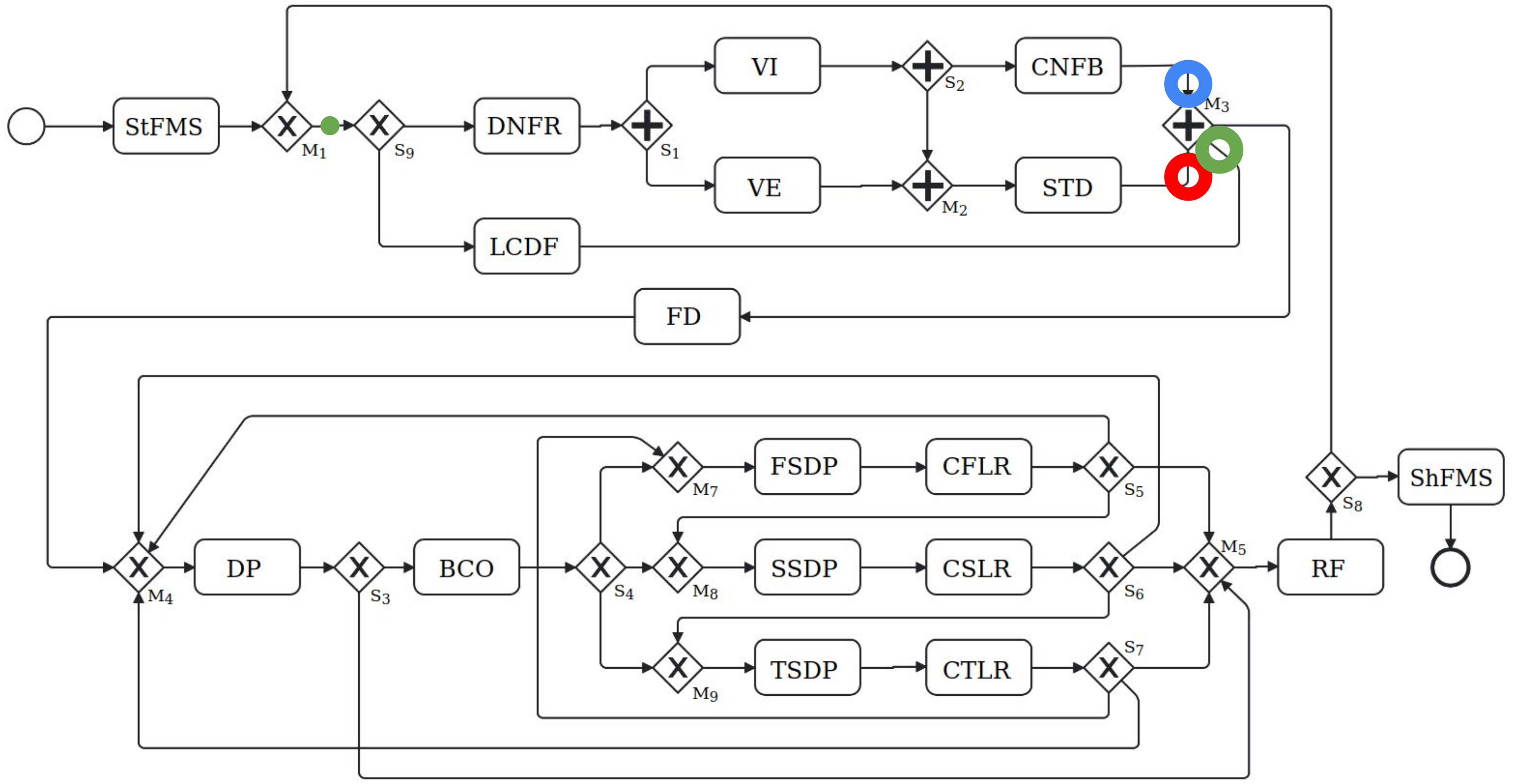
Detailed Approach – Step 6.3 – Parallelism

However, this process contains a **deadlock**:



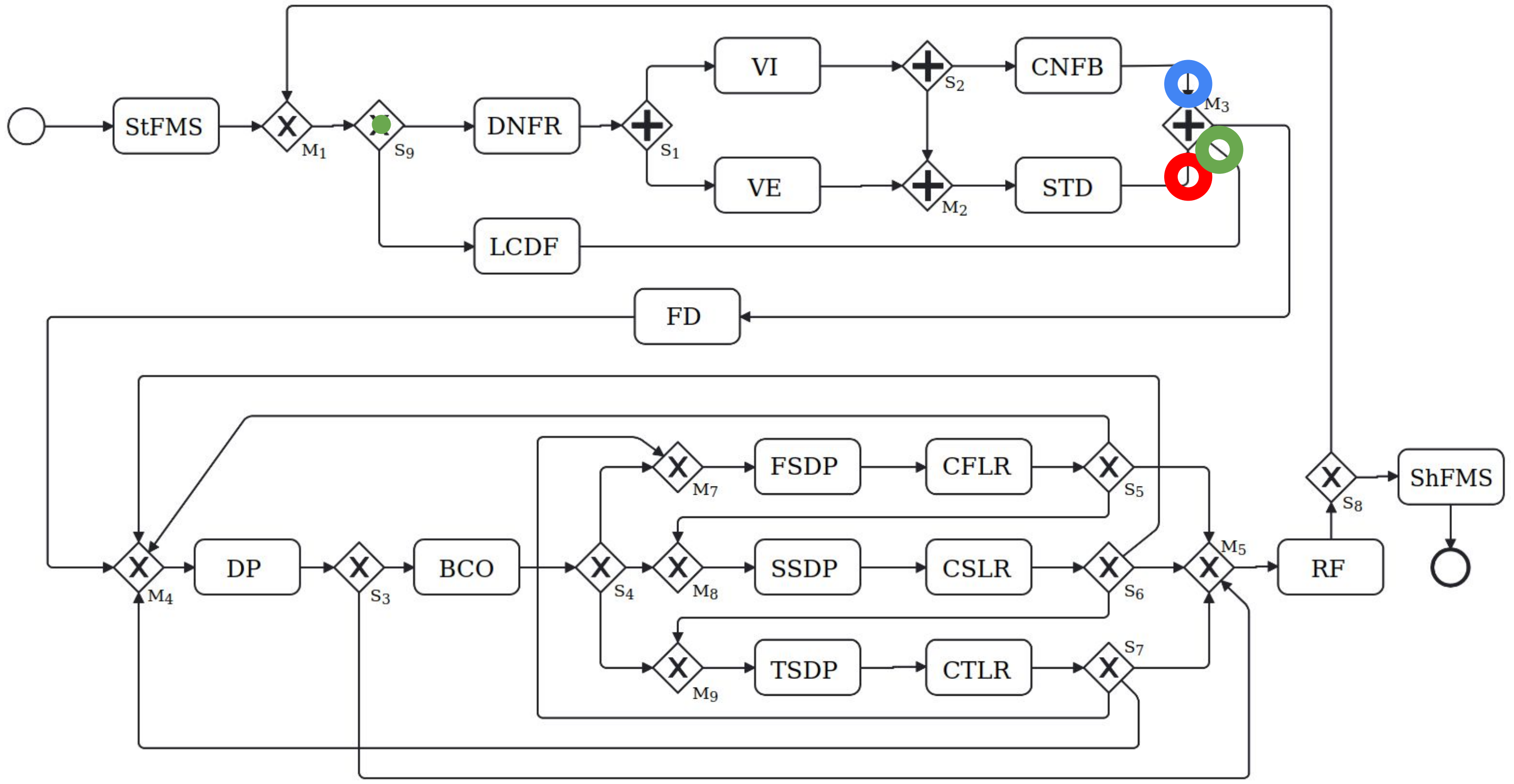
Detailed Approach – Step 6.3 – Parallelism

However, this process contains a **deadlock**:



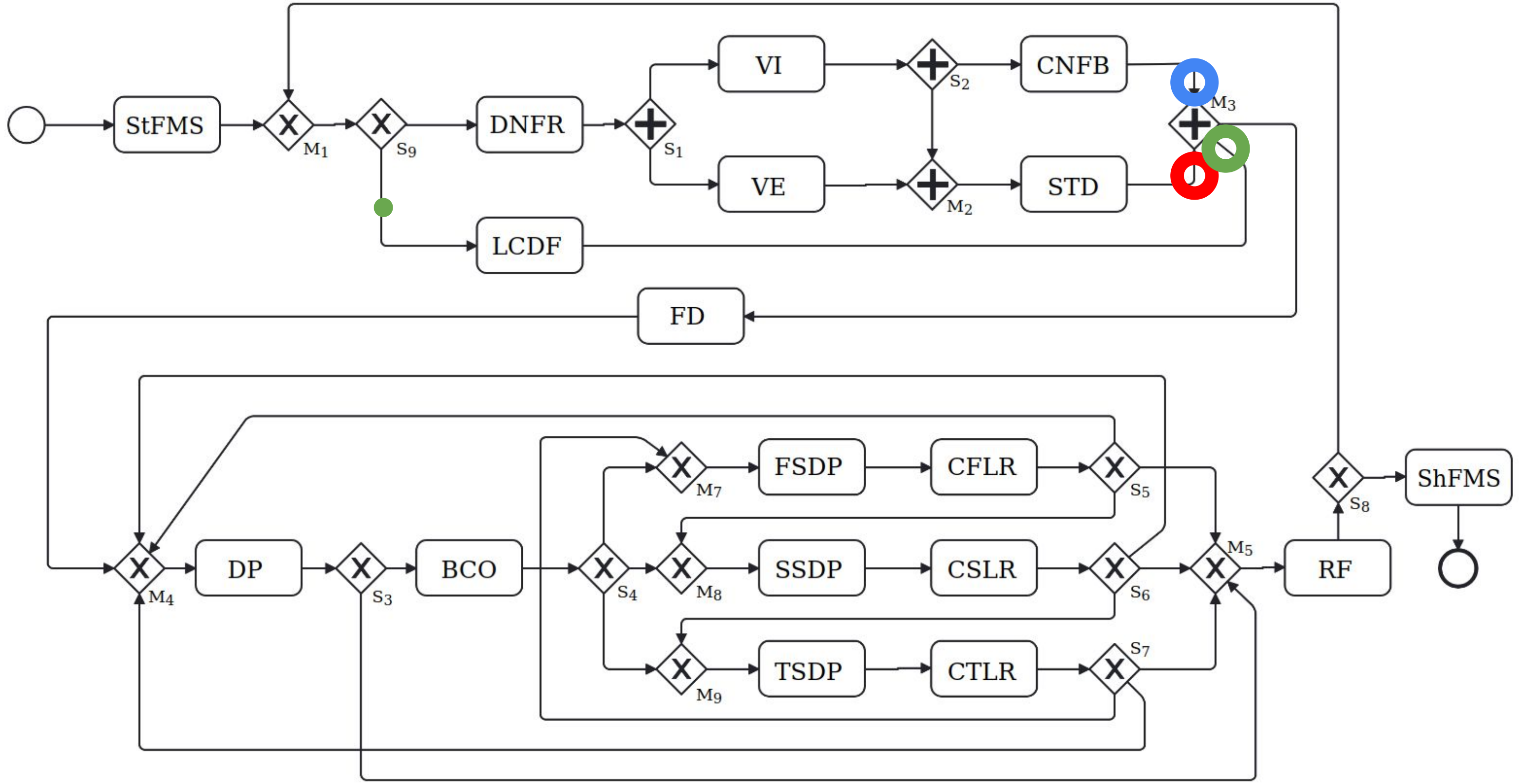
Detailed Approach – Step 6.3 – Parallelism

However, this process contains a **deadlock**:



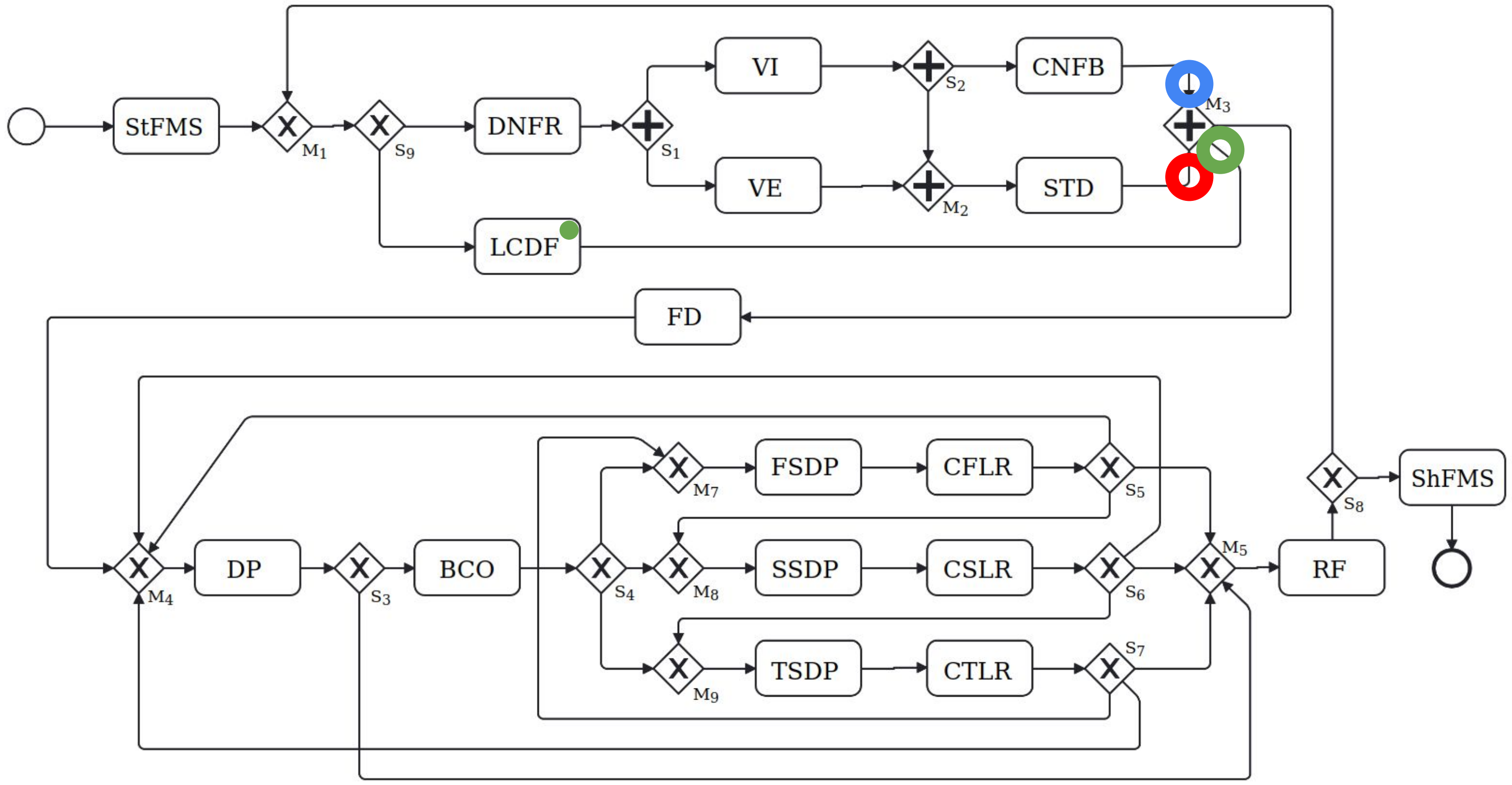
Detailed Approach – Step 6.3 – Parallelism

However, this process contains a **deadlock**:



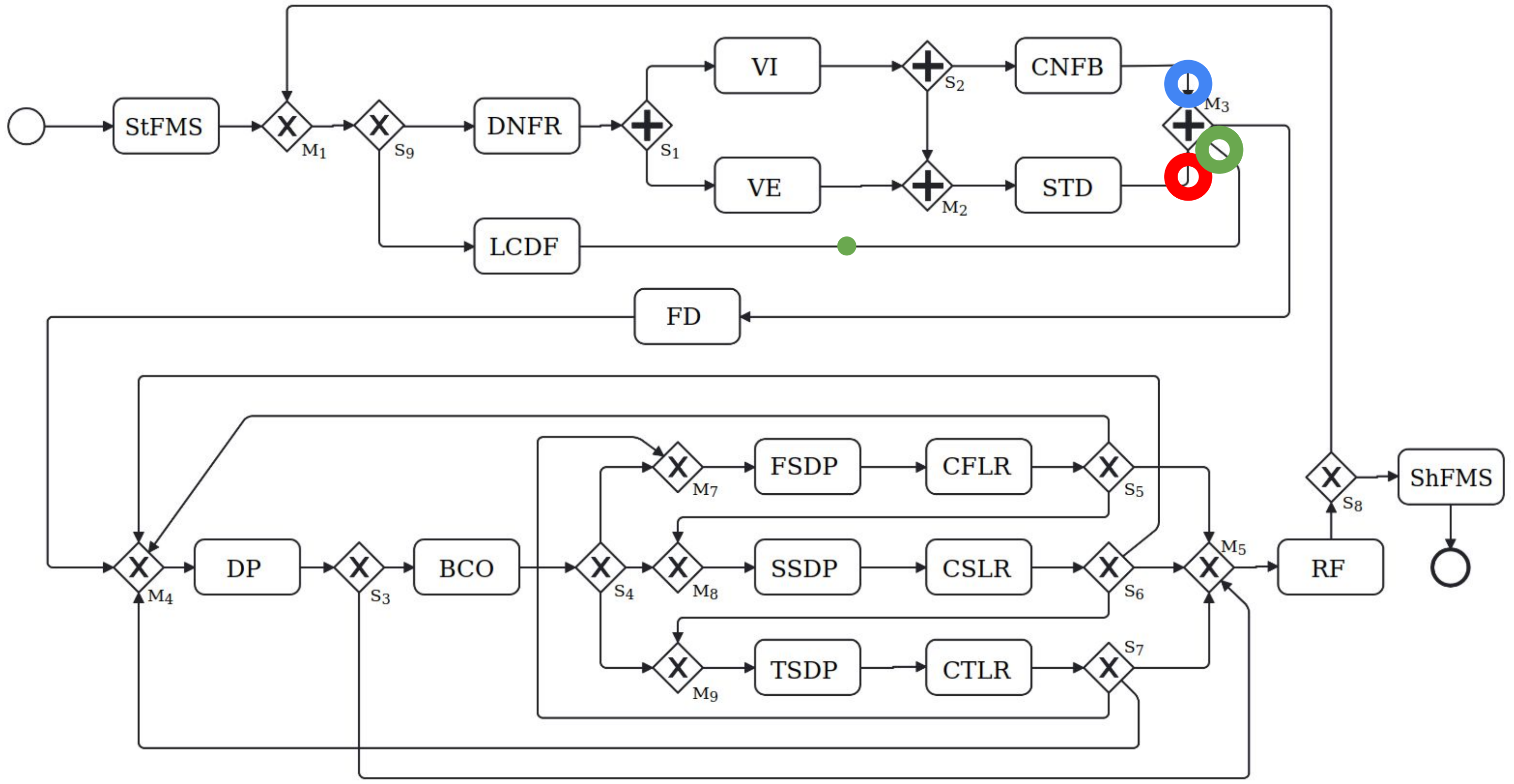
Detailed Approach – Step 6.3 – Parallelism

However, this process contains a **deadlock**:



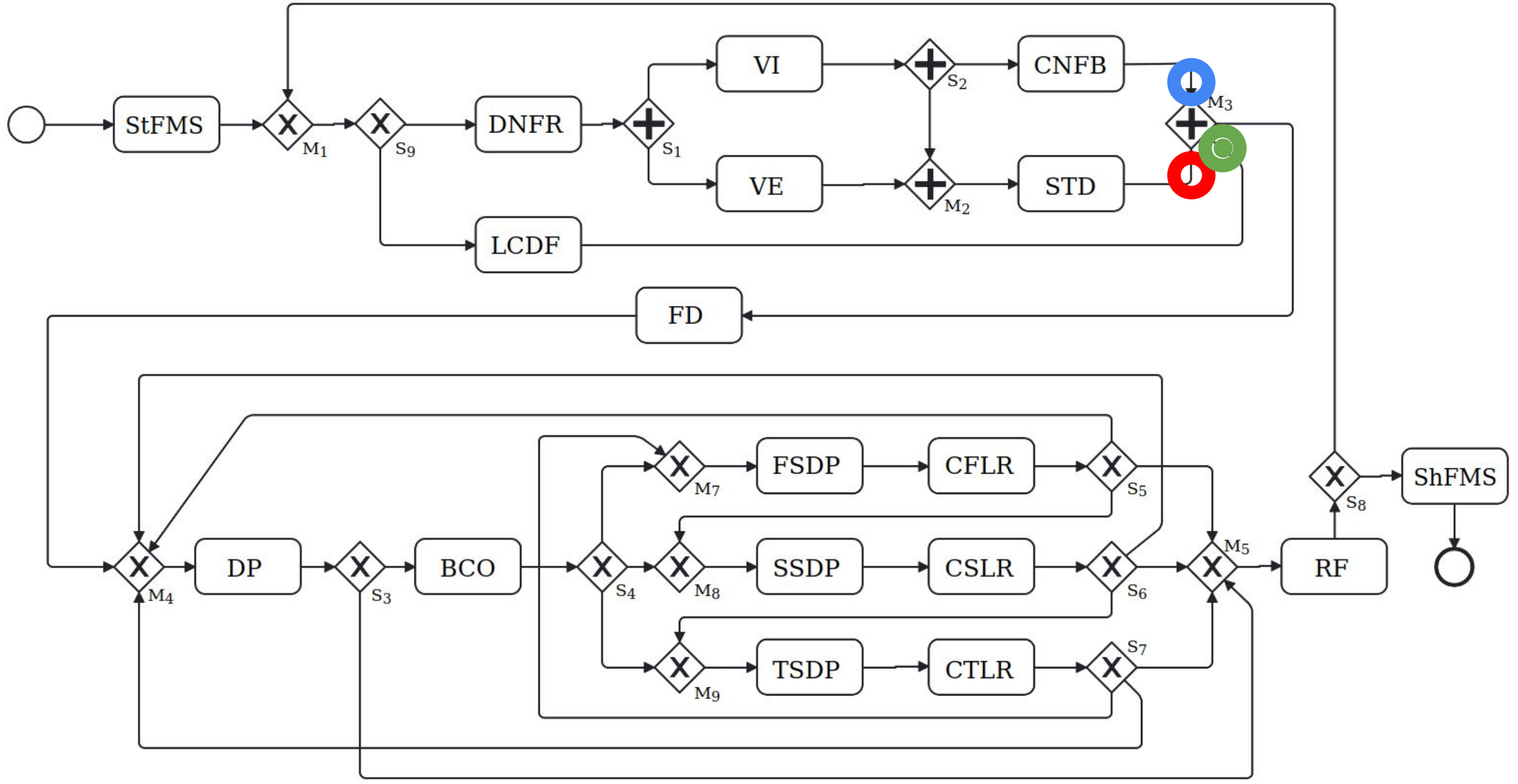
Detailed Approach – Step 6.3 – Parallelism

However, this process contains a **deadlock**:



Detailed Approach – Step 6.3 – Parallelism

However, this process contains a **deadlock**:

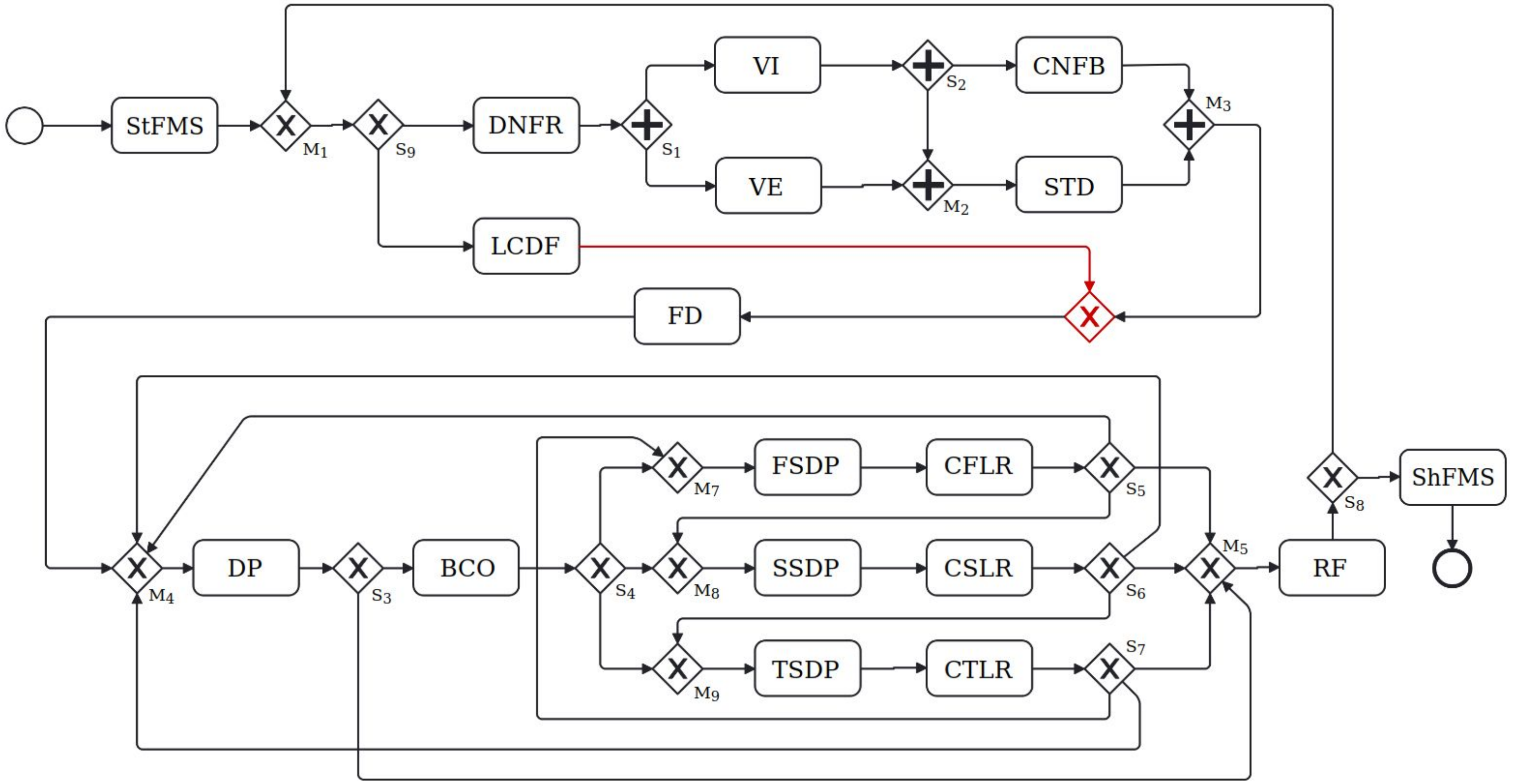


Detailed Approach – Step 6.3 – Parallelism

Our solution is to **remove partially** and **step-by-step** parallelism in the process.

Detailed Approach – Step 6.3 – Parallelism

Our solution is to **remove partially** and **step-by-step** parallelism in the process.

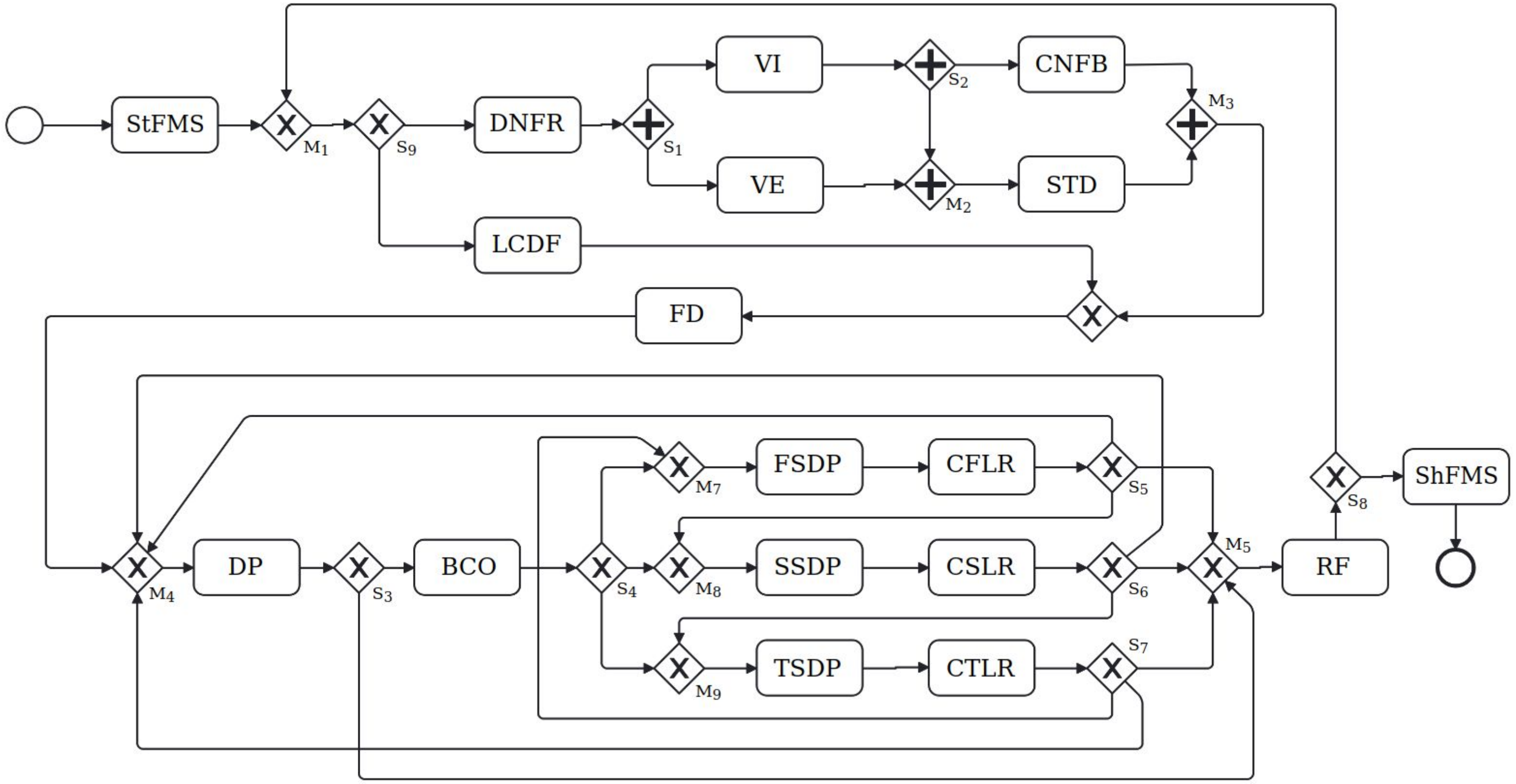


Detailed Approach – Step 6.3 – Parallelism

The **process** no longer contains deadlock nor livelocks, and is **complete**.

Detailed Approach – Step 6.3 – Parallelism

The **process** no longer contains deadlock nor livelocks, and is **complete**.



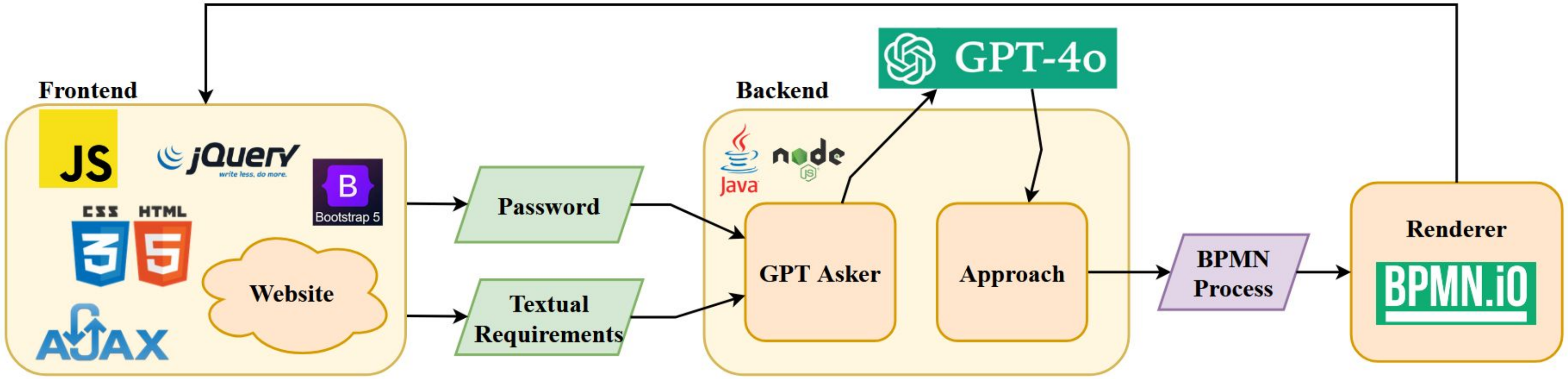
Tool Support

This approach has been entirely **developed** and **validated** with a tool written in **Java** and consisting of approximately **12k lines of code**.

Tool Support

This approach has been entirely **developed** and **validated** with a tool written in **Java** and consisting of approximately **12k lines of code**.

The Java code has been **embedded** in the backend of a **web server** freely accessible online (https://lig-givup.imag.fr/).



Experiments – Quality of the Process

Experiments were conducted on **200 examples**, **25%** coming from the **PET dataset** and the **literature**, and **75% handcrafted**.

Tool/Model	✓	?	✗	Avg. Ex. Time
Our tool	**83%**	9.8%	**7.2%**	**7.21s**
NaLa2BPMN	32.8%	8.9%	58.3%	68.7s
ProMoAI	50%	**8.7%**	41.2%	24.7s
Gemini	73.4%	13.8%	12.8%	7.67s
GPT-4-turbo	69.8%	19.3%	10.9%	11.8s

Experiments – Quality of the Process

Correct processes

Tool/Model	✓	?	✗	Avg. Ex. Time
Our tool	**83%**	9.8%	**7.2%**	**7.21s**
NaLa2BPMN	32.8%	8.9%	58.3%	68.7s
ProMoAI	50%	**8.7%**	41.2%	24.7s
Gemini	73.4%	13.8%	12.8%	7.67s
GPT-4-turbo	69.8%	19.3%	10.9%	11.8s

Experiments – Quality of the Process

Correct processes

Ambiguous processes

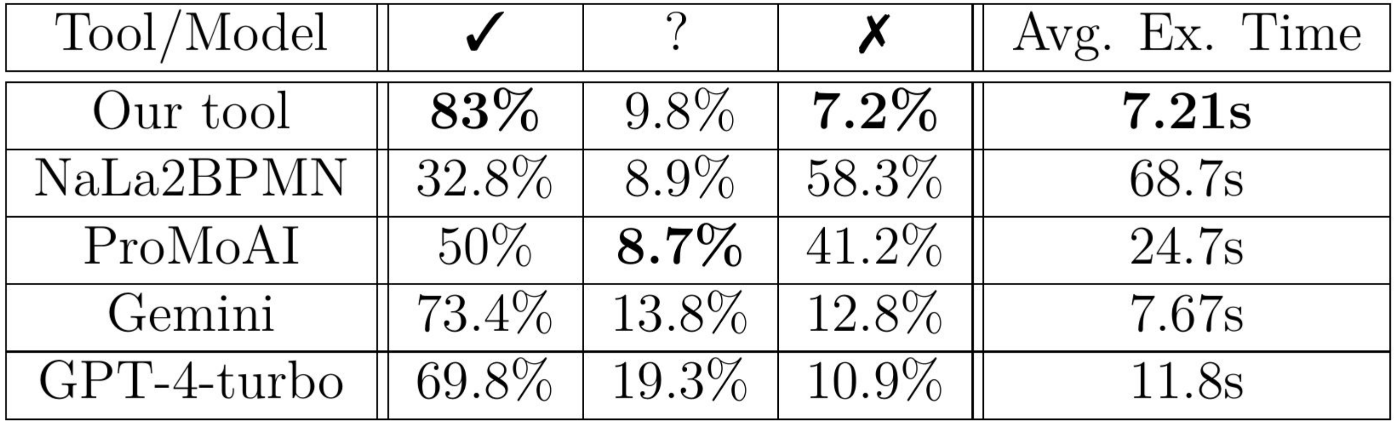
Tool/Model	✓	?	✗	Avg. Ex. Time
Our tool	**83%**	9.8%	**7.2%**	**7.21s**
NaLa2BPMN	32.8%	8.9%	58.3%	68.7s
ProMoAI	50%	**8.7%**	41.2%	24.7s
Gemini	73.4%	13.8%	12.8%	7.67s
GPT-4-turbo	69.8%	19.3%	10.9%	11.8s

Experiments – Quality of the Process

Correct processes

Ambiguous processes

Incorrect processes



Tool/Model	✓	?	✗	Avg. Ex. Time
Our tool	**83%**	9.8%	**7.2%**	**7.21s**
NaLa2BPMN	32.8%	8.9%	58.3%	68.7s
ProMoAI	50%	**8.7%**	41.2%	24.7s
Gemini	73.4%	13.8%	12.8%	7.67s
GPT-4-turbo	69.8%	19.3%	10.9%	11.8s

Experiments – Quality of the Process

Correct processes

Ambiguous processes

Incorrect processes

Tool/Model	✓	?	✗	Avg. Ex. Time
Our tool	**83%**	9.8%	**7.2%**	**7.21s**
NaLa2BPMN	32.8%	8.9%	58.3%	68.7s
ProMoAI	50%	**8.7%**	41.2%	24.7s
Gemini	73.4%	13.8%	12.8%	7.67s
GPT-4-turbo	69.8%	19.3%	10.9%	11.8s

An **ambiguous process** is a process that is **not incorrect** with regards to the description, but which **does not correspond to the expectations** of the experts.

Conclusion

In this work, we proposed an approach aiming at automatically designing **syntactically and semantically correct BPMN** processes from a **textual description** of the requirements.

Conclusion

In this work, we proposed an approach aiming at automatically designing **syntactically and semantically correct BPMN** processes from a **textual description** of the requirements.

The proposed approach performs in **6 main steps** that are in charge of **transforming** the **description** into expressions, ASTs, dependency graph and **BPMN**, and finally enrich this BPMN process.

Conclusion

In this work, we proposed an approach aiming at automatically designing **syntactically and semantically correct BPMN** processes from a **textual description** of the requirements.

The proposed approach performs in **6 main steps** that are in charge of **transforming** the **description** into expressions, ASTs, dependency graph and **BPMN**, and finally enrich this BPMN process.

It has been **fully implemented** and **tested** as a tool consisting of approximately **12k lines** of Java code, which was embedded in the backend of a **web server** for **distribution purposes**.

Plan

Existing Solutions

- **Operational research** is a field in which such problems are well-known;

Existing Solutions

- ➢ **Operational research** is a field in which such problems are well-known;
- ➢ **Resource balancing** allows to adapt the pool of resources according to the needs of the process;

Existing Solutions

- **Operational research** is a field in which such problems are well-known;
- **Resource balancing** allows to adapt the pool of resources according to the needs of the process;
- **Scheduling** permits to modify the priority of the tasks in order to fluidify their execution;

Existing Solutions

- ➢ **Operational research** is a field in which such problems are well-known;
- ➢ **Resource balancing** allows to adapt the pool of resources according to the needs of the process;
- ➢ **Scheduling** permits to modify the priority of the tasks in order to fluidify their execution;
- ➢ **Process refactoring** modifies the structure of the process with the goal of optimising it.

Existing Solutions

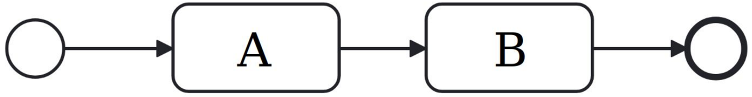
- ➢ **Operational research** is a field in which such problems are well-known;
- ➢ **Resource balancing** allows to adapt the pool of resources according to the needs of the process;
- ➢ **Scheduling** permits to modify the priority of the tasks in order to fluidify their execution;
- ➢ **Process refactoring** modifies the structure of the process with the goal of optimising it.

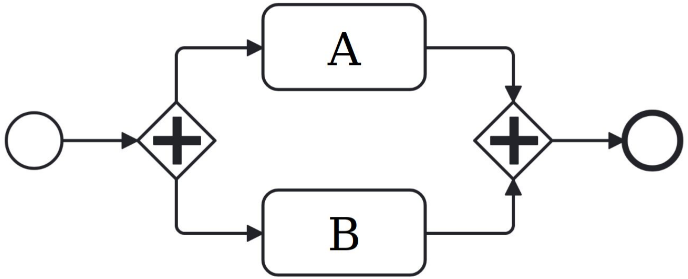
Process Refactoring

Refactoring a process consists in **modifying the order** in which the tasks are organised, in order to **optimise** the process with regards to some criteria (**execution time**, **resources usage**, **costs**, etc.).

Process Refactoring

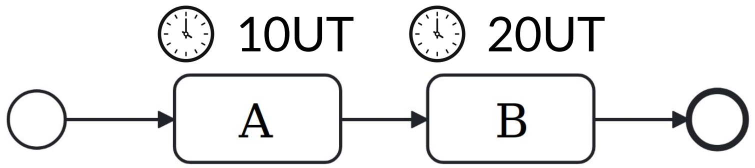
Refactoring a process consists in **modifying the order** in which the tasks are organised, in order to **optimise** the process with regards to some criteria (**execution time**, **resources usage**, **costs**, etc.).

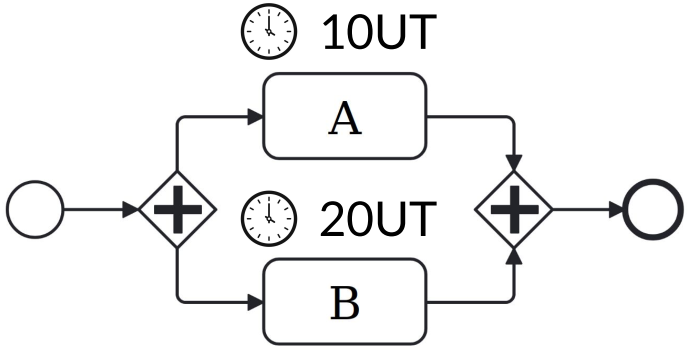




Process Refactoring

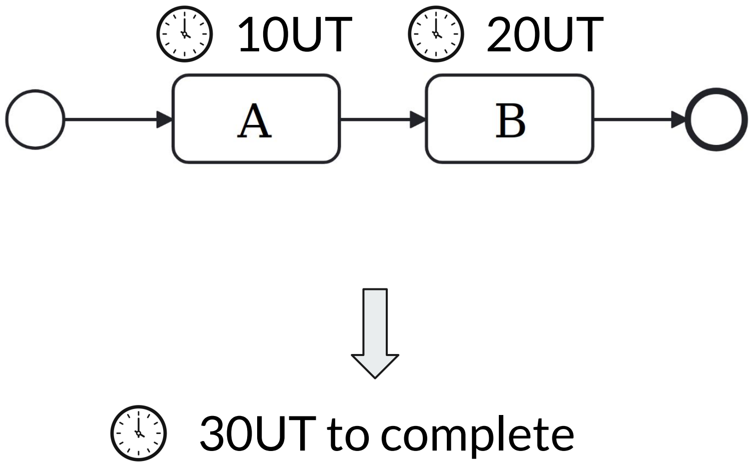
Refactoring a process consists in **modifying the order** in which the tasks are organised, in order to **optimise** the process with regards to some criteria (**execution time**, **resources usage**, **costs**, etc.).

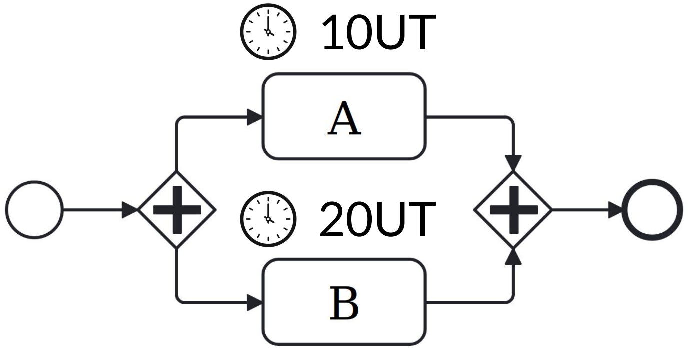




Process Refactoring

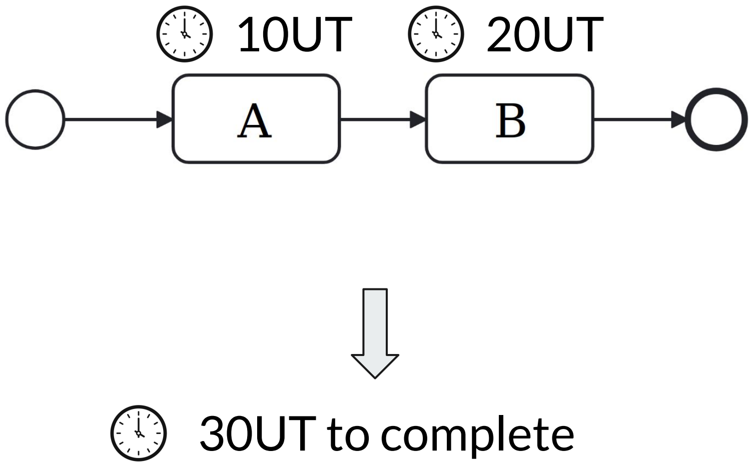
Refactoring a process consists in **modifying the order** in which the tasks are organised, in order to **optimise** the process with regards to some criteria (**execution time**, **resources usage**, **costs**, etc.).

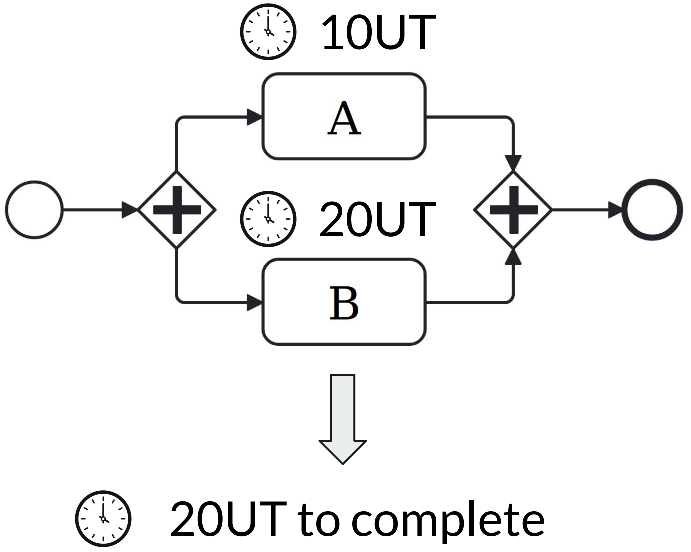




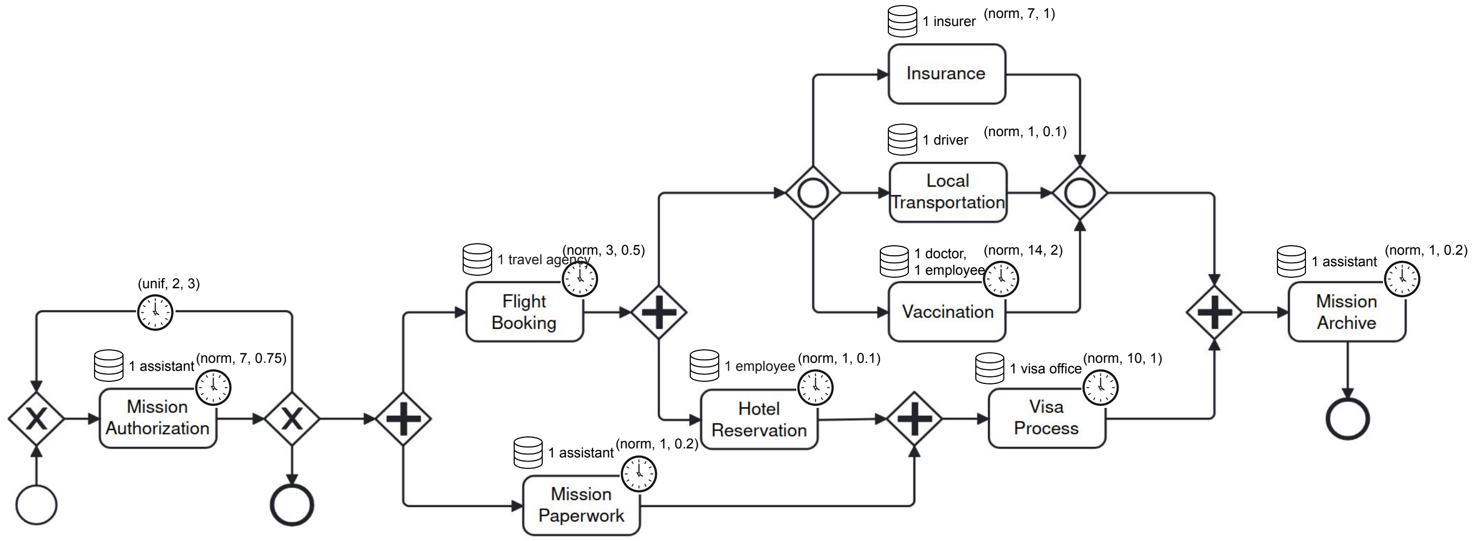
Process Refactoring

Refactoring a process consists in **modifying the order** in which the tasks are organised, in order to **optimise** the process with regards to some criteria (**execution time**, **resources usage**, **costs**, etc.).



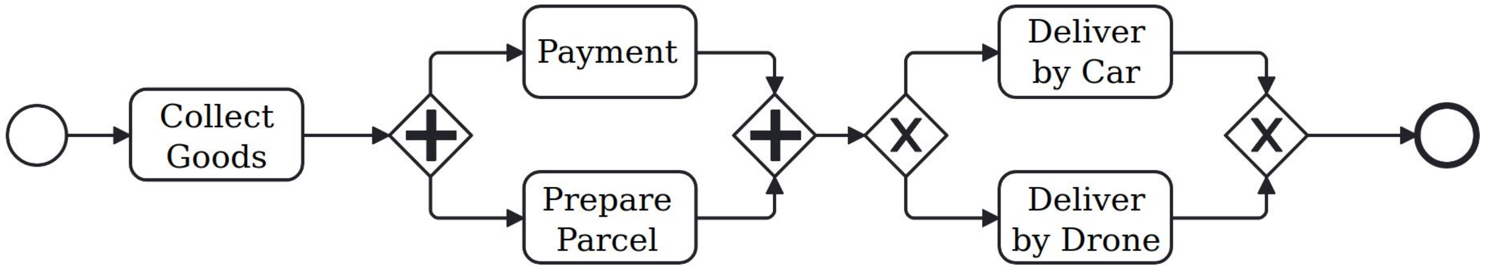


First Research Question

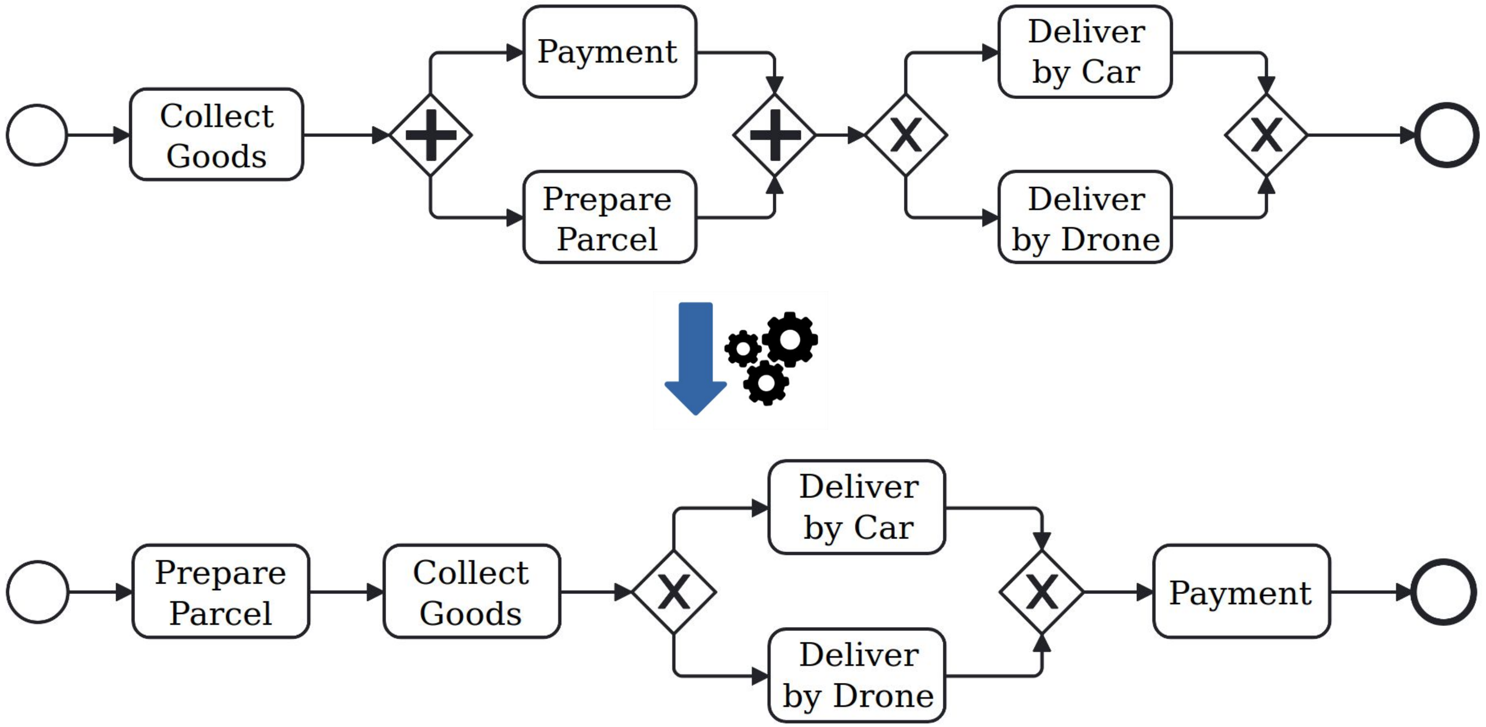


➢ How can you **optimise** a BPMN process in real-world conditions?

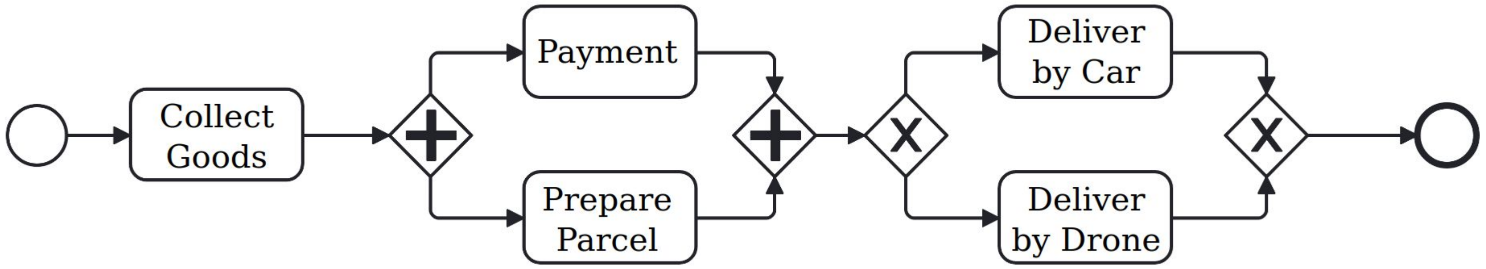
Second Research Question

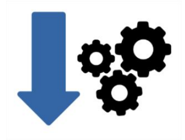


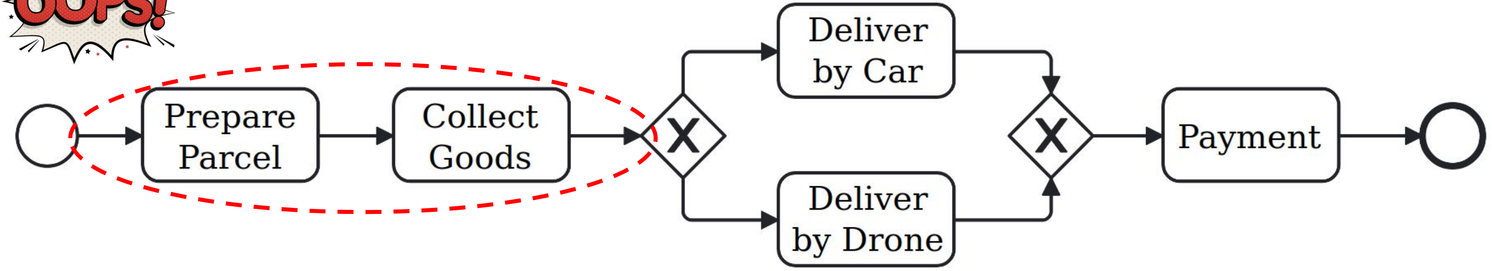
Second Research Question



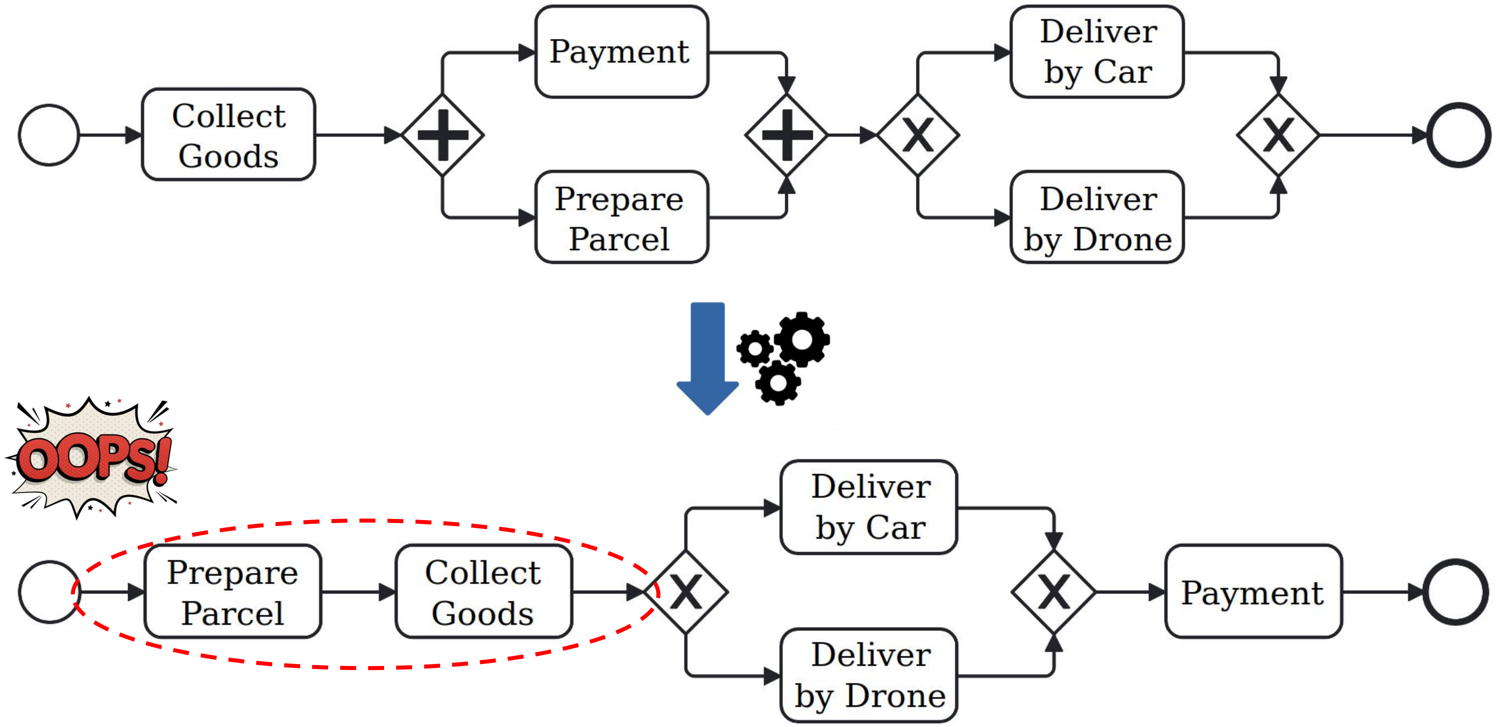
Second Research Question





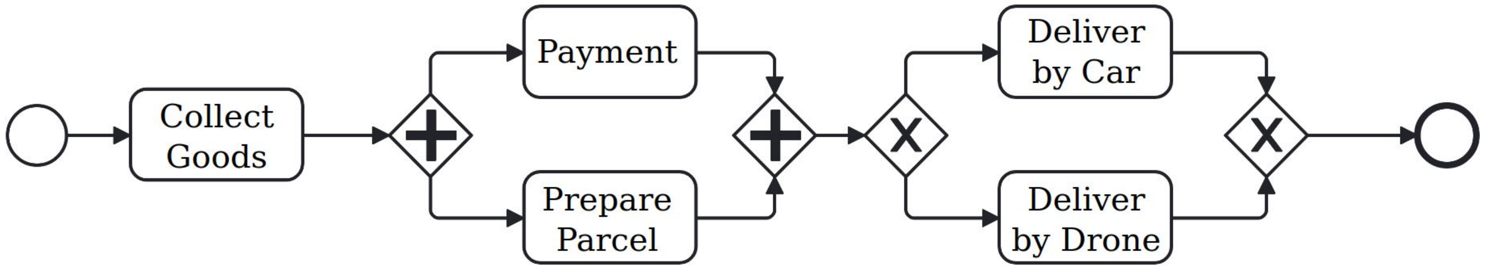


Second Research Question

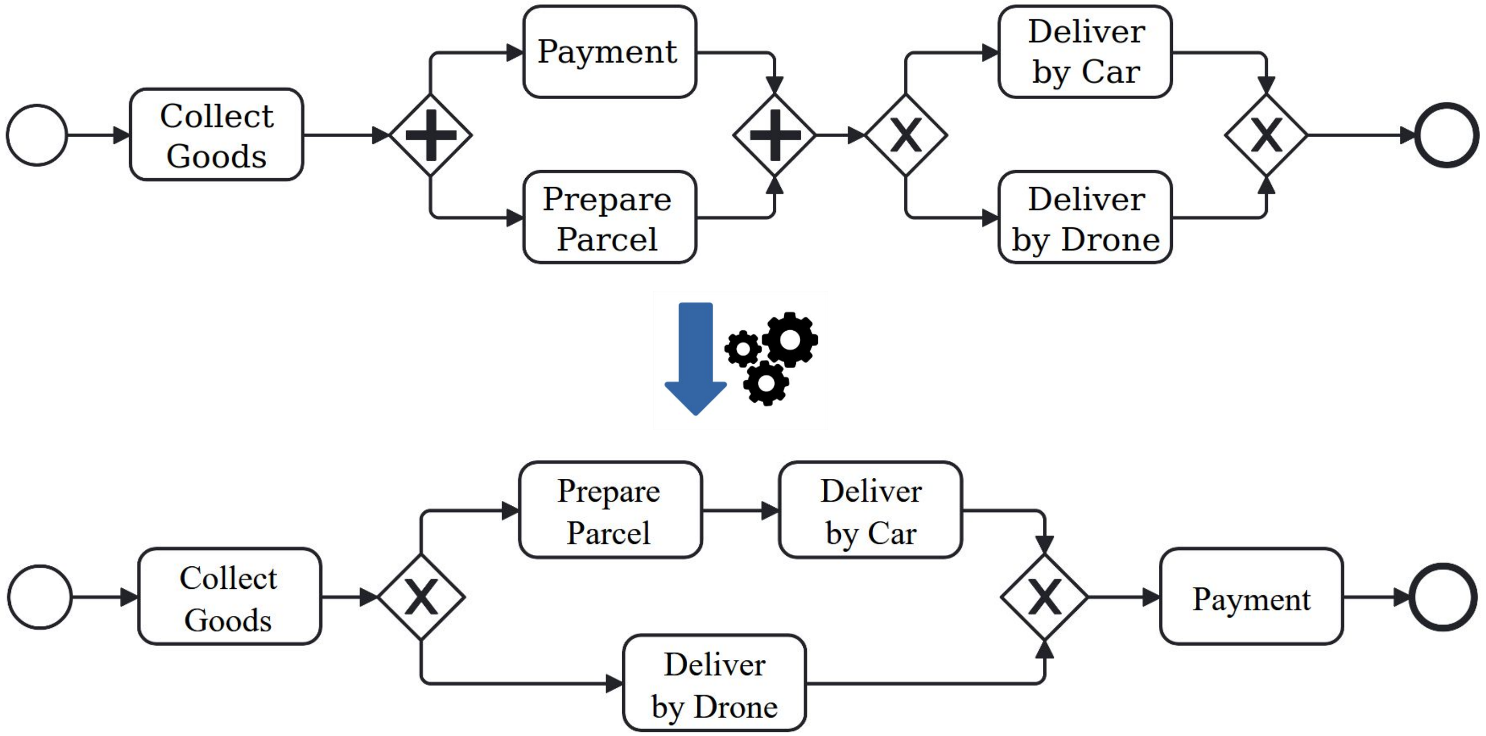


➢ How can you **preserve the logic/meaning** of the original process?

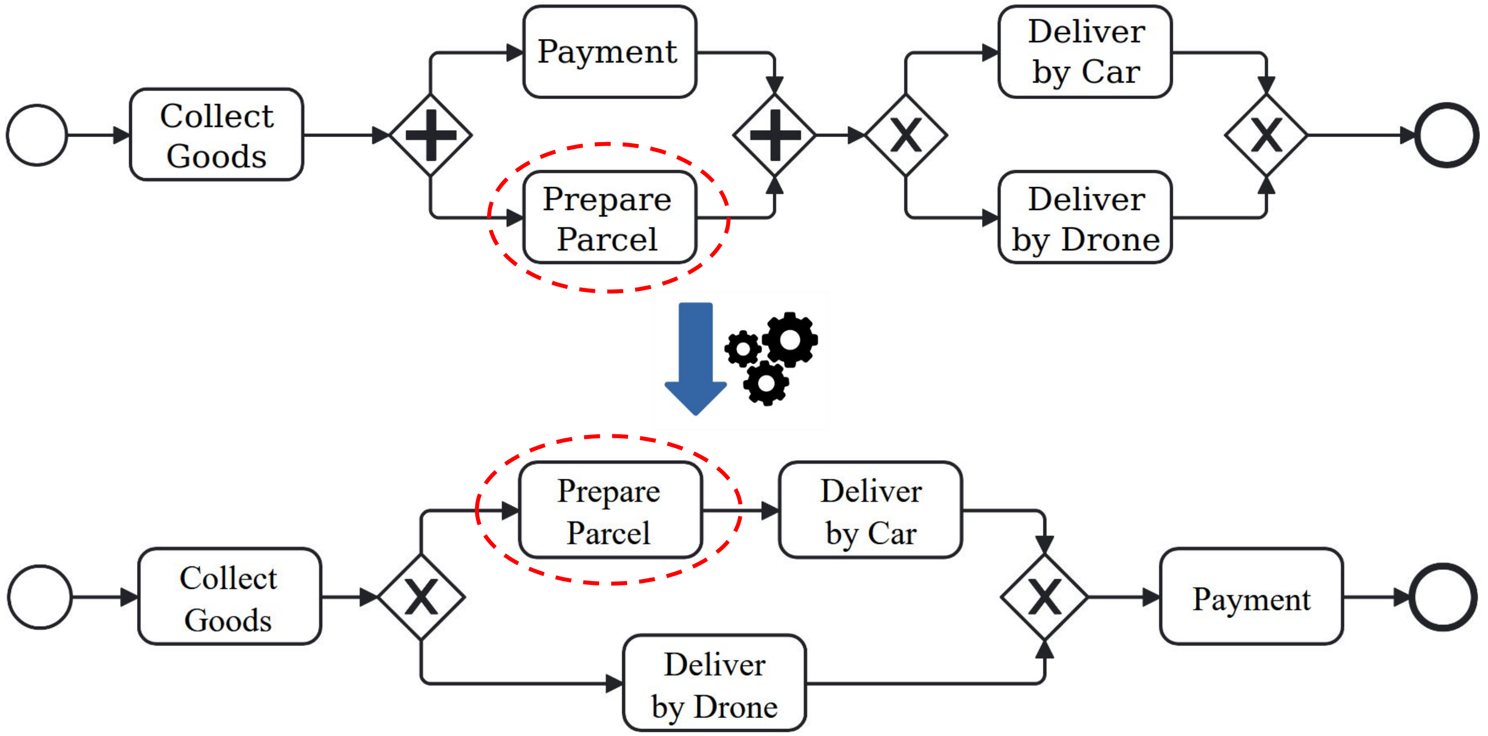
Third Research Question



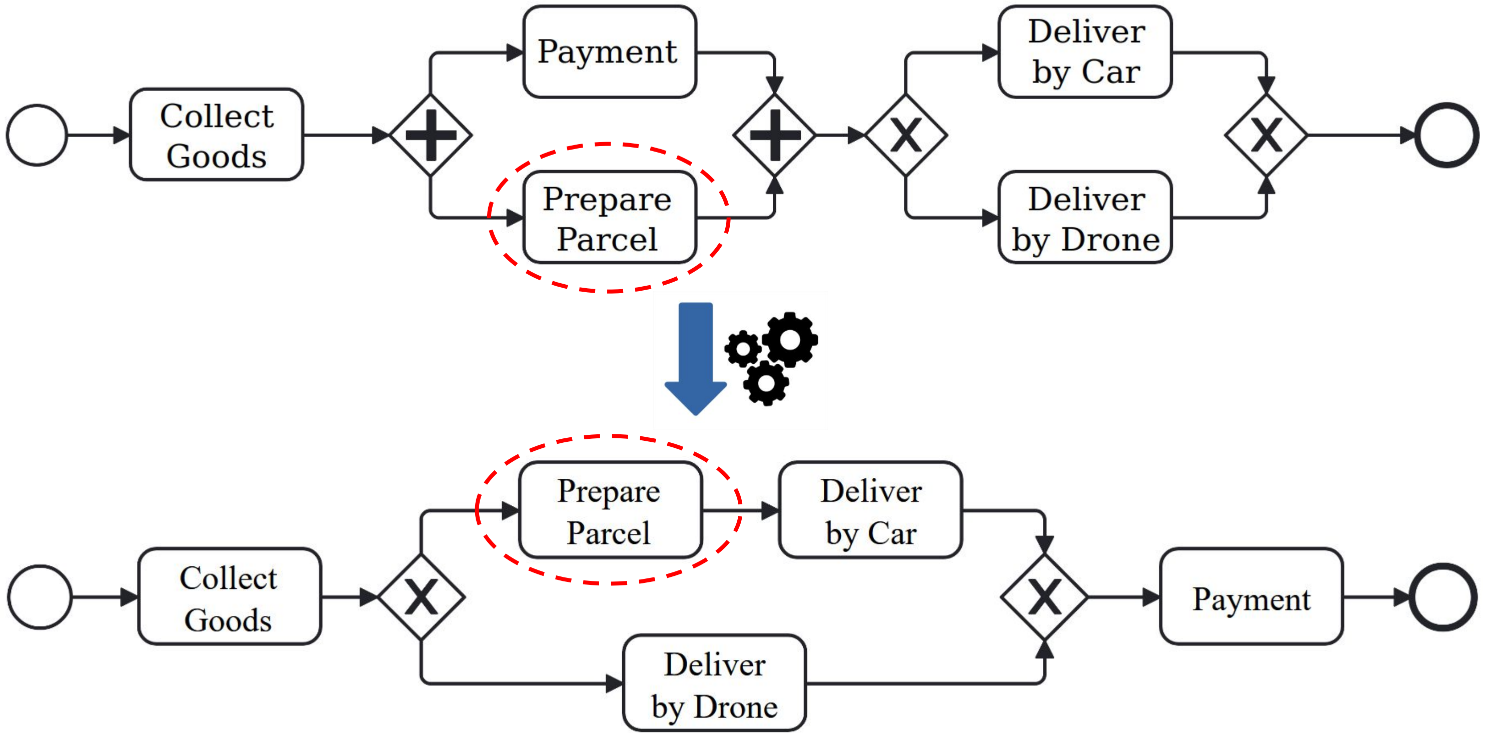
Third Research Question



Third Research Question

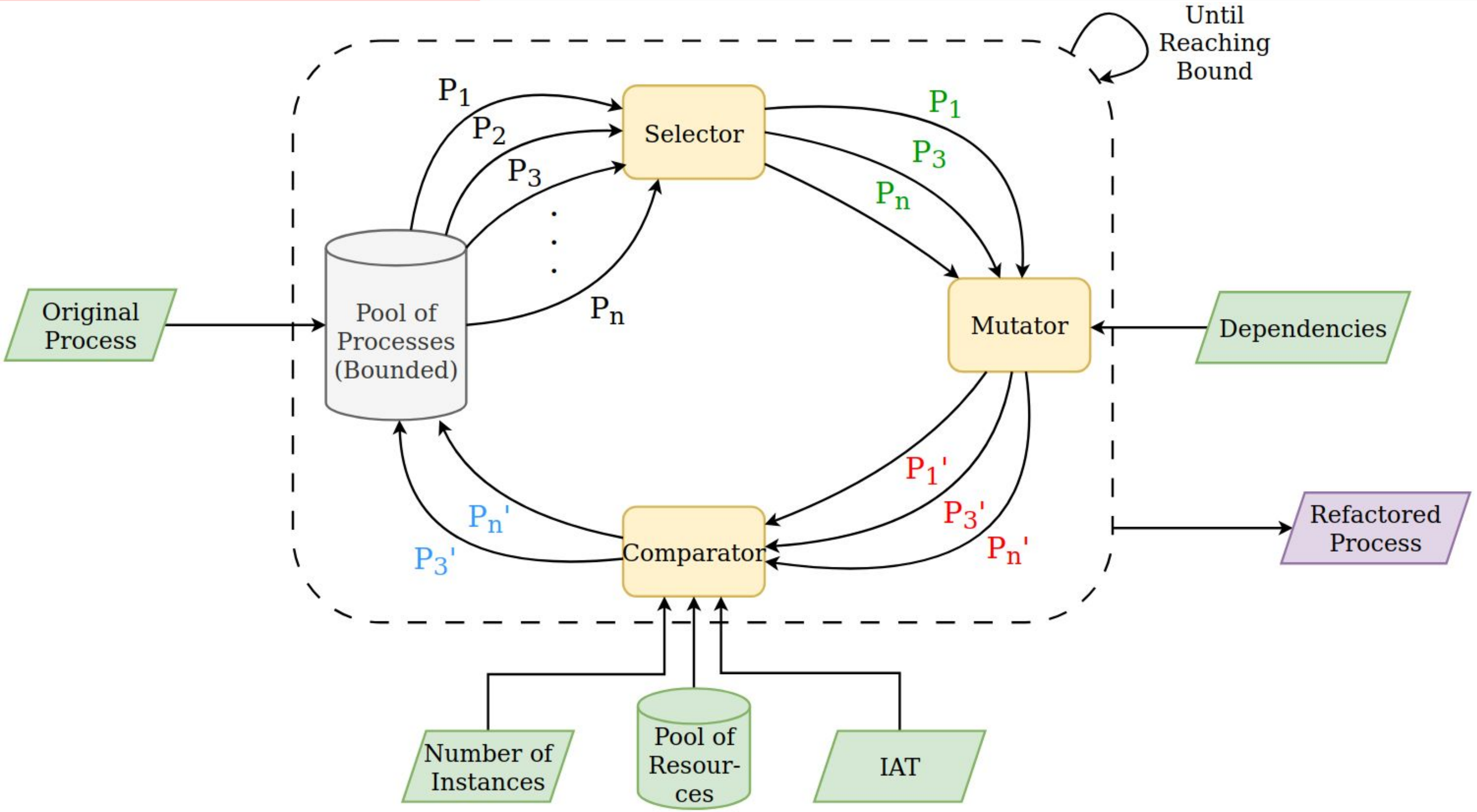


Third Research Question

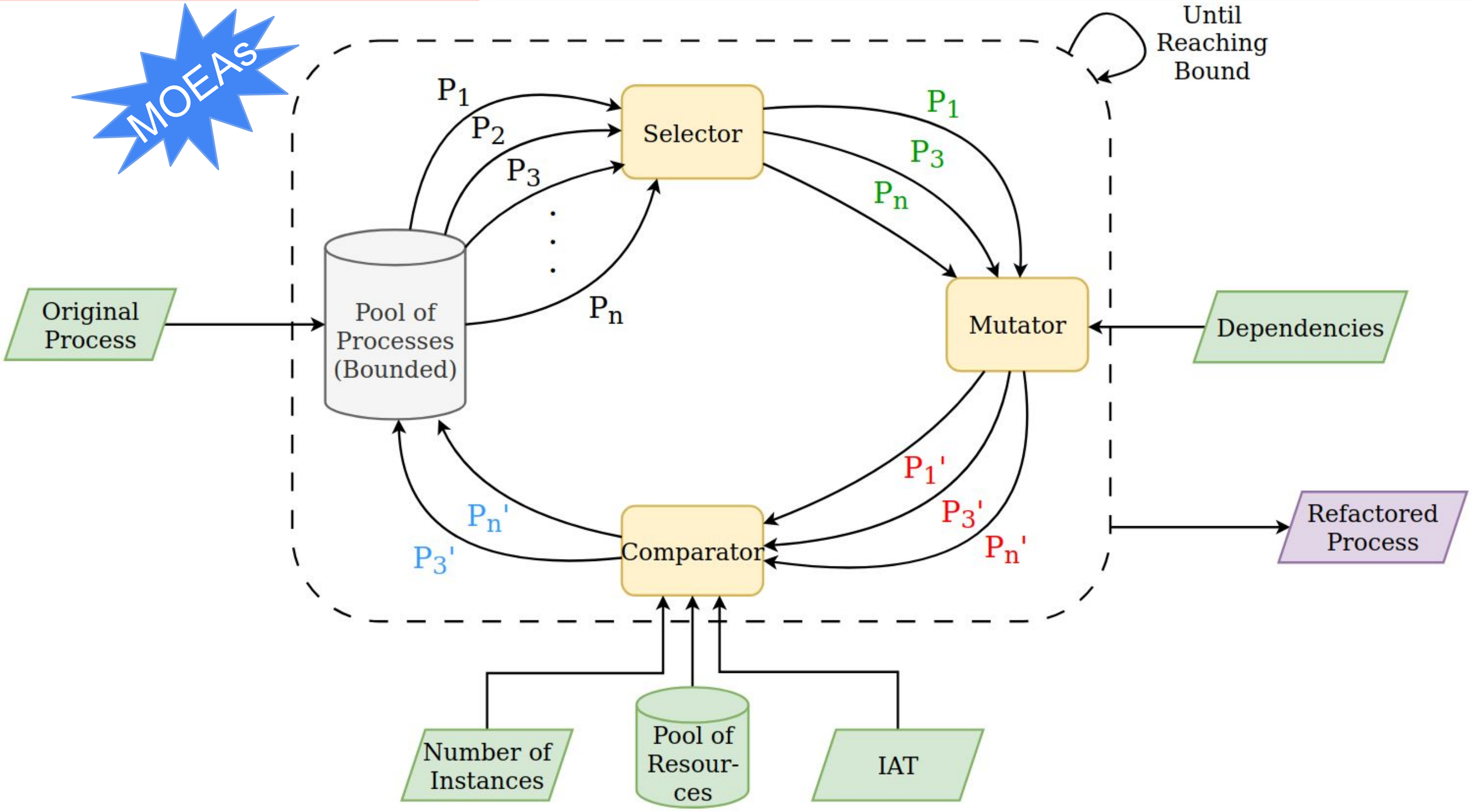


➢ How can you **preserve the structural semantics** of the original process?

Global Approach



Global Approach

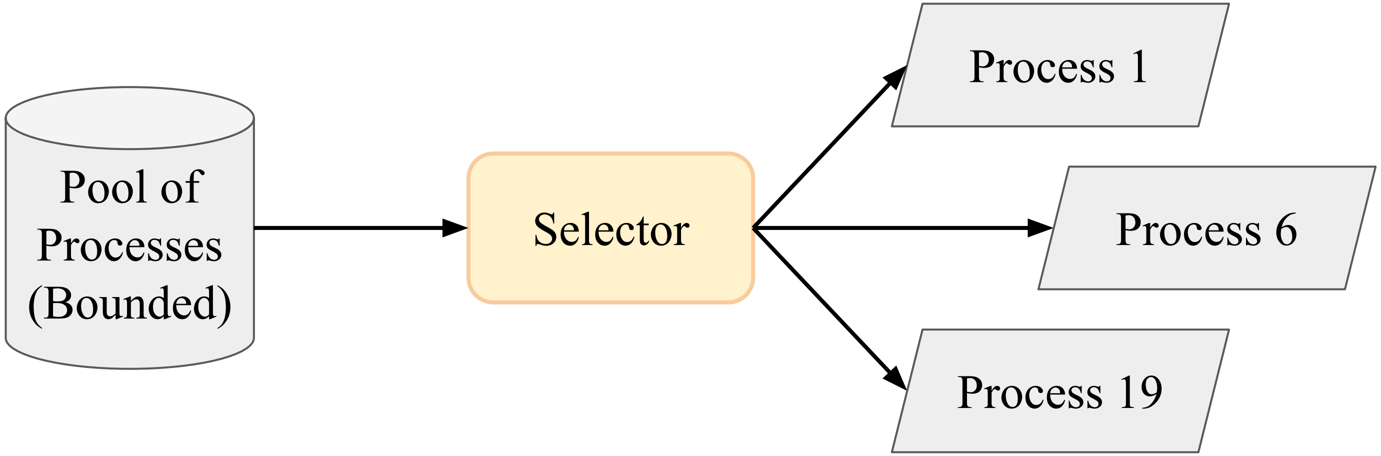


Step 1 – Selection

This step consists in **selecting** the processes that will be **mutated** from the **pool** of available **processes**.

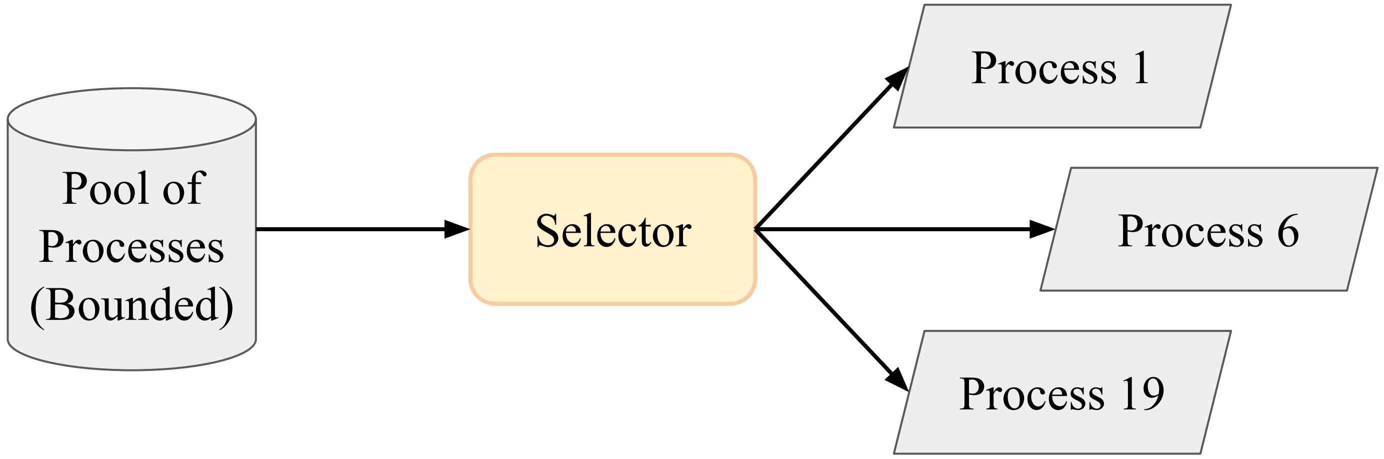
Step 1 – Selection

This step consists in **selecting** the processes that will be **mutated** from the **pool** of available **processes**.



Step 1 – Selection

This step consists in **selecting** the processes that will be **mutated** from the **pool** of available **processes**.



This step can be performed in **various ways**, but is usually at the **discretion of the used algorithm**, which is also the case in this approach.

Step 2 – Mutation

This step consists in **mutating** the selected processes.

Step 2 – Mutation

This step consists in **mutating** the selected processes.

In this approach, a **mutation** is a **refactoring operation** that changes the structure of the process by **moving one** of its **tasks** from one place to another.

Step 2 – Mutation

This step consists in **mutating** the selected processes.

In this approach, a **mutation** is a **refactoring operation** that changes the structure of the process by **moving one** of its **tasks** from one place to another.

This operation is **sensitive**, as it must **provide guarantees** regarding the preservation of the **meaning of the process**, and of its **structural semantics**.

Step 2 – Mutation

This step consists in **mutating** the selected processes.

In this approach, a **mutation** is a **refactoring operation** that changes the structure of the process by **moving one** of its **tasks** from one place to another.

This operation is **sensitive**, as it must **provide guarantees** regarding the preservation of the **meaning of the process**, and of its **structural semantics**.

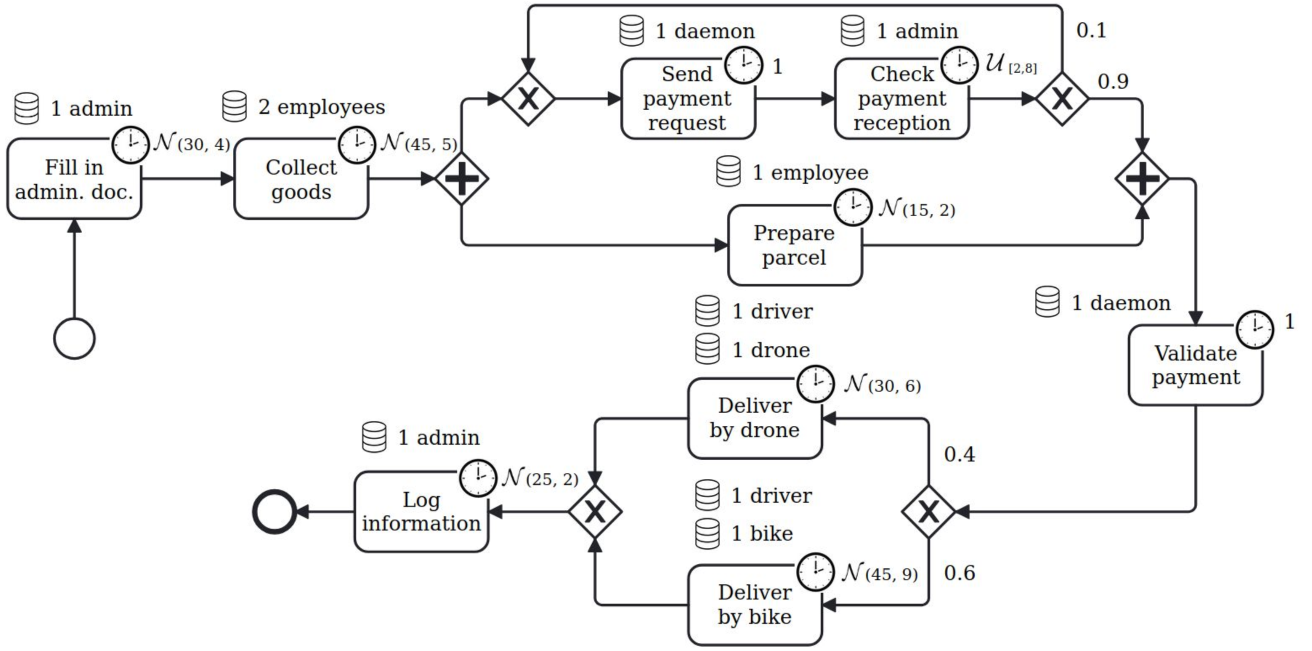
Such **guarantees** are **obtained** with the help of two mechanisms: **user-defined task dependencies**, and **refactoring patterns**.

Step 2 – Task Dependencies

A **dependency** between two tasks is a **relationship** indicating the (immutable) **order** in which two tasks of the process must be **executed** to preserve its meaning.

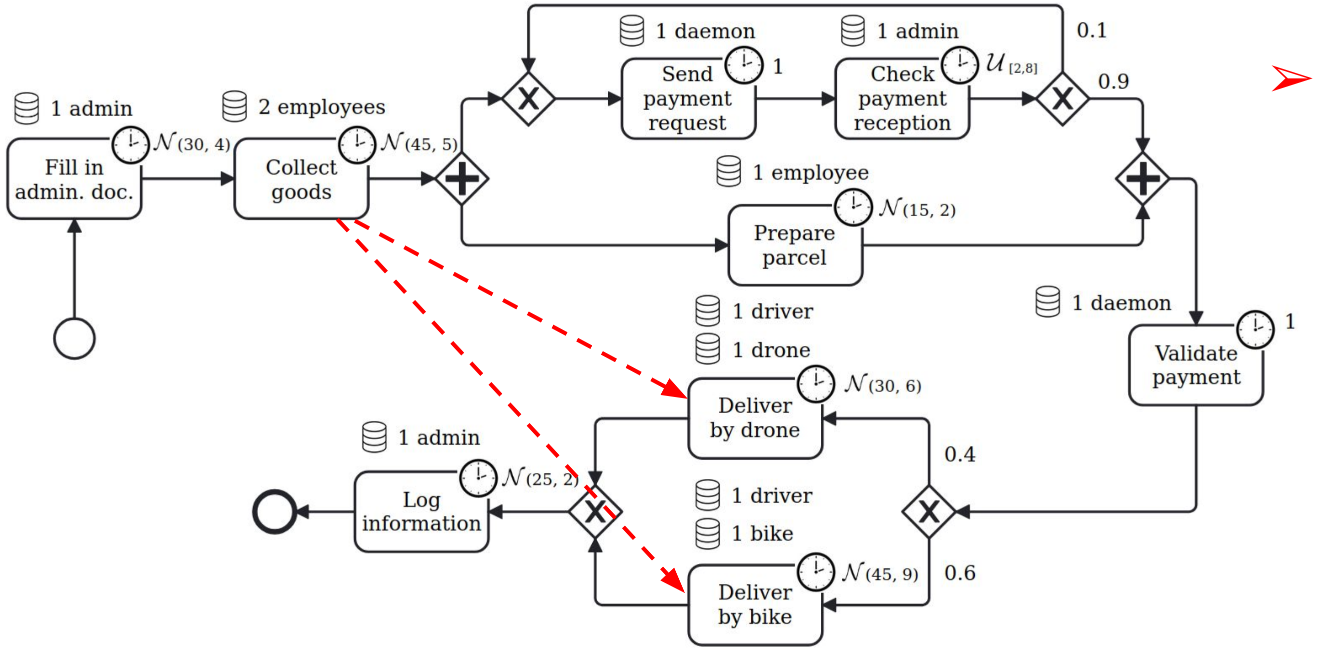
Step 2 – Task Dependencies

A **dependency** between two tasks is a **relationship** indicating the (immutable) **order** in which two tasks of the process must be **executed** to preserve its meaning.



Step 2 – Task Dependencies

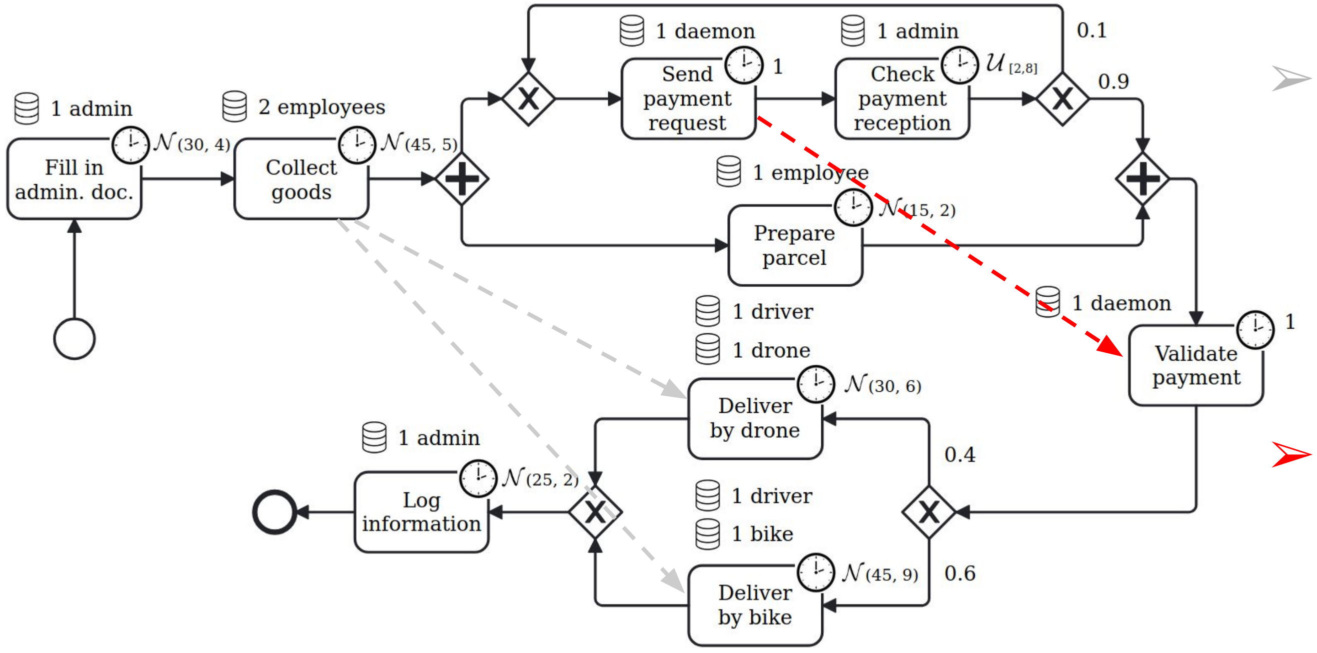
A **dependency** between two tasks is a **relationship** indicating the (immutable) **order** in which two tasks of the process must be **executed** to preserve its meaning.



➢ Goods must be collected before being delivered.

Step 2 – Task Dependencies

A **dependency** between two tasks is a **relationship** indicating the (immutable) **order** in which two tasks of the process must be **executed** to preserve its meaning.



- ➢ Goods must be collected before being delivered.
- ➢ A payment request must be sent before being validated.

Step 2 – Refactoring Patterns – Introduction

A **refactoring pattern** is an operation **applied to a task and a process** which aims at **moving this task** from its **original position** in the process to **another** well-defined **position** in that process.

Step 2 – Refactoring Patterns – Introduction

A **refactoring pattern** is an operation **applied to a task and a process** which aims at **moving this task** from its **original position** in the process to **another** well-defined **position** in that process.

This operation is in charge of **ensuring** the preservation of the **structural semantics** of the process.

Step 2 – Refactoring Patterns – Introduction

A **refactoring pattern** is an operation **applied to a task and a process** which aims at **moving this task** from its **original position** in the process to **another** well-defined **position** in that process.

This operation is in charge of **ensuring** the preservation of the **structural semantics** of the process.

To **facilitate** this preservation, it is **not based on the BPMN** process itself, **but on** another representation, called **sequence graph**.

Step 2 – Refactoring Patterns – Introduction

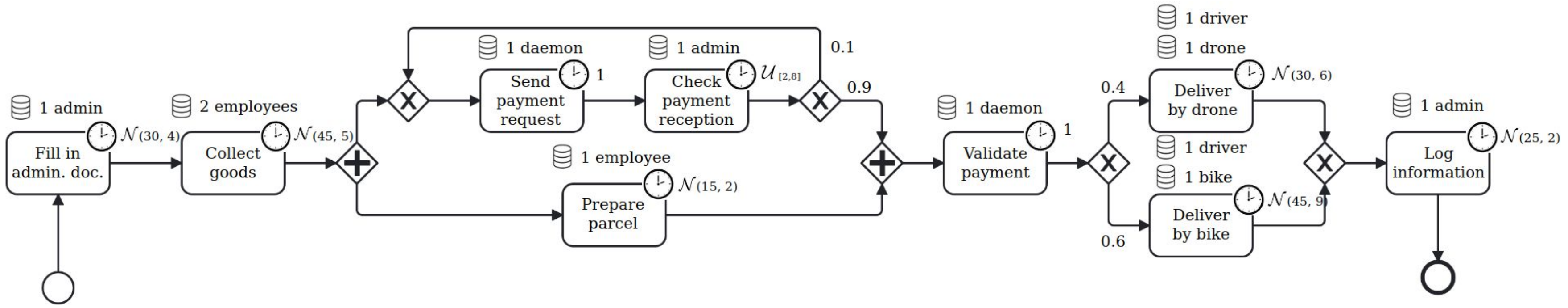
A **refactoring pattern** is an operation **applied to a task and a process** which aims at **moving this task** from its **original position** in the process to **another** well-defined **position** in that process.

This operation is in charge of **ensuring** the preservation of the **structural semantics** of the process.

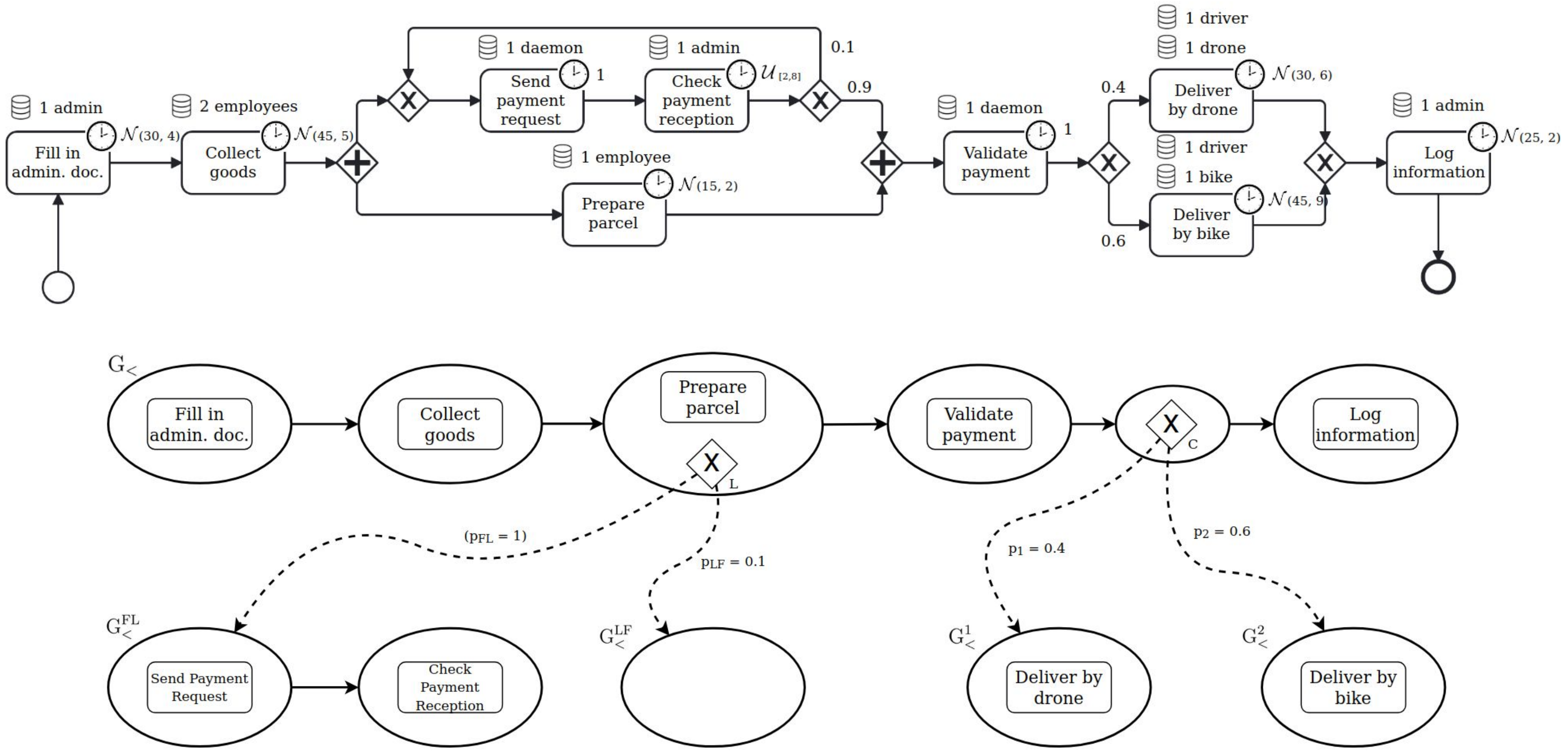
To **facilitate** this preservation, it is **not based on the BPMN** process itself, **but on** another representation, called **sequence graph**.

A sequence graph is a **hierarchical structure** composed of **nodes** and **edges**.

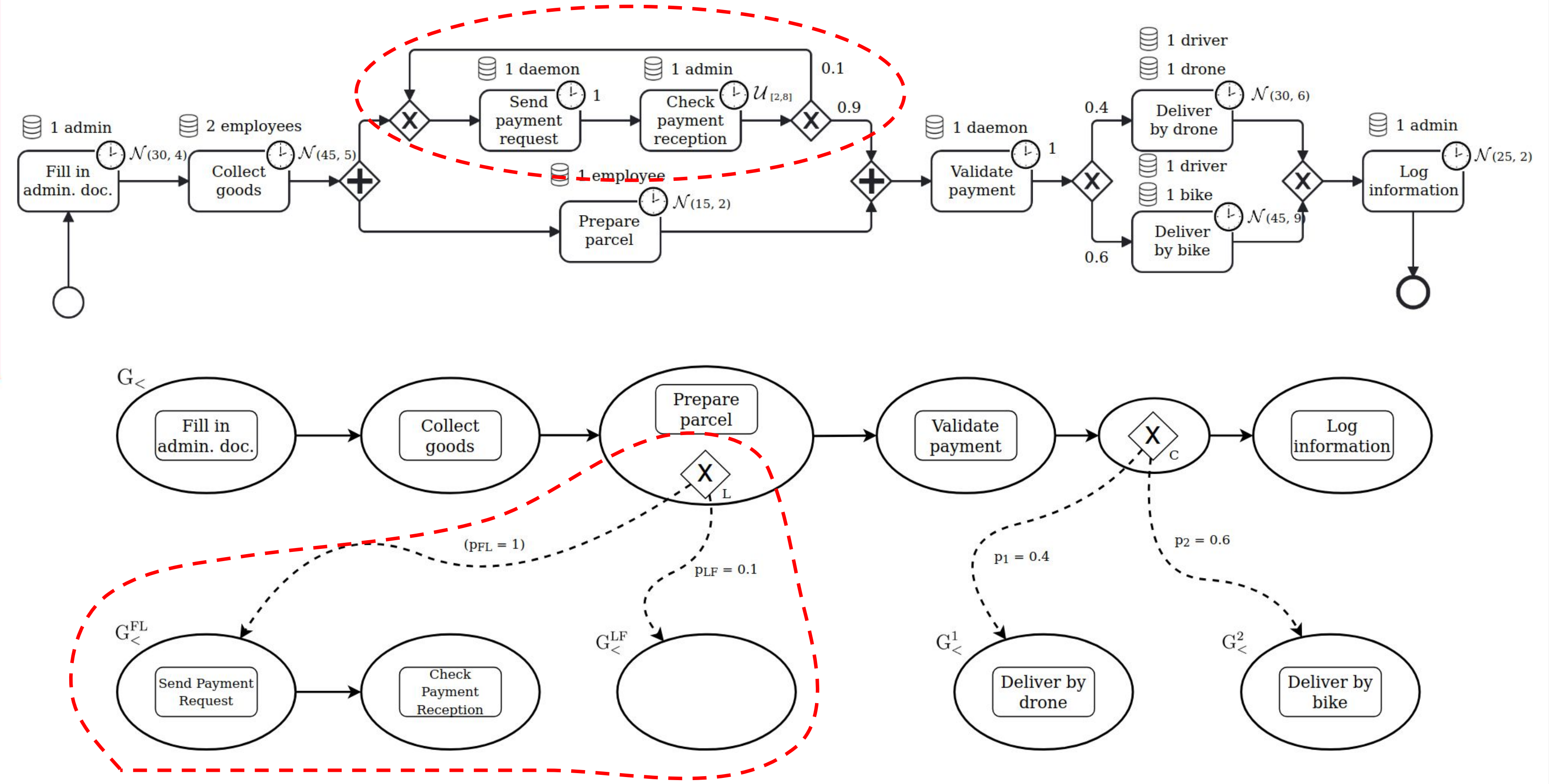
Step 2 – Sequence Graph



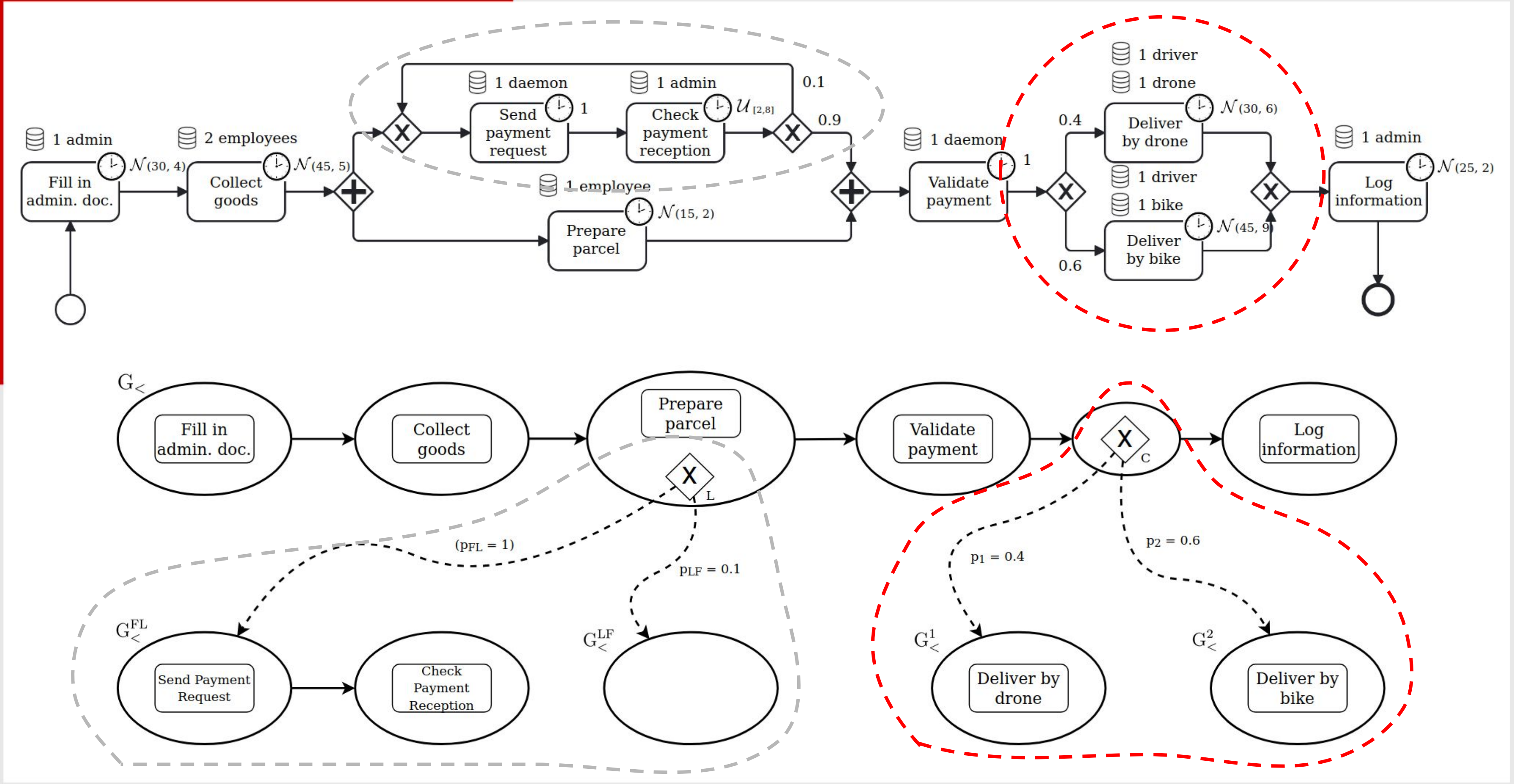
Step 2 – Sequence Graph



Step 2 – Sequence Graph

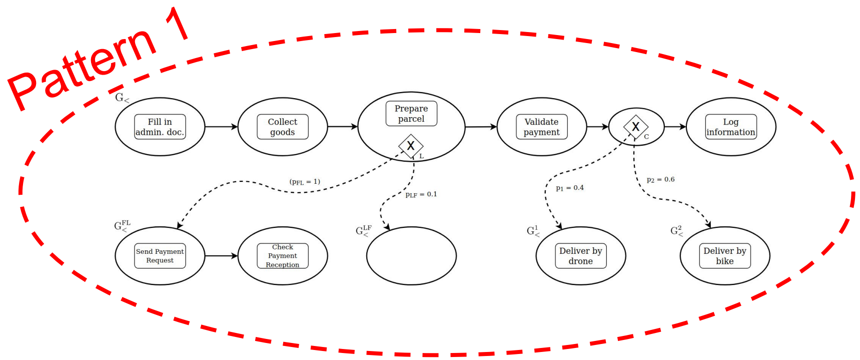


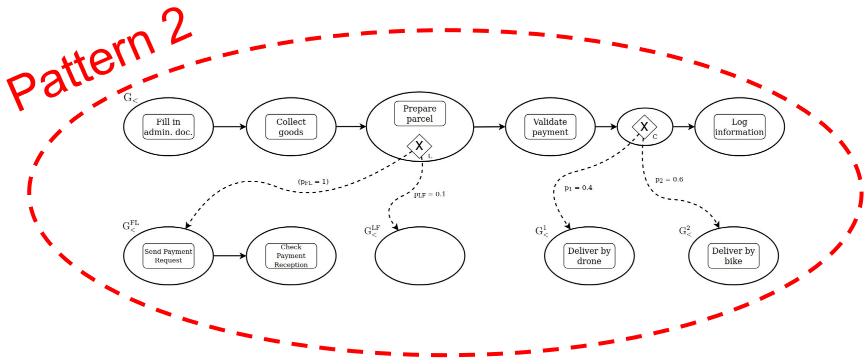
Step 2 – Sequence Graph

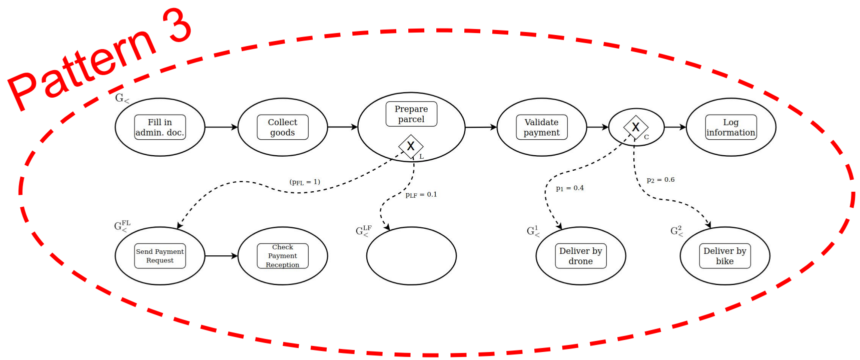


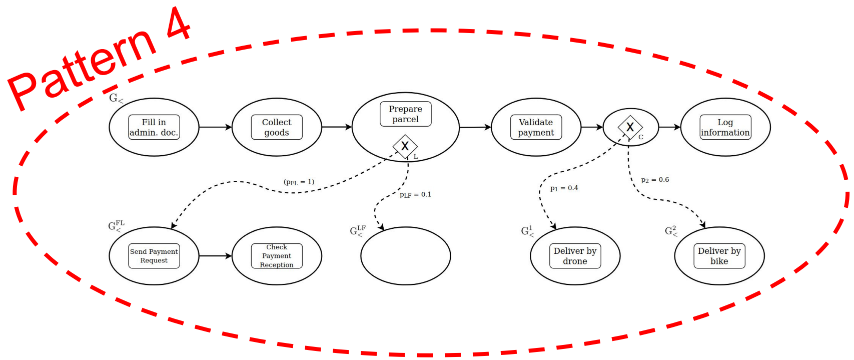
Step 2 – Refactoring Patterns

The **movement** of a task in the process is **ruled** by **4** refactoring **patterns**.



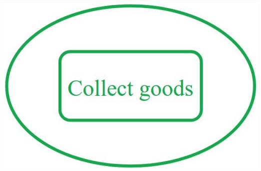


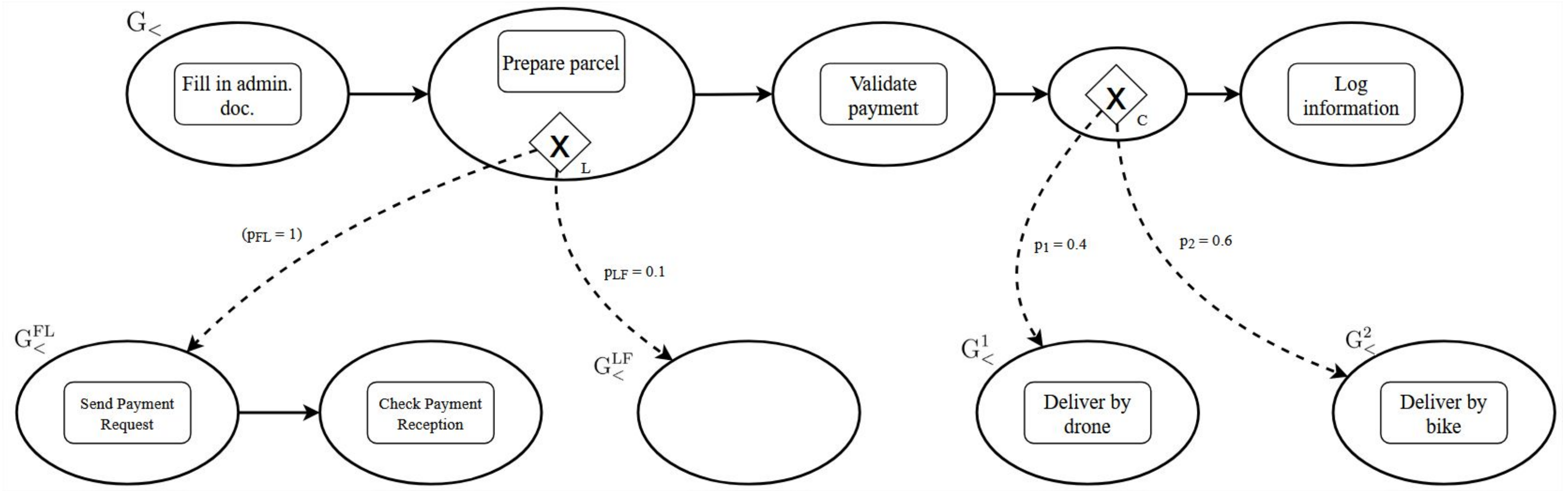




Step 2 – Pattern 1

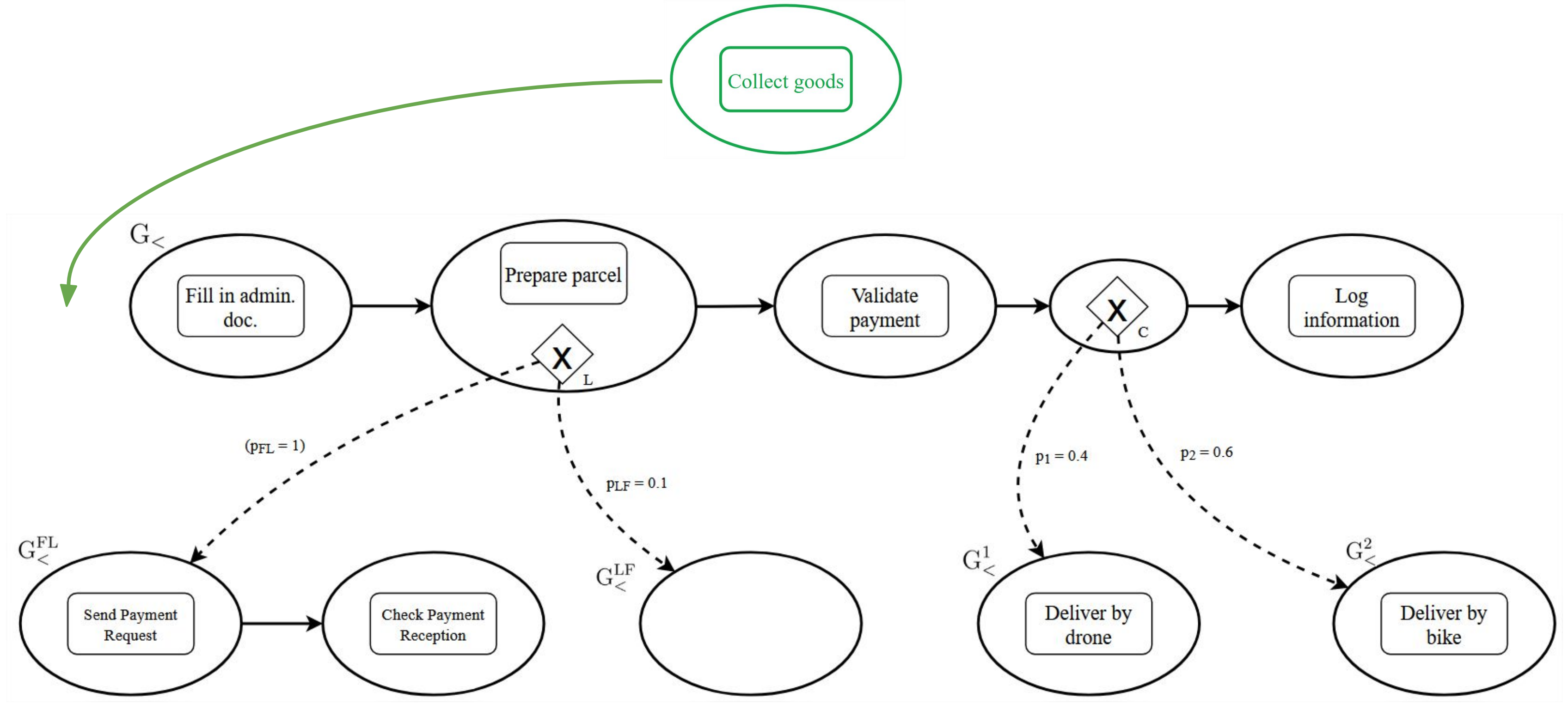
The **first pattern** consists in inserting the task before/after any node of the graph.





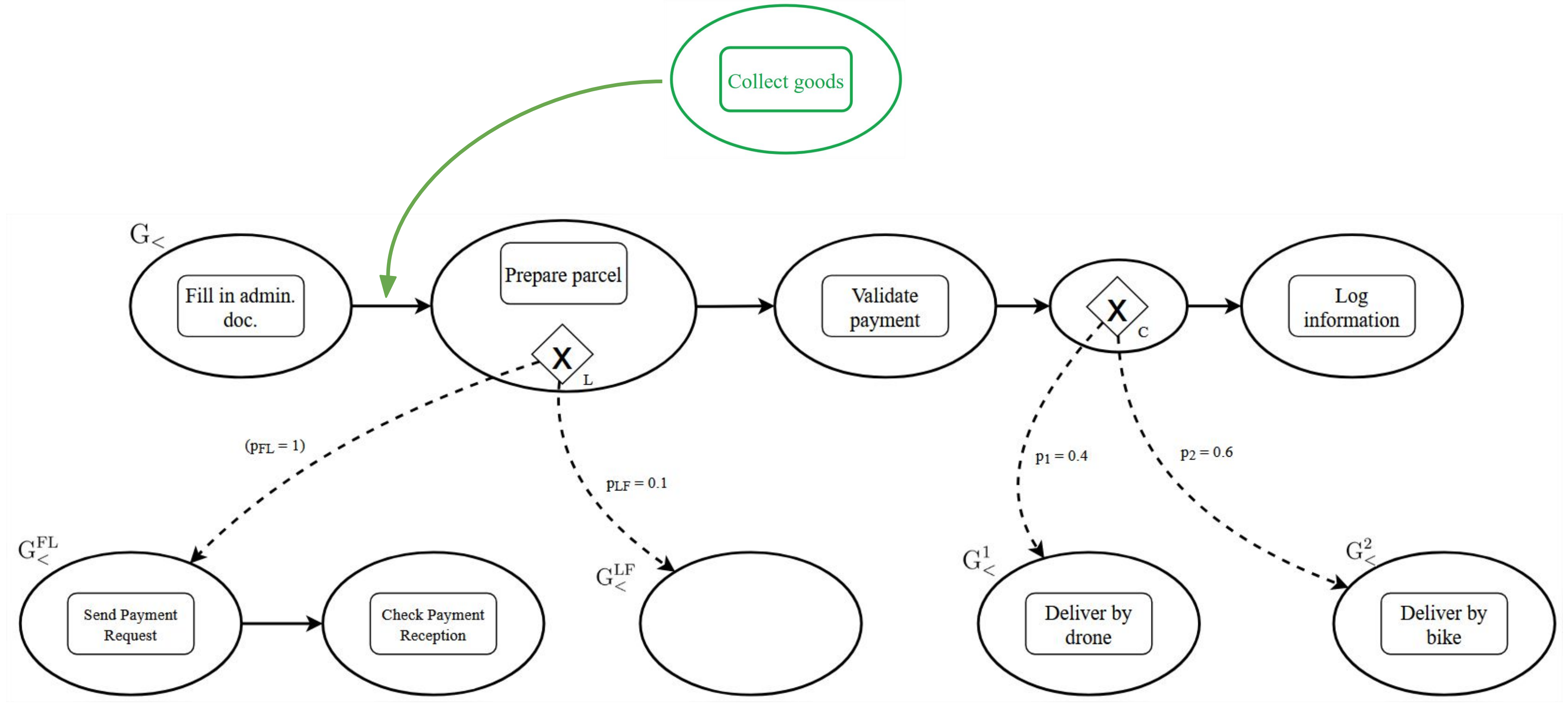
Step 2 – Pattern 1

The **first pattern** consists in inserting the task before/after any node of the graph.



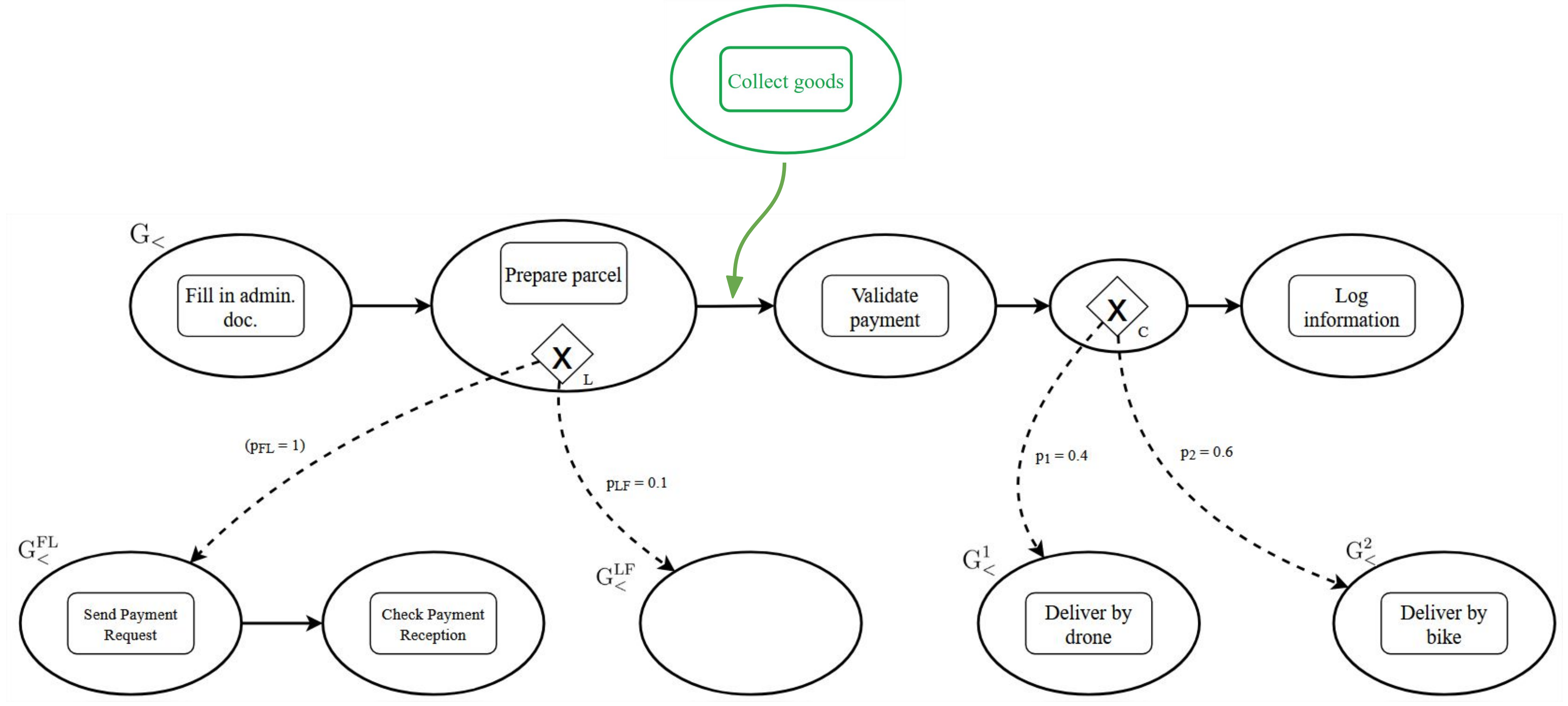
Step 2 – Pattern 1

The **first pattern** consists in inserting the task before/after any node of the graph.



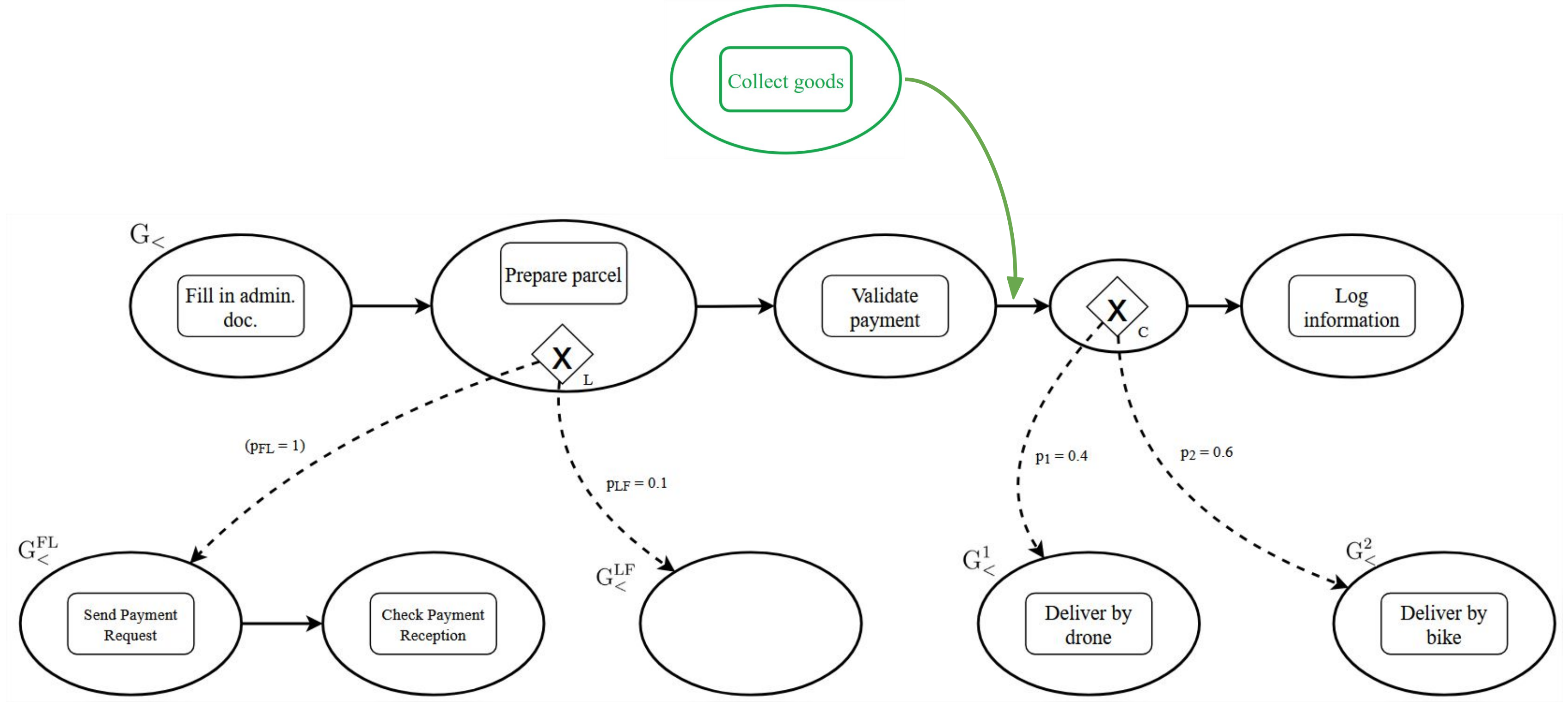
Step 2 – Pattern 1

The **first pattern** consists in inserting the task before/after any node of the graph.



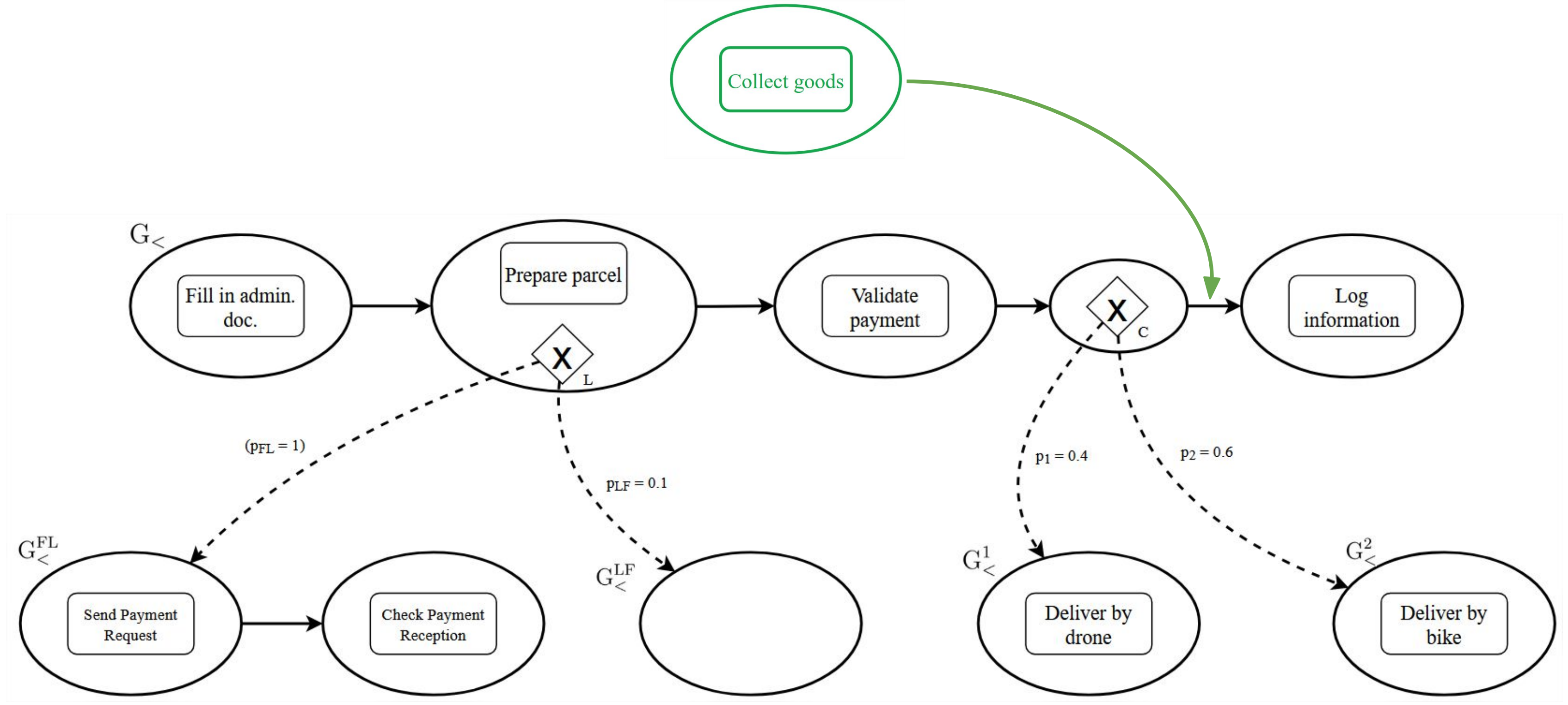
Step 2 – Pattern 1

The **first pattern** consists in inserting the task before/after any node of the graph.



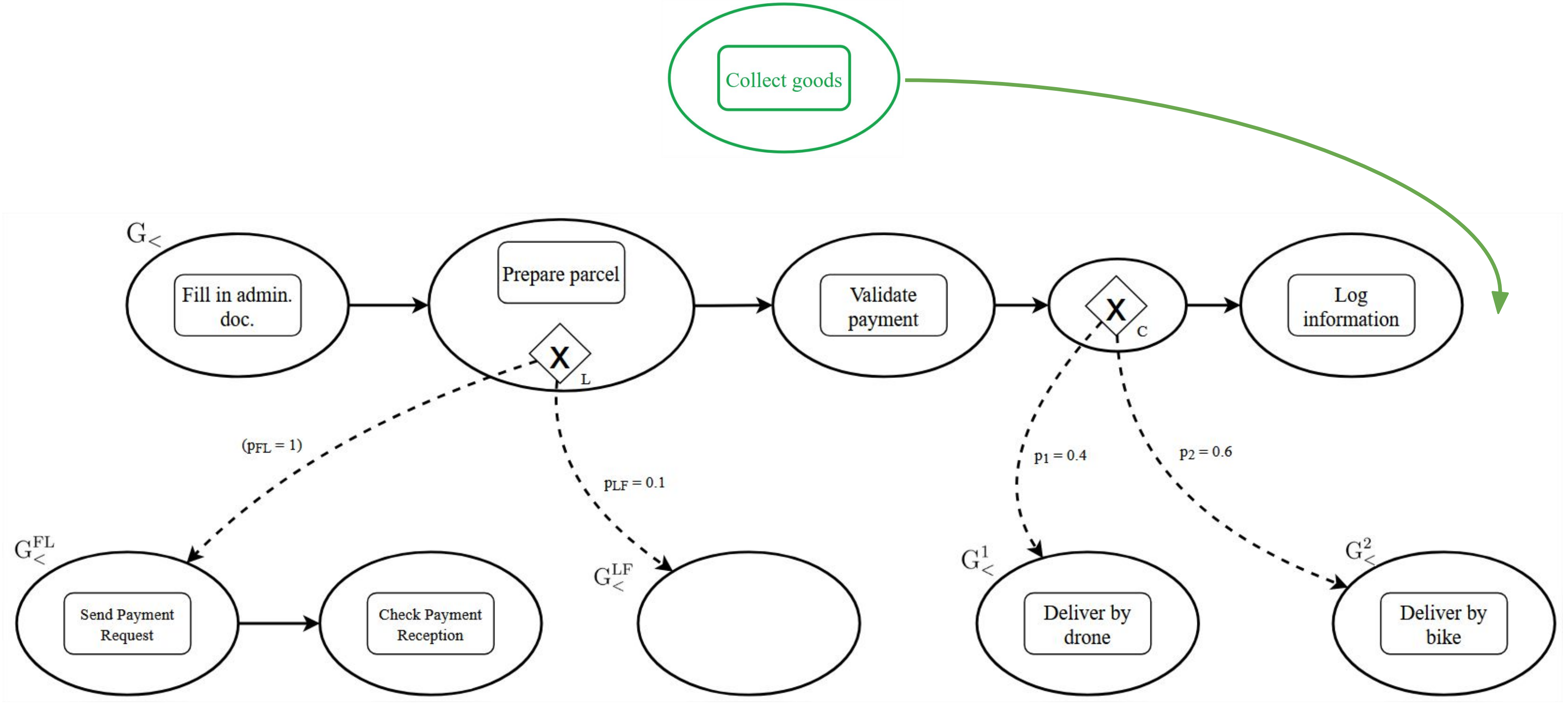
Step 2 – Pattern 1

The **first pattern** consists in inserting the task before/after any node of the graph.



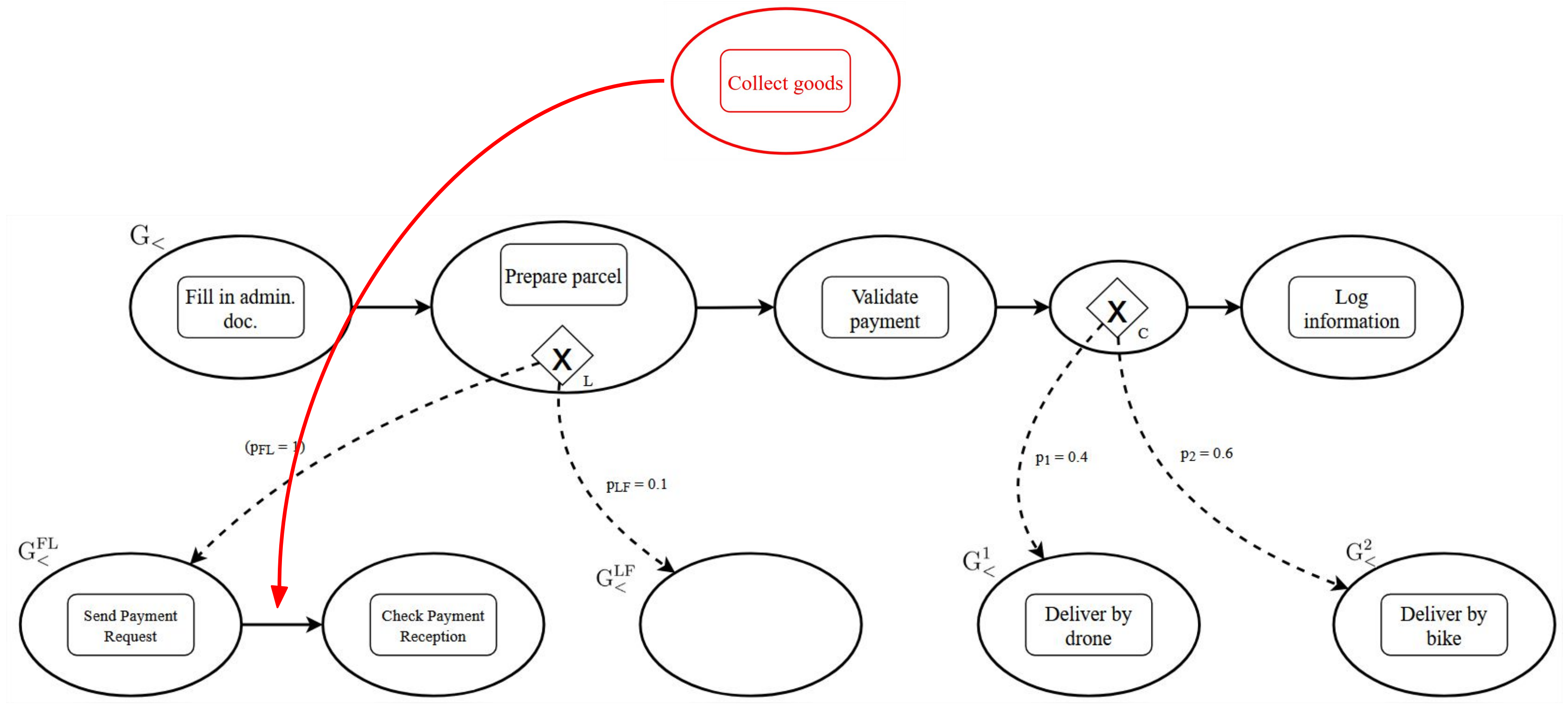
Step 2 – Pattern 1

The **first pattern** consists in inserting the task before/after any node of the graph.



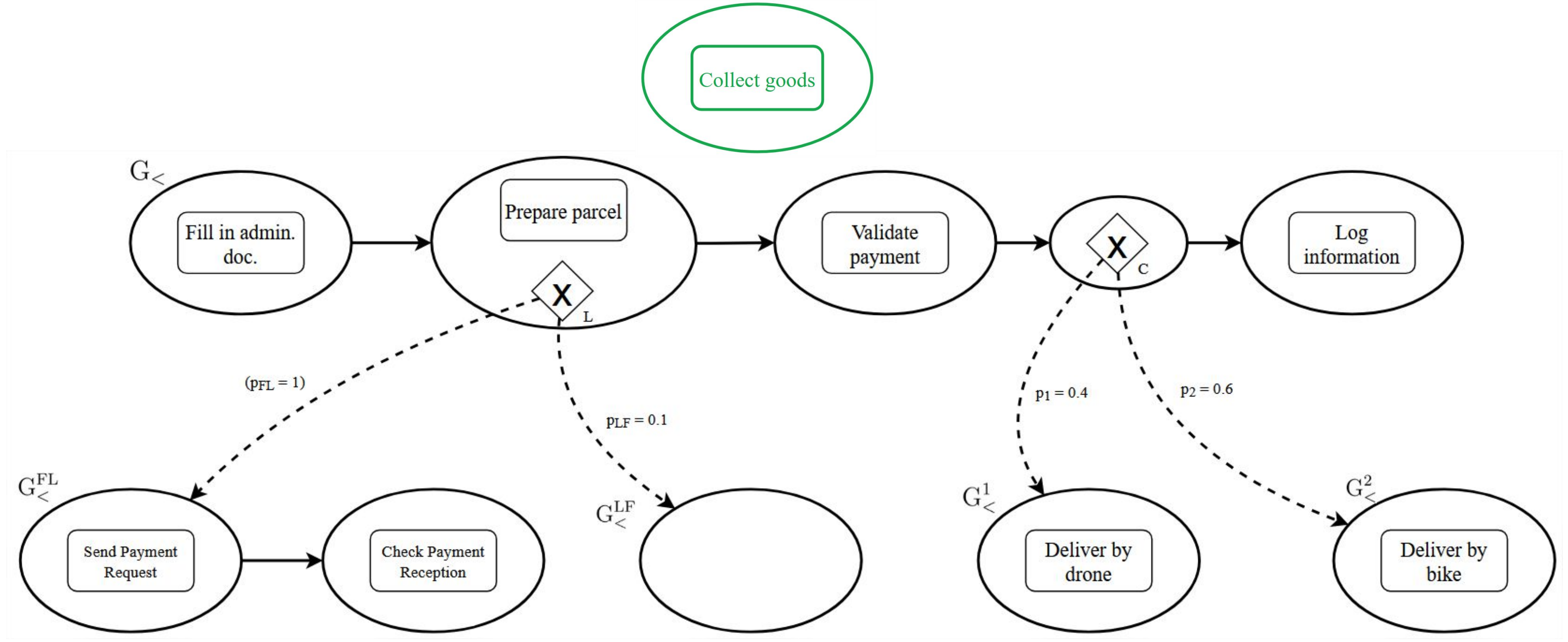
Step 2 – Pattern 1

The **first pattern** consists in inserting the task before/after any node of the graph.



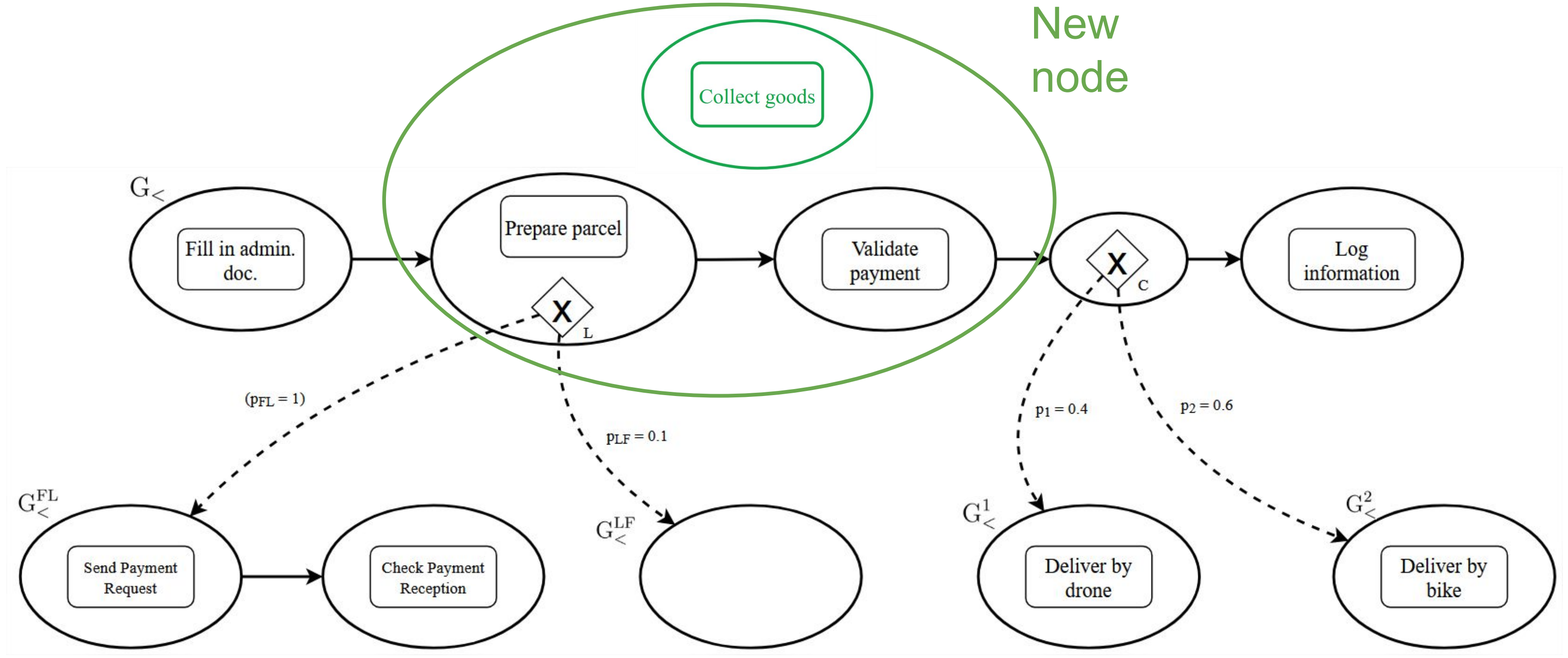
Step 2 – Pattern 2

The **second pattern** consists in inserting the task in parallel of any non-empty subsequence of nodes of the graph.



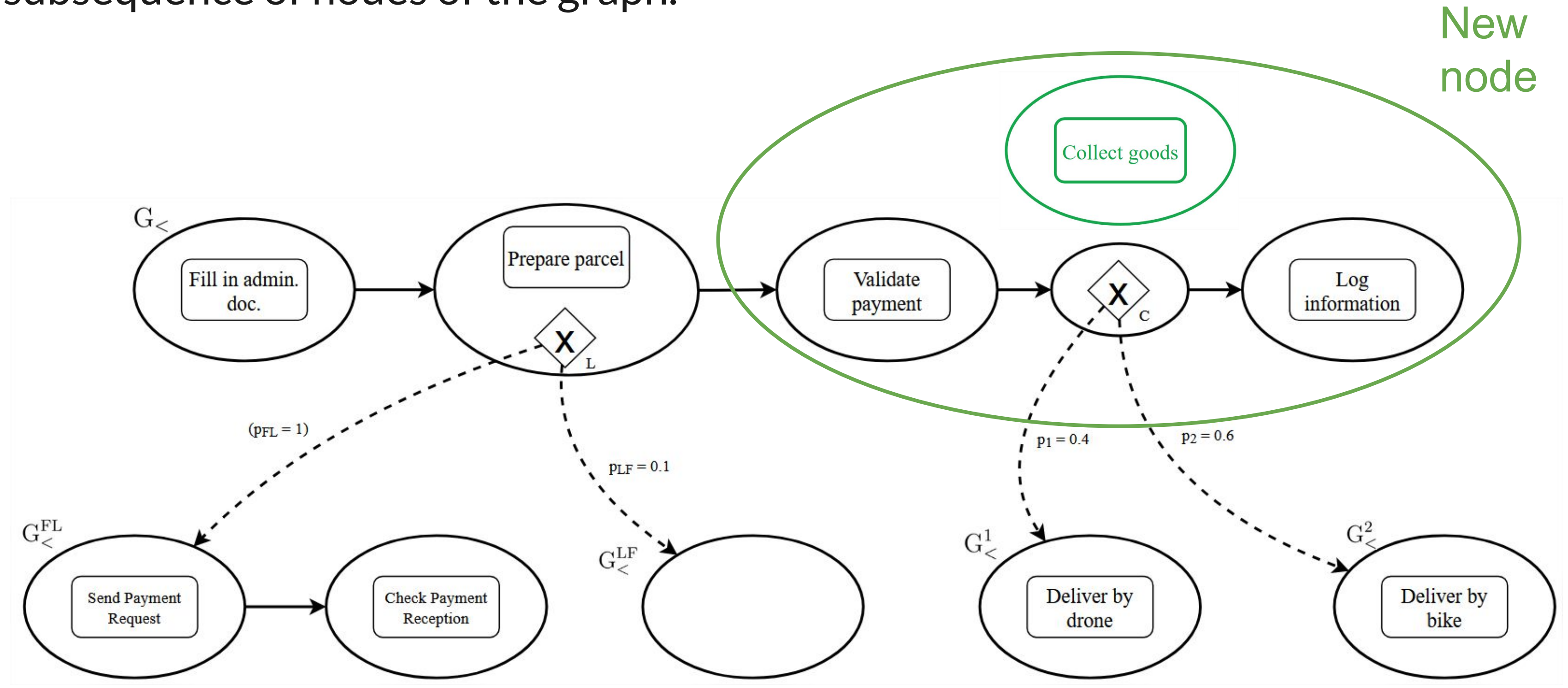
Step 2 – Pattern 2

The **second pattern** consists in inserting the task in parallel of any non-empty subsequence of nodes of the graph.



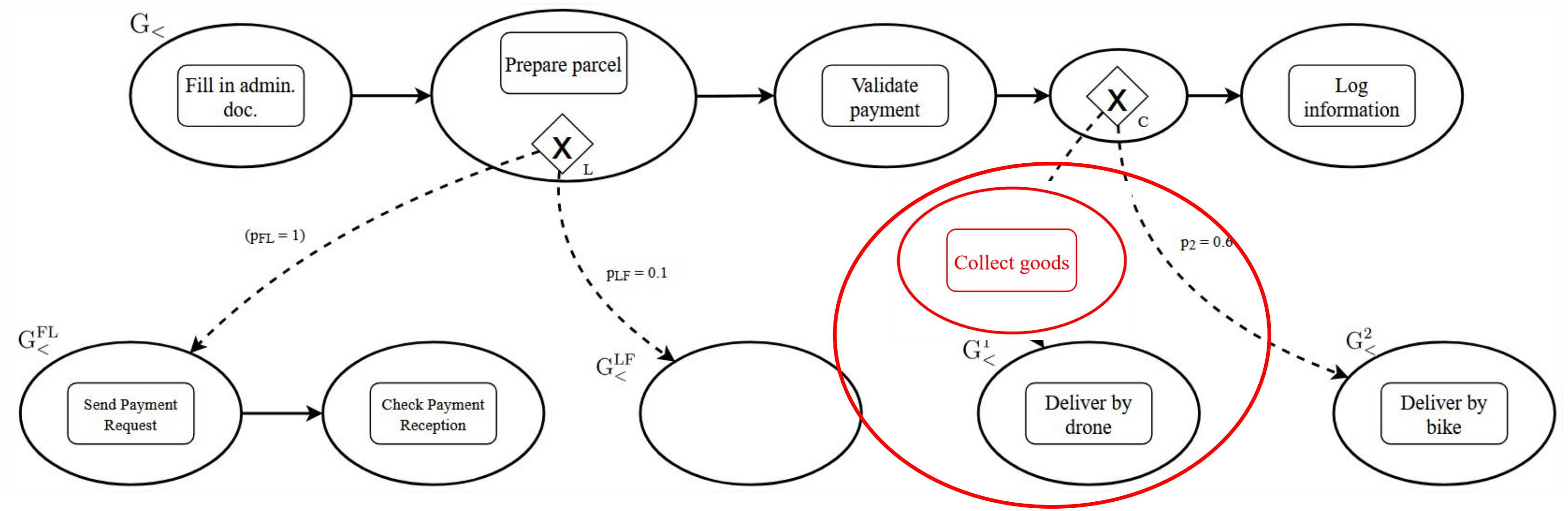
Step 2 – Pattern 2

The **second pattern** consists in inserting the task in parallel of any non-empty subsequence of nodes of the graph.



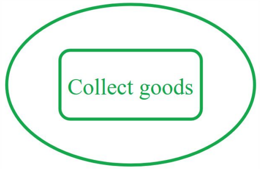
Step 2 – Pattern 2

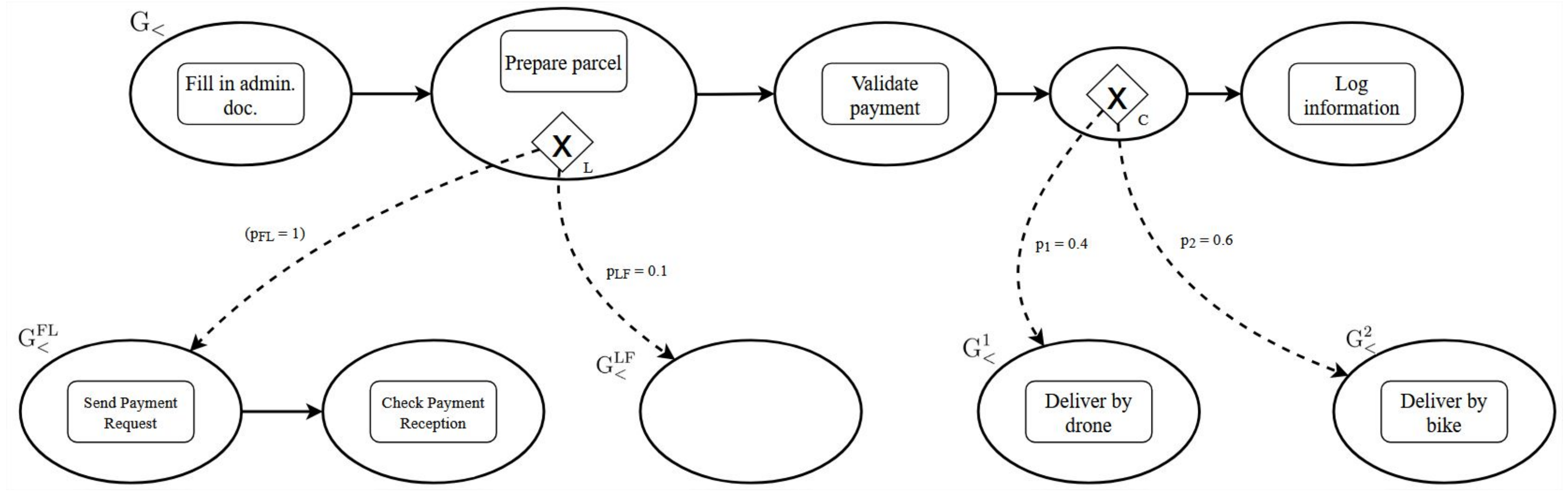
The **second pattern** consists in inserting the task in parallel of any non-empty subsequence of nodes of the graph.



Step 2 – Pattern 3

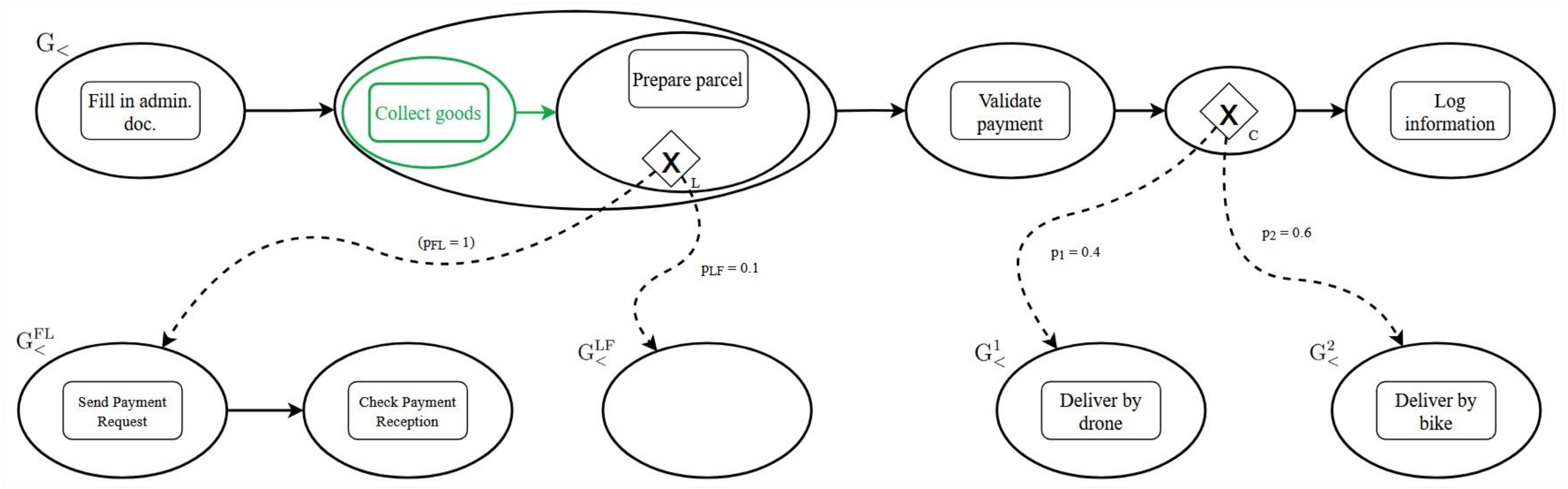
The **third pattern** consists in inserting the task before or after any combination of elements of any node of the graph.





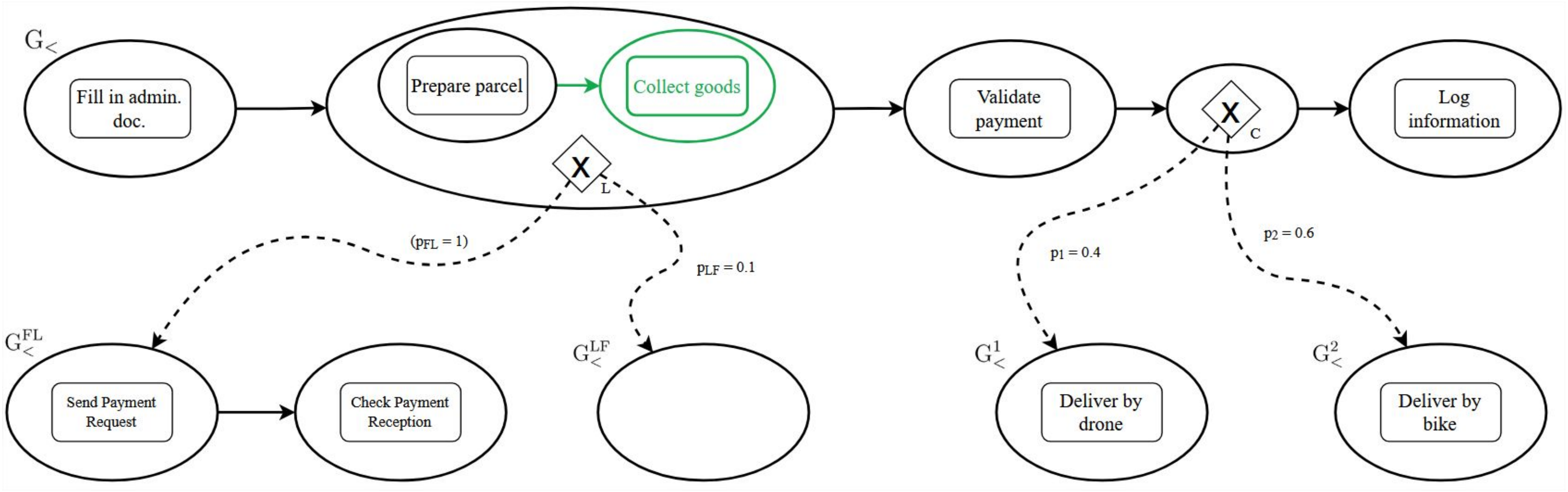
Step 2 – Pattern 3

The **third pattern** consists in inserting the task before or after any combination of elements of any node of the graph.



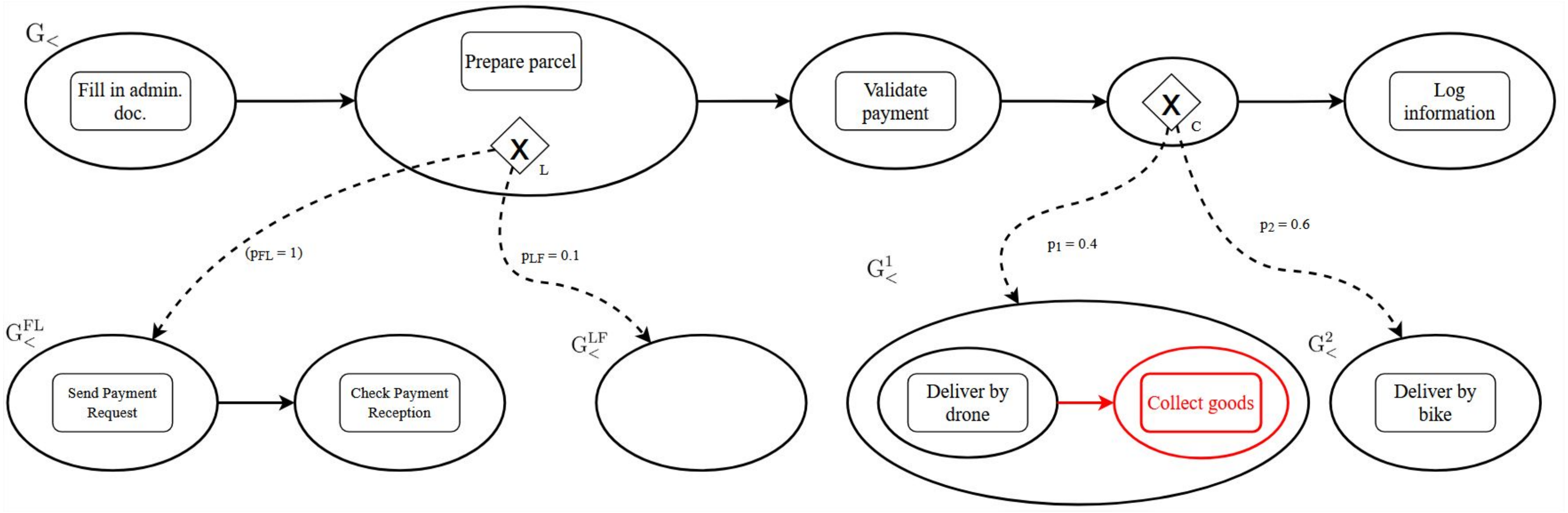
Step 2 – Pattern 3

The **third pattern** consists in inserting the task before or after any combination of elements of any node of the graph.



Step 2 – Pattern 3

The **third pattern** consists in inserting the task before or after any combination of elements of any node of the graph.



Step 2 – Pattern 4

The **fourth pattern** consists in inserting the task inside some **choice structures**.

Step 2 – Pattern 4

The **fourth pattern** consists in inserting the task inside some **choice structures**.

However, we said earlier that **choice structures must be preserved** to **guarantee** the structural **semantics** of the process.

Step 2 – Pattern 4

The **fourth pattern** consists in inserting the task inside some **choice structures**.

However, we said earlier that **choice structures must be preserved** to **guarantee** the structural **semantics** of the process.

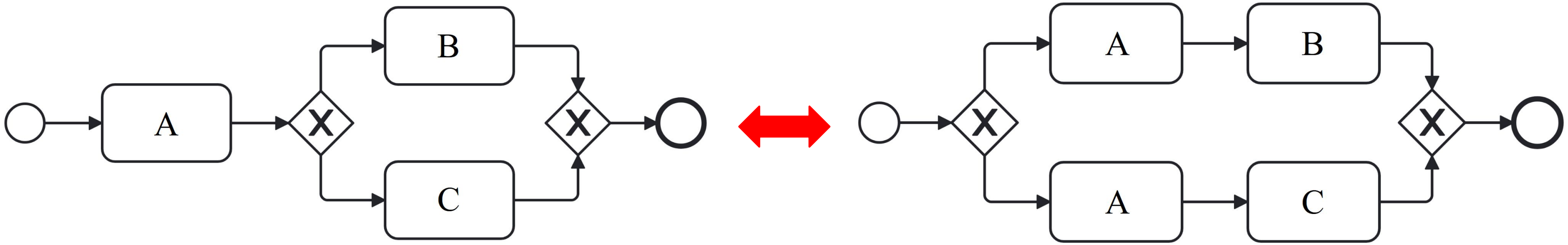
Nonetheless, there is a way to **modify a choice** structure while **preserving its semantics**:

Step 2 – Pattern 4

The **fourth pattern** consists in inserting the task inside some **choice structures**.

However, we said earlier that **choice structures must be preserved** to **guarantee** the structural **semantics** of the process.

Nonetheless, there is a way to **modify a choice** structure while **preserving its semantics**:



Step 2 – Pattern 4

In our context, Pattern 4 **recursively applies Patterns 1–4** to all the sequence graphs composing (~ branches of) the choice.

Step 2 – Pattern 4

In our context, Pattern 4 **recursively applies Patterns 1–4** to all the sequence graphs composing (~ branches of) the choice.

This creates **several new sequence graphs** for each branch of the choice.

Step 2 – Pattern 4

In our context, Pattern 4 **recursively applies Patterns 1–4** to all the sequence graphs composing (~ branches of) the choice.

This creates **several new sequence graphs** for each branch of the choice.

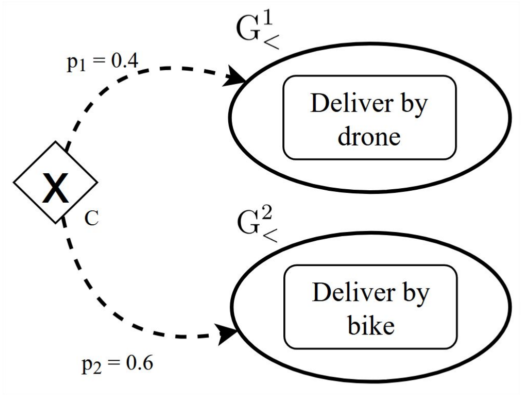
The **cartesian product** of the generated branches is then computed, which creates **several unique new** versions of the **choice**.

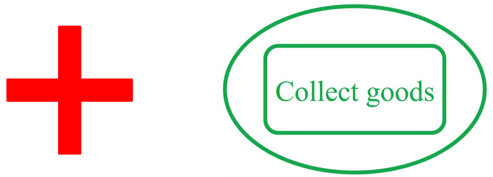
Step 2 – Pattern 4

In our context, Pattern 4 **recursively applies Patterns 1–4** to all the sequence graphs composing (~ branches of) the choice.

This creates **several new sequence graphs** for each branch of the choice.

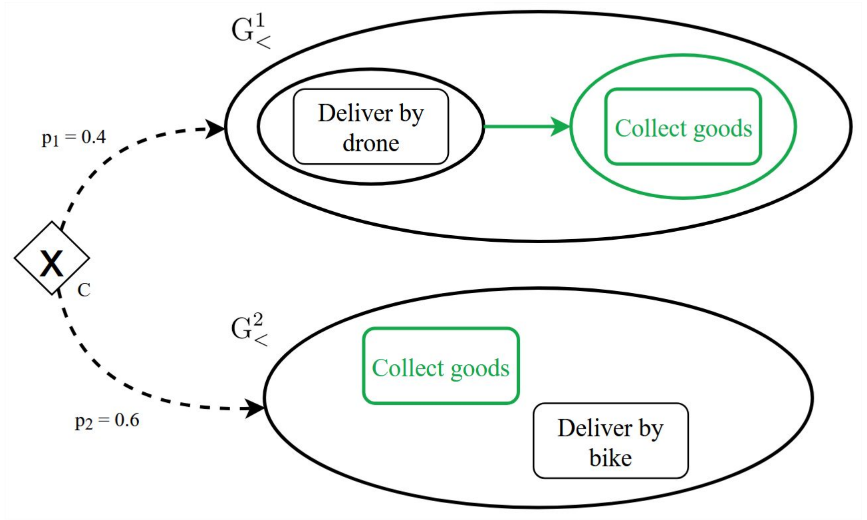
The **cartesian product** of the generated branches is then computed, which creates **several unique new** versions of the **choice**.

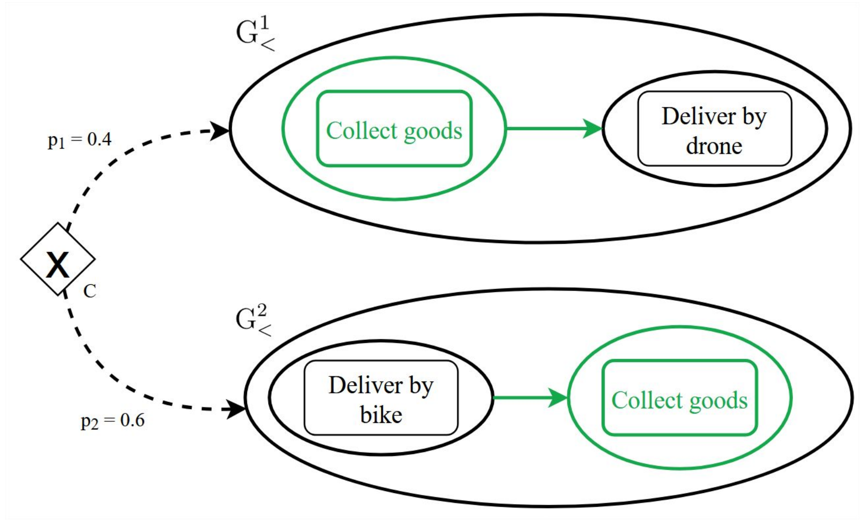




Step 2 – Pattern 4 – Example

This results in our case in **several possible choice structures**, two of them being illustrated below:





Step 3 – Comparison

When all the **selected processes** have been **mutated**, they have to be **compared**.

Step 3 – Comparison

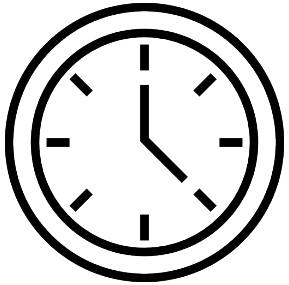
When all the **selected processes** have been **mutated**, they have to be **compared**.

This comparison is **based on metrics**, which are, in this approach:

Step 3 – Comparison

When all the **selected processes** have been **mutated**, they have to be **compared**.

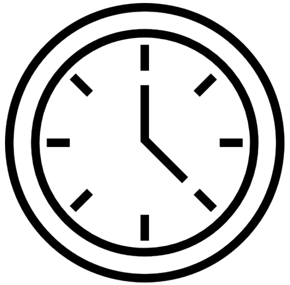
This comparison is **based on metrics**, which are, in this approach:



Step 3 – Comparison

When all the **selected processes** have been **mutated**, they have to be **compared**.

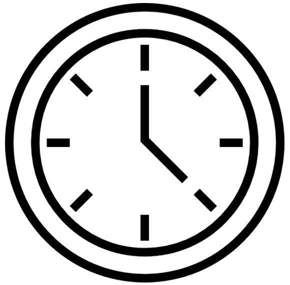
This comparison is **based on metrics**, which are, in this approach:



Step 3 – Comparison

When all the **selected processes** have been **mutated**, they have to be **compared**.

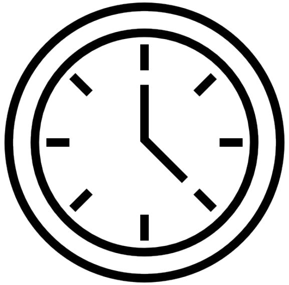
This comparison is **based on metrics**, which are, in this approach:



Step 3 – Comparison

When all the **selected processes** have been **mutated**, they have to be **compared**.

This comparison is **based on metrics**, which are, in this approach:



These metrics are obtained by **simulating** the process.

Step 3 – Comparison

The **metrics** are then **aggregated** in the form of a **score**, by **normalising** and **weighting** all the criteria to give them the **appropriate importance**.

Step 3 – Comparison

The **metrics** are then **aggregated** in the form of a **score**, by **normalising** and **weighting** all the criteria to give them the **appropriate importance**.

The **best score** thus **indicates** which **process** is the **most optimised** one.

$$\sum_{c \in C} \omega_c \times \frac{c(G_0)}{c(G_1)} > \sum_{c \in C} \omega_c \times \frac{c(G_0)}{c(G_2)}$$

Step 3 – Comparison

The **metrics** are then **aggregated** in the form of a **score**, by **normalising** and **weighting** all the criteria to give them the **appropriate importance**.

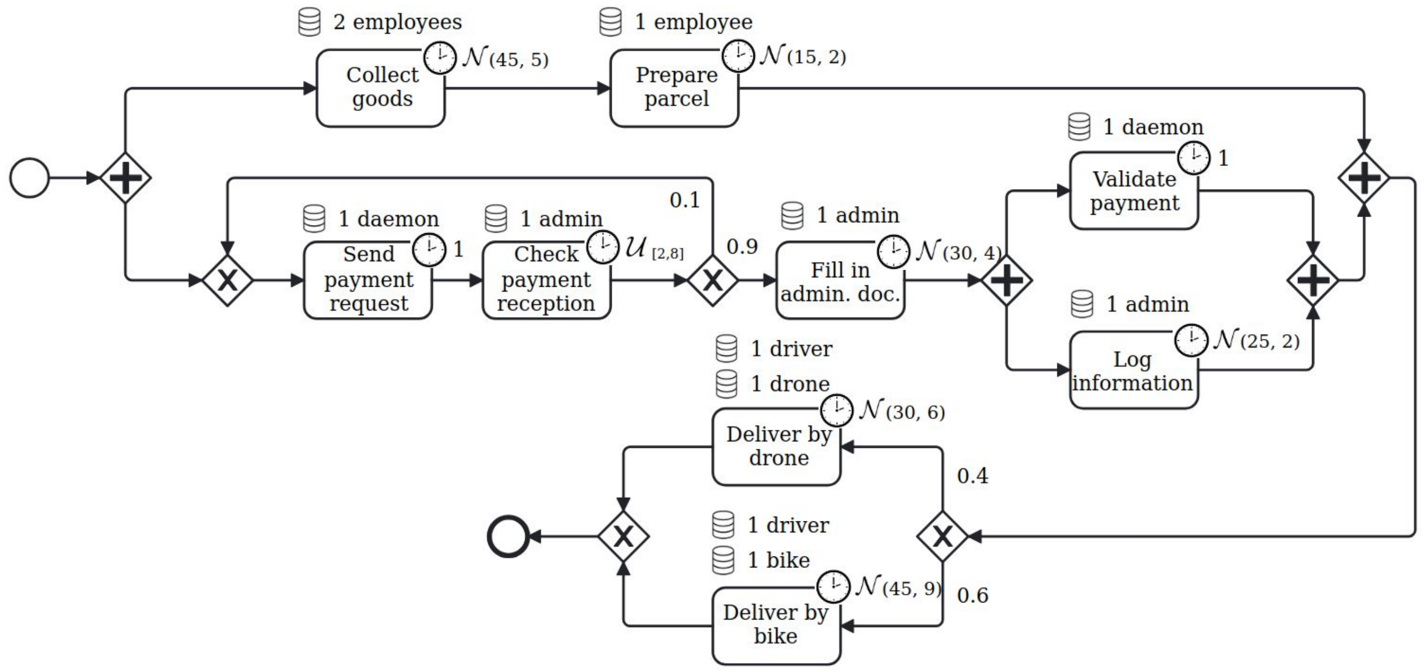
The **best score** thus **indicates** which **process** is the **most optimised** one.

$$\sum_{c \in C} \omega_c \times \frac{c(G_0)}{c(G_1)} > \sum_{c \in C} \omega_c \times \frac{c(G_0)}{c(G_2)}$$

The **decision** of **keeping** or **discarding** a process is taken at the **discretion** of the **MOEA** being used, and can be **counter-intuitive** (for instance, a worse process can be kept and a better one discarded).

Step 4 – Termination

When the **MOEA reaches its bound** (duration, number of iterations, manual stop, etc.), it returns an **optimised version** of the **original process**.



Tool Support

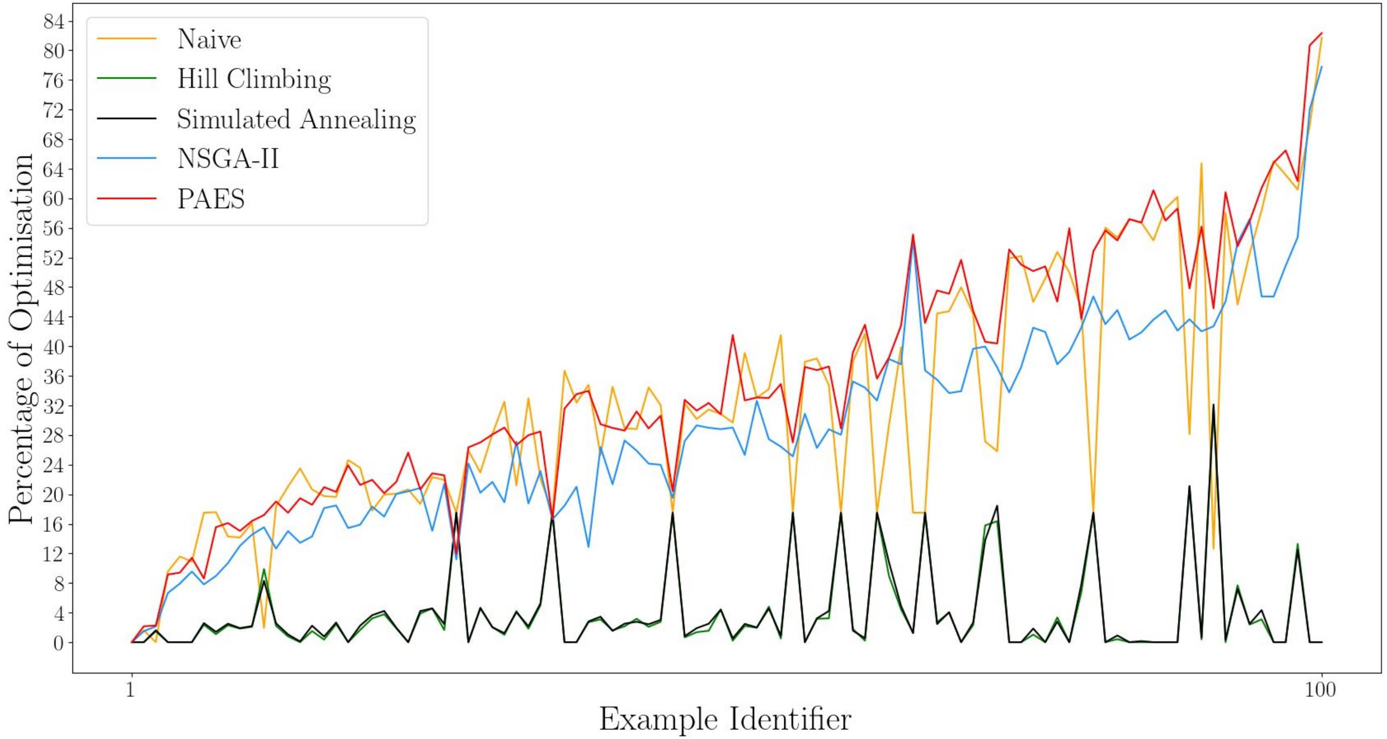
The approach has been **fully implemented** and consists of approximately **15k lines of Java code**.

It makes use of the **jMetal framework** [DN2011] which **implements** dozens of well-known **MOEAs** and provides **facilities** regarding their **utilisation** in various **contexts**.

This tool was used as a **support** for the **experimentations**.

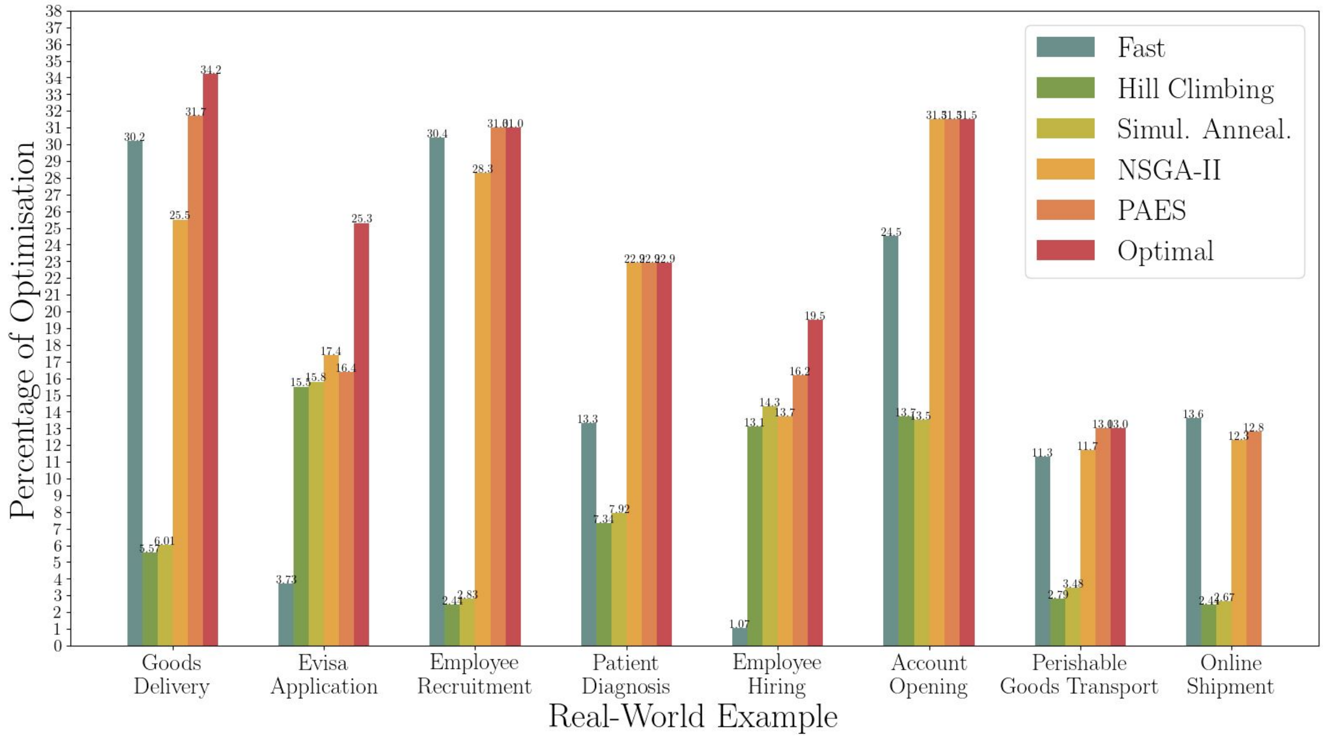
Experiments – Handcrafted Examples

The **first part** of these experiments consisted in **comparing** several **algorithms** on **100 handcrafted** examples.



Experiments – Real-World Examples

The **second part** of these experiments consisted in **comparing** these **algorithms** on **8 real-world** examples coming from the literature.



Conclusion

We proposed a **technique** aiming at **optimising BPMN** processes enriched with **durations**, **resources**, **costs**, and **executed multiple times**.

Conclusion

We proposed a **technique** aiming at **optimising BPMN** processes enriched with **durations**, **resources**, **costs**, and **executed multiple times**.

The proposed approach tackles the complex problem of **multi-objectives optimisation** while ensuring the **preservation of the semantics** of the process, and of its **behaviour**, by the meaning of **user-defined dependencies**.

Conclusion

We proposed a **technique** aiming at **optimising BPMN** processes enriched with **durations**, **resources**, **costs**, and **executed multiple times**.

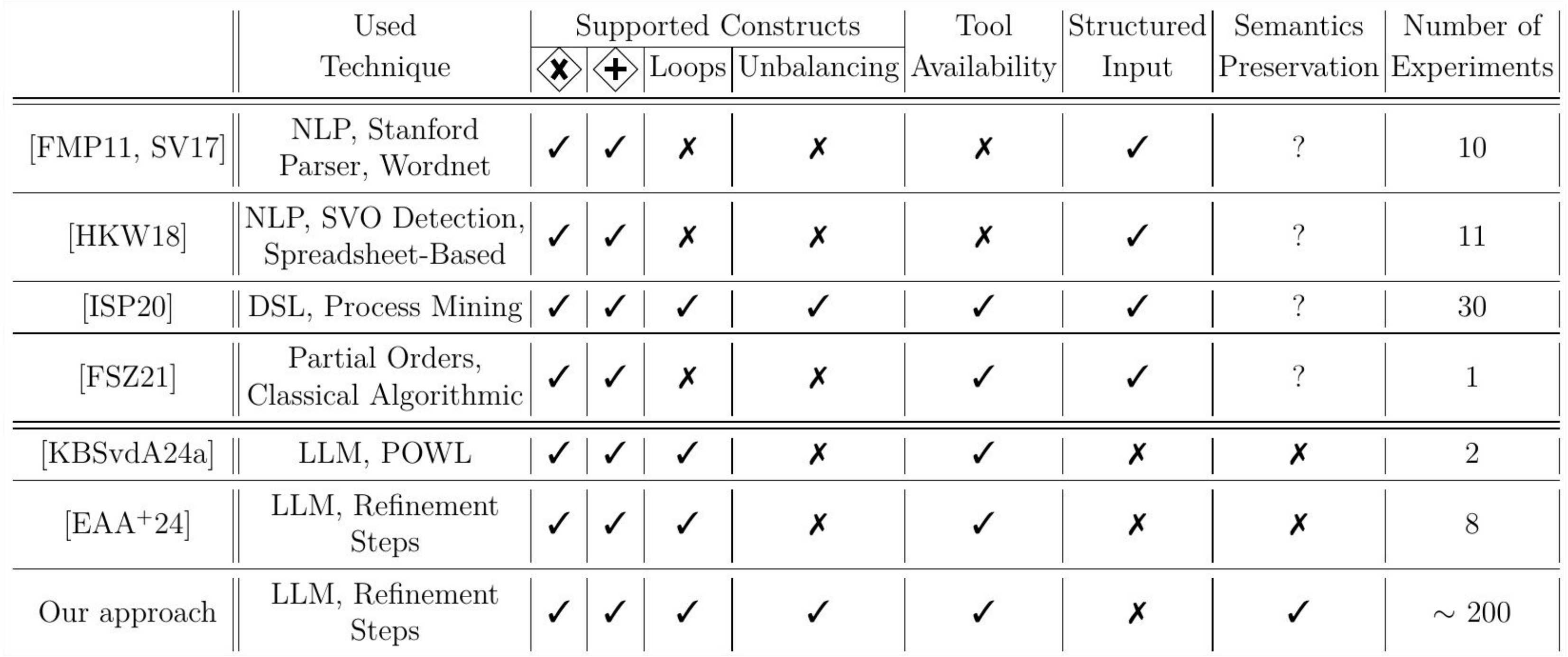
The proposed approach tackles the complex problem of **multi-objectives optimisation** while ensuring the **preservation of the semantics** of the process, and of its **behaviour**, by the meaning of **user-defined dependencies**.

The presented approach has been **fully implemented** and **validated** by a tool written in Java on a basis containing **more than 100 examples**.

Plan

Related Work – Modelling

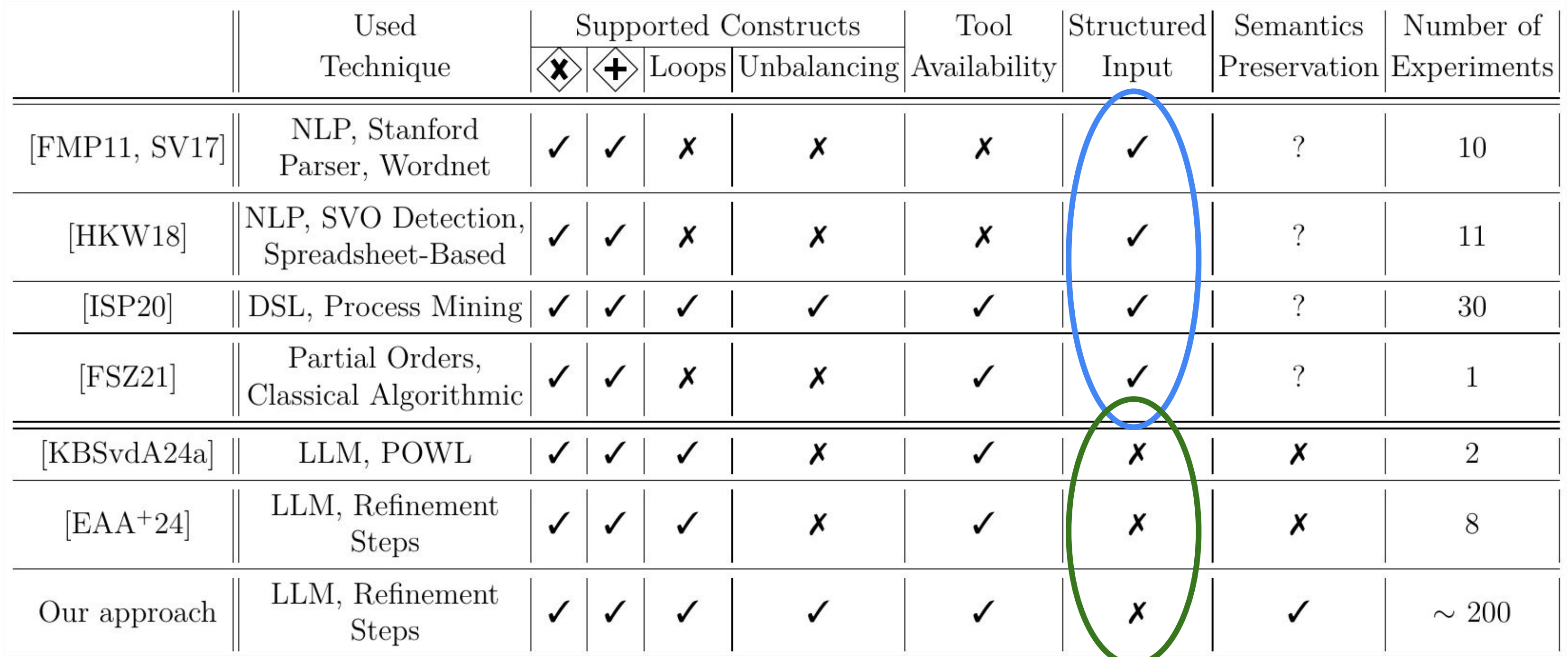
The **question of modelling** business processes has been a **topic of interest** since years, and **many approaches** tended to give it an answer.



	Used Technique	Supported Constructs				Tool Availability	Structured Input	Semantics Preservation	Number of Experiments
		⊗	⊕	Loops	Unbalancing				
[FMP11, SV17]	NLP, Stanford Parser, Wordnet	✓	✓	✗	✗	✗	✓	?	10
[HKW18]	NLP, SVO Detection, Spreadsheet-Based	✓	✓	✗	✗	✗	✓	?	11
[ISP20]	DSL, Process Mining	✓	✓	✓	✓	✓	✓	?	30
[FSZ21]	Partial Orders, Classical Algorithmic	✓	✓	✗	✗	✓	✓	?	1
[KBSvdA24a]	LLM, POWL	✓	✓	✓	✗	✓	✗	✗	2
[EAA$^+$24]	LLM, Refinement Steps	✓	✓	✓	✗	✓	✗	✗	8
Our approach	LLM, Refinement Steps	✓	✓	✓	✓	✓	✗	✓	~ 200

Related Work – Modelling

The **question of modelling** business processes has been a **topic of interest** since years, and **many approaches** tended to give it an answer.



	Used Technique	Supported Constructs				Tool Availability	Structured Input	Semantics Preservation	Number of Experiments
		⊗	⊕	Loops	Unbalancing				
[FMP11, SV17]	NLP, Stanford Parser, Wordnet	✓	✓	✗	✗	✗	✓	?	10
[HKW18]	NLP, SVO Detection, Spreadsheet-Based	✓	✓	✗	✗	✗	✓	?	11
[ISP20]	DSL, Process Mining	✓	✓	✓	✓	✓	✓	?	30
[FSZ21]	Partial Orders, Classical Algorithmic	✓	✓	✗	✗	✓	✓	?	1
[KBSvdA24a]	LLM, POWL	✓	✓	✓	✗	✓	✗	✗	2
[EAA+24]	LLM, Refinement Steps	✓	✓	✓	✗	✓	✗	✗	8
Our approach	LLM, Refinement Steps	✓	✓	✓	✓	✓	✗	✓	∼ 200

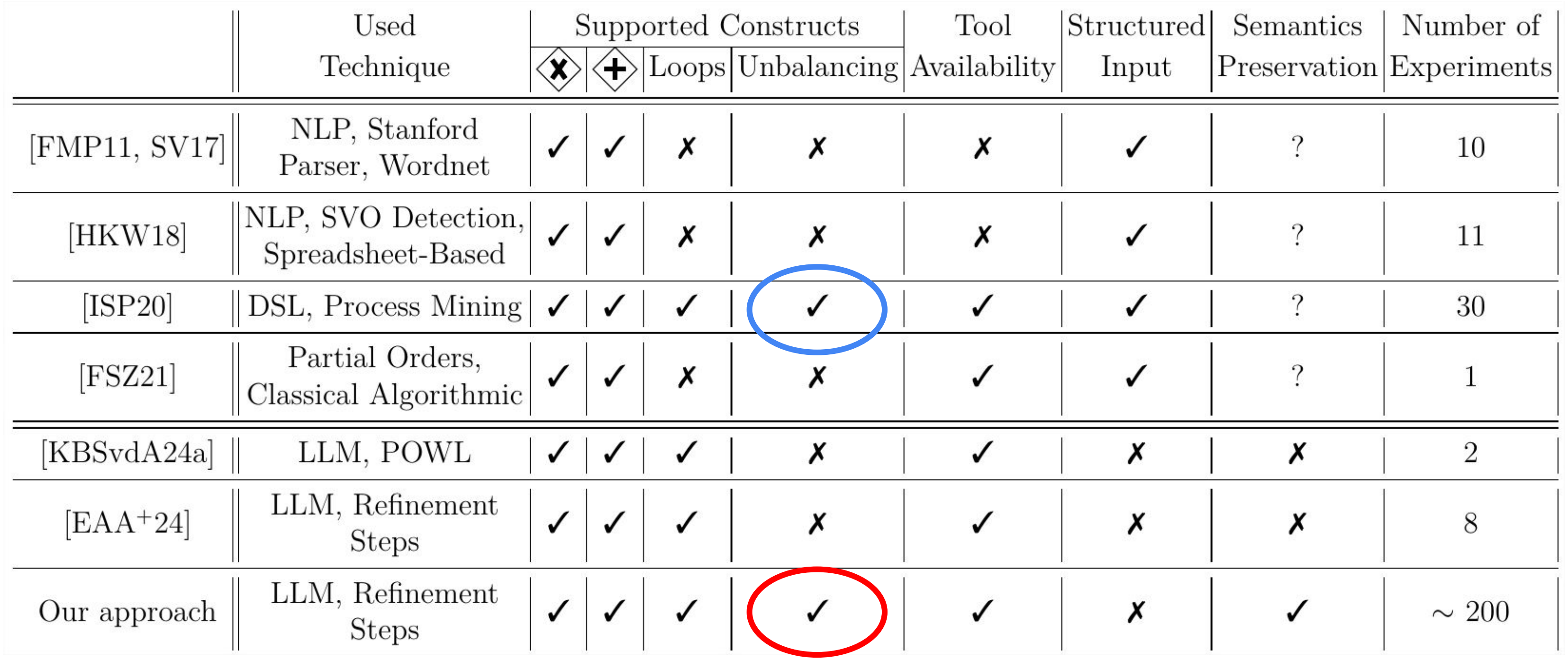
Related Work – Modelling

The **question of modelling** business processes has been a **topic of interest** since years, and **many approaches** tended to give it an answer.

	Used Technique	Supported Constructs				Tool Availability	Structured Input	Semantics Preservation	Number of Experiments
		⊗	⊕	Loops	Unbalancing				
[FMP11, SV17]	NLP, Stanford Parser, Wordnet	✓	✓	✗	✗	✗	✓	?	10
[HKW18]	NLP, SVO Detection, Spreadsheet-Based	✓	✓	✗	✗	✗	✓	?	11
[ISP20]	DSL, Process Mining	✓	✓	✓	✓	✓	✓	?	30
[FSZ21]	Partial Orders, Classical Algorithmic	✓	✓	✗	✗	✓	✓	?	1
[KBSvdA24a]	LLM, POWL	✓	✓	✓	✗	✓	✗	✗	2
[EAA+24]	LLM, Refinement Steps	✓	✓	✓	✗	✓	✗	✗	8
Our approach	LLM, Refinement Steps	✓	✓	✓	✓	✓	✗	✓	~ 200

Related Work – Modelling

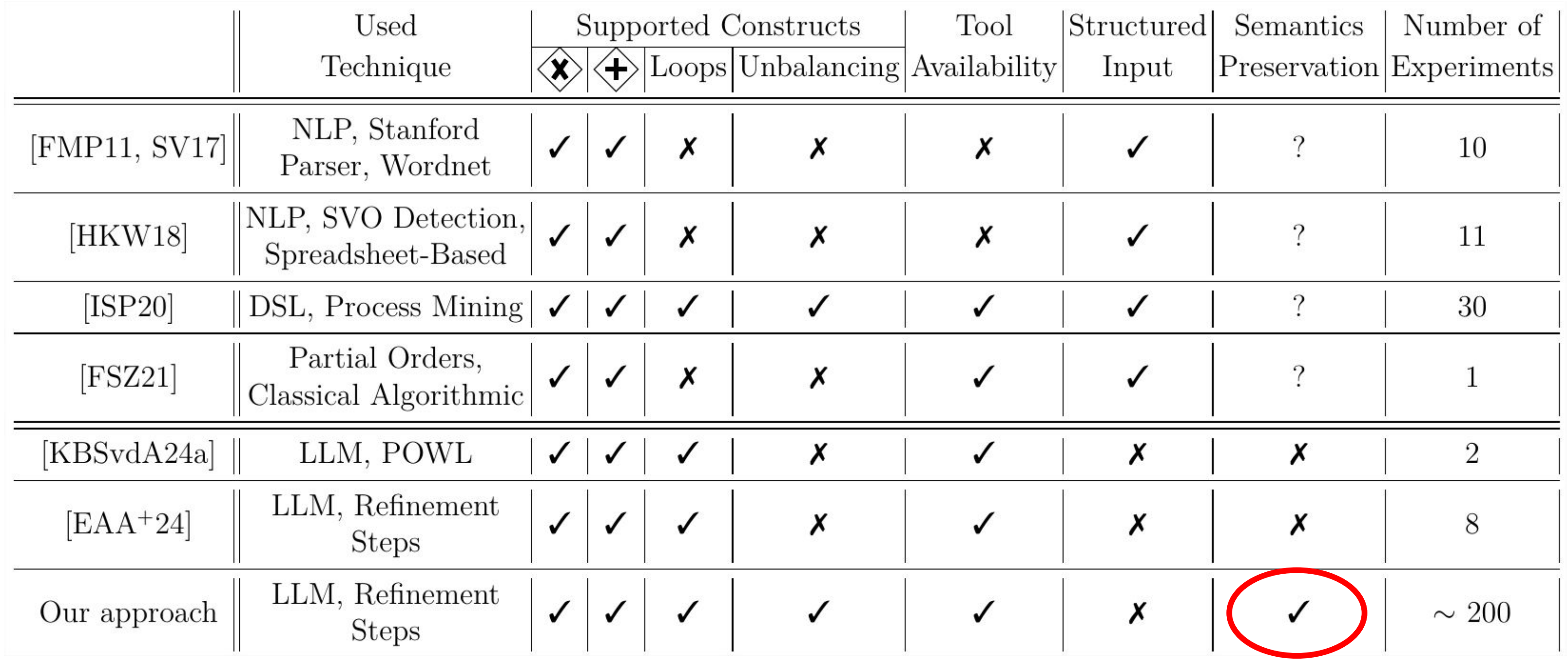
The **question of modelling** business processes has been a **topic of interest** since years, and **many approaches** tended to give it an answer.



	Used Technique	Supported Constructs				Tool Availability	Structured Input	Semantics Preservation	Number of Experiments
		⊗	⊕	Loops	Unbalancing				
[FMP11, SV17]	NLP, Stanford Parser, Wordnet	✓	✓	✗	✗	✗	✓	?	10
[HKW18]	NLP, SVO Detection, Spreadsheet-Based	✓	✓	✗	✗	✗	✓	?	11
[ISP20]	DSL, Process Mining	✓	✓	✓	✓	✓	✓	?	30
[FSZ21]	Partial Orders, Classical Algorithmic	✓	✓	✗	✗	✓	✓	?	1
[KBSvdA24a]	LLM, POWL	✓	✓	✓	✗	✓	✗	✗	2
[EAA^{+}24]	LLM, Refinement Steps	✓	✓	✓	✗	✓	✗	✗	8
Our approach	LLM, Refinement Steps	✓	✓	✓	✓	✓	✗	✓	$\sim$ 200

Related Work – Modelling

The **question of modelling** business processes has been a **topic of interest** since years, and **many approaches** tended to give it an answer.



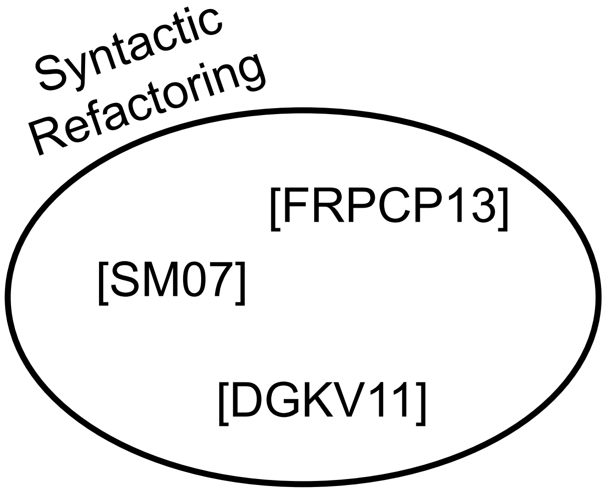
	Used Technique	Supported Constructs				Tool Availability	Structured Input	Semantics Preservation	Number of Experiments
		✖	✚	Loops	Unbalancing				
[FMP11, SV17]	NLP, Stanford Parser, Wordnet	✓	✓	✗	✗	✗	✓	?	10
[HKW18]	NLP, SVO Detection, Spreadsheet-Based	✓	✓	✗	✗	✗	✓	?	11
[ISP20]	DSL, Process Mining	✓	✓	✓	✓	✓	✓	?	30
[FSZ21]	Partial Orders, Classical Algorithmic	✓	✓	✗	✗	✓	✓	?	1
[KBSvdA24a]	LLM, POWL	✓	✓	✓	✗	✓	✗	✗	2
[EAA+24]	LLM, Refinement Steps	✓	✓	✓	✗	✓	✗	✗	8
Our approach	LLM, Refinement Steps	✓	✓	✓	✓	✓	✗	✓	~ 200

Related Work – Refactoring

Refactoring as an optimisation technique is less classical, thus there is no strictly identical approaches in the literature.

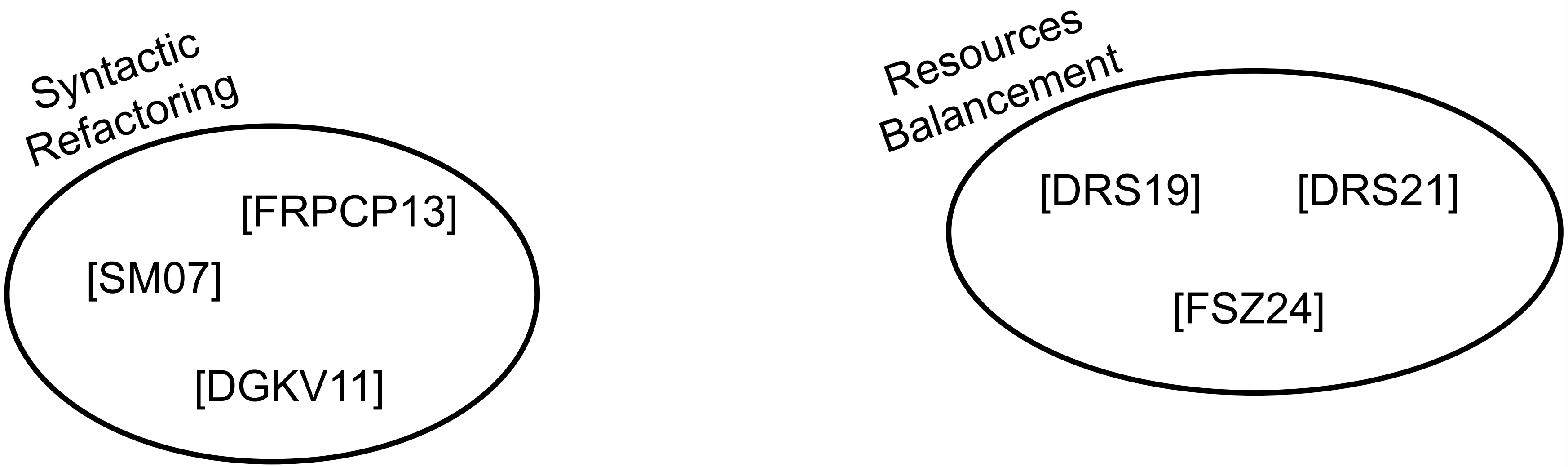
Related Work – Refactoring

Refactoring as an **optimisation** technique is **less classical**, thus there is **no** strictly identical **approaches** in the **literature**.



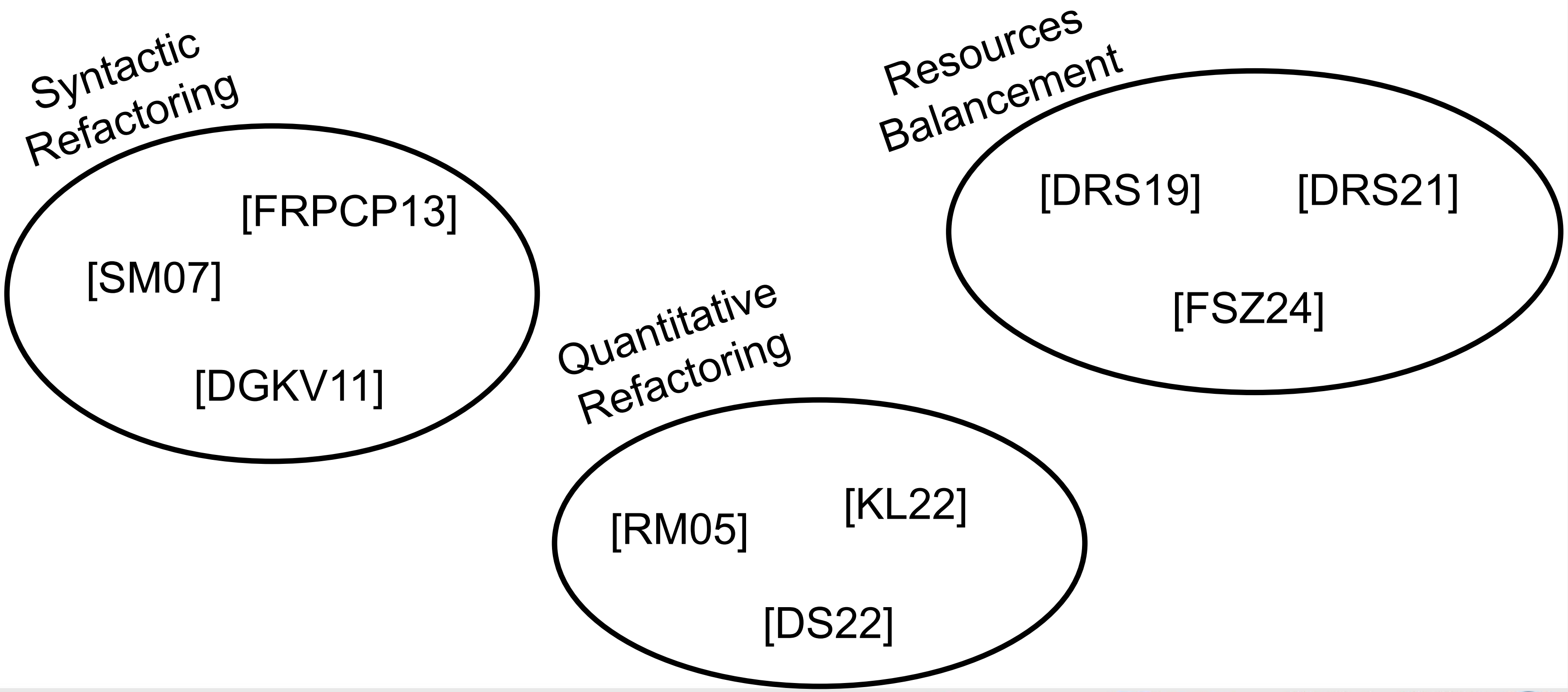
Related Work – Refactoring

Refactoring as an **optimisation** technique is **less classical**, thus there is **no** strictly identical **approaches** in the **literature**.



Related Work – Refactoring

Refactoring as an **optimisation** technique is **less classical**, thus there is **no** strictly identical **approaches** in the **literature**.



Plan

General Conclusion

In this **thesis**, we have proposed to dive into **two important topics** of business process management: **modelling** and **optimisation**.

General Conclusion

In this **thesis**, we have proposed to dive into **two important topics** of business process management: **modelling** and **optimisation**.

We proposed an approach to **formally model BPMN processes** from their textual representation, and **several approaches aiming at optimising** such processes.

General Conclusion

In this **thesis**, we have proposed to dive into **two important topics** of business process management: **modelling** and **optimisation**.

We proposed an approach to **formally model BPMN processes** from their textual representation, and **several approaches aiming at optimising** such processes.

Perspectives on modelling:

- ➢ **Cross-checking** the generated expressions **with** other **LLMs**;
- ➢ **Enlarging** the supported BPMN **syntax**;
- ➢ **Enlarging** and **improving** the **datasets** used for fine-tuning.

General Conclusion

In this **thesis**, we have proposed to dive into **two important topics** of business process management: **modelling** and **optimisation**.

We proposed an approach to **formally model BPMN processes** from their textual representation, and **several approaches aiming at optimising** such processes.

Perspectives on modelling:

- ➢ **Cross-checking** the generated expressions **with** other **LLMs**;
- ➢ **Enlarging** the supported BPMN **syntax**;
- ➢ **Enlarging** and **improving** the **datasets** used for fine-tuning.

Perspectives on refactoring:

- ➢ Getting **rid of sequence graphs** structures;
- ➢ **Removing** or **limiting** the use of **simulation**;
- ➢ Finding **better optimisation algorithms**.

References

- [BKO2010]: *An Empirical Comparison of the Usability of BPMN and UML Activity Diagrams for Business Users*, Dominik Q. Birkmeier, Sebastian Klöckner, and Sven Overhage, 2010.

- [BRJ2000]: *The UML User Guide*, Grady Booch, James Rumbaugh, and Ivar Jacobson, 2000.

- [Davenport1993]: *Process innovation: reengineering work through information technology*, Thomas Hayes Davenport, 1993.

- [DN2011]: *jMetal: A Java Framework for Multi-objective Optimization*, Juan José Durillon, Antonio Jesus Nebro, 2011.

- [Geambasu2012]: *BPMN vs. UML Activity Diagram for Business Process Modeling*, Cristina Venera Geambasu, 2012.

References

- ➢ [HC1993]: *Reengineering the Corporation: A Manifesto for Business Revolution*, Michael Hammer and James Champy, 1993.
- ➢ [Jackson2020]: *Understanding understanding and ambiguity in natural language*, Philip Jackson, 2020.
- ➢ [JMP1993]: *Business Process Reengineering: BreakPoint Strategies for Market Dominance*, Henry J. Johansson, Patrick McHugh, and A. John Pendlebury, 1993.
- ➢ [Lin1996]: *On the Structural Complexity of Natural Language Sentences*, Dekang Lin, 1996.
- ➢ [NK2006]: *Assessing Business Process Modeling Languages Using a Generic Quality Framework*, Anna Gunhild Nysetvold and John Krogstie, 2006.

References

➢ [OMG2011]: *Business Process Model and Notation (BPMN)*, OMG, 2011.

➢ [RB1990]: *Improving Performance: How to Manage the White Space on the Organization Chart*, Geary A. Rummler and Alan P. Brache, 1990.

➢ [Smith1776]: *An Inquiry into the Nature and Causes of the Wealth of Nations*, Adam Smith, 1776.

➢ [Taylor1911]: *The Principles of Scientific Management*, Frederick Winslow Taylor, 1911.

➢ [Weske2007]: *Business Process Management: Concepts, Languages, Architectures*, Mathias Weske, 2007.

➢ [White2004]: *Process Modeling Notations and Workflow Patterns*, Stephen White, 2004.