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?

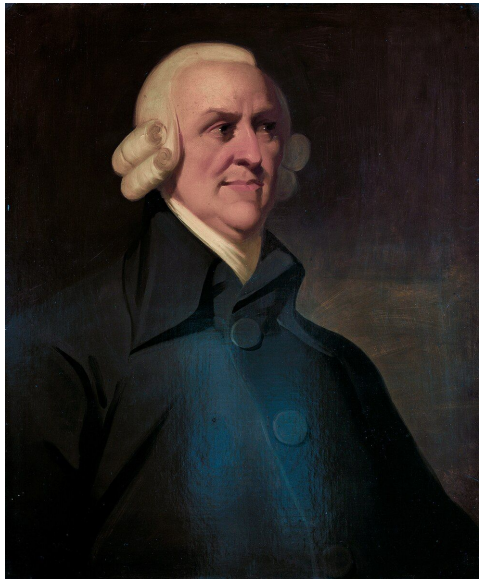
A Little Bit of History: BPMN

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

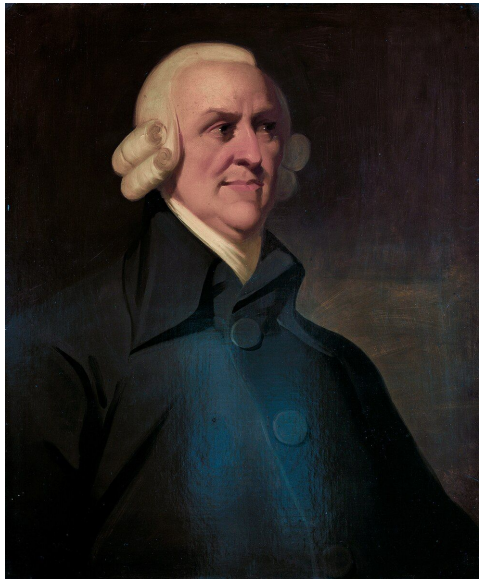
According to history, the **first** man to have ever evokated the term “business process” is the scottish economist Adam Smith in 1776.



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.”*

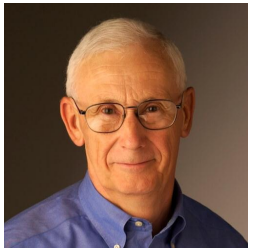
A Little Bit of History: …and His Successors

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

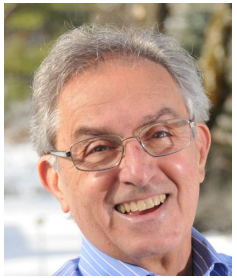
Frederick Winslow Taylor



Geary A. Rummler

Thomas H. Davenport

Michael Hammer



James Champy

Wil van der Aalst

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

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:

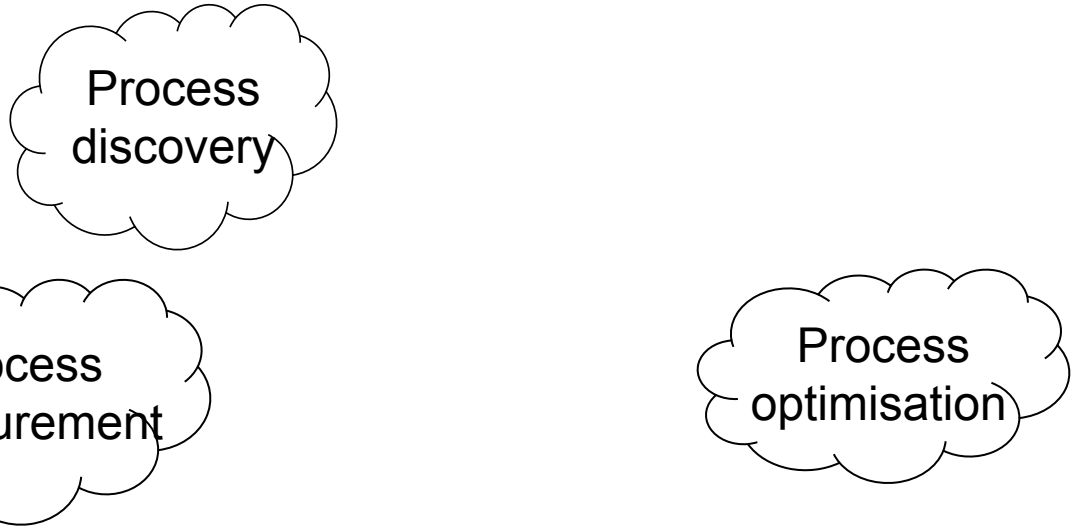
A Little Bit of History: Business Process Management

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:

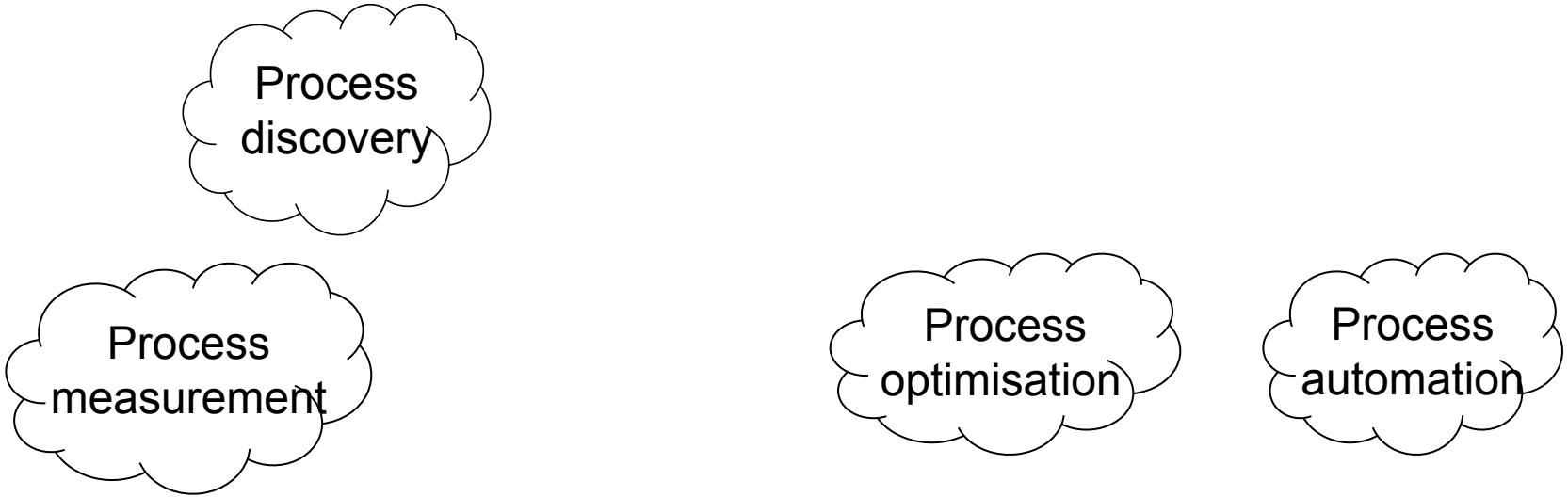
This desire to provide a **rigorous**, **unified** definition of business processes paved the way to the creation of a new discipline: the **business process management**.

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



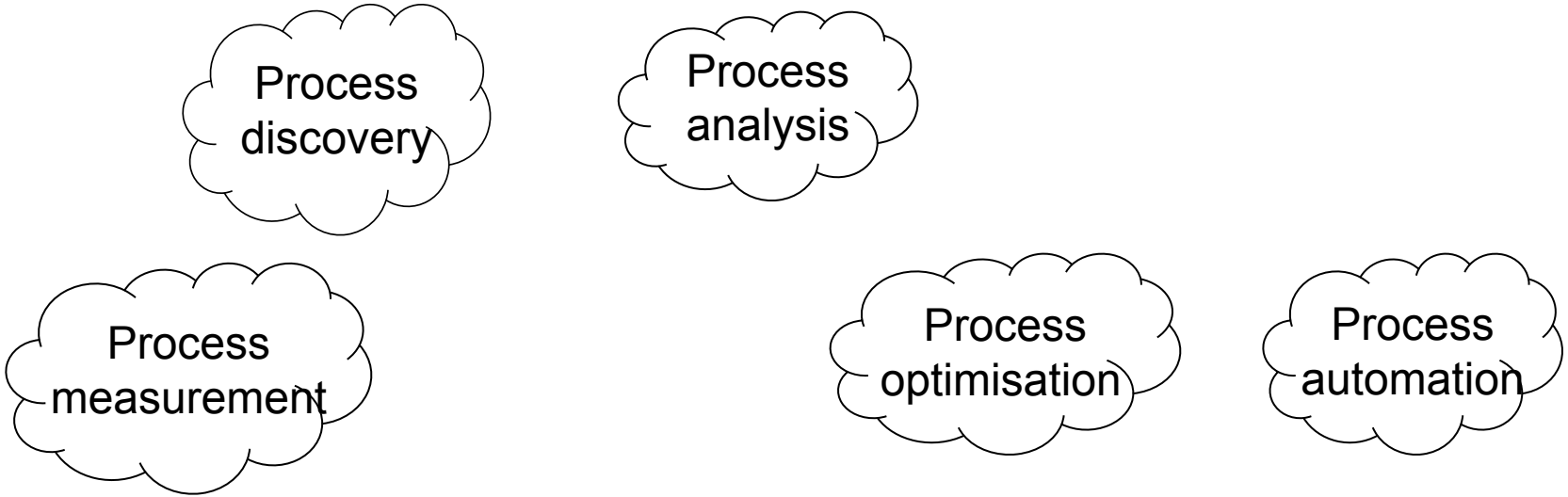
This desire to provide a **rigorous**, **unified** definition of business processes paved the way to the creation of a new discipline: the **business process management**.

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



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**.

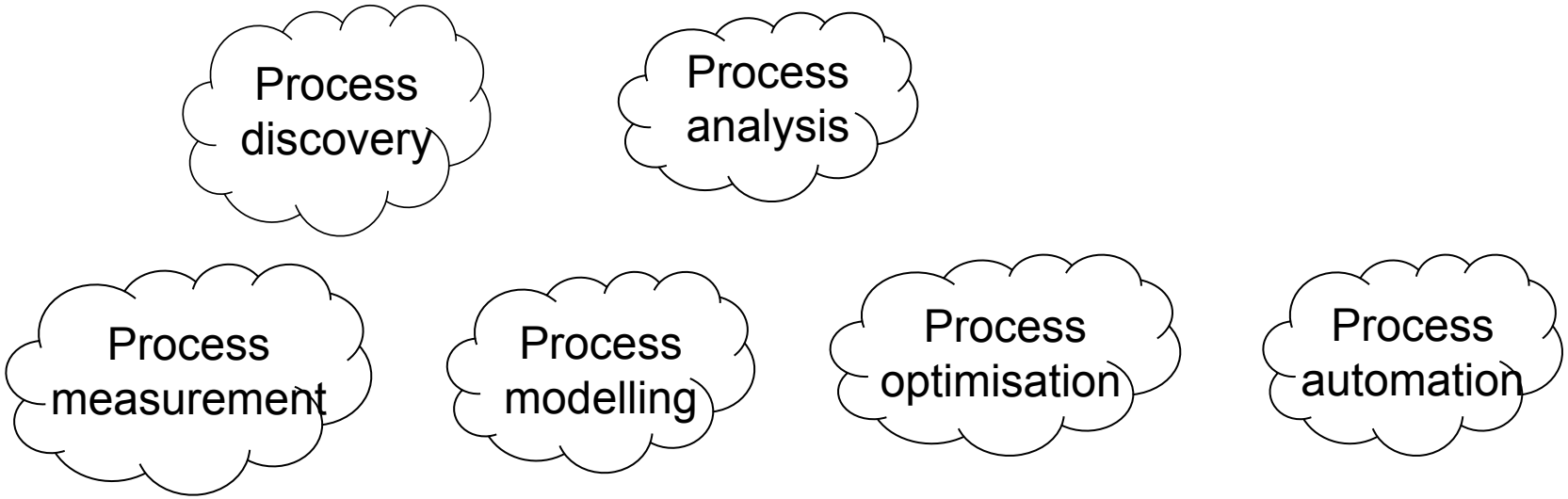
This **holistic discipline** encompasses all the fields related to business processes, such as:



A Little Bit of History: Business Process Management

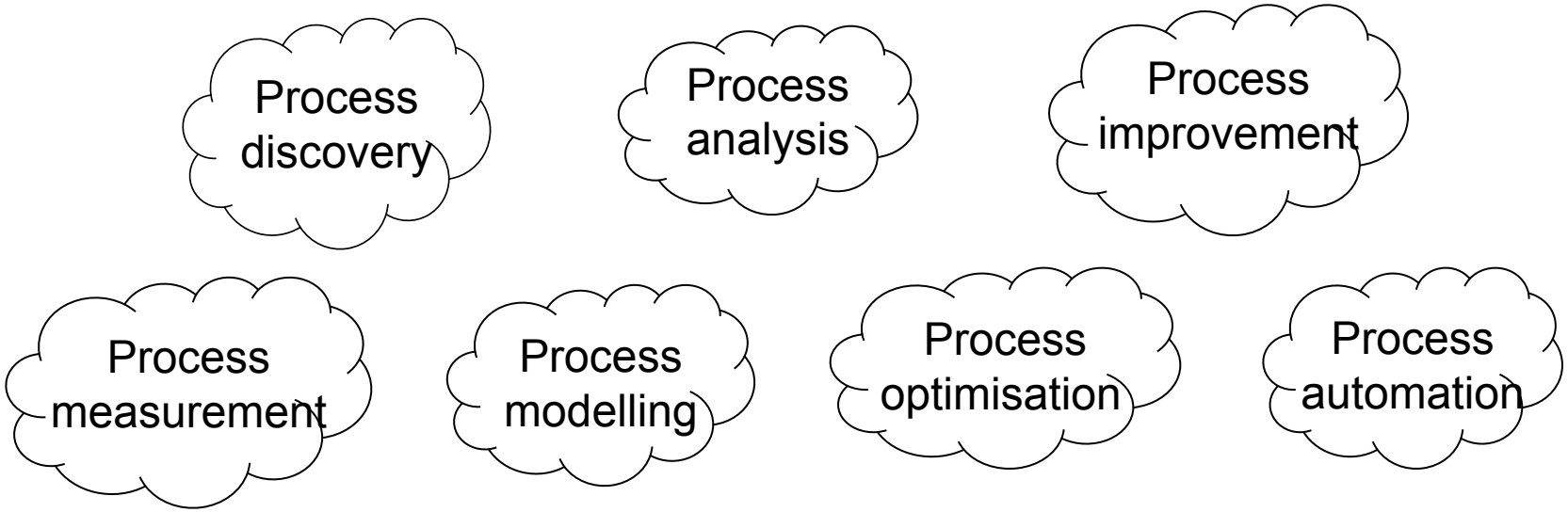
This desire to provide a **rigorous**, **unified** definition of business processes paved the way to the creation of a new discipline: the **business process management**.

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



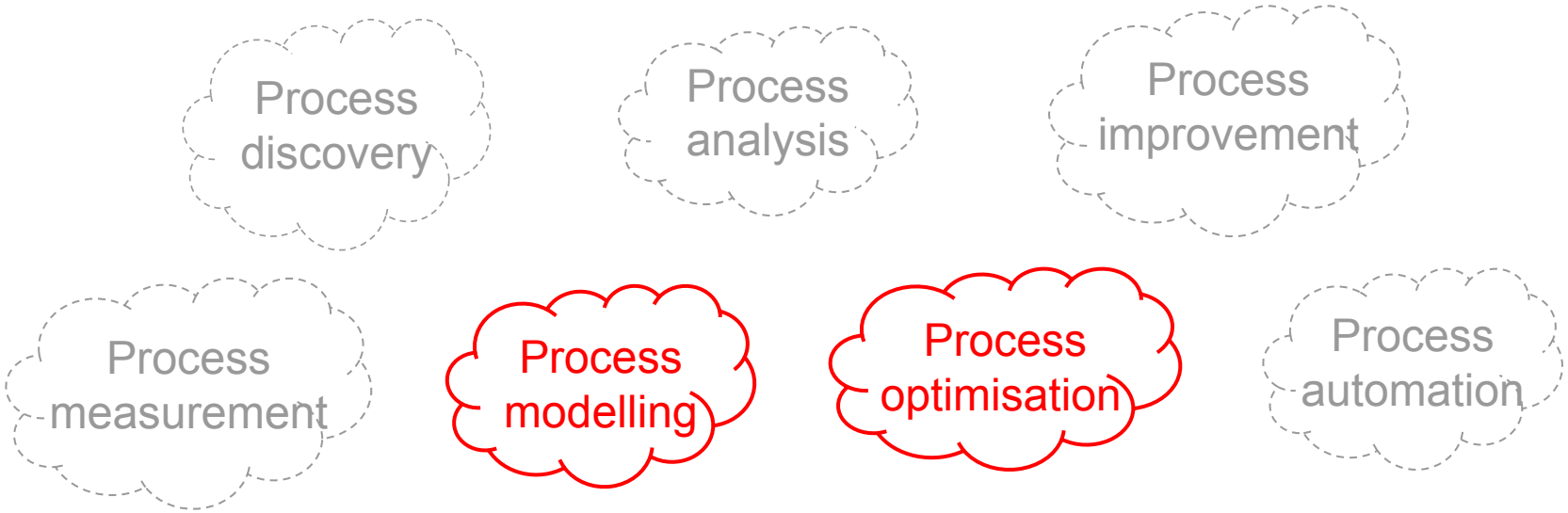
This desire to provide a **rigorous**, **unified** definition of business processes paved the way to the creation of a new discipline: the **business process management**.

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



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:

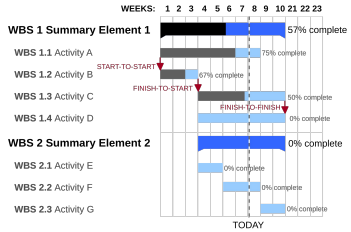


A Little Bit of History: How to Model a Process?

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?

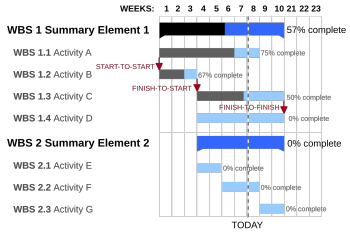
The term **business process modelling** was coined in the 1960s by Stanley Williams, but people were **interested in modelling** processes **years before**.



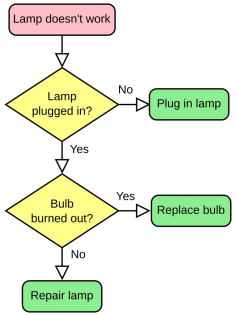
Gantt chart, 1910-15

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**.



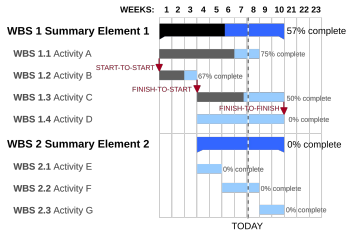
Gantt chart, 1910-15



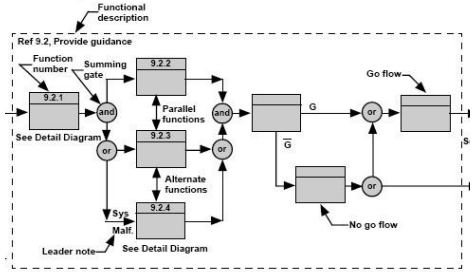
Flowchart, 1921

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

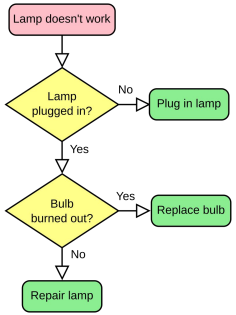
The term **business process modelling** was coined in the 1960s by Stanley Williams, but people were **interested in modelling** processes **years before**.



Gantt chart, 1910-15



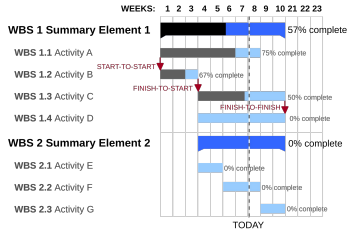
Functional flow block diagram (FFBD), 195X



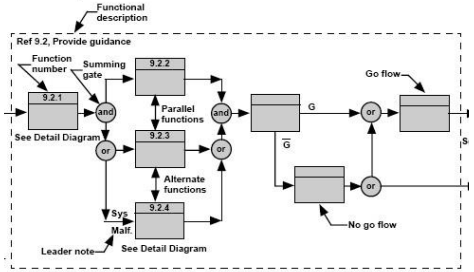
Flowchart, 1921

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

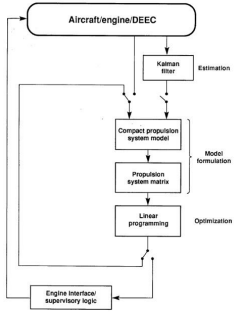
The term **business process modelling** was coined in the 1960s by Stanley Williams, but people were **interested in modelling** processes **years before**.



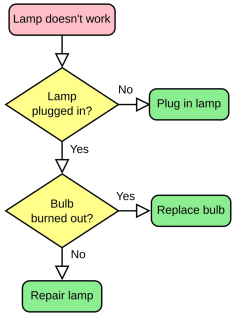
Gantt chart, 1910-15



Functional flow block diagram (FFBD), 195X



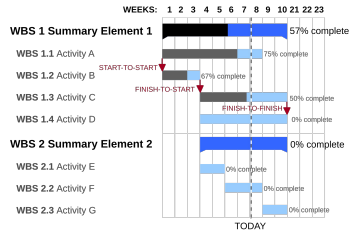
Control-flow diagram (CFD), 195X



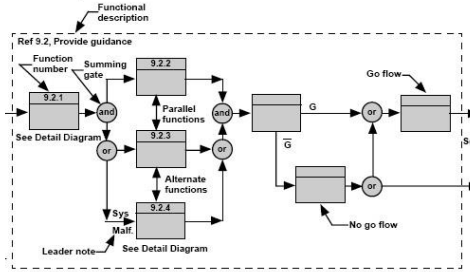
Flowchart, 1921

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

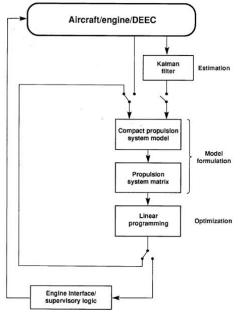
The term **business process modelling** was coined in the 1960s by Stanley Williams, but people were **interested in modelling** processes **years before**.



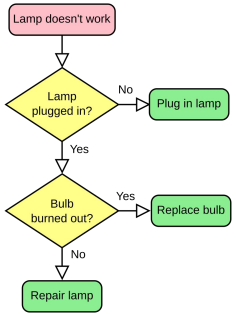
Gantt chart, 1910-15



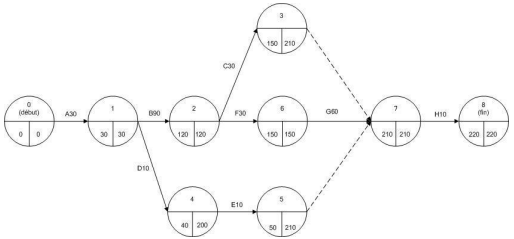
Functional flow block diagram (FFBD), 195X



Control-flow diagram (CFD), 195X



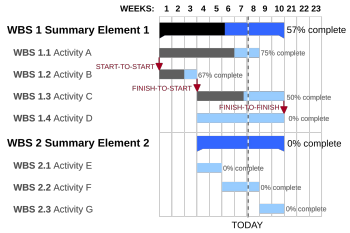
Flowchart, 1921



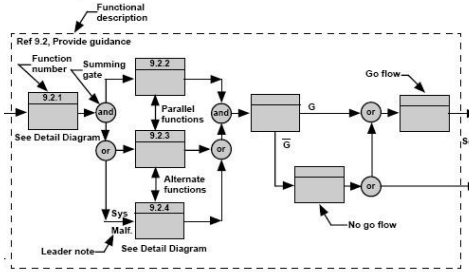
PERT diagram, 195X

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

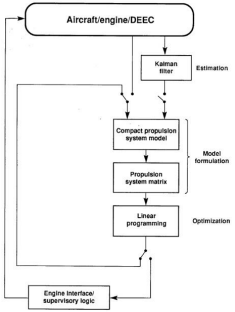
The term **business process modelling** was coined in the 1960s by Stanley Williams, but people were **interested in modelling** processes **years before**.



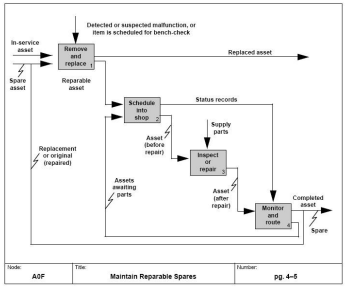
Gantt chart, 1910-15



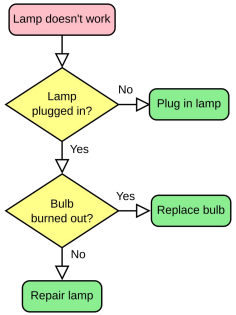
Functional flow block diagram (FFBD), 195X



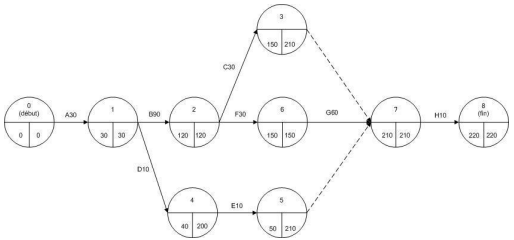
Control-flow diagram (CFD), 195X



IDEF diagram, 197X



Flowchart, 1921



PERT diagram, 195X

A Little Bit of History: BPMN

More recently, another notation, called **Business Process Management Notation (BPMN)** [OMG2011], emerged, and became **rapidly widely used** by companies and institutions.

A Little Bit of History: BPMN

More recently, another notation, called **Business Process Management Notation (BPMN)** [OMG2011], emerged, and became **rapidly widely used** by companies and institutions.

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?

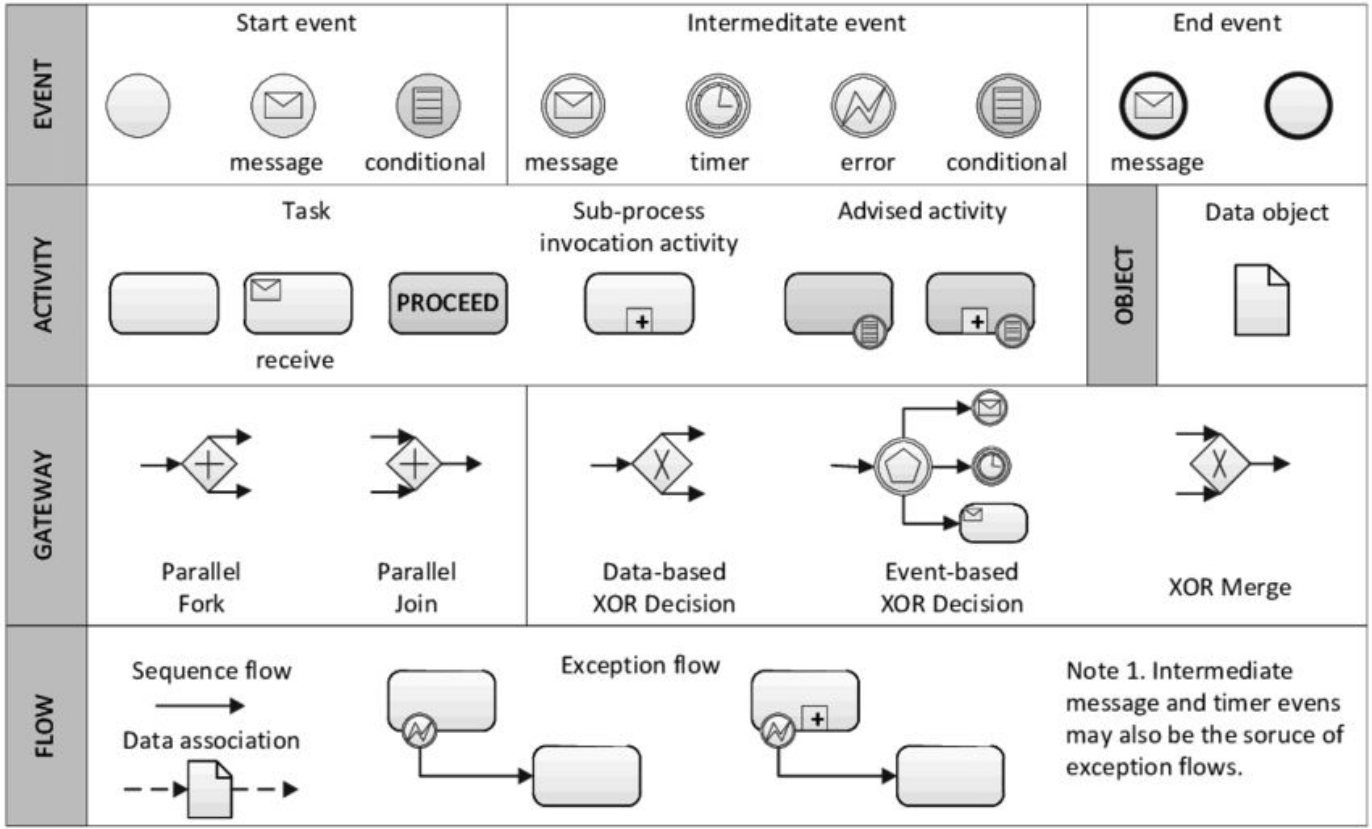
- A **workflow-based notation** created in 2004 by the Business Process Management Initiative (BPMI) and the Object Management Group (OMG).

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

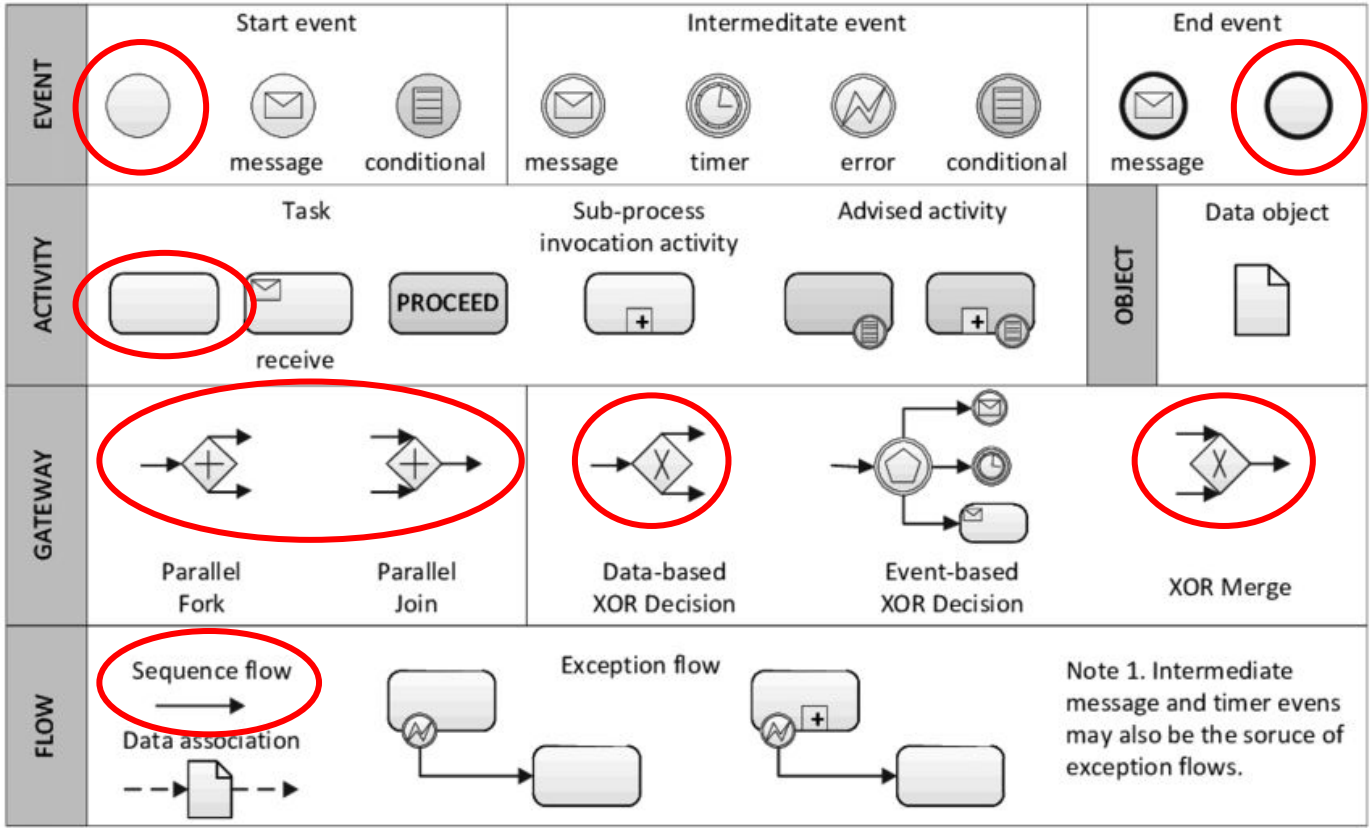
But what is BPMN?

- 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

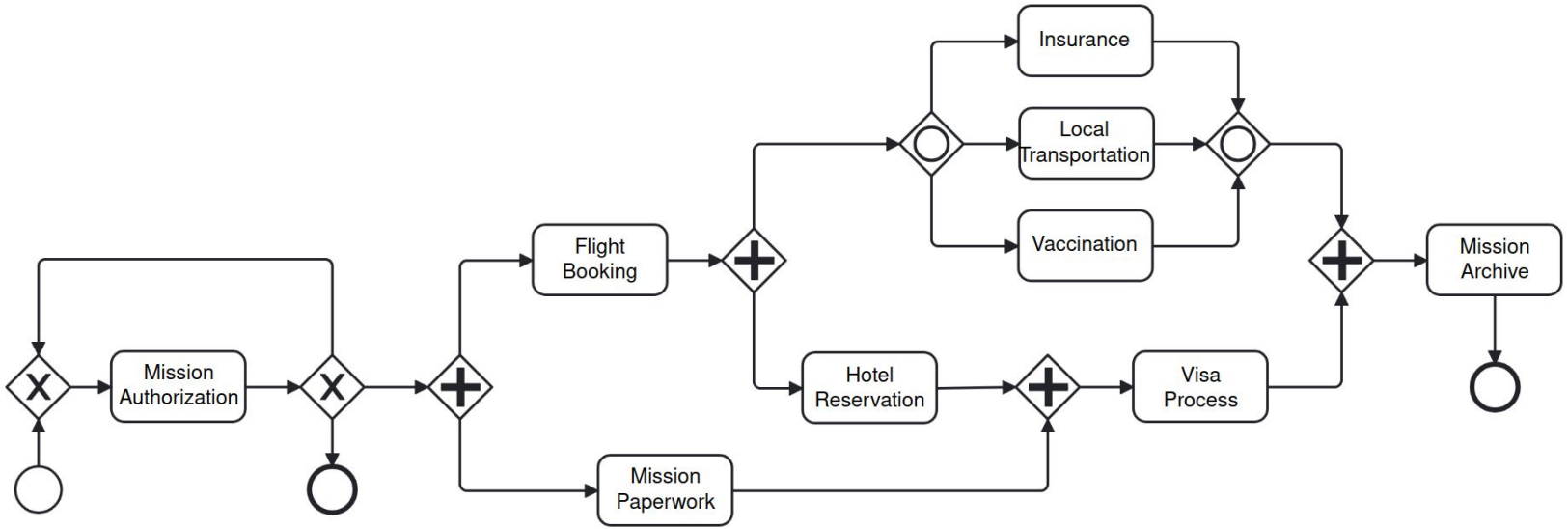


Excerpt of the BPMN Syntax



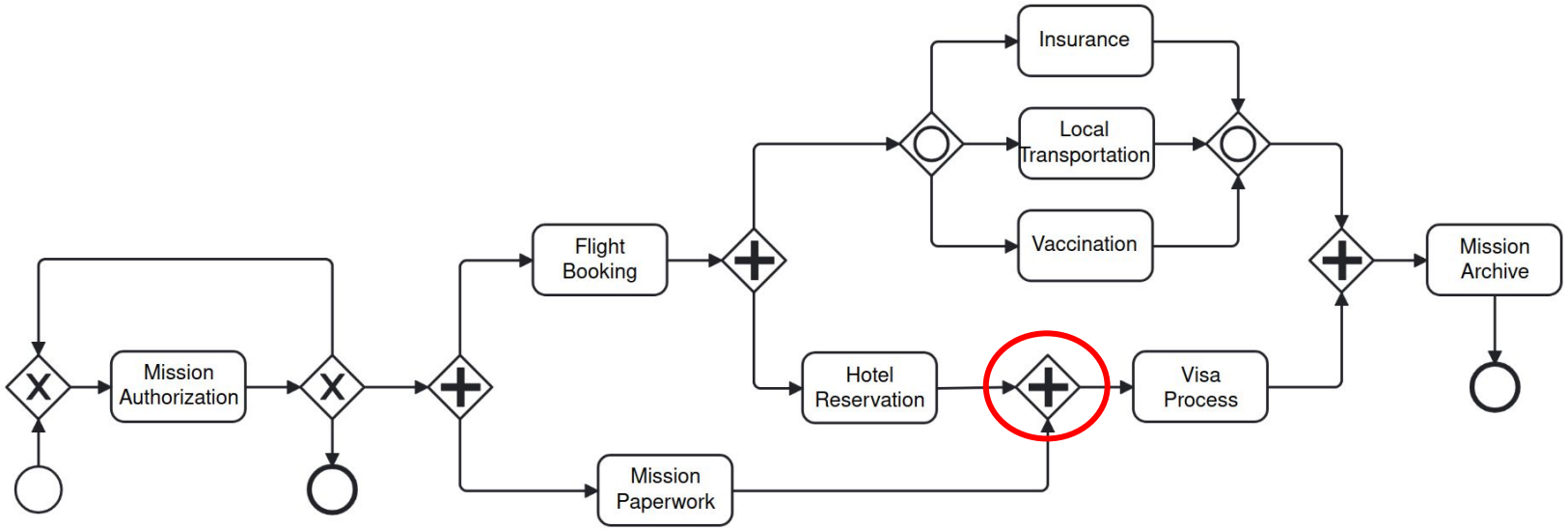
Example of BPMN Process

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



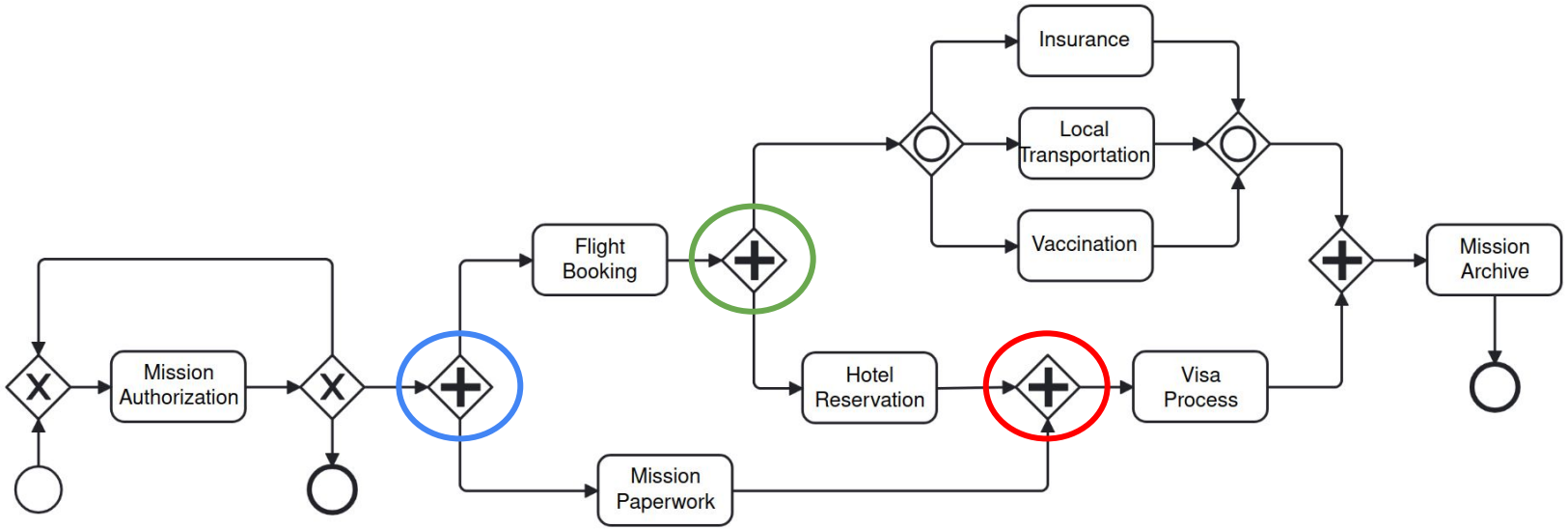
Example of BPMN Process

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



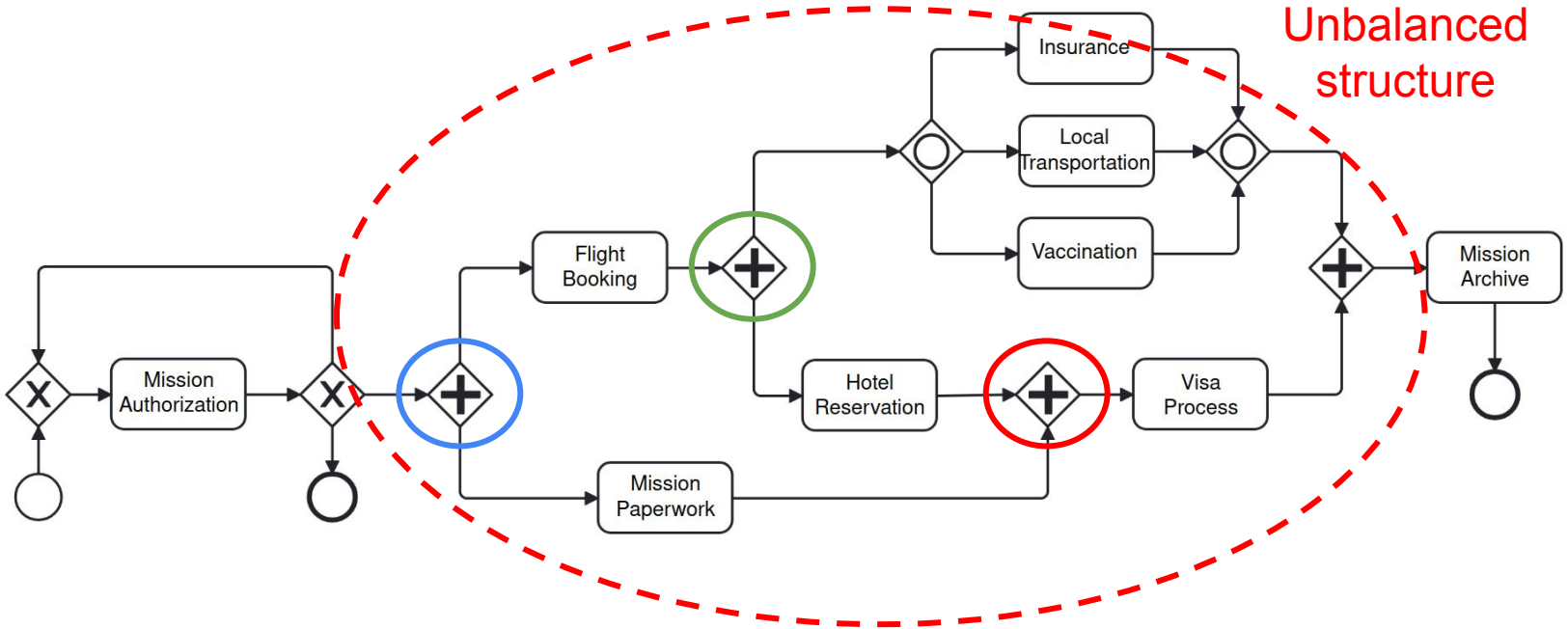
Example of BPMN Process

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



Example of BPMN Process

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

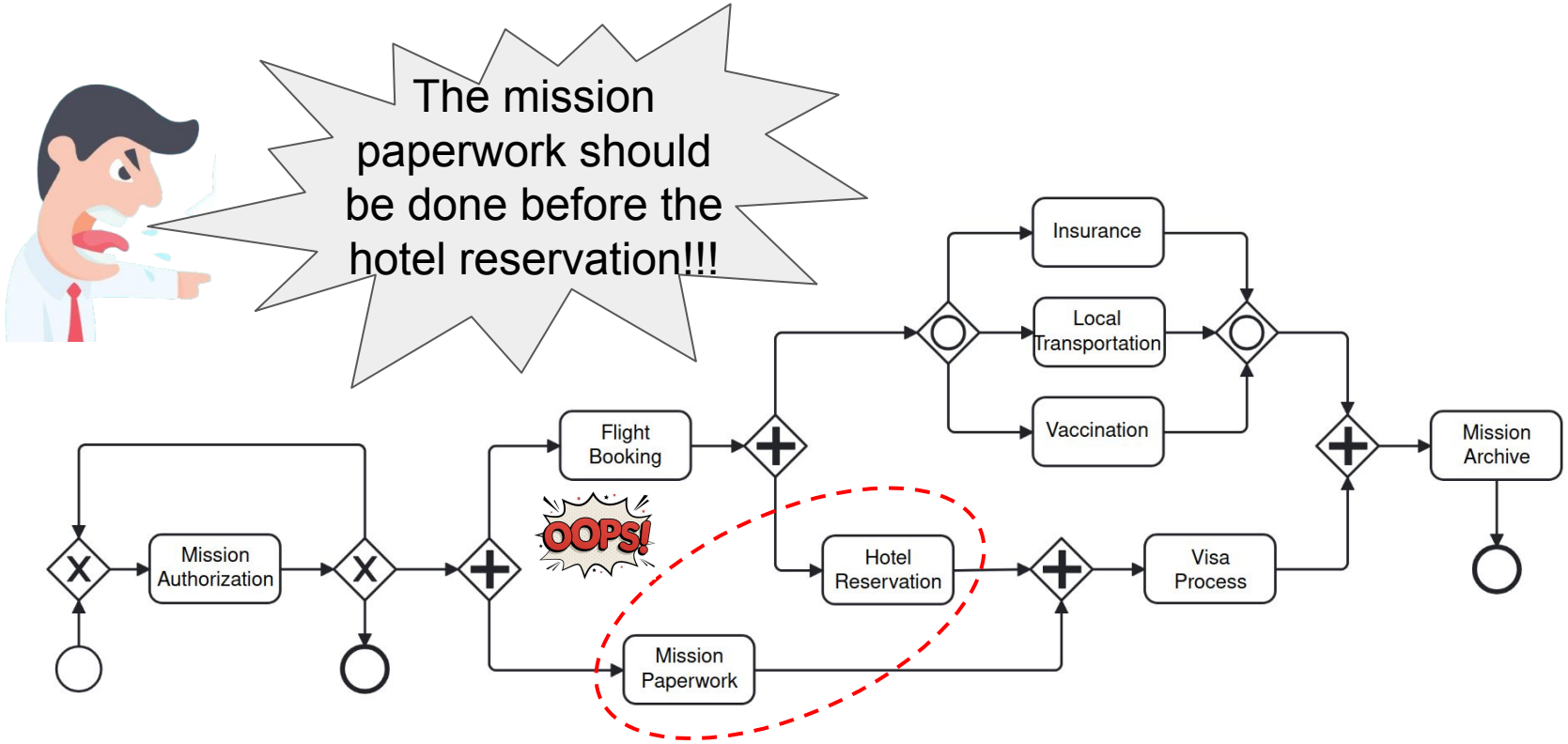


How to write a BPMN process?

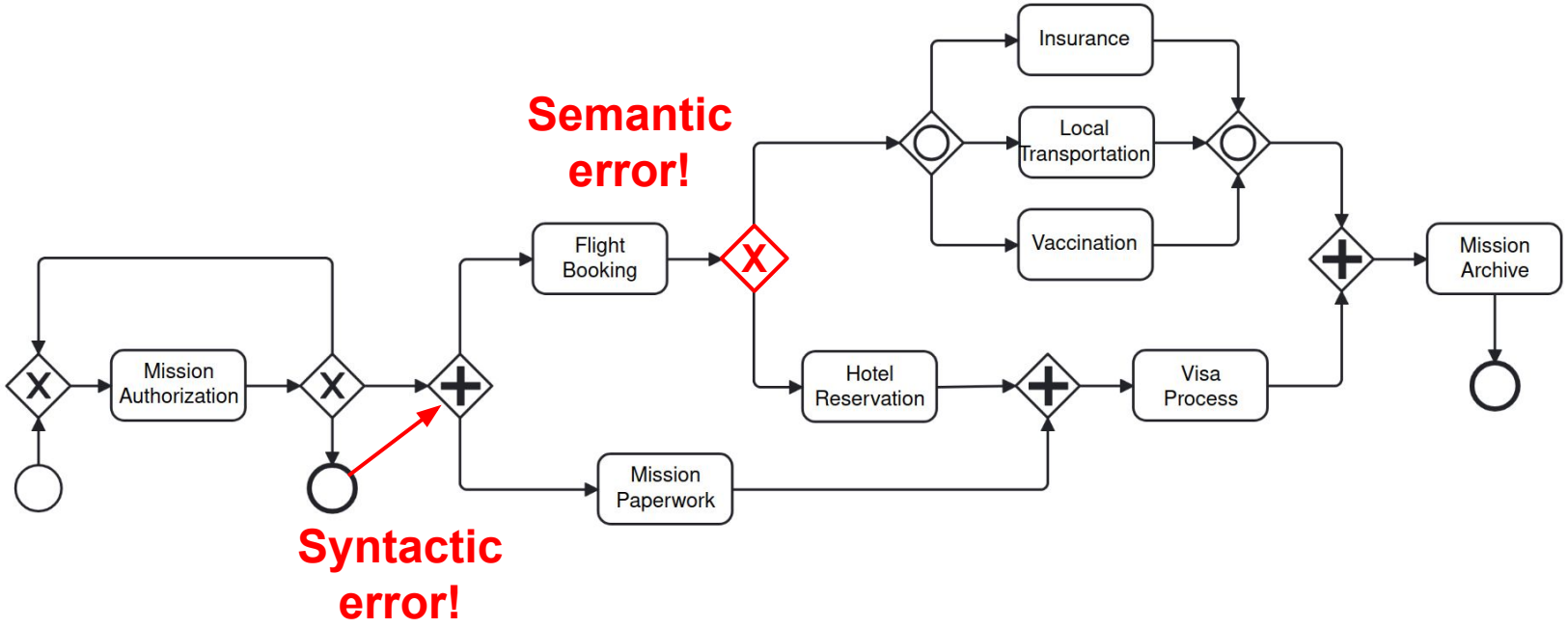
How to avoid wasting time designing?

~30m!

How to match the expected behaviour?



How to ensure syntactic/semantic correctness?



Modelling BPMN processes

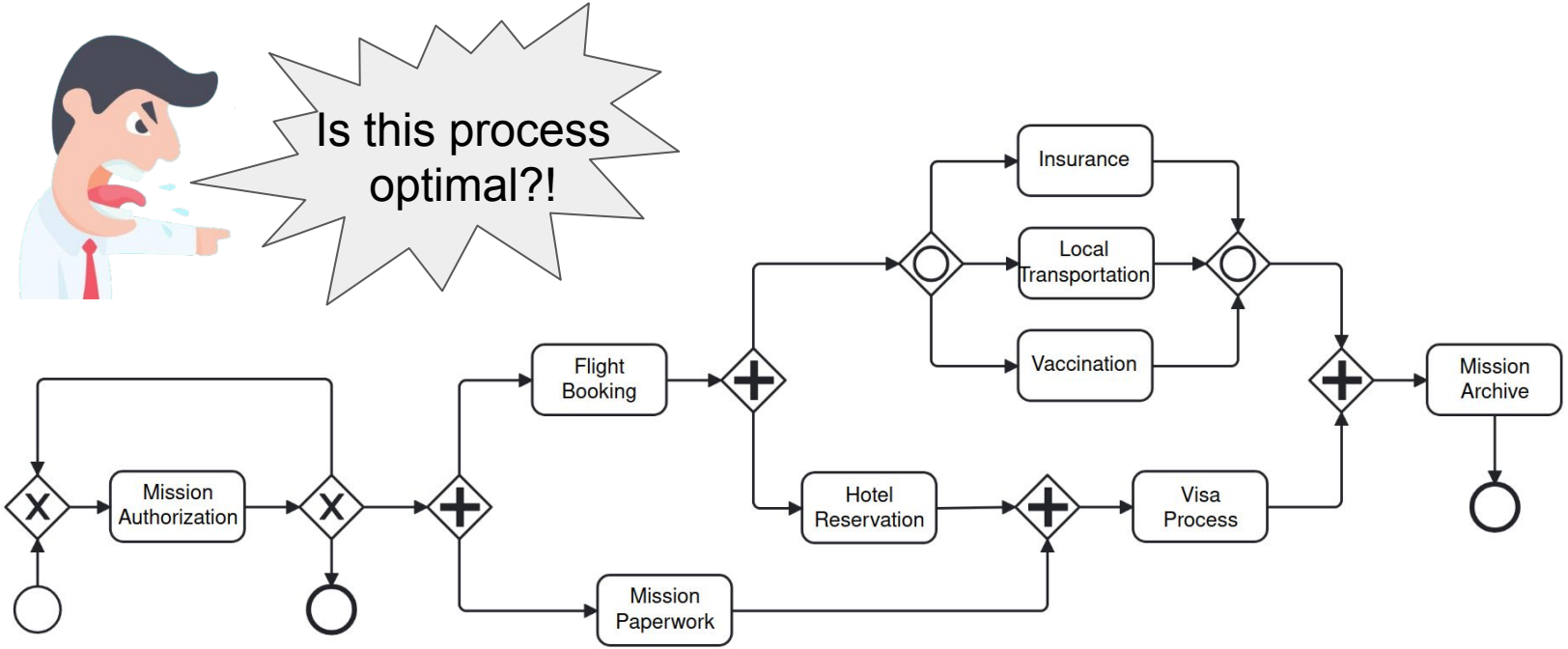
- How to **write** a **BPMN** process?
- How to **avoid wasting time** designing?
- How to **match** the expected **behaviour**?
- How to ensure **syntactic/semantic correctness**?

How to optimise a BPMN process?

How to optimise a BPMN process?

How to optimise a BPMN process?

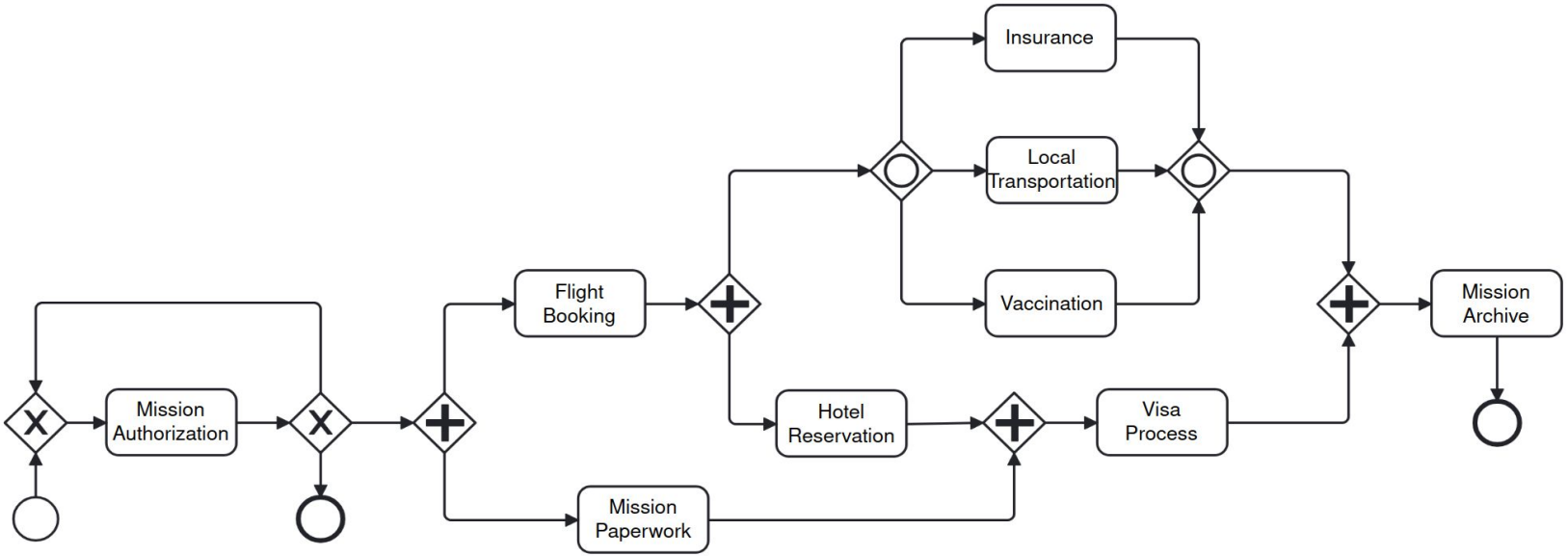
How to optimise a BPMN process?



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

How to optimise a BPMN process?

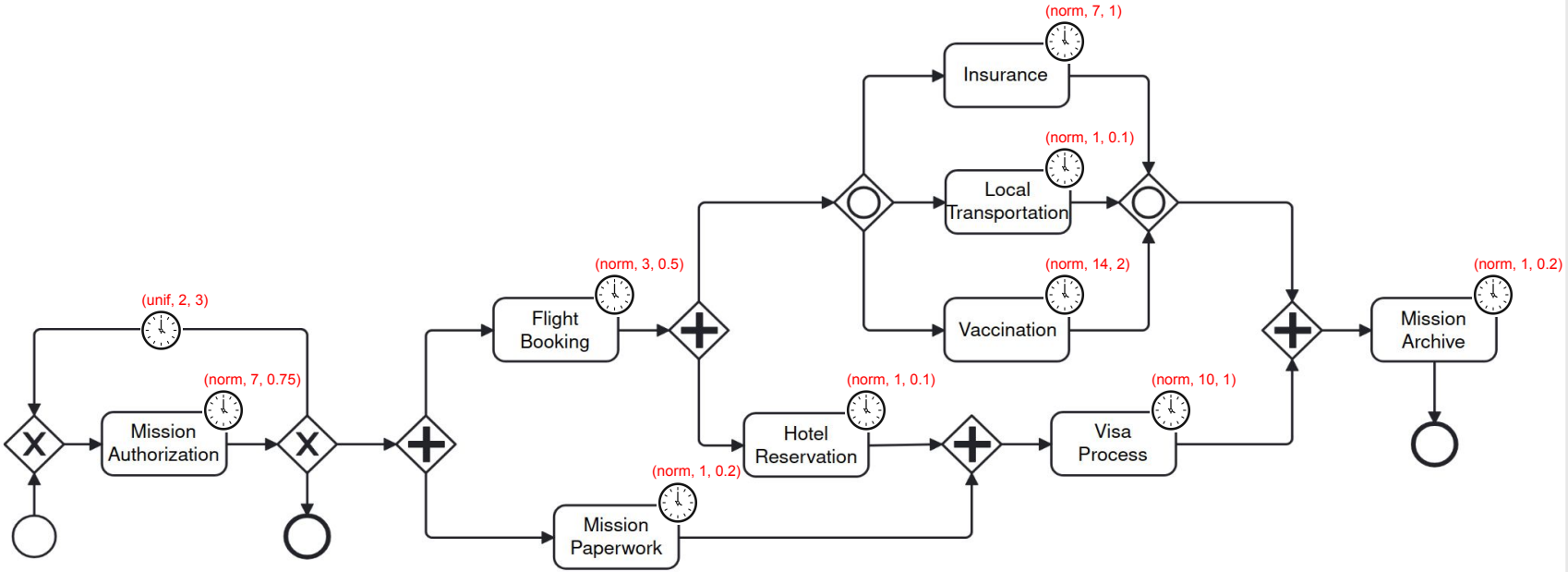
But what if we enrich the process with:



How to optimise a BPMN process?

But what if we enrich the process with:

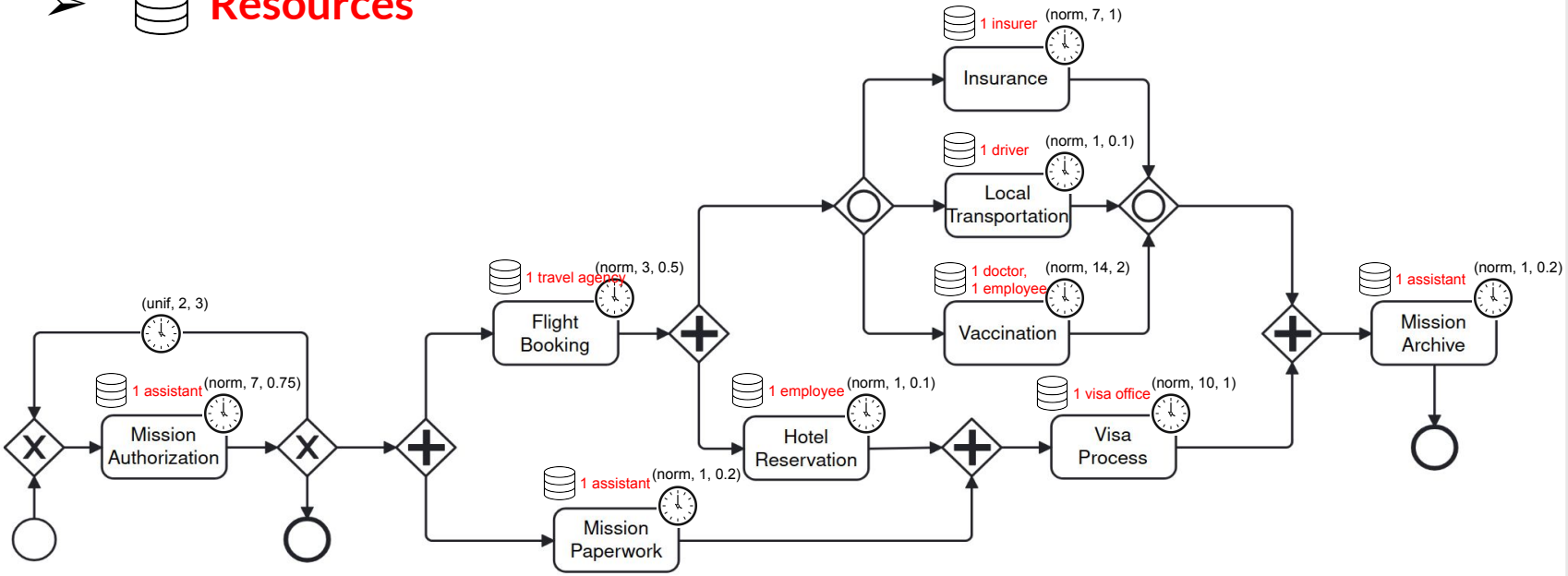
- **Durations** (following probabilistic distributions)



How to optimise a BPMN process?

But what if we enrich the process with:

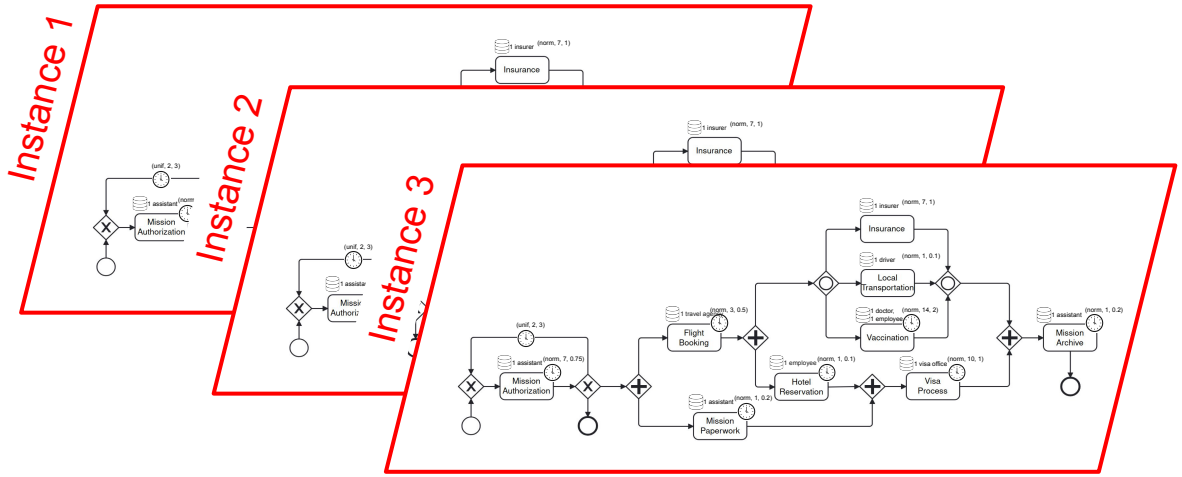
- Durations (following probabilistic distributions)
- **Resources**



How to optimise a BPMN process?

But what if we enrich the process with:

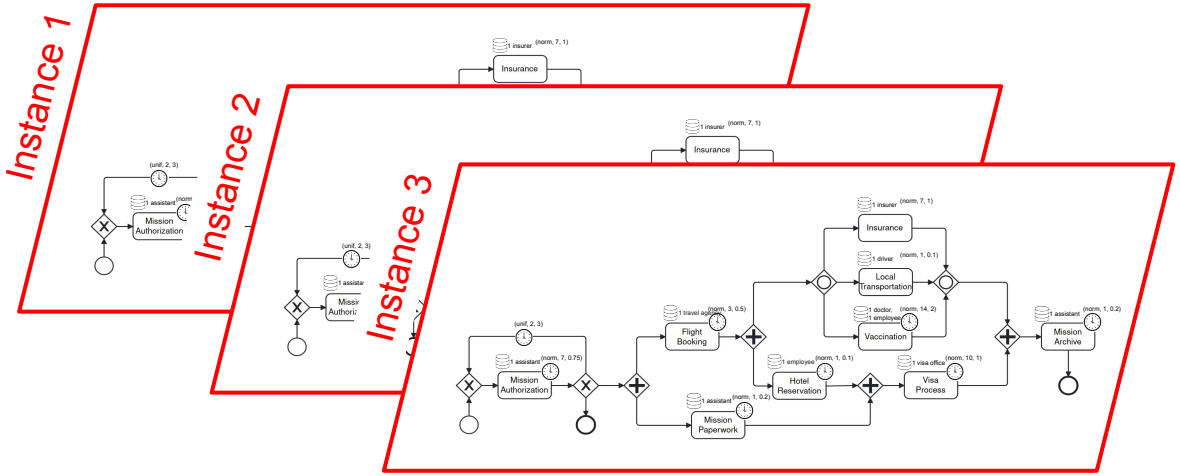
- Durations (following probabilistic distributions)
- Resources
- **Multiple Simultaneous Executions**



How to optimise a BPMN process?

But what if we enrich the process with:

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



⇒ The problem becomes **much more complex!**

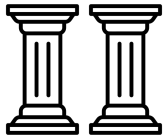
PhD Research Axes

Modelling BPMN processes

- How to **write** a **BPMN** process?
- How to **avoid wasting time** designing?
- How to **match** the expected **behaviour**?
- How to ensure **syntactic/semantic correctness**?

Optimising BPMN processes

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



Contributions of the Thesis – Modelling of Processes

Contributions of the Thesis – Modelling of Processes

- An approach **generating a BPMN process** from a textual description of its requirements which:
  - Manipulates **abstract syntax trees**
  - Handles **balanced BPMN** processes

ICSOC'24

Contributions of the Thesis – Modelling of Processes

- An approach **generating a BPMN process** from a textual description of its requirements which:
  - Manipulates **abstract syntax trees**
  - Handles **balanced BPMN** processes

  } ICSOC'24

- A **tool** approach coupling:
  - **Generation** of the BPMN process
  - **Verification** based on textual descriptions of temporal logic properties

  } FSE'25

Contributions of the Thesis – Modelling of Processes

- An approach **generating a BPMN process** from a textual description of its requirements which:
  - Manipulates **abstract syntax trees**
  - Handles **balanced BPMN** processes

  } ICSOC'24

- A **tool** approach coupling:
  - **Generation** of the BPMN process
  - **Verification** based on textual descriptions of temporal logic properties

  } FSE'25

- An **extension** of the **BPMN generation** approach to:
  - Handle **unbalanced** processes
  - Provide **strong semantical guarantees**

  } TSE'25 (submitted)

Contributions of the Thesis – Modelling of Processes

- An approach **generating a BPMN process** from a textual description of its requirements which:
  - Manipulates **abstract syntax trees**
  - Handles **balanced BPMN** processes

  } ICSOC'24

- A **tool** approach coupling:
  - **Generation** of the BPMN process
  - **Verification** based on textual descriptions of temporal logic properties

  } FSE'25

- An **extension** of the **BPMN generation** approach to:
  - Handle **unbalanced** processes
  - Provide **strong semantical guarantees**

  } TSE'25 (submitted)

Contributions of the Thesis – Optimisation of Processes

- An approach **refactoring a BPMN process** with:
  - **Static analysis** of the process
  - Computation of (theoretical) **optimal pool** of resources
  - Support for **constant** durations

SEFM'23

- An approach **refactoring a BPMN process** with:
  - **Static analysis** of the process
  - Computation of (theoretical) **optimal pool** of resources
  - Support for **constant** durations

SEFM'23

- An approach **refactoring a BPMN process** with:
  - **Simulation-based analysis** of the process
  - **Involvement of the user** in the decisions
  - Support for **non-constant** durations

QRS'24

Contributions of the Thesis – Optimisation of Processes

- An approach **refactoring a BPMN process** with:
  - **Static analysis** of the process
  - Computation of (theoretical) **optimal pool** of resources
  - Support for **constant** durations

  } SEFM'23

- An approach **refactoring a BPMN process** with:
  - **Simulation-based analysis** of the process
  - **Involvement of the user** in the decisions
  - Support for **non-constant** durations

  } QRS'24

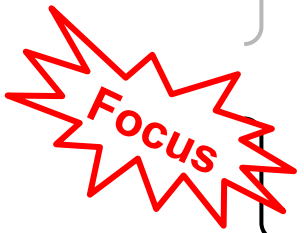
- An **extension** of the **second approach** to:
  - Handle **multiple optimisation criteria**

  } JSS'25 (submitted)

Contributions of the Thesis – Optimisation of Processes

- An approach **refactoring a BPMN process** with:
  - **Static analysis** of the process
  - Computation of (theoretical) **optimal pool** of resources
  - Support for **constant** durations

SEFM'23



- An approach **refactoring a BPMN process** with:
  - **Simulation-based analysis** of the process
  - **Involvement of the user** in the decisions
  - Support for **non-constant** durations

QRS'24

- An **extension** of the **second approach** to:
  - Handle **multiple optimisation criteria**

JSS'25 (submitted)

Plan

I/ Introduction

II/ Automated Generation of BPMN Processes from Textual Requirements

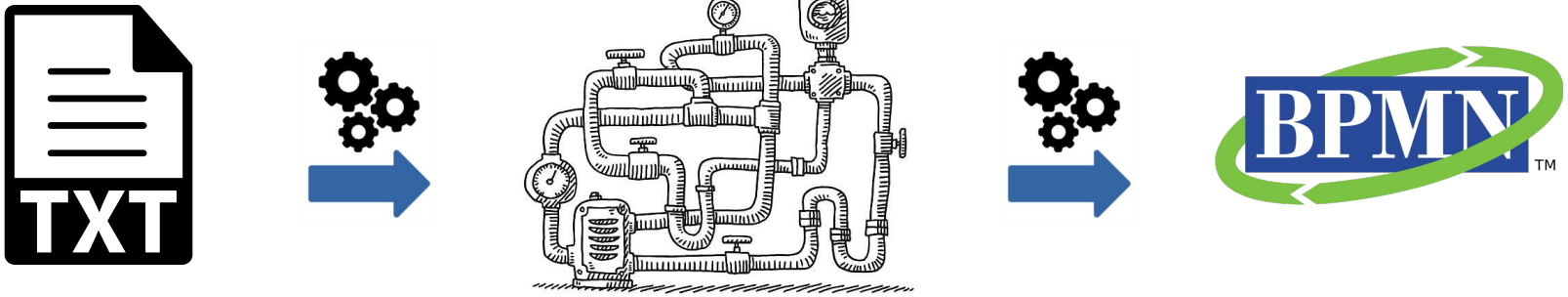
III/ Human-Centered Refactoring-Based Optimisation of BPMN Processes

IV/ Related Work

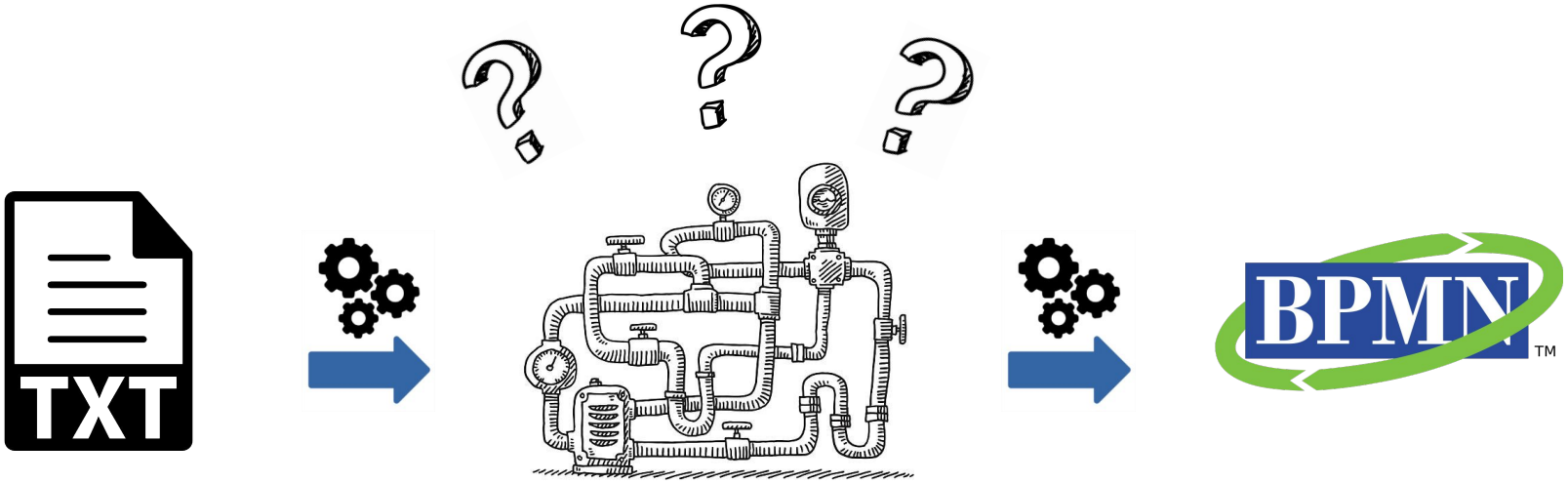
V/ Takeaways

VI/ References

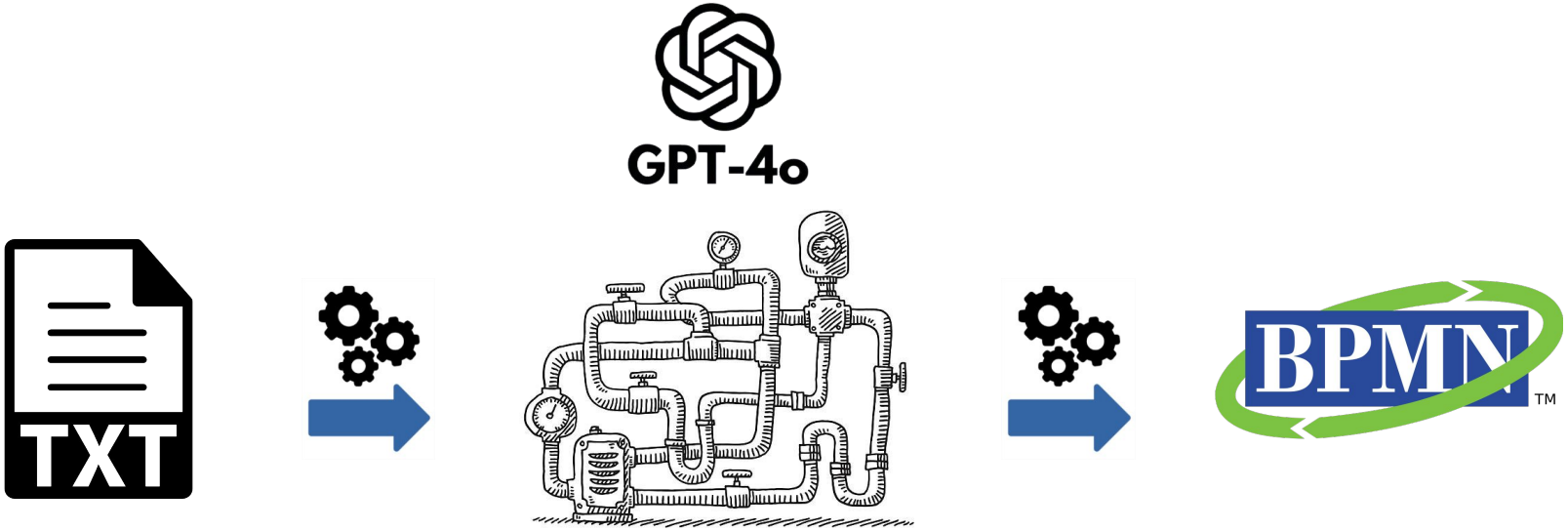
Automated Generation of BPMN – Big Picture



Automated Generation of BPMN – Big Picture



Automated Generation of BPMN – Big Picture



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)

$$\langle E \rangle ::= \texttt{t} \mid (\langle E \rangle) \mid \langle E_1 \rangle \langle op \rangle \langle E_2 \rangle \mid (\langle E_1 \rangle)*$$

$$\langle op \rangle ::= \text{`|'} \mid \text{`\&'} \mid \text{`<'} \mid \text{`,'}$$

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

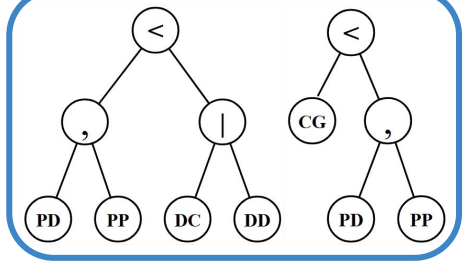
Large Language Model (LLM)

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

$$\langle E \rangle ::= \texttt{t} \mid (\langle E \rangle) \mid \langle E_1 \rangle \langle op \rangle \langle E_2 \rangle \mid (\langle E_1 \rangle)*$$

$$\langle op \rangle ::= \text{`|'} \mid \text{`\&'} \mid \text{`<'} \mid \text{`,'}$$

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

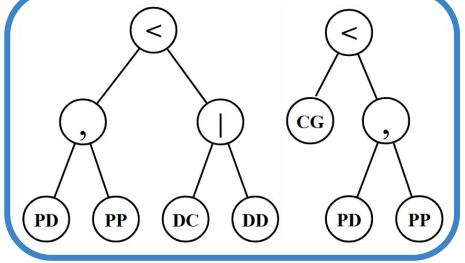
Large Language Model (LLM)

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

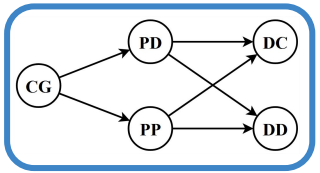
$$\langle E \rangle ::= \texttt{t} \mid (\langle E \rangle) \mid \langle E_1 \rangle \langle op \rangle \langle E_2 \rangle \mid (\langle E_1 \rangle)*$$

$$\langle op \rangle ::= \text{`|'} \mid \text{`\&'} \mid \text{`<'} \mid \text{`,'}$$

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

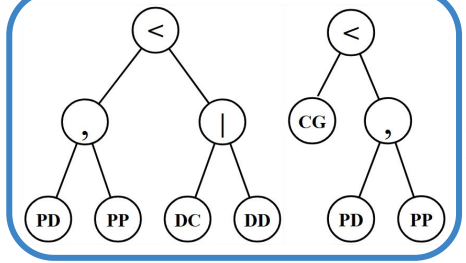
Large Language Model (LLM)

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

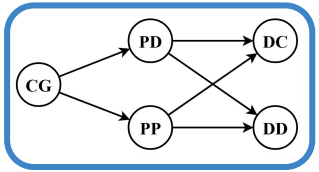
$$\langle E \rangle ::= \texttt{t} \mid (\langle E \rangle) \mid \langle E_1 \rangle \langle op \rangle \langle E_2 \rangle \mid (\langle E_1 \rangle)*$$

$$\langle op \rangle ::= \text{`|'} \mid \text{`\&'} \mid \text{`<'} \mid \text{`,'}$$

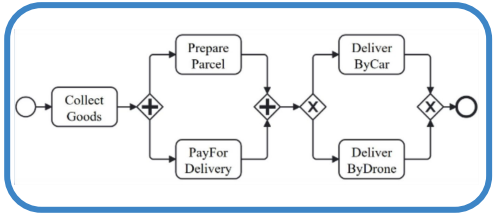
Expressions Following an Internal Grammar



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

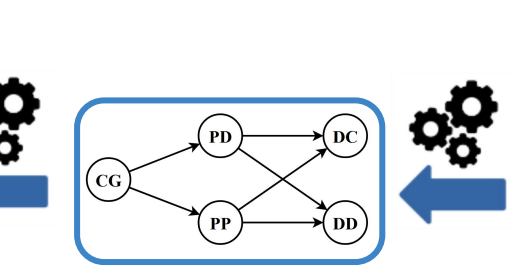
Large Language Model (LLM)

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

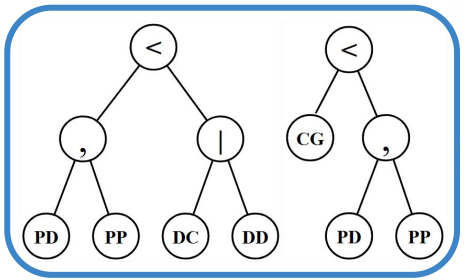
$$\langle E \rangle ::= \texttt{t} \mid (\langle E \rangle) \mid \langle E_1 \rangle \langle op \rangle \langle E_2 \rangle \mid (\langle E_1 \rangle)*$$

$$\langle op \rangle ::= \text{`|'} \mid \text{`\&'} \mid \text{`<'} \mid \text{`,'}$$

Expressions Following an Internal Grammar

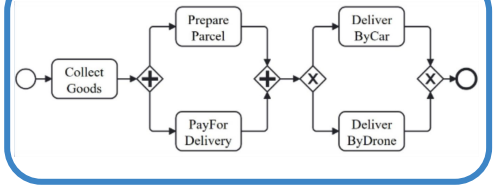


Dependency Graph (Skeleton of the Process)



Abstract Syntax Trees

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**).

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**).

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

$$\langle \mathrm{E} \rangle ::= \mathtt{t} \mid (\langle \mathrm{E} \rangle) \mid \langle \mathrm{E}_1 \rangle \langle \mathrm{op} \rangle \langle \mathrm{E}_2 \rangle \mid (\langle \mathrm{E}_1 \rangle)*$$
$$\langle \mathrm{op} \rangle ::= \text{`|'} \mid \text{`\&'} \mid \text{`<'} \mid \text{`,'}$$

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)*

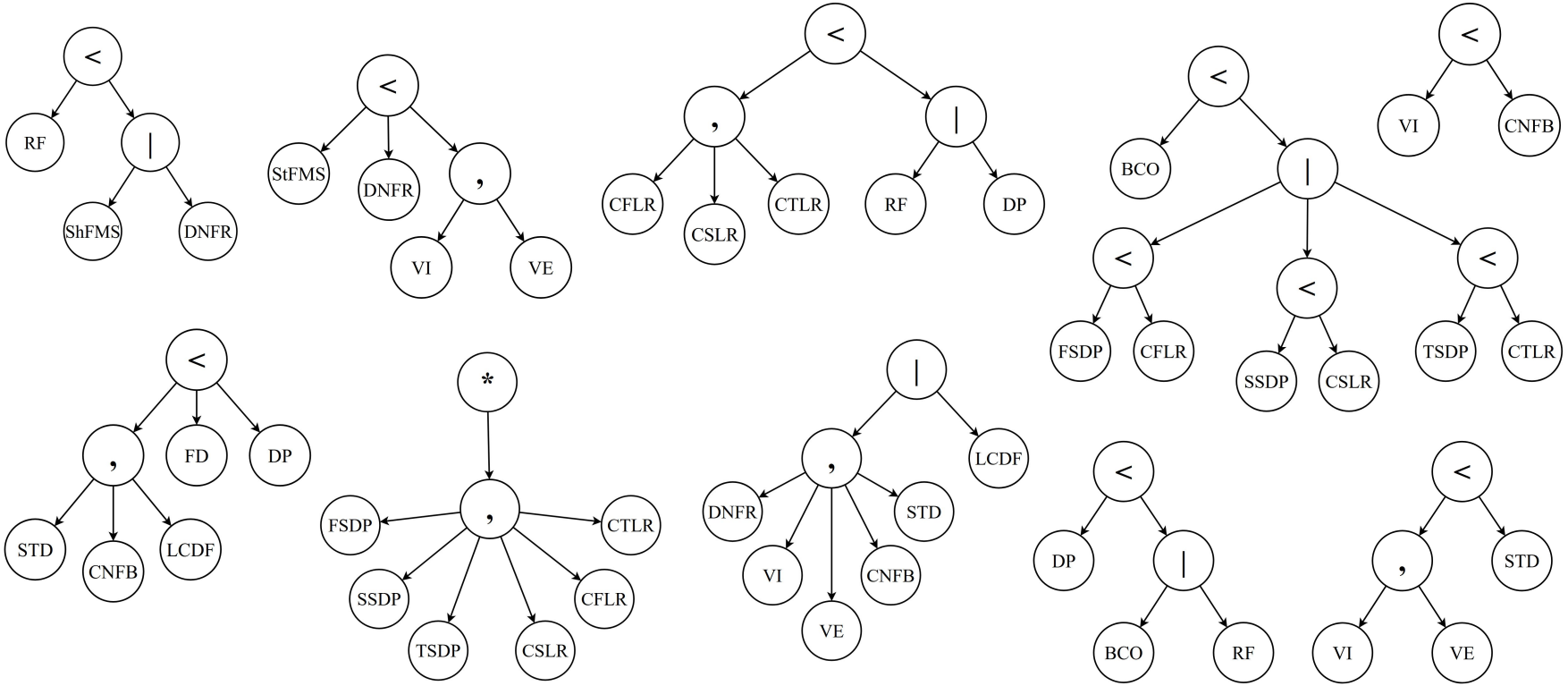
RF < (ShFMS | DNFR)

Detailed Approach – Step 4 – ASTs Mapping

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

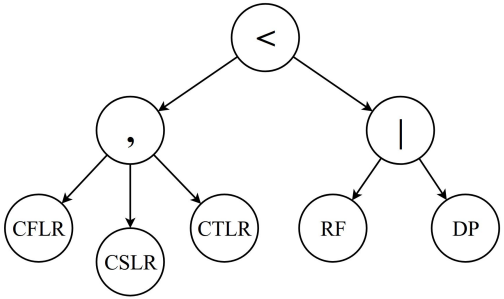
Detailed Approach – Step 4 – ASTs Mapping

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)**.

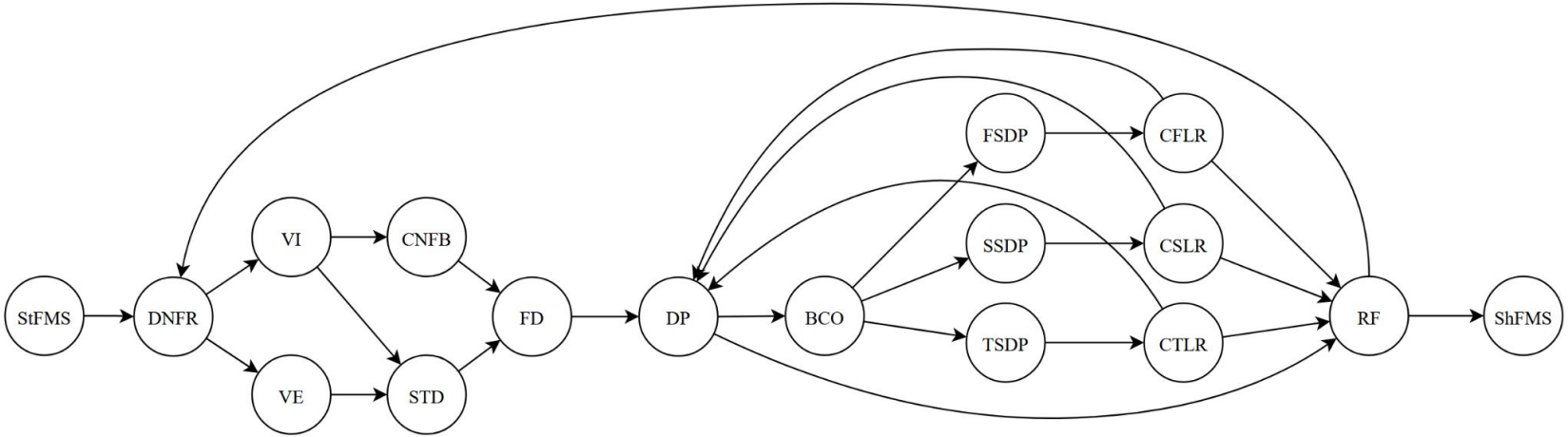


Detailed Approach – Step 5 – Dependency Graph

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**.

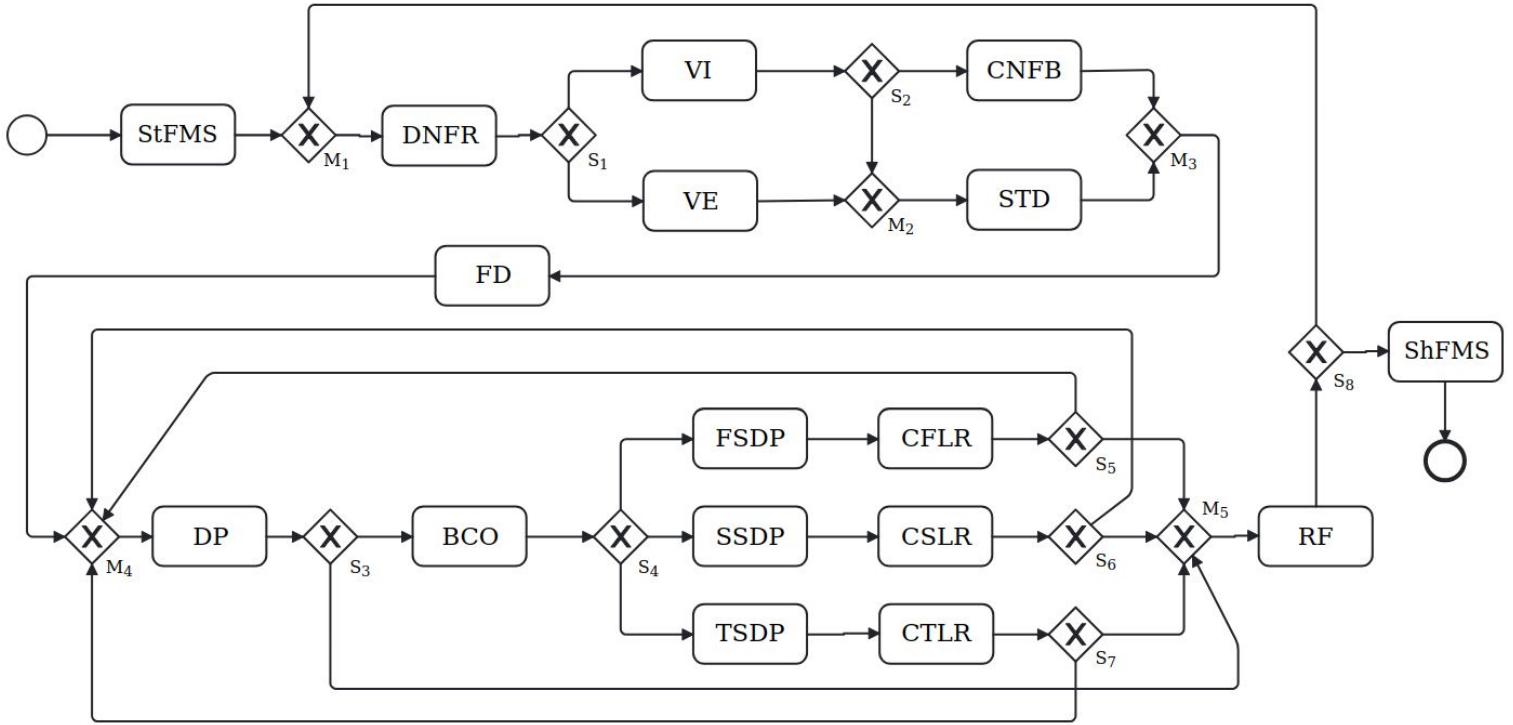


Detailed Approach – Step 6 – BPMN Generation

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**.

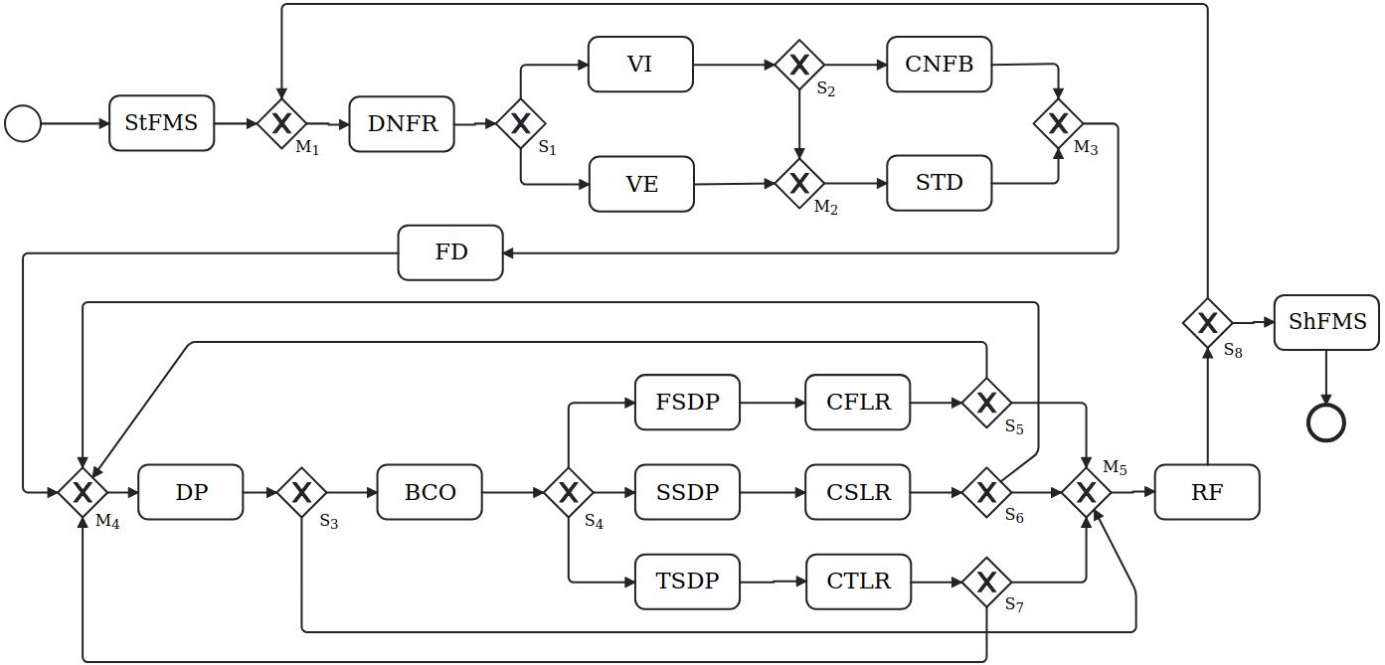


Detailed Approach – Step 7 – Refinement

However, this process is **incomplete** with regards to the expressions!

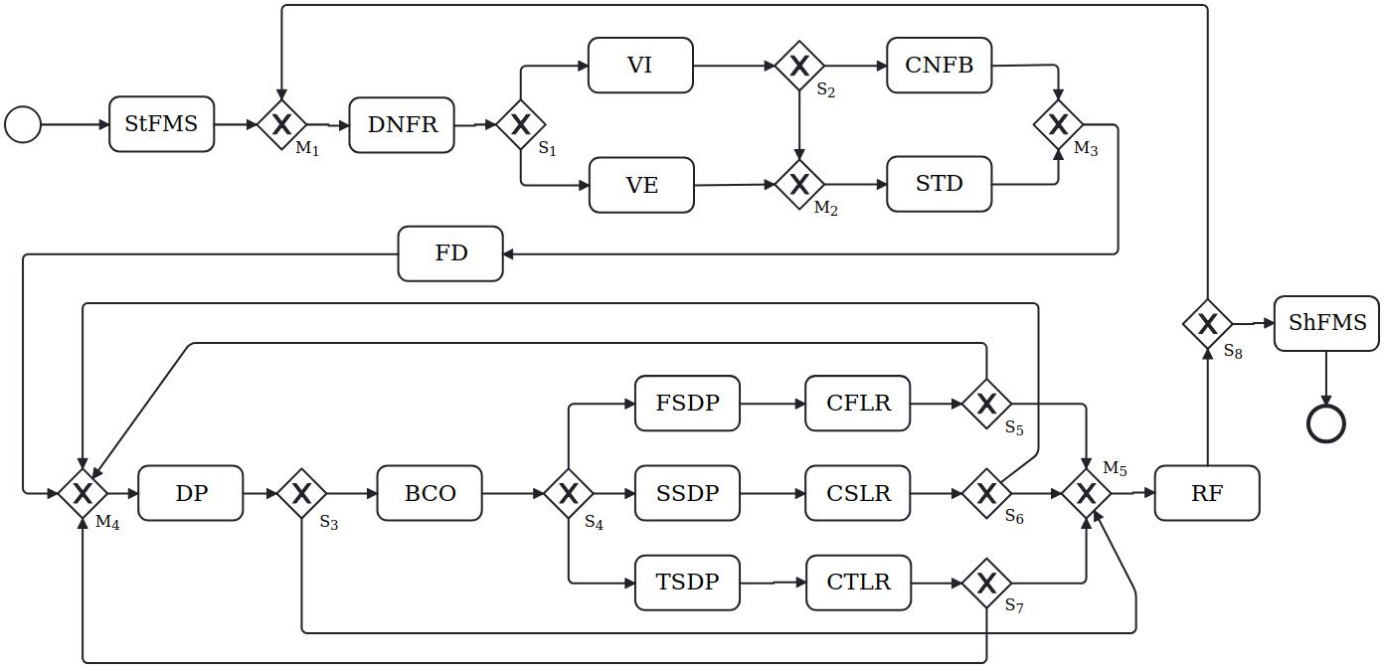
Detailed Approach – Step 7 – Refinement

However, this process is **incomplete** with regards to the expressions!



Detailed Approach – Step 7 – Refinement

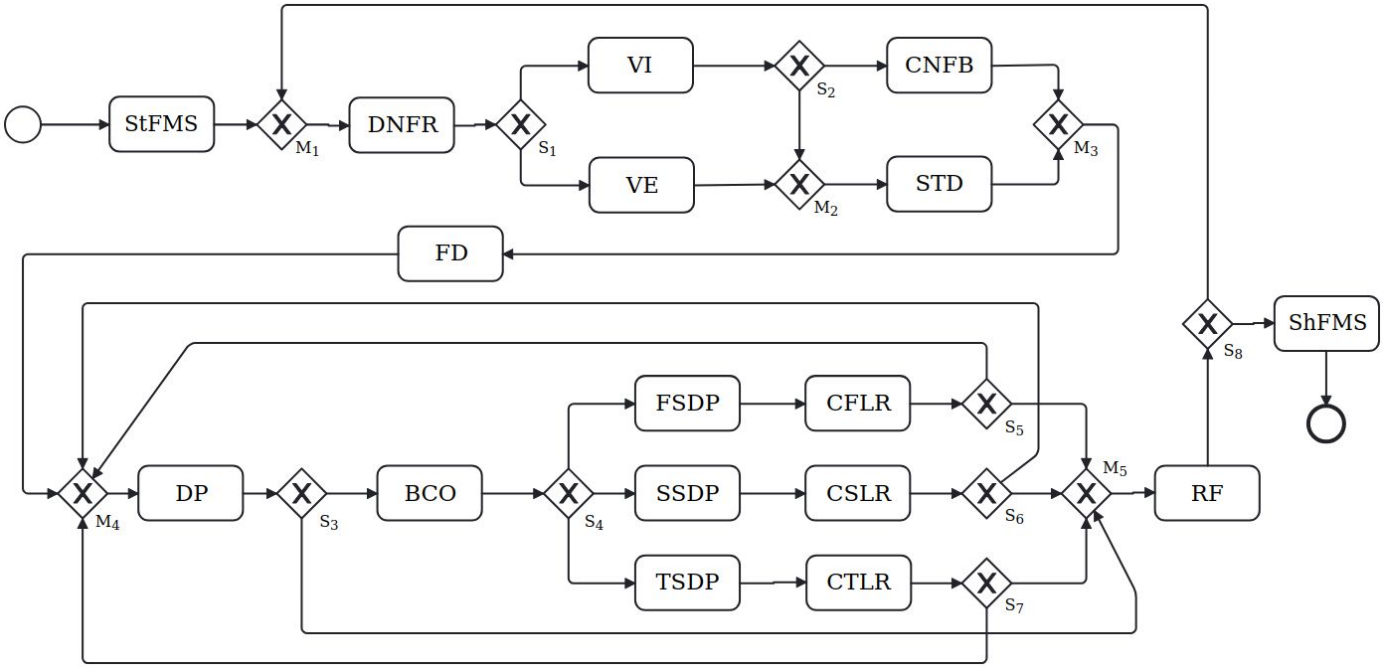
However, this process is **incomplete** with regards to the expressions!



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

Detailed Approach – Step 7 – Refinement

However, this process is **incomplete** with regards to the expressions!

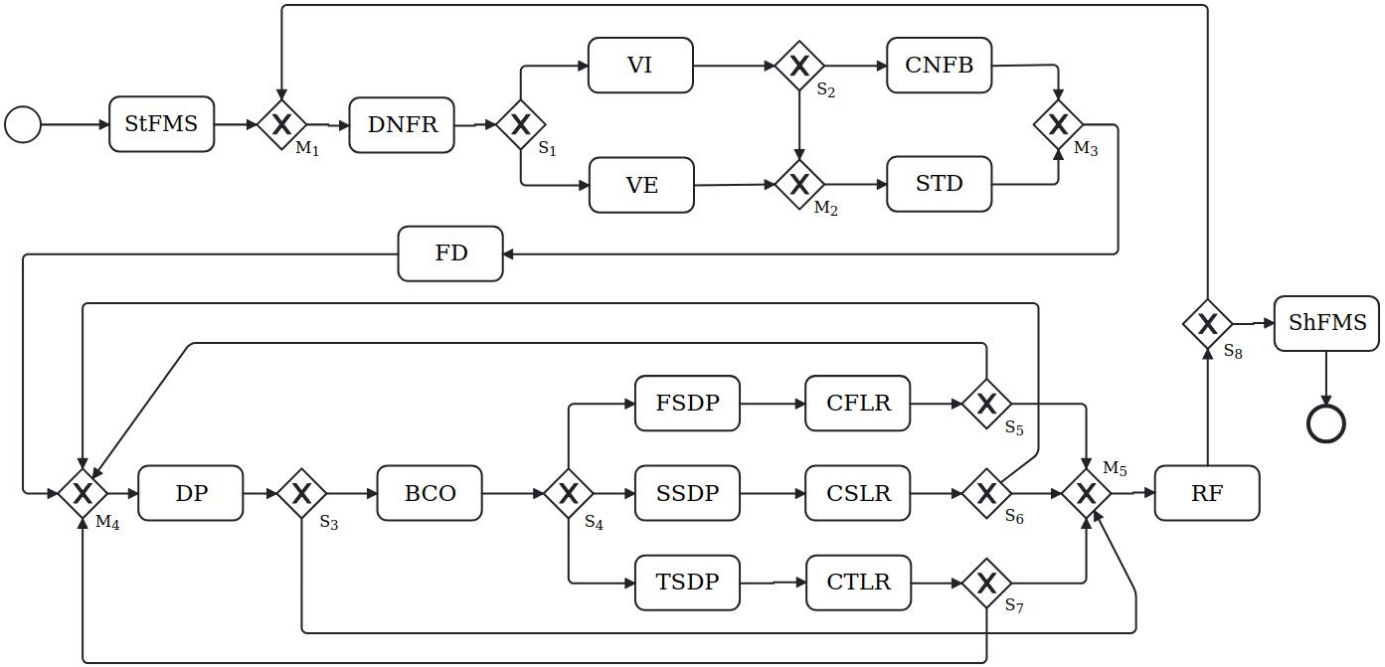


(DNFR, VI, VE, CNFB, STD) ⊢ LCDF

Task LCDF is not in the process!

Detailed Approach – Step 7 – Refinement

However, this process is **incomplete** with regards to the expressions!

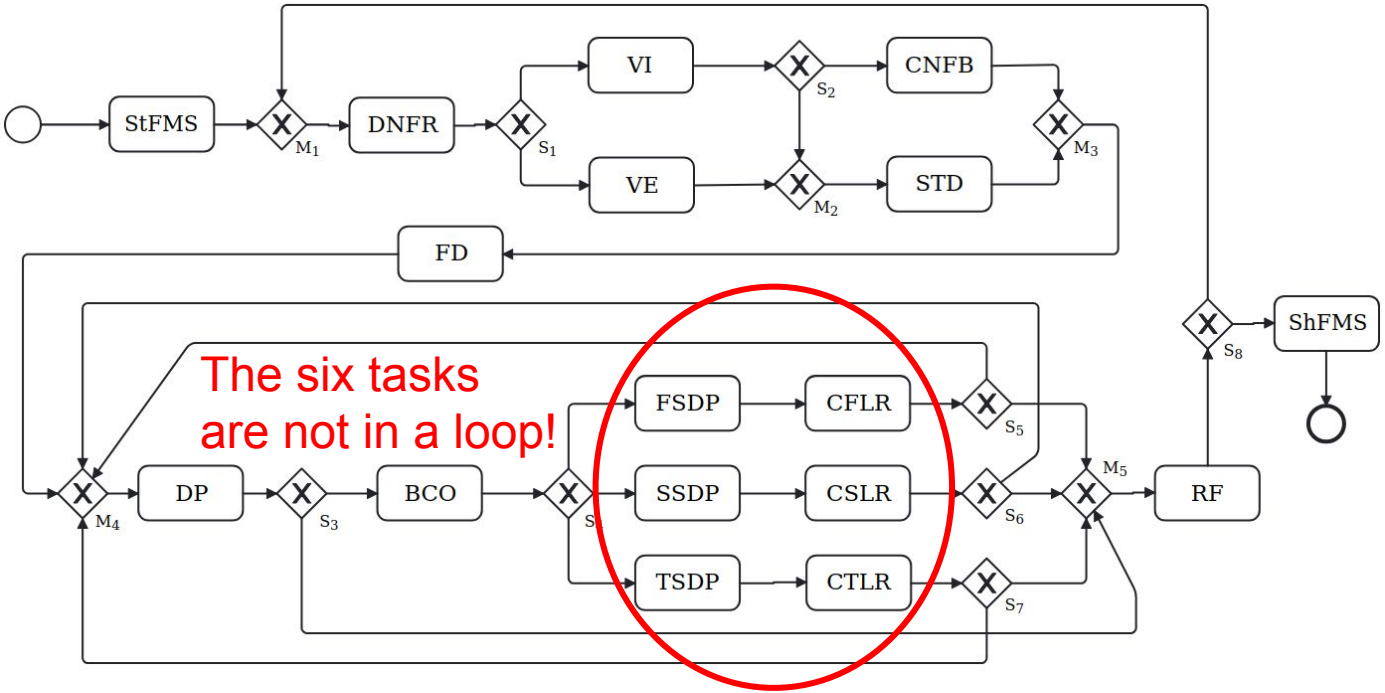


(DNFR, VI, VE, CNFB, STD) | LCDF

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

Detailed Approach – Step 7 – Refinement

However, this process is **incomplete** with regards to the expressions!

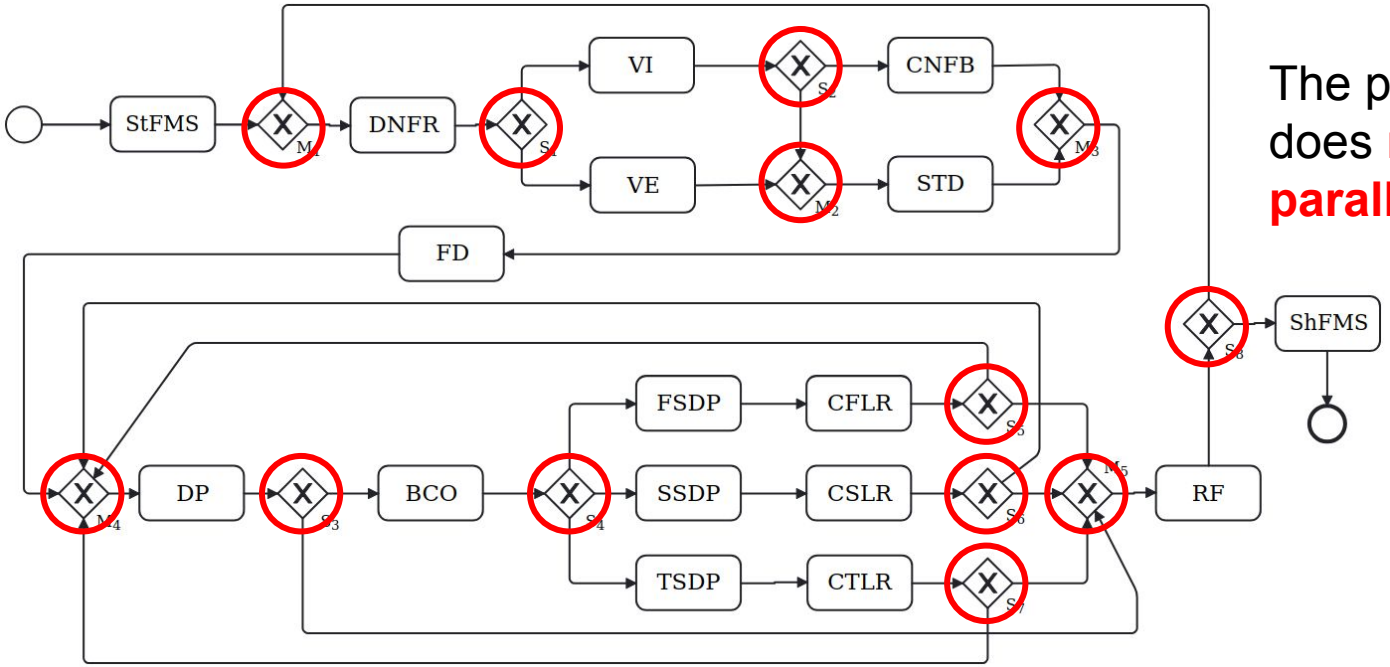


(DNFR, VI, VE, CNFB, STD) | LCDF

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

Detailed Approach – Step 7 – Refinement

However, this process is **incomplete** with regards to the expressions!



The process does **not contain parallelism**!

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

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

Detailed Approach – Step 7 – Refinement

The next step thus consists in **refining the** generated **process** by **adding** to it all the **missing information** stated in the expressions, and **parallelism**.

The next step thus consists in **refining the** generated **process** by **adding** to it all the **missing information** stated in the expressions, and **parallelism**.

Mutual exclusions handling

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

Detailed Approach – Step 7 – Refinement

The next step thus consists in **refining the** generated **process** by **adding** to it all the **missing information** stated in the expressions, and **parallelism**.

Mutual exclusions handling

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

Explicit loops handling

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

The next step thus consists in **refining the** generated **process** by **adding** to it all the **missing information** stated in the expressions, and **parallelism**.

Mutual exclusions handling

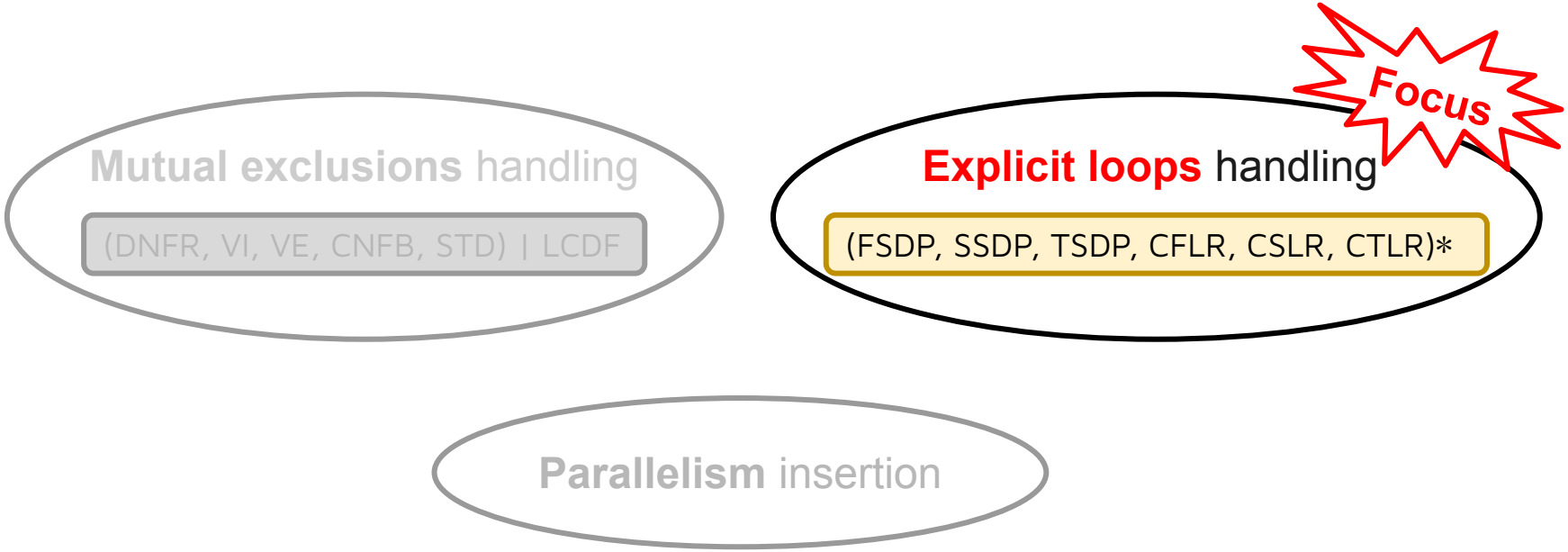
(DNFR, VI, VE, CNFB, STD) | LCDF

Explicit loops handling

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

Parallelism insertion

The next step thus consists in **refining the** generated **process** by **adding** to it all the **missing information** stated in the expressions, and **parallelism**.



Detailed Approach – Step 7.2 – Explicit Loops

In a graph, a loop can be seen as a **strongly connected component**.

In a graph, a loop can be seen as a **strongly connected component**.

Definition (Connected Graph)

Let $G = (V, E, \Sigma)$ be a graph. G is said to be *connected* if and only if for all $(v_1, v_2) \in V^2$, there exists a path p of G such that $v_1 \in p \ \wedge \ v_2 \in p$.

In a graph, a loop can be seen as a **strongly connected component**.

Definition (Connected Graph)

Let $G = (V, E, \Sigma)$ be a graph. G is said to be *connected* if and only if for all $(v_1, v_2) \in V^2$, there exists a path p of G such that $v_1 \in p \ \wedge \ v_2 \in p$.

Definition (Graph Component)

Let $G = (V, E, \Sigma)$ be a graph. A *component* of G is a subgraph $G_S = (V_S, E_S, \Sigma_S) \subseteq G$ such that:

- G_S is connected;
- $\nexists G'_S \subseteq G$ such that $G_S \subseteq G'_S \ \wedge \ G'_S$ is connected.

In a graph, a loop can be seen as a **strongly connected component**.

Definition (Connected Graph)

Let $G = (V, E, \Sigma)$ be a graph. G is said to be *connected* if and only if for all $(v_1, v_2) \in V^2$, there exists a path p of G such that $v_1 \in p \ \wedge \ v_2 \in p$.

Definition (Graph Component)

Let $G = (V, E, \Sigma)$ be a graph. A *component* of G is a subgraph $G_S = (V_S, E_S, \Sigma_S) \subseteq G$ such that:

- G_S is connected;
- $\nexists G'_S \subseteq G$ such that $G_S \subseteq G'_S \ \wedge \ G'_S$ is connected.

Definition (Strongly Connected Component)

Let $G = (V, E, \Sigma)$ be a graph. A *strongly connected component (SCC)* of G is a component $G_S = (V_S, E_S, \Sigma_S)$ of G such that for all $(v_1, v_2) \in V_S^2$, v_1 can reach v_2.

Our proposal thus consists in modifying the **graph restricted to the tasks of the loop** to make it become a **strongly connected component**.

Our proposal thus consists in modifying the **graph restricted to the tasks of the loop** to make it become a **strongly connected component**.

We define **the restriction of a graph** to a subset of its vertices as follows.

Definition (Graph Restriction)

Let $G = (V, E, \Sigma)$ be a graph. The *restriction* of G to the subset $\{v_1, ..., v_n\} \subseteq V$ of its vertices is defined as $G \restriction_{\{v_1,...,v_n\}} \stackrel{\text{def}}{=} (V\restriction, E\restriction, \Sigma\restriction)$ where:

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

Detailed Approach – Step 7.2 – Explicit Loops

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

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

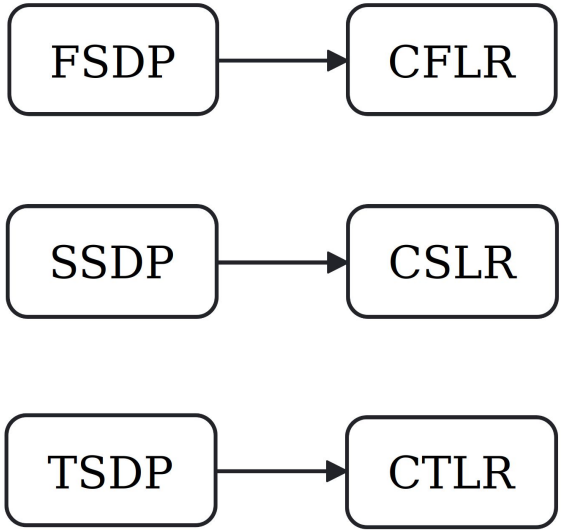
is:

Detailed Approach – Step 7.2 – Explicit Loops

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

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

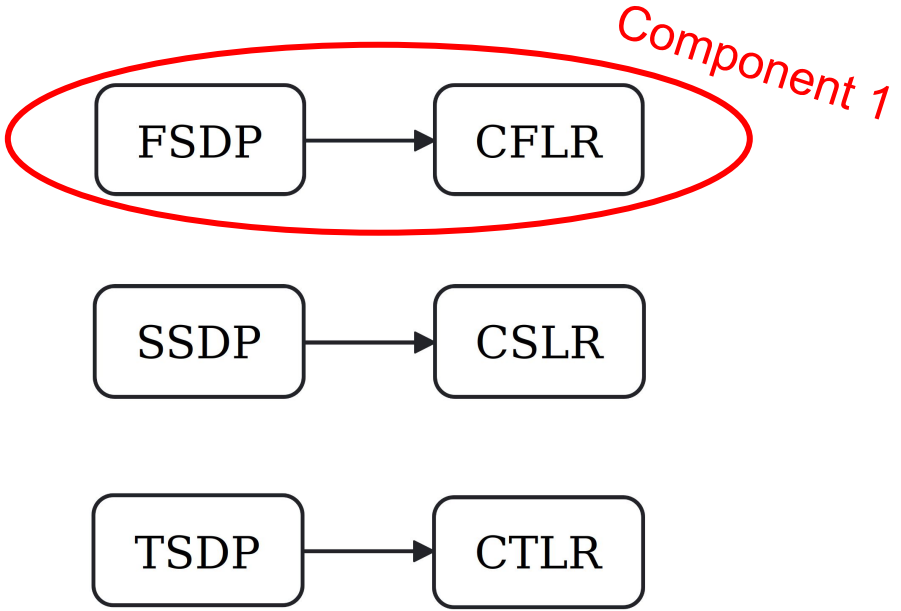
is:



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

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

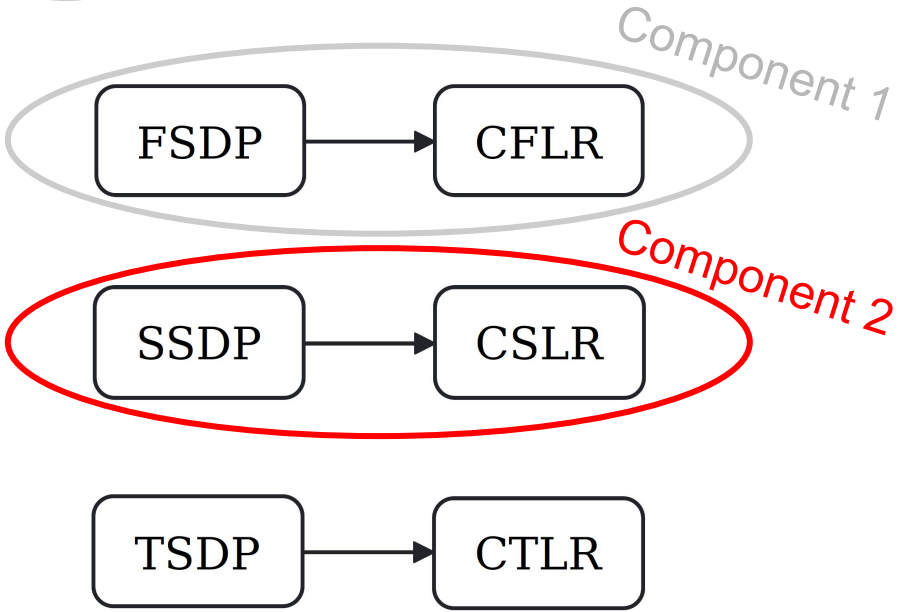
is:



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

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

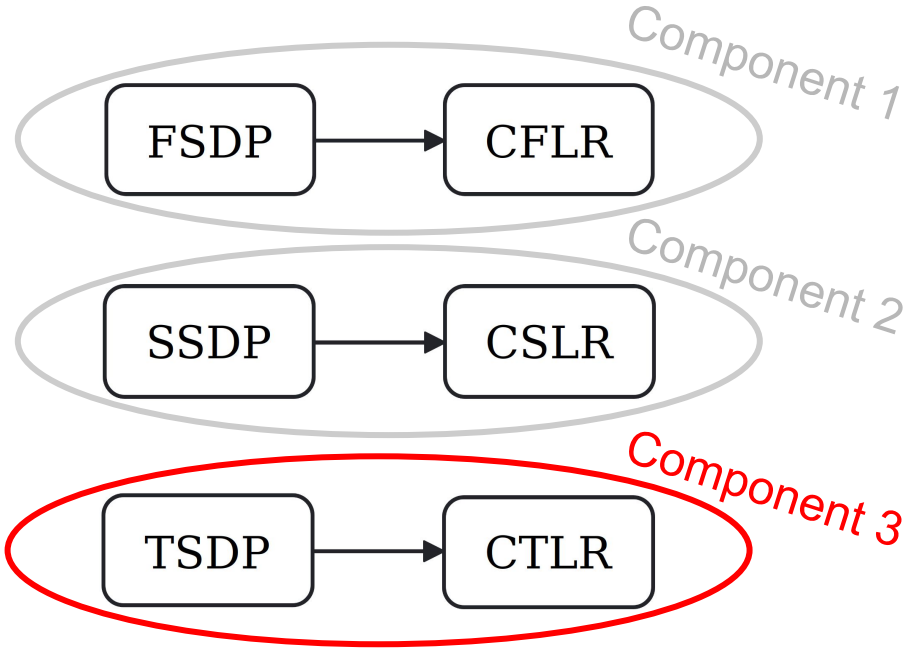
is:



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

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

is:



Detailed Approach – Step 7.2 – Explicit Loops

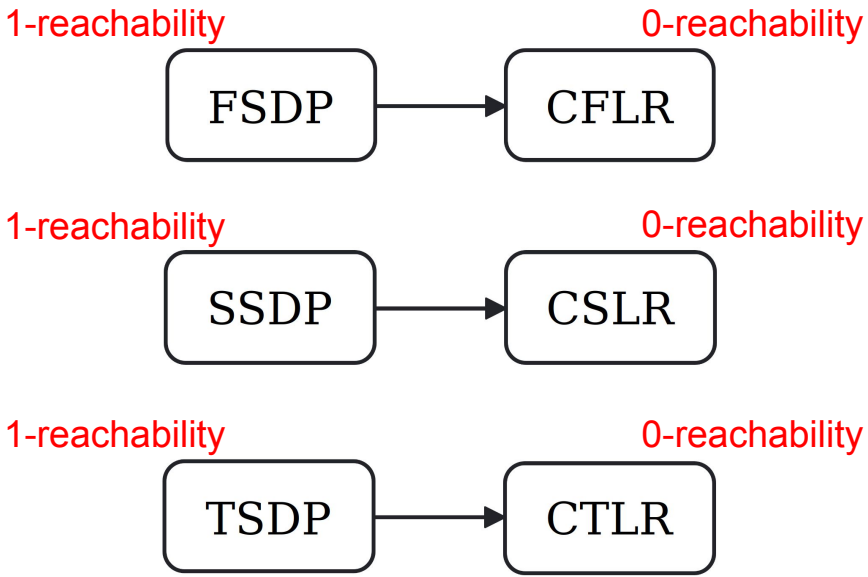
These **components** are then **connected** to create a **single component**.

These **components** are then **connected** to create a **single component**.

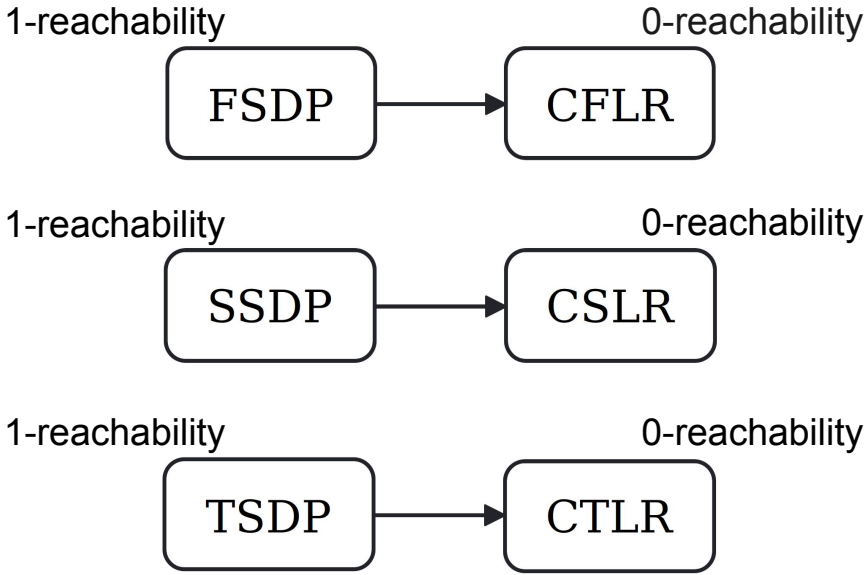
This connection is based on the ***n*-reachability** of each node, i.e., the number of nodes that they can reach.

These **components** are then **connected** to create a **single component**.

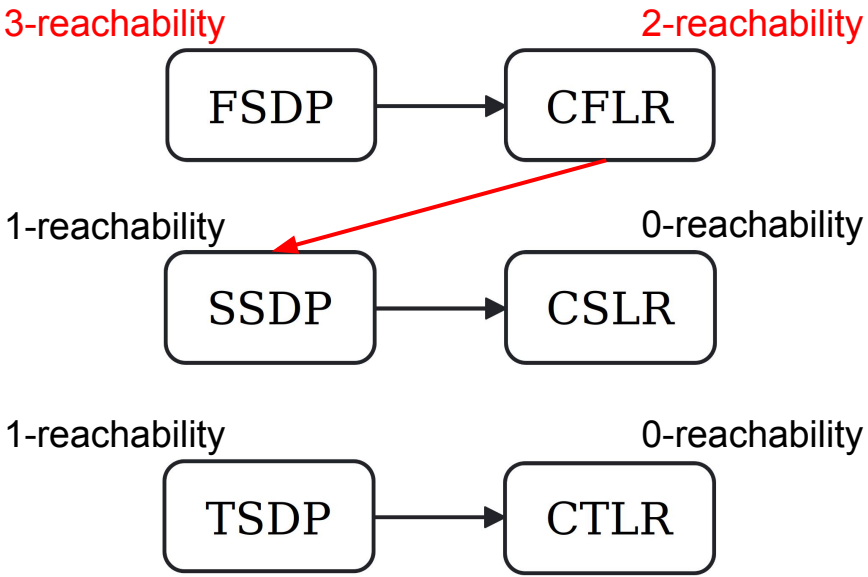
This connection is based on the ***n*-reachability** of each node, i.e., the number of nodes that they can reach.



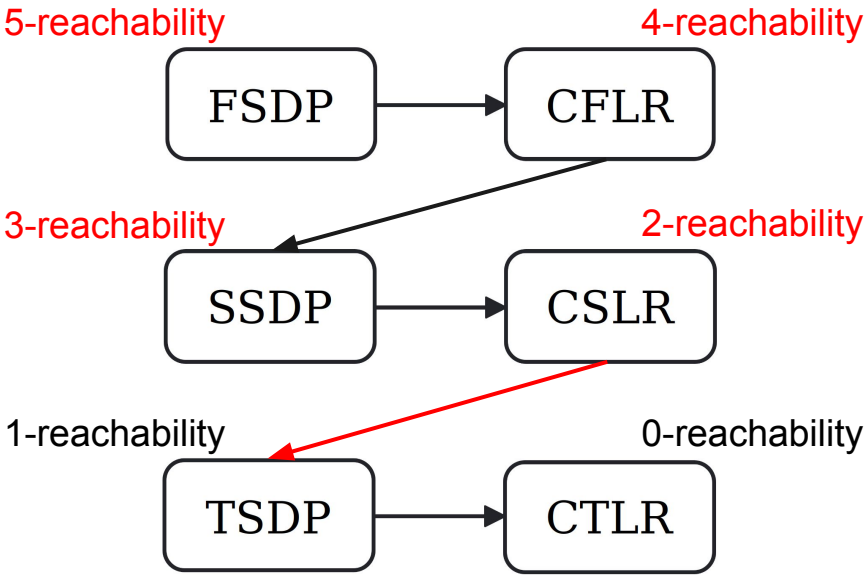
Each node of a component having a **0-reachability** must be **connected** to the node of **another component** having the **maximum reachability**, but this can be done in **any order**.



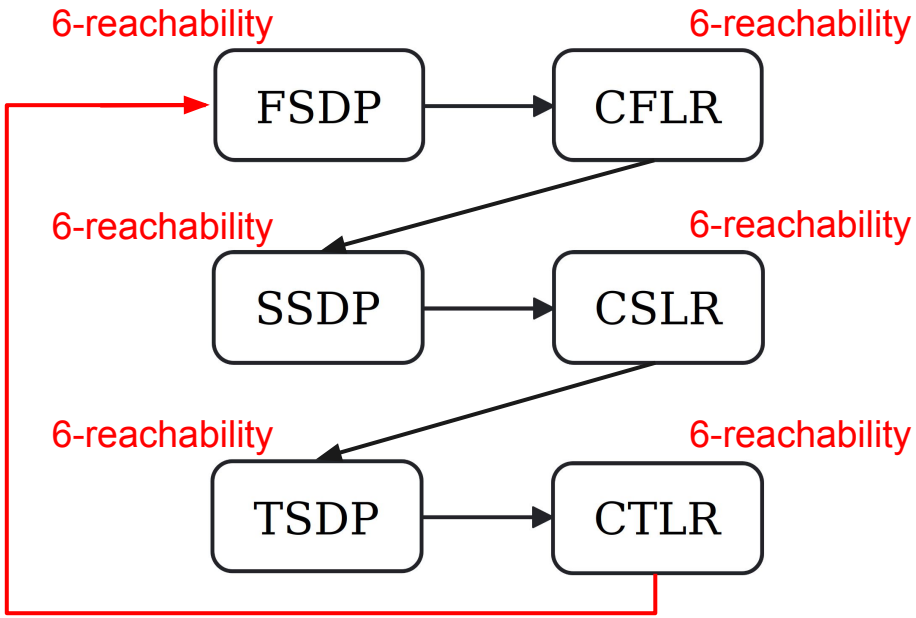
Each node of a component having a **0-reachability** must be **connected** to the node of **another component** having the **maximum reachability**, but this can be done in **any order**.



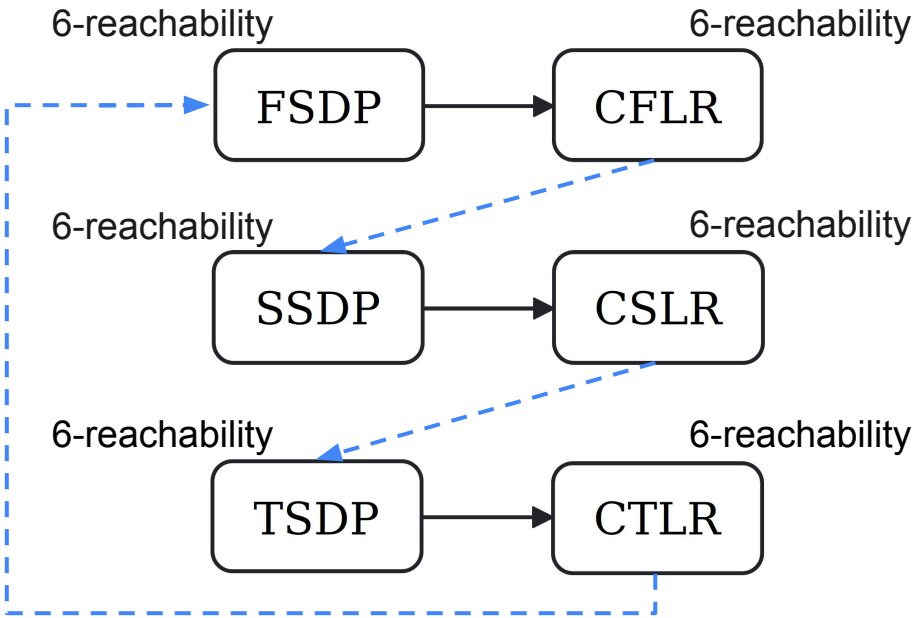
Each node of a component having a **0-reachability** must be **connected** to the node of **another component** having the **maximum reachability**, but this can be done in **any order**.



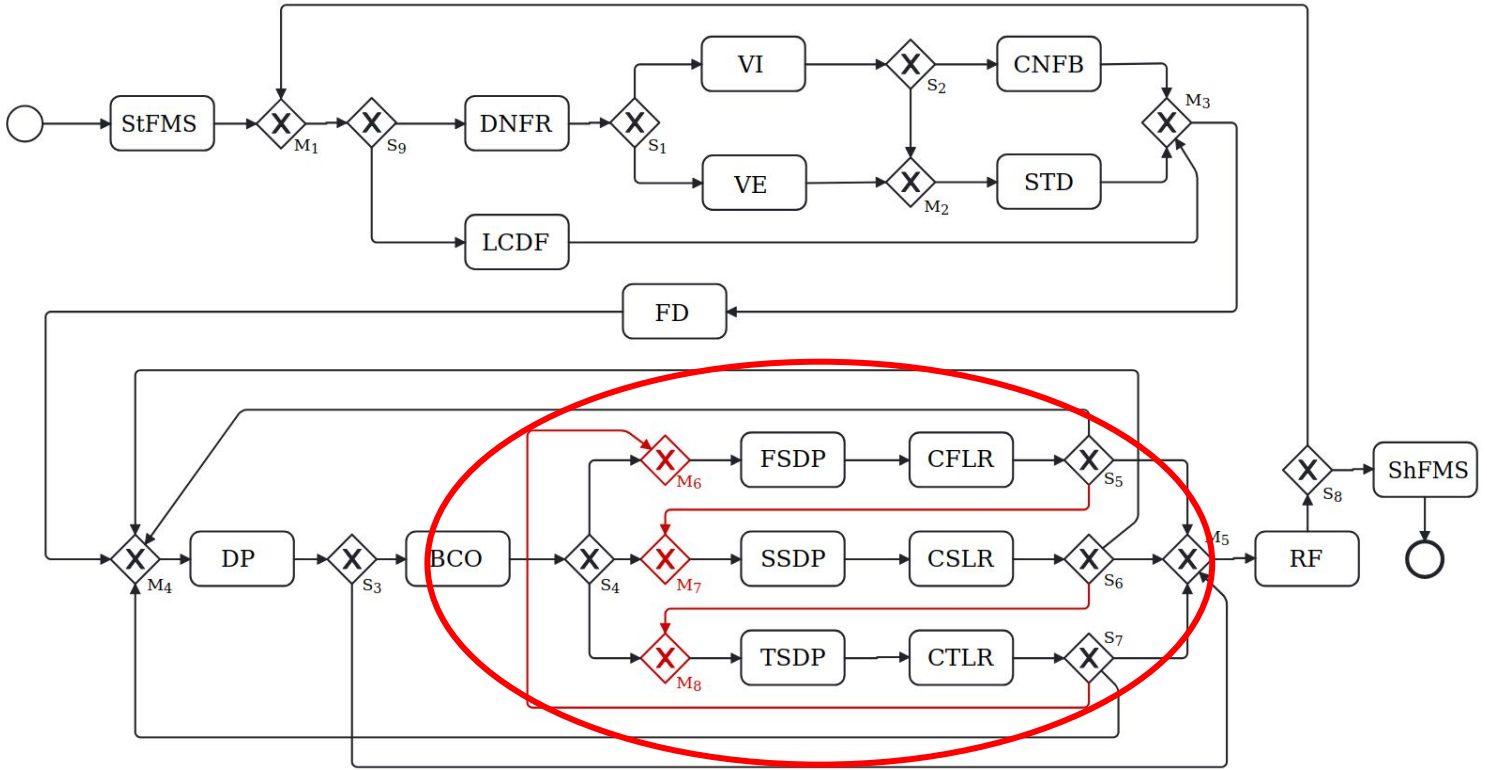
Each node of a component having a **0-reachability** must be **connected** to the node of **another component** having the **maximum reachability**, but this can be done in **any order**.



Each node of a component having a **0-reachability** must be **connected** to the node of **another component** having the **maximum reachability**, but this can be done in **any order**.

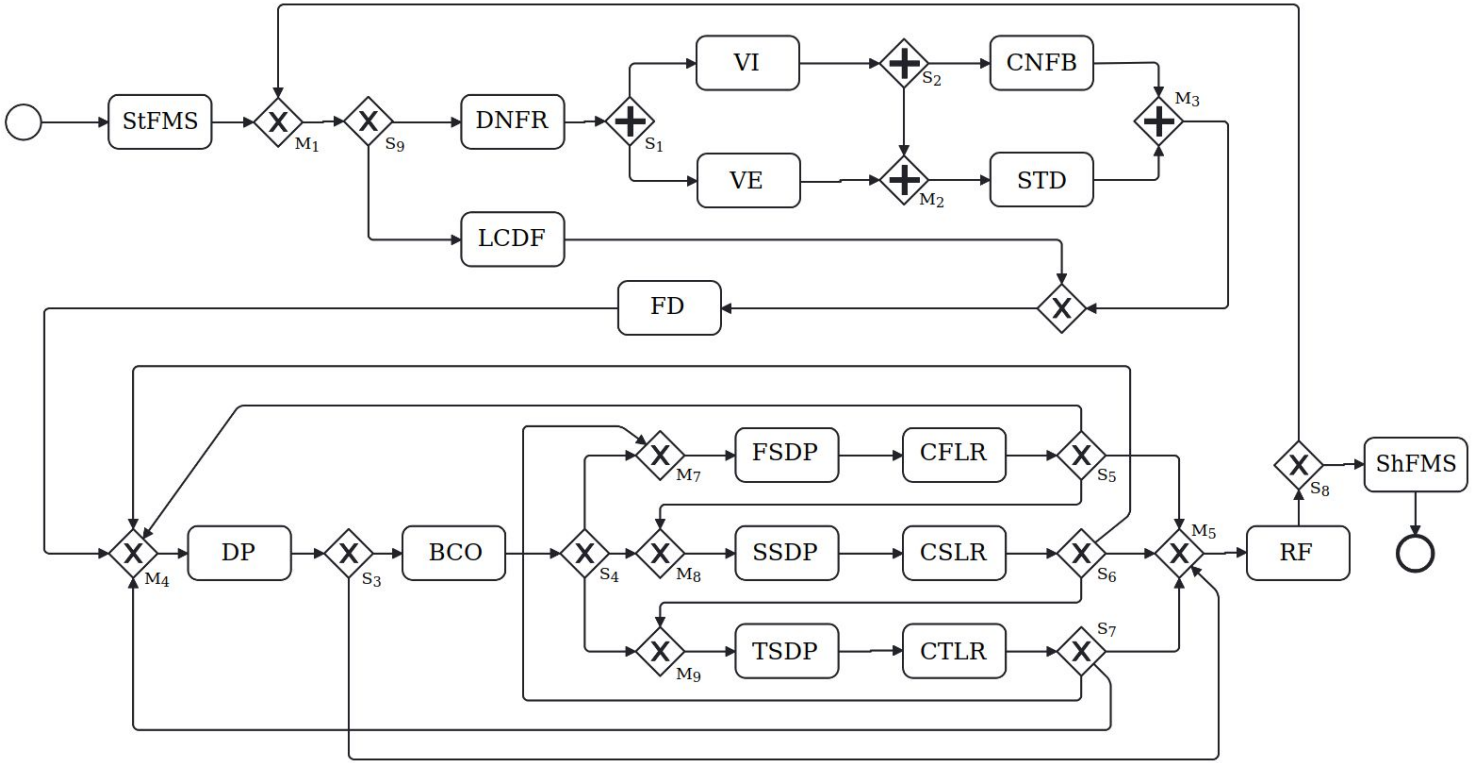


These **new edges** are then eventually **added to the BPMN** process to make the loop appear in it:



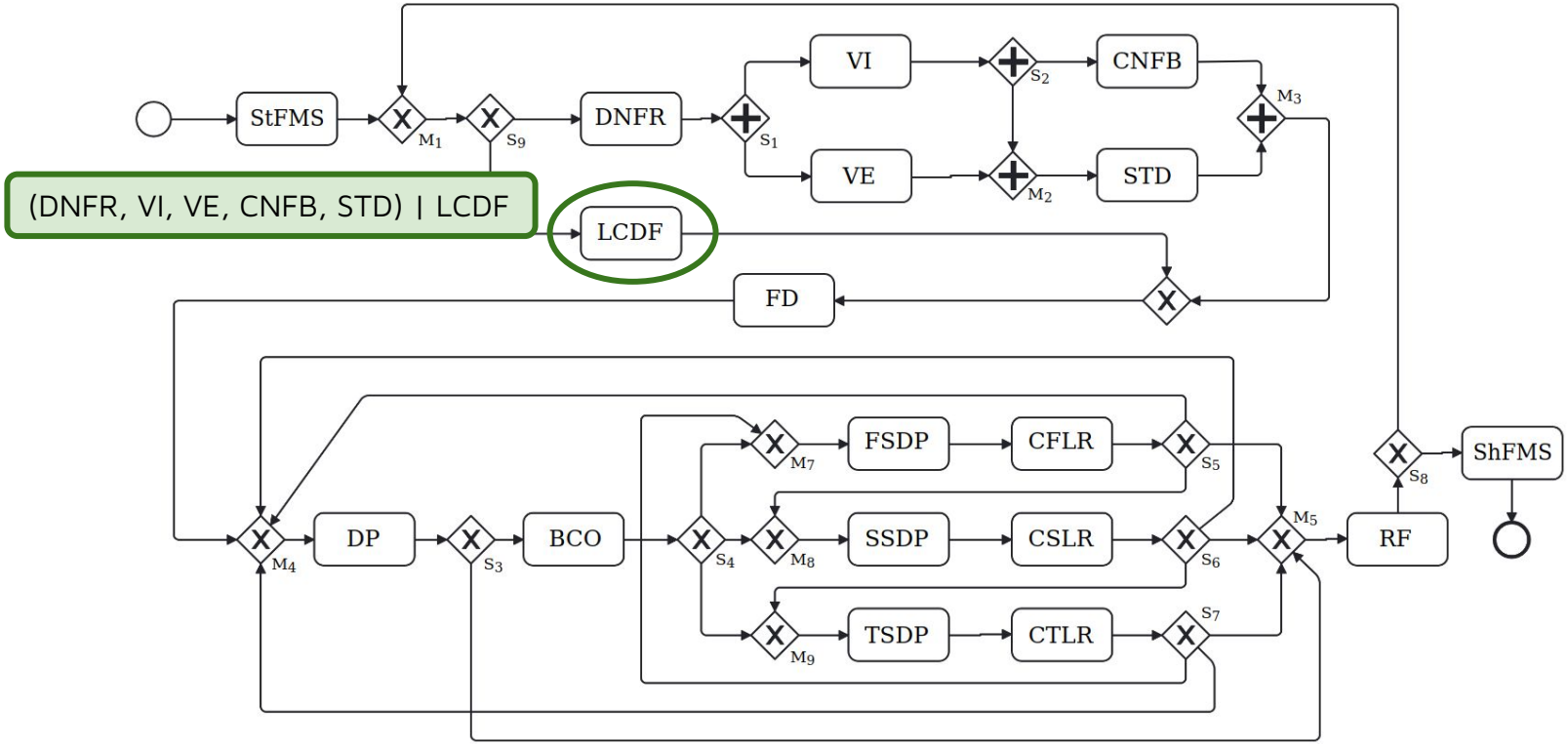
Detailed Approach – Final Process

After applying these successive refinement steps, the **process is complete**.



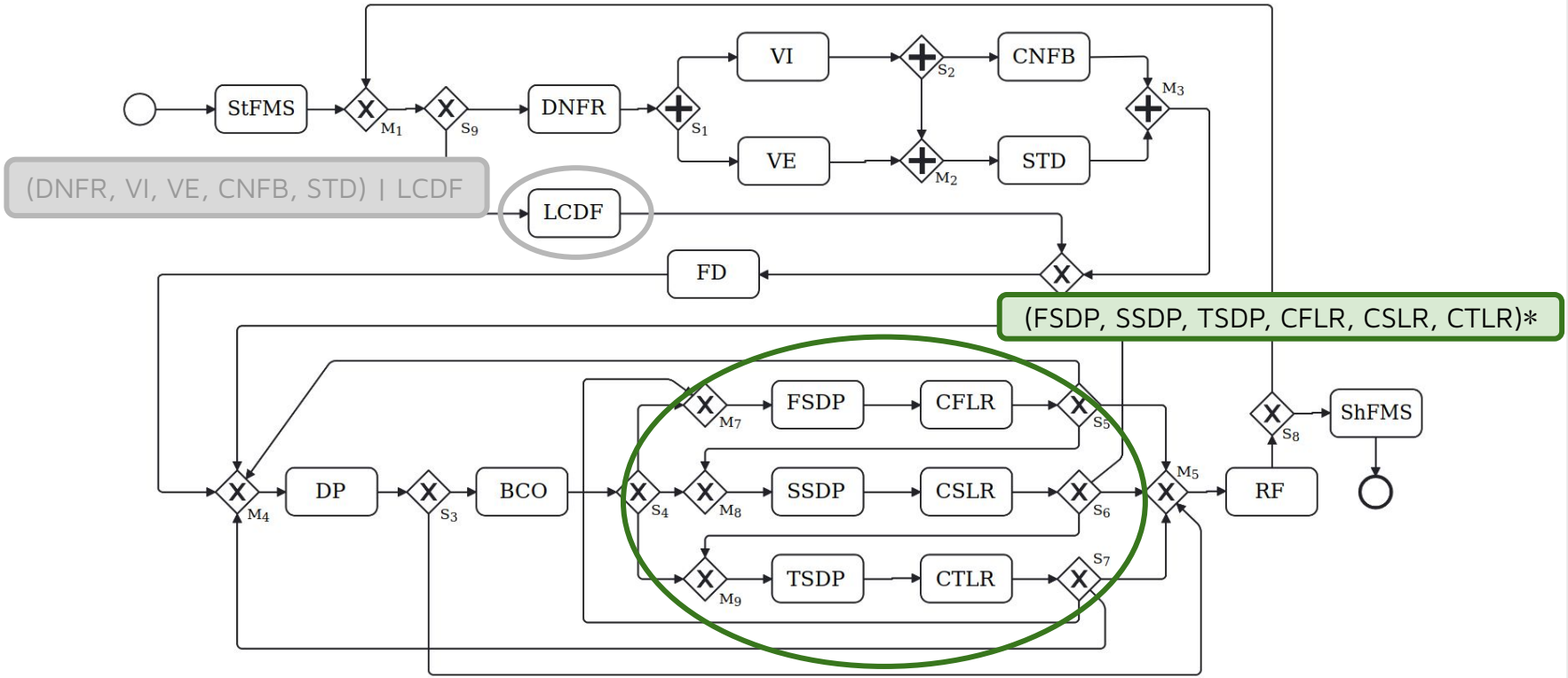
Detailed Approach – Final Process

After applying these successive refinement steps, the **process is complete**.



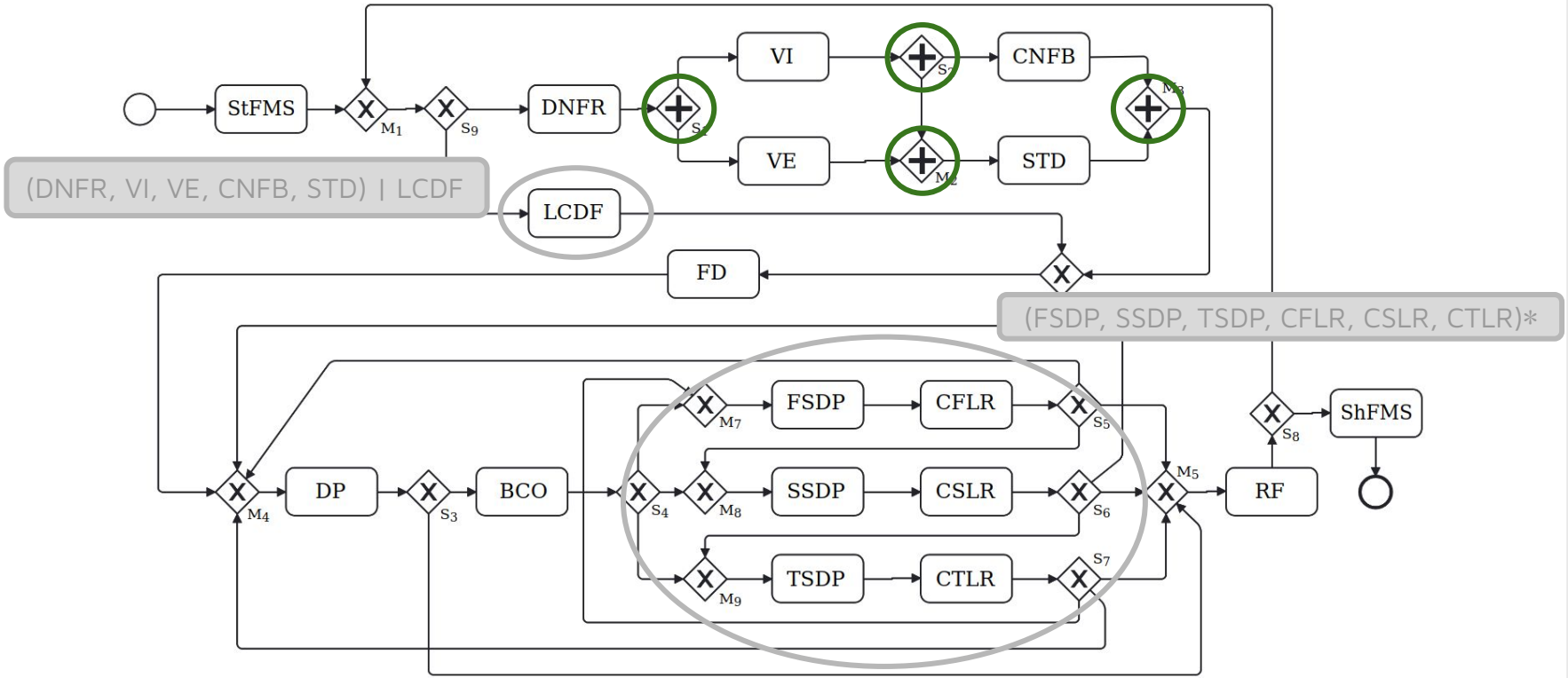
Detailed Approach – Final Process

After applying these successive refinement steps, the **process is complete**.



Detailed Approach – Final Process

After applying these successive refinement steps, the **process is complete**.

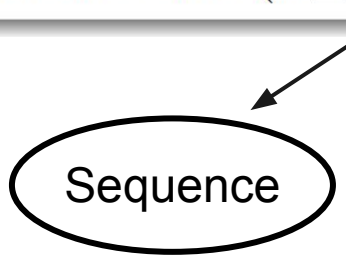


Theorem (Constraints Preservation)

Let $B = (V, E, \Sigma)$ be the BPMN process built from the sequential constraints Cons_1 and enriched with exclusive gateways and start/end events, and let Cons_2, Cons_3, and Cons_4 be the sets of constraints respectively satisfied by G after managing mutual exclusions, managing explicit loops, and inserting parallelism. We state that $(\emptyset \subseteq)\mathrm{Cons}_1 \subseteq \mathrm{Cons}_2 \subseteq \mathrm{Cons}_3 \subseteq \mathrm{Cons}_4$.

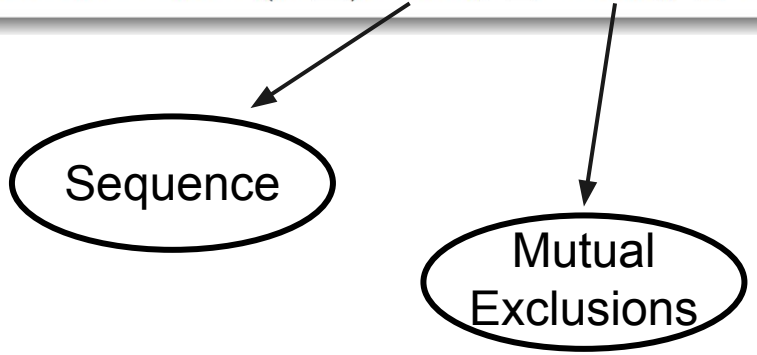
Theorem (Constraints Preservation)

Let $B = (V, E, \Sigma)$ be the BPMN process built from the sequential constraints Cons_1 and enriched with exclusive gateways and start/end events, and let Cons_2, Cons_3, and Cons_4 be the sets of constraints respectively satisfied by G after managing mutual exclusions, managing explicit loops, and inserting parallelism. We state that $(\emptyset \subseteq)\mathrm{Cons}_1 \subseteq \mathrm{Cons}_2 \subseteq \mathrm{Cons}_3 \subseteq \mathrm{Cons}_4$.



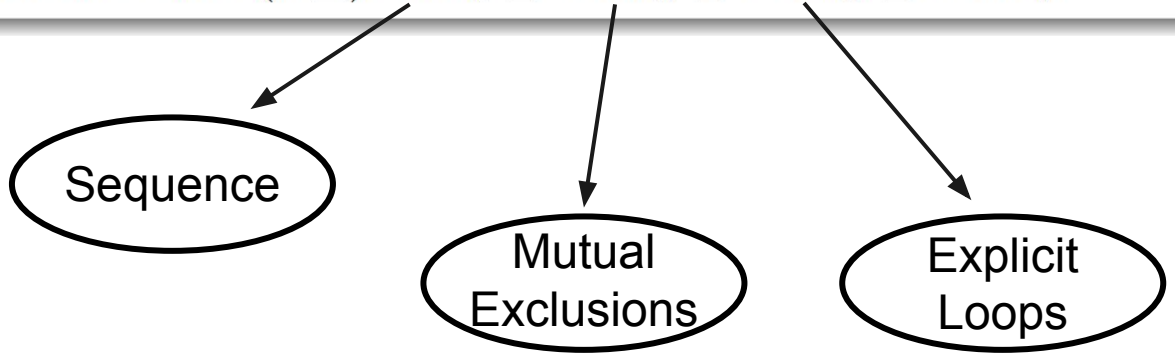
Theorem (Constraints Preservation)

Let $B = (V, E, \Sigma)$ be the BPMN process built from the sequential constraints Cons_1 and enriched with exclusive gateways and start/end events, and let Cons_2, Cons_3, and Cons_4 be the sets of constraints respectively satisfied by G after managing mutual exclusions, managing explicit loops, and inserting parallelism. We state that $(\emptyset \subseteq)\mathrm{Cons}_1 \subseteq \mathrm{Cons}_2 \subseteq \mathrm{Cons}_3 \subseteq \mathrm{Cons}_4$.



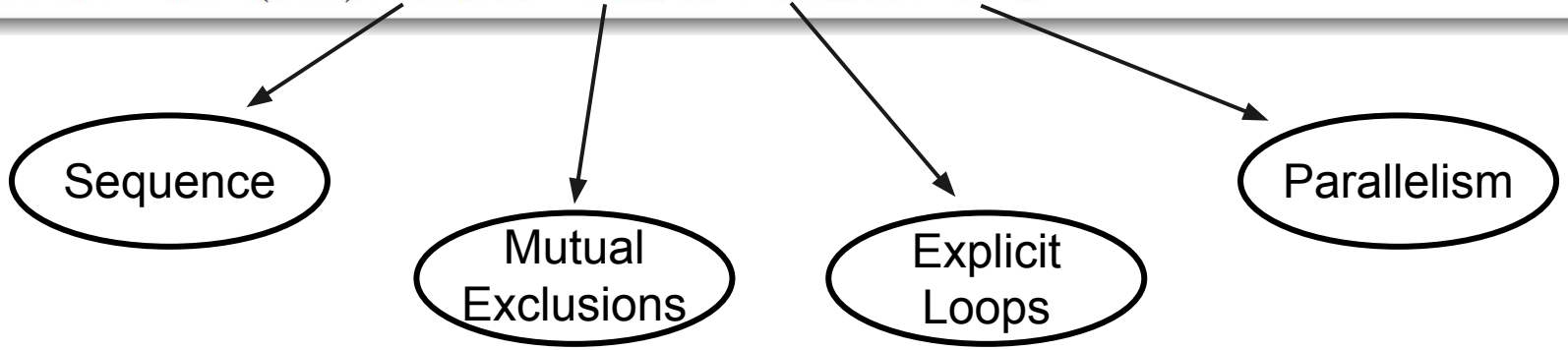
Theorem (Constraints Preservation)

Let $B = (V, E, \Sigma)$ be the BPMN process built from the sequential constraints Cons_1 and enriched with exclusive gateways and start/end events, and let Cons_2, Cons_3, and Cons_4 be the sets of constraints respectively satisfied by G after managing mutual exclusions, managing explicit loops, and inserting parallelism. We state that $(\emptyset \subseteq)\mathrm{Cons}_1 \subseteq \mathrm{Cons}_2 \subseteq \mathrm{Cons}_3 \subseteq \mathrm{Cons}_4$.



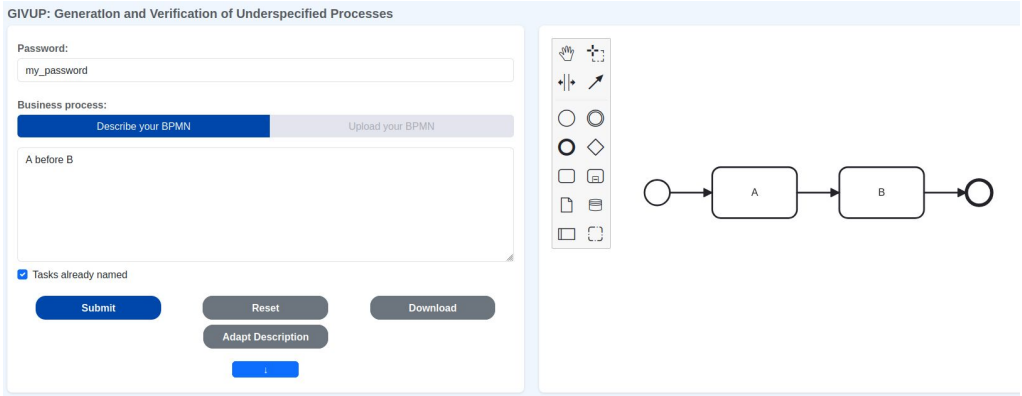
Theorem (Constraints Preservation)

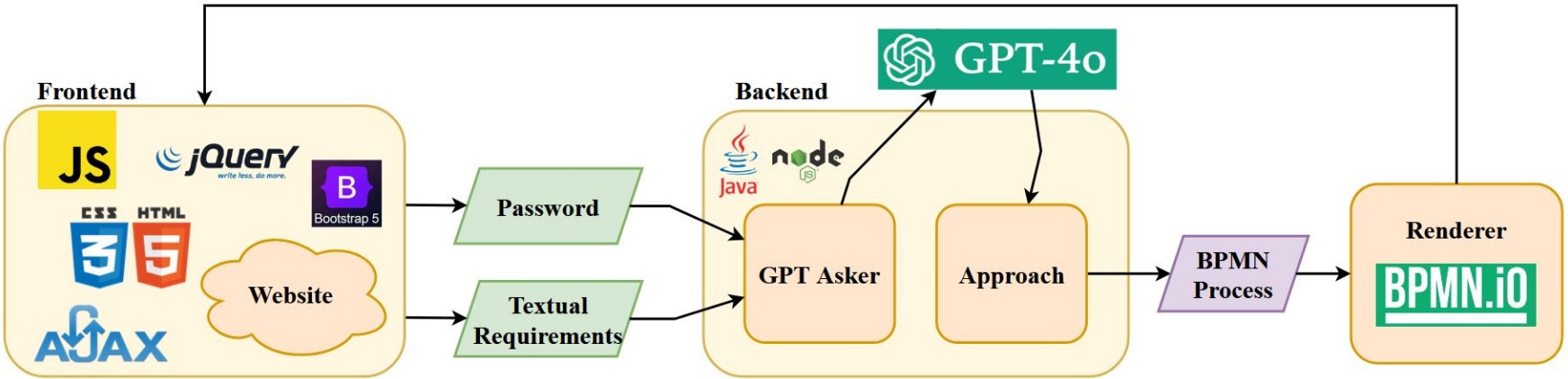
Let $B = (V, E, \Sigma)$ be the BPMN process built from the sequential constraints Cons_1 and enriched with exclusive gateways and start/end events, and let Cons_2, Cons_3, and Cons_4 be the sets of constraints respectively satisfied by G after managing mutual exclusions, managing explicit loops, and inserting parallelism. We state that $(\emptyset \subseteq)\mathrm{Cons}_1 \subseteq \mathrm{Cons}_2 \subseteq \mathrm{Cons}_3 \subseteq \mathrm{Cons}_4$.



Tool Support

- **12k lines of Java** code
- **Tool** available **online** (https://lig-givup.imag.fr/)





Experiments – Quality of the Generated 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 Generated 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 Generated 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 Generated 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 Generated 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.

Plan

I/ Introduction

II/ Automated Generation of BPMN Processes from Textual Requirements

III/ Human-Centered Refactoring-Based Optimisation of BPMN Processes

IV/ Related Work

V/ Takeaways

VI/ References

- **Operational research** deals with analytical problem solving

- **Operational research** deals with analytical problem solving
  ⇒ **no analytical representation** of our problem

- **Operational research** deals with analytical problem solving
  ⇒ **no analytical representation** of our problem

- **Resource balancing** allows to adapt the pool of resources

- **Operational research** deals with analytical problem solving
  ⇒ **no analytical representation** of our problem

- **Resource balancing** allows to adapt the pool of resources
  ⇒ requires **flexibility** in the available **resources**

- **Operational research** deals with analytical problem solving
  ⇒ **no analytical representation** of our problem

- **Resource balancing** allows to adapt the pool of resources
  ⇒ requires **flexibility** in the available **resources**

- **Scheduling** permits to modify order of execution of the tasks

- **Operational research** deals with analytical problem solving
  ⇒ **no analytical representation** of our problem

- **Resource balancing** allows to adapt the pool of resources
  ⇒ requires **flexibility** in the available **resources**

- **Scheduling** permits to modify order of execution of the tasks
  ⇒ **does not solve** inherent **structural issues**

- **Operational research** deals with analytical problem solving
  ⇒ **no analytical representation** of our problem

- **Resource balancing** allows to adapt the pool of resources
  ⇒ requires **flexibility** in the available **resources**

- **Scheduling** permits to modify order of execution of the tasks
  ⇒ **does not solve** inherent **structural issues**

- **Process refactoring** modifies the structure of the process with the goal of optimising it

- **Operational research** deals with analytical problem solving
  ⇒ **no analytical representation** of our problem

- **Resource balancing** allows to adapt the pool of resources
  ⇒ requires **flexibility** in the available **resources**

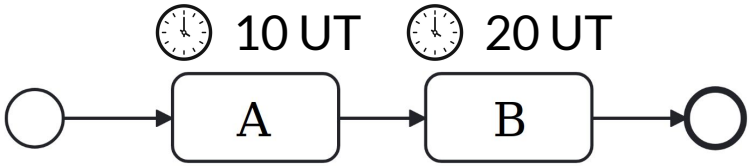
- **Scheduling** permits to modify order of execution of the tasks
  ⇒ **does not solve** inherent **structural issues**

- **Process refactoring** modifies the structure of the process with the goal of optimising it

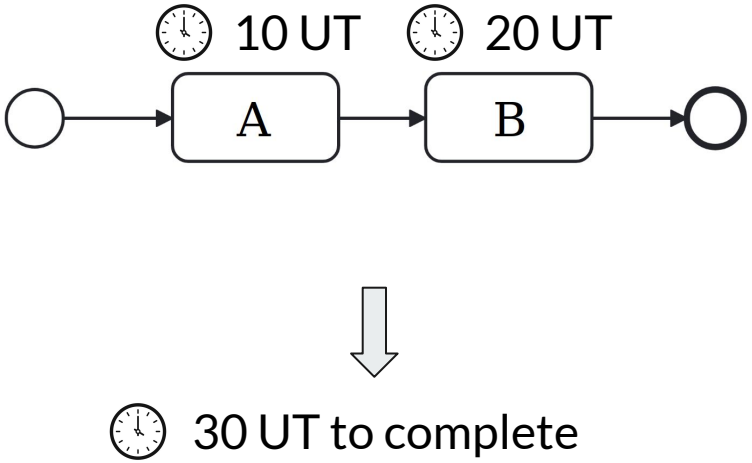
Process Refactoring

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

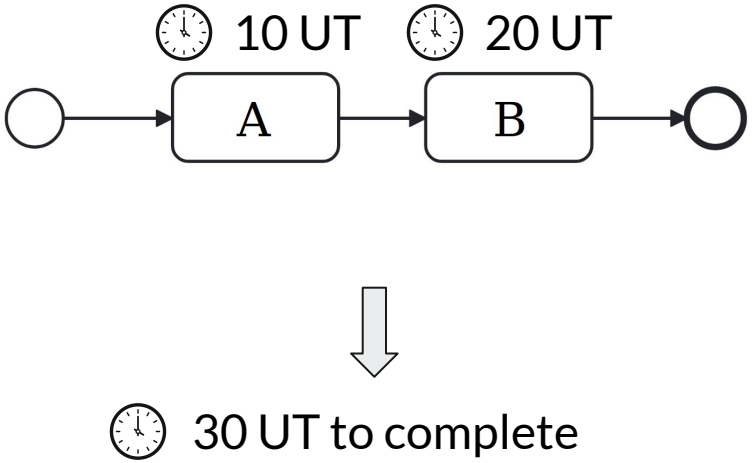
Refactoring a process consists in **modifying the order** in which the tasks are organised, in order to **optimise** the process with regards to some quantitative criteria (**execution time**, resources usage, costs, etc.).

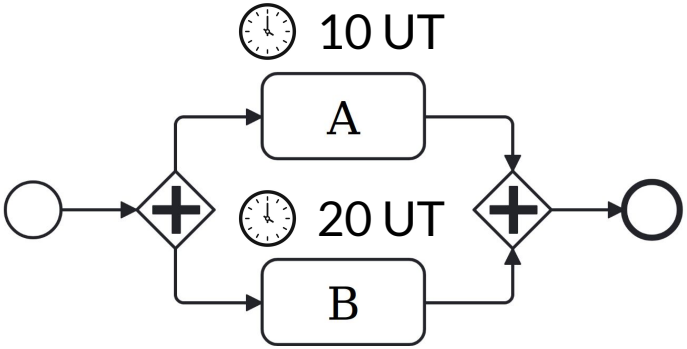


Refactoring a process consists in **modifying the order** in which the tasks are organised, in order to **optimise** the process with regards to some quantitative criteria (**execution time**, resources usage, costs, etc.).

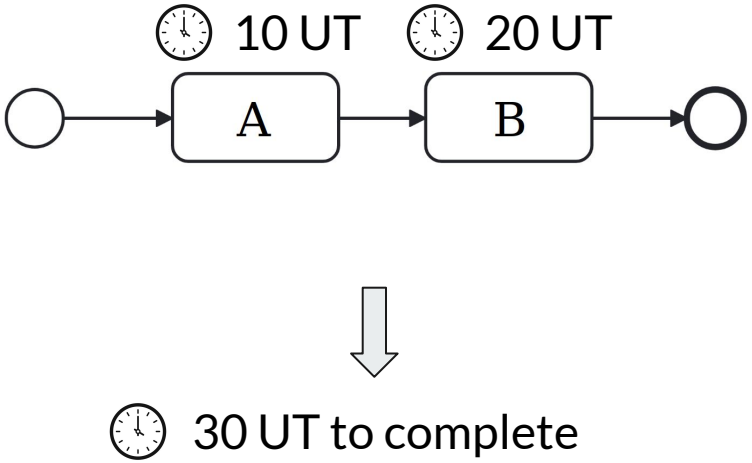


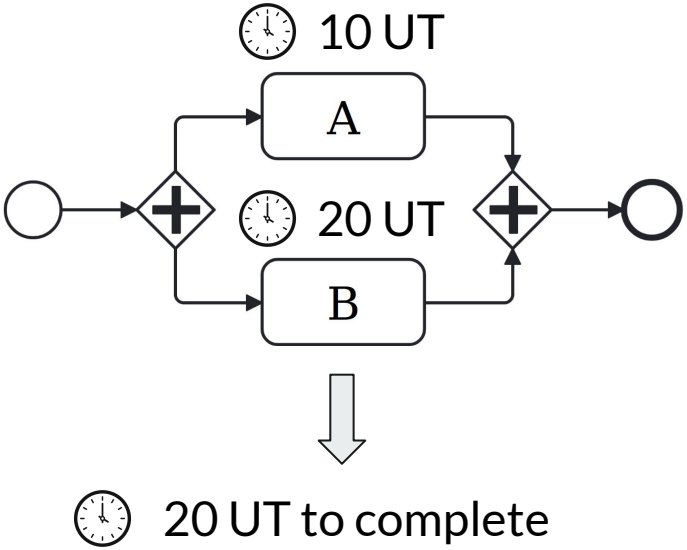
Refactoring a process consists in **modifying the order** in which the tasks are organised, in order to **optimise** the process with regards to some quantitative criteria (**execution time**, resources usage, costs, etc.).





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





Parallelising: an obvious solution?

However, **parallelising** a process **does not** necessarily **optimise** it!

Parallelising: an obvious solution?

However, **parallelising** a process **does not** necessarily **optimise** it!

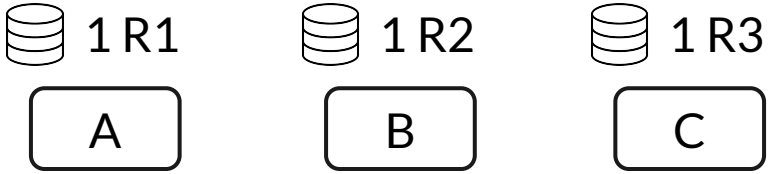
A

B

C

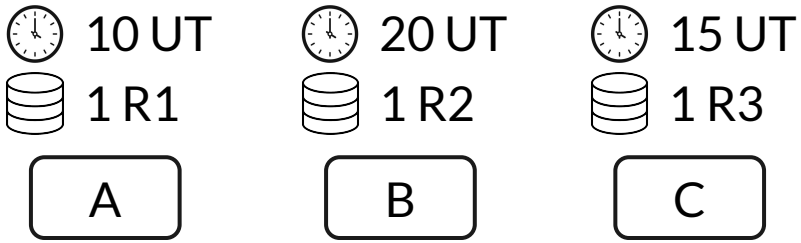
Parallelising: an obvious solution?

However, **parallelising** a process **does not** necessarily **optimise** it!



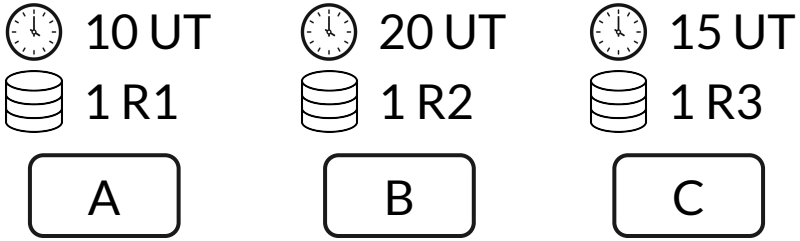
Parallelising: an obvious solution?

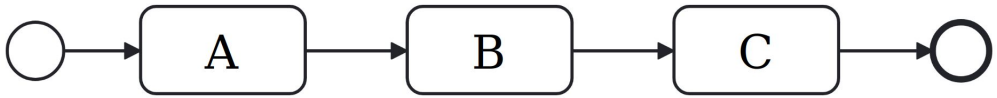
However, **parallelising** a process **does not** necessarily **optimise** it!

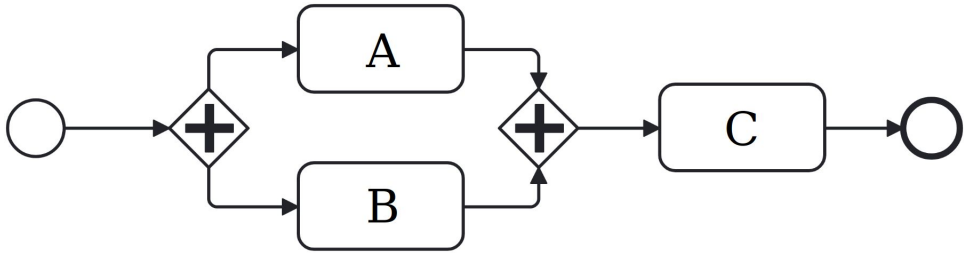


Parallelising: an obvious solution?

However, **parallelising** a process **does not** necessarily **optimise** it!

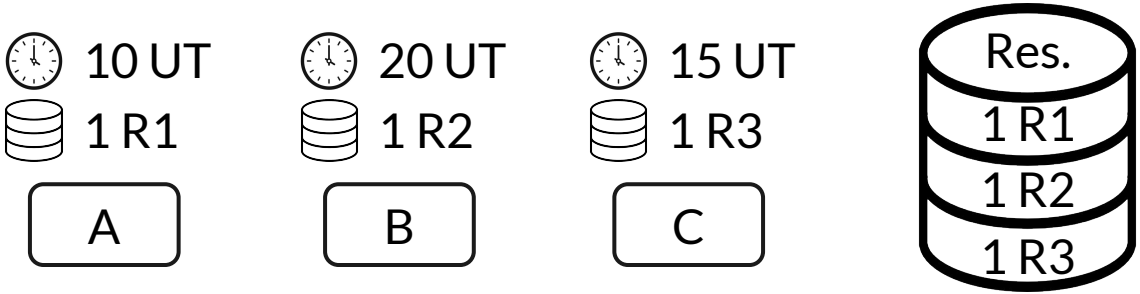


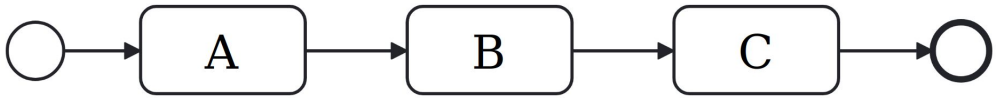


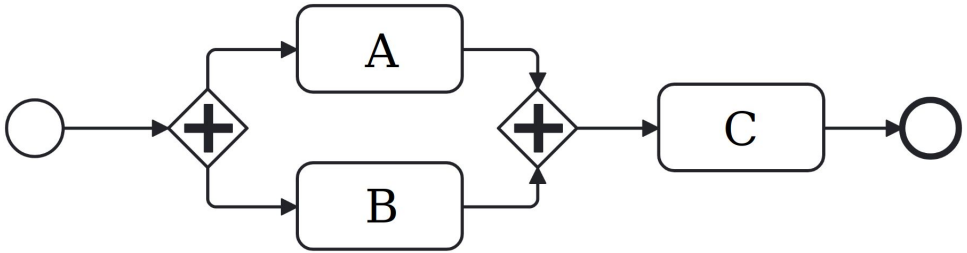


Parallelising: an obvious solution?

However, **parallelising** a process **does not** necessarily **optimise** it!

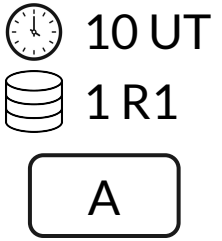


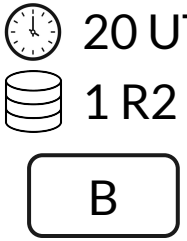


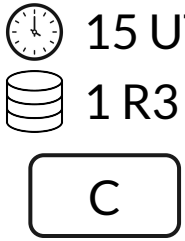


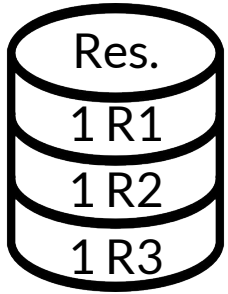
Parallelising: an obvious solution?

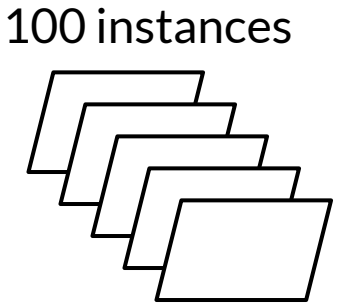
However, **parallelising** a process **does not** necessarily **optimise** it!

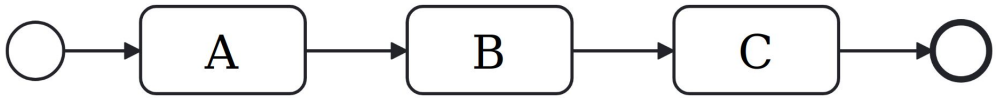


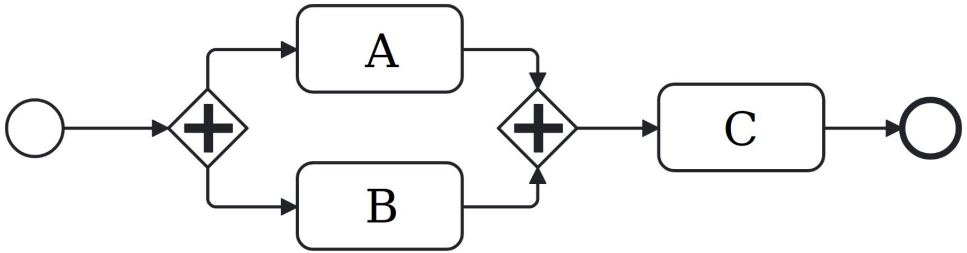






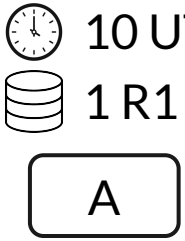


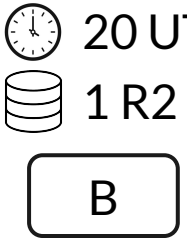


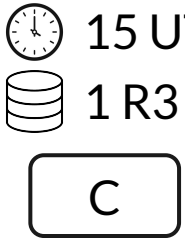


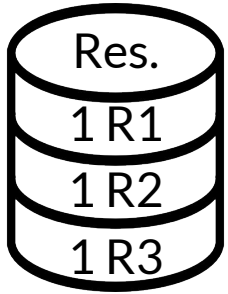
Parallelising: an obvious solution?

However, **parallelising** a process **does not** necessarily **optimise** it!

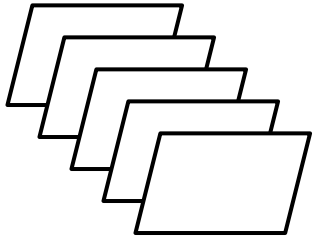


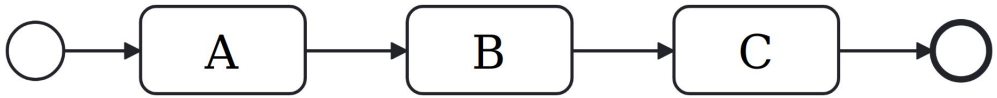




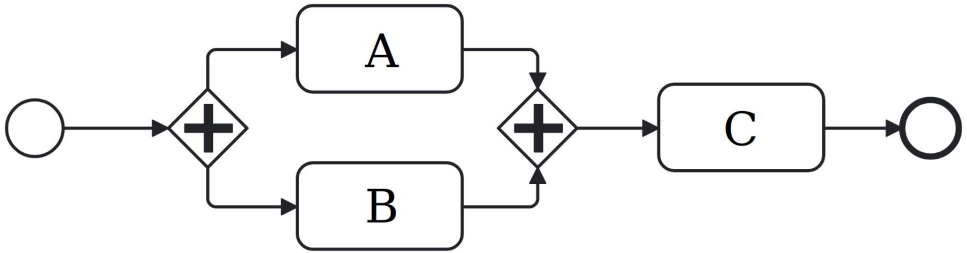


100 instances





AET = 787 UT



AET = 838 UT

Refactoring-related problematics

- How can you **guide** the refactoring to **optimise** the process?

Refactoring-related problematics

- How can you **guide** the refactoring to **optimise** the process?
- How can you **preserve the logic/meaning** of the original process?

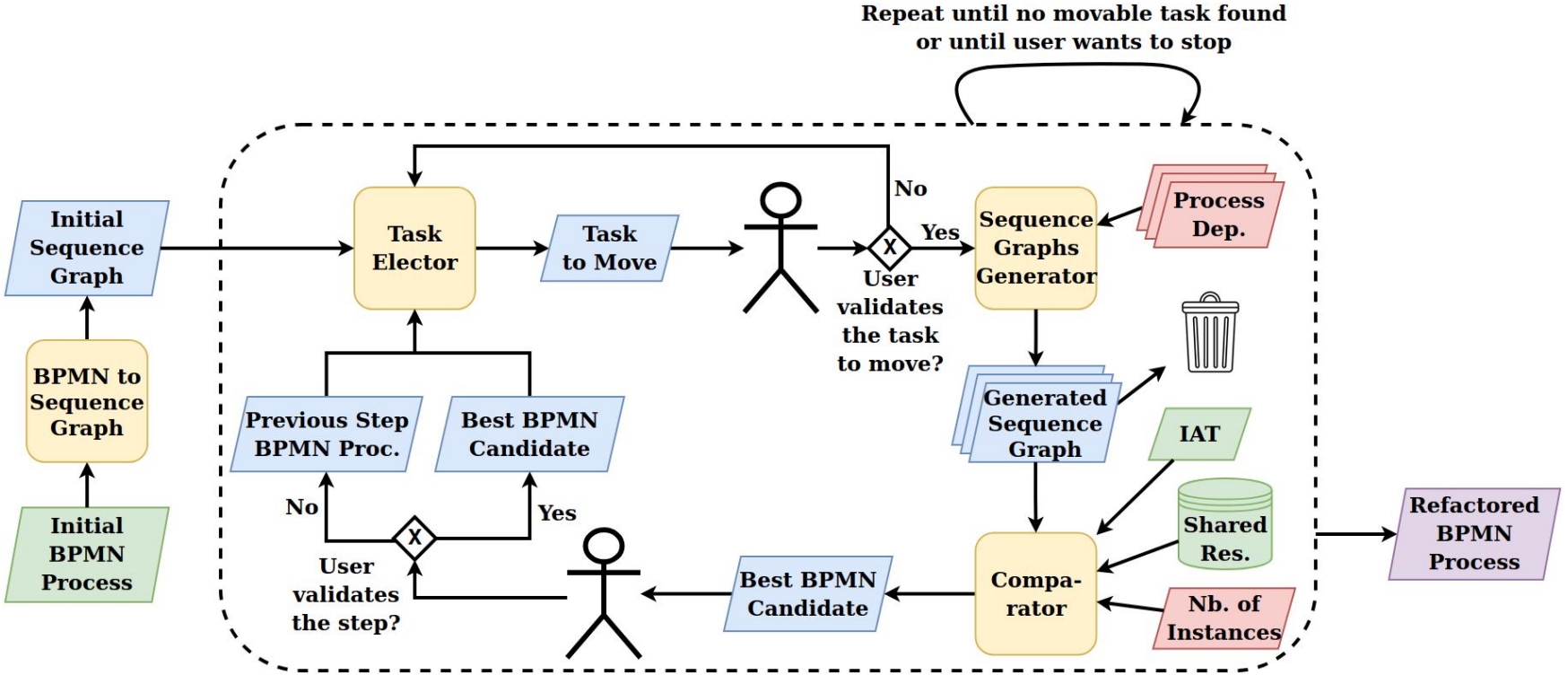
Refactoring-related problematics

- How can you **guide** the refactoring to **optimise** the process?
- How can you **preserve the logic/meaning** of the original process?
- How can you **preserve the structural semantics** of the original process?

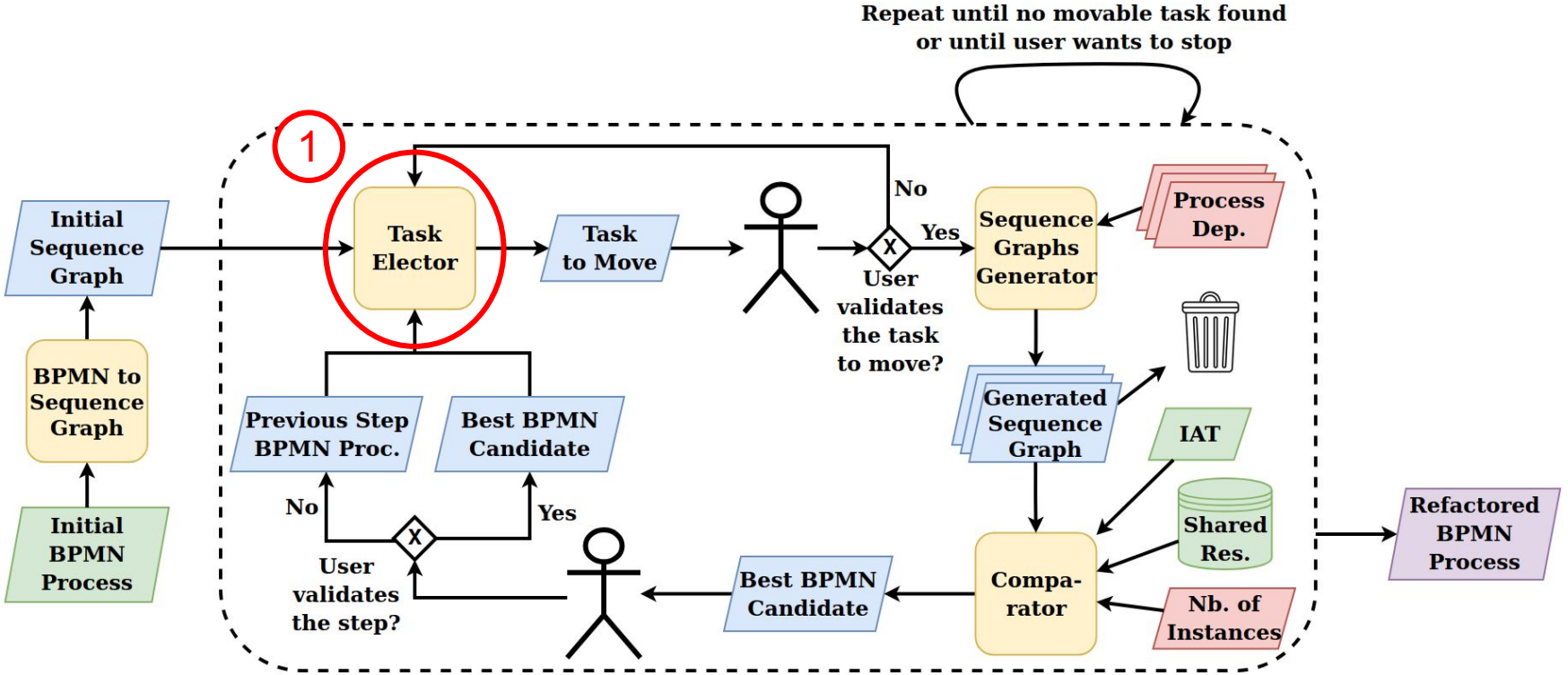
Refactoring-related problematics

- How can you **guide** the refactoring to **optimise** the process?
- How can you **preserve the logic/meaning** of the original process?
- How can you **preserve the structural semantics** of the original process?
- How can you **maximise the chances** of the user to **understand** the refactored **process**?

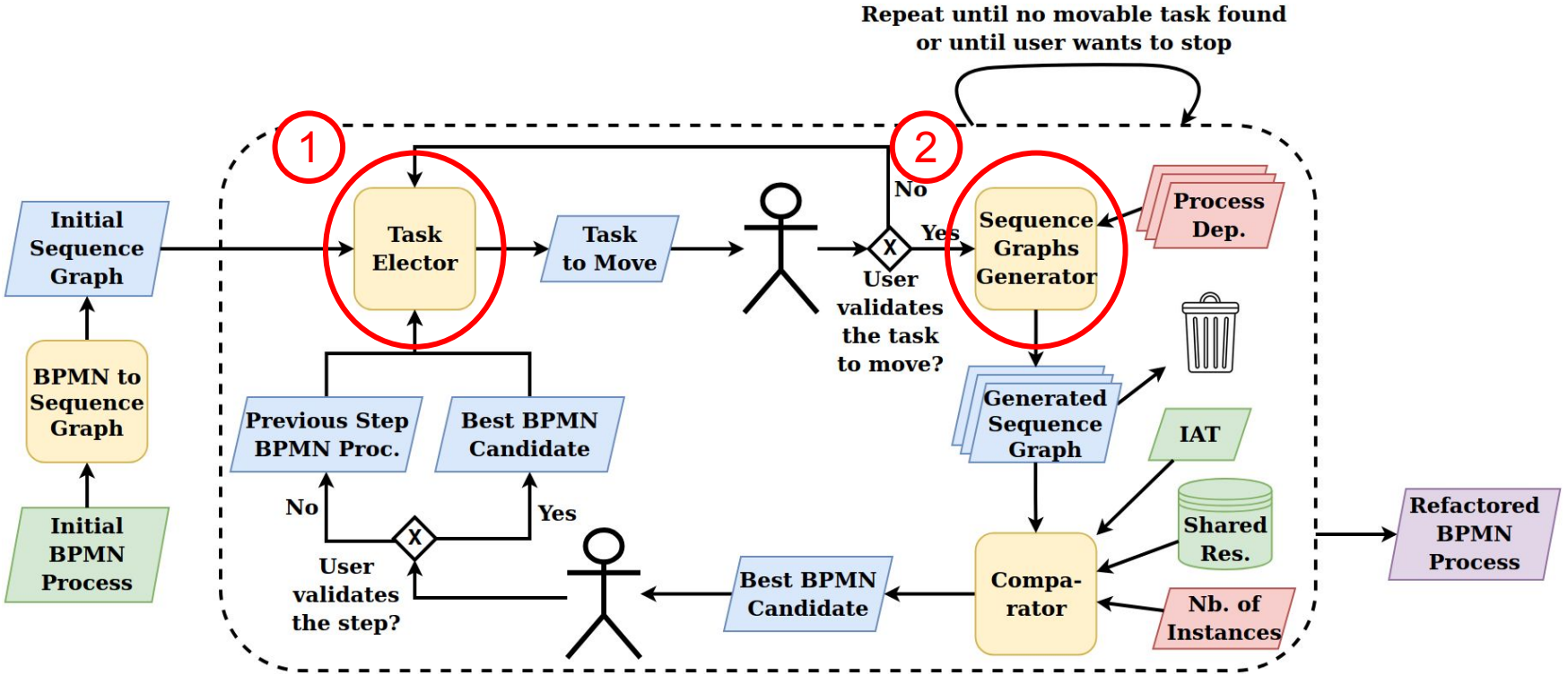
Global Picture of the Approach



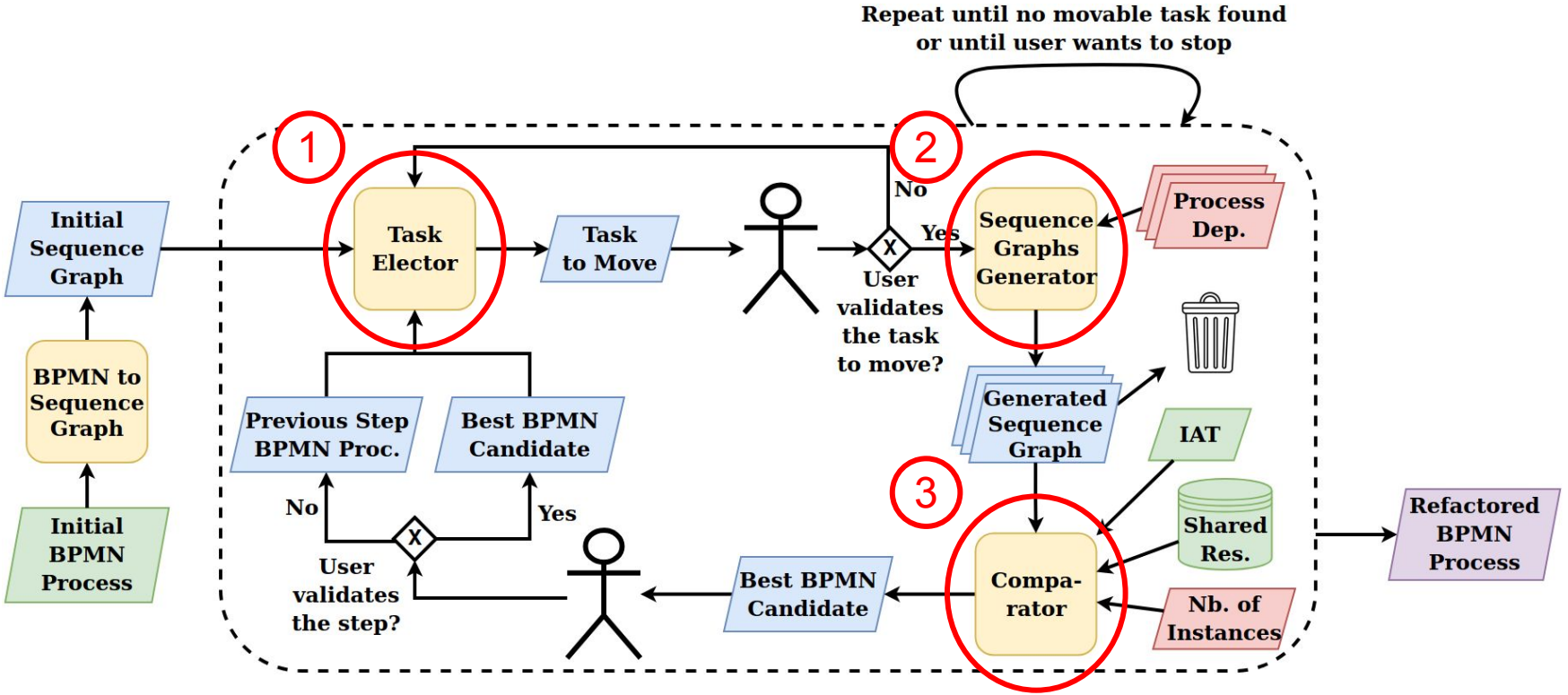
Global Picture of the Approach



Global Picture of the Approach

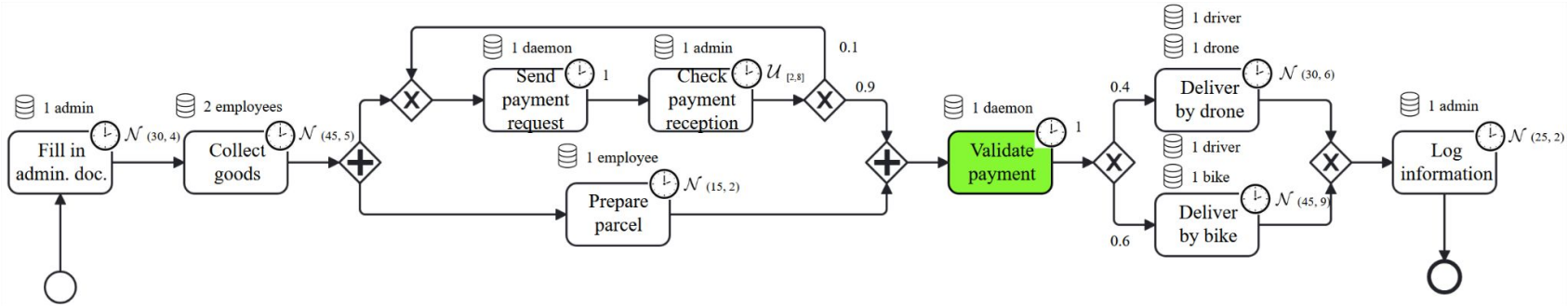


Global Picture of the Approach



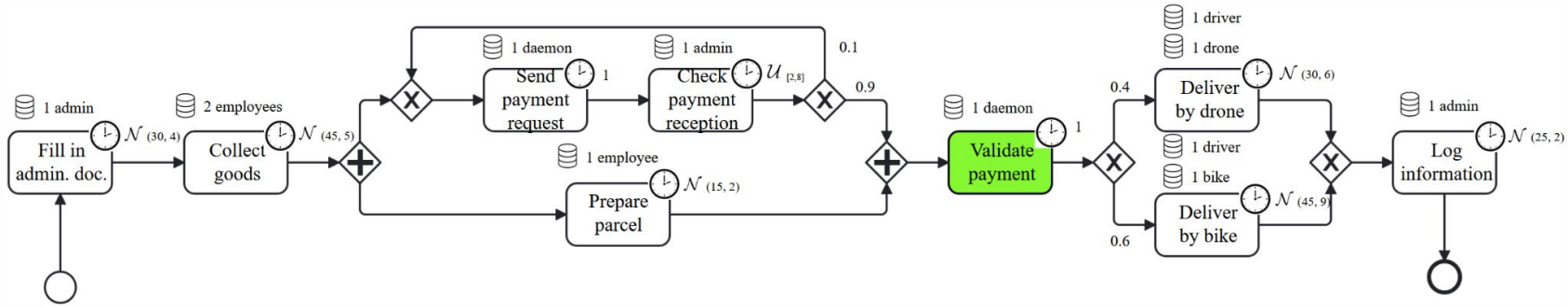
Step 1 – Election of the Task to Move

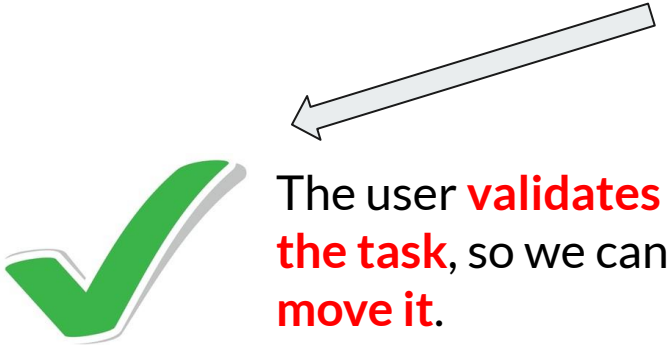
The **first step** consists in **proposing a task to move** to the user.



Step 1 – Election of the Task to Move

The **first step** consists in **proposing a task to move** to the user.

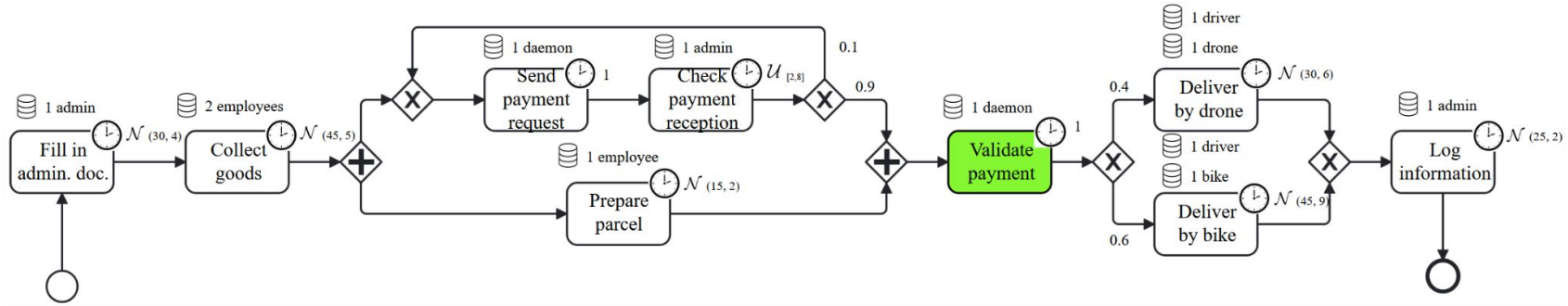




The user **validates the task**, so we can **move it**.

Step 1 – Election of the Task to Move

The **first step** consists in **proposing a task to move** to the user.



The user **validates the task**, so we can **move it**.

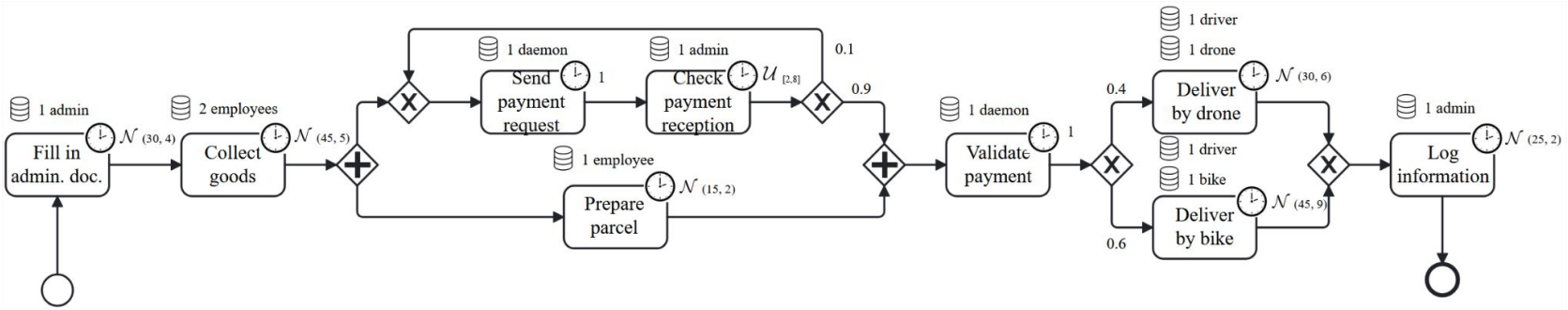
The user **declines the task**, so we **propose a new task** to move.

Step 2 – Relocation of the Task

The **relocation of the task** must preserve the **structural semantics** of the process.

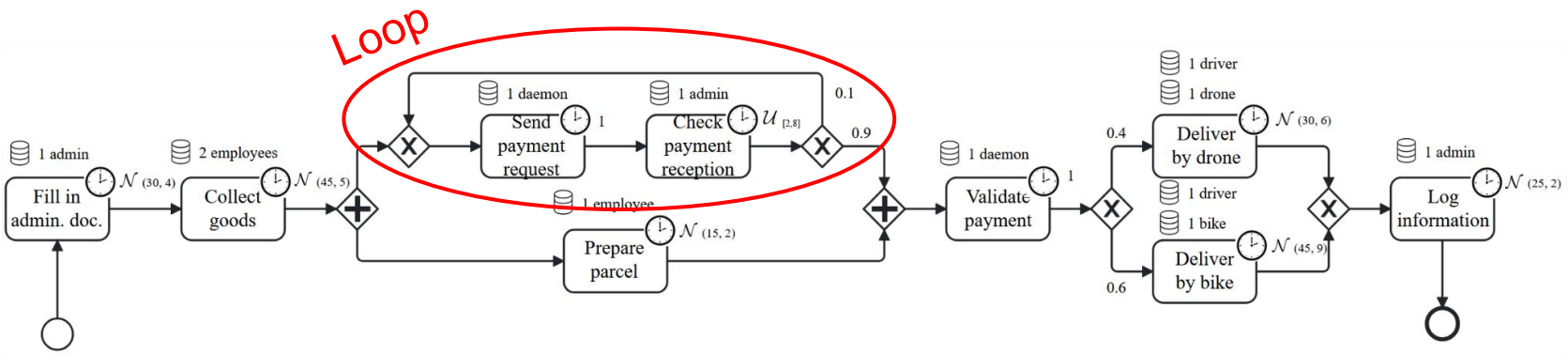
Step 2 – Relocation of the Task

The **relocation of the task** must preserve the **structural semantics** of the process.



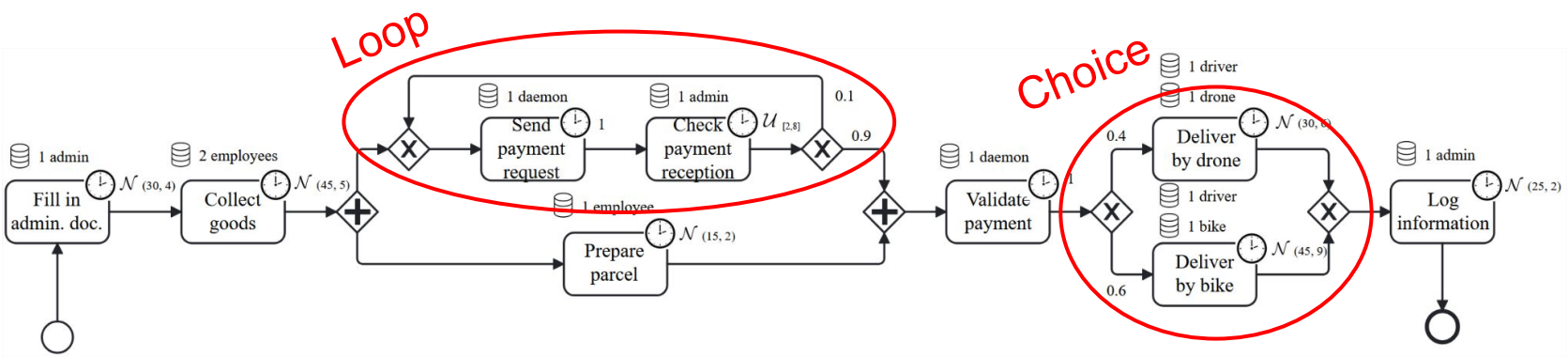
Step 2 – Relocation of the Task

The **relocation of the task** must preserve the **structural semantics** of the process.



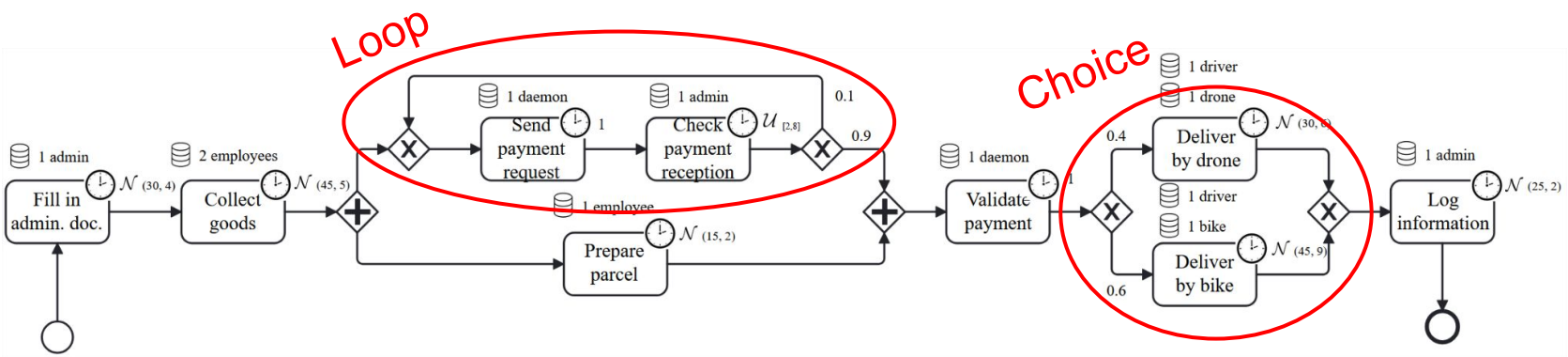
Step 2 – Relocation of the Task

The **relocation of the task** must preserve the **structural semantics** of the process.



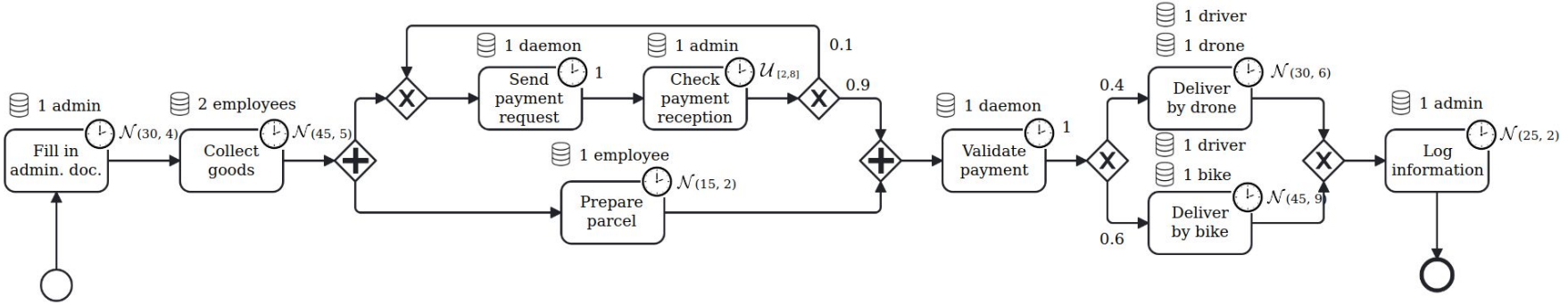
Step 2 – Relocation of the Task

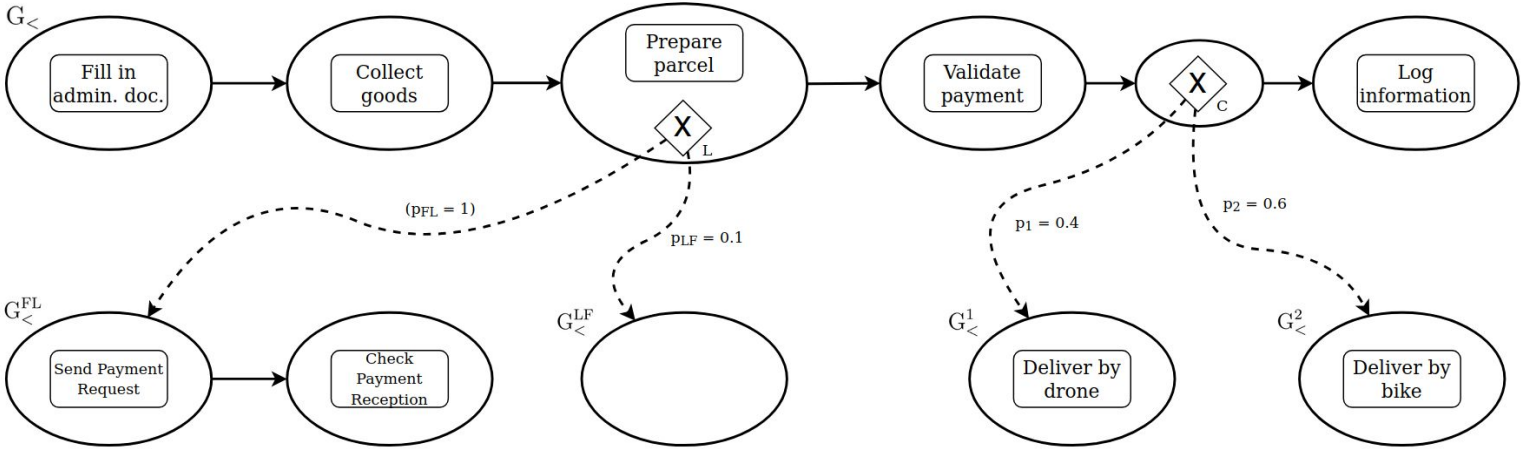
The **relocation of the task** must preserve the **structural semantics** of the process.



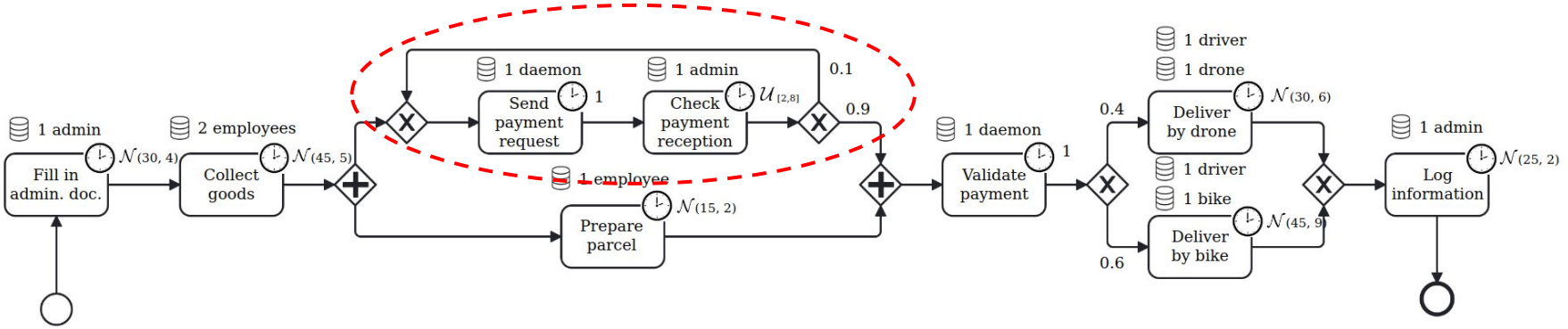
To **facilitate** this **preservation**, the relocation of the task is **not performed** on the **BPMN process**, **but on** another representation, called **sequence graph**.

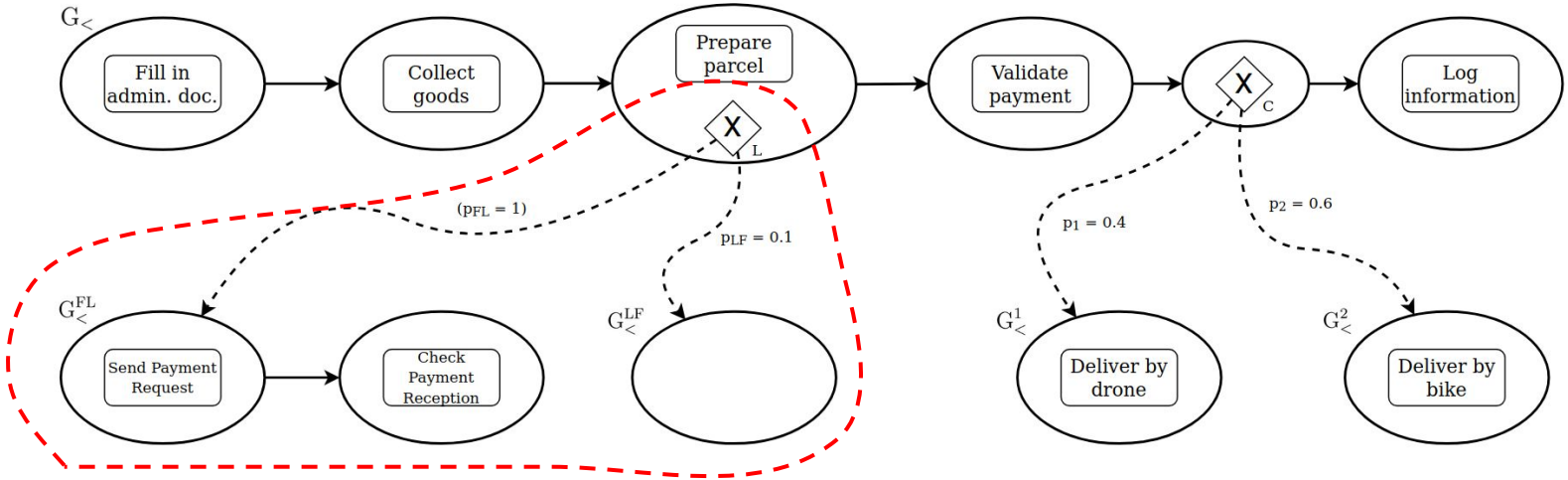
Step 2 – BPMN to Sequence Graph



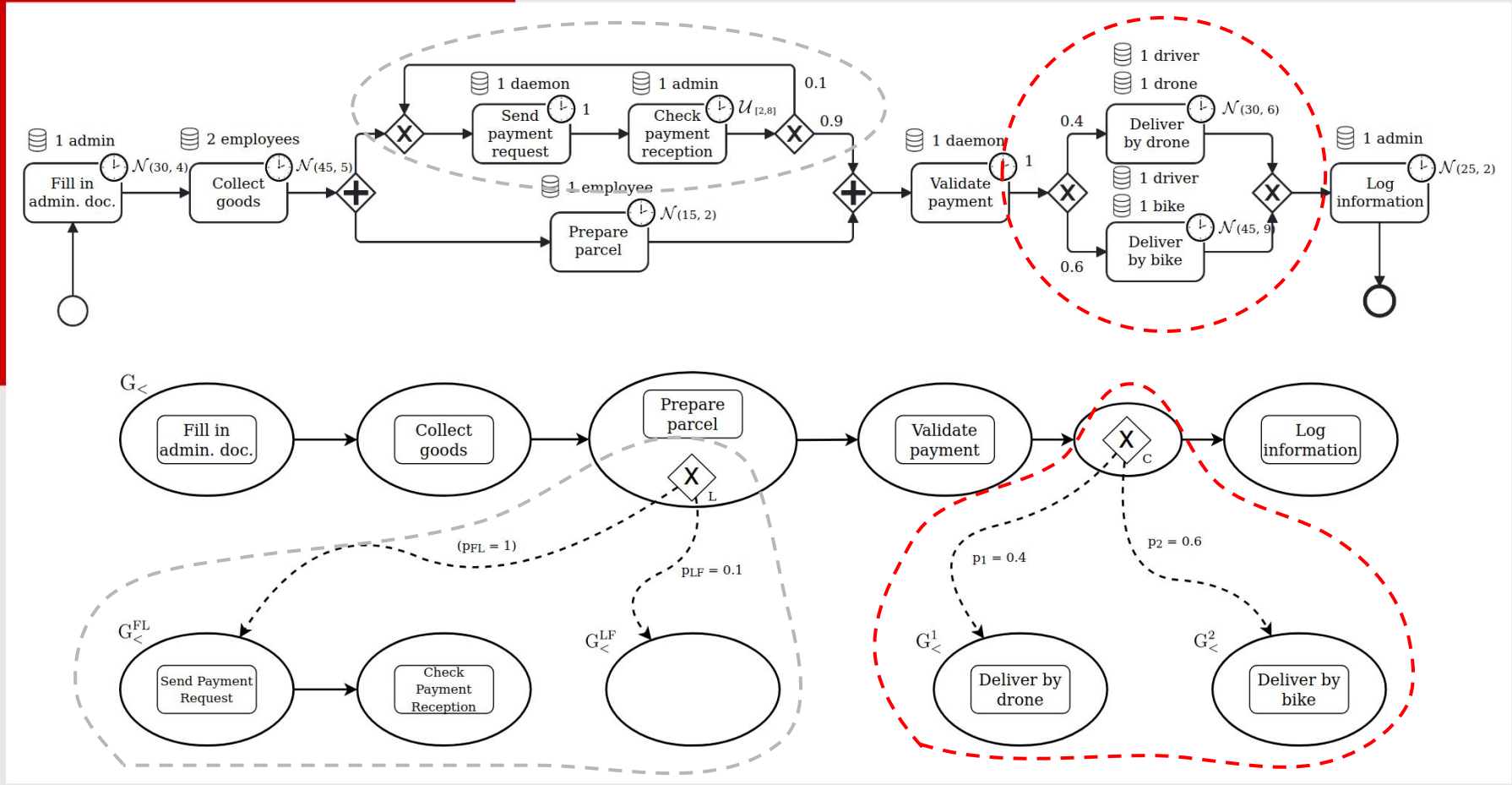


Step 2 – BPMN to Sequence Graph





Step 2 – BPMN to Sequence Graph



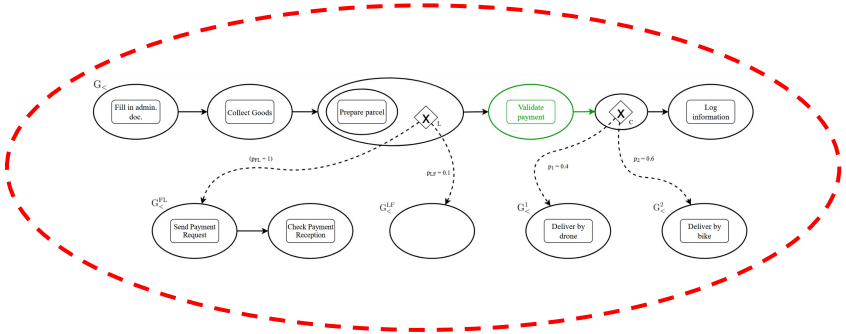
Step 2 – Refactoring Patterns

The selected task is **moved** thanks to **4 refactoring patterns**.

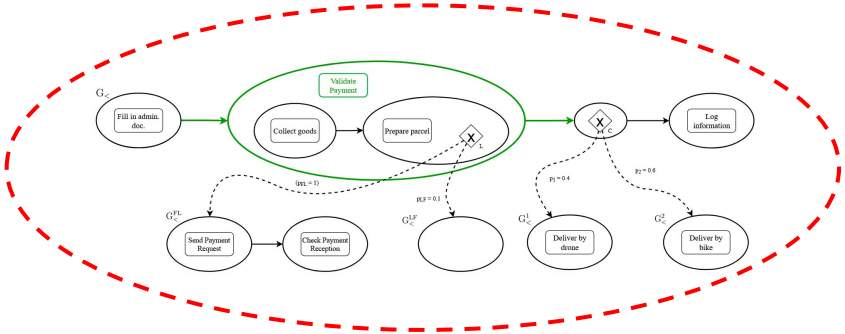
Step 2 – Refactoring Patterns

The selected task is **moved** thanks to **4 refactoring patterns**.

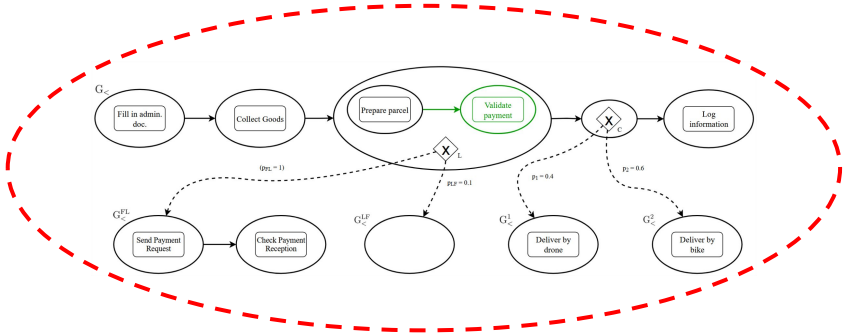
Task Between Nodes



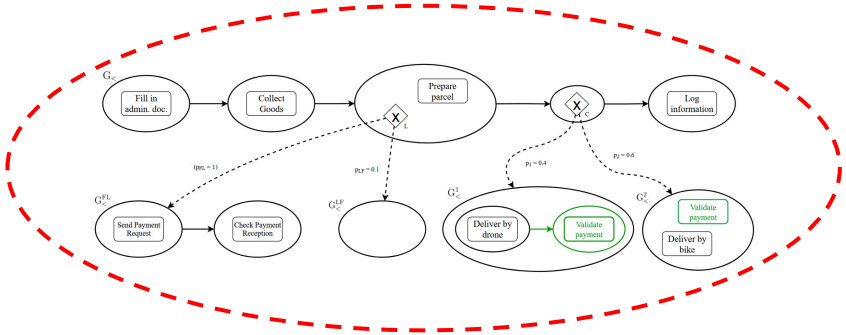
Task in Parallel of Sub-Sequences



Task Before/After Elements of a Node



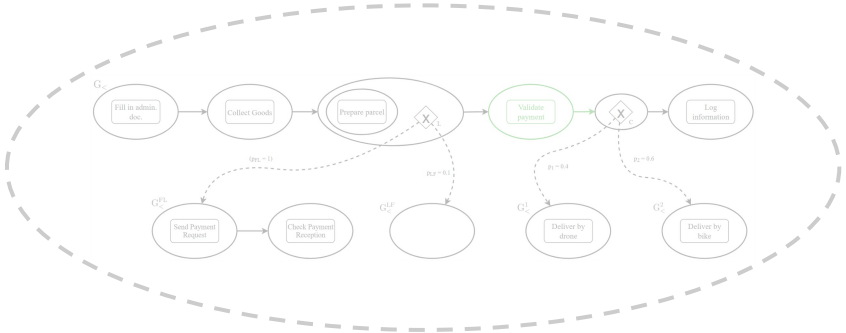
Task Inside Choices



Step 2 – Refactoring Patterns

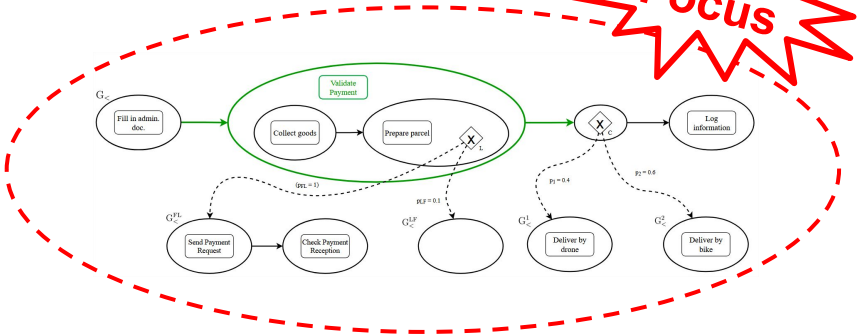
The selected task is **moved** thanks to **4 refactoring patterns**.

Task Between Nodes

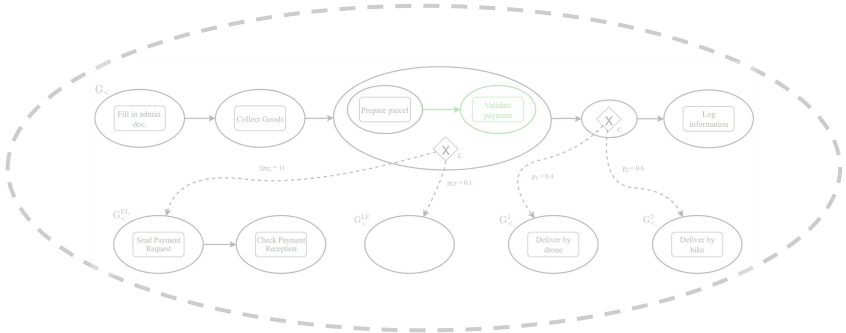


Task in Parallel of Sub-Se

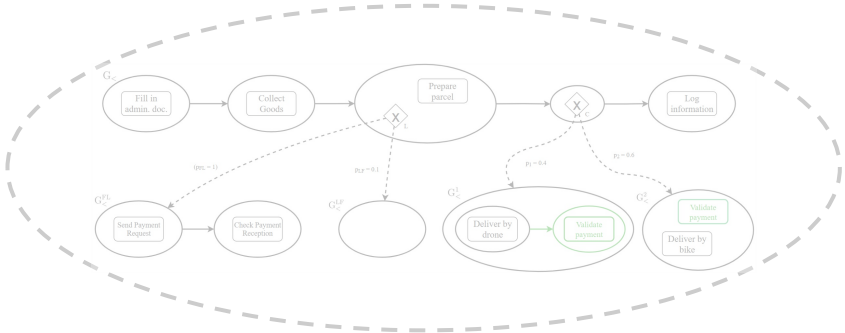
Focus



Task Before/After Elements of a Node

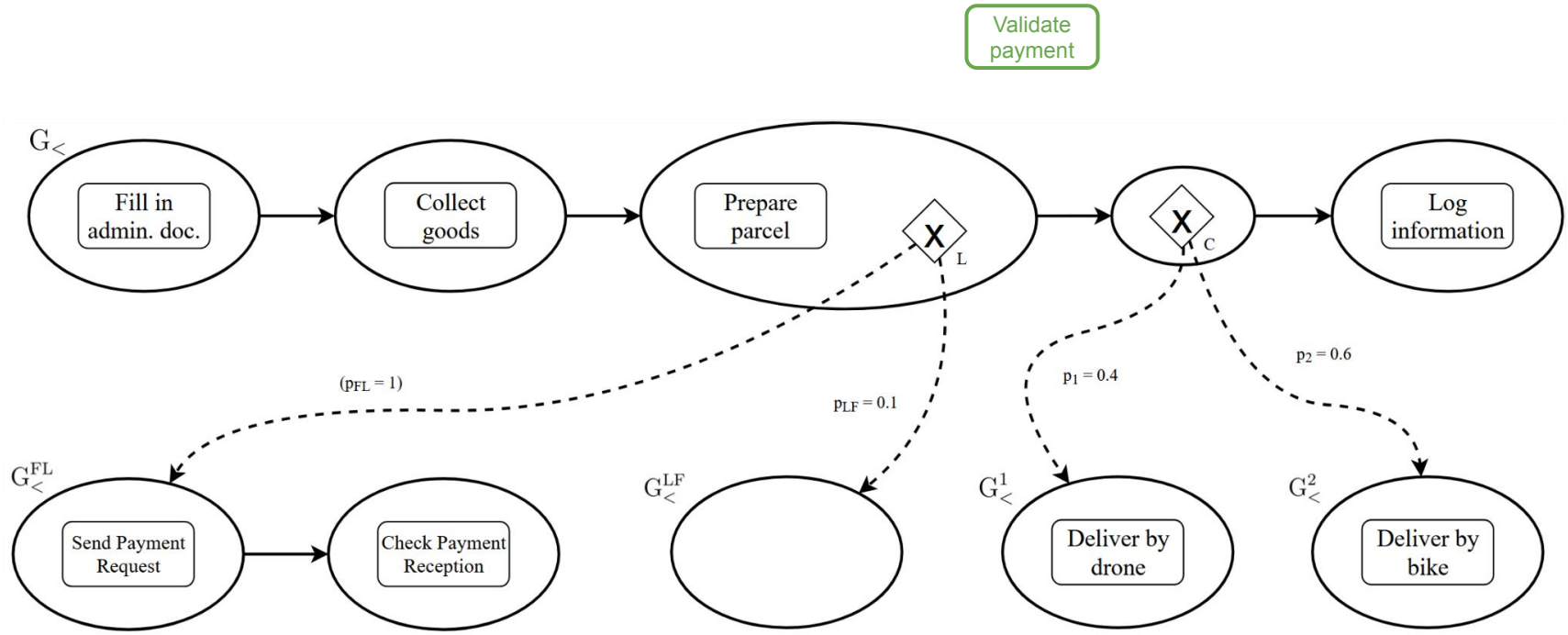


Task Inside Choices



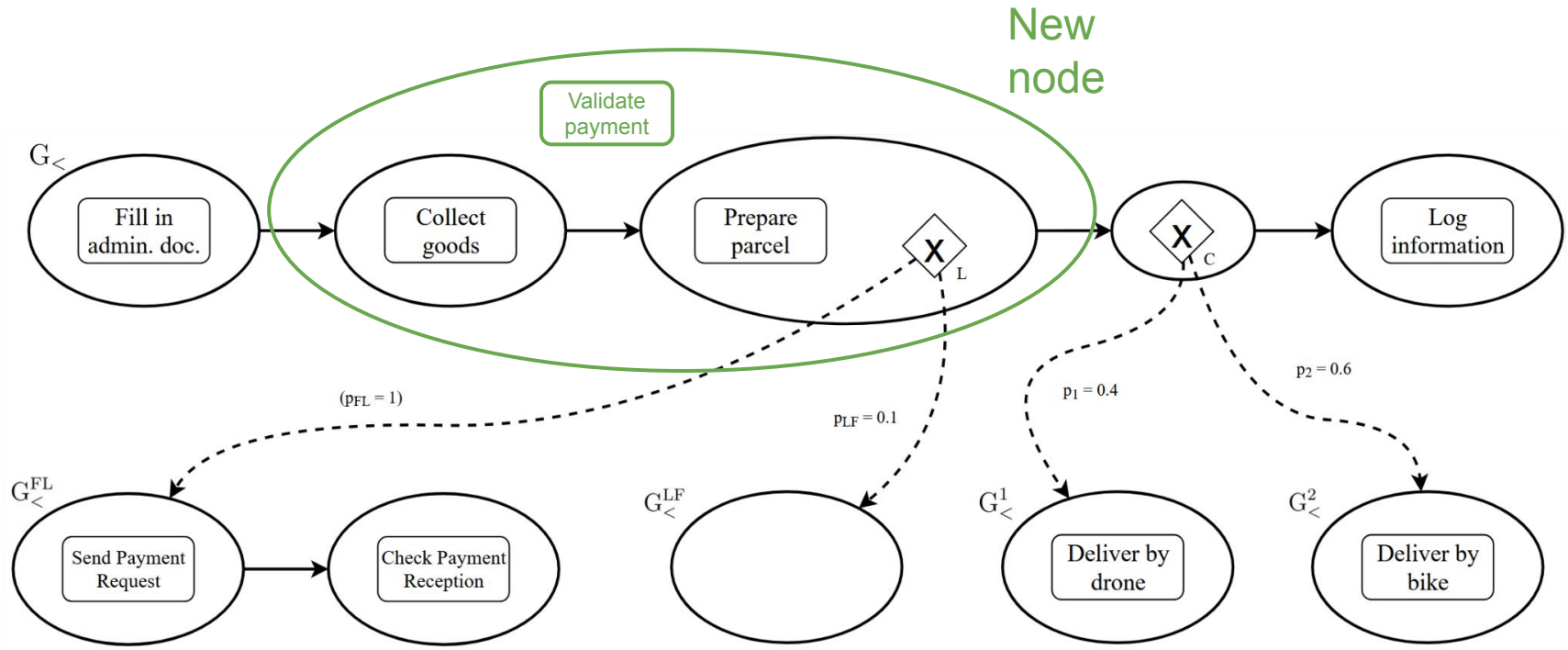
Step 2 – Pattern 2

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



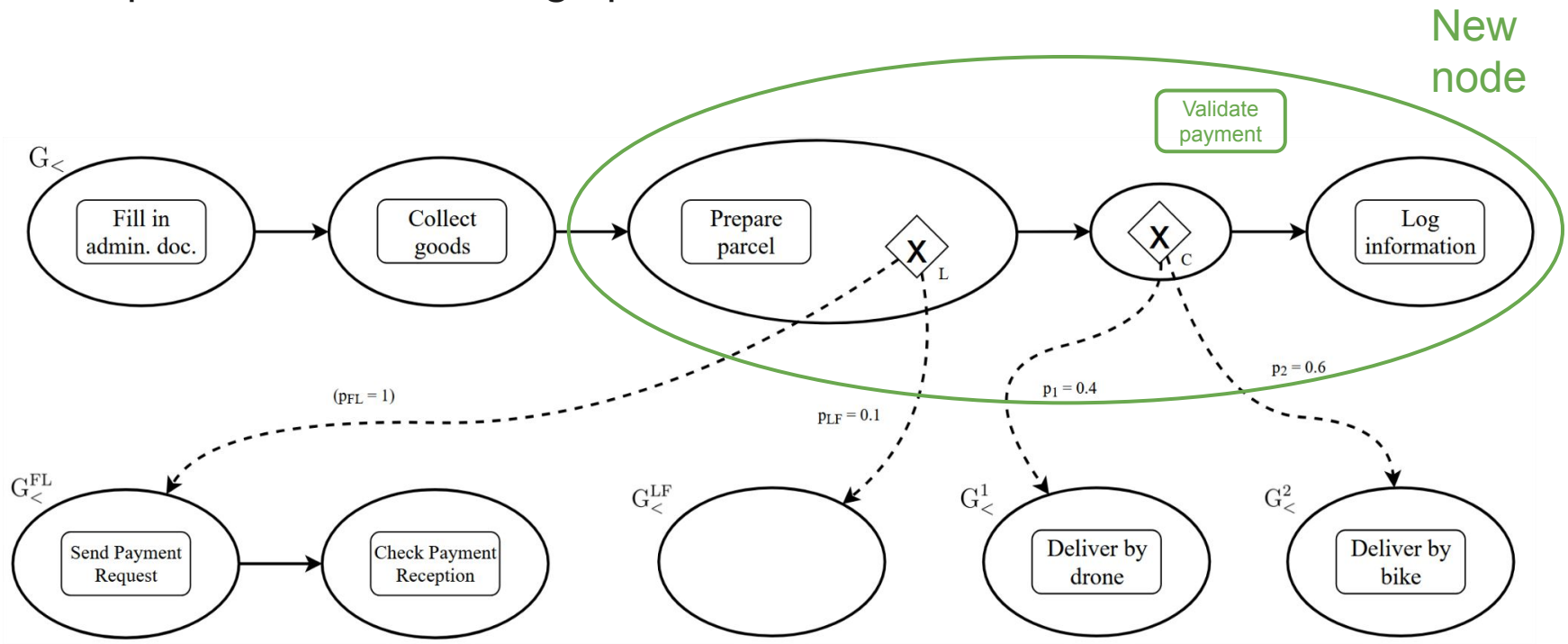
Step 2 – Pattern 2

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



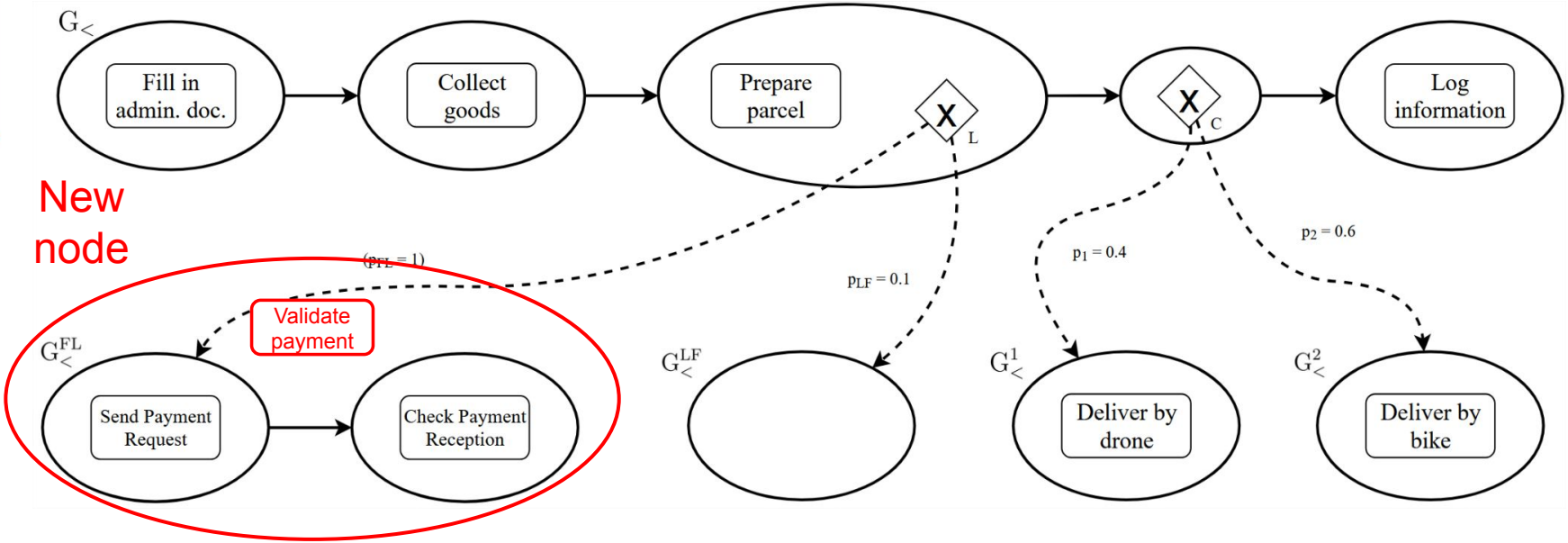
Step 2 – Pattern 2

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 – Structural Semantics Preservation

We **showed** in the manuscript that **the refactoring patterns preserve the structural semantics** of the process.

We **showed** in the manuscript that **the refactoring patterns preserve the structural semantics** of the process.

Proposition (Structural Semantics Preservation)

Let $G_< = (V_<, E_<, \Sigma_<)$ be a sequence graph, and let $t \in \mathcal{T}(G_<)$ be a task of $G_<$. We state that:

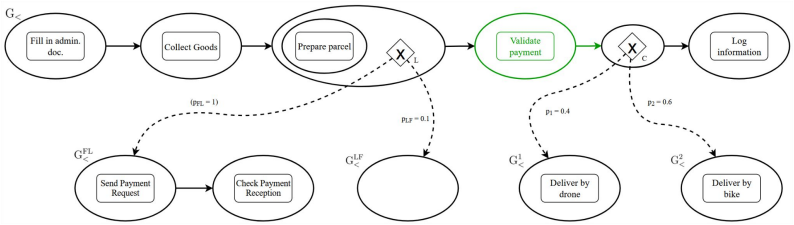
— $\forall G'_< \in \texttt{ins}(\texttt{rem}(G_<, t), t), \ \forall \lambda_< \in \Lambda(G_<), \ \exists \lambda' \in \mathfrak{S}(\Lambda(G'_<)) \mid \lambda_< = \lambda'_<$;

— $\forall G'_< \in \texttt{ins}(\texttt{rem}(G_<, t), t), \ \forall \lambda'_< \in \Lambda(G'_<), \ \exists \lambda \in \mathfrak{S}(\Lambda(G_<)) \mid \lambda'_< = \lambda_<$.

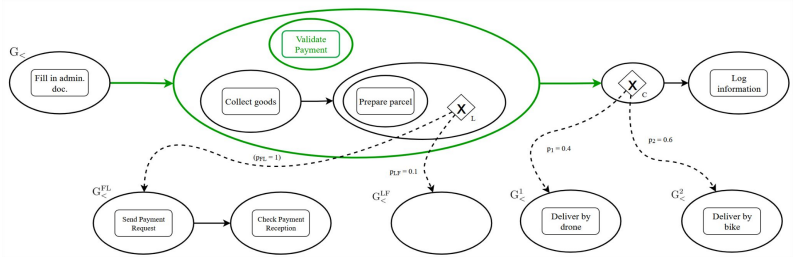
Step 3 – Comparison of the generated processes

The generated processes are **compared** based on their **average execution time** (AET), a metric obtained by **simulating** them in their **real conditions**.

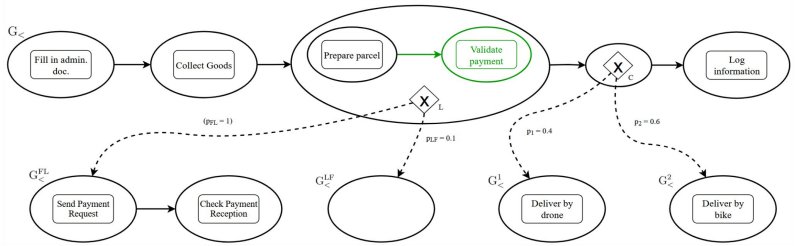
The generated processes are **compared** based on their **average execution time** (AET), a metric obtained by **simulating** them in their **real conditions**.



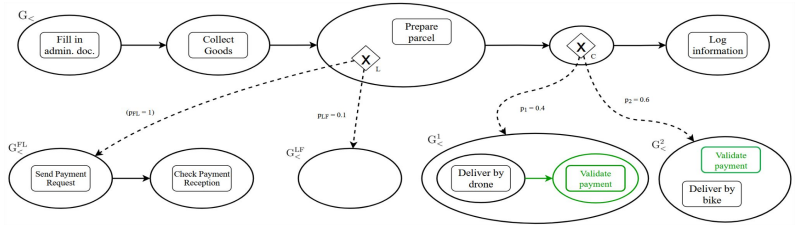
AET = 532 UT



AET = 512 UT



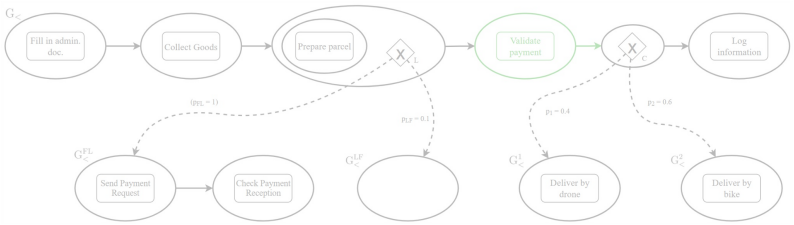
AET = 589 UT



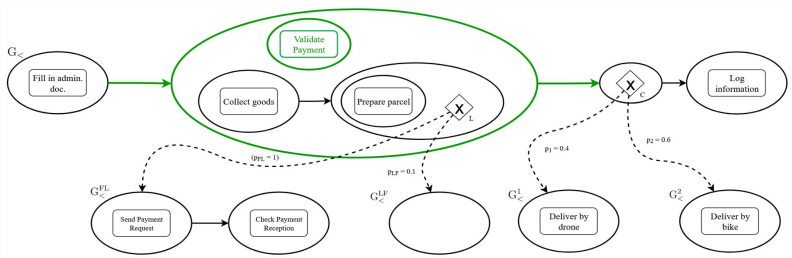
AET = 685 UT

Step 3 – Comparison of the generated processes

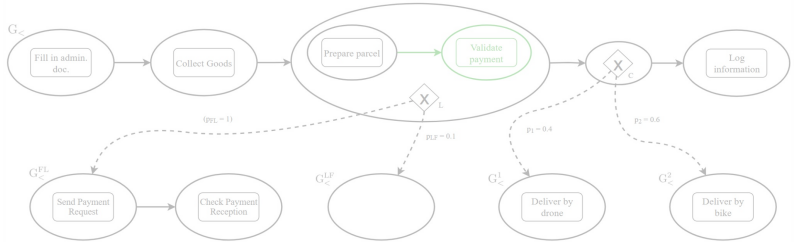
The generated processes are **compared** based on their **average execution time** (AET), a metric obtained by **simulating** them in their **real conditions**.



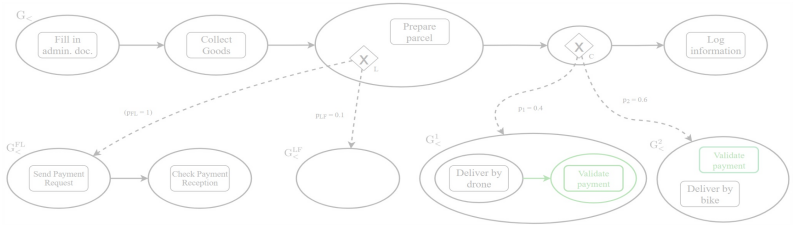
AET = 532 UT



AET = 512 UT



AET = 589 UT



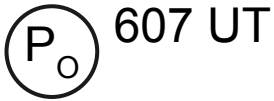
AET = 685 UT

Step 3 – Comparison bias: local optimum

However, the selected process is a **local optimum**!

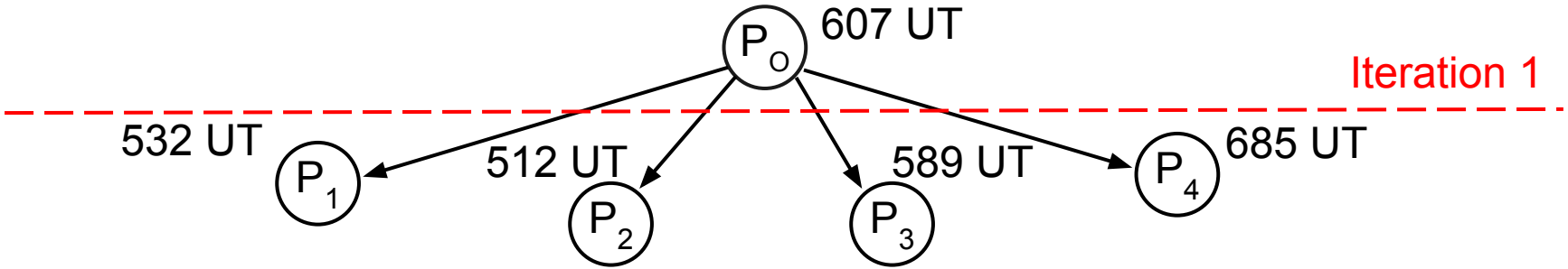
Step 3 – Comparison bias: local optimum

However, the selected process is a **local optimum**!



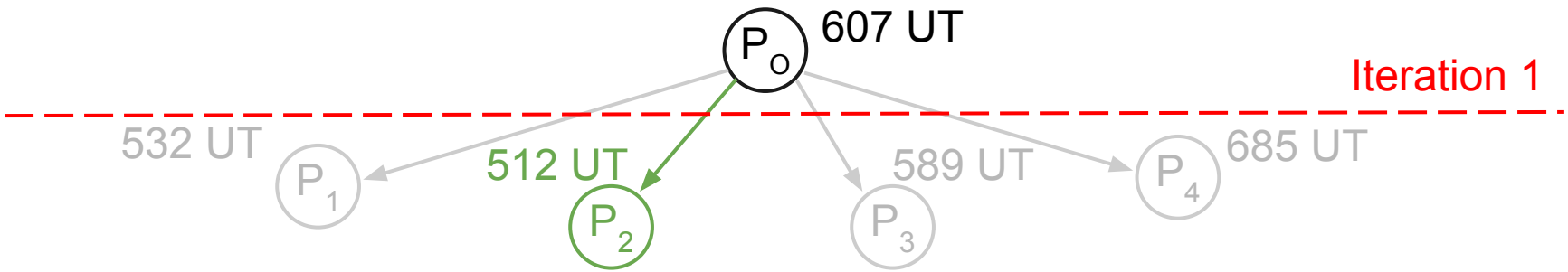
Step 3 – Comparison bias: local optimum

However, the selected process is a **local optimum**!



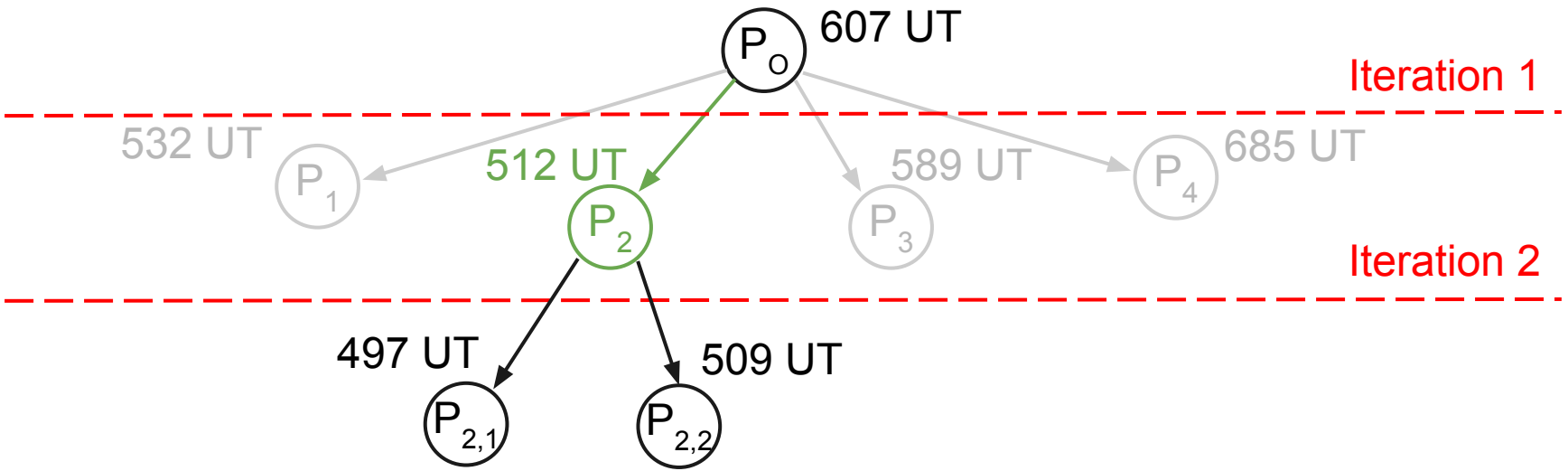
Step 3 – Comparison bias: local optimum

However, the selected process is a **local optimum**!



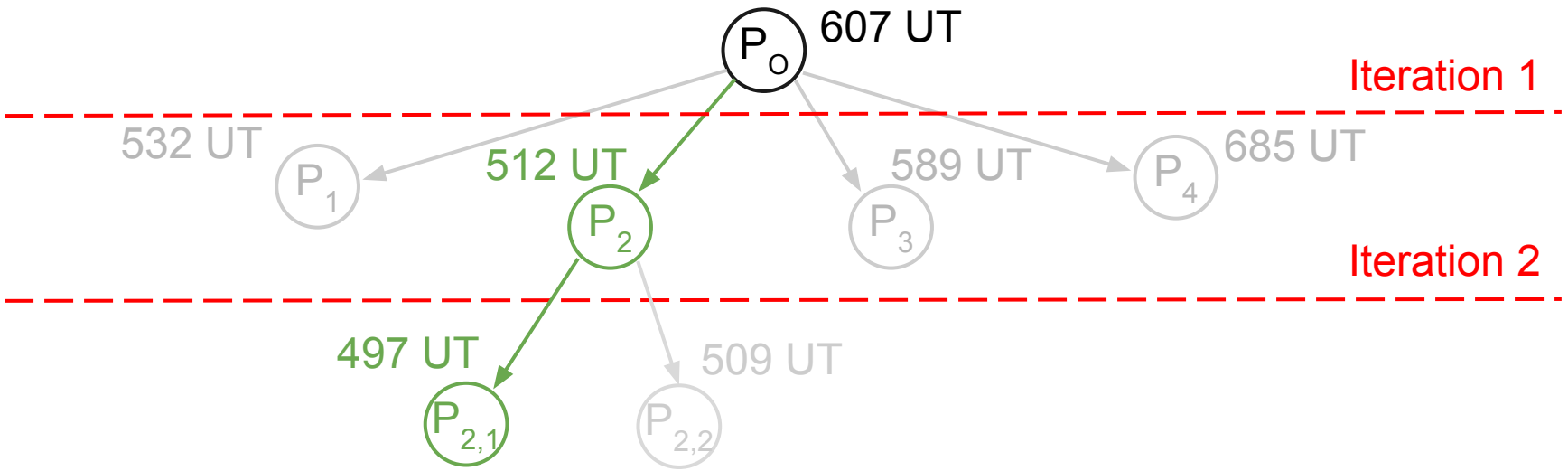
Step 3 – Comparison bias: local optimum

However, the selected process is a **local optimum**!



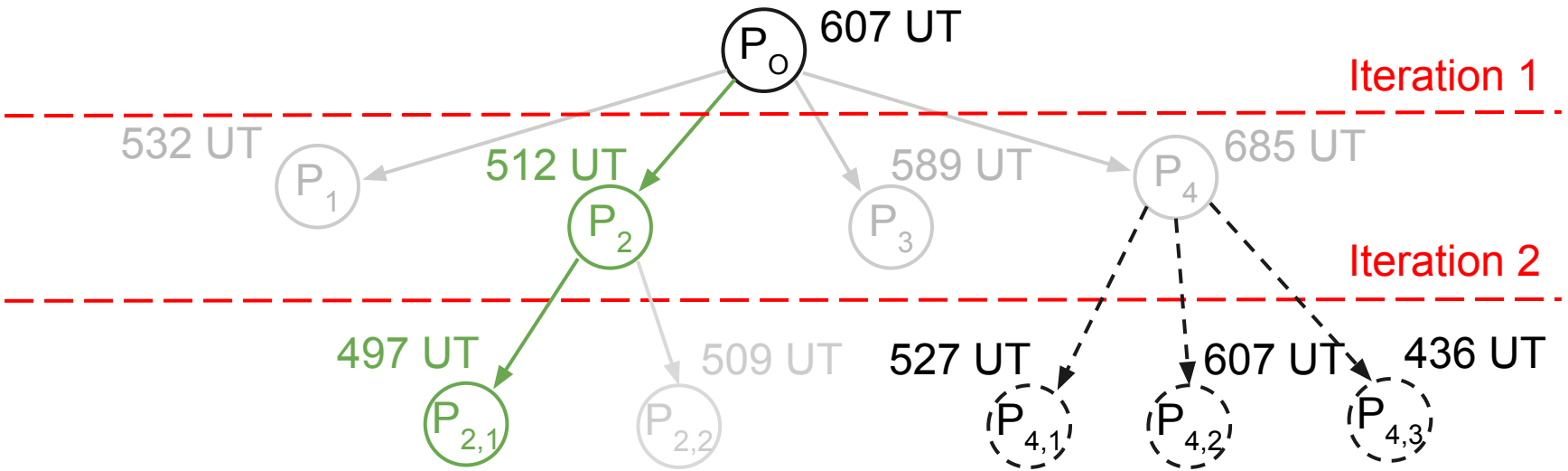
Step 3 – Comparison bias: local optimum

However, the selected process is a **local optimum**!



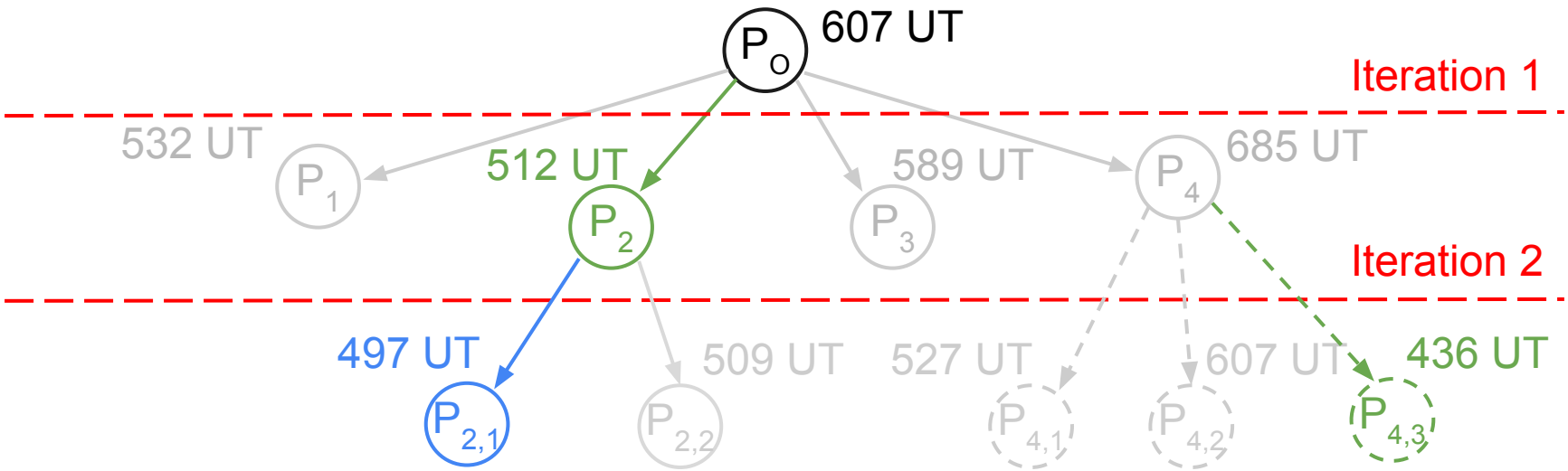
Step 3 – Comparison bias: local optimum

However, the selected process is a **local optimum**!



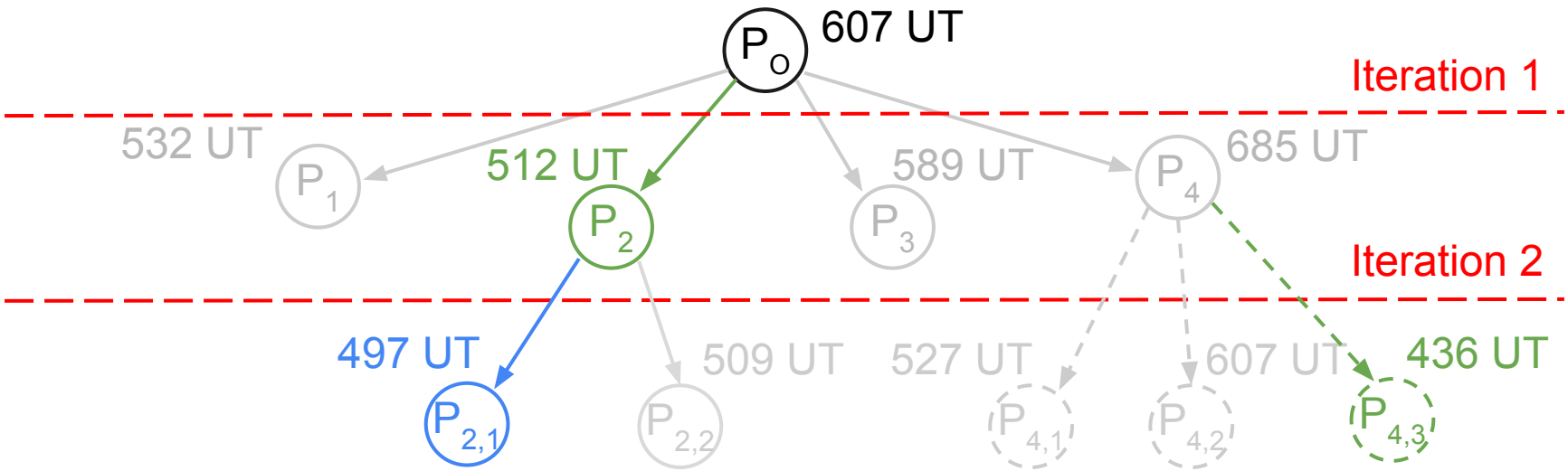
Step 3 – Comparison bias: local optimum

However, the selected process is a **local optimum**!



Step 3 – Comparison bias: local optimum

However, the selected process is a **local optimum**!



⇒ There is **no guarantee** that this **local optimum** will **lead to a global one**!

Step 3 – Comparison bias solution: full exploration

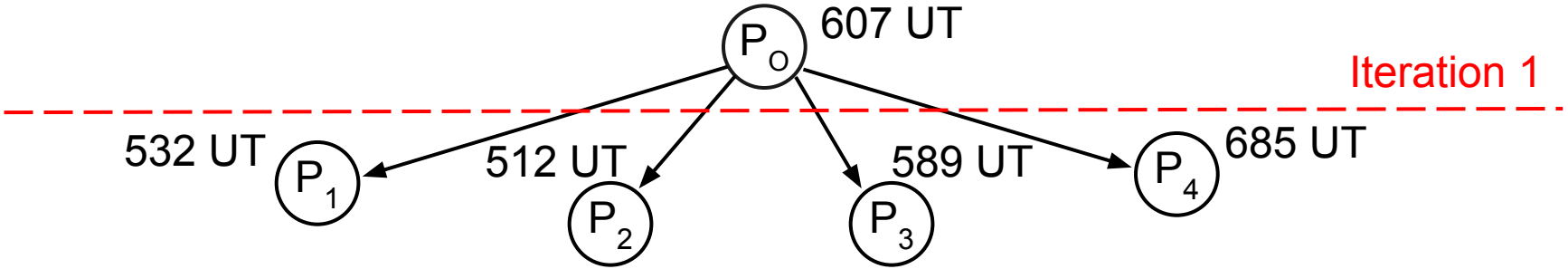
A solution is thus to **compute the whole tree** of solutions and **pick the best leaf**.

Step 3 – Comparison bias solution: full exploration

A solution is thus to **compute the whole tree** of solutions and **pick the best leaf**.

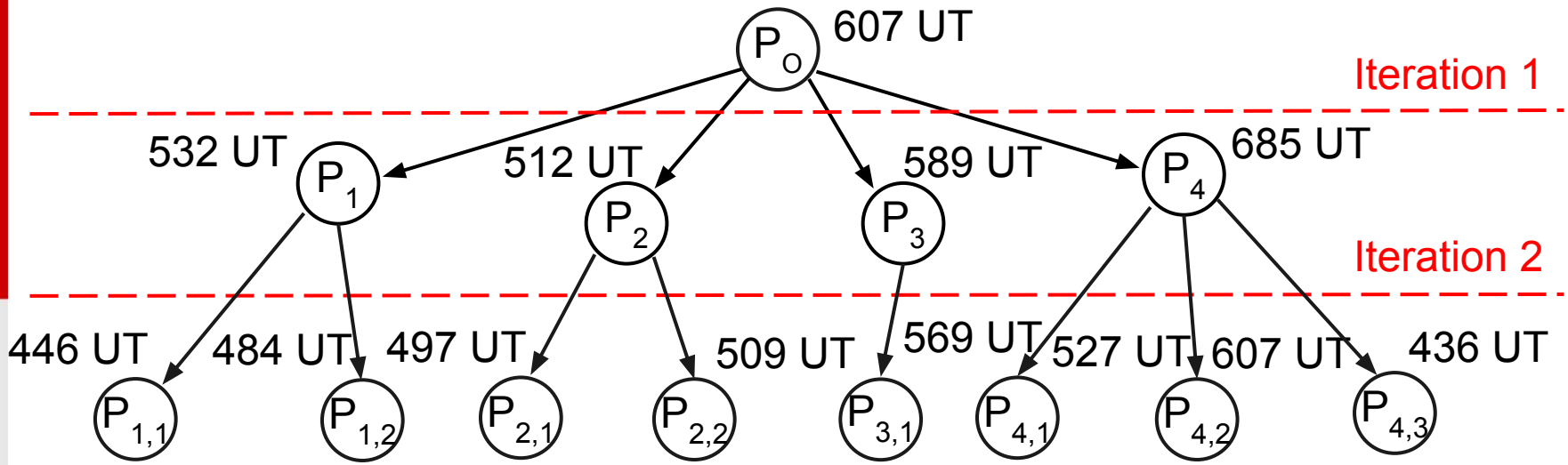
Step 3 – Comparison bias solution: full exploration

A solution is thus to **compute the whole tree** of solutions and **pick the best leaf**.



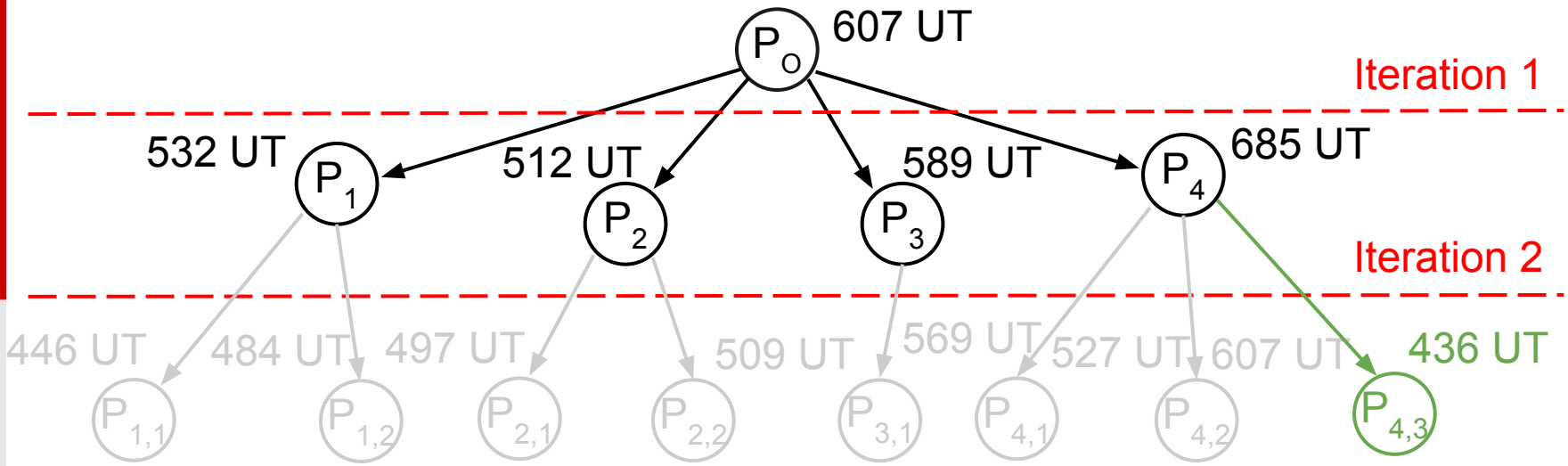
Step 3 – Comparison bias solution: full exploration

A solution is thus to **compute the whole tree** of solutions and **pick the best leaf**.



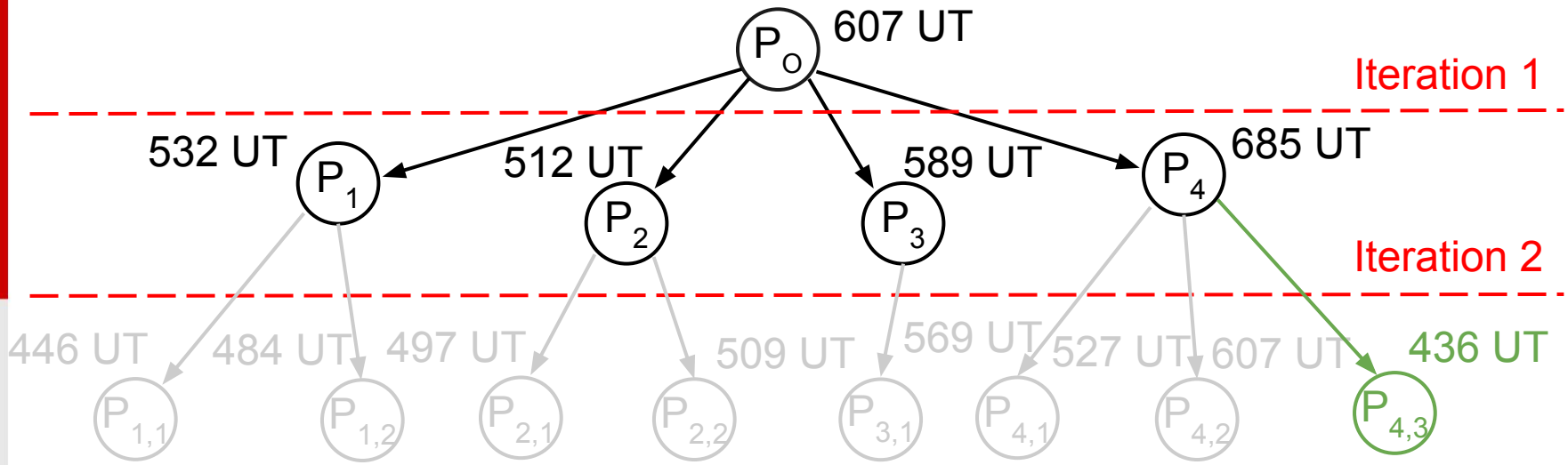
Step 3 – Comparison bias solution: full exploration

A solution is thus to **compute the whole tree** of solutions and **pick the best leaf**.



Step 3 – Comparison bias solution: full exploration

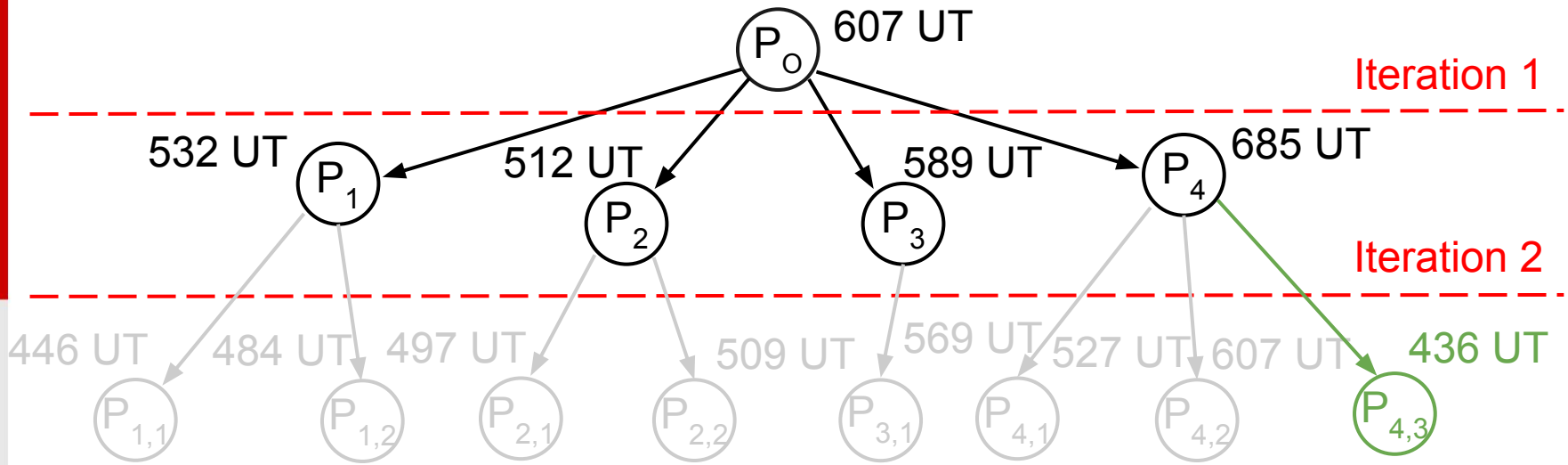
A solution is thus to **compute the whole tree** of solutions and **pick the best leaf**.



However, this is **not feasible** in practice, due to the **size of the** generated **tree**.

Step 3 – Comparison bias solution: full exploration

A solution is thus to **compute the whole tree** of solutions and **pick the best leaf**.



However, this is **not feasible** in practice, due to the **size of the** generated **tree**.

For instance, a BPMN process with **15 tasks** which can be **moved** to **20** different **places** generates a tree of **$15^{20} = 3 \times 10^{23}$ nodes**.

Step 3 – Comparison bias solution: heuristics

Thus, there is a need for **heuristics** aiming at **efficiently traversing the tree** of solutions.

Thus, there is a need for **heuristics** aiming at **efficiently traversing the tree** of solutions.

Our proposal consists in **attributing a (weighted) score** to each generated process, **based on** its **AET** and its **resources usage**.

Definition (Process Score)

Let $B = (V, E, \Sigma)$ be a BPMN process. The *score* of B is defined as

$$\text{score}(B) = \omega_{\text{AET}} \times (\delta_{\mu_{\text{AET}}} + \delta_{\sigma_{\text{AET}}} + \omega_{\text{loc}} \times \delta_{\text{AET}}) + \omega_{\text{res}} \times (\delta_{\mu_{\text{res}}} + \omega_{\text{loc}} \times \delta_{\text{res}})$$

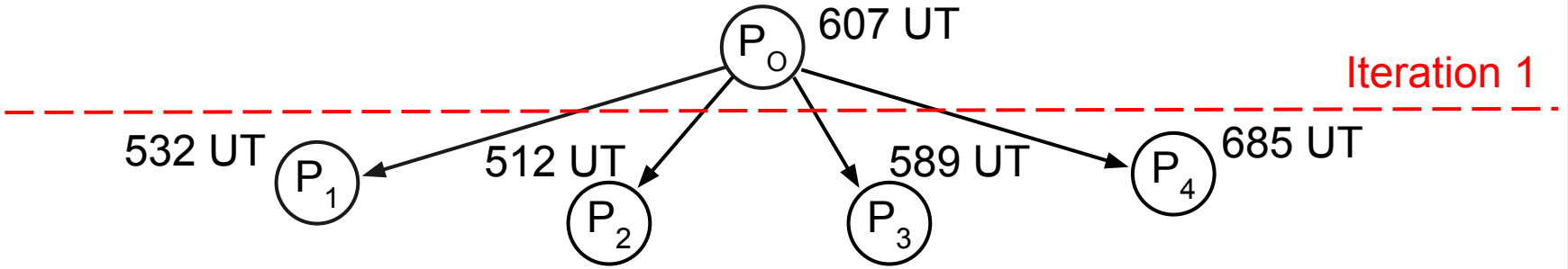
Step 3 – Comparison bias solution: heuristics

Based on this score, **one or several processes** of the current layer **are kept**, and **used** as basis for the **computation** of the **next layer**.

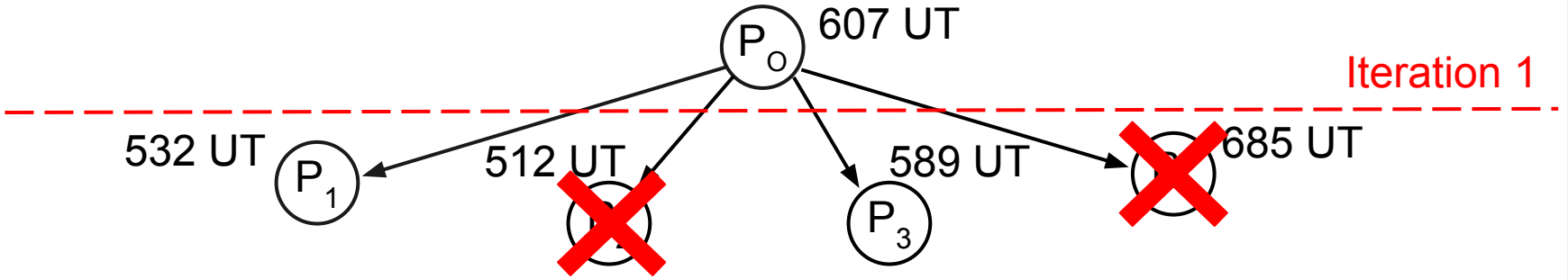
Step 3 – Comparison bias solution: heuristics

Based on this score, **one or several processes** of the current layer **are kept**, and **used** as basis for the **computation** of the **next layer**.

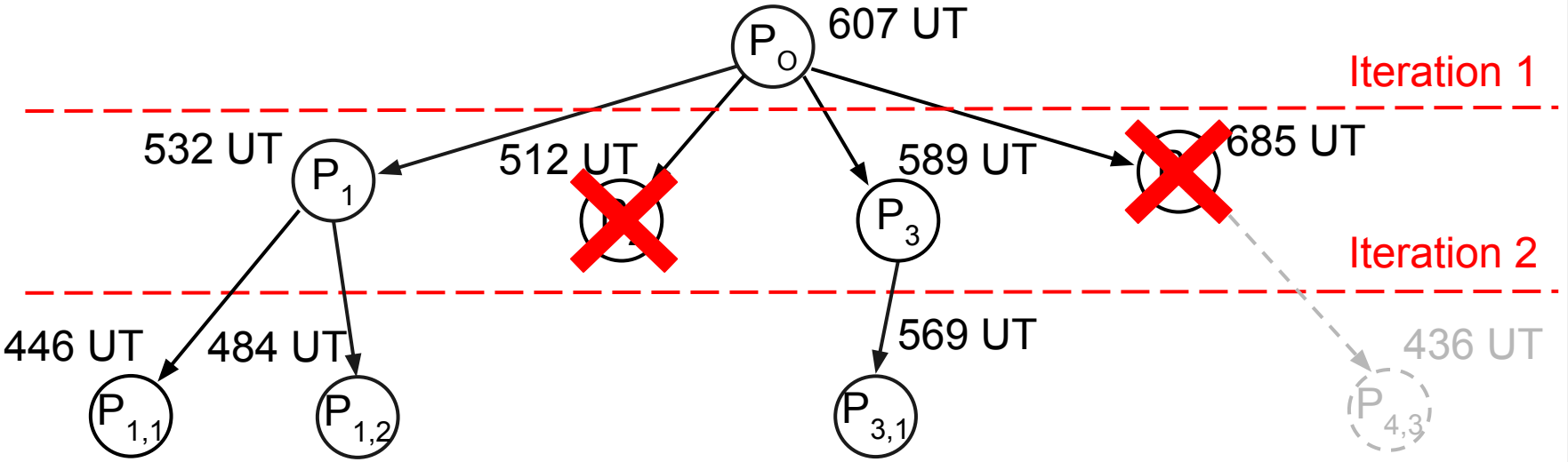
Based on this score, **one or several processes** of the current layer **are kept**, and **used** as basis for the **computation** of the **next layer**.



Based on this score, **one or several processes** of the current layer **are kept**, and **used** as basis for the **computation** of the **next layer**.

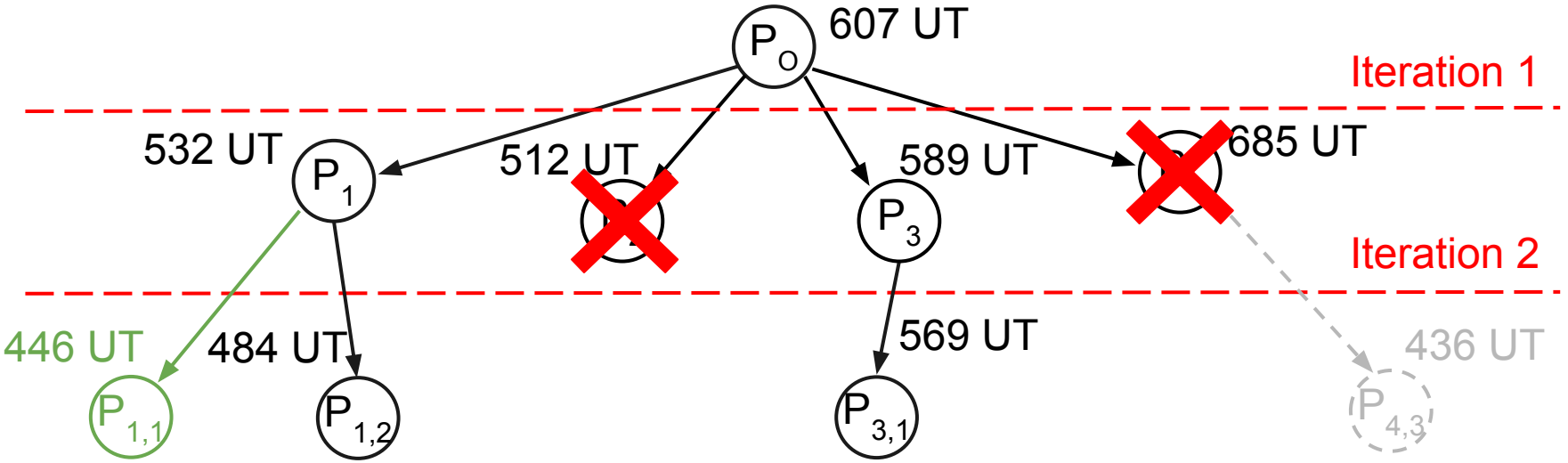


Based on this score, **one or several processes** of the current layer **are kept**, and **used** as basis for the **computation** of the **next layer**.



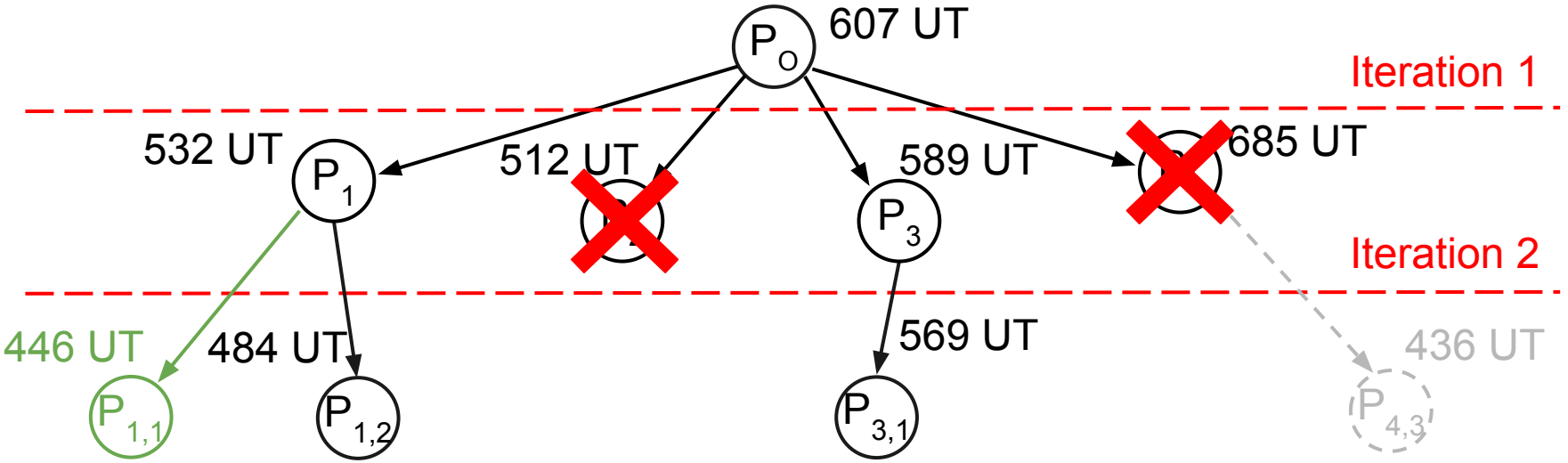
Step 3 – Comparison bias solution: heuristics

Based on this score, **one or several processes** of the current layer **are kept**, and **used** as basis for the **computation** of the **next layer**.



Step 3 – Comparison bias solution: heuristics

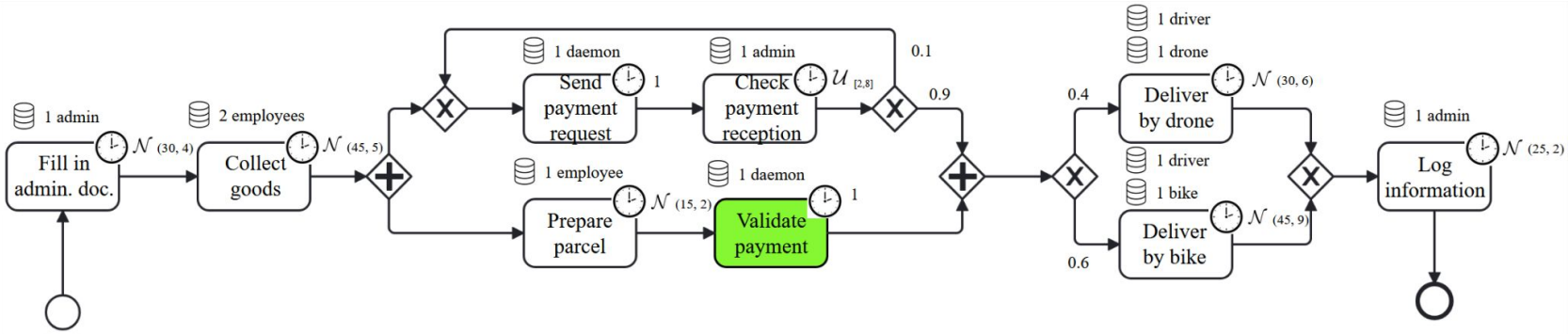
Based on this score, **one or several processes** of the current layer **are kept**, and **used** as basis for the **computation** of the **next layer**.



We obtain a process that is **close to the optimal** (446 UT / 436 UT) while **fastening** the **computations**.

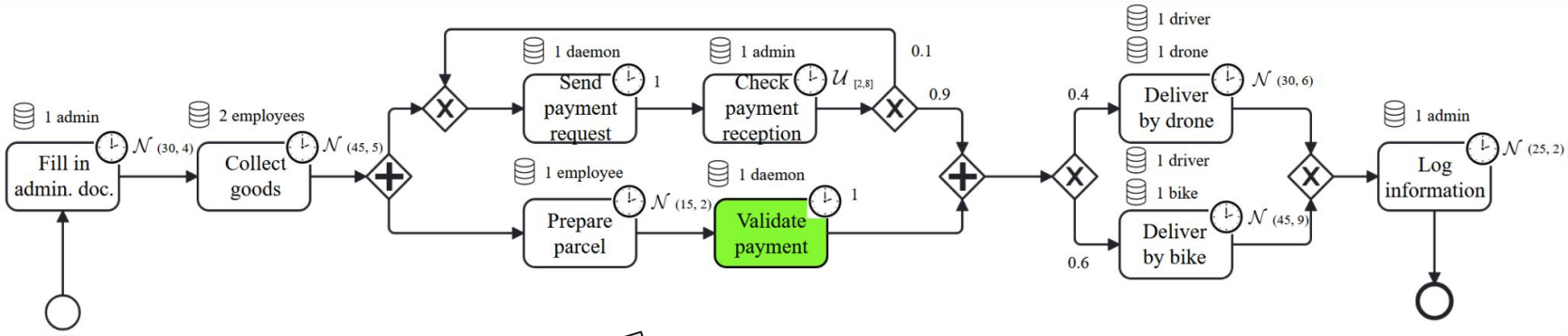
Step 3 – Human Process Validation

The resulting process is then **proposed to the user**.



Step 3 – Human Process Validation

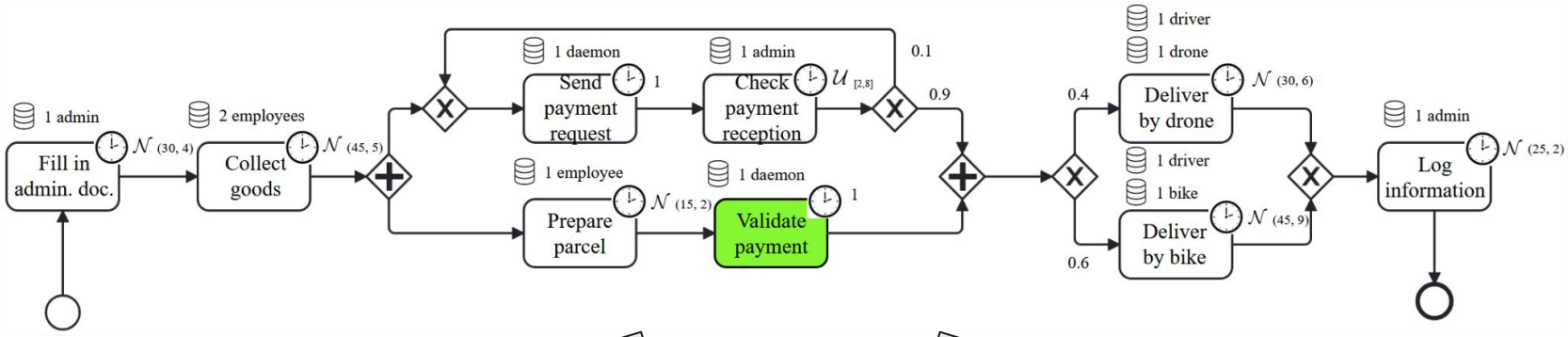
The resulting process is then **proposed to the user**.



The user **validates the process** ⇒ we propose a new task to move **on it**.

Step 3 – Human Process Validation

The resulting process is then **proposed to the user**.

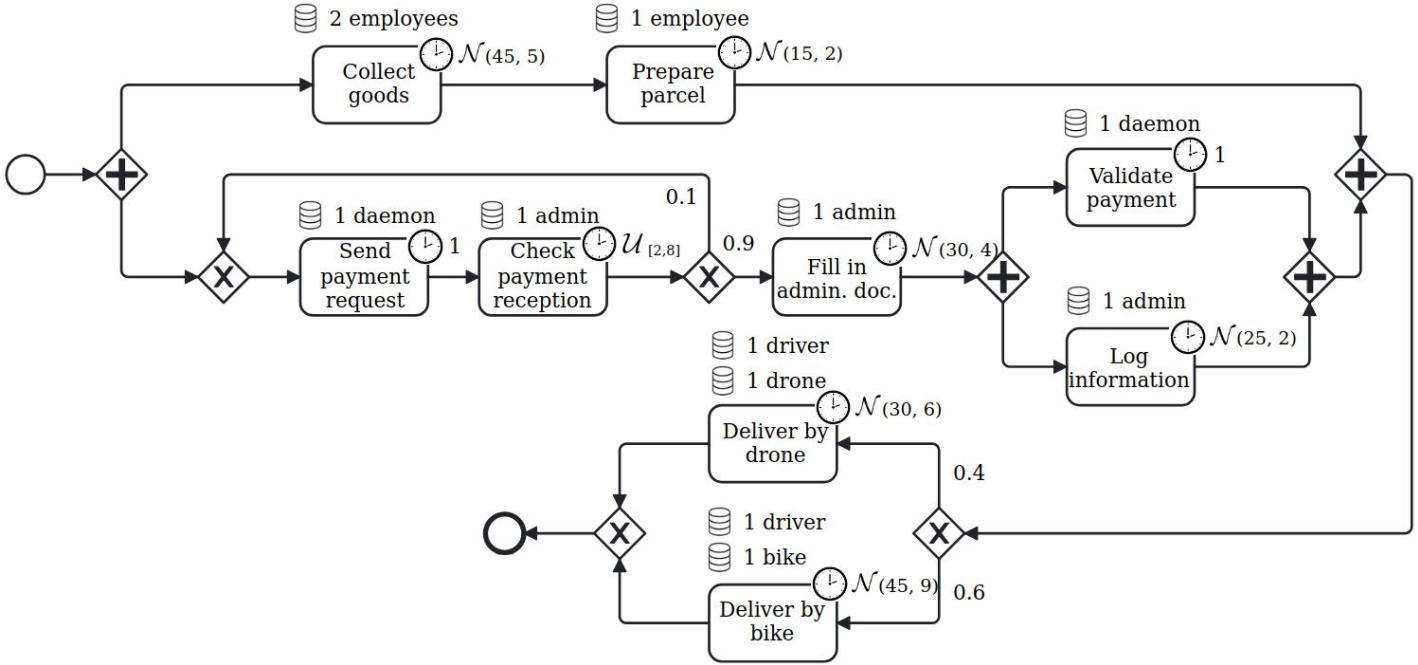


The user **validates the process** ⇒ we propose a new task to move **on it**.

The user **declines the process** ⇒ we propose a new task to move **on the previous process**.

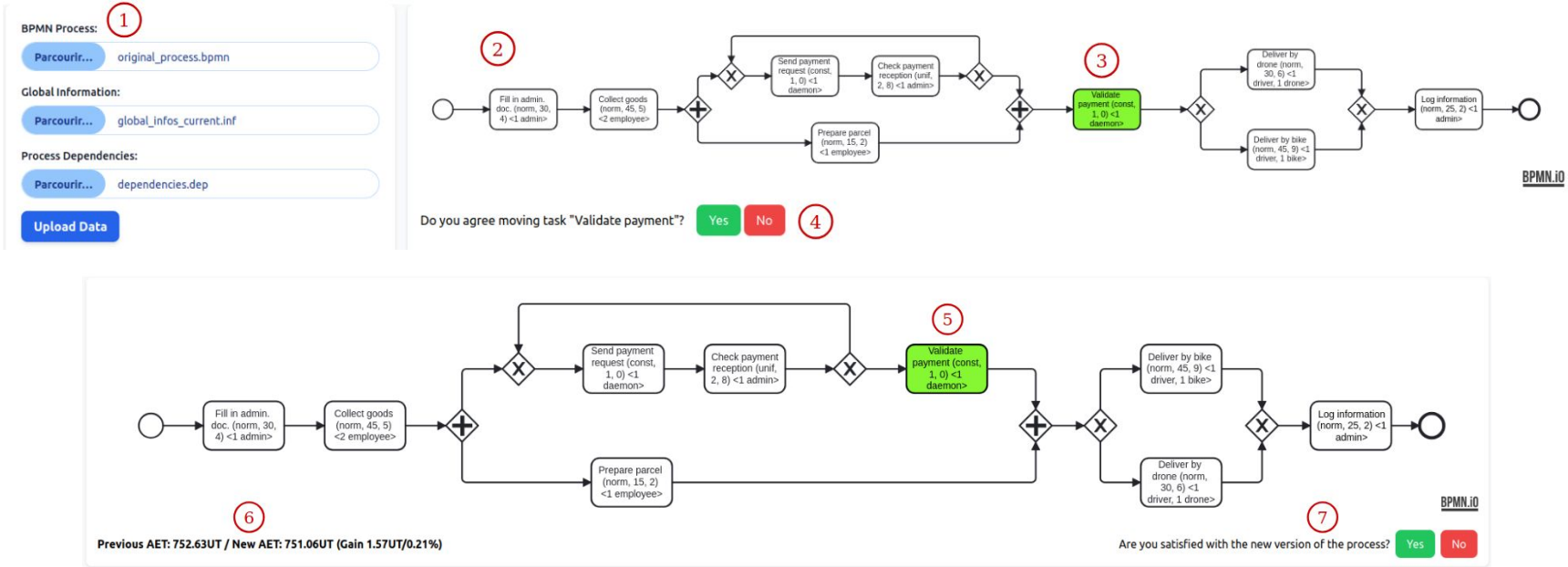
Step 3 – End of Refactoring Loop

When **all the tasks** of the process **have been moved**, or when the **user decides to stop**, the approach returns an **optimised version** of the **original process**.



Tool Support

- **15k lines of Java code**
- Executes in the backend of a **NodeJS server running locally**
- **Freely available online**



Experiments

Several **experiments were conducted to validate** the approach.

		Evisa App. [Sal22]	Empl. Rec. [FSZ21]	Patient Diag. [6]	Empl. Hir. [7]	Acc. Op. [NS22]	Per. Goods [VTS22]	Online Ship. [8]	Hand-Crafted 1	Hand-Crafted 2a	Hand-Crafted 2b
Characs.	Tasks	9	10	8	11	15	16	24	26	51	51
	$\diamondsuit_C$	1	1	2	2	2	2	3	4	1	1
	Deps.	3	9	3	7	7	10	27	43	43	23
	AET	36.1	30.9	67.2	24.7	51.9	15	85.9	232	323	323
Heuristic	AET	20	21.4	61.6	19	40.9	13.2	70.3	145*	244*	182*
	Gain	44.6%	30.7%	8.33%	23.1	21.2%	12.0%	18.2	37.5%	24.5%	43.7%
	Time	6.21s	32s	5s	26s	1.25m	14s	1.97m	6.37m	58m	2.12h
	μ_{time}	0.88s	0.32s	0.56s	2.36s	5s	1.75s	4.9s	15s	1.14m	2.54m
Full	AET	17.1	20.4	60.4	17.8	34.2	13.2	69.3	122	183	99
	Gain	52.6%	34.0%	10.1%	27.9%	34.1%	12.0%	19.3%	47.4%	43.3%	69.3%
	Time	17.0m	7.38m	11.2s	43.1h	26.8h	1.34h	>14d	1.7d	>14d	>14d

Experiments

Several **experiments were conducted to validate** the approach.

		Evisa App. [Sal22]	Empl. Rec. [FSZ21]	Patient Diag. 6	Empl. Hir. 7	Acc. Op. [NS22]	Per. Goods [VTS22]	Online Ship. 8	Hand-Crafted 1	Hand-Crafted 2a	Hand-Crafted 2b
Characs.	Tasks	9	10	8	11	15	16	24	26	51	51
	◈ $_C$	1	1	2	2	2	2	3	4	1	1
	Deps.	3	9	3	7	7	10	27	43	43	23
	AET	36.1	30.9	67.2	24.7	51.9	15	85.9	232	323	323
Heuristic	AET	20	21.4	61.6	19	40.9	13.2	70.3	145*	244*	182*
	Gain	44.6%	30.7%	8.33%	23.1	21.2%	12.0%	18.2	37.5%	24.5%	43.7%
	Time	6.21s	32s	5s	26s	1.25m	14s	1.97m	6.37m	58m	2.12h
	μ_{time}	0.88s	0.32s	0.56s	2.36s	5s	1.75s	4.9s	15s	1.14m	2.54m
Full	AET	17.1	20.4	60.4	17.8	34.2	13.2	69.3	122	183	99
	Gain	52.6%	34.0%	10.1%	27.9%	34.1%	12.0%	19.3%	47.4%	43.3%	69.3%
	Time	17.0m	7.38m	11.2s	43.1h	26.8h	1.34h	>14d	1.7d	>14d	>14d

Experiments

Several **experiments were conducted to validate** the approach.

		Evisa App. [Sal22]	Empl. Rec. [FSZ21]	Patient Diag. [6]	Empl. Hir. [7]	Acc. Op. [NS22]	Per. Goods [VTS22]	Online Ship. [8]	Hand-Crafted 1	Hand-Crafted 2a	Hand-Crafted 2b
Characs.	Tasks	9	10	8	11	15	16	24	26	51	51
	$\diamondsuit_C$	1	1	2	2	2	2	3	4	1	1
	Deps.	3	9	3	7	7	10	27	43	43	23
	AET	36.1	30.9	67.2	24.7	51.9	15	85.9	232	323	323
Heuristic	AET	20	21.4	61.6	19	40.9	13.2	70.3	145*	244*	182*
	Gain	44.6%	30.7%	8.33%	23.1	21.2%	12.0%	18.2	37.5%	24.5%	43.7%
	Time	6.21s	32s	5s	26s	1.25m	14s	1.97m	6.37m	58m	2.12h
	μ_{time}	0.88s	0.32s	0.56s	2.36s	5s	1.75s	4.9s	15s	1.14m	2.54m
Full	AET	17.1	20.4	60.4	17.8	34.2	13.2	69.3	122	183	99
	Gain	52.6%	34.0%	10.1%	27.9%	34.1%	12.0%	19.3%	47.4%	43.3%	69.3%
	Time	17.0m	7.38m	11.2s	43.1h	26.8h	1.34h	>14d	1.7d	>14d	>14d

Experiments

Several **experiments were conducted to validate** the approach.

		Evisa App. [Sal22]	Empl. Rec. [FSZ21]	Patient Diag. [6]	Empl. Hir. [7]	Acc. Op. [NS22]	Per. Goods [VTS22]	Online Ship. [8]	Hand-Crafted 1	Hand-Crafted 2a	Hand-Crafted 2b
Characs.	Tasks	9	10	8	11	15	16	24	26	51	51
	$\diamondsuit_C$	1	1	2	2	2	2	3	4	1	1
	Deps.	3	9	3	7	7	10	27	43	43	23
	AET	36.1	30.9	67.2	24.7	51.9	15	85.9	232	323	323
Heuristic	AET	20	21.4	61.6	19	40.9	13.2	70.3	145*	244*	182*
	Gain	44.6%	30.7%	8.33%	23.1	21.2%	12.0%	18.2	37.5%	24.5%	43.7%
	Time	6.21s	32s	5s	26s	1.25m	14s	1.97m	6.37m	58m	2.12h
	μ_{time}	0.88s	0.32s	0.56s	2.36s	5s	1.75s	4.9s	15s	1.14m	2.54m
Full	AET	17.1	20.4	60.4	17.8	34.2	13.2	69.3	122	183	99
	Gain	52.6%	34.0%	10.1%	27.9%	34.1%	12.0%	19.3%	47.4%	43.3%	69.3%
	Time	17.0m	7.38m	11.2s	43.1h	26.8h	1.34h	>14d	1.7d	>14d	>14d

Experiments

Several **experiments were conducted to validate** the approach.

		Evisa App. [Sal22]	Empl. Rec. [FSZ21]	Patient Diag. [6]	Empl. Hir. [7]	Acc. Op. [NS22]	Per. Goods [VTS22]	Online Ship. [8]	Hand-Crafted 1	Hand-Crafted 2a	Hand-Crafted 2b
Characs.	Tasks	9	10	8	11	15	16	24	26	51	51
	$\diamondsuit_C$	1	1	2	2	2	2	3	4	1	1
	Deps.	3	9	3	7	7	10	27	43	43	23
	AET	36.1	30.9	67.2	24.7	51.9	15	85.9	232	323	323
Heuristic	AET	20	21.4	61.6	19	40.9	13.2	70.3	145*	244*	182*
	Gain	44.6%	30.7%	8.33%	23.1	21.2%	12.0%	18.2	37.5%	24.5%	43.7%
	Time	6.21s	32s	5s	26s	1.25m	14s	1.97m	6.37m	58m	2.12h
	μ_{time}	0.88s	0.32s	0.56s	2.36s	5s	1.75s	4.9s	15s	1.14m	2.54m
Full	AET	17.1	20.4	60.4	17.8	34.2	13.2	69.3	122	183	99
	Gain	52.6%	34.0%	10.1%	27.9%	34.1%	12.0%	19.3%	47.4%	43.3%	69.3%
	Time	17.0m	7.38m	11.2s	43.1h	26.8h	1.34h	>14d	1.7d	>14d	>14d

Experiments

Several **experiments were conducted to validate** the approach.

		Evisa App. [Sal22]	Empl. Rec. [FSZ21]	Patient Diag. [6]	Empl. Hir. [7]	Acc. Op. [NS22]	Per. Goods [VTS22]	Online Ship. [8]	Hand-Crafted 1	Hand-Crafted 2a	Hand-Crafted 2b
Characs.	Tasks	9	10	8	11	15	16	24	26	51	51
	$\diamondsuit_C$	1	1	2	2	2	2	3	4	1	1
	Deps.	3	9	3	7	7	10	27	43	43	23
	AET	36.1	30.9	67.2	24.7	51.9	15	85.9	232	323	323
Heuristic	AET	20	21.4	61.6	19	40.9	13.2	70.3	145*	244*	182*
	Gain	44.6%	30.7%	8.33%	23.1	21.2%	12.0%	18.2	37.5%	24.5%	43.7%
	Time	6.21s	32s	5s	26s	1.25m	14s	1.97m	6.37m	58m	2.12h
	μ_{time}	0.88s	0.32s	0.56s	2.36s	5s	1.75s	4.9s	15s	1.14m	2.54m
Full	AET	17.1	20.4	60.4	17.8	34.2	13.2	69.3	122	183	99
	Gain	52.6%	34.0%	10.1%	27.9%	34.1%	12.0%	19.3%	47.4%	43.3%	69.3%
	Time	17.0m	7.38m	11.2s	43.1h	26.8h	1.34h	>14d	1.7d	>14d	>14d

Experiments

Several **experiments were conducted to validate** the approach.

		Evisa App. [Sal22]	Empl. Rec. [FSZ21]	Patient Diag. 6	Empl. Hir. 7	Acc. Op. [NS22]	Per. Goods [VTS22]	Online Ship. 8	Hand-Crafted 1	Hand-Crafted 2a	Hand-Crafted 2b
Characs.	Tasks	9	10	8	11	15	16	24	26	51	51
	◈ $_C$	1	1	2	2	2	2	3	4	1	1
	Deps.	3	9	3	7	7	10	27	43	43	23
	AET	36.1	30.9	67.2	24.7	51.9	15	85.9	232	323	323
Heuristic	AET	20	21.4	61.6	19	40.9	13.2	70.3	145*	244*	182*
	Gain	44.6%	30.7%	8.33%	23.1	21.2%	12.0%	18.2	37.5%	24.5%	43.7%
	Time	6.21s	32s	5s	26s	1.25m	14s	1.97m	6.37m	58m	2.12h
	μ_{time}	0.88s	0.32s	0.56s	2.36s	5s	1.75s	4.9s	15s	1.14m	2.54m
Full	AET	17.1	20.4	60.4	17.8	34.2	13.2	69.3	122	183	99
	Gain	52.6%	34.0%	10.1%	27.9%	34.1%	12.0%	19.3%	47.4%	43.3%	69.3%
	Time	17.0m	7.38m	11.2s	43.1h	26.8h	1.34h	>14d	1.7d	>14d	>14d

Experiments

Several **experiments were conducted to validate** the approach.

		Evisa App. [Sal22]	Empl. Rec. [FSZ21]	Patient Diag. 6	Empl. Hir. 7	Acc. Op. [NS22]	Per. Goods [VTS22]	Online Ship. 8	Hand-Crafted 1	Hand-Crafted 2a	Hand-Crafted 2b
Characs.	Tasks	9	10	8	11	15	16	24	26	51	51
	$\diamondsuit_C$	1	1	2	2	2	2	3	4	1	1
	Deps.	3	9	3	7	7	10	27	43	43	23
	AET	36.1	30.9	67.2	24.7	51.9	15	85.9	232	323	323
Heuristic	AET	20	21.4	61.6	19	40.9	13.2	70.3	145*	244*	182*
	Gain	44.6%	30.7%	8.33%	23.1	21.2%	12.0%	18.2	37.5%	24.5%	43.7%
	Time	6.21s	32s	5s	26s	1.25m	14s	1.97m	6.37m	58m	2.12h
	μ_{time}	0.88s	0.32s	0.56s	2.36s	5s	1.75s	4.9s	15s	1.14m	2.54m
Full	AET	17.1	20.4	60.4	17.8	34.2	13.2	69.3	122	183	99
	Gain	52.6%	34.0%	10.1%	27.9%	34.1%	12.0%	19.3%	47.4%	43.3%	69.3%
	Time	17.0m	7.38m	11.2s	43.1h	26.8h	1.34h	>14d	1.7d	>14d	>14d

Experiments

Several **experiments were conducted to validate** the approach.

		Evisa App. [Sal22]	Empl. Rec. [FSZ21]	Patient Diag. 6	Empl. Hir. 7	Acc. Op. [NS22]	Per. Goods [VTS22]	Online Ship. 8	Hand-Crafted 1	Hand-Crafted 2a	Hand-Crafted 2b
Characs.	Tasks	9	10	8	11	15	16	24	26	51	51
	◈ $_C$	1	1	2	2	2	2	3	4	1	1
	Deps.	3	9	3	7	7	10	27	43	43	23
	AET	36.1	30.9	67.2	24.7	51.9	15	85.9	232	323	323
Heuristic	AET	20	21.4	61.6	19	40.9	13.2	70.3	145*	244*	182*
	Gain	44.6%	30.7%	8.33%	23.1	21.2%	12.0%	18.2	37.5%	24.5%	43.7%
	Time	6.21s	32s	5s	26s	1.25m	14s	1.97m	6.37m	58m	2.12h
	μ_{time}	0.88s	0.32s	0.56s	2.36s	5s	1.75s	4.9s	15s	1.14m	2.54m
Full	AET	17.1	20.4	60.4	17.8	34.2	13.2	69.3	122	183	99
	Gain	52.6%	34.0%	10.1%	27.9%	34.1%	12.0%	19.3%	47.4%	43.3%	69.3%
	Time	17.0m	7.38m	11.2s	43.1h	26.8h	1.34h	>14d	1.7d	>14d	>14d

Plan

I/ Introduction

II/ Automated Generation of BPMN Processes from Textual Requirements

III/ Human-Centered Refactoring-Based Optimisation of BPMN Processes

IV/ Related Work

V/ Takeaways

VI/ References

Related Work – Modelling

	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

	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

	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

	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

Qualitative refactoring [SM2007, DGKV2011, FRPCP2013]

Restructures a process to solve **structural issues**:

- soundness issues
- bad design
- duplicated parts of the process

Related Work – Refactoring

Qualitative refactoring [SM2007, DGKV2011, FRPCP2013]

Restructures a process to solve **structural issues**:

- soundness issues
- bad design
- duplicated parts of the process

Resource optimisation [DRS2019, DRS2021, FSZ2024]

Optimises a process by **changing** its number of available **resources:**

- statically
- runtime
- predictically

Related Work – Refactoring

Qualitative refactoring [SM2007, DGKV2011, FRPCP2013]

Restructures a process to solve **structural issues**:

- soundness issues
- bad design
- duplicated parts of the process

Resource optimisation [DRS2019, DRS2021, FSZ2024]

Optimises a process by **changing** its number of available **resources:**

- statically
- runtime
- predictically

Quantitative refactoring [RM2005, KL2022, DS2022]

Restructures a process to **optimise** its **execution time:**

- optional tasks
- duration reduction
- split/merge of tasks
- local patterns

Plan

I/ Introduction

II/ Automated Generation of BPMN Processes from Textual Requirements

III/ Human-Centered Refactoring-Based Optimisation of BPMN Processes

IV/ Related Work

V/ Takeaways

VI/ References

Takeaways

We propose an approach to **generate BPMN** processes:

Takeaways

We propose an approach to **generate BPMN** processes:

➢ **Fully automated**, tested, and available online

Takeaways

We propose an approach to **generate BPMN** processes:

- **Fully automated**, tested, and available online
- **Correctly generating** the process in **more than 80%** of the cases

Takeaways

We propose an approach to **generate BPMN** processes:

- **Fully automated**, tested, and available online
- **Correctly generating** the process in **more than 80%** of the cases
- Providing **guarantees** regarding the **semantics** of the process

Takeaways

We propose an approach to **generate BPMN** processes:

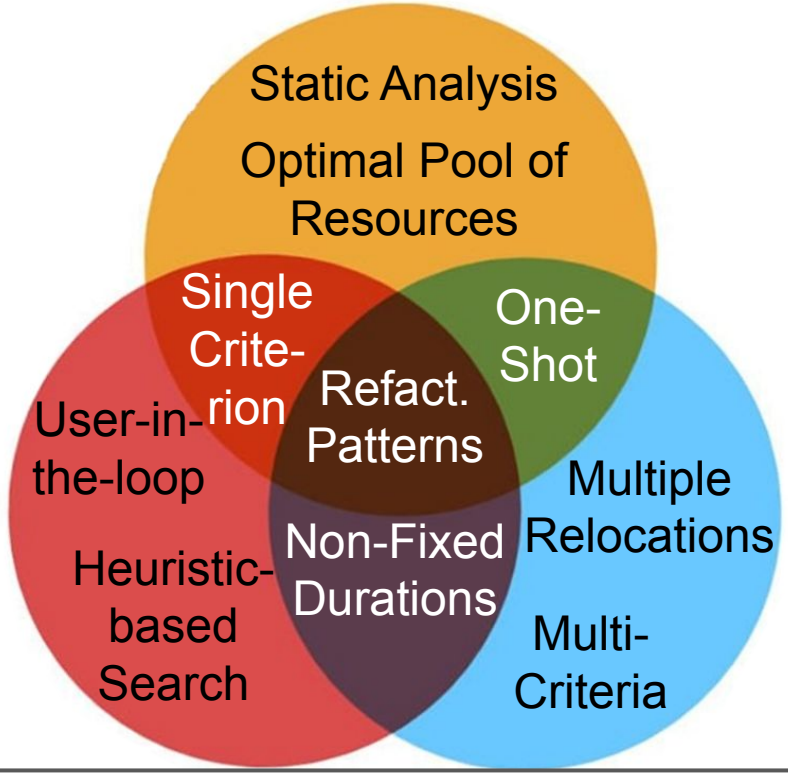
- **Fully automated**, tested, and available online
- **Correctly generating** the process in **more than 80%** of the cases
- Providing **guarantees** regarding the **semantics** of the process
- Including **behavioural verification** facilities

Takeaways

We propose an approach to **generate BPMN** processes:

- **Fully automated**, tested, and available online
- **Correctly generating** the process in **more than 80%** of the cases
- Providing **guarantees** regarding the **semantics** of the process
- Including **behavioural verification** facilities

We propose 3 approaches to **refactor BPMN** processes:



Perspectives

Regarding the **generation of processes**, we thought about several **perspectives**.

- **Cross-check** the generated expressions **with** other **LLMs** — Short-term
- Add **further information** during generation (resources, durations, …) — Mid-term
- Provide **advices** to **improve** the **quality** of the process
- **Enlarge** the supported BPMN **syntax** — Long-term
- **Synchronise the description** with the process changes — Transversal

Perspectives

Regarding the **refactoring of processes**, we thought about several **perspectives**.

- ➢ Explore possibilities offered by **scheduling** techniques — Short-term
- ➢ Look for better **heuristics**
- ➢ Extend the **support** (BPMN syntax, model of resources) — Mid-term
- ➢ **Remove sequence graphs** to increase the support
- ➢ **Limit** the usage of **simulation** (AI, SMT, analytics, …) — Long-term

References [1/5]

- [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.
- [DGKV2011]: *Identifying Refactoring Opportunities in Process Model Repositories*, Remco M. Dijkman, Beat Gfeller, Jochen Malte Küster, and Hagen Völzer, 2011.
- [DN2011]: *jMetal: A Java Framework for Multi-objective Optimization*, Juan José Durillon, Antonio Jesus Nebro, 2011.

References [2/5]

- [DRS2019]: *A Rewriting Approach to Resource Allocation Analysis in Business Process Models*, Francisco Duran, Camilo Rocha, and Gwen Salaün, 2019
- [DRS2021]: *Resource Provisioning Strategies for BPMN Processes: Specification and Analysis Using Maude*, Francisco Duran, Camilo Rocha, and Gwen Salaün, 2021
- [DS2022]: *Optimization of BPMN Processes via Automated Refactoring*, Francisco Duran and Gwen Salaün, 2022.
- [FRPCP2013]: *Graph-Based Business Process Model Refactoring*, Maria Fernandez-Ropero, Ricardo Pérez-Castillo, and Mario Piattini, 2013.
- [FSZ2024]: *Dynamic Resource Allocation for Executable BPMN Processes Leveraging Predictive Analytics*, Yliès Falcone, Gwen Salaün, and Ahang Zuo, 2024.

References [3/5]

- [Geambasu2012]: *BPMN vs. UML Activity Diagram for Business Process Modeling*, Cristina Venera Geambasu, 2012.
- [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.
- [KL2022]: *Business Workflow Optimization through Process Model Redesign*, Akhil Kumar and Rong Liu, 2022.

References [4/5]

- [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.
- [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.
- [RM2005]: *Best Practices in Business Process Redesign: An Overview and Qualitative Evaluation of Successful Redesign Heuristics*, H. A. Reijers and S. Liman Mansar, 2005.

References [5/5]

- [SM2007]: *Refactoring BPMN Models: From `Bad Smells' to Best Practices and Patterns*, Darius Silingas and Edita Mileviciene, 2007.
- [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.

Scientific Integrity Oath

"En présence de mes pairs.
Parvenu à l'issue de mon doctorat en 'Informatique', et ayant ainsi pratiqué, dans ma quête du savoir, l'exercice d'une recherche scientifique exigeante, en cultivant la rigueur intellectuelle, la réflexivité éthique et dans le respect des principes de l'intégrité scientifique, je m'engage, pour ce qui dépendra de moi, dans la suite de ma carrière professionnelle quel qu'en soit le secteur ou le domaine d'activité, à maintenir une conduite intègre dans mon rapport au savoir, mes méthodes et mes résultats."

"In the presence of my peers.
With the completion of my doctorate in 'Computer science', in my quest for knowledge, I have carried out demanding research, demonstrated intellectual rigour, ethical reflection, and respect for the principles of research integrity.
As I pursue my professional career, whatever my chosen field, I pledge, to the greatest of my ability, to continue to maintain integrity in my relationship to knowledge, in my methods and in my results."