TÉCNICO LISBOA

Enterprise Integration

Business Process Management

Prof. Sérgio Guerreiro
Sergio.guerreiro@tecnico.ulisboa.pt

Department of Computer Science and Engineering
Instituto Superior Técnico / Universidade de Lisboa
INESC-ID
URL: http://www.inesc-id.pt
Rua Alves Redol, 9
1000-029 Lisboa
Portugal

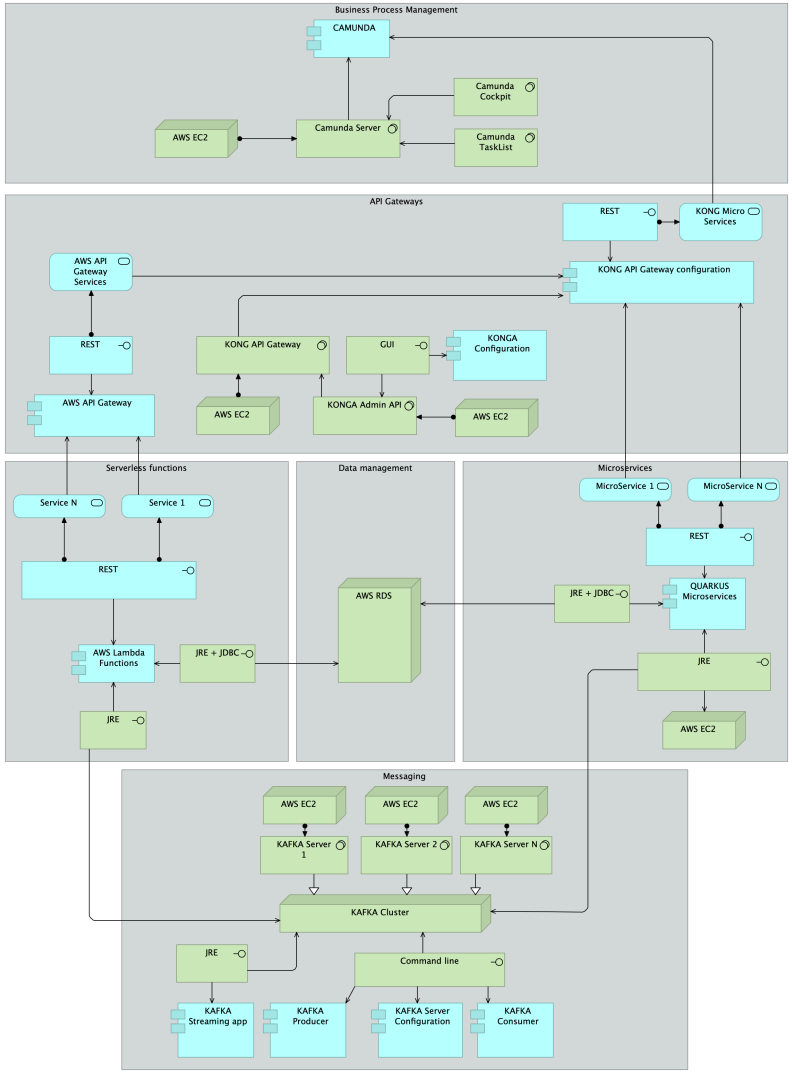
Camunda Engine is the business process executor that consumes the available microservices in the enterprise.

The goal is to enable the integration with all the remaining enterprise systems and to enable the representation and execution of the sequence of business behaviours in an easy way.

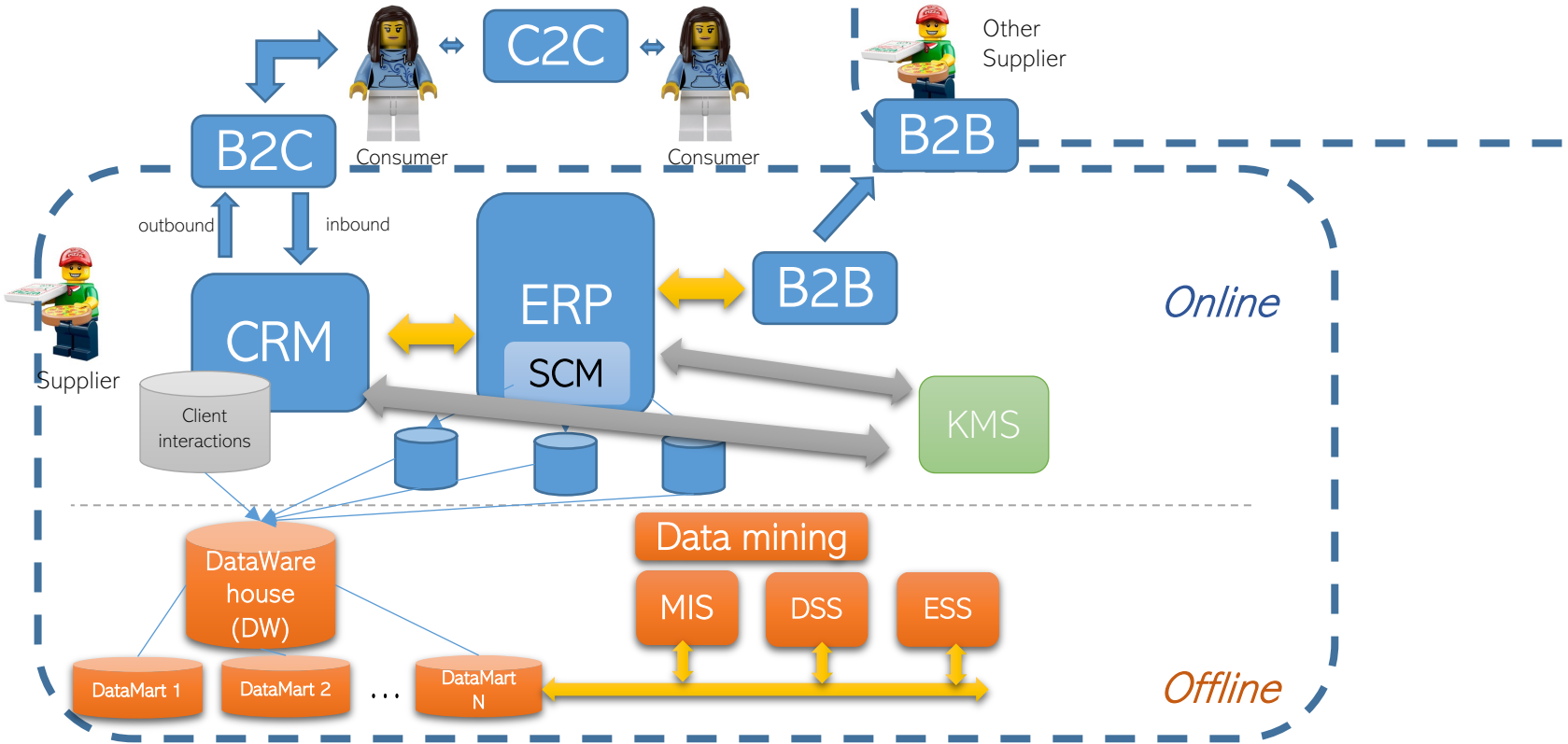
The process-oriented nature allows a closer approach to the business development of the enterprise.

IE Technology Stack

Overall view



ERP as the placeholder for Business Processes

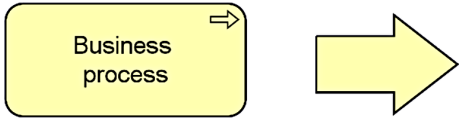


The precursors: **the Workflow systems**

- Workflow Management System is a software platform that supports the design, development, implementation and analysis of workflows.
- Create new applications as an exercise of composition (existing functions) rather them the traditional development of a new application by conventional programming
- Programming based on models, represented as straight forward visual constructs (visual programming)
- Workflows were considered, since the 90's also as a strategic approach for application integration
- For all of the above reasons considered as: programming in the large

TÉCNICO LISBOA

Then, What is a business process?



8.3.1 Business Process

A business process represents a sequence of business behaviors that achieves a specific result such as a defined set of products or business services.

A business process describes the internal behavior performed by a business role that is required to produce a set of products and services. For a consumer, the products and services are relevant and the required behavior is merely a black box, hence the designation “internal”.

A complex business process may be an aggregation of other, finer-grained processes. To each of these, finer-grained roles may be assigned.

There is a potential many-to-many relationship between business processes and business functions. Informally speaking, processes describe some kind of “flow” of activities, whereas functions group activities according to required skills, knowledge, resources, etc.

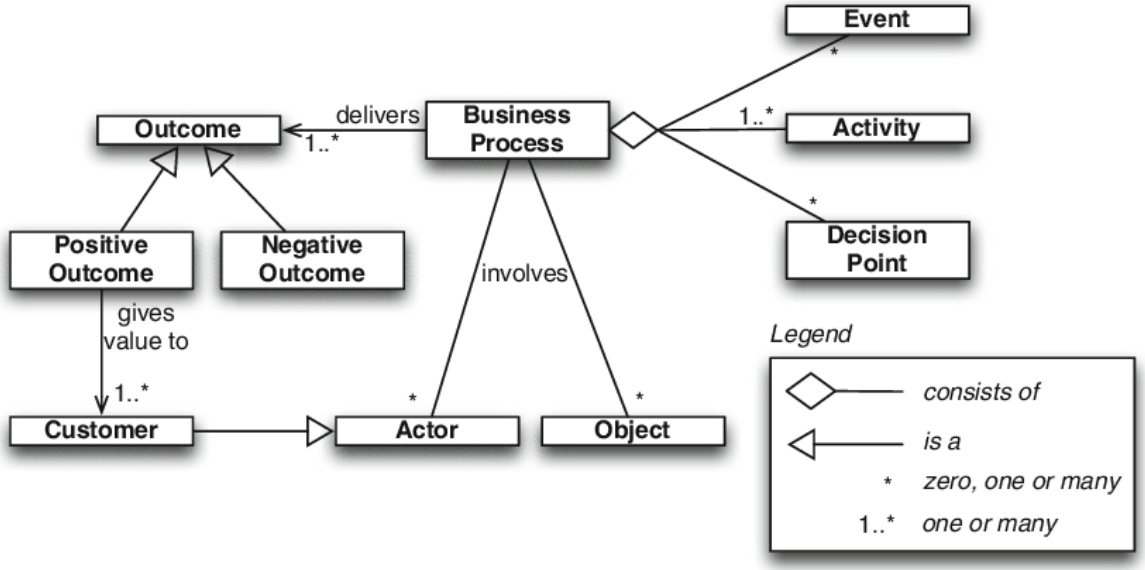
A business process may be triggered by, or trigger, any other business behavior element (e.g., business event, business process, business function, or business interaction). A business process may access business objects. A business process may realize one or more business services and may use (internal) business services or application services. A business role may be assigned to a business process to perform this process manually. An automated business process can be realized by an application process. The name of a business process should clearly indicate a predefined sequence of actions using a verb or verb-noun combination and may include the word “process”. Examples are “adjudicate claim”, “employee on-boarding”, “approval process”, or “financial reporting”.

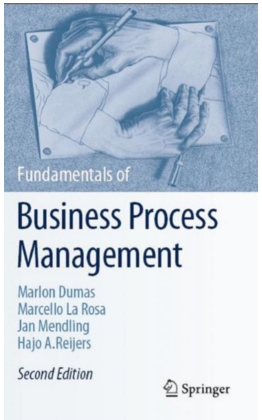
In an ArchiMate model, the existence of business processes is depicted. High-level business, end-to-end processes, macro flows, and workflows can all be expressed with the same business process element in the ArchiMate language. It does not, however, list the flow of activities in detail. This is typically done during business process modeling, where a business process can be expanded using a business process design language; e.g., BPMN [12].

https://pubs.opengroup.org/architecture/archimate3-doc/chap08.html#_Toc10045374

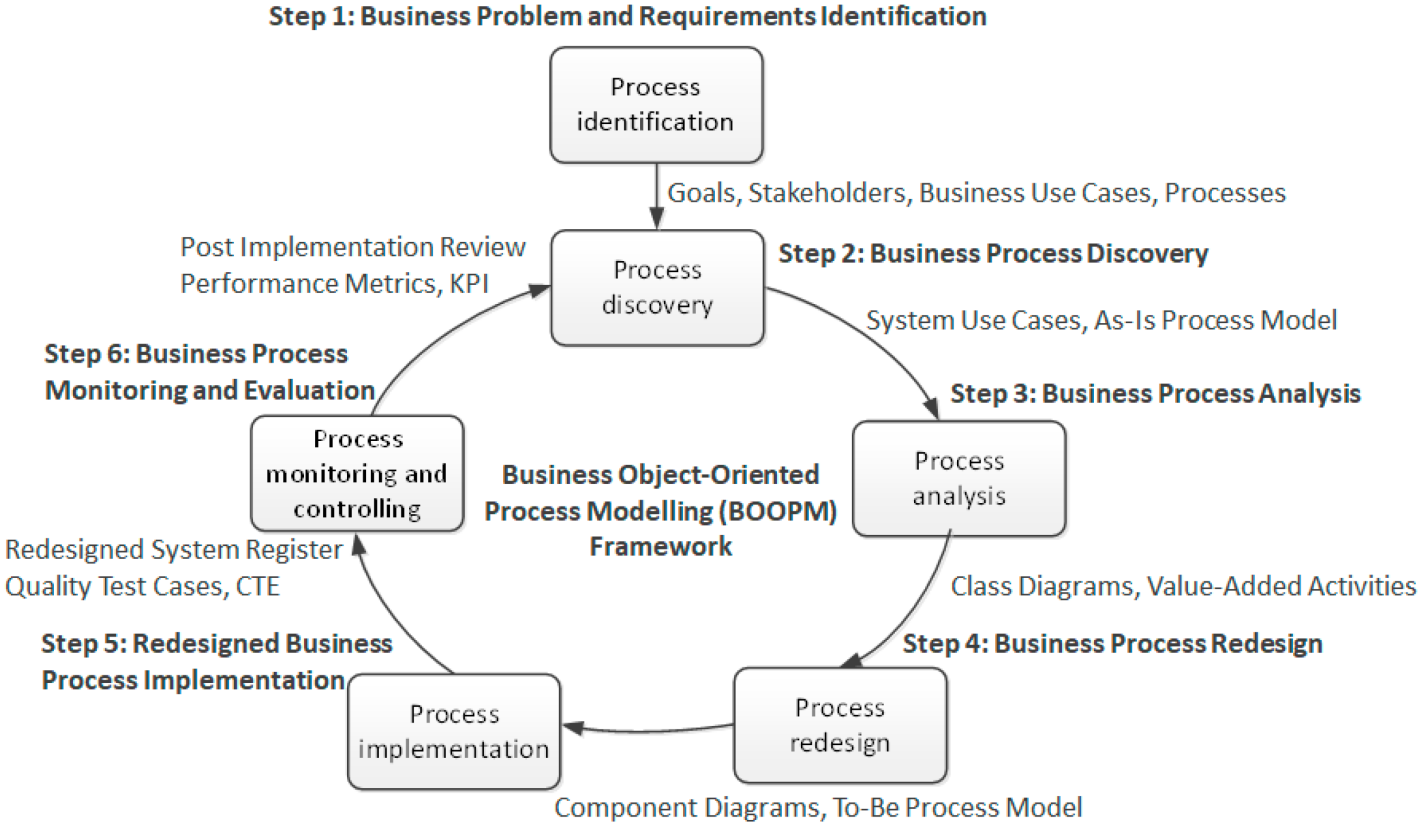
Business Process definition

- Set of interrelated activities that transform inputs into outputs in order to produce a service or product **to a specific customer**.
- The focus is always on the final product!!!





A (classic) reference framework for the BPM lifecycle



Process Innovation and Improvement Using Business Object-Oriented Process Modelling (BOOPM) Framework
https://www.mdpi.com/2571-5577/2/3/23/htm

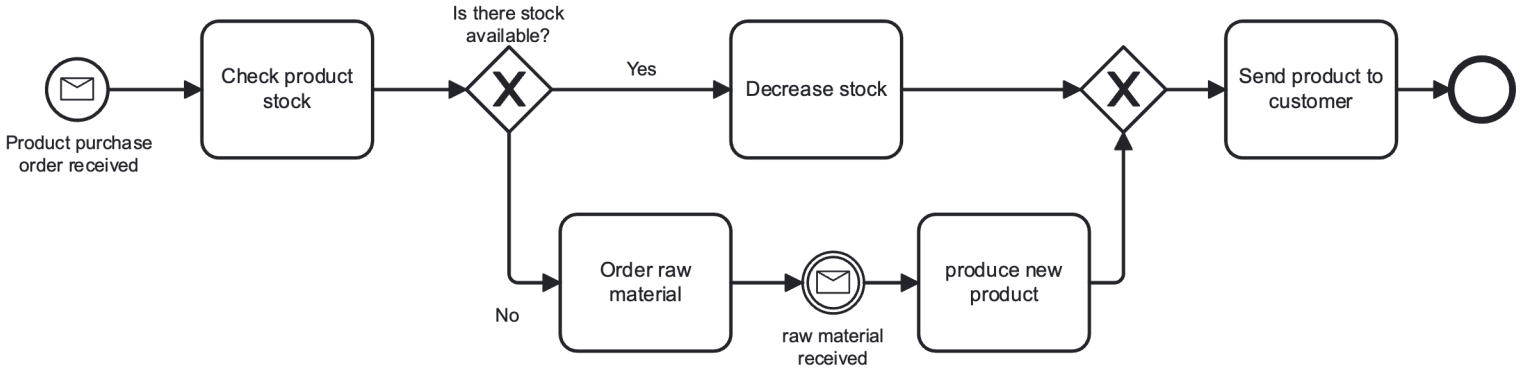
On “Process” versus “Business Process”

Process:

- “A set of interrelated and cooperative activities that transform inputs into outputs”. (ISO 9000)

Business Process:

- “A collection of activities that takes one or more kinds of input and creates an output that is of value to the customer.” (Hammer & Champy 1993)



Hello World! BPMN

When a seller receives a product purchase order, he always checks the available stock.

If there is stock available, update it and send the product to the customer.

If there is no stock available, order raw material and wait for it to be received. Then the new product is produced and sent to the customer.

“Business” Process Model and Notation (BPMN)

- BPMN is a OMG notation for Business Process Modelling.
- BPMN allows the graphical specification of business processes.
- BPMN can be mapped to execution languages and SOA environments…

- Timeline:
  - 2004: BPMI (Business Process Management Initiative) publishes BPMN 1.0
  - 2005: BPMI is integrated into OMG (Object Management Group)
  - 2006: OMG officially adopted the notation and publishes BPMN 1.0

  - 2008: BPMN 1.2 published
  - 2009: BPMN 2.0 draft release
  - 2011: BPMN 2.0 published - www.omg.org/spec/BPMN/2.0/

http://www.omg.org/bpmn/

…all about BPMN: http://www.bpmn.org/

Modelling in BPMN – Process types

TÉCNICO LISBOA

2.1 Process Modeling Conformance

The next eight sections describe **Process Modeling Conformance**.

2.1.1 BPMN Process Types

The implementations claiming **Process Modeling Conformance** MUST support the following **BPMN** packages:

- The **BPMN** core elements, which include those defined in the *Infrastructure*, *Foundation*, *Common*, and *Service* packages (see Chapter 8).
- **Process** diagrams, which include the elements defined in the **Process**, **Activities**, *Data*, and *Human Interaction* packages (see Chapter 10).
- **Collaboration** diagrams, which include **Pools** and **Message Flow** (see Chapter 9).
- **Conversation** diagrams, which include **Pools**, **Conversations**, and **Conversation Links** (see Chapter 9).

As an alternative to full **Process Modeling Conformance**, there are three conformance sub-classes defined:

- **Descriptive**
- **Analytic**
- **Common Executable**

Descriptive is concerned with visible elements and attributes used in high-level modeling. It should be comfortable for analysts who have used BPA flowcharting tools.

Analytic contains all of **Descriptive** and in total about half of the constructs in the full **Process Modeling Conformance** Class. It is based on experience gathered in BPMN training and an analysis of user-patterns in the Department of Defense Architecture Framework and planned standardization for that framework.

Both **Descriptive** and **Analytic** focus on visible elements and a minimal subset of supporting attributes/elements.

Common Executable focuses on what is needed for executable process models.

Elements and attributes not in these sub-classes are contained in the full **Process Modeling Conformance** class.

The elements for each sub-class are defined in the next section.

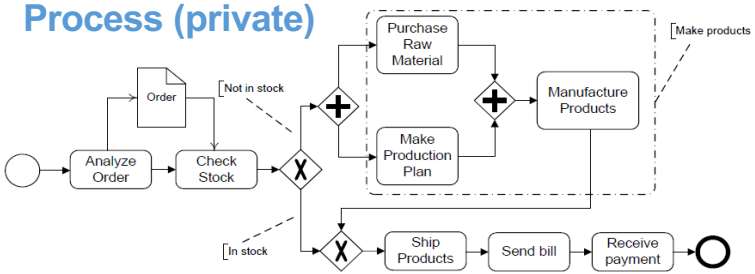
Types of BPMN Diagrams

- **Process**
  - Represents the public or private processes of a participant.
  - Focus on representing the (internal) orchestration of a process.
  - The participant can be subdivided into multiple Lanes.
  - All external pools (if any) must be black-box.
- **Collaboration**
  - Represents the message exchange between two or more participants .
  - Focus on representing the orchestration of a process across multiple participants.
- **Choreography**
  - Represent the information pertaining to each participant in the choreography.
  - The focus is not on orchestrations of the work performed within these Participants, but rather on the exchange of information between Participants.
- **Conversation**
  - Conversation diagrams visualize messages exchange between pools.
  - Design workflow with business process diagram and visualize communications with BPMN conversation diagrams.

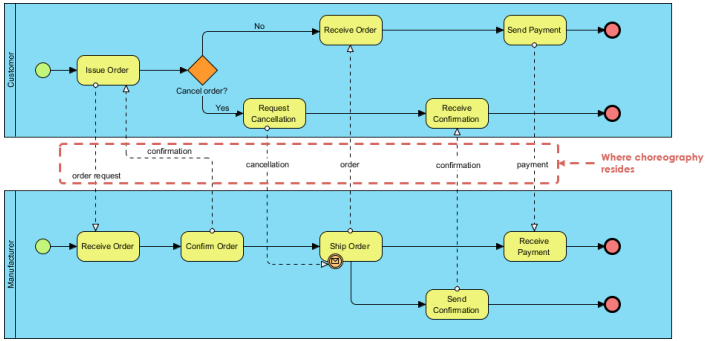
Types of BPMN Diagrams

TÉCNICO LISBOA

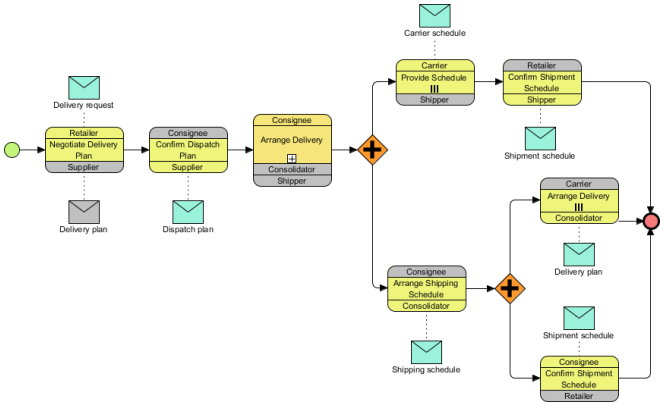
Process (private)

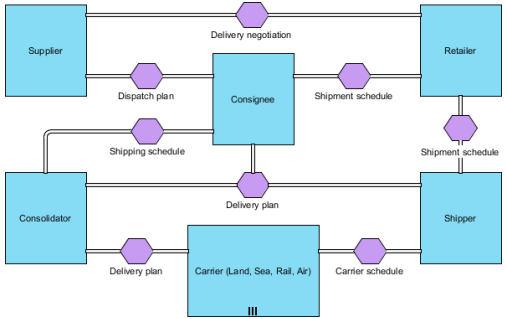


Collaboration



Choreography

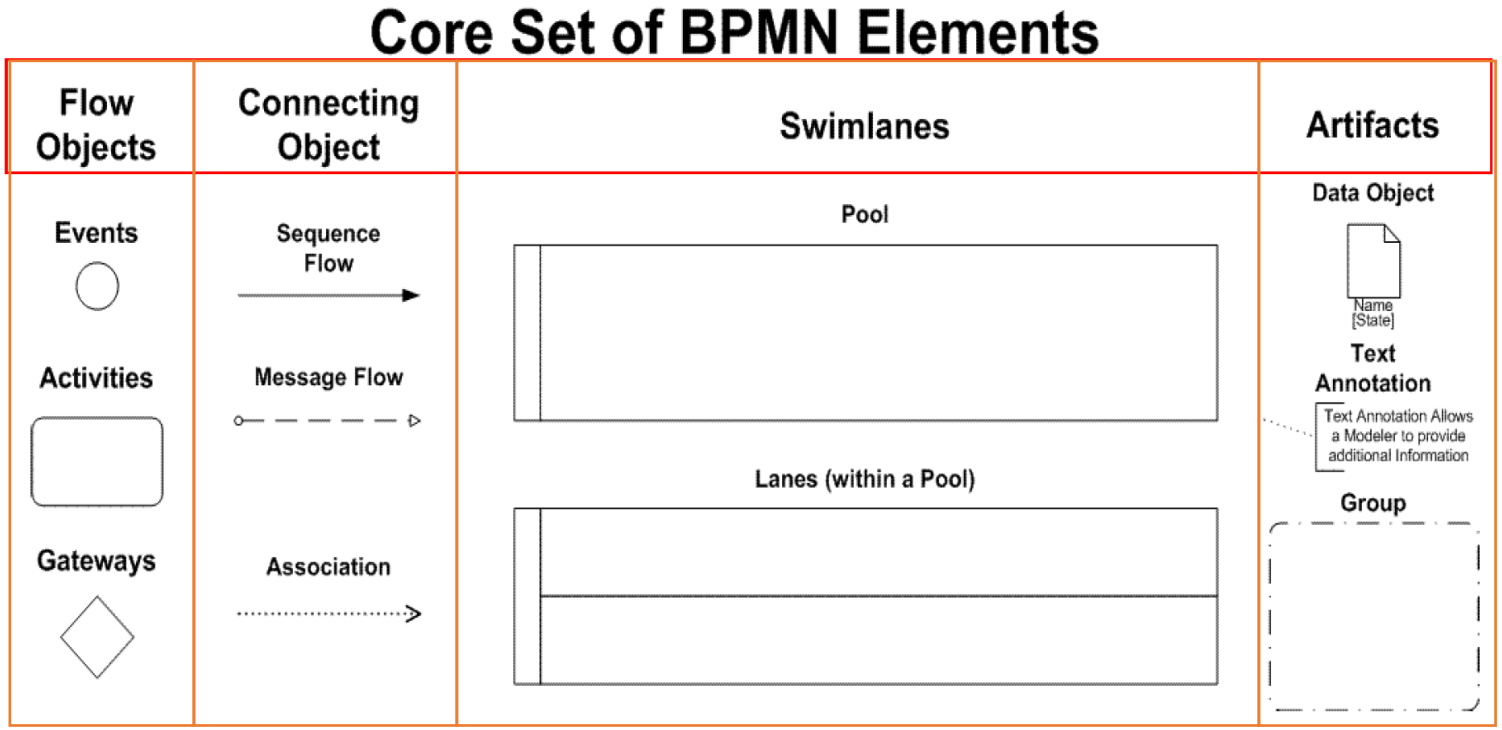




Conversation

Examples of Collaboration, Choreography and Conversation from:
https://www.visual-paradigm.com/guide/bpmn/bpmn-orchestration-vs-choreography-vs-collaboration/

Core BPMN Elements



Pools and Lanes

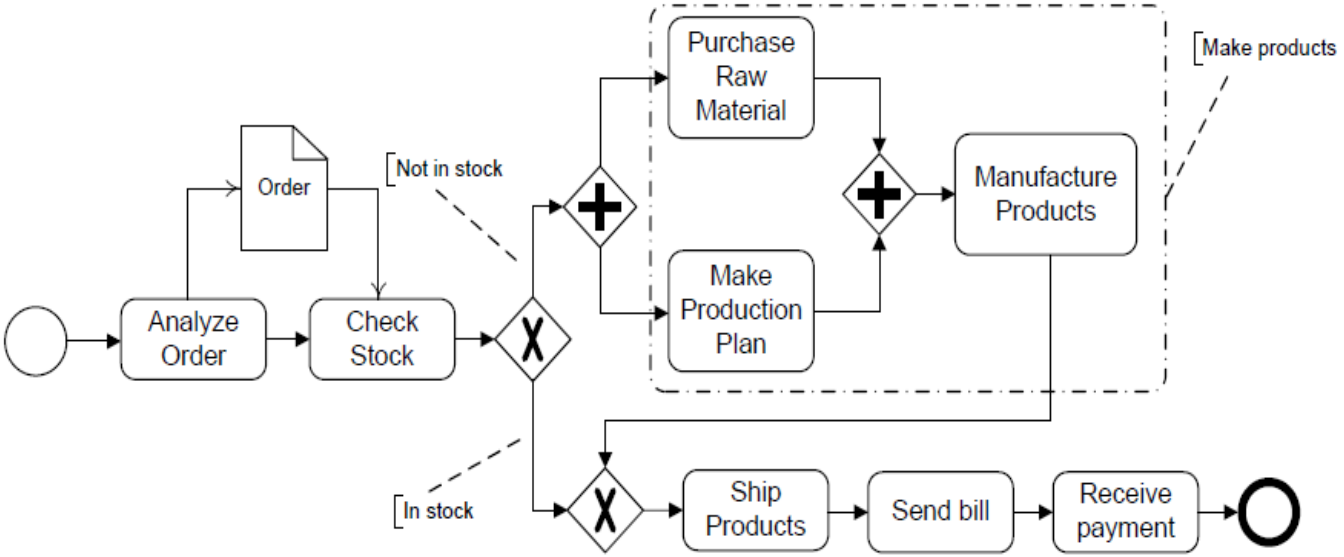
Pool: represents a process Participant.

- Sequence flows cannot cross Pool boundaries
- Message flows can cross Pool boundaries

Lane: a sub-division of a Pool. Used to organize and categorize the activities of a Participant.

- Sequence flows can cross Lane boundaries

Private Process Diagram (without pools)



- A **Process** focuses on a single **Participant**.
- A **Private Process** focuses on a **Participant** internal to the organization.

Private Process with one Pool and nested Lanes

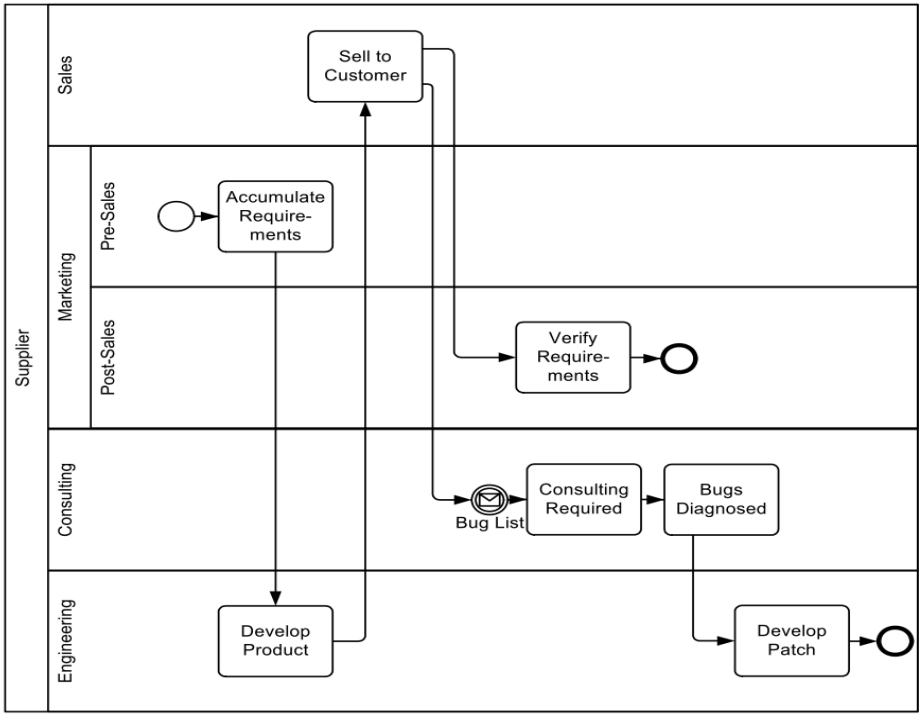


Figure 10.125 - An Example of Nested Lanes

This **Process** focuses on a single **Participant** with multiple Lanes. Therefore it is a **private Process.**

Public vs. Private Process

Figure 7.1 - Example of a *private* Business Process

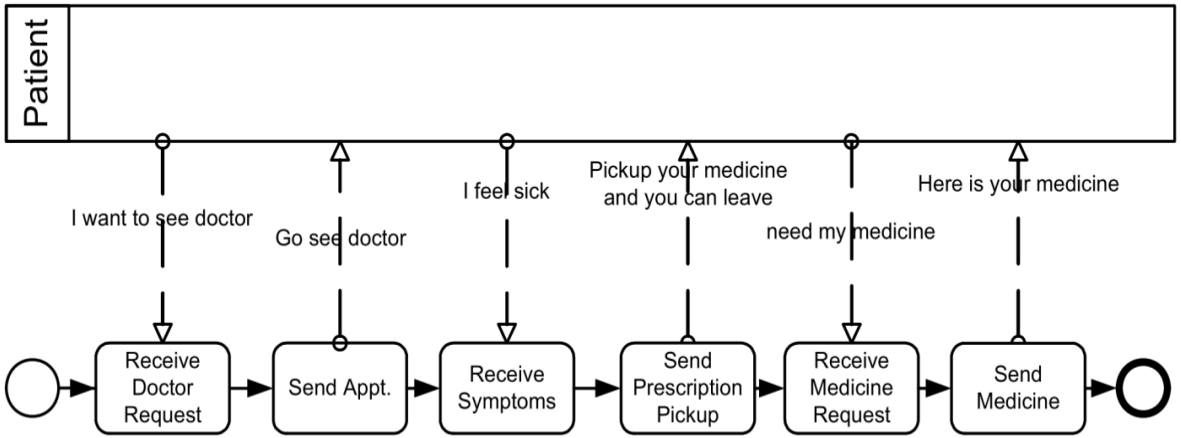
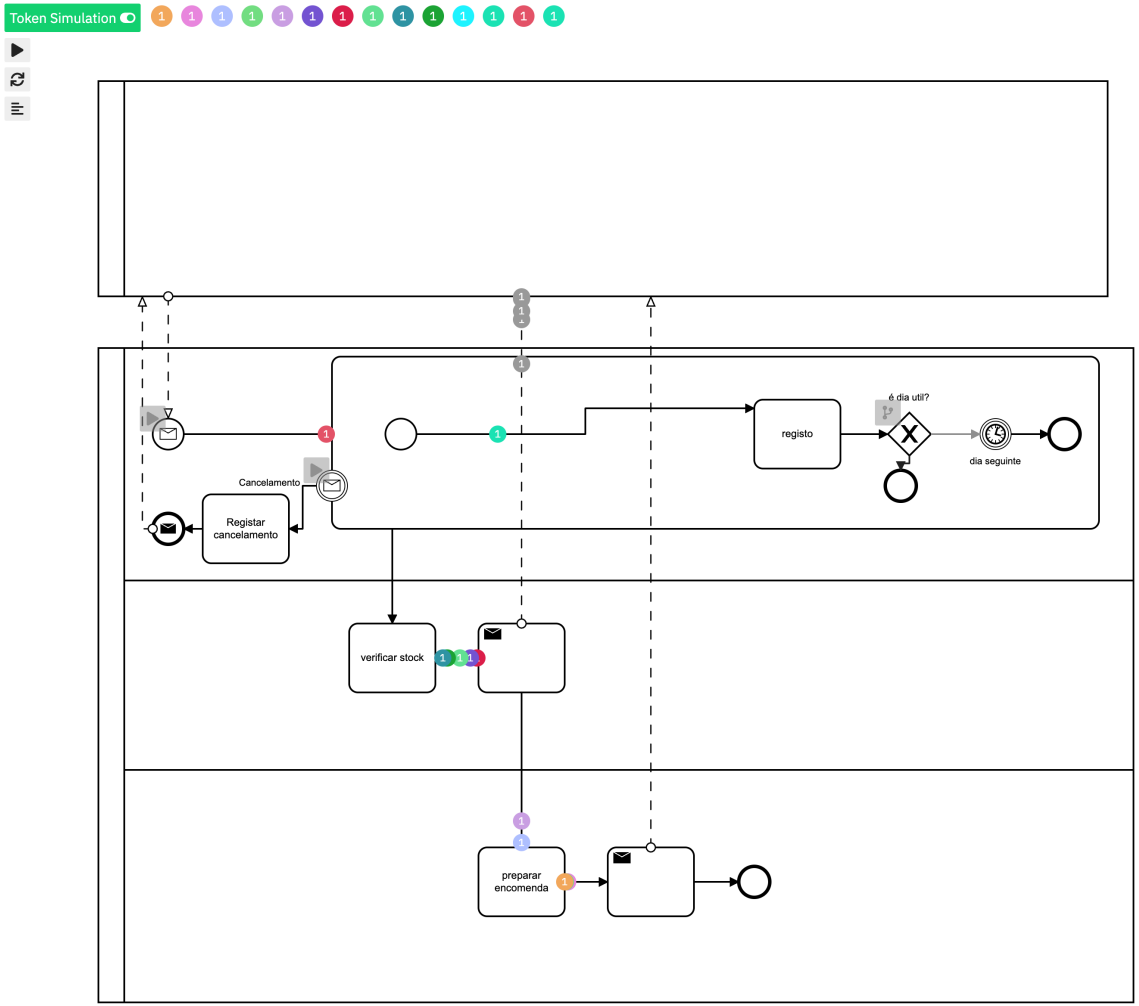


Figure 7.2 - Example of a *public* Process

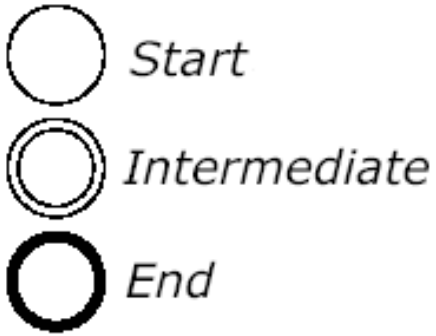


Process Definition *vs.* Process Instance

Flow Objects

- **Activity**: a unit of work.

- **Event**: an occurrence during a business process.



- **Gateway**: controls the flow of activities.

Connectors

Sequence Flow

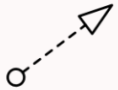
defines the execution order of activities.

Default Flow

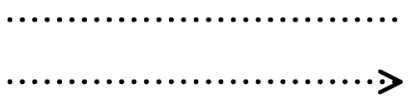
is the default branch to be chosen if all other conditions evaluate to false.

Conditional Flow

has a condition assigned that defines whether or not the flow is used.

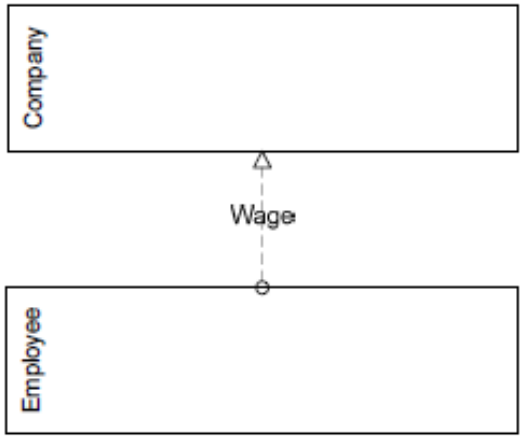


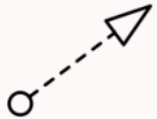
Message Flow
symbolizes information flow across organizational boundaries. Message flow can be attached to pools, activities, or message events.



An Association is used to link information and Artifacts with BPMN graphical elements (see page 67). Text Annotations (see page 71) and other Artifacts (see page 66) can be Associated with the graphical elements. An arrowhead on the Association indicates a direction of flow (e.g., data), when appropriate.

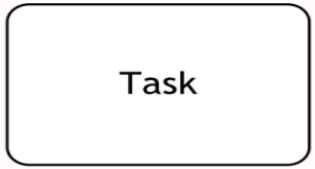
Message Flow





Message Flow symbolizes information flow across organizational boundaries. Message flow can be attached to pools, activities, or message events.

Activities



A **Task** is a unit of work, the job to be performed. When marked with a [+] symbol it indicates a **Sub-Process,** an activity that can be refined.

Activity Markers

Markers indicate execution behavior of activities:

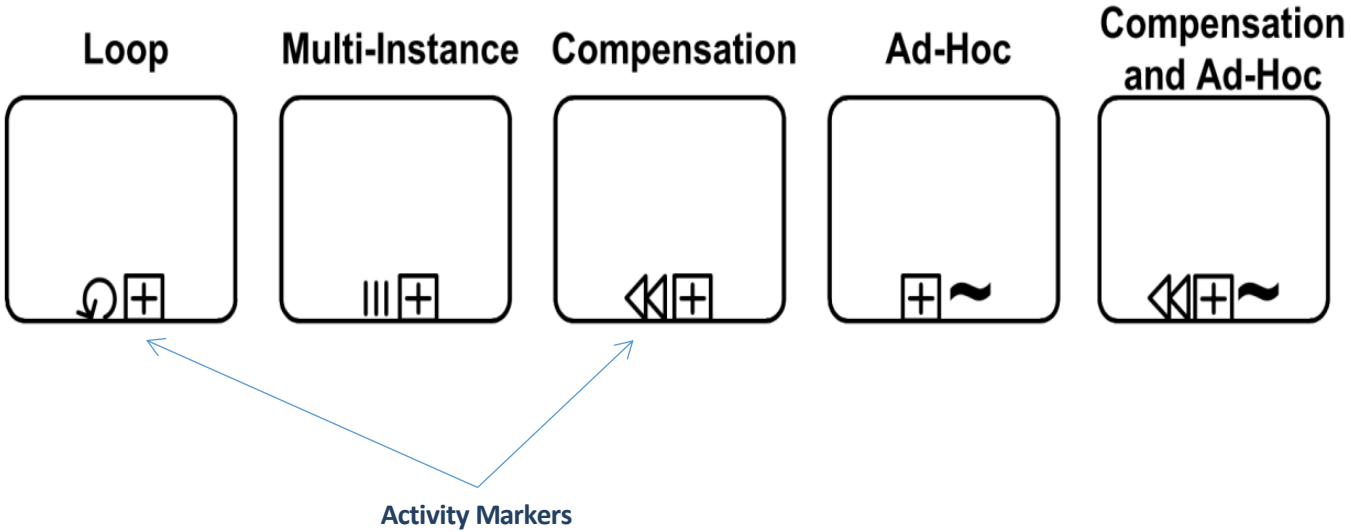
- Sub-Process Marker
- Loop Marker
- Parallel MI Marker
- Sequential MI Marker
- Ad Hoc Marker
- Compensation Marker

Task Types

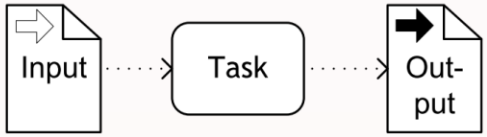
Types specify the nature of the action to be performed:

- Send Task
- Receive Task
- User Task
- Manual Task
- Business Rule Task
- Service Task

Activity Types

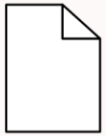


Data



A **Data Input** is an external input for the entire process. It can be read by an activity.

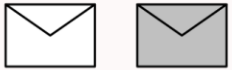
A **Data Output** is a variable available as result of the entire process.



A **Data Object** represents information flowing through the process, such as business documents, e-mails, or letters.

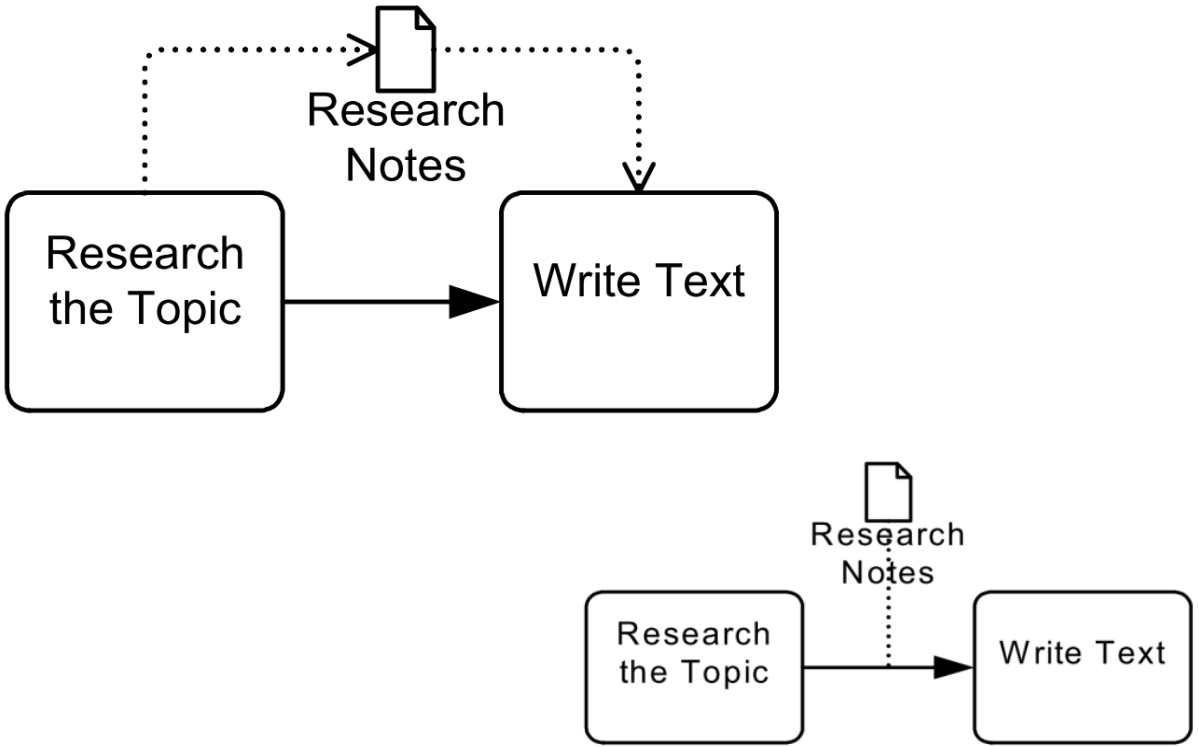
A **Collection Data Object** represents a collection of information, e.g., a list of order items.

A **Data Store** is a place where the process can read or write data, e.g., a database or a filing cabinet. It persists beyond the lifetime of the process instance.



A **Message** is used to depict the contents of a communication between two Participants.

Data Association within a Process



Event Triggers

- A Trigger specifies what causes the Event.
- The Trigger is shown as an icon inside the Event symbol.
- BPMN defines several types of Triggers:
  - None (i.e. no Trigger) and
  - 12 other (Message, Timer, Error, Signal,...)
- A Trigger may have two different **behaviours**:

Throw a Trigger  (throw/send a message)

Catch a Trigger  (catch/receive a message)

Event Behaviour (throw/catch)

- When **throwing** a Trigger the Event **waits for a token** and then ***produces* the Trigger**.
  The notation for throw is a *filled* Trigger.

- When **catching** a Trigger the Event **waits for a token** and then ***waits* for the Trigger**.
  The notation for catch is an *outlined* Trigger.
  "Catch" is *blocking* while it waits for the Trigger and the Token.

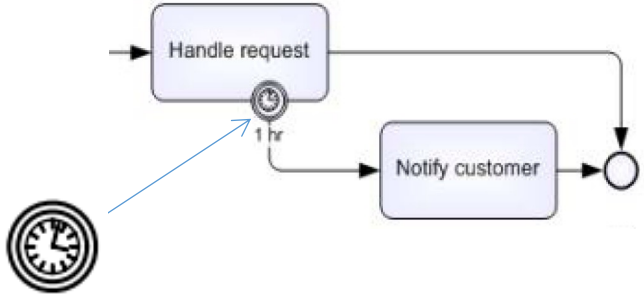
- **Start events**: can only **catch**
- **Intermediate events**: can **catch** and **throw**
- **End events**: can only **throw**

Interrupting and Non-Interrupting Events

Interrupting

Interrupting: when the event is thrown or caught the corresponding activity is interrupted and is not completed.

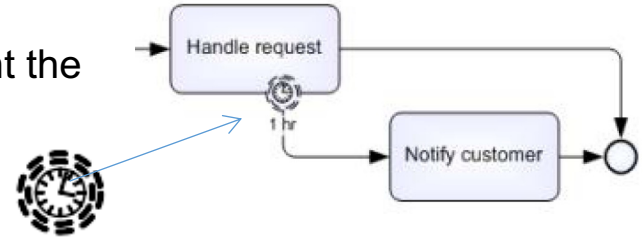
Notation is a **solid line**.



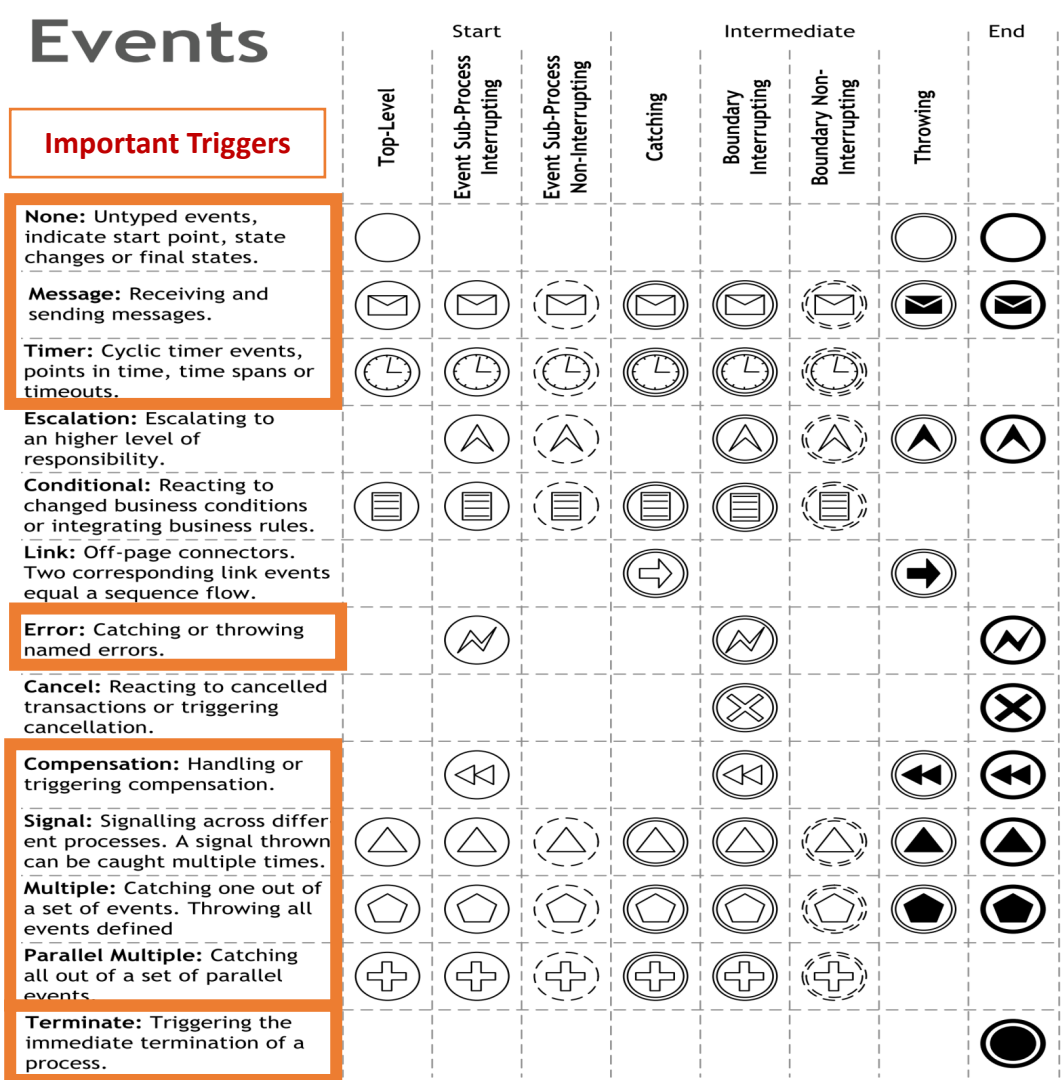
Non-Interrupting

Non-Interrupting: when the event is thrown or caught the corresponding activity continues its execution.

Notation is a **dotted line**.



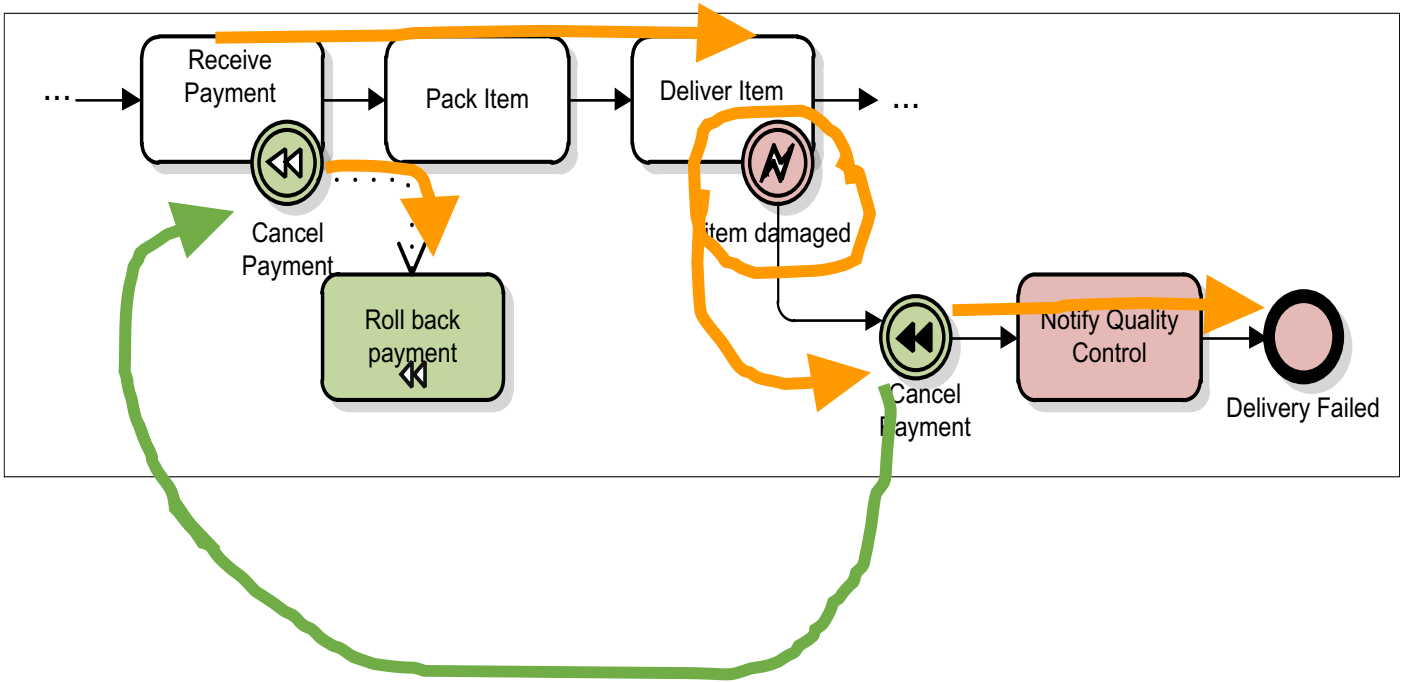
Events


Events
Start
Intermediate
End
Top-Level
Event Sub-Process Interrupting
Event Sub-Process Non-Interrupting
Catching
Boundary Interrupting
Boundary Non-Interrupting
Throwing
Important Triggers
None: Untyped events, indicate start point, state changes or final states.
Message: Receiving and sending messages.
Timer: Cyclic timer events, points in time, time spans or timeouts.
Escalation: Escalating to an higher level of responsibility.
Conditional: Reacting to changed business conditions or integrating business rules.
Link: Off-page connectors. Two corresponding link events equal a sequence flow.
Error: Catching or throwing named errors.
Cancel: Reacting to cancelled transactions or triggering cancellation.
Compensation: Handling or triggering compensation.
Signal: Signalling across different processes. A signal thrown can be caught multiple times.
Multiple: Catching one out of a set of events. Throwing all events defined
Parallel Multiple: Catching all out of a set of parallel events.
Terminate: Triggering the immediate termination of a process.

Compensation

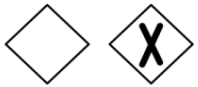
- Sometimes you need to undo the effects of a business process. You do this by rolling it backwards, one completed Activity at a time
- There are three options for undoing each completed Activity:
  1. Do nothing - there are no data changes to undo
  2. Overwrite - restore all data to its original state
  3. Undo - perform a specific Activity to undo the data changes - this is known as *compensation*

Errors and Compensation



Gateways

Exclusive Gateway



When splitting, it routes the sequence flow to exactly one of the outgoing branches. When merging, it awaits one incoming branch to complete before triggering the outgoing flow.

Event-based Gateway

Is always followed by catching events or receive tasks. Sequence flow is routed to the subsequent event/task which happens first.

Parallel Gateway

When used to split the sequence flow, all outgoing branches are activated simultaneously. When merging parallel branches it waits for all incoming branches to complete before triggering the outgoing flow.

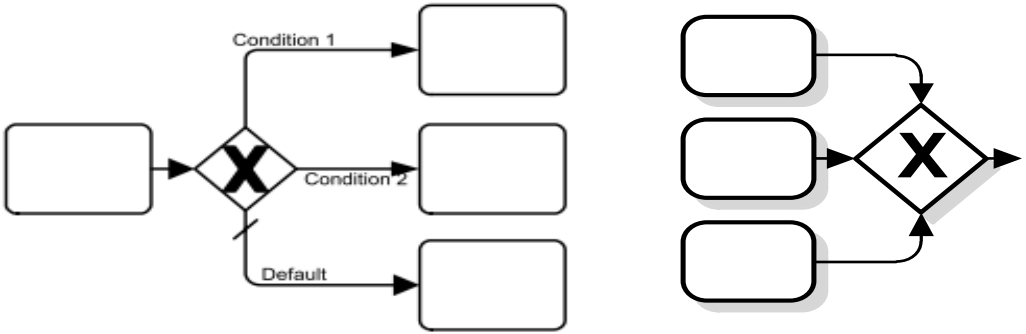
Inclusive Gateway
When splitting, one or more branches are activated. All active incoming branches must complete before merging.

Exclusive Event-based Gateway (instantiate)
Each occurrence of a subsequent event starts a new process instance.

Complex Gateway
Complex merging and branching behavior that is not captured by other gateways.

Parallel Event-based Gateway (instantiate)
The occurrence of all subsequent events starts a new process instance.

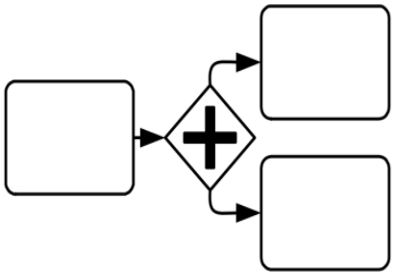
Exclusive Gateway (XOR-split/merge)



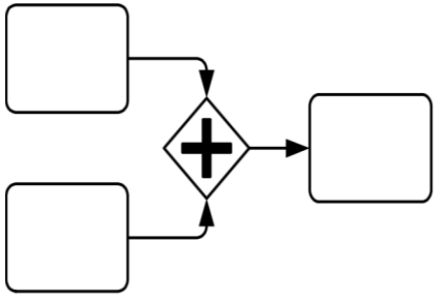
- XOR-split selects one and only one of the 2..* output flows.
- The conditions are evaluated top-to-bottom; the output flow associated to first condition evaluated true is selected.
- Optionally, a *default* flow (--/--) may be included. The default flow is unconditional and is selected whenever all the other conditions are evaluated false.
- The output flows of a XOR-split may be merged using a XOR-merge that continues as soon as one input flow arrives.

Parallel (AND-split/merge)

- AND-split forks one input flow into 2..* parallel (concurrent in time) output flows.
- AND-merge waits for *all* input flows before joining them into a single output flow.



AND-split (fork)



AND-merge(join)

Event-Based Gateway

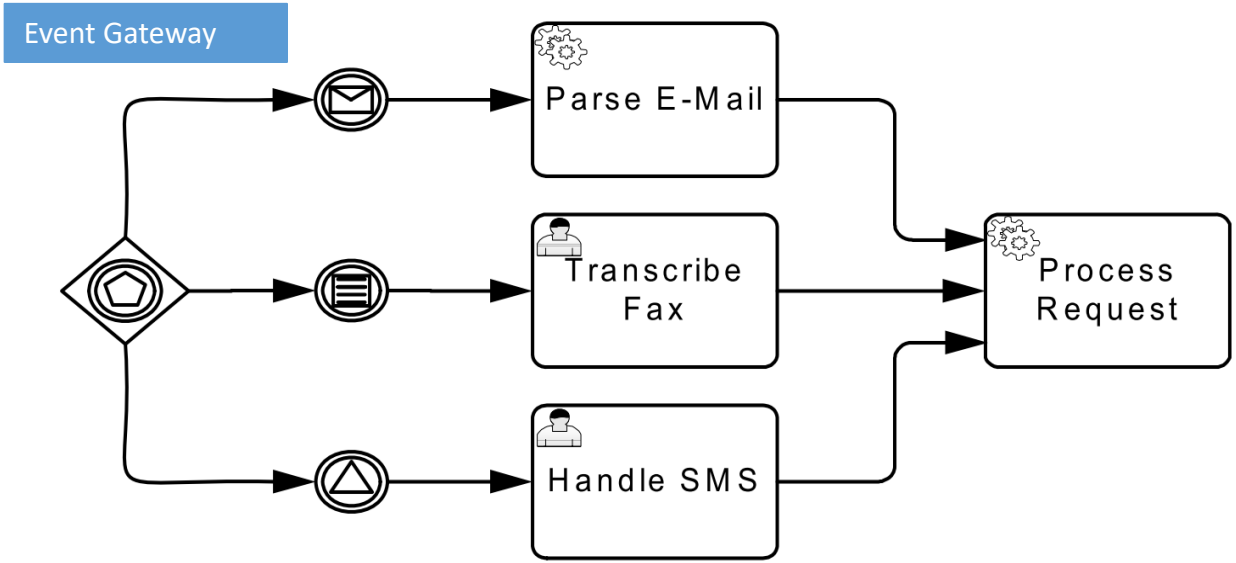
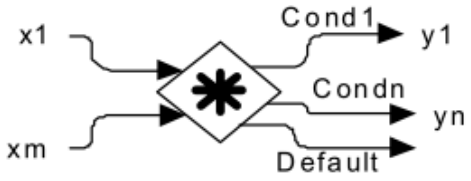


Figure 10.98 - A Process initiated by an Event-Based Gateway

Complex Gateway

- These have no default semantics!
  - Splitting - modeller provides an IncomingCondition
  - Merging - modeller provides an OutgoingCondition
- They *must* be supported by Text Annotations that describe their semantics, otherwise the BPD is unreadable!



The **Complex Gateway** can be used to model complex synchronization behavior. An `Expression activationCondition` is used to describe the precise behavior. For example, this `Expression` could specify that *tokens* on three out of five *incoming* **Sequence Flows** are needed to activate the **Gateway**. What *tokens* are produced by the **Gateway** is determined by conditions on the *outgoing* **Sequence Flows** as in the split behavior of the **Inclusive Gateway**. If *tokens* arrive later on the two remaining **Sequence Flows**, those *tokens* cause a reset of the **Gateway** and new *token* can be produced on the *outgoing* **Sequence Flows**. To determine whether it needs to wait for additional *tokens* before it can reset, the **Gateway** uses the synchronization semantics of the **Inclusive Gateway**.

TÉCNICO LISBOA

Executable Business Process

Modelling an executable BPMN

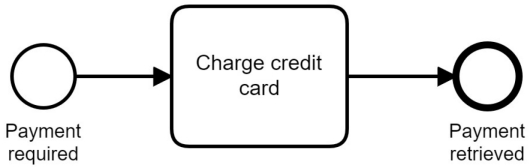


Figure 1-2. A very simple process, that can already handle many requirements in the credit card example

Practical Process Automation

by Bernd Ruecker

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

As shown in Figure 1-3 the workflow engine:

- keeps the state persistent,
- schedules the retries,
- gives you visibility into current process instances and historic information.

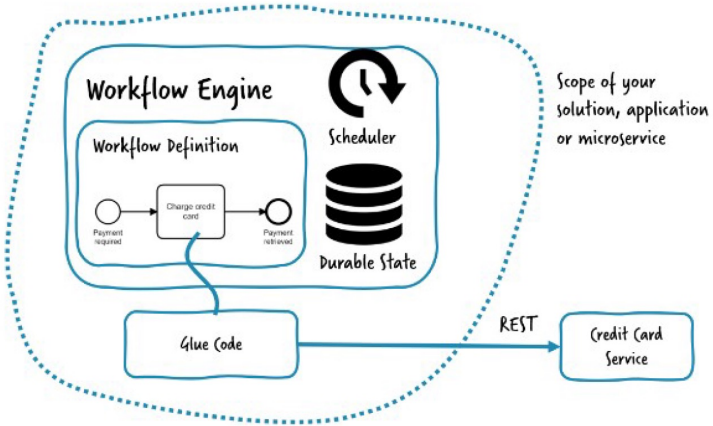


Figure 1-3. Workflow engine

Modelling an **executable** BPMN

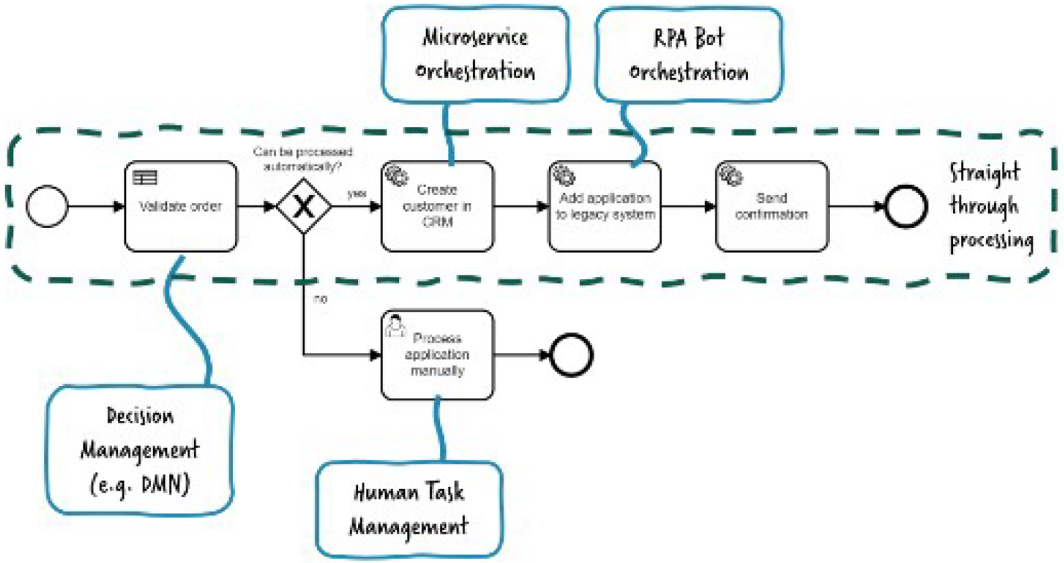


Figure 1-5. Use cases are typically mixed in real-life examples

Practical Process Automation

by Bernd Ruecker

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

Modelling an **executable** BPMN

TÉCNICO LISBOA

Process automation is about automating tasks within a process as well as the automation of the control flow between these tasks.

Three different combinations possible which define a kind of maturity levels of process automation:
1. The **human controls** the process, often passing paper around. You're probably familiar with this from physical inbox folders on your desk in an office environment. Software and office tools can, of course, make this process more efficient.

2. The **computer controls** the process and involves people whenever necessary, for example using task list user interfaces.

3. The **whole process** is fully automated and only requires manual intervention if something happens beyond the expected normal operations.

Practical Process Automation

by Bernd Ruecker

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North,
Sebastopol, CA 95472.

Modelling an **executable** BPMN

TÉCNICO LISBOA

Two types of automation:

1. **Automation of the control flow**: The interactions between tasks are automated, but the activities itself might not. In the previous example, this was the humans cooking the food. A process might not be fully automated as it cannot run without human interaction. This is often known as human task management.

2. **Automation of the tasks**: The task itself is automated. In the previous example, this would be the robots who cook the food. Automating the tasks leads to fully automated processes, which is typically named STP (straight through processing). Very often this is the most ideal path, where everything runs smoothly and is fully automated. Humans are involved in case any unforeseen exceptions or special cases occur. This approach is actually quite powerful, as you can invest the effort on automation on the majority of cases and save additional (manual) effort only on the rare cases.

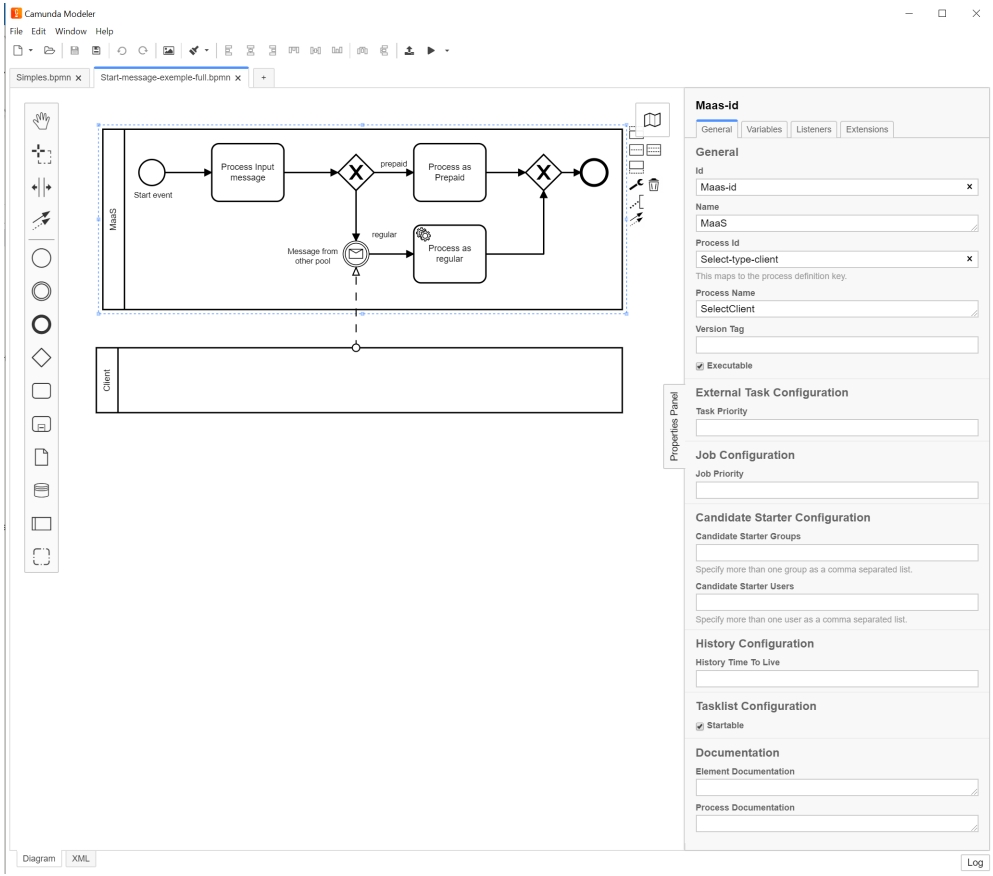
Practical Process Automation

by Bernd Ruecker

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North,
Sebastopol, CA 95472.

Modelling an executable BPMN

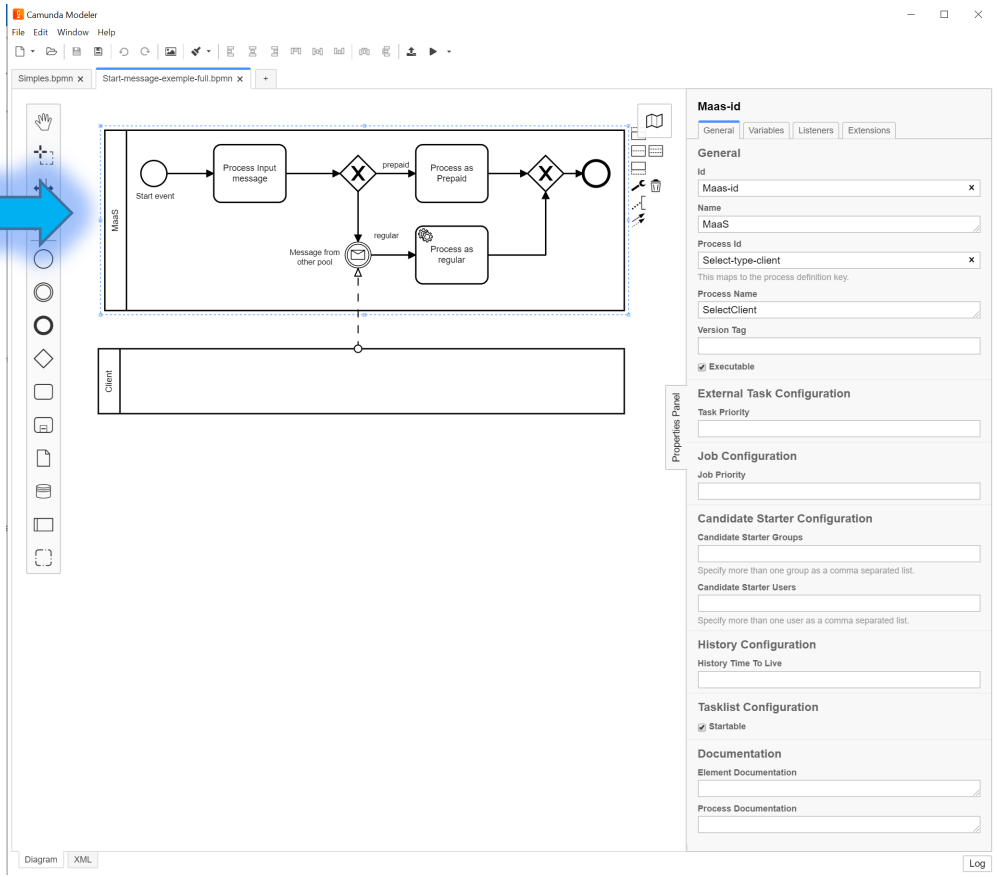


Modelling an **executable** BPMN – Separation of systems

Each system should be **modelled** as a separate pool

Then, each pool uses communication between them sending and receiving messages

Later, in terms of **execution**, a process engine supports many systems or each system a different engine



TÉCNICO LISBOA

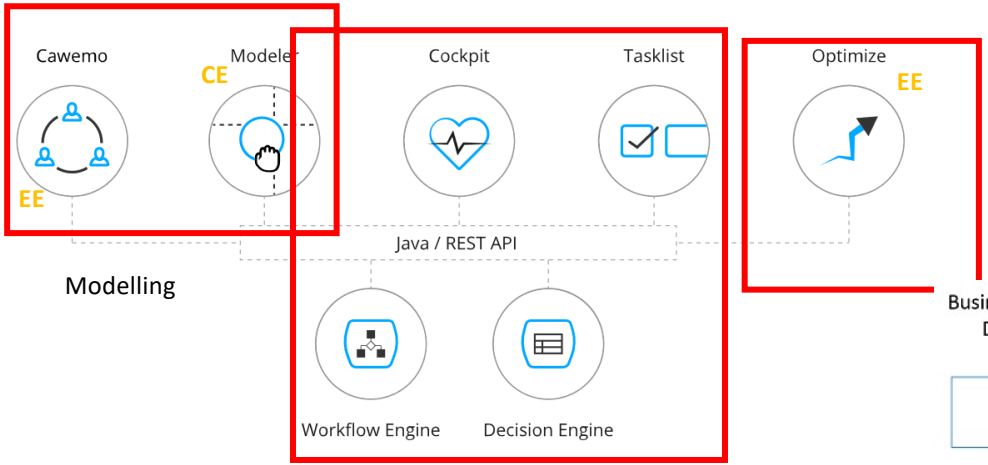
A Business Process Execution engine

The case of Camunda

TÉCNICO LISBOA

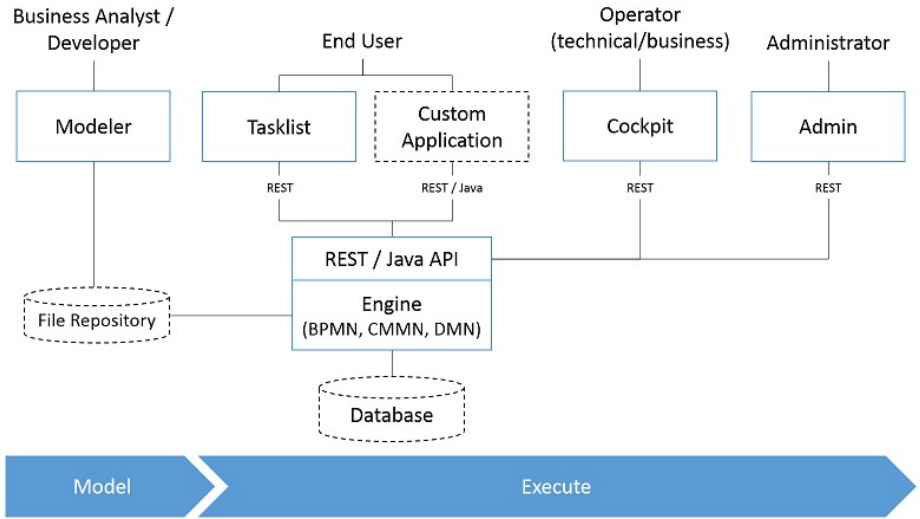
Camunda Architecture

Camunda Components

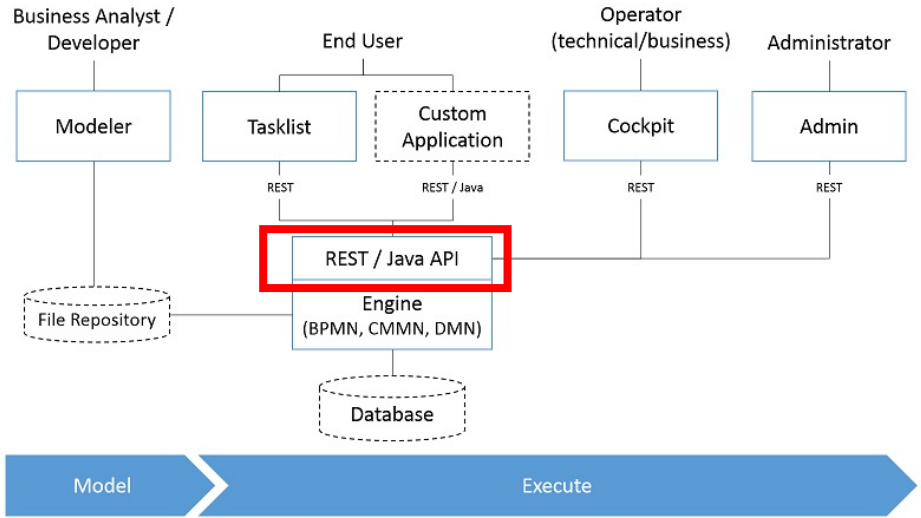


2 edições para optar:

CE - Community edition
EE – Enterprise edition

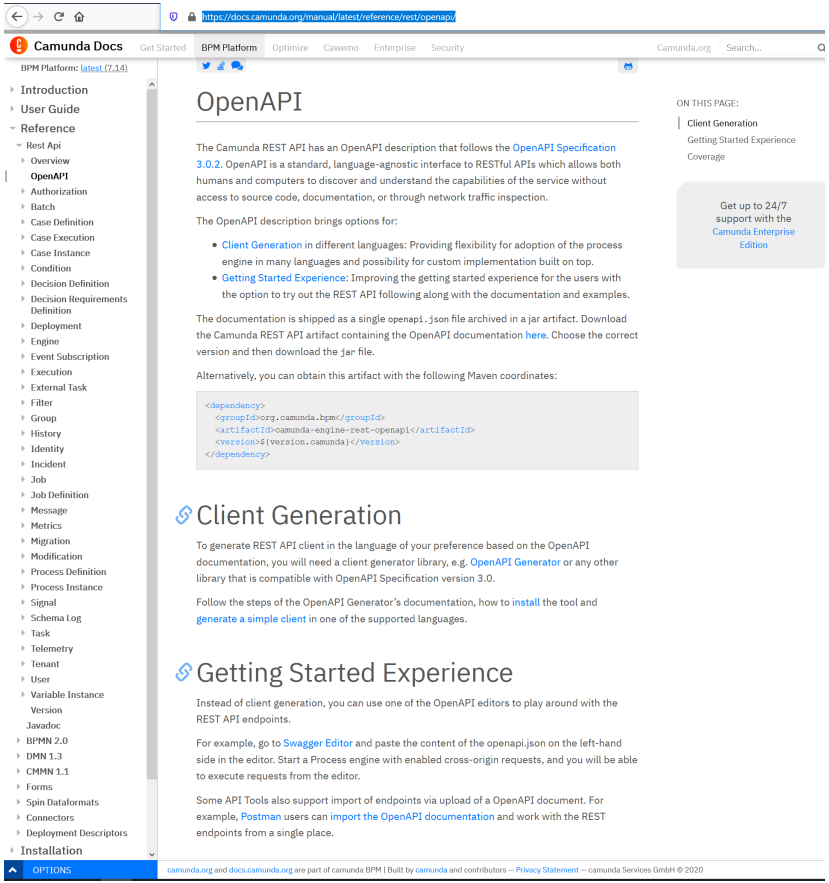


The Camunda REST API is provided by an OpenAPI description



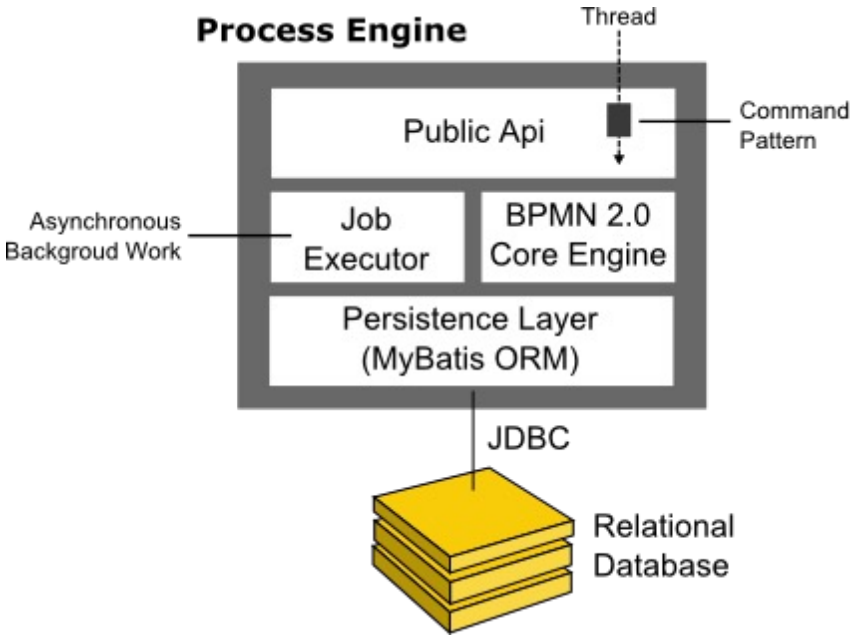
Process engine functionalities are exposed by REST API and thus can be used by other applications

Camunda Architecture



https://docs.camunda.org/manual/latest/reference/rest/openapi/

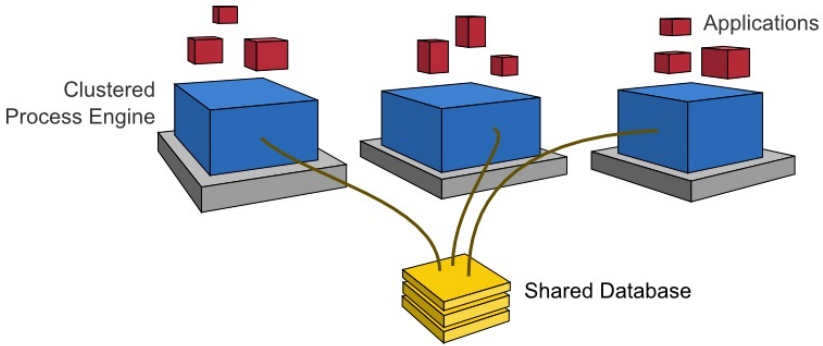
Camunda Architecture



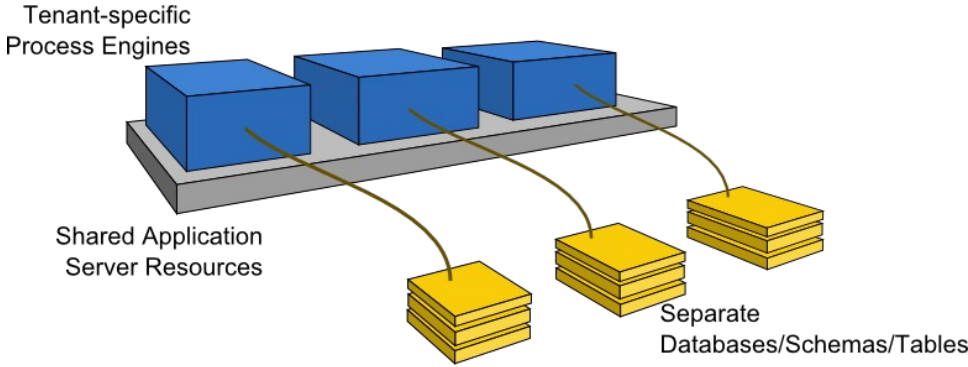
- Process Engine Public API: Service-oriented API allowing Java applications to interact with the process engine. The different responsibilities of the process engine (i.e., Process Repository, Runtime Process Interaction, Task Management, ...) are separated into individual services. The public API features a command-style access pattern:
- **BPMN 2.0 Core Engine**: this is the core of the process engine. It features a lightweight execution engine for graph structures (PVM - Process Virtual Machine), a BPMN 2.0 parser which transforms BPMN 2.0 XML files into Java Objects and a set of BPMN Behavior implementations (providing the implementation for BPMN 2.0 constructs such as Gateways or Service Tasks).
- Job Executor: the Job Executor is responsible for processing asynchronous background work such as Timers or asynchronous continuations in a process.
- **The Persistence Layer**: the process engine features a persistence layer responsible for persisting process instance state to a relational database.

https://stage.docs.camunda.org/manual/7.3/guides/user-guide/#process-engine-transactions-in-processes-understanding-asynchronous-continuations

Camunda Deployment Models



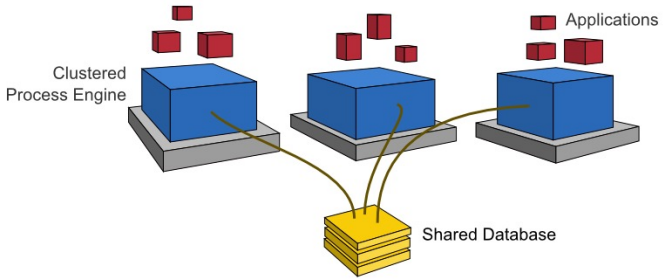
Clustering

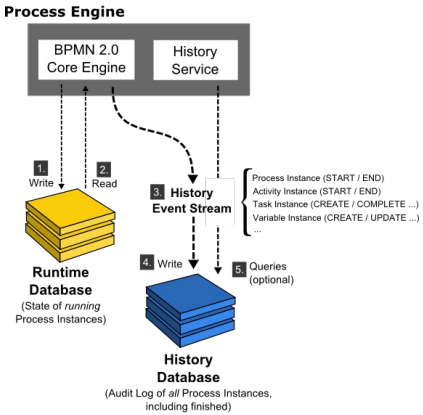


Multi-tenancy

https://stage.docs.camunda.org/manual/7.3/guides/user-guide/#process-engine-transactions-in-processes-understanding-asynchronous-continuations

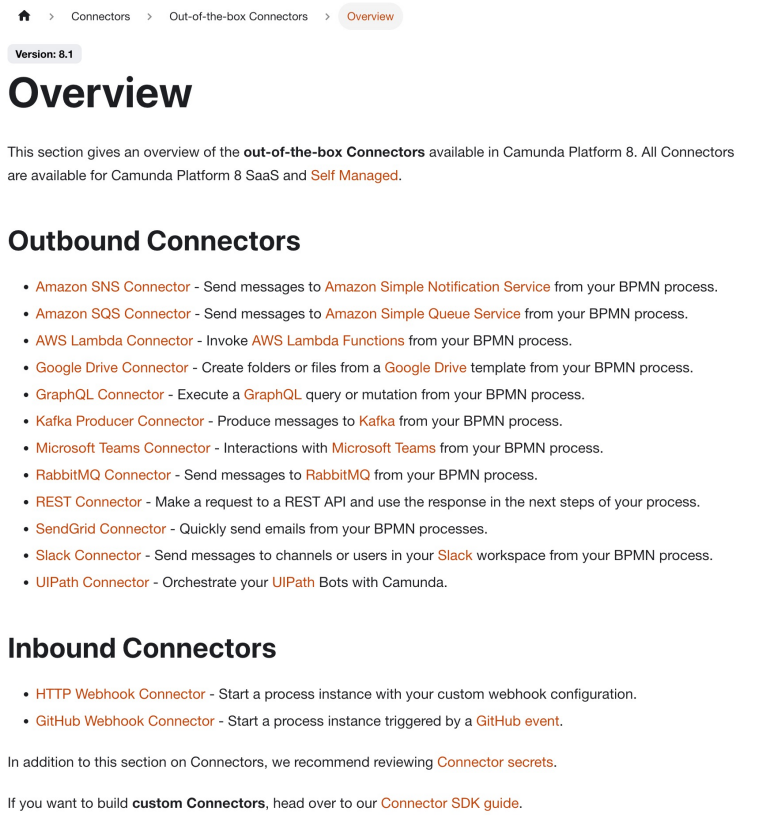
Camunda Deployment & Operation environment





- **JAVA** compatibility:
  - Oracle JDK 8 / 9 / 10 / 11 / 12 / 13 / 14 / 15
  - IBM JDK 8 (with J9 JVM)
  - OpenJDK 8 / 9 / 10 / 11 / 12 / 13 / 14 / 15, including builds of the following products: Oracle OpenJDK, AdoptOpenJDK (with HotSpot JVM), Amazon Corretto, Azul Zulu
- **Process Engine, Camunda cockpit, Tasklist** e **Admin** compatibility:
  - Apache Tomcat 9.0,
  - JBoss EAP 7.0 / 7.1 / 7.2 / 7.3 / 7.4,
  - Wildfly Application Server 13.0 / 14.0 / 15.0 / 16.0 / 17.0 / 18.0 / 19.0 / 20.0 / 21.0 / 22.0 / 23.0,
  - IBM WebSphere Application Server 8.5 / 9.0 (Enterprise Edition only)
  - Oracle WebLogic Server 12c (12R2) (Enterprise Edition only)
- **Shared database** compatibility:
  - MySQL 5.7 / 8.0
  - MariaDB 10.2 / 10.3
  - Oracle 12c / 19c
  - IBM DB2 11.1 (excluding IBM z/OS for all versions)
  - PostgreSQL 9.6 / 10 / 11 / 12 / 13
  - Amazon Aurora PostgreSQL compatible with PostgreSQL 9.6 / 10.4 / 10.7 / 10.13 / 12.4
  - Microsoft SQL Server 2014/2016/2017/2019 (see Configuration Note)
  - Microsoft Azure SQL with Camunda-supported SQL Server compatibility levels (see Configuration Note):
    - SQL Server on Azure Virtual Machines
    - Azure SQL Managed Instance
    - Azure SQL Database
  - H2 1.4 (not recommended for Cluster Mode - see Deployment Note)
  - CockroachDB v20.1.3 (see Configuration guide for more details)
- **Clustering** – active/ative, with shared database for process execution
- **Kubernetes** for dynamic installation
- **Separation** between execution data and historical data
- **Optimize** available with Enterprise Edition

Camunda 8 connectors



Connectors > Out-of-the-box Connectors > Overview

Version: 8.1

Overview

This section gives an overview of the **out-of-the-box Connectors** available in Camunda Platform 8. All Connectors are available for Camunda Platform 8 SaaS and Self Managed.

Outbound Connectors

- Amazon SNS Connector - Send messages to Amazon Simple Notification Service from your BPMN process.
- Amazon SQS Connector - Send messages to Amazon Simple Queue Service from your BPMN process.
- AWS Lambda Connector - Invoke AWS Lambda Functions from your BPMN process.
- Google Drive Connector - Create folders or files from a Google Drive template from your BPMN process.
- GraphQL Connector - Execute a GraphQL query or mutation from your BPMN process.
- Kafka Producer Connector - Produce messages to Kafka from your BPMN process.
- Microsoft Teams Connector - Interactions with Microsoft Teams from your BPMN process.
- RabbitMQ Connector - Send messages to RabbitMQ from your BPMN process.
- REST Connector - Make a request to a REST API and use the response in the next steps of your process.
- SendGrid Connector - Quickly send emails from your BPMN processes.
- Slack Connector - Send messages to channels or users in your Slack workspace from your BPMN process.
- UIPath Connector - Orchestrate your UIPath Bots with Camunda.

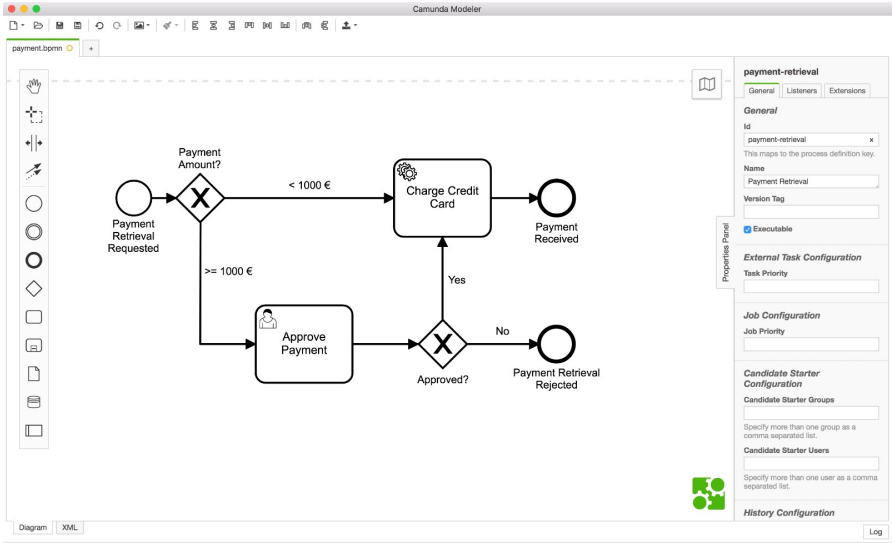
Inbound Connectors

- HTTP Webhook Connector - Start a process instance with your custom webhook configuration.
- GitHub Webhook Connector - Start a process instance triggered by a GitHub event.

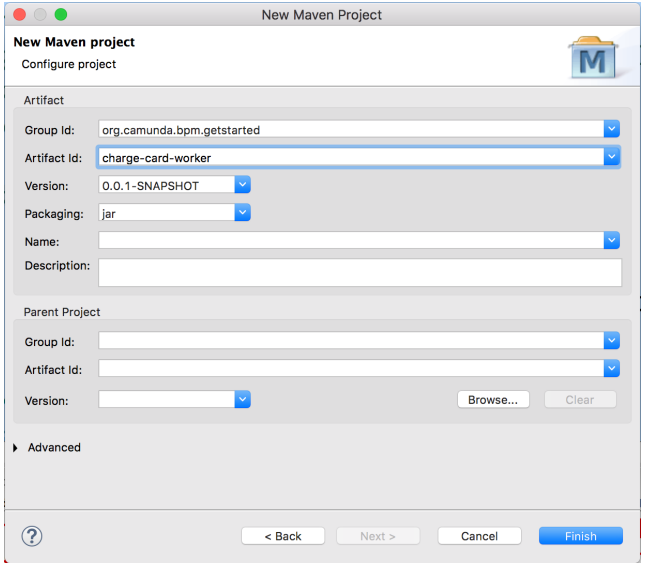
In addition to this section on Connectors, we recommend reviewing Connector secrets.

If you want to build **custom Connectors**, head over to our Connector SDK guide.

The development environment is based on

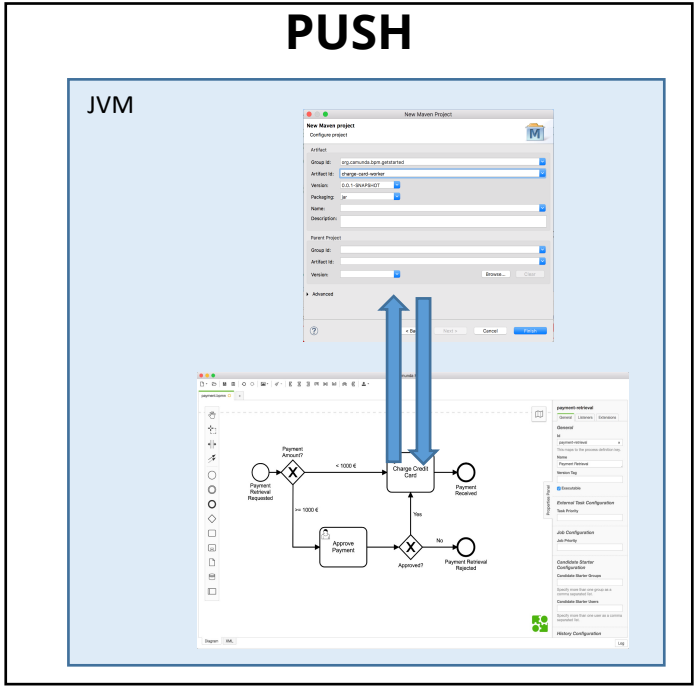


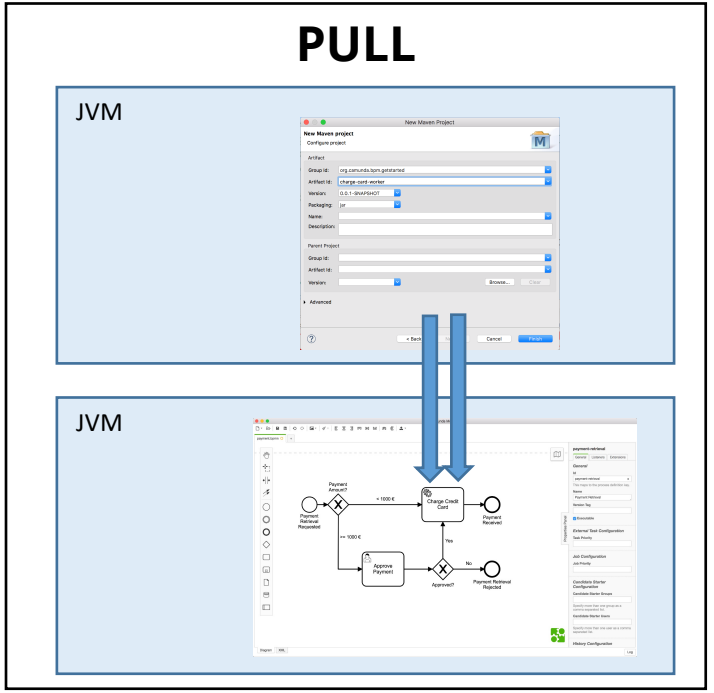
Camunda Modeler – this will be the option for Enterprise Integration course



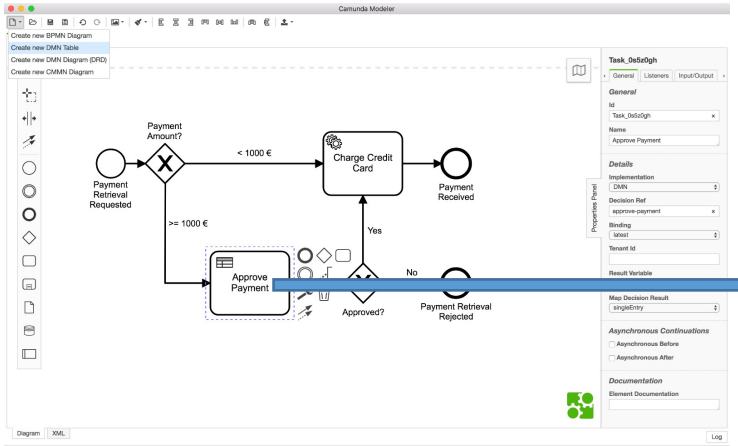
Eclipse with Maven for JAVA classes

Two architectural models for tasks' execution

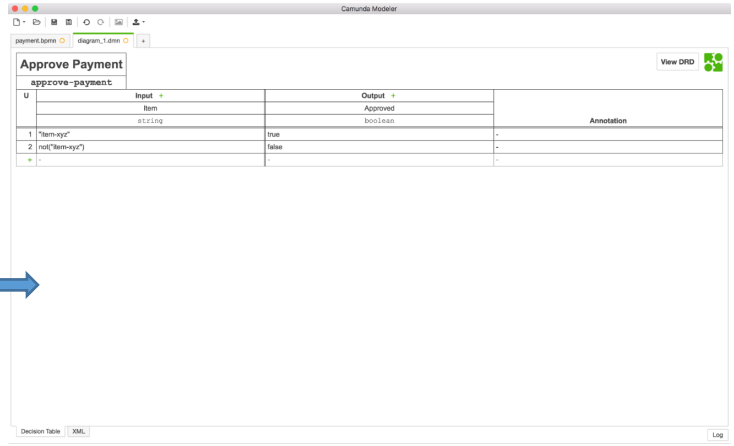




Business rules specification



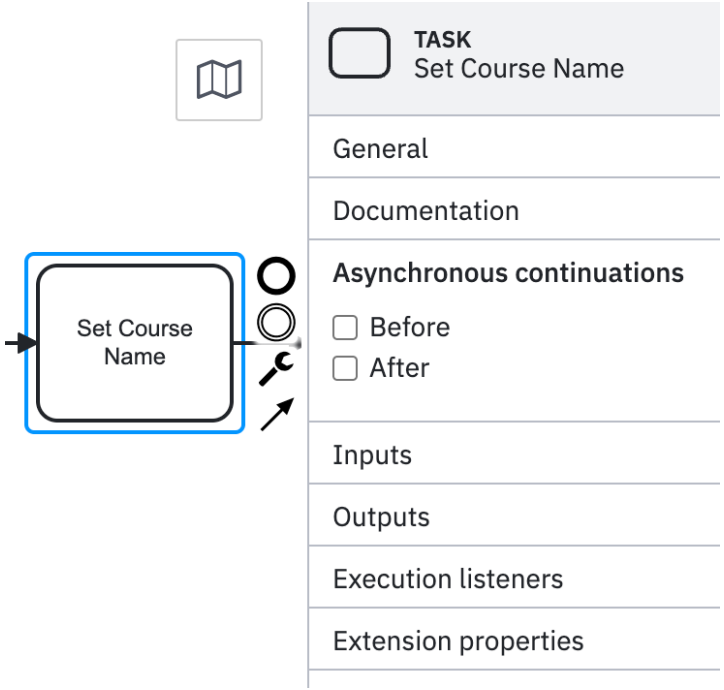
Camunda Modeler



DMN Table definition

Camunda implements the OMG's DMN specification, making it easy to model multiple Gateways into simpler-to-manage rule tables.

Asynchronous Continuations in Camunda BPM

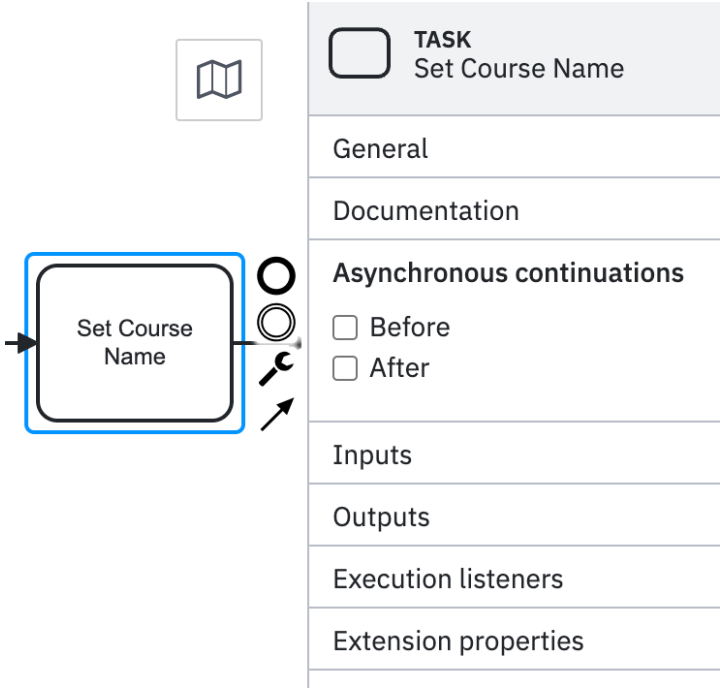


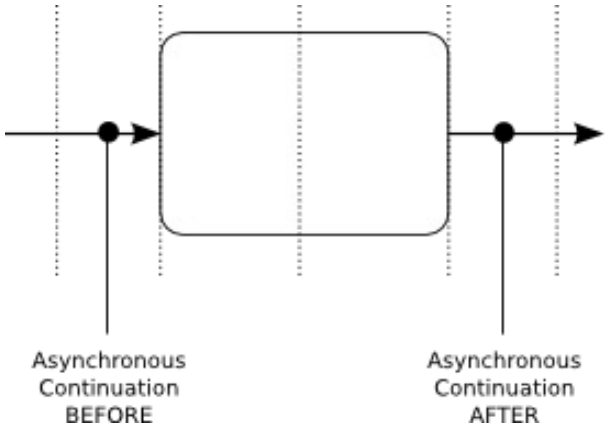
Asynchronous continuations are break-points in the process execution. They are used as transaction boundaries and allow another thread than the currently active thread to continue execution.

- Async is used for placing a safe-point before an activity such that the execution state is committed. If the activity then fails to execute, the transaction is rolled back only up to the safe point.
- Async also comes in handy if you have longer-running computations and do not want to block the calling thread (eg. HTTP Thread) but instead want to delegate the heavy lifting to a background thread.
- Finally, due to the fact that asynchronous continuations are executed by the job executor, the retry mechanism can be used in order to retry a failed activity execution.

https://camunda.com/blog/2014/07/advanced-asynchronous-continuations/

Asynchronous Continuations in Camunda BPM

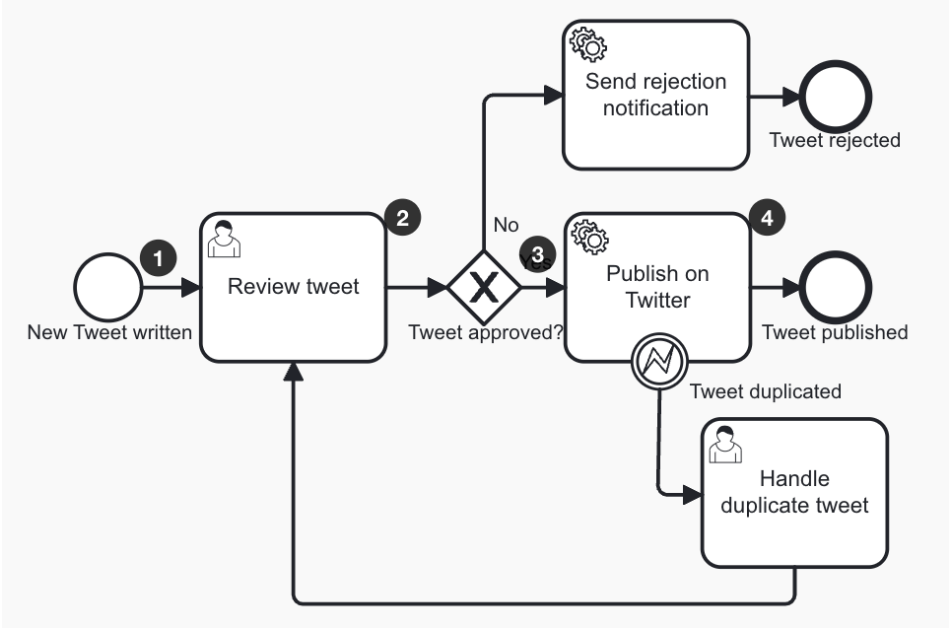




https://camunda.com/blog/2014/07/advanced-asynchronous-continuations/

Handling data in processes

When using Camunda, you have access to a dynamic map of process variables, which lets you associate data to every single process instance (and local scopes in case of user tasks or parallel flows).



1. The process instance starts with a freshly written `tweet` we need to remember.
2. We need to present this `tweet` so that the user can decide whether to `approve` it.
3. The gateway needs to have access to this information: was the tweet `approved`?
4. To publish the tweet, the service task again needs the `tweet` itself!

Therefore, the tweet approval process needs two variables:

Variable name	Variable type	Sample value
`tweet`	String	"@Camunda rocks"
`approved`	Boolean	true

Input and output variables

TÉCNICO LISBOA

- Input mappings can be used to create new variables. They can be defined on service tasks and subprocesses. When an input mapping is applied, it creates a new **local variable** in the scope where the mapping is defined.

Process instance variables	Input mappings	New variables
`orderId: "order-123"`	**source:** `=orderId` **target:** `reference`	`reference: "order-123"`
`customer:{"name": "John"}`	**source:** `=customer.name` **target:** `sender`	`sender: "John"`
`customer: "John"` `iban: "DE456"`	**source:** `=customer` **target:** `sender.name` **source:** `=iban` **target:** `sender.iban`	`sender: {"name": "John", "iban": "DE456"}`

- Output mappings can be used to customize how job/message variables are merged into the process instance. They can be defined on service tasks, receive tasks, message catch events, and subprocesses.

Job/message variables	Output mappings	Process instance variables
`status: "Ok"`	**source:** `=status` **target:** `paymentStatus`	`paymentStatus: "OK"`
`result: {"status": "Ok", "transactionId": "t-789"}`	**source:** `=result.status` **target:** `paymentStatus` **source:** `=result.transactionId` **target:** `transactionId`	`paymentStatus: "Ok"` `transactionId: "t-789"`
`status: "Ok"` `transactionId: "t-789"`	**source:** `=transactionId` **target:** `order.transactionId`	`order: {"transactionId": "t-789"}`

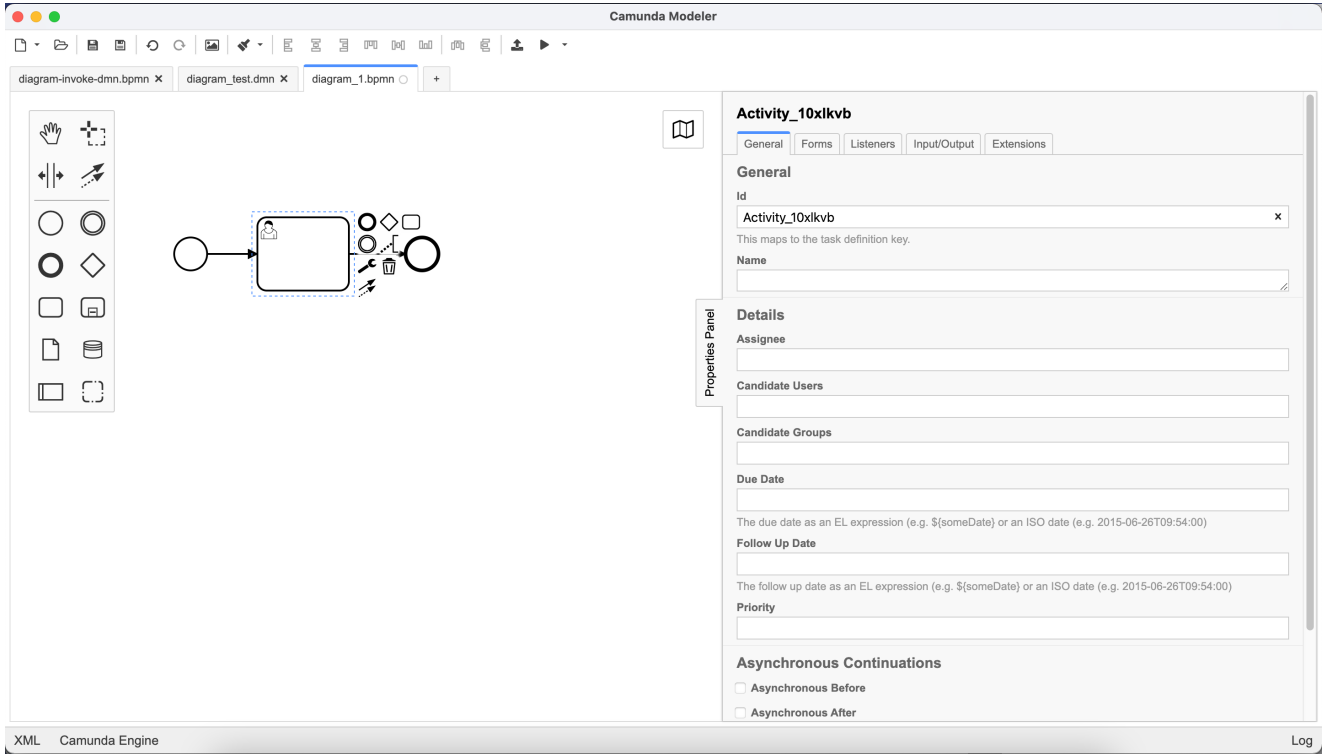
- Can be used to create new variables or customize how variables are merged into the process instance.

Source	Target
`customer.name`	`sender`
`customer.iban`	`iban`
`totalPrice`	`price`
`orderId`	`reference`

https://docs.camunda.io/docs/components/concepts/variables/#variable-values

TÉCNICO LISBOA

Camunda Authorization



- **Assignee**

a specific user who must perform the task

- **Candidate users**

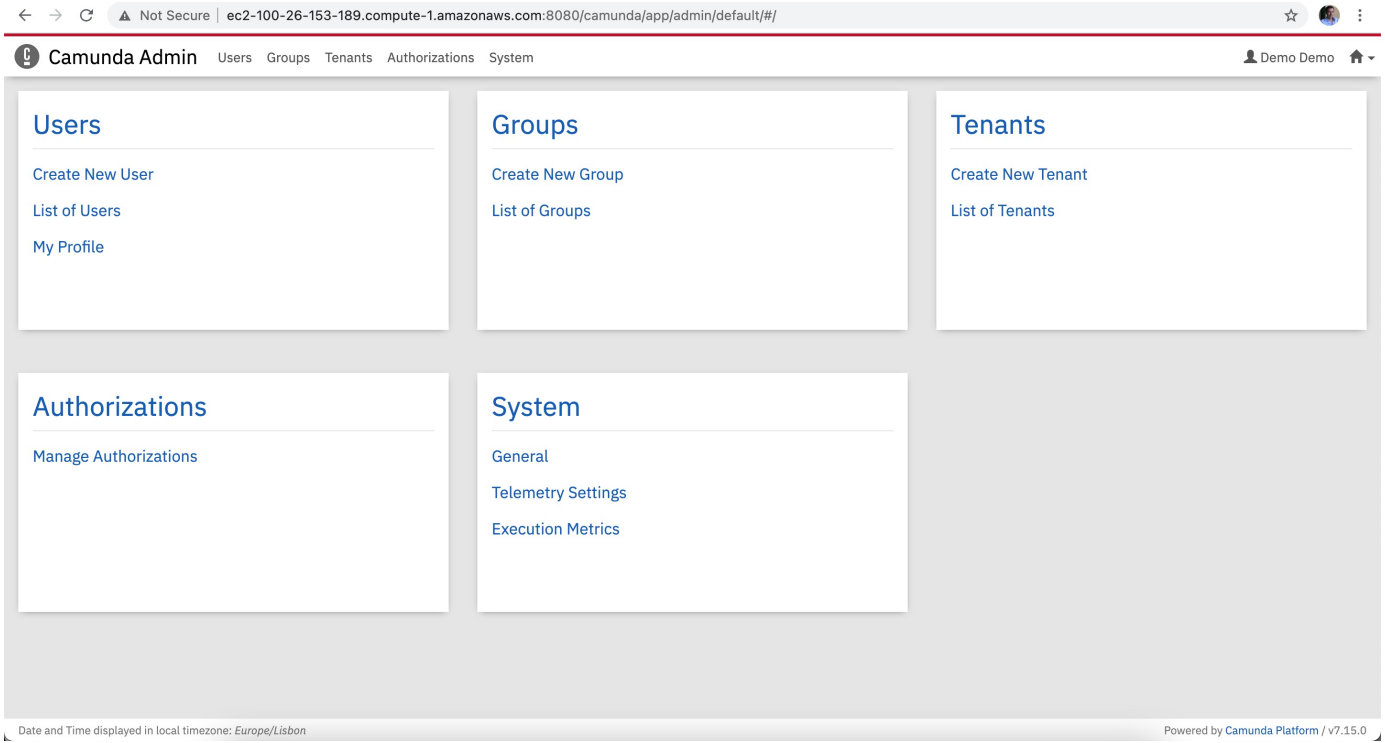
a list of specific users who can perform a task

- **Candidate groups**

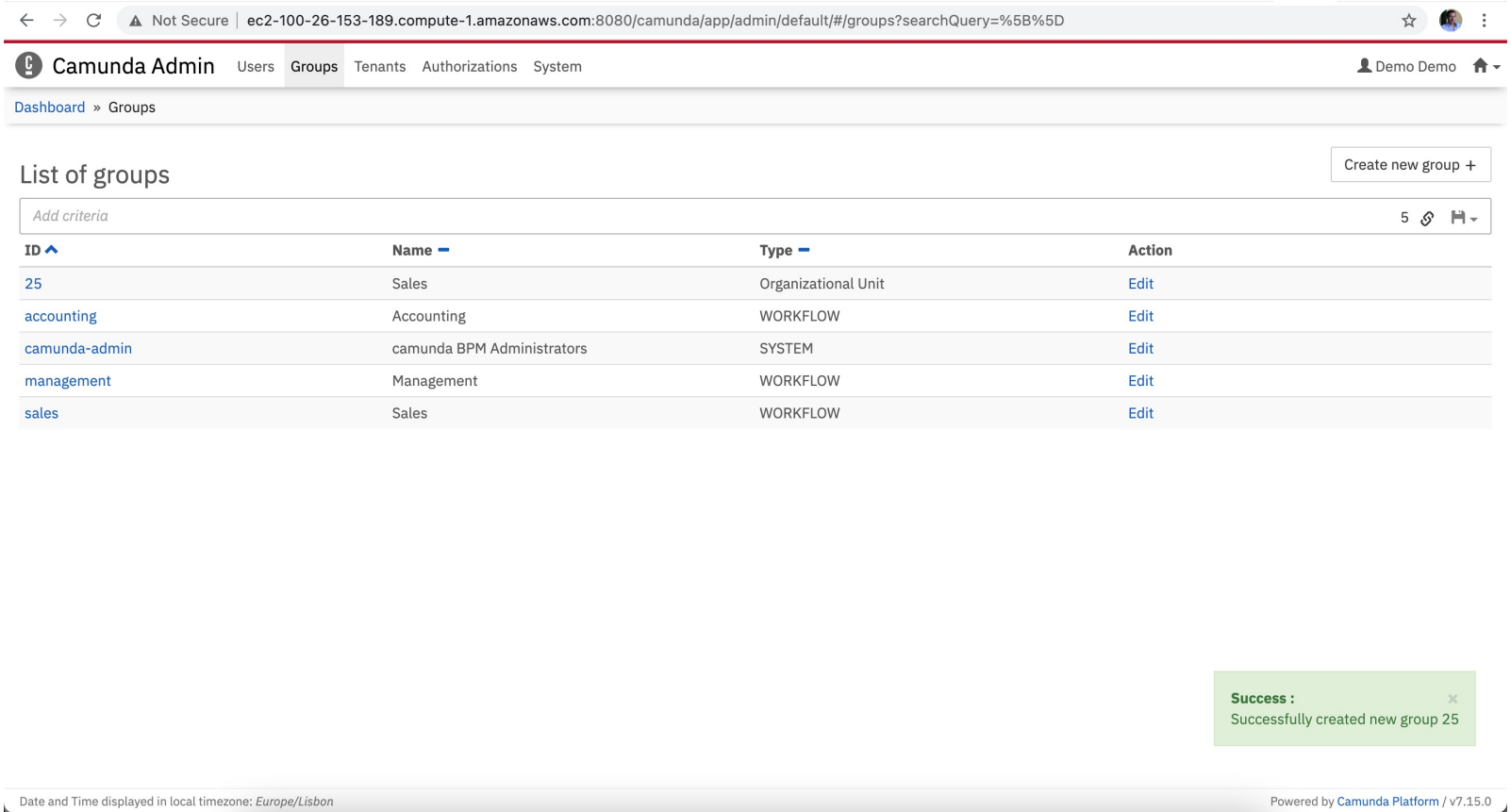
a list of user groups who can perform the task

Fine-grained authorizations definition

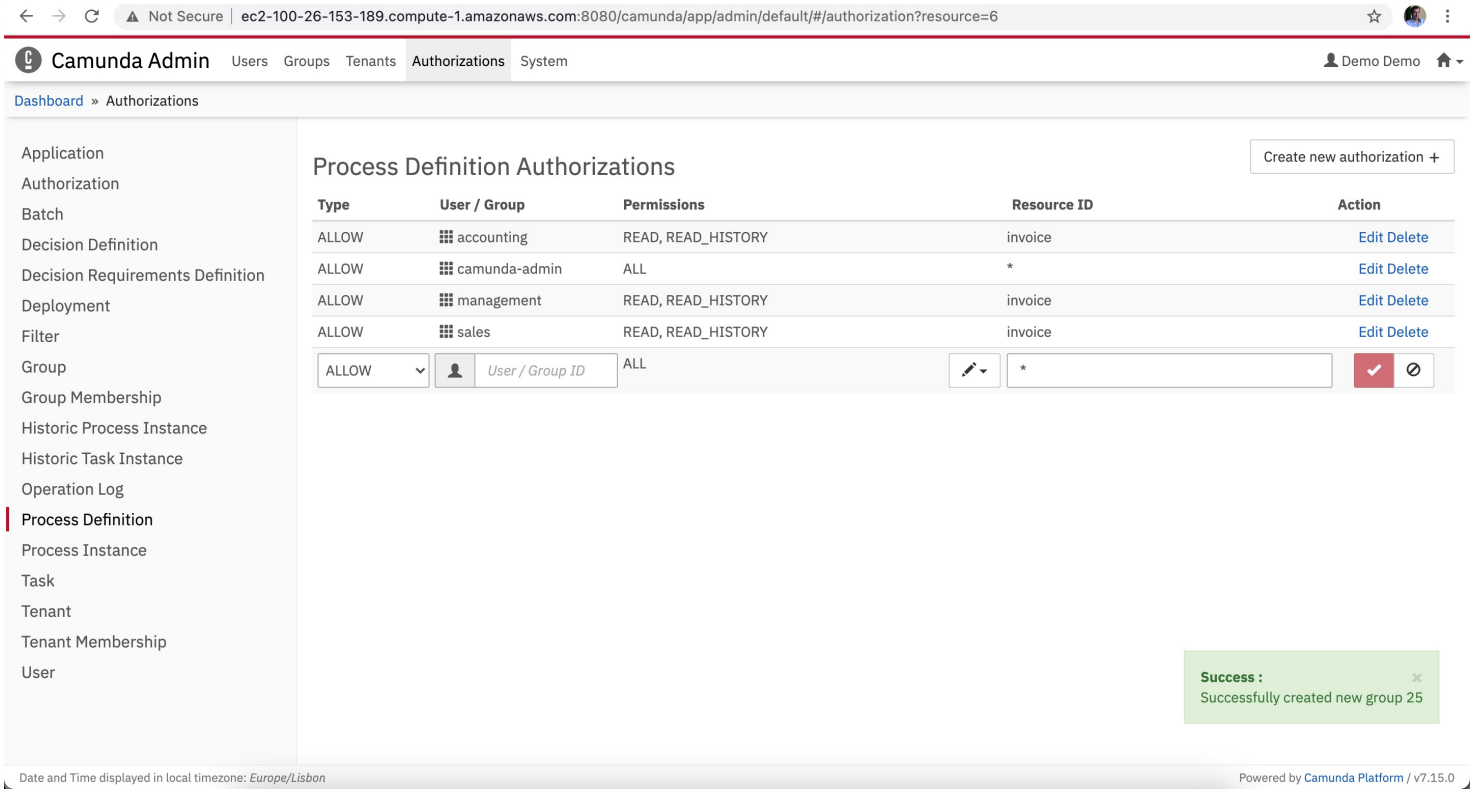
https://docs.camunda.org/manual/latest/user-guide/process-engine/authorization-service/
https://docs.camunda.org/optimize/2.2/technical-guide/authorization/



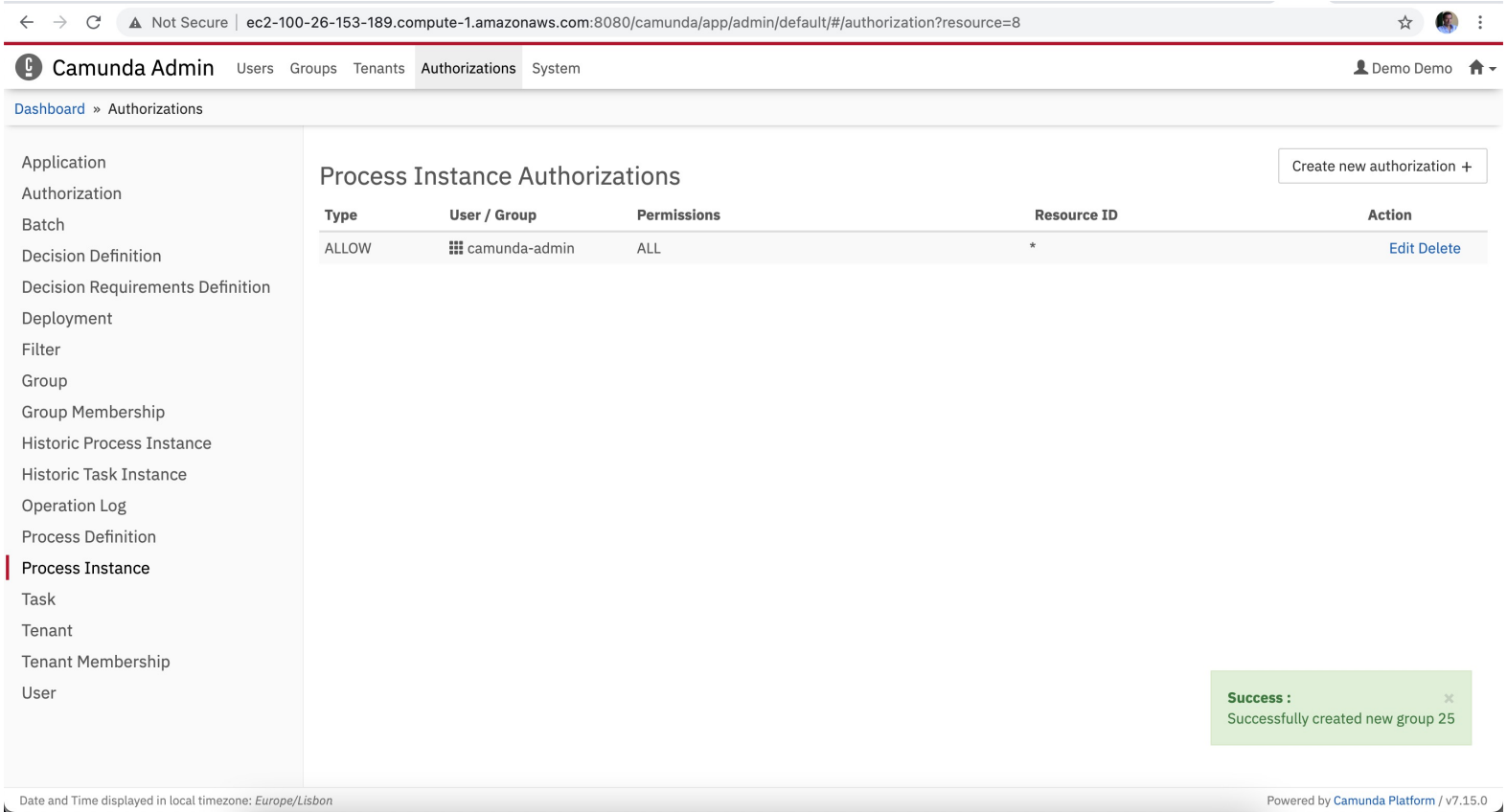
Creating a new group



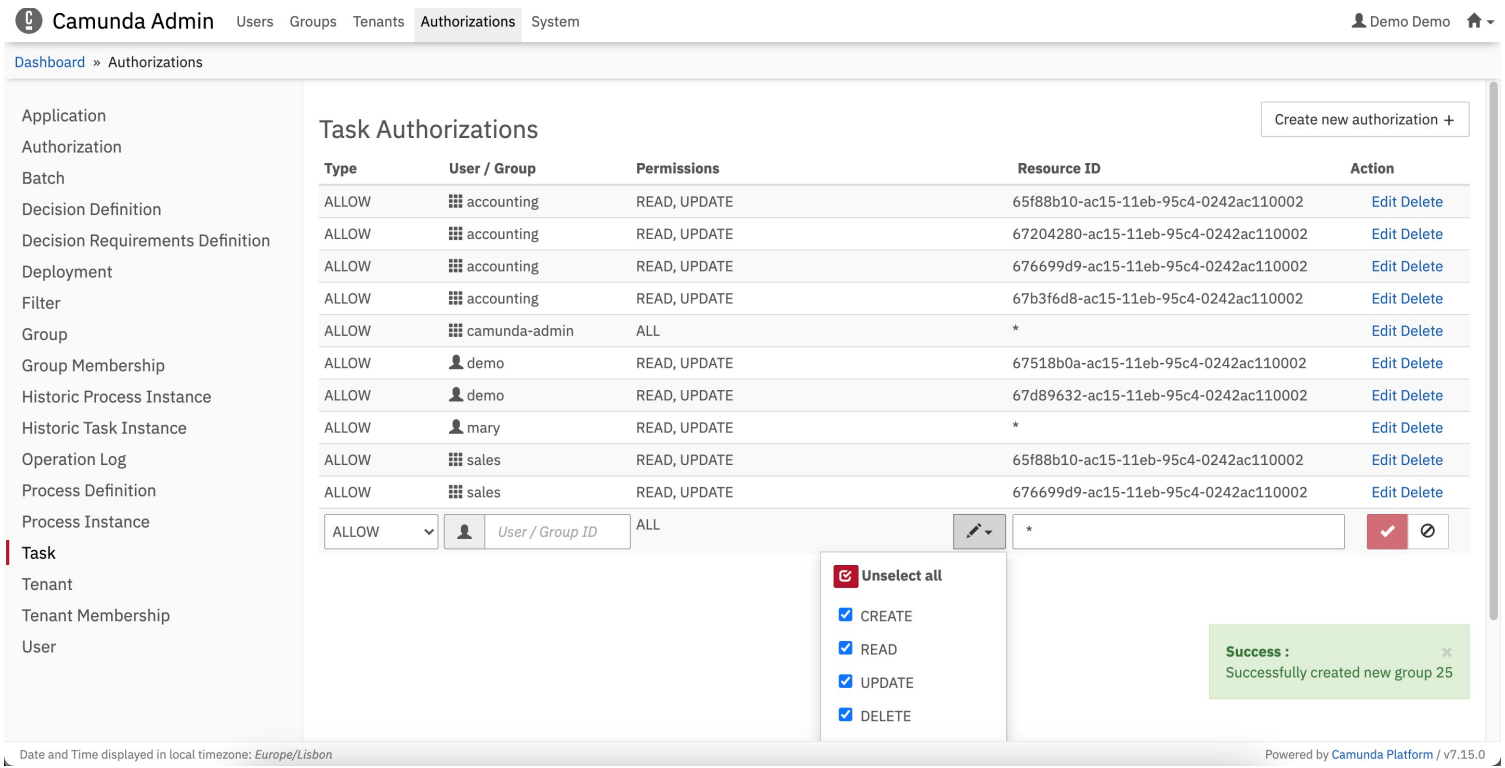
Authorizing a group to define a process



Authorizing a group to instantiate a process

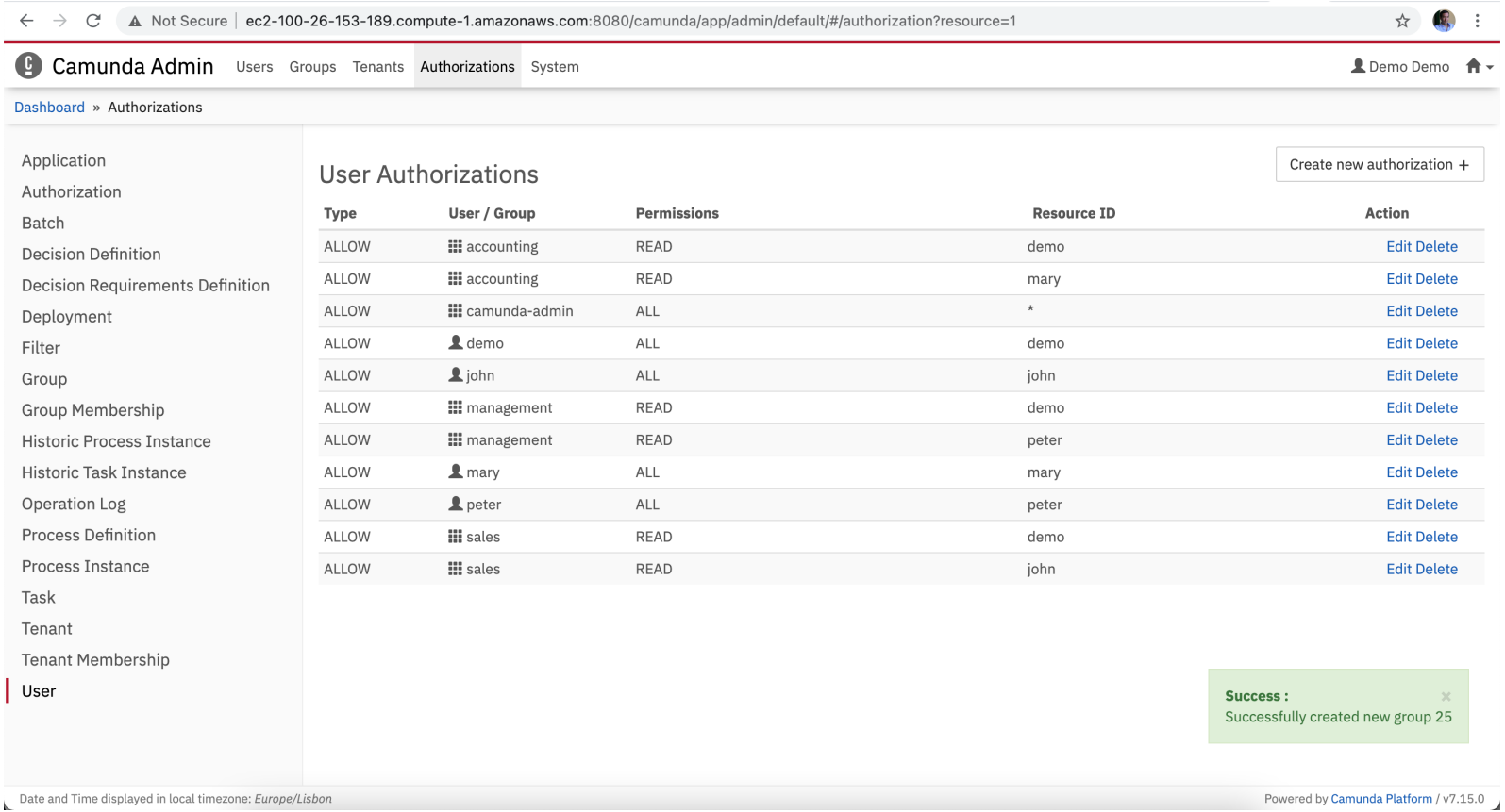


Authorizing a group to a specific task of a process



Assigning an user to a group

TÉCNICO LISBOA

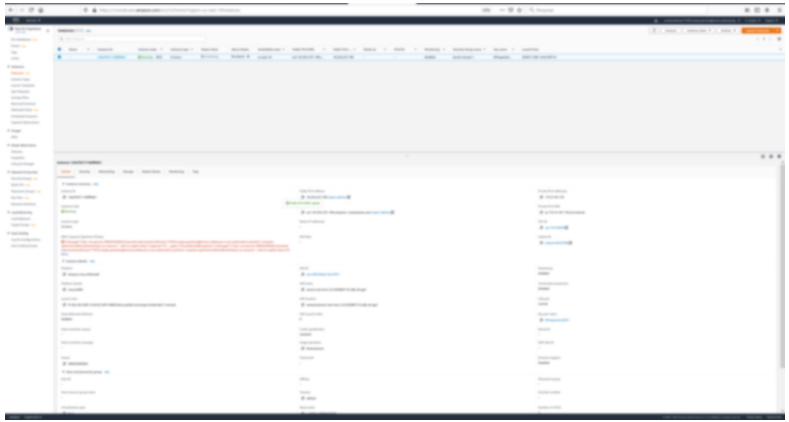


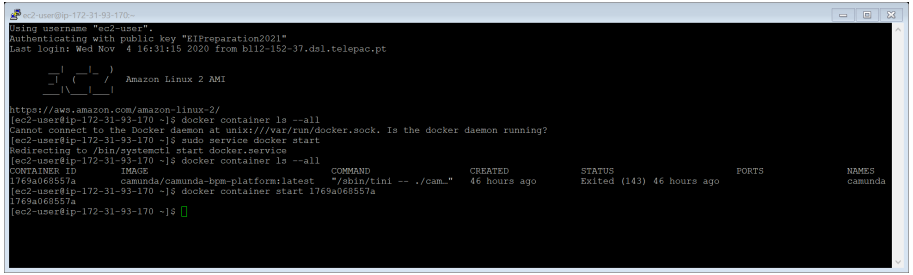
TÉCNICO LISBOA

Camunda Engine Deployment

TÉCNICO LISBOA

Camunda engine **deployment** on the cloud





C.1. Create an EC2 new instance of the type: Amazon Linux AMI 2018.03.0 (HVM), SSD Volume Type. The exact versions may change with time. And define the inbound rules to allow the 8080 and 22 accessed from anywhere.

C.2. Install the most recent version of camunda-modeler using the download public available at:

- https://camunda.com/download/modeler/

C.3. Access the new EC2 instance and execute the following commands:

```
sudo yum update -y
sudo yum install -y docker
sudo service docker start
sudo usermod -aG docker ec2-user
docker ps
exit
```

C.4. Then, access again your EC2 instance and install and execute the Camunda engine with the following command:

```
docker pull camunda/camunda-bpm-platform:latest
docker run -d --name camunda -p 8080:8080 camunda/camunda-bpm-platform:latest
```

Test the installed Camunda engine opening the following URL in your browser (the default user/password is demo/demo):

```
# open browser with url: http://<YOUR_AWS_EC2_NAME>:8080/camunda
```

Hint 1: Later if you would like to list all the available container use the following command:

```
PS C:\Users\Sérgio Guerreiro> docker container ls --all

CONTAINER ID        IMAGE                                   COMMAND                  CREATED             STATUS
PORTS               NAMES
44509250aa0c        camunda/camunda-bpm-platform:latest   "/sbin/tini -- ./cam…"   11 days ago         Exited
(143) 1 second ago                      camunda
```

Hint 2: Later if you would like to stop the container use the following command:

```
docker container stop <containerID_hash>
```

Hint 3: Later if you would like to restart the container use the following command:

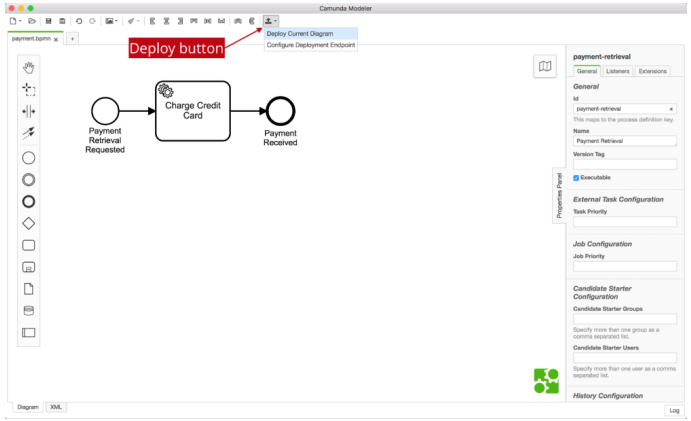
```
docker container start <containerID_hash>
```

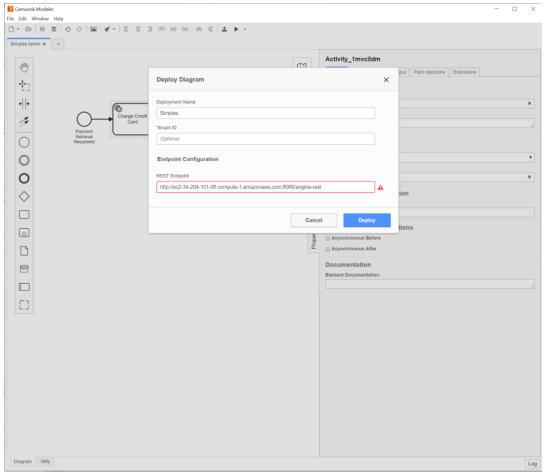
TÉCNICO LISBOA

Deployment on the cloud

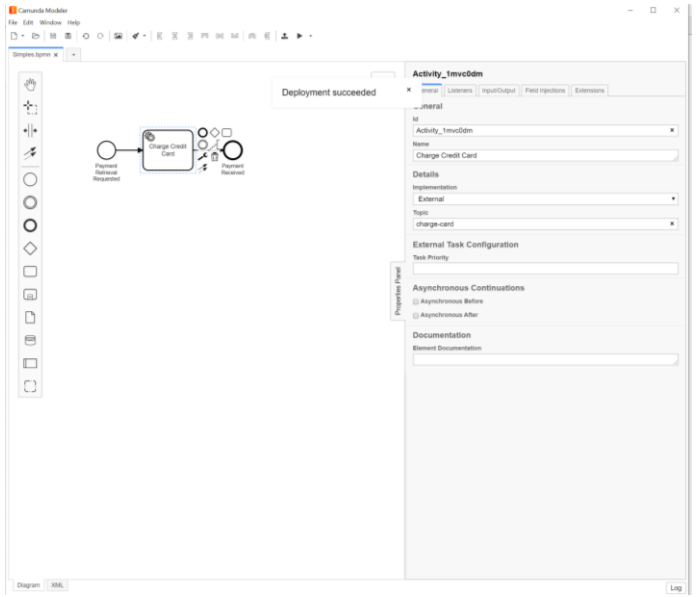
D.1. Use the Camunda Modeler to Deploy the Process

To deploy the Process, click on the deploy button in the Camunda Modeler, then give it the Deployment Name “Payment Retrieval” and click the Deploy button. From version 3.0.0 on, you will be required to provide an URL for an Endpoint Configuration along with Deployment Details. This can be either the root endpoint to the REST API (e.g. `http://localhost:8080/engine-rest`) or an exact endpoint to the deployment creation method (e.g. `http://localhost:8080/engine-rest/deployment/create`).





You should see a success message in the Camunda Modeler:

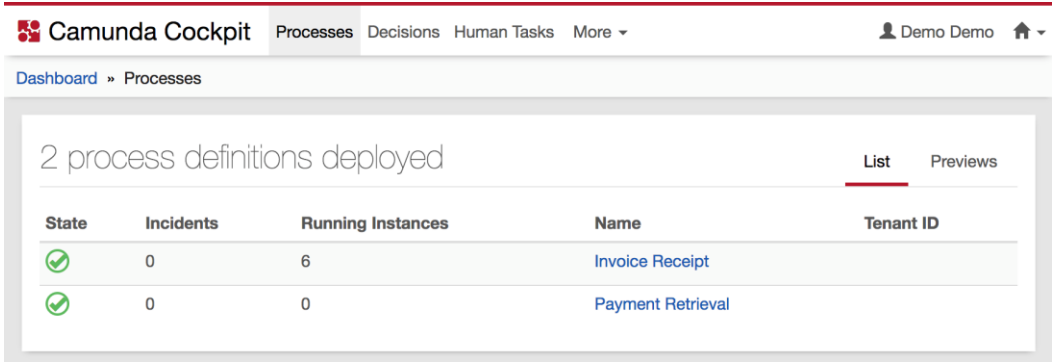


TÉCNICO LISBOA

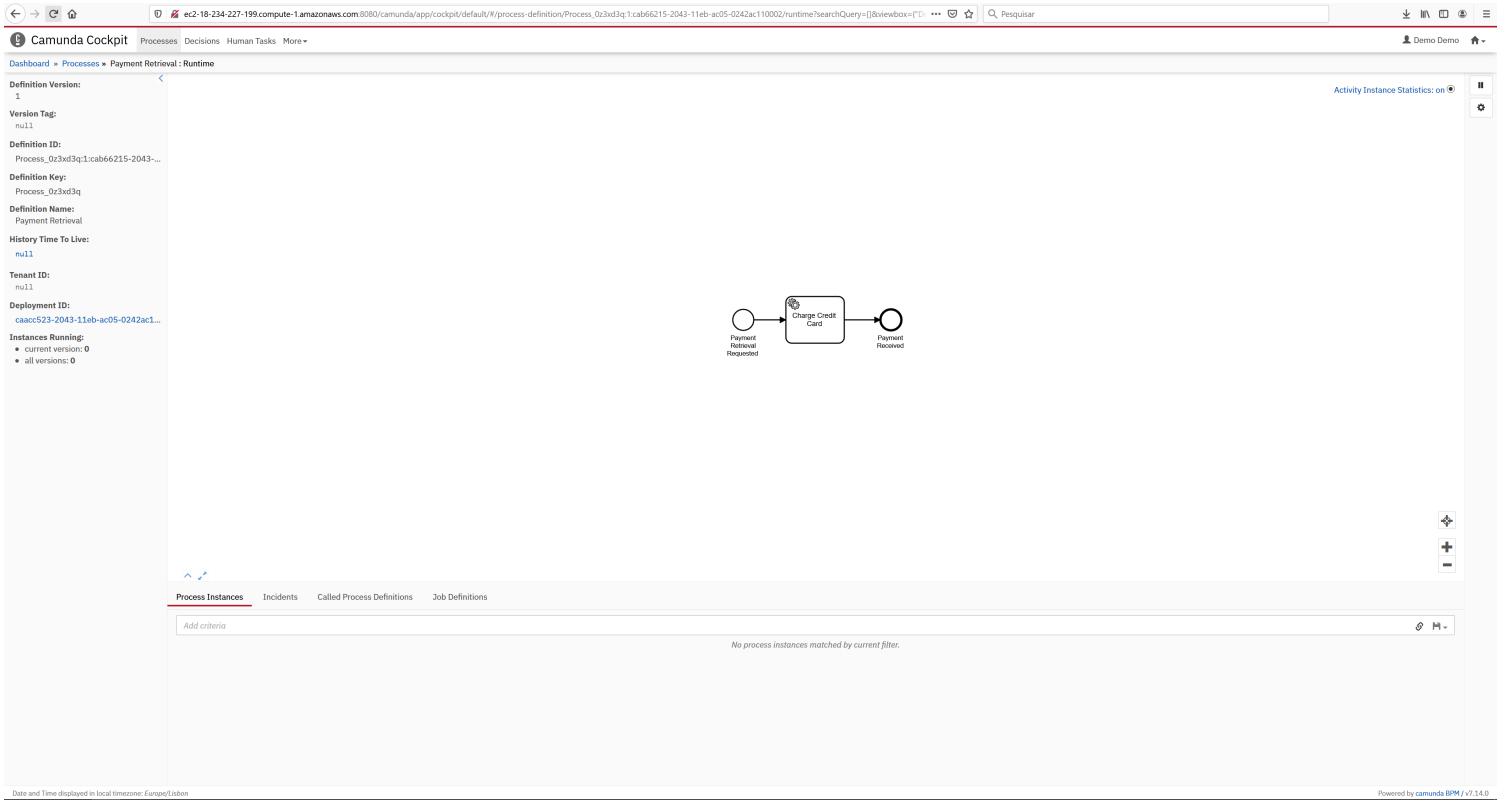
Deployment on the cloud

D.1. Verify the Deployment with Cockpit

Next, use Cockpit to see if the process was successfully deployed. Go to http://localhost:8080/camunda/app/cockpit and log in with the credentials demo / demo. Your process *Payment Retrieval* should be visible on the dashboard.



Deployment on the cloud

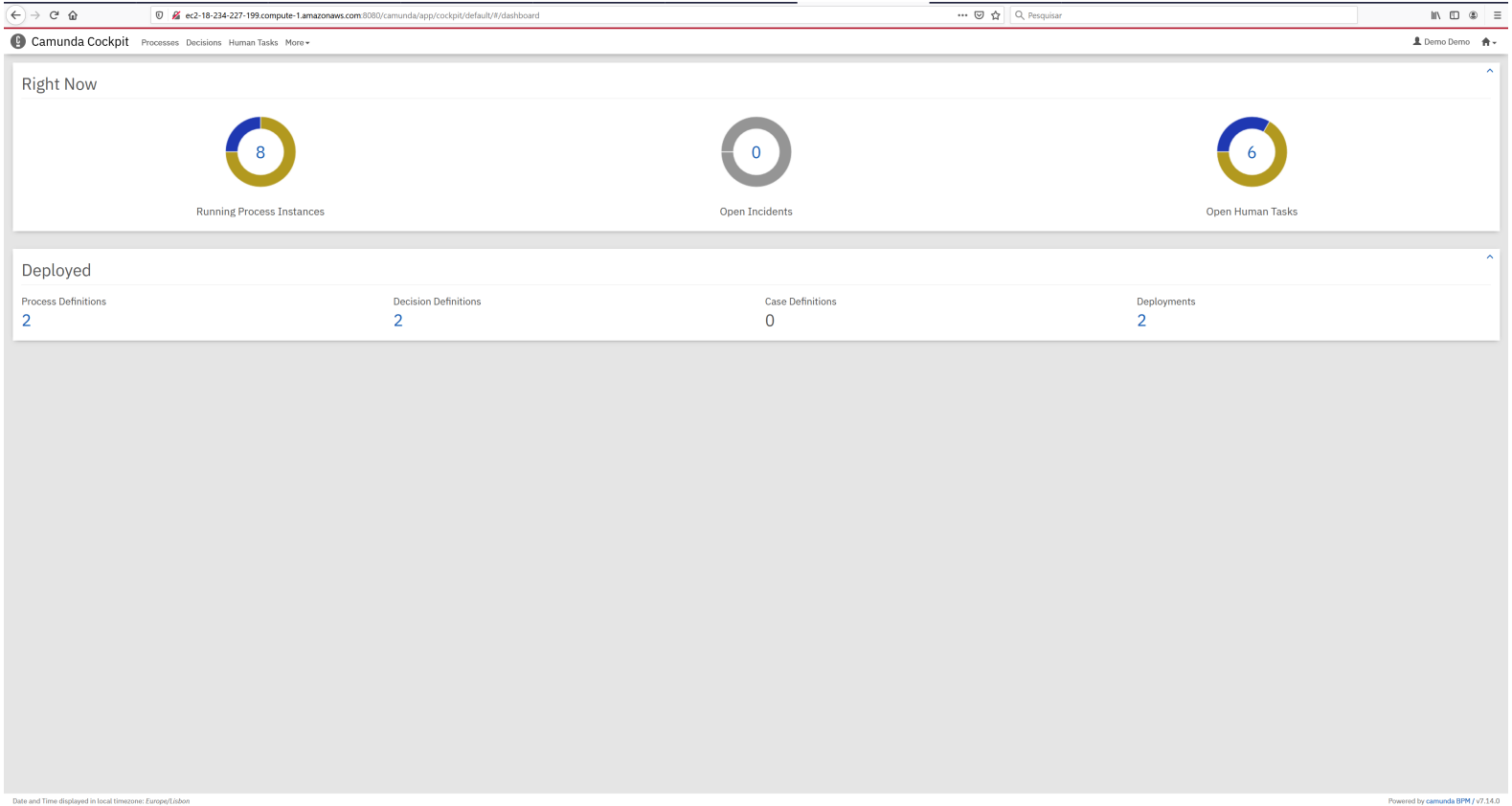


TÉCNICO LISBOA

Camunda Cockpit

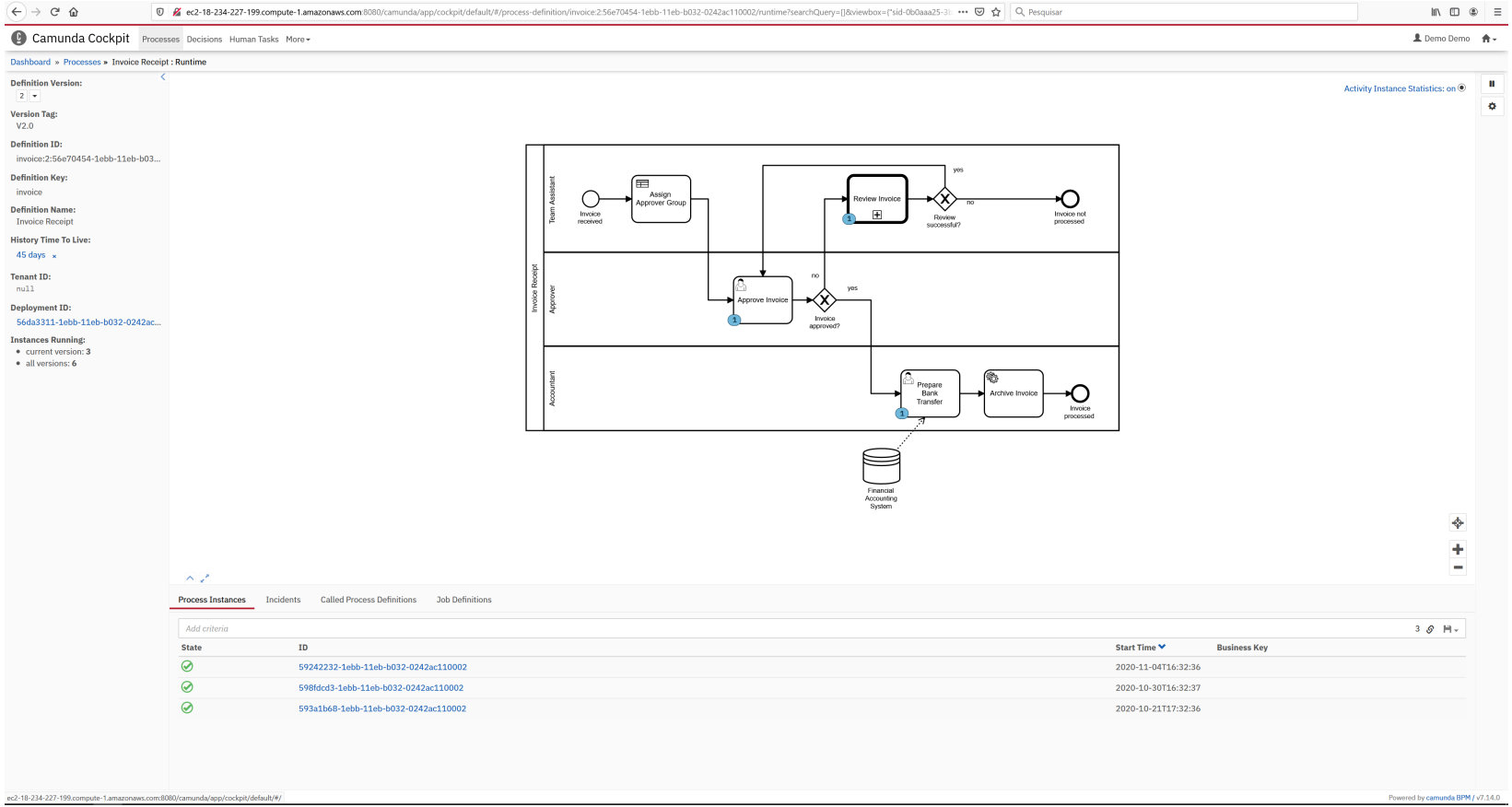
TÉCNICO LISBOA

Camunda **Cockpit**



TÉCNICO LISBOA

Camunda **Cockpit**



TÉCNICO LISBOA

Instantiating one process model

TÉCNICO LISBOA

Instantiating one process model

D.1. Start a process invocation of your BPMN using curl:

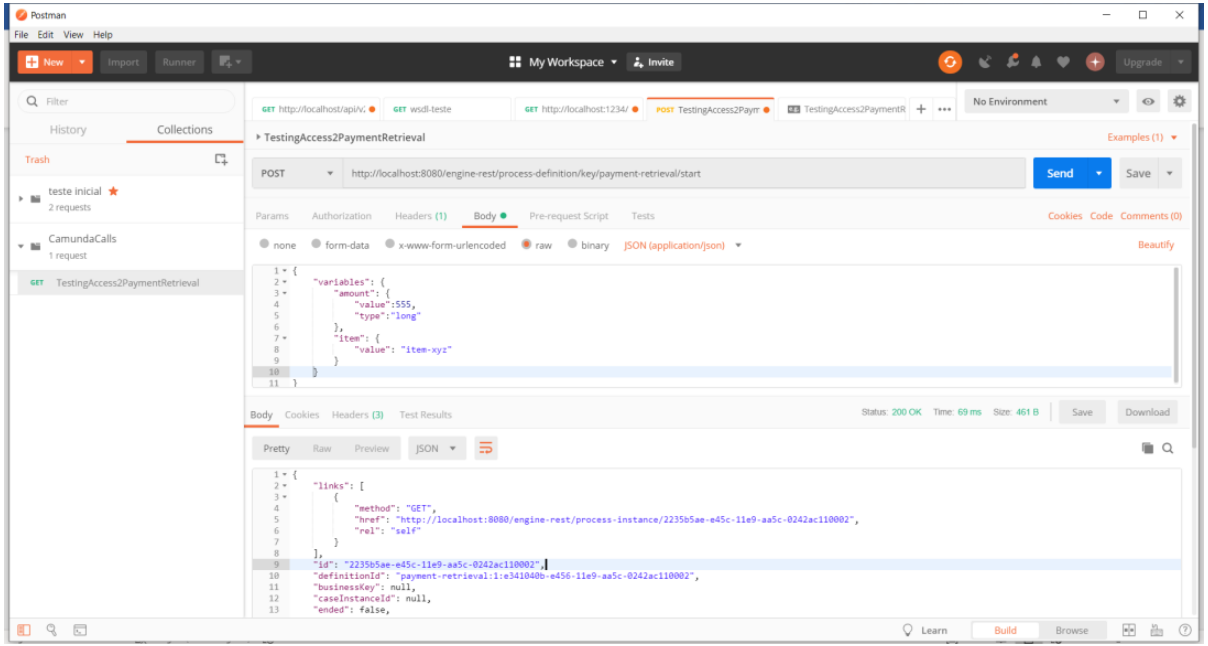
```
curl -H "Content-Type: application/json" -X POST -d '{"variables": {"amount": {"value":555,"type":"long"},
"item": {"value":"item-xyz"} } }' http://localhost:8080/engine-rest/process-definition/key/payment-
retrieval/start
```

Running example

```
Sérgio Guerreiro@LAPTOP-DF942S8T ~
$ curl -H "Content-Type: application/json" -X POST -d '{"variables": {"amount": {"value":555,"type":"long"},
"item": {"value":"item-xyz"} } }' http://localhost:8080/engine-rest/process-definition/key/payment-
retrieval/start
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   431    0   345  100    86    345     86  0:00:01 --:--:--  0:00:01  5525
{"links":[{"method":"GET","href":"http://localhost:8080/engine-rest/process-instance/a5c230ce-e459-11e9-aa5c-
0242ac110002","rel":"self"}],"id":"a5c230ce-e459-11e9-aa5c-0242ac110002","definitionId":"payment-
retrieval:1:e341040b-e456-11e9-aa5c-
0242ac110002","businessKey":null,"caseInstanceId":null,"ended":false,"suspended":false,"tenantId":null}
```

Instantiating one process model

Alternatively, install postman https://www.getpostman.com/downloads/ (for windows OS and Mac OS). Here is what the request might look like in Postman:



Note1: due to the asynchronous JAVA service, the instance could be listed as an active instance for a while in the camunda cockpit.

Note2: the external java client is not accessible remotely. It uses the org.camunda.bpm.client.ExternalTaskClient class which takes care of the invocation directly.

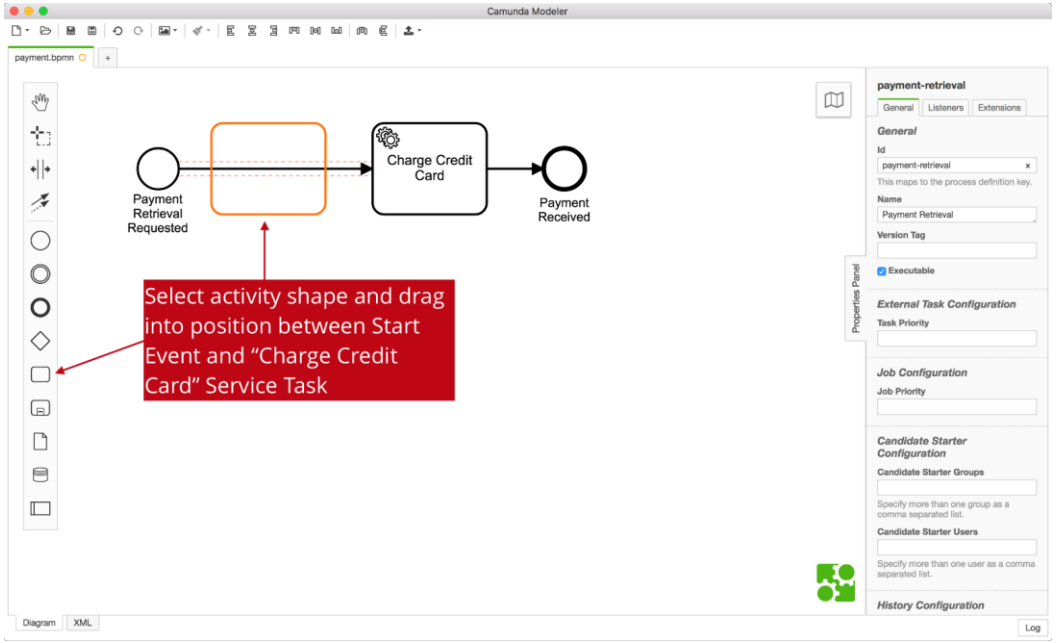
TÉCNICO LISBOA

Camunda Integrated Forms

A user interaction task

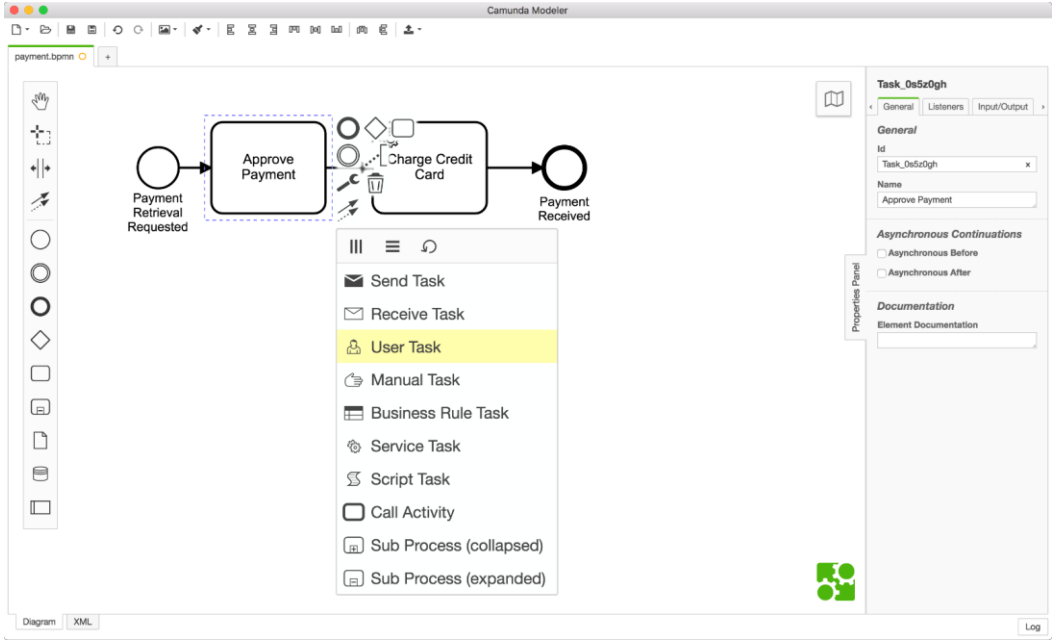
E.1. Add a User Task

We want to modify our process so that we can involve humans. To do so, open the process in the Camunda Modeler. Next, from the Modeler's left-hand menu, select the activity shape (rectangle) and drag it into position between the Start Event and the "Charge Credit Card" Service Task. Name it *Approve Payment*.



A user interaction task

Change the activity type to *User Task* by clicking on it and using the wrench button menu.

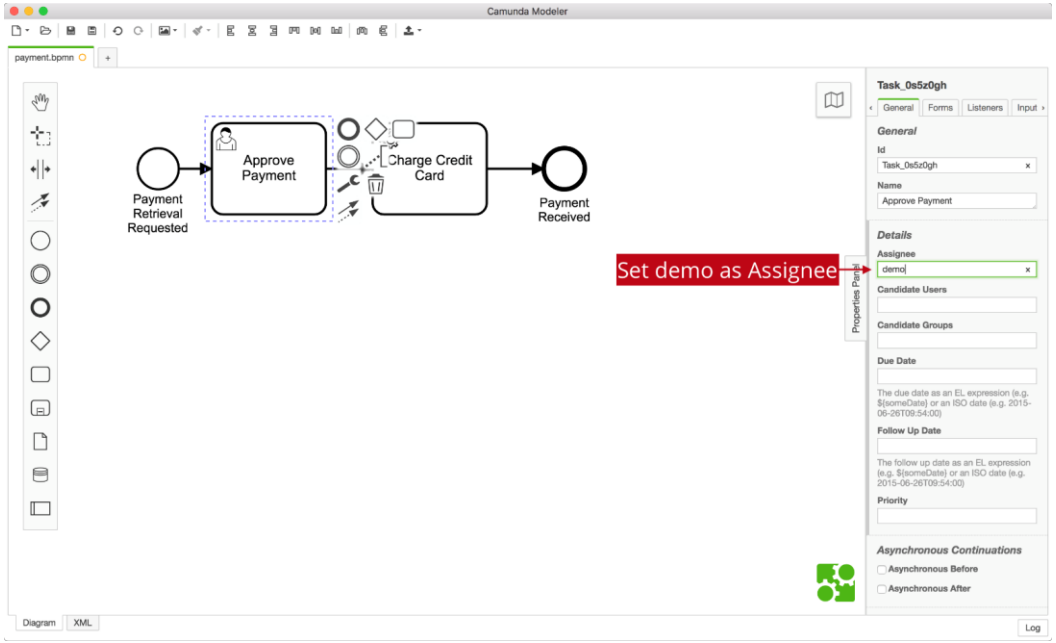


A user interaction task

E.1. Configure a User Task

Next, open the properties view. If the properties view is not already visible, click on the “Properties Panel” label on the right-hand side of the Modeler canvas.

Select the User Task on the canvas. This will update the selection in the properties view. Scroll to the property named `Assignee`. Type *demo*.



A user interaction task

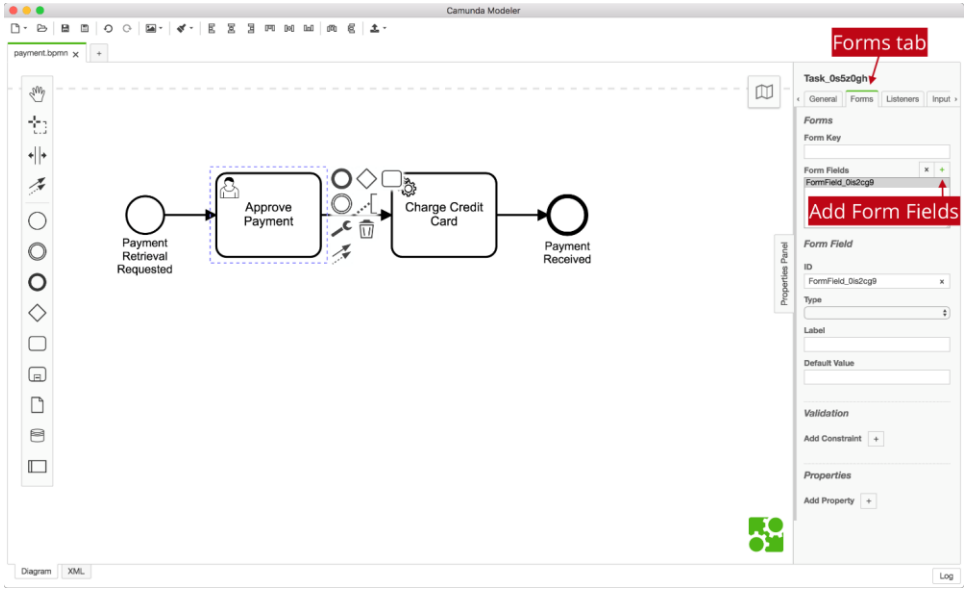
E.1. Configure a basic form in the User Task

This step will also take place in the properties panel. If the panel is not already visible, click on the “Properties Panel” label on the right-hand side of the Modeler canvas.

Select the User Task on the canvas. This will update the selection in the properties view.

Click on the Tab `Forms` in the properties panel.

Add three form fields by clicking on the `plus` button:



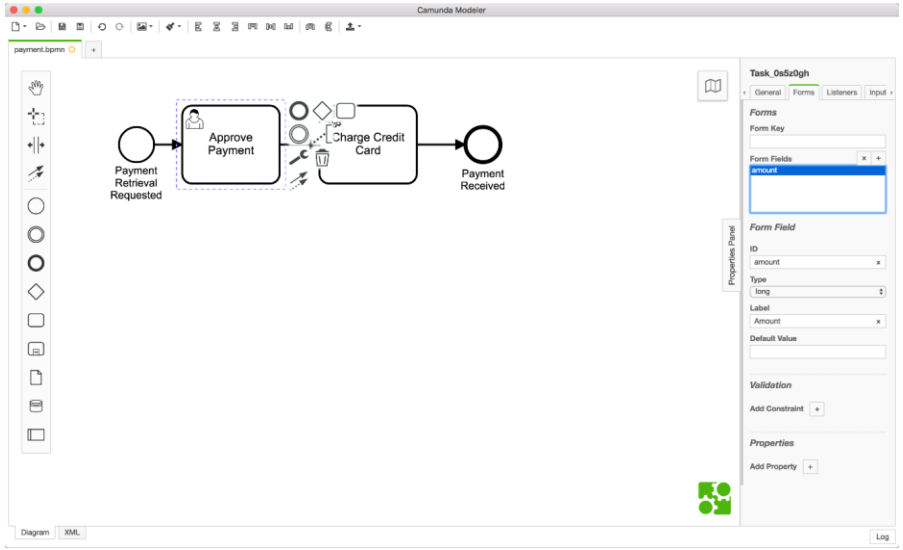
A user interaction task

Field 1:

ID: amount

Type: long

Label: Amount



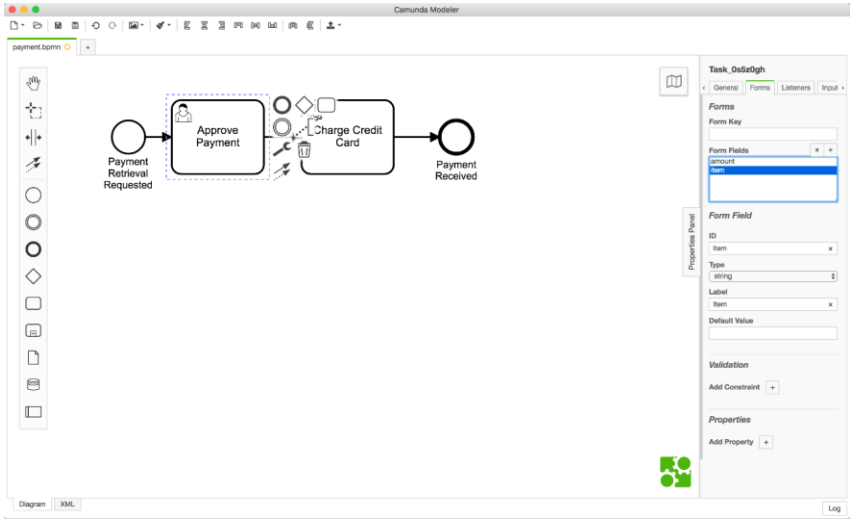
A user interaction task

Field 2:

ID: item

Type: string

Label: Item



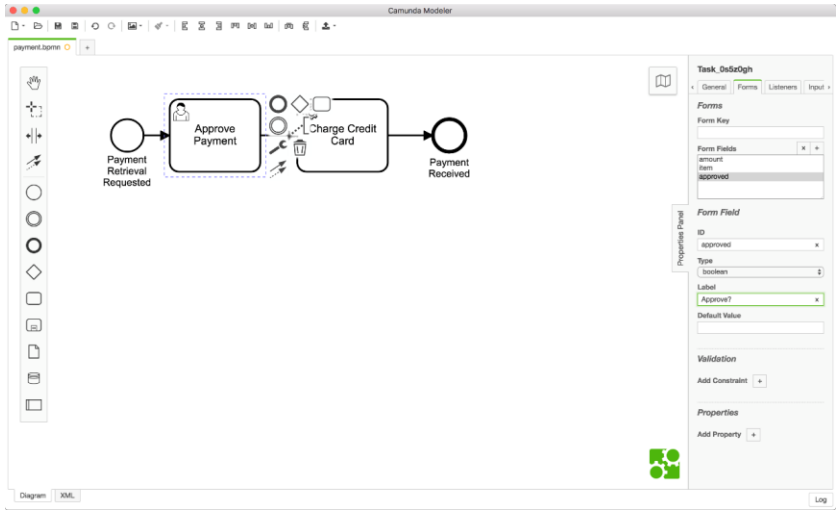
A user interaction task

Field 3:

ID: approved

Type: boolean

Label: Approved?



A user interaction task

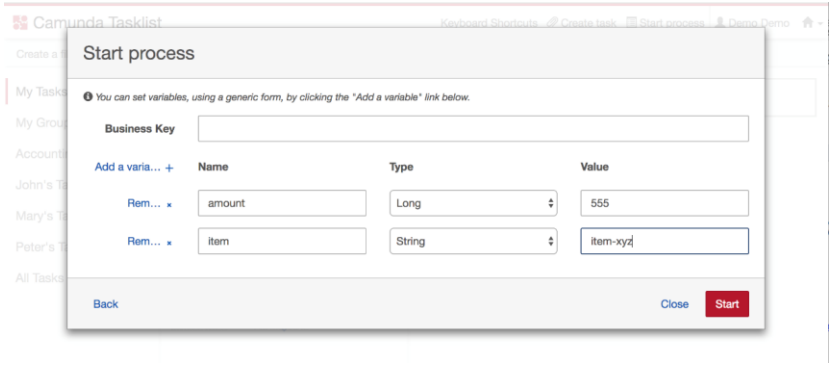
E.1. Deploy the Process

Use the `Deploy` Button in the Camunda Modeler to deploy the updated process to Camunda.

E.2. Work on the Task

Go to Tasklist (http://localhost:8080/camunda/app/tasklist) and log in with the credentials “demo / demo”. Click on the

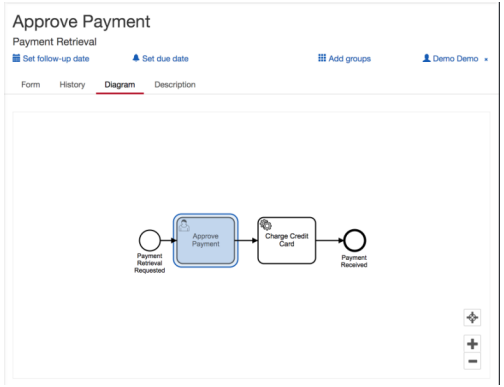
button to start a process instance. This opens a dialog where you can select *Payment Retrieval* from the list. Now you can set variables for the process instance using a generic form.



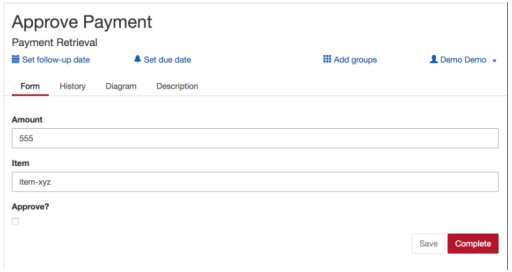
The generic form can be used whenever you have not added a dedicated form for a User Task or a Start Event. Click on the *Add a variable* button to create a new row. Fill in the form as shown in the screenshot. When you’re done, click *Start*.

A user interaction task

You should now see the *Approve Payment* task in your Tasklist. Select the task and click on the *Diagram* tab. This displays the process diagram highlighting the User Task that's waiting to be worked on.



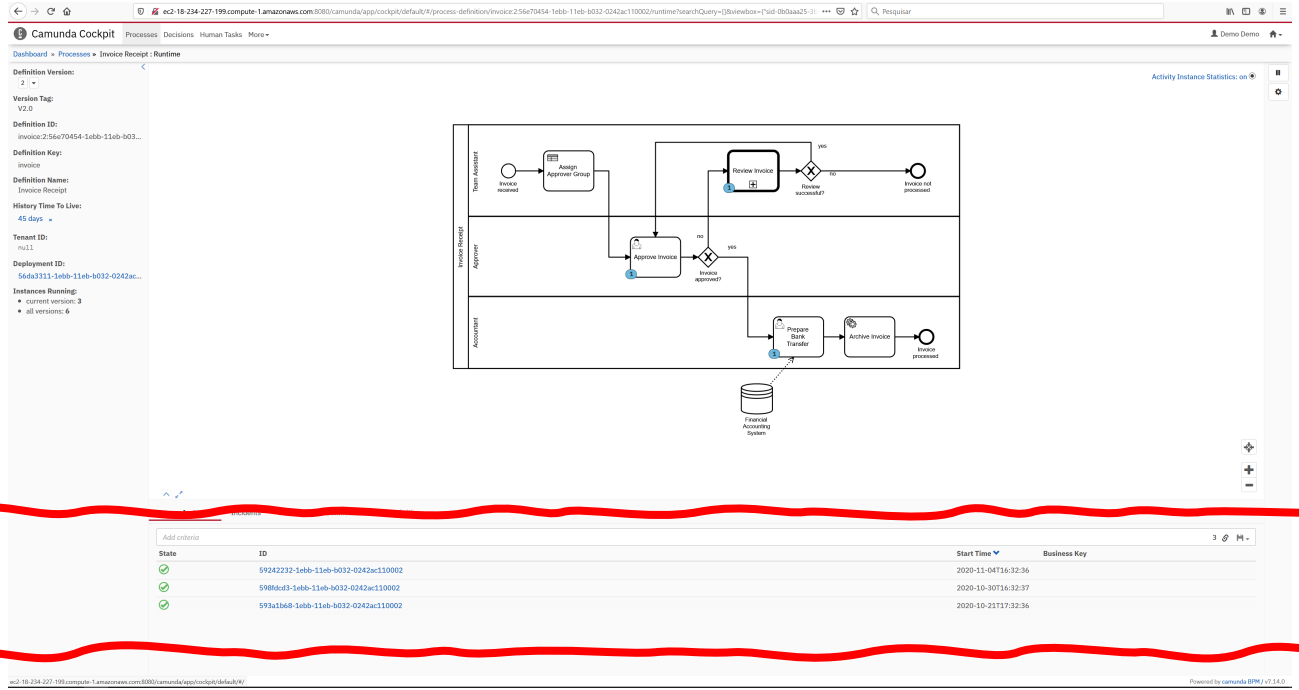
To work on the task, select the *Form* tab. Because we defined the variables in the Form Tab in the Camunda Modeler, the Tasklist has automatically generated form fields for us.



TÉCNICO LISBOA

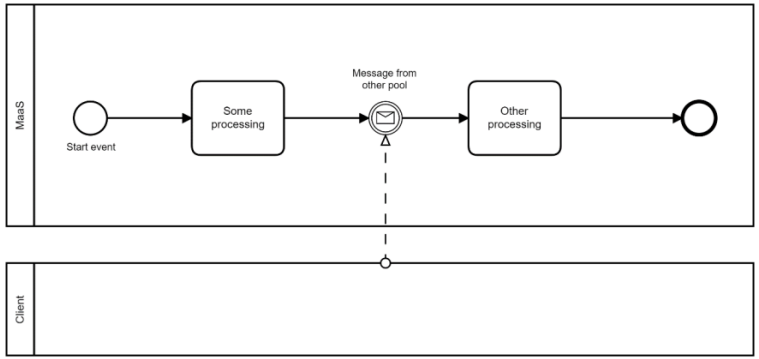
Multiple instances of a process

Multiple instances of the same process

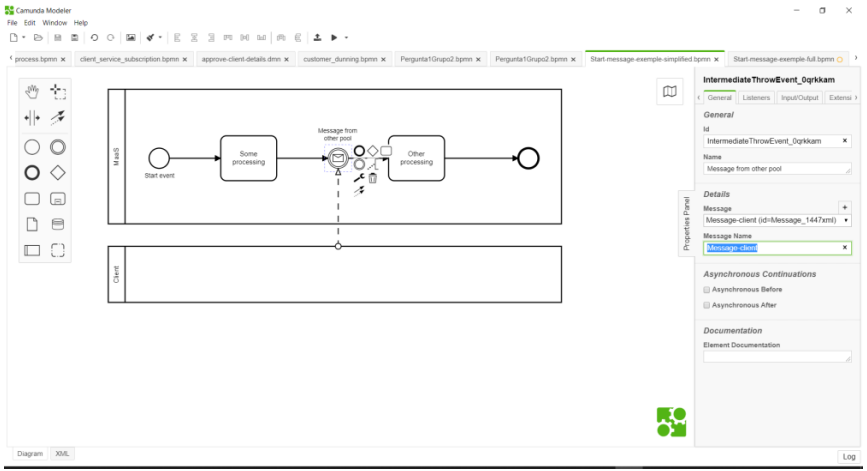


Multiple instances - 5.2. Correlation ID of a process to deal with messages

J1. Design the following collaboration in Camunda Modeller:



Where the intermediate message catch event should have a configuration similar with the following:



Multiple instances - 5.2. Correlation ID of a process to deal with messages

J1. Deploy the model in your Camunda engine.

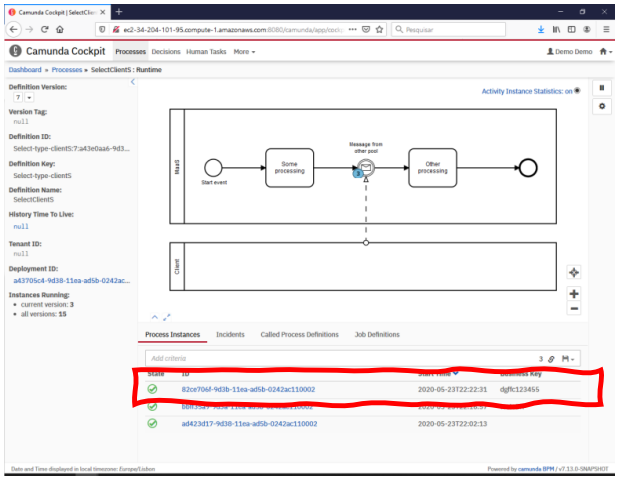
J2. Start the process with a command similar with the following:

```
curl -H "Content-Type: application/json" --data "@bodyStart.json" -X POST http://ec2-34-204-101-95.compute-1.amazonaws.com:8080/engine-rest/process-definition/key/Select-type-clientS/start
```

where bodyStart.json defines a key for that new instance of the collaboration:

```
{
  "businessKey" : "dgffc123455"
}
```

You can verify the creation of the instance in the camunda cockpit. You will obtain something similar with the following business key as defined:



Multiple instances - 5.2. Correlation ID of a process to deal with messages

J1. Then, the instance is expecting a message. To test the catch of a REST message, use a similar curl command:

```
curl -H "Content-Type: application/json" --data "@body.json" -X POST http://ec2-34-204-101-95.compute-1.amazonaws.com:8080/engine-rest/message
```

where bodyStart.json defines the message listener and the correlation with the key for your instance:

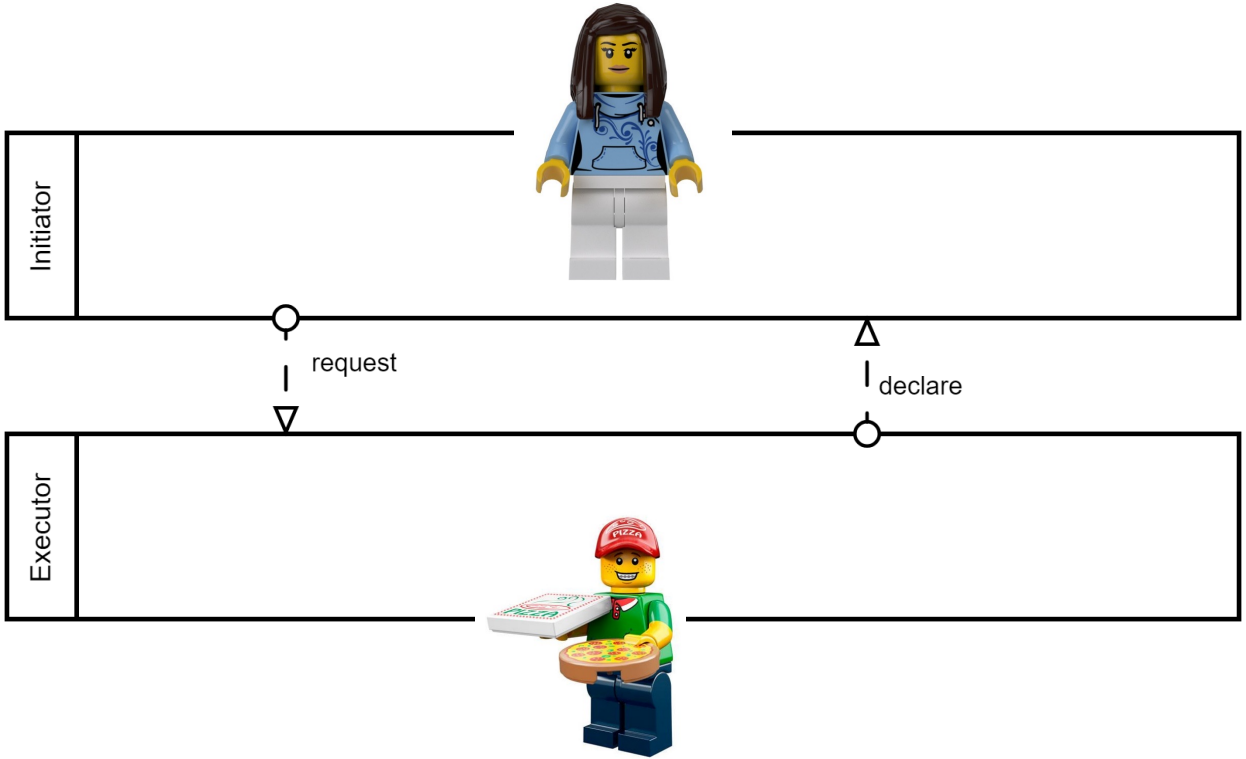
```
{
  "messageName" : "Message-client",
  "businessKey" : "dgffc123455",
  "resultEnabled" : true
}
```

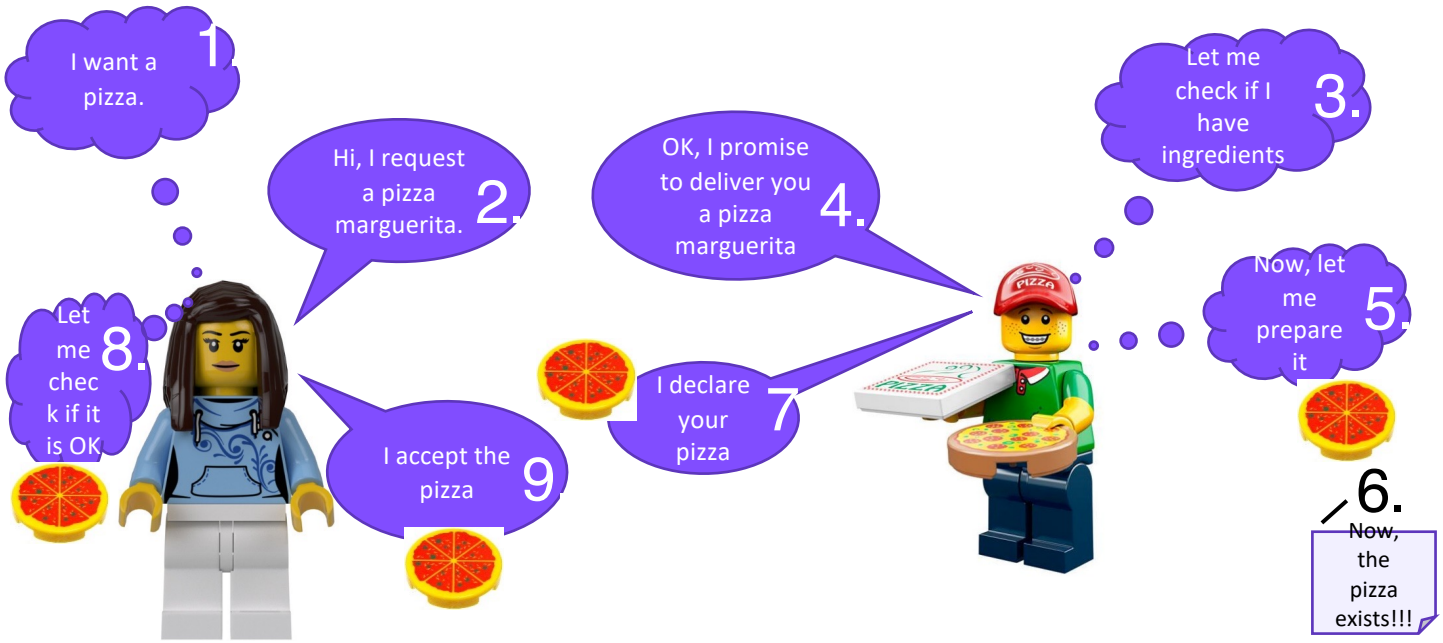
TÉCNICO LISBOA

Process discovery and Enrichment

A BPMN pattern oriented solution

Mapping DEMO complete pattern onto BPMN models

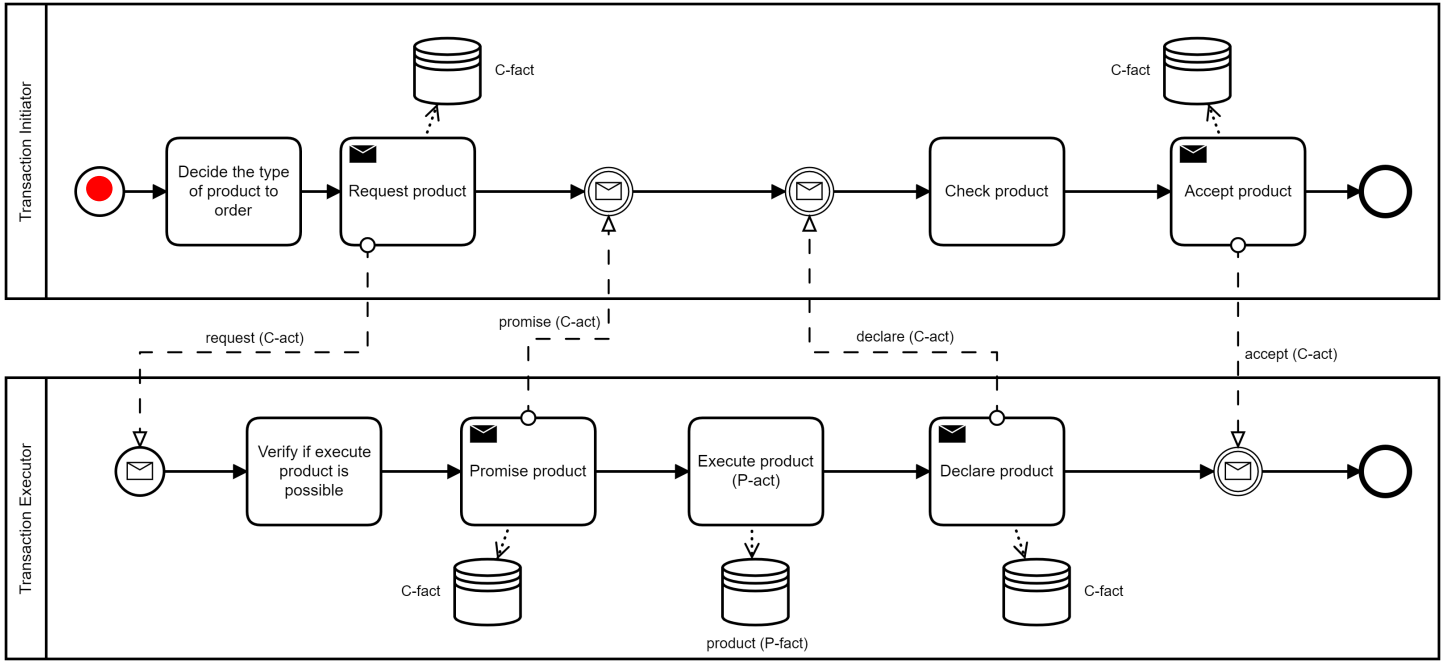




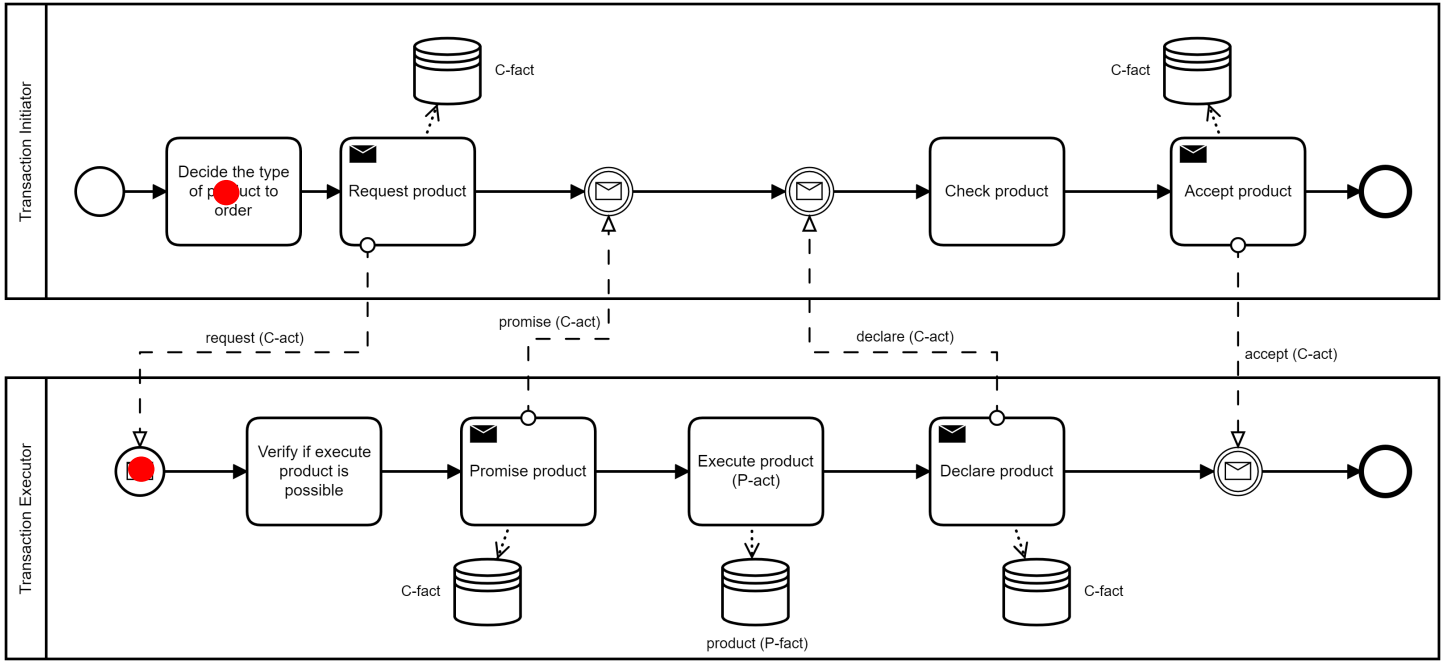
2 ACTORS COMMUNICATING…what happens… if everything runs happily?

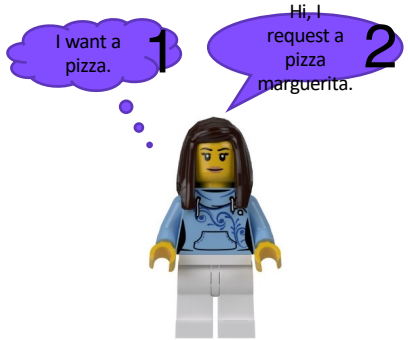
Simulating the Communication pattern between 2 participants, as a social system…

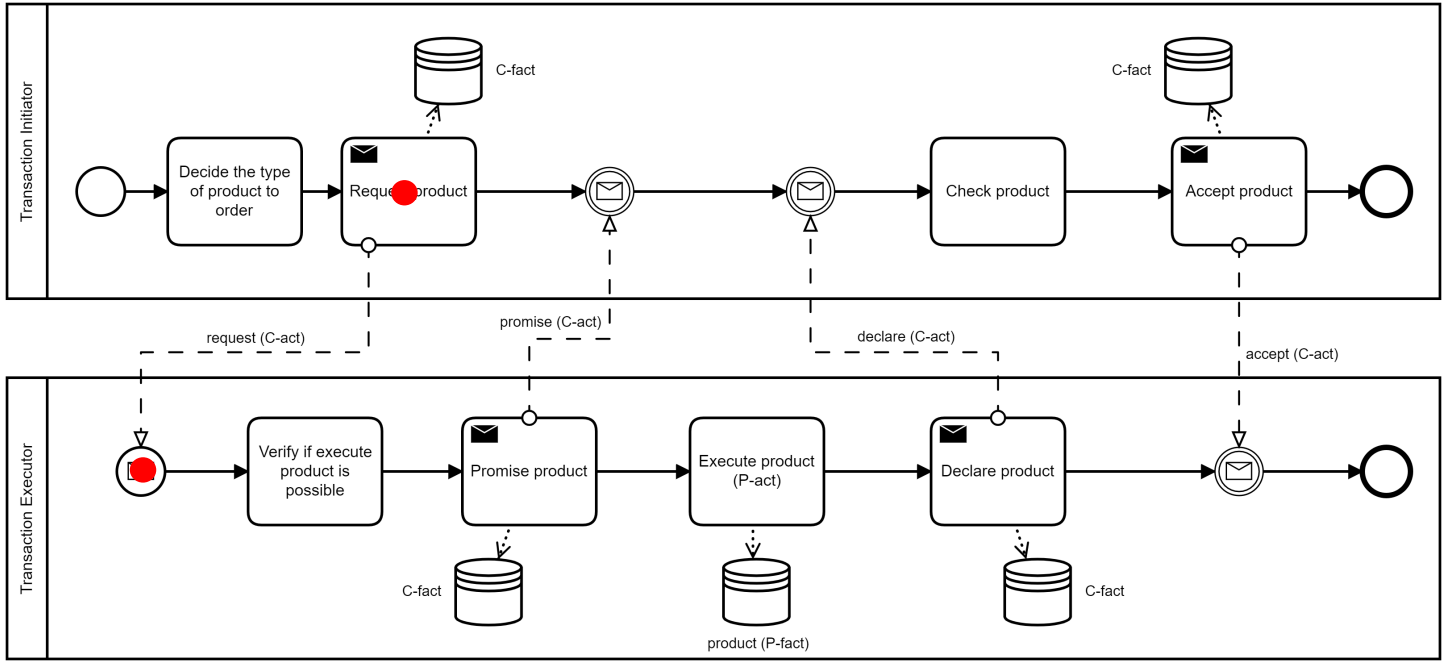
In BPMN


Transaction Initiator
C-fact
C-fact
Decide the type of product to order
Request product
Check product
Accept product
request (C-act)
promise (C-act)
declare (C-act)
accept (C-act)
Transaction Executor
Verify if execute product is possible
Promise product
Execute product (P-act)
Declare product
C-fact
C-fact
product (P-fact)

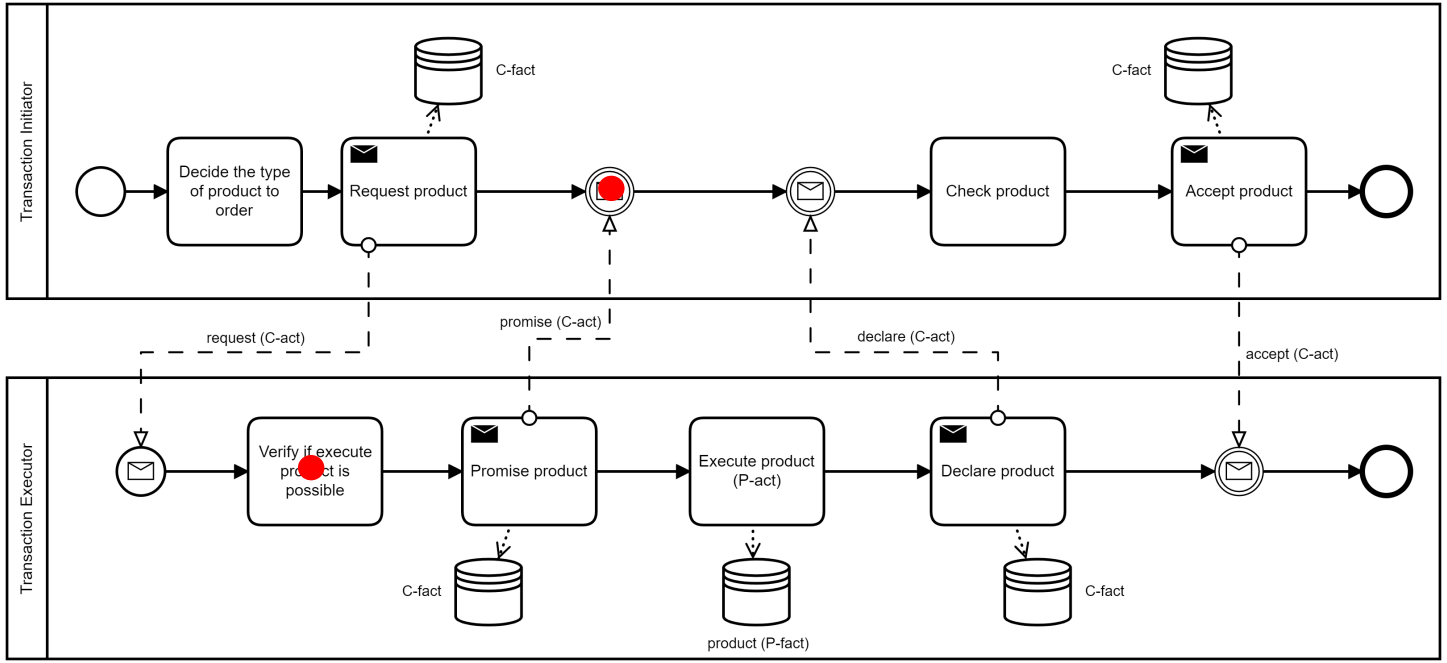

I want a pizza.

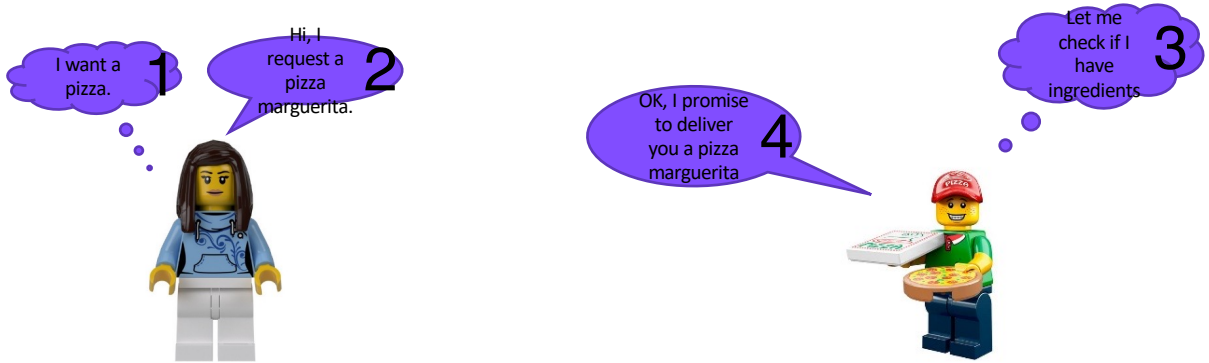

Transaction Initiator
C-fact
C-fact
Decide the type of product to order
Request product
Check product
Accept product
request (C-act)
promise (C-act)
declare (C-act)
accept (C-act)
Transaction Executor
Verify if execute product is possible
Promise product
Execute product (P-act)
Declare product
C-fact
C-fact
product (P-fact)

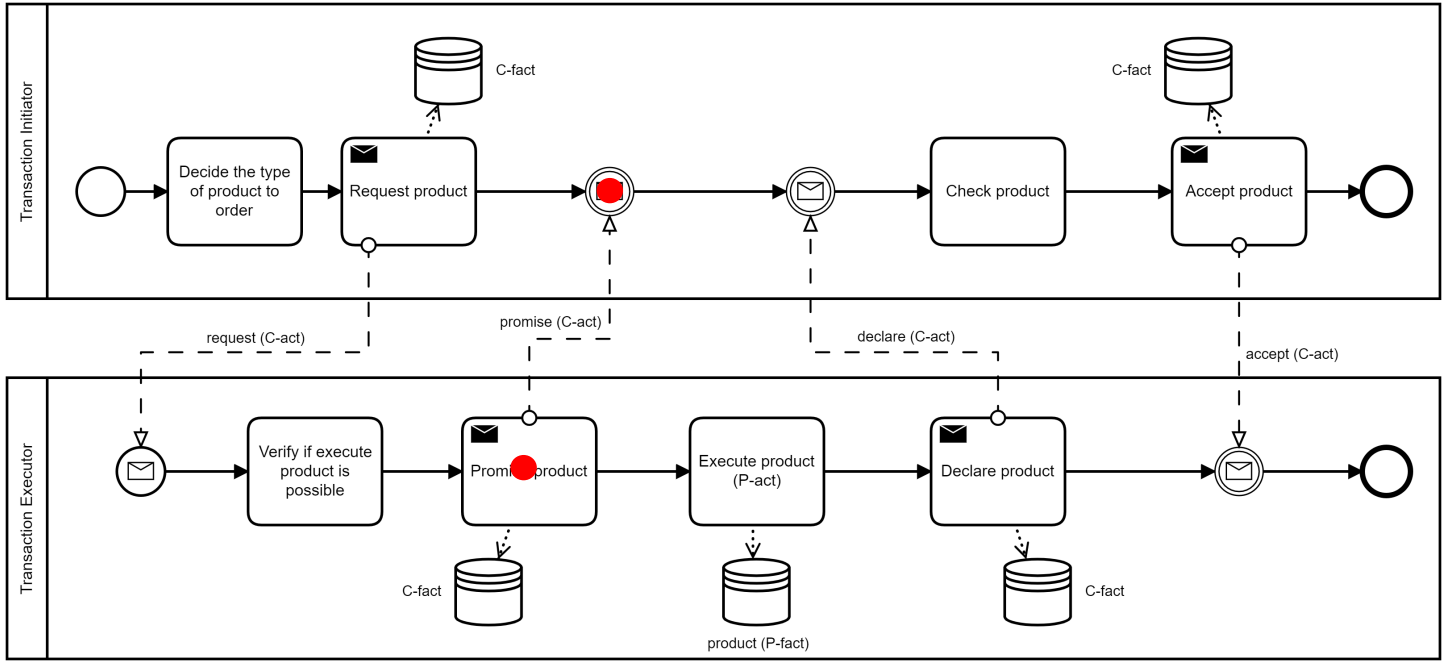

I want a pizza.
1
Hi, I request a pizza marguerita.
2


Transaction Initiator
C-fact
C-fact
Decide the type of product to order
Request product
Check product
Accept product
request (C-act)
promise (C-act)
declare (C-act)
accept (C-act)
Transaction Executor
Verify if execute product is possible
Promise product
Execute product (P-act)
Declare product
C-fact
C-fact
product (P-fact)

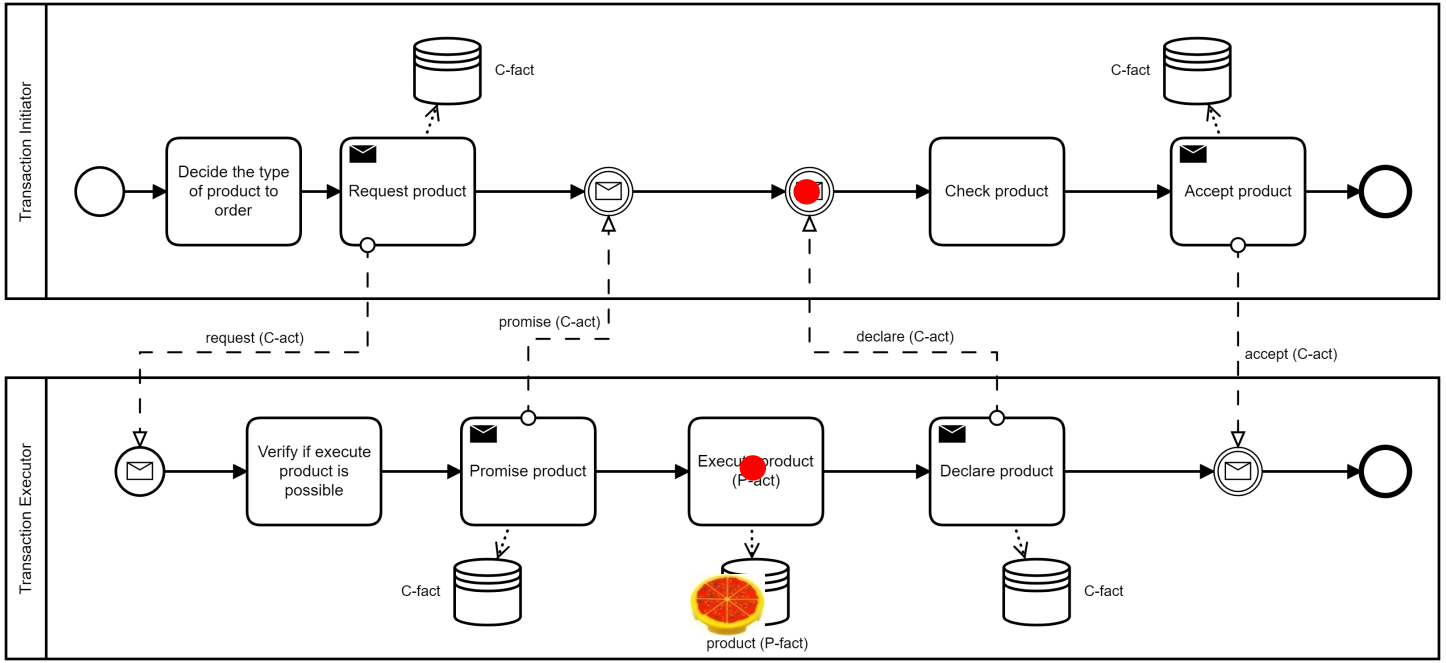

I want a pizza.
1
Hi, I request a pizza marguerita.
2
Let me check if I have ingredients
3


Transaction Initiator
C-fact
C-fact
Decide the type of product to order
Request product
Check product
Accept product
request (C-act)
promise (C-act)
declare (C-act)
accept (C-act)
Transaction Executor
Verify if execute product is possible
Promise product
Execute product (P-act)
Declare product
C-fact
C-fact
product (P-fact)

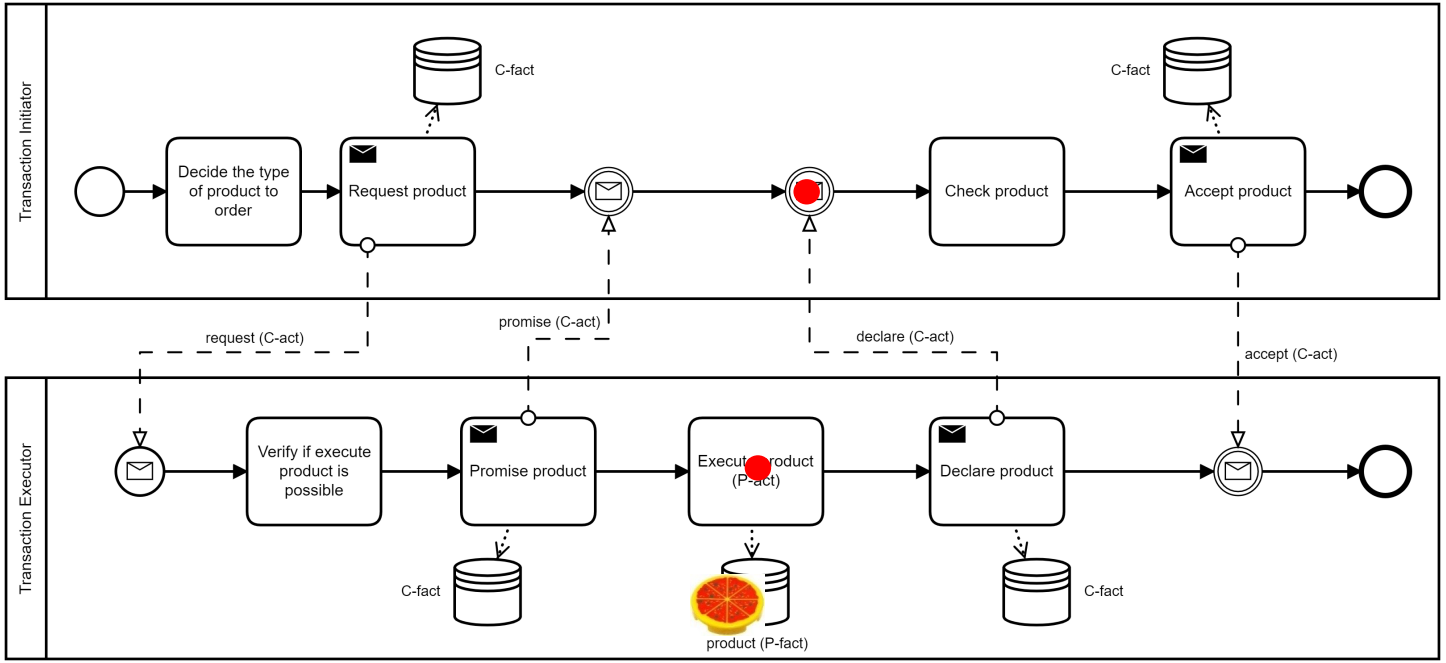

I want a pizza. 1
Hi, I request a pizza marguerita. 2
Let me check if I have ingredients 3
OK, I promise to deliver you a pizza marguerita 4

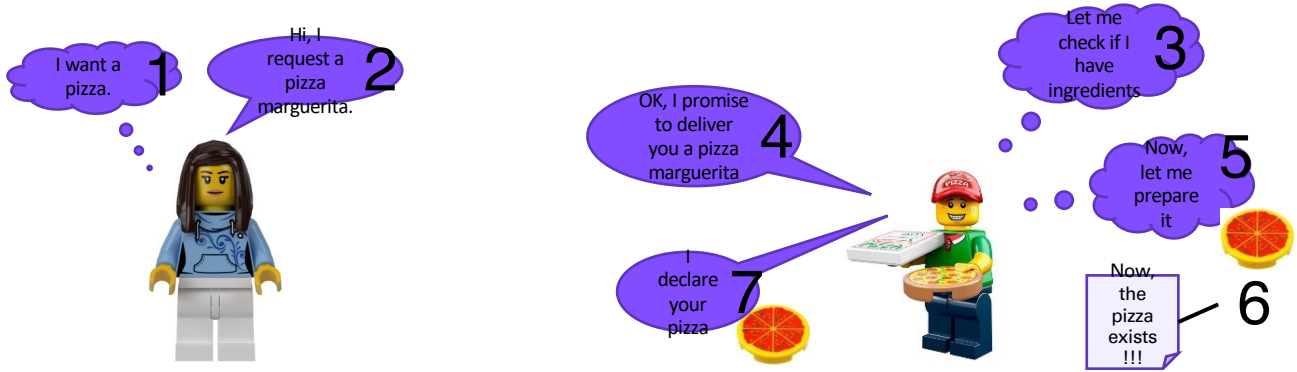

Transaction Initiator
C-fact
C-fact
Decide the type of product to order
Request product
Check product
Accept product
request (C-act)
promise (C-act)
declare (C-act)
accept (C-act)
Transaction Executor
Verify if execute product is possible
Execute product (P-act)
Declare product
C-fact
C-fact
product (P-fact)

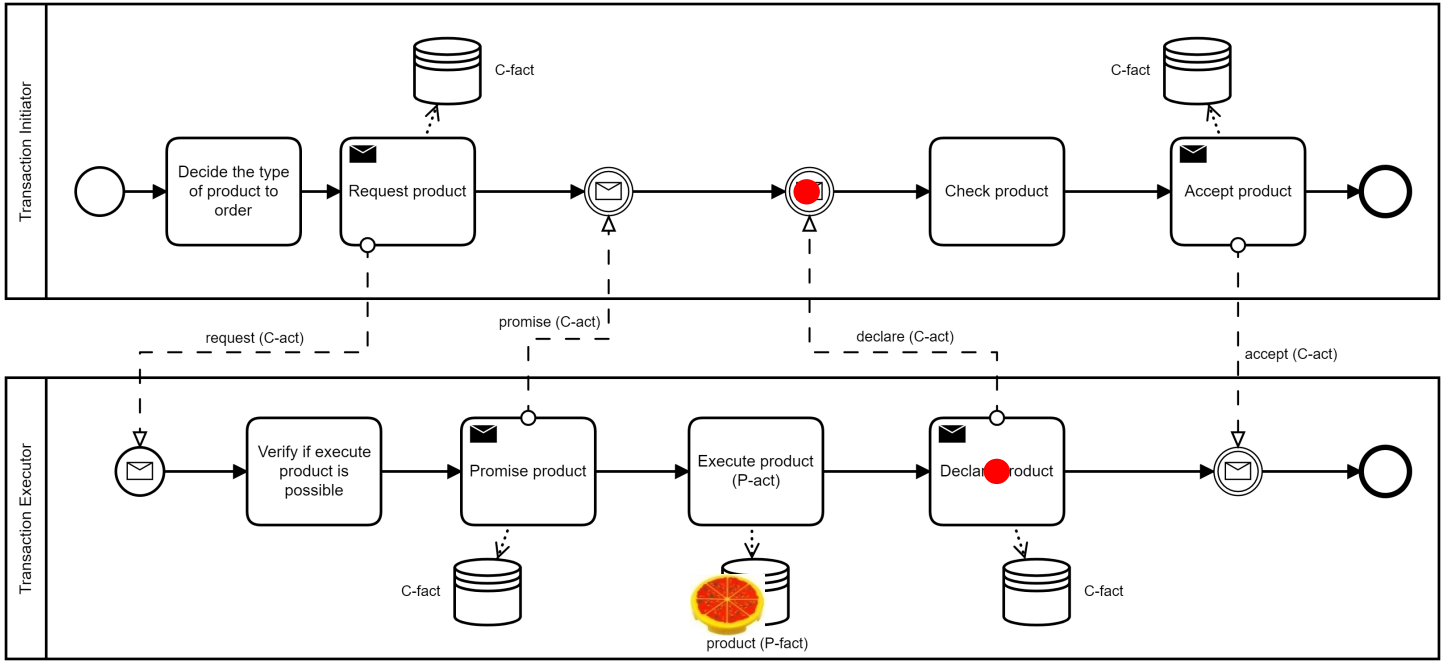

I want a pizza.
1
Hi, I request a pizza marguerita.
2
Let me check if I have ingredients
3
OK, I promise to deliver you a pizza marguerita
4
Now, let me prepare it
5

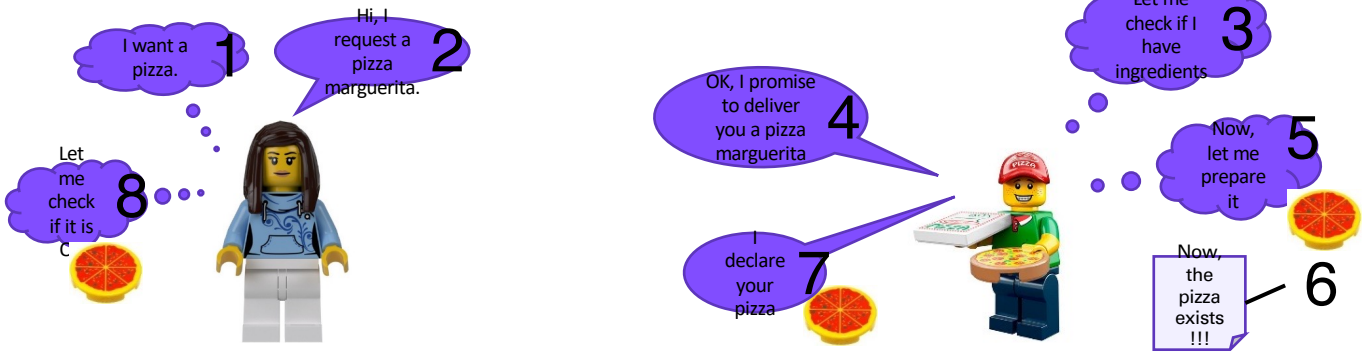

Transaction Initiator
C-fact
C-fact
Decide the type of product to order
Request product
Check product
Accept product
request (C-act)
promise (C-act)
declare (C-act)
accept (C-act)
Transaction Executor
Verify if execute product is possible
Promise product
Declare product
C-fact
C-fact
product (P-fact)

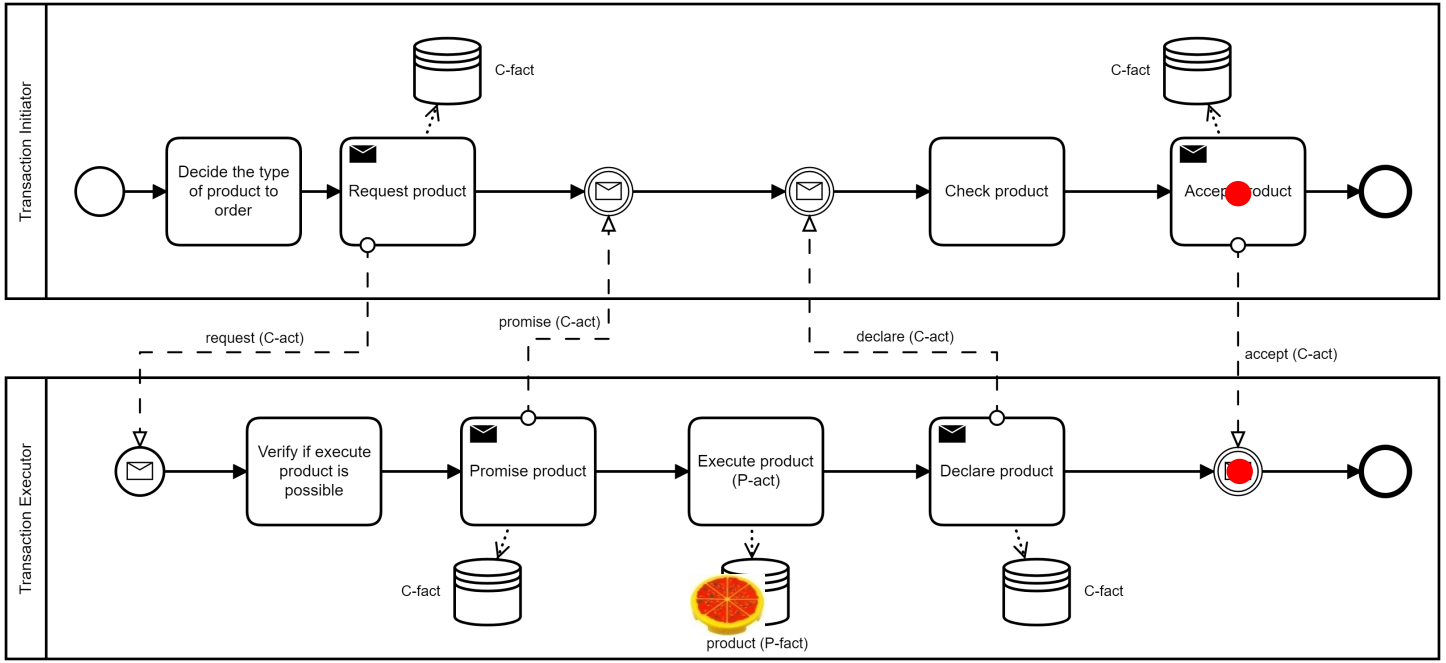

I want a pizza.
1
Hi, I request a pizza marguerita.
2
Let me check if I have ingredients
3
OK, I promise to deliver you a pizza marguerita
4
Now, let me prepare it
5
Now, the pizza exists in the world
P-fact
6

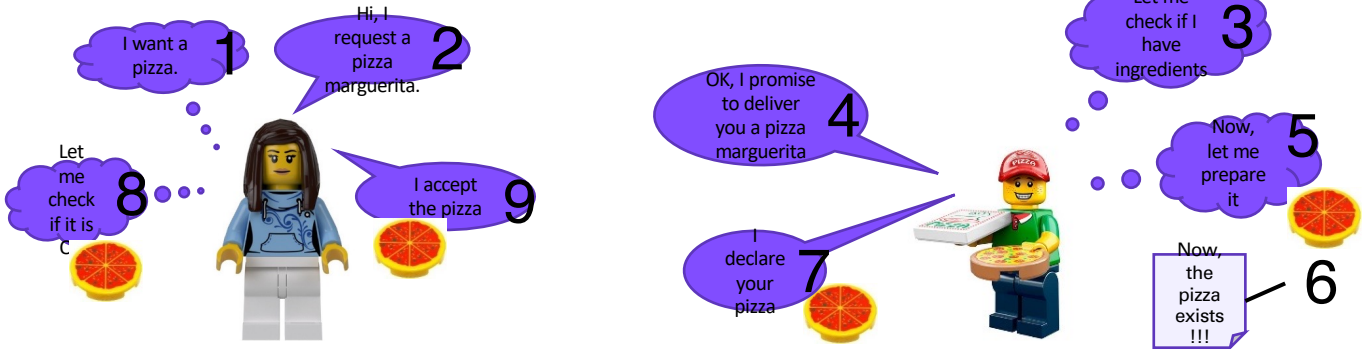

Transaction Initiator
C-fact
C-fact
Decide the type of product to order
Request product
Check product
Accept product
request (C-act)
promise (C-act)
declare (C-act)
accept (C-act)
Transaction Executor
Verify if execute product is possible
Promise product
Execute product (P-act)
Declare product
C-fact
C-fact
product (P-fact)


I want a pizza. 1
Hi, I request a pizza marguerita. 2
Let me check if I have ingredients 3
OK, I promise to deliver you a pizza marguerita 4
Now, let me prepare it 5
I declare your pizza 7
Now, the pizza exists !!! 6

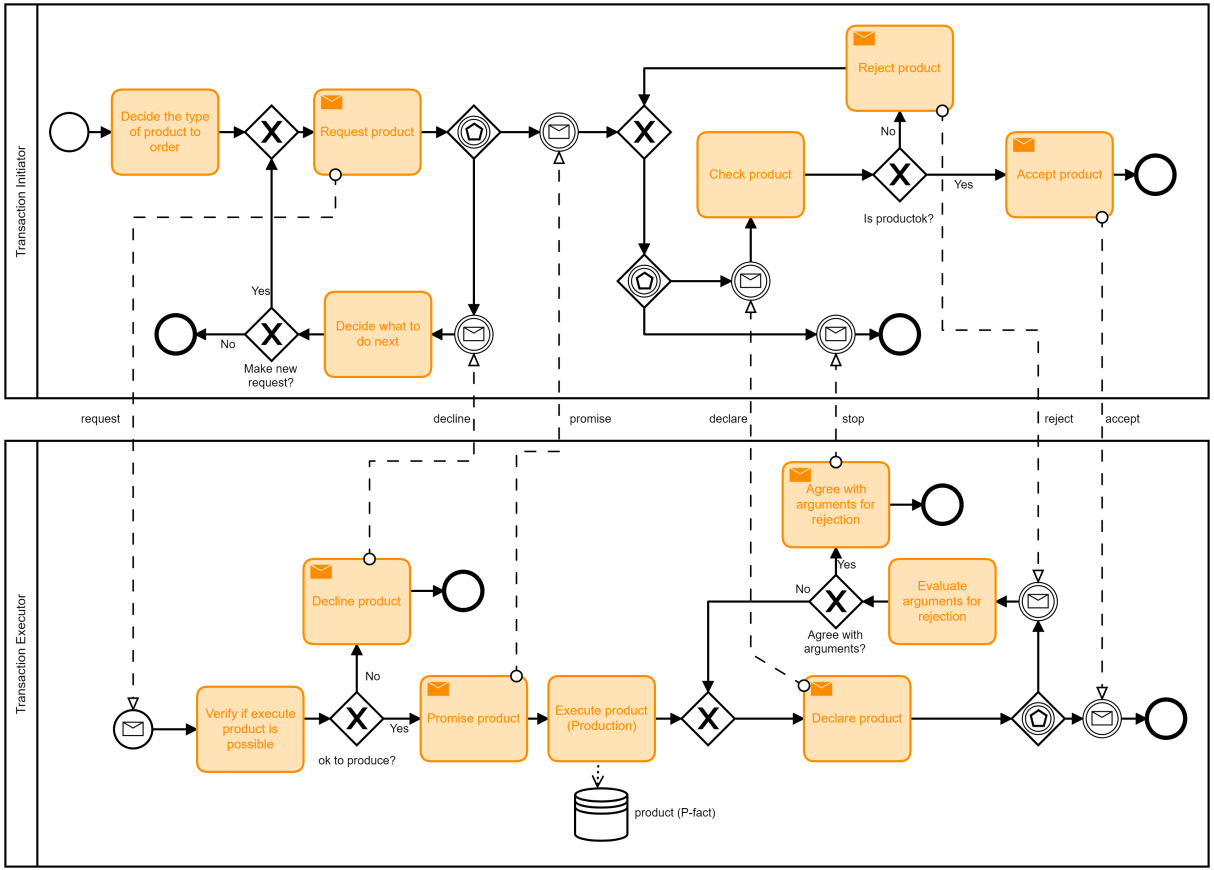

Transaction Initiator
C-fact
C-fact
Decide the type of product to order
Request product
Check product
Accept product
request (C-act)
promise (C-act)
declare (C-act)
accept (C-act)
Transaction Executor
Verify if execute product is possible
Promise product
Execute product (P-act)
C-fact
C-fact
product (P-fact)


I want a pizza. 1
Hi, I request a pizza marguerita. 2
Let me check if I have ingredients 3
OK, I promise to deliver you a pizza marguerita 4
Now, let me prepare it 5
Now, the pizza exists !!! 6
I declare your pizza 7
Let me check if it is 8


Transaction Initiator
C-fact
C-fact
Decide the type of product to order
Request product
Check product
Accept product
request (C-act)
promise (C-act)
declare (C-act)
accept (C-act)
Transaction Executor
Verify if execute product is possible
Promise product
Execute product (P-act)
Declare product
C-fact
C-fact
product (P-fact)


I want a pizza. 1
Hi, I request a pizza marguerita. 2
Let me check if I have ingredients 3
OK, I promise to deliver you a pizza marguerita 4
Now, let me prepare it 5
Let me check if it is
8
I accept the pizza 9
I declare your pizza 7
Now, the pizza exists !!! 6

Enrichment – business transaction pattern –
happy flow + rejections + declinations



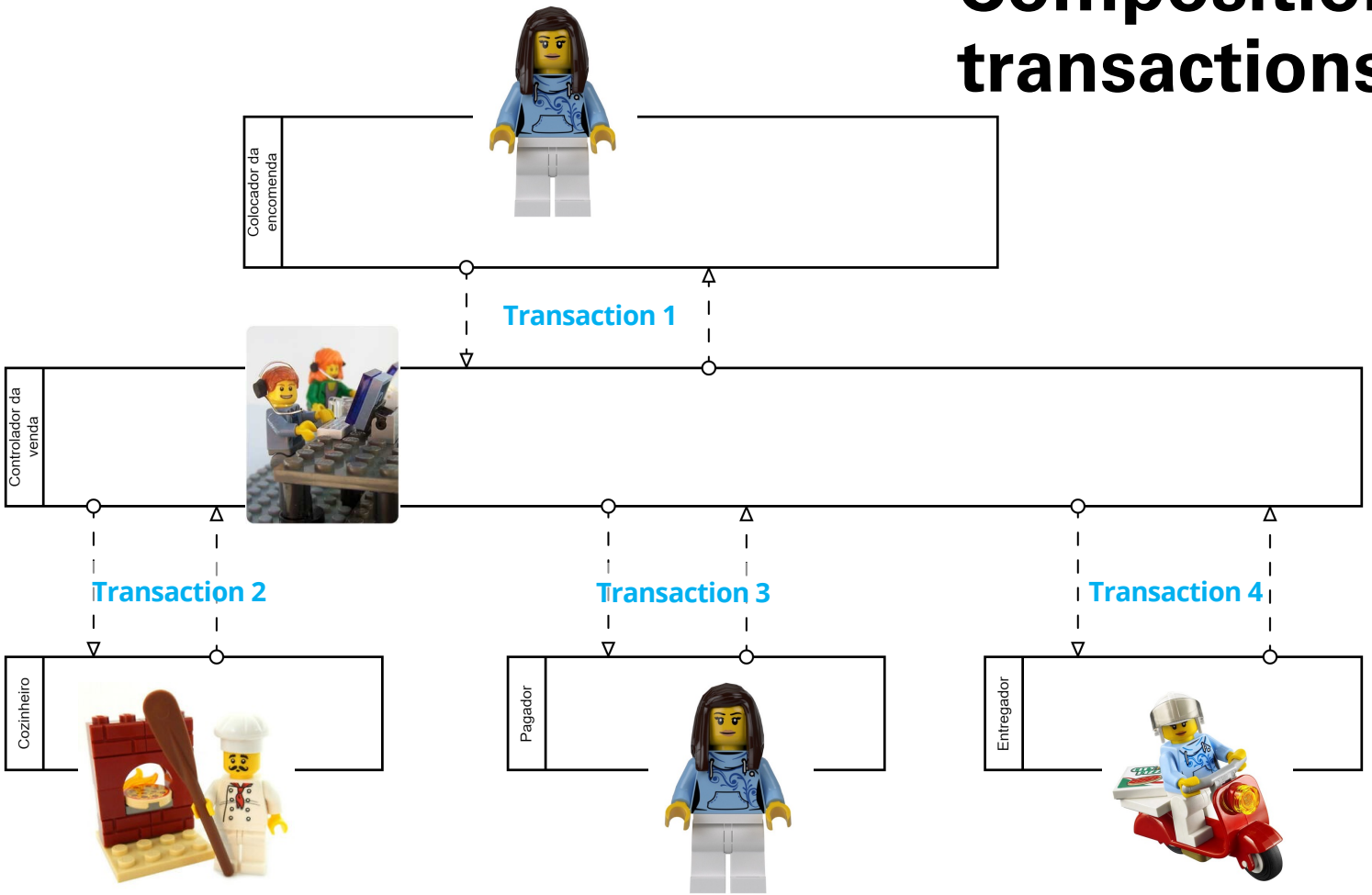
13 variability points =>

2^{13} =

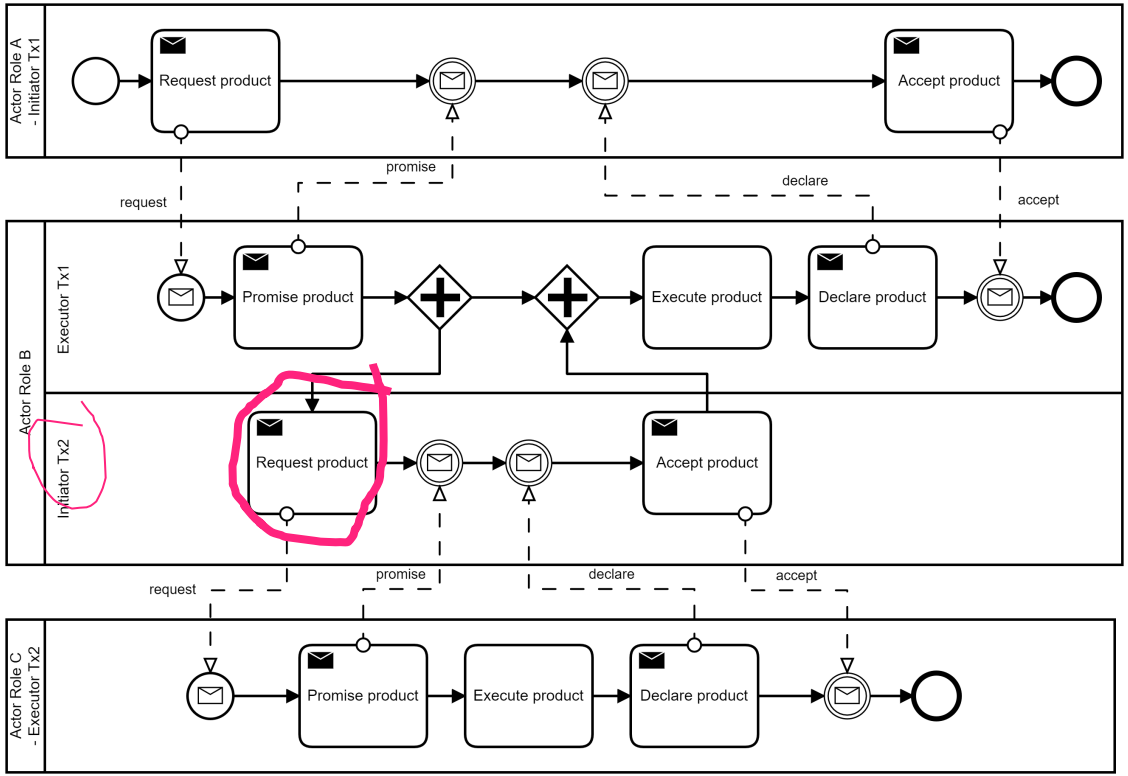
8192

process configurations

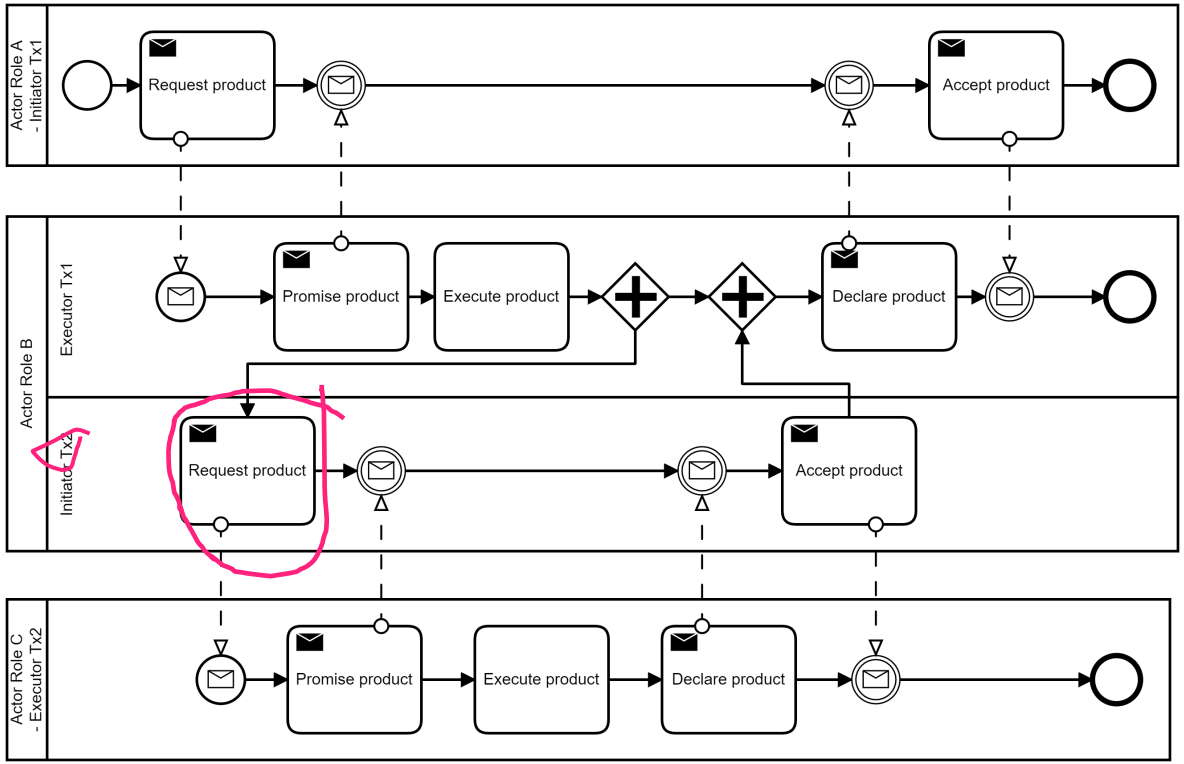
Composition of transactions…



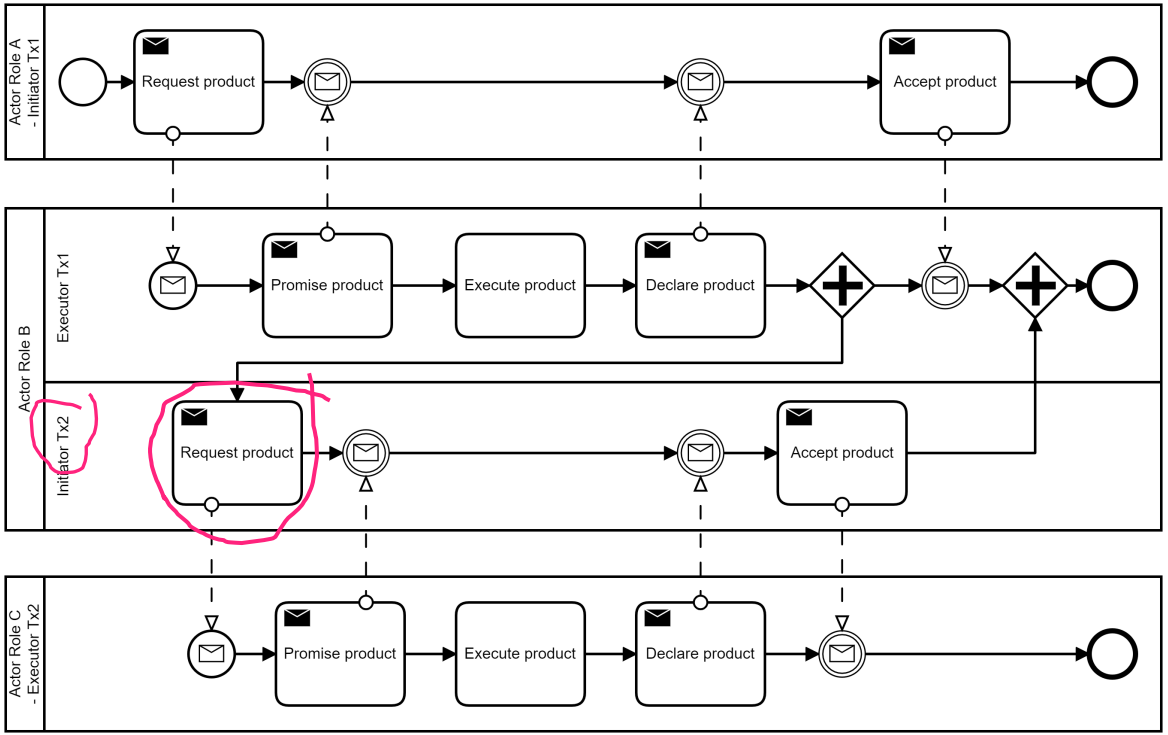
Transactions dependency – Request after Promise



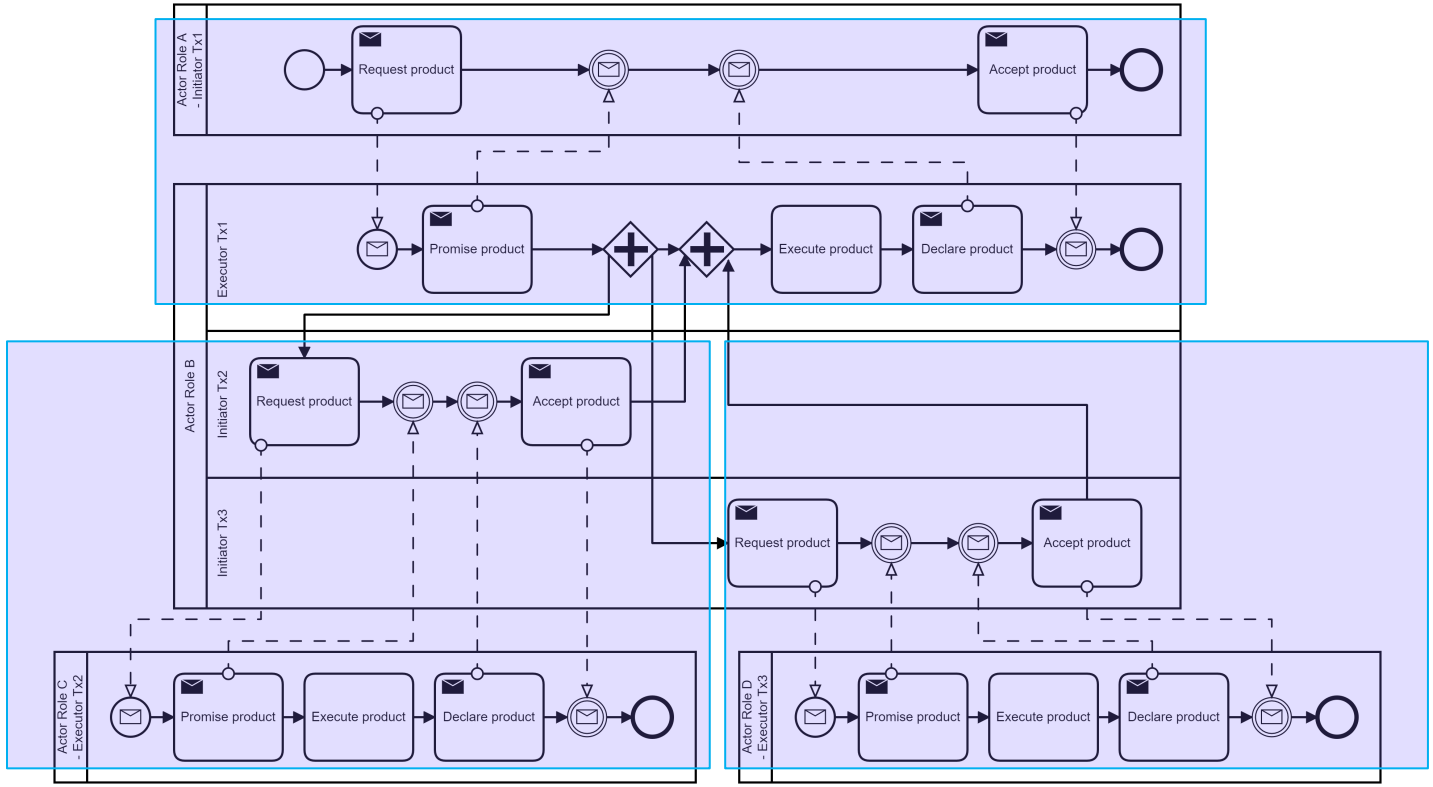
Transactions dependency – Request after Execute



Transactions dependency – Request after Declare



RaP composition pattern concerning 3 business transactions where TK2 and TK3 are performed simultaneously.



https://github.com/SemantifyingBPMN/SemantifyingBPMN

```
target % java -jar SemantifyingBPMN-4.0.1.jar

The usage of SemantifyingBPMN is the following.
SemantifyingBPMN-4.0.1 --actors <filename> --tpt <filename> --tkdepend <filename> --output-file-txt <filename> --output-file-bpmn <filename>
Credits: Sérgio Guerreiro (2022) (github: https://github.com/SemantifyingBPMN/SemantifyingBPMN)

where the parameters are,
--actors: is a csv file with the list of actor roles and is mandatory. Composed of 2 fields, in each line, with actor role name and description:
 (e.g.: A01 - Customer ; The role that initiates the business process).
--tpt: is a csv file with the Transactor Product Table and is mandatory. Composed of 6 fields, in each line, with TK name, TK description, Actor role
initiator, Actor role executor, Product kind , Product kind description:
 (e.g.: TK01; Sale completing ; A01 - Customer  ; A02 - Dispatcher ; PK01 ; [Product] is sold).
--tkdepend: is a csv file with the dependencies matrix N*N transactions and is mandatory. Composed of Strings with dependencies: RaP = Request after Promise
pattern, RaE = Request after Execution, RaD = Request after Declare pattern.
 (e.g.:
            ; TK01  ; TK02  ; TK03 ; TK04
       TK01 ;       ;       ;      ;
       TK02 ; RaP   ;       ;      ;
       TK03 ;       ; RaE   ;      ;
       TK04 ;       ;       ; RaE  ;
 )
--tkview: is a mandatory csv file with view definition for each transaction per line, acceptable values are: HappyFlow |
HappyFlowAndDeclinationsAndRejections | Complete | Custom | CustomHappyFlowOnly. Default value is HappyFlow.
         The Custom value accepts extra detail for each transaction step, even empty ones.
 (e.g.
      TransactionKind  ; View   ; Request Decision ; Request ; Promise Decision ; Promise ; Decline ; After Decline Decision ; Execute ; Declare       ;
Decision Accept ; Accept ; Reject ; Evaluate Rejection ; Stop
                TK01 ; HappyFlow
                TK02 ; HappyFlowAndDeclinationsAndRejections
                TK03 ; Custom ;                  ; Pedido  ;                ;         ; Executa ; how to decide          ; Produce ; Here it is     ;
Valida resultado;        ; not ok ; decide reject      ; ok
                TK04 ; Complete
                TK05 ;
 )
--businessObjects: is a csv file with the list of business object for each transaxction step.
 (e.g.
  TransactionKind ; TransactionStep ; BusinessObject
  TK01 ; Request    ; Order
  TK02 ; Promise   ; Receipt
)
--simplify on|off: create a BPMN model without communications for simplification purposes only. Optional, default is off.
--engine camunda|... : create a BPMN model ready for CAMUNDA execution. Optional.
--output-file-txt: is a file to store the model in txt format. Optional.
--output-file-bpmn: is a file to store the BPMN model. Optional.
```

BPMNSemantifying tool OUTCOME

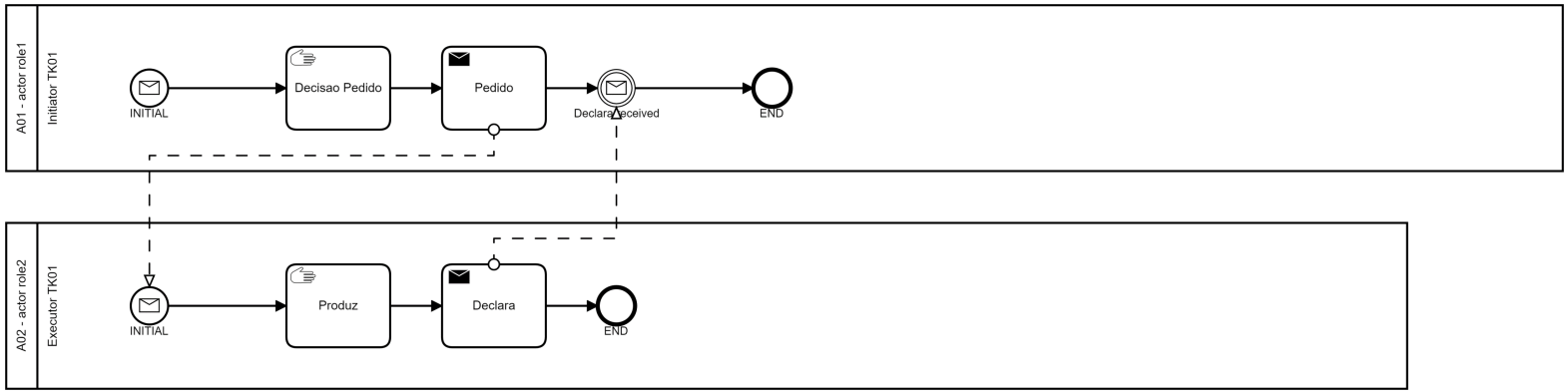
```
root@0736c3e84db2:~# more actors
A01 - actor role1 ; Budget transfer solicitor.
A02 - actor role2 ;  Executor of the first validation and creator of a budget transfer proposal.
root@0736c3e84db2:~# more tpt
TK01;Soliciting budget change; A01 - actor role1  ; A02 - actor role2 ; PK01 ; [budget] has been
changed
root@0736c3e84db2:~# more tkdepend
    ; TK01
TK01;
```

```
root@0736c3e84db2:~# more tkview3
TransactionKind  ; View               ; Request Decision ; Request ; Promise Decision ; Promise ; Decline ; After Decline Decision ; Execute ; Declare       ; Decision Accept      ; Accept ; Reject ; Evaluate Rejection ; Stop
TK01             ; CustomHappyFlowOnly ;                 ;         ;                 ;         ;         ;                       ;         ;               ;                     ;        ;        ;                   ;
```

BPMNSemantifying tool OUTCOME

```
root@0736c3e84db2:~# more actors
A01 - actor role1 ; Budget transfer solicitor.
A02 - actor role2 ;  Executor of the first validation and creator of a budget transfer proposal.
root@0736c3e84db2:~# more tpt
TK01;Soliciting budget change; A01 - actor role1  ; A02 - actor role2 ; PK01 ; [budget] has been
changed
root@0736c3e84db2:~# more tkdepend
    ; TK01
TK01;
```

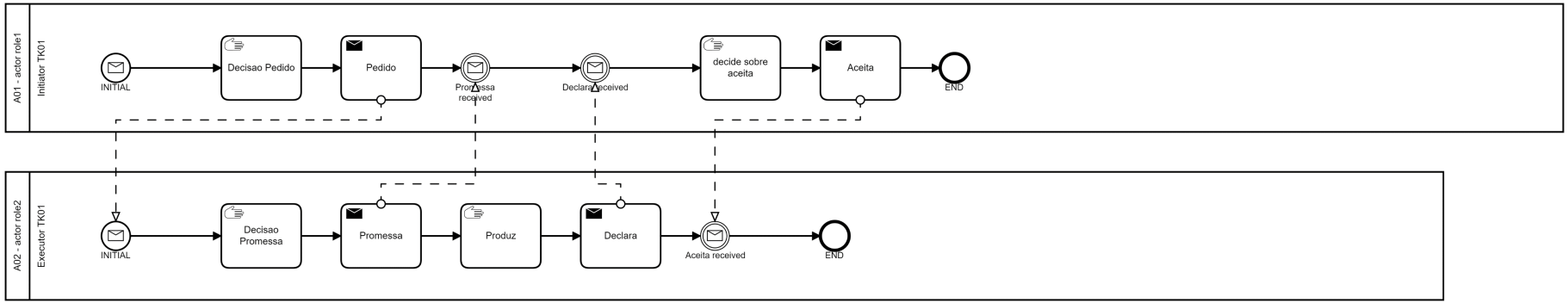
```
root@0736c3e84db2:~# more tkview2
TransactionKind  ; View                ; Request Decision ; Request ; Promise Decision ; Promise ; Decline ; After Decline Decision ; Execute ; Declare       ; Decision Accept      ; Accept ; Reject ; Evaluate Rejection ; Stop
TK01             ; CustomHappyFlowOnly  ; Decisao Pedido   ; Pedido  ;                ;         ;         ;                       ;   Produz;  Declara      ;                     ;        ;        ;                   ;
```



BPMNSemantifying tool OUTCOME

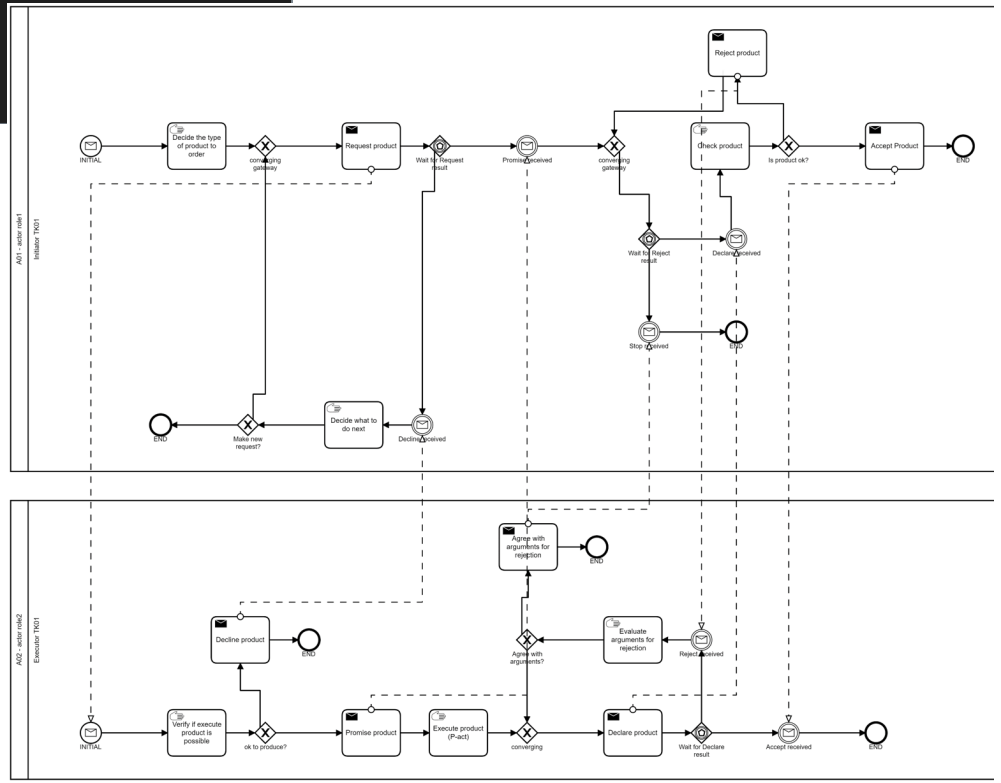
```
root@0736c3e84db2:~# more actors
A01 - actor role1 ; Budget transfer solicitor.
A02 - actor role2 ;  Executor of the first validation and creator of a budget transfer proposal.
root@0736c3e84db2:~# more tpt
TK01;Soliciting budget change; A01 - actor role1  ; A02 - actor role2 ; PK01 ; [budget] has been
changed
root@0736c3e84db2:~# more tkdepend
    ; TK01
TK01;
```

```
root@0736c3e84db2:~# more tkview1
TransactionKind  ; View                ; Request Decision ; Request ; Promise Decision ; Promise ; Decline ; After Decline Decision ; Execute ; Declare       ; Decision Accept      ; Accept ; Reject ; Evaluate Rejection ; Stop
TK01             ; CustomHappyFlowOnly  ; Decisao Pedido   ; Pedido  ; Decisao Promessa ; Promessa;         ;                       ;   Produz;  Declara      ; decide sobre aceita  ;  Aceita;        ;                    ;
```



```
root@0736c3e84db2:~# more tkview6
TransactionKind  ; View  ; Request Decision ; Request ; Promise Decision ; Promise ; Decline ; After Decline Decision ; Execute ; Declare       ; Decision Accept  ; Accept ; Reject ; Evaluate Rejection ; Stop
TK01             ; HappyFlowAndDeclinationsAndRejections ;  ;  ;  ;  ;  ;  ;  ;  ;  ;  ;  ;
```

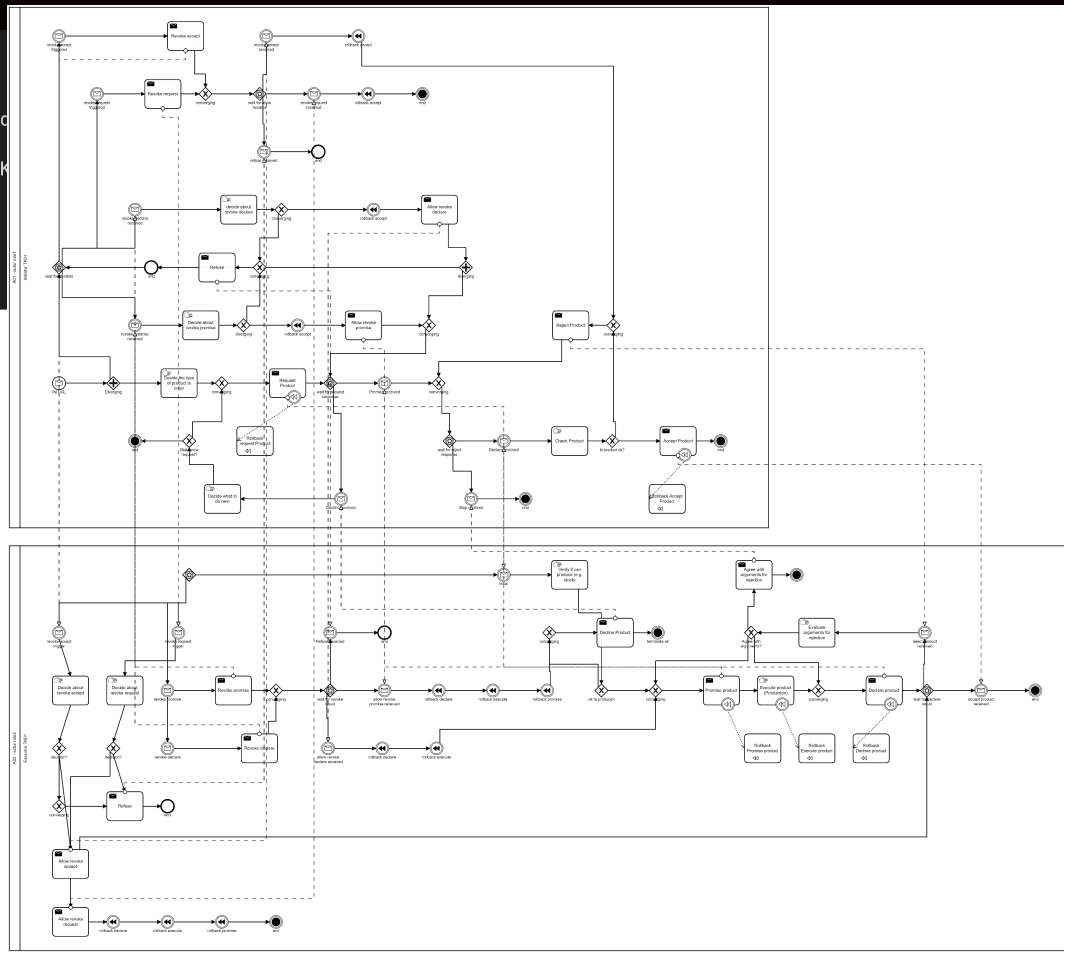
```
root@0736c3e84db2:~# more actors
A01 - actor role1 ; Budget transfer solicitor.
A02 - actor role2 ;  Executor of the first validation and creator of a budget transfer proposal.
root@0736c3e84db2:~# more tpt
TK01;Soliciting budget change; A01 - actor role1  ; A02 - actor role2 ; PK01 ; [budget] has been
changed
root@0736c3e84db2:~# more tkdepend
    ; TK01
TK01;
```



BPMNSemantifying tool OUTCOME

```
root@0736c3e84db2:~# more tkview6
TransactionKind  ; View  ; Request Decision ; Request ; Promise Decision ; Promise ; Decline ; After Decline Decision ; Execute ; Declare       ; Decision Accept  ; Accept ; Reject ; Evaluate Rejection ; Stop
TK01             ; Complete ;  ;  ;  ;  ;  ;  ;  ;  ;  ;  ;  ;  ;
```

```
root@0736c3e84db2:~# more actors
A01 - actor role1 ; Budget transfer solicitor.
A02 - actor role2 ;  Executor of the first validation and creator of a buc
root@0736c3e84db2:~# more tpt
TK01;Soliciting budget change; A01 - actor role1  ; A02 - actor role2 ; PK
changed
root@0736c3e84db2:~# more tkdepend
    ; TK01
TK01;
```

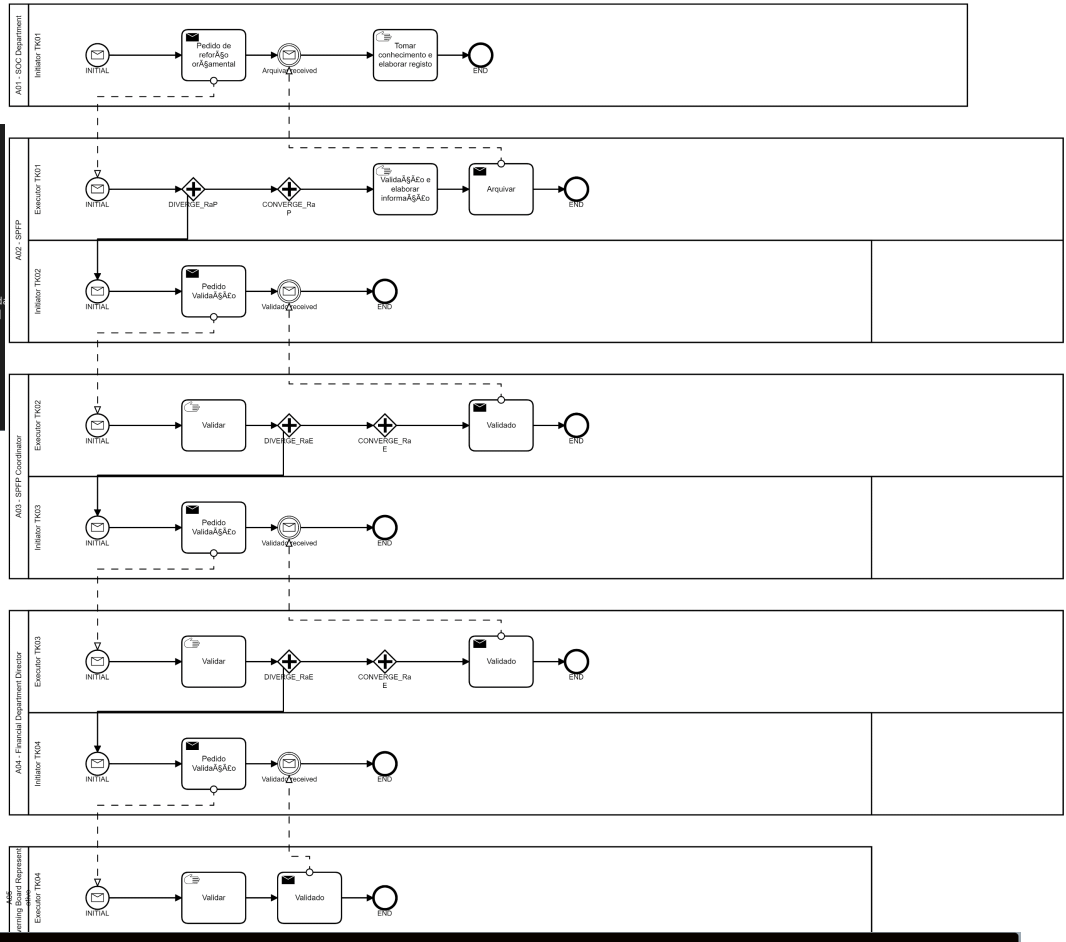


BPMNSemantifying tool OUTCOME

```
root@0736c3e84db2:~# more actors4
A01 - SOC Department ; Budget transfer solicitor.
A02 - SPFP                  ; Executor of the first validation and creator of a budget transfer proposal.
A03 - SPFP Coordinator  ;  Executor of the second validation of the budget transfer.
A04 - Financial Department Director          ;   Executor of the third validation of the budget transfer.
A05 - Governing Board Representative    ;   Executor of the fourth validation of the budget transfer.

root@0736c3e84db2:~# more tpt4
TK01;Soliciting budget change; A01 - SOC Department  ; A02 - SPFP ; PK01 ; [budget] has been changed
TK02;Validating second-level budget change; A02 - SPFP ; A03 - SPFP Coordinator; PK02 ; [budget change] has been validated by second level
TK03;Validating third-level budget change;A03 - SPFP Coordinator;A04 - Financial Department Director; PK03 ; [budget change] has been valid
TK04;Validating fourth-level budget change;A04 - Financial Department Director;A05 - Governing Board Representative; PK04 ; [budget change]

root@0736c3e84db2:~# more tkdepend4
    ; TK01; TK02;TK03;TK04
TK01;     ;     ;    ;
TK02; RaP ;     ;    ;
TK03;     ; RaE ;    ;
TK04;     ;     ;RaE ;
```



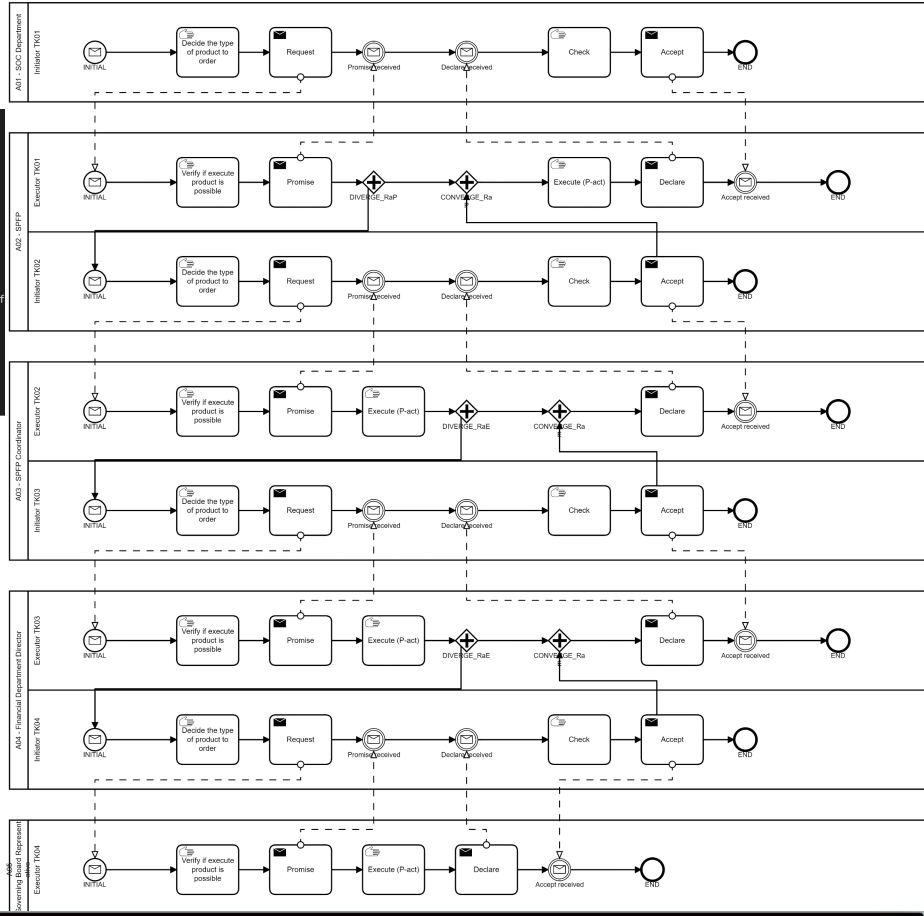
BPMNSemantifying tool OUTCOME

```
root@0736c3e84db2:~# more tkview4
TransactionKind ; View             ; Request Decision; Request                    ; Promise Decision   ; Promise ; Decline ; After Decline Decision ; Execute                    ; Declare       ; Decision Accept                        ; Accept ; Reject ; Evaluate Rejection ; Stop
TK01            ; CustomHappyFlowOnly  ;              ; Pedido de reforço orçamental ;                   ;         ;         ;                       ;Validação e elaborar informação;  Arquivar     ;   Tomar conhecimento e elaborar registo ;        ;        ;                    ;
TK02            ; CustomHappyFlowOnly  ;              ;    Pedido Validação          ;                   ;         ;         ;                       ;  Validar                  ;  Validado     ;                                        ;        ;        ;                    ;
TK03            ; CustomHappyFlowOnly ;               ;    Pedido Validação          ;                   ;         ;         ;                       ;  Validar                  ;  Validado     ;                                        ;        ;        ;                    ;
TK04            ; CustomHappyFlowOnly ;               ;    Pedido Validação          ;                   ;         ;         ;                       ;  Validar                  ;  Validado     ;                                        ;        ;        ;                    ;
```

```
root@0736c3e84db2:~# more actors4
A01 - SOC Department ; Budget transfer solicitor.
A02 - SPFP                  ;  Executor of the first validation and creator of a budget transfer proposal.
A03 - SPFP Coordinator  ;  Executor of the second validation of the budget transfer.
A04 - Financial Department Director          ;   Executor of the third validation of the budget transfer.
A05 - Governing Board Representative    ;   Executor of the fourth validation of the budget transfer.

root@0736c3e84db2:~# more tpt4
TK01;Soliciting budget change; A01 - SOC Department  ; A02 - SPFP ; PK01 ; [budget] has been changed
TK02;Validating second-level budget change; A02 - SPFP ; A03 - SPFP Coordinator; PK02 ; [budget change] has been validated by second level
TK03;Validating third-level budget change;A03 - SPFP Coordinator;A04 - Financial Department Director; PK03 ; [budget change] has been validated by third level
TK04;Validating fourth-level budget change;A04 - Financial Department Director;A05 - Governing Board Representative; PK04 ; [budget change] has been validated by f

root@0736c3e84db2:~# more tkdepend4
    ; TK01; TK02;TK03;TK04
TK01;     ;     ;    ;
TK02; RaP ;     ;    ;
TK03;     ; RaE ;    ;
TK04;     ;     ;RaE ;
```



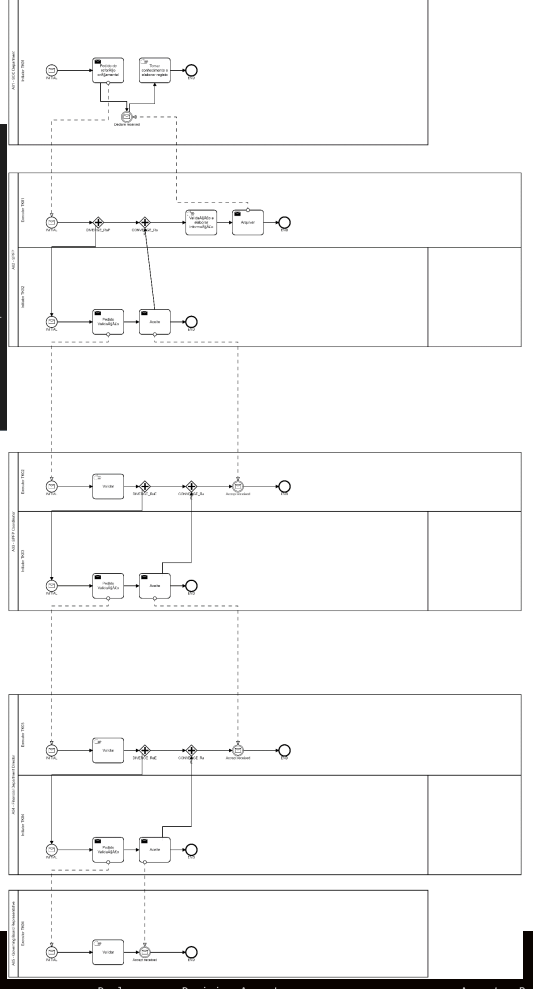
BPMNSemantifying tool OUTCOME

```
root@0736c3e84db2:~# more tkview5
TransactionKind ; View      ; Request Decision; Request                    ; Promise Decision   ; Promise ; Decline ; After Decline Decision ; Execute                    ; Declare       ; Decision Accept                          ; Accept ; Reject ; Evaluate Rejection ; Stop
TK01            ; HappyFlow  ;                 ; Pedido de reforço orçamental  ;                    ;         ;         ;                       ;Validação e elaborar informação;  Arquivar      ;   Tomar conhecimento e elaborar registo  ;        ;        ;                    ;
TK02            ; HappyFlow  ;                 ;   Pedido Validação           ;                    ;         ;         ;                       ;  Validar                   ;  Validado     ;                                        ;        ;        ;                    ;
TK03            ; HappyFlow ;                  ;   Pedido Validação           ;                    ;         ;         ;                       ;  Validar                   ;  Validado     ;                                        ;        ;        ;                    ;
TK04            ; HappyFlow ;                  ;   Pedido Validação           ;                    ;         ;         ;                       ;  Validar                   ;  Validado     ;                                        ;        ;        ;                    ;
```

```
root@0736c3e84db2:~# more actors4
A01 - SOC Department ; Budget transfer solicitor.
A02 - SPFP                  ;  Executor of the first validation and creator of a budget transfer proposal.
A03 - SPFP Coordinator  ;  Executor of the second validation of the budget transfer.
A04 - Financial Department Director          ;   Executor of the third validation of the budget transfer.
A05 - Governing Board Representative    ;   Executor of the fourth validation of the budget transfer.

root@0736c3e84db2:~# more tpt4
TK01;Soliciting budget change; A01 - SOC Department  ; A02 - SPFP ; PK01 ; [budget] has been changed
TK02;Validating second-level budget change; A02 - SPFP ; A03 - SPFP Coordinator; PK02 ; [budget change] has been validated by second level
TK03;Validating third-level budget change;A03 - SPFP Coordinator;A04 - Financial Department Director; PK03 ; [budget change] has been validated by third level
TK04;Validating fourth-level budget change;A04 - Financial Department Director;A05 - Governing Board Representative; PK04 ; [budget change] has been validated by fourth level

root@0736c3e84db2:~# more tkdepend4
    ; TK01; TK02;TK03;TK04
TK01;     ;     ;    ;
TK02; RaP ;     ;    ;
TK03;     ; RaE ;    ;
TK04;     ;     ;RaE ;
```



BPMNSemantifying tool OUTCOME

```
root@0736c3e84db2:~# more tkview6
TransactionKind ; View  ; Request Decision  ; Request                    ; Promise Decision   ; Promise ; Decline ; After Decline Decision ; Execute                    ; Declare    ; Decision Accept                          ; Accept ; Reject ; Evaluate Rejection  ; Stop
TK01            ; Custom;                   ; Pedido de reforço orçamental ;                   ;         ;         ;                       ;Validação e elaborar informação;  Arquivar  ;   Tomar conhecimento e elaborar registo  ;        ;        ;                     ;
TK02            ; Custom;                   ;   Pedido Validação          ;                   ;         ;         ;                       ;  Validar                  ;            ;                                            ; Aceita ;        ;                     ;
TK03            ; Custom;                   ;   Pedido Validação          ;                   ;         ;         ;                       ;  Validar                  ;            ;                                            ; Aceita ;        ;                     ;
TK04            ; Custom;                   ;   Pedido Validação          ;                   ;         ;         ;                       ;  Validar                  ;            ;                                            ; Aceita ;        ;                     ;
```

```
root@0736c3e84db2:~# more actors4
A01 - SOC Department ; Budget transfer solicitor.
A02 - SPFP                ;  Executor of the first validation and creator of a budget transfer proposal.
A03 - SPFP Coordinator  ;  Executor of the second validation of the budget transfer.
A04 - Financial Department Director          ;   Executor of the third validation of the budget transfer.
A05 - Governing Board Representative    ;   Executor of the fourth validation of the budget transfer.

root@0736c3e84db2:~# more tpt4
TK01;Soliciting budget change; A01 - SOC Department  ; A02 - SPFP ; PK01 ; [budget] has been changed
TK02;Validating second-level budget change; A02 - SPFP ; A03 - SPFP Coordinator; PK02 ; [budget change] has been validated by second level
TK03;Validating third-level budget change;A03 - SPFP Coordinator;A04 - Financial Department Director; PK03 ; [budget change] has been validated by third level
TK04;Validating fourth-level budget change;A04 - Financial Department Director;A05 - Governing Board Representative; PK04 ; [budget change] has been validated by fourth level

root@0736c3e84db2:~# more tkdepend4
    ; TK01; TK02;TK03;TK04
TK01;     ;     ;    ;
TK02; RaP ;     ;    ;
TK03;     ; RaE ;    ;
TK04;     ;     ;RaE ;
```

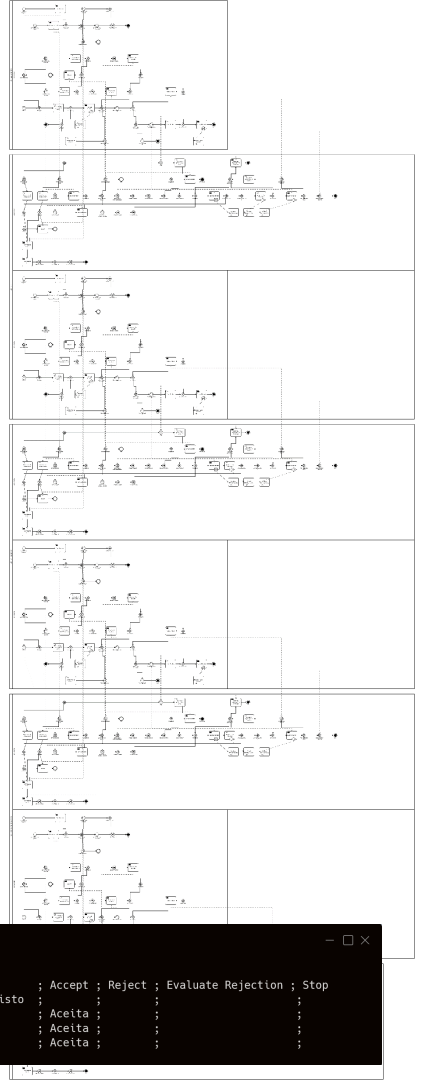
This BPMN XML file produced in some numbers:

8460 lines ♦ 26 send tasks ♦ 76 exclusive gateways ♦

14 parallel gateways ♦ 48 intermediateThrowEvent ♦

40 manualTasks ♦ 560 BPMN edges ♦ 68 messageFlows ♦

430 flowNodeRefs ...

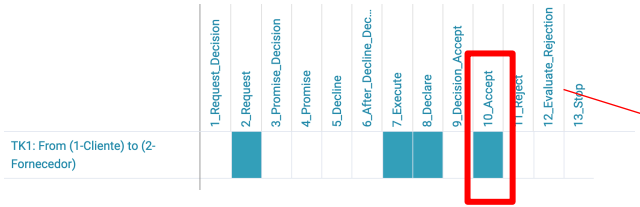


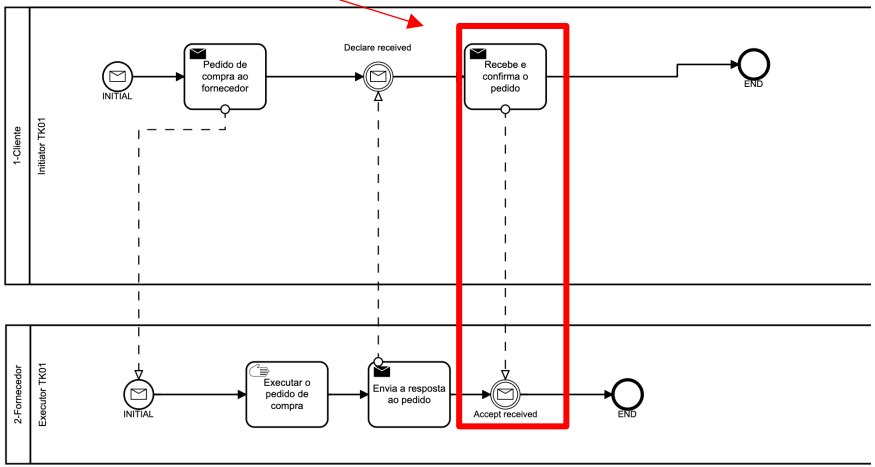
```
root@0736c3e84db2:~# more tkview6
TransactionKind ; View  ; Request Decision; Request                    ; Promise Decision   ; Promise ; Decline ; After Decline Decision ; Execute                    ; Declare    ; Decision Accept                    ; Accept ; Reject ; Evaluate Rejection ; Stop
TK01            ; Complete;               ; Pedido de reforço orçamental ;                  ;         ;         ;                       ;Validação e elaborar informação;  Arquivar  ;   Tomar conhecimento e elaborar registo ;        ;        ;                    ;
TK02            ; Complete;               ;   Pedido Validação          ;                  ;         ;         ;                       ;  Validar                   ;            ;                                       ; Aceita ;        ;                    ;
TK03            ; Complete;               ;   Pedido Validação          ;                  ;         ;         ;                       ;  Validar                   ;            ;                                       ; Aceita ;        ;                    ;
TK04            ; Complete;               ;   Pedido Validação          ;                  ;         ;         ;                       ;  Validar                   ;            ;                                       ; Aceita ;        ;                    ;
```

TÉCNICO LISBOA

Visualizing the patterns

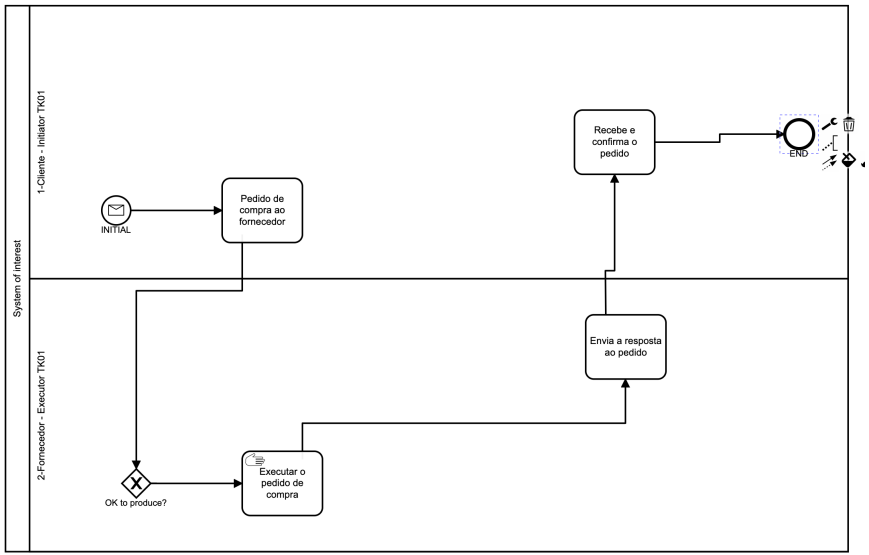
BPMN PRODUCED IN MULTIPLE POOLS



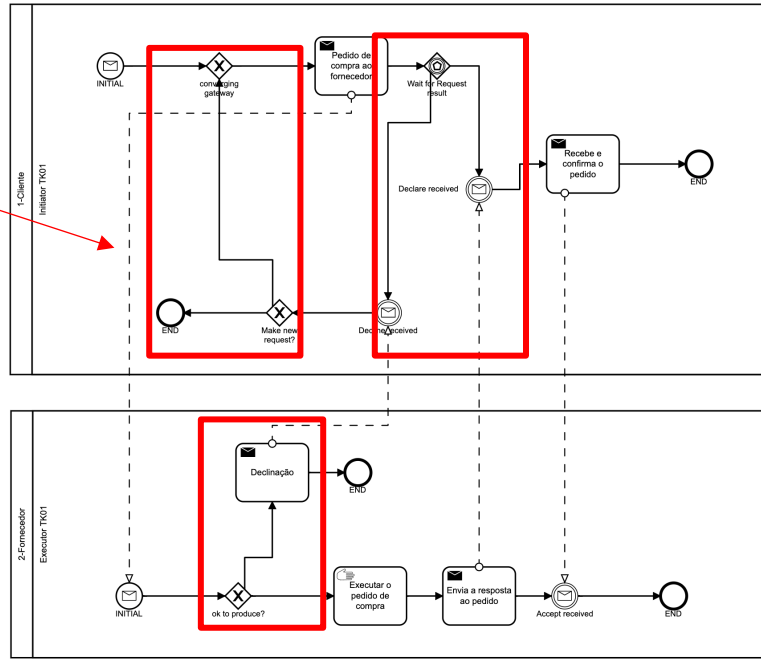


BPMN PRODUCED IN MULTIPLE LANES

	1_Request_Decision	2_Request	3_Promise_Decision	4_Promise	5_Decline	6_After_Decline_Dec...	7_Execute	8_Declare	9_Decision_Accept	10_Accept	11_Reject	12_Evaluate_Rejection	13_Stop
TK1: From (1-Cliente) to (2-Fornecedor)		■					■	■		■			

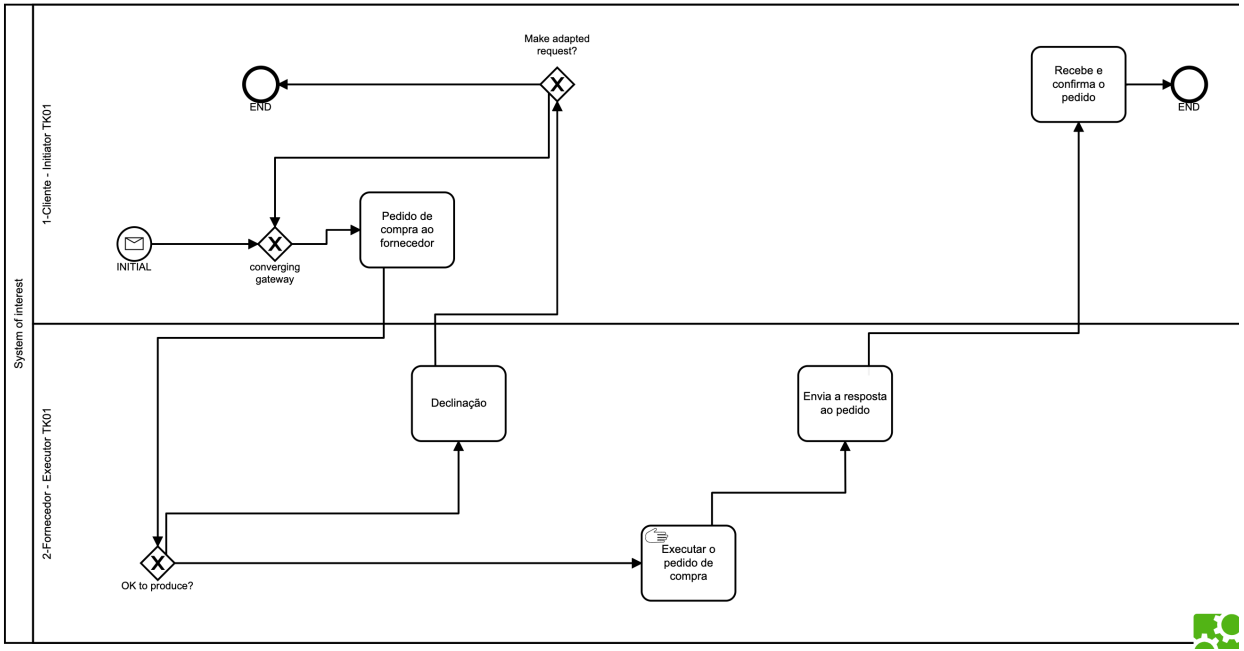


ADDING EXCEPTIONS: REQUEST DECLINATION USING POOLS

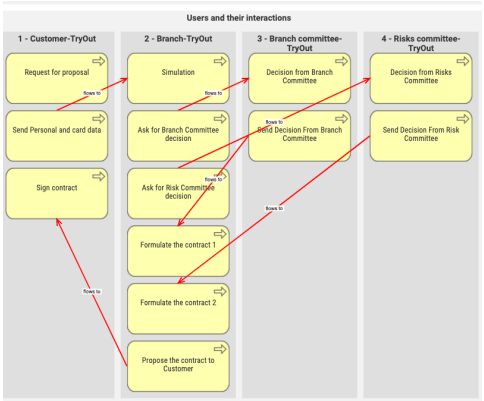


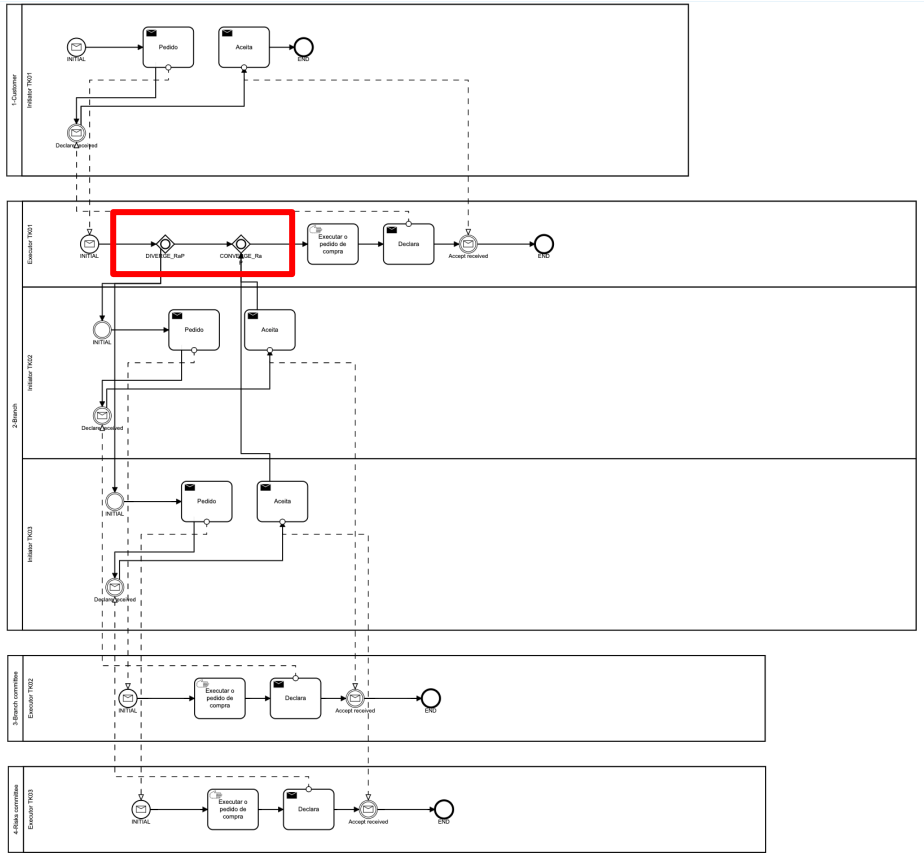
ADDING EXCEPTIONS: REQUEST DECLINATION USING LANES

	1_Request_Decision	2_Request	3_Promise_Decision	4_Promise	5_Decline	6_After_Decline_Dec...	7_Execute	8_Declare	9_Decision_Accept	10_Accept	11_Reject	12_Evaluate_Rejection	13_Stop
TK1: From (1-Cliente) to (2-Fornecedor)		■			■		■	■		■			



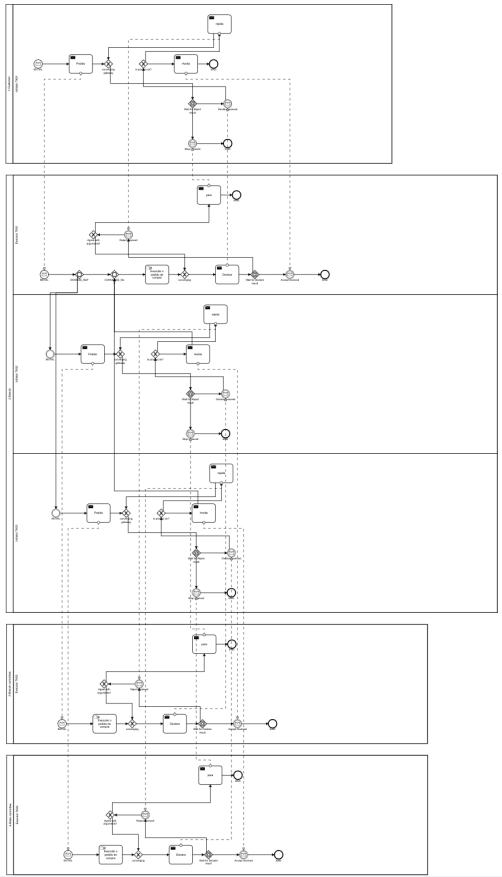
PRODUCING A NETWORK OF 4 ACTORS + 3 TKS USING POOLS





ADDING REJECTION TO THE NETWORK OF 4 ACTORS + 3 TKS USING POOLS

	1_Request_Decision	2_Request	3_Promise_Decision	4_Promise	5_Decline	6_After_Decline_Dec...	7_Execute	8_Declare	9_Decision_Accept	10_Accept	11_Reject	12_Evaluate_Rejection	13_Stop
TK1: From (1 - Customer-TryOut) to (2 - Branch-TryOut)		■					■	■		■	■		
TK2: From (2 - Branch-TryOut) to (3 - Branch committee-TryOut)		■					■	■		■	■		
TK3: From (2 - Branch-TryOut) to (4 - Risks committee-TryOut)		■					■	■		■	■		



TÉCNICO LISBOA

Q&A

TÉCNICO LISBOA