UNIVERSIDAD DE LA REPÚBLICA URUGUAY

FACULTAD DE INGENIERÍA

Minería de coreografías de procesos colaborativos en Gobierno Digital (e-Government)

Informe de Proyecto de Grado presentado por

Daniela Andrade

en cumplimiento parcial de los requerimientos para la graduación de la carrera de Ingeniería en Computación de Facultad de Ingeniería de la Universidad de la República

Supervisores

Andrea Delgado
Daniel Calegari

Montevideo, 19 de febrero de 2024

Resumen

Los procesos de negocio son un conjunto de actividades realizadas en coordinación en un entorno organizacional para alcanzar un objetivo de negocio. Un proceso colaborativo involucra dos o más organizaciones que actúan de forma coordinada para llevar a cabo distintas partes de un proceso. Para lograr esta coordinación intercambian mensajes entre las mismas, lo cual determina la coreografía del proceso.

La minería de procesos es una disciplina de la Ciencia de Datos que se basa en técnicas de minería de datos para analizar los registros (logs) de eventos asociados a la ejecución de procesos. La aplicación de minería de procesos sobre los datos de ejecución se han enfocado en procesos dentro de una sola organización, y no en procesos colaborativos, ni en sus coreografías.

El objetivo principal de este proyecto es la aplicación y/o definición de nuevas técnicas y algoritmos de minería de procesos para el descubrimiento y análisis de coreografías en procesos colaborativos en BPMN 2.0. Se propone una solución para descubrir el modelo de coreografía a partir de un registro de eventos, con aplicación a datos de un proceso real del Gobierno Digital en el contexto de AGESIC.

Palabras clave: Procesos colaborativos, minería de procesos, BPMN, ProM, coreografía

Índice general

1. Introducción **1**
- 1.1. Motivación y objetivos . . . 2
- 1.2. Aportes del proyecto . . . 2
- 1.3. Contexto de trabajo . . . 3
- 1.4. Estructura del documento . . . 3

2. Marco teórico **5**
- 2.1. Procesos de negocio . . . 5
  - 2.1.1. Coreografías en BPMN 2.0 . . . 8
- 2.2. Minería de procesos . . . 12
  - 2.2.1. Log de eventos . . . 13
  - 2.2.2. Herramientas . . . 17

3. Problemática y propuesta de solución **21**
- 3.1. Descripción del problema . . . 21
- 3.2. Relevamiento de requerimientos . . . 22
- 3.3. Propuesta de solución . . . 22
  - 3.3.1. Etapa 1 . . . 23
  - 3.3.2. Etapa 2 . . . 23
  - 3.3.3. Etapa 3 . . . 23
  - 3.3.4. Etapa 4 . . . 23

4. Desarrollo de prototipo **25**
- 4.1. Integración con ProM . . . 25
- 4.2. Diseño conceptual . . . 26
- 4.3. Lógica implementada . . . 27
  - 4.3.1. Template para el desarrollo de plug-ins . . . 27
  - 4.3.2. Detalle del desarrollo . . . 30

5. Casos de aplicación **37**
- 5.1. Caso de estudio: Solicitud de pasaporte . . . 37
  - 5.1.1. Descripción del caso de estudio . . . 37
  - 5.1.2. Aplicación del caso de estudio . . . 39
  - 5.1.3. Evaluación del resultado . . . 43

5.2. Casos de estudio adicionales 50
5.2.1. Caso 1 53
5.2.2. Caso 2 53
5.2.3. Caso 3 54
5.2.4. Caso 4 54

6. Conclusiones y Trabajo Futuro **57**

Referencias **59**

Capítulo 1

Introducción

Los procesos de negocio (Weske, 2019) son un conjunto de actividades realizadas en coordinación en un entorno organizacional para alcanzar un objetivo de negocio. Un proceso colaborativo involucra dos o más organizaciones que actúan de forma coordinada para llevar a cabo distintas partes de un proceso. Para lograr esta coordinación intercambian mensajes entre las mismas, lo cual determina la coreografía del proceso. Los procesos colaborativos tienen diversos desafíos tanto para su implementación como para el análisis de su ejecución. En particular, las trazas de ejecución de los procesos colaborativos se encuentran registradas en diferentes organizaciones con modelos de datos y tecnologías diferentes complejizando la recolección de los datos.

La minería de procesos (van der Aalst, 2016b) es una disciplina de la Ciencia de Datos que se basa en técnicas de minería de datos para analizar los registros (logs) de eventos asociados a la ejecución de procesos. Esto permite descubrir modelos de procesos a partir de los eventos, chequear la conformidad de una traza de eventos en un modelo determinado, esto es, qué tanto respeta una ejecución el modelo predefinido. Obtener medidas de ejecución de los procesos como su duración, cuellos de botella o la subutilización de recursos. La aplicación de minería de procesos sobre los datos de ejecución de procesos de negocio se han enfocado en procesos dentro de una sola organización, y no en procesos colaborativos, ni en sus coreografías. Además, los modelos descubiertos por las implementaciones existentes no incluyen en su visualización los participantes (organizaciones) o roles asociados, lo cual limita la comprensión y utilidad de los modelos descubiertos.

El objetivo principal de este proyecto es la aplicación y/o definición de nuevas técnicas y algoritmos de minería de procesos para el descubrimiento y análisis de coreografías en procesos colaborativos en BPMN 2.0.

Se propone una solución para descubrir el modelo de coreografía a partir de un registro de eventos con datos reales de un proceso del Gobierno Digital en el contexto de AGESIC. Además, se evalúa la solución con logs de eventos de procesos colaborativos de una propuesta existente. Se desarrolla la solución propuesta conformada por un plug-in que se integra en la plataforma ProM, el

cual es un entorno Java principalmente de uso académico para tareas de minería de procesos.

1.1. Motivación y objetivos

La motivación de este proyecto es la no existencia de propuestas que permitan descubrir modelos de coreografías de procesos de negocio colaborativos.

Es por esta razón, que el objetivo general de este proyecto es **desarrollar o extender una herramienta que permita el descubrimiento de coreografías en procesos colaborativos a partir de la información almacenada en los registros de eventos de diferentes organizaciones**.

Como objetivos específicos se tienen:

- (OE1) Estudiar conceptos de BPM, Minería de Procesos, configuración y ejecución de procesos, lenguajes BPMN 2.0, Petri Nets y herramientas (plataformas BPMS, ProM, etc.)
- (OE2) Analizar técnicas, algoritmos, herramientas y propuestas existentes para minería de coreografías en procesos colaborativos
- (OE3) Generar propuesta/extensión de minería de procesos para el descubrimiento de coreografías en procesos colaborativos y su visualización en BPMN 2.0
- (OE4) Desarrollar/extender/adaptar herramienta de soporte a la propuesta
- (OE5) Aplicar la propuesta a datos reales y soporte extendido de herramientas

1.2. Aportes del proyecto

- Un enfoque o propuesta y herramienta de soporte al descubrimiento de modelos de coreografías en procesos de negocio colaborativos a partir de un registro de eventos en BPMN 2.0.
- Conjunto de casos de estudio proveniente de datos reales de un proceso de negocio del Gobierno Digital y con registros de eventos de procesos colaborativos de una propuesta existente con los que se probó la herramienta.
- Repositorio en gitlab (Andrade D., Delgado A., Calegari D., 2023) con el código fuente de la solución desarrollada.
- Publicación científica (Peña, Andrade, Delgado, y Calegari, 2023) en la que se resume la solución presentada para el workshop Collaboration Mining for Distributed Systems 2023 (COMINDS) de la International Conference on Process Mining 2023 (ICPM).

1.3. Contexto de trabajo

Este proyecto se enmarca en el proyecto de investigación "Minería de procesos y datos para la mejora de procesos colaborativos aplicada a e-Government" del programa del Fondo María Viñas (FMV) de la ANII, en ejecución por el grupo COAL del INCO.

1.4. Estructura del documento

A continuación de detalla como está organizado el resto del informe:

- Capítulo 2 - Marco teórico: describe de forma introductoria los conceptos de procesos colaborativos, minería de procesos y coreografías.
- Capítulo 3 - Problemática y propuesta de solución: describe la problemática que se quiere resolver y se plantea una posible solución.
- Capítulo 4 - Desarrollo de prototipo: se detalla la solución implementada.
- Capítulo 5 - Casos de aplicación: se evalúa la solución desarrollada con diferentes casos de aplicación.
- Capítulo 6 - Conclusiones y trabajo futuro: se plantean las conclusiones del proyecto y los posibles trabajos a futuro.

Capítulo 2

Marco teórico

En este capítulo se resume el marco teórico del proyecto, incluyendo los conceptos de procesos de negocio, procesos de negocio colaborativos, minería de procesos, log de eventos, coreografía de procesos y también se presenta la herramienta ProM.

2.1. Procesos de negocio

Un proceso de negocio se puede definir como un conjunto de actividades realizadas en coordinación en un entorno organizacional y técnico, para alcanzar un objetivo del negocio. Los procesos de negocio (PNs) tienen un ciclo de vida que consta de las siguientes etapas (ver Figura 2.1) (Weske, 2019):

- Diseño y Análisis, donde se modelan, validan y verifican los PNs.
- Configuración, en la cual se implementa, integra, implanta y testea el sistema.
- Ejecución, donde se ejecuta y monitorea los PNs.
- Evaluación, aquí se analizan resultados de medición de las ejecuciones.

Para poder gestionarlos se utilizan conceptos, métodos y técnicas que apoyan el diseño, la administración, la configuración, la ejecución y el análisis de los mismos. Todas estas actividades involucran al personal de la organización, los cuales pueden utilizar sistemas de software que facilitan la ejecución de los PNs. Un Business Process Management System (BPMS) es un sistema de software genérico guiado por representaciones explícitas de procesos para coordinar la ejecución de procesos de negocio (Weske, 2019).

El Business Process Modeling and Notation (BPMN) (OMG, 2011) es una notación gráfica que permite describir un PN. En la Figura 2.2 se muestra un ejemplo de un modelo de proceso donde el mismo comienza al momento de realizar un pedido de compra, donde luego en simultaneo se procesa la recepción

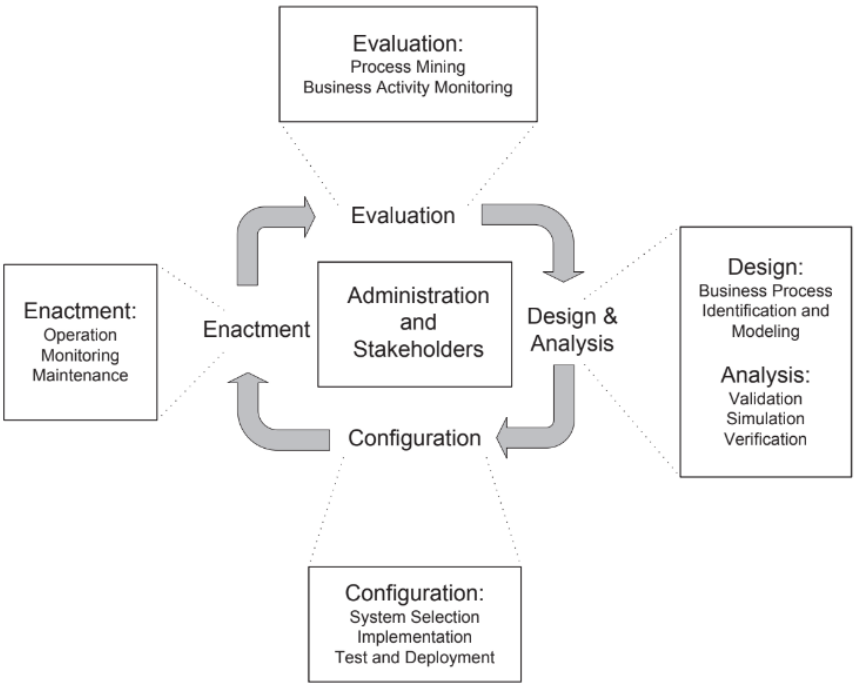


Figura 2.1: Ciclo de vida de un proceso de negocio (Weske, 2019)

de la factura y su pago, y por otro lado se recibe el producto. Recién cuando se completan ambas actividades finaliza el proceso de compra.

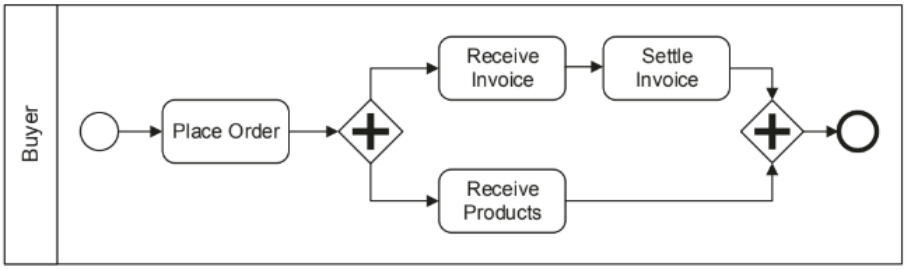


Figura 2.2: Ejemplo de modelo de proceso de pedido de compra (Weske, 2019)

Los modelos de procesos muestras el flujo de actividades que se realizan dentro de una organización para alcanzar un objetivo. Este proceso también puede involucrar actividades realizadas por otra organización dentro de otro proceso de negocio. Si en el ejemplo anterior, al momento de realizar el pedido de compra se le envía un mensaje al vendedor, el cual comienza a realizar la orden de compra, con esto le envía la factura y el producto al comprador (de forma

paralela) y luego de que el comprador recibe tanto la factura como el producto, realiza el pago de la factura el vendedor finaliza su proceso de negocio. En la Figura 2.3 se muestra la situación descripta (Weske, 2019).

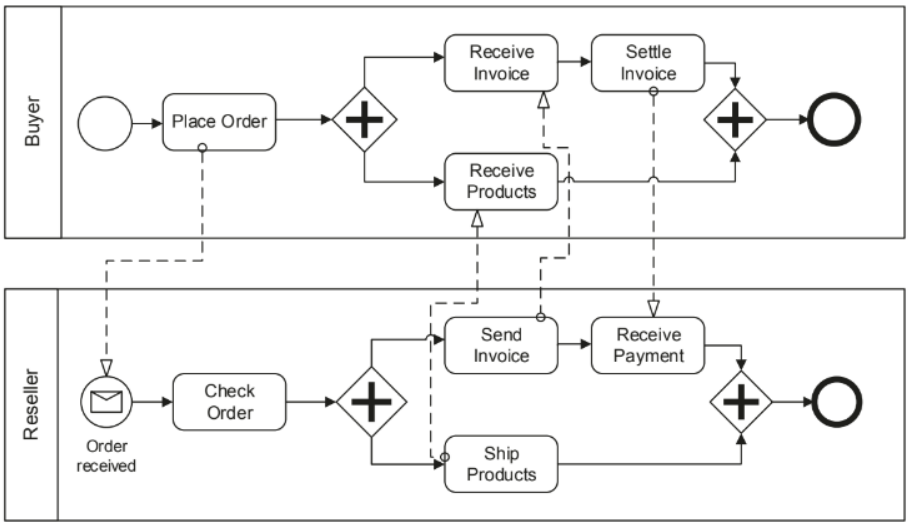


Figura 2.3: Ejemplo de modelo de proceso colaborativo (Weske, 2019)

La situación que se muestra en la Figura 2.3 se denomina proceso colaborativo, ya que participan dos (pueden ser más) organizaciones para llegar a un objetivo común donde a su vez, cada una de ella tiene su propio proceso de negocio. Existen cinco formas de interactuar (van der Aalst, 2016a):

- Capacidad compartida, el control del proceso de negocio está en una organización pero las tareas se distribuyen entre el resto de las organizaciones.
- Ejecución en cadena, las tareas se realizan de forma secuencial entre las organizaciones donde una organización activa la ejecución de tareas en otra.
- Subcontratación, una organización subcontrata subprocesos a otras organizaciones.
- Transferencia de casos, las organizaciones tienen el mismo proceso de negocio y se distribuye la carga de trabajo entre las mismas.
- Débilmente acoplado, cada organización tiene su proceso de negocio donde sólo conocen el protocolo que utilizan para comunicarse.

Continuando con el ejemplo de la Figura 2.3, cuando se tiene un proceso colaborativo, existe una interacción entre las organizaciones la cual es enviando y recibiendo mensajes. A este intercambio de mensajes se le denomina coreografía de procesos. Para realizar una interacción correcta, las organizaciones

deben acordar una coreografía común antes de interactuar (Weske, 2019). En la próxima sección se da una descripción más detallada sobre esto.

2.1.1. Coreografías en BPMN 2.0

BPMN (*Business Process Model and Notation (BPMN)*, 2011) proporciona una serie de notaciones que permiten la visualización de forma gráfica de los procesos de negocio, así como también permite la portabilidad de los modelos de procesos con sus definiciones entre diferentes herramientas.

BPMN 2.0 extiende a BPMN donde una de las extensiones es la definición del modelo de coreografía.

BPMN 2.0 tiene dos especificaciones:

- Semántica, se definen los elementos BPMN que participan en el proceso de negocio y la relación que hay entre los mismos. En el Listing 2.1 se muestra una parte de un ejemplo de la parte semántica en donde desde el startEvent sale un sequenceFlow con identificador element5-element7 (flecha) que une el startEvent con el elemento task de nombre Activity A. Luego, Activity A tiene como elemento entrante un sequenceFlow descripto antes y como elemento saliente un sequenceFlow que une la Activity A con el parallelGateway.

```
....
<task id="element7" name="Activity A">
  <incoming>element5-element7</incoming>
  <outgoing>element7-element8</outgoing>
</task>
<startEvent id="element5" name="Start">
  <outgoing>element5-element7</outgoing>
</startEvent>
<parallelGateway id="element8" name="">
  <incoming>element7-element8</incoming>
  <outgoing>Flow_09xrdfq</outgoing>
  <outgoing>Flow_0116fjb</outgoing>
</parallelGateway>
<sequenceFlow id="element5-element7" name="" isImmediate="true" sourceRef="
    element5" targetRef="element7" />
<sequenceFlow id="element7-element8" name="" isImmediate="true" sourceRef="
    element7" targetRef="element8" />
....
```

Listing 2.1: Especificación semántica en BPMN 2.0 (*Colliery Validation*, s.f.-a)

- Diagrama (BPMNDI), esta especificación requiere de la especificación semántica ya que establece la posición y el tamaño de los elementos de la parte semántica en el plano. En el Listing 2.2 se muestra una parte de un ejemplo de BPMNDI correspondiente a los elemento mostrados en el ejemplo Listing 2.1. En el ejemplo se puede ver que se definen dos tipos de elementos: BPMNShape y BPMNEdge. BPMNEdge se corresponde con los elementos que son del tipo sequenceFlow, mientras que el resto de los

elementos se corresponden con BPMNShape. Estos dos tipos de elementos están contenidos dentro de un BPMNPlane, el cual es una colección ordenada de los elementos BPMNShape y BPMNEdge.

```
....
....
<bpmndi:BPMNShape id="element7_gui" bpmnElement="element7">
    <omgdc:Bounds x="292" y="180" width="70" height="30" />
    <bpmndi:BPMNLabel labelStyle="sid-da45f934-3c3c-438b-bfa2-f8f62914fb67"
        >
      <omgdc:Bounds x="89.54285430908203" y="37" width="50.91429138183594"
          height="12" />
    </bpmndi:BPMNLabel>
</bpmndi:BPMNShape>
<bpmndi:BPMNShape id="element5_gui" bpmnElement="element5">
    <omgdc:Bounds x="212" y="180" width="30" height="30" />
    <bpmndi:BPMNLabel labelStyle="sid-5b971192-0a62-4aa8-acd5-7ad65e64ae6b"
        >
      <omgdc:Bounds x="216" y="212" width="24" height="14" />
    </bpmndi:BPMNLabel>
</bpmndi:BPMNShape>
<bpmndi:BPMNShape id="Gateway_144vk4m_di" bpmnElement="element8">
    <omgdc:Bounds x="407" y="175" width="40" height="40" />
</bpmndi:BPMNShape>
<bpmndi:BPMNEdge id="element5-element7_gui" bpmnElement="element5-element7"
     >
    <omgdi:waypoint x="242" y="195" />
    <omgdi:waypoint x="292" y="195" />
</bpmndi:BPMNEdge>
<bpmndi:BPMNEdge id="element7-element8_gui" bpmnElement="element7-element8"
     >
    <omgdi:waypoint x="362" y="195" />
    <omgdi:waypoint x="407" y="195" />
</bpmndi:BPMNEdge>
....
....
```

Listing 2.2: Especificación BPMNDI en BPMN 2.0 (*Colliery Validation*, s.f.-a)

Las coreografías son usadas en BPMN para representar el intercambio de información entre los participantes en un proceso de negocio colaborativo, donde los participantes se corresponden con las organizaciones intervinientes. La coreografía sólo existe si hay dos o más participantes. Lo que interesa en este contexto, son los mensajes intercambiados entre los participantes donde teniendo en cuenta el sentido de la comunicación se puede determinar el participante emisor y el participante receptor. Al mensaje intercambiado se le denomina Tareas de Coreografía (Choreography Task) que involucran al participante emisor, los participantes receptores y al mensaje intercambiado. En la Figura 2.4 se muestra un ejemplo de un modelo de coreografía para tener una visualización de lo que se describe. En el ejemplo, se tienen tres participantes: Comercio, Proveedor y Proveedor externo. Siguiendo las enumeraciones que se remarcan en rojo:

1. Event, es algo que sucede durante el curso de un proceso o coreografía que

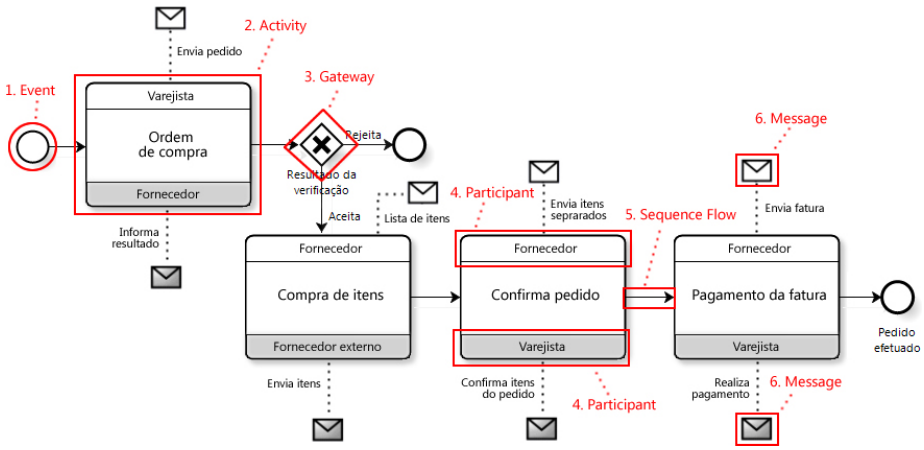


Figura 2.4: Ejemplo de modelo de coreografía en BPMN 2.0 (*iProcess*, s.f.)

afecta el flujo del mismo. Existen tres tipos de eventos: Start, Intermediate y End. De los cuales para la solución sólo se utilizan Start y End debido a que el BPMN Miner genera un diagrama BPMN a partir de Petri Nets y estas no detectan eventos intermedios. A continuación se detallan los eventos utilizados:

- Start Event: indican gráficamente el inicio de una coreografía.
- End Event: indican gráficamente el final de una coreografía.

2. Activity (Choreography Activity), representa la interacción entre dos o más participantes. En un mismo elemento se representa el mensaje intercambiado, el participante emisor y los participantes receptores.

3. Gateway, son utilizados para controlar el flujo de un proceso o coreografía. Existen cinco tipos de gateway: Exlusive, Parallel, Event-Based, Inclusive y Complex. De estos, los que se utilizan en la solución, debido a la misma razón que los eventos, son:

   - Exclusive, son usados para crear caminos alternativos mediante una expresión de condición. Es el que se puede ver en la Figura 2.4.
   - Parallel, son utilizados cuando se quiere tener caminos paralelos y que ejecuten al mismo tiempo. No hay una condición de que camino seguir en el gateway. En la Figura 2.5 se muestra un ejemplo.

4. Participant, representa las organizaciones que participan en el intercambio de mensajes. El participante emisor se muestra con el fondo blanco y los participantes receptores se muestras con fondo en color gris.

5. Sequence Flow, son usadas para mostrar el orden de los elementos BPMN en un diagrama de coreografía. Cada Sequence Flow tiene un único origen

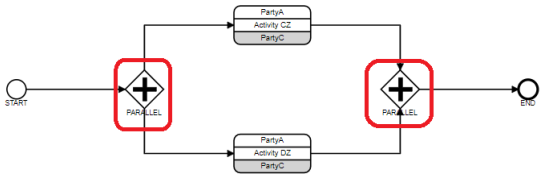


Figura 2.5: Ejemplo de Parallel Gateway

y un único destino. Tanto el origen como el destino pueden ser Events, Activities, Choreography Activities o Gateways.

6. Message, representa el mensaje que se pasa de un participante a otro.

En el ejemplo, la comunicación inicia cuando el Comercio le solicita una orden de compra al Proveedor, comenzando el diagrama de coreografía con un Start Event. El proveedor puede aceptar o rechazar el pedido, en caso de rechazar finaliza el diagrama teniendo un Sequence Flow a un End Event. En caso de aceptar, el Proveedor le solicita al Proveedor externo la compra. Luego el Proveedor le confirma el pedido al Comercio y por último, le solicita el pago. Finalizando así el diagrama de coreografía con el End Event.

A continuación se detallan algunas clases que conforman el metamodelo de BPMN 2.0 que son de interés en este proyecto que se muestran en la Figura 2.6. La clase BaseElement es la clase más abstracta de todos los elementos BPMN. Provee el atributo Id que permite la identificación única de los elementos BPMN y documentación sobre los elementos que heredan de la misma. Definitions es la clase más externa que contiene todos los elementos BPMN. Especifica la visibilidad y el espacio de nombres para todos los elementos. A su vez, Definitions está compuesta por un conjunto de RootElement, que también hereda de la clase BaseElement. RootElement es la super clase abstracta de todos los elementos BPMN que se contienen dentro de Definitions, donde Collaboration, Process y Choreography son clases concretas de ésta.

Por otra parte, se tiene la clase Choreography Activity que se corresponde con el diagrama de Coreografía, que hereda de la clase FlowElement. A su vez, Choreography Activity tiene asociada un conjunto de participantes donde se determina el participante que inicia la interacción y el que la recibe. Las Choreography Activity representan la interacción entre dos o más participantes. Además, se tiene la clase Choreography Task que hereda de Choreoagraphy Activity, donde la Choreography Task es la interacción entre dos participantes y se representa como un único objeto. Las clases Event y Gateway heredan también de FlowElement y son clases que son comunes a los tres tipos de modelo (Proceso, Colaboración y Coreografía). Por último, se tienen las Task que son una clase que heredan de la clase Activity que representan las actividades dentro de un proceso de negocio. Son utilizadas en el modelo de Proceso y de Colaboración. Y también heredan de la clase FlowElement.

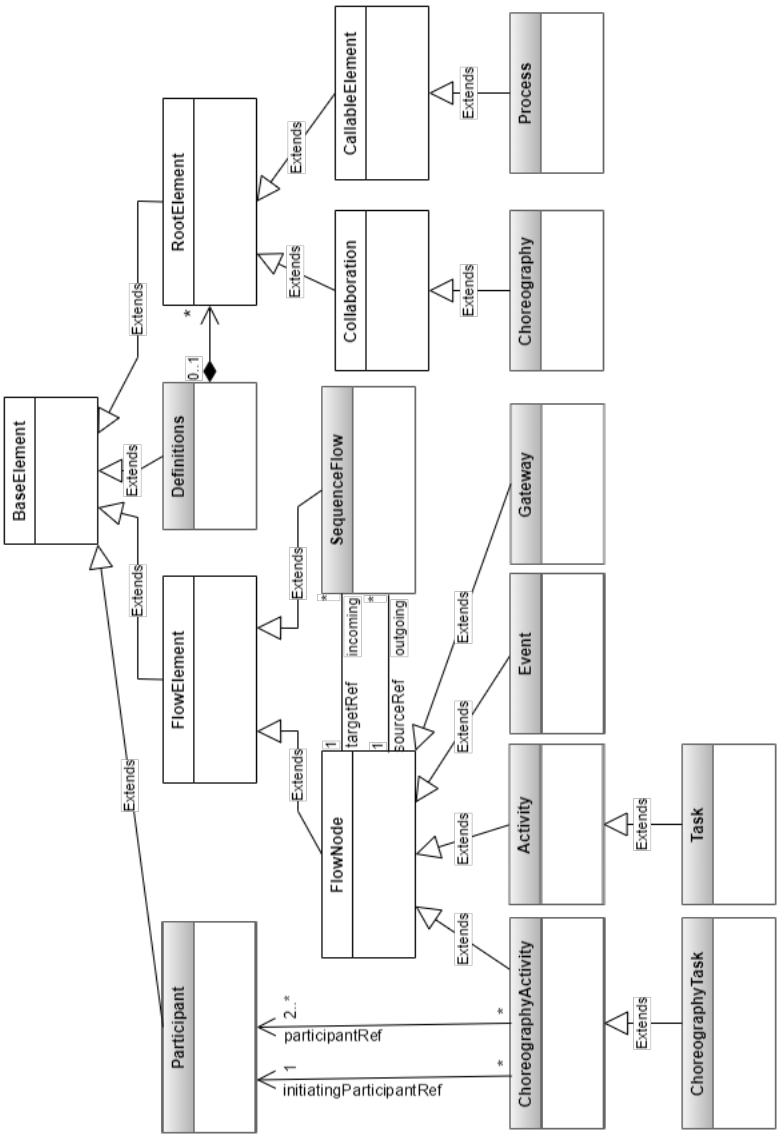


Figura 2.6: Parte del metamodelo de BPMN 2.0 (*Business Process Model and Notation (BPMN)*, 2011)

2.2. Minería de procesos

La minería de procesos es una disciplina que se encuentra entre la Ciencia de Datos y la Ciencia de Procesos. La finalidad de esta disciplina es descubrir, chequear la conformidad y mejorar los procesos de negocio mediante la extracción de la información que se obtiene de los registros de eventos de los sistemas

involucrados (van der Aalst, 2016a).

La minería de procesos permite cerrar el ciclo de vida de un proceso de negocio, los datos que se registran en los sistemas de información pueden ser utilizados para mejorar el proceso de negocio haciendo un análisis de los datos para evaluar posibles desviaciones y en tal caso, mejorar el modelo de negocio (van der Aalst, 2016a).

Se identifican tres tipos de minería de procesos (van der Aalst, 2016a):

- Descubrimiento de modelos, a partir de un registro de eventos (registros de datos de ejecuciones), se produce un modelo de proceso de negocio (por ejemplo BPMN, redes de Petri) sin usar ninguna otra información adicional.
- Conformidad de modelos, dado un modelo de proceso y un registro de eventos con ejecuciones del mismo proceso, se chequea o comprueba que el modelo se ajusta a la realidad registrada en el registro de eventos y viceversa. La comprobación puede utilizarse para detectar y localizar desviaciones, y para medir la gravedad de las mismas.
- Extensión de modelos, aquí la idea es extender o mejorar un modelo de proceso existente utilizando información sobre el proceso real registrado en el registro de eventos teniendo en cuenta por ejemplo la duración, cuellos de botella, recursos involucrados, entre otros.

2.2.1. Log de eventos

En la minería de procesos, el registro de eventos forma un papel importante ya que permite analizar el comportamiento de los sistemas informáticos en tiempo de ejecución. Para esto, se desarrolló un formato de registro de eventos denominado XES (eXtensible Event Stream) el cual permite almacenar registros de eventos de diferentes sistemas de información.

En el Listing 2.3 se muestra un ejemplo de XES tomado del caso de estudio de solicitud de pasaporte.

```
<log xmlns="http://your_namespace"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://your_namespace"
       xes.version="1.0" xes.features="nested-attributes" >
       <extension name="Organizational" prefix="org" uri="http://www.xes-
            standard.org/org.xesext"/>
       <extension name="Time" prefix="time" uri="http://www.xes-standard.org/
            time.xesext"/>
       <extension name="Concept" prefix="concept" uri="http://www.xes-standard.
            org/concept.xesext"/>

       <string key="collab:processType" value="Choreography" />
       <trace>
              <string key="concept:name" value="Trace 0" />
              <event>
                     <string key="concept:name" value="Get available dates" />
                     <date key="time:timestamp" value="2020-02-05T06
                          :50:54.000-03:00" />
                     <string key="lifecycle:transition" value="complete" />
                     <string key="org:resource" value="Miguel Gonzalez" />
                     <string key="org:role" value="User Support" />
              </event>
              <event>
                     <string key="concept:name" value="Confirm appointment" />
                     <date key="time:timestamp" value="2020-02-06T17
                          :56:55.000-03:00" />
                     <string key="lifecycle:transition" value="complete" />
                     <string key="org:resource" value="Pedro Garcia" />
                     <string key="org:role" value="Coordination" />
              </event>
       </trace>
</log>
```

Listing 2.3: Ejemplo XES - Solicitud de pasaporte (caso de estudio)

Las etiquetas Log, Trace y Event solo definen la estructura del documento, no contienen ninguna información. Tanto como los Log, los Trace y los Event contienen atributos. Los atributos tienen una clave, un tipo (String, Date, Int, Float o Boolean) y un valor que se corresponde con el tipo. A su vez, los atributos pueden tener atributos en sí mismos para proporcionar información adicional. Los atributos pueden estar definidos por una extensión donde las más utilizadas se detallan en la Tabla 2.1. A su vez, en el Log se pueden definir clasificadores que estos definen atributos, esto es útil para darle una identidad a los atributos que se especifican y poder comparar los eventos entre sí.

En la Figura 2.7 se muestra el metamodelo de XES donde se puede ver la correspondencia con lo antes explicado. La etiqueta Log contiene etiquetas de Trace, Extension, Classifier y Attribute. El Trace contiene etiquetas de Event y Attribute. El Event contiene etiqueta de Attribute. Luego, la etiqueta Extension contiene un name, prefix, URI y Attribute. Y los Classifier contienen Attribute. Los Attribute pueden contener Attribute en sí mismos, una key y pueden ser de diferentes tipos (String, Date, Int, Float, Boolean) con sus respectivos valores.

Como se pudo ver anteriormente, los XES tienen un formato particular que los sistemas de almacenamiento de la información pueden no tener dicho forma-

Tabla 2.1: Extensiones para XES

Extension	Key	Level	Type	Description
concept	name	log, trace, event	string	Nombre descriptivo.
lifecycle	transition	event	string	Indica la transición del ciclo de vida que se encuentra cada evento.
organizational	resource	event	string	Especifica el nombre o identificador de quien originó evento.
organizational	role	event	string	Especifica el rol de quien originó evento a nivel de estructura organizacional.
time	timestamp	event	date	Especifica la fecha y la hora en que ocurrió el evento.

to, por lo que se proporciona la herramienta XESame que permite extraer los registros de eventos de diversas fuentes de datos sin necesidad de tener conocimientos de programación. La transformación de los datos al registro de eventos se puede realizar con la interfaz que proporciona la herramienta (Verbeek et al., 2011).

Ahora, para poder trabajar con registros de eventos que se correspondan con un registro de eventos de una coreografía, es necesario agregar la extensión (González y Delgado, 2021) que se muestra en la Tabla 2.2 como adicional a las extensiones mostradas en la Tabla 2.1. Esto, permite agregar las etiquetas correspondientes a los participantes que están interactuando en el evento, donde se agrega la etiqueta de fromParticipant, toParticipant y elemType del tipo Message.

Por lo tanto, si agregamos al ejemplo anterior dicha extensión tendríamos un log de eventos como se muestra en Listing 2.4. Se agregan las etiquetas de elemType, fromParticipant y toParticipant. Donde el participante AGESIC le solicita a DNIC las fechas disponibles para así, finalmente confirmar la cita por parte de AGESIC.

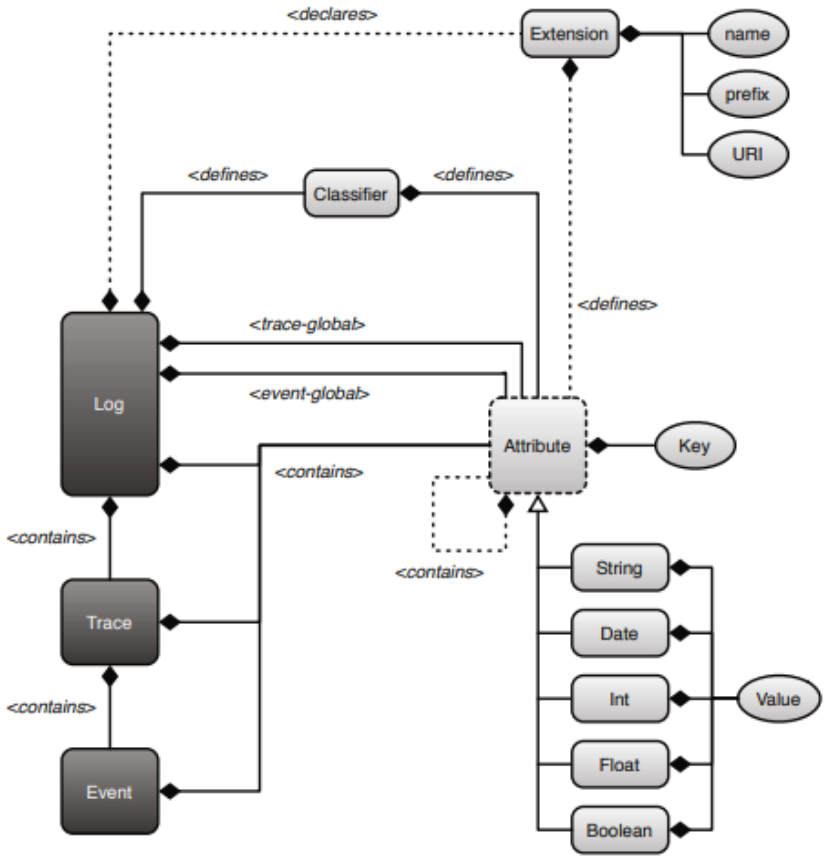


Figura 2.7: Metamodelo de XES (Verbeek et al., 2011)

Tabla 2.2: Extensión para coreografía

Extension	Key	Level	Type	Description
collab	elemType	event	string	Especifica el tipo de evento.
collab	fromParticipant	event	string	Especifica el participante que inicio la interacción.
collab	toParticipant	event	string	Especifica el participante que recibe la interacción.

```
<log xmlns="http://your_namespace"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://your_namespace"
        xes.version="1.0" xes.features="nested-attributes" >
        <extension name="Organizational" prefix="org" uri="http://www.xes-standard.
            org/org.xesext"/>
        <extension name="Time" prefix="time" uri="http://www.xes-standard.org/time.
            xesext"/>
        <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org
            /concept.xesext"/>
        <extension name="Collaboration" prefix="collab" uri="http://www.xes-standard
            .org/collab.xesext"
        />
        <string key="collab:processType" value="Choreography" />
        <trace>
                <string key="concept:name" value="Trace 0" />
                <event>
                        <string key="concept:name" value="Get available dates" />
                        <date key="time:timestamp" value="2020-02-05T06
                            :50:54.000-03:00" />
                        <string key="lifecycle:transition" value="complete" />
                        <string key="org:resource" value="Miguel Gonzalez" />
                        <string key="org:role" value="User Support" />
                        <string key="collab:elemType" value="Message" />
                        <string key="collab:fromParticipant" value="AGESIC" />
                        <string key="collab:toParticipant" value="DNIC" />
                </event>
                <event>
                        <string key="concept:name" value="Confirm appointment" />
                        <date key="time:timestamp" value="2020-02-06T17
                            :56:55.000-03:00" />
                        <string key="lifecycle:transition" value="complete" />
                        <string key="org:resource" value="Pedro Garcia" />
                        <string key="org:role" value="Coordination" />
                        <string key="collab:elemType" value="Message" />
                        <string key="collab:fromParticipant" value="AGESIC" />
                        <string key="collab:toParticipant" value="DNIC" />
                </event>
        </trace>
</log>
```

Listing 2.4: Ejemplo XES - Solicitud de pasaporte (caso de estudio)

2.2.2. Herramientas

ProM Tools

ProM es un Framework de minería de procesos de código abierto que permite procesar archivos XES mediante distintos plug-ins (Verbeek et al., 2011). En la Figura 2.8 se muestra un ejemplo donde se tiene cargado un XES donde en verde aparecen los posibles plug-ins que se pueden aplicar y los resultados del mismo. Mientras que en la Figura 2.9 se muestra el resultado.

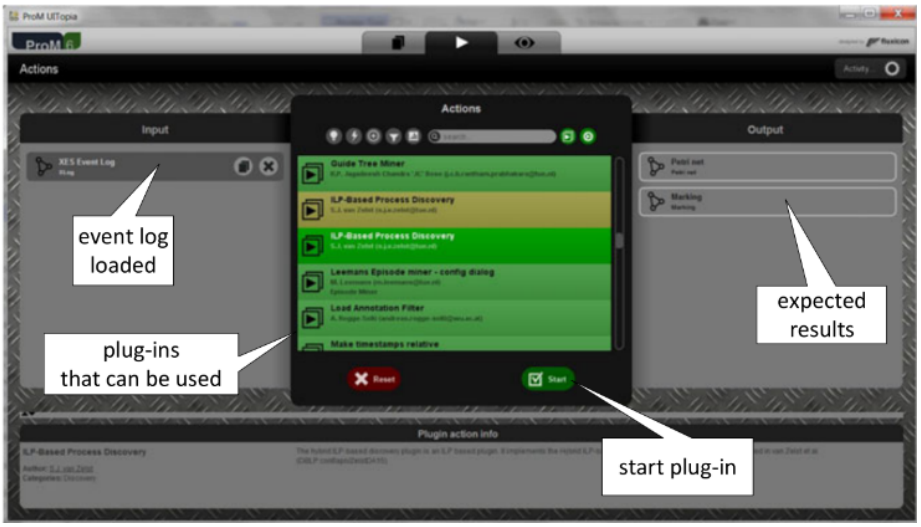


Figura 2.8: ProM: cargar y ejecutar un plug-in (van der Aalst, 2016b)

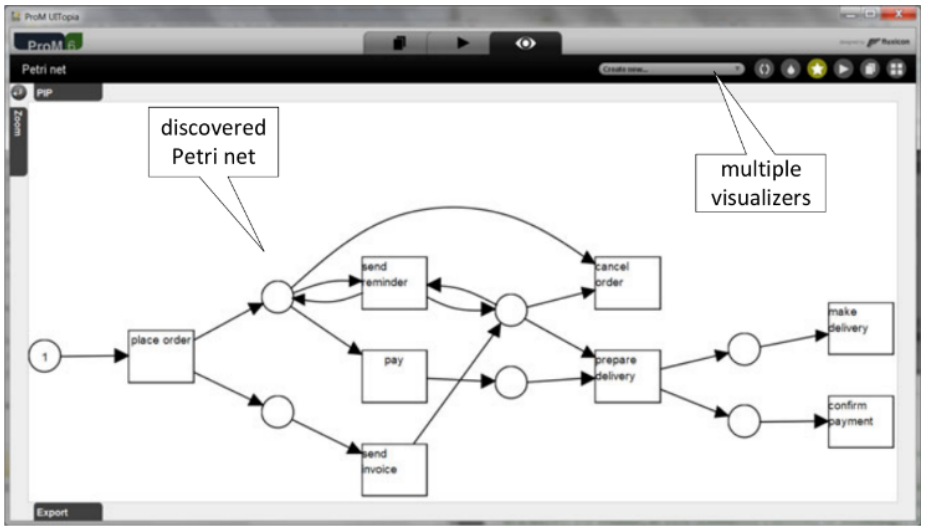


Figura 2.9: ProM: ejemplo de red de petri (van der Aalst, 2016b)

Entre los plug-ins integrados al Framework, se tiene el BPMN Miner que es una técnica de descubrimiento de procesos que genera modelos BPMN a partir de registros de eventos. Es una herramienta que permite descubrir modelos de procesos jerárquicos utilizando técnicas de descubrimiento de modelos de procesos existentes para generar el modelo plano correspondiente dentro de la jerarquía (Conforti, Dumas, García-Bañuelos, y Rosa, 2016) para luego trans-

formar este resultado en un modelo BPMN.

Antes de describir las técnicas de descubrimiento que se utilizan para generar el modelo, se define lo que es una red de Petri. Es el lenguaje de modelado de procesos representado como un grafo bipartito (esto es porque se arman caminos que son disjuntos) que consta de lugares y transiciones, donde se puede transitar por la misma mediante reglas de activación en los posibles caminos (van der Aalst, 2016b).

También, se definen los criterios de calidad para los modelos descubiertos y para cada uno de ellos se puede especificar que tanto están acorde a los mismos:

- Fitness, el modelo descubierto permite el comportamiento observado en el registro de eventos.
- Precision, el modelo descubierto no permite un comportamiento que no esté observado en el registro de eventos.
- Generalization, el modelo descubierto generaliza el comportamiento observado en el registro de eventos.
- Simplicity, el modelo descubierto debe ser lo màs simple posible.

Entonces, las técnicas de descubrimiento de modelos de procesos planos que se utilizan para generar el modelo BPMN son (Conforti et al., 2016):

- Inductive Miner (van der Aalst, 2016b), divide el registro de eventos de forma recursiva formando subgrupos. Utiliza la técnica de dividir y conquistar para detectar las relaciones en el registro de eventos y devuelve una red de Petri con el resultado. Es capaz de detectar opciones, concurrencia y bucles. Este algoritmo permite generar modelos con un alto fitness (comentado en la sección 2.2).
- Heuristics Miner ProM6, identifica las dependencias entre los registros de eventos y su frecuencia. El algoritmo construye un gráfico en base a los valores antes mencionados donde se pueden aplicar umbrales que permiten que se tenga una mayor resistencia al ruido. Este gráfico permite construir una red de Petri como resultado. Genera buenos resultados en términos de precisión (comentado en la sección 2.2), donde la precisión mide el comportamiento adicional al registro de eventos permitido por el modelo.
- ILP Miner, genera una red de Petri representando las relaciones del registro de eventos en un problema de programación lineal entera. Genera buenos resultados en términos de precisión (comentado en la sección 2.2).
- Alpha Algorithm, construye redes de Petri identificando relaciones entre pares de eventos de registro. Es sensible al ruido (registros de eventos incorrectos o faltantes) y a al comportamiento poco frecuente, como tampoco puede manejar registros de eventos complejos. Tiene baja precisión (comentado en la sección 2.2).

Capítulo 3

Problemática y propuesta de solución

En este capítulo se presenta la problemática a resolver y la propuesta de solución.

3.1. Descripción del problema

La minería de procesos se ha enfocado principalmente en procesos de negocio llevados a cabo por una sola organización (intra-organizacionales) y no en procesos colaborativos donde participan dos o más organizaciones (inter-organizacionales), ni en sus coreografías. Las implementaciones existentes para el descubrimiento de modelos no incluyen el modelado de coreografías en el lenguaje de coreografía de BPMN 2.0 y tampoco se ha encontrado ninguna propuesta que permita obtener el modelo de coreografía en procesos colaborativos.

Por esta razón, se quiere contar con un enfoque o propuesta y herramienta que permita el descubrimiento de coreografías en procesos colaborativos y su visualización en el lenguaje BPMN 2.0 con el soporte de alguna herramienta existente. En la Figura 3.1 se presenta gráficamente la problemática a resolver, donde se recibe como dato de entrada un archivo XES extendido de coreografía y mediante alguna herramienta, se genere un archivo BPMN con el modelo de coreografía correspondiente al XES extendido. El XES extendido de coreografía del cual se parte incluye los datos de los mensajes intercambiados con sus correspondientes participantes (emisor y receptor). La información sobre los mensajes intercambiados son obtenidos de trabajos previos que se enfocan en obtener los datos desde las organizaciones e integrarlos (González y Delgado, 2021).

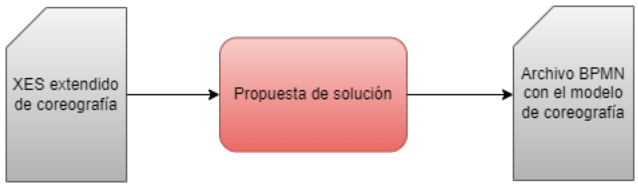


Figura 3.1: Representaicón de la problemática

3.2. Relevamiento de requerimientos

A continuación se listan los requerimientos solicitados para el presente proyecto.

- (R1) La herramienta debe estar incluida como un plug-in de ProM, por lo que debe ser implementada en la plataforma Java Empresarial (Java EE).
- (R2) La herramienta debe recibir como datos de entrada un archivo de formato XES extendido de coreografía del proceso de negocio que se quiere representar.
- (R3) La herramienta debe generar un archivo resultado con la especificación del diagrama de coreografía correspondiente al archivo XES extendido pasado como dato de entrada en lenguaje BPMN 2.0.
- (R4) La herramienta debe permitir descargar el archivo resultado de ProM para poder visualizar la coreografía en alguna herramienta existente que soporte los elementos de coreografía.

3.3. Propuesta de solución

Para poder resolver la problemática presentada, se propone como una posible solución reutilizar plug-ins existentes para resolver el problema de descubrimiento, en particular el BPMN Miner.

El punto de partida del plug-in será un archivo XES extendido que contiene la información del intercambio de mensaje entre varias organizaciones. Por lo que las trazas en el XES estarán conformadas por eventos que representan los mensajes intercambiados. La coreografía determina el comportamiento del flujo de estos mensajes, por lo que bajo este contexto, el problema se puede ver como el descubrimiento de un proceso de negocio. Esto nos permite reutilizar un algoritmo existente, donde el resultado del descubrimiento no estará en el formato de coreografía, con lo que será necesario trasformarlo en el formato esperado, respetando el proceso descubierto.

En la Figura 3.2 se muestra un diagrama de la propuesta de solución y a continuación se dará una breve explicación de cada etapa y el flujo entre los mismos.

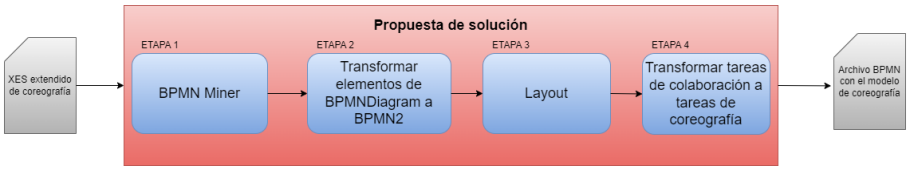


Figura 3.2: Propuesta de solución

3.3.1. Etapa 1

En esta primera etapa, se utiliza el plug-in BPMN Miner para el XES extendido de coreografía pasado como dato de entrada. El BPMN Miner permite seleccionar diferentes algoritmos que generan redes de Petri de diferente calidad, mencionados en la sección 2.2.2, para luego transformarlas en modelos BPMN.

3.3.2. Etapa 2

El metamodelo que utiliza el BPMN Miner para trabajar con elementos de BPMN no contiene los elementos de coreografía. Por lo que fue necesario transformar cada elemento del modelo descubierto en elementos pertenecientes al metamodelo de BPMN 2.0 y así, poder trabajar con tareas de coreografía en pasos posteriores. Esto deja como resultado un modelo de proceso en BPMN 2.0.

3.3.3. Etapa 3

Al resultado anterior, se le aplica el algoritmo BPMN Layout Generators (*BPMN Layout Generators*, 2020). El cual permite generar el diseño gráfico del proceso BPMN (BPMNDI), posicionando en el plano los elementos del modelo para su adecuada visualización.

3.3.4. Etapa 4

En este paso se recibe el modelo en BPMN 2.0 con la información gráfica (BPMNDI) para sustituir las tareas en tareas de coreografía, utilizando la información del XES extendido para obtener los participantes actuantes en cada tarea. Devolviendo finalmente, un archivo en formato BPMN que contiene el modelo de coreografía del XES en cuestión.

Capítulo 4

Desarrollo de prototipo

En este capítulo se presenta el desarrollo de la solución propuesta, el diseño conceptual del mismo, así como también, la lógica implementada para alcanzar los objetivos y como se integra con ProM.

4.1. Integración con ProM

Se pretende incluir la solución propuesta dentro de la plataforma ProM. La misma, como se mencionó anteriormente, cuenta con una gran variedad de plug-ins con técnicas de minería de procesos. La solución construida descubre el modelo de coreografía de un registro de eventos extendido, generando un archivo en formato BPMN como resultado.

En la Figura 4.1 se muestra una descripción general del marco ProM. Se puede ver que hay cinco tipos de plug-ins (van Dongen, de Medeiros, Verbeek, Weijters, y van der Aalst, 2005):

- Plug-ins de minería, los cuales implementan algún algoritmo de minería como puede ser construir Redes de Petri o construir un modelo de coreografía a partir de un registro de eventos.
- Plug-ins de exportación, permiten la funcionalidad de "guardar como"para algunos objetos.
- Plug-ins de importación, implementa una funcionalidad para procesar objetos exportados, como puede ser una Red de Petri.
- Plug-ins de análisis, realizan el análisis de propiedades de algunos resultados de minería.
- Plug-ins de conversión, realizan conversiones entre diferentes tipos de datos.

El plug-in Log Filter de la Figura 4.1 lee los registros del archivo XML generado a partir de algún sistema de información. Puede cargar otros modelos

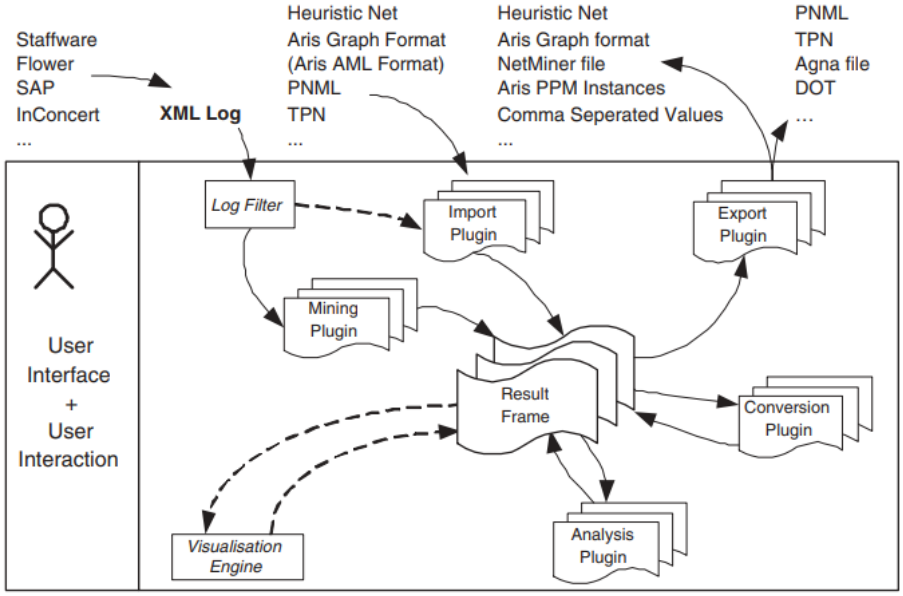


Figura 4.1: Marco ProM

a partir de los plug-ins de importación (Import Plugin), como también, puede aplicar plug-ins de minería (Mining Plugin) para realizar la extracción necesaria y finalmente generar el resultado para el marco (Result frame). Al resultado, se le pueden aplicar plug-ins de exportación (Export plugin), para así permitirle al usuario almacenar el resultado por fuera de la herramienta ProM. Se puede aplicar algún plug-in de conversión (Conversion plugin) para transformar el tipo del resultado generado en otro tipo de datos o se puede aplicar algún plug-in de análisis (Analysis plugin) para analizar el resultado, como puede ser calcular un lugar invariante para una Red de Petri. Además, ProM incluye visualizadores de los resultados generados para el marco (Visualization Engine).

4.2. Diseño conceptual

Se plantea desarrollar una herramienta que permita descubrir el modelo de coreografía en un proceso colaborativo, partiendo de un XES extendido de coreografía, permitiendo visualizar el resultado de forma gráfica. En la Figura 4.2 se muestra el diseño del prototipo.

Para alcanzar este objetivo, se desarrollaron dos plug-in, uno del estilo del Log Filter (Figura 4.1) el cual se encarga de leer el registro de eventos en formato XES extendido de coreografía, mantener una correspondencia entre el nombre de las tareas y los participantes de las mismas en una estructura de datos que es de utilidad para el procesamiento. Luego, obtiene el modelo BPMN generado por el plug-in BPMN Miner (incluído en ProM). Se hace una transformación

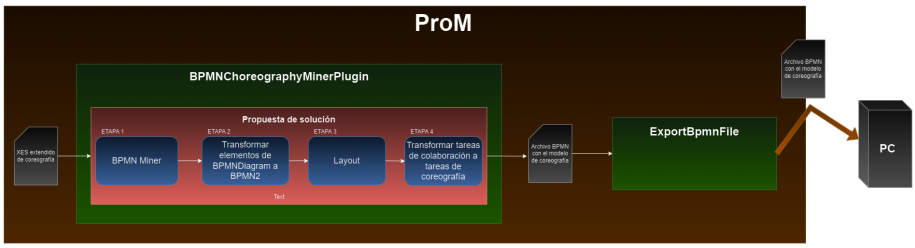


Figura 4.2: Prototipo de la solución

de los tipos de datos devueltos utilizando el metamodelo de BPMN 2.0. Se utiliza el BPMN Layout Generators con el resultado obtenido para generar el diseño gráfico del proceso BPMN pasándole únicamente la parte semántica del proceso. Por último, se intercambian las tareas de colaboración del resultado del BPMN Layout Generators por tareas de coreografía de BPMN 2.0 y se utiliza la estructura de datos definida anteriormente para obtener la correspondencia de las tareas con los participantes. El plug-in desarrollado, devuelve un archivo en formato BPMN con el resultado del modelo de coreografía correspondiente al registro de eventos.

Como ProM no incluye un visualizador de modelos BPMN de coreografías, se desarrolló un plug-in más del tipo Export Plugin (Figura 4.1), el cual toma como parámetro de entrada un objeto del tipo `Resource` y permite al usuario descargar el archivo BPMN de coreografía generado para que pueda ser abierto con alguna herramienta que admita elementos BPMN de coreografía (como puede ser chor-js (Ladleif, von Weltzien, y Weske, 2019)).

4.3. Lógica implementada

4.3.1. Template para el desarrollo de plug-ins

Para poder llevar a cabo el desarrollo del prototipo fue necesario instalar Java y Eclipse. Para iniciar el desarrollo de la solución, se consideró un ejemplo de implementación de plug-ins para ProM (*Template complementos para ProM*, 2011) y su guía (*Guía para crear complementos en ProM*, 2011). El código del template se encuentra almacenado en el control de versiones Subversion y para descargarlo se instaló la herramienta TortoiseSVN (*TortoiseSVN*, s.f.) para el sistema operativo Windows.

Los plug-ins en ProM son básicamente métodos a los cuales se le agregan anotaciones Java para que ProM detecte el plug-in y se pueda llamar desde la herramienta.

```
public class BPMNChoreographyMinerPlugin {
    @Plugin
    (
        name = "priced-choreography",
        parameterLabels = {"Log"},
        returnLabels = { "BPMN file" },
        returnTypes = { Resource.class },
        userAccessible = true,
        help = ""
    )
    @UITopiaVariant
    (
        affiliation = "My company",
        author = "My name",
        email = "My e-mail address"
    )
    public static Resource bpmnChoreographyMinerPlugin(final UIPluginContext context,
          XLog log)
    {
        //TODO: El cuerpo del plug-in
    }
}
```

Listing 4.1: Ejemplo de definición de un plug-in para ProM

En el ejemplo de definición de un plug-in para ProM se puede ver el método `bpmnChoreographyMinerPlugin` con un par de anotaciones Java. La anotación `@Plugin` es la que le indica al Framework ProM que el método es un plug-in con el nombre "priced-choreography" (en "name"), que recibe una lista de parámetros de entrada (en "parameterLabels"), que en este caso se etiqueta con el nombre "Log". La lista de etiquetas de los objetos devueltos por el plug-in (returnLabels), en este caso sólo se devuelve un objeto: "BPMN file". El tipo de retorno (en "returnTypes") "Resource.class". Además, se indica si el usuario puede tener acceso o no al plug-in (en "userAccessible"). Por último, se puede brindar algún texto explicativo sobre el plug-in (en "help").

La anotación `@IUTopiaVariant` le informa a la GUI de ProM de la existencia del plug-in y proporciona datos sobre el autor del mismo.

Para poder ejecutar el plug-in, es necesario hacer coincidir los tipos de datos de entrada con los especificados en el método que define el plug-in. El nombre del parámetro se especifica en la anotación y el tipo en el método. En la Figura 4.3 se muestra como ingresando los datos de entrada correctos para el plug-in, se muestra el mismo en las opciones de ProM. Mientras que en la Figura 4.4 se muestra que no hay plug-ins que reciban como dato de entrada un objeto del tipo `Resource`.

Ahora, en la Figura 4.5 se muestra que la salida del plug-in del ejemplo se muestra tal como se define en la anotación. El nombre de la etiqueta de los datos de salida "BPMN file" y el tipo del mismo `''Resource.class''`.

El Framework no se da cuenta si los objetos devueltos por el plug-in son los correctos hasta que el mismo no termina su ejecución, en caso de no serlo genera una excepción.

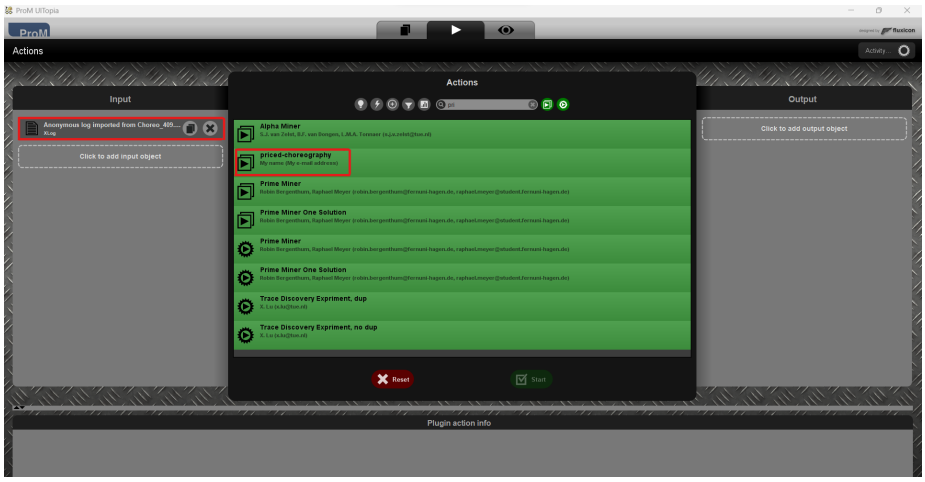


Figura 4.3: Ejemplo de datos de entrada correctos

Figura 4.4: Ejemplo de datos de entrada incorrectos

En la definición del plug-in "priced-choreography" se ve que en el método que implementa el plug-in se le pasa un parámetro `PluginContext` el cual es obligatorio y proporciona las interfaces necesarias para comunicarse con el Framework, con otros plug-ins y con el usuario. Se uso el tipo específico de contexto `UIPluginContext` debido a que es un plug-in que requiere interacción con el usuario.

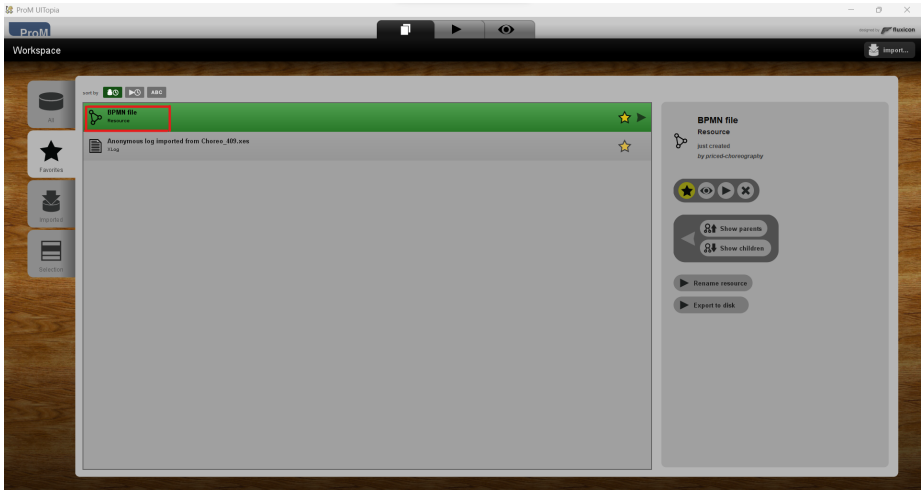


Figura 4.5: Ejemplo de datos de salida

4.3.2. Detalle del desarrollo

Anteriormente, se mencionó algunos conceptos claves para el desarrollo de plug-ins para poder ser integrados en el Framework ProM. Ahora se explica paso a paso como fue desarrollado el plug-in propuesto como solución en la Figura 3.2.

En eclipse se instaló el software BPMN2 Modeler (*BPMN2 Modeler*, 2011) que permite crear modelos que cumplen con el estándar de BPMN 2.0. El mismo, se incluyó en el proyecto para poder reutilizar las clases mencionadas en la Figura 2.6. También, se incluyeron los binarios de BPMN Miner y BPMN Layout Generator al proyecto.

En la Figura 4.6 se muestra el diagrama de clases implementado para alcanzar la solución propuesta. `BPMNChoreographyMinerPlugin` es la clase principal del plug-in, la cual contiene las anotaciones mencionadas anteriormente. Posee el método `bpmnChoreographyMinerPlugin` encargado de realizar la serie de pasos que se ven en la Figura 3.2, con parámetros `UIPluginContext` (necesario en todos los plug-in) y `XLog` (registro de eventos). A lo largo de todo el proceso de generación del modelo de coreografía, la solución desarrollada genera tres archivos BPMN intermedios. El primero contiene los elementos BPMN 2.0 luego de realizar la transformación de los elementos devueltos en el plug-in BPMN Miner. El segundo, contiene los elementos en BPMN 2.0 devueltos por el BPMN Layout Generator. Y finalmente, el último contiene los elementos de coreografía que permiten la visualización del diagrama. Para generar estos archivos se utilizó la clase `URI` (Identificador Uniforme de Recursos (*URI*, s.f.)), para indicar la ruta donde se quieren dejar los archivos. Se utilizó también, `Resource` (*Resource*, s.f.) para indicar el recurso que se quiere generar, que en este caso es un archivo del tipo BPMN y donde se le indica la ruta donde depositar el archivo con la

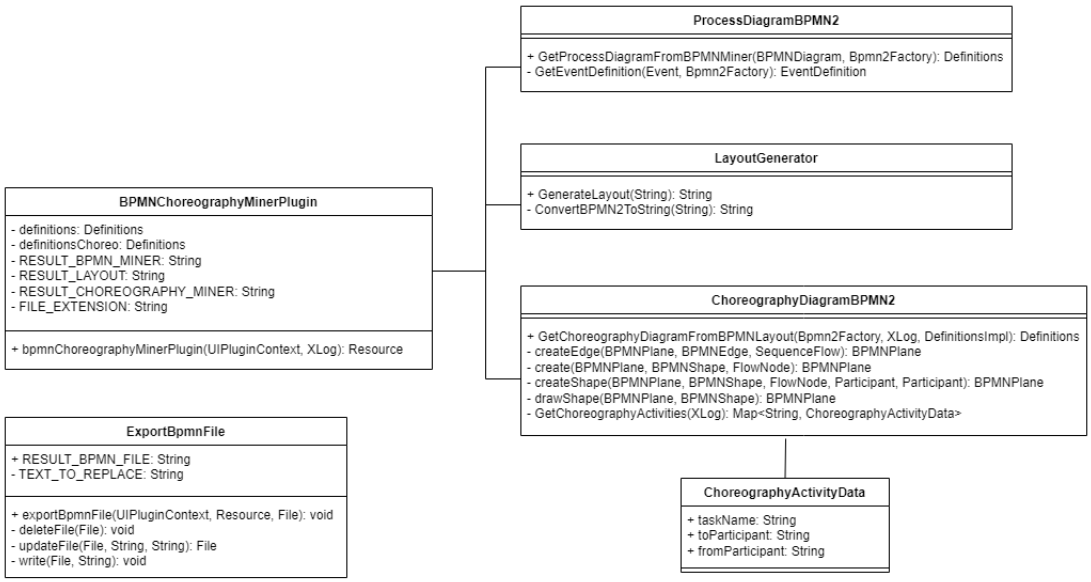


Figura 4.6: Diagrama de clases de la solución implementada

`URI` definida.

BPMN Miner y transformación de los tipos de datos

Como primer paso del método `bpmnChoreographyMinerPlugin`, se invoca el método `mineBPMNModelWithException` del plug-in BPMN Miner (*BPMN Miner*, 2016), con los mismos parámetros que el método principal (`UIPluginContext` y `XLog`). Este último devuelve un objeto del tipo `BPMNDiagram` (creado por los desarrolladores del plug-in BPMN Miner), con el diagrama de proceso obtenido a partir del registro de eventos (se corresponde con lo mencionado en la Etapa 1 de la subsección 3.3.1).

Este resultado es usado en la clase `ProcessDiagramBPMN2` invocando el método `GetProcessDiagramFromBPMNMiner` con los parámetros `BPMNDiagram` y `Bpmn2Factory` (utilizada para la generación de los elementos BPMN 2.0) donde transforma los elementos BPMN del tipo `BPMNDiagram` en elementos BPMN 2.0. Los elementos generados son incluidos en el objeto `Definitions`, que como se mencionó anteriormente incluye todos los elementos BPMN, y el método devuelve como resultado un objeto de este tipo. En esta clase también se desarrolló un método privado que determina el tipo de evento a generar en BPMN 2.0. Sin embargo, el BPMN Miner devuelve únicamente eventos con el `EventDefinition` con valor None, porque el mismo no determina el tipo a partir de las redes de Petri. De todas formas, se hizo el desarrollo para poder determinarlo en caso de poder utilizar un algoritmo que sea capaz de identificar el `EventDefinition`. El objeto `Definitions` devuelto, es almacenado en un archivo del

tipo BPMN (utilizando `URI` y `Resource`), quedando almacenado internamente para su posterior procesamiento (se corresponde con lo mencionado en la Etapa 2 de la subsección 3.3.2).

BPMN Layout Generator

Hasta ahora, tenemos únicamente la definición de los elementos que forman parte del diagrama de proceso de BPMN. Resta generar el diseño gráfico de estos elementos (BPMNDI) y transformar las tareas de proceso en coreografía. Para esto se invoca el método `GenerateLayout` de la clase `LayoutGenerator` con la ruta del archivo generado anteriormente como parámetro. Esta clase además posee un método privado que recibe la ruta del archivo como parámetro y devuelve una transformación al tipo `String` del mismo. Este resultado es utilizado para poder invocar el método `generateLayoutFromBPMNSemantic` de la clase `BPMNLayoutGenerator` (*BPMN Layout Generators*, 2020). Dicho método es invocado con el `String` definido anteriormente y con el tipo de archivo que se espera, en nuestro caso es del tipo BPMN. Este último, devuelve como resultado un `String` que contiene las especificaciones tanto semánticas como el diseño gráfico de los elementos BPMN. Retornando dicho resultado el método `GenerateLayout`, donde se genera un archivo con el contenido (utilizando `URI` y `Resource`). Lo antes descripto se corresponde con lo mencionado en la Etapa 3 de la subsección 3.3.3.

Transformar el modelo descubierto en coreografía

Por último (se corresponde con lo mencionado en la Etapa 4 de la subsección 3.3.4), con el resultado anterior, se almacenan los objetos BPMN en un objeto del tipo `Definitions` para poder invocar el método `GetChoreographyDiagramFromBPMNLayout` de la clase `ChoreographyDiagramBPMN2`, con los parámetros del tipo `Bpmn2Factory` (para generar los elementos BPMN 2.0 necesarios), `XLog` (el registro de eventos) y `DefinitionsImpl` (los elementos BPMN obtenidos del resultado anterior). Esta clase contiene varios métodos privados:

- `GetChoreographyActivities`, recibe como parámetro un objeto del tipo `XLog`. Este método recorre todo el registro de eventos para detectar los participantes de cada evento y poder realizar la asociación tarea-participantes (emisor y receptor). Esta asociación la guarda en la estructura de datos `ChoreographyActivityData` (almacena nombre de la tarea, participante emisor y participante receptor).

- `createEdge`, recibe como parámetro objetos del tipo `BPMNPlane` (contiene la especificación del diseño gráfico de todos los elementos BPMN), `BPMNEdge` (contiene la especificación del elemento del tipo `Edge` del diseño gráfico) y `SequenceFlow` (contiene la especificación del elemento BPMN al cual debe estar asociado el elemento `BPMNEdge` que se va a crear). Este método se encarga de generar un nuevo objeto del tipo `BPMNEdge` con la misma especificación que el objeto `BPMNEdge` que se pasa como parámetro,

asigna el objeto `SequenceFlow` que le corresponde de la parte semántica y agrega el objeto creado al `BPMNPlane` para que sea visible en el modelo.

- `create`, recibe como parámetros `BPMNPlane`, `BPMNShape` (contiene la especificación de alguno de los elementos BPMN que no son del tipo `Egde` del diseño gráfico) y `FlowNode` (contiene la especificación de alguno de los elementos del tipo semántico que no son del tipo `SequenceFlow`). Este método se encarga de crear un nuevo objeto del tipo `BPMNShape` con la misma especificación del objeto pasado por parámetro, se asocia el `SequenceFlow` correspondiente y se agrega el objeto creado al `BPMNPlane` para que sea visible en el modelo.

- `createShape`, recibe como parámetro objetos del tipo `BPMNPlane`, `BPMNShape`, `FlowNode`, `Participant` (contiene la especificación del tipo semántico del participante emisor) y `Participant` (contiene la especificación del tipo semántico del participante receptor). Este método se encarga de crear objetos del tipo `BPMNShape` para objetos del tipo `ChoreographyTask`. Tiene el mismo procedimiento que el método anterior, solo que además asigna los valores del diseño gráfico para los participantes.

- `drawShape`, recibe como parámetro objetos del tipo `BPMNPlane` y `BPMNShape`. Este método se encarga de realizar la asignación de los objetos creados en los métodos mencionados anterior mente al plano.

El método principal `GetChoreographyDiagramFromBPMNLayout`, se encarga de recorrer todos los elementos BPMN que vienen en el parámetro `Definitions`, y para cada elemento del tipo `Task` crea un objeto nuevo del tipo `ChoreographyTask` con la asociación almacenada en la estructura de datos `ChoreographyActivityData` y genera una copia de la parte de diseño gráfica con los método privados mencionados antes. Ahora, para los elementos que no son del tipo `Task`, se crea un objeto nuevo con las mismas especificaciones, con el mismo diseño gráfico (con los métodos privados mencionados antes). Retornando como resultado un nuevo objeto del tipo `Definitions`.

El resultado anterior es tomado por el método principal del plug-in desarrollado como resultado final del mismo, almacenando el resultado en un archivo utilizando `URI` y `Resource`. Por último, el plug-in devuelve este objeto, con las especificaciones semánticas y de diseño gráfico del modelo de coreografía.

En el Listing 4.2 se muestra la parte semántica del resultado obtenido aplicado al caso de estudio de solicitud de pasaporte.

```
<bpmn2:choreography id="Choreography_1" name="Default_Choreography">
    <bpmn2:participant id="Participant_0" name="AGESIC"/>
    <bpmn2:participant id="Participant_1" name="DNIC"/>
    ....
    ....
    <bpmn2:exclusiveGateway id="Gateway_0" name="DATABASED" gatewayDirection=
        "Diverging"/>
    <bpmn2:choreographyTask id="Task_5" name="Notify appointment result"
        initiatingParticipantRef="Participant_1">
      <bpmn2:participantRef>Participant_1</bpmn2:participantRef>
      <bpmn2:participantRef>Participant_0</bpmn2:participantRef>
    </bpmn2:choreographyTask>
    <bpmn2:choreographyTask id="Task_3" name="Judicial records response"
        initiatingParticipantRef="Participant_2">
      <bpmn2:participantRef>Participant_2</bpmn2:participantRef>
      <bpmn2:participantRef>Participant_1</bpmn2:participantRef>
    </bpmn2:choreographyTask>
    <bpmn2:exclusiveGateway id="Gateway_2" name="DATABASED" gatewayDirection=
        "Diverging"/>
    <bpmn2:exclusiveGateway id="Gateway_4" name="DATABASED" gatewayDirection=
        "Converging"/>
    <bpmn2:choreographyTask id="Task_4" name="Notify appointment cancellation"
        initiatingParticipantRef="Participant_1">
      <bpmn2:participantRef>Participant_1</bpmn2:participantRef>
      <bpmn2:participantRef>Participant_0</bpmn2:participantRef>
    </bpmn2:choreographyTask>
    <bpmn2:choreographyTask id="Task_2" name="Has judicial records"
        initiatingParticipantRef="Participant_1">
      <bpmn2:participantRef>Participant_1</bpmn2:participantRef>
      <bpmn2:participantRef>Participant_2</bpmn2:participantRef>
    </bpmn2:choreographyTask>
    <bpmn2:exclusiveGateway id="Gateway_1" name="DATABASED" gatewayDirection=
        "Diverging"/>
    <bpmn2:exclusiveGateway id="Gateway_3" name="DATABASED" gatewayDirection=
        "Converging"/>
    <bpmn2:endEvent id="Event_1" name="END"/>
    <bpmn2:sequenceFlow id="Flow_0" sourceRef="Task_0" targetRef="Gateway_0"/>
    <bpmn2:sequenceFlow id="Flow_1" sourceRef="Task_2" targetRef="Gateway_1"/>
    <bpmn2:sequenceFlow id="Flow_2" sourceRef="Task_3" targetRef="Gateway_2"/>
    <bpmn2:sequenceFlow id="Flow_3" sourceRef="Gateway_4" targetRef="Task_4"/>
    ....
    ....
  </bpmn2:choreography>
```

Listing 4.2: Resultado plug-in: semántica de los elementos BPMN - Solicitud de pasaporte

En el Listing 4.3 se muestra la parte del diseño gráfico del resultado obtenido aplicado al caso de estudio de solicitud de pasaporte.

```
<bpmndi:BPMNDiagram id="BPMNDiagram_1">
    <bpmndi:BPMNPlane id="BPMNPlane_1" bpmnElement="Choreography_1">
      <bpmndi:BPMNShape id="_0Lf9UWMmEe6MRNnSsJshzA" bpmnElement="
          Event_0">
        <dc:Bounds height="30.0" width="30.0" x="85.0" y="35.0"/>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="_0Lf9UmMmEe6MRNnSsJshzA" bpmnElement="Task_1
          ">
        <dc:Bounds height="60.0" width="120.0" x="240.0" y="20.0"/>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="BPMNShape_FPTask_1" bpmnElement="Participant_0"
          choreographyActivityShape="_0Lf9UmMmEe6MRNnSsJshzA">
        <dc:Bounds height="60.0" width="120.0" x="240.0" y="20.0"/>
      </bpmndi:BPMNShape>
      ....
      ....
      <bpmndi:BPMNShape id="_0LgkZ2MmEe6MRNnSsJshzA" bpmnElement="Event_1
          ">
        <dc:Bounds height="30.0" width="30.0" x="2285.0" y="135.0"/>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNEdge id="_0LgkaGMmEe6MRNnSsJshzA" bpmnElement="Flow_0"
          >
        <di:waypoint xsi:type="dc:Point" x="560.0" y="50.0"/>
        <di:waypoint xsi:type="dc:Point" x="670.0" y="50.0"/>
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNEdge id="_0LgkaWMmEe6MRNnSsJshzA" bpmnElement="Flow_1"
          >
        <di:waypoint xsi:type="dc:Point" x="960.0" y="150.0"/>
        <di:waypoint xsi:type="dc:Point" x="1070.0" y="150.0"/>
      </bpmndi:BPMNEdge>
      ....
      ....
```

Listing 4.3: Parte del resultado: diseño gráfico de los elementos BPMN - Solicitud de pasaporte

Descargar el modelo de coreografía generado

Para poder visualizar el modelo generado, se implemento el plug-in `ExportBpmnFile` que posee el método `exportBpmnFile`. El mismo recibe como parámetros de entrada objetos del tipo `UIPluginContext`, `Resource` (definido para poder descargar el archivo generado por el plug-in anterior) y `File` (tiene el contenido del objeto `Resource`). Para este plug-in, fue necesario agregar método privados para el manejo del archivo y así poder editar el contenido de los mismo. Esto es debido a que al momento de salvar los archivos generados, cuando se hace la asignación de los identificadores a los elementos BPMN se asociaba con una referencia incorrecta lo que ocasionaba que el archivo no se pudiera visualizar.

Capítulo 5

Casos de aplicación

En este capítulo se va a analizar los resultados obtenidos de aplicar la solución desarrollada a un registro de eventos con datos de un proceso real de Gobierno Digital. También, se evaluará con logs de eventos de procesos colaborativos de una propuesta existente.

5.1. Caso de estudio: Solicitud de pasaporte

En el Gobiernos Digital (e-Govermment) los procesos utilizados suelen ser colaborativos. La plataforma de Gobierno Electrónico (PGE) de la Agencia de Gobierno Electrónico y Sociedad de la Información y del Conocimiento (AGESIC) de Uruguay brinda una plataforma de interoperabilidad que permite a las organizaciones gubernamentales llevar a cabo procesos de negocio colaborativos entre ellas y los ciudadanos. Entre los procesos de negocio que participan en la plataforma de interoperabilidad se encuentra la solicitud de pasaporte.

5.1.1. Descripción del caso de estudio

El caso de estudio trata con datos de un proceso real de AGESIC, proceso de solicitud de pasaporte. Para este caso, en la Figura 5.1 se muestra el diagrama de coreografía (González y Delgado, 2021) que se corresponde con el proceso.

Si observamos la Figura 5.1, AGESIC inicia la interacción con la DNIC solicitando las fechas disponibles. Luego, AGESIC le envía la confirmación de la cita, que puede ser aceptar o no aceptar. En caso de no aceptar, el proceso finaliza. En caso de aceptar, el proceso continúa y la DNIC le consulta a DNPT si tiene registros judiciales. Si pasan 24 horas sin recibir respuesta, la DNIC le cancela la cita a AGESIC y finaliza el proceso. Si la DNPT le envía a la DNIC la respuesta indicando si posee registros judiciales antes de las 24 horas, el proceso continua. En caso de tener registros, la DNIC le cancela la cita a AGESIC y el proceso finaliza. En caso de no tener registros, la DNIC le confirma la cita a AGESIC y finaliza el proceso.

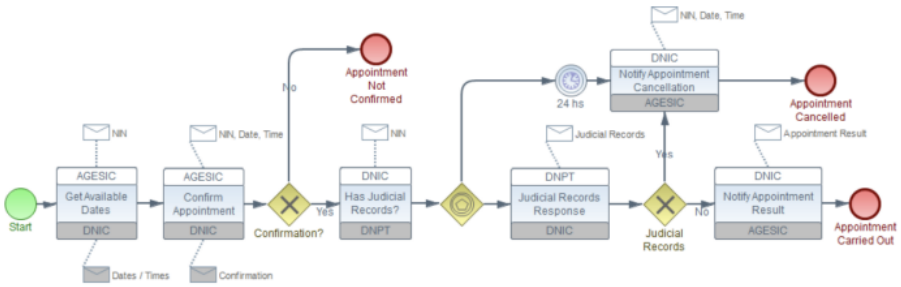


Figura 5.1: Diagrama de coreografía de solicitud de pasaporte

Dado que se tiene el diagrama de coreografía, se podrá realizar una comparación del mismo con el resultado obtenido de la solución desarrollada.

En el Listing 5.1 se muestra un ejemplos de traza que completa el proceso de solicitud de pasaporte. Al pasarle este registro de eventos con trazas similares a esta a la solución desarrollada, se van a detectar las actividades: "`Get available dates`", "`Confirm appointment`", "`Has judicial records`", "`Judicial records response`", "`Notify appointment result`" y "`Notify appointment cancellation`". Junto con los participantes: AGESIC, DNIC y DNPT.

En el ejemplo de traza, se pueden ver los detalles que se explicaron antes. Donde el camino que se sigue en dicha traza es el de la confirmación de la cita, ya que no se presentan registros judiciales y el DNPT respondió antes de las 24 horas de la consulta (esto se puede ver en la fecha que se indica en los eventos correspondientes).

```
<trace>
    <string key="concept:name" value="Trace 4" />
    <event>
        <string key="concept:name" value="Get available dates" />
        <date key="time:timestamp" value="2020-04-15T21:32:15.000-03:00" />
        <string key="lifecycle:transition" value="complete" />
        <string key="org:resource" value="Miguel Gonzalez" />
        <string key="org:role" value="User Support" />
        <string key="collab:elemType" value="Message" />
        <string key="collab:fromParticipant" value="AGESIC" />
        <string key="collab:toParticipant" value="DNIC" />
    </event>
    <event>
        <string key="concept:name" value="Confirm appointment" />
        <date key="time:timestamp" value="2020-04-17T19:34:38.000-03:00" />
        <string key="lifecycle:transition" value="complete" />
        <string key="org:resource" value="Pedro Garcia" />
        <string key="org:role" value="Coordination" />
        <string key="collab:elemType" value="Message" />
        <string key="collab:fromParticipant" value="AGESIC" />
        <string key="collab:toParticipant" value="DNIC" />
    </event>
    <event>
        <string key="concept:name" value="Has judicial records" />
        <date key="time:timestamp" value="2020-04-17T19:35:07.000-03:00" />
        <string key="lifecycle:transition" value="complete" />
        <string key="org:resource" value="invalid" />
        <string key="org:role" value="invalid" />
        <string key="collab:elemType" value="Message" />
        <string key="collab:fromParticipant" value="DNIC" />
        <string key="collab:toParticipant" value="DNPT" />
    </event>
    <event>
        <string key="concept:name" value="Judicial records response" />
        <date key="time:timestamp" value="2020-04-17T21:37:36.000-03:00" />
        <string key="lifecycle:transition" value="complete" />
        <string key="org:resource" value="invalid" />
        <string key="org:role" value="invalid" />
        <string key="collab:elemType" value="Message" />
        <string key="collab:fromParticipant" value="DNPT" />
        <string key="collab:toParticipant" value="DNIC" />
    </event>
    <event>
        <string key="concept:name" value="Notify appointment result" />
        <date key="time:timestamp" value="2020-04-18T21:37:05.000-03:00" />
        ...
```

Listing 5.1: Ejemplo de una traza del registro de eventos de coreografía de solicitud de pasaporte

5.1.2. Aplicación del caso de estudio

Como primer paso, cargamos el registro de eventos al ProM y seleccionamos la opción ProM log files (Naive) como se muestra en la Figura 5.2. A continua-

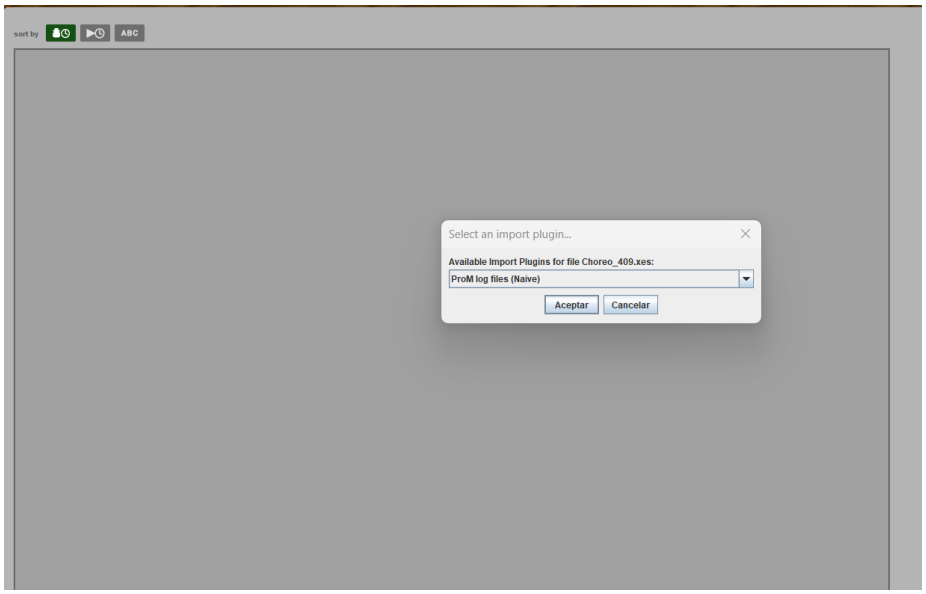


Figura 5.2: Cargar XES en ProM

ción, hacemos click en el botón remarcado en la Figura 5.3 y buscamos el plugin desarrollado (priced-choreography) como se muestra en la Figura 5.4 y le damos al botón Start.

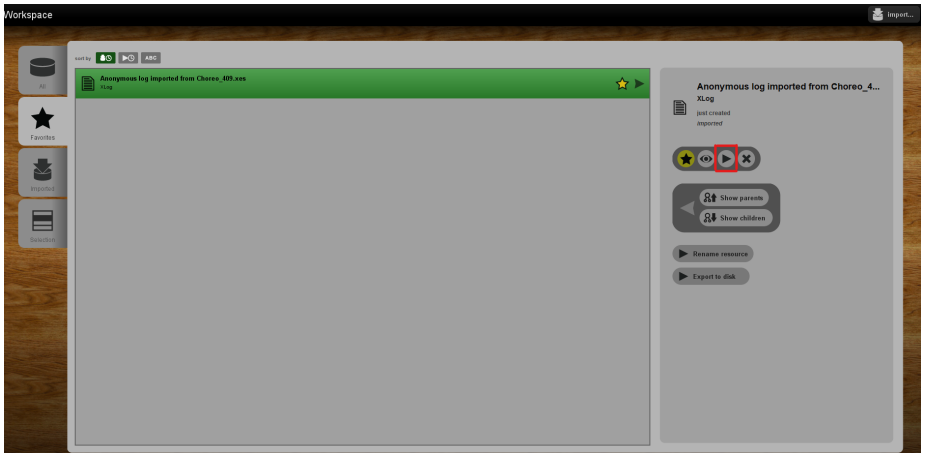


Figura 5.3: Usar el recurso importado

Como primera instancia del proceso, se pide que se seleccione el algoritmo de descubrimiento que se quiere utilizar, este paso forma parte del plugin BPMN Miner (Figura 5.5). En este caso utilizamos el algoritmo Heuristics Miner

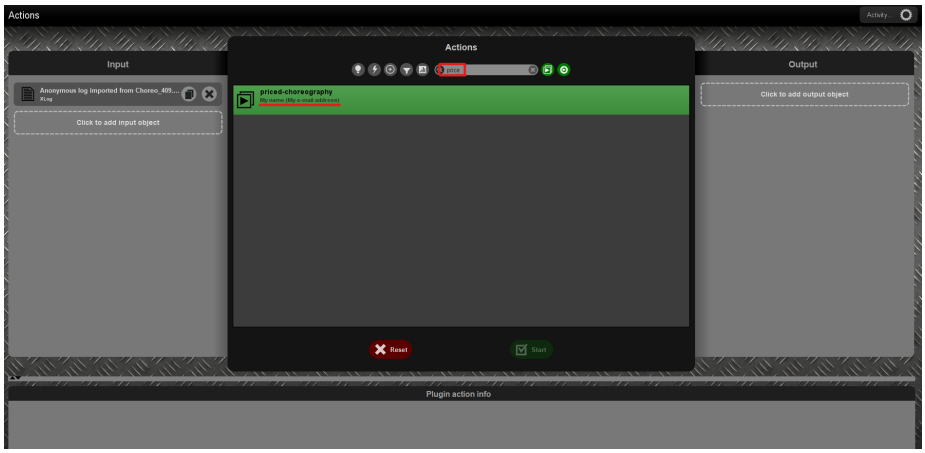


Figura 5.4: Seleccionar el plugin desarrollado

ProM6.

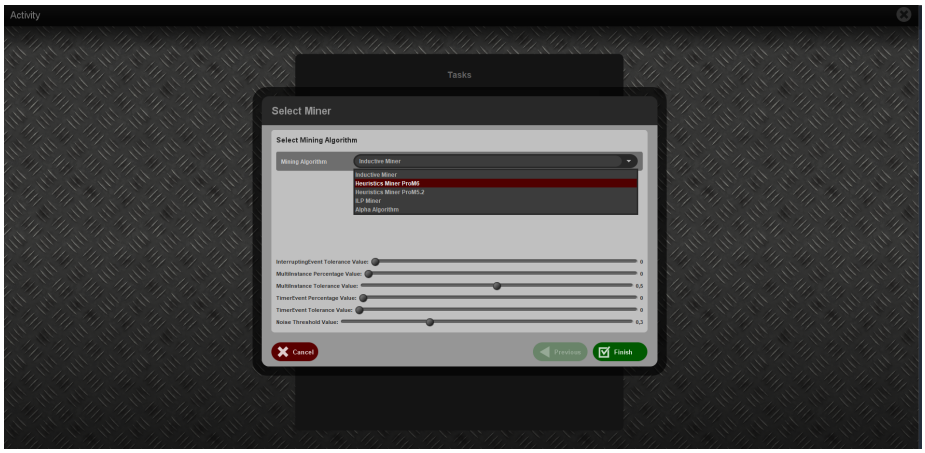


Figura 5.5: Selección del algoritmo de descubrimiento

A continuación, se muestra una ventana con las posibles claves que se quieren usar para la identificación de las jerarquías a identificar, en nuestro caso, como se dijo en secciones anteriores, no se tienen jerarquías en los registros de eventos utilizados. Por esta razón, dejamos deshabilitados todos los check boxes de las posibles claves (Figura 5.6).

El siguiente paso se selecciona el clasificador utilizado, por defecto viene el valor Event Name que es el que dejaremos (Figura 5.7).

Luego de los pasos anteriores, ya vemos el archivo BPMN resultado (Figura 5.8), el cual lo podemos descargar de la plataforma para poder visualizar su contenido o el diagrama en alguna herramienta que soporta elementos de co-

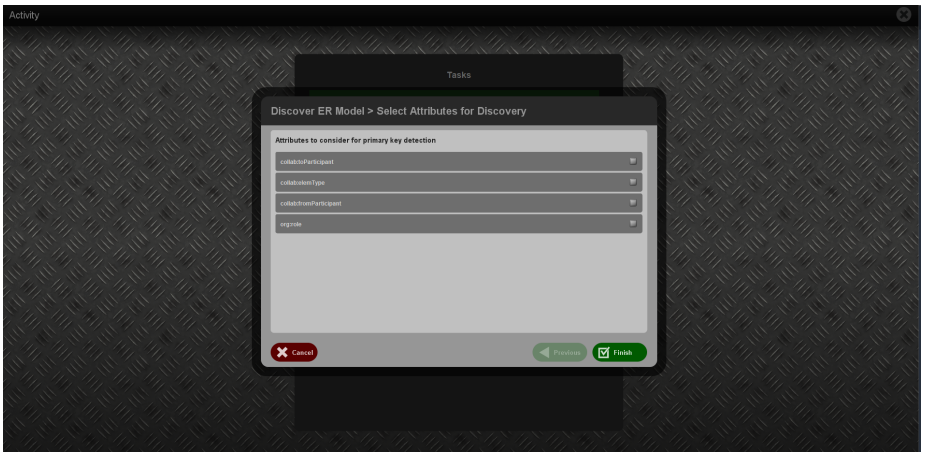


Figura 5.6: Selección de claves a agrupar

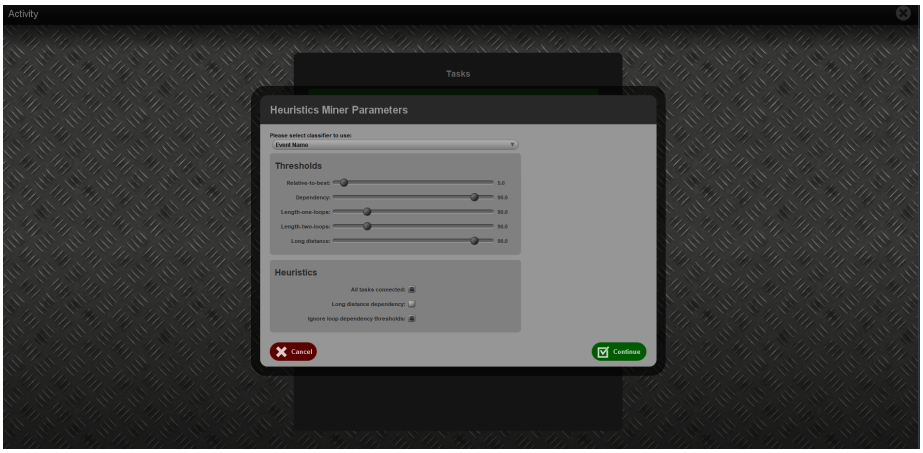


Figura 5.7: Selección del clasificador

reografía. Para descargar el archivo generado, se utiliza el plugin implementado para exportar archivos del tipo Resource, el cual habilita el botón Export to disk (Figura 5.9).

Para visualizar el diagrama generado se utilizó la herramienta chor-js (Ladleif et al., 2019), donde cargamos el archivo que se descargó anteriormente presionando en el botón que se recuadra en rojo en la Figura 5.10. En la Figura 5.11 se puede ver el resultado en dicho visualizador.

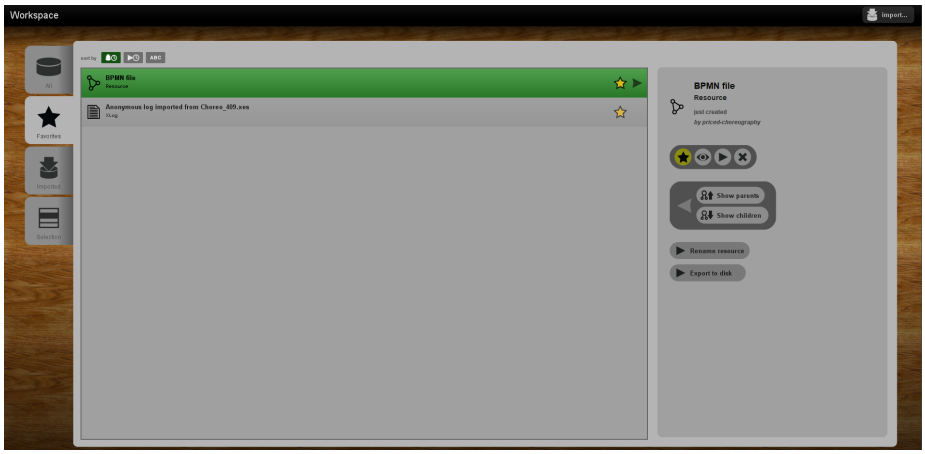


Figura 5.8: Resultado del plugin desarrollado

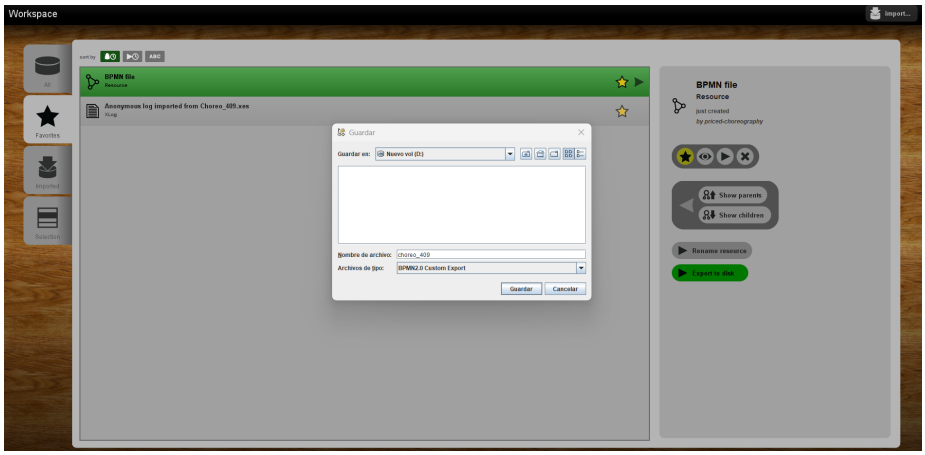


Figura 5.9: Exportar el resultado obtenido

5.1.3. Evaluación del resultado

Para analizar el resultado del plug-in, se van a hacer dos comparaciones. Por un lado se comparará con el resultado del plug-in Filter Event Log de ProM aplicado al XES extendido de coreografía, el cual además de permitir el filtrado del log muestra las variantes de los caminos que aparecen en el mismo. Y por otro, con el diagrama de coreografía de la Figura 5.1.

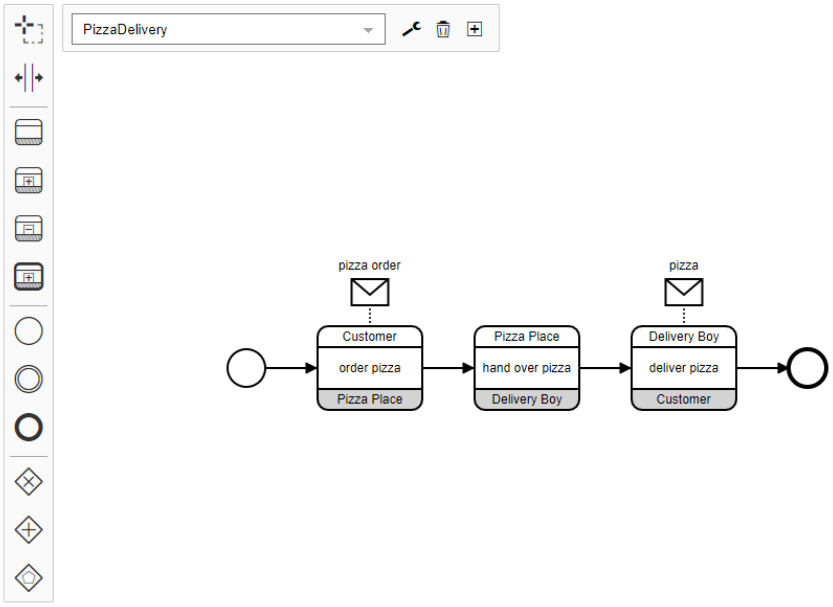


Figura 5.10: Visualizador chor-js

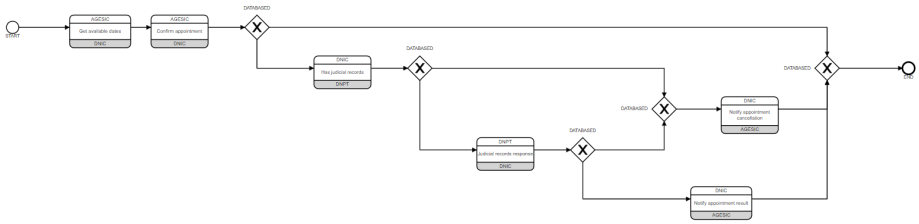


Figura 5.11: Coreografía de solicitud de pasaporte - Visualizado en chor-js

Comparación con la variante de caminos del plug-in Filter Event Log de ProM

Las variantes de los caminos del plug-in Filter Event Log de ProM son los que se muestran en la Figura 5.12. Como se puede ver, existen cuatro posibles

caminos. Para validar que el resultado obtenido se corresponde con el log, verificaremos que existan esos cuatro caminos en la coreografía descubierta. En las

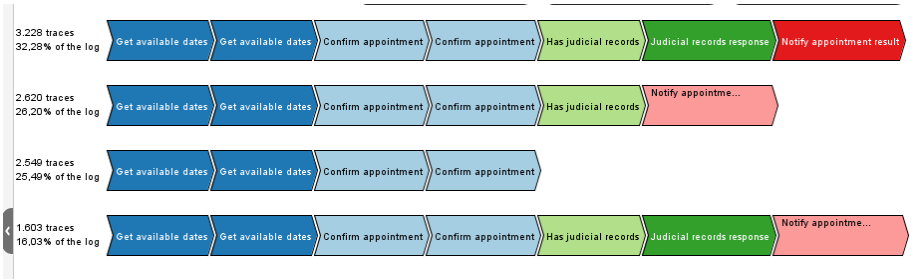


Figura 5.12: Variante de caminos del plug-in Filter Event Log de ProM

Figuras 5.13, 5.14, 5.15 y 5.16 se muestra cada uno de los caminos en el mismo orden que se muestran en la Figura 5.12.

El Camino 1 (Figura 5.13) representa el caso exitoso del proceso, donde AGESIC le solicita fechas disponibles y se confirma una de ellas a la DNIC. La DNIC le solicita a DNPT los registros judiciales de la persona y recibe como respuesta que la misma no posee registros judiciales, con lo cual la DNIC le confirma la solicitud a AGESIC.

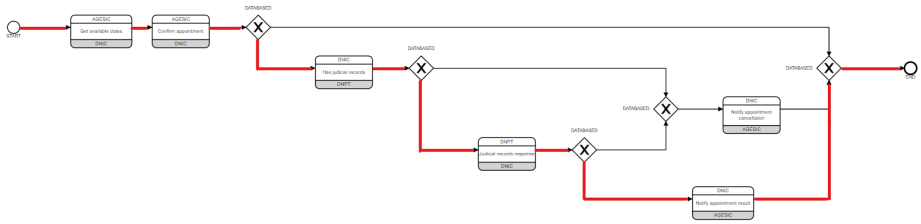


Figura 5.13: Camino 1 de solicitud de pasaporte descubierto por el plug-in

El Camino 2 (Figura 5.14) representa el caso en que el DNPT no responde o tarda en responder si la persona posee registros judiciales por lo que la DNIC le cancela la solicitud a AGESIC.

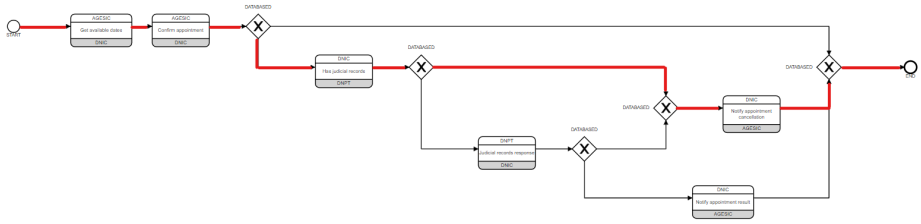


Figura 5.14: Camino 2 de solicitud de pasaporte descubierto por el plug-in

El Camino 3 (Figura 5.15) representa el caso en que la interacción finaliza

cuando AGESIC le confirma una de las fechas disponibles, quedando incompleto el proceso ya que no hubo más intercambio de mensajes.

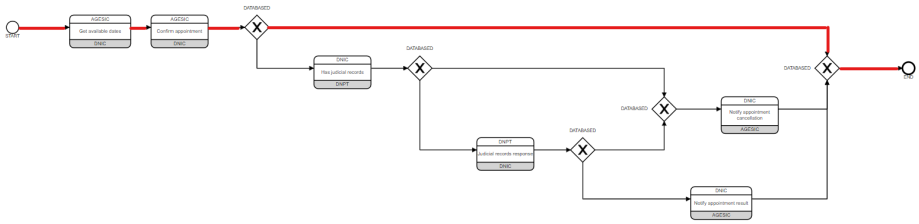


Figura 5.15: Camino 3 de solicitud de pasaporte descubierto por el plug-in

El Camino 4 (Figura 5.16) representa el caso en que el DNPT le indica a la DNIC que la persona posee registros judiciales con lo cual la DNIC le cancela la solicitud a AGESIC.

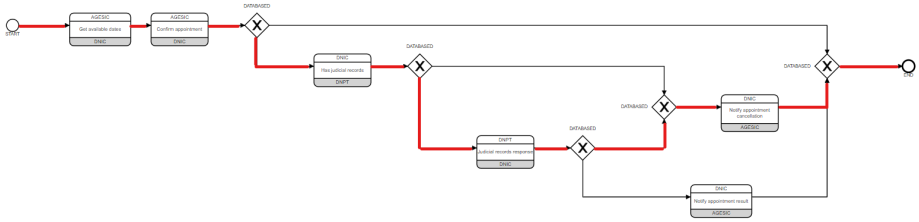
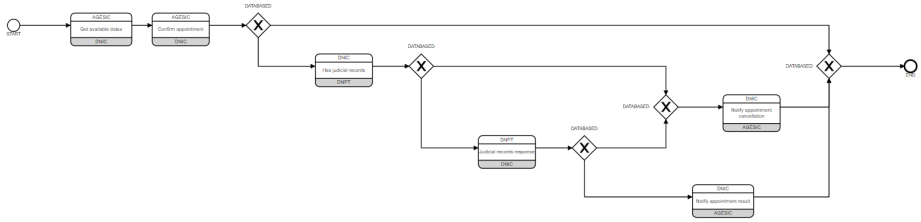


Figura 5.16: Camino 4 de solicitud de pasaporte descubierto por el plug-in

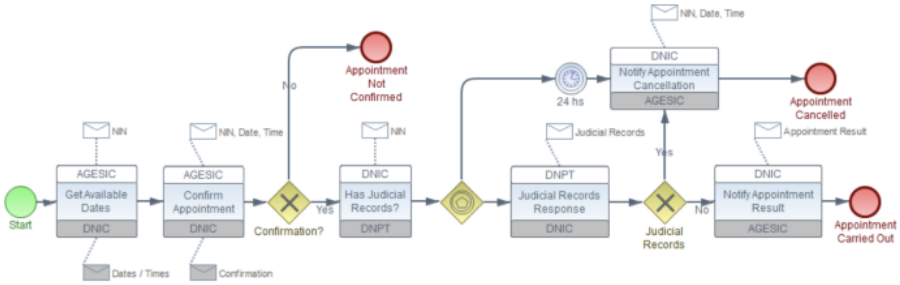
Debido a que el resultado obtenido cubre las variantes de caminos identificados en el log de eventos, se puede decir que la solución desarrollada se adecúa a los objetivos esperados.

Comparación con la Figura 5.1

En la Figura 5.17 se puede ver que las actividades que son antecesoras al evento final se corresponden con las que se tienen en el resultado: `Confirm appointment`, `Notify appointment cancellation` y `Notify appointment result`.



(a) Coreografía de solicitud de pasaporte descubierta por el plug-in



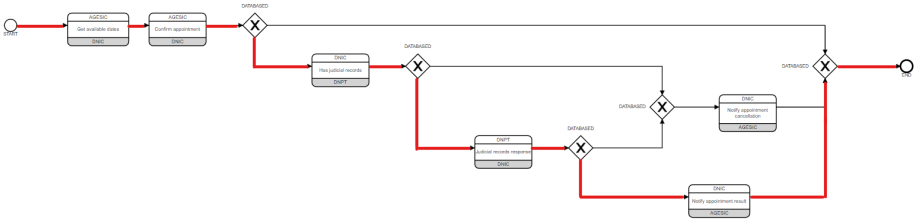
(b) Coreografía de solicitud de pasaporte (González y Delgado, 2021)

Figura 5.17: Comparación de resultados

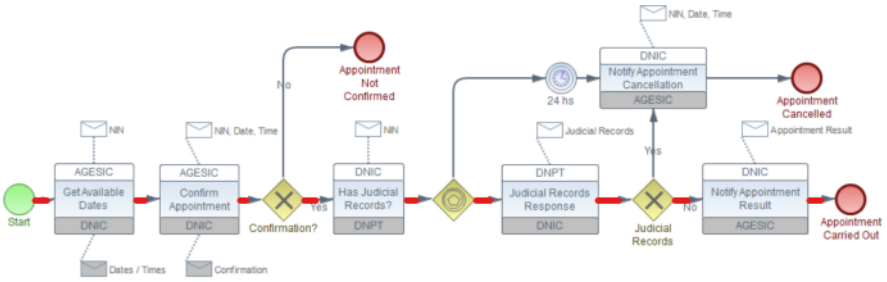
Los eventos que aparecen no son los mismos debido a lo que se comentó en secciones anteriores, el BPMN Miner solo detecta los eventos de inicio y de fin. Lo mismo ocurre con los gateways, solo se detectan los del tipo Exclusive y Parallel.

En el resultado también se tienen dos gateway más debido al algoritmo utilizado. El BPMN Miner trabaja con un evento inicial y un evento final, por lo que para poder abarcar los tres posibles caminos que llegan al evento final, es necesario agregar un gateway Exclusive y así representar que se va a seguir un único camino para llegar al mismo. Lo mismo ocurre con la tarea `Notify appointment cancellation`, donde se llega a ella mediante dos posibles caminos y el algoritmo lo representa agregando un gateway Exclusive.

A pesar de las diferencias mencionadas anteriormente, el resultado obtenido representa cada uno de los posibles caminos se pueden ver en las Figuras 5.18, 5.19, 5.20 y 5.21 junto a los resultados obtenidos.

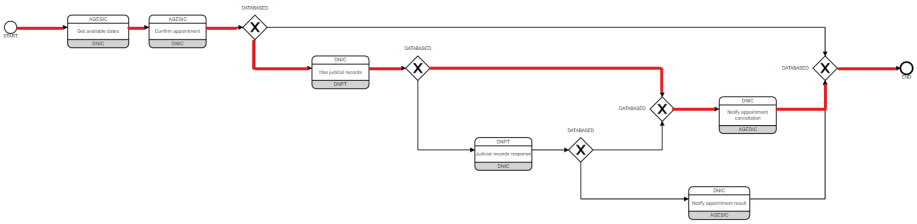


(a) Camino 1 de solicitud de pasaporte descubierto por el plug-in

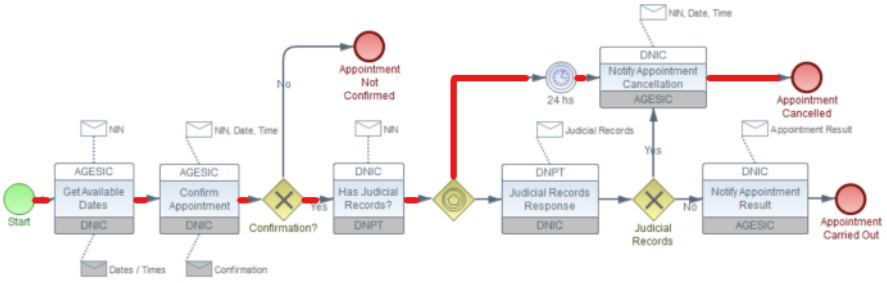


(b) Camino 1 de solicitud de pasaporte (González y Delgado, 2021)

Figura 5.18: Comparación de resultados de camino 1

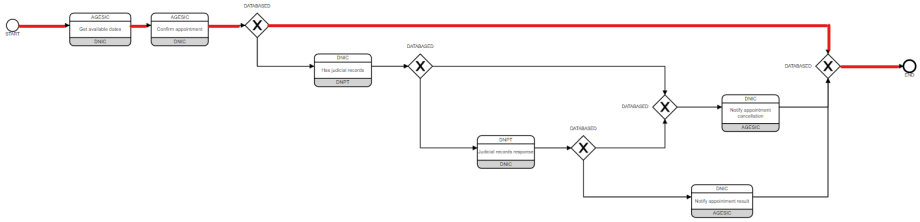


(a) Camino 2 de solicitud de pasaporte descubierto por el plug-in

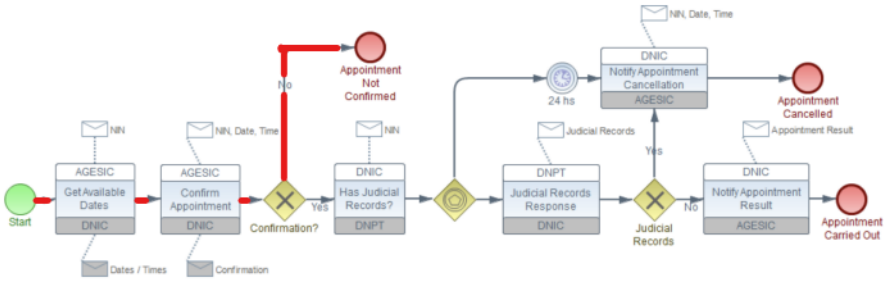


(b) Camino 2 de solicitud de pasaporte (González y Delgado, 2021)

Figura 5.19: Comparación de resultados de camino 2

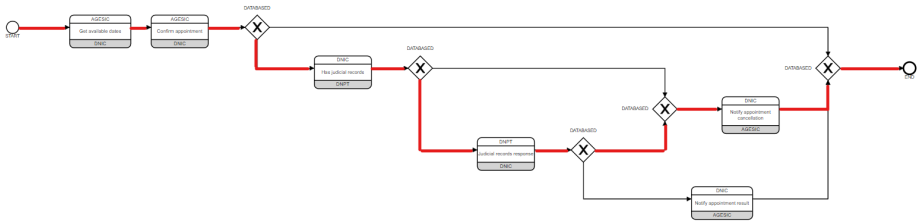


(a) Camino 3 de solicitud de pasaporte descubierto por el plug-in

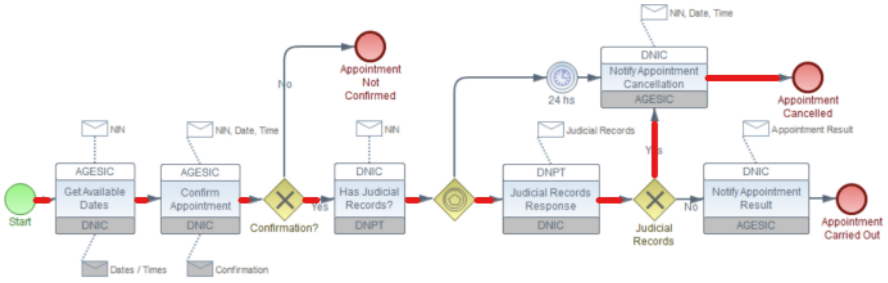


(b) Camino 3 de solicitud de pasaporte (González y Delgado, 2021)

Figura 5.20: Comparación de resultados de camino 3



(a) Camino 4 de solicitud de pasaporte descubierto por el plug-in



(b) Camino 4 de solicitud de pasaporte (González y Delgado, 2021)

Figura 5.21: Comparación de resultados de camino 4

5.2. Casos de estudio adicionales

En esta sección se van a mostrar los resultados obtenidos para diferentes logs (Corradini, Re, Rossi, y Tiezzi, 2022), los cuales se utilizaron para la evaluación del plug-in desarrollado para descubrir el modelo de colaboración de procesos colaborativos (Peña L., Delgado A., Calegari D., 2022). Algunos logs (*Colliery Validation*, s.f.-b) fueron generados con datos de procesos reales y otros con procesos artificiales. Los mismos fueron transformados a logs extendidos de coreografía para poder ser utilizados por el plug-in desarrollado. En el Listing 5.2 se muestra una traza de uno de los XES artificiales, donde para transformarlo en un XES de coreografía se tomó la información de los eventos de `key="msgInstanceId"` con `value="m2_124"` y `value=‘‘m2_125’’`.

Estos eventos (son cuatro) son los correspondientes a los mensjaes intercambiados, uno del tipo “send” y el otro del tipo “receive”. En el Listing 5.3 se muestra la traza correspondiente para el XES de coreografía con la información de los eventos del tipo “send” (incluyendo el participante que inicia la comunicación) y agregando la información del participante para los eventos del tipo “receive”. De esta forma se hizo la transformación de los XES de colaboración en XES de coreografía para poder utilizarlos en el proyecto.

```
<trace>
    <string key="concept:name" value="case_188"/>
    <event>
        <string key="org:group" value="PartyA"/>
        <string key="concept:name" value="Activity A"/>
        <date key="time:timestamp" value="2021-06-23T11:00:05.205+02:00"/>
    </event>
    <event>
        <string key="org:group" value="PartyC"/>
        <string key="concept:name" value="Activity ZA"/>
        <date key="time:timestamp" value="2021-06-23T11:00:08.182+02:00"/>
    </event>
    <event>
        <string key="msgInstanceId" value="m2_124"/>
        <string key="msgType" value="send"/>
        <string key="org:group" value="PartyC"/>
        <string key="concept:name" value="Activity CZ"/>
        <date key="time:timestamp" value="2021-06-23T11:00:08.508+02:00"/>
        <string key="msgFlow" value="m2"/>
    </event>
    <event>
        <string key="msgInstanceId" value="m2_124"/>
        <string key="msgType" value="receive"/>
        <string key="org:group" value="PartyA"/>
        <string key="concept:name" value="Activity C"/>
        <date key="time:timestamp" value="2021-06-23T11:00:14.126+02:00"/>
        <string key="msgFlow" value="m2"/>
    </event>
    <event>
        <string key="msgInstanceId" value="m1_125"/>
        <string key="msgType" value="send"/>
        <string key="org:group" value="PartyC"/>
        <string key="concept:name" value="Activity DZ"/>
        <date key="time:timestamp" value="2021-06-23T11:00:15.495+02:00"/>
        <string key="msgFlow" value="m1"/>
    </event>
    <event>
        <string key="msgInstanceId" value="m1_125"/>
        <string key="msgType" value="receive"/>
        <string key="org:group" value="PartyA"/>
        <string key="concept:name" value="Activity D"/>
        <date key="time:timestamp" value="2021-06-23T11:00:23.433+02:00"/>
        <string key="msgFlow" value="m1"/>
    </event>
    <event>
        <string key="org:group" value="PartyC"/>
        <string key="concept:name" value="Activity BZ"/>
        <date key="time:timestamp" value="2021-06-23T11:00:23.515+02:00"/>
    </event>
    ...
    ...
</trace>
```

Listing 5.2: Ejemplo de una traza de un XES (Corradini et al., 2022) - collectivelog_artificial1.xes

```
<trace>
        <string key="concept:name" value="case_188"/>
        <event>
            <string key="ref_org:group" value="PartyC"/>
            <string key="msgInstanceId" value="m2_124"/>
            <string key="msgType" value="send"/>
            <string key="org:group" value="PartyC"/>
            <string key="ref_concept:name" value="Activity CZ"/>
            <string key="collab:elemType" value="Message"/>
            <string key="collab:toParticipant" value="PartyA"/>
            <date key="time:timestamp" value="2021-06-23T06
                :00:08.508-03:00"/>
            <string key="ref_msgType" value="send"/>
            <string key="concept:name" value="Activity CZ"/>
            <string key="ref_msgFlow" value="m2"/>
            <string key="collab:fromParticipant" value="PartyC"/>
            <string key="ref_time:timestamp" value="2021-06-23T06
                :00:08.508-03:00"/>
            <string key="ref_msgInstanceId" value="m2_124"/>
            <string key="msgFlow" value="m2"/>
        </event>
        <event>
            <string key="ref_org:group" value="PartyC"/>
            <string key="msgInstanceId" value="m1_125"/>
            <string key="msgType" value="send"/>
            <string key="org:group" value="PartyC"/>
            <string key="ref_concept:name" value="Activity DZ"/>
            <string key="collab:elemType" value="Message"/>
            <string key="collab:toParticipant" value="PartyA"/>
            <date key="time:timestamp" value="2021-06-23T06
                :00:15.495-03:00"/>
            <string key="ref_msgType" value="send"/>
            <string key="concept:name" value="Activity DZ"/>
            <string key="ref_msgFlow" value="m1"/>
            <string key="collab:fromParticipant" value="PartyC"/>
            <string key="ref_time:timestamp" value="2021-06-23T06
                :00:15.495-03:00"/>
            <string key="ref_msgInstanceId" value="m1_125"/>
            <string key="msgFlow" value="m1"/>
        </event>
    </trace>
```

Listing 5.3: Ejemplo de una traza de un XES extendido de coreografía correspondiente a la traza del Listing 5.2

El plug-in (Peña L., Delgado A., Calegari D., 2022) junto con el que se desarrolló en este proyecto, forman parte del proyecto "Minería de procesos y datos para la mejora de procesos colaborativos aplicada a e-Government" financiado por la Agencia Nacional de Investigación e Innovación (ANII), Fondo María Viñas (FMV) "Proyecto ANII N° FMV_1_2021_1_167483", Uruguay.

Para evaluar los resultados del plug-in desarrollado vamos a aplicar el plug-in Filter Event Log para ver los posibles caminos que puedan existir en el log.

5.2.1. Caso 1

En la Figura 5.22 se pueden ver los posibles caminos que se representan en el registro de eventos para este caso. Este caso es bastante sencillo, solo hay dos actividades las cuales ocurren siempre pero en diferente orden.

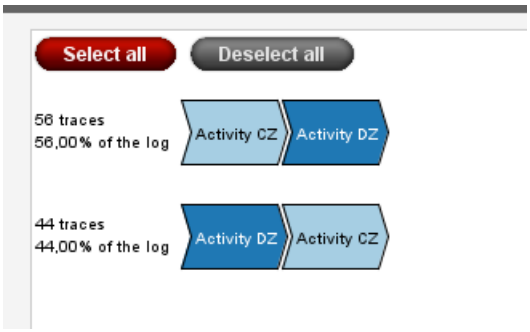


Figura 5.22: Caso 1

Si comparamos el resultado obtenido que se puede visualizar en la Figura 5.23, se puede ver que se tienen gateway del tipo Parallel, lo cual permite que ambas actividades se ejecuten de forma paralela y pueden ocurrir en diferente orden.

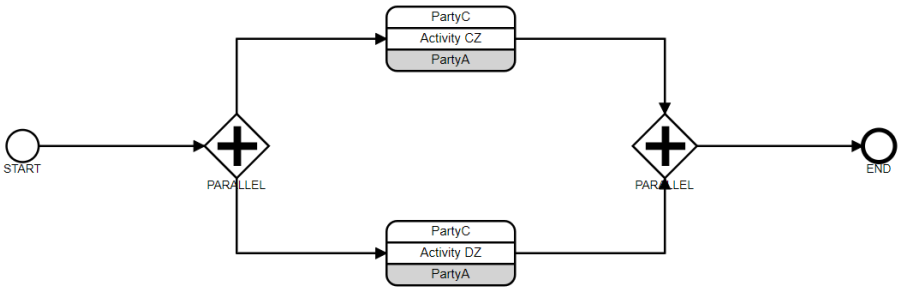


Figura 5.23: Resultado caso 1

5.2.2. Caso 2

En la Figura 5.24 se pueden ver los posibles caminos que se representan en el registro de eventos para este caso. En este caso ocurre algo similar que el caso anterior, solo que se tienen tres actividades las cuales pueden ocurrir de forma paralela y en diferente orden. Es por esto que aparece un posible recorrido más que en el caso anterior.

Si comparamos el resultado obtenido que se puede visualizar en la Figura 5.25, se puede ver que se tienen gateway del tipo Parallel, lo cual permite que las

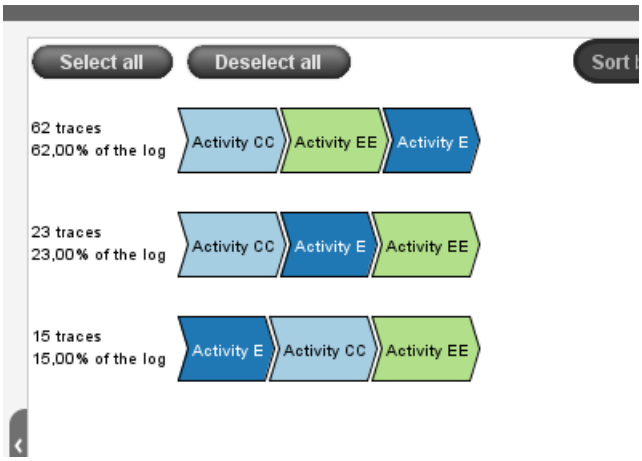


Figura 5.24: Caso 2

actividades se ejecuten de forma paralela y pueden ocurrir en diferente orden.

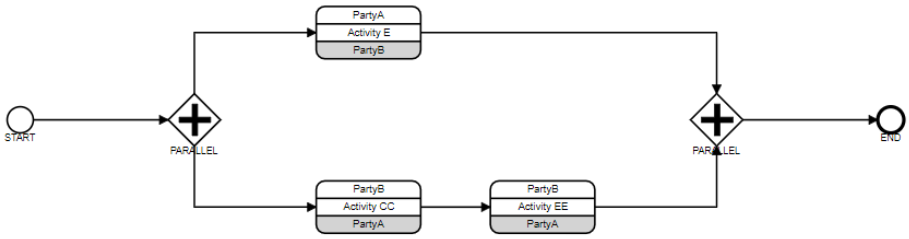


Figura 5.25: Resultado caso 2

5.2.3. Caso 3

En la Figura 5.26 se pueden ver los posibles caminos que se representan en el registro de eventos para este caso.

Si comparamos el resultado obtenido que se puede visualizar en la Figura 5.27, se puede ver que se tienen gateway del tipo Parallel, lo cual permite que las actividades se ejecuten de forma paralela y pueden ocurrir en diferente orden pudiendo representar todos los caminos.

5.2.4. Caso 4

En la Figura 5.28 se pueden ver los posibles caminos que se representan en el registro de eventos para este caso. Este caso es bastante simple, hay dos actividades donde solo ocurre una de ellas.

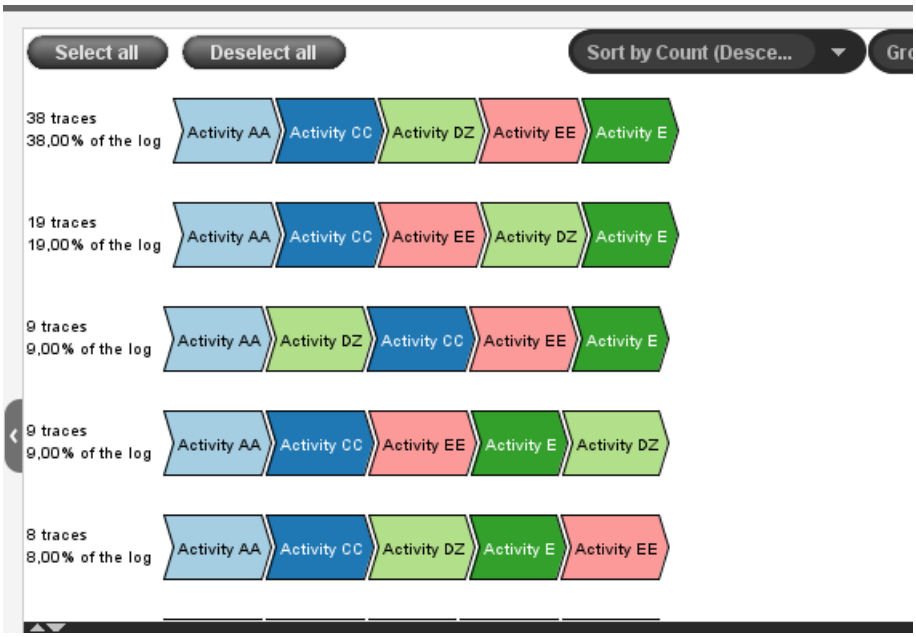


Figura 5.26: Caso 3

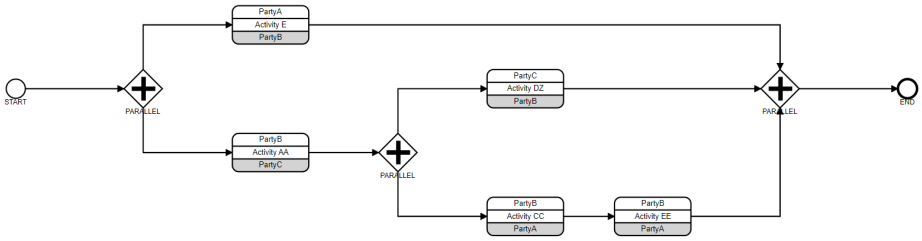


Figura 5.27: Resultado caso 3

Si comparamos el resultado obtenido que se puede visualizar en la Figura 5.29, se puede ver que se tienen gateway del tipo Exclusive, lo cual permite que se realice solo una de las actividades.

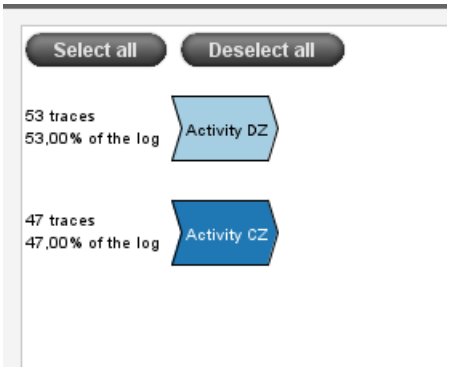


Figura 5.28: Caso 4

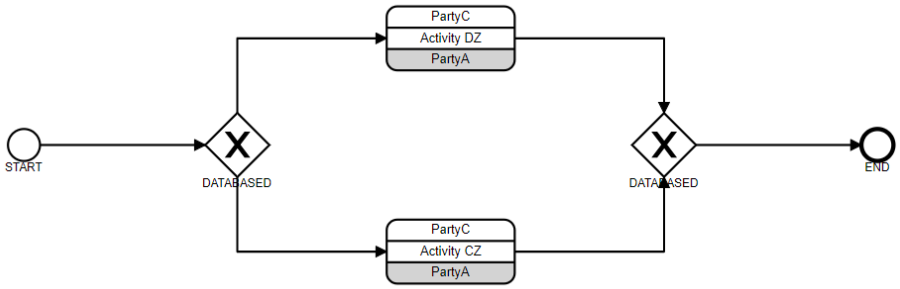


Figura 5.29: Resultado caso 4

Capítulo 6

Conclusiones y Trabajo Futuro

El proyecto tuvo como objetivo la apliación y/o definición de nuevas técnicas y algoritmos de minería de procesos para el descubrimiento y análisis de coreografías de procesos colaborativos en BPMN 2.0 a partir de la información almacenada en los registros de eventos de diferentes organizaciones.

El objetivo OE1 (estudiar conceptos de BPM, Minería de Procesos, configuración y ejecución de procesos, lenguajes BPMN 2.0, Petri Nets y herramientas) fue alcanzado realizando la investigación correspondiente al marco teórico con los conceptos que fueron necesarios para tener una introducción en el tema de Minería de Procesos. Con esto se puede decir que dicho objetivo fue alcanzado.

Para el objetivo OE2 (analizar técnicas, algoritmos, herramientas y propuestas existentes para minería de coreografías en procesos colaborativos), se investigo herramientas existentes para minería de coreografías. No se encontró ningún artículo que planteara una solución para el descubrimiento de coreografías en procesos colaborativos. Además, se investigaron algoritmos existentes que puedan ser integrados en la solución. Entre los cuales, se llegó a que el BPMN Miner (integrado en la herramienta ProM como un plug-in) genera el modelo BPMN de los registros de eventos que se pasan como entrada. Además, se encontró el BPMN Layout Generator que permite generar el diseño gráfico de los elementos BPMN. Con esto podemos decir que alcanzamos el objetivo planteado.

En el objetivo OE3 (generar propuesta/extensión de minería de procesos para el descubrimiento de coreografías en procesos colaborativos y su visualización en BPMN 2.0) también podemos decir que fue un objetivo alcanzado ya que se propuso una solución que integra el BPMN Miner para descubrir el modelo BPMN colaborativo del registro de eventos proporcionado. El BPMN Layout Generator, que permitió facilitar la ubicación gráfica de los elementos BPMN descubiertos en el BPMN Miner. Y por último, tomar el resultado de la integración anterior para transformar las actividades descubiertas en actividades de

coreografía, pudiendo devolver el modelo esperado.

El objetivo OE4 (desarrollar/extender/adaptar herramienta de soporte a la propuesta) también fue alcanzado, ya que se desarrolló la propuesta de solución planteada en el objetivo anterior como un plug-in que se puede integrar en la herramienta ProM. Un trabajo a futuro sobre este punto es investigar el manejo de URI y Resource en el código, ya que debido a tiempos estipulados no se pudo realizar y a la hora de guardar un archivo utilizando esta herramienta, los elementos no se almacenaban de forma correcta.

Por último, para el objetivo OE5 (aplicar la propuesta a datos reales y soporte extendido de herramientas) se probó la herramienta desarrollada con datos de un proceso real del Gobierno Digital y con logs de una propuesta con datos reales y artificiales. Para todos los casos evaluados el resultado obtenido permite los mismos posibles caminos que se presentan en los logs, luego de aplicar el plug-in Filter Event Log de ProM. En el caso de solicitud de pasaporte, el modelo descubierto presenta algunas diferencias con el modelos del caso de estudio (gateways adicionales y menos eventos) pero permite el mismo flujo de caminos posibles. Con lo antes dicho, podemos decir que se alcanzó el objetivo y se tuevieron buenos resultados.

Como trabajo futuro, se podría investigar otras herramientas de descubrimiento o desarrollar una nueva con el fin de poder tener resultados más exactos. Además, sería bueno poder determinar que tipo de eventos corresponden ya que el BPMN Miner solo detecta eventos del tipo Start y End.

También, como trabajo a futuro se puede pensar en la evaluación de la herramienta desarrollada con una mayor variedad de casos y poder validar los resultados con expertos en el tema. De la misma forma, sería ideal poder contar con la visualización del modelo dentro de la herramienta de ProM y que permita, además, modificar el modelo. También se podría desarrollar el plug-in como una aplicación web que reutilice herramientas existentes para la visualización del modelo descubierto ya que ProM no provee facilidad para poder visualizar el resultado.

Referencias

Andrade D., Delgado A., Calegari D. (2023). *BPMN 2.0 choreography process discovery ProM plugin.* https://gitlab.fing.edu.uy/dandrade/priced-choreographypm.

Bpmn2 modeler. (2011). https://wiki.eclipse.org/BPMN2-Modeler. (Accessed: 2023-12-04)

BPMN layout generators. (2020). https://github.com/process-analytics/bpmn-layout-generators. (Accessed: 2023-06-05)

BPMN Miner. (2016). https://svn.win.tue.nl/repos/prom/Packages/BPMNMiner/Trunk/. (Accessed: 2023-07-18)

Business process model and notation (BPMN). (2011). https://www.omg.org/spec/BPMN/2.0/PDF. (Accessed: 2023-09-11)

Colliery validation. (s.f.-a). https://bitbucket.org/proslabteam/colliery_validation/src/master/artificial1/model.bpmn. (Accessed: 2023-11-20)

Colliery validation. (s.f.-b). https://bitbucket.org/proslabteam/colliery_validation/src/master/. (Accessed: 2023-12-05)

Conforti, R., Dumas, M., García-Bañuelos, L., y Rosa, M. L. (2016). BPMN miner: Automated discovery of BPMN process models with hierarchical structure. *Inf. Syst.*, *56*, 284–303.

Corradini, F., Re, B., Rossi, L., y Tiezzi, F. (2022). A technique for collaboration discovery. En *Enterprise, business-process and information systems modeling* (pp. 63–78). Springer.

González, L., y Delgado, A. (2021). Towards compliance requirements modeling and evaluation of e-government inter-organizational collaborative business processes. En *54th hawaii intl. conf. on system sciences, HICSS 2021* (pp. 1–10). ScholarSpace.

González, L., y Delgado, A. (2021). Compliance requirements model for collaborative business process and evaluation with process mining. En *XLVII latin american computing conference (CLEI).*

Guía para crear complementos en ProM. (2011). https://svn.win.tue.nl/trac/prom/wiki/setup/HowToCreatePluginsInProM. (Accessed: 2023-06-18)

iProcess. (s.f.). https://blog.iprocess.com.br/2013/09/bpmn-2-0-novos-diagramas-e-elementos-coreografia-no-detalhe/

diagrama-de-coreografia-diagrama-com-elementos/. (Accessed: 2023-11-30)

Ladleif, J., von Weltzien, A., y Weske, M. (2019). chor-js: A modeling framework for BPMN 2.0 choreography diagrams. En *Proceedings of the ER forum and poster & demos session 2019 on publishing papers with CEUR-WS co-located with 38th intl. conf. on conceptual modeling (ER 2019)* (Vol. 2469, pp. 113–117). CEUR-WS.org. https://bpt-lab.org/chor-js-demo/.

OMG. (2011). *Business Process Model and Notation (BPMN) 2.0* (Inf. Téc.). Autor.

Peña, L., Andrade, D., Delgado, A., y Calegari, D. (2023). An approach for discovering inter-organizational collaborative business processes in bpmn 2.0. En *2nd workshop on collaboration mining for distributed systems (cominds 2023) held in conjunction with the 5th international conference on process mining (icpm 2023), rome, italy, 2023, in press.*

Peña L., Delgado A., Calegari D. (2022). *BPMN 2.0 collab. process discovery ProM plugin.* https://gitlab.fing.edu.uy/open-coal/bpmncollaborativepm.

Resource. (s.f.). https://download.eclipse.org/modeling/emf/emf/javadoc/2.7.0/org/eclipse/emf/ecore/resource/Resource.html. (Accessed: 2023-09-24)

Template complementos para ProM. (2011). https://svn.win.tue.nl/repos/prom/Packages/GettingStarted/Trunk/. (Accessed: 2023-06-06)

TortoiseSVN. (s.f.). https://tortoisesvn.uptodown.com/windows. (Accessed: 2023-06-06)

URI. (s.f.). https://download.eclipse.org/modeling/emf/emf/javadoc/2.4.3/org/eclipse/emf/common/util/URI.html. (Accessed: 2023-09-24)

van der Aalst, W. M. P. (2016a). Intra- and inter-organizational process mining: Discovering processes within and between organizations. En P. Johannesson, J. Krogstie, y A. L. Opdahl (Eds.), *The practice of enterprise modeling* (pp. 1–11). Springer.

van der Aalst, W. M. P. (2016b). *Process mining - data science in action, second edition.* Springer. Descargado de https://doi.org/10.1007/978-3-662-49851-4 doi: 10.1007/978-3-662-49851-4

van Dongen, B., de Medeiros, A., Verbeek, H., Weijters, A., y van der Aalst, W. (2005). *The prom framework: A new era in process mining tool support.*

Verbeek, H. M. W., Buijs, J. C. A. M., van Dongen, B. F., y van der Aalst, W. M. P. (2011). Xes, xesame, and prom 6. En P. Soffer y E. Proper (Eds.), *Information systems evolution* (pp. 60–75). Springer.

Weske, M. (2019). *Business process management - concepts, languages, architectures, third edition.* Springer.