UNIVERSIDAD DEL BÍO-BÍO
FACULTAD DE CIENCIAS EMPRESARIALES
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN Y TECNOLOGÍAS DE LA INFORMACIÓN
ESCUELA DE INGENIERÍA CIVIL EN INFORMÁTICA

# TRANSFORMACIÓN CIM A PIM: OBTENCIÓN DE CASOS DE USO A PARTIR DE PROCESOS DE NEGOCIO ENRIQUECIDOS

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

AUTOR: LEONEL IGNACIO MUÑOZ VERGARA

PROFESOR GUÍA: DR. ALFONSO RODRÍGUEZ RÍOS

CHILLÁN, 27 DE FEBRERO DE 2019

# Agradecimientos

Quiero agradecer en primera instancia a mis padres Leonel Muñoz y Erika Vergara, los cuales me formaron como un hombre con valores y con la capacidad de superar todos los desafíos que la vida me propone, por el gran apoyo durante todos estos años de estudio y el gran amor que me brindaron día a día para seguir adelante.

También agradecer a mis amigos, familiares y hermanos que siempre me apoyaron y tuvieron palabras de ánimo en los momentos cruciales, a los que me ayudaron e inspiraron a seguir en esta hermosa carrera y los que siempre creyeron en mí.

Quiero agradecer a Constanza De la Hoz quien en este último año se convirtió en un pilar fundamental en el ámbito académico y de la vida cotidiana, dándome su apoyo y energía en todo momento para poder lograr mis objetivos en la vida.

Finalmente agradecer al Dr. Alfonso Rodríguez Ríos, él me dio la oportunidad de adentrarme en esta área de la informática, agradezco mucho su apoyo e inspiración en cuanto lo académico. Agradezco todos los consejos que me brindó para poder realizar mi memoria.

Muchas gracias a todos.

# Resumen

En la actualidad las organizaciones prestan mucha atención a los procesos de negocio, reconociendo estos procesos como un recurso importante los cuales les permiten situarse en una mejor posición respecto a sus competidores. Junto con la importancia de los procesos de negocio, las organizaciones comienzan a preocuparse tanto de la calidad de datos como la seguridad de su información, ya que esto permite llevar a cabo sus tareas en forma más segura y confiable. Este interés por parte de las organizaciones y la dificultad para definir los requisitos del software, son la motivación para el desarrollo de esta memoria, en la cual se busca lograr expresar seguridad y calidad de datos en los modelos de procesos de negocio descritos con BPMN y proporcionar una transformación en la cual se toman estos modelos enriquecidos y se convierten en Diagramas de Casos de Uso. Para lograr esto fue necesario mejorar la herramienta existente de manera que sea posible modelar procesos de negocio con seguridad y calidad de datos, además de definir un conjunto de reglas para realizar la transformación hacia Diagramas de Casos de Uso.

# Abstract

Currently, organizations pay much attention to business processes, recognizing these processes as an important resource which allows them to be in a better position with respect to their competitors. Along with the importance of business processes, organizations begin to worry as much about the quality of data as the security of their information, as this allows them to carry out their tasks in a more secure and reliable way. This interest on the part of the organizations and the difficulty to define the requirements of the software, are the motivation for the development of this memory, in which the aim is to express security and data quality in the business process models described with BPMN and provide a transformation in which these enriched models are taken and converted into use case diagrams. To achieve this it was necessary to improve the existing tool so that it is possible to model business processes with security and data quality, in addition to defining a set of rules to perform the transformation to use case diagrams.

# Índice general

# Índice de figuras

# Índice de tablas

# Capítulo 1

# Introducción

Uno de los principales problemas para las organizaciones es poder definir claramente los requisitos que deberá soportar el software que apoya a los Sistemas de Información (SI). Una temprana definición de los requisitos, en la etapa de construcción del sistema permite ayudar a los encargados del desarrollo y de alguna manera facilita el trabajo en la industria (Rodríguez and Caro, 2012). Por otra parte, la Calidad de Datos (DQ, Data Quality) y la seguridad son aspectos importante a tener en cuenta por las organizaciones para el buen funcionamiento de sus procesos y la claridad de su información. Junto a lo antes mencionado cabe destacar la nueva escena de negocio, en la cual encontramos muchos participantes y un uso intensivo de las Tecnologías de la Información (TI) y las comunicaciones. Esto implica que las empresas no sólo expanden su negocio, sino que también aumentan su vulnerabilidad (Rodríguez et al., 2007). Vulnerabilidad que tarde o temprano será aprovechada para causar daño. En este contexto han cobrado importancia la DQ y la seguridad puesto que ambos factores contribuyen a preservar el negocio al tener mejores datos para las decisiones y mayor seguridad en cuanto a la operación de sistema. Tanto DQ y la seguridad se están volviendo populares y fundamentales para las organizaciones por el hecho de influir en gran medida en su éxito o fracaso. Las organizaciones que incorporen una estrategia de gestión de DQ y seguridad en sus estrategias de negocio serán capaces de convertir sus datos en ventajas competitivas.

En cuanto a la especificación temprana de los requisitos de un SI constituye un desafío permanente para los ingenieros encargados de desarrollar esta labor. Las inquietudes de abordar y tener una buena gestión de DQ y seguridad, de igual manera obtener una temprana adopción de los requisitos de un sistema es un factor importante para el desarrollo de este trabajo.

Tanto DQ como la seguridad, es bueno capturarlos en altos niveles de abstracción a la hora de realizar desarrollo de software, debido a que esto contribuye a una buena y completa definición de los requisitos del sistema. Si representamos estos requisitos en un Proceso de Negocio (BP, Business Process), lograremos obtener un modelo en el cual se es consciente de la DQ y la seguridad. En un enfoque Model Driven Architecture (MDA) estos BP enriquecidos se pueden situar en el nivel Computational Independent Model (CIM), lo que permitiría transformarlos en Diagramas de Casos de Uso (UCD, Use Case Diagram) que en el enfoque MDA corresponden al nivel Platform Indepedent Model (PIM).

Para la representación de seguridad y DQ en los diagramas de procesos de negocio se utilizará como base la herramienta propuesta en (Fuentes et al., 2015). Esta herramienta permite utilizar un modelador de procesos de negocio con la extensión DQ, la cual consiste en agregar una marca "DQ" a los elementos que se desea indicar esta característica. Esta herramienta ha sido modificada, como parte del trabajo realizado en esta memoria, de manera que sea posible representar requisitos de seguridad. Esto quiere decir que se agregaron los elementos correspondientes para que los usuarios de dicha herramienta puedan enriquecer los modelos BP descritos con BPMN(Business Process Model and Notation). Para llevar a cabo esta tarea se ha propuesto una variante del método BPiDQ* (Business Process including Data Quality) presentado en (Rodríguez and Caro, 2012), al cual llamaremos BPiDQSec (Business Process including Data Quality and Security), señalando mediante la "Sec" la integración de los requisitos de seguridad.

El siguiente objetivo que se abordará en este proyecto es la realización de una transformación de modelos, la cual considerará como elementos de origen los modelos BP enriquecido y descritos con BPMN. Sobre estos modelos se aplicará un conjunto predefinido de reglas descritas en Atlas Transformation Language (ATL) que permiten obtener como salida de estas transformaciones un UCD.

El resto del trabajo está organizado como sigue. En la Sección 2 se presentan los antecedentes relacionados con el tema de la memoria. Las tecnologías que se utilizan para desarrollar la memoria se encuentran en la Sección 3. Las propuestas presentadas las cuales consideran el método BPiDQSec, metamodelos, modificaciones realizadas al *plugin* DQ, las transformaciones y la herramienta desarrollada son presentadas en la Sección 4. Los resultados del proyecto se presentan en la Sección 5. Para finalizar con las conclusiones y trabajos futuros presentadas en la Sección 6.

## 1.1. Objetivo General

El objetivo general de esta memoria es:

- Proponer un nuevo método BPiDQSec, basado en BPiDQ*, que permita especificar procesos de negocio enriquecidos con requisitos de calidad de datos y seguridad, complementar modelador de BPMN propuesto por (Fuentes et al., 2015) para que sea posible incorporar la calidad de datos y la seguridad en los modelos de procesos de negocio y construir una herramienta con la cual se puedan realizar transformaciones desde los modelos descritos con BPMN hacia diagramas de casos de uso.

## 1.2. Objetivos Específicos

Los objetivos específicos del proyecto son:

- Comprender y aplicar conceptos referentes a la calidad de datos y a la seguridad dentro de los modelos BP descritos con BPMN. Esto consiste en comprender la notación y método propuesto para el desarrollo de modelos enriquecidos.
- Modificar el *plugin* DQ (Fuentes et al., 2015). Para ello se debe analizar el código fuente con el fin de comprenderlo y luego generar las clases para agregar la extensión de seguridad.
- Aprender el funcionamiento de BPiDQ*. Esto con el fin de poder presentar una modificación que sea consistente con lo antes presentado.
- Aprender el lenguaje de transformación de modelos ATL y generar las reglas que ayudan a pasar del nivel CIM a PIM. Específicamente de modelos BP a UCD.

## 1.3. Enfoque del Trabajo

El enfoque que se toma en esta investigación es de tipo cualitativo ya que se constituye una idea de investigación (Transformación CIM a PIM: Obtención de Casos de Uso a partir de Procesos de Negocio Enriquecidos), la cual consiste en realizar una transformación de modelos con distintos niveles de abstracción de MDA, también proponer una variante al *plugin* DQ presentado en (Fuentes et al., 2015).

## 1.4. Contexto

Dentro del marco de DQ abordado en (Rodríguez and Caro, 2012), se propone una extensión a los modelos BP descritos con BPMN en la cual se considera agregar exactitud, completitud y oportunidad de los datos y un método para llegar a realizar una transformación de un modelo BP a un UCD. Todo esto situado en el contexto de BPiDQ*. Este método se toma como base para presentar BPiDQSec. Este nuevo método considera los aportes realizados en el ámbito de la seguridad por (Rodríguez et al., 2007). Con lo cual se propone una nueva extensión en la que se consideran los requisitos de seguridad en los modelos BP. Los requisitos de seguridad que se consideran son Attack Harm Detection, Integrity, Privacy , Access Control. En el caso de la calidad de datos, en esta nueva extensión, se realiza un manejo distinto de

la Completitud, Exactitud y Oportunidad, no se considerará el uso de patrones que son utilizados en la propuesta original, con el objeto de homologar con la seguridad su representación en los modelos de procesos de negocio. Patrones que para la facilidad de este proyecto se omiten. De esta manera al momento de realizar los modelos BP, la calidad de datos o la seguridad representada en los elementos se ve reflejada en el ID de estos con el siguiente formato "ID_Extensión" pudiendo agregar la extensión:

- DQEX: *Accuracy* (Exactitud)
- DQCO: *Completeness*
- DQOP: *Opportunity*
- SECAC: *Access Control*
- SECAD: *Attack Harm Detection*
- SECIN: *Integrity*
- SECPR: *Privacy*

Los modelos BP enriquecidos son considerados como el modelo de entrada para el método BPiDQSec, el cual tiene por objetivo aplicar reglas de transformación en ATL, con el fin de obtener UCD descrito con UML. La transformación propuesta están en el contexto de la propuesta MDA (ver Figura 1.1), en la cual se propone pasar del nivel CIM a PIM.

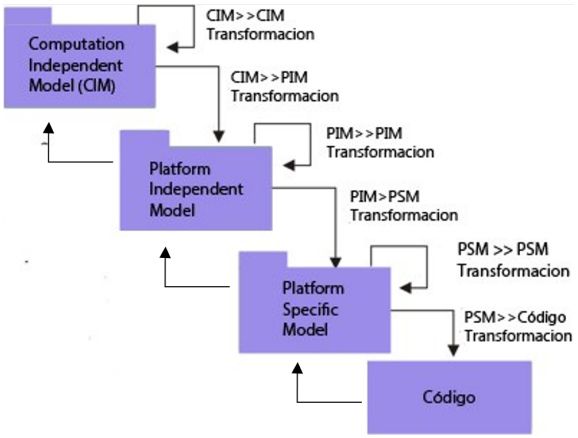


Figura 1.1: Niveles de MDA (Ordoñez et al., 2016)

MDA está compuesto por varios niveles. MDA provee los estándares y mecanismos para concebir una arquitectura alrededor de modelos y su proceso de transformación

(Ordoñez et al., 2016). Son estos procesos de transformación los cuales permiten pasar entre los distintos niveles que esta arquitectura presenta, con el fin de obtener modelos con mayor o menor grado de abstracción.

Los elementos de BPMN en los cuales se puede añadir la extensión de DQ o seguridad a la hora de realizar un modelo BP enriquecido son presentados en la Tabla 1.1 y Tabla 1.2 respectivamente. Para decidir a qué elementos es posible agregar alguna de las extensiones, se consideraron los presentados en (Rodríguez and Caro, 2012) y en (Rodríguez et al., 2007).

| **Elementos de Calidad de datos** | **Data Object** | **Data Store** | **Message** | **Task** |
|---|---|---|---|---|
| Accuracy | X | X | X | X |
| Completeness | X | X | X | X |
| Opportunity | X | X | X | X |

Tabla 1.1: Elementos BPMN que Soportan Calidad de Datos

| **Elementos de Seguridad** | **Data Object** | **lane** | **Pool** | **Task** |
|---|---|---|---|---|
| Access Control | | X | X | X |
| Attack Harm Detection | X | X | X | X |
| Integrity | X | | | |
| Privacy | | X | X | |

Tabla 1.2: Elementos de BPMN que Soportan Seguridad

En la Tabla 1.1 y Tabla 1.2 se presentan los elementos que soportan la extensión de DQ y/o Seguridad, los elementos que no se encuentran presentes en estas tablas no se extienden el dominio de DQ o seguridad para ellos.
Ahora se indicará la repercusión que tiene en los modelos BP el agregar característica de seguridad o DQ a los elementos de BPMN.

En cuanto a la calidad de datos que podemos agregar a los modelos BP tenemos:

- *Accuracy*: con este atributo de DQ podremos controlar el grado en que los datos tienen atributos correctos para un contexto específico de uso, debiendo cumplir con ser datos precisos, representar el estado real de la información y no causar ambigüedad a la hora de representar los datos, la exactitud se verificará para todos lo elementos de forma igual (Caro et al., 2013; Rodríguez and Caro, 2012).

- *Completeness*: ayuda a controlar que todos los datos contengan los valores necesarios para lograr una ejecución exitosa de un proceso de negocio, la verificación de esta cualidad será igual en todos los elementos.(Caro et al., 2013; Rodríguez and Caro, 2012).

- *Opportunity*: permite controlar la actualidad de los datos utilizados en el proceso de negocio, además de ayudar a saber su disponibilidad a la hora de solicitarlos, para todos los elementos se aplicara de la misma manera (Caro et al., 2013; Rodríguez and Caro, 2012).

En relación con los requisitos de seguridad que podemos agregar a los modelos BP se consideran los propuestos por (Rodríguez et al., 2007).

- *Access Control*: ayuda a evitar que terceros no autorizados en el sistema obtengan información sensible de los elementos que implementan este requisito.
- *AttackHarmDetection*: los elementos deben considerar un mecanismo para registrar y notificar un intento de ataque o ataque exitoso dentro del BP.
- *Integrity*: se preocupa de evitar la corrupción intencional y no autorizada del contenido de los elementos que implementen este requisito.
- *Privacy*: esta característica evita que terceros no autorizados obtengan información sensible sobre la identidad de los participantes del modelo BP.

# Capítulo 2

# Conceptos Relacionados

En este capítulo, se busca familiarizar al lector con los principales conceptos que son utilizados en esta memoria, exponiendo de manera clara y precisa de que trata cada uno de estos. A continuación, procederemos a indicar los conceptos (presentados en orden alfabético) que el autor considera clave.

## 2.1. ATL (Atlas Transformation Language)

ATL es un lenguaje de transformación de modelos desarrollado y mantenido por OBEO y AtlanMod (OBEO, 2014). Fue iniciado por el equipo ATLAS en el campo de la ingeniería basada en modelos. ATL proporciona formas de producir un modelo de destino a partir de un modelo de origen. La forma en que ATL funciona es tomando un modelo origen $Ma$ el cual se transforma en un modelo destino $Mb$, dicha transformación está dirigida por *mma2mmb.atl* (programa que realiza la transformación). ATL, $MMa$ y $MMb$ se ajustan a un metamodelo $KM3$ lo que es representado por la Figura 2.1. ATL es un lenguaje híbrido ya que contiene reglas declarativas e imperativas, además de destacar que ATL sólo permite transformaciones unidireccionales, lo que considera un modelo de origen que sólo se puede leer y un modelo destino que sólo se puede escribir (Jouault, 2006). En esta memoria ATL fue utilizado para describir las reglas que hicieron posible la transformación de los modelos BP enriquecidos con DQ y seguridad en UCD.

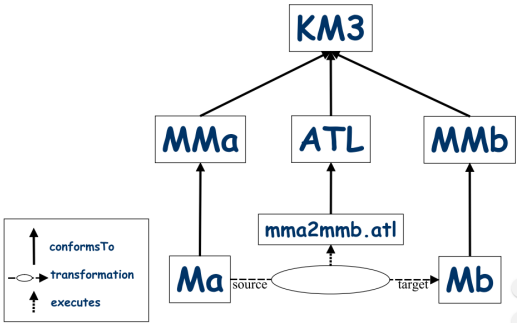


Figura 2.1: Composición de ATL (Jouault, 2006)

## 2.2. BP (Proceso de Negocio)

Un proceso de negocio es una colección coordinada de actividades diseñadas para producir una salida que proporciona valor al cliente (Sanchis et al., 2009), centrando un fuerte énfasis en cómo funciona el negocio de una organización. Los procesos de negocio se componen por actividades que son realizadas por actores que desempeñan roles particulares (Mili et al., 2004). Dichas actividades pueden vincularse con los actores que operan dentro de los límites de la organización. Los modelos BP enriquecidos son utilizados en esta memoria como un modelo de entrada para el sistema que finalmente convertirá dichos modelos en UCD.

## 2.3. BPMN (Business Process Model and Notation)

BPMN (Silver, 2011) es una notación gráfica que describe la lógica de los pasos de un BP, propuesta en el año 2004 por Business Process Modeling Iniciative. Su creciente popularidad entre las organizaciones provocó que fuera adoptada por la OMG (Object Management Group) en el año 2006. Esta notación ha sido especialmente diseñada para coordinar la secuencia de los procesos y los mensajes que fluyen entre los participantes de las diferentes actividades. BPMN proporciona un lenguaje común con el fin de facilitar la comprensión mediante una notación simple para los usuarios del negocio que permite representar modelos BP de manera sencilla. En sus últimas versiones BPMN formaliza una semántica de ejecución de todos los elementos que lo componen. También presenta un mecanismo de extensibilidad el cual permite realizar extensiones a los modelos BP (Chinosi and Trombetta, 2012). BPMN fue utilizado como el lenguaje para realizar los modelos BP enriquecidos. Los principales elementos de la notación de BPMN se presentan en la Figura 2.2.

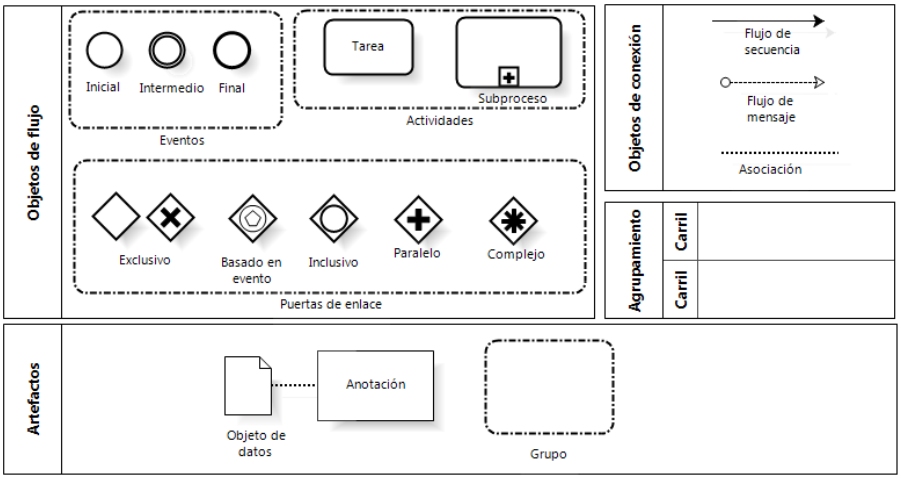


Figura 2.2: Notación Básica BPMN (Ciaramella et al., 2009)

## 2.4. MM (Metamodelo)

Los metamodelos se encuentran situados en el tercer nivel del estándar Meta Object Facility (Alanen et al., 2004) (MOF) promovido por la OMG. Los metamodelos ayudan a definir nuevos lenguajes de modelado para un dominio determinado. Con ellos se define la sintáxis abstracta de un lenguaje de modelado, también se definen los tipos que se pueden usar como objetos en un modelo por medio de clases similares a UML (Boronat and Meseguer, 2010). Adicionalmente se especifica que la base conceptual de un lenguaje de modelado se denomina metamodelo(Jouault et al., 2008). Consecuentemente se puede decir que el desarrollo de metamodelos contempla una disciplina en la cual se aplica análisis, construcción, desarrollo de esquemas, reglas, restricciones y teorías aplicables en un dominio en particular. En esta memoria se especifican los metamodelos de la extensión DQSec y el de UCD.

## 2.5. UCD (Diagrama de Casos de Uso)

Un UCD (Bittner, 2002) describe el comportamiento del sistema en las distintas situaciones en las que éste responde a una petición de alguno de sus participantes. En esencia, un diagrama de casos de uso describe cómo interactúa un usuario final (que puede tener varios roles) con el sistema en circunstancias específicas (Pressman and Troya, 1988). Los usuarios finales o entidades que se encuentran en un UCD se denominan actores y la forma que tienen estos actores para poder interactuar con el sistema es mediante las relaciones que se forman entre estos y los casos de uso, siendo estos últimos los que entregan una funcionalidad específica que deberá realizar

el sistema con el que se interactúa. La notación que se utiliza para crear los UCD es presentada en la Figura 2.3.

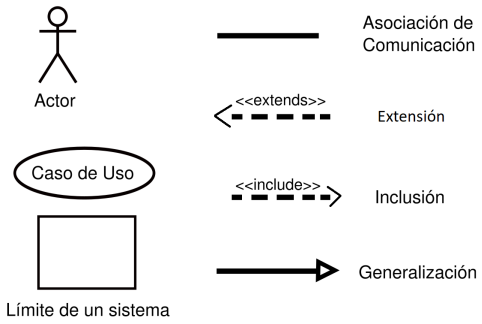


Figura 2.3: Notación de Casos de Uso

## 2.6. Transformación de Modelos

Las transformaciones de modelos consisten principalmente en pasar de uno o más modelos de origen a un modelo destino. Generalmente, las transformaciones son pasos de refinamiento generando modelos que disminuyen el nivel de abstracción (Jouault et al., 2008). A su vez existen distintos tipos de transformaciones.

- Transformaciones Endógenas: estas trasformaciones consideran el mismo metamodelo de origen y destino, permite trasformaciones de tipo model merging y model modification.

- Transformaciones Exógenas: estas trasformaciones consideran distintos metamodelos de origen y destino, esto permite realizar model transformation, model linking y model Marking.

En una transformación se dispone de distintos elementos y estos son los metamodelos, un lenguaje de transformación y el modelo a transformar. En los últimos años también se ha planteado realizar transformaciones que aumenten el nivel de abstracción. La mayoría de las transformaciones de modelos realizadas se consideran en un sentido, lo que quiere decir es que, si tomamos un modelo A, aplicamos las reglas de transformación se obtendrá un modelo B, pero no corresponde que tomando el modelo B y aplicando las mismas reglas se obtenga el modelo A.

## 2.7. UML (Unified Modeling Language)

UML es un lenguaje de modelado visual, que es utilizado para especificar, visualizar, construir y documentar los artefactos de un sistema de software (Booch, 1999). En general, los diagramas UML describen los límites, la estructura, el comportamiento del sistema y los objetos que contiene. UML es un conjunto de lenguajes que se utilizan para modelar clases, casos de uso, diagramas de actividad, etc. Aunque UML no es considerado un lenguaje de programación, hay herramientas que pueden llegar a convertir en código un modelo realizado con UML. Este lenguaje guarda una relación directa con el análisis y el diseño orientados a objetos.

## 2.8. XML (Extensible Markup Language)

XML (Bray et al., 1997) es un formato universal para datos y documentos estructurados. XML no requiere ni favorece una interfaz o clase de interfaces específica. XML presenta el conjunto de información como un árbol modificado en favor de la claridad y la simplicidad, pero no es un requisito que el conjunto de información esté disponible a través de esta estructura (Cowan and Megginson, 1999). Los archivos con extensión "xml" basan su desarrollo mediante etiquetas, para así poder estructurar los datos que contienen. XML es una tecnología relativamente sencilla, pero se rodea de tecnologías que la complementan y la hacen mucho más grande. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir información de una manera segura, fiable y fácil. En el proyecto, XML es usado debido a que los modelos generados con el nuevo *plugin* DQSec, se pueden exportar en formato XML lo cual hace de fácil lectura gracias a las librerías que Java dispone para esto, las especificaciones de la extensión en el código XML del modelo diseñado, se ven reflejadas mediante los ID's de los elementos de BPMN, los cuales quedan de la forma "ID-Extension".

# Capítulo 3

# Tecnologías Utilizadas

En este capítulo se presentará (en orden alfabético) las funcionalidades principales de las herramientas o tecnologías utilizadas en el proyecto.

## 3.1. BPMN 2.0 Modeler

Eclipse BPMN 2.0 Modeler (BPMN, 2011) es una herramienta de modelado gráfico para la creación de modelos BP. El objetivo principal de BPMN 2.0 Modeler es proporcionar un marco de edición de flujo de trabajo gráfico que se puede personalizar fácilmente para cualquier motor de ejecución compatible con BPMN 2.0. Esta herramienta fue utilizada para el desarrollo del *plugin* DQ proporcionado en (Fuentes et al., 2015). Por lo tanto, BPMN 2.0 Modeler se utilizó para poder extender el dominio de seguridad en la herramienta propuesta anteriormente. BPMN 2.0 Modelar utiliza EMF (Eclipse Modeling Framework). Este fue utilizado en este trabajo para realizar los metamodelos de BP enriquecidos y de UC.

## 3.2. Eclipse

Eclipse (Eclipse, 2007) es una herramienta de software desarrollada por IBM, éste es un entorno de desarrollo que es utilizado específicamente para poder trabajar con BPMN 2.0 Modeler, editar el *plugin* DQ, utilizar EMF para generar los metamodelos necesario para el proyecto, ATL para realizar las reglas de transformación entre modelos y Java para realizar el programa que tome el XML de la salida de las transformaciones y crea un archivo reconocible para una representación gráfica de los UC cuyo formato es Graph Modelling Language (GML). Las utilidades presentadas por Eclipse en este proyecto se ven reflejadas en la Figura 3.1.

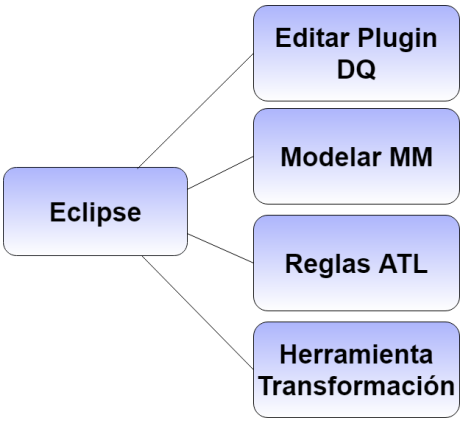


Figura 3.1: Utilidades de Eclipse en el Proyecto

## 3.3. GML (Graph Modelling Language)

GML (Himsolt, 1997) es un formato de archivos jerárquicos basados en ASCII, utilizado especialmente para representaciones gráficas considerado como un formato portátil. La característica clave de GML consiste en una lista jerárquica con una fácil representación; un ejemplo de esto lo podemos apreciar en la Figura 3.2, este cuenta con un ID el cual nos ayudara a identificar el elemento, el Label es el nombre que recibe este elemento en este caso "Actor 2", las posiciones $x$ e $y$ representan la posición del elemento, $w$ y $h$ representan el alto y el ancho que tendrá el elemento, Image indica la ruta donde está el actor. Este lenguaje es utilizado debido a su compatibilidad con yEd Graph lo que facilita la representación de los UCD.

```
node
[
    id  0
    label   "Actor 2"
    graphics
    [
        x   374.0
        y   321.0
        w   60.0
        h   100.0
        image   "file:/C:/Users/PC-Leonel/Desktop/yEd/img/UC1.png
        fill    "#CCCCFF"
        outline "#000000"
    ]
]
```

Figura 3.2: Representación de un Elemento con GML

## 3.4. JAVA

Java (Gosling et al., 2014) es un lenguaje de programación y plataforma informática, ese fue comercializado por primera vez en el año 1995 por Sun Microsystems. Se centra en el paradigma orientado a objetos. El lenguaje Java es utilizado en el proyecto para crear las clases con las cuales se podrán extender el nuevo dominio de seguridad. Apoyar el desarrollo de la aplicación la cual convierte la representación XML de los modelos generados con el *plugin* DQS en XML compatible con las reglas descritas en ATL, una vez aplicada las reglas se genera un archivo XML el cual es una representación de los UC, en este caso se utiliza el Java para convertir este XML en un archivo GML el cual podrá ser interpretado por yEd Graph Editor.

## 3.5. yEd Graph Editor

yEd Graph Editor (yWorks, 2014) es un editor gráfico con el cual podemos realizar una gran cantidad de modelos dentro de los que se encuentran diagramas de flujo, diagramas de organización, modelos con BPMN, modelos con UML, entre otros. Este editor es utilizado debido a la facilidad con la que se pueden representar los UCD mediante un archivo con extensión GML.

# Capítulo 4

# Propuesta

En este capítulo se presenta el método BPiDQSec, metamodelos propuestos, destacando la inclusión de la extensión de seguridad y DQ al metamodelo de BPMN, las modificaciones realizadas al *plugin* DQ para generar el *plugin* DQSec, indicando tanto los elementos gráficos agregados y la explicación de las secciones más relevantes de las clases generadas. Además se presentan transformaciones que se realizaron como parte de este proyecto, de igual manera se presentará el modelo BP y la representación UCD de cada uno de estos. Finalmente se presenta la herramienta que permite implementar el método BPiDQSec.

## 4.1. Metodología BPiDQSec

El método BPiDQSec se basa en BPiDQ* (Rodríguez and Caro, 2012). BPiDQSec consta de 4 etapas (ver Figura 4.1 ).

- Agregar la capacidad para representar requisitos de seguridad en los modelos BP. En esta etapa se realizan las modificaciones a nivel de código para agregar la capacidad de representar seguridad en el *plugin* DQ (Fuentes et al., 2015).

- Modelado de procesos de negocio consciente de DQ y seguridad. En esta etapa el modelador se encarga de una temprana captura de requisitos de DQ y seguridad. Dando como resultado un modelo BP enriquecido que denotan el interés del modelador por profundizar en los requisitos que son importantes para un buen desempeño del BP.

- Definición de reglas de transformación de modelos en ATL. Aquí se definen todas las reglas necesarias para convertir el modelo BP a una representación en XML del UCD.

- Generación de UCD. En esta última etapa se toma el XML del UCD y se convierte en una representación en GML, la cual podrá ser interpretada por yEd graph para generar los UCD descritos con UML.

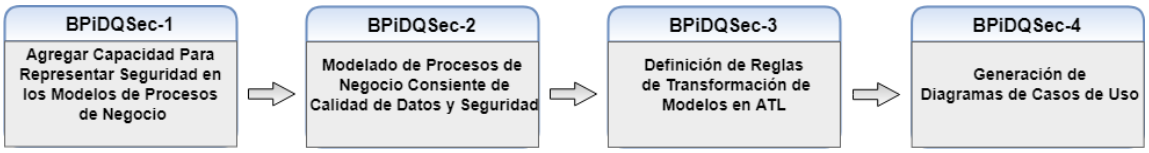


Figura 4.1: Proceso BPiDQSec

## 4.2. Metamodelos

En la Figura 4.2 se presenta el metamodelo propuesto para satisfacer tanto la calidad de datos como la seguridad dentro de los modelos BP descritos con BPMN. Se puede apreciar en este metamodelo un elemento central denominado *BusinessProcessDiagram* el cual cuenta con una relación de composición con todos los elementos que lo rodean. Estos elementos son *Artifacts*, *Connection Event*, *Flow Object* y *Swimlanes*. También se agrega la extensión de DQ y seguridad mediante *Extension*, quien al igual que los elementos anteriores se encuentra relacionado con el elemento central mediante una relación de composición. *Extension* ayuda a representar *Accuracy*, *Completeness* y *Opportunity* en el contexto de DQ, en el caso de seguridad representa *Access Control*, *Attack Harm Detection*, *Privacy* e *Integrity*. Para lograr implementar este metamodelo se tomó la idea de los metamodelos de seguridad y DQ propuestos en (Rodríguez et al., 2007) y (Rodríguez and Caro, 2012) respectivamente.

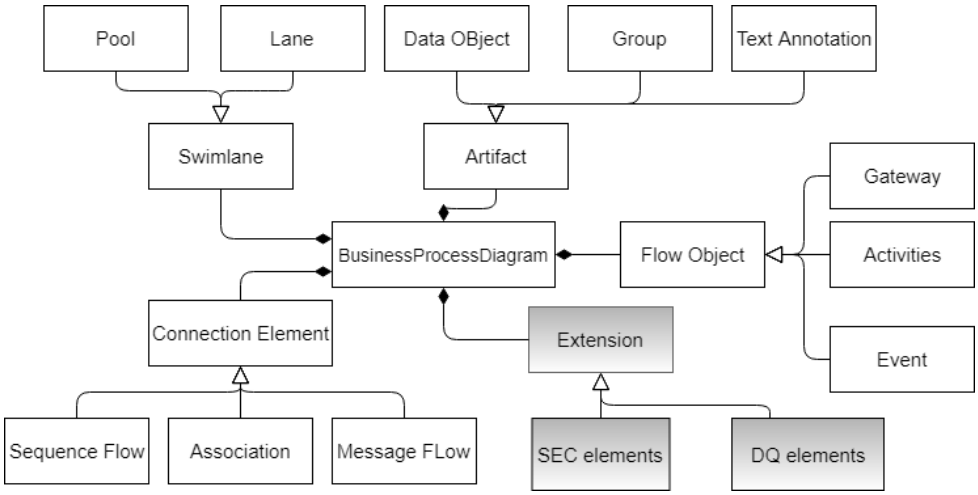


Figura 4.2: Metamodelo Extensión DQSec

También se presenta el metamodelo de los casos de uso (ver Figura 4.3). Para este caso se consideró un modelo básico de casos de uso, de manera similar al anterior en el metamodelo planteado tenemos un elemento central llamado *UseCaseMode*, el cual

se relaciona mediante una composición con *UseCase*, este se relaciona de la misma manera con Actor. En cuanto a los casos de usos extendidos e incluidos consideramos una relación de *UseCase* a *UseCase* y para los actores generalizados se considera una relación de *Actor* a *Actor*.

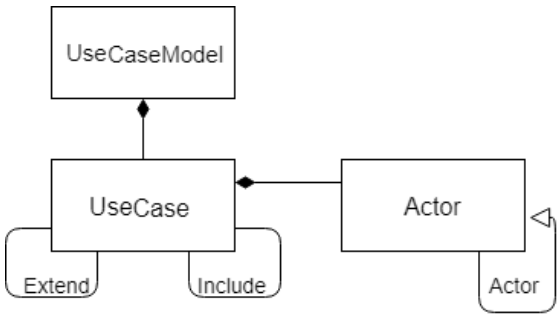


Figura 4.3: Metamodelo Casos de Uso

## 4.3. Modificaciones al Plugin DQ

Para desarrollar el nuevo *plugin* DQSec, se toma como base las clases y lo expuesto por (Fuentes et al., 2015), para así lograr implementación de manera eficiente de la extensión de seguridad a los modelos BP. Con el fin de lograr mayor facilidad a la hora de identificar los distintos requisitos de seguridad y DQ que pudiera solicitar el usuario que utiliza la herramienta de modelado, se presentan una serie de figuras las que ayudan a identificar de manera fácil qué extensión se está agregando al elemento. Esto se logra creando elementos para cada ámbito de DQ y seguridad. Las figuras propuestas son presentadas en la Tabla 4.1 y Tabla 4.2.

| **Security Elements** | **Representación Gráfica** |
|---|---|
| Access Control |  |
| Attack Harm Detection |  |
| Integrity |  |
| Privacy |  |

Tabla 4.1: Elementos Sec Agregados

| DQ Elements | Representación Gráfica |
|---|---|
| Accuracy | |
| Completeness | |
| Opportunity | |

Tabla 4.2: Elementos DQ Agregados

Una vez identificados los elementos gráficos necesarios para poder crear el nuevo *plugin* DQSec, se explica cuáles fueron las principales modificaciones a nivel de código que se realizaron al *plugin* DQ. Al igual que en la extensión DQ se modifica la clase *ShapeDecoratorUtil*, que es la clase encargada de proporcionar las distintas opciones que son desplegadas al momento de posar el cursor sobre un elemento del modelo BP. Un ejemplo de esto se puede apreciar en la Figura 4.4. En este ejemplo se puede observar cómo se verán las marcas que pueden ser agregadas por el usuario (la distribución de éstas dentro del menú no son controladas por el desarrollador).

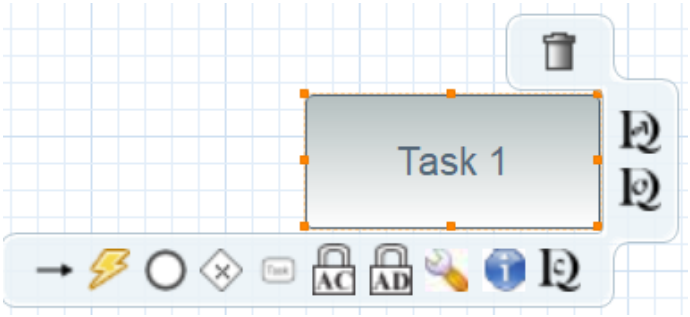


Figura 4.4: Opciones Presentadas en Activity

Para agregar las marcas presentadas anteriormente (Ver Tabla 4.1 y Tabla 4.2) es necesario crear una clase para cada una de ellas (ver Tabla 4.3). Cada una de estas clases toman de base *ShowDQBPFeature* propuesta en (Fuentes et al., 2015), esta clase además de agregar la marca solicitada por el usuario se encarga de modificar el ID de los elementos del modelo BP que ahora contengan alguna de las extensiones. Dentro de las novedades que propone el *plugin* DQSec es permitir agregar más de un elemento de extensión a los distintos componentes del modelo BP.

| Clases |
|---|
| ShowDQBPFeatureAC (para representar Calidad, Accuracy) |
| ShowDQBPFeatureCO (para representar Calidad, Completeness) |
| ShowDQBPFeatureOP (para representar Calidad, Opportunity) |
| ShowSecFeatureAC (para representar Seguridad, Access Control) |
| ShowSecFeatureAD (para representar Seguridad, Attack Harm Detection) |
| ShowSecFeatureIN (para representar Seguridad, Integridad) |
| ShowSecFeatureNR (para representar Seguridad, Non Repudiation) |
| ShowSecFeaturePR (para representar Seguridad, privacidad) |

Tabla 4.3: Clases Agregadas al Plugin DQSec

En el *plugin* DQ (Fuentes et al., 2015) al momento de agregar una marca a los componentes del modelo BP, se necesita la clase *ShowDQBPFeature*, ésta solicita la imagen/marca a *ImageProvider* para luego llamar dentro de *ShowDQBPFeature* a las funciones de la clase *ShapeDecoratorUtil*, las cuales agregan los componentes gráficos dentro de los elementos del modelo BP.

Luego de la implementación de las clases señaladas en la Tabla 4.3, podemos apreciar un menú que contiene varios elementos de extensión para agregar, a diferencia del *plugin* DQ el cual sólo permite agregar una marca. Un ejemplo de las marcas que se pueden agregar a las actividades dentro de los modelos BP se muestra en la Figura 4.5.

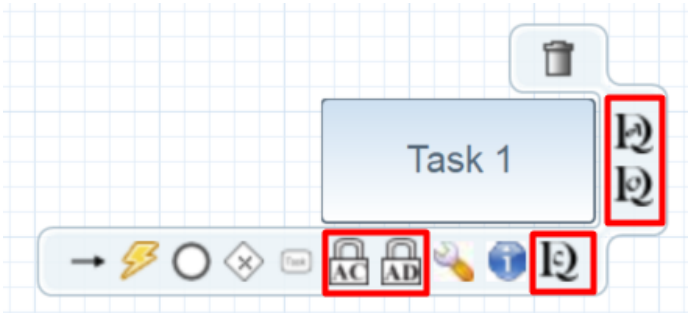


Figura 4.5: Elementos Disponibles en Activity

También se proporciona la opción de agregar las marcas en el menú que despliega el elemento a la hora de hacer click derecho sobre él (ver Figura 4.6). Estas opciones se pueden agregar gracias a la clase *Bpmn2ToolBehaviorProvider*. Una vez comprendidas las clases principales que intervienen en el desarrollo del *plugin* DQ se procede a la implementación de las clases que son utilizadas para generar el *plugin* DQSec.

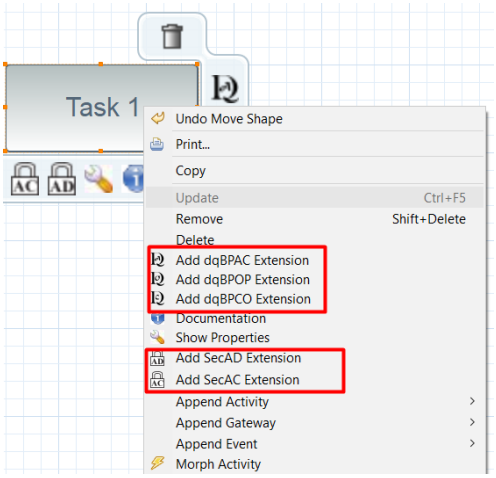


Figura 4.6: Menú Desplegado con Doble Click

Junto con la implementación de las clases que hacen posible agregar las marcas a los elementos del modelo BP descrito con BPMN (ver Tabla 4.2 y Tabla 4.1), se agregan las respectivas clases para eliminar cada una de las marcas gráficas presentadas. Las clases que cumplen la función de eliminar las marcas deben tener varias restricciones, si queremos eliminar una marca y el elemento contiene más de una, se debe eliminar el elemento exacto y modificar el ID de tal manera que el modelo visto por el usuario sea consistente con las acciones realizadas sobre él. En cuanto a las clases generadas para lograr la modificación al *plugin* DQ, se presentará el funcionamiento general de dos de ellas (ver Figura 4.7). Una sirve para agregar elementos de DQ y la otra agrega elementos de seguridad, el resto de las clases tienen funcionalidades similares.

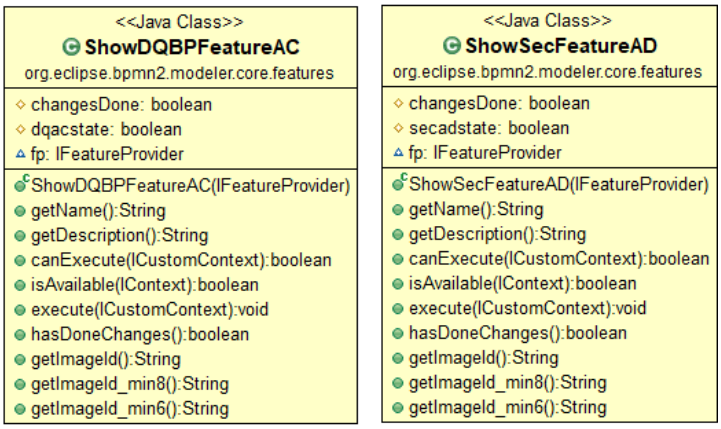


Figura 4.7: Clases para Agregar Elementos en Plugin DQSec

La manera con la cual se procederá a explicar las clases, es mediante descripción del código fuente de cada uno de las funciones y métodos que estas presentan, también se presenta las variaciones de estas clases entre sí.

Se comienza explicando las funciones encargadas de solicitar las distintas marcas gráficas (ver Código Fuente 4.8). estas devuelven elementos de la clase *Iconstants*, los elementos que se solicitan en esta función son agregados anteriormente para no tener problemas a la hora de ocuparlos dentro de la herramienta. Se puede apreciar que la diferencia sustancial de ambas secciones de código tiene que ver con un dominio en particular. De esta forma, las marcas retornadas tendrán relación con la especificación de seguridad y/o DQ indicadas. BPSECAD retorna las marcas correspondientes a *Attack Harm Detection* y DQBPAC hace referencia a los elementos de *Completeness*, ambas clases constan de 3 funciones las cuales se encarga de solicitar marcas de distintos tamaños, esto se debe a que hay elementos que necesitan marcas más pequeñas. Esta es una sección de código que comparten todas las clases implementadas con la particularidad de pedir las marcas correspondientes.

Figura 4.8: getImageId()

```
1  DQ
2          public String getImageId() {
3                  return IConstants.ICON_BPSECAD_16;
4          }
5          public String getImageId_min8() {
6                  return IConstants.ICON_BPSECAD_8;
7          }
8          public String getImageId_min6() {
9                  return IConstants.ICON_BPSECAD_6;
10         }
11 Seguridad
12         public String getImageId() {
13                 return IConstants.ICON_DQBPAC_16;
14         }
15         public String getImageId_min8() {
16                 return IConstants.ICON_DQBPAC_8;
17         }
18         public String getImageId_min6() {
19                 return IConstants.ICON_DQBPAC_6;
20         }
```

Para conocer qué elemento soportará una marca en particular se proporciona la clase *isAvailable()*, la cual se comunica con la clase *Bpmn2ToolBehaviorProvider* presentada en (Fuentes et al., 2015). Esta clase indica qué opciones se despliegan en los elementos, por lo tanto, es aquí donde debemos agregar la validación a los componentes del modelo BP. Podemos ver que en la línea 7 de la Figura 4.9 y la Figura 4.10 se verifica si alguna de estas instancias de los elementos de BPMN contienen una marca, en este caso en particular *Acuracy* y *Attack Harm Detection*. Esto lo sabemos debido a que en la línea 8 se pregunta si el ID de estos elementos contiene DQAC o SECAD (palabra agregado al ID en caso que ya tenga la extensión). Si este elemento no contiene unas de estas palabras en su ID, es candidato para agregar la marca. Las particularidades que presentan el resto de las clases implementadas, es

que en la línea 7 consulta por elementos distintos y en la línea 8 pregunta por otra palabra clave contenida en el ID.

Figura 4.9: isAvailable() DQ

```
1  public boolean isAvailable(IContext context) {
2     if ((context instanceof ICustomContext)) {
3       PictogramElement[] pes=((ICustomContext)context).getPictogramElements();
4       DiagramEditor editor = (DiagramEditor) getDiagramBehavior().
            getDiagramContainer();
5       editor.setPictogramElementForSelection(pes[0]);
6       EObject obj = BusinessObjectUtil.getBusinessObjectForPictogramElement(pes[0]);
7       if (((obj instanceof Message)) || ((obj instanceof DataStoreReference)) || ((
            obj instanceof DataObject)) || ((obj instanceof Task)) || ((obj instanceof
             MessageFlow)) || ((obj instanceof Conversation)) || ((obj instanceof
            DataInput)) || ((obj instanceof DataOutput))) {
8           if (!((BaseElement) obj).getId().contains("_DQAC")) {
9               return true;
10          }
11      }
12      return false;
13    }
14    return false;
15 }
```

Figura 4.10: isAvailable() Sec

```
1  public boolean isAvailable(IContext context) {
2     if ((context instanceof ICustomContext)) {
3       PictogramElement[] pes = ((ICustomContext) context).getPictogramElements();
4       DiagramEditor editor = (DiagramEditor) getDiagramBehavior().getDiagramContainer
            ();
5       editor.setPictogramElementForSelection(pes[0]);
6       EObject obj = BusinessObjectUtil.getBusinessObjectForPictogramElement(pes[0]);
7       if (((obj instanceof Group)) || ((obj instanceof DataObject)) || ((obj
            instanceof Task)) || ((obj instanceof MessageFlow)) || ((obj instanceof
            Lane)) || ((obj instanceof Participant ))) {
8         if (!((BaseElement) obj).getId().contains("_SECAD")) {
9           return true;
10        }
11      }
12      return false;
13    }
14    return false;}
```

Sabiendo si los elementos del modelo BP tienen la capacidad de soportar la extensión de DQ o seguridad, se procede a realizar la inserción de la marca. Para esto utilizamos el método *execute()* (Ver Figura 4.11 y Figura 4.12), en el cual pregunta si el estado del *plugin* está habilitado con el fin de ofrecer la funcionalidad de agregar las marcas. Si el elemento no ha sido intervenido anteriormente se pregunta si éste en particular contiene dentro de su ID alguna de las palabras claves que hacen referencia a las extensiones. Sí dicho elemento no contiene palabras claves, se puede agrega el elemento gráfico con la función *createImageDQ / createImageSEC*. Estas clases pertenecen a *ShapeDecoratorUtil*,en el caso en que esta no sea la primera marca agregada al elemento, se debe adjuntar la nueva palabra clave al ID y agregando la nueva marca dentro del elemento del modelo BP.

Figura 4.11: Execute() DQ

```
1  public void execute(final ICustomContext context) {
2    TransactionalEditingDomain domain = getDiagramEditor().getEditingDomain();
3    domain.getCommandStack().execute(new RecordingCommand(domain) {
4    protected void doExecute() {
5    PictogramElement pes = context.getPictogramElements()[0];
6    DiagramEditor editor = (DiagramEditor) ShowDQBPFeatureAC.this.getDiagramBehavior
         ().getDiagramContainer();
7    editor.setPictogramElementForSelection(pes);
8    EObject businessObject = BusinessObjectUtil.getBusinessObjectForPictogramElement(
         pes);
9    if (!ShowDQBPFeatureAC.this.dqacstate) {
10     ontainerShape x = (ContainerShape) ShowDQBPFeatureAC.this.fp .
           getPictogramElementForBusinessObject(businessObject);
11     if ((businessObject instanceof Message)) {
12       Message r = (Message) businessObject;
13       if (!r.getId().contains("DQAC")) {
14         r.setId(r.getId() + "_DQAC");
15         int cont =1;
16         if (r.getId().contains("DQCO")) {
17                   cont = cont +6;}
18         if (r.getId().contains("DQOP")) {
19                   cont= cont +6; }
20                 ShapeDecoratorUtil .createImageDQ( ShowDQBPFeatureAC.this.fp.
                       getPictogramElementForBusinessObject(businessObject) .
                       getGraphicsAlgorithm(), ShowDQBPFeatureAC.this.getImageId_min6
                       (), 12, cont, 6); }
21                  }
```

Figura 4.12: Execute() Sec

```
1 if((businessObject instanceof Lane)) {
2    Lane r = (Lane) businessObject;
3    if(!r.getId().contains("_SECAD")) {
4      r.setId(r.getId() + "_SECAD");
5      int cont=2;
6      if(r.getId().contains("_SECAC")) {
7            cont = cont+17;}
8          if(r.getId().contains("_SECPR")) {
9            cont = cont+17;}       ShapeDecoratorUtil.createImageSec(x.
                 getGraphicsAlgorithm(), ShowSecFeatureAD.this.getImageId(), cont, 2)
                 ;}
10           }
```

Estas son las principales funcionalidades de las clases presentadas para realizar modificación del *plugin* DQ y obtener el *plugin* DQSec. La mayoría de las clases se ven diferenciadas en los elementos que solicitan para agregar la extensión DQ o de seguridad presentadas en la Tabla 4.1 y Tabla 4.2. Otro punto que considerar entre las clases generadas son las modificaciones realizadas para soportar varios marcas gráficas dentro de los elementos del modelo BP.

## 4.4. Transformaciones

En esta sección se presentarán dos ejemplos de transformación de modelos. En la Tabla 4.4 y Tabla 4.5 se muestran los elementos que pueden contener algún requisito de seguridad o DQ.

| **Elementos de Seguridad** | **Data Object** | **Lane** | **Pool** | **Task** |
|---|---|---|---|---|
| Access Control | | | | X |
| Attack Harm Detection | | | X | X |
| Integrity | X | | | |
| Privacy | | X | X | |

Tabla 4.4: Implementación de Transformaciones Sec

| **Elementos de Calidad de Datos** | **Data Store** | **Message** | **Task** |
|---|---|---|---|
| accuracy | X | | |
| completeness | | | X |
| opportunity | | X | |

Tabla 4.5: Implementación de Transformaciones DQ

Como primera transformación presentada se tiene un modelo BP enriquecido con *Access Control* agregado en una activity, específicamente en "Procesar Solicitud", el modelo presenta *Privacy* en el *Pool* "Banco" y el *Lane* "Usuario", Se comprobará la exactitud de una base de datos denominada "Data Store 1" y finalmente se considera *Integrity* en un *Data Object*, el cual en el modelo se denomina "solicitudes". Las tareas, eventos de inicio y eventos de finalización, no necesitan gran explicación, debido a que las tareas normales (sin extensión DQ o seguridad), se transforman en un caso de uso sin considerar casos especiales y tanto los eventos de iniciación como de finalización no se consideran en la transformación. Lo descrito anteriormente es representado en la Figura 4.13.

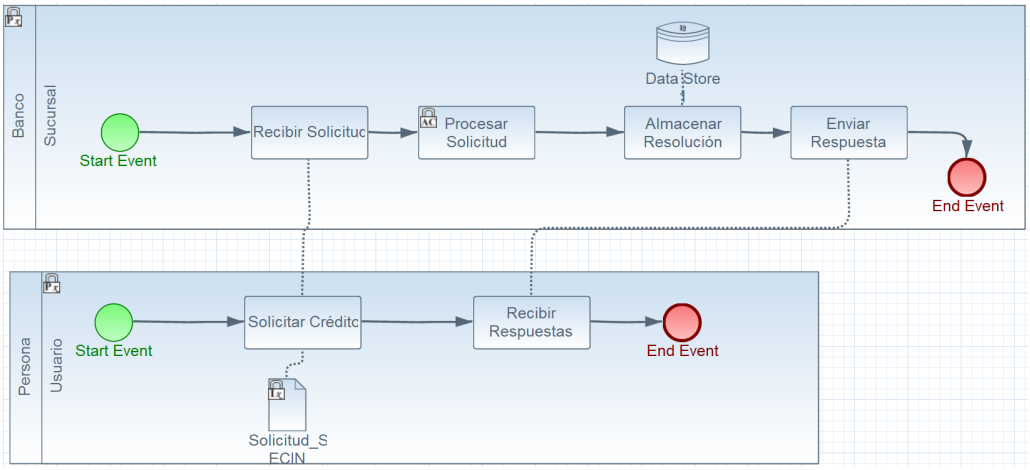


Figura 4.13: Modelo de Procesos de Negocio Caso 1

El UCD obtenido se presenta en la Figura 4.14, se aprecia que todas las tareas en el modelo BP se convirtieron en casos de uso, los *Pools* y *Lanes* presentes se convierten en actores del UCD. Para el caso particular de "Banco" el cual contiene un requisito de *privacy*, se le asocia una tarea la cual verifica esta cualidad, para el *Lane* "Usuario" que presenta un requisito de *Privacy* se extiende una tarea que verifica la privacidad de todas las tareas que este *Lane* realiza, en relación a la tarea que contiene el *Access Control* se relaciona mediante una inclusión con un caso de uso que verifica si el usuario que realiza esta tarea cumple con *Access Control* necesario para "Procesar Solicitud". De una manera similar se verifica la cualidad de *Access control* que se debería tener a la hora "Almacenar Resolución" y Para finalizar, el caso de uso "Verificar Integridad de Solicitud" se extiende a "Solicitar Crédito".

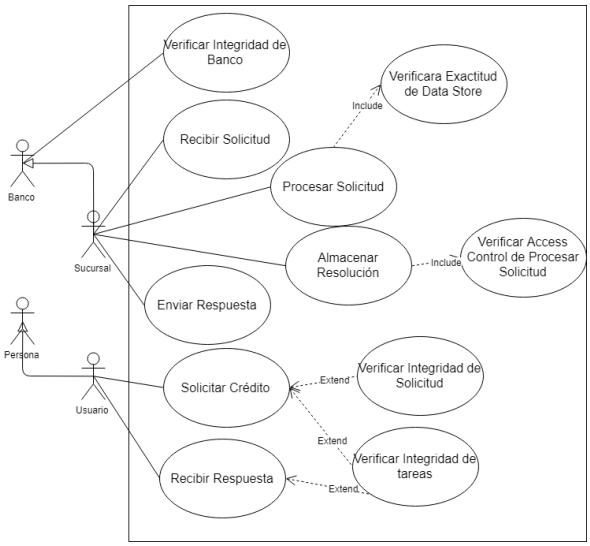


Figura 4.14: Modelo de Casos de Uso Transformado Caso 1

El segundo modelo BP, contiene un requisito de *Attack Harm Detection* en el *Pool* llamado "Tienda". Se agrega *Attack Harm Detection* y *Completeness* en las tareas "Verificar Pago" y "Enviar Cotización" respectivamente. Esas son las consideraciones en relación con las extensiones implementadas en este modelo BP. Al igual que en la situación anterior una representación de lo descrito se aprecia en la Figura 4.15.

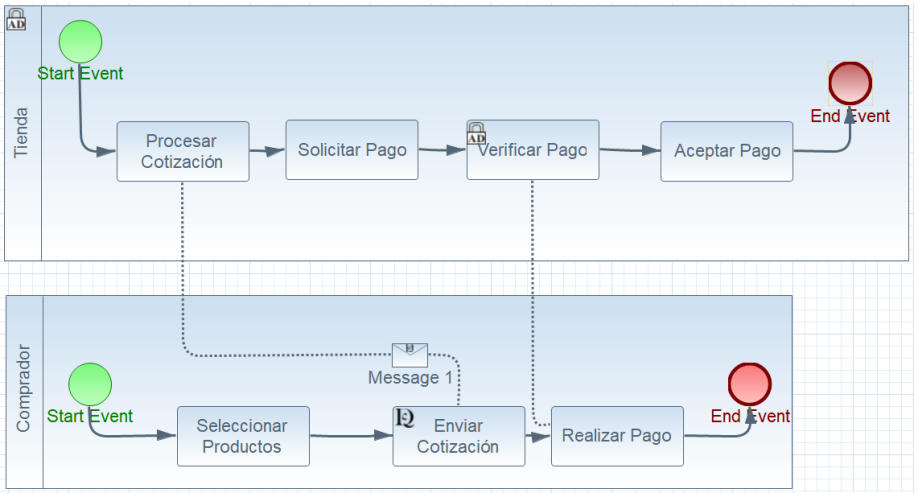


Figura 4.15: Modelo de Procesos de Negocio Caso 2

A modo de explicación de las transformaciones realizadas para generar el UCD de la Figura 4.16, se indican los pasos más importantes, que para este caso son, la definición de una marca que representa *Attack Harm Detection* en el *Pool* "Tienda", en esta situación se generará un nuevo caso de uso que se asocia con el actor "Tienda", al cual se le agrega la capacidad de notificar y registrar si sufrió un ataque, en relación con el mensaje que se presenta entre "Enviar Cotización" y "Procesar Cotización", ambos deben verificar *Opportunity* de este mensaje. Para finalizar "Verificar Pago" se relaciona mediante una extensión con un caso de uso que agrega la posibilidad de recibir notificaciones y registrar ataques sufridos en el proceso de realización de esta tarea y "Enviar Cotización" se relaciona mediante una inclusión con un caso de uso que verifica la oportunidad de los datos del mensaje enviado.

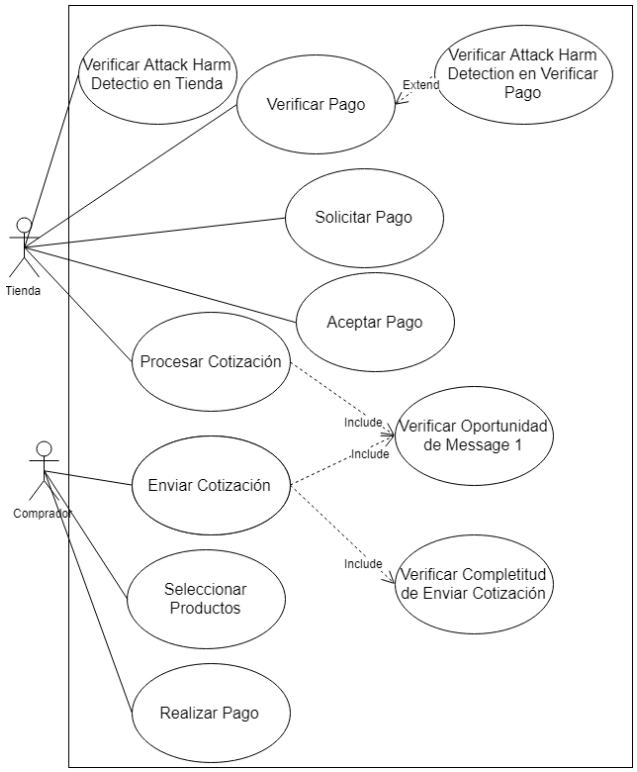


Figura 4.16: Modelo de Casos de Uso Transformado Caso 2

## 4.5. Herramienta BPiDQSec

En esta sección se presentará la manera como funcionan las transformaciones dentro de la herramienta BPiDQSec, especificando los pasos que esta sigue, para luego presentar los principales aspectos gráficos de la herramienta.

La manera en que funcionan las transformaciones para pasar de modelos BP a UCD es:

- Generar el modelo BP descrito con BPMN, lo cual se realiza con el *plugin* DQSec el cual tendrá una extensión "bpmn".
- Realizar las transformaciones de este modelo en un archivo XML el cual sea compatible con las reglas propuestas para la transformación.
- Aplicar las reglas de transformación descritas con ATL al archivo XML que representa el modelo BP, con el fin de obtener un archivo XML que represente un UCD.
- Transformar el XML que representa al UCD, para convertirlo en su representación en lenguaje GML.

Una representación gráfica de los pasos utilizados para realizar las transformaciones se presenta en la Figura 4.17. Donde el archivo de entrada "BPMN" es la representación en XML del modelo BP enriquecido generado por el *plugin* DQSec y el Archivo de Salida "GML" es la representación de los UCD que será interpretada por yEd graph para mostrar el UCD descrito con UML.

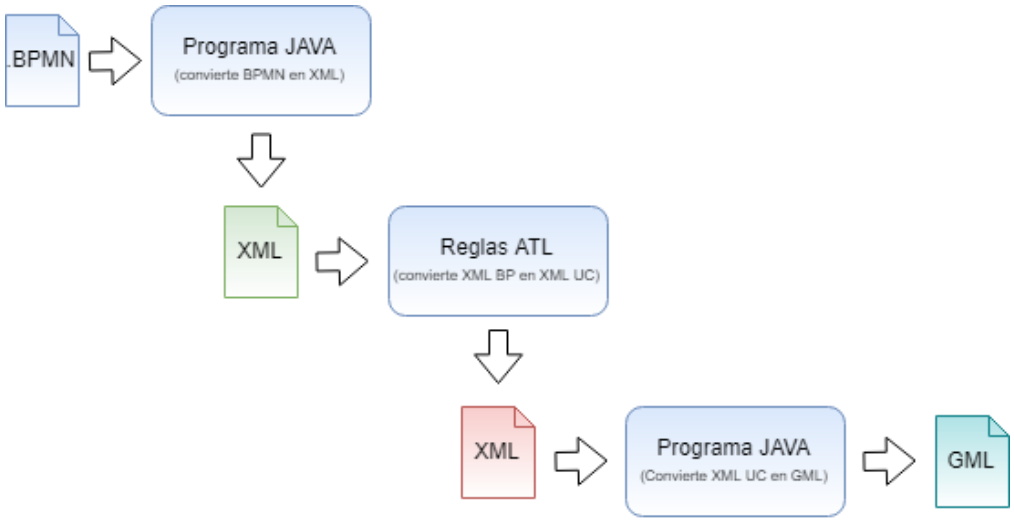


Figura 4.17: Proceso de Transformación

Los programas que intervienen en el proceso presentado en la Figura 4.17, se unifica generando la herramienta BPiDQSec (Ver Figura 4.18). Se puede apreciar que esta herramienta cuenta de un botón denominado "Seleccionar Archivo" este se encarga de seleccionar el archivo "bpmn"que se transformará.

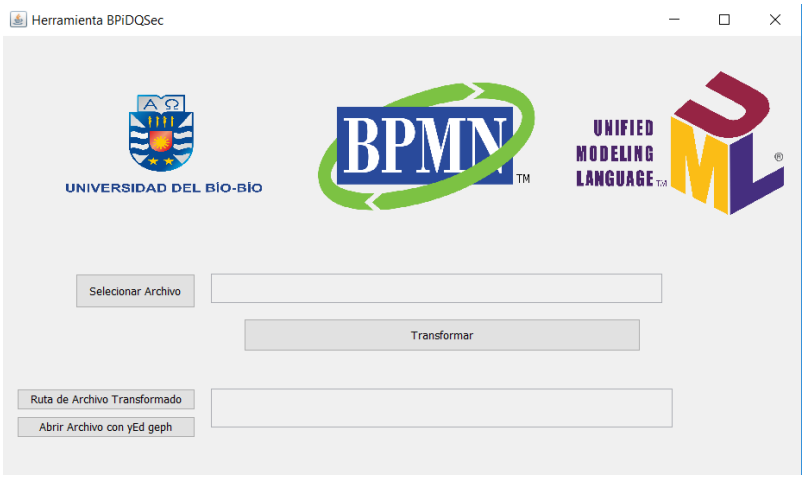


Figura 4.18: Herramienta BpiDQSec

Una vez seleccionada la ruta donde se encuentra el archivo que se quiere convertir, esta aparecerá en el recuadro de texto dispuesto para esto (ver Figura 4.19), se debe presionar el botón llamado "transformar" el cual se encarga de ejecutar los distintos pasos presentados en la figura 4.17.

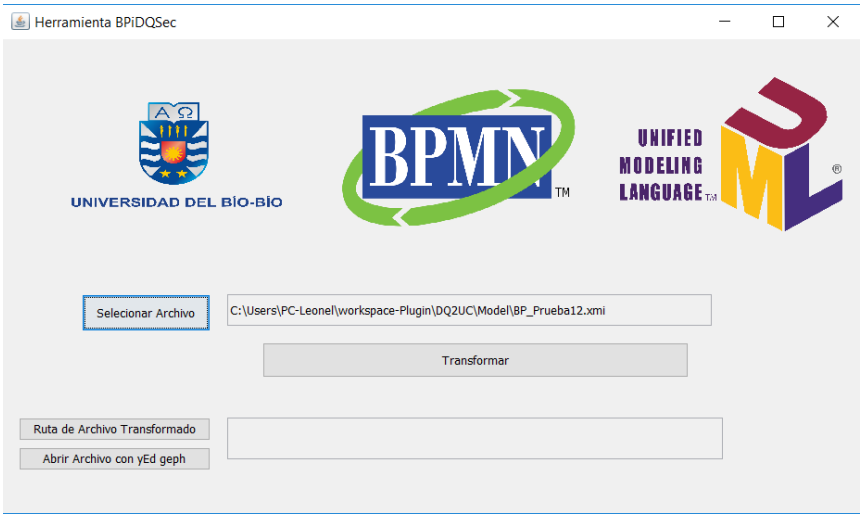


Figura 4.19: Herramienta BpiDQSec

Una vez completado el proceso de transformación se dispondrá de la ruta del archivo generado en el segundo cuadro de texto (ver Figura 4.20), presentando 2 botones. Uno llamado "Ruta de Archivo Transformado" el cual nos permite abrir la ruta donde se encuentra el archivo generado por la herramienta y el botón "Abrir archivo con yEd graph" el cual abrirá el archivo con el programa yEd graph el cual mostrará una representación en UML del UCD.

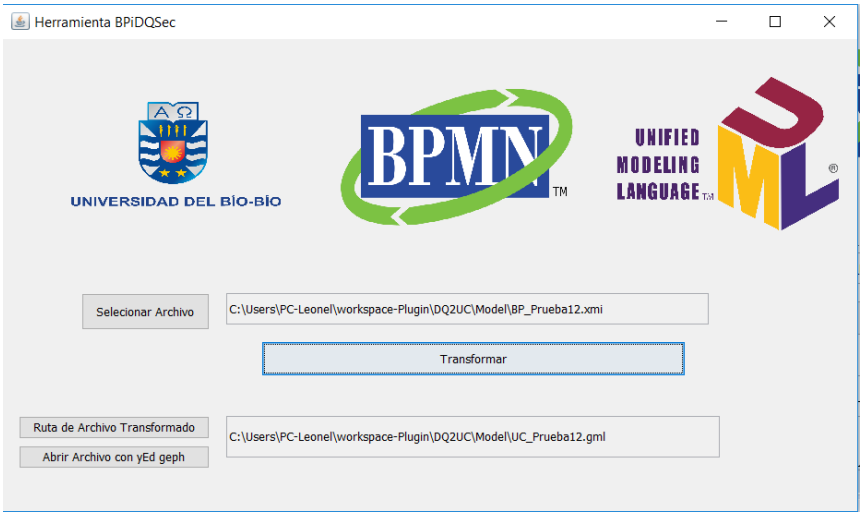


Figura 4.20: Herramienta BpiDQSec

# Capítulo 5

# Resultados

Este capítulo se divide en dos secciones. En la Sección 5.1 se muestran los resultados obtenidos mediante el *plugin* DQSec que nos permitirá crear modelos BP enriquecidos, mostrando mediante imágenes los resultados obtenidos a la hora ejecutar el *plugin* DQSec y el *plugin* DQ. En la Sección 5.2 se muestran los resultados obtenidos mediante la realización de las transformaciones, en esta sección se presentarán, además, los resultados mediante explicación de código fuente, dentro del cual se encuentran los modelos XML generados, las reglas que se utilizan a la hora de realizar una transformación y la representación gráfica y en lenguaje GML del UCD generado.

## 5.1. Plugin DQSec

La herramienta DQSec presentada en este trabajo esta basada en la herramienta propuesta por (Fuentes et al., 2015). Esta nueva herramienta incluye el dominio de seguridad. A continuación, se presenta las diferencias entre ambas herramientas desde el punto de vista visual en la construcción de los modelos.

Podemos apreciar que la Figura 5.1 y la Figura 5.2 son actividades del *plugin* DQ y el *plugin* DQSec respectivamente. Se puede ver en la Figura 5.2 que soporta una mayor cantidad de elementos. Lo cual se debe a la separación que se realiza de DQ dentro de DQSec considerando exactitud, completitud y oportunidad de forma separada.

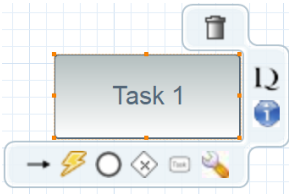


Figura 5.1: Actividad en Plugin DQ


Figura 5.2: Actividad en Plugin DQSec

Ahora se presenta en la Figura 5.3 un *Pool* del *plugin* DQSec y en la Figura 5.4 un *Pool* del *plugin* DQ, de igual manera que en el caso anterior notamos que en el *Pool* del *plugin* DQSec podemos agregar marcas de extensión al elemento. Hay que considerar que en el *plugin* DQ no se aplica la extensión a los *Pools*.

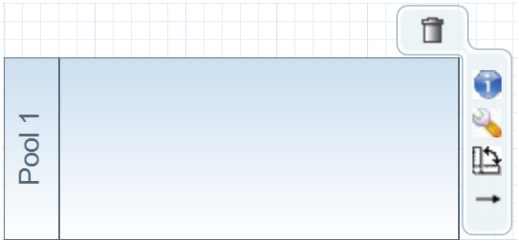


Figura 5.3: Pool en Plugin DQ

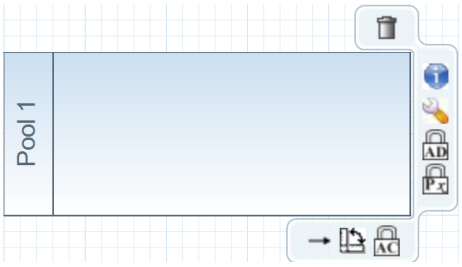


Figura 5.4: Pool en Plugin DQSec

En relación con la capacidad de soportar varias marcas en un elemento, esto es presentado en la Figura 5.5 en la que se muestra una *Activity*, con las marcas *Access Control*, *Attack Harm Detection* y *Oportunity* y en la Figura 5.6 se presenta un *Pool* que contiene las marcas de *Attack Harm Detection* y *Privacy*. Además de reflejar el enriquecimiento mediante marcas gráficas, los componentes de los modelos ven su ID modificado según las marcas agregadas; para el caso de la *Activity* el ID es "Task_SECAD_SECAC_BPOP" y en el caso del *Pool* el ID es "Pool_SECAD_SECPR".

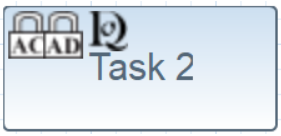


Figura 5.5: Actividad DQSec con Varios Marcas Gráficas


Figura 5.6: Pool DQSec con Varios Marcas Gráficas

## 5.2. Transformaciones BPiDQSec

En esta sección se muestra los pasos llevados a cabo para hacer una transformación. Esto considera el modelo BP enriquecido, su representación en XML, algunas de las reglas que intervienen en este proceso y finalmente, el UCD de manera gráfica junto a la representación de algunos de sus elementos en el lenguaje GML.

En la Figura 5.7, se muestra un proceso de negocio para la compra con tarjeta de crédito en el cual se han representado requisitos de calidad de datos (*Completeness* en la actividad "solicitar autorización de tarjeta" y *Opportunity* en "Message 1") y seguridad (*Privacy* en el actor "Comprador").

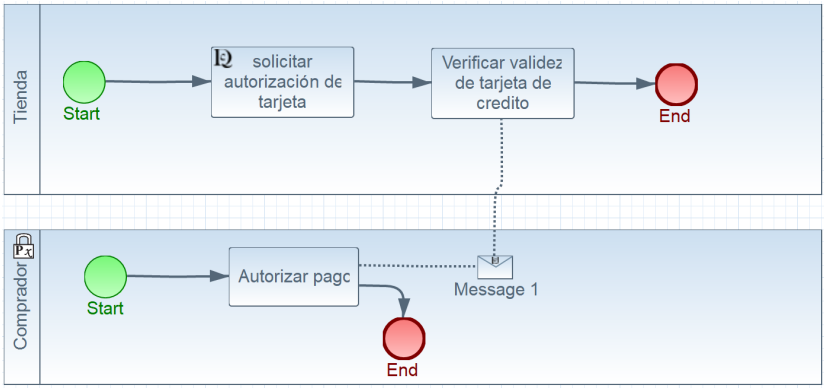


Figura 5.7: Modelo de Procesos de Negocio Enriquecido

En la Figura 5.8 se muestra alguno de los elementos que componen el modelo BP de la Figura 5.7 en el lenguaje XML.

Figura 5.8: XML de modelo BP

```
<BPMNDqSec:BusinessProcessDiagram
Pool
  <pools Name="Comprador"
       Id="Pool2"/>
       RQPR="//@Extension.4
Association
  <ConnectingObjects
       xsi:type="BPMNDqSec:Association"
       Name="Asso1"
       Id="Asso1"
       ActAsso="//@FlowObjects.2"
       DataObjcAsso="//@Artifacts.0"/>
Artifacts
  <Artifacts
       xsi:type="BPMNDqSec:DataObject"
       Name="message 1"
       Id="Mess1"
       RQOP="//@Extension.1"
       Tipo="message"/>
Activity
  <FlowObjects
       xsi:type="BPMNDqSec:Activity"
       Name="solicitar autorizacion de tarjeta"
       Id="Task1"
       RQCO="//@Extension.0"
       ActPool="//@pools.0"/>
Extension
  <Extension
       xsi:type="BPMNDqSec:DQCompleteness"/>
 </BPMNDqSec:BusinessProcessDiagram>
```

El total de las reglas que intervienen para realizar las transformaciones se pueden encontrar en el siguiente anexo (https://docs.google.com/document/d/1X1jrH6nO9PtkZ9mmU25MEaX5oeXUnxUYwRE/edit?usp=sharing)

Una regla correspondiente para convertir un *Pool* en un actor son presentadas en la Figura 5.9, existen reglas que convierten *Pool* con y sin extensión.

Figura 5.9: Regla ATL Pool-Actor

```
rule PoolToAct {
        from
                Bp : MM!Pool(not Bp.PoolAC(Bp) and not Bp.PoolPR(Bp)and not Bp.
                    PoolAD(Bp))
        to
                Uc : MM1!Actor (
                        Id <- Bp.Id,
                        Name <- Bp.Name,
                        ActUc <- Bp,
                        AAct <- Bp.Lanes
```

En la Figura 5.10 se presentan las reglas necesarias para convertir una actividad a un caso de uso. Cabe destacar que hay reglas correspondientes tanto para actividades sin extensión y con extensión.

Figura 5.10: Regla ATL Actividad-CasoDeUso

```
rule ActCOtoUCP {
        from
                Bp : MM!Activity(
                        Bp.ActCO(Bp) and not Bp.LaneDef(Bp))
        to
                Uc2: MM1!UsesCase(
                        Id <- Bp.Id,
                        Name <- Bp.Name,
                        UcAct <- Bp.ActPool,
                        IncludeUC <- Bp,
                        ExtendUC <- Bp),
                Uc : MM1!UsesCase (
                        Id <- Bp.Id,
                        Name <- 'validar la completitud de '+ Bp.Name,
                        IncludeUC <- Bp)}
```

Se muestra la representación mediante XML del UCD generado (ver Figura 5.11). En el código se puede ver de forma detallada cuales son los actores, casos de uso, casos de uso extendidos y casos de uso incluidos. Su representación gráfica con UML se muestra en la Figura 5.13.

Figura 5.11: Representación UCD en XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:UsesCase="http://UsesCase">
  <UsesCase:UsesCaseModel/>
  <UsesCase:Actor Name="Tienda" Id="Pool1" ActUc="/1"/>
  <UsesCase:Actor Name="Comprador" Id="Pool2" ActUc="/2"/>
  <UsesCase:UsesCase Name="Verificar validez de tarjeta de credito" Id="Task2" ExtendUC="/3" IncludeUC="/3" UcAct="/1"/>
  <UsesCase:UsesCase Name="Autorizar Pago" Id="Task3" ExtendUC="/4" IncludeUC="/4" UcAct="/2"/>
  <UsesCase:UsesCase Name="solicitar autorizacion de tarjeta" Id="Task1" ExtendUC="/5" IncludeUC="/5" UcAct="/1"/>
  <UsesCase:UsesCase Name="validar la completitud de solicitar autorizacion de tarjeta" Id="Task1" IncludeUC="/5"/>
  <UsesCase:UsesCase Name="verificar la Oportunidad de message 1" Id="Task3" IncludeUC="/4"/>
  <UsesCase:UsesCase Name="verificar la Oportunidad de message 1" Id="Task2" IncludeUC="/3"/>
</xmi:XMI>
```

en la Figura 5.12 se presentan algunos de los elementos que componen el UCD generado, en el lenguaje GML (Ver Figura /5.12).

Figura 5.12: Representación GML del UCD

```
node
    [
            id      0
            label   "Tienda"
            graphics
            [
                    x       304.0
                    y       212.0
                    w       60.0
                    h       100.0
            ]]
        node
        [
            id      3
            label   "Autorizar Pago"
            graphics
            [
                    x       576.0
                    y       545.5
                    w       224.0
                    h       120.0
            ]]
```

Finalmente se muestra le UCD generado, en este diagrama se aprecia el manejo de los requisitos de calidad mediante casos de uso Incluidos. Se aprecia que "Verificar Completitud de Solicitar autorización de tarjeta" se incluye a "Solicitar autorización de tarjeta" esto con el fin de corroborar *Completeness* en la información utilizada en este caso de uso, se muestra que el actor "Tienda" y el actor "Comprador" verifican la oportunidad del mensaje y se agrega una tarea al actor comprador para verificar su *Privacy* y así no poner en riesgo su información personal.

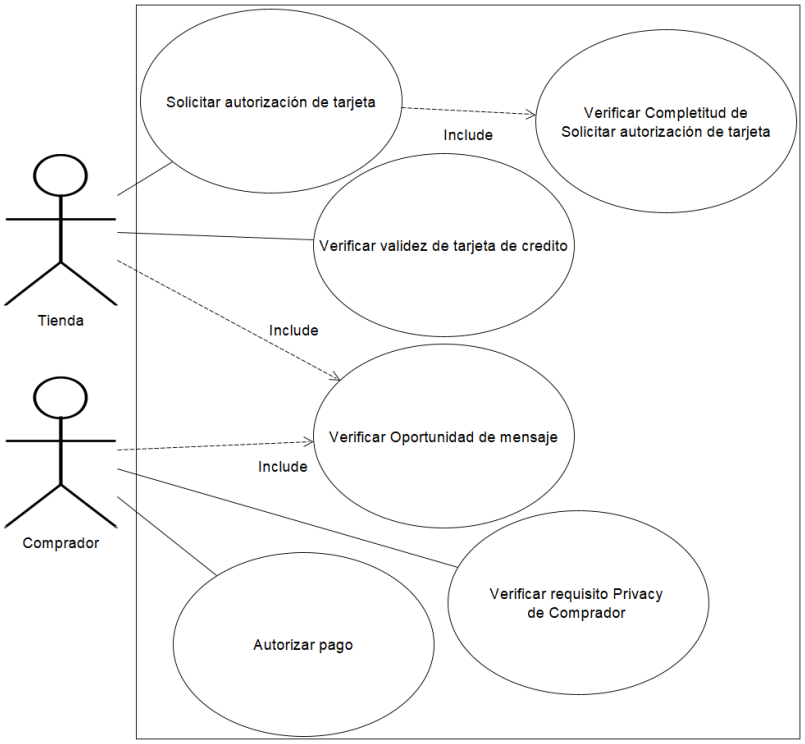


Figura 5.13: Modelo de Casos de Uso Generado

En esta sección se mostró de manera gráfica y mediante secciones de código, todos lo elementos que se van generando a la hora de realizar una transformación.

# Capítulo 6

# Conclusiones

Con la modificación realizada al *plugin* DQ, conseguimos no solo poder modelar calidad de datos en los procesos de negocio descritos con BPMN, sino que también se consigue modelar seguridad. Otra característica agregada en el *plugin* DQSec, es poder añadir más de una marca a los elementos del modelo BP y separar la representación de DQ de exactitud, completitud y oportunidad mediante marcas distintas.

Las transformaciones se lograron mediante la creación de reglas de transformación de modelos en ATL y la creación de una herramienta que se encarga de automatizar este proceso.

En este trabajo se ha presentado un contexto general donde es posible realizar estas especificaciones y transformaciones, los conceptos claves necesarios para entender el problema que se soluciona, las tecnologías utilizadas, la solución propuesta y los resultados obtenidos.

A modo personal cabe señalar que una de las tareas que más trabajo me llevó, fue el comprender el *plugin* DQ para lograr realizar las modificaciones necesarias con el fin de añadir la extensión de seguridad. En relación a las reglas en ATL, el mayor esfuerzo se realizo en lograr que las reglas especificadas se pudieran ejecutar mediante una aplicación en JAVA.

Como trabajo futuro, en cuanto a esta memoria, se podrían proporcionar más reglas descritas con ATL para poder procesar varios elementos de calidad o seguridad presentados en un elemento de los modelos BP a la vez y seguir agregando extensiones al *plugin* DQSec para que este sea cada vez más completo y abarque la mayor cantidad de dominios posibles.

# Bibliografía

[Alanen et al. 2004] Alanen, Marcus ; Porres, Ivan et al.: *A relation between context-free grammars and meta object facility metamodels.* Turku Centre for Computer Science, Book, 2004

[Bittner 2002] Bittner, Kurt: *Use case modeling.* Addison-Wesley Longman Publishing Co., Inc., 2002

[Booch 1999] Booch, Grady: UML in action. In: *Communications of the ACM* 42 (1999), Nr. 10, S. 26–28

[Boronat and Meseguer 2010] Boronat, Artur ; Meseguer, José: An algebraic semantics for MOF. In: *Formal Aspects of Computing* 22 (2010), Nr. 3-4, S. 269–296

[BPMN 2011] BPMN: *BPMN 2.0 Modeler.* 2011. – URL https://www.eclipse.org/proposals/soa.bpmn2-modeler/. – Accedido 07-12-2018

[Bray et al. 1997] Bray, Tim ; Paoli, Jean ; Sperberg-McQueen, C M. ; Maler, Eve ; Yergeau, François: Extensible markup language (XML). In: *World Wide Web Journal* 2 (1997), Nr. 4, S. 27–66

[Caro et al. 2013] Caro, Angélica ; Fuentes, Alejandra ; Soto, M A.: Desarrollando sistemas de información centrados en la calidad de datos. In: *Ingeniare. Revista chilena de ingeniería* 21 (2013), Nr. 1, S. 54–69

[Chinosi and Trombetta 2012] Chinosi, Michele ; Trombetta, Alberto: BPMN: An introduction to the standard. In: *Computer Standards & Interfaces* 34 (2012), Nr. 1, S. 124–134

[Ciaramella et al. 2009] Ciaramella, Alessandro ; Cimino, Mario G. ; Lazzerini, Beatrice ; Marcelloni, Francesco: Using BPMN and Tracing for Rapid Business Process Prototyping Environments. In: *ICEIS (3)*, 2009, S. 206–212

[Cowan and Megginson 1999] Cowan, John ; Megginson, David: XML information set. In: *World Wide Web Consortium, W3C Working Draft WD-xml-infoset-19991220, http://www. w3. org/TR/xml-infoset* (1999)

[Eclipse 2007] Eclipse, IDE: *Eclipse Foundation.* 2007. – URL https://www.eclipse.org/

[Fuentes et al. 2015] Fuentes, Quijada ; Ignacio, Guillermo et al.: Implementación de un prototipo de la Extensión dqBP en BPMN, Tesis. (2015)

[Gosling et al. 2014] Gosling, James ; Joy, Bill ; Steele, Guy ; Bracha, Gilad ; Buckley, Alex: *The Java Language Specification, Java SE 8 Edition (Java Series)*. 2014

[Himsolt 1997] Himsolt, Michael: GML: A portable graph file format. In: *Html page under http://www. fmi. uni-passau. de/graphlet/gml/gml-tr. html, Universität Passau* (1997)

[Jouault 2006] Jouault, Frédéric: *Contribution à l'étude des langages de transformation de modèles, PhdTesis*, Nantes, Dissertation, 2006

[Jouault et al. 2008] Jouault, Frédéric ; Allilaire, Freddy ; Bézivin, Jean ; Kurtev, Ivan: ATL: A model transformation tool. In: *Science of computer programming* 72 (2008), Nr. 1-2, S. 31–39

[Mili et al. 2004] Mili, Hafedh ; Jaoude, Guitta B. ; Lefebvre, Éric ; Tremblay, Guy: Going beyond MDA: Business process modeling for software reuse. In: *Proceedings of the Workshop on Legacy Transformation: Capturing Business Knowledge from Legacy Systems-OOPSLA* Bd. 2004, 2004

[OBEO 2014] OBEO, INRIA: *ATLAS transformation language (ATL)*. 2014

[Ordoñez et al. 2016] Ordoñez, AF ; Suarez, P ; Villavicencio, M: *Metodologías ágiles en el desarrollo de software: un enfoque basado en Scrum y MDD.* 2016

[Pressman and Troya 1988] Pressman, Roger S. ; Troya, Jose M.: Ingeniería del software. (1988)

[Rodríguez and Caro 2012] Rodríguez, Alfonso ; Caro, Angélica: Obteniendo Casos de Uso centrados en la Calidad de los Datos desde Procesos de Negocio descritos con BPMN. In: *RISTI-Revista Ibérica de Sistemas e Tecnologias de Informação* (2012), Nr. 10, S. 65–80

[Rodríguez et al. 2007] Rodríguez, Alfonso ; Fernández-Medina, Eduardo ; Piattini, Mario: A BPMN extension for the modeling of security requirements in business processes. In: *IEICE transactions on information and systems* 90 (2007), Nr. 4, S. 745–752

[Sanchis et al. 2009] Sanchis, Raquel ; Poler, Raúl ; Ortiz, Ángel: Técnicas para el Modelado de Procesos de Negocio en Cadenas de Suministro. In: *Información tecnológica* 20 (2009), Nr. 2, S. 29–40

[Silver 2011] Silver, Bruce: *BPMN Method and Style, with BPMN Implementer's Guide: A structured approach for business process modeling and implementation using BPMN 2.0*. Cody-Cassidy Press Aptos, 2011

[yWorks 2014] yWorks: yEd Graph Editor. (2014). – URL https://www.yworks.com/products/yed

# Apéndice A

# Reglas ATL

## Transformaciones BPMN (nombre) a UCD (nombre)

```
rule BpmnMToUseCaseM {
        from
                Bp : MM!BusinessProcessDiagram
        to
                Uc : MM1!UsesCaseModel (
                        Name <- Bp.Name
                )
}
```

## Transformaciones de Pool-Actor

```
rule PoolToAct {
        from
                Bp : MM!Pool(not Bp.PoolAC(Bp) and not Bp.PoolPR(Bp)and not Bp.
                    PoolAD(Bp))
        to
                Uc : MM1!Actor (
                        Id <- Bp.Id,
                        Name <- Bp.Name,
                        ActUc <- Bp,
                        AAct <- Bp.Lanes
                )
}
```

```
rule PoolToActAC {
        from
                Bp : MM!Pool(Bp.PoolAC(Bp))
        to
                Uc : MM1!Actor (
                        Id <- Bp.Id,
                        Name <- Bp.Name,
                        ActUc <- Bp,
                        AAct <- Bp.Lanes
                ),
                Uc2 : MM1!UsesCase(
                        Id <- 'ext_' + Bp.Id,
                        Name <- 'verificar Access Control de ' + Bp.Name,
                        IncludeUC <- Bp
                )
}
```

```
rule PoolToActAD{
        from
                Bp : MM!Pool(Bp.PoolAD(Bp))
        to
                Uc : MM1!Actor (
                        Id <- Bp.Id,
                        Name <- Bp.Name,
                        ActUc <- Bp,
                        AAct <- Bp.Lanes
                ),
                Uc2 : MM1!UsesCase(
                        Id <- 'ext_' + Bp.Id,
                        Name <- 'verificar_Attack_Harm_Detection_de_' + Bp.Name,
                        ExtendUC <-  Bp
                )
}
```

```
rule PoolToActPR{
        from
                Bp : MM!Pool((Bp.PoolPR(Bp)))
        to
                Uc : MM1!Actor (
                        Id <- Bp.Id,
                        Name <- Bp.Name,
                        ActUc <- Bp,
                        AAct <- Bp.Lanes
                ),
                Uc2 : MM1!UsesCase(
                        Id <- 'ext_' + Bp.Id,
                        Name <- 'verificar_Privacity_' + Bp.Name,
                        ExtendUC <- Bp
                )
}
```

## Transformaciones de Lane-Actor

```
rule LaneToAct {
        from
                Bp : MM!Lane(not Bp.LanePR(Bp) and not Bp.LaneAC(Bp) and not Bp.
                    LaneAD(Bp))
        to
                Uc : MM1!Actor (
                        Name <- Bp.Name,
                        Id <- Bp.Id,
                        ActUc <- Bp,
                        AAct <- Bp.PoolLane

                )
}
```

```
rule LaneToActAC {
        from
                Bp : MM!Lane(Bp.LaneAC(Bp))
        to
                Uc : MM1!Actor (
                        Name <- Bp.Name,
                        Id <- 'EXT-AC',
                        ActUc <- Bp,
                        AAct <- Bp.PoolLane

                )
}
```

```
rule LaneToActAD {
        from
                Bp : MM!Lane(Bp.LaneAD(Bp))
        to
                Uc : MM1!Actor (
                        Name <- Bp.Name,
                        Id <- 'EXT-AD',
                        ActUc <- Bp,
                        AAct <- Bp.PoolLane

                )
}
```

```
rule LaneToActPR {
        from
                Bp : MM!Lane(Bp.LanePR(Bp))
        to
                Uc : MM1!Actor (
                        Name <- Bp.Name,
                        Id <- 'EXT-PR',
                        ActUc <- Bp,
                        AAct <- Bp.PoolLane

                )
}
```

## Transformaciones de Activity-Caso de Uso

```
rule ActividadNormalP {
        from
                Bp : MM!Activity (not Bp.ActAC(Bp) and not Bp.ActAD(Bp) and not Bp.
                    ActCO(Bp)and not Bp.ActACC(Bp)and not Bp.ActOP(Bp) and not Bp.
                    LaneDef(Bp))
        to
                Uc : MM1!UsesCase (
                        Id <- Bp.Id,
                        Name <- Bp.Name,
                        UcAct <- Bp.ActPool,
                        IncludeUC <- Bp,
                        ExtendUC <- Bp
                )
}
```

```
rule ActividadNormalL {
        from
                Bp : MM!Activity (not Bp.ActAC(Bp) and not Bp.ActAD(Bp) and not Bp.
                    ActCO(Bp)and not Bp.ActACC(Bp)and not Bp.ActOP(Bp) and Bp.
                    LaneDef(Bp))
        to
                Uc : MM1!UsesCase (
                        Id <- Bp.Id,
                        Name <- Bp.Name,
                        UcAct <- Bp.ActLane,
                        IncludeUC <- Bp,
                        ExtendUC <- Bp
                )
}
```

```
rule ActACtoUCL {
        from
                Bp : MM!Activity(
                        Bp.ActAC(Bp) and Bp.LaneDef(Bp)
                )
        to
                Uc2: MM1!UsesCase(
                        Id <- Bp.Id,
                        Name <- Bp.Name,
                        UcAct <- Bp.ActLane,
                        IncludeUC <- Bp,
                        ExtendUC <- Bp
                ),
                Uc : MM1!UsesCase (
                        Id <- Bp.Id,
                        Name <- 'validar␣a␣'+ Bp.Name + '␣si␣sufrio␣ataque␣de␣acceso
                            ',
                        IncludeUC <- Bp
                )
}
```

```
rule ActACtoUCP {
        from
                Bp : MM!Activity(
                        Bp.ActAC(Bp) and not Bp.LaneDef(Bp)
                )
        to
                Uc2: MM1!UsesCase(
                        Id <- Bp.Id,
                        Name <- Bp.Name,
                        UcAct <- Bp.ActPool,
                        IncludeUC <- Bp,
                        ExtendUC <- Bp
                ),
                Uc : MM1!UsesCase (
                        Id <- Bp.Id,
                        Name <- 'validar␣a␣'+ Bp.Name + '␣si␣sufrio␣ataque␣de␣acceso
                            ',
                        IncludeUC <- Bp
                )
}
```

```
rule ActACCtoUCL {
        from
                Bp : MM!Activity(
                        Bp.ActACC(Bp) and Bp.LaneDef(Bp)
                )
        to
                Uc: MM1!UsesCase(
                        Id <- Bp.Id,
                        Name <- Bp.Name,
                        UcAct <- Bp.ActLane,
                        IncludeUC <- Bp,
                        ExtendUC <- Bp
                ),

                Uc2 : MM1!UsesCase (
                        Id <- Bp.Id,
                        Name <- 'validar la Exactitud de '+ Bp.Name,
                        IncludeUC <- Bp
                )
}
```

```
rule ActACCtoUCP {
        from
                Bp : MM!Activity(
                        Bp.ActACC(Bp) and not Bp.LaneDef(Bp)
                )
        to
                Uc: MM1!UsesCase(
                        Id <- Bp.Id,
                        Name <- Bp.Name,
                        UcAct <- Bp.ActPool,
                        IncludeUC <- Bp,
                        ExtendUC <- Bp
                ),

                Uc2 : MM1!UsesCase (
                        Id <- Bp.Id,
                        Name <- 'validar la Exactitud de '+ Bp.Name,
                        IncludeUC <- Bp
                )
}
```

```
rule ActADtoUCL {
        from
                Bp : MM!Activity(
                        Bp.ActAD(Bp) and Bp.LaneDef(Bp)
                )
        to
                Uc: MM1!UsesCase(
                        Id <- Bp.Id,
                        Name <- Bp.Name,
                        UcAct <- Bp.ActLane,
                        IncludeUC <- Bp,
                        ExtendUC <- Bp
                ),
                Uc2 : MM1!UsesCase (
                        Id <- Bp.Id,
                        Name <- 'validar si '+ Bp.Name+ ' detecto un attack harm',
                        ExtendUC  <- Bp
                )
}
```

```
rule ActADtoUCP {
        from
                Bp : MM!Activity(
                        Bp.ActAD(Bp) and not Bp.LaneDef(Bp)
                )
        to
                Uc: MM1!UsesCase(
                        Id <- Bp.Id,
                        Name <- Bp.Name,
                        UcAct <- Bp.ActPool,
                        IncludeUC <- Bp,
                        ExtendUC <- Bp
                ),
                Uc2 : MM1!UsesCase (
                        Id <- Bp.Id,
                        Name <- 'validar si '+ Bp.Name+ ' detecto un attack harm',
                        ExtendUC  <- Bp
                )
}
```

```
rule ActOPtoUCL {
        from
                Bp : MM!Activity(
                        Bp.ActOP(Bp) and Bp.LaneDef(Bp)
                )
        to
                Uc2: MM1!UsesCase(
                        Id <- Bp.Id,
                        Name <- Bp.Name,
                        UcAct <- Bp.ActLane,
                        IncludeUC <- Bp,
                        ExtendUC <- Bp
                ),
                Uc : MM1!UsesCase (
                        Id <- Bp.Id,
                        Name <- 'verificar la oportunidad de '+ Bp.Name,
                        IncludeUC <- Bp
                )
}
```

```
rule ActOPtoUCP {
        from
                Bp : MM!Activity(
                        Bp.ActOP(Bp) and not Bp.LaneDef(Bp)
                )
        to
                Uc2: MM1!UsesCase(
                        Id <- Bp.Id,
                        Name <- Bp.Name,
                        UcAct <- Bp.ActPool,
                        IncludeUC <- Bp,
                        ExtendUC <- Bp
                ),
                Uc : MM1!UsesCase (
                        Id <- Bp.Id,
                        Name <- 'verificar la oportunidad de '+ Bp.Name,
                        IncludeUC <- Bp
                )
}
```

```
rule ActCOtoUCL {
        from
                Bp : MM!Activity(
                        Bp.ActCO(Bp) and Bp.LaneDef(Bp)
                )
        to
                Uc2: MM1!UsesCase(
                        Id <- Bp.Id,
                        Name <- Bp.Name,
                        UcAct <- Bp.ActLane,
                        IncludeUC <- Bp,
                        ExtendUC <- Bp
                ),
                Uc : MM1!UsesCase (
                        Id <- Bp.Id,
                        Name <- 'validar la completitud de '+ Bp.Name,
                        IncludeUC <- Bp

                )
}
```

```
rule ActCOtoUCP {
        from
                Bp : MM!Activity(
                        Bp.ActCO(Bp) and not Bp.LaneDef(Bp)
                )
        to
                Uc2: MM1!UsesCase(
                        Id <- Bp.Id,
                        Name <- Bp.Name,
                        UcAct <- Bp.ActPool,
                        IncludeUC <- Bp,
                        ExtendUC <- Bp
                ),
                Uc : MM1!UsesCase (
                        Id <- Bp.Id,
                        Name <- 'validar la completitud de '+ Bp.Name,
                        IncludeUC <- Bp
                )
}
```

### Transformaciones de ConnectingObject-Casos de Uso (Extend o Include)

```
rule TaskAssoDsEX {
        from
                Bp : MM!Association(
                        Bp.DataObjcAsso.DataStoreEX(Bp.DataObjcAsso)
                )
        to
                Uc : MM1!UsesCase (
                        Name <- 'verificar la Exactitud de '+ Bp.DataObjcAsso.Name,
                        Id <- Bp.ActAsso.Id,
                        IncludeUC <- Bp.ActAsso
                )
}
```

```
rule TaskAssoDsOP {
        from
                Bp : MM!Association(
                        Bp.DataObjcAsso.DataStoreOP(Bp.DataObjcAsso)
                )
        to
                Uc : MM1!UsesCase (
                        Name <- 'verificar la Oportunidad de '+ Bp.DataObjcAsso.Name
                                ,
                        Id <- Bp.ActAsso.Id,
                        IncludeUC <- Bp.ActAsso
                )
}
```

```
rule TaskAssoDsCO {
        from
                Bp : MM!Association(
                        Bp.DataObjcAsso.DataStoreCO(Bp.DataObjcAsso)
                )
        to
                Uc : MM1!UsesCase (
                        Name <- 'verificar la Completitud de '+ Bp.DataObjcAsso.Name
                                ,
                        Id <- Bp.ActAsso.Id,
                        IncludeUC <- Bp.ActAsso
                )
}
```

```
rule TaskAssoMSEX {
        from
                Bp : MM!Association(
                        Bp.DataObjcAsso.MenssageEX(Bp.DataObjcAsso)
                )
        to
                Uc : MM1!UsesCase (
                        Name <- 'verificar la Exactitud de '+ Bp.DataObjcAsso.Name,
                        Id <- Bp.ActAsso.Id,
                        IncludeUC <- Bp.ActAsso
                )
}
```

```
rule TaskAssoMSCO {
        from
                Bp : MM!Association(
                        Bp.DataObjcAsso.MenssageCO(Bp.DataObjcAsso)
                )
        to
                Uc : MM1!UsesCase (
                        Name <- 'verificar la Completitud de '+ Bp.DataObjcAsso.Name
                                ,
                        Id <- Bp.ActAsso.Id,
                        IncludeUC <- Bp.ActAsso
                )
}
```

```
rule TaskAssoMSOP {
        from
                Bp : MM!Association(
                        Bp.DataObjcAsso.MenssageOP(Bp.DataObjcAsso)
                )
        to
                Uc : MM1!UsesCase (
                        Name <- 'verificar la Oportunidad de '+ Bp.DataObjcAsso.Name
                                ,
                        Id <- Bp.ActAsso.Id,
                        IncludeUC <- Bp.ActAsso
                )
}
```

```
rule TaskAssoDOAD {
        from
                Bp : MM!Association(
                        Bp.DataObjcAsso.DataObjectAD(Bp.DataObjcAsso)
                )
        to
                Uc : MM1!UsesCase (
                        Name <- 'verificar si '+ Bp.DataObjcAsso.Name + ' sufrio AHD
                                ',
                        Id <- Bp.ActAsso.Id,
                        ExtendUC <- Bp.ActAsso
                )
}
```

```
rule TaskAssoDOIN {
        from
                Bp : MM!Association(
                        Bp.DataObjcAsso.DataObjectIN(Bp.DataObjcAsso)
                )
        to
                Uc : MM1!UsesCase (
                        Name <- 'verificar integridad de '+ Bp.DataObjcAsso.Name,
                        Id <- Bp.ActAsso.Id,
                        ExtendUC <- Bp.ActAsso
                )
}
```

```
rule TaskAssoDOEX {
        from
                Bp : MM!Association(
                        Bp.DataObjcAsso.DataObjectEX(Bp.DataObjcAsso)
                )
        to
                Uc : MM1!UsesCase (
                        Name <- 'verificar Exactitud de '+ Bp.DataObjcAsso.Name,
                        Id <- Bp.ActAsso.Id,
                        IncludeUC <- Bp.ActAsso
                )
}
```

```
rule TaskAssoDOOP {
        from
                Bp : MM!Association(
                        Bp.DataObjcAsso.DataObjectOP(Bp.DataObjcAsso)
                )
        to
                Uc : MM1!UsesCase (
                        Name <- 'verificar Oportunidad de '+ Bp.DataObjcAsso.Name,
                        Id <- Bp.ActAsso.Id,
                        IncludeUC <- Bp.ActAsso
                )
}
```

```
rule TaskAssoDOCO {
        from
                Bp : MM!Association(
                        Bp.DataObjcAsso.DataObjectCO(Bp.DataObjcAsso)
                )
        to
                Uc : MM1!UsesCase (
                        Name <- 'verificar Completitud de '+ Bp.DataObjcAsso.Name,
                        Id <- Bp.ActAsso.Id,
                        IncludeUC <- Bp.ActAsso
                )
}
```

# Apéndice B

# Clases Plugin DQSec

### Clase getImageId() DQ

```
1 public String getImageId() {
2         return IConstants.ICON_DQBPAC_16;
3 }
4 public String getImageId_min8() {
5         return IConstants.ICON_DQBPAC_8;
6 }
7 public String getImageId_min6() {
8         return IConstants.ICON_DQBPAC_6;
9 }
```

```
1 public String getImageId() {
2         return IConstants.ICON_DQBPCO_16;
3 }
4 public String getImageId_min8() {
5         return IConstants.ICON_DQBPCO_8;
6 }
7 public String getImageId_min6() {
8         return IConstants.ICON_DQBPCO_6;
9 }
```

```
1 public String getImageId() {
2         return IConstants.ICON_DQBPOP_16;
3 }
4 public String getImageId_min8() {
5         return IConstants.ICON_DQBPOP_8;
6 }
7 public String getImageId_min6() {
8         return IConstants.ICON_DQBPOP_6;
9 }
```

### Clase getImageId() Sec

```
1 public String getImageId() {
2         return IConstants.ICON_BPSECAC_16;
3 }
4 public String getImageId_min8() {
5         return IConstants.ICON_BPSECAC_8;
6 }
7 public String getImageId_min6() {
8         return IConstants.ICON_BPSECAC_6;
9 }
```

```
public String getImageId() {
        return IConstants.ICON_BPSECAD_16;
}
public String getImageId_min8() {
        return IConstants.ICON_BPSECAD_8;
}
public String getImageId_min6() {
        return IConstants.ICON_BPSECAD_6;
}
```

```
public String getImageId() {
        return IConstants.ICON_BPSECIN_16;
}
public String getImageId_min8() {
        return IConstants.ICON_BPSECIN_8;
}
public String getImageId_min6() {
        return IConstants.ICON_BPSECIN_6;
}
```

```
public String getImageId() {
        return IConstants.ICON_BPSECPR_16;
}
public String getImageId_min8() {
        return IConstants.ICON_BPSECPR_8;
}
public String getImageId_min6() {
        return IConstants.ICON_BPSECPR_6;
}
```

### Clase isAvailable() DQ

```
public boolean isAvailable(IContext context) {
        if ((context instanceof ICustomContext)) {
                PictogramElement[] pes = ((ICustomContext) context).
                    getPictogramElements();
                DiagramEditor editor = (DiagramEditor) getDiagramBehavior().
                    getDiagramContainer();
                editor.setPictogramElementForSelection(pes[0]);
                EObject obj = BusinessObjectUtil.
                    getBusinessObjectForPictogramElement(pes[0]);
                if (((obj instanceof Message)) || ((obj instanceof
                    DataStoreReference)) || ((obj instanceof DataObject))
                                || ((obj instanceof Task)) || ((obj instanceof
                                    MessageFlow)) || ((obj instanceof Conversation))
                                || ((obj instanceof DataInput)) || ((obj instanceof
                                    DataOutput))) {
                        if (!((BaseElement) obj).getId().contains("_DQAC")) {
                                return true;
                        }
                }
                return false;
        }
        return false;
}
```

```
1  public boolean isAvailable(IContext context) {
2          if ((context instanceof ICustomContext)) {
3                  PictogramElement[] pes = ((ICustomContext) context).
                       getPictogramElements();
4                  DiagramEditor editor = (DiagramEditor) getDiagramBehavior().
                       getDiagramContainer();
5                  editor.setPictogramElementForSelection(pes[0]);
6                  EObject obj = BusinessObjectUtil.
                       getBusinessObjectForPictogramElement(pes[0]);
7                  if (((obj instanceof Message)) || ((obj instanceof
                       DataStoreReference)) || ((obj instanceof DataObject))
8                                  || ((obj instanceof Task)) || ((obj instanceof
                                       MessageFlow)) || ((obj instanceof Conversation))
9                                  || ((obj instanceof DataInput)) || ((obj instanceof
                                       DataOutput))) {
10                         if (!((BaseElement) obj).getId().contains("_DQCO")) {
11                                 return true;
12                         }
13                 }
14                 return false;
15         }
16         return false;
17 }
```

```
1  public boolean isAvailable(IContext context) {
2          if ((context instanceof ICustomContext)) {
3                  PictogramElement[] pes = ((ICustomContext) context).
                       getPictogramElements();
4                  DiagramEditor editor = (DiagramEditor) getDiagramBehavior().
                       getDiagramContainer();
5                  editor.setPictogramElementForSelection(pes[0]);
6                  EObject obj = BusinessObjectUtil.
                       getBusinessObjectForPictogramElement(pes[0]);
7                  if (((obj instanceof Message)) || ((obj instanceof
                       DataStoreReference)) || ((obj instanceof DataObject))
8                                  || ((obj instanceof Task)) || ((obj instanceof
                                       MessageFlow)) || ((obj instanceof Conversation))
9                                  || ((obj instanceof DataInput)) || ((obj instanceof
                                       DataOutput))) {
10                         if (!((BaseElement) obj).getId().contains("_DQOP")) {
11                                 return true;
12                         }
13                 }
14                 return false;
15         }
16         return false;
17 }
```

### Clase isAvailable() Sec

```
1  public boolean isAvailable(IContext context) {
2          if ((context instanceof ICustomContext)) {
3                  PictogramElement[] pes = ((ICustomContext) context).
                       getPictogramElements();
4                  DiagramEditor editor = (DiagramEditor) getDiagramBehavior().
                       getDiagramContainer();
5                  editor.setPictogramElementForSelection(pes[0]);
6                  EObject obj = BusinessObjectUtil.
                       getBusinessObjectForPictogramElement(pes[0]);
7                  if (((obj instanceof Group))
8                                  || ((obj instanceof Task))
9                                  || ((obj instanceof Lane)) || ((obj instanceof
                                       Participant ))) {
10                         if (!((BaseElement) obj).getId().contains("_SECAC")) {
11                                 return true;
12                         }
13                 }
14                 return false;
15         }
16         return false;}
```

```
1  public boolean isAvailable(IContext context) {
2          if ((context instanceof ICustomContext)) {
3                  PictogramElement[] pes = ((ICustomContext) context).
                       getPictogramElements();
4                  DiagramEditor editor = (DiagramEditor) getDiagramBehavior().
                       getDiagramContainer();
5                  editor.setPictogramElementForSelection(pes[0]);
6                  EObject obj = BusinessObjectUtil.
                       getBusinessObjectForPictogramElement(pes[0]);
7                  if (((obj instanceof Group)) || ((obj instanceof DataObject))
8                                  || ((obj instanceof Task)) || ((obj instanceof
                                       MessageFlow))
9                                  || ((obj instanceof Lane)) || ((obj instanceof
                                       Participant ))) {
10                         if (!((BaseElement) obj).getId().contains("_SECAD")) {
11                                 return true;
12                         }
13                 }
14                 return false;
15         }
16         return false;}
```

```
1  public boolean isAvailable(IContext context) {
2          if ((context instanceof ICustomContext)) {
3                  PictogramElement[] pes = ((ICustomContext) context).
                       getPictogramElements();
4                  DiagramEditor editor = (DiagramEditor) getDiagramBehavior().
                       getDiagramContainer();
5                  editor.setPictogramElementForSelection(pes[0]);
6                  EObject obj = BusinessObjectUtil.
                       getBusinessObjectForPictogramElement(pes[0]);
7                  if (((obj instanceof DataObject))|| ((obj instanceof MessageFlow)))
                       {
8                          if (!((BaseElement) obj).getId().contains("_SECIN")) {
9                                  return true;
10                         }
11                 }
12                 return false;
13         }
14         return false;}
```

## Clase Execute() DQ

```
1  public void executeDQAC(final ICustomContext context) {
2          TransactionalEditingDomain domain = getDiagramEditor().getEditingDomain();
3          domain.getCommandStack().execute(new RecordingCommand(domain) {
4          protected void doExecute() {
5                  PictogramElement pes = context.getPictogramElements()[0];
6                  DiagramEditor editor = (DiagramEditor) ShowDQBPFeatureAC.this.
                       getDiagramBehavior().getDiagramContainer();
7                  editor.setPictogramElementForSelection(pes);
8                  EObject businessObject = BusinessObjectUtil.
                       getBusinessObjectForPictogramElement(pes);
9                  if (!ShowDQBPFeatureAC.this.dqacstate) {
10                         ContainerShape x = (ContainerShape) ShowDQBPFeatureAC.this.
                               fp
11                         .getPictogramElementForBusinessObject(businessObject);
12                         if ((businessObject instanceof Message)) {}
13                         if ((businessObject instanceof DataStoreReference)) {}
14                         if ((businessObject instanceof Conversation)) {}
15                         if (((businessObject instanceof DataInput)) || ((
                               businessObject instanceof DataOutput))) {}
16                         if ((businessObject instanceof MessageFlow)) {}
17                         if (((businessObject instanceof BusinessRuleTask)) || ((
                               businessObject instanceof ManualTask))
18                                 || ((businessObject instanceof UserTask)) || ((
                                       businessObject instanceof ScriptTask))
19                                 || ((businessObject instanceof ServiceTask)) || ((
                                       businessObject instanceof SendTask))
20                                 || ((businessObject instanceof ReceiveTask))) {}
21         ShowDQBPFeatureAC.this.dqacstate = true;
22 }}});}
```

```
1  public void executeDQCO(final ICustomContext context) {
2          TransactionalEditingDomain domain = getDiagramEditor().getEditingDomain();
3          domain.getCommandStack().execute(new RecordingCommand(domain) {
4          protected void doExecute() {
5                  PictogramElement pes = context.getPictogramElements()[0];
6                  DiagramEditor editor = (DiagramEditor) ShowDQBPFeatureCO.this.
                       getDiagramBehavior().getDiagramContainer();
7                  editor.setPictogramElementForSelection(pes);
8                  EObject businessObject = BusinessObjectUtil.
                       getBusinessObjectForPictogramElement(pes);
9                          if (!ShowDQBPFeatureCO.this.dqcostate) {
10                                 ContainerShape x = (ContainerShape)
                                       ShowDQBPFeatureCO.this.fp
11                                 .getPictogramElementForBusinessObject(businessObject
                                       );
12                         if ((businessObject instanceof Message)) {}
13                         if ((businessObject instanceof DataStoreReference)) {
14                         if ((businessObject instanceof Conversation)) {
15                         if (((businessObject instanceof DataInput)) || ((
                               businessObject instanceof DataOutput))) {
16                         if ((businessObject instanceof MessageFlow)) {
17                         if (((businessObject instanceof BusinessRuleTask)) || ((
                               businessObject instanceof ManualTask))
18                                 || ((businessObject instanceof UserTask)) || ((
                                       businessObject instanceof ScriptTask))
19                                 || ((businessObject instanceof ServiceTask)) || ((
                                       businessObject instanceof SendTask))
20                                 || ((businessObject instanceof ReceiveTask))) {
21                 ShowDQBPFeatureCO.this.dqcostate = true;
22 }}});}
```

```
1  public void executeDQOP(final ICustomContext context) {
2          TransactionalEditingDomain domain = getDiagramEditor().getEditingDomain();
3          domain.getCommandStack().execute(new RecordingCommand(domain) {
4          protected void doExecute() {
5                  PictogramElement pes = context.getPictogramElements()[0];
6                  DiagramEditor editor = (DiagramEditor) ShowDQBPFeatureOP.this.
                       getDiagramBehavior().getDiagramContainer();
7                  editor.setPictogramElementForSelection(pes);
8                  EObject businessObject = BusinessObjectUtil.
                       getBusinessObjectForPictogramElement(pes);
9                  if (!ShowDQBPFeatureOP.this.dqopstate) {
10                         ContainerShape x = (ContainerShape) ShowDQBPFeatureOP.this.
                               fp
11                         .getPictogramElementForBusinessObject(businessObject);
12                 if ((businessObject instanceof Message)) {}
13         if ((businessObject instanceof DataStoreReference)) {}
14         if ((businessObject instanceof Conversation)) {}
15         if (((businessObject instanceof DataInput)) || ((businessObject instanceof
               DataOutput))) {
16         if ((businessObject instanceof MessageFlow)) {
17         if (((businessObject instanceof BusinessRuleTask)) || ((businessObject
               instanceof ManualTask))
18                 || ((businessObject instanceof UserTask)) || ((businessObject
                       instanceof ScriptTask))
19                 || ((businessObject instanceof ServiceTask)) || ((businessObject
                       instanceof SendTask))
20                 || ((businessObject instanceof ReceiveTask))) {}
21                 Task r = (Task) businessObject;
22                 if (!r.getId().contains("DQOP")) {}
23         } else if (((businessObject instanceof DataObject)) || ((businessObject
               instanceof Task))) {}
24                 if ((businessObject instanceof DataObject)) {}
25                 if ((businessObject instanceof DataObject)) {}
26                 if ((businessObject instanceof Task)) {}
27         }
28         ShowDQBPFeatureOP.this.dqopstate = true;
29 }}});}
```

### Clase Execute() Sec

```
1  public void executeSECAD(final ICustomContext context) {
2          TransactionalEditingDomain domain = getDiagramEditor().getEditingDomain();
3          domain.getCommandStack().execute(new RecordingCommand(domain) {
4          protected void doExecute() {
5                  PictogramElement pes = context.getPictogramElements()[0];
6                  DiagramEditor editor = (DiagramEditor) ShowSecFeatureAD.this.
                       getDiagramBehavior().getDiagramContainer();
7                  editor.setPictogramElementForSelection(pes);
8                  EObject businessObject = BusinessObjectUtil.
                       getBusinessObjectForPictogramElement(pes);
9                  if (!ShowSecFeatureAD.this.secadstate) {
10                         ContainerShape x = (ContainerShape) ShowSecFeatureAD.this.fp
11                         .getPictogramElementForBusinessObject(businessObject);
12                         if ((businessObject instanceof MessageFlow)) {}
13         if((businessObject instanceof Lane)) {}
14         if((businessObject instanceof Group)) {}
15         if((businessObject instanceof Participant)) {}
16         if (((businessObject instanceof BusinessRuleTask)) || ((businessObject
               instanceof ManualTask))
17                 || ((businessObject instanceof UserTask)) || ((businessObject
                       instanceof ScriptTask))
18                 || ((businessObject instanceof ServiceTask)) || ((businessObject
                       instanceof SendTask))
19                 || ((businessObject instanceof ReceiveTask))) {}
20                 Task r = (Task) businessObject;
21                 if (!r.getId().contains("_SECAD")) {
22                         r.setId(r.getId() + "_SECAD");
23                         int cont =0;
24                         if(r.getId().contains("_DQCO")) {}
25                         if(r.getId().contains("_DQOP")) {}
26                         if(r.getId().contains("_DQAC")) {}
27                         if(r.getId().contains("_SECAC")) {}
28                 ShapeDecoratorUtil.createImageSec(((Shape) x.getChildren().get(0)).
                       getGraphicsAlgorithm(),
29                 }
                                   ShowSecFeatureAD.this.getImageId(), 20+cont, 2);
30         }
31         if (((businessObject instanceof DataObject))) {}
32         if ((businessObject instanceof Task)) {}
33         ShowSecFeatureAD.this.secadstate = true;
34 }}});
```

```
public void execute(final ICustomContext context) {
        TransactionalEditingDomain domain = getDiagramEditor().getEditingDomain();
        domain.getCommandStack().execute(new RecordingCommand(domain) {
        protected void doExecute() {
                PictogramElement pes = context.getPictogramElements()[0];
                DiagramEditor editor = (DiagramEditor) ShowSecFeatureIN.this.
                    getDiagramBehavior().getDiagramContainer();
                editor.setPictogramElementForSelection(pes);
                EObject businessObject = BusinessObjectUtil.
                    getBusinessObjectForPictogramElement(pes);
                if (!ShowSecFeatureIN.this.secinstate) {
                        ContainerShape x = (ContainerShape) ShowSecFeatureIN.this.fp
                        .getPictogramElementForBusinessObject(businessObject);
                        if ((businessObject instanceof MessageFlow)) {
                        if (((businessObject instanceof DataObject)) ) {
                        if ((businessObject instanceof DataObject)) {
                                DataObject r = (DataObject) businessObject;
                                if (!r.getId().contains("_SECIN")) {
                                        r.setId(r.getId() + "_SECIN");
                                        int cont =0;
                                        }
                                if(r.getId().contains("_DQCO")) {}
                                if(r.getId().contains("_DQOP")) {}
                                if(r.getId().contains("_SECAD")) {}
                                if(r.getId().contains("_DQAC")) {}
                                if(cont < 32) {}
                                if (cont == 32){}
                                if (cont ==48 ){}
                                if (cont==64) {}
                                }
                        }       ShowSecFeatureIN.this.secinstate = true;
}}});
```

```
public void executePR(final ICustomContext context) {
        TransactionalEditingDomain domain = getDiagramEditor().getEditingDomain();
        domain.getCommandStack().execute(new RecordingCommand(domain) {}
        protected void doExecute() {
                PictogramElement pes = context.getPictogramElements()[0];
                DiagramEditor editor = (DiagramEditor) ShowSecFeaturePR.this.
                    getDiagramBehavior().getDiagramContainer();
                editor.setPictogramElementForSelection(pes);
                EObject businessObject = BusinessObjectUtil.
                    getBusinessObjectForPictogramElement(pes);
                if (!ShowSecFeaturePR.this.secprstate) {
                        ContainerShape x = (ContainerShape) ShowSecFeaturePR.this.fp
                        .getPictogramElementForBusinessObject(businessObject);
                        if((businessObject instanceof Lane)) {}
                        if((businessObject instanceof Group)) {}
                        if((businessObject instanceof Participant)) {}
                                Participant r = (Participant) businessObject;
                                if(!r.getId().contains("_SECPR")) {
                                r.setId(r.getId() + "_SECPR");
                                int cont=2;
                                if(r.getId().contains("_SECAD")) {}
                if(r.getId().contains("_SECAC")) {}
                ShapeDecoratorUtil.createImageSec(x.getGraphicsAlgorithm(),
                    ShowSecFeaturePR.this.getImageId(), 2, cont);
                }
        }
        ShowSecFeaturePR.this.secprstate = true;
}}});
```

```
1  public void executeSECAC(final ICustomContext context) {
2          TransactionalEditingDomain domain = getDiagramEditor().getEditingDomain();
3          domain.getCommandStack().execute(new RecordingCommand(domain) {
4          protected void doExecute() {
5                  PictogramElement pes = context.getPictogramElements()[0];
6                  DiagramEditor editor = (DiagramEditor) ShowSecFeatureAC.this.
                       getDiagramBehavior().getDiagramContainer();
7                  editor.setPictogramElementForSelection(pes);
8                  EObject businessObject = BusinessObjectUtil.
                       getBusinessObjectForPictogramElement(pes);
9                  if (!ShowSecFeatureAC.this.secacstate) {
10                         ContainerShape x = (ContainerShape) ShowSecFeatureAC.this.fp
11                         .getPictogramElementForBusinessObject(businessObject);
12                         if ((businessObject instanceof MessageFlow)) {}
13                         if((businessObject instanceof Lane)) {}
14                         if((businessObject instanceof Group)) {}
15                         if((businessObject instanceof Participant)) {}
16                         if (((businessObject instanceof BusinessRuleTask)) || ((
                               businessObject instanceof ManualTask))
17                                 || ((businessObject instanceof UserTask)) || ((
                                       businessObject instanceof ScriptTask))
18                                 || ((businessObject instanceof ServiceTask)) || ((
                                       businessObject instanceof SendTask))
19                                 || ((businessObject instanceof ReceiveTask))) {
20                                 Task r = (Task) businessObject;
21                                 if (!r.getId().contains("_SECAC")) {}
22                 }else if (((businessObject instanceof DataObject)) || ((
                       businessObject instanceof Task))) {
23                                 if ((businessObject instanceof DataObject)) {}
24                                 if ((businessObject instanceof Task)) {}
25                 }       ShowSecFeatureAC.this.secacstate = true;
26 }}});}
```

La especificación completa de todas las clases implementadas en el *Plugin* DQSec, se encuentran en la siguiente URL: https://drive.google.com/drive/folders/1wZmOMg9tpExxIVwpjRzHOQG8YhcUuSww?usp=sharing