UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

PVModeler: Herramienta de modelado de BPMN 2.0 extendido con variabilidad de procesos

Trabajo de Fin de Máster

Máster Universitario en Ingeniería y Tecnología de Sistemas Software

Departamento de Sistemas Informáticos y Computación

Autor/a: André de Oliveira Penteado, Fernando

Tutor/a: Penadés Gramage, María Carmen

Sánchez Díaz, Juan

Curso 2021-2022

*"**Kannagara Tamachihaemase**. A través de esta frase hacemos una oración especial, en que prometemos cumplir con la voluntad de Dios, obedeciéndole a sus órdenes (**Kannagara**); pedimos también el fortalecimiento de nuestra alma (**Tamachihaemasse**)"*

Meishu Sama

Dedico este trabajo final de máster a mi esposa y a mis padres

Fernando André de Oliveira Penteado

Agradecimientos

Primeramente, agradezco a Dios y a Meishu Sama por el permiso de poder cumplir con mi sueño de estudiar en el extranjero, realizar un máster y vivir en Europa.

Agradezco a mi esposa Sandra, por acompañarme y apoyarme en todo lo que hago.

Agradezco a mis padres, Luiz (*in memoriam*) y Rita (*in memoriam*) por la base sólida que me dieron y por haberme enseñado la importancia de los estudios en nuestras vidas. Siempre les recordaré.

Agradezco al Ministro Marcelo, por el apoyo espiritual durante toda mi trayectoria.

Agradezco a Sergi por haberme ayudado en mi adaptación y por haberme enseñado sobre la cultura local, además de todo el apoyo que me brindó.

Agradezco a Nacho y Toni, por la oportunidad y la confianza colocada en mi persona.

Agradezco a la Universidad Politécnica de Valencia y a todos los profesores del Máster Universitario en Ingeniería y Tecnología de Sistemas de Software.

En especial, agradezco a mis tutores, María Carmen y Juan por el apoyo en este último año de desenvolvimiento del presente trabajo final de máster, especialmente agradezco a María Carmen por el apoyo durante todo el máster.

Muchas gracias a todos.

Fernando

Resumen

La variabilidad en modelos de procesos permite derivar un conjunto o familia de procesos a partir de un modelo de referencia, el cual contiene toda la parte común a la familia. Esta parte puede ser instanciada para generar un representante de esa familia. Existen diversas técnicas para definir familias de procesos, en la tesis final de máster combinaremos dos aproximaciones: utilizaremos la aproximación propuesta en el proyecto PESOA que permite definir diversas relaciones entre actividades de un modelo de procesos, en particular una actividad puede ser extendida añadiendo más componentes al subproceso que es extendido; la aproximación PROVOP, que permite marca puntos de ajustes en el diagrama. PESOA utiliza estereotipos a la idea de UML 2.0 y las actividades pueden recibir estereotipos con las etiquetas: abstract, null, optional y default. Provop utiliza una notación propia en el que adiciona un diamante negro para marcar los puntos de ajuste. El objetivo de este trabajo de fin de máster es crear un editor web de modelos de procesos en BPMN (Business Process Modelling Notation) que permita incorporar las características propuestas en PESOA y PROVOP. De esta forma los modeladores podrán definir su familia de procesos, almacenarla en un repositorio para posteriormente editarla y eventualmente generar un representante de la familia de procesos.

Resum

La variabilitat en models de processos permet derivar un conjunt o familia de processos a partir d'un model de referencia, el qual conté tota la part comú a la familia. Aquesta part pot ser instanciada per a generar un representant d'eixa familia. Existeixen diverses tècniques per a definir families de processos, en la tesi final de màster en la tesi final de màster combinarem dos aproximacions: utilitzarem l'aproximació proposta en el projecte PESOA que permiteix definir diverses relacions entre activitats d'un model de processos, en particular una activitat pot ser estesa afegint més components al subprocés que es estés; l'aproximació PROVOP, que permet marca punts d'ajustos en el diagrama. PESOA utilitza estereotips a la idea de UML 2.0 i les activitats poden rebre estereotips com les etiquetes: abstract, null, optional i default. Provop utilitza una notació pròpia en què addiciona un diamant negre per a marcar els punts d'ajust. L'objectiu d'este treball de fi de màster és crear un editor web de models de processos en BPMN (Business Process Modelling Notation) que permeta incorporar les característiques propostes en PESOA i PROVOP. D'aquesta forma els modeladors podran definir la seua familia de processos, emmagatzenar-la en un repositori per a posteriormente editar-la i eventualment generar un representant de la familia de processos

Abstract

Process models variability allows to derive a set or family of processes from a reference model, which contains all the common components of the process family. This part can be instantiated to generate a representative of that process family. There are various techniques to define families of processes, in this final master's thesis we will combine two approaches: the approach proposed in the PESOA project that allows defining various relationships between activities of a process model, in particular an activity can be extended by adding more components to the sub-process that is extended; and the PROVOP approach, which allows setting adjustment points on the diagram. PESOA uses stereotypes to the idea of UML 2.0 and activities can receive stereotypes with the tags: abstract, null, optional and default. PROVOP uses its own notation in which it adds a black diamond to mark the adjustment points. The objective of this master's thesis is to create a web editor of process models in BPMN (Business Process Modeling Notation) that allows incorporating the features proposed in PESOA and PROVOP. In this way, modelers will be able to define their family of processes, store it in a repository to later edit it and eventually generate a representative of the family of processes.

ÍNDICE

Índice de Ilustraciones

Índice de Tablas

Capítulo 1: Introducción

En el presente capítulo se introduce la motivación para este trabajo de fin de máster, los objetivos perseguidos, la planificación del trabajo y la estructura de esta memoria.

1.1 Motivación

Un proceso de negocios es un conjunto de actividades coordinadas que se realizan en un ambiente técnico y organizacional [23]. También puede ser entendido por un conjunto de actividades que recibe uno o más tipos de entradas y entrega una salida en forma de valor al cliente. [23] Estas actividades en conjunto son el objetivo del negocio [23]. Cada proceso de negocio es asignado a una organización específica, pero que se puede interactuar con los procesos de otras organizaciones.

Los procesos de negocio se modelan utilizando una notación específica para tal efecto, llamada Business Process Model and Notation (BPMN), que ha sido creada por el Business Process Mangement Initiative, pero que actualmente es mantenida por el Object Management Group. Su primera versión fue publicada en 2007 y ha ido evolucionando hasta su versión actual, que se encuentra en la 2.0.2 y ha sido lanzada en 2014 [1].

Al modelar un proceso, un analista puede encontrarse con que diversas partes de este modelo sean semejantes, o, muchas veces, idénticas. Además, puede también variar según el contexto en el que se encuentre, según modificaciones de los requisitos, según la evolución de la aplicación, o para incluir nuevos roles o nuevas estrategias de negocio [22]. La existencia de procesos semejantes que pueden variar según el contexto u otros factores se llama Variabilidad de Procesos.

Existen dos formas posibles para representar la variabilidad de procesos utilizando los métodos tradicionales de modelado (como BPMN) [8]: la primera es que el analista cree diversos modelos para representar cada una de las situaciones en las que se tiene que utilizar un proceso especifico, haciendo con que exista la replicación innecesaria de los mismos fragmentos de modelo [8]; o diseñando un modelo extremamente complejo abarcando todas las condiciones, haciendo que el modelo se torne difícil de leer, comprender y mantener en cada una de las condiciones específicas del proceso [8].

Teniendo esto en cuenta, diversas estrategias para modelar variabilidad de procesos han sido propuestas en los últimos años, muchas con la idea de crear un único modelo que soporte todas las variabilidades. Este tipo de abordaje es llamado Modelado de Procesos Personalizables [8].

Una de estas propuestas es PESOA (Process Family Engineering in Service-Oriented Applications), que propone adaptar el concepto de estereotipos de la especificación del UML2 al BPMN, haciendo que cada actividad, asociación o artefacto pueda tener un estereotipo asignado [10].

Otra propuesta es Provop framework (PROcess Variants by OPtions), que es un abordaje que permite la configuración de la estructura del proceso para lograr la variabilidad, en el que familias de procesos son compuestas de variabilidades de procesos, pueden ser creadas desde un modelo de proceso base, aplicando un conjunto predefinido de cambios estructurales llamados adaptaciones, que se pueden añadir, modificar o borrar actividades o fragmentos de procesos en algunos puntos de referencias conocidos como puntos de ajustes [10].

Además de estas dos, existe la propuesta de dependencias verticales y horizontales [9], que propone que existen procesos que necesiten de que pasos de otros procesos ocurran para que se avance, y esta notación permite hacer una ligación entre dos procesos distintos.

En el artículo [10], el autor propone un abordaje mixto entre PESOA, Provop, más una notación propia que ha sido creada para la comprensión de dependencias verticales, para crear una notación de modelado para la resiliencia urbana.

1.2 Objetivos

En el presente trabajo de fin de máster, el objetivo es desarrollar una herramienta web que permita modelar la variabilidad de procesos y composición de los mismos de forma dinámica.

Para lograr este objetivo se pretende:

- Analizar las propuestas ya existentes
- En caso de no existir una propuesta semejante, diseñarla
- Desarrollar la herramienta de soporte
- Validar la herramienta a través de un caso de estudio

1.3 Planificación

Para el desarrollo de este trabajo final de máster ha sido aplicada la siguiente planificación:

1. Análisis de estado del arte: revisar en la literatura científica la existencia o inexistencia de herramientas que cumplan con la propuesta del trabajo;
2. Análisis del trabajo a ser hecho: revisar cuál es la necesidad y que se puede hacer en un determinado período de tiempo;
3. Desarrollo del trabajo: programación de la herramienta;

4. Escribir la memoria: condensar el trabajo en la memoria a ser entregada.

De esta forma, ha sido planificado que el análisis del estado del arte se realizariá en 3 semanas. Para el análisis del trabajo a ser hecho se necesitaría de una semana. Para el desarrollo del trabajo ha sido utilizada la metodología scrum (cuyos conceptos serán introducidos en el capítulo 4) y se ha dividido en 4 sprints, cada uno con duración de 3 semanas.

1.4 Estructura de la memoria

La memoria está dividida en 6 capítulos.

- Capítulo 1: Introducción del presente trabajo de fin de máster.
- Capítulo 2: Estudio del estado del arte sobre herramientas y variabilidad de procesos.
- Capítulo 3: Visión global del trabajo de fin de máster, introducción de conceptos sobre los cuáles se desarrollarán la herramienta.
- Capítulo 4: Desarrollo de la herramienta propuesta.
- Capítulo 5: Implementación de un Caso de estudio para la validación la herramienta.
- Capítulo 6: Conclusión y los trabajos futuros.

Capítulo 2: Estado del Arte

En el desarrollo del presente capítulo, se presentará el estudio del estado del arte, que consiste en la revisión de la literatura científica existente para verificar el estado actual en el que se encuentra la presente rama científica. De esta forma, este capítulo está formado por la introducción de la metodología utilizada para la revisión bibliográfica, la investigación y los resultados de la revisión sistemática.

2.1 Metodología

Para desarrollar el trabajo de estudio del estado del arte, fue utilizada la metodología de revisión sistemática. La revisión sistemática es una metodología de búsqueda que tiene el propósito de poder reducir el sesgo, identificando, valorando y sintetizando los estudios relevantes de un tema específico [20].

Una revisión sistemática puede ser dividida en 6 pasos: [7]

1. Definición de la pregunta a ser buscada y el criterio de elegibilidad: en este paso, se define el punto de partida de la revisión sistemática, proponiendo cuál será la pregunta (o hipótesis) en la que se tiene que centrar el estudio, además se define cuál será el criterio en el que se basará para categorizar un artículo como siendo de interés o no.
2. Búsqueda por estudios: en este paso se busca por los estudios en los repositorios existentes, además de otras posibles fuentes que puedan ser consideradas.
3. Selección de los estudios: una vez que se buscan los artículos, los mismos son seleccionados siguiendo los criterios definidos en el punto 1.
4. Extracción de los datos: en este paso se categoriza los estudios seleccionados.
5. Sintetización de los datos: en este paso lo que se hace es coger los datos, y presentarlos de alguna forma, sea como tablas o gráficos.
6. Interpretación de los resultados: en este paso se interpreta los resultados obtenidos con el propósito de obtener una conclusión.

Esta metodología es la que será utilizada para el desarrollo del estudio del estado del arte.

De esta forma, la pregunta a ser buscada será “¿existe una herramienta que implemente las notaciones de variabilidad de procesos?”. El criterio de elegibilidad es el que si lo que es propuesto en los artículos decir, de alguna forma, una herramienta que tenga una notación para variabilidad de procesos.

La ejecución de la revisión sistemática es explicada en el siguiente apartado.

2.2 Revisión sistemática

Previo al desarrollo de la herramienta de estudio del presente trabajo final de máster, se ha realizado una investigación profunda sobre del estado del arte, de manera que se confirme la existencia o inexistencia de una herramienta similar o idéntica a la propuesta del presente trabajo.

Para estudiar el estado del arte, utilizamos como punto de partida el artículo: Business process variability modeling: A survey (La Rosa, M; et al) [8], artículo que desarrolla el estudio del estado del arte del modelado de la variabilidad de procesos hasta el año 2017, sin embargo, en dicho estudio no se encontraron referencias que cumplan o que respondan a los objetivos buscados con la herramienta que el presente trabajo busca proponer.

A partir de este resultado, se determinó ampliar la búsqueda del estado del arte iniciada por el artículo mencionado, realizando la investigación de referencias que puedan existir de entre los años posteriores al 2017, buscando en los repositorios IEEE, ACM Digital library (ACM DL), Scopus, World of science (WOS)

Como punto partida, fue definido que sería utilizada la misma cadena de búsqueda, filtrando los años 2017 a 2020, añadiendo palabras clave que estuviesen relacionadas a “herramienta”. De esta manera se identifican el número de artículos generados en este período de tiempo obteniendo los resultados que se muestran en la Tabla 1, Cadena 1.

En segunda instancia, se ha introducido un filtro añadiendo la palabra “tool” a la cadena obteniendo los resultados que se muestran en la Tabla 1, Cadena 2.

Para intentar tener nuevos resultados, fue añadido también la palabra “framework” con la conectiva OR junto a “tool” obteniendo los resultados que se muestran en la Tabla 1, Cadena 3.

CADENA 1 ((customization OR customizability OR customizable OR flexibility OR flexible) OR (variation OR variability) OR (configuration OR configurability OR configurable)) AND "business process"	• IEEE – 180 artículos • ACM DL – 1399 artículos • Scopus - 13020 artículos • WOS – 688 artículos
CADENA 2 ((customization OR customizability OR customizable OR flexibility OR flexible) OR (variation OR variability) OR (configuration OR configurability OR configurable)) AND "business process" AND "tool"	• IEEE – 11 artículos • ACM DL – 890 artículos • Scopus - 6172 artículos • WOS – 68 artículos
CADENA 3 ((customization OR customizability OR customizable OR flexibility OR flexible) OR (variation OR variability) OR (configuration OR configurability OR configurable)) AND "business process" AND ("tool" OR "framework")	• IEEE – 48 artículos • ACM DL – 1148 artículos • Scopus - 10748 artículos • WOS – 109 artículos

Tabla 1 Cadenas de Búsqueda y Resultados. Fuente: Elaboración propia.

Finalmente, una vez obtenido los resultados de las tres cadenas, se ha decidido que se utilizaría la segunda cadena de búsqueda, por tener una cantidad más razonable de artículos.
Una vez obtenido el resultado de los filtros aplicados, se realizó el trabajo de filtro manual, buscando, de entre los títulos de los 7.141 artículos encontrados, similitudes a nuestra herramienta de estudio.

El resultado de esta búsqueda manual es de 93 artículos que podrían obtener títulos que estén relacionados con la temática.

Enseguida, siguiendo con la búsqueda manual, se ha añadido un nuevo filtro: la lectura manual de los resúmenes de cada uno de los 93 artículos encontrados en el párrafo anterior.

En este punto se ha realizado la clasificación de los artículos etiquetándolos de la siguiente forma:

- Propuesta de implementación,
- Implementación,
- Optimización de alguna herramienta o algoritmo existente o si eran relacionados a un contexto especifico.

Adicionalmente, en esta etapa también se identificaron aquellos artículos que pudieran tener una herramienta que podría responder al objetivo del presente trabajo.

En consecuencia, de los 93 resumes estudiados manualmente, 22 tienen relaciones directas con herramientas según los parámetros explicados anteriormente.

En la Tabla 2 se listan esos 22 artículos mencionados, indicando:

- Autores,
- Año de publicación
- Fuente de origen,
- El número de referencia que será utilizado para los gráficos siguientes.

Referencia	Artículo	Autor	Año	Fuente
1	Automatic tool support possibilities for the text-based S-BPM process modelling methodology	Elstermann, Matthes Heuser, Tobias	2016	ACM
2	ASSEMBLE: A collaborative business process development tool	Afzal, Ayesha, et al.	2018	ACM
3	Designing a Simple and Flexible Workflow Management System for Collective Decision Making	Pham, Hanh	2019	ACM
4	S-BPM implementation in CUBA platform for rapid application development	Lednev, Andrey, et al.	2019	ACM
5	A multicriteria method based approach to the BPMM selection problem	Lima, Eliana , et al.	2017	IEEE
6	Automated generation of variants in business process families based on the Common Variability Language (CVL)	Calegari, Daniel, et al.	2019	IEEE
7	Towards supporting modeling variability in E-learning application: A case study	Azouzi, Sameh, et al.	2018	IEEE
8	An approach implementing template-based process development on BPMN	Cui, Xiaofeng	2017	IEEE
9	Compliance Analysis of Configurable Business Process Model Based on Extend CTL	Huang, Yiwang, et al.	2017	IEEE
10	Research and implementation of smart substation configuration file management system based on BPM process engine	Luo, Mei Ling, et al.	2017	IEEE
11	Towards Automated and Fine-Grain Reuse of Configurable Business Process Models	Marwa, Mdimagh Sami, Bhiri	2019	IEEE
12	Dynamic business process generation and verification	Wu, Budan, et al.	2016	IEEE
13	Towards rule-based pattern perspective for BPMN 2.0 business process models	Kluza, Krzysztof Nalepa, Grzegorz J.	2016	IEEE
14	A semantic framework supporting business process variability using event logs	Yongsiriwit, Karn, et al.	2016	IEEE
15	Strategies to automatically derive a process model from a configurable process model based on event data	Arriagada-Benítez, et al.	2017	Scopus
16	A framework for generating domain-specific rule for process model customization	Mani, Neel, et al.	2017	Scopus
17	A novel tool for configurable process evolution and service derivation	Sbai, Hanae, et al.	2019	scopus
18	Supporting customizable business process models using graph transformation rules	Geist, Verena, et al.	2016	Scopus
19	BVCCON-TOOL: A modeling tool to support dynamic business process configuration approach	Pereira, Tarcísio, et al.	2016	Scopus
20	BPFlexTemplate: A software tool to derive flexible process model templates	Ilahi, Latifa, et al.	2017	wos
21	Measuring Business Process Consistency Across Different Abstraction Levels	Zhang, Xuewei, et al.	2019	wos
22	A knowledge-based approach to manage configurable business processes	Benallal, Wehbi, et al.	2020	wos

Tabla 2 Listado de artículos de la revisión sistemática. Fuente: Elaboración propia

A continuación, en la Ilustración 1, se han dividido los artículos de acuerdo con la temática presentada, identificándolos por su número de referencia.

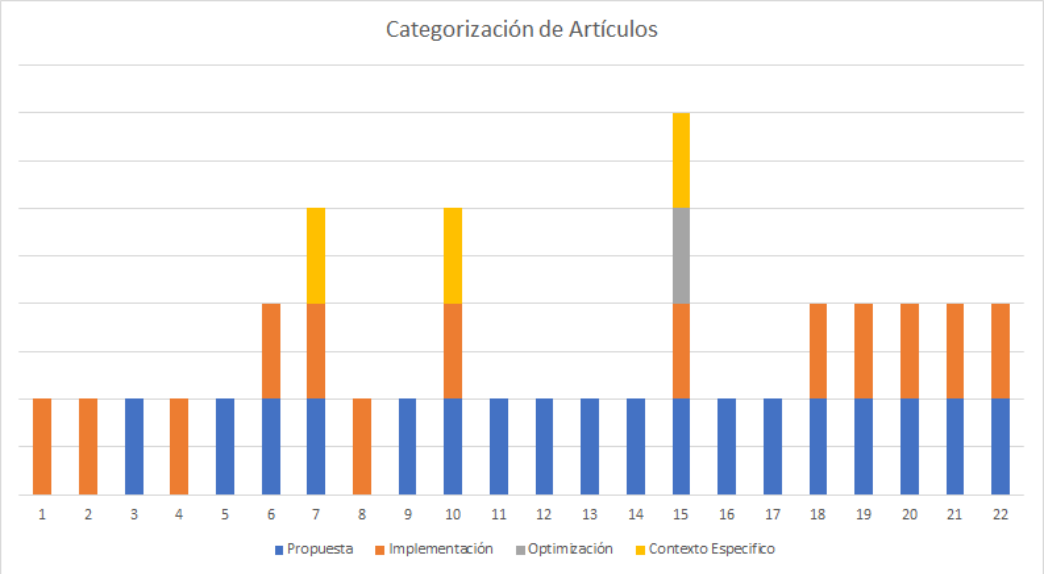

Ilustración 1 Categorización de Artículo. Fuente: Elaboración propia

En esta Ilustración 2, se ilustran el total de artículos por categoría.

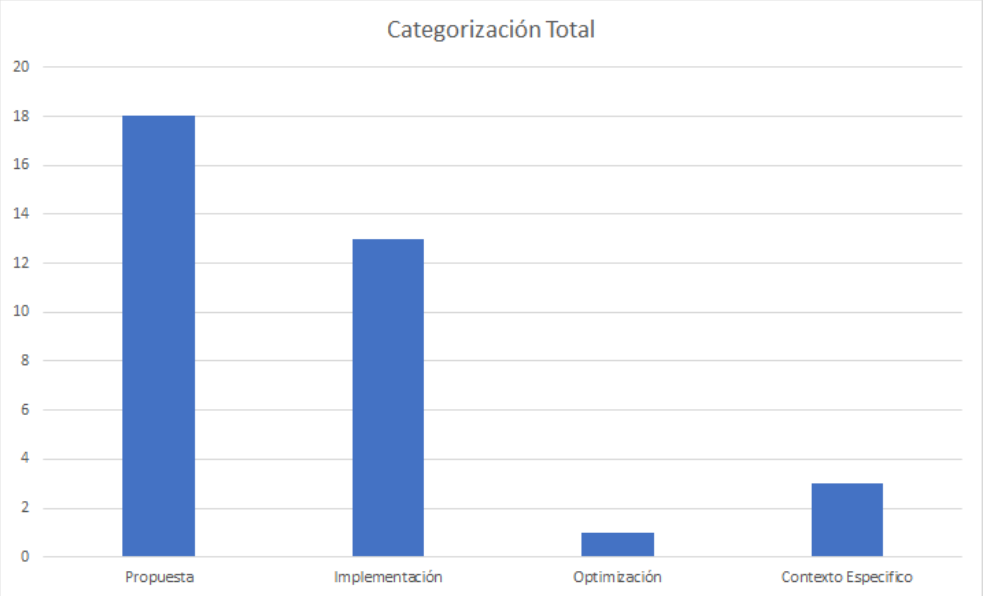

Ilustración 2 Categorización Total. Fuente: Elaboración propia

A continuación, en la Ilustración 3, se presentan el porcentaje del total de la categorización que hay de artículos que están presentes en cada categoría.

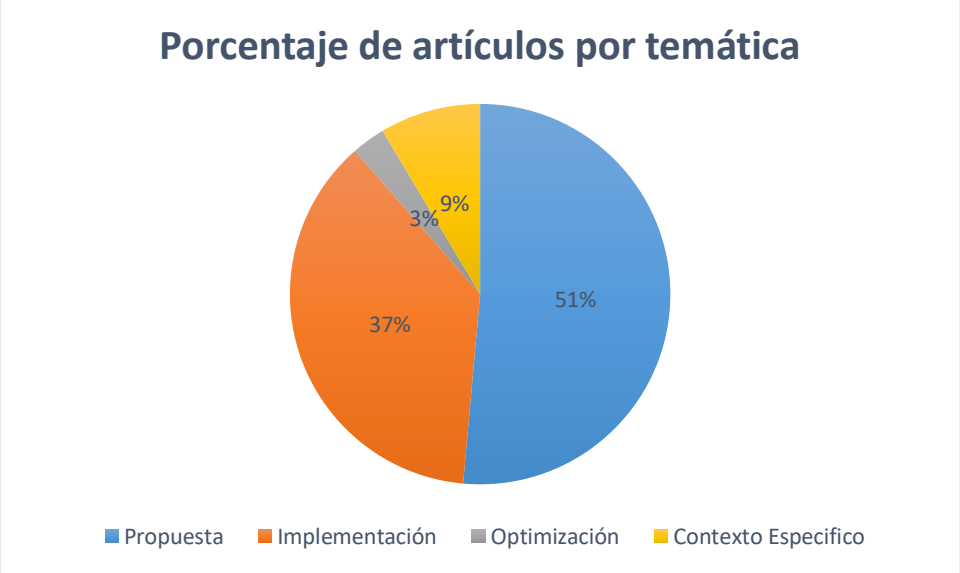

Ilustración 3 Porcentaje de artículos por temática. Fuente: Elaboración propia

En consecuencia, de los 22 artículos se identifica que:

- 18 artículos tienen propuestas de implementación,
- 13 artículos tienen las implementaciones,
- 1 artículo establece la optimización de un algoritmo
- 3 artículos están relacionados a un contexto específico.

A su vez, de estos 22 artículos:

- 3 artículos están relacionados a herramientas sobre modelado de variabilidad de proceso de negocio [11][14].

A continuación, se ilustra un estudio de cada una de estas herramientas.

2.2.1 BPFlexTemplate: a software tool to derive flexible process model templates

Este artículo presenta la creación de una herramienta web que puede comparar diversos procesos y generar un modelo flexible de este proceso [9].

La propuesta es que el modelo final tenga no solamente la inclusión de elementos de proceso que son comunes a todas las variantes comparadas, sino que también elementos no comunes que resultan de los ajustes realizados a cada variante [9].

Su propósito es que el modelo final permita a los ingenieros que utilicen un modelo de proceso similar en toda organización y, así, controlar todos los procesos de negocio de

forma eficiente y permitir que las unidades de la organización se beneficien de los ajustes realizados a lo largo del tiempo [9].

La herramienta que creada sigue la propuesta BPFlexTemplate, que ha sido idealizada por los autores, que incluye un modelo de ciclo de vida de proceso personalizado, buenas prácticas y un algoritmo con el propósito de generar una plantilla de modelo de proceso, controlar la proliferación de variables de procesos y mejorar el modelado de procesos de negocio [9].

En la herramienta, el ingeniero de procesos puede:

- Seleccionar modelos BPMN de nuevos modelos BPMN cargados o de plantillas modelo generadas (Ilustración 4 y 5),
- Comparar dos o más modelos BPMN,
- Producir la distancia del grafico final de edición (Ilustración 6)
- Explotar la comparación de estos resultados y derivar una plantilla modelo. (Ilustración 7) [9]

La comparación de los modelos BPMN utilizan los siguientes algoritmos que pueden ser elegidos por los usuarios:

- Tipo de similitud (entre similitud estructural, nodos coincidentes y comportamental);
- Algoritmos de similitud para comparación de elementos de procesos (entre sintáctico, distancia de grafos, semántica, atributos, tipos, contextuales);
- Clasificación (entre 0 y 100), en el que el ingeniero puede seleccionar el rango en el que un elemento del proceso es considerado similar a otro, y, entonces, ser clasificado como perteneciendo a una parte común del proceso. [9]

La derivación de una plantilla modelo es lo que permite a un usuario visualizar el modelo de plantilla generado anteriormente. En este momento se puede diferenciar cuales procesos son considerados comunes y cuales son considerados flexibles y posibles de ser ejecutados según algún mecanismo de flexibilidad. [9]

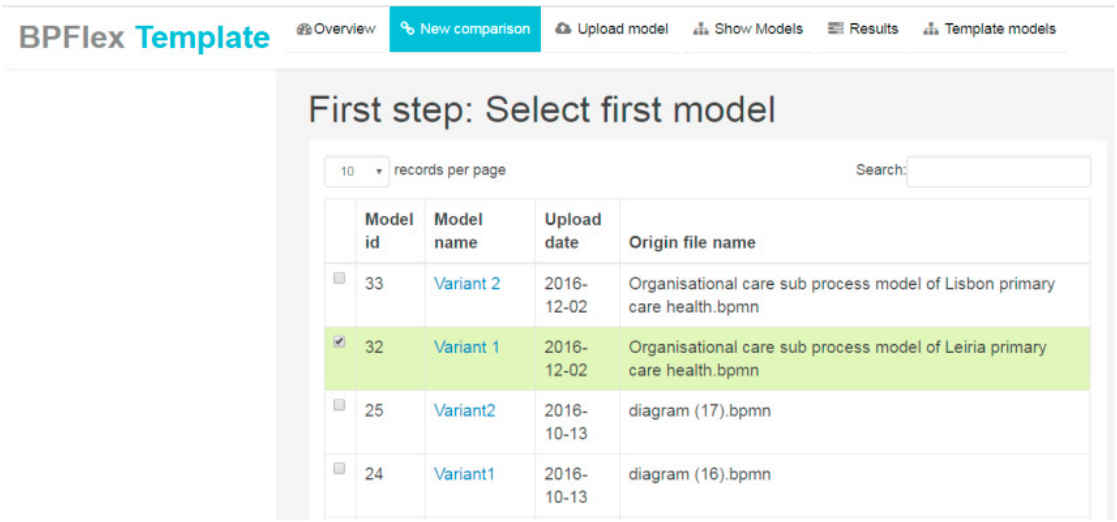

Ilustración 4 Paso 1 en el BPFlexTemplate Tool. Fuente: [9]

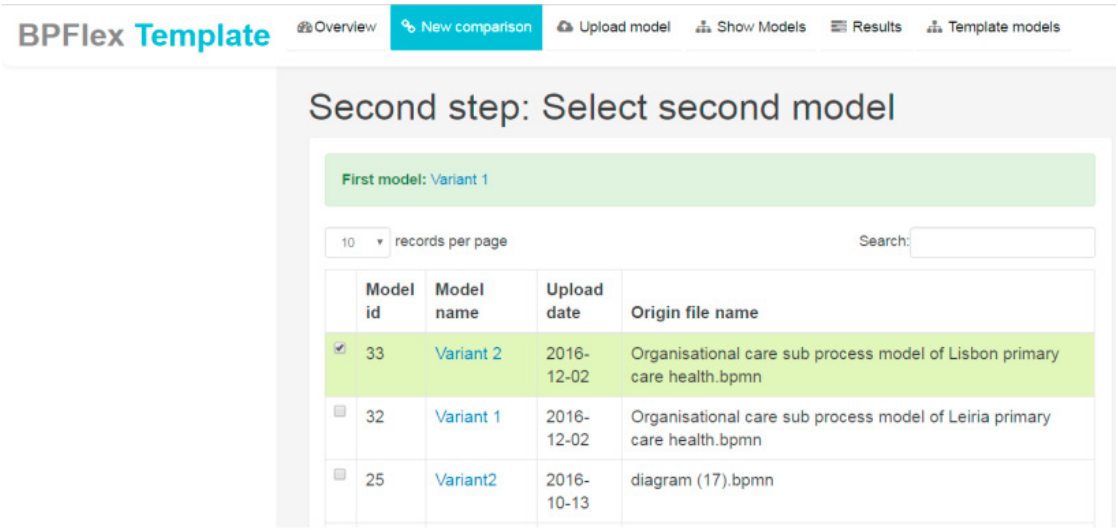

Ilustración 5 Paso 2 en el BPFlexTemplate Tool. Fuente: [9]

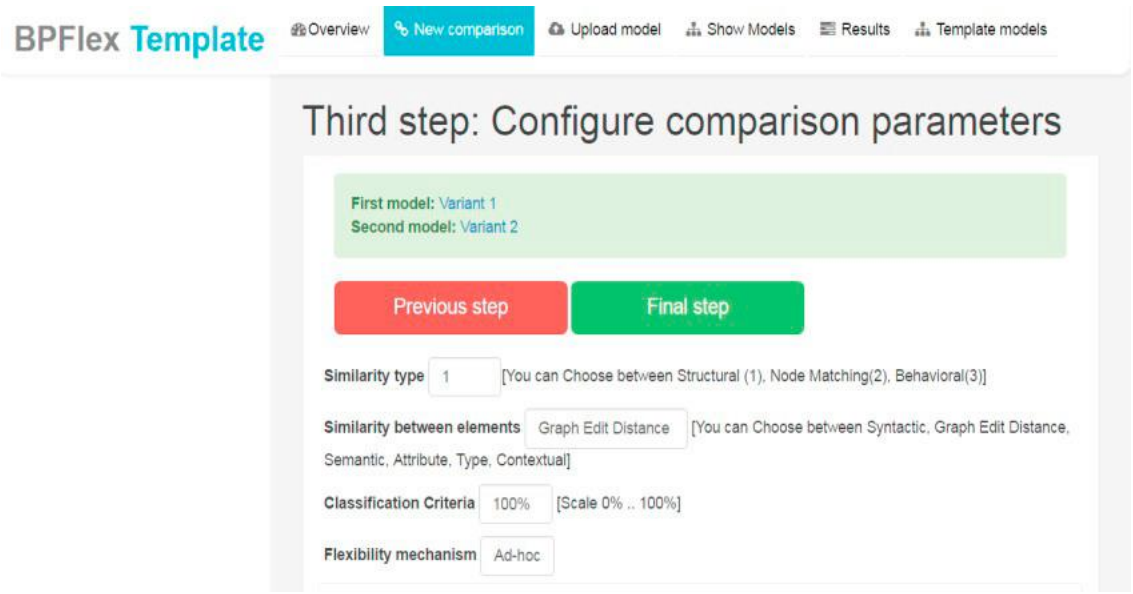

Ilustración 6 Paso 3 en el BPFlexTemplate Tool. Fuente: [9]

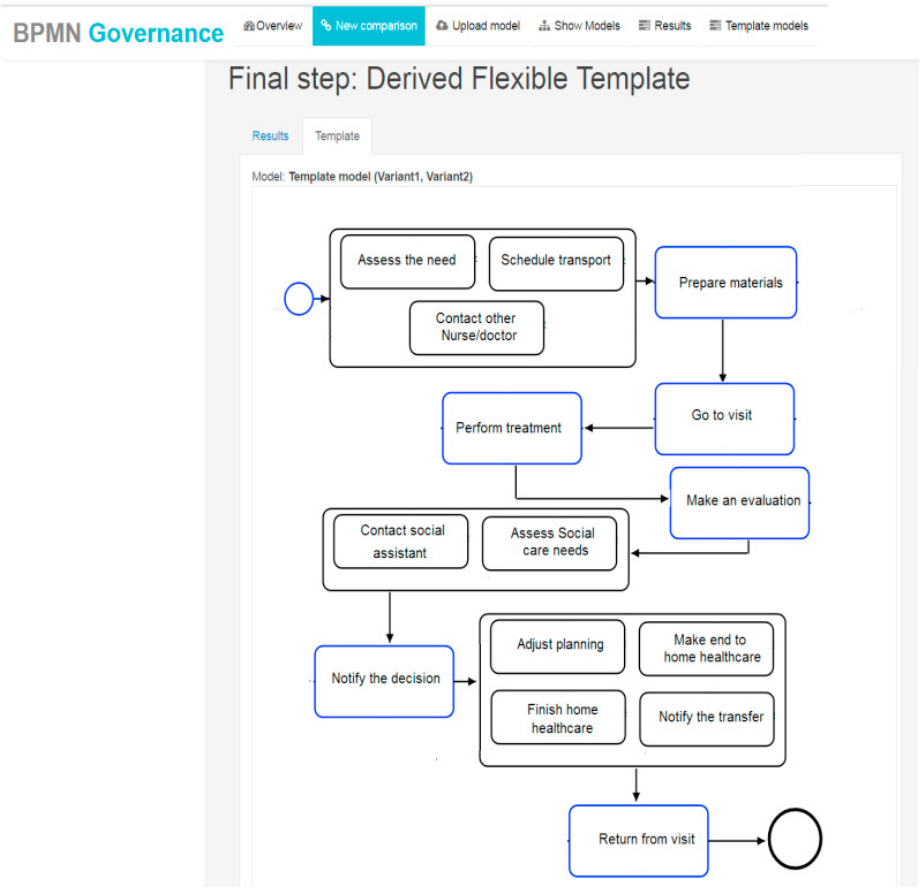

Ilustración 7 - Paso final en el BPFlexTemplate Tool con el modelo derivado. Fuente: [9]

2.2.2 BVCCON-TOOL: A modeling tool to support dynamic business process configuration approach.

La propuesta BVCCON presentado en el artículo: A modeling tool to support dynamic business process configuration approach [11], amplía la variabilidad de procesos con requisitos no funcionales e información contextual.

Su configuración utiliza información del contexto para identificar posibles casos de cambio, y la información de contexto también puede ser usada para definir restricciones esenciales para el proceso [10]. Así, las informaciones de contexto son utilizadas como criterio cualitativo para guiar los modelos de procesos de negocio y la aplicación de contexto se responsabiliza por los cambios de entorno [11].

La propuesta BVCCoN posee cinco tareas principales: [11]

1. Obtener la variabilidad: se obtiene la variabilidad analizando un modelo BPMN de referencia, con la ayuda de un conjunto de preguntas y respuestas se puede identificar puntos de variabilidad en la forma que se realiza el proceso. [11]
2. Describir la variabilidad: se expresa utilizando puntos de variabilidad [11]
3. Analizar el contexto: Asociar los contextos a las variantes del proceso. La información contextual está representada por expresiones lógicas que pueden evaluarse según los datos [11]

4. Enlazar variables de contexto con las variantes: las variables de contexto son utilizadas para describir los atributos de calidad que pueden ser relevante a las partes del proceso [11]
5. Realizar la configuración: generar un nuevo modelo de proceso de negocio como resultado final.

Esa propuesta es considerada compleja, una vez que está relacionada con los modelos de procesos de negocio, modelos de requerimientos no funcionales y modelos información contextual [11].

Todos estos modelos son desarrollados en las tareas 2,3 y 4, debido a eso se concluye que una gran parte de la propuesta está en el desarrollo de modelos y estos modelos siempre estarán conectados [11]. La ausencia de una herramienta que permita integrar estas tareas hace con que el proceso se ralentice, sea difícil de entender y sea pasible de errores humanos.

Con el propósito de facilitar el modelo utilizando la propuesta BVCCoN, fue creada la herramienta BVCCoN-Tool, que implementa las cinco tareas que componen esa propuesta. La Ilustración 8 muestra la interfaz de esta herramienta.

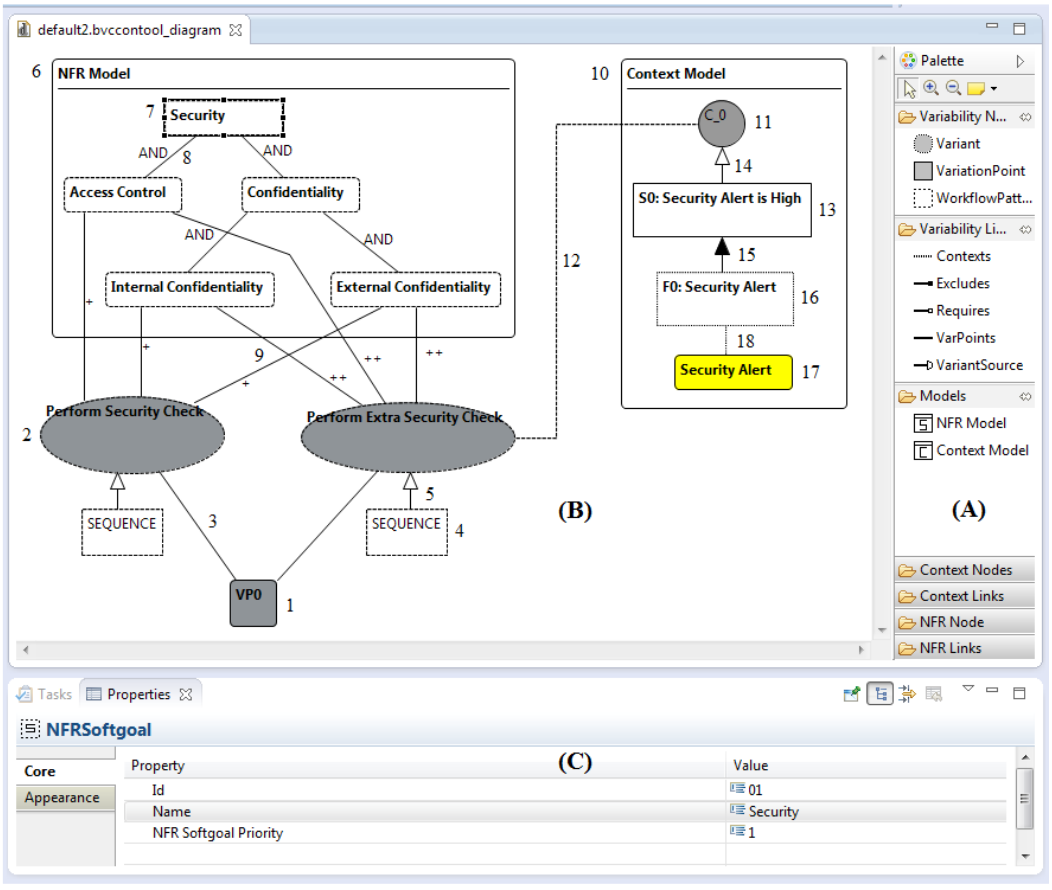

Ilustración 8 Interfaz gráfica del BVCCON-TOOL. Fuente: [11]

2.2.3 A novel tool for configurable process evolution and service derivation

Según (Latifa Ilahi, et al) [14], junto con la mejora y reutilización de procesos de negocio para la introducción de la gestión de la variabilidad, surgió una necesidad de alineación entre negocio y equipo de tecnología.

El estudio hecho por los autores descubrió que las propuestas existentes no permiten la generación de servicios configurables desde procesos configurables.

En este contexto ha sido introducido un nuevo concepto: modelado de procesos basado en servicios. [14] Debido a esto, los autores han estudiado la relación entre los procesos configurables y las aplicaciones empresariales, enfocándose en servicios, con el propósito de construir sistemas de información regidos por procesos que permitan la orientación a servicios. [14]

El surgimiento de la gestión de la variabilidad en los procesos y servicios comerciales hace que los usuarios adopten los procesos configurables en la capa comercial y los servicios configurables en la capa de TI [14].

De esta forma se ha propuesto un abordaje dirigido a modelos para la generación de servicios configurables.

Partiendo de esta propuesta, han desarrollado la herramienta CPMEv, que permite modelar estos servicios configurables. En la Ilustración 9, se muestra la interfaz gráfica de CPMEv.

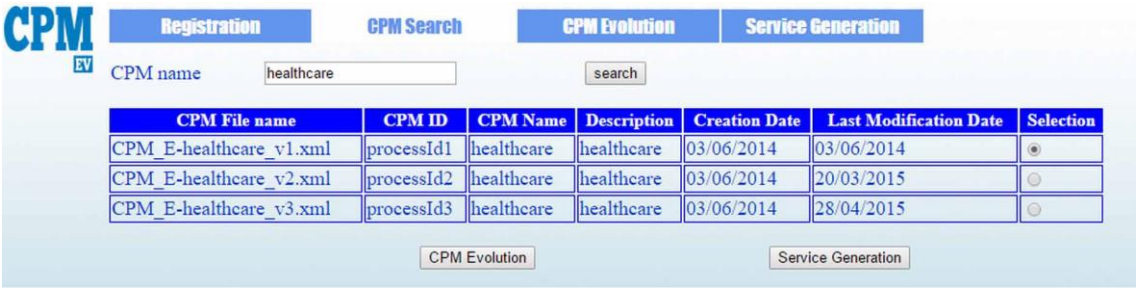

Ilustración 9 Interfaz gráfica del CPMEv. Fuente: [14]

2.3 Conclusión

En síntesis, de estos 3 artículos, ninguno hace referencia a una herramienta que se pudiera utilizar para el objetivo del presente trabajo, o que esté relacionado con la herramienta propuesta.

En consecuencia se concluye, que la idea propuesta por este trabajo final de curso es una idea no preexistente en el medio científico.

2.4 Debilidades de la revisión sistemática:

Se debe tomar en cuenta de que la revisión sistemática ha sido realizada en base a búsqueda de estudios realizados en el idioma inglés, por lo que artículos o estudios publicados en otros idiomas no se han considerado en la búsqueda.

Adicionalmente, se han utilizado apenas los repositorios de artículos ACM Digital Library, IEEE Xplore, ISCRAM, Scopus y Web of Science, en consecuencia, los artículos que no están en ninguno de estos repositorios no se toman en cuenta en la búsqueda.

Finalmente, en la búsqueda manual, en el primer filtro se toma en cuenta sólo los títulos de los artículos, por lo que existe la probabilidad de que no fueran considerados aquellos estudios que podrían haber estado relacionado con nuestro objetivo, pero que el titulo

propuesto por sus actores no hayan hecho referencia a esta herramienta, por lo que no ha podido ser identificado en este filtro.

Capítulo 3: Visión global

En este capítulo se explica, en grandes rasgos, el trabajo que será realizado en la presente memoria. Adicionalmente, se introducirá los conceptos de PESOA, PROVOP y dependencias verticales y horizontales.

Por último, se realiza el estudio de los editores open source de BPMN existentes y los argumentos de elección utilizados.

3.1 Visión global del trabajo

El presente trabajo de fin de máster tiene como objetivo desarrollar una herramienta web que permita modelar variabilidad de procesos utilizando diversas notaciones distintas.

Para tal objetivo, ha sido elegido la solución propuesta por el artículo Penadés, M. C., et al, Building Urban Resilience : A Dynamic Process Composition Approach. [10], en el que se propone un abordaje mixto entre las notaciones PESOA, PROVOP y dependencias verticales y horizontales, permitiendo que se modele utilizando las características de cada una de estas notaciones.

Para lograr este objetivo, se utilizará un editor web open source, que permita modificar y distribuir la solución modificada, y así implementar las notaciones en este editor.

3.2 PESOA

La notación de PESOA ha sido introducida en el artículo: Variability Mechanisms in E-Business Process Families (Schnieders, A., et al) [15] que presenta una notación de variabilidad de procesos para familia de procesos digitales.

Para entender esta notación primero se necesita introducir algunos conceptos.

3.2.1 Familias de procesos

Familias de procesos son conjuntos de procesos de negocio que poseen una similitud bien definida y una variabilidad previamente conocida. [26]

3.2.2 Ingeniería de familias de procesos

Ingeniería de familias de procesos es un paradigma de desarrollo de software que utiliza un subconjunto de subsistemas e interfaces que son partes de una estructura que se basa en las necesidades individuales de los clientes finales y que puede ser desarrollado con este fin [12].

Es clasificado por un llamado “ciclo de vida dual”, en el que la primera parte del desarrollo de familias de procesos, llamado ingeniería de familias de procesos, y

artefactos de desarrollo genéricos, llamados infraestructura de familias de procesos, son desarrollados tomando como punto de partida cuales miembros de la familia de procesos son derivados en la fase correspondiente de la segunda sección, llamada ingeniería de aplicación, de este proceso. [15]

3.2.3 Arquitectura de familias de procesos

La arquitectura de familias de procesos es la referencia para los miembros de la familia de procesos, que define y describe la estructura básica para las aplicaciones de una familia de procesos.

Esta describe cuales son las partes reutilizables de un sistema con sus interfaces y abarca los requisitos funcionales y no funcionales de una familia de procesos. Adicionalmente, describe las técnicas necesarias para reproducir dicha variabilidad y en cuáles puntos de variabilidad dicha técnica debe ser aplicada. [5]

3.2.4 Mecanismos de variabilidad para familias de procesos de negocios

Un diagrama de procesos que contempla variabilidad necesita tener tres adiciones a los diagramas de procesos de negocios [15]:

1. Donde se produce la variabilidad (variation point)
2. Las posibles soluciones para cada variabilidad (variants)
3. Debe ser mostrado el mecanismo de variabilidad usado para la resolución del conflicto entre las variantes

De manera que, Schnieders, A. en su artículo [15], ha propuesto la identificación de los puntos de variabilidad añadiendo el concepto de estereotipos de la notación de UML2 al BPMN.

En UML, los estereotipos son marcaciones que identifican el propósito de ciertos elementos del modelo. Adicionalmente, se puede utilizar para describir que ciertas partes del modelo se difieren en significado o usabilidad de otras partes del modelo. Su sintaxis es el nombre o tipo del estereotipo entre comillas francesas, como en el siguiente ejemplo: <<*nombre estereotipo*>> [21].

A continuación, se ilustra cómo han sido introducidos los siguientes estereotipos: [15]

- <<*VarPoint*>>: representa el punto de variabilidad y puede ser detallado.
- <<*Variant*>>: representa una variabilidad que es suficiente como tal.
- <<*Abstract*>>: representa un comportamiento alternativo, debe ser identificado por una variabilidad específica.
- <<*Null*>>: representa un comportamiento opcional.
- <<*Optional*>>: representa un camino opcional para el proceso, y es una representación mínima del estereotipo <<*Null*>> y necesita una variabilidad específica.
- <<*Default*>>*:* representa la implementación por defecto de la variabilidad.

De manera que, usando la notación propuesta podemos tener un modelo como el siguiente:

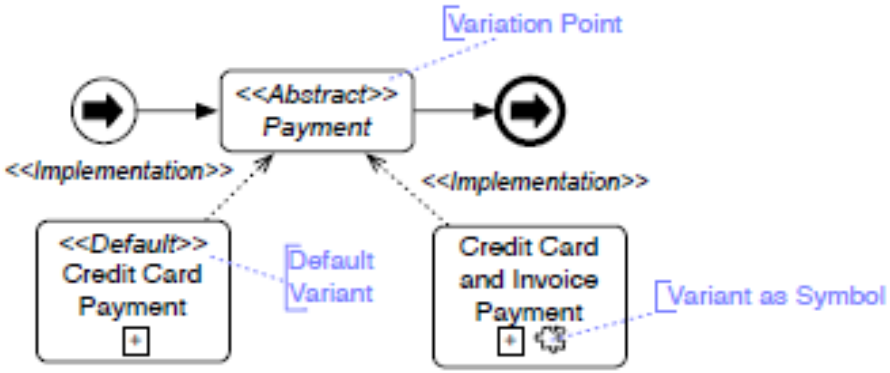

Ilustración 10 Encapsulation in BPMN. Fuente [15]

Interpretando el modelo de la Ilustración 10, tenemos el comportamiento alternativo - marcado por el estereotipo <<*Abstract*>> - (o también se puede entender como un proceso abstracto, que debe tener una implementación especificada en seguida, como por ejemplo un proceso de caja negra) Payment, que es implementado (la implementación es representada por las líneas descontinuas) por el subproceso por defecto de pago por tarjeta de crédito – marcado por el estereotipo <<*Default*>> - e implementado por el subproceso alternativo pago por tarjeta de crédito y facturación (que en el [15] se denota también por el símbolo de "pieza de rompecabezas", pero que en el presente trabajo ser utilizará apenas el estereotipo <<*Variant*>> que tiene el mismo significado).

3.3 PROVOP

PROVOP (PROcess Variants by Options – Variabilidades de procesos por opciones) es un marco de trabajo que no ha sido definido para un formalismo específico de variabilidad, sino para ser un marco genérico para la gestión y el formalismo de la variabilidad de procesos [6].

La Ilustración 11 ejemplifica el funcionamiento de PROVOP. En el modelo del proceso de origen (3a), hay un modelo formado por 5 actividades; A, B, C, D y E, en el que se utiliza el símbolo "diamante negro" (♦) como sintaxis para denotar el punto en el que se ejecutará los ajustes (aquí llamado de "puntos de ajustes").

Tomando como base este proceso, se deriva en otros tres procesos:

1 Variabilidad 1, en el que se borra un fragmento de procesos entre los puntos de ajustes X e Y y se modifica la actividad D;

2 Variabilidad 2, en el que se añade un fragmento de proceso entre los puntos de ajustes X e Y; y,

3 Variabilidad 3, en el que se borra la actividad C y cambia el orden de las actividades. [6]

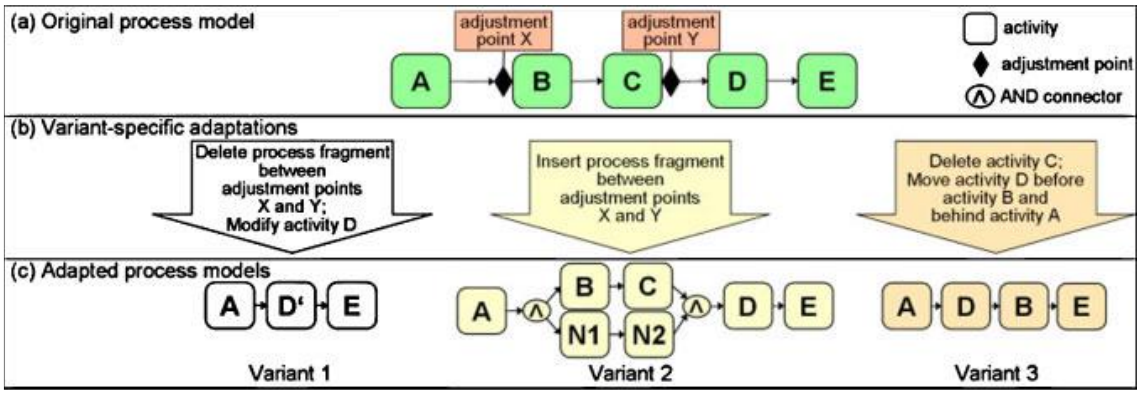

Ilustración 11 Definición General del modelo de variabilidad. Fuente [6]

Adicionalmente, un proceso puede ser modificado aplicando un conjunto de operaciones sobre el que son implementados por PROVOP: [6]

1. INSERT – añadir un fragmento de proceso a un proceso base
2. DELETE – borrar un fragmento de proceso de un proceso base
3. MOVE – mover un fragmento de proceso de un proceso base
4. MODIFY – modificar un atributo específico de un proceso base

Las operaciones de 1 a 3 pueden ser aplicadas a un fragmento de modelo, mientras que la operación 4 es utilizada para cambiar el valor de un atributo específico del proceso base.

Un punto de ajuste puede ser colocado en la entrada o salida de una actividad o un conector en un proceso. Eso permite que ingenieros de procesos puedan marcar donde se crearán nuevos subprocesos, restringiendo las regiones en las que se aplicará la variabilidad. [6]

3.4 Dependencias verticales y horizontales

La notación de dependencias verticales ha sido introducida en el artículo de Penadés, M. C., et al, Building Urban Resilience : A Dynamic Process Composition Approach. [10].

Este artículo hace referencia específicamente al modelado de procesos para el diseño de modelos de madurez de resiliencia en ciudades (Resiliency Maturity Model o RMM). El modelado de RMM se ha realizado usando familias de procesos (concepto ya introducido en el punto 3.3.4). El proceso de desarrollo de resiliencia de una ciudad debe seguir procesos específicos e inicia en el nivel más bajo y va progresando. A cualquier punto de los procesos el estado de madurez puede depender de políticas que hayan sido implementadas con éxito anteriormente [10].

Las políticas no son independientes entre sí, pero algunas dependencias deben ser consideradas cuando se define el plan de acción, además, para seguir al estado siguiente en algún nivel, todas las políticas deben ser implementadas [10]. A esta definición se le llama dependencias horizontales. Sin embargo, también, la activación de algunas políticas en una dimensión puede depender de que algunas políticas en otras dimensiones sean completadas. A esto se llama dependencias verticales.

En el artículo de Penadés [10] se ha definido que las dependencias horizontales serian denotadas por los conectores por defecto de BPMN, y se ha introducido una nueva notación para las dependencias verticales: el hexágono de doble borde y el hexágono de borde grueso.

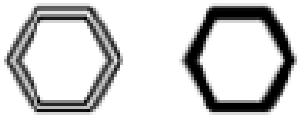

Ilustración 12 Hexágonos de doble borde y de borde grueso. Fuente: [10]

El hexágono de doble borde significa que es el origen de la dependencia vertical, mientras que el hexágono de borde grueso significa que es el destino de la dependencia vertical [10]. Es decir, cuando se encuentra un hexágono de doble borde, en este punto se puede referenciar a otro proceso en el que estaría un hexágono de borde grueso.

A continuación, ilustramos un modelo [10]:

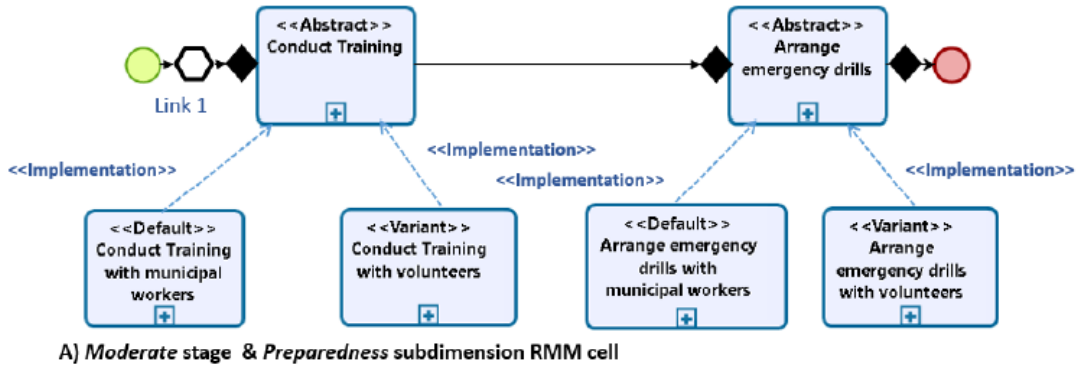

Ilustración 13 Proceso con dependencia vertical de destino. Fuente: [10]

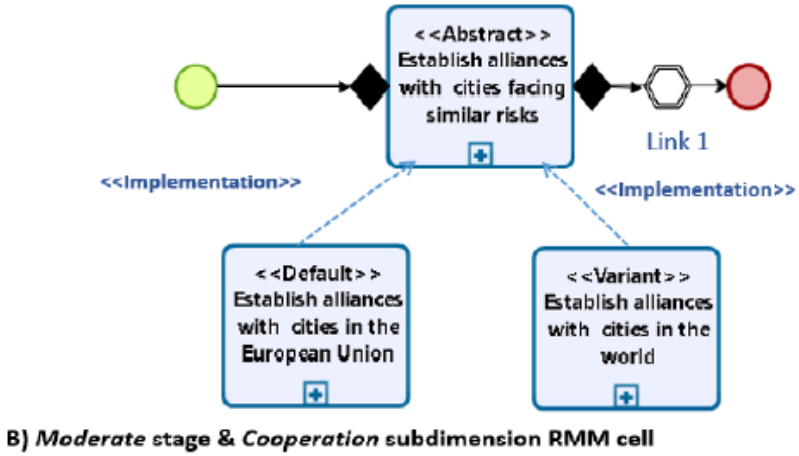

Ilustración 14 Proceso con dependencia vertical de origen. Fuente: [10]

En la ilustración 14, observamos que el link 1, que crea una dependencia vertical al proceso que se encuentra en la ilustración 13.

3.5. Editores open source

Una vez que el proyecto es el desarrollo de un editor online de BPMN 2.0 extendido, el primer paso es tener el propio editor BPMN 2.0. Teniendo esto en cuenta, se ha tomado como base un editor open source que permitie añadir nuevos componentes para, así, implementar el modelo de BPMN extendido que se quiere obtener.

Se han encontrado los siguientes editores open source de plataforma web:

Editor open source:	Características:
Bpmn.io:	Proyecto hecho y mantenido por el equipo de Camunda (un conocido software propietario de modelado en diversas notaciones) en el que se tiene un editor completo de BPMN, en el que todo el código fuente está disponible y su licencia permite cambios, modificaciones y distribuciones según las necesidades del programador final. Este proyecto es hecho en javascript.
Diagrams.net:	Proyecto opensource mantenido por el equipo que tiene el mismo nombre. Es un editor completo de BPMN, UML y otras notaciones con su código fuente disponible, permite también el cambio en su código y su posterior distribución. Este proyecto es hecho en Java.
Simpl4:	Plataforma para creación de aplicaciones controladas por procesos opensource, creada y mantenida por transparent solutions. Es un framework bien completo para ese tipo de aplicaciones. Este proyecto está hecho en Java.
Modelio.org:	Plataforma open source que permite modelar diversas notaciones, como BPMN, UML, ArchiMate y otros. Tiene muchas herramientas de generación de código y publicación de documentos. Es creado y mantenido por Modelio Soft. Está hecho mayoritariamente en Java.

Tabla 3 Editor open source. Fuente: Elaboración propia

Luego de analizar los editores indicados, se ha tomado en cuenta como base del proyecto el bpmn.io, mantenido por Camunda, por las siguientes razones:

- Simplicidad: a pesar de ser un editor completo de BPMN 2.0, es un editor simple. Tiene una interfaz bien limpia y, al contrario de los otros editores, es un editor de apenas BPMN 2.0. Sus desarrolladores no han hecho un editor con varias notaciones, sino que han hecho varios editores para distintas notaciones (además de bpmn.io, Camunda también mantiene DMN-JS).

- Mantenimiento por una empresa conocida: una vez que Camunda es una empresa ya consolidada en el mercado y tiene herramientas de modelado completas, el nombre ha dado confianza para decidir por esta herramienta.

- Documentación clara: a pesar de que en el desarrollo de la herramienta se ha sentido escasa en algunas partes, su documentación está básicamente hecha por ejemplos, en el que se puede observar cómo ha sido hecha alguna solución directamente en el código, y es fácil de entenderlo sin tener la necesidad de una explicación compleja en lenguaje natural.

- Diversos ejemplos: como dicho anteriormente, existen muchos ejemplos de que se pueden tomar como base para el desarrollo de las aplicaciones.

- Foro y comunidad: bpmn.io posee un foro en el que responden principalmente el equipo de mantenimiento de la herramienta, en el que, en muchas situaciones, responden las preguntas con trozos de códigos.

- Arquitectura simple de replicar y mantener: los módulos del editor están bien definidos, permitiendo así tener mayor facilidad en hacer los cambios necesarios y crear las distintas notaciones que necesarias.

- Hecho en javascript y completamente front-end, sin necesidad de procesamiento del lado del servidor.

Una vez elegido el editor, se ha creado el ambiente de trabajo para su uso.

Capítulo 4: Desarrollo

En este capítulo se presentan los trabajos necesarios para el desarrollo de la herramienta. Inicialmente se explica la metodología utilizada. Enseguida, se muestra cómo fue realizado el análisis y la planificación del trabajo. A continuación, se explica los trabajos que anticiparon el desarrollo, como preparación del ambiente de trabajo y pruebas con el editor open source. Finalmente se continúa con las explicaciones del desarrollo de las notaciones PESOA, PROVOP y dependencias verticales.

4.1 Metodología

Para desarrollar la herramienta, se ha decidido utilizar algunas de las metodologías ágiles existentes. Metodologías ágiles son metodologías de desarrollo de proyectos, además de una filosofía de trabajo y organización [13].

En suma, el principio de las metodologías ágiles es dividir los proyectos en proyectos menores, y hacer entregas continuas. Esto permite tener una mejor calidad del proyecto, una vez que el mismo está dividido en problemas menores; un mayor compromiso del equipo; las entregas son más rápidas y permiten una mejora en la productividad [13].

De las metodologías ágiles existentes, se ha utilizado una forma adaptada del Scrum.

Scrum es un marco de trabajo que ayuda a equipos a generar valor por soluciones adaptativas a problemas complejos [17]. Los equipos de scrum normalmente son formados por individuos de diversas áreas y son, por si mismas, autosuficientes [16].

Algunos conceptos importantes de scrum a ser introducidos son:

- Product backlog: es un listado de ordenado de necesidades del producto [17]
- Sprint: es un periodo de tiempo en el que el proyecto es dividido [17]. Durante el sprint es hecho todo el trabajo de desarrollo.
- Sprint Planning: Es la planificación de lo que será hecho en un sprint en el que se define: ¿por qué el sprint será valioso? ¿qué es lo que se puede hacer en este sprint? Y, ¿quién hará las tareas definidas? [17].
- Sprint Review: es la revisión del trabajo hecho en el sprint, en el que se revisa si el trabajo ha sido concluido, si ha sido concluido con éxito y qué es lo que se puede mejorar para el siguiente sprint [17].

Normalmente un equipo de scrum es formado por un product owner, que es el profesional que tiene el conocimiento del producto y es el encargado en poner los ítems al product backlog; un scrum máster, que es el profesional encargado de aplicar las metodologías scrum al equipo y a gestionar el avance del proyecto; y los desarrolladores, que son los encargados en la creación del proyecto. [17]

La adaptación hecha en el scrum para los tres roles existentes ha sido ejecutada por apenas una persona, y no todo un equipo.

4.2 Análisis y planificación del trabajo

4.2.1 Desarrollo del product backlog

El product backlog es una lista ordenada con las peticiones necesarias para desarrollar o mejorar un producto. Es responsabilidad del product owner llenar el product backlog y definir las prioridades en conjunto con el cliente [25].

De esta forma, se ha diseñado el product backlog para el desarrollo de la herramienta:

1 Implementar la notación PESOA

2 Implementar la notación PROVOP

3 Implementar la notación de dependencias verticales y horizontales

4 Imprimir el diagrama en formato XML con la notación BPMN extendida

5 Imprimir el diagrama en formato de imagen

Se ha decidido utilizar la herramienta DevOps de Microsoft como repositorio para poner las informaciones del backlog y para utilizar el protokanban que tiene para ordenar el trabajo. Este protokanban estaba formado por apenas 3 estados: To Do, Doing, Done.

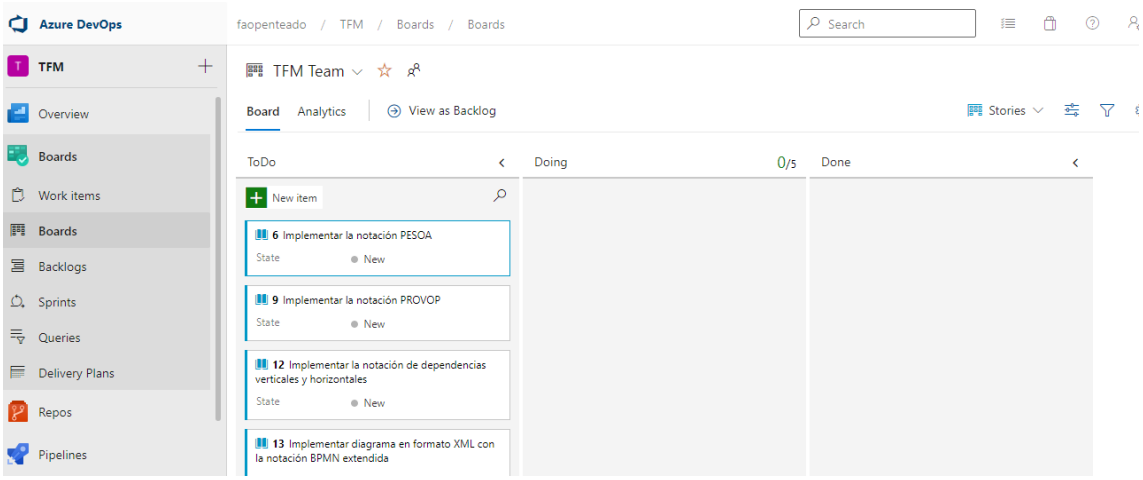

Ilustración 15 DevOps. Fuente: Elaboración propia.

Una vez adicionadas las historias y definido el protokanban, el tablero del devops tiene el siguiente formato:

De esta forma, se tiene el trabajo ordenado y ya definido que es lo que se necesita hacer.

Seguidamente, se ha planificado el número de sprints en el que estas tareas serían realizadas y, en cuáles sprints cada tarea sería implementada.

De tal manera que, ha sido decidido que:

1 En el Sprint 0 se prepararía el proyecto para se pudiera empezar el desarrollo.

2 En el sprint 1 se haría las tareas 1, 4 y 5

3 En el sprint 2 se haría la tarea 2.

4 En el sprint 3 se haría la tarea 3.

4.3 Sprint 0: Preparación del proyecto

4.3.1 Sprint Planning

Para este sprint el objetivo es preparar el proyecto para su desarrollo

4.3.2 Preparación del ambiente de trabajo para el uso del editor

Se ha decidido utilizar el VSCode como IDE (Integrated Development Enviroment o Ambiente de Desarrollo Integrado).

En seguida, se ha preparado el proyecto del bpmn.io para que se pudiera desarrollar las modificaciones. Bpmn.io tiene dos formas de integrar su editor al navegador:

4.3.2.1 Versión pre-empaquetada:

Esta versión permite incorporar el editor/visualizador con una simple referencia a un script utilizando la tag script del html.

En la Ilustración 16, se muestra el visualizador más simple: [5]

```
<!—El contenedor del visualizador -->
<div id="canvas"></div>

<!—la url del codigo en javascript del visualizador -->
<script src="https://unpkg.com/bpmn-js/dist/bpmn-viewer.development.js"></script>
```

Ilustración 16 Código HTML del visualizador simple. Fuente: Elaboración propia

Acontinuación, se añade un script simple en javascript para inicializar los componentes y delimitar en la página qué es el editor, como se muestra en la Ilustración 17 [5].

```
<script>
    // el diagram que se mostrará
    const bpmnXML;
    // BPMNJs es la instancia del visualizador
    const viewer = new BpmnJS({ container: '#canvas' });
    // Importar un diagrama BPMN 2.0
    try {
      await viewer.importXML(bpmnXML);
      viewer.get('canvas').zoom('fit-viewport');
    } catch (err) {
    }
  </script>
```

Ilustración 17 Codigo javascript que inicializa los componentes. Fuente: Elaboración propia.

Este código mostrará un modelo por defecto de ejemplo y el resultado final en pantalla es el siguiente:

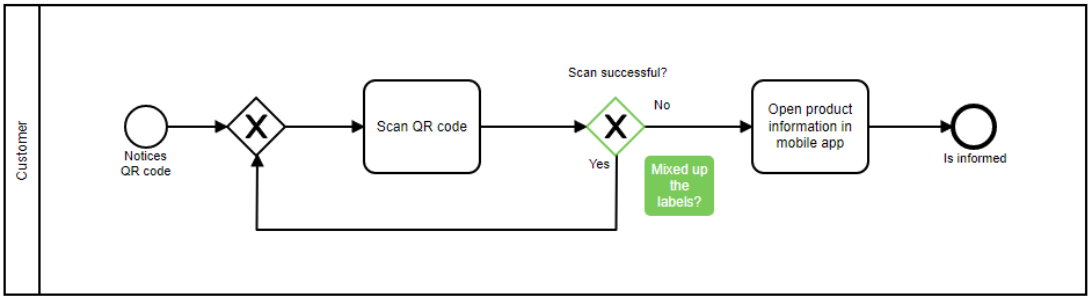

Ilustración 18 Visualizador BPMN de bpmn.io. Fuente: Elaboración propia.

4.3.2.2 Instalación por npm

El abordaje anterior es óptimo para la situación en el que no se desee obtener un editor con características propias, sino un editor simple para la notación BPMN 2.0 en el que se permite las operaciones básicas de creación, edición y descarga del modelo.

Sin embargo, para la situación que se necesita para el desarrollo del presente trabajo, que es la opción de crear un editor de una notación BPMN 2.0 extendida, no es posible utilizar este abordaje.

Pero bpmn.io permite hacer una instalación a través de npm (Node Package Manager). Npm es un repositorio online para publicación de proyectos open source de Node.js, además de un utilitario de línea de comando para adicionar paquetes, gestión de versionamento y gestión de dependencias [24]. Npm viene en conjunto con la instalación del Node.Js, así que primero es necesario instalar el Node.Js en el ordenador para tener acceso al utilitario de línea de comando del npm.

A continuación, teniendo el Node.Js instalado, dentro del propio VSCode, utilizando el terminal que el editor ofrece, se ejecuta el siguiente comando:

```
npm install bpmn-js
```

Este comando instala todos los componentes del bpmn.io, y, así se tiene acceso a todo el código, permitiendo que se pueda añadir características propias al editor:

```
PS C:\Users\fer-p\Desktop\Pruebas TFM> npm install bpmn-js
npm WARN saveError ENOENT: no such file or directory, open 'C:\Users\fer-p\Desktop\Pruebas TFM\package.json'
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN enoent ENOENT: no such file or directory, open 'C:\Users\fer-p\Desktop\Pruebas TFM\package.json'
npm WARN Pruebas TFM No description
npm WARN Pruebas TFM No repository field.
npm WARN Pruebas TFM No README data
npm WARN Pruebas TFM No license field.

+ bpmn-js@8.8.0
added 21 packages from 11 contributors and audited 21 packages in 24.829s
found 0 vulnerabilities
```

Ilustración 19 Comando de instalación del bpmn.io. Fuente: Elaboración propia.

La instalación del bpmn.io adiciona en la carpeta "node-modules" diversos componentes, como se ilustra en la siguiente captura:

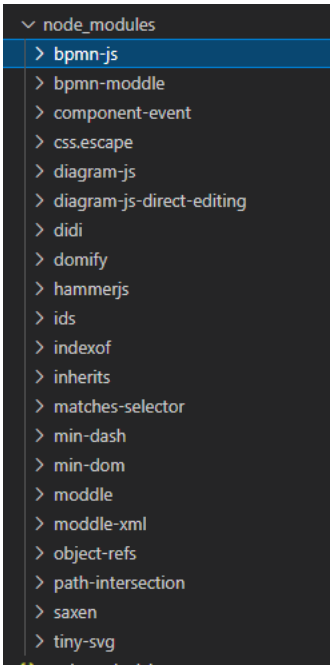

Ilustración 20 Componentes añadidos. Fuente: Elaboración propia.

De estos componentes, los que realmente es necesario modificar directamente son el bpmn-js que tiene todas las características de la notación BPMN y el bpmn-moddle, que tiene las características generales del editor.

Para comprobar que la instalación ha sido ejecutada de forma correcta, se ha creado una página que permite cargar el editor tomando el ejemplo base de bpmn.io.

Primeramente, se ha descargado el proyecto que posee la siguiente estructura (que también se repite en los demás proyectos que aparecerán en este trabajo final de master:

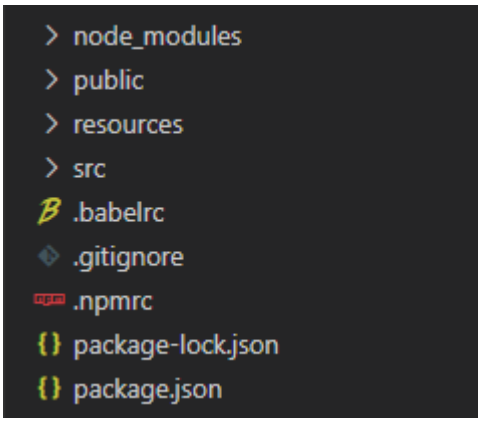

Ilustración 21 Proyecto base bpmn.io. Fuente: Elaboración propia.

- node_modules: posee todos los módulos de node que serán necesarios para la ejecución del proyecto.
- resources: que posee los recursos como imágenes o ficheros auxiliares.
- public: carpeta con los ficheros de la aplicación que serán públicos, como el index.html.
- src: que posee el fichero javascript con el código en javascript para la ejecución de este proyecto simple.

El código en javascript utilizado para el ejemplo es el que se muestra en la Ilustración 22 [4].

```
2   // el codigo donde se encuentra el diagrama BPMN de ejemplo
3   import pizzaDiagram from '../resources/pizza-collaboration.bpmn';
4   // Importar las dependencias del bpmn-js ya instaladas
5   import BpmnViewer from 'bpmn-js';
6   // Inicializar el visualizador en el contenedor de nombre canvas
7   var viewer = new BpmnViewer({
8     container: '#canvas'
9   });
10  // Importar el diagrama previamente inicializado al visualizador
11  viewer.importXML(pizzaDiagram).then(function(result) {
12        viewer.get('canvas').zoom('fit-viewport');
13    }).catch(function(err) {
14  });
```

Ilustración 22 Código javascript para cargar el ejemplo. Fuente: Elaboración propia.

Antes de ejecutar el proyecto, primeramente, es necesario instalar el Webpack.

Webpack es un empaquetador de módulos estáticos para aplicaciones Javascript. Este empaquetador crea internamente un grafo de dependencias a partir de uno o más puntos de entrada, y, en seguida, combina cada módulo que el proyecto necesita en uno o más paquetes [3].

El webpack es necesario para empaquetar todos los paquetes de bpmn-js para poder ser ejecutados por el navegador.

Para utilizar webpack en el proyecto, es necesario instalarlo al proyecto utilizando el siguiente comando en consola:

```
npm install --save-dev webpack
```

Ese comando instalará los paquetes *webpack* y *webpack-sources* al proyecto, permitiendo así poder ejecutar los comandos de ejecución del npm.

Para ejecutar el proyecto primero se debe instalar las dependencias con el siguiente comando en la consola:

```
npm install
```

Y para ejecutar el programa se debe utilizar el siguiente comando en la consola:

```
npm run dev
```

Una vez ejecutado el proyecto, aparecerá el siguiente diagrama en el navegador en la dirección de localhost en el que el node decide apuntar:

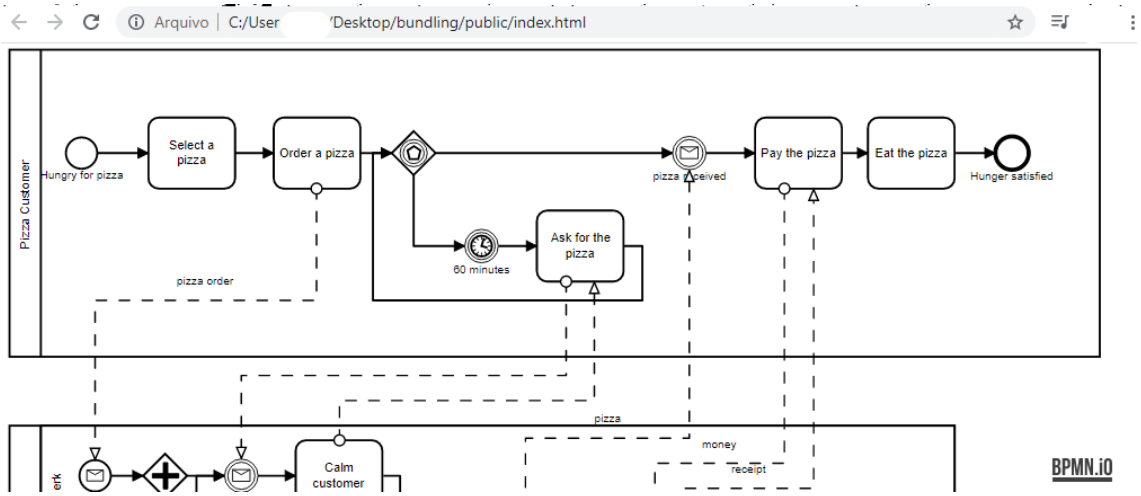

Ilustración 23 - Proyecto ejecutado con bpmn.io utilizando el instalador del npm. Fuente: Elaboración propia.

Por consiguiente, se pude concluir que ha sido posible instalar un proyecto de bpmn.io y ejecutarlo desde el ordenador local, permitiendo así seguir con el desarrollo de la herramienta.

4.3.3 Sprint Review

Para este sprint el objetivo es buscar los editores open source existentes en el mercado que permitan edición de su código fuente y encontrar a uno que pudiera ser utilizado como base para el desarrollo de la herramienta que se pretende desarrollar.

Para tal efecto, se han encontrado cuatro editores open source. De estos cuatro editores se ha decidido utilizar sólo el bpmn.io.

A continuación, es necesario instalarlo, comprobarlo, y preparar el ambiente de trabajo para el desarrollo subsecuente.

La instalación se ha ejecutado con éxito y las pruebas han sido realizadas según se esperaba.

Todos los objetivos del sprint han sido ejecutados con éxito.

En el siguiente sprint se iniciará el desarrollo de la herramienta como tal.

4.3 Sprint 1: Notación PESOA

4.3.1 Sprint Planning

En este sprint, se iniciará el desarrollo de la herramienta.

Antes de iniciar el desarrollo, se ha utilizado un ejemplo del bpmn.io para entender cómo se podría implementar la aplicación. El ejemplo elegido ha sido el bpmn-js-example-custom-rendering [4]. Este ejemplo pone colores a ciertas formas, añade formas customizadas y exporta una extensión del BPMN 2.0 que pudiera leer estas modificaciones.

De manera que, lo primero que se realiza es crear una sintaxis para la exportación de las modificaciones, es decir, una extensión al formato BPMN 2.0 que pueda especificar las características de variabilidad de procesos.

4.3.2 Extensión del BPMN 2.0

Bpmn.io permite extender los formatos de BPMN 2.0 utilizando un fichero en JSON, en el que se define las propiedades como nombre de la extensión, prefijo de la extensión y el comportamiento de la extensión [5].

Para extender el modelo, se ha utilizado un formato de anotación dentro de los nodos, en los que permite identificar si este nodo es un nodo BPMN 2.0 normal o un nodo de variabilidad de procesos y a cuál característica se refiere. En seguida se explicará mejor como se ha decidido implementar la variabilidad de procesos y qué significa exactamente cada caso, pero por ahora, se puede ilustrar que, un nodo de task en BPMN 2.0 ordinario sería algo semejante a esto:

```
<bpmn2:task id="Activity_1p4xg34" name="Variation Point">
```

Ilustración 24 Nodo ordinario de Task en BPMN 2.0. Fuente: Elaboración propia.

A su vez, un nodo extendido a la notación propuesta sería algo como este:

```
<bpmn2:task id="Activity_1p4xg34" name="Variation Point" pv
        :PVNodeType="&#60;&#60;VarPoint&#62;&#62;">
```

Ilustración 25 Nodo extendido a la notación propuesta. Fuente: Elaboración propia.

Como se puede observar, existe el atributo *pv:PVNodeType*, que define qué significa este nodo, en el caso del ejemplo arriba, significa que este nodo task ha sido extendido a un *VarPoint*.

Para lograr esto, se ha utilizado el JSON. Este JSON primeramente define el nombre de la extensión, que ha sido llamado de ProcessVaribility y tendrá un prefijo pv. En seguida se ha definido que esta extensión tendrá un tipo llamado ProcessVariabilityNodes, que aplicará las extensiones a bpmn:FlowNode (hecho para poder extender las características de la notación PESOA y PROVOP), que representa actividades en la notación BPMN (como actividades se entienden taks, subprocess y call activity) y bpmn:SequenceFlow (hecho para poder extender una característica del PESOA) que representa los flujos entre actividades (como flujos de secuencia y flujos de mensajería). La Ilustración 26, muestra el JSON creado para extender la notación BPMN.

```
{
  "name": "ProcessVariability",
  "uri": "http://some-company/schema/bpmn/pv",
  "prefix": "pv",
  "xml": {
    "tagAlias": "lowerCase"
  },
  "types": [
    {
      "name": "ProcessVariabilityNodes",
      "extends": [
        "bpmn:FlowNode",
        "bpmn:SequenceFlow"
      ],
      "properties": [
        {
          "name": "PVNodeType",
          "isAttr": true,
          "type": "String"
        }
      ]
    }

  ]
}
```

Ilustración 26 JSON que implementa la extensión del BPMN. Fuente: Elaboración propia.

4.3.3 Ficheros con las personalizaciones

Una vez concluido el punto anterior, es necesario entender cómo ha sido dividida la aplicación para poder implementar características personalizadas a la herramienta.

Los proyectos personalizados en bpmn.io suelen tener por lo menos 5 ficheros en los que se definen las personalizaciones a ser realiadas: CustomContextPad.js, CustomPallete.js, CustomRenderer.js, CustomRules.js, CustomUpdater.js y el index.js

de la carpeta custom, en la que se concatena e inicializa todas las reglas personalizadas al proyecto. En seguida se definirá la función de cada uno de estos ficheros:

4.3.3.1 CustomContextPad.js

El “Pad” es la “caja de herramientas” que se encuentra al lado derecho del editor original, en el que se encuentran las opciones para el editor, en el que se muestran los elementos BPMN para añadir al diagrama (como ejemplo los eventos, los gateways, las tareas, etc) y las herramientas de manejo del diagrama (como ejemplo la mano para mover el diagrama). La Ilustración 27, muestra el pad original del bpmn.io.

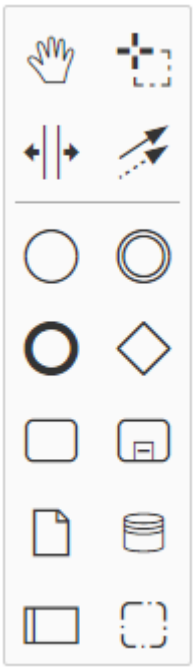

Ilustración 27- Pad original de bpmn.io. Fuente: Elaboración propia.

4.3.3.2 CustomPallete.js

El “Pallete” son las herramientas que aparecen como acceso directo al hacer clic en algún elemento del diagrama, en el que añadirá el elemento que se ha hecho clic en secuencia en el diagrama. Por ejemplo, al hacer clic en un evento de inicio de diagrama por defecto, abrirá el pallete para añadir evento de fin, gateway, tasks, evento intermediario, anotaciones, tipos, borrar el elemento y flujos. En la ilustración 28 se puede observar el pallete por defecto.

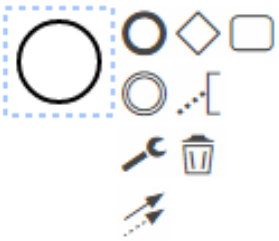

Ilustración 28 - Pallete por defecto de evento de inicio. Fuente: Elaboración propia.

4.3.3.3 CustomRenderer.js

Es el fichero en el que se define cómo se mostrarán las formas personalizadas. Por ejemplo, aquí se definió como se escribiría las anotaciones de PESOA en las actividades personalizadas.

4.3.3.4 CustomRules.js

En este fichero se define la lógica de las reglas personalizadas, por ejemplo, se ha definido que cuando son dos objetos PESOA, el sequence flow que conecta los dos posee un formato distinto.

4.3.3.5 CustomUpdater.js

En este fichero se unen todas las partes para diseñar los diagramas personalizados y permite que siempre se lea y se escriba los ficheros de las mismas formas. Funciona como si fuera un auditor del diagrama, siempre se actualizará de la misma forma.

4.3.4 Implementación

Teniendo los conceptos introducidos, se deduce que, para implementar esta notación habría que extender las tareas y subprocesos del BPMN para recibir la notación de los estereotipos presentados por PESOA.

De esta manera, lo que se deber realizar es crear tareas y subprocesos personalizados con las notaciones. Para las tareas, las notaciones recibidas serían: *<<VarPoint>>, <<Abstract>>, <<Null>>* y *<<Optional>>.* Las tareas tienen como significado en esta notación algo que será todavía especificado o una abstracción de un nivel más alto.

Para los subprocesos, las notaciones recibidas serian *<<Defect>>, <<Variant>>, <<VarPoint>>, <<Abstract>>, <<Null>> y <<Optional>>*. Los subprocesos en este contexto se entienden por la especificación de una tarea o a una tarea que todavía también será especificada, pero con un proceso interno.

4.3.4.1 Custom palette

Como anteriormente indicado, el palette es la “caja de herramientas” del editor. Es donde se añade los símbolos que serán utilizados en el editor.

Primeramente, se añade aquí los símbolos para que después sean implementadas.

Visualmente hablando, se han creado las imágenes que estarían en el palette. Estas imágenes son del formato svg, debido a esto, se ha utilizado el editor de imágenes web Photopea (https://www.photopea.com/).

Photopea es un editor de imágenes online gratuito que permite crear imágenes en los formatos JPG, PNG o SVG, además de permitir que se exporte el proyecto en el formato PSD. La Ilustración 29 muestra la interfaz de este editor. Como lo que se necesita no es algo demasiado complejo, la solución que este editor gratuito proporciona es suficiente.

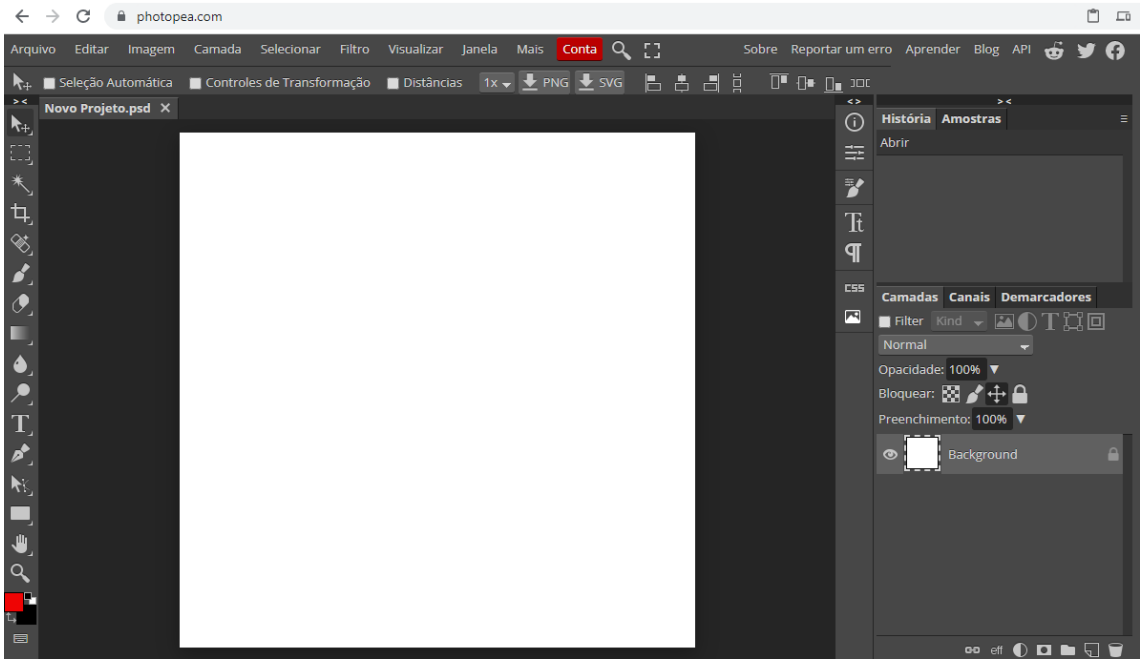

Ilustración 29 Editor Photopea

En este editor han sido creadas las Ilustraciones de la 30 a 41 que se mostrarán en el palette.

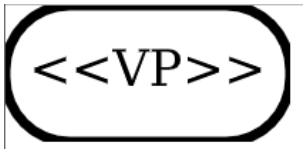

Ilustración 30 - Icono Tarea VarPoint

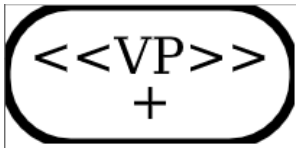

Ilustración 31 - Icono Subproceso VarPoint

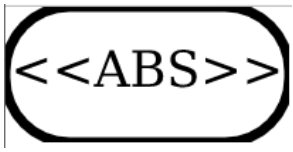

Ilustración 32 - Icono Tarea Abstract

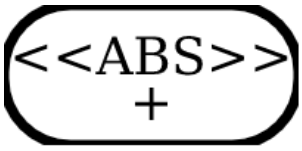

Ilustración 33 - Icono Subproceso Abstract

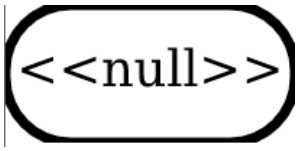

Ilustración 34 - Icono tarea NULL

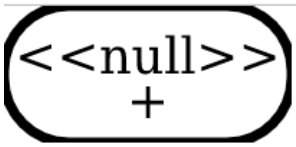

Ilustración 35 - Icono subproceso NULL

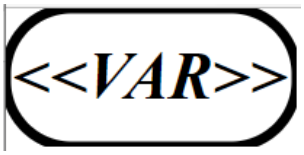

Ilustración 36 - Icono tarea Variant

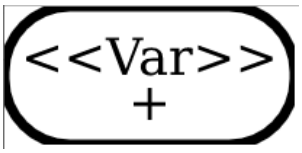

Ilustración 37 - Icono subproceso Variant

Ilustración 38 - Icono tarea Default

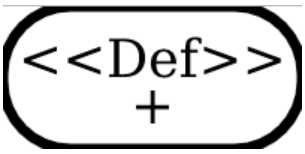

Ilustración 39 - Icono subproceso Default

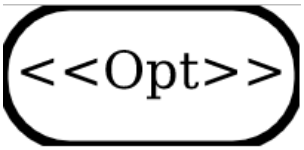

Ilustración 40- Icono tarea Optional

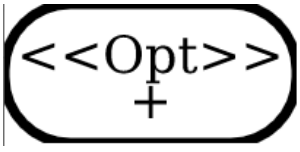

Ilustración 41 - Icono Subproceso Optional

Teniendo las imágenes hechas, la forma que bpmn.io trabaja para asignar imágenes al palette es asignándolas a una clase CSS. De esta forma, han sido creadas las clases icono-varpoint (relacionado al icono de la tarea extendida VarPoint), icono-abstract (relacionado al icono de la tarea extendida Abstract), icono-default (relacionado al icono de la tarea extendida Default), icono-null (relacionado al icono de la tarea extendida Null), icono-optional (relacionado al icono de la tarea extendida Optional), icono-variant (relacionado al icono de la tarea extendida Variant), icono-varplus (relacionado al icono del subproceso extendido Variant), icono-defplus (relacionado al icono del subproceso extendido Defect), icono-vpplus (relacionado al icono del subproceso extendido VarPoint), icono-absplus (relacionado al icono del subproceso extendido Abstract), icono-nullplus (relacionado al icono del subproceso extendido NULL) e icono-optplus (relacionado al icono del subproceso extendido Optional). Las clases de CSS tienen la estructura como se muestra en la Ilustración 42, cambiando apenas la URL de la imagen de background:

```
182 .icono-optplus{
183     background: url("imgs/icono_optplus.svg") ;
184     background-size: contain;
185     background-repeat: no-repeat;
186     background-position: center;
187     margin: 0 2px 0 2px;
188     width: 46px !important;
189 }
```

Ilustración 42 Código CSS de las imagenes que irán al palette. Fuente: Elaboración propia.

Teniendo las imágenes, se las debe añadir en el palette y asignarles un método para renderizar el componente en el editor, una vez seleccionado.

En el fichero del CustomPalette.js hay una clase llamada CustomPalette. En esta clase se define qué es lo que va a estar en el palette y cuál es el comportamiento que cada ítem del palette tendrá cuando se lo selecciona.

Para definir qué es lo que va a estar en el palette, se define que retorne una lista de ítems para esta clase, y en cada ítem se define en qué grupo se pondrá dicho campo, cuál es el título textual que va tener este campo y cuáles acciones este campo tendrá.

Los ítems de esta lista tienen una estructura semejante a la de la Ilustración 43.

```
100         'create.varpoint': {
101             group: 'activity',
102             className: 'icono-varpoint',
103             title: translate('Create VarPoint'),
104             action: {
105                 dragstart: createTask("<<VarPoint>>"),
106                 click: createTask("<<VarPoint>>")
107             }
108         },
```

Ilustración 43 Estructura de código que pone la imagen al palette. Fuente: Elaboración propia.

En este ejemplo especifico, se define que el identificador de lo que se va añadir a esta lista es “create.varpoint”, que tiene que estar en el grupo de los “activity” (que es el grupo que se encuentran las tareas y subprocesos), el titulo será “Create VarPoint” y las acciones que tendrá son, que cuando se hace clic se llamará el método createTask(“<<VarPoint>>”) y cuando se arrastre también llamará al mismo método.

Después de añadir todos los ítems al palette, el resultado en pantalla será el siguiente:

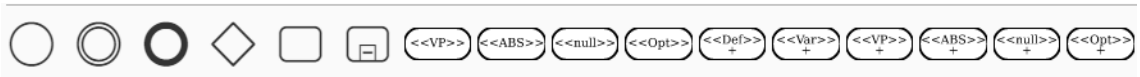

Ilustración 44 - Items añadidos al palette. Fuente: Elaboración propia.

Para renderizar en pantalla las tareas o subprocesos personalizados han sido creados dos métodos: createTask(string) (explicitado en la Ilustración 45) y createSubprocess(string). Los dos tienen un funcionamiento semejante, cambiando apenas a qué tipo de objeto se referencia: bpmn:Task o bpmn:Subprocess.

```
27  function createTask(suitabilityScore) {
28    return function(event) {
29      const businessObject = bpmnFactory.create('bpmn:Task');
30      businessObject.PVNodeType = suitabilityScore;
31
32      const shape = elementFactory.createShape({
33        type: 'bpmn:Task',
34        businessObject: businessObject
35      });
36
37      create.start(event, shape);
38    };
39  }
```

Ilustración 45 Método createTask(string). Fuente: Elaboración propia.

Este método retorna la función de creación que es definido en el CustomRenderer, explicado en el apartado siguiente.

Sin embargo, primeramente se crea un nuevo objeto de bpmn:Task, y asigna al atributo PVNodeType (que es el atributo que ha sido definido que tendrá los datos relacionados a la variabilidad de procesos) el valor de name, que es el texto del estereotipo que estará en este objeto. En seguida es creada la forma como tal que será renderizada y se llama el método que creará dicha forma.

4.3.4.2 Custom Renderer

Como explicado anteriormente, el fichero CustomRenderer.js tiene los métodos para renderizar las formas personalizadas.

Dentro de este fichero está la clase CustomRenderer que extiende la clase del bpmn.io BaseRenderer. En la práctica, todos los objetos, una vez que se extiende el BaseRenderer, entrarán en este fichero para ver si son personalizados o no, si no, no hace nada, si sí, ejecutan los métodos personalizados para su impresión en pantalla.

Los dos principales métodos del Custom Renderer son el drawShape(parentNode, element) y el drawCustomConnection(po, element).

El método drawShape diseña los objetos en el editor. Para la notación del PESOA, el fragmento de código es el mostrado en la Ilustración 46.

```
97      else{
98        var text = svgCreate('text');
99
100       svgAttr(text, {
101         fill: '#000',
102         transform: 'translate(23, 15)',
103         fontSize: '0.65em',
104         fontStyle: 'italic',
105         fontFamily: 'Arial, Helvetica, sans-serif'
106       });
107       svgAppend(text, document.createTextNode(elementText));
108       svgAppend(parentNode, text);
109     }
```

Ilustración 46 Fragmento de código que diseña la notación PESOA. Fuente: Elaboración propia.

En resumen, crea un svg de texto con el estereotipo (ejemplo <<*VarPoint*>>), asigna los atributos de CSS, en el que se posiciona ese texto en la posición superior izquierda de la tarea o subproceso y lo imprime en pantalla en la posición decidida.

El método drawCustomConnection(po,element) lo que hace es que, cuando el objeto tiene el atributo PVNodeType = “custom” crea la línea descontinua que significa la implementación de la tarea.

4.3.4.3 Custom Rules

En el fichero CustomRules.js, para la notación PESOA, hay el método que define cuando la línea que conecta dos elementos debe tener valor en el atributo PVNodeType, lo que significa que es una línea personalizada. La función de este método es:

1. Verificar si los objetos de entrada y salida son objetos con valor en el atributo PVNodeType
2. Si por lo menos uno no es, retorna falso y hace una conexión normal
3. Si los dos son, asigna a la línea valor en el atributo PVNodeType y retorna verdadero.

4.3.5 Resultados de la implementación

Al hacer clic en un elemento personalizado y arrastrarlo al espacio del editor, el elemento tendrá una apariencia “fantasma” hasta que uno hace clic donde quiere ponerlo:

Ilustración 47 - Objeto antes de posicionarlo en el editor. Fuente: Elaboración propia.

Una vez posicionado, se abre el texto del elemento para introducir su texto indicativo.

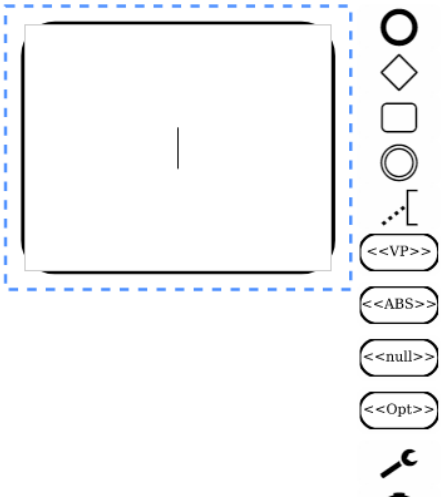

Ilustración 48 - Objeto recién posicionado. Fuente: Elaboración propia.

Al hacer clic fuera del objeto, este sería el resultado final:

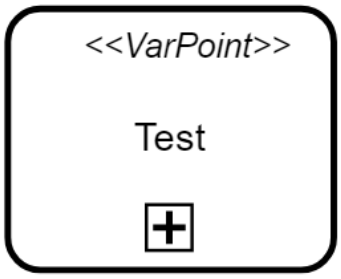

Ilustración 49 – Objeto. Fuente: Elaboración propia.

El objeto usado como ejemplo anteriormente es un subproceso, también es posible expandirlo para poner dentro otro proceso, haciendo clic en y seleccionando

Sub Process (expanded) . Así el resultado sería el siguiente mostrado en la ilustración 50.

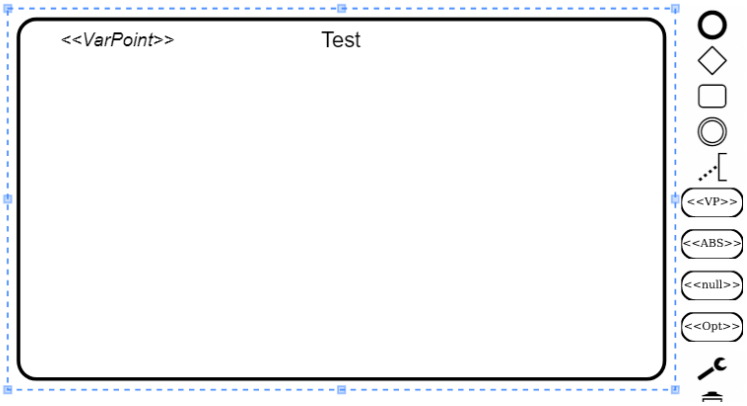

Ilustración 50 - Subproceso expandido. Elaboración propia.

Como se puedo observar, el estereotipo se ha movido al extremo superior izquierdo.

Adicionalmente, al conectar dos elementos personalizados se encuentra una línea descontinua que muestra la implementación del punto de variabilidad.

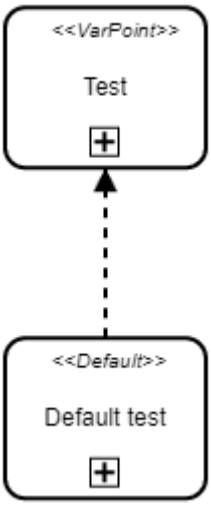

Ilustración 51 – Implementación. Elaboración propia.

Un diagrama completo hecho en el editor creado para la notación PESOA puede ser el siguiente:

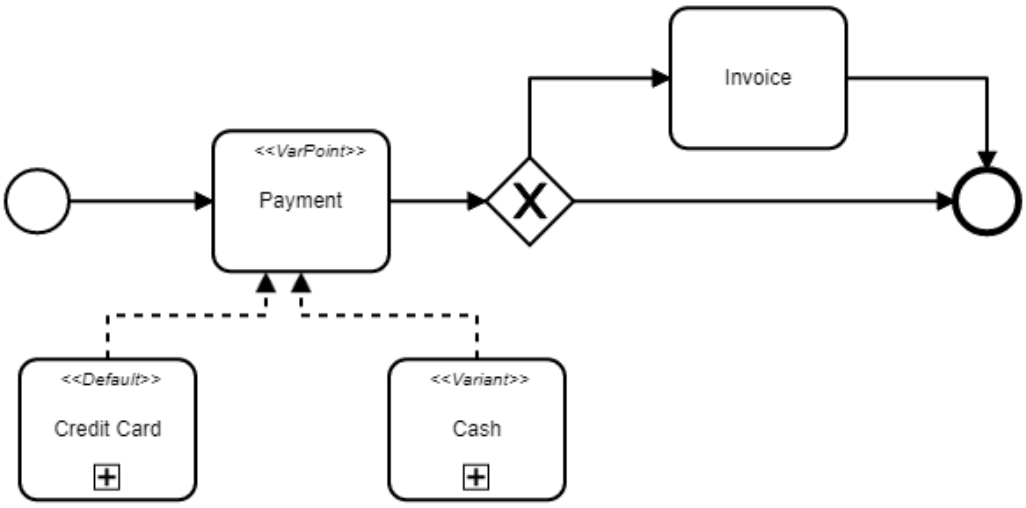

Ilustración 52 - Ejemplo de proceso modelado en el editor creado. Elaboración propia.

Este es un proceso simple de pago, en el que el "Payment" es un punto de variabilidad que puede ser implementado por defecto por el subproceso "Credit card" pero también tiene la variabilidad "Cash", que, a su vez si hay éxito entra en la tarea "Invoice", sino, termina el proceso.

Este modelo puede ejemplificar cómo en un mismo diagrama modelado se puede obtener objetos con y sin los estereotipos inicialmente creados, respetando las reglas que han sido elaboradas.

Adicionalmente, se puede exportar el diagrama en formato XML.

```
<bpmn2:task id="Activity_0jicqjw" name="Payment" pv:PVNodeType="&#60;&#60;VarPoint&#62;&#62;">
  <bpmn2:incoming>Flow_0jzora4</bpmn2:incoming>
  <bpmn2:incoming>Flow_0qlvcbc</bpmn2:incoming>
  <bpmn2:incoming>Flow_05rlm1d</bpmn2:incoming>
  <bpmn2:outgoing>Flow_12k8vux</bpmn2:outgoing>
</bpmn2:task>
<bpmn2:sequenceFlow id="Flow_0jzora4" sourceRef="Event_0546ni0" targetRef="Activity_0jicqjw" />
<bpmn2:endEvent id="Event_1djwkpc">
  <bpmn2:incoming>Flow_1okzv49</bpmn2:incoming>
  <bpmn2:incoming>Flow_0rd5576</bpmn2:incoming>
</bpmn2:endEvent>
<bpmn2:subProcess id="Activity_1p9h6ke" name="Credit card" pv:PVNodeType="&#60;&#60;Default&#62;&#62;">
  <bpmn2:outgoing>Flow_0qlvcbc</bpmn2:outgoing>
</bpmn2:subProcess>
<bpmn2:sequenceFlow id="Flow_0qlvcbc" sourceRef="Activity_1p9h6ke" targetRef="Activity_0jicqjw" pv:PVNodeType="custom" />
<bpmn2:subProcess id="Activity_0zrbnx1" name="Cash" pv:PVNodeType="&#60;&#60;Variant&#62;&#62;">
  <bpmn2:outgoing>Flow_05rlm1d</bpmn2:outgoing>
</bpmn2:subProcess>
<bpmn2:sequenceFlow id="Flow_05rlm1d" sourceRef="Activity_0zrbnx1" targetRef="Activity_0jicqjw" pv:PVNodeType="custom" />
<bpmn2:exclusiveGateway id="Gateway_0q0lohn">
  <bpmn2:incoming>Flow_12k8vux</bpmn2:incoming>
  <bpmn2:outgoing>Flow_0mv8xpe</bpmn2:outgoing>
  <bpmn2:outgoing>Flow_0rd5576</bpmn2:outgoing>
</bpmn2:exclusiveGateway>
<bpmn2:sequenceFlow id="Flow_12k8vux" sourceRef="Activity_0jicqjw" targetRef="Gateway_0q0lohn" />
```

Ilustración 53 - Parte del diagrama exportado. Elaboración propia.

Como se puede observar, en el diagrama exportado se tiene los estereotipos marcados, además de las líneas de secuencia que tienen el atributo PVNodeType como custom, que es lo que significa que son líneas descontinuas.

4.3.7 Sprint Review

En este sprint el objetivo era crear en el editor elegido la notación PESOA.

La notación PESOA es formada por estereotipos que se añaden a las tareas y subprocesos.

En este sprint fue posible crear los diseños y reglas para la notación PESOA.

En el siguiente sprint sigue el desarrollo para la notación PROVOP

4.4 Sprint 2: Notación PROVOP

4.4.1 Sprint Planning

En este sprint el objetivo es desarrollar la notación PROVOP dentro de la herramienta.

4.4.2 Introducción

La notación PROVOP permite definir variabilidad de procesos de una forma bien sencilla, utilizando apenas un símbolo. De manera que, antes de mostrar el desarrollo por detrás de la notación de PROVOP, es importante introducir algunos conceptos.

4.4.3 Implementación

4.4.3.1 Custom Palette

Como ya explicado, el palette es donde se encuentran los atributos de las notaciones. Así que, el primer punto a realizar, es añadir la notación PROVOP a nuestro pallete. Usando la misma herramienta de diseño generada para crear los diseños de la notación PESOA, se ha generado el “diamante negro” en png para añadirlo al palette:

Ilustración 54 Notación de punto de ajuste. Elaboración propia.

Sin embargo, para PROVOP ha sido utilizado otro abordaje para añadir la imagen en el palette. Se ha utilizado este abordaje para poder reaprovechar la imagen en el propio diagrama, como una forma personalizada, es decir, una forma que no se encuentra en la notación BPMN de origen. Como la notación PESOA está basada en añadir estereotipos, es decir, “anotaciones” dentro de atributos ya existentes en BPMN, este abordaje no es necesario, teniendo apenas que añadir los estereotipos dentro de las actividades o subprocesos nativos de BPMN.

De manera que, lo se ha creado la imagen como si fuera un módulo propio, teniendo su index.js y su imagen en png.

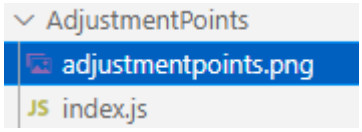

Ilustración 55 Estructura dentro del proyecto del punto de ajuste. Fuente: Elaboracion propia.

Dentro del index.js, hay una instrucción de exportación de la imagen en formato base64. Esta forma de trabajar es la forma indicada en los tutoriales del bpmn.io para añadir formas no existentes en el BPMN nativo [2].

```
module.exports.dataURL = '
```

Ilustración 56 Fragmento del código de la imagen convertida a string base64 de la imagen de puntos de ajuste. Fuente: Elaboración propia.

De esta forma, para poder utilizar este módulo en el CustomPallete.js, primeramente hay que importarlo, como se muestra en la Ilustración 57.

```
1  import AP from '../adjustmentpoints'
```

Ilustración 57 Importación del módulo de puntos de ajuste. Fuente: Elaboración propia.

En seguida, para poner esta imagen al pallete y ser utilizada como botón, ha sido añadido el fragmento de código que se encuentra en la Ilustración 58, junto a los demás códigos de PESOA:

```
189  'create.adjustmentpoints': {
190    group: 'activity',
191    title: 'Create Adjustment Points',
192    imageUrl: AP.dataURL,
193    action: {
194      dragstart: startCreateAP("AP"),
195      click: startCreateAP("AP")
196    }
197  },
```

Ilustración 58 Fragmento de código que añade la imagen de puntos de ajuste al palette. Fuente: Elaboración propia.

En este código lo que se realiza es añadir esta forma al pallete junto del grupo de actividades, que la imagen sea el dataURL del módulo AP, es decir, del módulo AdjustmentPoints que ha sido importado con el nombre de AP y se define las acciones como distintas a las de PESOA, llamando el método de “starteCreateAP”.

En el método, startCreateAP tiene la misma función que createTask y la createSubprocess utilizadas para el método PESOA, pero, para PROVOP, lo que realiza es crear un “gateway" como base. La notación de punto de ajuste tendrá la misma funcionalidad de un puerto en el BPMN, por eso se utiliza como base.

Adicionalmente, para el atributo PVNodeType, que es el que marca la extensión del BPMN que usamos, se asigna el valor “AP”, sigla de “Adjustment Points” (puntos de ajuste en inglés). Por último, se llama el método de creación de las formas create.start()

que llamará el método de creación que se encuentra en el fichero CustomRenderer.js, que será explicado en el siguiente apartado.

De esta forma, añadiendo la forma del punto de ajuste en el palette, tenemos el siguiente resultado:

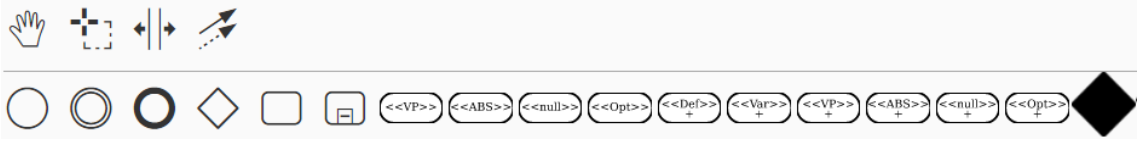

Ilustración 59 Custom Palette con el punto de ajuste. Fuente: Elaboración propia.

4.4.3.2 Custom Renderer

Como ya explicado, el fichero CustomRenderer.js es el encargado en diseñar la forma personalizada en la pantalla del editor.

Para que se pueda imprimir en pantalla el “diamante negro” de la notación de punto de ajuste, es necesario añadir una condición al método drawShape, introducido en el capítulo anterior, presentado en la Ilustración 60.

```
59      if(elementText == "AP"){
60          var url = AdjustmentPoint.dataURL;
61
62          var apGfx = svgCreate('image', {
63            x: 0,
64            y: 0,
65            width: 50,
66            height: 50,
67            href: url
68          });
69
70          svgAppend(parentNode, apGfx);
71      }
```

Ilustración 60 Fragmento de código que modifica drawShape para imprimir en pantalla en diamante negro. Fuente: Elaboración propia.

En resumen, primeramente se toma la imagen base64 del módulo AdjustmentPoint, que también tuvo que ser importado en este fichero, y se lo ha asignado a la variable url; en seguida, se crea una imagen svg en el que el “href” (atributo en el que se enlaza el valor de la url de una imagen en el lenguaje de marcación HTML) tiene el valor del dataURL del módulo AdjustmentPoint, y asigna los valores de posición x e y en el bloque svg, y la altura y ancho de la imagen.

4.4.3.3 Resultados de la implementación

Al hacer clic en la imagen del palette, obtenermos la siguiente figura para posicionarla en el editor:

Ilustración 61 Punto de ajuste sombreado de azul antes de ser posicionado en el editor. Fuente: Elaboración propia.

Y al hacer clic en el editor para posicionarlo, el punto de ajuste se muestra de la siguiente forma:

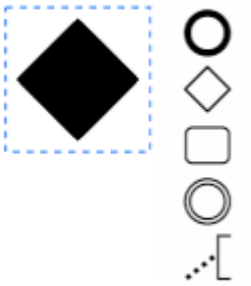

Ilustración 62 Punto de ajuste posicionado. Fuente: Elaboración propia.

De esta forma, el diagrama de ejemplo dado por [6] que está en la figura 3 pudo ser modelado de la siguiente manera en el editor:

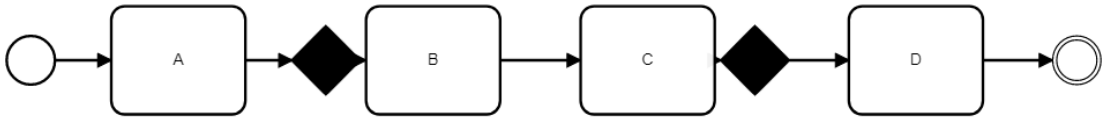

Ilustración 63 Punto de ajuste posicionado. Fuente: Elaboración propia.

Adicionalmente, por no haber ninguna restricción, también es posible hacer diagramas mesclando las dos notaciones hechas hasta el momento, es decir, hacer un diagrama que tenga las mismas características de PESOA y de PROVOP, como el ejemplo abajo:

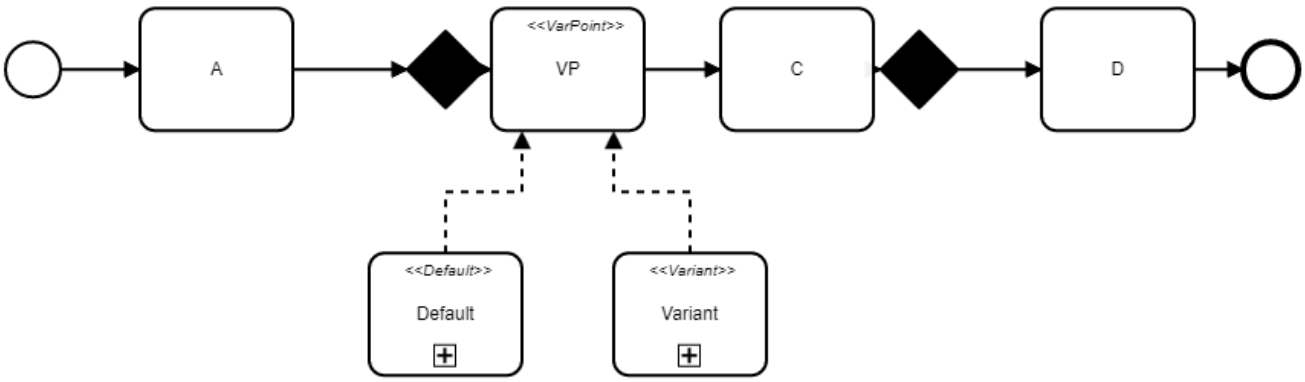

Ilustración 64 Diagrama con notaciones PESOA y PROVOP. Fuente: Elaboración propia.

Continuando, en el fichero exportado con el diagrama, se puede observar que los puertos que son los “diamante negro” tienen el siguiente formato:

```
<bpmn2:gateway id="Gateway_0fhkqxu" pv:PVNodeType="AP">
```

Es decir, tiene el atributo pv:PVNodeType con el valor “AP”, mostrando la marcación de que este Gateway es un punto de ajuste.

4.4.4 Sprint Review

En este sprint el objetivo es implementar la notación PROVOP en el editor.

La notación PROVOP consta de una sintaxis simple formada apenas por un atributo: el “diamante negro”.

Dicha notación ha sido implementada con éxito, permitiendo adjuntarla a la notación PESOA dentro del editor.

Para el siguiente sprint se deberá crear la notación de dependencias verticales y horizontales.

4.5 Sprint 3: Notación de dependencias verticales y horizontales

4.5.1 Sprint Planning

En este sprint el propósito es desarrollar la notación de dependencias verticales y horizontales en la herramienta.

4.5.2 Implementación

4.5.2.1 Custom Palette

Lo primero es crear las imágenes para poner en el palette. Usando la misma herramienta que ha sido utilizada para la creación de las imágenes tanto para PESOA como para PROVOP, se ha creado las imágenes para el hexágono de doble borde y para el hexágono de borde grueso:

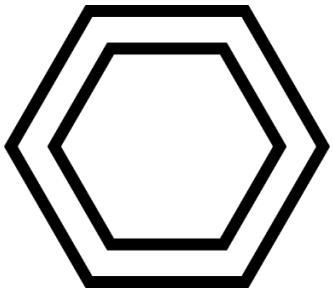

Ilustración 65 - Notación del Hexágono de Doble Borde

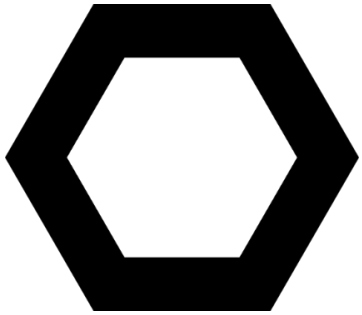

Ilustración 66 - Notación del Hexágono de Borde Grueso

Para añadir las imágenes al palette, se ha utilizado el mismo abordaje que el que ha sido usado para la notación PROVOP. Como se tiene que añadir una nueva forma al personalizada que no existe en la notación BPMN al diagrama, se ha creado un módulo para el hexágono de borde grueso y otro modulo para el hexágono de doble borde, usando la misma estrategia para el punto de ajuste.

De manera que para el hexágono de borde grueso ha sido creado el módulo llamado VerticalSincronizationFull, en el que hay el imagen png de este hexágono y el fichero index.js en el que tiene la misma estructura del diamante negro, es decir, dentro se crea el atributo dataURL con la representación de la imagen en base64.

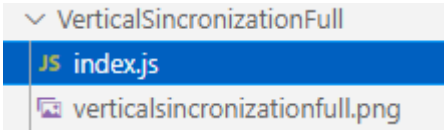

Ilustración 67 - Módulo del hexágono de borde grueso. Fuente: Elaboración propia.

```
module.exports.dataURL = '
iVBORw0KGgoAAAANSUhEUgAAAiQAAAHbCAYAAAD734EhAAAg
```

Ilustración 68 – Parte del código de la imagen del hexágono de borde grueso en base64. Fuente Elaboración propia.

Para el hexágono de doble borde ha sido creado el módulo llamado VerticalSincronizationEmpty, con la misma estructura del VerticalSincronizationFull, es decir, la imagen png y el fichero index.js en el que dentro se crea el atributo dataURL con la imagen en formato base64.

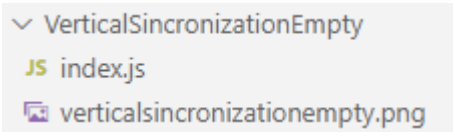

Ilustración 69 - Módulo del hexágono de doble borde. Fuente: Elaboración propia.

```
module.exports.dataURL =  '
iVBORw0KGgoAAAANSUhEUgAAAiQAAAHbCAYAAAD734EhAAAgAEl
```

Ilustración 70 - Parte del código del index.js del VerticalSincronizationEmpty. Fuente: Elaboración propia.

Teniendo estas dos imágenes, el procedimiento de añadir las imágenes al palette es el mismo que el hecho anteriormente para el PROVOP. Primeramente, se ha añadido en el fichero CustomPalette.js las referencias a los módulos creados anteriormente, en el que se tiene las imágenes, ilustración 71.

```
2   import VSF from '../VerticalSincronizationFull'
3   import VSE from '../VerticalSincronizationEmpty'
```

Ilustración 71 Referencias a las imágenes de PROVOP. Fuente: Elaboración propia.

En seguida, debajo del código en el .que se ha metido el icono del diamante negro se adiciona los códigos para poner los iconos del hexágono de borde grueso y del hexágono de doble borde, como se muestra en la Ilustración 72.

```
198     'create-verticalsincronizationfull': {
199       group: 'activity',
200       title: 'Create Vertical Sincronization',
201       imageUrl: VSF.dataURL,
202       action: {
203         dragstart: startCreateVSF("VSF"),
204         click: startCreateVSF("VSF")
205       }
206     },
```

Ilustración 72 Fragmento de código para añadir el hexágono de borde grueso al palette. Fuente: Elaboración propia.

El código del hexágono de doble borde es idéntico, cambiando apenas las siglas “VSF” para “VSE”. Lo que hace ese fragmento de código es añadir la imagen que se encuentra en el VSF.dataURL en el grupo de actividades, y asignar el método startCreateVSF para los eventos de arrastrar y clicar.

El método startCreateVSF (que es idéntico al startCreateVSE para el hexágono de doble borde) hace lo mismo que el método startCreateAP del PROVOP. Crea un gateway como figura base para los hexágonos, y adiciona en la variable PVNodeType el valor “VSF” para el hexágono de borde grueso y el valor “VSE” para el hexágono de doble borde. En seguida, llama al método create.start, en el que irá al método que se encuentra en el fichero CustomRenderer.js para generar las imágenes, que será explicado en el siguiente apartado.

De manera que, después de añadir las imágenes para las notaciones de dependencias verticales hexágono de borde grueso y hexágono de doble borde, el palette ha adquirido la siguiente forma, una vez que ya tiene todas las formas de las notaciones que serán utilizadas:

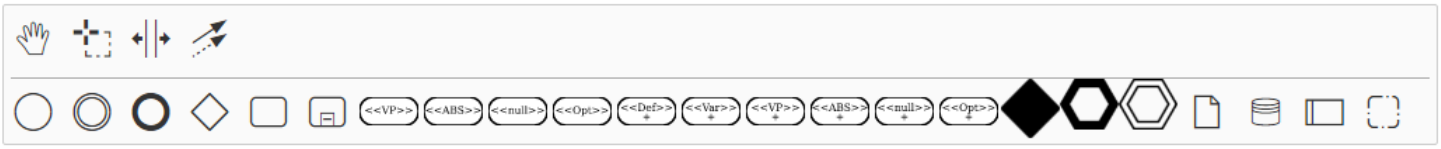

Ilustración 73 - Palette con todos los símbolos. Fuente: Elaboración propia.

4.5.2.2 Custom Renderer

Para que se imprima en pantalla, se ha tenido que modificar, en el fichero CustomRenderer.js, en el método drawShape, para reconocer cuando es uno de los hexágonos de la notación de dependencias verticales.

Primeramente, es necesario añadir la referencia a los módulos del VerticalSincronizationFull y VerticalSincronizationEmpty en el fichero. Teniendo esto, se ha añadido el fragmento de código mostrado en la Ilustración 74 al método drawShape.

```
72    else if(elementText == 'VSF'){
73      var url = VerticalSincronizationFull.dataURL;
74        parentNode.innerHTML = "";
75        var apGfx = svgCreate('image', {
76          x: 0,
77          y: 0,
78          width: 50,
79          height: 50,
80          href: url
81        });
82
83        svgAppend(parentNode, apGfx);
```

Ilustración 74 Fragmento que renderiza el hexágono de borde grueso. Fuente: Elaboración propia.

Tanto para el hexágono de doble borde (VSE) como para el hexágono de borde grueso (VSF) el abordaje es el mismo. Lo primero a realizar es añadir a la variable url el valor del atributo dataURL, que contiene la representación en base64 de la imagen de la notación. En seguida, se limpia el nodo padre. Eso hará con que se quite la imagen original del gateway. Después, se crea una imagen svg con el atributo href teniendo la representación en base64 de la imagen, y asignando los valores de altura y ancho de la imagen a 50px.

4.5.3 Resultados de la implementación

Al hacer clic en los iconos de los dos hexágonos en el palette, se muestra las siguientes imágenes para posicionarlas en el editor:

Ilustración 75 - Hexágono de borde grueso antes de ser posicionado. Fuente: Elaboración propia.

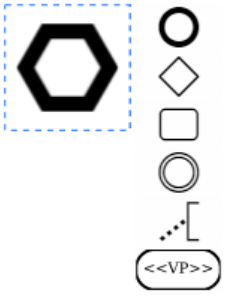

Ilustración 76 - Hexágono de borde grueso una vez posicionado. Fuente: Elaboración propia.

Ilustración 77 - Hexágono de doble borde antes de ser posicionado. Fuente: Elaboración propia.

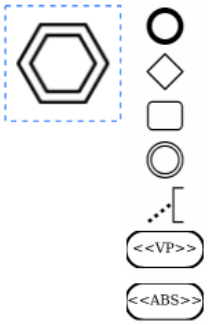

Ilustración 78 - Hexágono de doble borde una vez posicionado. Fuente: Elaboración propia.

De esta forma, los ejemplos de las ilustraciones 13 y 14 pueden ser modelados en el editor, probando que es posible modelar utilizando las notaciones PESOA, PROVOP y de dependencia vertical en el mismo diagrama.

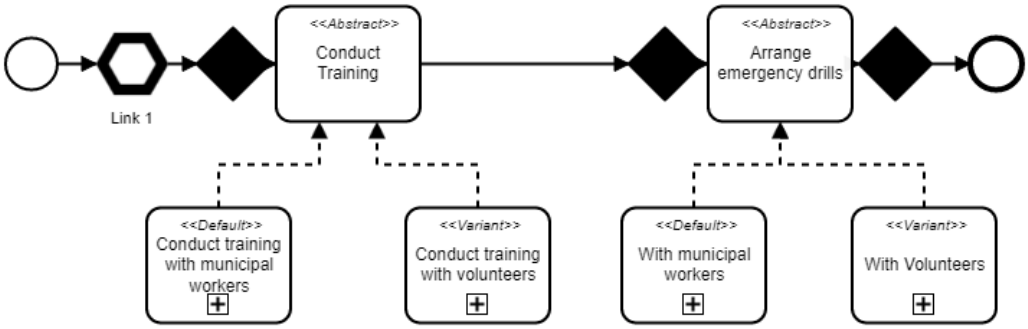

Ilustración 79 - Diagrama equivalente al de la ilustración 13 hecho en el editor. Fuente: Elaboración propia.

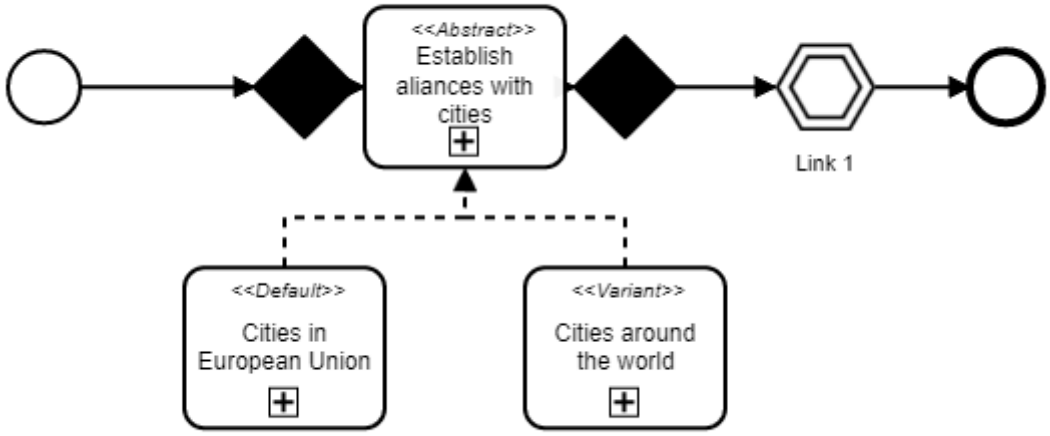

Ilustración 80 - Diagrama equivalente al de la ilustración 14 hecho en el editor. Fuente: Elaboración propia.

Además, al exportar el diagrama BPMN en XML, se puede ver los atributos PVNodeType con los valores “VSE” para el hexágono de doble borde y “VSF” para el hexágono de borde gruesa:

```
<bpmn2:gateway id="Gateway_0lsd89k" name="Link 1" pv:PVNodeType="VSE">
  <bpmn2:incoming>Flow_02d9d6i</bpmn2:incoming>
  <bpmn2:outgoing>Flow_01pfx7g</bpmn2:outgoing>
</bpmn2:gateway>
```

Ilustración 81 - XML exportado con el valor "VSE". Fuente: Elaboración propia.

```
<bpmn2:gateway id="Gateway_1j1atoa" name="Link 1" pv:PVNodeType="VSF">
  <bpmn2:incoming>Flow_0tn9ape</bpmn2:incoming>
  <bpmn2:outgoing>Flow_1a4uctn</bpmn2:outgoing>
</bpmn2:gateway>
```

Ilustración 82 - XML exportado con el valor "VSF". Fuente: Elaboración propia.

4.5.4 Sprint Review

En este sprint el propósito era desarrollar la notación de dependencias verticales en el editor.

La notación de dependencias verticales es compuesta de dos símbolos: el hexágono de borde grueso y el hexágono de doble borde.

Dicha notación ha sido implementada con éxito dentro del editor, permitiendo también utilizarla con las demás notaciones presentes.

Con esto, se concluye el trabajo de desarrollo de la herramienta.

Capítulo 5: Caso de estudio para validación de la herramienta

En este capítulo se implementará un caso de estudio en la herramienta desarrollada con el propósito de validarla. En seguida se mostrará los diagramas diseñados en la herramienta y un fragmento del XML exportado.

5.1 Introducción

Con el propósito de validar el funcionamiento de la herramienta, se ha modelado un modelo del proyecto SMR (Smart Mature Resilience). El proyecto SMR tiene como objetivo entregar un modelo de gestión de resiliencia a ciudades para que estas puedan gestionar su resiliencia. Por resiliencia se entiende la capacidad que una ciudad tiene de resistir, absorber, acomodar y recuperarse de un posible desastre. [18]

El proyecto SMR tiene un modelo llamado Resilience Maturity Model, que es un modelo que muestra el proceso que una ciudad tiene que hacer para alcanzar un nivel más alto de resiliencia. Este modelo puede ser usado para planificar estrategias, identificar los niveles de madurez de resiliencia, desarrollar e implementar políticas de desarrollo de resiliencia, establecer un punto de referencia de auto-critica para que una ciudad sepa su nivel de resiliencia y entregar justificativas suficientes para que puedan pedir recursos para poder mejorar su resiliencia. [19]

Este modelo es compuesto por 5 grados: Inicial, Moderado, Avanzado, Robusto y Vertebrado; identifica 13 actores: gobierno local, servicios de emergencia, infraestructuras críticas, empresas público y privadas, ONGs, voluntarios, gobiernos regionales, prensa, ciudadanos, entidades académicas y científicas, gobiernos nacionales, políticos a nivel europeo, organizaciones internacionales. Está dividida en 4 dimensiones: liderazgo y gobernabilidad, preparación, infraestructura y recursos y cooperación.

Finalmente, el modelo utilizado para validar la herramienta fue presentado en el artículo [10] presentado en la Ilustración 83. En este fragmento se muestra las dependencias entre las etapas P2 – educación y formación de la dimensión de preparación y C2 – intervención en redes de ciudades sobre resiliencia de la dimensión de cooperación.

En la Ilustración 83, las dependencias verticales son definidas por las flechas verdes y las dependencias horizontales son definidas por las flechas naranjas. A su vez, se entiende que las políticas estarán dentro de puntos de variabilidad de la notación Provop, y, en general, las políticas tendrán serán del tipo *Abstract* de la notación PESOA. Adicionalmente, las flechas verdes, es decir: las dependencias verticales, mostrarán los puntos de la notación de dependencias verticales.

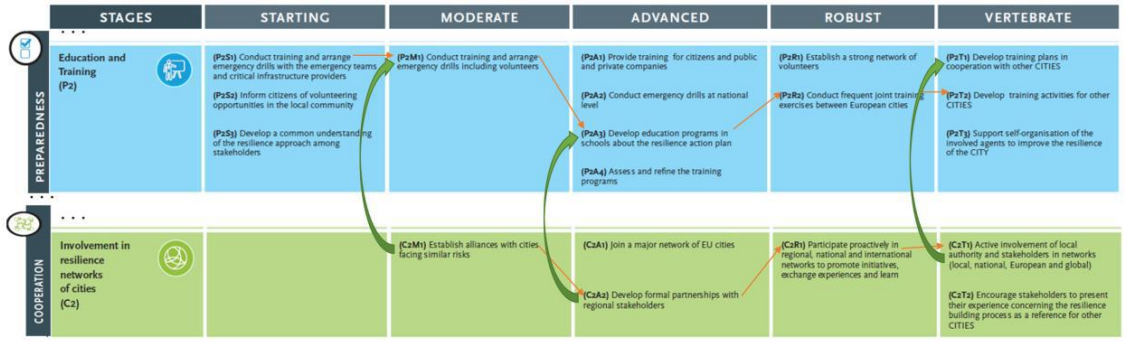

Ilustración 83 - Fragmento de la matriz RMM. Fuente: [10]

5.2 Modelado de Políticas

Siguiendo el proceso descrito en la ilustración 83, se ha diseñado el modelado de sus procesos en la herramienta desarrollada.

5.2.1 Education and training (P2) – Starting

La etapa starting de la dimensión Education and training está compuesta por 3 tareas que han sido etiquetadas como “Abstract”. Adicionalmente, estas tareas se encuentran en puntos de ajuste, que significa que pueden ser perfeccionadas. Las tres tareas pueden ser ejecutadas en paralelo. La ilustración 84 muestra esta etapa modelada en el editor.

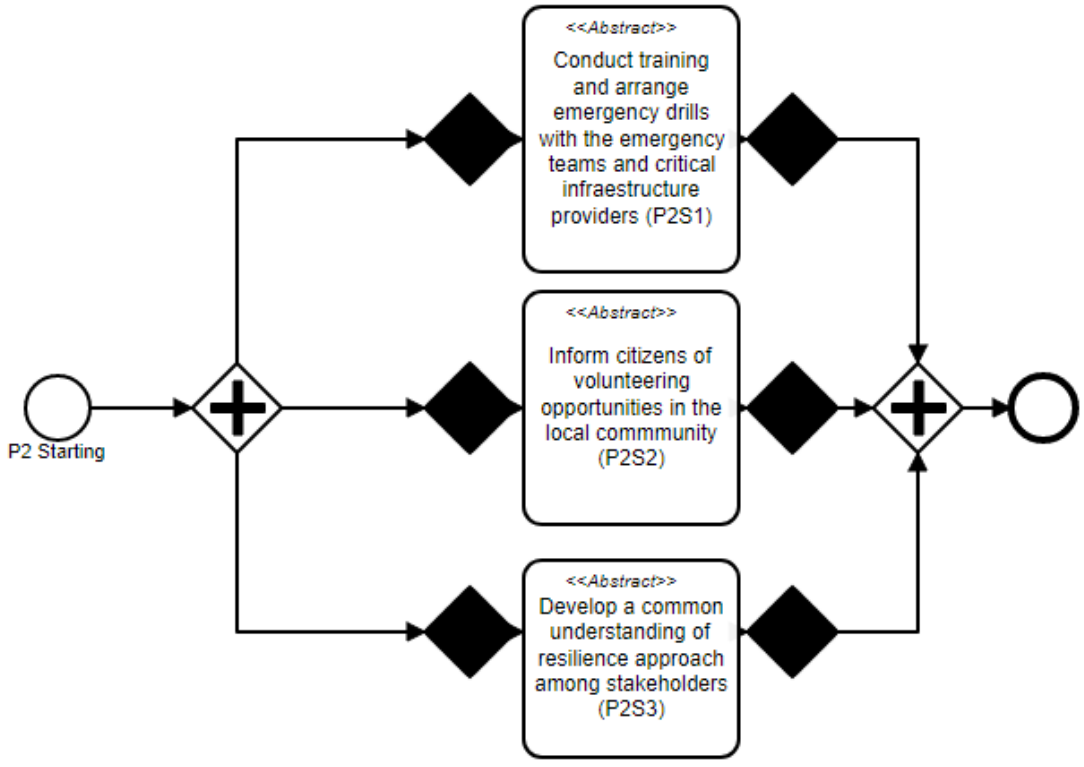

Ilustración 84 - Education and training (P2) - Starting. Fuente: Elaboración propia

5.2.2 Education and training (P2) - Moderate

La etapa Moderate está compuesta apenas de una tarea “Abstract”, que también puede ser especificada en un futuro (debido a esto los puntos de ajuste). Para poder hacer esta tarea, primeramente se debe concluir la tarea C2M1 – Establish alliances with cities facing similar risks, es decir, existe una dependencia vertical, así que se añade la notación de entrada de dependencias verticales para establecer este punto. La ilustración 85 muestra esta etapa modelada en el editor.

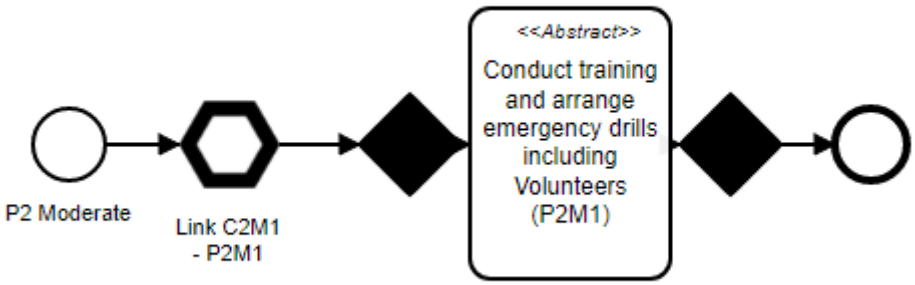

Ilustración 85 - Educational and training (P2) – Moderate. Fuente: Elaboración propia

5.2.3 Education and training (P2) - Advanced

La etapa Advanced está compuesta de cuatro tareas, de la misma manera que las anteriores etapas, estas tareas son “Abstract” y pueden ser especificadas, así que están entre puntos de ajuste. Dichas tareas se ejecutan en paralelo. Adicionalmente, para poder realizar la tarea P2A3 - Develop education program in schools about the resilience action plan, es necesario cumplir con la tarea C2A2 - Develop formal partnerships with regional stakeholders, es decir, una dependencia vertical. De manerea que, antes de la tarea P2A3 se añade una notación de entrada de dependencia vertical. La ilustración 86 muestra esta etapa modelada en el editor.

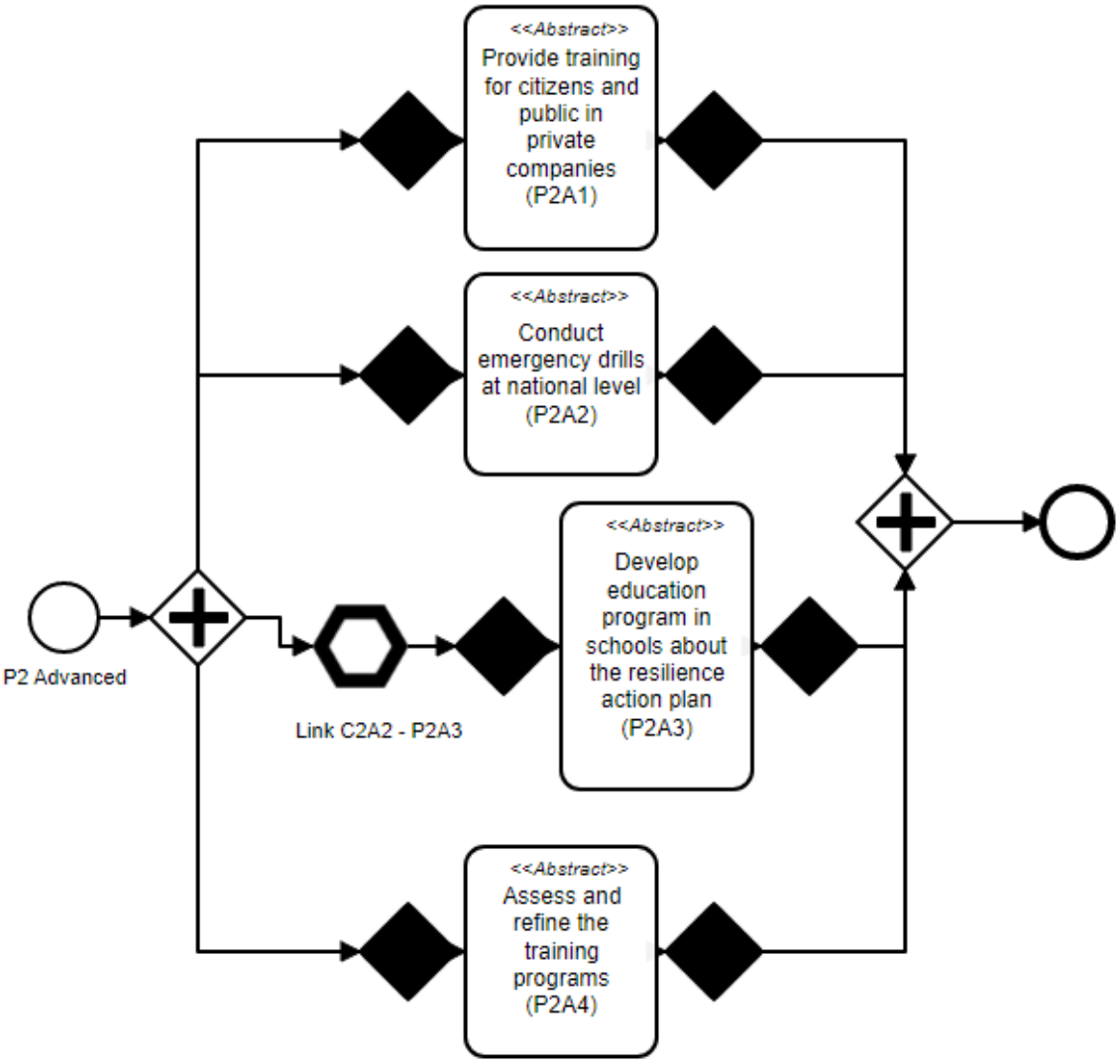

Ilustración 86 - Educational and Training (P2) – Advanced. Fuente: Elaboración propia

5.2.4 Education and training (P2) - Robust

Para la etapa Robust, son necesarias apenas dos tareas, que han sido etiquetadas como “Abstract”, y también pueden ser especificadas, de tal forma, han sido colocadas entre puntos de ajuste y se ejecutan en paralelo. La ilustración 87 muestra esta etapa modelada en el editor.

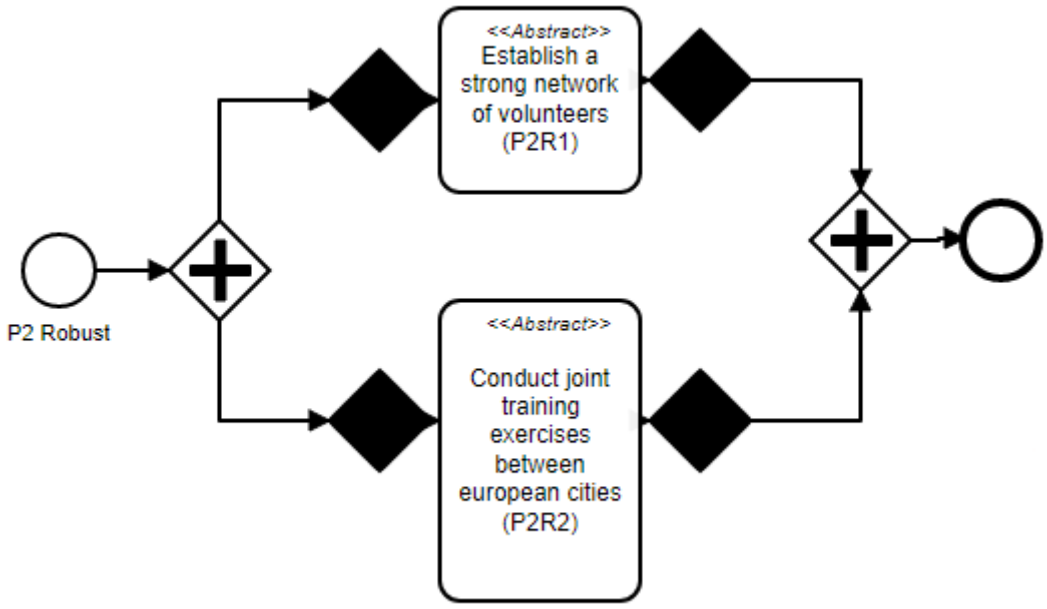

Ilustración 87 - Education and training (P2) – Robust. Fuente: Elaboración propia

5.2.5 Education and training (P2) - Vertebrate

Esta etapa es compuesta también por tres actividades “Abstract”, y posee una dependencia vertical con la tarea C2T1 - Active involvement of local authority and stakeholdesin networks antes de la tarea P2T1 - Develop training plans in cooperation with other cities (P2T1). La ilustración 88 muestra esta etapa modelada en el editor.

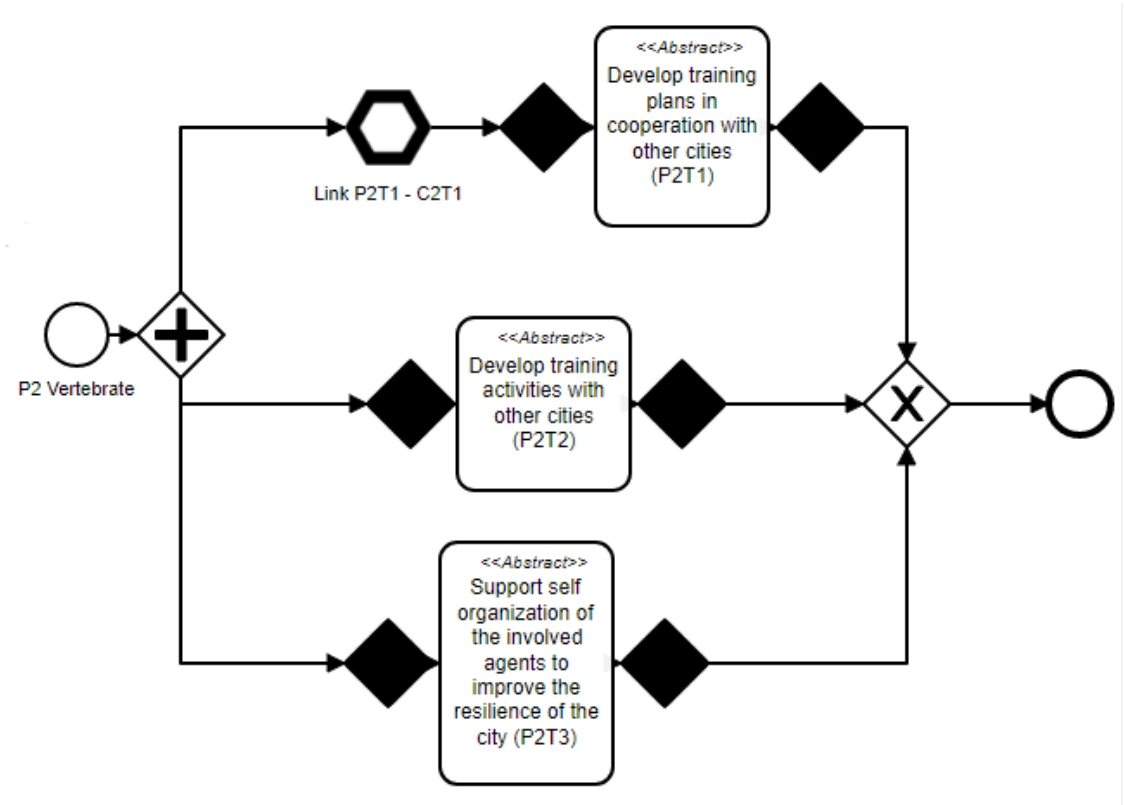

Ilustración 88 - Education and training (P2) – Vertebrate. Fuente: Elaboración propia

5.2.6 Involvement in resilience network of cities (C2) - Moderate

La etapa C2 está formada por apenas una tarea, pero esta tarea es el prerrequisito para la tarea P2M1, de esta forma, se crea una dependencia vertical. Debido a esto, se adiciona un punto de dependencia vertical de salida al terminar la tarea. La ilustración 89 muestra esta etapa modelada en el editor.

Ilustración 89 - Involvement in resilience network of cities (C2) - Moderate. Fuente: Elaboración propia

5.2.7 Involvement in resilience network of cities (C2) - Advanced

Esta etapa está compuesta de dos tareas, al igual que las demás etapas, entre puntos de ajustes y del tipo “Abstract”. Asimismo, la tarea C2A2 - Develop

formal partnerships with regional stakeholders es prerrequisito para la tarea P2A3 - Develop education program in schools about the resilience action plan, generando así una dependencia vertical. Debido a esto se ha añadido un punto de dependencia vertical de salida al final de esta tarea. La ilustración 90 muestra esta etapa modelada en el editor.

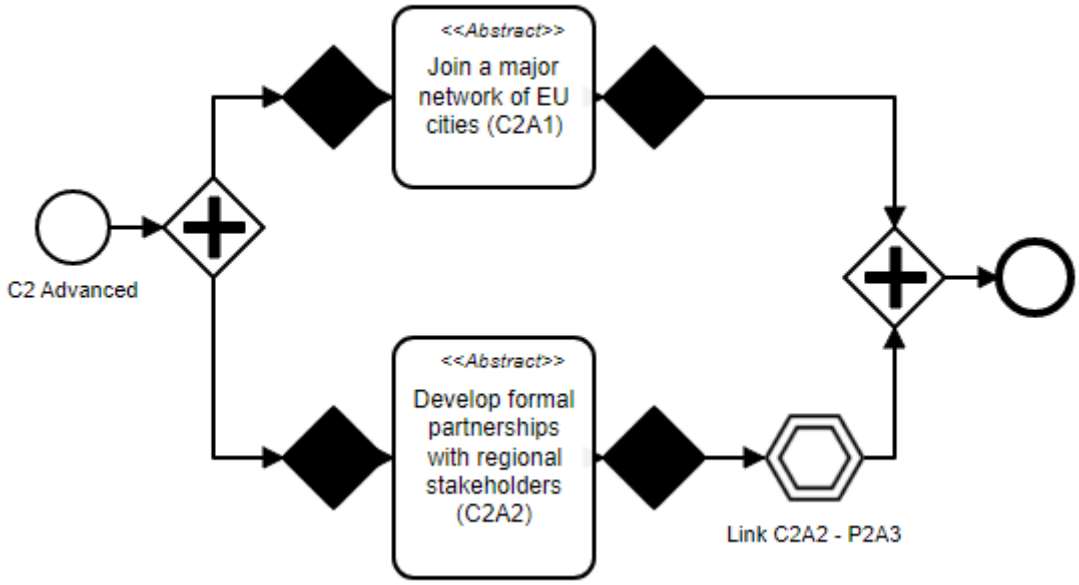

Ilustración 90 - Involvement in resilience network of cities (C2) – Advanced. Fuente: Elaboración propia

5.2.8 Involvement in resilience network of cities (C2) - Robust

La etapa Robust está compuesta apenas por una tarea, también marcada como "Abstract" y está posicionada entre puntos de ajuste. La ilustración 91 muestra esta etapa modelada en el editor.

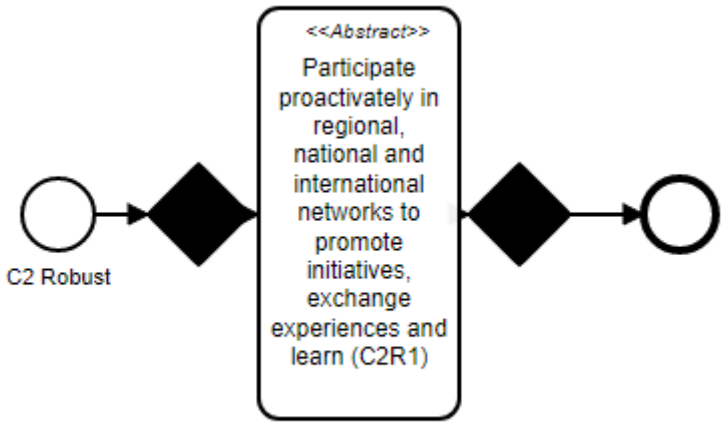

Ilustración 91 - Involvement in resilience network of cities (C2) – Robust. Fuente: Elaboración propia

5.2.9 Involvement in resilience network of cities (C2) – Vertebrate

La etapa Vertebrate está compuesta por dos tareas, siendo que la tarea C2T1 - Active involvement of local authority and stakeholdesin networks es prerrequisito para la tarea P2T1 - Establish a strong network of volunteers, generando así una dependencia vertical. Debito a esto, se añade al final de esta tarea la notación de dependencia vertical de salida. La ilustración 92 muestra esta etapa modelada en el editor.

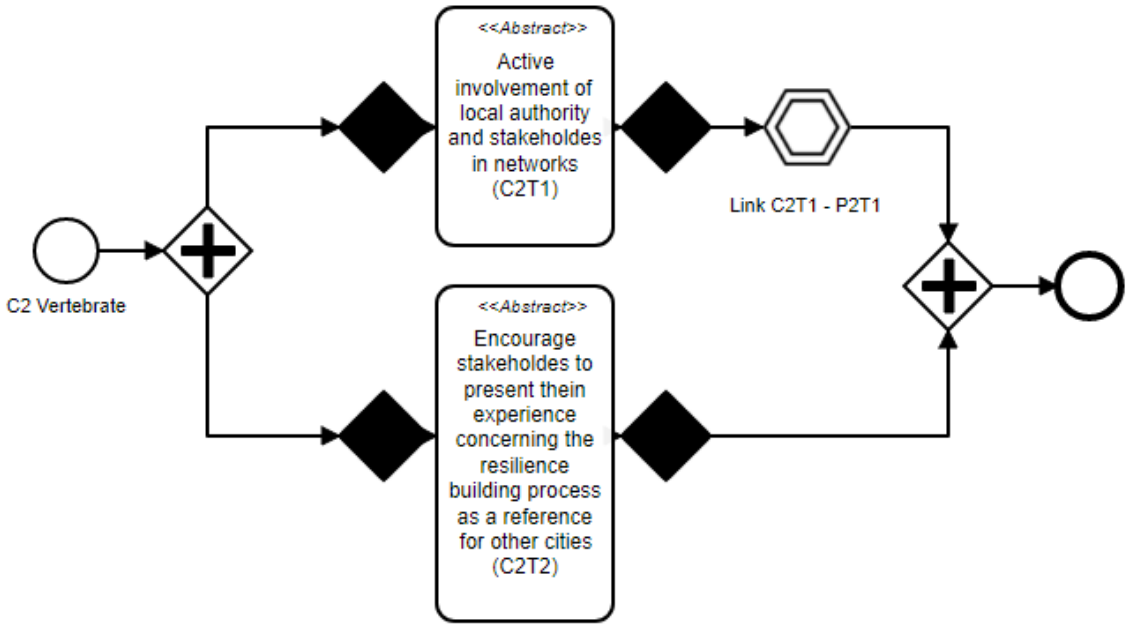

Ilustración 92 - Involvement in resilience network of cities (C2) – Vertebrate. Fuente: Elaboración propia

5.3 Representación XML de los procesos

Teniendo construido el diagrama, es posible exportarlo en un fichero XML.

Este fichero posee la notación BPMN 2.0 extendida creada para el fin de poder contemplar las notaciones de variabilidad de procesos. El fichero XML del diagrama expuesto en el apartado 5.2 posee 1211 líneas. La lustración 93 muestra las primeras líneas de este diagrama. En esta ilustración es posible contemplar algunos nodos que implementan la notación extendida, con el atributo PVNodeType=”AP”

```
<?xml version="1.0" encoding="UTF-8"?>
<bpmn2:definitions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xm
  <bpmn2:process id="Process_1">
    <bpmn2:startEvent id="Event_14cqhti" name="P2 Starting">
      <bpmn2:outgoing>Flow_0sb45p4</bpmn2:outgoing>
    </bpmn2:startEvent>
    <bpmn2:sequenceFlow id="Flow_0sb45p4" sourceRef="Event_14cqhti" targetR
    <bpmn2:parallelGateway id="Gateway_1jc2941">
      <bpmn2:incoming>Flow_0sb45p4</bpmn2:incoming>
      <bpmn2:outgoing>Flow_0xy3zn3</bpmn2:outgoing>
      <bpmn2:outgoing>Flow_11g8ozm</bpmn2:outgoing>
      <bpmn2:outgoing>Flow_1m2xr7g</bpmn2:outgoing>
    </bpmn2:parallelGateway>
    <bpmn2:task id="Activity_1qv96db" name="Conduct training and arrange em
      <bpmn2:incoming>Flow_0wc0gg0</bpmn2:incoming>
      <bpmn2:outgoing>Flow_0detxes</bpmn2:outgoing>
    </bpmn2:task>
    <bpmn2:gateway id="Gateway_07qv37o" pv:PVNodeType="AP">
      <bpmn2:incoming>Flow_0xy3zn3</bpmn2:incoming>
      <bpmn2:outgoing>Flow_0wc0gg0</bpmn2:outgoing>
    </bpmn2:gateway>
    <bpmn2:sequenceFlow id="Flow_0wc0gg0" sourceRef="Gateway_07qv37o" targe
    <bpmn2:sequenceFlow id="Flow_0xy3zn3" sourceRef="Gateway_1jc2941" targe
    <bpmn2:gateway id="Gateway_0qw67wm" pv:PVNodeType="AP">
      <bpmn2:incoming>Flow_11g8ozm</bpmn2:incoming>
      <bpmn2:outgoing>Flow_01gdimw</bpmn2:outgoing>
    </bpmn2:gateway>
    <bpmn2:sequenceFlow id="Flow_11g8ozm" sourceRef="Gateway_1jc2941" targe
    <bpmn2:task id="Activity_0q05vcv" name="Inform citizens of volunteering
      <bpmn2:incoming>Flow_01gdimw</bpmn2:incoming>
      <bpmn2:outgoing>Flow_0xqrb2a</bpmn2:outgoing>
    </bpmn2:task>
    <bpmn2:sequenceFlow id="Flow_01gdimw" sourceRef="Gateway_0qw67wm" targe
    <bpmn2:gateway id="Gateway_1iq1l3l" pv:PVNodeType="AP">
      <bpmn2:incoming>Flow_1m2xr7g</bpmn2:incoming>
      <bpmn2:outgoing>Flow_1rcarjp</bpmn2:outgoing>
    </bpmn2:gateway>
    <bpmn2:sequenceFlow id="Flow_1m2xr7g" sourceRef="Gateway_1jc2941" targe
    <bpmn2:task id="Activity_0k1zj8e" name="Develop a common understanding
      <bpmn2:incoming>Flow_1rcarjp</bpmn2:incoming>
      <bpmn2:outgoing>Flow_1edrpmt</bpmn2:outgoing>
    </bpmn2:task>
    <bpmn2:sequenceFlow id="Flow_1rcarjp" sourceRef="Gateway_1iq1l3l" targe
    <bpmn2:gateway id="Gateway_00ds502" pv:PVNodeType="AP">
      <bpmn2:incoming>Flow_0detxes</bpmn2:incoming>
      <bpmn2:outgoing>Flow_1322hjv</bpmn2:outgoing>
    </bpmn2:gateway>
    <bpmn2:sequenceFlow id="Flow_0detxes" sourceRef="Activity_1qv96db" targ
    <bpmn2:gateway id="Gateway_0ofiq5e" pv:PVNodeType="AP">
      <bpmn2:incoming>Flow_0xqrb2a</bpmn2:incoming>
      <bpmn2:outgoing>Flow_0jkezaa</bpmn2:outgoing>
    </bpmn2:gateway>
    <bpmn2:sequenceFlow id="Flow_0xqrb2a" sourceRef="Activity_0q05vcv" targ
    <bpmn2:gateway id="Gateway_0ua1qig" pv:PVNodeType="AP">
      <bpmn2:incoming>Flow_1edrpmt</bpmn2:incoming>
      <bpmn2:outgoing>Flow_1fvteyj</bpmn2:outgoing>
    </bpmn2:gateway>
    <bpmn2:sequenceFlow id="Flow_1edrpmt" sourceRef="Activity_0k1zj8e" targ
    <bpmn2:parallelGateway id="Gateway_14lrtrp">
      <bpmn2:incoming>Flow_1322hjv</bpmn2:incoming>
      <bpmn2:incoming>Flow_0jkezaa</bpmn2:incoming>
      <bpmn2:incoming>Flow_1fvteyj</bpmn2:incoming>
      <bpmn2:outgoing>Flow_1mb93qs</bpmn2:outgoing>
    </bpmn2:parallelGateway>
```

Ilustración 93 - Diagrama BPMN exportado en XML. Fuente: Elaboración propia

Capítulo 6: Conclusión y Trabajos Futuros

6.1 Conclusión

Este trabajo de final de máster tenía como propósito desarrollar una herramienta que permitiera, en un mismo editor de notación BPMN añadir tres notaciones personalizadas distintas: PESOA, PROVOP y Dependencias Verticales.

La notación PESOA es una notación de variabilidad de procesos para familias de procesos digitales. Su forma de representar la variabilidad de procesos es añadiendo estereotipos, propios del lenguaje UML, al BPMN. Estereotipos en UML son marcaciones que identifican los propósitos de ciertos elementos. De esta forma, PESOA propone que se adicione estereotipos a tareas y subprocesos del BPMN, siendo estos estereotipos <<VarPoint>>, representando un punto de variabilidad; <<Variant>>, representando una variabilidad; <<Abstract>>, representando un comportamiento alternativo; <<Null>>, representando un comportamiento opcional.

La notación PROVOP es un marco que permite marcar en el diagrama los puntos en los que serán hechos variabilidades en el proceso. Estos puntos son llamados de puntos de ajuste, y su notación es el diamante negro.

La notación de Dependencias Verticales es una notación que permite dar una continuidad entre dos procesos separados, mostrando que existe una dependencia entre estos dos procesos. Sus notaciones son los hexágonos de borde grueso, que tiene como propósito marcar el punto destino que debe ser conectado en un proceso, y hexágonos de doble borde, que tiene como propósito marcar el punto de origen que debe ser conectado en un proceso.

Para desarrollar esta herramienta se ha elegido como punto de partida un editor previamente existente, en el que se pudiera hacer las modificaciones pertinentes para poder implementar las notaciones citadas anteriormente.

Una vez hecha la búsqueda de los editores existentes en el mercado, se ha decidido utilizar el editor bpmn.io, hecho en javascript y puramente front-end.

De esa forma, han sido desarrolladas las modificaciones y se ha permitido tener un editor final que permita modelar procesos de negocios con notaciones específicas de variabilidad de procesos. Este editor ha sido bautizado como PVModeler. En la Ilustración 94 se muestra la interfaz del editor.

En consecuencia, se concluye que ha sido posible desarrollar y crear la herramienta propuesta inicialmente para el presente trabajo final de master.

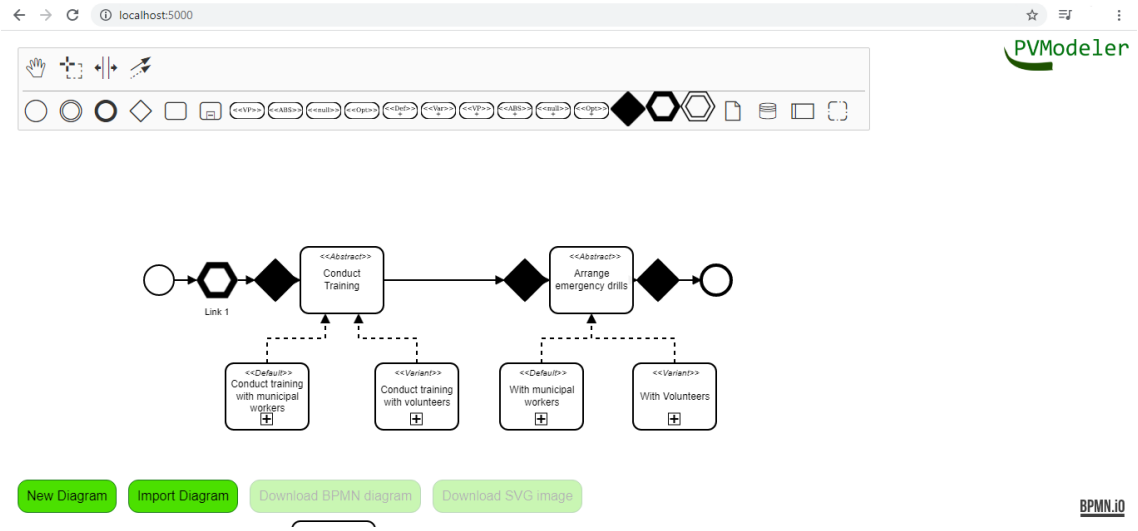

Ilustración 94 PVModeler. Fuente: Elaboración propia

6.2 Trabajos Futuros

Este editor de modelos de la notación BPMN 2.0 extendido será utilizado en el proyecto de investigación incremental para introducir la transformación digital en el marco SMR, modelando las políticas a aplicar en una ciudad para mejorar la resiliencia urbana. Esto abre nuevas vías de continuación del trabajo:

- Validación de la herramienta en entornos reales o proyectos pilotos
- Extensión de la herramienta para que soporte la composición de fragmentos de procesos, de acuerdo a las reglas indicadas por el usuario
- Analizar la ejecución de los modelos en motores de proceso como CAMUNDA
- Creación de herramientas que puedan utilizar los procesos exportados y la notación introducida por este trabajo de fin de máster para distintas funciones.

Bibliografía

[1] ABOUT THE BUSINESS PROCESS MODEL AND NOTATION SPECIFICATION VERSION 2.0.2, https://www.omg.org/spec/BPMN/#spec-versions. Accedido en 16/10/2021

[2] bpmn-js-examples/custom-elements at master · bpmn-io/bpmn-js-examples · GitHub. https://github.com/bpmn-io/bpmn-js-examples/tree/master/custom-elements. Accedido en 20/10/2021

[3] Concepts webpack. https://webpack.js.org/concepts/. Accedido en 07/10/2021

[4] GitHub - bpmn-io/bpmn-js-example-custom-rendering: An example of creating custom rendering for bpmn-js. https://github.com/bpmn-io/bpmn-js-example-custom-rendering. Accedido en 07/10/2021

[5] GitHub - bpmn-io/bpmn-js-example-model-extension: An example of creating a model extension for bpmn-js. https://github.com/bpmn-io/bpmn-js-example-model-extension. Accedido en 08/10/2021

[6] Hallerbach, Alena & Bauer, Thomas & Reichert, Manfred. (2010). Capturing variability in business process models: The Provop approach. Journal of Software Maintenance and Evolution: Research and Practice. 22. n/a-n/a. 10.1002/spip.446.

[7] Impellizzeri, Franco & Bizzini, Mario. (2012). Systematic review and meta-analysis: A primer. International journal of sports physical therapy. 7. 493-503.

[8] La Rosa, M., Van Der Aalst, W. M. P., Dumas, M., & Milani, F. P. (2017, March 1). Business process variability modeling: A survey. ACM Computing Surveys. Association for Computing Machinery. https://doi.org/10.1145/3041957

[9] Latifa Ilahi, Ricardo Martinho, Sonia Ayachi Ghannouchi, BPFlexTemplate: a software tool to derive flexible process model templates, Procedia Computer Science, Volume 121, 2017, Pages 1096-1103, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2017.11.140.

[10] Penadés, M. C., & Sánchez-diaz, J. (n.d.). Building Urban Resilience: A Dynamic Process Composition Approach. Aceptado, pendiente de publicación.

[11] Pereira, T., Alencar, F., & Castro, J. (2016). BVCCON-TOOL: A modeling tool to support dynamic business process configuration approach. CIBSE 2016 - XIX Ibero-American Conference on Software Engineering, April, 39–52.

[12] Pohl, K., B¨ockle, G., van der Linden, F.: Software Product Line Engineering: Foundations, Principles, and Techniques. Springer (2005)

[13] ¿Qué es la metodología 'agile'?, https://www.bbva.com/es/metodologia-agile-la-revolucion-las-formas-trabajo/ accedido en 05/10/2021

[14] Sbai, H., Faquih, L. El, & Fredj, M. (2019). A novel tool for configurable process evolution and service derivation. International Journal of Enterprise Information Systems, 15(2), 58–75. https://doi.org/10.4018/IJEIS.2019040104

[15] Schnieders, A. & Puhlmann, F., (2006). Variability mechanisms in e-business process families. In: Abramowicz, W. & Mayr, H. C. (Hrsg.), Business Information Systems – 9th International Conference on Business Information Systems (BIS 2006). Bonn: Gesellschaft für Informatik e.V.. (S. 583-601).

[16] 'Scrum', 'agile'... así son las nuevas formas de trabajo para la transformación de BBVA, https://www.bbva.com/es/bbva-asi-son-las-nuevas-formas-de-trabajo-para-acelerar-la-transformacion/. Accedido en 05/10/2021

[17] Schwaber, K, & Sutherland, J.: The scrum guide https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf

[18] SMR, About SMR. https://smr-project.eu/aboutsmr/. Accedido en 25/11/2021

[19] SMR, Maturity model guide. https://smr-project.eu/tools/maturity-model-guide/. Accedido en 25/11/2021

[20] Uman L. S. (2011). Systematic reviews and meta-analyses. Journal of the Canadian Academy of Child and Adolescent Psychiatry = Journal de l'Academie canadienne de psychiatrie de l'enfant et de l'adolescent, 20(1), 57–59.

[21] UML stereotypes - IBM Documentation. https://www.ibm.com/docs/en/rsm/7.5.0?topic=elements-uml-stereotypes. Accedido en 10/10/2021

[22] Variability (BPM 2016) – PINOT, https://www.isa.us.es/ppinot/variability-bpm2016/. Accedido en 15/10/2021

[23] Weske, M. (2007). Business Process Management: Concepts, Languages, Architectures. 978-3-540-73521-2. Springer-Verlag Berlin Heidelberg 2007.

[24] What is npm? – Node JS. https://nodejs.org/en/knowledge/getting-started/npm/what-is-npm/. Accedido en 07/10/2021

[25] What is a Product Backlog?. https://www.scrum.org/resources/what-is-a-product-backlog. Accedido en 01/11/2021

[26] Zur Muehlen, M., & Su, J. (2011). Lecture Notes in Business Information Processing: Preface. Lecture Notes in Business Information Processing, 66 LNBIP(September). https://doi.org/10.1007/978-3-642-20511-8

Anexos

Anexo 1 – Manual de uso del PVModeler

Sobre PVModeler

PVModeler es un editor web desarrollado para modelar diagramas en BPMN 2.0 con notación extendida para permitir modelar variabilidad de procesos. Las notaciones de variabilidad de procesos que están implementadas son PESOA, introducida en el artículo Variability Mechanisms in E-Business Process Families (Schnieders, A., et al), PROVOP, introducido en el artículo Capturing variability in business process models: The Provop approach (Hallerbach, Alena & Bauer, et al), y la notación de dependencias verticales introducidas por [10] Penadés, M. C., & Sánchez-diaz, J. (n.d.). Building Urban Resilience: A Dynamic Process Composition Approach.

Requisitos

PVModeler ha sido comprobado y validado con éxito en los navegadores Google Chrome, versión 96, Mozilla Firefox versión 94, Microsoft Edge versión 96. El editor también ha sido probado en el navegador Microsoft Internet Explorer pero sin éxito. De los tres navegadores que han funcionado con éxito, el editor ha presentado mejor rendimiento en el Mozilla Firefox y en el Google Chrome.

Funcionamiento

Al abrir PVModeler en el navegador se cargado un ejemplo de diagrama modelado con variabilidad de procesos y todas las notaciones implementadas. La interfaz que se muestra es la siguiente:

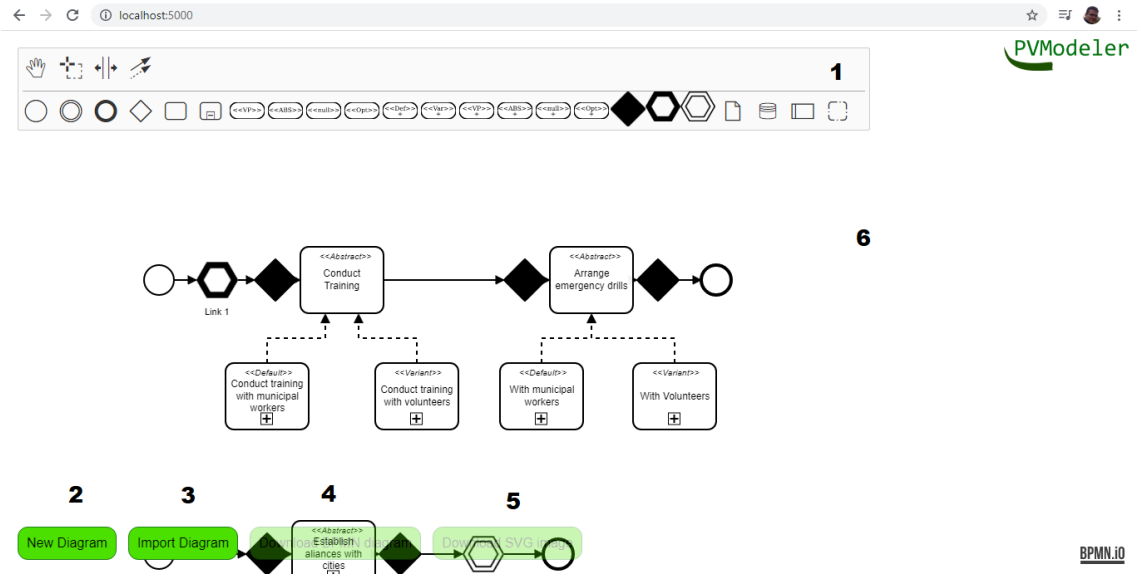

Ilustración 95 Interfaz gráfica PVModeler. Fuente: Elaboración propia.

En la figura se muestra:

1. Palette: área en el que se encuentran los objetos (tasks, subprocess, gateways, etc.) para uso en el diagrama
2. New Diagram: Botón que limpiará el editor.
3. Import Diagram: Botón que permite importar un diagrama existente
4. Download BPMN Diagram: botón que permite descargar el diagrama en formato XML
5. Download SVG Image: botón que permite descargar la imagen del diagrama en formato svg.
6. Espacio del editor para el modelado.

Modelado

Para añadir un elemento al espacio para modelado, el usuario debe hacer clic en el elemento en el palette, posicionarlo donde quiera en el espacio para modelado, y hacer clic:

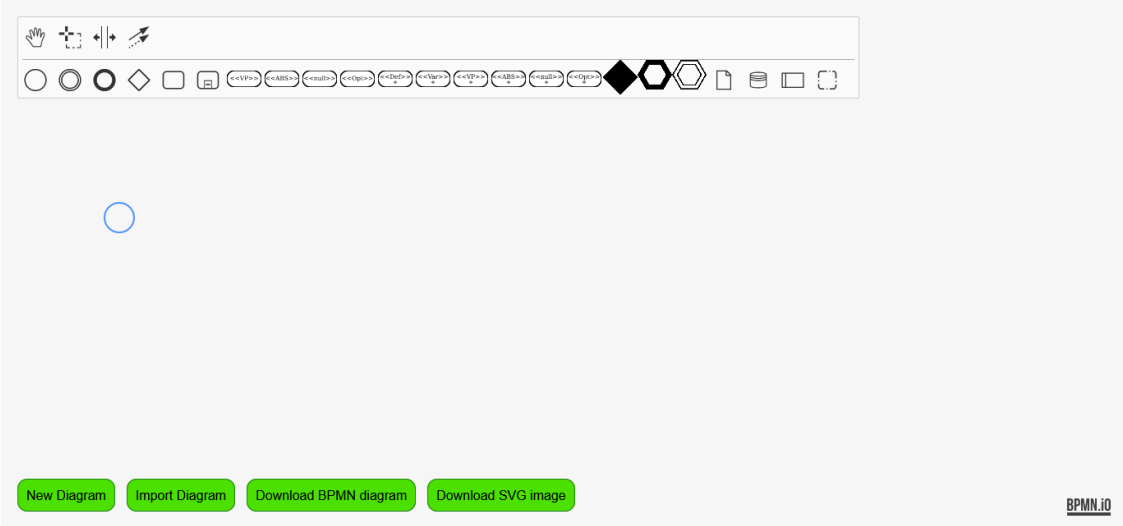

Ilustración 96 Objeto sobre editor. Fuente: Elaboración propia.

Al posicionarlo abrirá el Pad, con algunas opciones de objetos que se pueden añadir a continuación.

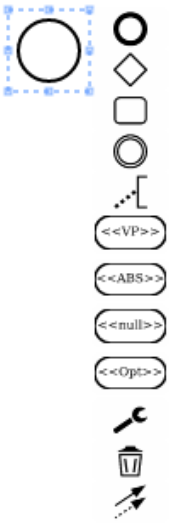

Ilustración 97 Pad de objeto BPMN por defecto. Fuente: Elaboración propia.

Algunos objetos tienen opciones adicionales, como los gateways, que pueden tomar forma como parallel gateway, inclusive gateway, complex gateway, event based gateway y exclusive gateway. De esta forma, para acceder a estos objetos, se debe hacer clic en el icono , que abrirá una ventana para elegir estas opciones.

Es posible agrandar o achicar cualquier elemento arrastrando los puntos azules más grandes que están alrededor del objeto.

Para conectar dos elementos, el usuario debe hacer clic en en el palette, y hacer clic en el primer en el elemento de origen, y en seguida en el elemento de destino. De esta forma se creará el conector direccional.

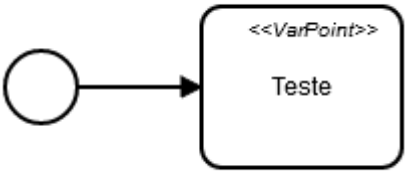

Ilustración 98 Notación PESOA. Fuente: Elaboración propia.

Al conectar dos elementos de la notación PESOA, el conector tendrá forma de línea discontinua, y la orientación seguirá la orientación del clic.

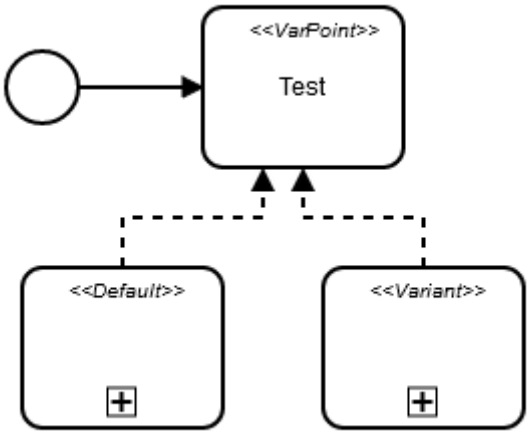

Ilustración 99 Notación PESOA. Fuente: Elaboración propia.

Los objetos de la notación PESOA tienen el mismo funcionamiento que tareas o subprocesos de la notación BPMN. Para expandir un subproceso, el usuario debe hacer clic en  y seleccionar Sub Process (Expanded).

Para adicionar un punto de variabilidad, el usuario debe hacer clic en el icono  , y posicionarlo en el editor. En seguida, debe hacer clic en el elemento que estará dentro del punto de variabilidad y conectarlos.

Ilustración 100 Notación PESOA con PROVOP. Fuente: Elaboración propia.

Después, puede posicionar el icono  al final del punto de variabilidad y conectar el objeto anterior a este icono.

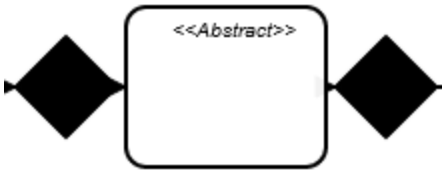

Ilustración 101 Notación PESOA con PROVOP. Fuente: Elaboración propia.

Si fuese necesario, el usuario puede arrastrar el objeto más cerca del punto de variabilidad para que no muestre la línea.

Para posicionar un elemento de la notación de dependencias verticales, el usuario sólo debe posicionarlo en el editor, y conectarlo con su elemento de entrada, y después conectarlo con su elemento de salida.

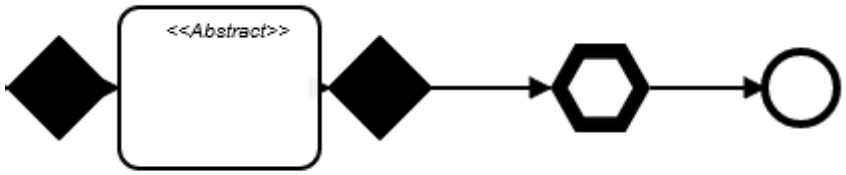

Ilustración 102 Notación PESOA, PROVOP y dependencias verticales. Fuente: Elaboración propia.

Al hacer doble clic en cualquier elemento permite poner un título

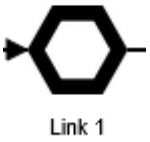

Ilustración 103 Objeto con título. Fuente: Elaboración propia.

Si se desea borrar un elemento, el usuario debe hacer clic en  y hacer clic en  . El editor irá reposicionar o borrar los conectores de este elemento de forma automática.

Exportación

Al hacer clic en “Download BPMN diagram”, se descargará el diagrama XML de la notación BPMN extendida.

Importación

Al hacer clic en “Import Diagram”, se abrirá la ventana de ficheros del navegador que permite hacer el upload de un diagrama para editarlo. Es importante aclarar que los formatos aceptables son apenas el formato de diagrama exportado por Camunda (que no posee las notaciones del BPMN Extendido para variabilidad de procesos) y el diagrama exportado por la propia herramienta.

Descargar imagen del diagrama

Al hacer clic en “Download SVG Image”, descargará la imagen del diagrama en formato SVG.

Anexo 2 – Manual de publicación

Para publicar la aplicación, el usuario debe tener, inicialmente el proyecto con el código del editor PVModeler.

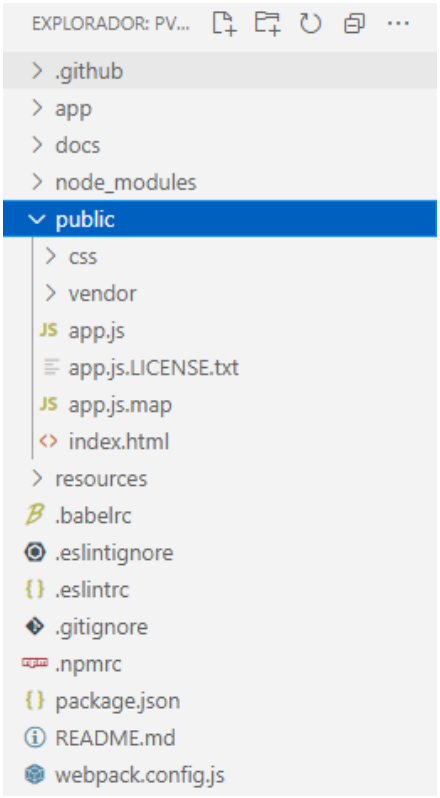

Ilustración 104 Listado de carpetas del proyecto con detalle a la carpeta "public". Fuente: Elaboración propia

Con el propósito de ilustrar, se el panel CPanel como ejemplo.

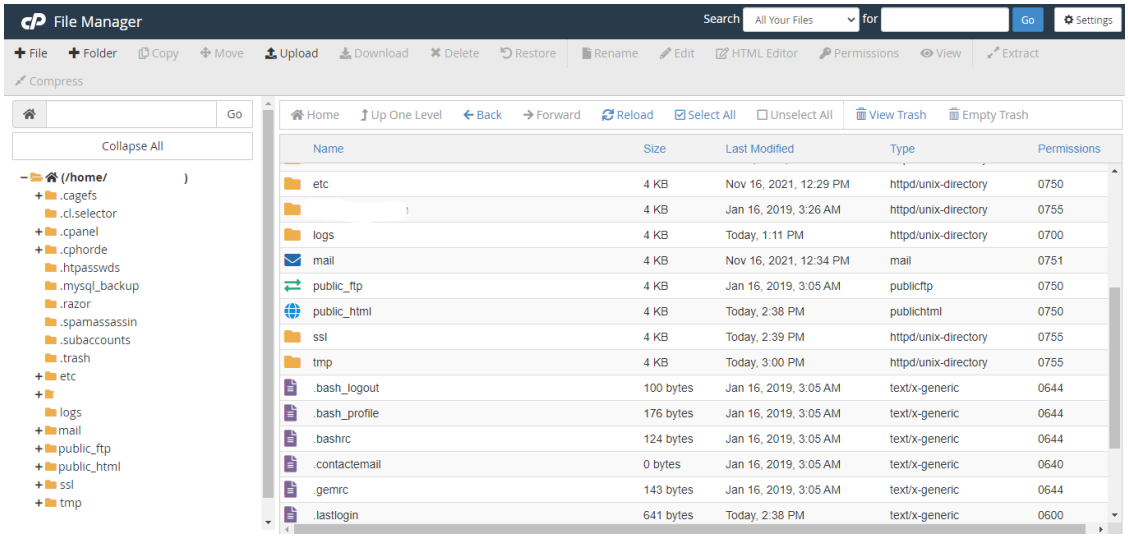

Ilustración 105 - Explorador de ficheros de CPanel. Fuente: Elaboración propia

Para publicar en el CPanel, el usuario debe dirigirse a la carpeta public_html, que es el directorio para publicar páginas web en servidores Linux.

Dentro del directorio el usuario debe copiar todos los ficheros de la carpeta “public” del proyecto. La carpeta public tiene los ficheros preparados para la publicación.

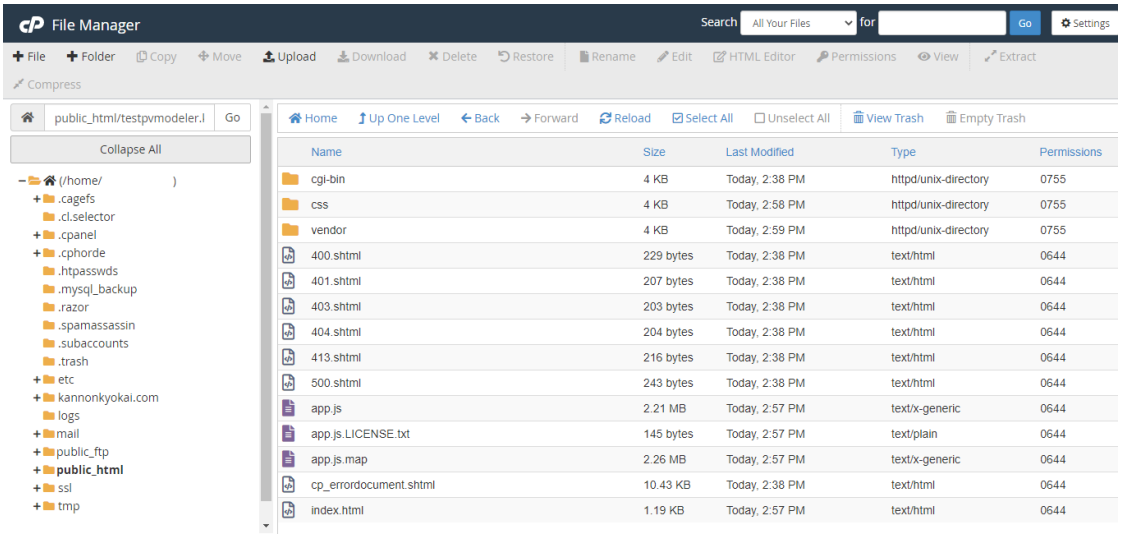

Ilustración 106 Ficheros publicados en el servidor. Fuente: Elaboración propia

De esta forma, al acceder la URL del servidor, abrirá el PVModeler.

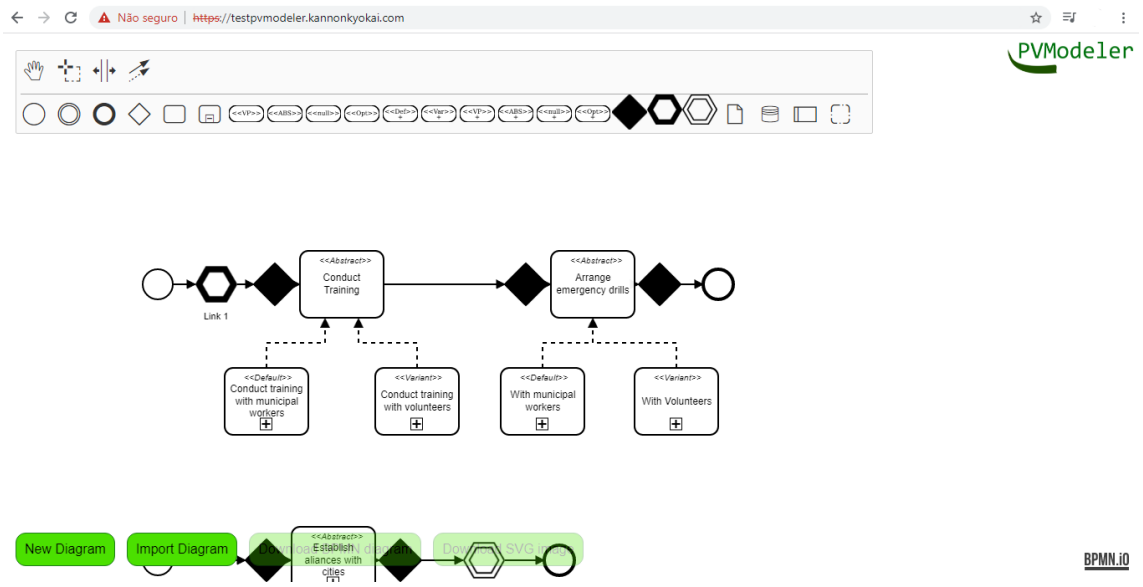

Ilustración 107 PVModeler cargado desde el servidor. Fuente: Elaboración propia