Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

# Trabajo de Fin de Grado

Escuela Superior de Ingeniería y Tecnología

Grado en Ingeniería Informática

# Simulación de BPMN mediante una librería de simulación de eventos discretos

*BPMN simulation through a discrete event simulation library*

La Laguna, 3 de julio de 2019

D. **Iván Castilla Rodríguez**, con N.I.F. 78.565.451-G profesor Ayudante Doctor adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor.

**C E R T I F I C A**

Que la presente memoria titulada:

*"Simulación de BPMN mediante una librería de simulación de eventos discretos"*

ha sido realizada bajo su dirección por D. **Andrea Pérez Quintana**,

con N.I.F. 43.834.204-E.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 3 de julio de 2019

# Agradecimientos

En primer lugar, agradecerle a mi tutor, Iván Castilla, por la orientación que me ha prestado en todo momento y su gran predisposición para ayudarme.

Y en especial, agradecerle a mi familia, pareja y amigos por haberme aportado los medios y el apoyo que he necesitado y por darme la motivación y las ganas para continuar.

Agradezco también a mis compañeros y profesores, por los momentos compartidos y las lecciones aprendidas que me han llevado a este instante de mi vida.

# Licencia

**Resumen**

*Las empresas deben analizar sus procesos para optimizar su productividad. Hasta los procesos más rutinarios pueden mejorarse. Muchas veces por falta de tiempo o por haber realizado ya análisis con anterioridad no se dan cuenta de los posibles errores; pero dedicar un poco de tiempo al análisis y optimización de procesos hace notar las mejoras de inmediato.*

*El análisis de estos procesos se puede llevar a cabo mediante tres fases; comenzando con el adecuado modelado, por medio del cual se describe la organización actual de la empresa. A continuación, realizando la implementación y ejecución de los procesos y, por último, la monitorización de los mismos.*

*Con el modelado conseguimos representar la situación actual de la organización, pero se requieren herramientas de simulación para poder analizar el impacto de modificaciones en estos procesos. La información que recibimos de ambos conceptos, nos ayuda a optimizarlos y corregirlos.*

*En este Trabajo de Fin de Grado se estudia la posibilidad de conversión entre ambos conceptos. Esto aporta una facilidad a la hora de modelar y simular los procesos, ya que no habrá que realizarlo de manera aislada, sino existirá la opción de hacerlo conjuntamente.*

*La Universidad de La Laguna dispone de un proyecto con una librería de simulación de eventos discretos, en el cual, participa el tutor de este TFG, Iván Castilla.*

*Siendo así, el objetivo de este proyecto es lograr convertir de un modelo con un lenguaje específico, en nuestro caso será BPMN, a Workflow Patterns (patrones de flujo de trabajo), y simular estos mediante una librería de simulación de eventos discretos.*

*El resultado obtenido es un análisis exhaustivo de la relación y las estrategias de conversión de un modelo a otro, y el desarrollo de una aplicación capaz de generar modelos de simulación a partir de modelos estáticos de descripción de procesos.*

**Palabras clave:** Modelado, simulación, procesos, BPMN, patrones de flujo de trabajo, simulación de eventos discretos.

**Abstract**

*Companies must analyse their processes to optimize their productivity and improve their routine processes. Frequently, due to not having time or because they have already performed analyses before, they do not find the possible errors; but if they take a bit of time to analyse and optimize processes, they can notice the improvements immediately.*

*The analysis of these processes can be carried out through three phases; starting with the appropriate modeling, that aims to consider the current enterprise organization. Then, performing the implementation and execution of the processes and, finally, the monitoring of them.*

*With modelling, we can represent the current situation of the organization, but simulation tools are required to analyse the impact of changes in these processes. The information we receive from both concepts helps us optimize and correct them.*

*This dissertation studies the possibility of conversion between the concepts of modelling and simulation. In this way, it is easier to simulate and model any process. Because of that, it will not have to be done separately; also, there will be the option to do it together.*

*The University of La Laguna has a project with a discrete events simulation library, whose main developer is the tutor of this dissertation.*

*The objective of this project is to convert a model with a specific language to Workflow Patterns (in our case, from BPMN language to Workflow Patterns), and simulate them through the discrete events simulation library.*

*The result obtained is an exhaustive analysis of the relationship and conversion strategies from one model to another, and the development of an application capable of generating simulation models from static models of process description.*

**Keywords:** Modeling, simulation, processes, BPMN, Workflow Patterns, discrete events simulation.

# Índice general

# Índice de figuras

# Índice de tablas

# Capítulo 1

# Introducción.

Actualmente estamos en una era de constante avance tecnológico donde la tecnología y la informática forman parte de nuestro día a día. Es por ello por lo que las empresas deben adaptarse a esta transformación digital si quieren sobrevivir.

El objetivo general de las empresas es maximizar los beneficios minimizando los riesgos y las pérdidas. Por tanto, la introducción de la tecnología en el mundo empresarial se centra en ayudar a cumplir con dichos objetivos, y la mejor manera de hacerlo es analizando los procesos y los recursos que utilizan para optimizarlos lo máximo posible. Es en este punto donde interviene la informática. Un programa informático puede encargarse de realizar dichos análisis de forma automatizada mediante el modelado y la simulación de los procesos de negocio.

Un proceso de negocio [1] es un conjunto de tareas que son realizadas en un ambiente organizacional para lograr un objetivo. Un ejemplo de proceso de negocio es la petición del periodo vacacional de los empleados. En la **figura 1.1** se muestra una posible representación del mismo.

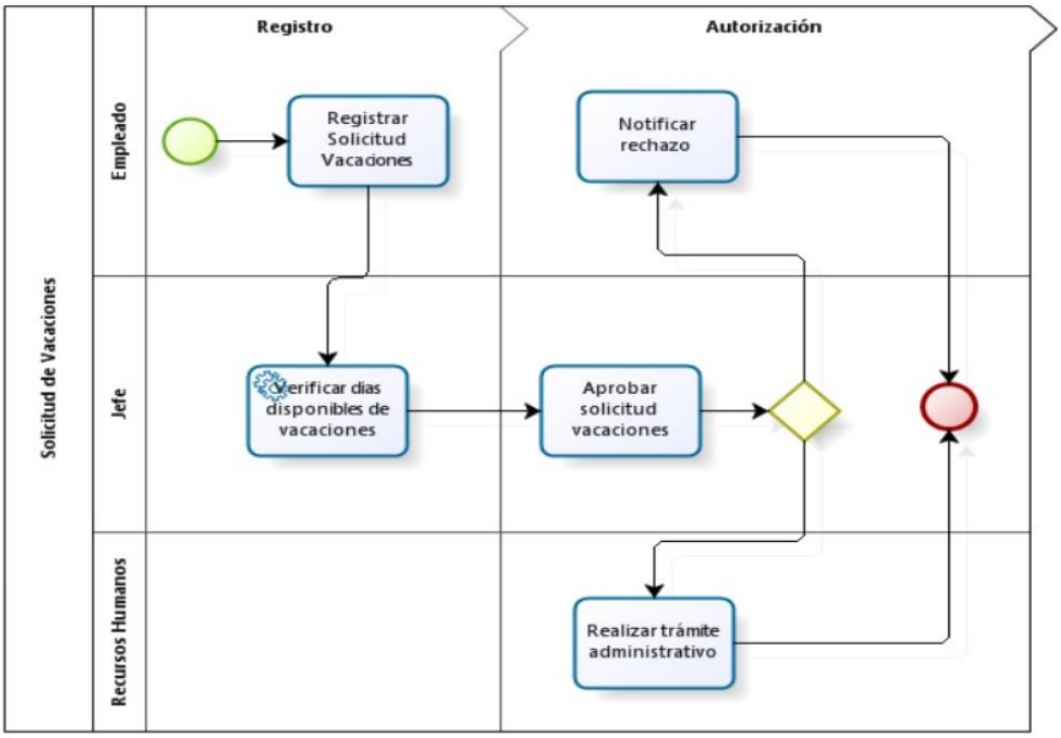


*Figura 1: 1.1. Ejemplo de proceso de negocio* Fuente: [30]

## 1.1 Modelo y Simulación

Modelo y simulación [2] son dos conceptos muy parecidos, pero no iguales. Un modelo es una representación que permite analizar, explicar y describir procesos. Sin embargo, una simulación no describe el proceso, sino que lo imita para conocer su comportamiento mediante la construcción de escenarios bajo distintas condiciones de operación.

En general, la principal diferencia es que el modelado se utiliza para describir el proceso y la simulación para predecir su comportamiento en función del tiempo, pero ambos dan lugar a datos que ayudan a optimizar los procesos.

Estas son herramientas rentables para diseñar o entender el comportamiento de los procesos. Es por ello, que las empresas hacen uso de ellos para obtener datos que le ayuden a optimizar los procesos antes de ponerlos en marcha, pero, sobre todo, para mejorar los procesos que ya realizan.

## 1.2 Motivación para el trabajo

Como se ha señalado, los métodos de modelado y simulación tienen una gran importancia en el ámbito empresarial [3] y gracias al avance tecnológico se ha facilitado la aplicación de los mismos.

Actualmente existe gran variedad de herramientas que permiten modelar los procesos de manera gráfica y sencilla y muchas otras que permiten el diseño y la simulación de éstos, aplicando diferentes condiciones para predecir el comportamiento. Unas herramientas son más completas que otras, algunas de ellas, se complementan entre sí y otras son completamente incompatibles.

La mayoría de las herramientas que aportan todo lo necesario y facilitan el uso y la transmisión y recepción de información son herramientas pagas. Sin embargo, existen muchas otras que no lo son y también pueden resultar muy útiles.

El problema reside en el trabajo de realizar dos veces el diseño para un mismo proceso, no tiene sentido realizar un modelo estático y luego crear un modelo de simulación basado en el primero.

Sabiendo esto, sería ideal obtener a partir de una herramienta de modelado gráfico de procesos una simulación que aporte la información necesaria.

En resumidas cuentas, se intenta conjugar la visión del negocio centrada en mejorar sus procesos mediante el análisis de los mismos y la visión de la tecnología centrada en digitalizarlos y automatizarlos aplicando metodologías que hagan uso de la tecnología y la informática.

Es por ello que, con este proyecto, se pretende elaborar una solución que nos permita desarrollar un solo modelo estático mediante alguna herramienta de BPM y, a través del mismo, realizar una simulación.

Para ello, se realizará un análisis de la herramienta de simulación diseñada en la Universidad de La Laguna, PSIGHOS, y un estudio de los datos que podemos extraer de un BPM (Business Process Model) para poder alcanzar una solución factible que nos facilite aprovechar todos los recursos sin necesidad de realizar dos veces el mismo trabajo.

# Capítulo 2
# Antecedentes

El caso de estudio de este trabajo es la continuación de diferentes proyectos de investigación y desarrollo a través de los cuales se ha logrado la creación de una herramienta de simulación que ya ha sido mencionada y que se describe en esta sección.

## 2.1 Patrones de flujo de trabajo

En la época en la que nos encontramos nos resultaría imposible producir todo lo que utilizamos para nuestra vida diaria nosotros mismos, hablamos de cosas materiales, como son la comida o la ropa, y de servicios, como los medios de transporte. Todo ello sin mencionar el mundo del entretenimiento. Esto también se aplica a las empresas, pues es muy difícil que una mismo departamento o unidad pueda encargarse de realizar diferentes trabajos.

Cuando hablamos de patrones de *workflow* [4], nos encontramos con terminología específica que deberíamos conocer, en su mayoría son términos en inglés, que aunque tengan traducción al español, se utilizan en su idioma original.

Hay muchos tipos de trabajo, cada uno de ellos dan como resultado un producto o un servicio al que llamaremos *items* cuando hablemos de *workflow*. Estos *items* no tienen que ser objetos tangibles, como por ejemplo, una demanda judicial, es un objeto no tangible que proporciona un *workflow*.

Cada *item* implica un proceso (aunque también lo podemos encontrar como procedimiento) para su realización. Cada uno de éstos, se divide en un grupo de tareas o *tasks* que forman la unidad básica de un trabajo que hay que llevar a cabo, por tanto, una tarea es un proceso que no puede ser subdividido.

Un proceso de negocio, a parte de las tareas a realizar, necesita tener en cuenta todos los recursos que se requieren para hacer un trabajo, ya que la mayoría necesitan de interacción humana.

Los patrones de *workflow* son capaces de interpretar la definición del proceso, interactuar con los participantes e invocar el uso de herramientas y aplicaciones si fuere necesario. Son básicos en el modelado de los procesos de negocio para el enrutamiento de las tareas y establecen un criterio para evaluar la expresividad de los lenguajes de modelación, ya que sería un requisito importante que las notaciones de las herramientas de modelado tengan soporte de patrones de *workflow*.

Las principales características de éstos y por las cuales son tan importantes en el modelado y la simulación de procesos son las siguientes:

- Capacidad de representar roles y asignarlos a tareas.
- Capacidad para especificar las características de los procesos.
- Capacidad para especificar atributos que nos permiten gestionar los procesos (planificación, monitorización, control).

Los profesores *Arthur ter Hofstede* y *Wil van der Aalst* [5] son dos de los principales y más importantes investigadores y conocedores de todo lo referido a patrones de *workflow*. El trabajo de la librería SIGHOS y, en su defecto, este proyecto, se basa en las especificaciones y definiciones de los patrones de workflow que aportan estos investigadores [6].

Una de sus aportaciones más importantes son las perspectivas [7] que definen el comportamiento de los flujos. Un esquema básico de ello se muestra en la **tabla 2.1**.

| **Perspectiva** | **Descripción** |
|---|---|
| Control | Representa las actividades y su orden de ejecución. |
| Datos | Datos (Documentos, objetos…) que fluyen entre actividades. |
| Recursos | Intervención de personas o dispositivos en una tarea. |
| Operaciones | Acciones elementales que se realizan en las actividades (por ejemplo, invocar un servicio con ciertos datos). |

*Tabla 1: 2.1. Perspectivas de Workflows*

## 2.2 Herramientas de modelado y simulación

Teniendo en cuenta nuestro objetivo, es importante que las herramientas de modelado que se usen tengan una notación que de soporte a los patrones de *Workflow*. Existen diversos patrones de control de flujo, sería ideal encontrar una notación que los soporte todos, pero es muy complejo, así que normalmente se analizan para que tengan soporte para la mayoría de ellos o al menos para los patrones básicos.

Existen diversas herramientas de modelado que hacen uso de diferentes notaciones. Las notaciones más extendidas y conocidas son **BPMN** y **UML**, pero existen más como **BPEL** o **XPDL**. Se debe analizar el soporte que dan a *Workflow* para saber cuál es más adecuada para nuestro objetivo.

Lo mismo pasa con las herramientas de simulación, en nuestro caso este no es un tema de discusión, pues este proyecto se centra en hacer uso de una herramienta diseñada a través de la librería de simulación de eventos discretos **SIGHOS**. Ésta tiene una relación directa con patrones de *workflow* ya que se basa en sus fundamentos, por tanto no tendremos problemas de compatibilidad por su parte.

Existen diversos artículos y estudios que analizan el soporte de *workflow* que ofrecen algunas de las notaciones de modelado más extendidas. Entre estos, un análisis de la notación BPMN [8], que nos ofrece una evaluación comparativa de cuatro lenguajes de modelado diferentes divididos por el soporte a patrones de *workflow* en tres de las perspectivas mencionadas con anterioridad (control, datos y recursos).

A parte de esto, el artículo también analiza y describe cada uno de los patrones de *workflow* definidos en cada perspectiva.

Aunque para este proyecto se han analizado los resultados de cada una de las perspectivas y lenguajes, nosotros nos centraremos en la perspectiva de control, que es la única relevante para este estudio.

Las notaciones analizadas son BPMN, UML2.0, BPEL y Oracle BPEL PM, puestas en ese orden en las tablas de resultados del artículo. Los resultados los vemos en la tabla de la **figura 2.1**.

| | 1 | 2 | 3 | 4 | | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|---|---|---|---|
| Basic Control–flow | | | | | 11. Implicit Termination | + | + | + | + |
| 1. Sequence | + | + | + | + | Multiple Instances Patterns | | | | |
| 2. Parallel Split | + | + | + | + | 12. MI without Synchronization | + | + | + | + |
| 3. Synchronisation | + | + | + | + | 13. MI with a priori Design Time Knowledge | + | + | + | + |
| 4. Exclusive Choice | + | + | + | + | 14. MI with a priori Runtime Knowledge | + | + | – | + |
| 5. Simple Merge | + | + | + | + | 15. MI without a priori Runtime Knowledge | – | – | – | +/– |
| Advanced Synchronisation | | | | | State-Based Patterns | | | | |
| 6. Multiple Choice | + | + | + | + | 16. Deferred Choice | + | + | + | + |
| 7. Synchronising Merge | +/– | – | + | + | 17. Interleaved Parallel Routing | +/– | – | +/– | – |
| 8. Multiple Merge | + | + | – | – | 18. Milestone | – | – | – | +/– |
| 9. Discriminator | +/– | + | – | – | Cancellation Patterns | | | | |
| Structural Patterns | | | | | 19. Cancel Activity | + | + | + | +/– |
| 10. Arbitrary Cycles | + | + | – | – | 20. Cancel Case | + | + | + | + |

*Figura 2: 2.1. Soporte Workflow - Perspectiva de control.* Fuente: [8]

*1- BPMN, 2- UML 2.0, 3- BPEL, 4- Oracle BPEL PM*
*+: Soporte completo, -: No soporta, +/-: Soporte Parcial*

En el **Capítulo 4** de este documento se analizarán más detenidamente estos resultados. En este sección solo nos interesa saber que existen análisis de compatibilidad de distintos lenguajes de modelado con respecto a los patrones de *workflow* y que se han utilizado para llevar a cabo nuestro propio estudio y realizar un proyecto propio.

## 2.3 Librería de simulación de eventos discretos SIGHOS

La simulación de eventos discretos [9] es un tipo de simulación muy utilizado en el ámbito empresarial debido a su utilidad para apoyar la toma de decisiones relacionadas con la planificación de producción e inventarios entre otras cosas.

Un sistema de eventos discretos se corresponde con un sistema en el cual los eventos cambian en instantes espaciados de tiempo, es decir, no están en continuo cambio.

Dada la estructura de este tipo de simulación se pueden obtener todo tipo de estadísticas sobre las operaciones modeladas, como diagramas de gantt de los recursos utilizados entre otros.

La simulación de eventos discretos puede ser utilizada en diferentes niveles tecnológicos:

- A través de librerías en algún lenguaje como C++ o java.
- A través de software especializado de simulación. Esta forma es menos flexible, por ello es también menos utilizada.
- A través de una mezcla de ambas con software de simulación multipropósito.

Los modelos de simulación de eventos discretos permiten la interacción entre los elementos, modelando así relaciones entre los distintos actores. Esta es una característica importante, ya que es también la característica principal de los patrones de workflow.

SIGHOS o PSIGHOS [10] es un proyecto de simulador de eventos discretos desarrollado por el profesor y tutor de este TFG D. Iván Castilla en su tesis doctoral [11] y por otros componentes de la Universidad de La Laguna que han seguido con su desarrollo a lo largo del tiempo.

SIGHOS es una librería desarrollada en java y contiene las funciones necesarias para realizar una simulación completa con todos sus componentes, incluido un gestor temporal. Por tanto para hacer uso de la herramienta se deben tener conocimientos de programación en java y también, sería interesante, tener algunas nociones sobre la simulación de eventos discretos para comprenderlo mejor.

Los componentes básicos que se manejan son los siguientes:

- Elementos: Son las entidades que interaccionan con el sistema, dan inicio a las actividades y usan los recursos. Representan los actores principales de la simulación y pueden ser de diversos tipos.

- Recursos: Diseñados dos tipos de recurso en función del tiempo de disponibilidad; los básicos que están disponibles el 100% del tiempo y los activos que se rigen mediante un horario, por ejemplo una consulta médica o el aula de un colegio.

- Actividades: Conjunto de tareas que llevan a cabo los elementos del sistema.

- Grupo de trabajo: Representan los recursos con su tipo y cantidad que son necesarios para desarrollar cada actividad.

- Flujo de trabajo: Conjunto de tareas a las que se le aplica una serie de patrones de control (Patrones de flujo de trabajo o Workflow Patterns).

Actualmente SIGHOS es una librería en desarrollo que permite realizar simulaciones orientadas a procesos basadas en las definiciones de patrones de *Workflow*. Para ello se hace uso del lenguaje de programación java, lo que permite flexibilidad en las simulaciones.

No dispone aún de interfaz gráfica que facilite el uso al usuario, aunque se han abierto algunos proyectos para ello.

SIGHOS hace uso de varios elementos en su código para representar las características de los modelos en la simulación. En el **apartado 2.3** hemos mencionado algunos de ellos, pero hay más componentes relevantes que se referencian en el **anexo 1** de la memoria.

# Capítulo 3
# Objetivos y fases

## 3.1 Objetivos

La idea principal de este proyecto es lograr convertir de un modelo con un lenguaje específico, a patrones de *Workflow*, y simular estos mediante la librería SIGHOS. Para ello se deben llevar a cabo diferentes hitos que se describen en el **apartado de fases**.

## 3.2 Fases

1. **Documentación**: Adquirir los conocimientos necesarios y reunir información importante. Principalmente sobre los distintos modelos de simulación, más concretamente, el modelo de simulación de eventos discretos; los métodos de simulación orientada al proceso; los patrones de workflow; y los distintos lenguajes y notaciones de modelado de procesos.

2. **Análisis y estudios previos**: Antes de comenzar debemos analizar las herramientas a utilizar. Debemos saber cómo está implementado y cómo de comporta el simulador SIGHOS para luego poder escoger la herramienta de implementación adecuada al mismo.

3. **Estudio de compatibilidad y soporte**:

    3.1. **Lenguaje de modelado**. Encontrar un lenguaje de modelado que tenga soporte de workflow mediante estudios de compatibilidad.
    3.2. **Herramienta de implementación.** Conseguir una herramienta que utilice dicho lenguaje y se adapte a las necesidades y los requisitos que necesitamos.

4. **Implementación**:

    4.1. **Plantear una relación 1 a 1** entre los patrones de workflow y dicho lenguaje.
    4.2. **Implementar las funciones necesarias** para convertir un modelo en la notación elegida en un modelo simulable con SIGHOS.

5. **Integración con SIGHOS**: Comprobar el funcionamiento mediante la implementación de un caso de estudio.

6. **Testeo y análisis de resultados**.

# Capítulo 4
# Tecnologías

En este capítulo se expondrán las diferentes tecnologías utilizadas durante el desarrollo del proyecto teniendo en cuenta una serie de requisitos y análisis realizados como parte de las tres primeras fases del proyecto, expuestas en la **sección 3** del informe.

## 4.1 Requisitos del proyecto.

Antes de comenzar con el desarrollo y la programación, se ha realizado un estudio de los requisitos que debe cumplir el proyecto.

1. La herramienta de simulación a usar es SIGHOS, y está basada en patrones *workflow*.
2. La herramienta de modelado es de libre elección bajo una serie de condiciones:
    - Uso de un estándar de modelado basado en *workflow* o que tenga soporte para e mismo.
    - Preferiblemente que sea una herramienta libre.
    - Se debe poder obtener un fichero de datos legibles de esa herramienta para poder analizarlo.

3. Como este proyecto, es la continuación de otros, el lenguaje a usar en la implementación debe ser, preferiblemente, el mismo que se utiliza en el resto del proyecto.

4. Los elementos de programación utilizados deben tener una licencia que permita su uso y modificación.

Teniendo en cuenta estos requisitos, se ha llevado a cabo un estudio del uso de diferentes lenguajes de implementación, estándares de modelado y herramientas de modelado previas al desarrollo. Los resultados resumidos de esta investigación se encuentran a lo largo de este capítulo.

También se han investigado diferentes metodologías de implementación usando códigos libres, librerías o cualquier recurso que esté a disposición vía internet. No se presentan todos los detalles, pero si las opciones más factibles al final del capítulo.

## 4.2 Lenguaje de modelado.

En la sección de antecedentes que tenemos disponible en el **capítulo 2** de esta memoria, se hace referencia a los resultados de un análisis de soporte de patrones de workflow. Nos centraremos en los resultados de soporte para la perspectiva de control, ya que, por el momento, es la única que vamos a integrar en SIGHOS.

Este análisis se realiza sobre cuatro lenguajes de modelado diferentes, que podemos ver en la **tabla 4.1**, BPMN [12], UML 2.0 [13], BPEL [14] y Oracle BPEL PM [15].

| **Lenguaje** | **Descripción** | **Ventajas** | **Desventajas** |
|---|---|---|---|
| BPMN | Es una notación gráfica que permite representar únicamente procesos de negocio. | 1) Es totalmente compatible con los elementos básicos de Workflow.<br>2) Es sencillo de representar e interpretar.<br>3) Existen variedad de herramientas libres que permiten desarrollar modelos con este lenguaje. | 1) No tiene mucha compatibilidad con la perspectiva de recursos de workflow.<br>2) Sólo representa procesos de negocio, no tiene soporte para otros tipos de proceso. |
| UML 2.0 | Es un lenguaje gráfico que unifica varias técnicas de modelado en una sola. | 1) Es casi totalmente compatible con los elementos básicos de Workflow.<br>2) Puede realizar diversos tipos de diagrama, entre ellos los procesos de negocio. | 1) No tiene mucha compatibilidad con la perspectiva de recursos de workflow.<br>2) Es un poco más complejo que BPMN.<br>3) No está especialmente diseñado para los procesos de negocio. |

| Lenguaje | Descripción | Ventajas | Desventajas |
|---|---|---|---|
| BPEL | Lenguaje estandarizado para la descripción de servicios web. Está basado en XML. | 1) Tiene tanta compatibilidad como UML.<br><br>2) Está diseñado para definir procesos de negocio y manipularlos.<br>3) Lenguaje en alto nivel, en el que se puede programar diversas estructuras. | 1) Es bastante más complejo que los lenguajes anteriores.<br><br>2) No tiene una interfaz gráfica, aunque es compatible con BPMN.<br>3) No tiene ninguna compatibilidad para soporte de recursos. |
| Oracle BPEL Process Manager | Ofrece una infraestructura completa para implementar y administrar procesos de negocio BPEL. | 1) Incorpora compatibilidad para la perspectiva de recursos.<br>2) Tiene tanta compatibilidad para workflow como UML.<br>3) Es sencillo de utilizar. | 1) Se centra más en los recursos humanos que en el proceso en sí.<br>2) Es sencillo de utilizar pero no de instalar, tienen que realizarse varias instalaciones y actualizaciones.<br>3) Está pensado para usuarios con conocimientos avanzados. |

*Tabla 2: 4.1. Lenguajes de modelado.*

Considerando las ventajas y desventajas de cada lenguaje, se ha escogido utilizar **BPMN**, debido a que es el más compatible con los patrones de workflow, es sencillo de utilizar y un lenguaje gráfico.

Aunque no tenga soporte para recursos, y en este caso, Oracle BPEL PM sea más adecuado, éste se centra en los recursos humanos, y lo que necesitamos es un lenguaje que nos permita luego simular teniendo en cuenta todas las variables, no solo los recursos humanos.

Por el momento, no necesitamos tener en cuenta la representación de los recursos, pero podemos estudiar diversas opciones para incluirlos de alguna manera. Es por ello que se ha escogido utilizar BPMN en vez de oracle BPEL PM.

## 4.3 Herramientas de modelado y simulación.

Existen varias herramientas de modelado que utilizan notación BPMN, hemos tenido en cuenta varias de ellas; BPMN-JS [16], Bonita Studio [17], Adonis [18], Eclipse 2.0 [19] y Microsoft Visio [20]. Podemos verlas en la **tabla 4.2**.

| **Herramienta** | **Descripción** | **Ventajas** | **Desventajas** |
|---|---|---|---|
| BPMN-JS | Es una herramienta web para el modelado BPMN usando javascript como lenguaje base de la herramienta. | 1) Es una herramienta web.<br>2) Es "Open Source".<br>3) Permite exportación XML.<br>4) Es compatible con todos los navegadores web.<br>5) Interfaz llamativa y con varias características de diseño. | 1) No es tan sencilla de utilizar.<br>2) Aunque permita exportación XML, no contiene una sintaxis sencilla para traducir y ensamblar con SIGHOS. |
| Adonis | Es una herramienta que permite el desarrollo de modelos con BPMN, basada en el diseño de procesos actuales (AS-IS) y futuros (TO-BE) | 1) Herramienta sencilla e intuitiva.<br>2) Totalmente personalizable.<br>3) Basada en web.<br>4) Está orientada a objetos, por ello ofrece infinidad de características y posibilidades de diseño.<br>5) Previsión de Recursos Humanos implementada. | 1) Es una herramienta de pago, muy cara.<br>2) No tiene versión gratuita limitada, solo una versión de prueba de 30 días que te limita bastante la actuación. |

| Herramienta | Descripción | Ventajas | Desventajas |
|---|---|---|---|
| Bonita Studio | Es una herramienta que permite diseñar modelos en BPMN a medida. Entorno basado en Eclipse. | 1) Hay diseños reciclables en su página web. 2) Es una herramienta "Open Source". 3) Tiene una interfaz web sencilla de utilizar. 4) Tiene una versión gratuita limitada, pero aún así permite desarrollar todo lo mencionado. 5) La versión Pro, permite todo lo mencionado y aún más características. 6) Basada en Eclipse, por tanto será compatible. | 1) A pesar de tener la interfaz web, es necesaria una instalación de escritorio, la cual es bastante pesada. 2) No resulta factible utilizar su versión gratuita para nuestro propósito, ya que no podríamos tener todo gestionado por la misma herramienta. Aún así, se podría tener en consideración usarla para modelar y luego enlazar el modelo. |
| BPMN 2.0 (Eclipse) | Es una herramienta de modelado para eclipse. Plugin de instalación que permite abrir una interfaz de diseño BPM. | 1) Herramienta gratuita. 2) Permite gestionar todo el proyecto en la herramienta eclipse. 3) Genera un documento XML simple, con los datos básicos necesarios. 4) Hay varios modelos reciclables en la web y varios tutoriales de uso y de bpm en general que se pueden consultar. | 1) No tiene características de personalización de interfaz. 2) No incluye recursos humanos entre los objetos de diseño posibles, aunque sí incluye "lanes" en los que pueden especificarse los actores de un conjunto de procesos. |
| Microsoft Visio | Herramienta de diseño de diagramas de varios tipos, entre ellos BPM. Exclusiva de Microsoft. | 1) Fácil de usar. 2) Permite trabajo colaborativo. 3) Se pueden obtener datos en tiempo real. 4) Tiene versión web. | 1) Es un servicio pago. 2) Es exclusiva de microsoft. 3) Para todas las características se aplica un precio mayor. 4) Contrato con permanencia de 1 año mínimo. |

*Tabla 3: 4.2. Herramientas de modelado.*

Teniendo en cuenta las consideraciones anteriores, se ha escogido utilizar el plugin de eclipse, **eclipse BPMN 2.0**, debido a que permite la gestión del proyecto completo en la misma herramienta, tanto el modelado como la simulación.

También por su facilidad de uso y sus características generales, que nos permiten realizar un modelo completo compatible con SIGHOS. Todo ello de manera gratuita, sin coste económico para el desarrollador ni el usuario final.

Al utilizar Eclipse, podemos completar diferentes datos útiles en la simulación desde la creación del modelo estático. Algunos de esos datos son los recursos humanos, materiales y temporales.

Se indaga más en las funcionalidades de esta herramienta en el **capítulo 5**.

## 4.4 Lenguaje de desarrollo e integración con el proyecto SIGHOS.

La herramienta SIGHOS necesita recibir un fichero de entrada con las especificaciones de la simulación que va a realizar. Dicho fichero debe estar desarrollado en lenguaje java para poder ser procesado por la herramienta.

El fichero de salida del modelado BPM tiene la extensión de archivo ".bpmn2", lo que significa que está desarrollado utilizando el estándar BPMN 2.0 [21] que es una evolución de BPMN e incorpora varias ventajas con respecto a otros. Éste fichero, si lo abrimos con un editor, veremos que está implementado en lenguaje de datos XML.

Para poder ensamblar ambos conceptos en una misma herramienta, modelado y simulación, se debe crear un método de conversión del fichero de salida en XML a un fichero de entrada compatible en java.

Se ha desarrollado un **conversor** entre el fichero de salida del modelado en xml y el fichero de entrada de SIGHOS, creando un pequeño programa en un lenguaje de programación orientado a objetos, para facilitar el tratamiento. Se ha escogido el lenguaje **java**, para mantener el mismo lenguaje en todas las partes de proyecto y no empezar a mezclar partes de códigos en diferentes lenguajes. Con ello se pretende facilitar el futuro tratamiento y gestión.

Esto no conlleva una integración completa, pero puede gestionarse todo desde la herramienta de Eclipse en un mismo proyecto, no se necesita tener proyectos separados para cada parte del diseño del proceso.

## 4.5 Librerías de desarrollo.

El programa comenzó a hacerse programado completamente, cuando sólo era una función. Tras ver que la implementación se volvía más compleja, se decidió hacer uso de librerías especializadas. Se investigaron varias librerías, al final nos quedamos con la que más se ajusta a nuestro proyecto.

Una de las librerías que estudiamos y, aunque no es la que utilizamos, resulta interesante para el parseo de ficheros XML en general. Se llama *DOM XML Parser* y pertenece al paquete *com.mkyong.seo* [22]. Es una librería muy simple y fácil de usar, que tiene soporte y de la cual obtenemos cada uno de los elementos y atributos de un XML de manera muy sencilla.

Descartamos esta idea porque es una librería para cualquier tipo de fichero XML y posteriormente encontramos una librería específica para ficheros XML de BPMN 2.0; *Camunda* [23].

Camunda dispone de plataformas de código abierto de flujo de trabajo y automatización de decisiones. Está desarrollada en Java y tiene licencia Apache, por lo cual podemos utilizar su código para integrarlo en el nuestro, respetando la licencia.

Puede integrarse en aplicaciones de java y de otros lenguajes mediante REST, en este caso no hace falta, ya que el desarrollo se ha decidido realizar en java. Las aplicaciones no solo están disponibles para usarse con BPMN, también pueden hacer uso de ellas con otros estándares permitidos. Las ventajas más relevantes del uso de Camunda son:

- Optimización: Es un código optimizado, empaquetado y reducido.
- Simplicidad y sencillez: Sus librerías están muy completas, por lo que casi todo lo tendremos implementado y únicamente haremos uso de su código con algunas modificaciones para poder ajustarlo al nuestro.
- Código abierto.
- Diferentes maneras de saber y controlar errores.
- Desarrollado en Java: Hemos decidido usar este lenguaje en nuestro proyecto, por lo cual es una gran ventaja haber encontrado una librería implementada en el mismo.

De las herramientas que tenemos disponibles para BPMN con Camunda, únicamente hemos hecho uso de *camunda-xml-model*, que nos proporciona un código en java para poder obtener y gestionar un fichero XML de BPMN 2.0.

# Capítulo 5
# Metodología y resultados

En esta sección se detallan los aspectos más relevantes en la implementación y la integración con la herramienta SIGHOS, siendo ambos resultados del desarrollo de las fases 4 y 5 expuestas en las fases del **capítulo 3**.

## 5.1 Requerimientos.

Para el desarrollo e implementación se hizo uso de diferentes herramientas y librerías que se deben instalar.

- Eclipse Java Oxygen [25]. Tiene que ser esta edición, debido a que el plugin que estamos utilizando no está aún disponible para versiones más modernas y no se desarrolló para todas las versiones anteriores a la misma.
- Eclipse Plugin JBPM [26]. Es un plugin de JBOSS que debe instalarse para poder incorporar en eclipse las funcionalidades de modelado gráfico en BPMN. Existe material de ayuda y tutoriales disponibles para la obtención e instalación del mismo [27].
- Librerías Camunda BPM: Librerías usadas en la implementación. Se pueden obtener todas fácilmente desde un repositorio Maven [28].

    1. *assertj-core-3.5.2*
    2. *camunda-bpmn-model-7.10.0*
    3. *camunda-example-invoice-7.10.0-javadoc*
    4. *camunda-example-wait-states-7.1.0-Final-sources*
    5. *camunda-xml-model-7.10.0*

- Librería JUnit [29] para desarrollo en Java. Hemos utilizado la versión 4.3, pero ya se encuentra disponible la versión 5.

## 5.2 Relación entre BPMN y SIGHOS.

BPMN es compatible con los patrones de *workflow* pero no al 100%, esto puede ocasionar problemas al querer representar modelos con funciones que no sean las funciones básicas. Es por ello, que no vamos a tener en cuenta la totalidad de posibilidades de esta notación, sino que se representará parte de la misma.

Se ha establecido una relación entre BPMN y workflow para poder realizar la traducción entre ambos, teniendo en cuenta los análisis de compatibilidad.

En la relación establecida encontraremos las funciones básicas del modelo, además de esto, existen diferentes normas que se deben cumplir en el diseño para poder obtener un modelo simulable por SIGHOS, si no se cumple con esta serie de reglas no se puede elaborar la simulación debido a que se necesita hacer uso de un conjunto mínimo de datos. Estas normas se describen a lo largo del capítulo.

## 5.3 Esquema de funcionamiento básico del conjunto de herramientas.

Todas las herramientas y librerías pueden integrarse en Eclipse. Desde eclipse gestionaremos la totalidad del proyecto sin necesidad de hacer uso de herramientas externas. El funcionamiento básico del conjunto se muestra en la **figura 5.1** y se puede resumir en los siguientes pasos:

Creación del modelo estático usando el plugin de eclipse siguiendo una serie de reglas de modelado necesarias para la conversión. El fichero generado con la extensión ".bpmn2" en formato XML será analizado y traducido por el programa desarrollado en este trabajo final de carrera. Del paso anterior, se obtiene un fichero java con el modelo listo para ser simulado por PSIGHOS. Simular el modelo en PSIGHOS.

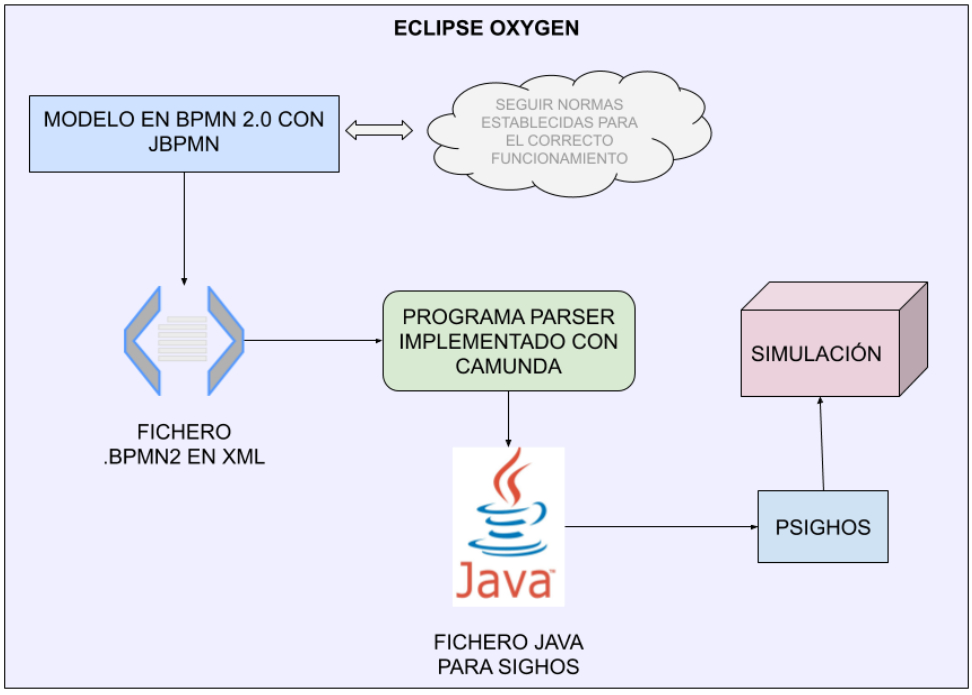


*Figura 3: 5.1. Esquema de funcionamiento básico.*

## 5.4 Implementación.

El código del programa se ha desarrollado a partir de las librerías *Camunda XML model*. Como hemos dicho, todo está encapsulado en el mismo proyecto gracias a la herramienta Eclipse. Los ficheros que necesitamos en nuestro programa de conversión son:

- Fichero con extensión ".bpmn2": Modelo estático creado con JBPMN.
- Main.java.
- Parser3.java: Tercera versión del código conversor.

### 5.4.1 Fichero .bpmn2.

Archivo en XML resultado del modelo bpmn desarrollado. Existen una serie de reglas a seguir y datos a cumplimentar para desarrollar este modelo. Deben cumplirse todas para poder elaborar el fichero de simulación, si no, no habrán datos suficientes.

1. Se puede hacer uso de todos los elementos incluidos en el estándar BPM, pero solo los elementos básicos serán traducidos. En el ejemplo de la **figura 5.2** tenemos representado un proceso con Lanes, eventos, tareas, exclusive gateways y agrupaciones. Los lanes y las agrupaciones nos ayudan en el modelo estático a distinguir mejor entre las funciones y los recursos, pero no forman parte de los elementos básicos que se han programado para la traducción, por tanto esos elementos pueden insertarse en el modelo gráfico, pero no serán traducidos.

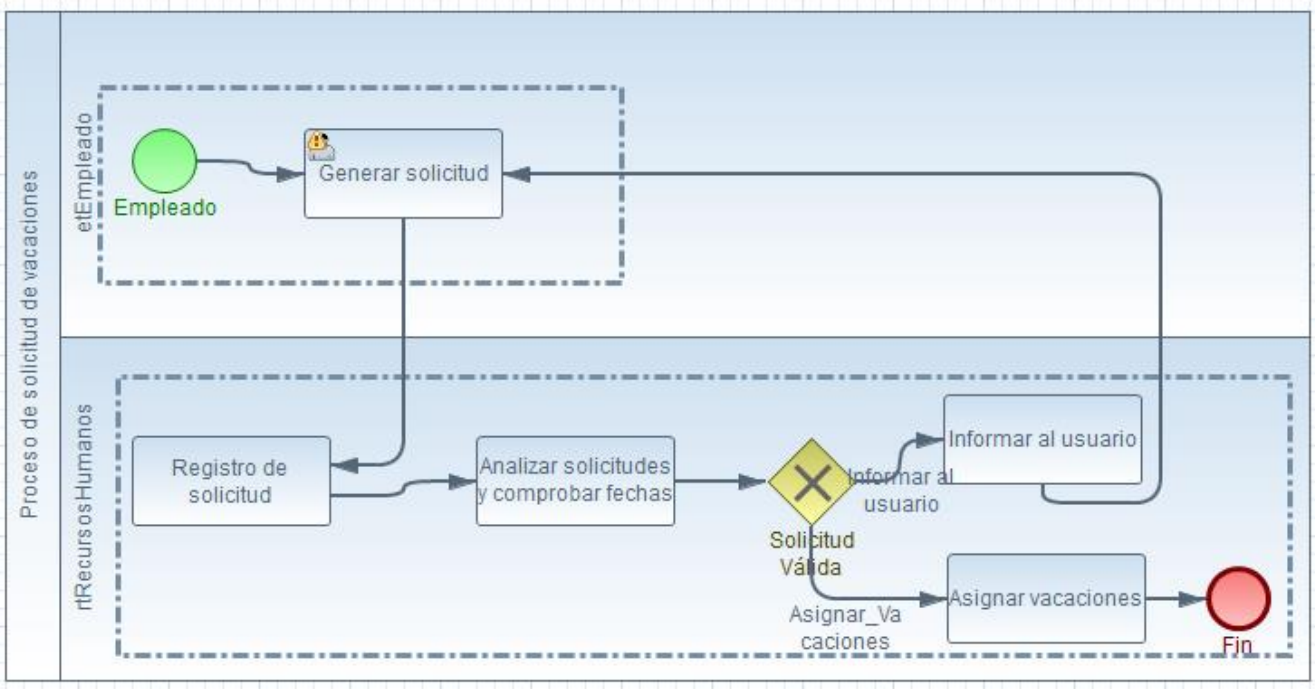


*Figura 4: 5.2. Ejemplo de proceso con diferentes elementos BPMN.*

2. Se ha determinado que el evento de inicio del proceso haga el papel del elemento que inicia el proceso en la simulación. Para el modelo simulable necesitaremos el nombre del elemento y un valor medio de elementos que se introducen en un corto periodo de tiempo. Como observamos en la **figura 5.3**, estos datos se añaden en la descripción de elemento en los atributos "Name" y "Documentation" respectivamente.

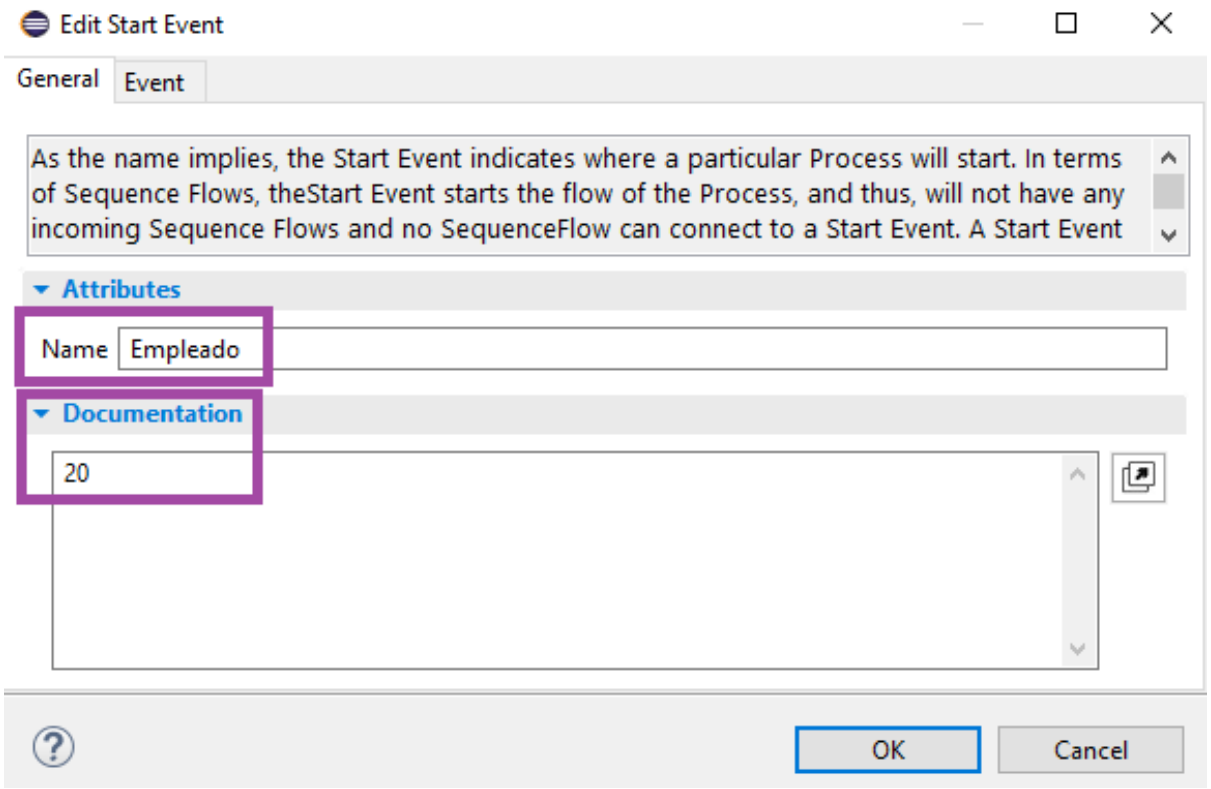


*Figura 5: 5.3. Evento de inicio del proceso.*

3. Existen varios tipos de tareas en esta herramienta (de usuario, de servicio, de scripts, genéricas..). No importa el tipo de tarea que usemos en el modelo, PSIGHOS las simula todas como si fueran iguales. Lo único que debe gestionarse en ellas, sean del tipo que sean, es que deben disponer de un nombre y de los recursos que van a utilizar en su descripción.

   Eclipse nos permite introducir los recursos en uno de sus atributos. Esto no forma parte de este proyecto, pero se ha tomado tiempo de investigación y el introducir estas metodologías es una posibilidad a tener en cuenta en un futuro. En la **figura 5.4** podemos observar que existen estas opciones.

   En nuestro caso, podemos ver en la **figura 5.5** que para mayor facilidad usaremos el atributo "Documentations" y el atributo "Name" para asignar los datos que necesitamos, introduciendo un recurso y la cantidad necesaria del mismo en el primer campo mencionado y el nombre de la actividad en el segundo.

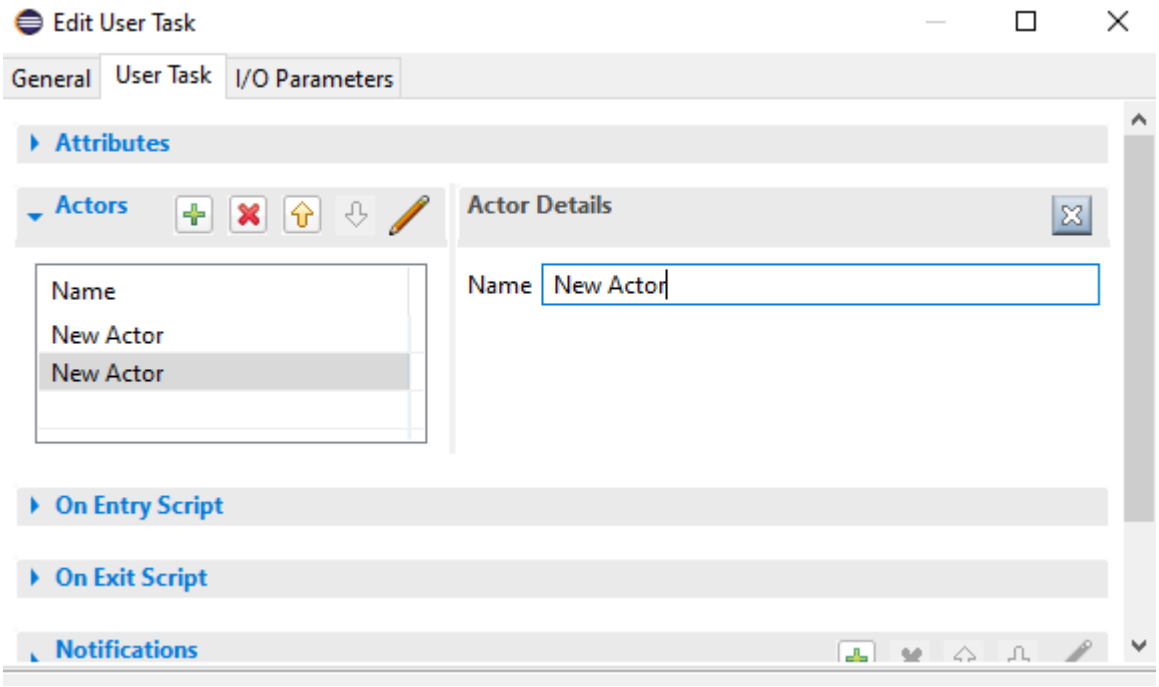


*Figura 6: 5.4. Inserción de actores.*

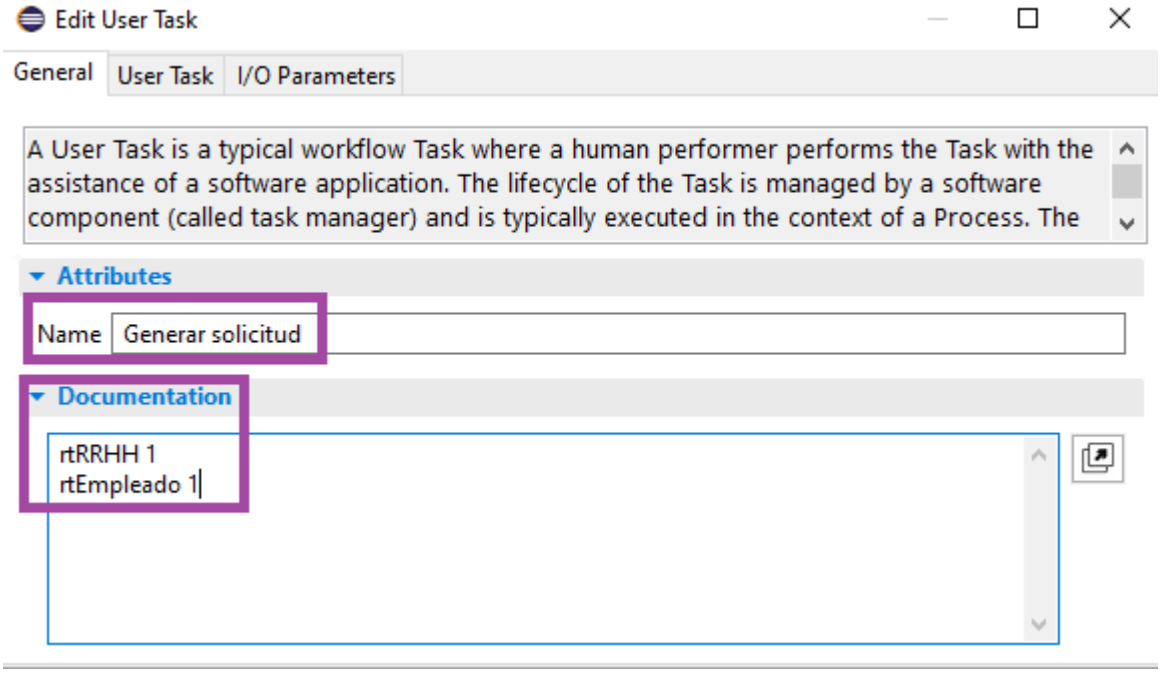


*Figura 7: 5.5. Inserción de recursos y cantidad por tarea.*

4. En la **figura 5.6**, se muestra como las exclusive gateways deben tener cumplimentado un atributo en forma de tabla denominado "Sequence Flow List" en el que se asigna a cada posible salida los siguientes datos:

    - Sequence FLow será el nombre del elemento al que da lugar.
    - Condition será el porcentaje de probabilidad de que se escoja un camino u otro.
    - Is Default es un booleano al que se le dejará asignado "False", ya que los porcentajes nos indican ya todo lo que necesitamos saber para la simulación sobre el camino a escoger

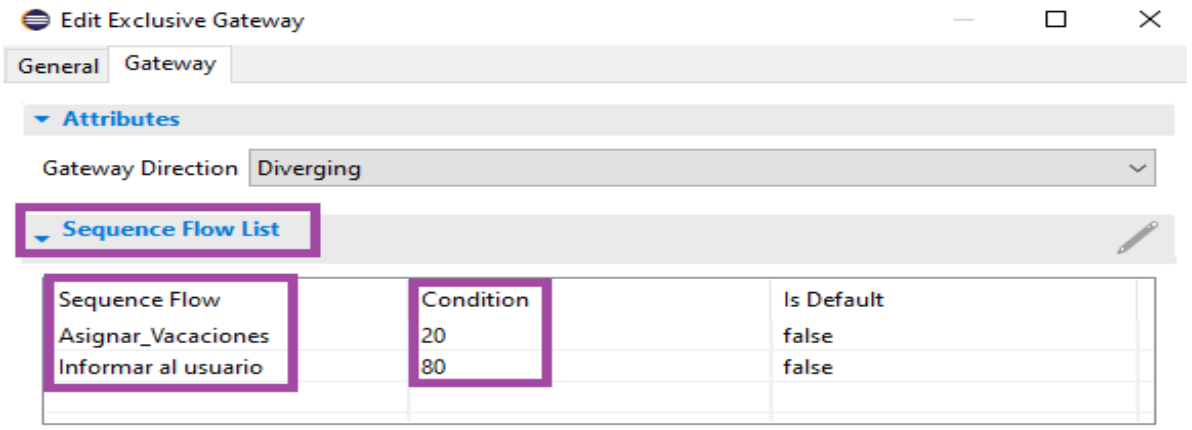


*Figura 8: 5.6. Atributos en Exclusive Gateways.*

5. Siempre debe haber al menos un evento de fin.

Deben cumplirse las cinco normas mencionadas para la simulación, además de esto, debe ser un modelo que cumple los estándares BPMN y contiene elementos básicos compatibles con patrones de workflow.

No todos los elementos de BPMN son compatibles, hay que intentar ajustarse a una serie de elementos, es estrictamente necesario que no se inserten elementos que se consideren importantes en el modelo con componentes que no sean compatibles.

El conjunto de los elementos de PSIGHOS pueden consultarse en los anexos. En el apartado de "caso de estudio" de este capítulo, se muestran ejemplos del código XML generado por el programa.

### 5.4.2 Main.java.

Crea el objeto y lo llama pasando como parámetro el fichero del modelo estático y un nombre para almacenar el fichero de salida simulable en java, tal como se muestra en el código de la **figura 5.7**.

```
1
2  public class Main {
3      static Parser3 par;
4
5      public static void main(String[] args)
6      {
7          par = new Parser3("lavanderia.bpmn2", "lavanderia.java");
8      }
9  }
10
```

*Figura 9: 5.7. Código main.*

### 5.4.3 Parser3.java.

Tercera versión del código finalizado. Se elaboraron dos versiones antes de llegar a esta. En la primera versión se comenzó con una implementación propia del conversor, en cuanto comenzó a complicarse la implementación surgió una nueva versión ya utilizando la librería de camunda, la cual se escogió después de un análisis de sus pros y contras.

La versión tres es una mejora de la versión dos, en la cual se eliminan los elementos innecesarios del código, se corrigen algunos errores y se añaden nuevas funcionalidades. En la **figura 5.8** se muestran las librerías importadas y las funciones que se han implementado.

Se hace uso de librerías java de gestión de ficheros, de tipos de datos y de validación de expresiones regulares; también se importan las librerías de camunda que estamos usando, aunque los ejemplos en el repositorio github de camunda hacen uso de varias librerías más.

Algunas variables se definen de manera global para facilitar la gestión, pero la mayoría son variables locales que no existen fuera de las funciones que las usan. La más importante es la declarada como "*protected BpmnModelInstance modelInstance*", que es la que nos da acceso al modelo BPM y las funciones de la librería camunda, a través de ésta accederemos a todo lo que necesitamos del modelo estático.

El código completo se encuentra en el **anexo 2**. En este capítulo se hará referencia a ciertas partes interesantes del mismo, pero no al código en su totalidad.

```
1  import java.io.File;
2  import java.io.FileWriter;
3  import java.io.IOException;
4  import java.util.ArrayList;
5  import java.util.Collection;
6  import java.util.Scanner;
7  import java.util.regex.Matcher;
8  import java.util.regex.Pattern;
9
10 import org.camunda.bpm.model.bpmn.Bpmn;
11 import org.camunda.bpm.model.bpmn.BpmnModelInstance;
12 //import org.camunda.bpm.model.bpmn.impl.BpmnModelConstants;
13 import org.camunda.bpm.model.bpmn.instance.ConditionExpression;
14 import org.camunda.bpm.model.bpmn.instance.Documentation;
15 import org.camunda.bpm.model.bpmn.instance.FlowNode;
16 import org.camunda.bpm.model.bpmn.instance.SequenceFlow;
17 /* These libraries are used in functions ..
```

```
37 public class Parser3 {
38     //Input Bpmn2 File
39     File input;
40     //Output Java File
41     FileWriter output;
42     protected BpmnModelInstance modelInstance;
43     protected String links ="";
44     protected String nEt, ActnEt = "";
45
46     //Constructor function
47     Parser3(String file_, String outputFile)..
78
79     //Function that loads the initial part of the java file: Packages, libraries, etc
80     private void preCharge(String model)..
119
120    //Search elements by their type
121    private void findByType() ..
205
206    //Get all sequence flows in the model
207    private void getSequenceFlows(FlowNode e, String name)..
249
250    private String getNameById(String id)..
270
271    //Create WorkGroups
272    private void createWorkgroup(FlowNode e, String name)..
342
343    //Close files
344    private void closeFiles()..
355 }
356
```

*Figura 10: 5.8. Librerías, variables globales y funciones.*

Centrándose en el funcionamiento interno del programa, el main llama al constructor y a partir de aquí se ejecutan una serie de funciones hasta obtener la salida deseada:

1. En el **constructor**, a parte de la creación del objeto, se ha dispuesto una sección de código que permite introducir por teclado el nombre del modelo de simulación.

2. Desde éste se llama a la **función de precarga** que se encarga de cargar en el fichero java de salida el código de importación de las librerías necesarias para la simulación y de crear la clase con el nombre de modelo suministrado.

3. Al ejecutarse la función de precarga, se retorna al constructor y éste llama a la **función de principal interés, "*findByType()*"**. En ella se recorre una colección con todos los nodos del modelo de la manera que se muestra en el código de la **figura 5.9**. Se utilizan expresiones regulares para comparar con los id obtenidos de los nodos del modelo. Dependiendo del tipo de nodo que estemos analizando el programa realizará una acción u otra:

    - Start Event: Para este caso Se obtiene el nombre y la documentación y se genera en el fichero java una entrada de código en la que se crea un elemento que inicia el proceso, denominado "ElementType".
    - Task: Las tareas pueden ser de varios tipos, pero todas se convertirán en "ActivityFlow" al pasarlas al modelo de simulación, por tanto todas serán tratadas del mismo modo. De éstas se obtiene el nombre de la tarea y luego se llama a la función "*createWorkGroup(FlowNode , String )*" a la que se le pasa el nodo de la tarea y el nombre de la actividad.
    - Exclusive Gateways: Son los nodos más complejos de esta función, de ellos hay que obtener el nombre, cada uno de las secuencias que puede seguir y las condiciones con los porcentajes de que se siga cada una de esas secuencias. Para obtener las secuencias se hace uso de la función "*getSequenceFlows(*FlowNode , String*)*" y para las condiciones se debe usar una colección de secuencias que se obtienen del nodo en cuestión. En la **figura 5.10** se presenta el código utilizado para ello y la forma que se ha utilizado para obtener las condiciones.

```
//get all flow nodes in the model
Collection<FlowNode> flowNodes = modelInstance.getModelElementsByType(FlowNode.class);

for(FlowNode e : flowNodes)
{
```

*Figura 11: 5.9. Colección de nodos del modelo.*

```
Collection<SequenceFlow> list = e.getOutgoing();
for (SequenceFlow seq : list) {
    //There should always be one or more conditions.
    ConditionExpression cond = seq.getConditionExpression();
    if(cond != null) {
        percentageName = seq.getName();
        percentageName = percentageName.replaceAll(" ", "_");

     }
```

*Figura 12: 5.10. Colección de secuencias y sus condiciones.*

4. La **función "*createWorkGroup(FlowNode , String )*"** accede a la documentación de las tareas, obtiene los datos de los recursos que necesita para la simulación y luego los traduce y escribe en el fichero java de salida. En la **figura 5.11** se observa que para acceder a la documentación se utiliza una colección de ese tipo de dato.

```
ArrayList<String> wg = new ArrayList<String>();
Collection <Documentation> doc = e.getDocumentations();
for(Documentation d:doc)
```

*Figura 13: 5.11. Acceso al atributo de documentación.*

5. La **función "*getSequenceFlows(*FlowNode , String*)*"** es la segunda más importante de la implementación. En ella se obtienen los enlaces de cada uno de los nodos con el nodo siguiente y el anterior en caso de que los tuviera. Se realiza de forma similar que en la función de búsqueda por tipo. Se utiliza una colección, esta vez de secuencias no de nodos, y se recorre obteniendo los valores de los atributos que relacionan cada nodo con su nodo sucesor y predecesor; éstos son "targetRef" y "sourceRef" respectivamente. Se analiza cada sucesor y predecesor usando expresiones regulares para averiguar el tipo y añadirlo al fichero de salida con el tipo y nombre correcto. En la **figura 5.12** se ha dispuesto una porción de código donde se ven estos detalles entre otros.

   De los atributos obtenemos el id del nodo, pero en la simulación se hace uso del nombre, así que llamamos a la **función "*getNameById(String )*"** a la que le pasamos el id y nos retorna el nombre.

   Además, se ha dispuesto una variable para controlar que en los nodos de tipo Exclusive Gateway se le establezca la conexión insertando también los porcentajes de las condiciones, en la **figura 5.12** se muestra que esta variable únicamente se utiliza para saber si escribe la condición o no.

6. Por último, el constructor llama a la **función "*closeFiles( )*"** para cerrar el fichero java que estaba generando.

   Se ha de dejar constancia que la librería camunda en uso tiene gran variedad de clases y funciones que no han sido utilizadas en este proyecto, pero que pueden resultar interesantes.

   Es una librería extensa, de la cual solo se ha hecho uso de lo necesario para atender a las necesidades del proyecto en cuestión.

```
//Get all sequence flows in the model
private void getSequenceFlows(FlowNode e, String name)
{
    String id, source, target = "";
    //Exclusive Gateway?
    Boolean eg_ = false;

    Collection<SequenceFlow> sequenceFlows = modelInstance.getModelElementsByType(SequenceFlow.class);
    for(SequenceFlow sf : sequenceFlows)
    {
        SequenceFlow flow = modelInstance.getModelElementById(sf.getId());
        id = flow.getAttributeValue("targetRef");
        target = getNameById(id);
                ...
        id = flow.getAttributeValue("sourceRef");
        source = getNameById(id);
                ...
        pat = Pattern.compile("ExclusiveGateway.*");
        mat = pat.matcher(id);
        if (mat.matches())
            eg_ = true;

        if(eg_ == true)
        {
            links = links + ("      " + source + ".link(" + target + ", " + target + "Condition);\r\n");
            eg_ = false;
        }
        else
            links = links + ("      " + source + ".link(" + target + ");\r\n");
                ...
   }
```

*Figura 14: 5.12. Obtención de nodos predecesores y sucesores.*

## 5.5 Ejemplo: Caso de Estudio de una lavandería hospitalaria.

Como descripción inicial y requisitos previos del caso se dispone del siguiente enunciado:

Se trata de una lavandería hospitalaria, la cual dispone de un sistema industrial que recibe la ropa sucia de un centro de uso (el hospital) y devuelve al mismo centro la ropa limpia. La ropa se recibe organizada en bolsas de 50kg y categorizadas por tipo:

1. Sábanas blancas; 2. Colchas; 3. Toallas; 4. Metidas; 5. Pijamas; 6. Mantas; 7. Uniformes

Se requiere observar en un turno de 8 horas de trabajo. La ropa que entra en el sistema se lava, suponemos que llegan unos 12 sacos cada media hora, en cada bolsa sólo se recibe ropa del mismo tipo. Al llegar la ropa, operarios distribuyen los sacos en varias cintas de transporte en el almacén. En estas cintas, se llevan las bolsas hasta la zona de lavado-secado, donde otro operario va descargando las bolsas de las cintas y metiéndolas en carros.

La proporción de prendas de cada tipo es aproximadamente: aproximadamente, 1: 20%, 2: 10%, 3: 20%, 4: 15%, 5: 10%, 6: 15%, 7:10%.

Hay un total de 5 operarios para realizar todas las tareas. Una bolsa de ropa tarda en lavarse un tiempo medio de 30 minutos. Para el secado exterior se dispone de espacio para 15 bolsas. Una vez que la ropa ha sido lavada y secada, las sábanas, colchas y metidas se llevan directamente a las planas, donde se planchan; y el resto se envía a secarse en el exterior.

Tras el secado en el exterior, pijamas y toallas se acaban en la plana; y las mantas, manualmente. Los uniformes se terminan aparte.

En cada tipo de acabado se necesita un técnico, y para el acabado en plana se requiere también una plana.

### 5.5.1 Diseño del modelo estático.

Lo primero es crear un nuevo proceso con JBPM como en la **figura 5.13**.

El comienzo del proceso es tras recibir el elemento de "ropa sucia" y el final al "Devolver ropa limpia". Se expresa que la entrada de ropa sucia será de unas 12 bolsas en un periodo corto. En la **figura 5.14** se muestra como configurar el elemento inicial con estos datos.

Se crean diferentes tareas, de los tipos que se estimen oportunos en el modelo. Se dispone de 5 operarios para realizar el total de las tareas, así que no se pueden utilizar los 5 en cada tarea. Se deben gestionar y dividir. En la **figura 5.15** se muestra la configuración de las tareas:

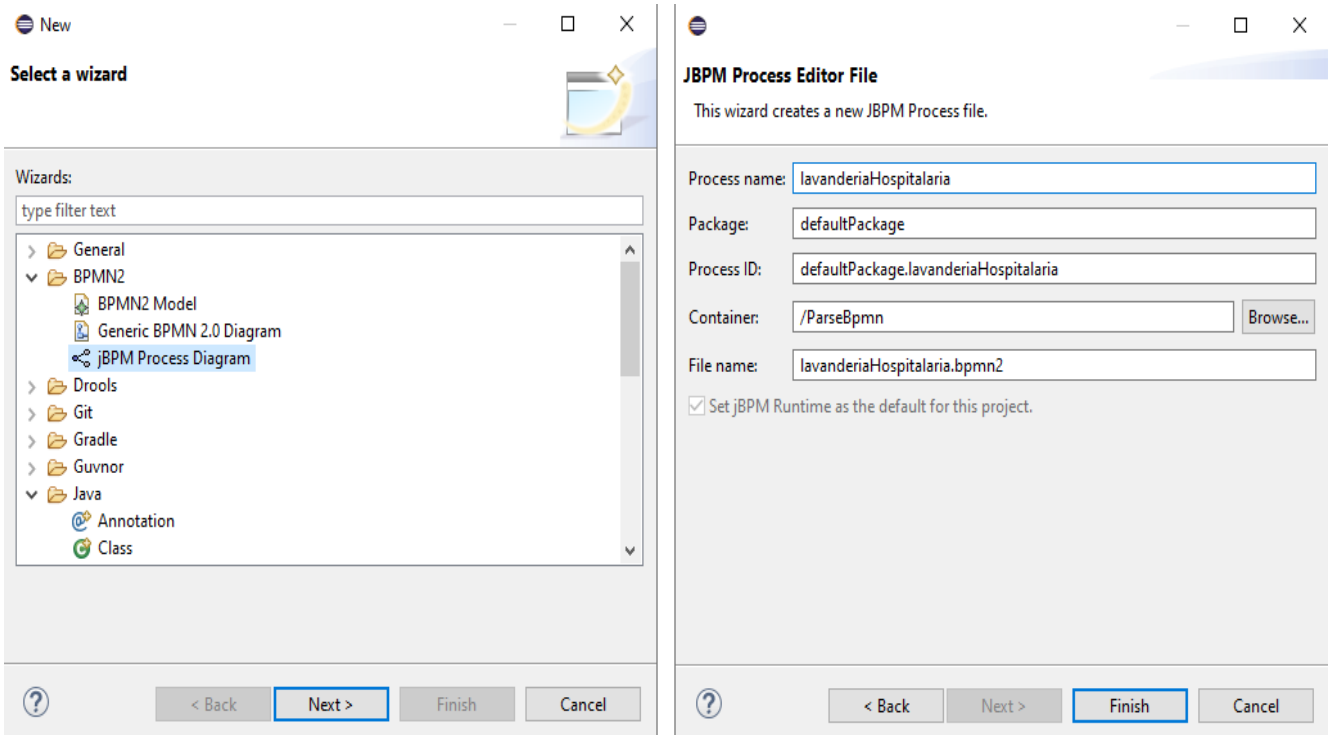


*Figura 15: 5.13. Crear nuevo proceso con JBPM.*

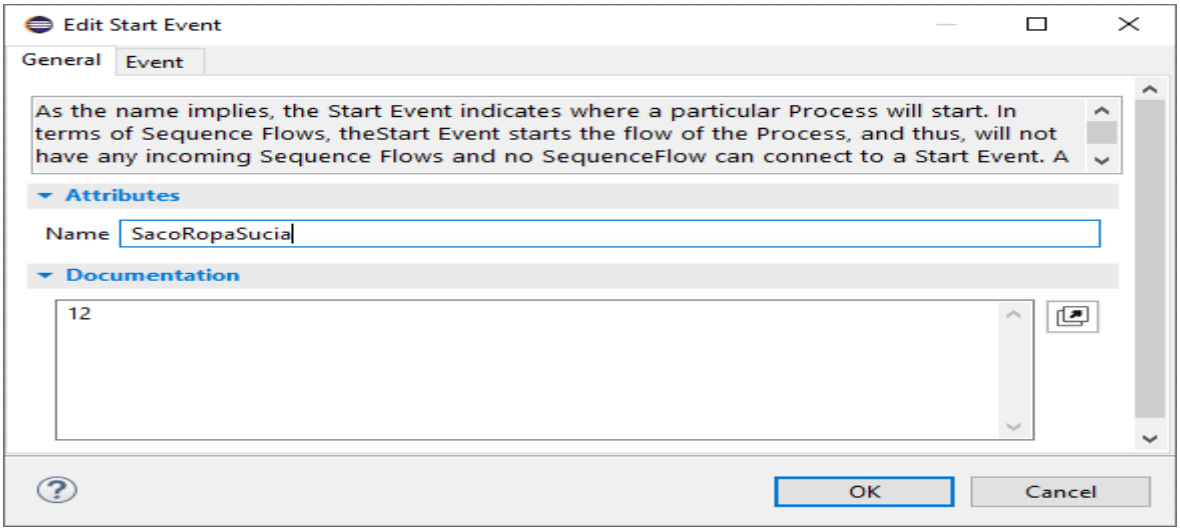


*Figura 16: 5.14. Elemento de inicio de proceso.*

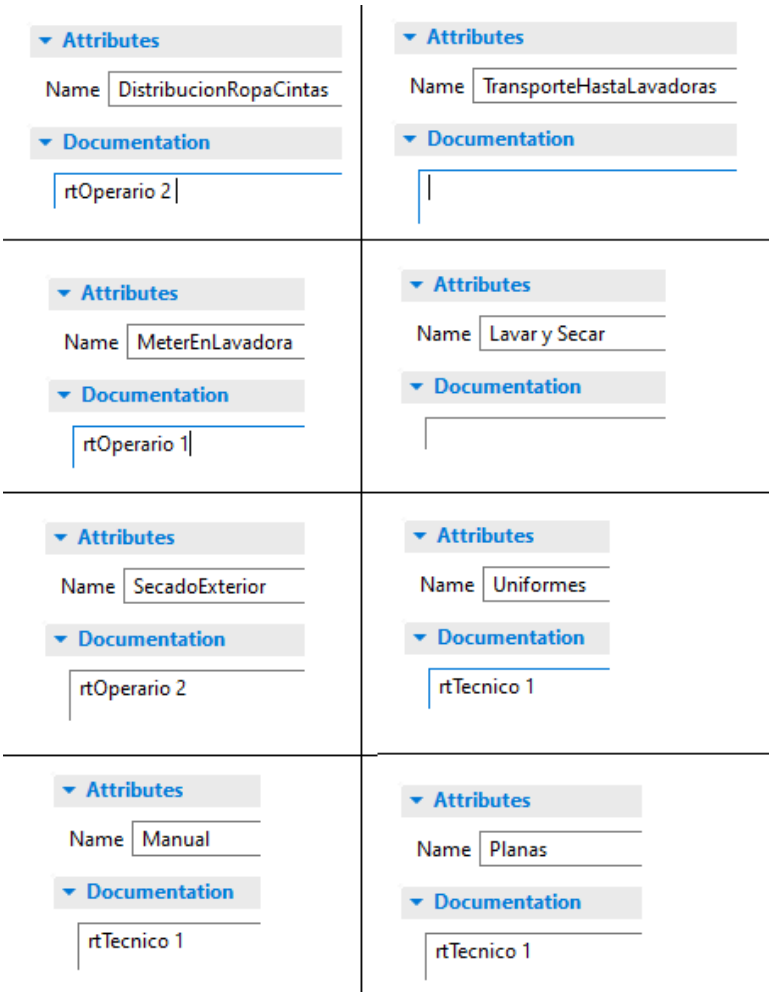


*Figura 17: 5.15. Tareas y recursos humanos.*

En último lugar, según la descripción disponemos de dos etapas en las que se debe realizar una elección; la primera dice así "Una vez que la ropa ha sido lavada y secada, las sábanas, colchas y metidas se llevan directamente a las planas, donde se planchan; y el resto se envía a secarse en el exterior." y la segunda "Tras el secado en el exterior, pijamas y toallas se acaban en la plana; y las mantas, manualmente. Los uniformes se terminan aparte.". Éstas se traducen en exclusive gateways, para poder configurarlos adecuadamente necesitamos indicarle como condición a cada secuencia de salida un porcentaje de probabilidad. En la **figura 5.16** se muestra la configuración y las condiciones de cada Exclusive Gateway.

En la descripción se nos facilita la siguiente información sobre los porcentajes de probabilidad: "1: 20%, 2: 10%, 3: 20%, 4: 15%, 5: 10%, 6: 15%, 7:10%." Por lo cual, solo debemos sumar los porcentajes de los tipos que van a usar una secuencia u otra, es decir:

- En la primera elección las sábanas (tipo 1), las colchas (tipo 2) y las metidas (tipo 4) se llevan directamente a las planas: Esta secuencia tiene una condición de 45 (20% + 10% + 15%). El resto irá a la segunda secuencia con una condición de 65.
- En la segunda elección hay tres secuencias de salida: La primera considera toallas (tipo 3) y pijamas (tipo 5) por tanto tendrá una condición de 30 (20% + 10%); La segunda considera mantas (tipo 6) que tiene una condición de 15; y la tercera y última considera los uniformes (tipo 7) con una condición de 10.

Tras disponer cada uno de los elementos obtenemos el modelo gráfico de la **figura 5.17**.

Parte del fichero generado se muestra en la **figura 5.18** a modo de ejemplo para la comprensión del código. Se ha marcado el comienzo del proceso en la figura. En ésta se muestra el evento de inicio, dos tareas y sus respectivos sequence flows.

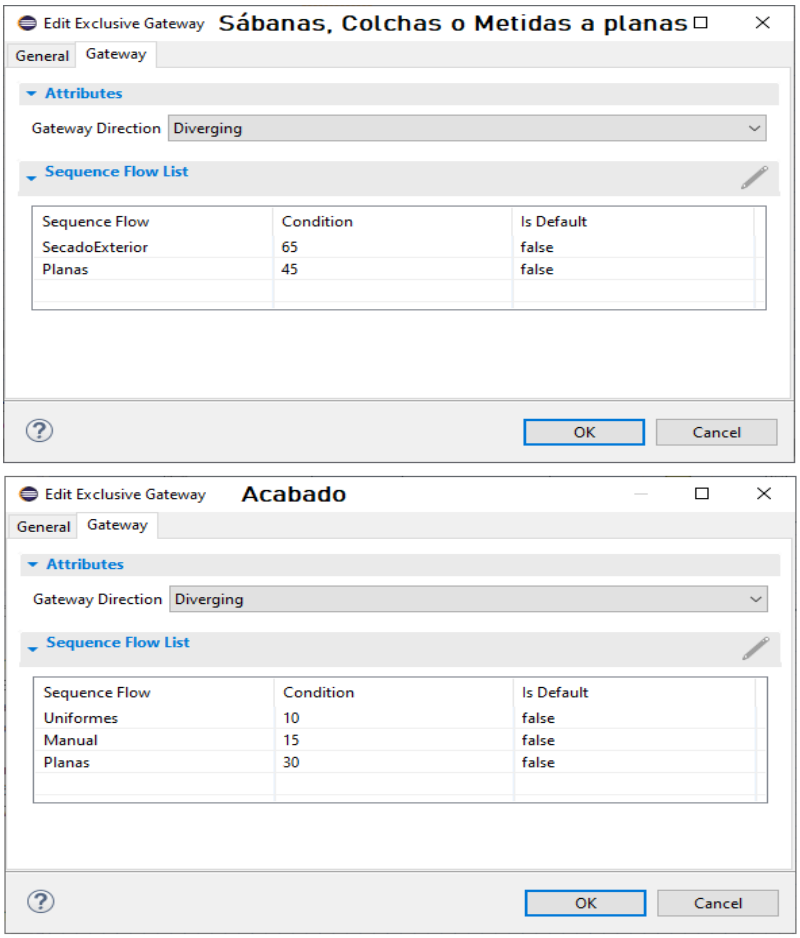


*Figura 18: 5.16. Exclusive Gateways.*

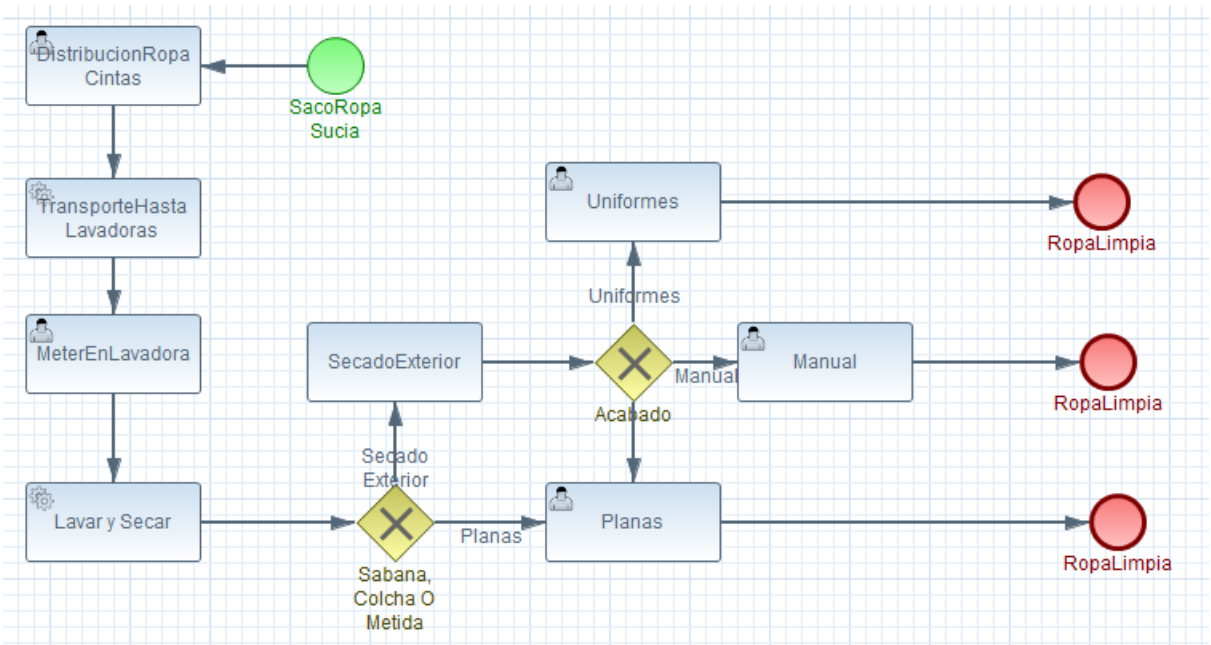


*Figura 19: 5.17. Modelo BPM.*

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- origin at X=0.0 Y=0.0 -->
<bpmn2:definitions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:bpmn2="http://www.omg.org/spec/BPMN/201(
  <bpmn2:itemDefinition id="ItemDefinition_1" isCollection="false" structureRef="java.lang.String"/>
  <bpmn2:itemDefinition id="ItemDefinition_2" isCollection="false" structureRef="java.lang.Integer"/>
  <bpmn2:itemDefinition id="ItemDefinition_3" isCollection="false" structureRef="java.lang.Boolean"/>
  <bpmn2:process id="defaultPackage.lavanderiaHospitalaria" tns:packageName="defaultPackage" name="lavanderiaHospitalar
    <bpmn2:startEvent id="StartEvent_1" name="SacoRopaSucia">
      <bpmn2:documentation id="Documentation_2"><![CDATA[12]]></bpmn2:documentation>
      <bpmn2:extensionElements>
      <bpmn2:outgoing>SequenceFlow_1</bpmn2:outgoing>
    </bpmn2:startEvent>
    <bpmn2:sequenceFlow id="SequenceFlow_1" tns:priority="1" sourceRef="StartEvent_1" targetRef="UserTask_2"/>
    <bpmn2:sequenceFlow id="SequenceFlow_3" tns:priority="1" sourceRef="UserTask_2" targetRef="ServiceTask_1"/>
    <bpmn2:serviceTask id="ServiceTask_1" name="TransporteHastaLavadoras" implementation="Java">
      <bpmn2:extensionElements>
        <tns:metaData name="elementname">
          <tns:metaValue><![CDATA[TransporteHastaLavadoras]]></tns:metaValue>
        </tns:metaData>
      </bpmn2:extensionElements>
      <bpmn2:incoming>SequenceFlow_3</bpmn2:incoming>
      <bpmn2:outgoing>SequenceFlow_6</bpmn2:outgoing>
    </bpmn2:serviceTask>
    <bpmn2:userTask id="UserTask_2" name="DistribucionRopaCintas">
      <bpmn2:documentation id="Documentation_11"><![CDATA[rtOperario 2 ]]></bpmn2:documentation>
```

*Figura 20: 5.18. Porción de código XML analizado.*

### 5.5.2 Fichero java simulable.

En la **figura 5.19** observamos que al principio de la ejecución se pide un nombre de modelo por consola, una vez se introduzca dicho dato, se genera el fichero sin problemas. El fichero generado se muestra en el **anexo 3**.

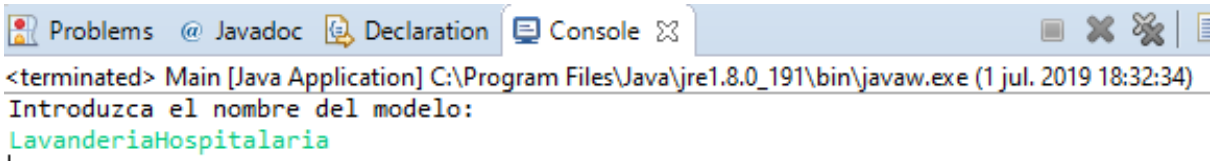


*Figura 21: 5.19. Introducir nombre del modelo.*

# Capítulo 6
# Conclusiones y líneas futuras

Con el desarrollo de este proyecto se han adquirido conocimientos sobre las herramientas y lenguajes de modelado que hay actualmente disponibles. También sobre las necesidades de las empresas y la importancia de actualizarse en un mundo de constante cambio tecnológico y la relevancia del papel de la informática en el mismo.

Como resultado, se ha logrado un programa que permite realizar una simulación en la herramienta SIGHOS partiendo de un modelo gráfico estático. Todo ello sin elaborar un modelo dinámico desde cero, que era el objetivo principal de este informe.

En este proyecto se han analizado diferentes lenguajes y herramientas. Los resultados han demostrado que, para la mayoría de casos implementados, las tecnologías elegidas eran las que más se adecuaban a nuestro objetivo cumpliendo los requisitos establecidos en el comienzo del proyecto.

Aunque se ha demostrado la validez y utilidad de este proyecto, han surgido varias opciones de desarrollo que no han sido implementadas y que podrían complementar el proyecto actual:

- **Lanes en BPM**: Los lanes son una estructura que no utilizan todos los modelos, pero se ha incorporado para una división más clara de las funciones que realizan y los recursos que utilizan los procesos representados. Actualmente, aunque se puedan utilizar en el modelo para tener una representación más clara, no son de utilidad en la simulación. Se podría estudiar el posible uso de los lanes para agrupar tareas y recursos.

- **Incorporación de los Recursos Humanos**: En principio BPMN no tiene soporte para definir o representar los recursos humanos que intervienen en las tareas. Sin embargo SIGHOS utiliza información de los mismos en la simulación. Existen alternativas para conseguir este objetivo, que han sido explicadas en el **capítulo 5** de la memoria. Sin embargo, también se ha llegado a la conclusión en estos estudios de que oracle BPEL PM sí tiene soporte para recursos humanos y además es compatible con BPMN. Una futura línea de investigación a seguir sería la opción de incorporar los recursos humanos desde oracle BPEL PM en los modelos realizados con BPMN.

- **Implementación de nuevas características**: Actualmente el código está limitado a que solo soporte las características básicas de patrones workflow. Se puede mejorar el programa incluyendo las características más complejas de workflow.

- **Incorporación de los Recursos Espaciales**: Los recursos espaciales son los espacios que se utilizan en cada tarea, éstos también intervienen en la gestión de recursos, SIGHOS los contempla en su última versión, pero no este programa.

- **Work groups**: Actualmente, éstos se crean para cada actividad, es decir, cada actividad tiene un workgroup asignado y no se repite para varias actividades diferentes. El código está preparado para poder implementar esta función con pocas modificaciones, teniendo en cuenta los recursos y los tiempos, que pueden determinarse en el modelo estático con eclipse.

Este proyecto cumple con los objetivos y requerimientos iniciales, aún así tiene ciertas limitaciones, pues es cierto que no hace falta elaborar dos modelos, sino que de un modelo estático obtenemos también la simulación; pero no podemos modificar los valores de la simulación.

Debido a esto, nos surgen nuevos problemas, que pueden ser objeto de futuros proyectos:

- **Traducción inversa**: Un buen complemento a este proyecto sería elaborar justo lo contrario, una conversión del modelo dinámico que usamos en SIGHOS al modelo estático en BPMN. Con ello, podremos modificar el modelo de simulación y realizar, a partir de éste, la conversión al modelo estático, sin necesidad de implementar los cambios de nuevo en el modelo con BPMN.

**Incorporar opciones de edición de datos**: Actualmente el conversor nos permite establecer algunos datos de forma dinámica, como son el nombre del modelo o los porcentajes de las condiciones, pero no podemos modificarlos una vez que tengamos el fichero java para SIGHOS, a no ser que lo hagamos directamente en el código del fichero. Sería interesante elaborar un proyecto en el que se incorpore una interfaz de edición de ciertos valores seleccionados para la simulación sin tener que acudir al código java o al modelo estático para ello.

# Capítulo 7
# Summary and conclusions.

With the development of this project, I have acquired knowledge about the modelling tools and languages that are currently available, and also on the needs of companies and the importance of updating in a world of constant technological change and the relevance of the role of computing in it.

As a result, a program has been achieved that allows a simulation based on a static graphic model to be carried out using SIGHOS. This functionality prevents the user from elaborating a dynamic model again, which is the main objective of this project.

In this project different languages and tools have been analysed. The results have shown that, for the major part of cases implemented, the chosen technologies were the best fitted to our objective, following the requirements established at the beginning of the project.

Although the validity and usefulness of this project has been demonstrated, several development options have not been implemented and could complement the current project:

- **Lanes in BPM:** Lanes are a structure that is not used by all models, but has been incorporated for a clearer division of the functions and the resources used by he processes. Currently, although they can be used in the model to have a clearer representation, they are not useful in the simulation. One possible future research line could be the use of lanes to group tasks and resources.

- **Incorporation of Human Resources:** Although BPMN lacks from support to define the human resources involved in the tasks, SIGHOS uses this information in the simulation. There are alternatives to achieve this goal, which have been explained in chapter 5 of the report. However, it has also been concluded in these studies that Oracle BPEL PM does have support for human resources and is also compatible with BPMN. A future line of research could be the option to incorporate human resources from Oracle BPEL PM in the models made with BPMN.

- **Implementation of new features:** The code is limited to only support the basic characteristics of workflow patterns. It can be improved including the most complex workflow properties.

- **Incorporation of Space Resources:** Space resources are the places that we use to do the tasks; they also take part in the management of resources, but this program do not contemplate them.

- **Work groups:** Currently, these are created for each activity. A work group is not repeated for several different activities. The code is ready to implement this function with few modifications, taking into account the resources and times, which can be determined in the static model with eclipse.

This project satisfies the initial objectives and requirements, but it has some limitations. Now it is not necessary to elaborate two models; from a static model we can obtain the simulation; but we cannot modify the values of the simulation.

Due to this, we have new problems that may be the subject of future projects:

- **Reverse translation:** A good complement to this project would be to create just the opposite, a conversion of the dynamic model that we use in SIGHOS to the static model in BPMN. We could modify the simulation model and do the conversion to the static model, without having to implement the changes again in the model with BPMN.

- **Incorporate data editing options:** The converter allows us to establish some data dynamically, such as the name of the model or the percentages of the conditions, but we cannot modify them once we have the java file for SIGHOS, unless we do it directly in the code of the file. It would be interesting to elaborate a project in which an interface of edition of certain selected values for the simulation is incorporated without having to resort to the java code or to the static model for it.

# Capítulo 8
# Presupuesto

En esta sección se especifican los costes que supone el proyecto, teniendo en cuenta tanto los recursos humanos como materiales.

| **Descripción** | **Coste por hora** | **Coste total** |
|---|---|---|
| Hardware: Equipo Informático completo | - | 2000 € |
| Software: Eclipse, plugin bpmn, SIGHOS... | - | 0 € |
| Desarrollador (Sobre 250 horas de trabajo) | 12€ | 3000€ |
| **Total** | | **5000€** |

*Tabla 4: 8.1. Presupuesto*

# Capítulo 9

# ANEXO I: Elementos de PSIGHOS

| Clase | Descripción |
|---|---|
| Experiment | En esta clase se define el número de simulaciones y el tipo. |
| Simulation | Clase principal del modelo. En ella se definen todos los componentes del modelo. |
| ElementType | Son entidades que interaccionan con el sistema. Es el que da inicio al proceso. Puede hacer uso de recursos. |
| ResourceType | Representan los diferentes roles que puede asumir un recurso en cada momento. Por ejemplo, un recurso Doctor, puede asumir el rol de cirujano o de médico pediatra. |
| Resource | Los recursos son los medios materiales o humanos necesarios para llevar a cabo una tarea. Pueden asignarse horarios de disponibilidad y asignarles un rol para cada horario. |
| WorkGroup | Definen el tipo y cantidad de recursos para realizar una tarea. |
| Activity | Son las tareas que pueden realizar los elemento. Cada una viene definida por un grupo de trabajo o varios. |
| ExclusiveChoice | Es un flujo de decisión, consiste en un elemento que dada una entrada puede tomar diferentes salidas. |
| PercentageCondition | Representa el porcentaje de probabilidad de que se coja una salida u otra en un ExclusiveChoice. Se le asigna un porcentaje a cada salida. |
| ElementGenerator | Controla el tiempo entre la creación de elementos. Es decir, genera un elementType cada x tiempo. |
| SimulationPeriodicCycle | Asigna la cantidad de horas de una jornada a un recurso. |

*Tabla 5: A.1. Clases de SIGHOS.* Fuente: [10]

# Capítulo 10

# ANEXO II: código del programa.

## 10.1 Main.java

```
public class Main {
    static Parser3 par;

    public static void main(String[] args) {
        par = new Parser3("lavanderiaHospitalaria.bpmn2", "lavanderiaHospitalaria.java");

}
```

## 10.2 Parser3.java

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collection;
import java.util.Scanner;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import org.camunda.bpm.model.bpmn.Bpmn;
import org.camunda.bpm.model.bpmn.BpmnModelInstance;
//import org.camunda.bpm.model.bpmn.impl.BpmnModelConstants;
import org.camunda.bpm.model.bpmn.instance.ConditionExpression;
import org.camunda.bpm.model.bpmn.instance.Documentation;
import org.camunda.bpm.model.bpmn.instance.FlowNode;
import org.camunda.bpm.model.bpmn.instance.SequenceFlow;
/* These libraries are used in functions that we are not currently using:

    import org.camunda.bpm.model.bpmn.instance.EndEvent;
```

```
import org.camunda.bpm.model.bpmn.instance.Event;
import org.camunda.bpm.model.bpmn.instance.ExclusiveGateway;
import org.camunda.bpm.model.bpmn.instance.ExtensionElements;
import org.camunda.bpm.model.bpmn.instance.Gateway;
import org.camunda.bpm.model.bpmn.instance.Script;
import org.camunda.bpm.model.bpmn.instance.ScriptTask;
import org.camunda.bpm.model.bpmn.instance.ServiceTask;
import org.camunda.bpm.model.bpmn.instance.StartEvent;
import org.camunda.bpm.model.bpmn.instance.UserTask;
import org.camunda.bpm.model.bpmn.instance.bpmndi.BpmnPlane;
import org.camunda.bpm.model.bpmn.instance.camunda.CamundaExecutionListener;
import org.camunda.bpm.model.bpmn.instance.camunda.CamundaFormData;
import org.camunda.bpm.model.xml.instance.ModelElementInstance;
*
*/

public class Parser3 {
    //Input Bpmn2 File
    File input;
    //Output Java File
    FileWriter output;
    protected BpmnModelInstance modelInstance;
    protected String links = "";
    protected String nEt, ActnEt = "";

    //Constructor function
    Parser3(String file_, String outputFile) {
        input = new File (file_);
        modelInstance = Bpmn.readModelFromFile(input);

        try {
            output = new FileWriter(outputFile);
        }
        catch (IOException e) {
            e.printStackTrace();
        }

        System.out.println ("Introduzca el nombre del modelo:");
        Scanner scannerIN = new Scanner (System.in);
        String model = scannerIN.nextLine();
        preCharge(model);
        scannerIN.close();
        findByType();
```

```
    try {
        output.write("\r\n        //Links: \r\n" + links
        +"        new TimeDrivenElementGenerator(this, " + nEt + ActnEt
        + ", SimulationPeriodicCycle.newDailyCycle(unit, 8 * 60));" + "\r\n");
    }
    catch (IOException e) {
        e.printStackTrace();
    }
    closeFiles();



//Function that loads the initial part of the java file: Packages, libraries, etc
private void preCharge(String model)

    String static_ =
        "package es.ull.iis.simulation.examples.hospital;\r\n" +
        "\r\n" +
        "import es.ull.iis.function.TimeFunctionFactory;\r\n" +
        "import es.ull.iis.simulation.condition.PercentageCondition;\r\n" +
        "import es.ull.iis.simulation.model.ElementType;\r\n" +
        "import es.ull.iis.simulation.model.Simulation;\r\n" +
         "import es.ull.iis.simulation.model.SimulationPeriodicCycle;\r\n" +
         "import es.ull.iis.simulation.model.Resource;\r\n" +
     "import es.ull.iis.simulation.model.ResourceType;\r\n" +
         "import es.ull.iis.simulation.model.TimeDrivenElementGenerator;\r\n" +
         "import es.ull.iis.simulation.model.TimeUnit;\r\n" +
         "import es.ull.iis.simulation.model.WorkGroup;\r\n" +
         "import es.ull.iis.simulation.model.flow.ActivityFlow;\r\n" +
         "import es.ull.iis.simulation.model.flow.ExclusiveChoiceFlow;\r\n" +
         "\r\n" +
         "public class model" + model + " extends Simulation {\r\n" +
         "\r\n" +
         "    /**\r\n" +
         "    * @param id\r\n" +
         "    * @param description\r\n" +
         "    * @param unit\r\n" +
         "    * @param startTs\r\n" +
         "    * @param endTs\r\n" +
         "    */\r\n" +
         "    \r\n" +
         "    public nuevaSimulacion(int id, TimeUnit unit, long startTs, long endTs) {" +
```

```
        "\r\n        super(id, \"" + model + "\", unit, startTs, endTs); \r\n"+
        "\r\n";

    try {
        output.write(static_);
    }
    catch (IOException e) {
        e.printStackTrace();
    }


//Search elements by their type
private void findByType()
{
     //get all flow nodes in the model
    Collection<FlowNode> flowNodes =
    modelInstance.getModelElementsByType(FlowNode.class);

    for(FlowNode e : flowNodes)
    {
        String attribute = e.getId();
        String name_ = e.getName();
        name_ = name_.replaceAll(" ", "_");

        //The start Event Contain the name of Element Type.
        Pattern pat = Pattern.compile("StartEvent.*");
        Matcher mat = pat.matcher(attribute);
        if (mat.matches())
        {
            Collection <Documentation> D = e.getDocumentations();
            for(Documentation d:D)
               nEt = d.getTextContent();

            try{
               output.write("        ElementType et" + name_ + " = new
                             ElementType(this, \""+ name_+ "\");\r\n");
               nEt = nEt + ", et" + name_;
            }
            catch (IOException c) {
                c.printStackTrace();
            }
        }
```

```
//Exclusive Gateways.
String percentageName = "";
pat = Pattern.compile("ExclusiveGateway.*");
mat = pat.matcher(attribute);
if (mat.matches())
{
    try{
       output.write("        ExclusiveChoiceFlow " + name_ + " =
                        new ExclusiveChoiceFlow(this);"
       + "\r\n");
    }
    catch (IOException c) {
        c.printStackTrace();
    }

    Collection<SequenceFlow> list = e.getOutgoing();
    for (SequenceFlow seq : list) {
        //There should always be one or more conditions.
        ConditionExpression cond = seq.getConditionExpression();
        if(cond != null) {
           percentageName = seq.getName();
           percentageName = percentageName.replaceAll(" ", "_");

            try {
               output.write("        PercentageCondition " +
                                percentageName + "Condition = new
                                PercentageCondition(" + cond.getTextContent() +
                                "); \r\n");
            } catch (IOException c) {
                c.printStackTrace();
            }
        }
    }
    try {
        output.write("\r\n");
    } catch (IOException c) {
        c.printStackTrace();
    }
    getSequenceFlows(e, name_);
}

//At the moment, we do not distinguish the tasks by types.
pat = Pattern.compile(".*Task.*");
```

```
        mat = pat.matcher(attribute);
        if (mat.matches())
        {
            try{
               output.write("        ActivityFlow " + name_ + " = new
                            ActivityFlow(this, \""+ name_+ "\");\r\n");
            }
            catch (IOException c) {
                c.printStackTrace();
            }

            createWorkgroup(e, name_);
        }
    }



//Get all sequence flows in the model
private void getSequenceFlows(FlowNode e, String name)

    String id, source, target = "";
    //Exclusive Gateway?
    Boolean eg_ = false;

   Collection<SequenceFlow> sequenceFlows =
     modelInstance.getModelElementsByType(SequenceFlow.class);

    for(SequenceFlow sf : sequenceFlows)
    {
        SequenceFlow flow = modelInstance.getModelElementById(sf.getId());
        id = flow.getAttributeValue("targetRef");
        target = getNameById(id);

        Pattern pat2 = Pattern.compile("EndEvent.*");
        Matcher mat2 = pat2.matcher(id);

        id = flow.getAttributeValue("sourceRef");
        source = getNameById(id);

        Pattern pat = Pattern.compile("StartEvent.*");
        Matcher mat = pat.matcher(id);
        if(mat.matches())
            ActnEt = (", " + target);
```

```
        if(!mat.matches() && !mat2.matches())
        {
            pat = Pattern.compile("ExclusiveGateway.*");
            mat = pat.matcher(id);
            if (mat.matches())
                eg_ = true;

            if(eg_ == true)
            {
                links = links + ("        " + source + ".link(" + target + ", " +
                                target + "Condition);\r\n");
              eg_ = false;
            }
            else
                links = links + ("        " + source + ".link(" + target + ");\r\n");
        }
    }
    links = links +  ("\r\n");

//Create WorkGroups
private void createWorkgroup(FlowNode e, String name)

    ArrayList<String> wg = new ArrayList<String>();
    Collection <Documentation> doc = e.getDocumentations();
    for(Documentation d:doc)
    {
        String str = d.getTextContent();
        String[] words = str.split("\\n");

        for (String w:words)
        {
            String[] rec = w.split("\\s");
            for (String r:rec)
                wg.add(r);
        }
    }

    try{
      output.write("        WorkGroup wg" + name + " = new WorkGroup(this,
            new ResourceType[] {");
    }
    catch (IOException c) {
```

```
    c.printStackTrace();
}

for(int i=0; i<wg.size(); i=i+2)
{
    try{
        if(i!=wg.size()-2)
                        output.write(wg.get(i)+ ", ");
        else
                        output.write(wg.get(i));
    }
    catch (IOException c) {
        c.printStackTrace();
    }
}

try{
    output.write("}, new int[] {");
}
catch (IOException c) {
    c.printStackTrace();
}

for(int i=1; i<wg.size(); i=i+2)
{
    try{
        if(i!=wg.size()-1)
                        output.write(wg.get(i) + ", ");
        else
                        output.write(wg.get(i));
    }
    catch (IOException c) {
        c.printStackTrace();
    }
}

try{
    output.write("});\r\n");
}
catch (IOException c) {
    c.printStackTrace();
}
```

```
        try{
            output.write("        act" + name + ".addWorkGroup(0, wg" + name + ",
                TimeFunctionFactory.getInstance(\"UniformVariate\", 7, 10)); \r\n\r\n");
        }
        catch (IOException c) {
            c.printStackTrace();
        }


private String getNameById(String id) {
         //get all flow nodes in the model
        Collection<FlowNode> flowNodes =
         modelInstance.getModelElementsByType(FlowNode.class);

        for(FlowNode e : flowNodes)
        {
            String attribute = e.getId();
            String name_ = e.getName();
            name_ = name_.replaceAll(" ", "_");

            Pattern pat = Pattern.compile(id);
            Matcher mat = pat.matcher(attribute);
            if (mat.matches())
            {
                return name_;
            }
        }
        return id;



    //Close files
    private void closeFiles()

        try {
            output.write("    }\r\n"
                    +"}");
            output.close();
        }
        catch (IOException e) {
                e.printStackTrace();
        }
    }
```

# Capítulo 11

# ANEXO III: código java simulable.

```
package es.ull.iis.simulation.examples.hospital;

import es.ull.iis.function.TimeFunctionFactory;
import es.ull.iis.simulation.condition.PercentageCondition;
import es.ull.iis.simulation.model.ElementType;
import es.ull.iis.simulation.model.Simulation;
import es.ull.iis.simulation.model.SimulationPeriodicCycle;
import es.ull.iis.simulation.model.Resource;
import es.ull.iis.simulation.model.ResourceType;
import es.ull.iis.simulation.model.TimeDrivenElementGenerator;
import es.ull.iis.simulation.model.TimeUnit;
import es.ull.iis.simulation.model.WorkGroup;
import es.ull.iis.simulation.model.flow.ActivityFlow;
import es.ull.iis.simulation.model.flow.ExclusiveChoiceFlow;

public class modelLavanderiaHospitalaria extends Simulation {

        /**
        * @param id
        * @param description
        * @param unit
        * @param startTs
        * @param endTs
        */

        public nuevaSimulacion (int id, TimeUnit unit, long startTs, long endTs) {
                super(id, "LavanderiaHospitalaria", unit, startTs, endTs);
```

```
ExclusiveChoiceFlow Sabana,_Colcha_O_Metida = new ExclusiveChoiceFlow(this);
PercentageCondition SecadoExteriorCondition = new PercentageCondition(65);
PercentageCondition PlanasCondition = new PercentageCondition(45);

ExclusiveChoiceFlow Acabado = new ExclusiveChoiceFlow(this);
PercentageCondition UniformesCondition = new PercentageCondition(10);
PercentageCondition ManualCondition = new PercentageCondition(15);

ActivityFlow SecadoExterior = new ActivityFlow(this, "SecadoExterior");
WorkGroup wgSecadoExterior = new WorkGroup(this, new ResourceType[] {rtOperario}, new int[] {2});
actSecadoExterior.addWorkGroup(0, wgSecadoExterior, TimeFunctionFactory.getInstance("UniformVariate", 7, 10));

ActivityFlow TransporteHastaLavadoras = new ActivityFlow(this, "TransporteHastaLavadoras");
WorkGroup wgTransporteHastaLavadoras = new WorkGroup(this, new ResourceType[] {}, new int[] {});
actTransporteHastaLavadoras.addWorkGroup(0, wgTransporteHastaLavadoras, TimeFunctionFactory.getInstance("UniformVariate", 7, 10));

ActivityFlow Lavar_y_Secar = new ActivityFlow(this, "Lavar_y_Secar");
WorkGroup wgLavar_y_Secar = new WorkGroup(this, new ResourceType[] {}, new int[] {});
actLavar_y_Secar.addWorkGroup(0, wgLavar_y_Secar, TimeFunctionFactory.getInstance("UniformVariate", 7, 10));

ActivityFlow DistribucionRopaCintas = new ActivityFlow(this, "DistribucionRopaCintas");
WorkGroup wgDistribucionRopaCintas = new WorkGroup(this, new ResourceType[] {rtOperario}, new int[] {2});
actDistribucionRopaCintas.addWorkGroup(0, wgDistribucionRopaCintas, TimeFunctionFactory.getInstance("UniformVariate", 7, 10));

ActivityFlow MeterEnLavadora = new ActivityFlow(this, "MeterEnLavadora");
```

```
WorkGroup wgMeterEnLavadora = new WorkGroup(this, new ResourceType[] {rtOperario}, new int[] {1});

actMeterEnLavadora.addWorkGroup(0, wgMeterEnLavadora, TimeFunctionFactory.getInstance("UniformVariate", 7, 10));


ActivityFlow Manual = new ActivityFlow(this, "Manual");

WorkGroup wgManual = new WorkGroup(this, new ResourceType[] {rtTecnico}, new int[] {1});

actManual.addWorkGroup(0, wgManual, TimeFunctionFactory.getInstance("UniformVariate", 7, 10));


ActivityFlow Uniformes = new ActivityFlow(this, "Uniformes");

WorkGroup wgUniformes = new WorkGroup(this, new ResourceType[] {rtTecnico}, new int[] {1});

actUniformes.addWorkGroup(0, wgUniformes, TimeFunctionFactory.getInstance("UniformVariate", 7, 10));


ActivityFlow Planas = new ActivityFlow(this, "Planas");

WorkGroup wgPlanas = new WorkGroup(this, new ResourceType[] {rtTecnico}, new int[] {1});

actPlanas.addWorkGroup(0, wgPlanas, TimeFunctionFactory.getInstance("UniformVariate", 7, 10));


ElementType etSacoRopaSucia = new ElementType(this, "SacoRopaSucia");


//Links:

DistribucionRopaCintas.link(TransporteHastaLavadoras);

TransporteHastaLavadoras.link(MeterEnLavadora);

MeterEnLavadora.link(Lavar_y_Secar);

Sabana,_Colcha_O_Metida.link(SecadoExterior, SecadoExteriorCondition);

Sabana,_Colcha_O_Metida.link(Planas, PlanasCondition);

Lavar_y_Secar.link(Sabana,_Colcha_O_Metida);

SecadoExterior.link(Acabado);

Acabado.link(Uniformes, UniformesCondition);

Acabado.link(Manual, ManualCondition);

Acabado.link(Planas, PlanasCondition);
```

```
            DistribucionRopaCintas.link(TransporteHastaLavadoras);
            TransporteHastaLavadoras.link(MeterEnLavadora);
            MeterEnLavadora.link(Lavar_y_Secar);
            Sabana,_Colcha_O_Metida.link(SecadoExterior, SecadoExteriorCondition);
            Sabana,_Colcha_O_Metida.link(Planas, PlanasCondition);
            Lavar_y_Secar.link(Sabana,_Colcha_O_Metida);
            SecadoExterior.link(Acabado);
            Acabado.link(Uniformes, UniformesCondition);
            Acabado.link(Manual, ManualCondition);
            Acabado.link(Planas, PlanasCondition);

            new TimeDrivenElementGenerator(this, 12, etSacoRopaSucia,
DistribucionRopaCintas, SimulationPeriodicCycle.newDailyCycle(unit, 8 * 60));
        }
    }
```

# Bibliografía


[1] -. (2019). Business Process Definition. -, de Appian Sitio web: https://es.appian.com/bpm/definition-of-a-business-process/

[2] José Pablo. (2013). Diferencia entre modelo y simulacion. -, de Blog Sitio web: https://www.elconspirador.com/2013/12/15/diferencia-entre-modelo-y-simulacion/

[3] -. (2016). Trucos para mejorar la productividad de tu empresa. -, de WorkMeter Sitio web: https://es.workmeter.com/blog/trucos-mejorar-productividad-empresa

**[Recurso en papel: Disponible en la Biblioteca de la facultad de Física y Matemáticas de la Universidad de la Laguna]**
[4] Wil Van der Aalst y Kees Van Hee. (2004). Workflow Management. Massachusetts: The MIT Press.

[5] -. (2016). Kees van Hee. -, de Wikipedia Sitio web: https://en.wikipedia.org/wiki/Kees_van_Hee

-. (2019). Wil van der Aalst. -, de Wikipedia Sitio web: https://en.wikipedia.org/wiki/Wil_van_der_Aalst

[6] Wil Van der Aalst y Kees Van Hee. (2010 - 2017). Workflow patterns. -, de Workflow Patterns Initiative Sitio web: http://www.workflowpatterns.com/

[7] -. Sistemas Workflow. -, de EcuRed Sitio web: https://www.ecured.cu/Sistemas_Workflow#Tipos_de_Workflow

[8] Petia Wohed, Wil M. P. van der Aalst, Marlon Dumas, Arthur H. M. ter Hofstede y Nick Russell. (2005). *Pattern-based Analysis of BPMN -an extensive evaluation of the Control-flow, ¿the Data and the Resource Perspectives?.* -, de Semantic Scholar Sitio web: https://www.semanticscholar.org/paper/Pattern-based-Analysis-of-BPMN-an-extensive-of-the-Wohed-Aalst/64fd40201cad2c240d942de3729f1db02c86ab21

[9] Wildredo Guaita. Modelos de Simulación de Eventos Discretos y de Procesos Continuos. -, de Boletín de Dinámicas de Sistemas Sitio web: http://dinamica-de-sistemas.com/revista/0608o.htm

[10] Iván Castilla - ISAATC. (2019). PSIGHOS. -, de GitHub Sitio web: https://github.com/ull-isaatc/sighos

[11] Iván Castilla Rodríguez. (2012). Tesis Doctoral "Explotación de los sistemas multi-núcleo para la simulación paralela de eventos discretos en Java". -, de Universidad de La Laguna Sitio web: https://riull.ull.es/xmlui/handle/915/3383

**[Recurso en papel: Disponible en la Biblioteca de la facultad de Física y Matemáticas de la Universidad de la Laguna]**
[12] Patrice Briol. (2008). BPMN the Business Process Modeling Notation. -: Pocket Handbook.

[13] Quirón. (2005). Introducción a UML 2.0. -, de Epiwiki Sitio web: http://www.epidataconsulting.com/tikiwiki/tiki-read_article.php?articleId=15

[14] -. (2018). WS-BPEL. -, de Wikipedia Sitio web: https://es.wikipedia.org/wiki/WS-BPEL

[15] -. Oracle BPEL Proccess Manager. -, de Oracle Sitio web: https://www.oracle.com/technetwork/es/middleware/bpel/overview/index.html

[16] -. (2019). BPMN 2.0 for the web. -, de NPM Sitio web: https://www.npmjs.com/package/bpmn-js

[17] -. Plataforma de código abierto para la automatización de procesos de negocios. -, de Bonitasoft, S.A Sitio web: https://es.bonitasoft.com/

[18] -. ADONIS: La nueva dimensión en la Gestión de Procesos de Negocios. -, de BOC ES Sitio web: https://es.boc-group.com/adonis-E

[19] -. (2018). BPMN2 Modeler. -, de Eclipse Foundation, Inc Sitio web: https://www.eclipse.org/bpmn2-modeler/

[20] -. (2019). Visio. Trabaja de forma visual. Diagramación de forma sencilla. -, de Microsoft Sitio web: https://products.office.com/es-es/visio/flowchart-software

[21] José Pacheco. (2018). ¿Por qué y cómo utilizar la notación más ampliamente aceptada?. -, de HEFLO Sitio web: https://www.heflo.com/es/blog/modelado-de-procesos/notacion-bpmn-2/

[22] Mkyong. (2008). DOM XML Parser. -, de Mkyong Sitio web: https://www.mkyong.com/java/how-to-read-xml-file-in-java-dom-parser/2/

[23] LuisMi Gracia. (2018). Un poco de Camunda. -, de Un poco de Java y + Sitio web: https://unpocodejava.com/2018/09/24/un-poco-de-camunda/

[24] -. (2019). Librerías Camunda. -, de GitHub Sitio web: https://github.com/camunda/camunda-xml-model/blob/master/src/main/java/org/camunda/bpm/model/xml/Model.java

[25] -. Eclipse Oxygen. -, de Eclipse Foundation Sitio web: https://www.eclipse.org/oxygen/

[26] -. (2006 - 2019). Open Source Business Automation Toolkit. -, de jBPM Sitio web: https://www.jbpm.org/

[27] Carla Soler Lucero. (2019). Guia de instalación Plugin Eclipse JBPM. -, de JBOSS Sitio web: https://docplayer.es/1839752-Disenador-grafico-de-proceso-jboss-jbpm.html

[28] -. (2006 - 2019). Repositorio Maven - Librerías Camunda. -, de Maven Ropository Sitio web: https://mvnrepository.com/artifact/org.camunda.bpm.model

[29] JUnit Team. (2019). JUnit 5. -, de JUnit 5 Sitio web: https://junit.org/junit5/

[30] Ana María Benítez Toledo (Bizagi). (2016). Mpdelamiento de Procesos usando BPMN y BizAgi. -, de SlidePlayer Sitio web: https://slideplayer.es/slide/8599953/

[31] -. Generados de bibliografía en formato APA. -, disponieble en sitio web: http://www.cva.itesm.mx/biblioteca/pagina_con_formato_version_oct/apaweb.html