Trabajo de Fin de Grado

Interfaz de modelado gráfico para la librería SIGHOS

Graphic modeling interface for the SIGHOS library

Samer Al Khindi Ternovskiy

La Laguna, 31 de mayo de 2018

D. **Iván Castilla Rodríguez,** con N.I.F. 78.565.451-G profesor 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:

"Interfaz de modelado gráfico para la librería SIGHOS"

ha sido realizada bajo su dirección por D. **Samer Al Khindi Ternovskiy**, con N.I.F. 54.669.442-Y.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 31 de mayo de 2018

Agradecimientos

En especial a mi familia, por su paciencia apoyo incondicional.

A Iván Castilla, por su gran ayuda y constancia en todo momento.

Licencia

Resumen

Este trabajo fin de grado consiste en el desarrollo de una herramienta que proporcione una interfaz de modelado gráfico a la librería de simulación de eventos discretos SIGHOS (SImulación para la Gestión HOSpitalaria). Dicha herramienta permitirá traducir un diagrama de flujo basado en la descripción de un proceso de negocio en declaraciones de métodos y atributos de la librería SIGHOS. Finalmente, todos los métodos obtenidos podrán ser descargados en un fichero fuente Java, con el que se podrá trabajar para completar la descripción del modelo de simulación.

Palabras clave: simulación, diagrama de flujo de trabajo, SIGHOS.

Abstract

This end-of-degree project consists of the development of a tool that provides a graphical modeling interface to the SIGHOS discrete events simulation library (SImulation for HOSpital Management). This tool will translate a flowchart based on the description of a business process into declarations of methods and attributes of the SIGHOS library. Finally, a user will be able to download a Java source file containing these methods and attributes, with which the user will be able to work to complete the description of the simulation model.

Keywords: simulation, workflow diagram, SIGHOS.

Índice General

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1 Introducción

Hoy en día, la complejidad de la estructura empresarial de las grandes organizaciones hace de la simulación un pilar en la producción a gran escala. Ejecutar un proyecto que disponga de un soporte de reproducción de los posibles escenarios, permite tomar decisiones con mayor certeza y efectividad. La estructura enrevesada de las grandes instituciones es el contexto adecuado para la aplicación de las técnicas de simulación, sin embargo, su ámbito complejo obliga a implementar una planificación precisa y descriptiva. Además, es importante acercar la simulación a mercados distintos de las multinacionales. Poseer un sistema de información optimizado permite a la mediana empresa, a las organizaciones públicas o al sistema sanitario optimizar el uso de sus recursos materiales y humanos, mejorando de esta manera la calidad de sus servicios.

1.2 Motivación para el trabajo

Actualmente existe un gran número de herramientas de simulación que permiten el diseño y la simulación de un modelo. Podemos encontrar este tipo de software de simulación tanto con licencia gratuita, como en el ámbito comercial. Una herramienta de estas características fue desarrollada el Grupo de Simulación de la ULL, el nombre de la librería de simulación e SIGHOS. Dicha librería nos ofrece la posibilidad de implementar modelos de simulación utilizando Java, aunque la falta de una interfaz gráfica complica su utilización.

A partir de estas consideraciones se plantea abordar el desarrollo de una interfaz gráfica que permita la utilización de la librería de simulación de eventos discretos SIGHOS. Con este fin se realizará un estudio de la parte funcional y operativa de la herramienta de simulación y se precederá al planteamiento de una solución grafica al manejo de ésta y al aprovechamiento de sus recursos.

Capítulo 2

Antecedentes

2.1 La Simulación.

La simulación es un acto que se basa en imitar el desempeño de una acción cuando en realidad esta no se está realizando. Generalmente, esta reproducción del funcionamiento de un elemento a lo largo del tiempo se lleva a cabo para tareas de estructuras complejas [1].

Previo al planteamiento de una simulación como solución, es necesario determinar si se trata del tratamiento adecuado al problema. Para ello es necesario evaluar los siguientes puntos:

- Magnitud del problema: En caso de que sea posible, obtener la solución al problema en cuestión mediante un algoritmo matemático o incluso mediante el sentido común. Es necesario tener en cuenta que plantear un sistema de simulación añadirá un coste económico y temporal innecesario
- Disponer del simulador adecuado para el problema: es esencial disponer de la herramienta adecuada y de un profesional que sea capaz de abstraer la información necesaria del problema para

ajustar la herramienta de forma apropiada. Es importante conocer el nivel de especificación adecuado al estudio, esto supondrá un ahorro de tiempos y recursos. De esta forma, no disponer de la herramienta adecuada y de un analista experimentado, implicará el planteamiento de una simulación cuyo desempeño no sea provechos para el tratamiento del problema.

- Recursos económicos: es necesario realizar un estudio de los costes económicos que conllevará el planteamiento de un sistema de simulación y de si el gasto realizado es justificado por los beneficios esperados.

- Obtención de datos: cualquier proceso de simulación debe ser alimentado con datos sobre el entrono a emular. En caso de que no se disponga de la información necesaria, es primordial cuantificar los recursos que se necesitarán para recopilar dicha información. Así mismo, realizar un estudio de simulación sin los datos necesarios para ello devolverá unas conclusiones poco exactas o, en el peor de los cosos, inservibles.

2.2 Modelos de simulación de eventos discretos

Dada la cantidad de diversos tipos de simulación existentes, cabe destacar que este trabajo se centra en la simulación de eventos discretos. La principal característica de la simulación de eventos discretos es que es dinámica, es decir, tiene en cuanta la progresión del sistema en el tiempo. Por el contario, tendríamos los modelos de simulaciones estáticos, que son modelo que no tienen en cuenta al tiempo como variable [2].

Los modelos de simulación de eventos discretos también permiten modelar un sistema de forma estocástica. Esto implica la inclusión de la aleatoriedad y el imprevisto en el sistema. Así mismo, nos permite modelar sistemas donde no tenemos una visión exacta del problema. Esta propiedad nos proporciona la ventaja de manejar un alto nivel de abstracción en nuestro modelo, sin que tengamos la necesidad de especificar todos los detalles de los elementos de la simulación. Estas características de este tipo de modelos de simulación nos permiten construir nuestro modelo centrándonos en los elementos fundamentales de la simulación y no centrarnos en las propiedades de los elementos que los componen.

Por definición, el manejo discreto del tiempo se basa en realizar una división de la línea temporal, disminuyendo la frecuencia del muestreo y de esta forma rebajando significativamente la complejidad del modelo. Dentro del manejo discreto del tiempo, la simulación de eventos discretos da un

paso más en esa dirección y se centra sólo en las ocurrencias principales que afectan a un elemento.

Como aclaración, un planteamiento opuesto se encuentra cuando trabajamos con simulaciones de tiempo continuo. Éstas utilizan ecuaciones dependientes del tiempo, que permiten establecer el estado de los elementos en cada instante del tiempo. Esto aumenta en gran medida la complejidad del modelo, disminuye el rendimiento y en muchos casos no aporta información relevante a la solución del problema.

Los modelos de simulación de eventos discretos también tienen la propiedad de permitir la interaccionen entre los elementos. De esta forma se permite modelar la relación entre los distintos actores que forman parte del modelo. Dichas interacciones se pueden describir partiendo de que las acciones de un actor puedan afectar a otro actor, o mediante una competencia por recursos entre actores.

2.3 Patrones de control de flujos de trabajo (workflow patterns).

Para un correcto estudio de una herramienta de simulación, es necesario determinar el tipo de control de flujos que esta implementa. La evaluación de las estrategias que ofrecen los patrones de modelado de un lenguaje de flujo, permite analizar la idoneidad de un sistema de simulación o su eficacia para abordar un problema en particular.

Los patrones de control de los flujos de trabajo en la librería SIGHOS están basados en las definiciones de los profesores Arthur ter Hofstede y Wil van der Aalst sobre especificaciones de los flujos de trabajo. Estos patrones posibilitan representar cualquier estructura que implemente un proceso de simulación [3]. Según los autores mencionados, existen diferentes perspectivas para definir el comportamiento de un flujo de trabajo:

- La perspectiva flujo-control: utiliza diferentes constructores para describir las actividades y su orden de ejecución.

- La perspectiva de datos: proporciona un control de la perspectiva de las capas de negocio del entorno a analizar y de su estructura de datos.

- La perspectiva de recursos: proporciona una visión de la estructura organizativa al flujo de trabajo, en forma de roles encargados de la ejecución de actividades. Estos roles se puedes presentar como dispositivos o como trabajadores.

- La perspectiva de operaciones: define las actuaciones elementales realizadas por las actividades. Estas acciones retribuyen a las aplicaciones subyacentes.

2.4 SIGHOS

La librería SIGHOS, SImulación para la Gestión HOSpitalaria, es un simulador de eventos discretos desarrollado como parte de una tesis doctoral defendida en la ULL ("Explotación de los Sistemas Multi-Núcleo para la Simulación Paralela de Eventos Discretos con Java"). Se trata de una librería diseñada en Java 6 para ser ejecutada en una consola [4].

Para la implementación de las simulaciones, la librería maneja una serie de componentes que permiten describir las características de la simulación y su entorno. Estos componentes son:

- Elementos: son las entidades que describen a los actores principales de la simulación. Estas entidades interaccionan con el sistema, haciendo usos de los recursos del sistema. De esta forma los elementos dan comienzo a las actividades definidas en la simulación. La librería permite definir elementos de distintos tipos, de esta forma, mediante la clase "ElementType" se agrupan todos los elementos que pertenecen a un tipo concreto.

- Recursos: son los medios materiales o humanos necesarios para llevar a cabo una tarea definida en la simulación. Existe la posibilidad de que los recursos estén disponibles para uno o varios roles, según un horario establecido. Para definir los roles que puede

asumir un determinado recurso en un momento determinado se utiliza la clase “ResourseType”.

- Grupos de trabajo: definen el tipo de recursos y el número de estos, que son necesarios para llevar a cabo una determinada actividad. Cabe destacar, que la librería SIGHOS ofrece la posibilidad de que una actividad se realice utilizando distintos grupos de trabajo.

- Actividades: se definen como las tareas que realizan los elementos. La actividad es definida por uno o varios grupos de trabajo con los que es posible realizarla. Para cada actividad es necesario especificar el tiempo que es necesario para llevarla a cabo. El tiempo necesario en finalizar una tarea puede depender de una función del tiempo o de un conjunto de subtareas.

- Flujos de trabajo: las tareas que deben ser llevadas a cabo por un elemento, son modeladas mediante patrones de flujo de trabajo. Dichos patrones proporcionan una estructura para definir un proceso estándar, como son: bucles, condiciones, secuencias, etc.

Capítulo 3

Objetivo y alcance

3.1 Objetivos

El objetivo de este proyecto es el desarrollo de una herramienta que proporcione una interfaz gráfica que permita implementar un diagrama de eventos y posteriormente traducir dicho diagrama a código de la librería SIGHOS.

Se comprende como Interfaz Gráfica de Usuario GUI, del inglés Graphical User Interface, al ambiente virtual donde el usuario puede interactuar con una herramienta informática, evitando así una comunicación puramente textual con la herramienta [5].

La idea de la inclusión de una interfaz gráfica en la librería SIGHOS surge de la necesidad de proporcionar una comunicación intuitiva con la herramienta de simulación.

Se pretende que mediante la manipulación directa de objetos gráficos el usuario pueda realizar una descripción esquemática de un modelo de simulación y posteriormente transformar ese diagrama a código comprensible por la librería. Con este planteamiento se pretende lograr una

comunicación intuitiva y visualmente atractiva con la herramienta de simulación, proporcionando un entorno confortable de interacción con la librería.

3.2 Alcance

Para dar respuesta a las necesidades del apartado anterior es necesario el desarrollo de una interfaz gráfica que permita la creación, de forma sencilla, de los elementos fundamentales que maneja la librería de simulación. Es fundamental que la herramienta tenga una interfaz gráfica que permita arrastrar los distintos componentes del modelo, para así construir el diagrama de eventos.

En esta primera versión de la interfaz, todavía no se contempla la posibilidad de programar código de la librería SIGHOS dentro de la misma interfaz. Así mismo, la implementación de partes del código se realizará una vez generados los ficheros Java. También cabe destacar que la funcionalidad de la interfaz se centrará en la definición de los workflow, obviando de esta forma la necesidad de incluir todos los elementos de SIGHOS.

El diseño de la aplicación deberá permitir la realización de las siguientes operaciones:

- Implementar diagramas que describan una simulación SIGHOS.
- Añadir información a los distintos elementos del diagrama.

- Generar un paquete que contenga la descripción y la estructura del diagrama.

- Generar a partir del paquete de información, código Java con las clases de la librería SIGHOS.

3.2.1 Destinatarios.

El desarrollo de este proyecto va dirigido al ámbito docente y a la investigación. Se espera que esta herramienta permita un uso más accesible e intuitivo de la librería SIGHOS, permitiendo la generación de código a partir de las herramientas gráficas de la aplicación. De esta forma se pretende que el usuario pueda utilizar la librería sin necesidad de declarar clases o realizar una implementación mediante código Java de sus necesidades.

Más adelante se espera que una versión más adelantada y enfocada a un ámbito más empresarial, pudiendo abarcar campos desde el sector clínico, industrial agrario, hasta los transportes o la organización pública. Tratándose de una herramienta de aplicabilidad genérica, se plantea su potencial en diversos campos donde se requiera una organización de medios y personal.

Capítulo 4

Tecnologías empleadas

4.1 Elección de entorno

Antes del comienzo de la implementación de la interfaz gráfica, se realizó un planteamiento de la plataforma sobre la que se desea que funcione la interfaz. Una de las opciones posibles era la implementación de una aplicación de escritorio. La ventaja de este planteamiento era la posibilidad de desarrollar una aplicación desde los cimientos. Desarrollar una aplicación totalmente nueva permite adaptar todas sus características a las necesidades de la aplicación, de la misma forma que se excluye todas las características que nos van a ser requeridas por el sistema. A pesar del atractivo de la idea de tener una herramienta totalmente personalizada, se decidió dar más importancia a tener una herramienta multiplataforma y de fácil acceso al público, evitando así la necesidad de instalación adicional de software.

Otro planteamiento fue el diseño de un plugin para el entorno de desarrollo Eclipse [6]. Esta propuesta poseía la ventaja de comenzar el desarrollo del proyecto partiendo de un entorno conocido y testeado como es Eclipse. Para la implementación del plugin se utilizaría la versión de

Eclipse: Eclipse for RCP/Plugin Developers [7]. Esta versión ofrece al programador una variedad de herramientas y extensiones para el desarrollo de plugins para Eclipse y aplicaciones del lado del cliente. Este planteamiento permitía la implementación de una herramienta multiplataforma, pero obligaba al usuario a la instalación y a la utilización de un entorno de desarrollo concreto. Otra de las desventajas era la futura obsolescencia de esta estrategia.

Por otra parte, volviendo a los objetivos del proyecto, es importante que la herramienta sea accesible en un futuro para todo tipo profesionales, minimizando en lo posible la necesidad de instalaciones y de configuración de la herramienta por éstos. Para alcanzar este objetivo, es sumamente favorecedor la utilización de una tecnología asequible y puntera, donde la instalación de entornos de desarrollo, de complementos y plugins, puede llegar a complicar la experiencia de un usuario no familiarizado con dichas herramientas.

Buscando una solución actual y con una gran accesibilidad, surge la idea de implementar la herramienta utilizando las vigentes tecnologías web. Utilizando este enfoque, se logra alejar al usuario de posibles instalaciones y configuración del funcionamiento de la herramienta, permitiendo una interacción fácil e intuitiva con la librería.

Para concluir, reseñar que aunque el planteamiento del desarrollo de una aplicación de escritorio o plugin implementados en java facilita la

integración de la librería SIGHOS, tener una herramienta de fácil manejo, multiplataforma e implementada con tecnologías actuales permite tener una herramienta más acorde a las necesidades del usuario al que está destinado.

4.2 Herramientas de modelado

Tras el análisis de los objetivos del proyecto se determinó no realizar un desarrollo desde cero de la interfaz web de la aplicación. Por el contrario, se decidió utilizar una aplicación web disponible que permita la implementación de diagramas.

Cabe destacar que una de las fases más complejas fue la elección de las herramientas de modelado de los diagramas. Para ello se realizó un estudio previo de soluciones posibles, sus ventajas e inconvenientes y de su compatibilidad con los requisitos de la librería SIGHOS. Para formar parte del estudio previo, las herramientas a analizar debían cumplir las características de poseer una interfaz gráfica que permita un modelado descriptivo de procesos.

Una vez realizada la búsqueda de herramientas genéricas para la implementación de diagramas, el estudio de soluciones candidatas se centró en herramientas basadas en BPMN. Al ser los conceptos que manejan dichas soluciones similares a los que utilizan los patrones de workflow, se planteó la posibilidad de aprovechar estas herramientas en el modelado de los elementos de SIGHOS.

4.2.1 BPMN

Antes de mencionar las herramientas estudiadas para la implementación del proyecto, es necesario hablar de BPMN (Business Process Model and Notation), una notación gráfica estándar del modelado de administración empresarial, y por tanto, presente en la mayoría de herramientas de modelado de procesos [8].

BPMN es una notación estándar para el modelado de procesos de negocios que reduce la distancia entre el diseño de un proceso de negocio y su implementación. Está basada en diagramas de flujo y permite la representación de procesos de negocios, siendo una notación gráfica y fácil de entender. BPMN agrupa la planificación y gestión del flujo de trabajo, así como el modelado y la arquitectura, proporcionando de esta forma, un mecanismo para la generación de procesos de negocios ejecutables (BPEL - Business Process Execution Language) [9].

BPMN utiliza una Arquitectura Orientada por Servicios (SOA), con el objetivo de adaptarse rápidamente a los cambios y oportunidades del negocio, además, combina las capacidades del software y la experiencia de negocio para optimizar los procesos y facilitar la innovación del negocio [10].

Las características principales de BPMN:

- Se define como una notación gráfica que describe la lógica de los pasos en un proceso de negocio.

- Es un lenguaje formal que permite modelar, simular y ejecutar un proceso de negocio.

- Proporciona un método normalizado para representar procesos de negocio.

- Es legible y de complejidad media.

- Propone un lenguaje común entre los usuarios de negocio y los técnicos.

- Facilita la implementación gráfica de los procesos de negocio.

- Determina y define los requerimientos del sistema.

4.2.2 Bonita Studio

Desarrollada por Bonitasoft S.A.Bonita BPM, Bonita Studio es una herramienta basada en BPMN diseñada con el fin de permitir al usuario implementar aplicaciones empresariales a su medida. Esta herramienta presenta un entorno de desarrollo basado en eclipse y consta de una interfaz gráfica, que permite el diseño de modelos BPMN arrastrando los distintos elementos al área de trabajo. Esta aplicación consta de un portal web donde los diseñadores y los clientes autorizados pueden obtener información y visualizar los modelos gráficos sobre el trabajo realizado. Bonita Studio

permite el diseño de interface basadas en HTML5 y Angular JS para mostrar los resultados obtenidos al usuario. El flujo de trabajo puede ser actualizado en todo momento, permitiendo de este modo el ajuste de parámetros en tiempo real y la visualización de las modificaciones por el usuario [11].

Para nuestro estudio se ha analizado la versión con código abierto de la herramienta, sin embargo, existe una versión de pago que ofrece ventajas como mayor personalización de las interfaces, soporte global, permite el diseño de interfaces para dispositivos móviles y tabletas.

Ventajas de Bonita BPM:

- Es Open Source
- Posee una interfaz intuitiva
- Herramienta personalizable

Desventajas encontradas:

- A pesar de tener una interfaz personalizable, la adaptación a las necesidades de la librería SIGHOS no parecía factible.
- Es una interfaz potente en cuanto a las opciones que ofrece, pero sufre en el apartado del rendimiento, también es algo pesada.
- La única forma de modelado permitida era BPMN 2.0

- Aun disponiendo de una aplicación web para la visualización de los datos obtenidos es necesaria la instalación de una aplicación de escritorio para el diseño de los modelos.

4.2.3 GoJS

GoJS es una librería implementada JavaScript y HTML5 que permite el diseño de diagramas y gráficos interactivos en un navegador web. Desarrollada por Northwoods Software, esta herramienta proporciona una interfaz gráfica para construir diagramas, mediante un sistema de arrastre de elementos GoJS [12].

Tras analizar los tipos de diagramas de los que dispone GoJS, el tipo de diagrama que permite implementar un modelo de las características de la librería SIGHOS es el diagrama de nodos, dado que la herramienta permite desplegar un panel para cada nodo e introducir la información que atañe a cada uno. La cantidad de información que se puede introducir es limitada y solo admite descripciones sencillas.

El tipo de gráficos que soporta la herramienta son:

- Diagramas de flujo
- UML
- Árboles familiares
- Diagramas de nodos

Ventajas de la herramienta GoJS:

- Permite exportar los modelos a XML
- Diseño intuitivo.
- La herramienta posee una amplia documentación oficial.

Desventajas:

- No es software libre.
- Las propiedades que se pueden adjuntar al modelo son limitadas.

4.2.4 BPMN-JS

BPMN-JS es una herramienta web, lanzada por OMG en 2011, que permite mediante una notación gráfica renderizar diagramas BPMN 2.0. Esta librería está implementada en JavaScript y ofrece una solución al estudio de los procesos de negocio. BPMN-JS no precisa de servidor BackEnd y permite visualizar y modelar diagramas BPMN en cualquier navegador [13].

Ventajas:

- Permite exportar los modelos a XML
- Es de código abierto
- Fácil de integrar a cualquier aplicación web

- Los diagramas se pueden enriquecer con datos organizados en paneles.

Desventajas:

- Sólo dispone de un tipo de diagrama: el BPMN 2.0

4.2.5 Elección de herramienta

Tras analizar las ventajas y desventajas que se muestran en la Tabla 4.1, se decidió usar BPMN-JS como base para la implementación de la interfaz gráfica.

Herramienta	Ventajas	Desventajas
Bonita Studio	- Open Source - Interfaz intuitiva - Herramienta personalizable	- Difícil adaptación a las necesidades de SIGHOS - Rendimiento regular - Precisa de la instalación de una aplicación de escritorio La única forma de modelado
GoJS	- Permite exportar los modelos a XML - Diseño intuitivo - Amplia documentación oficial	- No es software libre - Las propiedades que se pueden adjuntar al modelo son limitadas
BPMN-JS	- Permite exportar los modelos a XML - Es de código abierto - Fácil de integración web - Ofrece diagramas enriquecidos con	La única forma de modelado permitida era BPMN 2.0

Tabla 4.1: Tabla resumen de ventajas y desventajas de herramientas analizadas.

4.3 Herramientas de ensamblaje

Para integrar la librería BPMN-JS a un proyecto web era necesario instalar Node.js [14] y hacerlo mediante el gestor de paquetes NPM [15]. De esta manera era posible personalizar la librería, teniendo acceso a todos sus componentes de forma individual.

Node.js es una plataforma construida sobre el motor de JavaScript de Google Chrome. Lanzada en el 2009 y desarrollada por Ryan Dahl, esta herramienta implementada en C++, permite usar a JavaScript como lenguaje de BackEnd. Su principal característica es que utiliza un modelo de programación asíncrono dirigido por eventos. Esta peculiaridad permite que las páginas y aplicaciones web no necesiten volver a ser cargadas cada vez que actualizan su contenido.

NPM es el gestor de paquetes por defecto que utiliza Node.js. Esta herramienta permite integrar librerías y herramientas en JavaScript a nuestra aplicación web mediante instrucciones por línea de comando. Todo proyecto que use NPM, consta de un fichero "package.json", que contendrá la configuración y las dependencias de la aplicación.

Capítulo 5

Diseño e Implementación

En este capítulo se explicarán los aspectos más importantes en el desarrollo de esta herramienta. Así mismo, se procederá a detallar el funcionamiento de la interfaz, los cambios aplicados a la herramienta BPMN-JS para adaptarla al funcionamiento de la librería SIGHOS y como se obtienen las clases Java de la librería SIGHOS a partir del diagrama de elementos.

5.1 Requisitos

Para la utilización de esta herramienta es necesario tener instalado la librería NODE-JS. Para el control de paquetes se utiliza la herramienta NPM y se utilizara GRUNT [16] como lanzador de tareas. En el archivo JSON [17] se especifica que la tarea va a correr en un servidor local en el puerto :9013 del navegador web que tengamos predefinido .

5.2 Adaptar BPMN.JS a SIGHOS

Plantear una interfaz para la librería SIGHOS mediante diagramas BPMN, plantea una solución un tanto imprecisa al problema. Los diagramas BPMN disponen de una estructura y una gran cantidad de elementos que

no van a ser utilizados por la librería. Por otro lado, plantear una simulación con SIGHOS requiere de la especificación de ciertos parámetros que el modelo BPMN no permite hacer. Por estas razones, tras varios intentos fallidos de implementar un modelo de simulación de eventos utilizando BPMN, se decidió que el planteamiento adecuado era modificar la herramienta BPMN-JS para adaptarla a las necesidades de la librería SIGHOS.

Los cambios planteados en la librería BPMN-JS son:

- Eliminar elementos de modelado BPMN innecesarios para la representación de simulaciones SIGHOS mediante diagramas.
    - Eliminar elementos del panel de selección de la interfaz.
    - Eliminar elementos del panel individual de cada elemento.
- Implementar sistema de parseo en JavaScript que permita obtener código Java con clases de la librería SIGHOS a partir del modelo BPMN.
- Obtener un fichero .java con las clases de la librería SIGHOS generadas a partir del modelo gráfico.

5.3 Implementación de un modelo de simulación.

Como se especificó anteriormente, la interfaz de la herramienta permite representar la estructura de una simulación de eventos discretos mediante

un diagrama de flujo de trabajo. La conceptualización del modelo de simulación es el paso más significativo para poder implementar un modelo de simulación que defina fielmente el problema.

La Tabla 5.1 presenta los elementos de modelado que permiten la implementación de diagramas.

Tipo de elemento	Figura
Inicio	
Fin	
Selección exclusiva	
Actividad	
Simulación	

Tabla 5.1: Elementos principales de modelado de la interfaz gráfica de SIGHOS.

Por otro lado, la Tabla 5.2 muestra los elementos auxiliares de modelado.

Tipo de elemento	Figura
Mover elementos	
Herramienta selección	
Modificar espacio de trabajo	
Conectar elemento	

Tabla 5.2: Elementos auxiliares de la interfaz gráfica de SIGHOS.

La Figura 5.3 muestra el diseño final que tiene la herramienta de modelado de simulaciones SIGHOS.

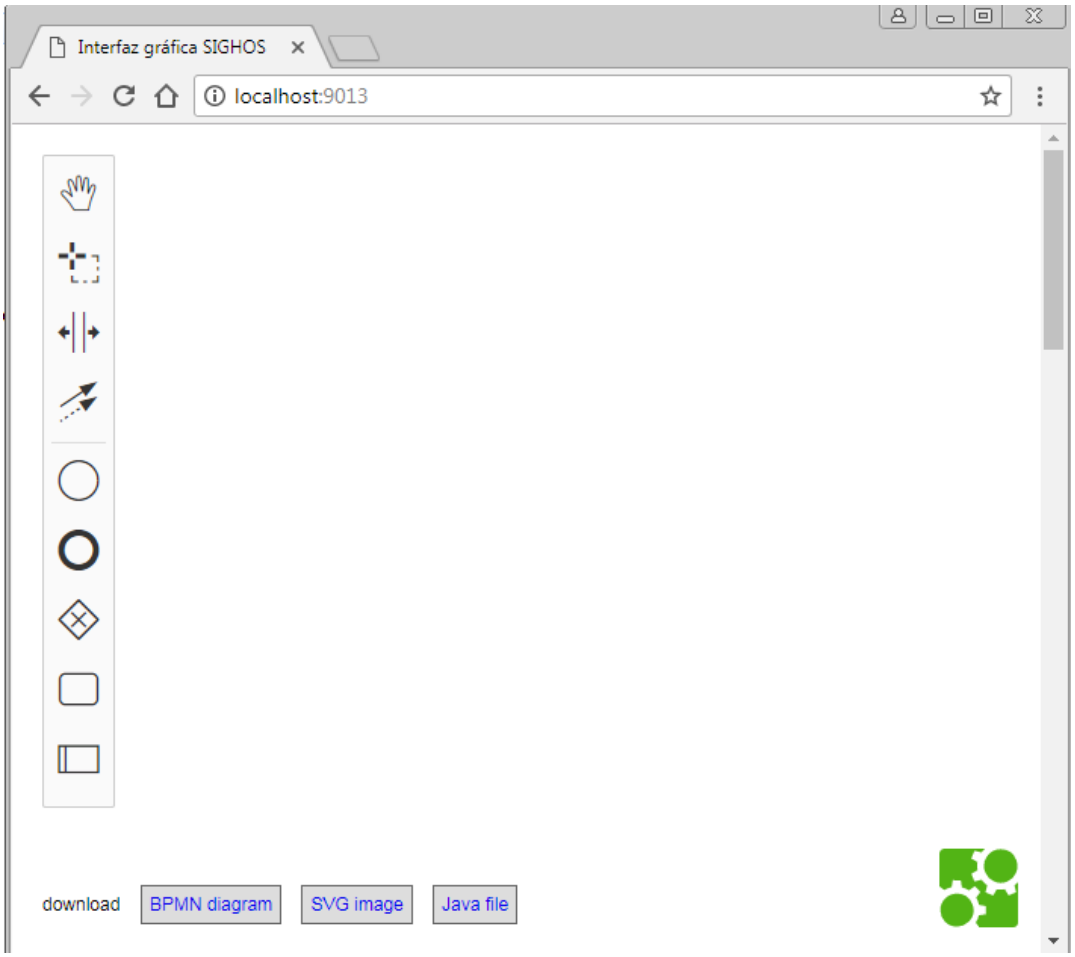


Figura 5.3: Imagen de la interfaz grafía de modelado de diagramas de simulación.

5.4 Generación de los elementos de modelado

Para obtener la herramienta de modelado de diagramas para la librería SIGHOS en primer lugar fue necesario eliminar los elementos innecesarios de la interfaz gráfica y modificar los elementos de la librería BPMN-JS. Para

ello fue necesario modificar la estructura y el código de la librería BPMN-JS, y así mismo, su comportamiento.

5.4.1 Descripción BPMN-JS

La librería BPMN-JS se basa en dos bibliotecas: diagram-js y bpmn-moddle, tal como muestra el diagrama de la Figura 5.4.1.

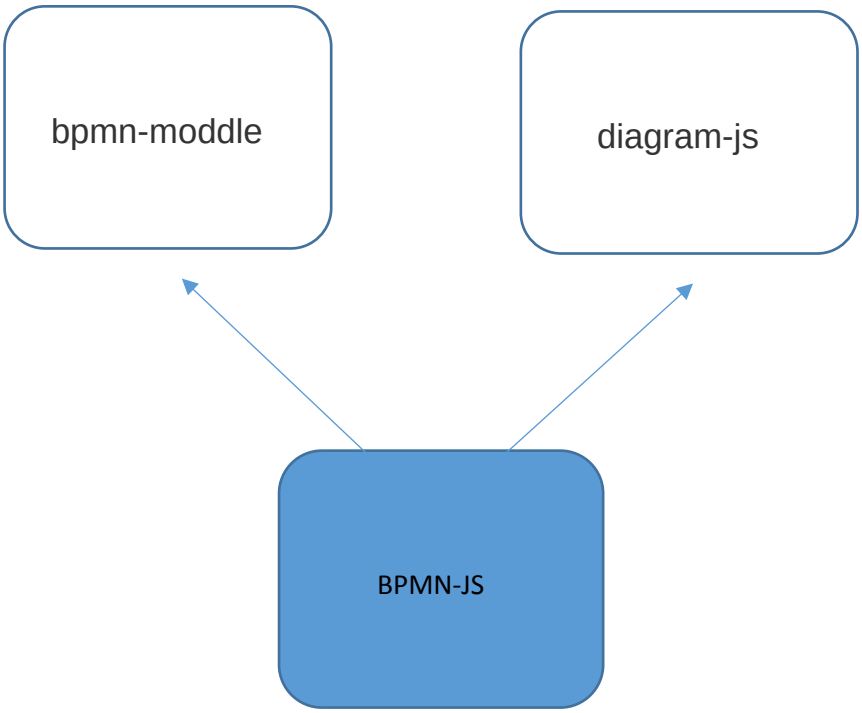


Figura 5.4.1: Diagrama de arquitectura de la librería BPMN-JS.

La biblioteca diagram-js es utilizada para el modelado gráfico de los elementos y la conexiones. Nos permite la interacción con los elementos gráficos y a su visualización y renderizado en pantalla.

El funcionamiento de diagram-js se basa en las siguientes clases:

- ElementRegistry: reconoce todos los elementos que forman el diagrama de flujo y permite su representación gráfica.

- ElementFactory: permite crear las formas y conexiones en función del modelo de datos de diagrama-js.

- GraphicsFactory: clase encargada de generar las representaciones gráficas de las formas y conexiones. También proporciona los renderizadores que definen el aspecto y el movimiento de los elementos dibujados
- Canvas: permite agregar y eliminar los elementos al modelo gráfico. También posibilita desplazar y acercar el diagrama de flujo.

A su vez, la biblioteca bpmn-moddle comprende la notación BPMN 2.0 y permite obtener y generar documentos XML [18] compatibles con el esquema, a partir de en un árbol de objetos de JavaScript. Dado que bpmn-moddle encapsula el funcionamiento de la lógica BPMN 2.0, es posible validar el modelo generado por usuario durante su importación y modelado. Esto permite generar acciones de modelado o advertencias de error, así como mensajes informativos para el usuario.

Por su parte, bpmn-moddle se apoya en las siguientes clases:

- Moddle: permite definir los meta-modelos en JavaScript de forma precisa.

- Moddle-xml: permite generar y leer documento en fomaro XML basados en los meta-modelos en JavaScript definidos en la clase Moddle.

5.4.1 Eliminar elementos innecesarios.

El conjunto de elementos de la interfaz que se muestra por pantalla se especifica en el fichero "PaletteProvider.js". Este fichero contiene las funciones JavaScript necesarias para representar los elementos en el navegador. Modificando dichas funciones se consiguió eliminar los elementos BPMN irrelevantes para nuestro diseño.

Teniendo en cuanta la estructura escalable de BPMN-JS y de cara a futuras modificaciones, se ha evitado modificar la función "createParticipant" la cual se encarga de crear los elementos del diagrama. Para eliminar los elementos prescindibles se ha utilizado una hoja de estilos "elementos.css" para ocultar la visibilidad de estos elementos.

elementos.css

```
/* Permite ocultar los elementos de modeladoprescindibles en la interfaz*/

div.entry.bpmn-icon-data-object,
div.entry.bpmn-icon-data-store,
div.entry.bpmn-icon-subprocess-expanded{

    display:none !important;
}
```

Figura5.4: Hoja de estilos elementos.css

La función “createParticipant” obtiene información de los elementos del diagrama, la cual se pasará al fichero “index.html”. Para evitar que los títulos de los elementos se obtengan de los elementos definidos en la herramienta, se ha evitado que los títulos se obtengan en la función “createAction” y se han pasado por parámetro directamente en la función “createParticipant”. De esta forma se especifican los títulos de cada elemento de una manera familiar a la librería SIGHOS.

5.4.2 Modificar el comportamiento del sistema de conexión y modificación de los elementos.

La librería BPMN-JS consta de un sistema de configuración individual de los elementos, el cual es innecesario para la librería SIGHOS. Para evitar confusión, algunas de estas opciones de configuración fueron eliminadas de la herramienta. Por un lado, el modelo BPMN consta de diferentes tipos de eventos, los cuales son innecesarios para nuestro propósito, exceptuando el evento de inicio sin condición. Con el resto de elementos ocurre lo mismo, se fue eliminando los distintos tipos de elementos presentes en el modelo BPMN, pero innecesarios para el modelado de SIGHOS. Para ellos se modificó el fichero “ReplaceOptions.js”, donde cada elemento tiene una lista de elementos candidatos para ser enlazados, de la cual fueron eliminados los elementos innecesarios.

También fue preciso eliminar la opción que tiene cada elemento de adjuntar otros elementos, previamente eliminados de la herramienta por su

falta de funcionalidad para la librería SIGHOS. Para ello se modificó el fichero "ContextPadProvider.js", modificando la nomenclatura de cada elemento y eliminando otras por completo.

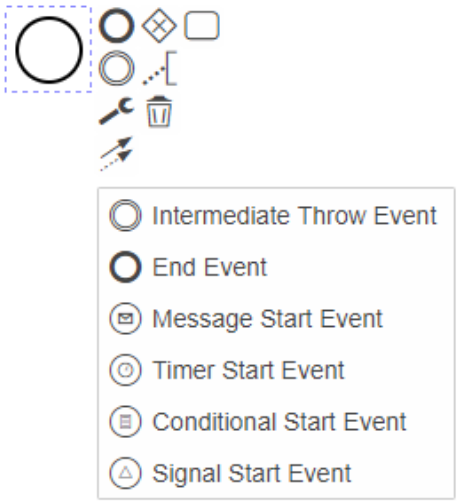
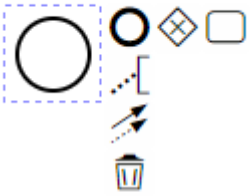
BPMN-JS	Herramienta gráfica SIGHOS
	

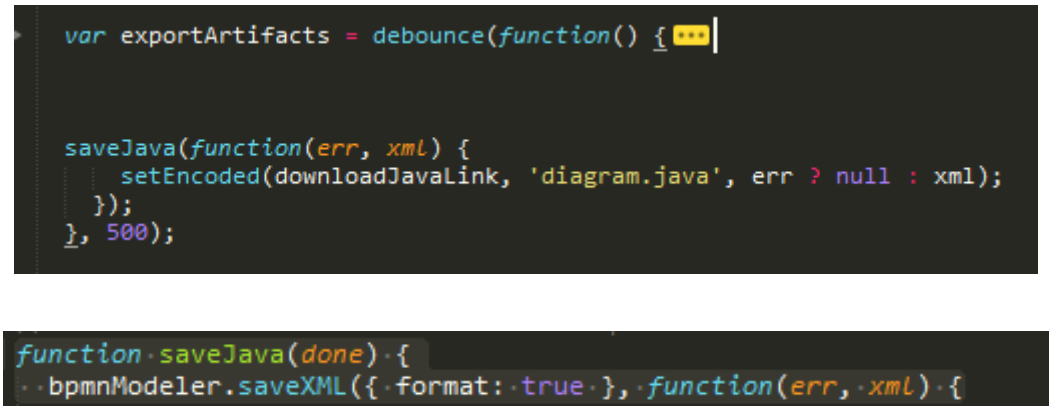
Tabla 5.5: Diferencia entre el sistema de conexión del elemento de inicio en la librería BPMN-JS y en la herramienta gráfica SIGHOS.

5.4.3 Generación del fichero Java con las clases SIGHOS.

La descarga del fichero Java que contiene el modelo implementado con las clases de la librería SIGHOS, se realiza mediante un botón dinámico. Es decir, el botón de descarga implementa la funcionalidad de permanecer inactivo mientras no se arrastre un elemento al área de trabajo.

Cabe destacar, que la generación del modelo de simulación a partir de los elementos añadidos al diagrama de flujo se actualiza con cada cambio en

el diagrama. De esta forma es posible obtener nuestro fichero con clases Java, luego realizar una modificación en el modelo y proceder de nuevo a obtener un fichero con las clases de la librería SIGHOS, sin necesidad de generar un nuevo modelo.



```
var exportArtifacts = debounce(function() {...

saveJava(function(err, xml) {
    setEncoded(downloadJavaLink, 'diagram.java', err ? null : xml);
  });
}, 500);
```

```
function saveJava(done) {
  bpmnModeler.saveXML({ format: true }, function(err, xml) {
```

Figura 5.6: Implementación del botón de descarga del fichero Java, contenedor de las clases SIGHOS.

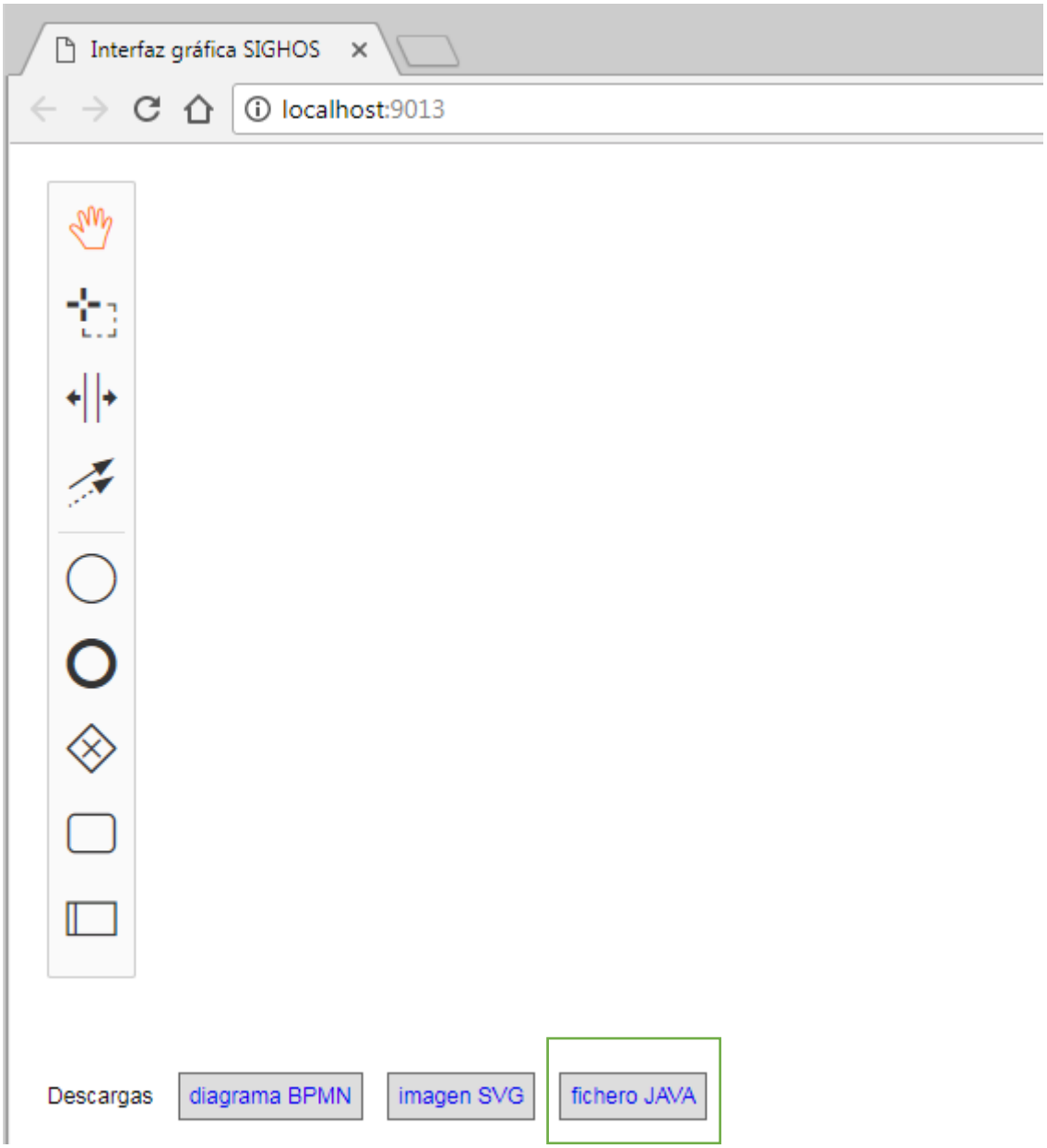


Figura 5.7: En la parte inferior, botón de descarga del fichero con las clases Java.

5.5 Generación de las clases de la librería SIGHOS.

Para generar las clases JAVA de la librería SIGHOS a partir del diagrama de flujo, es necesario realizar un parseo de los distintos elementos del XML generado que describe el diagrama. En concreto se analizan las etiquetas del fichero XML y se relacionan entre sí, hasta obtener la lógica necesaria para derivar las clases que definan la simulación SIGHOS.

5.5.1 Función de parseo y librerías necesarias para la simulación SIGHOS.

Al comienzo del fichero .java se incluyen los paquetes y clases necesarias para llevar a cabo la simulación. Para ello se ha generado la función "saveJava", la cual obtiene la información del modelo a partir del XML que se genera por la librería BPMN-JS y almacena dicha información en una cadena de texto "xmlDoc". El elemento de parseo se obtiene instanciando la clase "DOMParser", clase JavaScript nativa, que permite acceder a los elementos del modelo gráfico mediante sus respectivos árboles de etiquetas.

```
function saveJava(done) {
  bpmnModeler.saveXML({ format: true }, function(err, xml) {
  var i;
  var parser = new DOMParser();
  var xmlDoc = parser.parseFromString(xml,"text/xml");
  var texto = "";



  var librerias = "package es.ull.iis.simulation.examples;" + "\r\n" +
                  "\r\n" +
                  "import es.ull.iis.simulation.condition.Condition;"  + "\r\n" +
                  "import es.ull.iis.simulation.condition.TrueCondition;"  + "\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" ;

  texto += librerias;
```

Figura 5.8: Clase “saveJava” y las librerías de la simulación SIGHOS.

5.5.2 Extracción de la clase de simulación.

Se define la clase principal de simulación, la cual contendrá todos los componentes de nuestro modelo. Para ello se analiza el diagrama en busca de la etiqueta "bpmn2:participant" y luego se obtiene en dicha etiqueta el contenido del atributo “name”, el cual contiene el nombre de la simulación.

```
var java = xmlDoc.getElementsByTagName("bpmn2:participant");
  for (i = 0; i < java.length; i++){
      texto += "public class " + java[i].getAttribute('name') +
               " (int id, TimeUnit unit, long startTs, long endTs) {"  + "\r\n" +
               "\t" + "super(id," + "\"" +
               java[i].getAttribute('name') + "\"" + ", unit, startTs, endTs);";
  }
```

Figura 5.9: Código para obtener la clase que engloba de simulación.

Después de la generación de la clase que engloba la simulación, se procede a añadir los recursos y las tablas de trabajo del experimento. Dado que no es posible detallar en el modelo gráfico los periodos de tiempo de disponibilidad y de trabajo de los recursos, así como la especificación de los grupos de trabajo, dichas líneas aparecen comentadas, se detalla al usuario la necesidad de implementarlas. Lo mismo ocurre con la descripción de los elementos y con el generador automático, el cual los instantes de creación de los elementos.

5.5.3 Extracción de las actividades

Las actividades se describen dentro del modelo de simulación como las tareas que son desarrolladas por los elementos. Las actividades son definidas por uno o varios equipos de trabajo en los que se realiza dicha actividad.

Para obtener las actividades especificadas en el modelo gráfico, se analiza el diagrama en busca de la etiqueta "bpmn2:task" y se accede al contenido del atributo “name”, donde se almacena el nombre de la actividad.

```
128   var activityFlow = xmlDoc.getElementsByTagName("bpmn2:task");
129     for (i = 0; i < activityFlow.length; i++){
130       texto += "ActivityFlow act" + activityFlow[i].getAttribute('name') + "= new ActivityFlow(this, \"" +
131       activityFlow[i].getAttribute('name') + "\");\r\n";
132     }
```

Figura 5.10: Código para obtener las actividades.

5.5.4 Extracción de los flujos de selección.

Los flujos de selección permiten el paso de una actividad a otra durante el transcurso de la simulación. De esta forma el transcurso de la simulación pasa de la forma establecida por las distintas actividades hasta el estado de fin de la simulación. Los flujos que se pueden emplear en el diagrama son el flujo de selección exclusiva (ExclusiveChoiceFlow) y el flujo de selección múltiple (MultiChoiceFlow).

Para obtener las flujos de selección exclusiva especificadas en el modelo gráfico, se analiza el diagrama en busca de la etiqueta "bpmn2:exclusiveGateway" y se accede al contenido del atributo “name", donde se almacena el nombre de la selección exclusiva en concreto.

```
var exclusiveGateway = xmlDoc.getElementsByTagName("bpmn2:exclusiveGateway");

for (i = 0; i < exclusiveGateway.length; i++){
  texto += "ExclusiveChoiceFlow f" + exclusiveGateway[i].getAttribute('name') +
  " = new ExclusiveChoiceFlow(this);\r\n";
}
```

Figura 5.11: Código para obtener los flujos de selección exclusiva.

Para obtener los flujos de selección multicondicional se repite el mismo procedimiento que para las selecciones exclusivas, pero implementando la

búsqueda para la etiqueta "MultiChoiceFlow". Luego se accede al contenido del atributo “name” para obtener el nombre de la decisión.

```
var parallelGateway = xmlDoc.getElementsByTagName("bpmn2:parallelGateway");

for (i = 0; i < parallelGateway.length; i++){
  texto += "MultiChoiceFlow f" + parallelGateway[i].getAttribute('name') +
  " = new MultiChoiceFlow(this);\r\n";
}
```

Figura 5.12: Código para obtener los flujos de selección multicondicional.

5.5.5 Extracción de la relación entre pasos del flujo.

Para encadenar los pasos del flujo en la simulación se requiere la utilización del método “link”, implementado en la librería SIGHOS. Para el caso de las selecciones exclusiva, el orden de vinculación entre los pasos es significativo. Para este tipo de selección, el elemento involucrado en la simulación seguirá el camino del flujo que primero se cumpla, prescindiendo así del resto de caminos alternativos.

Para extraer los conexiones entre los pasos de flujo del XML generado es necesario analizar la etiqueta "bpmn2:sequenceFlow". Uno de los datos que contiene esta etiqueta es el código único que referencia a los pasos que son encadenados. Este código también es almacenado en la declaración de cada paso en el documento XML. Esto permite relacionar la definición de cada

encadenamiento entre pasos, con la etiqueta contenedora de los atributos que almacenan los datos descriptivos del paso encadenado.

```
<bpmn2:task id="Task_1p69ktf" name="Consulta">
  <bpmn2:incoming>SequenceFlow_09ej5xv</bpmn2:incoming>
  <bpmn2:incoming>SequenceFlow_0wxf63b</bpmn2:incoming>
  <bpmn2:outgoing>SequenceFlow_0mdvtbx</bpmn2:outgoing>
</bpmn2:task>

<bpmn2:sequenceFlow id="SequenceFlow_09ej5xv" sourceRef="StartEvent_1" targetRef="Task_1p69ktf" />
<bpmn2:exclusiveGateway id="ExclusiveGateway_1t22dhm" name="operar">
  <bpmn2:incoming>SequenceFlow_0mdvtbx</bpmn2:incoming>
  <bpmn2:outgoing>SequenceFlow_0wc7tjb</bpmn2:outgoing>
  <bpmn2:outgoing>SequenceFlow_1vuglb7</bpmn2:outgoing>
</bpmn2:exclusiveGateway>

<bpmn2:sequenceFlow id="SequenceFlow_0mdvtbx" sourceRef="Task_1p69ktf" targetRef="ExclusiveGateway_1t22dhm" />
<bpmn2:task id="Task_0xzn895" name="Quirófano">
  <bpmn2:incoming>SequenceFlow_0wc7tjb</bpmn2:incoming>
  <bpmn2:outgoing>SequenceFlow_0wxf63b</bpmn2:outgoing>
</bpmn2:task>

<bpmn2:sequenceFlow id="SequenceFlow_0wc7tjb" name="SI" sourceRef="ExclusiveGateway_1t22dhm" targetRef="Task_0xzn895" />
<bpmn2:endEvent id="EndEvent_173xai6" name="Fin">
  <bpmn2:incoming>SequenceFlow_1vuglb7</bpmn2:incoming>
</bpmn2:endEvent>
<bpmn2:sequenceFlow id="SequenceFlow_1vuglb7" name="NO" sourceRef="ExclusiveGateway_1t22dhm" targetRef="EndEvent_173xai6" />
<bpmn2:sequenceFlow id="SequenceFlow_0wxf63b" sourceRef="Task_0xzn895" targetRef="Task_1p69ktf" />
```

Figura 5.13: Extracto del documento XML dónde se especifican los pasos “Consulta”, “Quirófano” y “operar”, encadenados entre sí.

```
var lista_conectores = [];
var conectores_xml = xmlDoc.getElementsByTagName("bpmn2:sequenceFlow");

for (i = 0; i < conectores_xml.length; i++){
  var conector = {Name:null, Tag1:null, Tag2:null, Tag_name1:null, Tag_name2:null, Tag_type1:null, Tag_type2:null};
  conector.Name = conectores_xml[i].getAttribute('name');
  conector.Tag1 = conectores_xml[i].getAttribute('sourceRef');
  conector.Tag2 = conectores_xml[i].getAttribute('targetRef');
  lista_conectores.push(conector);
}

var todos = xmlDoc.getElementsByTagName("*");
for (i = 0; i < lista_conectores.length; i++){

    for (var j = 0; j < todos.length; j++){

      if (todos[j].getAttribute('id') == lista_conectores[i].Tag1){
        lista_conectores[i].Tag_name1 = todos[j].getAttribute('name');
        lista_conectores[i].Tag_type1 = todos[j].nodeName;
      }

      if (todos[j].getAttribute('id') == lista_conectores[i].Tag2){
        lista_conectores[i].Tag_name2 = todos[j].getAttribute('name');
        lista_conectores[i].Tag_type2 = todos[j].nodeName;
      }
    }
}
```

Figura 5.14: Párrafo para extraer del documento XML los datos necesarios sobre el encadenamiento entre pasos.

Los datos de los encadenamientos que se extraen del documento XML para implementar el método "link" de la simulación SIHGOS se realizan entre las siguientes conexiones entre pasos:

- Entre una actividad y un elemento de selección.
- Entre una actividad y otra actividad.
- Entre una selección y una actividad.

```
for (i = 0; i < lista_conectores.length; i++){

  if (lista_conectores[i].Tag_type1 == "bpmn2:task"){

      if (lista_conectores[i].Tag_type2 == "bpmn2:exclusiveGateway" || lista_conectores[i].Tag_type2 == "bpmn2:parallelGateway"){
      //  actAppointment.link(fRequireSurgery);
        texto += "act" + lista_conectores[i].Tag_name1 + ".link(f" + lista_conectores[i].Tag_name2 + ");\r\n";
      }

      if (lista_conectores[i].Tag_type2 == "bpmn2:task"){
       // actSurgery.link(actAppointment);
        texto += "act" + lista_conectores[i].Tag_name1 + ".link(act" + lista_conectores[i].Tag_name2 + ");\r\n";
      }
  }


  if ((lista_conectores[i].Tag_type1 == "bpmn2:exclusiveGateway" || lista_conectores[i].Tag_type1 == "bpmn2:parallelGateway")
      && lista_conectores[i].Tag_type2 == "bpmn2:task"){
      //fRequireSurgery.link(actSurgery, cond);
      texto += "f" + lista_conectores[i].Tag_name1 + ".link(act" + lista_conectores[i].Tag_name2 + ", cond);\r\n";
  }

}
```

Figura 5.15: Párrafo para implementar los métodos "link" de la simulación.

5.6 Caso de estudio.

La visualización de los resultados obtenidos tras analizar el diagrama implementado mediante la herramienta se hace a través de la descarga del fichero "simulacion.java".

A continuación, se muestra el ejemplo de un diagrama de flujo basado en un hospital y el fichero generado a partir del diagrama, el cual contiene las clases Java de la simulación SIGHOS. El diagrama simula el paso de los pacientes entre consulta y quirófano hasta su alta. Nuestra simulación consta de 6 médicos, 3 de los cuales enfermeros y 3 cirujanos.

Los médicos y enfermeros disponen del mismo horario laboral: comienza a las 8:00 y dura 7 horas. Las cirugías se realizan entre las 11:00 y las 14:00.

Cada día a las 8:00 llegan 20 pacientes al hospital. Tras pasar la consulta, se determina de forma aleatoria el 5% de los pacientes que precisa de una intervención quirúrgica. Una vez realizada la intervención, el paciente pasará de nuevo por consulta para determinar su alta o una nueva intervención.

Se precisa de un médico para el paso de la consulta. Por otro lado, para cada intervención quirúrgica es necesaria la presencia de 2 cirujanos y 1 enfermero, o en su defecto, la intervención se podrá realizar con 1 cirujano y un enfermero, pero con una duración de la intervención mayor a lo normal.

Teniendo en cuenta los componentes mencionados anteriormente, la simulación dispone de:

- Tipos de elemento: los pacientes. Disponemos de un solo tipo de pacientes.

- Tipos de recursos: médico, cirujano y enfermero.

- Recursos: 3 enfermeros y 6 médicos (3 de los médicos son del tipo de recurso cirujano).

- Grupos de trabajo: Para la realización de las cirugías disponemos de dos posibles grupos de trabajo: 2 cirujanos y 1 enfermero, y por otro lado, 1 cirujano y 1 enfermero. Para las consultas sólo es necesaria la presencia de 1 cirujano.

- Actividades: Consulta y Quirófano.

- Generación de elementos: cada 10 minutos se genera un paciente, dentro del horario de trabajo del hospital.

- Flujo de trabajo: Se describe mediante la interfaz de modelado de diagramas de flujo en la siguiente imagen:

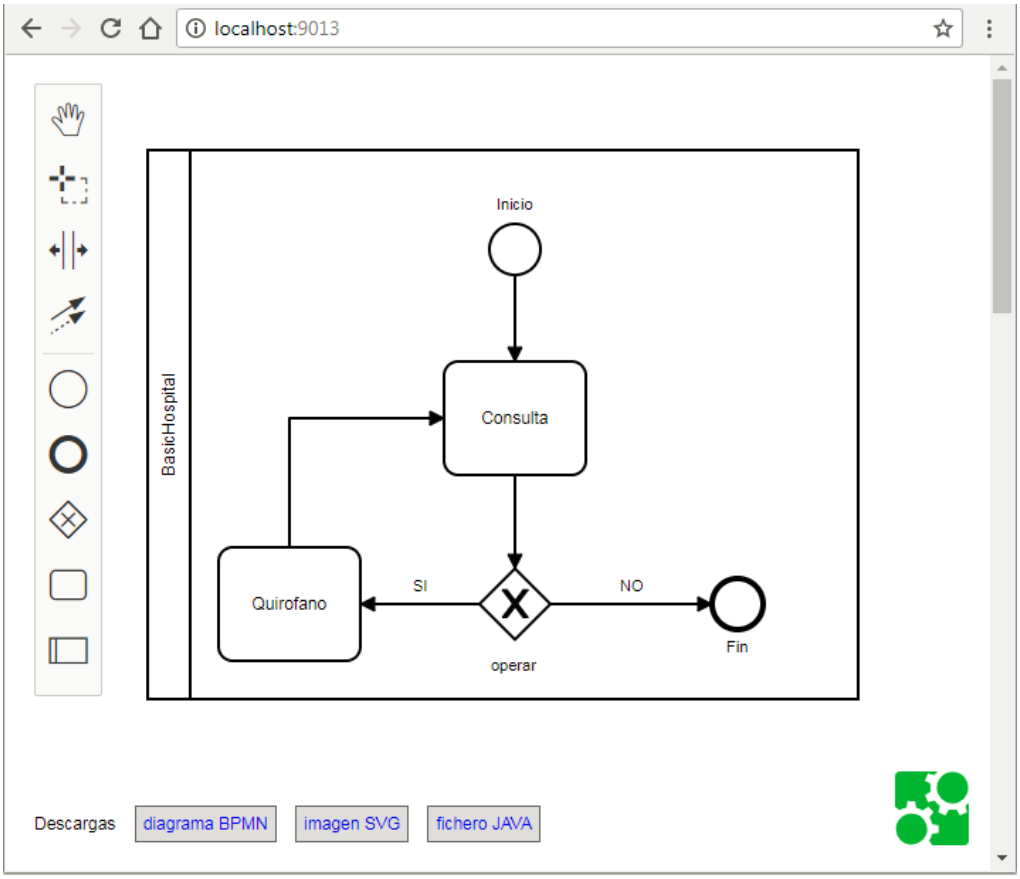


Figura 5.16: Diagrama de flujo basado en hospital.

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

import es.ull.iis.simulation.condition.Condition;
import es.ull.iis.simulation.condition.TrueCondition;
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 BasicHospital (int id, TimeUnit unit, long startTs, long endTs) {
    super(id,"BasicHospital", unit, startTs, endTs);// Define the model

// Please add Element Types
// ElementType etPatient = new ElementType(this, "Patient");

// Please add Resource types
// ResourceType rtDoctor = new ResourceType(this, "Doctor");

// Please add Resources
// Resource resDoctor1 = new Resource(this, "Dr. J. Martin");

// Please define the work timetables
// ... for doctors and nurses (starting at 8:00)
// SimulationPeriodicCycle docCycle = SimulationPeriodicCycle.newDailyCycle(unit, 8 * 60);

// Please define the roles of the resources
// resDoctor1.addTimeTableEntry(docCycle, 7 * 60, rtDoctor);

// Create the activities: Automatically generated from NOMBRE_APLICAION
ActivityFlow actConsulta= new ActivityFlow(this, "Consulta");
ActivityFlow actQuirofano= new ActivityFlow(this, "Quirofano");

// Please define the workgroups
// WorkGroup wgAppointment = new WorkGroup(this, rtDoctor, 1);

// Please assign duration and workgroups to activities
// actAppointment.addWorkGroup(0, wgAppointment, TimeFunctionFactory.getInstance("UniformVariate", 7, 10));

// Automatically generated from NOMBRE_APLICAION
ExclusiveChoiceFlow foperar = new ExclusiveChoiceFlow(this);

// Please replace this condition by the desired one
Condition cond = new TrueCondition();

// Automatically generated from NOMBRE_APLICAION
actConsulta.link(foperar);
foperar.link(actQuirofano, cond);
actQuirofano.link(actConsulta);

// Please define element generator
// new TimeDrivenElementGenerator(this, 1, etPatient, actAppointment, docCycle);
}
}
```

Figura 5.17: Fichero “simulacion.java” generado a partir del diagrama de flujo del hospital.

Capítulo 6

Conclusiones y líneas futuras

Con este proyecto se ha logrado fijar unas bases sólidas y una primera versión de una interfaz gráfica que permita el aprovechamiento del potencial de la librería SIGHOS.

Durante la implementación de este proyecto, se han adquirido conocimientos sobre diversas herramientas de diseño de procesos, quedando claro la importancia y la demanda de estos por los distintos organismos. También ha sido de gran interés trabajar por primera vez utilizando tecnologías web, desarrollando una herramienta totalmente multiplataforma.

Cabe destacar, que la adaptación de la librería BPMN-JS a las necesidades del proyecto ha sido la parte que más compleja en el desarrollo de este proyecto.

De cara a proyectos futuros, será interesante la implementación de una nueva capa de la interfaz que permita rellenar los datos de la simulación que no son extraíbles del modelo gráfico. De esta forma se permitirá la generación completa de una simulación que podrá ser interpretada por la librería SIGHOS.

Así mismo tras el desarrollo de este trabajo, surgen distintos puntos para continuar realizando progresos en la herramienta:

- Montar la interfaz implementada en un proyecto web, permitiendo que la librería SIGHOS acceda al código generado por la interfaz gráfica.

- Añadir otra vista a la interfaz que permita al usuario cuantificar las diversas variables de la simulación. Algunas de estas serían los tiempos requeridos para finalizar cada tarea, los horarios de trabajo de cada recurso, etc.

Capítulo 7

Summary and Conclusions

With this project it has been possible to establish solid foundations and a first version of a graphical interface that allows the exploitation of the potential of the SIGHOS library.

During the implementation of this project, it has been acquired knowledge about various process design tools, and it has become clear the importance and demand of these tools by the different organisms. Also, it has been of great interest to work for the first time using web technologies, developing a totally multi-platform tool.

It should be noted that the adaptation of the BPMN-JS library to the needs of the project has been the most complex part of the development of this project.

For future projects, it will be interesting to implement a new layer of the interface that allows filling the simulation data that are not extractable from the graphic model. This will allow the complete generation of a simulation that can be interpreted by the SIGHOS library.

As well, after the development of this work, different ideas emerge to continue making progress in the tool:

- Mounting the implemented interface in a web project, allowing the SIGHOS library to access the generated code by the graphic interface.

- Adding another view to the interface that allows the user to quantify the various simulation variables. Some of these would be the times required to complete each task, the work schedules of each resource, etc.

Presupuesto.

En este capítulo se procede a especificar los costes de las herramientas utilizadas y los tiempos de trabajo. El presupuesto del desarrollo de la herramienta se deduce en función del coste de software y del cómputo de las horas trabajadas.

Descripción	Coste
200 Horas de trabajo	1500 €
Librería BPMN.JS	0 € (licencia gratuita)
Ordenador para desarrollo	1000 €
Total:	**2500 €**

Tabla 7.1: Presupuesto del proyecto

Bibliografía

[1] George S. Fishman. Discrete-Event Simulation: Modeling, Programming, and Analysis. 2013

[2] Gabriel A. Wainer, Pieter J. Mosterman. Discrete-Event Modeling and Simulation: Theory and Applications. 2016.

[3] Arthur H. M. ter Hofstede, Wil M. P. van der Aalst, Michael Adams. Modern Business Process Automation. 2014.

[4] SIGHOS: Iván Castilla Rodríguez. Explotación de los Sistemas

Multi-Núcleo para la Simulación Paralela de Eventos Discretos con Java. 2010.

[5] GUI: Graphical User Interface

https://www.britannica.com/technology/graphical-user-interface

[6] Plugin. https://en.wikipedia.org/wiki/Plug-in_(computing)

[7] Eclipse for RCP/Plugin Developers

http://www.eclipse.org/callisto/plugin-dev.php

[8] BPMN: Business Process Model and Notation

http://www.bpmn.org/

[9] BPEL: Business Process Execution Language

https://en.wikipedia.org/wiki/Business_Process_Execution_Language

[10] SOA: Arquitectura Orientada por Servicios

https://ingenieriadelsoftwareuah2015.wordpress.com/2015/03/22/arquitectura-orientada-a-servicios-soa/

[11] Bonita Studio. https://www.bonitasoft.com/

[12] GoJS https://gojs.net/latest/index.html

[13] BPMN-JS https://bpmn.io/toolkit/bpmn-js/walkthrough/

[14] Node.js https://nodejs.org/en/about

[15] NPM https://docs.npmjs.com/getting-started/what-is-npm

[16] GRUNT https://gruntjs.com/

[17] JSON: JavaScript Object Notation https://www.json.org/

[18] XML : Extensible Markup Language

https://en.wikipedia.org/wiki/XML