Simulación de procesos de negocio en BPMN 2.0 con el estándar BPSim

Lic. Juan Larrayoz

Trabajo de Tesis para
la obtención del título de
Magister en Sistemas de Información

Centro de Posgrados y Actualización Profesional en Informática, Instituto de Computación, Facultad de Ingeniería, Universidad de la República

Octubre 2015
Montevideo, Uruguay

Tutor: Dra. Ing. Andrea Delgado

Agradecimientos

En primer lugar a mi hija Sofia, quien durante sus 4 añitos de vida a tenido que saber compartir el tiempo que por derecho le correspondía con los cursos de posgrado y este trabajo de tesis.

En segundo lugar a Sabrina, mi señora quien sin duda se llevo la peor parte al tener que asumir todas las responsabilidades que por falta de tiempo no pude cumplir.

A mi tutora Andrea Delgado, quien siempre supo como guiarme y motivarme en este gran emprendimiento personal.

A todos ellos les doy las gracias por aportar su granito de arena y ayudarme a concluir este sueño finalmente convertido en realidad.

Resumen

La Gestión de Procesos de Negocio (Business Process Management, BPM) es un paradigma que ayuda a las organizaciones a mejorar su desempeño a través de la mejora continua de los procesos que la componen. BPM provee un conjunto de métodos, técnicas y herramientas que permiten modelar, simular, implementar, ejecutar y analizar un proceso de negocio. Cuando se desea implementar un nuevo proceso o rediseñar uno existente uno de los problemas a los que se enfrenta la organización es que resulta sumamente difícil predecir cual será su desempeño (performance).

La simulación de procesos (Business Process Simulation, BPS) es parte fundamental de la gestión de procesos de negocio; Le brinda herramientas a las organizaciones para diseñar y modificar sus procesos utilizando métodos estadísticos permitiendo obtener una mejor comprensión de/los comportamientos posibles en tiempo de ejecución, asociados a los escenarios de ejecución definidos. BPSim es el estándar propuesto en 2013 por la WfMC (Workflow Management Coalition) con la meta de definir una especificación para la parametrización e intercambio de datos provenientes de un escenario de simulación.

Se hace difícil encontrar en el mercado herramientas de BPS que provean completo soporte para realizar simulaciones adecuadas, y las pocas disponibles son sumamente costosas, haciéndolas poco interesantes para la mayoría de las empresas. Además estas herramientas son propietarias de cada proveedor, no existiendo la posibilidad de compartir escenarios de simulación entre diferentes implementaciones.

El objetivo general de esta tesis es comprender los conceptos involucrados detrás de la simulación de procesos, la adecuación de las definiciones del estándar BPSim y su integración con el lenguaje de notación del estándar BPMN 2.0. Para esto se analizó la adopción e implementación del estándar BPSim para la simulación de procesos de negocio basados en el estándar BPMN 2.0 desarrollando un prototipo que permite simular procesos en BPMN 2.0 con BPSim. Para construir este simulador de procesos se evaluaron una variedad de plataformas BPMS Open Source disponibles en la actualidad.

Finalmente se realizó un caso de estudio para poder apreciar como el prototipo debe de configurado y usado, y evaluar tanto la aplicabilidad como la utilidad de la propuesta.

Índice de contenido

Índice de figuras

Índice de tablas

1 Introducción

En esta sección del documento se presentan los principales conceptos a desarrollar en este trabajo de tesis. Se definen los conceptos de proceso de negocio, sistema de gestión de procesos, simulación y los estándares vigentes involucrados como ser BPMN2 y BpSim.

Siendo que esta sección es un extracto de la información contenida en el Anexo A, en caso de que el lector desee ampliar la información aquí presentada puede recurrir a dicho anexo.

1.1 Motivación

Actualmente casi toda organización puede ser descripta como sistemas de procesos de negocios que por su propia y compleja estructura exigen un análisis profundo mediante procedimientos tales como el Modelado de Procesos de Negocio y la Simulación de Procesos de Negocio (Business Process Simulation, BPS). Esta situación a provocado que desde hace un tiempo exista un marcado interés creciente por parte de las empresas desarrolladoras de software en la confección de herramientas de simulación, pasando a formar parte vital esta de muchas de las suites de BPM (Business Process Management, BPM) disponibles en el mercado.

La Gestión de Procesos de Negocio es un paradigma que ayuda a las organizaciones a mejorar su desempeño a través de la mejora continua de los procesos que la componen. BPM provee un conjunto de métodos, técnicas y herramientas que permiten modelar, simular, implementar, ejecutar y analizar un proceso de negocio.

La simulación de procesos es parte fundamental de la gestión de procesos de negocio, permite a las organizaciones diseñar y modificar sus procesos de forma más eficiente, reducir riesgos inherentes al cambio y obtener información relevante que permita tomar mejores decisiones. Las técnicas de simulación ofrecen grandes beneficios frente a otros métodos matemáticos a la hora de estudiar el comportamiento de un sistema, permiten la experimentación sin interrumpir la actividad del sistema real evitando la posibilidad de provocar daños en el mismo, además proporcionan una mejor comprensión del comportamiento del sistema y su dinámica interna, gracias a la obtención de resultados numéricos sobre ello y la posibilidad de experimentar diferentes escenarios.

Todas estas herramientas de simulación carecen de un estándar que las sustente, que permita y facilite compartir información entre ellas, llevando al usuario a caer muchas veces en una solución del tipo “Vendor lock-in” [VenLock15]. A principios del año 2013, surge como esfuerzo conjunto de diversas empresas y organizaciones la especificación de BPSim, con el objetivo de promover una estándar para la simulación de procesos de negocio.

BPSim es una iniciativa llevada acabo por el grupo BPSWG (Business Process Simulation Working Group) de WfMC (Workflow Management Coalition) [WfMC15] y uno de sus objetivos principales es fomentar una mayor adopción de la simulación de procesos de negocio dentro de la comunidad BPM a través de un enfoque guiado por estándares. Además fue concebido originalmente para ser utilizado como complemento del principal lenguaje de modelado disponible actualmente, BPMN 2 (Business Process Model and Notation) [BPMN15], permitiendo incorporar información sobre simulación a los procesos ya existentes modelados con este estándar.

En el contexto presentado surge la idea de involucrarse con el estándar BPSim para la simulación de procesos de negocio. Al momento de comenzar con este trabajo de tesis se constató que BPSim es un estándar reciente, en incipiente adopción donde la oportunidad de conjugarlo con BPMN 2.0 promete como resultado un “lenguaje” sumamente expresivo tanto para modelado como para el análisis de procesos de negocio a través de las simulación.

Se vio en esta situación una muy buena oportunidad de contribuir a la adopción y desarrollo de este nuevo estándar mediante la realización del trabajo de tesis que se presenta en este informe: “Simulación de procesos de negocio en BPMN 2.0 con el estándar BpSim”

1.2 Objetivos

El OG (objetivo general) de esta tesis es comprender los conceptos involucrados detrás de la simulación de procesos, la adecuación de las definiciones del estándar BPSim y su integración con el lenguaje de notación del estándar BPMN 2.0.

Los OE (objetivos específicos) de la tesis son:

- **OE1:** Investigar los conceptos detrás de la simulación de procesos de negocios y de los principales simuladores del mercado.
- **OE2:** Estudiar diferentes implementaciones de BPMS Open Source que implementen BPMN 2.0 para utilizar una de ellas como plataforma base para la construcción del prototipo.
- **OE3:** Analizar la integración definida en BpSim con el metamodelo del estándar BPMN 2.0 para proveer un simulador de procesos especificado con dicha notación.
- **OE4:** Desarrollar un prototipo de herramienta que permita simular un proceso especificado en BPMN 2.0 con base en el estándar BpSim y analizar los datos obtenidos.

Los resultados esperados asociados a los objetivos anteriormente descriptos son los siguientes:

- **R1:** Investigación de alternativas de simulación de procesos propuestas por diferentes proveedores con el foco en las que potencialmente brinden soporte para el estándar BpSim. Este resultado esta asociado al **OE1**.
- **R2:** Análisis de la integración del estándar BpSim con el metamodelo BPMN 2.0 para simulación de procesos de negocio, y de la extensión de modeladores BPMN 2.0 existentes con información requerida para la simulación de procesos. Este resultado esta asociado al **OE2**.
- **R3:** Prototipado de herramienta que permita simular un proceso de negocio modelado en BPMN 2.0 con base en el estándar BpSim de simulación y en herramientas de ejecución de procesos BPMN 2.0 existentes. Este resultado esta asociado al **OE4**.
- **R4:** Desarrollo de un caso de estudio que permita presentar la viabilidad de las definiciones y prototipo realizado. Este resultado esta asociado al **OE4**.

1.3 Contribuciones

El primer aporte de este trabajo de tesis es el estudio del estado del arte asociado al soporte del estándar BPSim a la simulación de procesos en BPMN 2 por parte de las herramientas de simulación disponibles en el mercado tanto propietarias como Open Source.

Dicha investigación proporciona un panorama claro de la adopción por parte del mercado de BPSim y del grado de soporte que brindan las herramientas disponibles.

El estándar para simulación de procesos de negocio es sumamente reciente, dicha especificación tiene fecha de publicación febrero del 2013, por lo que realmente se trata de algo incipiente que esta comenzando a tomar fuerza. Esta situación es fácilmente comprobable por varias razones, la primera es la calidad de la documentación oficial, si bien la especificación esta “completa” en cuanto a conceptos necesarios, la guía de implementación es eso, una simple guía a la cual le falta detalles, ejemplos y contiene varios errores / contradicciones con la especificación. Otra de las razones es la falta herramientas que implementen el estándar, a la fecha las que proveen soporte para BPSim no son más de dos, siendo este en realidad un soporte parcial del estándar, pues no se encontró ninguna herramienta que soporte al 100% la especificación.

Como BPMN en su momento, BPSim intenta constituirse en un lenguaje común para la simulación de procesos de negocio, por este motivo el segundo aporte que pretende realizar este trabajo es ayudar a difundir la adopción del estándar BPSim, que consideramos tiene gran potencial para avanzar un paso en la simulación de procesos de negocio hacia la interoperabilidad, posicionando este tipo de herramientas como parte fundamental en el ciclo de vida para la gestión de procesos.

En este sentido el tercer aporte refiere al análisis detallado de las definiciones que propone el estándar BPSim y como este se integra con un modelo de BPMN 2.0, permitiendo incorporar al lenguaje de modelado la perspectiva de simulación mediante la definición de escenarios de simulación basados en sus elementos.

Finalmente, otro aporte realizado es la contribución que implica a la comunidad Open Source el prototipo realizado, no solo a la comunidad Open Source, también a la comunidad de Activiti, de donde se tomaron los componentes base para la construcción del simulador y cuya suite de BPM no provee dentro de su batería de herramientas.

El prototipo desarrollado creemos contribuye a la comunidad proveyéndole a Activiti (una de las suite Open Source BPM mas reconocidas a nivel mundial que implementa casi completamente el estándar BPMN 2.0) de la capacidad de simulación de procesos BPMN 2.0 con el estándar BPSim.

1.4 Organización del documento

El documento está estructurado en ocho capítulos. En el primer capitulo se hace una introducción al trabajo de tesis, cual fue la motivación y los objetivos que sirvieron de marco para dicho trabajo.

En el segundo capítulo se presentan los principales conceptos y estándares involucrados. Se definen conceptos como proceso de negocio, gestión de procesos (BPM), ciclo de vida, en este último haciendo énfasis en las fases de diseño y análisis. Se presenta el estudio de sistemas de BPM o BPMS y sus principales componentes, destacando las capacidades de los módulos de modelado y simulación. También se presentan los principales conceptos detrás de la simulación de procesos, como ser tipos de simulación y diferente estilos de modelado. Se describe el estándar BPSim y en su integración con BPMN 2.0.

En el tercer capítulo se presenta un análisis exhaustivo de herramientas de modelado de procesos y de simulación, ya sea que formen parte de soluciones BPMS o sean distribuidas de

forma independiente. Se describen y evalúan herramientas propietarias y Open Source, haciendo énfasis en estas últimas.

El capítulo cuatro presenta en profundidad el análisis del prototipo, las decisiones relevantes de diseño y las funcionalidades y requerimientos a implementar en cada una de las diferentes iteraciones planificadas para su construcción.

El capítulo cinco presenta el diseño del prototipo, se destaca la selección de componentes a utilizar como punto de partida. También se da una introducción a estos componentes y una breve reseña a su arquitectura y principales componentes. Se profundiza en el análisis de Activiti, solución BPMS utilizada como base para la construcción del simulador.

En el capítulo seis se da a conocer la implementación del prototipo, se describe las dependencias de software necesarias y se detalla el funcionamiento de los principales módulos que componen la solución arquitectónica planteada.

En el capítulo siete se presenta un caso de estudio para demostrar la factibilidad de la propuesta en base al prototipo construido. Se utilizó como caso de estudio uno de los tres ejemplos presentados en el manual de implementación provisto por el estándar BPSim.

Para finalizar en el capítulo ocho se presentan las conclusiones finales del trabajo y los puntos pendientes que por diferentes motivos quedaron fuera del alcance de la tesis, pero su continuación a futuro sería por demás interesante.

2 Estado del arte

En este capítulo se introducen los elementos principales asociados a procesos de negocios y su gestión. Se definen conceptos como proceso de negocio, gestión de procesos, ciclo de vida, en este último haciendo énfasis en las fases de diseño y análisis.

También se abordará el tema de las herramientas BPMS y sus principales componentes, haciendo énfasis en los módulos que permiten modelar y simular un proceso.

2.1 Proceso de negocio

¿Que es un proceso de negocio?

Existen tantas definiciones como autores, pero a continuación presentaremos algunas de las definiciones propuestas por los autores utilizados como referencia en este trabajo de tesis.

“Un proceso de negocio consiste en un conjunto de actividades que se realizan en coordinación en un entorno organizativo y técnico . Estas actividades dan cuenta conjuntamente a un objetivo de negocio . Cada proceso de negocio es creado por una sola organización , pero puede interactuar con los procesos de negocio realizadas por otras organizaciones”.[WES07]

“Proceso de negocio se define como un flujo coordinado y estandarizado de actividades realizadas por personas o máquinas, que pueden traspasar límites funcionales o departamentales, para lograr un objetivo de negocio, que crea valor para clientes internos o externos”.[VERM09]

“La automatización de un proceso de negocio , en su totalidad o en parte, durante el cual documentos, información o tareas pasan de un participante a otro para la acción de acuerdo a un conjunto de normas de procedimiento”.[WfMC99]

Resumiendo las definiciones mencionadas, podemos inferir que básicamente un proceso de negocio es o esta compuesto por un conjunto de actividades que son realizadas por una máquina de forma automática o por personas en una organización. Estas actividades son completadas de forma coordinada contribuyendo a cumplir un objetivo de negocio.

A lo largo del tiempo han existido diversas notaciones gráficas para modelar proceso de negocio. Actualmente la mas utilizada es BPMN (Business Process Model and Notation) en su versión 2.0. El objetivo principal de BPMN es proporcionar una notación estandarizada que sea entendible por los usuarios tanto de de negocio como técnicos. BPMN 2.0 tiene la particularidad que además de permitir diagramar un proceso de negocio, también provee la especificación para que pueda ser ejecutado por un motor de procesos, definiendo un formato de intercambio en XML.

En la Figura 1 podemos ver un diagrama de proceso modelado utilizando la notación BPMN 2.0.

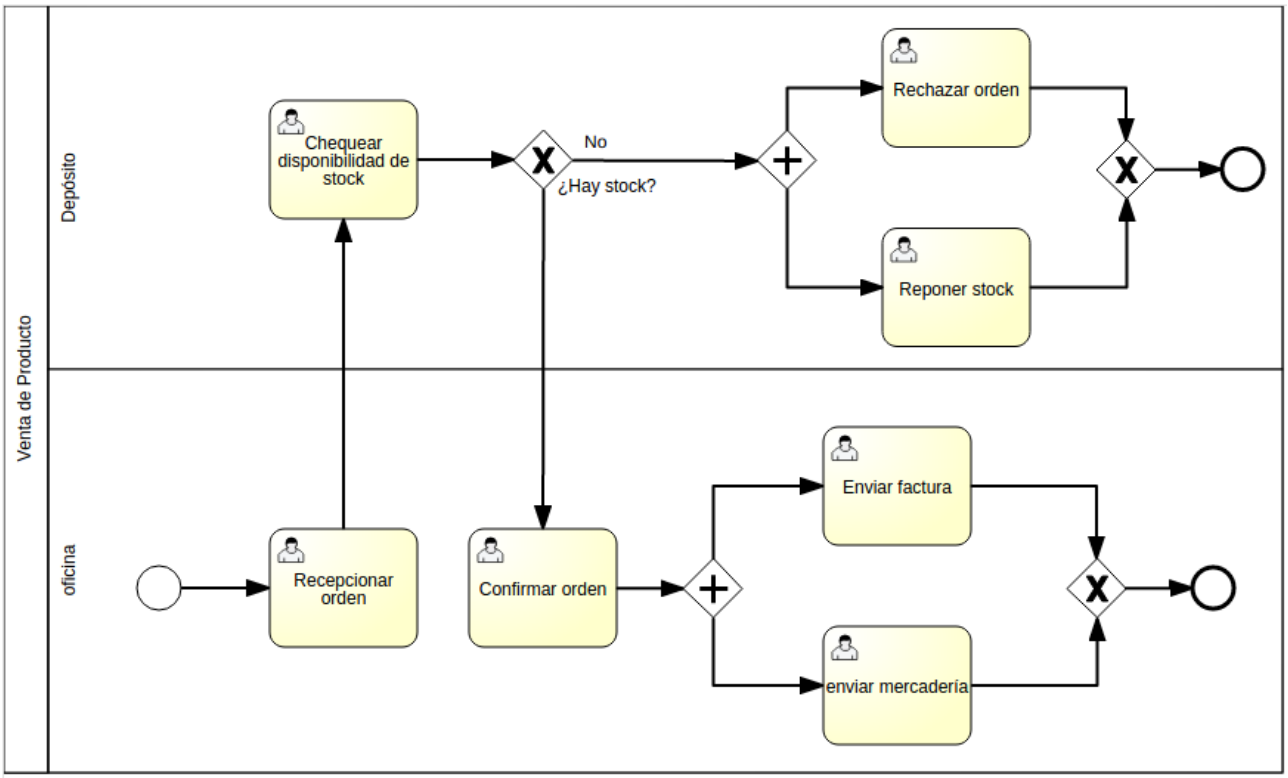


Figura 1: Modelo en BPMN 2.0 del proceso de venta de producto de [WES12]

El proceso comienza cuando se recibe una orden de compra, luego se pasa verificar la existencia en stock del o los productos que componen la orden. En el caso de que no exista stock suficiente se rechaza la orden y simultáneamente se repone el stock realizando las acciones que correspondan, y luego se finaliza el proceso, en caso contrario se confirma la orden, y de forma paralela se envía la factura y se envía la mercadería. Una vez finalizadas estas dos tareas se finaliza el proceso.

Como se puede apreciar en la Figura 1 las tareas “Recepcionar orden” y “Chequear disponibilidad de stock” se encuentran en dos lanes o carriles separados, la primera se encuentra en el lane Oficina y la segunda en el lane Depósito. Un lane se asocia a un rol o participante del proceso de negocio encargado de desempeñar las tareas que este contenga. También es posible apreciar en el diagrama otro elemento que se conoce como pool (Venta de producto), un pool contiene o delimita un proceso de negocio. Generalmente el pool se identifica con el nombre del proceso de negocio y los lanes con los roles que van a desempeñar las tareas que contienen.

De este ejemplo se desprenden dos conceptos medulares en BPM como son “modelo de proceso de negocio” e “instancia de proceso de negocio”.

El modelo es lo que esta representado en la Figura 1, que actúa como “plantilla” o template para todas las ejecuciones posibles del proceso, en cambio cada orden procesada siguiendo el modelo será una instancia particular del modelo con datos específicos asociados.

Según las definiciones dadas en [WES07], un modelo de proceso consiste en un diagrama

compuesto de actividades y sus relaciones o restricciones. Una instancia de proceso representa un caso en particular en la operativa del negocio de una empresa, y se compone de instancias de las actividades que componen el modelo.

2.2 Gestión de procesos (Business Process Management)

Una vez identificados los procesos de negocio estos deben ser gestionados. Podemos definir a la gestión de proceso de negocio (Business Process Management, BPM) de la siguiente forma:

"La gestión de procesos de negocio abarca conceptos, métodos y técnicas para dar soporte al diseño, administración, configuración, ejecución y análisis de proceso de negocio". [WES07]

"Los servicios y herramientas que apoya la gestión de procesos (por ejemplo, análisis, definición, elaboración, monitoreo y administración), incluido el apoyo para la interacción a nivel de aplicación y usuarios". [BPMNSpec11]

Las actividades o tareas que se desarrollan en BPM se organizan en etapas o fases, cada una con un enfoque específico. Estas fases constituyen lo que se conoce como **Ciclo de vida BPM.**

Existen diversos diagramas de ciclo de vida BPM propuestos por diversos autores, la mayoría de ellos concuerdan en muchas de las fases que los componen y todos ellos en que es un esquema cíclico, esto quiere decir que una vez que se llego a la última fase se vuelve a comenzar nuevamente.

Uno de los diagramas mas aceptados y difundidos es el propuesto por Weske, en el cual propone cuatro etapas para el ciclo de vida de un proceso de negocio, como se puede apreciar en la Figura 2.

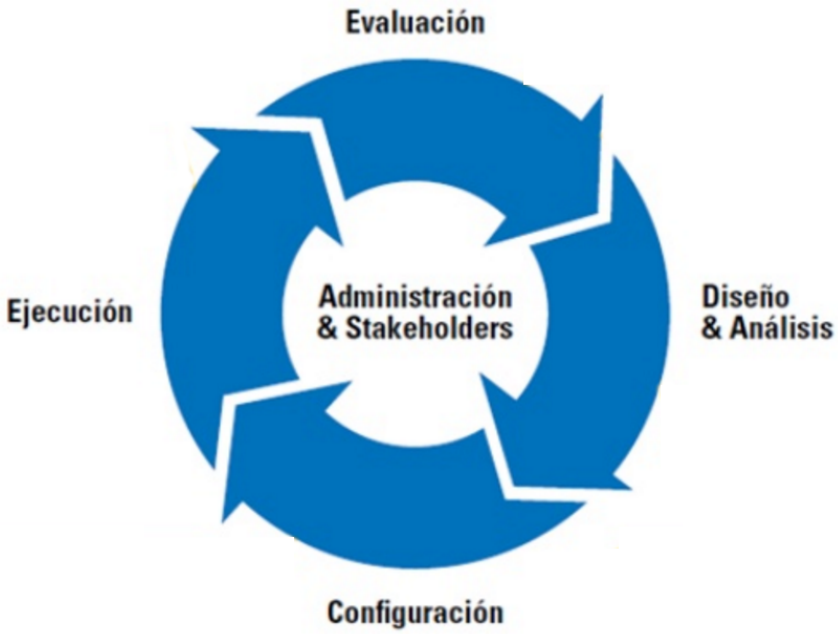


Figura 2: Ciclo de vida BPM de [WES07]

A continuación se presenta la definición de actividades en cada fase según lo propuesto por Weske.

Análisis y Diseño

En esta etapa se identifican los procesos de negocio, luego son revisados y representados como modelos de procesos de negocio. El modelado de procesos de negocio mediante el uso de notaciones gráficas como puede ser BPMN facilita la comunicación y compresión de los modelos por parte de todos los interesados. En esta fase se emplean técnicas de modelado, verificación, simulación y validación.

Las técnicas de simulación se utilizan como soporte para las tareas de validación debido a que experimentar y analizar posibles comportamientos del modelo no deseados en las fases iniciales, da lugar a correcciones tempranas minimizando el impacto que podrían llegar a tener si se detectan en fases posteriores.

Configuración

Luego que se modeló, validó y verificó el modelo es necesario implementarlo. Existen diferentes formas de implementación, la mas básica es mediante un conjunto de reglas y procedimientos que los empleados de la compañía deben respetar y seguir, o se puede utilizar un sistema informático para la gestión del proceso de negocio. Hoy en día el uso de un sistema para la implementación de los procesos es bastante común.

Una vez implementado, dicha implementación debe ser testeada. Este testeo se realiza con las herramientas básicas de testing que se utilizan a la hora de probar una aplicación. Una vez testeado, el sistema es puesto en producción.

Ejecución

Una vez implementado, testeado y puesto en producción es momento de crear instancias del modelo de procesos, cada vez que el proceso es ejecutado en la organización. En esta fase se desarrollan las actividades de mantenimiento y monitoreo. Se obtiene información importante sobre de la salud de las diferentes instancias, información que en etapas posteriores se utilizará para lograr optimizar el modelo.

Uno de los productos mas conocidos que se puede asociar con esta etapa son los logs de ejecución del sistema encargado de ejecutar los procesos. Estos logs contienen información detallada de por ejemplo tiempo de inicio y fin de una actividad, duración, usuario asignado, etc.

Evaluación y Optimización

En esta fase se utiliza la información recabada en la fase anterior para someter a evaluación los modelos originales y optimizar su implementación. Una forma posible de evaluarlos es utilizar técnicas de minería de procesos (Process Mining) para identificar aspectos de ejecución del modelo y su adecuación con la operativa definida.

2.3 Sistema de Gestión de procesos de Negocio

Un Sistema de Gestión de procesos de Negocio (Business Process Management Systems, BPMS) también conocido como Business Process Management Suite es:

"Un BPMS es un software genérico manejado por representaciones explicitas de procesos, utilizado para coordinar la ejecución de procesos de negocio". [WES07]

"Es la convergencia de varias tecnologías de integración, que culminan en una plataforma de TI centrada en procesos para poder entregar soluciones BPM". [VERM09]

"La tecnología que permite BPM". [BPMNSpec11]

Una de las formas mas tradicionales de dar soporte a BPM es contar con un conjunto de herramientas que permitan implementar las diferentes actividades que se llevan a cabo en las distintas fases del ciclo de vida.

Hacer que un modelo se convierta en un proceso ejecutable requiere de la interacción de diversas tecnologías, las cuales deben formar parte del BPMS. Dependiendo del autor de referencia y con la evolución del tiempo los BPMS se han vuelto cada vez mas completos involucrando cada vez mas componentes, a continuación se listan diferentes componentes que pueden encontrarse como parte de un BPMS:

- Modelador de procesos
- Motor de workflow
- Herramienta de monitoreo de procesos
- Simulación
- Motor de reglas de negocio
- Herramientas de análisis y Business Intelligence
- Herramienta de orquestación y SOA
- Herramienta para el desarrollo de formularios

Alguno de estos componentes son básicos, como ser el motor de workflows y el modelador de procesos (presente en todas las suites BPM analizadas) y otros se pueden encontrar o no dependiendo de cada suite en cuestión.

2.4 Business Process Model and Notation

El Object Managment Group (OMG) es la responsable de mantener el estándar Business Process Model and Notation (BPMN) que fue inicialmente desarrollado por la Business Process Management Initiative (BPMI).

El objetivo principal fue lograr un estándar que pueda ser fácilmente entendido e interpretado por usuarios del negocio y todos los involucrados hasta llegar a los usuarios mas técnicos de IT (Tecnologías de la Información, Information Technology). Los usuarios de negocio son los responsables de diseñar los borradores iniciales del proceso que luego se trabajan con los usuarios de IT para que implementen esos procesos con las herramientas correspondientes. La

versión vigente del estándar BPMN es 2.0.2.

El modelado BPMN se realiza mediante diagramas compuestos por diversos elementos gráficos, cada uno con un significado y comportamiento definido. En la Figura 3 podemos apreciar un diagrama modelado en BPMN 2.0.

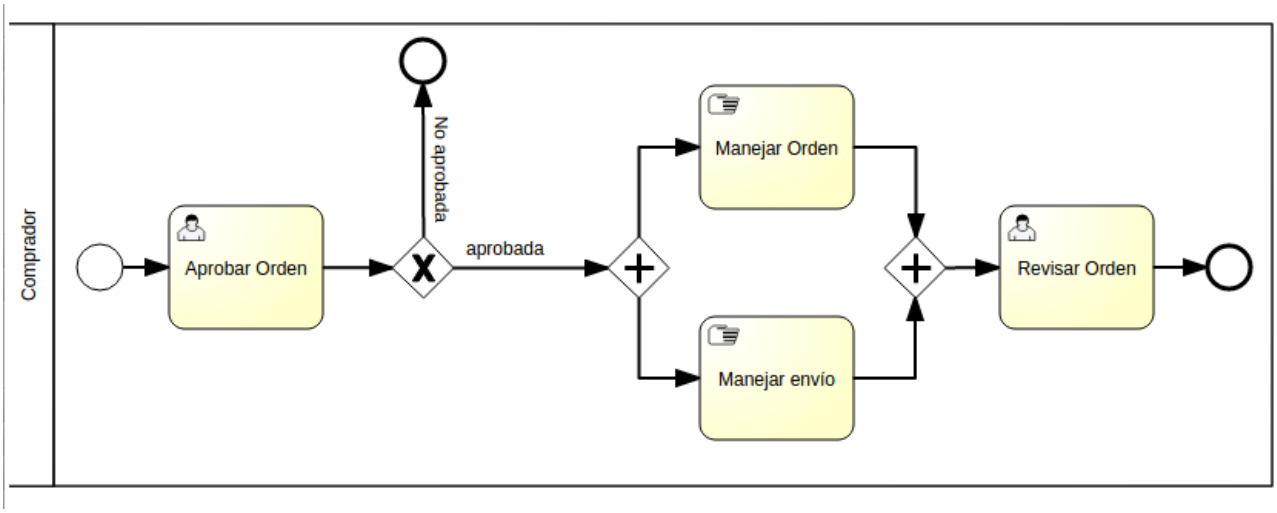


Figura 3: Ejemplo de proceso modelado en BPMN 2 de [BPMaN12]

Los elementos se pueden clasificar en cuatro categorías básicas:

- **Objetos de flujo:** actividades, eventos, gateways
- **Objetos de conexión:** flujos de mensaje y secuencia, asociación
- **Objetos contenedores:** pools y swimlanes
- **Artefactos:** anotaciones, grupos, objetos de datos

En la Figura 4 se pueden apreciar algunos de los elementos de BPMN enumerados anteriormente.

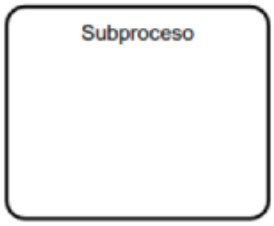
Elementos	Representación
Evento de inicio	
Evento de fin	
Evento Intermedio	
Tarea	
Subproceso	
Flujo de secuencia	
Compuerta OR o exclusiva	
Compuerta AND o paralela	
Pool	

Figura 4: Ejemplo de elementos BPMN 2

Una de las principales diferencias con otros lenguajes de modelado de procesos es que BPMN 2 puede ser ejecutado en un motor de procesos.

Por mas información sobre BPMN 2, ir al documento “Anexo A”, sección “BPMN 2”.

2.5 Simulación

La simulación es una técnica experimental para explorar el comportamiento de los sistemas modelados. Un modelo es siempre una abstracción de la realidad, una simplificación, aplicada con el objetivo de gestionar la complejidad.[P&K05]

Igualmente aunque se trate de una simplificación del mundo real, esta puede ser arbitrariamente compleja, especialmente cuando una gran cantidad de componentes y eventos entran en juego conjuntamente.
Si pensamos estrictamente en procesos de negocios, la complejidad emerge de tener muchas instancias de procesos, en paralelo, cada una posiblemente en un estado diferente, consumiendo diferentes recursos, a su vez cada uno esperando a que se liberen los recursos compartidos requeridos, fácilmente se puede apreciar que existe una casuística considerable que puede otorgar a un escenario de simulación un grado de complejidad considerable.

Por complejo que resulte realizar una simulación, siempre es mas simple que realizar una optimización matemática del problema (además no siempre el problema lo permite). Generalmente este tipo de optimizaciones resulta sumamente compleja de realizar, pero como aspecto positivo a destacar es que persiguen una solución óptima.

Los modelos analíticos difieren de la simulación en que estos proveen un conjunto de ecuaciones de las que se puede obtener una solución del tipo “forma cerrada”. La desventaja de este tipo de optimización es que pone muchas mas restricciones en el modelo y lo vuelven menos “real”.

Uno de los puntos negativos de la simulación es que no provee una solución óptima al problema planteado, solamente calculan los resultados para determinado caso en particular y da pistas del comportamiento del modelo. En estos casos la optimización se debe realizar como un proceso separado, “jugando” con diferentes escenarios para poder encontrar buenos modelos, a este procedimiento se lo conoce como análisis de sensibilidad.

Cuando hablamos procesos de negocio, el enfoque de simulación y optimización pareciera ser el enfoque mas adecuado. Generalmente los consultores y expertos en procesos no dominan el campo de las matemáticas y estadísticas como para realizar análisis matemáticos complejos.

En la bibliografía podemos encontrar muchas definiciones para el término simulación, pero en este caso vamos a hacer referencia a una del año 1975, “La simulación es el proceso de describir un sistema real y el uso de este modelo para la experimentación , con el objetivo de comprender el comportamiento del sistema o para explorar estrategias alternativas para su funcionamiento” [SHAN75].

Generalmente se distingue dos tipos de simulaciones, discreta o continua. La diferencia entre una y otra esta dada por la forma en que las variables de estado cambian.

Simulación de eventos discretos: En este tipo de simulación las variables de estado cambian instantáneamente en puntos distintos en el tiempo, por lo tanto, nada pasa entre dos puntos contiguos en el tiempo. La ejecución de la simulación se traduce en una secuencia finita de estados.

Simulación continua: En una simulación continua, las variables cambian continuamente de estado, por lo general a través de una función en la que el tiempo es una variable.

En la práctica, la mayoría de las simulaciones utilizan ambos tipos de variables de estado, discretas y continuas , pero una de ellas predomina y lleva a la clasificación de toda la simulación.

2.5.1 SIMULACIÓN DE EVENTOS DISCRETOS (DES)

Este tipo de simulación se presta mejor para simular un proceso de negocio si tenemos en cuenta que todo en el contexto de un proceso puede ser expresado como un evento discreto en el tiempo.

Por ejemplo, el proceso llevado a cabo cuando se compra un articulo en una tienda online como Amazon, seleccionar el articulo, agregarlo al carro, confirmar la compra y finalmente abonar el artículo, todos estos pasos del proceso de compra pueden ser vistos como un determinado evento que ocurren en un momento dado en el tiempo.

Una aclaración importante a realizar es que la calidad del resultado obtenido de una simulación depende directamente de la correctitud del modelo y de los datos de entrada. Especialmente en modelos complejos, si se comete errores en el modelado del problema o en los datos de entrada utilizados, esto sumado a la no exactitud de los resultados provistos por la simulación pueden hacer que el resultado final cambie drásticamente. Es por esto que los resultados nunca deben considerarse como un hecho, sino que deben validarse cuidadosamente.

Un modelo DES consiste básicamente en entidades que cambian su estado en el transcurso del tiempo, siguiendo determinadas reglas.

Aleatoriedad

Un DES debe de incluir cierta aleatoriedad, en realidad esta es una de las principales razones para utilizar este tipo de simulación en lugar de utilizar optimización matemática. En una simulación esta aleatoriedad es implementada por componentes estocásticos [ESTOC15], los cuales a menudo en sus cimientos se basan en distribuciones estadísticas.

La aleatoriedad es parte de la realidad, ¿cuantas ordenes de compras realizarán los clientes en un determinado momento del tiempo?, la respuesta es un valor al azar, al igual que lanzar un dado. La aleatoriedad en los sistemas informáticos es un problema debido a que las computadoras solo pueden realizar cálculos. La verdadera aleatoriedad solo se puede obtener mediante el uso de hardware específico, pero esto no es necesario en la mayoría de los casos gracias a los generadores de números pseudoaleatorios, que se calculan mediante algoritmos especiales [PSEU15].

Proceso de simulación

La creación y operación de una simulación del tipo DES en su momento era considerada sumamente compleja, solo algunos privilegiados eran capaz de lidiar con este tipo de problemas, sin embargo, durante las últimas décadas se ha estandarizado un proceso bien definido para la validación, operación y análisis de los resultados de una simulación.

En la Figura 5 se puede apreciar el proceso mencionado anteriormente:

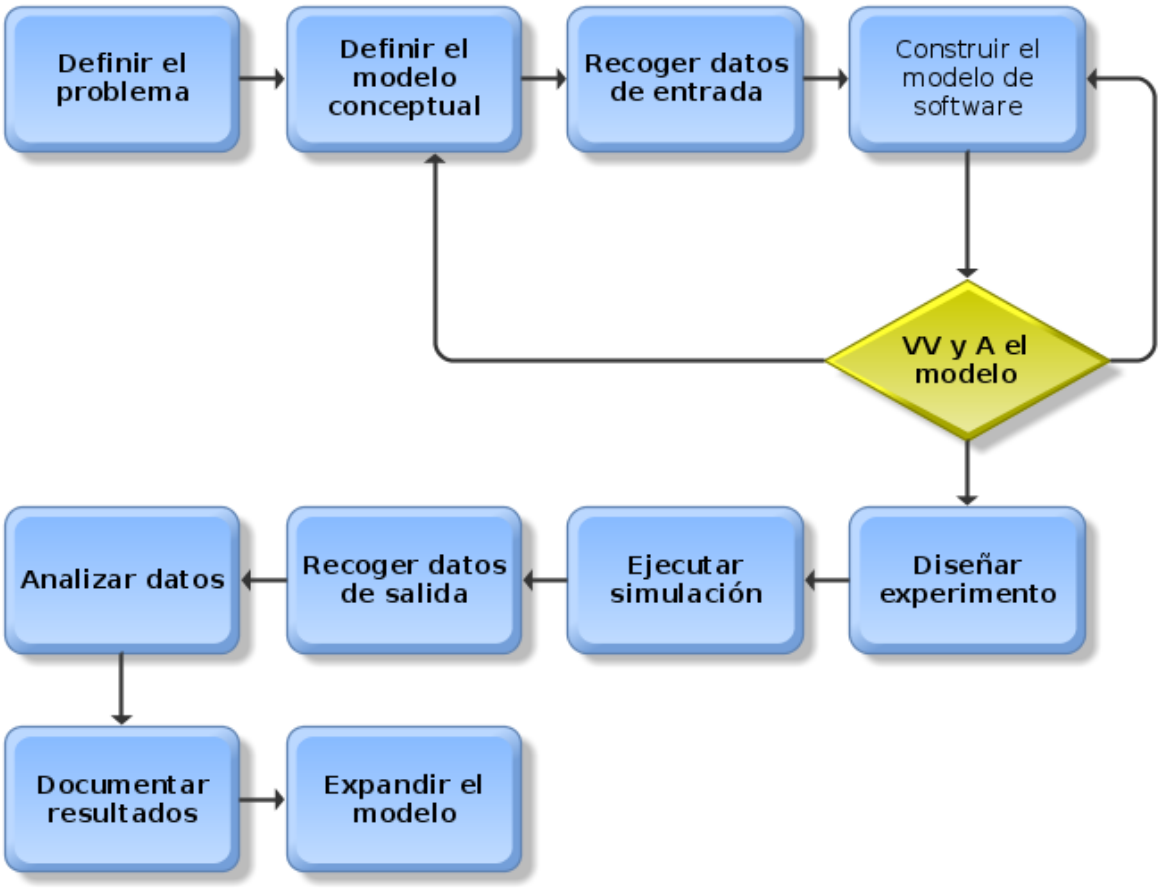


Figura 5: Proceso de simulación de [SHAN75]

Definir el problema: El primer paso en una simulación es definir concretamente cual es el problema a resolver por el modelo. Se deben de definir claramente los límites entre el problema a modelar y el contexto.

Definir el modelo conceptual: Una vez definido el problema a abordar, se debe definir el modelo conceptual a simular. Este debe incluir los algoritmos utilizados para describir el sistema, las necesidades de entrada y las salidas generadas. Todas las suposiciones que se hagan sobre el sistema deben de quedar claras en esta etapa.

En esta etapa también se definen los recursos tanto humanos como materiales, tiempos, etc. para hacer “funcionar” el modelo.

Recoger datos de entrada: En esta etapa se debe proporcionar la información de entrada necesaria para alimentar el modelo. Estos datos pueden ser por ejemplo datos estadísticos conocidos del comportamiento del sistema. La correcta ejecución de esta etapa compromete seriamente los resultados obtenidos.

Construir el modelo de software: El modelo de simulación se construye en base a la solución definida y a los datos de entrada recolectados anteriormente. Esta etapa se puede realizar utilizando software de simulación existente o creando el software necesario a medida.

Verificar, Validar y acreditar el modelo (VV y A): En esta fase se debe asegurar la correctitud de los datos de entrada, de las suposiciones y de los algoritmos del modelo. Esta etapa pensada para identificar los problemas que puedan surgir por mala definición de los elementos detallados anteriormente.

Diseñar experimento: Esta fase identifica la forma mas precisa y productiva de ejecutar la simulación. En esta fase es donde por ejemplo se debe seleccionar el tipo de distribución estadística que se utilizará para simular el comportamiento de algunas o todas las variables de entrada.

Ejecutar la simulación: En esta fase se pone en ejecución el modelo construido y validado en la fase anterior. La ejecución de la simulación genera los datos de salida (que serán o no, dependiendo de que tan bien se hayan realizado los pasos anteriores) la solución al problema inicialmente propuesto.

Recoger datos de salida: Luego de realizada la ejecución los datos de salida se procesan y almacenan para su posterior análisis.

Analizar datos: La cantidad de datos obtenidos de la fase anterior puede ser considerable a lo largo del tiempo. Estos deben ser analizados con detalle para por ejemplo, extraer tendencias a largo plazo y para cuantificar las respuestas a las preguntas que motivaron la simulación. Este análisis generalmente proporciona información en formato de tablas, gráficos, etc.

Documentar resultados: Los resultados de la sesión de simulación deben ser documentados y comunicados a las partes interesadas.

Expandir el modelo: Los modelos de simulación generalmente son caros y difíciles de construir, por lo tanto una vez que se genera un modelo, lo que se estila es ir adaptándolo a medida que van surgiendo nuevos requerimientos, o incluso, se reutiliza para otros proyectos similares.

Aunque el esquema detallado anteriormente es bastante genérico, los simuladores de procesos de negocio actuales tienen muchas de estas características en común.

Componentes de un DES

Si bien es cierto que existen tipos diferentes de simulación, que varían en sus fases de diseño e implementación, la mayoría de ellas tienen mucho en común. A continuación se presenta lista de los componentes que comúnmente se encuentran en todo simulador, especialmente en los orientados a simular procesos.

Reloj de simulación: Este componente es el encargado de contabilizar el tiempo de la simulación, siempre que se quiera saber algo relacionado al tiempo de simulación del modelo se debería consultar a este componente. Cabe destacar que nada tiene que ver con el tiempo real. El tiempo del modelo es independiente del tiempo real, esto hace que sea posible por ejemplo computar modelos que en tiempo real llevaría años en pocos segundos.

Existen 2 formas diferentes de simular el tiempo, basado en eventos (event stepped) o de forma incremental (time stepped). En el primer caso los cambios de tiempo solo se dan en los puntos que ocurre un evento, es decir, el modelo salta de un evento a otro omitiendo la representación de tiempos intermedios.
El segundo tipo de simulación de tiempo es utilizado cuando existe una gran cantidad de

interacción entre entidades basadas en eventos compartidos. Por ejemplo, cuando existe la necesidad de presentar un flujo constante de eventos y tiempo para una persona interactuando con la simulación. El tiempo en estos casos salta de una posición a otra en incrementos de tiempo generalmente fijos.

El manejo del tiempo también puede variar si se requiere realizar simulaciones en paralelo y/o distribuidas. En estos casos el manejo del reloj de puede realizar también de dos maneras diferentes, de forma conservativa (conservative) u optimista (optimistic).

En el primer caso se opta por mantener la consistencia entre todos los procesos durante la ejecución, por lo tanto se asume que la sincronización entre procesos ocurre frecuentemente por lo que se opta por la sincronización conservativa.

En el caso de sincronizar de forma optimista , lo que se hace es dejar que la ejecución de los procesos se suceda en el tiempo lo mas rápido que se pueda computar, cuando se recibe un evento de otro proceso que afecta la ejecución actual, se hace rollback en el tiempo para incluir la nueva interacción.

Lista de eventos: Es necesario mantener una lista ordenada de los eventos que el simulador tendrá que ir ejecutando a medida que transcurre el tiempo.

Controlador central: Este componente oficia de scheduler, es el encargado de ejecutar la simulación, lo que implica seleccionar eventos, encargarse del manejo del tiempo, ejecutar eventos, realizar los cálculos necesarios de las variables que se presentarán como resultado y finalizar la simulación.

Estados del modelo /Model State: En cada momento del tiempo, las entidades a simular deben de tener un estado definido, y ese estado debe ser representado de alguna manera en el sistema. En el caso específico de este trabajo de tesis, el motor de workflow se encargará de cumplir esta función.

Generador de números aleatorios: Los modelos a simular generalmente necesitan el uso de números aleatorios para introducir la variabilidad. Como se menciono anteriormente uno de los pasos mas importantes para una simulación es la recolección de datos, en el caso de que no se cuente con datos históricos que determinen el comportamiento de las variables a simular, se hace necesario recurrir a técnicas estadísticas para emular el comportamiento “real” de esas variables.

Los generadores de números aleatorios se utilizan para reproducir la creación de una serie de números que supuestamente son aleatorios e independientes el uno del otro.

Recolector de datos: Pieza encargada de recolectar los datos estadísticos obtenidos durante la simulación para posteriormente presentarlos de forma amigable para que sean interpretados por el usuario final.

2.5.2 BPSIM

BPSim (Business Process Simulation) es un estándar para la parametrización e intercambio de datos de análisis de procesos de negocio diseñado para promover una adopción más amplia de la simulación dentro de la comunidad BPM a través de estándares.

BPSim es un estándar perteneciente a la WfMC (Workflow Management Coalition) que comenzó a

tomar forma por el 2011 con la formación del grupo de trabajo BSWG (Business Simulation Working Group), este grupo conformado por diversos especialistas en procesos de negocios la mayoría de ellos asociados a empresas reconocidas del medio libero la primera versión del estándar para comienzos del año 2013.

La documentación disponible al momento de ser analizada consta básicamente de 2 documentos, el primero redacta la especificación del estándar y el segundo una breve guía de implementación para desarrolladores.

Si comparamos la cantidad y calidad de la documentación con la de otros estándares, por ejemplo BPMN 2.0, llama notoriamente la atención la cantidad de material disponible, incluyendo el oficial. Realmente es muy poca la documentación a la fecha, esto denota claramente que se trata de algo sumamente nuevo, que recién esta cobrando fuerza dentro del mundo de la gestión de procesos de negocio.

¿Que viene a solucionar BPSim?

Una de los principales problemas en el campo de simulación asociado a procesos de negocios es la falta de estándares a seguir por parte de los implementadores, esta falta lleva a que las herramientas de simulación no sean compatibles entre si.

A contrario de lo que pasa con las herramientas de modelado de procesos que implementan BPMN 2 por ejemplo, que teóricamente una vez modelado un proceso en una herramienta X, este modelo podría ser utilizado desde cualquier otra herramienta que implemente el estándar. Teóricamente entre comillas porque esto no se cumple en todos los casos, muchas de las herramientas que dicen ser compatibles con el estándar en realidad no lo son, o no lo son al 100%.

BPSim esta pensado para contribuir al negocio desde los siguientes puntos de vista:

- Ayudar a validar y diseñar un proceso
- Reducir el riesgo frente a los cambios
- Predecir la performance de un proceso
- Servir como soporte a la toma de decisiones
- Asignación y gestión de recursos

Desde el inicio fue concebido como un complemento para dos de los principales estándares de proceso de negocio, XPDL 2.2 y BPMN 2.0. A continuación se presenta la Figura 6 donde se puede apreciar de que manera se integra con dichos estándares.

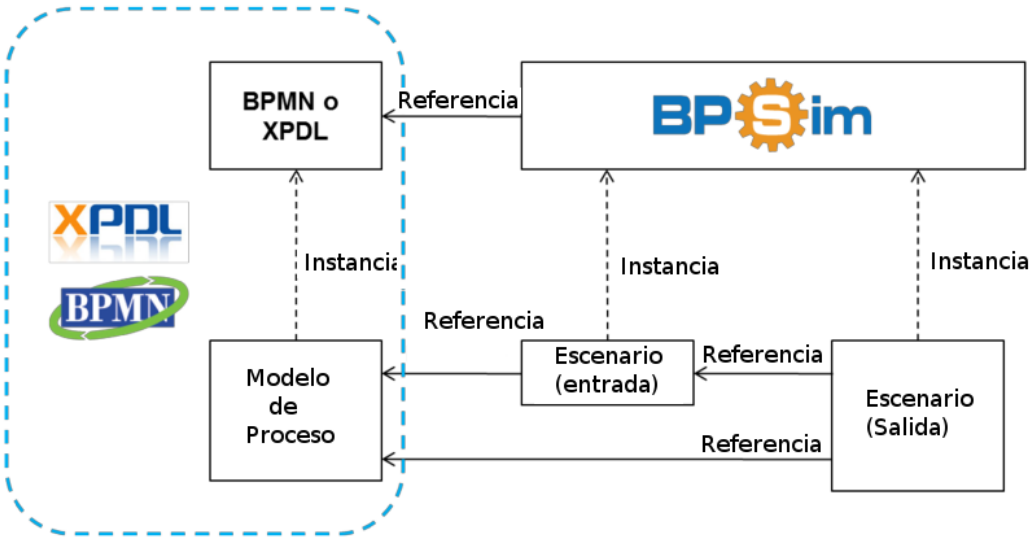


Figura 6: Diagrama de componentes de BPSim [BPSim2013]

En la figura anterior se pueden apreciar los componentes claves que integra BPSim. El estándar introduce el concepto de escenario, donde se especifica el comportamiento asociado a la simulación de los elementos del modelo BPMN 2. El escenario hace referencia a los elementos del modelo de proceso a través del identificador de cada elemento.

El escenario se puede visualizar como un archivo XML donde en notación BPSim se definen los parámetros específicos para la simulación, como por ejemplo, tiempo de procesamiento y espera de una tarea, probabilidad de ejecución de un determinado flujo de secuencia, etc. la información del escenario puede ser almacenada de forma independiente (Archivo XML separado) o integrado como extensión al archivo XML del proceso de negocio.

En el escenario de simulación se pueden definir tanto parámetros de entrada (los que determinan el comportamiento de la simulación) como parámetros de salida (datos que se quieren consultar como resultado de la simulación). Cada uno de estos parámetros de entrada o salida hacen referencia a un elemento del proceso de negocio a simular.

Especificación

BPSim básicamente se conforma de dos partes:

- Meta-modelo
- Definición del formato de archivo de para almacenamiento e intercambio de los datos.

El meta-modelo esta expresado utilizando UML (Unified Modeling Language) y el formato de archivo de intercambio esta definido usando XSD (XML Schema Definition). Estos dos elementos juntos representan el core de la especificación.

El meta-modelo de BPSim esta expresado mediante diagramas UML y a grandes rasgos esta compuesto por los siguientes elementos:

- BPSimData
- Scenario
- ScenarioParameters

- VendorExtension
- ElementParameters

En la Figura 7 se pueden apreciar los elementos mas relevantes del core de BPSim.

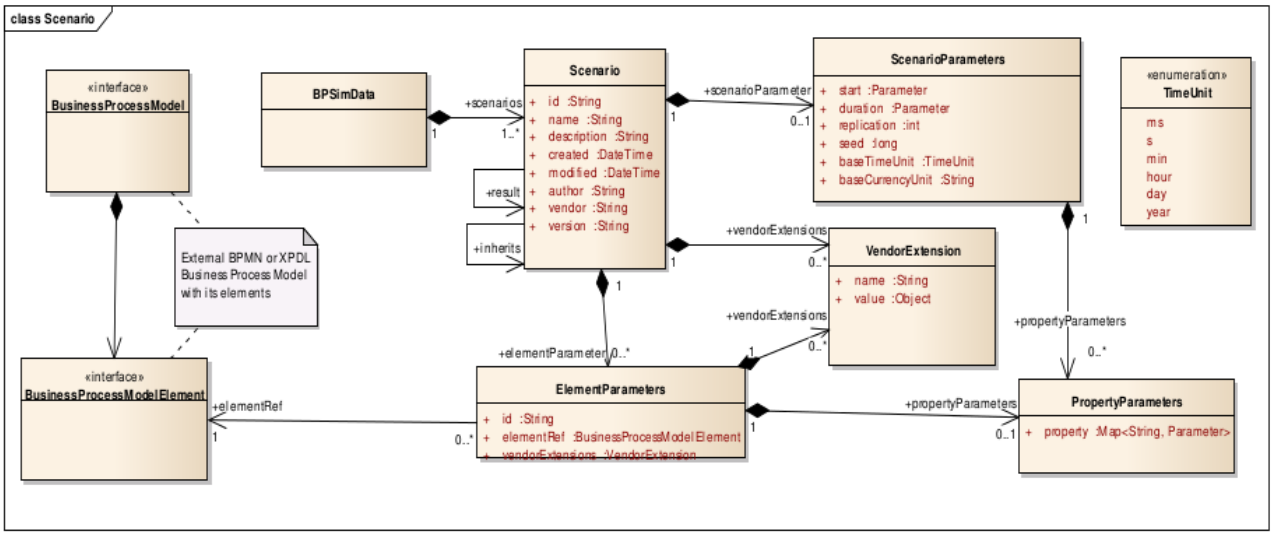


Figura 7: Diagrama de clases del metamodelo de BPSim de [BPSimSpec13]

BPSimData: Es el elemento raíz del modelo y contiene todos los escenarios definidos.

Scenario: Un escenario representa un caso de prueba asociado a un modelo de proceso de negocio. Esta compuesto por toda la información de entrada necesaria para poder simular el proceso y de la información de salida resultado del análisis realizado. También esta permitida la herencia entre escenarios, teniendo que definir nuevamente el comportamiento de los atributos que se desea modificar o los nuevos atributos.

Un escenario esta compuesto por una colección de elementos de la clase “ElementParameters”. Cada uno de estos parámetros debe hacer referencia a un elemento específico de los elementos que componen el proceso de negocio asociado. En la Tabla 1 se presentan los atributos definidos para un escenario.

Tabla 1: Atributos de la clase Scenario

Atributos	Descripción
String: id	Identificador único del escenario
String: name	Nombre del escenario
String: description	Descripción del escenario
DateTime: created	Fecha hora de cuando fue creado el escenario
DateTime: modified	Fecha hora de cuando fue modificado el escenario
String: author	El nombre del autor del escenario
String:vendor	El nombre de la herramienta de software utilizada para creer el escenario
String:version	El N° de versión del escenario
Scenario:Result	En el caso de que el escenario sea el resultado de

	una simulación y el código del escenario que generó dicho resultado se encuentra en el mismo modelo, este campo debería referenciar a ese escenario.
Scenario:inherits	Referencia al escenario del cual este escenario heredará.
ElementParameter:elementParameter	Colección de elementos que hacen referencia a los elementos del proceso de negocio a simular.
ScenarioParameter:scenarioParameter	Parámetros de configuración que definen el comportamiento del escenario
VendorExtension:vendorExtension	Mecanismo que provee la posibilidad de extender por parte de un implementador las capacidades definidas en el estándar

ScenarioParameters: La información contenida en esta clase determina el comportamiento de un escenario. En la Tabla 2 se muestran los atributos definidos en la clase.

Tabla 2: Parámetros de un escenario

Atributos	Descripción
Parameter: start	Define el tiempo de inicio del escenario
Parameter: duration	Determina la duración del escenario
Int:replication	Numero de replicaciones (cantidad de veces) que el escenario debe ser ejecutado
Long:seed	Semilla randómica utilizada para inicializar un pseudo generador de números aleatorios.
TimeUnit:baseTimeUnit	Define la unidad de tiempo a utilizar en el escenario. Todos los parámetros de tiempo (a no ser que se redefinan localmente) deben considerarse expresados en esta unidad
String:baseCurrencyUnit	Define la moneda por defecto para el escenario. Esta moneda deberá expresarse usando el estándar ISO 4217. Al igual que en el caso anterior, todos los parámetros que representen costos (a no ser que se redefinan localmente) deben considerarse expresados en esta moneda

ElementParameters: Cada elemento de esta clase hace referencia a un elemento del modelo de proceso de negocio asociado. Los parámetros se encuentran clasificados en seis perspectivas diferentes (tiempo, costo, recursos, control, prioridad y propiedades). Cada una de estas perspectivas contiene diferentes propiedades relacionadas entre si que permiten definir el comportamiento de los diferentes elementos del modelo a simular.

En la Tabla 3 se presentan los atributos pertenecientes a la clase.

Tabla 3: Clase ElementParameters

Atributos	**Descripción**
String: id	Identificador único del parámetro
BusinessProcessModelElement: elementRef	Mediante este atributo se relacionan los parámetros definidos en BPSim con los elementos del proceso

	de negocio a simular.
VendorExtension: vendorsExtensions	Extensiones de terceros asociadas a este parámetro en particular

VendorExtension: Esta clase permite que terceros puedan incorporar al estándar sus propias extensiones. Si bien la recomendación de los creadores de BPSim es mantenerse dentro del estándar ya que consideran que este soporta la mayoría de los elementos necesarios en un ambiente de simulación, en determinadas ocasiones puede ser necesario querer extender las funcionalidades ofrecidas. En la Tabla 4 se presentan los atributos pertenecientes a la clase.

Tabla 4: Atributos VendorExtension

Atributos	Descripción
String: name	El nombre que identifica a la extensión. Este nombre debe ser único por lo que se recomienda utilizar un prefijo que identifique al proveedor para evitar conflictos con extensiones de otros proveedores
Object: value	Valor de la extensión

Perspectivas de simulación

BPSim propone complementar XPDL y BPMN permitiendo parametrizar un proceso de negocio desde diferentes perspectivas que contribuyen al análisis, simulación y optimización del mismo. En la Figura 8 se pueden apreciar las diferentes perspectivas definidas:

Figura 8: Perspectivas de BPSim de [BPSim2013]

BPSim permite parametrizar un proceso de negocio desde las seis perspectivas que muestra la Figura 8. Cada parámetro definido en el escenario de simulación referencia un elemento específico del modelo de proceso y cada elemento del modelo puede tener cero o muchos parámetros asociados.

Perspectiva tiempo: Esta perspectiva esta representada por la clase TimeParameters del modelo

UML. Aquí se encuentran agrupados todos los parámetros de tiempo de los diferentes elementos del modelo de proceso de negocio. Todos los parámetros reflejan básicamente intervalos de tiempo.

Existen dos intervalos de tiempo que son los más destacados, el primero es la duración (duration time) y el segundo es el tiempo de retraso (lag time). Estas dos variables en realidad son valores calculados, por lo tanto no pertenecen a la especificación del metamodelo.

La duración puede ser calculada mediante la suma de los siguiente parámetros:

duration = setupTime + processingTime + validationTime + reworkTime

En los casos que no sea necesario llegar a este nivel de granularidad, el parámetro utilizado para representar el tiempo de ejecución de una actividad cualquiera es processingTime.

El tiempo de retraso puede ser calculado de la siguiente forma:

lagTime = transferTime + quequeTime + waitTime

Un parámetro del tipo TimeParameters debe ser representado por alguna de estas tres clases:

- NumericParameter
- FloatingParameter
- DurationParameter

En la Tabla 5 se presenta la lista de atributos que componen la perspectiva.

Tabla 5: Atributos perspectiva tiempo

Atributos	Descripción
Parameter: transferTime	Tiempo que transcurre desde que finalizo el paso anterior y se llego al paso actual
Parameter: queueTime	El tiempo transcurrido entre que la tarea esta disponible y la misma es asignada
Parameter: waitTime	El tiempo transcurrido entre que la tarea es asignada y esta comienza
Parameter: setupTime	El tiempo gastado preparándose para realizar el trabajo actual
Parameter: processingTime	Tiempo empleado en completar el trabajo actual
Parameter: validationTime	El tiempo empleado revisando el trabajo realizado
Parameter: reworkTime	El tiempo empleado corrigiendo el trabajo realizado

Perspectiva Costo: Esta perspectiva esta representada por la clase CostParameters del modelo UML. Aquí se encuentran agrupados todos los parámetros que permiten especificar el costo de los elementos del proceso de negocio.

En la Tabla 6 se presenta la lista de atributos que componen la perspectiva.

Tabla 6: Atributos perspectiva costo

Atributos	Descripción
Parameter: fixedCost	Especifica el costo fijo que se pagará por única vez cada vez que se realice la “tarea”. Por defecto esta expresado en la moneda definida como parámetro del escenario (BaseCurrencyUnit)
Parameter: unitCost	Especifica el costo que se pagará por cada unidad de tiempo que insuma la “tarea”. Al igual que en el caso anterior, por defecto esta expresada en la moneda del escenario.

Perspectiva Control: Esta perspectiva esta representada por la clase ControlParameters del modelo UML. Aquí se encuentran agrupados todos los parámetros que permiten controlar el flujo de ejecución de los diferentes elementos de un proceso de negocio.

En la Tabla 7 se presenta la lista de atributos que componen la perspectiva.

Tabla 7: Atributos perspectiva control

Atributos	Descripción
Parameter: interTriggerTimer	Especifica el intervalo de tiempo entre dos ocurrencias de un evento. El evento ocurrirá constantemente cada intervalo de tiempo hasta que se alcance la cantidad establecida en el parámetro triggerCount
Parameter: triggerCount	Especifica la cantidad máxima de veces que se deberá repetir determinado evento
Parameter: probability	Define la probabilidad que el control del flujo pase a determinado elemento
Parameter: condition	Define una condición que de cumplirse se pasa el control del flujo a determinado elemento. Esta propiedad es complementaria de la probabilidad, esto quiere decir que en un elemento es posible definir solamente una de las dos a la vez.

Perspectiva Prioridad: Esta perspectiva esta representada por la clase PriorityParameters del modelo UML. Aquí se encuentran agrupados todos los parámetros que permiten especificar la prioridad de los diferentes elementos de un proceso de negocio.

En la Tabla 8 se presenta la lista de atributos que componen la perspectiva.

Tabla 8: Atributos perspectiva prioridad

Atributos	Descripción
Parameter: interruptible	Determina si la ejecución del elemento es o no interrumpible
Parameter: priority	Determina la prioridad que un elemento a la hora de competir por la asignación de recursos

Perspectiva Recursos: Esta perspectiva esta representada por la clase ResourceParameters del modelo UML. Aquí se encuentran agrupados todos los parámetros que permiten especificar los recursos de los diferentes elementos de un proceso de negocio. A continuación se presenta la lista de atributos que componen la perspectiva.

Tabla 9: Atributos perspectiva recursos

Atributos	Descripción
Parameter: availability	Determina cuando un recurso esta disponible o no. La disponibilidad de un recurso puede variar en función de un calendario
Parameter: quantity	Determina la cantidad de recursos disponibles o necesarios
Parameter: selection	Define el criterio de selección para asignar un determinado recurso.
List<Parameter>: role	Los roles a los cuales pertenece el recurso. La lista de roles que se especifica en este atributo puede ser utilizada en por el parámetro Selection

Perspectiva Propiedades: Esta perspectiva esta representada por la clase PropertyParameters del modelo UML. Aquí se encuentran agrupados todos los parámetros que permiten especificar propiedades adicionales a ser asignadas a los diferentes elementos de un proceso de negocio.

En la Tabla 10 se presenta la lista de atributos que componen la perspectiva.

Tabla 10: Atributos perspectiva propiedades

Atributos	Descripción
Map<String, Parameter>	Define propiedades adicionales a ser asignadas a un elemento del proceso de negocios. Estas propiedades pueden ser utilizadas posteriormente como input de un ExpressionParameters. Este tipo de propiedades son evaluadas inmediatamente que el token entra en el elemento correspondiente.

En la Figura 9 podemos apreciar un proceso de negocio en BPMN 2.0 y un ejemplo de como sería la parametrización de las diferentes perspectivas asociadas a los elementos del modelo.

Escenario_1

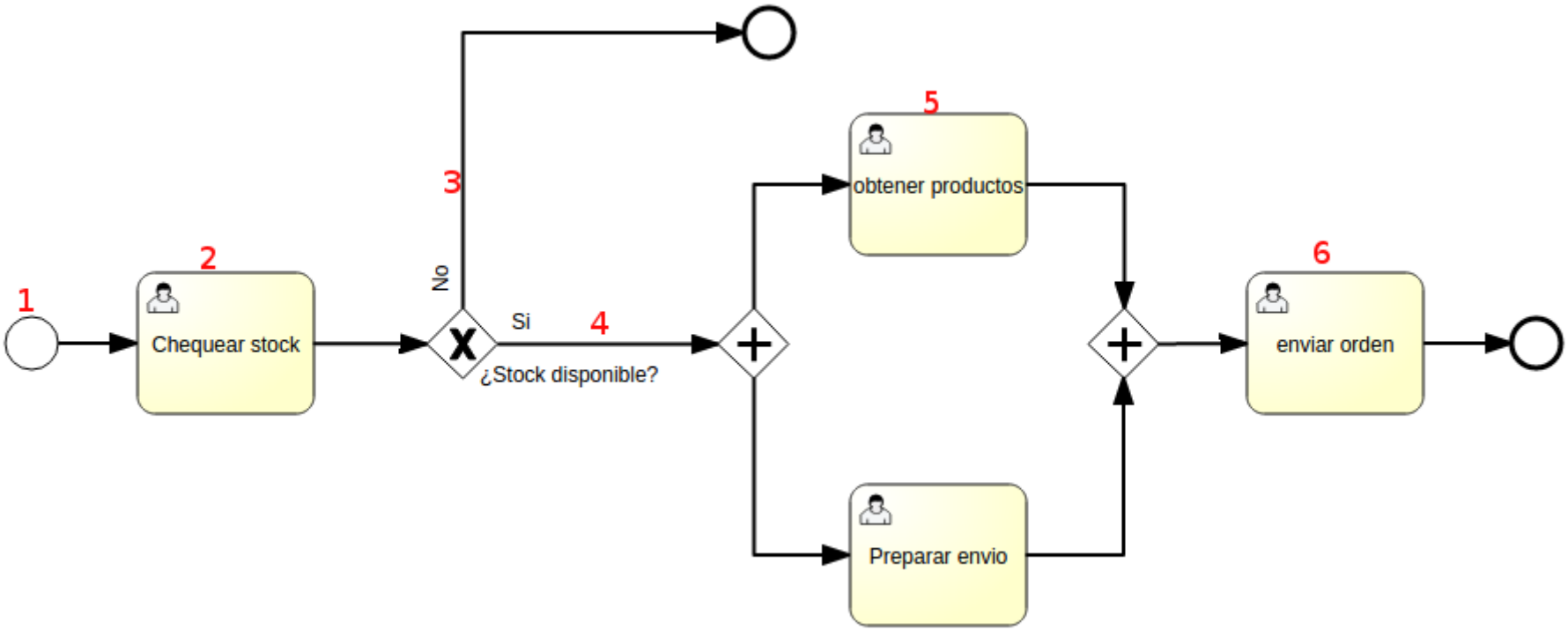


Figura 9: Ejemplo parametrización proceso BPMN 2

En la figura 10 se pude apreciar el código XML del proceso modelado en la figura 9

```
<process id="process" isExecutable="true">
  <startEvent id="inicio" name="inicio"></startEvent>
  <userTask id="ChequearStock" name="Chequear stock"></userTask>
  <exclusiveGateway id="eg1"></exclusiveGateway>
  <endEvent id="fin1"></endEvent>
  <parallelGateway id="pg1"></parallelGateway>
  <userTask id="obtenerProducto" name="obtener productos"></userTask>
  <userTask id="prepararEnvio" name="preparar envio"></userTask>
  <parallelGateway id="pg2"></parallelGateway>
  <userTask id="enviarOrden" name="enviar orden"></userTask>
  <endEvent id="fin"></endEvent>
  <sequenceFlow id="flow1" sourceRef="inicio" targetRef="ChequearStock"></sequenceFlow>
  <sequenceFlow id="flow2" sourceRef="ChequearStock" targetRef="eg1"></sequenceFlow>
  <sequenceFlow id="flow3" sourceRef="eg1" targetRef="fin1"></sequenceFlow>
  <sequenceFlow id="flow4" sourceRef="eg1" targetRef="pg1"></sequenceFlow>
  <sequenceFlow id="flow5" sourceRef="pg1" targetRef="obtenerProducto"></sequenceFlow>
  <sequenceFlow id="flow6" sourceRef="pg1" targetRef="prepararEnvio"></sequenceFlow>
  <sequenceFlow id="flow7" sourceRef="obtenerProducto" targetRef="pg2"></sequenceFlow>
  <sequenceFlow id="flow8" sourceRef="prepararEnvio" targetRef="pg2"></sequenceFlow>
  <sequenceFlow id="flow9" sourceRef="pg2" targetRef="enviarOrden"></sequenceFlow>
  <sequenceFlow id="flow10" sourceRef="enviarOrden" targetRef="fin"></sequenceFlow>
</process>
```

Figura 10: Código XML del proceso modelado en la figura 9

Se pueden apreciar los diferentes elementos del modelo de proceso y ver como cada uno de ellos esta identificado a través del atributo **“id”**. Este identificador es el que se utiliza desde BPSim para definir el comportamiento asociado al escenario de simulación para los elementos del modelo BPMN 2.0.

```
<bpsim:ScenarioParameters baseCurrencyUnit="US$" baseTimeUnit="min">
  <bpsim:Start>
    <bpsim:DateTimeParameter value="2014-01-01T00:00:00"/>
  </bpsim:Start>
  <bpsim:Duration>
    <bpsim:DurationParameter value="PT10H"/>
  </bpsim:Duration>
</bpsim:ScenarioParameters>
```

Figura 11: BPSim - Parametrización Escenario

baseCurrencyUnit y baseTimeUnit: Definen la moneda y la unidad de tiempo por defecto para el escenario. Estos parámetros pueden ser sobrescritos para cada elemento de forma particular.
DateTimeParameter: Determina la fecha de inicio para el reloj de la simulación.
DurationParameter: Determina la duración del escenario en formato ISO 8601, pasado ese tiempo se aborta la simulación.

```
<bpsim:ElementParameters elementRef="inicio">
  <bpsim:ControlParameters>
    <bpsim:InterTriggerTimer>
      <bpsim:NumericParameter timeUnit="min" value="3"/>
    </bpsim:InterTriggerTimer>
    <bpsim:TriggerCount>
      <bpsim:NumericParameter value="100"/>
    </bpsim:TriggerCount>
  </bpsim:ControlParameters>
</bpsim:ElementParameters>
```

Figura 12: BPSim - Parametrización nodo inicio

InterTriggerTimer: Determina cada cuanto tiempo se dispara la simulación de una nueva

instancia del proceso.
TriggerCount: Determina la cantidad de instancias a simular.

Se puede apreciar en el atributo **elementRef** como se hace referencia al id del elemento descripto en el XML del modelo BPMN 2.0.

```
<bpsim:ElementParameters elementRef="ChequearStock">
  <bpsim:TimeParameters>
    <bpsim:WaitTime>
      <bpsim:NumericParameter timeUnit="min" value="1"/>
    </bpsim:WaitTime>
    <bpsim:ProcessingTime>
      <bpsim:NormalDistribution timeUnit="min" mean="5.0" standardDeviation="3.0"/>
    </bpsim:ProcessingTime>
  </bpsim:TimeParameters>
  <bpsim:ResourceParameters>
    <bpsim:Quantity>
      <bpsim:NumericParameter value="1"/>
    </bpsim:Quantity>
  </bpsim:ResourceParameters>
  <bpsim:CostParameters>
    <bpsim:FixedCost>
      <bpsim:FloatingParameter currencyUnit="US$" value="100.0"/>
```

Figura 13: BPSim - Parametrización tarea "chequear stock"

WaitTime: Define el tiempo de espera antes de comenzar la tarea.
ProcessingTime: Determina el tiempo que se tarda en resolver la tarea. En este caso el parámetro fue definido utilizando un parámetro del tipo distribución normal.
Quantity: Cantidad de recursos asignados a resolver la tarea
FixedCost: Costo fijo involucrado en la ejecución de la tarea.

```
<bpsim:ControlParameters>
  <bpsim:Probability>
    <bpsim:FloatingParameter value="20.0"/>
  </bpsim:Probability>
</bpsim:ControlParameters>
```

Figura 14: BPSim - Parametrización flujos y condiciones

Probability: Probabilidad que el flujo de ejecución transite por este camino o flujo de secuencia.

```
<bpsim:ElementParameters elementRef="obtenerProducto">
  <bpsim:TimeParameters>
    <bpsim:WaitTime>
      <bpsim:UniformDistribution timeUnit="min" max="6.0" min="4.0"/>
    </bpsim:WaitTime>
    <bpsim:ProcessingTime>
      <bpsim:NumericParameter timeUnit="min" value="10"/>
    </bpsim:ProcessingTime>
  </bpsim:TimeParameters>
  <bpsim:ResourceParameters/>
  <bpsim:CostParameters>
    <bpsim:FixedCost>
      <bpsim:FloatingParameter currencyUnit="US$" value="0.0"/>
    </bpsim:FixedCost>
    <bpsim:UnitCost>
      <bpsim:FloatingParameter currencyUnit="US$" value="20.0"/>
    </bpsim:UnitCost>
  </bpsim:CostParameters>
</bpsim:ElementParameters>
```

Figura 15: BPSIM - Parametrización obtener producto y envío orden

BPSim permite parametrizar lo que serían los parámetros de entrada que alimentan el escenario de simulación (algunos se describieron anteriormente), pero además permite parametrizar los parámetros de salida del escenario, es decir, que es lo que se quiere medir o examinar como resultado de la simulación.

Documentación

Como se menciono anteriormente, la documentación oficial disponible se encuentra dividida en dos documentos, el primero es la documentación completa de la especificación del estándar y el segundo es la guía para el implementador.

Además de estos dos documentos en la página oficial se pueden encontrar el material expuesto por el grupo BSWG en diferentes presentaciones que han dado sobre BPSim.

Al margen de la documentación oficial, al momento de realizar este trabajo de tesis casi que no existe otro tipo de documentación del estándar que se pueda tomar como referencia.

Si hacemos referencia a la guía para el implementador por ejemplo, vemos como en tres cortos ejemplos basados en diferentes procesos de negocio se trata de explicar la mayor cantidad de funcionalidades disponibles en BPSim, sin llegar a entrar en detalle en ninguna de ellas.

En resumen, la cantidad y calidad de información disponible sobre el estándar al momento de comenzar el trabajo de tesis deja que desear, sobre todo cuando se necesita realizar un análisis en profundidad para por ejemplo como en este caso, poder desarrollar una herramienta basada en el. Existen contradicciones entre lo que dice la definición del estándar y lo que se presenta en el manual del implementador, conceptos que no están desarrollados en profundidad que se prestan a confusión y además no existe ningún libro escrito sobre el tema.

A la fecha actual esta situación no cambió demasiado, los documentos oficiales siguen en la misma versión y todavía sigue sin existir ningún libro asociado a la temática. Probablemente la falta de información se deba a que el estándar en si es muy reciente, en 2013 recién se hizo pública la versión 1.0, y aunque a la fecha ya han transcurrido dos años desde su primera versión se nota que su difusión se viene dando de forma paulatina.

3 Herramientas evaluadas de soporte

En este capítulo se presenta un resumen del análisis realizado de las herramientas de modelado y simulación disponibles en el mercado para tomarlas como punto de partida a la hora de construir el prototipo planteado. No solo se investigaron las que estaban alineadas con el objetivo de la tesis, sino que se trato de relevar e investigar todo el universo de herramientas disponibles.

Por ejemplo, una de las condiciones preestablecidas para este trabajo de tesis es que las herramientas seleccionadas deben de ser Open Source, pero igualmente se investigaron soluciones pagas para poder apreciar el estado actual de la evolución de este tipo de herramientas en toda su amplitud.

3.1 Criterios de selección

Existen en el mercado una cantidad considerable de modeladores y herramientas BPM que dicen ser compatibles con el estándar BPMN 2. Podemos encontrar pagas, gratuitas, de código libre, multiplataforma o para un sistema operativo en particular, con interfaz desktop o web, etc.

Durante la primera etapa de investigación se encontraron más de 70 herramientas en principio compatibles con BPMN 2, por lo que fue necesario aplicar determinados criterios de selección para poder filtrar y reducir la lista a una cantidad más manejable para su análisis.

Para confeccionar la lista definitiva se utilizo como fuente de información principal las siguientes páginas web:

- http://www.bpmn.org/
- http://en.wikipedia.org/wiki/Comparison_of_Business_Process_Modeling_Notation_tools
- http://bpm-conference.org/bpt-resource-management/

El primer link es la página oficial de la OMG del estándar BPMN, en la página se puede encontrar un apartado con herramientas que implementan el estándar. El segundo link es un artículo en Wikipedia que contiene una lista de herramientas compatibles con BPMN 2 y el tercer link es la página del evento BPM Conference organizado por algunos de los referentes más destacados en procesos de negocio (Ejemplo: Will Van Der Aalst, Mathias Weske).

3.1.1 CRITERIOS DE SELECCIÓN APLICADOS

A continuación se detallan los criterios aplicados para confeccionar la lista de herramientas a analizar.

- Soporte para BPMN 2
- Se priorizaron las herramientas más conocidas del mercado (Pagas o libres)
- Las herramientas Open Source mencionadas en los links anteriores.
- Posibilidad de conseguir una versión de la herramienta para pruebas.
- Soporte para simulación

Luego de haber realizado un primer filtro se confeccionó una lista de 21 candidatos a analizar. De las herramientas seleccionadas se identifico en primer lugar la información asociada a las características que se presentan en la Tabla 11:

Tabla 11: Características evaluadas

Atributo	Descripción
Nombre	Nombre de la herramienta
Empresa	Empresa fabricante de la herramienta
Plataforma	Sistema Operativo en la cual ejecuta
Features destacables	Funcionalidades que valga la pena destacar
Tipo de Licenciamiento	Tipo de licencia mediante la cual se distribuye la herramienta
URL	URL del sitio web oficial de la herramienta
Soporte de simulación	Soporta o no simulación

En la Tabla 12 se puede apreciar un resumen de los candidatos evaluados.

Tabla 12: Herramientas BPM evaluadas con soporte BPMN 2.0						
Nombre	**Compania**	**Plataforma**	**Features**	**Licencia**	**Sitio**	**Simulación**
Activiti Designer	Alfresco y la comunidad de Activiti	Eclipse plugin	Modelado, simulación y ejecución	Apache License 2.0	http://www.activiti.org/	Si
ADONIS CE	BOC	Windows	Modelado, análisis, simulación, evaluación	Freeware	http://www.adonis-community.com/	Si
Altova UModel	Altova	Windows	Modelado	Propietario	http://www.altova.com/es/umodel.html	No
ARIS Express	Software AG	Multiplataforma	Modelado	Freeware	http://www.ariscommunity.com/aris-express	No
Bizagi BPM Suite + Modeler	Bizagi	Windows	Modelado, simulación, ejecución	Propietario + Freeware	https://www.bizagi.com/	Si
Bonita BPM	BonitaSoft	Multiplataforma	Modelado, simulación, ejecución	GPL v2	http://es.bonitasoft.com/	Si
Borland Together	Borland	Multiplataforma	Modelado, simulación, validación	Propietario	http://www.borland.com/products/together/	Si
Camunda Modeler	Camunda	Multiplataforma	Modelado – Plugin para eclipse	Apache License 2.0	http://camunda.org/features/modeler.html	No
Enterprise Architect	Sparx Systems	Windows	Modelado, simulación	Propietario	http://www.sparxsystems.com.au/	Si
Genexus WorkFlow	Artech	Windows	Modelado	Propietario	http://www.genexus.com/productos/business-process-management-suite?es	No
Yaoqiang BPMN	Blenta	Multiplataforma	Modelado, simulación	Propietario (V3), GPL (V2.2)	http://sourceforge.net/projects/bpmn/	Si
IBM Process Designer	IBM	Multiplataforma	Modelado, simulación, ejecución	Propietario		Si
Igrafx Process	Igrafx	Multiplataforma	Modelado, simulación, análisis	Propietario	http://www.igrafx.com/	Si
INNOVATOR for Business Analysis	MID GmbH	Windows	Modelado, simulación, análisis	Propietario	http://www.mid.de/en/products/innovator-for-business-analysts.html	Si
JBPM	RedHat	Multiplataforma	Modelado, simulación y ejecución	Apache License	https://www.jboss.org/jbpm	Si
Visio 2013 (trial)	Microsoft	Windows	Modelado	Propietario	http://office.microsoft.com/en-us/visio	No
Modelio	ModelioSoft	Multiplataforma	Modelado	Open Source	http://www.modelio.org/	No

Signavio Process Editor	Signavio	Web, Windows y Linux	Modelado, simulación	Propietario	http://www.signavio.com/products/process-editor/process-modeling/	Si
Tibco ActiveMatrix	TIBCO Software	Windows, Linux	Modelado, simulación, ejecución	Propietario	http://www.tibco.be/products/automation/business-process-management/process-workforce-automation/activematrix-bpm/default.jsp	Si
Eclipse BPMN2 Modeler	Eclipse	Eclipse plugin	Modelado, simulación	Open Source	http://eclipse.org/bpmn2-modeler/	Si
BPMN Web Modeler	TrisoTech	Web	Modelado, simulación	Propietario	https://www.businessprocessincubator.com/bpmnwebmodeler/	Si
L-SIM	Lanner	Desconocido	Simulación	Propietario	http://www.lanner.com/en/l-sim.cfm	Si

Para cada herramienta analizada (en el caso de que corresponda) se completo el siguiente cuadro:

Tabla 13: Características analizadas por herramienta

Features	Soporta
Simulación	Si / No / No corresponde
Soporte para BpSim	Si / No / No corresponde
Exporta diagrama a archivo xml BPMN 2	Si / No
Importa diagrama de archivo xml BPMN 2	Si / No
Conformidad con nivel Descriptive	Si / No
Calidad del código fuente	Mala / Buena / Excelente
Complejidad del proyecto	Baja / Media / Elevada

Las últimas tres características del cuadro no se contemplaron en el análisis de las herramientas propietarias. A continuación se presentan los criterios utilizados para la definición de las escalas de valores correspondientes a las categorías de calidad del código fuente y complejidad del proyecto.

Calidad del código fuente: la calidad del código se consideró **mala** cuando este carece totalmente de comentarios que expliquen su comportamiento, cuando tiene comentarios pero existe poca o nula documentación sobre su arquitectura interna se catalogó como **buena** y

cuando tiene comentarios y además existe documentación relevante sobre su arquitectura interna se catalogó como **excelente**.

Complejidad del proyecto: la complejidad del proyecto se definió en base a la cantidad de clases que lo componen. Menos de 500 clases se catalogó como complejidad **baja**, entre 500 y 1000 clases como complejidad **media**, y mas de 1000 clases complejidad **elevada**.

El proceso de análisis se dividió en dos etapas, una primera de evaluación de herramientas propietarias, y una segunda de evaluación de herramientas Open Source.

3.2 Herramientas propietarias

La conformidad con el nivel 1 de BPMN no fue tomada en cuenta debido a que como estas herramientas son pagas, no se considero necesario analizar este ítem en detalle ya que al no poder ser utilizadas como base para el prototipo este dato pasa a ser secundario. La calidad del código y complejidad del proyecto no se analizaron debido a que sus fuentes no están disponibles.

A continuación en la Tabla 14 se presenta resumen del análisis realizado.

Tabla 14: Resumen análisis herramientas propietarias

Nombre	Simulación	BpSim	Exporta a BPMN2	Importa de BPMN2
ADONIS CE	Si	No	No	No
Altova Umodel	No	N/C	No	No
ARIS Express	No	N/C	No	No
Bizagi BPM Suite + Modeler	Si	Si	No	No
Borland Together	Si	No	No	No
Enterprise Architect	No	N/C	Si	Si
Yoaqiang BPMN V3	No	N/C	Si	Si
Igrafx Process	Si	No	No	No
INNOVATOR for Business	No	N/C	Si	No
Signavio Process Editor	Si	No	Si	Si
BPMN Web Modeler	Si	Si	Si	Si
L-SIM	Si	Si	Desconocido	Desconocido

Para las herramientas Genexus WorkFlow, IBM Process Designer y L-SIM no fue posible conseguir versión para realizar pruebas. Tampoco se pudo probar Visio 2013 porque requiere Windows Vista o superior y solo se contaba con una Maquina Virtual con Windows XP para realizar las pruebas.

La herramienta L-SIM es un caso particular ya que fue la única en su tipo que se encontró. L-SIM es un servidor de simulación de procesos de negocio en BPMN 2.0 sobre el estándar BPSim. Si bien en la página web oficial de producto se comenta que brinda soporte para BPSim no fue

posible comprobarlo debido a que no existe la posibilidad de descargar la herramienta para realizar pruebas. En la página del producto también se menciona que es utilizado como servidor de simulación por otras herramientas (Incluidas suites BPMS), como por ejemplo Oracle BPA Suite, SAP Business Process Optimization, etc.. Tampoco fue posible conseguir una versión de prueba de estos productos. Debido a que no fue posible conseguir una versión de prueba de L-SIM ni de ninguno de los productos que lo incorporan no se realizo un análisis detallado de este producto.

Varias de las herramientas analizadas brindan soporte para simulación, pero solo dos de ellas brindan soporte de simulación sobre BPSim. Otro dato a resaltar es que la mayoría de ellas no brinda funcionalidades de importación/exportación de modelos en BPMN 2.0 (6 no exportan y 7 no importan). De las herramientas Propietarias analizadas se destacan dos en particular, Bizagi Modeler y BPMN Web Modeler. Ambas soportan simulación de procesos pero además basada en BpSim.

3.3 Herramientas Open Source

A continuación en la Tabla 15 se presenta resumen del análisis realizado.

Tabla 15: Resumen análisis herramientas Open Source

Nombre	Simulación	BpSim	Exporta a BPMN2	Importa de BPMN2	CL1	Calidad código	Complejidad proyecto
Activiti Designer y Modeler	No	N/C	Si	Si	No	Excelente	Elevada
Camunda Modeler	No	N/C	Si	Si	Si	Excelente	Elevada
Yaoqiang BPMN 2.2	No	N/C	Si	Si	Si	Buena	Baja
JBPM Tools	Si	Si	Si	Si	No	Buena	Media
Modelio	No	N/C	No	No	No	Excelente	Elevada
Eclipse BPMN2 Modeler	No	N/C	Si	Si	Si	Buena	Media
Bonita BPM	Si	No	No	No	Si	Buena	Elevada

En el caso de las herramientas Open Source un porcentaje menor de ellas soporta simulación (28,5% frente al 54,5% de las herramientas propietarias). Por otro lado podemos apreciar claramente como se prioriza la interoperabilidad entre herramientas soportando todas ellas (A excepción de Modelio) importación y exportación de modelos en BPMN 2.0. Otro punto relevante es que solo JBPM brinda soporte para simulación de procesos usando BPSim.

En general, la calidad del código analizado es buena, todas ellas poseen un mínimo de comentarios al menos en las partes mas relevantes del código. La complejidad del código (Sin considerar Yoaqiang que es solamente modelador de procesos) no es despreciable, todos ellos son proyectos conformados por diversos módulos (en algunos casos mas de 20 sub proyectos) donde se ven involucrados diversos framework y tecnologías.

De este conjunto destacan dos en particular, Activiti y JBPM. Ambas son parte de suites BPMS que incluyen entre sus componentes motor de workflow, haciéndolas sumamente interesantes a los efectos de la tesis pues dentro de sus objetivos se encuentra modificar un motor de procesos

para convertirlo en simulador.

3.4 Conclusiones

Actualmente existe numerosas herramientas de modelado de procesos en BPMN2. En este trabajo se analizaron 21 de ellas, las mas relevantes, pero a priori y sin mucho esfuerzo se pueden encontrar mas de un centenar de ellas.

Del análisis realizado se desprenden las siguientes conclusiones:

Herramientas propietarias:

- Existen muchas mas opciones de herramientas propietarias que Open Source.
- En comparación la calidad de las herramientas de modelado tanto propietarias como Open Source no difieren demasiado, no pasa lo mismo con las de simulación donde las herramientas propietarias son considerablemente superior a las de código libre.
- En general muchas de ellas manejan un formato de archivo propietario, y no proveen funcionalidades de importar o exportar modelos en el formato nativo de BPMN2, es decir, archivos XML, lo que hace que la interoperabilidad entre herramientas sea limitada.
- Varias de estas herramientas soportan simulación, aunque la mayoría lo hace de una forma muy rudimentaria (Exceptuando Bizagi y BPMN Modeler).
- De las herramientas que soportan simulación, dos de ellas brindan soporte para el estándar BpSim.

Herramientas Open Source:

- Proyectos activos, con una comunidad que le brinde soporte, bien documentados encontramos pocos, lo que si existen son diversos “proyectos personales” y académicos pero que no llegan a tener un nivel como para ser utilizadas profesionalmente.
- La mayoría de las herramientas de modelado en este mundo se pueden clasificar en 2 categorías, plugin de eclipse o aplicación web, en el segundo caso la mayoría de ellas comenzaron en algún momento del tiempo como un fork del proyecto Oryx [Oryx06].
- Estas herramientas no están pensadas desde el punto de vista de un usuario final de negocios, sino desde la óptica de un desarrollador, sobre todo las que se presentan como plugin de eclipse.
- Como es de suponerse, la gran mayoría de estas herramientas soporta importación y exportación de archivos BPMN2 en XML, lo que facilita la interoperabilidad con otras herramientas.
- Prácticamente no existen herramientas de simulación, de las siete analizadas solo JBPM provee esa funcionalidad, además de proveer soporte para BpSim.

- La mayoría de estas herramientas no dan soporte completo al primer nivel de conformidad de BPMN2. La principal razón sea seguramente que la mayoría de ellas forma parte de una suite BPMS, por lo que la paleta de modelado esta acotada a los elementos soportados por el motor de workflow asociado a la suite.
- Las herramientas web de modelado tienen una excelente usabilidad, por lo que se puede llegar a desarrollar procesos complejos en poco tiempo.

Luego de la evaluación 4 herramientas destacaron sobre las demás:

- Bizagi BPM
- BMN Web Modeler
- Activiti
- JBPM

Las primeras dos son la que mejor soporte de simulación brindan de todas las herramientas analizadas, y además ambas soportan BPSim. Lamentablemente al ser propietarias no esta disponible su código fuente para realizar un análisis en mayor profundidad.

JBPM es la única herramienta Open Source que soporta simulación de procesos mediante BPSim, igualmente si la comparamos con las dos anteriores tiene menor cubrimiento. El soporte para simulación parece estar claramente en una fase beta, donde recién se esta poniendo el foco en integrar a la suite BPMS este tipo de herramienta.

Los componentes de modelado de Activiti (Modeler y Designer) no destacan particularmente en nada de otros modeladores analizados, sobre todo si lo comparamos con modeladores Open Source. Todos estos son muy parecidos y están basados prácticamente en las mismas tecnologías, plugin de eclipse utilizando Graphiti, Eclipse EMF, etc., y aplicaciones web que surgen como un fork del proyecto Oryx.

El atractivo principal de los modeladores de Activiti es que al ser parte de una suite BPMS, también cuenta con un motor de workflow, y tanto modelador y workflow se integran perfectamente (La paleta de elementos BPMN 2 soportada por los modeladores se ajusta perfectamente a los elementos soportados por el motor de procesos).

Para finalizar, de las mas de de 70 herramientas con las que se comenzó la investigación inicialmente solo se pudo encontrar 3 que soportan simulación de proceso basados en BPSim, de lo que se puede desprender que el uso de BPSim como estándar de simulación para la construcción de herramientas todavía no está muy adoptado en el ámbito profesional ni lo académico.

3.5 Actualización herramientas de soporte

Previo a la finalización del trabajo de tesis se realizo una nueva búsqueda con el objetivo de visualizar como evolucionó el mercado de las herramientas con soporte para BPSim desde el momento que se elaboro el estado del arte a la fecha actual.

Luego de transcurridos aproximadamente 2 años desde el primer análisis se pudo apreciar que no a variado muy poco el mercado de herramientas con soporte para BPSim. Solo se pudo encontrar dos herramientas asociadas a la simulación de procesos de negocio que no estaban disponibles

en primera instancia, BIMP y Enterprise Architect de la empresa Sparx Systems.

3.5.1 BIMP – THE BUSINESS PROCESS SIMULATOR

BIMP [BIMP15] es un simulador de procesos de negocio que se encuentra disponible en tres sabores diferentes, como aplicación en la nube (Esta es la versión disponible para pruebas), como API REST y como API Java. Estas dos últimas implementaciones no están disponibles para prueba.

Si bien BIMP es un simulador de procesos de negocio que comparte muchos conceptos propuestos por BPSim y en una primera impresión transmite la sensación de que se basa en este estándar, luego de una investigación exhaustiva se comprobó que no es así, si bien es un simulador de procesos de negocio no esa construido sobre BPSim.

Durante la investigación se llego al sitio web http://fundamentals-of-bpm.org/supplementary-material/miscellanea/, sitio que corresponde a curso online dictado por la Universidad de Tecnología de Queensland por el profesor Marlon Dumas (Contribuyo en el trabajo académico de investigación que tuvo como resultado a BIMP) . En dicho curso hay un capítulo dedicado a la herramienta BIMP. Dentro del material de este capítulo se puedo encontrar dos archivos con ejemplos de escenarios de simulación parametrizados para su uso con la herramienta BIMP. Cuando se analiza los archivos XML que contienen el proceso a simular se puede apreciar que la información del escenario de simulación esta embebida en el archivo del proceso de negocio pero no esta expresada en el formato propuesto por BPSim, sumado a que en el sitio oficial no se hace mención a la adopción de dicho estándar se llego a la conclusión que BIMP no implementa BPSim.

3.5.2 ENTERPRISE ARCHITECT

Enterprise Architect fue una de las herramientas propietarias que se probó en su momento y se llego a la conclusión que solo soportaba animación de token. En esta nueva revisión de herramientas se encontró información de que la versión 12.1 de Enterprise Architect (Se había probado la versión 10) soporta simulación de proceso de negocio BPMN 2.0 mediante BPSim.

Se procedió a bajar el trial de la versión 12.1 del siguiente link, http://www.sparxsystems.com/products/ea/trial.html. Luego se siguieron los pasos detallados en el tutorial disponible en la página web que se menciona a continuación:

http://www.sparxsystems.com/enterprise_architect_user_guide/12.1/business_engineering/setup_bpsim.html

Luego de completar el primer paso del tutorial, “Create a Business Process Model” se procedió a realizar el segundo paso, “Create a Business Process Simulation Artifact”. No fue posible completar este paso ya que el Artifact “Business Process Simulation” que se sugiere utilizar no esta disponible, por lo que no fue posible probar las funcionalidades de simulación de esta herramienta.

3.5.3 CONCLUSIÓN

Luego de evaluar la aparición de nuevas herramientas en el mercado que brinden soporte para la

simulación de procesos de negocio en BPMN 2.0 sobre BPSim se llego a la conclusión que la situación no varió demasiado con respecto al análisis realizado al comenzar el trabajo de tesis.

Solo se encontró una herramienta previamente evaluada que en su nueva versión incorporó soporte para BPSim (Enterprise Architect) la cual no fue posible evaluar en detalle debido a que aparentemente la versión trial no provee el soporte para la simulación de procesos sobre BPSim. En resumen, en el transcurso de estos dos años la cantidad de herramientas que brindan soporte a BPSim se mantiene prácticamente igual.

4 Análisis del prototipo

En este capítulo se presenta el análisis del prototipo a construir. En base al conocimiento adquirido durante la etapa de estado del arte se detallaran los requerimientos básicos a cubrir por el simulador, decisiones relevantes de diseño y las diferentes iteraciones para llegar a su estado actual.

4.1 Introducción

Para la construcción del prototipo de herramienta que permita simular un proceso de negocio modelado con BPMN 2.0 con base en el estándar BPSim.

Se definieron los siguiente requisitos:

1. Los procesos de negocio a simular deben estar modelados en BPMN 2.
2. El prototipo de simulador se tiene que basar en el estándar BPSim para simulación de procesos de negocio.
3. La interfaz del simulador debe de ser amigable al usuario, por lo que un usuario de negocio tiene que ser capaz de simular un proceso sin mayores dificultades.
4. No solo la interfaz debe de ser amigable, el resultado de la simulación se deberá presentar en forma de gráficos o tabular que permitan una fácil comprensión.
5. La solución a presentar se basará en componentes de software existentes con la finalidad de reutilizar lo más posible propuestas existentes y “no reinventar la rueda”. Se realizarán las modificaciones necesarias para adaptarlos a las necesidades del prototipo.

Como elementos principales del diseño de la solución, se definió que el prototipo estará conformado por dos grande módulos interconectados tanto entre ellos como al motor de procesos que sea tomado como base:

1. Motor de simulación
2. Herramienta gráfica de simulación

Para la construcción del framework de simulación se tomará como componente base un motor de WorkFlow que brinde soporte para el estándar BPMN 2.0. Este motor se deberá basar en la filosofía Open Source. Además se tendrá en cuenta que se pueda ejecutar en un ambiente standalone (básicamente que este implementado sobre POJOs), de esta manera se asegura que el simulador pueda ser incluido en cualquier tipo de ambiente (Web, Entreprise, etc.).

En el caso de la herramienta gráfica se tomara como base un modelador de procesos web existente, y se le realizarán las modificaciones necesarias para incluir las funcionalidades de simulación requeridas. El hecho de que la herramienta seleccionada sea web se debe al punto 3 mencionado anteriormente; este tipo de herramientas en general ya de por si son mas amigables con el usuario final, por lo que son mas aptas para usuarios de negocio.

Un punto no menor a destacar es que ambos componentes deberán ser 100% compatibles entre si, con el objetivo de minimizar las modificaciones a realizar y reducir los riesgos de incompatibilidades. Una selección acertada sería apuntar a una solución BPMS, que provea tanto motor de WorkFlow como editor web de procesos de negocio.

Como se observó en la etapa de investigación de herramientas, las aplicaciones de modelado que forman parte de una suite de BPMS proveen una paleta de elementos para modelar acorde al motor de WorkFlow incluido en la suite, y los motores soportan el estándar BPMN 2 pero no al 100%, es muy común que tengan pequeñas modificaciones propias de cada implementador que hace que sean incompatibles con herramientas de modelado de terceros.

4.2 Requerimientos

Considerando las funcionalidades a proveer por el prototipo, se definió realizar el proceso de desarrollo del prototipo en tres iteraciones e ir incorporando nuevos requerimientos en cada una de ellas. Parte del trabajo realizado en esta etapa de diseño del prototipo incluyo determinar a que dimensiones y propiedades de BPSim se les daba soporte y a cuales no.

A continuación en la Tabla 16 se listan las propiedades expuestas por el estándar y se detalla a cuales se le dará soporte en el simulador y a cuales no.

Tabla 16: Propiedades BPSim soportadas por el simulador

Propiedades/Dimensión	Control	Tiempo	Costos	Recursos
fixedCost			Si	
UnitCost			Si	
interTriggerTimer	Si			
triggerCount	Si			
condition	No			
probability	Si			
availability				No
quantity				Si
selection				No
role				No
transferTime		Si		
quequeTime		Si		
WaitTime		Si		
SetupTime		Si		
ProcessingTime		Si		
ValidationTime		Si		
reworkTime		Si		

Cabe destacar que de las seis perspectivas de BPSim se brindo soporte a cuatro de ellas, control, tiempo, costos y recursos. Sin duda estas cuatro perspectivas además de ser las fundamentales son la que aportan mayor valor agregado a la hora de simular un escenario y obtener información sobre dicha simulación. Las perspectivas Prioridad y Propiedades aportan valor en casos mas específicos de simulación y no en los escenarios mas comunes, por lo que se decidió postergar su implementación y pasar a tenerlas en cuenta para trabajos a futuro de forma de complementar el soporte anterior.

En esta etapa también se debió definir a que elementos de BPMN 2.0 se le brindaba soporte con la finalidad de acotar el alcance del prototipo y que fuera posible realizarlo dentro de los plazos establecidos. El criterio establecido fue optar por dar soporte a los elementos de modelado mas utilizados, que generalmente son los que abarca el nivel 1 de conformidad para BPMN 2.0.

A continuación se detallan los elementos básicos de BPMN 2.0 que componen el nivel 1 de conformidad:

- Activity: Task (User, Service, Abstract), Subprocess, Call Activity
- Gateway: Exclusive, Parallel
- Start event: None, Message, Timer
- End event: None, Message, Terminate
- Sequence flow and Message flow
- Pool and Lane
- Data object, Data store, and Data association
- Documentation
- Artifact: Text annotation, Association, and Group

Cabe destacar que ciertos elementos de modelado mencionados anteriormente tienen solamente un efecto visual sobre el diagrama y no afectan la simulación en si. Por ejemplo los diferentes tipos de artifacts.

En el caso de las tareas se decidió ampliar el soporte agregando los siguientes elementos:

- Send Task
- Receive Task
- Manual Task
- Script Task
- Business Rule Task

Como se presento en la sección 2.5.2, “BPSim – Perspectivas de Simulación” del capítulo Estado del Arte, en la cual se puede apreciar con un ejemplo la definición de atributos BPSim y su relación con elementos BPMN 2.0, a continuación en la Tabla 17 se puede apreciar el detalle completo de las perspectivas y parámetros de BPSim soportadas y su asociación con los elementos de BPMN 2.0

Tabla 17: Perspectivas y atributos BPSim soportados y su relación con elementos BPMN 2.0

Perspectiva	Aplica a	Atributos BPSim soportados
Tiempo	Tareas y subprocesos	WaitTime, processingTime
Costo	Tareas, subprocesos y recursos	FixedCost, UnitCost
Control	Eventos	InterTriggerTimer, triggerCount
	Flujo de secuencia	Probability
Recursos	Tareas de usuario	quantity

Por mas información de como aplica cada perspectiva de BPSim a los elementos de BPMN 2.0 ir al documento Anexo A, sección “Introducción a BPSim”.

El trabajo fue pensado para ser desarrollado en forma incremental, comenzando en primera instancia por el simulador y continuando luego con el modelador de procesos. Fue necesario dividir el desarrollo en tres iteraciones, la primera se concentro en desarrollar el simulador y la implementación de las funcionalidades básicas para poder simular un proceso de negocio inicialmente por linea de comandos. En la segunda iteración el foco se puso sobre la herramienta de modelado, incorporando la perspectiva simulación e integrando este módulo con el simulador. Finalmente la tercera instancia se centró en el soporte a la perspectiva recursos por parte de ambos módulos y además se potenciaron alguno de los requerimientos de etapas anteriores.

En la Tabla 18 se muestra las funcionalidades a implementar organizadas por iteración.

Tabla 18: Detalle de funcionalidades a implementar por iteración

Funcionalidad	Iteración 1	Iteración 2	Iteración 3
F01 - Soporte al metamodelo de BPSim	X		
F02 - Parametrización de escenario e implementación de perspectiva de control	X		
F03 - Implementación de perspectiva tiempo	X		
F04 - Implementación de perspectiva costos	X		
F05 - Soporte a elementos básicos de BPMN 2.0	X		
F06 - Implementación de tipos de parámetros básicos de BPSim	X		
F07 - Inclusión de propiedades de simulación al modelador		X	
F08 - Exportación a formato XML		X	
F09 - Implementación de tipos de parámetros avanzados de BPSim		X	
F10 - Soporte a elementos avanzados de BPMN 2.0		X	
F11 - Presentación básica de resultados		X	
F12 - Implementación de perspectiva de recursos			X
F13 – Persistir resultados de la simulación de forma transparente			X
F14 – Presentación avanzada de resultados			X

A continuación se describen en detalle las funcionalidades incluidas para ser implementadas en cada iteración definida.

4.2.1 PRIMERA ITERACIÓN

En esta primera etapa el foco fue puesto sobre el motor de simulación, incluyendo únicamente funcionalidades básicas que permitan simular un proceso de negocio independientemente de la interfaz gráfica. A continuación se detallan los requerimientos implementados en esta etapa:

F01 - Soporte al metamodelo de BPSim

Como primer paso fundamental para implementar el simulador surgió la necesidad de poder manipular y validar archivos XML de BPSim con la información del escenario a simular. Se manejaron dos opciones en este punto:

- Utilizar herramientas como Eclipse EMF (Eclipse Modeling Framework) para modelar el metamodelo de BPSim en base a la especificación disponible en la documentación. EMF permite generar automáticamente clases de implementación en Java para los elementos de nuestro metamodelo, pudiendo posteriormente procesar un XML de BPSim, validarlo y consultarlo programaticamente.
- Investigar la existencia de alguna implementación Open Source para Java del metamodelo de BPSim. Este punto requirió volver a revisar las herramientas de simulación analizadas con anterioridad en busca de una implementación del metamodelo. La única herramienta Open Source con soporte de simulación BPSim analizada fue JBPM, y esta poseía una implementación del metamodelo BPSim realizada con Eclipse EMF bajo licencia “Apache License 2.0” [Apache04], por lo que se reutilizo el código en la construcción del prototipo.

F02 - Parametrización de escenario e implementación de perspectiva de control

En este requerimiento se incluyen las funcionalidades básicas para poder parametrizar un escenario y ejecutar una simulación sin la intervención del usuario. Estas funcionalidades brindan la posibilidad de determinar cuantas instancias se van a simular y poder especificar el porcentaje de probabilidad de ejecución de cada PATH del modelo. En la figura 16 se presenta un ejemplo.

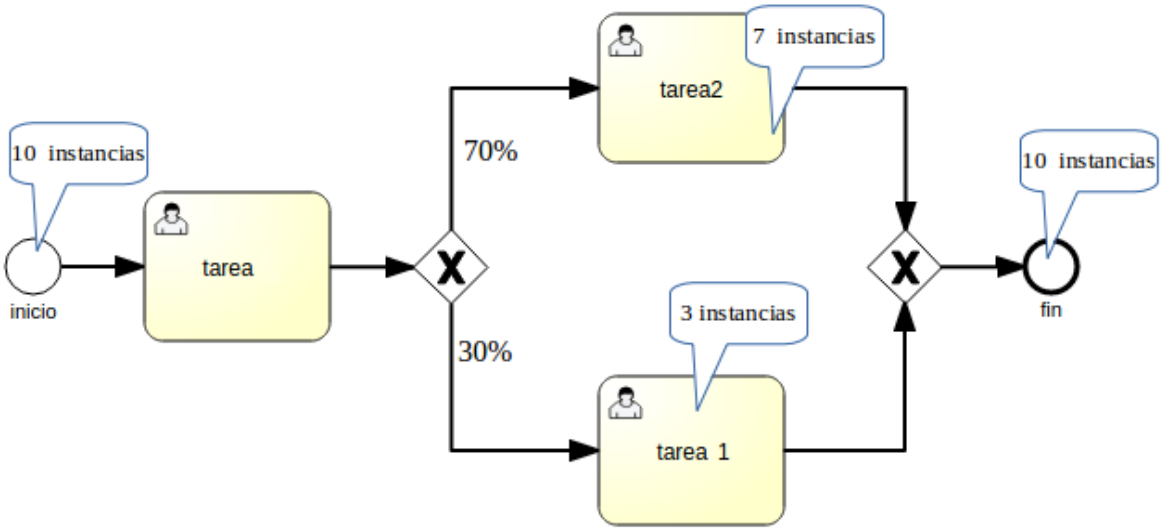


Figura 16: Pathfinder - Ejemplo de escenario

Como se puede apreciar en la Figura 16, el escenario consta de 10 instancias a simular y dos posibles caminos (Paths) de ejecución. Siendo el nodo de decisión un XOR cada tipo de instancia ejecutará solo un camino, que dependerá en ejecución de la condición evaluada. En simulación se necesita esta información como parámetro de entrada “representando” lo que pasaría cuando la condición se evalúa realmente. A continuación se describen los dos posibles caminos de ejecución.

Path 1

En la Figura 17 se muestra el primer de los dos caminos posibles. En esta caso el camino corresponde a la ejecución de las tareas: tarea y tarea2, que será recorrido por el 70% de las instancias del proceso a ejecutar. Estos valores en general se obtienen de dos formas, del análisis

de información histórica de ejecución del proceso o de los usuarios expertos en el negocio.

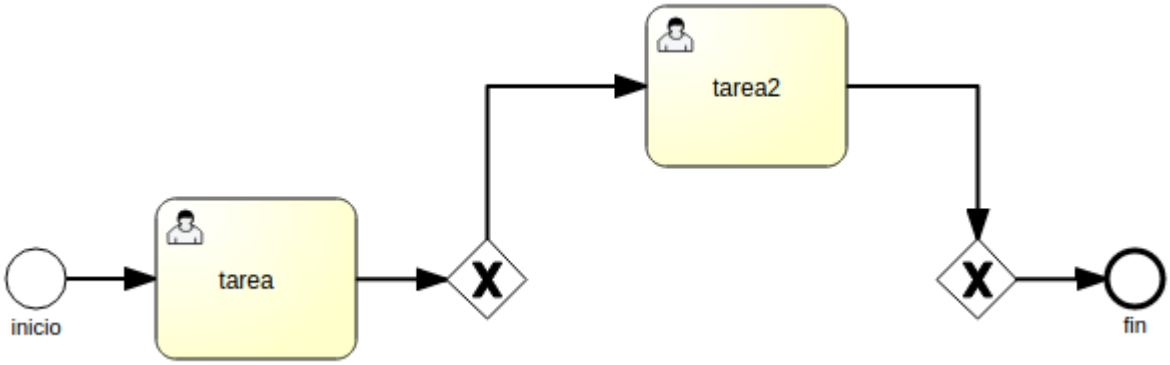


Figura 17: Pathfinder - primer path identificado

Path 2

En la Figura 18 se muestra el segundo de los caminos posibles. En esta caso el camino corresponde a la ejecución de las tareas: tarea y tarea 1, que será recorrido el 30% de las instancias del proceso a ejecutar. El path 2 se ejecutará un total de 3 veces mientras que el path 1 se ejecutará un total de 7 veces.

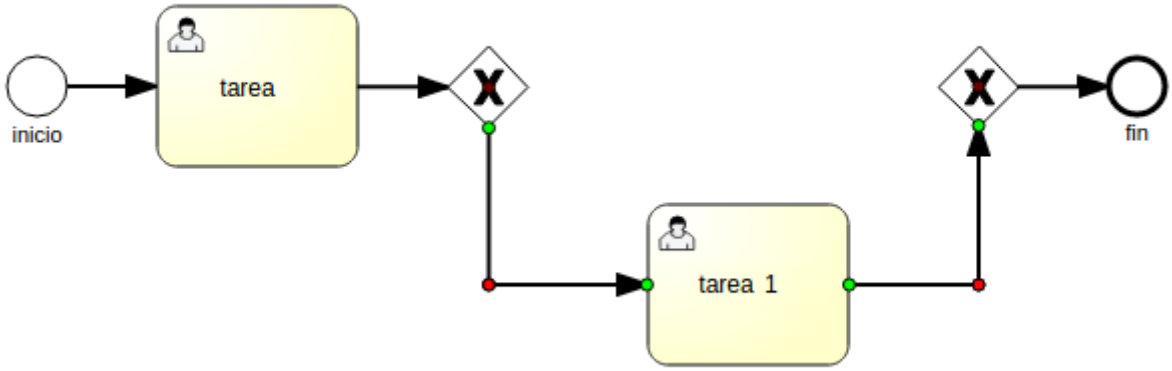


Figura 18: Pathfinder - segundo path identificado

F03 - Implementación de perspectiva tiempo

Este requerimiento además de la implementación de los parámetros de BPSim asociados a la perspectiva tiempo implica la construcción de los componentes necesarios para controlar el reloj de la simulación (ver sección 3.1.2 - “Componentes de un DES”).

La perspectiva tiempo abarca la implementación de 7 parámetros propuestos por BPSim, de los cuales los siguientes son clave:

- **Processing Time (Tiempo de procesamiento):** es el tiempo empleado para completar el trabajo actual, por ejemplo una tarea de usuario.

- **Wait Time (Tiempo de espera):** es el tiempo transcurrido entre que la tarea es asignada y esta comienza a ser ejecutada.

Estos son los dos parámetros mas relevantes de la perspectiva y los que están presentes en la mayoría de los simuladores analizados. Los 5 parámetros restantes de la perspectiva son los siguientes:

- **Transfer Time (Tiempo de transferencia):** es el tiempo que transcurre desde que finalizo el paso anterior y se llego al paso actual.
- **Queue Time (Tiempo de cola):** es el tiempo transcurrido entre que la tarea esta disponible y la misma es asignada.
- **Setup Time (Tiempo de preparación):** es el tiempo gastado preparándose para realizar el trabajo actual.
- **Validation Time (Tiempo de validación):** tiempo empleado revisando el trabajo realizado.
- **Rework Time (Tiempo de retrabajo):** tiempo empleado corrigiendo el trabajo realizado.

F04 - Implementación de la perspectiva costos

Este requerimiento implica la implementación de los parámetros implicados en la perspectiva costos. Estos parámetros son:

- **Fixed Cost (Costo fijo):** Especifica el costo fijo que se pagará por única vez cada vez que se realice la “tarea”. Por defecto esta expresado en la moneda definida por defecto como parámetro del escenario (BaseCurrencyUnit).
- **Unit Cost (Costo unitario):** Especifica el costo que se pagará por cada unidad de tiempo que insuma la “tarea”. Al igual que en el caso anterior, por defecto esta expresada en la moneda del escenario.

En esta primera instancia se incluyó el soporte de esta perspectiva para los diferentes tipos de actividades presentes en el nivel descriptive (Básicamente diferentes tipos de tareas), dejando el soporte de costos asociado a recursos para una siguiente iteración.

F05 - Soporte a elementos básicos de BPMN 2.0

Ya se menciono con anterioridad que se daría soporte a los elementos de BPMN 2.0 que pertenecen al nivel uno de conformidad y a elementos particulares que son generalmente utilizados en el modelado de procesos pero no pertenecen al nivel uno, sino a niveles superiores (Ejemplo, rule task, intermediate events).

En esta primera instancia se dio soporte a los siguientes tipos de elementos:

- Evento de inicio
- Evento de fin
- Gateways
  - exclusivo
  - paralelo
- Actividades
  - Tarea de usuarios
  - Tareas manuales
- Conectores

- Flujos de secuencia (Sequence flow)

Estos son los elementos básicos de modelado a los cuales se les dio soporte en esta primera etapa, dejando para las siguientes etapas elementos mas complejos como ser, sub procesos, eventos intermedios, diferentes tipos de eventos de inicio y fin, etc.

F06 - Implementación de tipos de parámetros básicos de BPSim

Los diferentes atributos pertenecientes a las perspectivas mencionadas anteriormente están asociadas a un parámetro y este parámetro debe proporcionar un valor por defecto. (por mas información ver documento Anexo A, sección 12 – Tipos de Parámetros).

Los tipos de parámetros mas destacados son dos:

- **ConstantParameter:** se basan en una constante, ya sea entera, decimal, cadena de caracteres, etc.
- **DistributionParameter:** Este tipo de parámetro se caracteriza porque su valor puede variar en función del tiempo acorde a un determinado tipo de distribución dado.

En esta primera instancia se opto por implementar parámetros del tipo constante, dejando el de tipo distribución para las siguientes iteraciones.

4.2.2 SEGUNDA ITERACIÓN

En esta iteración se puso el foco en la herramienta gráfica de simulación, con el objetivo de una vez finalizada la etapa poder simular un proceso con las funcionalidades descriptas en el paso anterior tanto de forma standalone como gráfica. Además del desarrollo de la herramienta gráfica también se complemento el motor de simulación con nuevas funcionalidades. A continuación se detallan los requerimientos implementados en esta etapa:

F07 - Inclusión de propiedades de simulación al modelador

Generalmente los modeladores Web investigados poseen el mismo comportamiento, al seleccionar un elemento del diagrama despliegan un menú contextual con las propiedades de BPMN 2.0 que aplican a dicho elemento. La idea es mantener ese comportamiento y dependiendo de que elemento se seleccione en el diagrama mostrar las propiedades de simulación correspondientes de forma independiente a las propiedades de modelado.

En esta iteración se implementaron las propiedades definidas en la iteración 1 para las perspectivas de control, costos y tiempo, además de las asociadas al escenario de simulación.

F08 - Exportación a formato XML

Esta funcionalidad brinda la posibilidad de visualizar y exportar a formato XML el escenario parametrizado de BPSim, con el objetivo que pueda ser utilizando a futuro en otra herramienta. En caso de la herramienta de modelado base que se seleccione no posea la funcionalidad de visualizar el XML de BPMN 2.0, se implementará dicha funcionalidad.

F09 - Implementación de tipos de parámetros avanzados de BPSim

En esta etapa se dará soporte a los tipos de parámetros “avanzados”. Se seleccionarán algunos de los tipos de distribuciones propuestas por el estándar y se implementarán asociadas a los parámetros de la perspectiva tiempo. Los tipos de distribuciones a implementar serán:

- Distribución normal
- Distribución normal trunca
- Distribución uniforme
- Distribución Poisson

F10 - Soporte a elementos avanzados de BPMN 2.0

En esta instancia se dará soporte a los elementos mas complejos de BPMN 2.0 como ser subprocesos y eventos intermedios. Además se complementarán las categoría de actividades y eventos de inicio y fin, extendiendo el soporte a los elementos de modelado extra que soporte la herramienta utilizada como base (Ejemplo: script task, mail task, start timer, start message, etc.).

F11 - Presentación básica de los resultados

Los resultados de la simulación se presentarán de forma básica en formato de tabla, con la única finalidad de poder apreciar los resultados. Este requerimiento se terminará de desarrollar en la siguiente iteración.

4.2.3 TERCERA ITERACIÓN

En esta iteración se incorporo la perspectiva de recursos tanto al motor de simulación como a la herramienta gráfica, además se terminaron de desarrollar funcionalidades planteadas en las etapas anteriores.

A continuación se detallan los requerimientos implementados en esta etapa:

F12 - Implementación de perspectiva recursos

En primera instancia se incorporo el soporte de la perspectiva recursos al motor de simulación, esto implicó poder definir diferentes pools de recursos y asignarlos a las diferentes tareas del modelo a simular. Finalmente se incorporo a la herramienta gráfica la opción de definición de pools de recursos y asignación de los mismos.

F13 - Persistir resultados de la simulación de forma transparente

Este requerimiento refiere a que el resultado de la ejecución de la simulación debe de dejar constancia en los logs del motor de workflow como si se tratara de una ejecución real de un proceso de negocios, no pudiendo distinguir una simulación de una ejecución manual del proceso.

La finalidad de esta funcionalidad es permitir a futuro extraer la información almacenada en los logs de ejecución y mediante la utilización de técnicas como Minería de procesos (Data Mining) [Aalst11] poder descubrir nuevo conocimiento, y/o analizar los datos registrados. Además también se puede utilizar estos logs para en trabajos a futuro implementar el tercer tipo de parámetros soportado por BPSim, datos históricos.

Presentación avanzada de resultados

Se mejoro la forma de presentar los resultados, logrando ser mas amigable al usuario final. Se graficó una selección de indicadores para ejemplificar los resultados posibles, como ser:

- Tiempo de uso de los recursos
- Tiempo de procesamiento y de espera de las diferentes actividades
- Costos por recurso y actividad
- Cantidad de instancias simuladas, finalizadas y abortadas
- Cantidad de ejecuciones de cada elemento del modelo

5 Diseño del prototipo

En este capitulo se describen los aspectos mas relevantes del diseño del prototipo, se describirán los componentes base seleccionados para su construcción y se ahondará en el análisis de algunos de los requerimientos descritos en el capitulo anterior. En el capítulos seis del documento se profundizará sobre el diseño del prototipo y como se realizo su implementación.

Una de las decisiones mas importantes a tomar fue la selección de los componentes de software a utilizar como base para la construcción del prototipo. Antes de comenzar a diseñar fue necesario conocerlos y analizar su arquitectura, tener en cuenta como esta podría o no afectar al prototipo y comprender la complejidad y costo de realizar las modificaciones necesarias para los objetivos de la tesis.

La primera condición a cumplir por parte de los componentes a utilizar fue que debían ser distribuidos bajo algún tipo de licencia Open Source, también fue determinante el hecho de que tanto el motor de workflow como el editor gráfico de procesos pertenecieran al mismo paquete de software, tratando de evitar de esta manera cualquier inconveniente de incompatibilidad que pudiese surgir.

Del análisis e investigación de herramientas presentado anteriormente se puede apreciar básicamente tres tipos de herramientas:

- **Modeladores (Business Process Modeling Tools):** Este tipo de herramientas están concebidas para describir y analizar procesos de negocio. El resultado obtenido con este tipo de herramientas generalmente solo se utiliza como documentación. El gran inconveniente con este tipo de herramientas es que funcionan de forma aislada y no están integradas en el ciclo de vida de BPM. Igualmente existen herramientas de este tipo que se pueden integrar con herramientas de terceros para pasar a formar parte del ciclo de vida BPM, si por ejemplo implementan el estándar BPMN 2.0.

- **Herramientas BPM (Business Process Management Tools):** Este tipo de herramientas abarcan todo el ciclo de vida BPM, también son conocidas con el nombre de BPMS. Brindan la posibilidad de modelar y posteriormente ejecutar el proceso de negocio en un motor de workflow. Alguna de estas suites incluyen una herramienta de BPS como parte del paquete y varios otros módulos de soporte a las actividades del ciclo de vida de los procesos.

- **Herramientas de simulación de propósito general:** Este tipo de herramientas generalmente soportan diferentes tipos de simulación asociadas a un dominio específico. Alguna de estas permiten simular procesos de negocio pero el gran inconveniente es que lo hacen en un lenguaje especifico que debe aprenderse, y este no tiene nada que ver con BPMN 2. Esto hace que sea más complicado y casi imposible de integrar en el ciclo de vida de BPM. Los productos de esta categoría no fueron tenidos en cuenta en esta tesis.

Como este trabajo de tesis persigue como uno de sus objetivos lograr integrar el prototipo de BPS en en ciclo de vida de BPM la única forma de lograrlo es centrándose en el segundo tipo de herramientas, por lo tanto se agrega una nueva condición a la selección de los componentes, además de ser Open Source deben ser parte de una suite BPMS.

5.1 Componentes seleccionados

De los productos investigados se obtuvieron cuatro posibles candidatos del tipo Open Source, estos candidatos se presentan como suites BPM por lo que integran tanto modelador como motor de procesos. Los posibles candidatos son:

- JBPM
- Bonita BPM
- Camunda
- Activiti

La versión de JBPM evaluada ya integra simulador de procesos y además basado en BPSim, por lo que se descarto como opción, ya que no parece tener demasiado sentido construir un simulador que se integre a su suite de BPM siendo que dispone de una herramienta propia con el mismo propósito.

Bonita BPM es una buena suite de BPM pero su principal contra es que su herramienta de modelado de procesos esta construida sobre Eclipse Workbench, lo que complica enormemente su modificación debido a la complejidad de los componentes que lo conforman. Además integra en su suite BPM la posibilidad de simular procesos, aunque no sigue el estándar BPSim.

Camunda se presentó como una buena opción pero al igual que Bonita BPM su principal contra es que el modelador de procesos esta disponible como plugin de eclipse, siendo difícil de modificar y complejo en su uso para un usuario de negocios. Además cabe destacar que Camunda surge como un fork del proyecto Activiti [Barrez2013].

Por último Activiti cumple con las dos condiciones preestablecidas, es Open Source y como parte de su suite BPM integra tanto motor de procesos como modelador. Este último disponible en dos versiones, como plugin de eclipse y como herramienta web. Además no dispone de ningún tipo de simulador, por lo que el prototipo podría ser una buena contribución a su comunidad.

Activiti es un motor de procesos Open Source escrito en el lenguaje Java compatible con el estándar BPMN 2.0. Nace en el año 2010 como iniciativa de Alfresco con el objetivo de construir el primer motor de procesos BPMN 2.0 bajo licencia de la Apache Software Foundation para integrarlo a su gestor de contenidos empresarial (ECM – Enterprise Content Management).

En marzo de 2010, dos de los desarrolladores claves en el proyecto JBPM, Tom Baeyens y Joram Barrez abandonan RedHat para unirse como desarrolladores y lideres de proyecto.

Si bien sus desarrolladores principales lideraron el proyecto JBPM hasta la versión 4, Activiti fue construido íntegramente de cero, basado en el know how que poseían estos dos desarrolladores en la construcción de herramientas BPM. Debido a esto fue que la primera versión de Activiti fue la 5.0, dando a entender de alguna manera que este proyecto era la continuación de la experiencia ganada por parte de sus desarrolladores líderes en JBPM hasta la versión 4.

Actualmente es una suite de BPM que además del motor de de workflow ofrece otras herramientas que permiten administrar el ciclo completo de un proceso de negocio. En la Figura 19 se presentan los módulos que componen el proyecto.

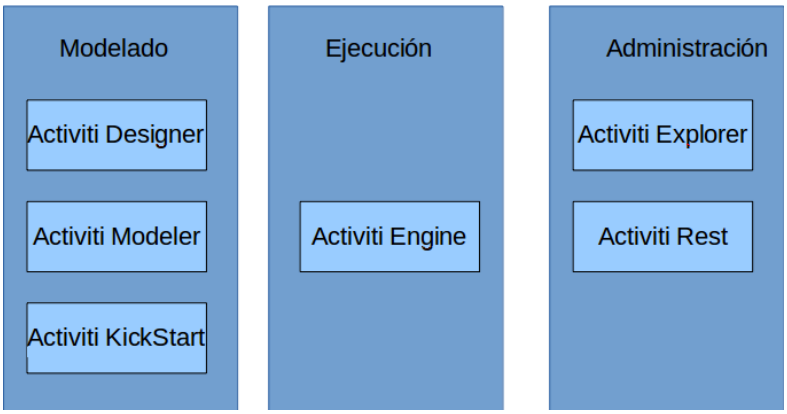


Figura 19: Módulos de Activiti de [Activiti15]

La principal ventaja sobre otros BPMS es que Activiti pone énfasis en la óptica del desarrollador por lo que centra su esfuerzo es permitir la facilidad en integrar el sistema en cualquier entorno Java. Todo esto es logrado sin dejar de lado la óptica de negocio ofreciendo un conjunto herramientas que permiten la colaboración entre usuarios de negocio y desarrolladores. A continuación se describen los componentes más importantes de Activiti.

5.1.1 ACTIVITI ENGINE

El motor de Activiti (Activiti engine) es el núcleo de las herramientas BPM de Activiti, se trata de un motor de workflow construido íntegramente en Java con soporte nativo para el lenguaje BPMN 2.0. Es capaz de desplegar definiciones de procesos, iniciar instancias de esas definiciones, ejecutar tareas de usuario y otras funciones previstas por BPMN 2.0.

Si miramos los cimientos del motor nos encontramos con una “simple” máquina de estados. Cuando hacemos el despliegue (deploy) de una definición de proceso e iniciamos una nueva instancia de este, los elementos de BPMN 2.0 son ejecutados por el motor uno por uno, donde hay un estado activo, estos estados están conectados entre si y la ejecución avanza transitando esas conexiones basándose en determinadas condiciones.

Básicamente la mayoría de los elementos de BPMN 2.0 están implementados como estados y cada uno pude tener una porción de código asociada que se ejecuta cuando la instancia de proceso arriba a dicho estado.

Esta máquina de estado sobre la que esta construido el resto de la librería se conoce con el nombre de “Virtual Process Machine” y es el corazón del motor.

En la Figura 20 podemos apreciar las clases de Activiti encargadas de implementar este concepto.

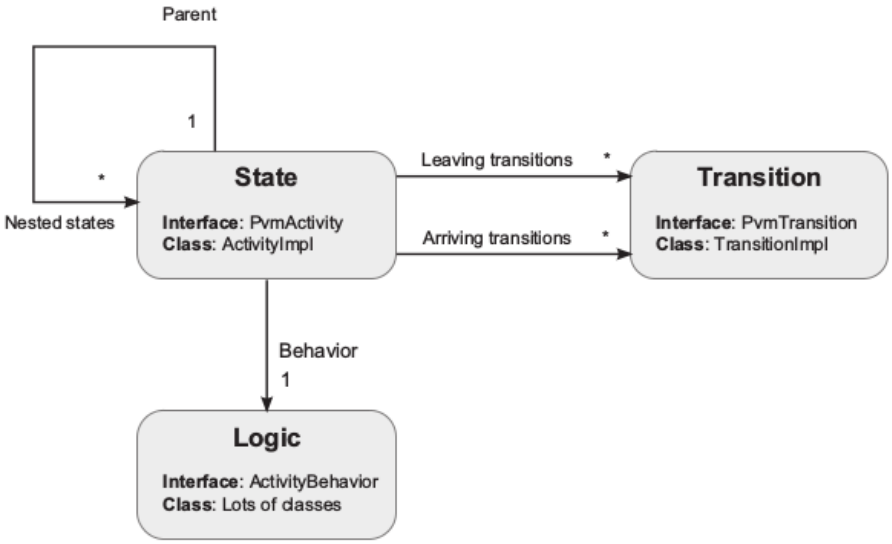


Figura 20: Diagrama Virtual Process Machine de [ActInAc12]

Cada estado tiene transiciones de entrada y de salida, además los estados pueden estar anidados y pueden contener lógica que se implementa en la clase ActivityBehavior. En esta clase es donde esta programado el comportamiento de los diferentes elementos de BPMN 2.0. En la Figura 21 se puede apreciar un diagrama que muestra los componentes principales definidos por el motor.

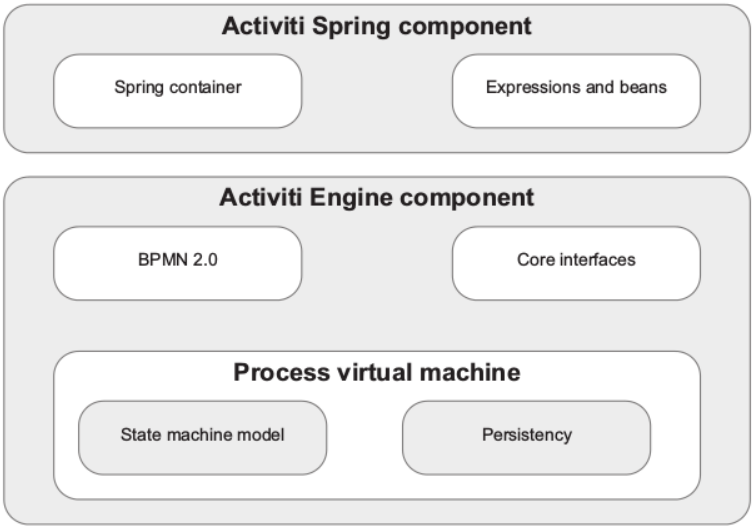


Figura 21: Activiti -Niveles de abstracción de [ActInAc12]

El componente Activiti Spring es opcional y permite integrar Activiti a un contenedor Spring para poder por ejemplo ejecutar Beans de Spring directamente desde una tarea de servicio.
El componente BPMN 2.0 es el encargado de brindar soporte al metamodelo BPMN y el componente Core interfaces expone todos los servicios básicos de Activiti para poder interactuar con el motor. El componente Activiti Engine que incluye la Process Virtual Machine que es la encargada de traducir la lógica del motor desde BPMN 2.0 al modelo propuesto por la máquina de estados.

5.1.2 ACTIVITI MODELER

La suite BPM integra dos herramientas para modelado de procesos, Activiti Desinger y Activiti Modeler. La primera se presenta como plugin de Eclipse con una orientación clara hacia un perfil de usuario técnico, por ejemplo, un desarrollador.

Activiti Modeler es la herramienta de modelado pensada para un usuario final, con un perfil mas orientado al negocio. La mecánica de modelado es prácticamente la misma utilizado por los Plugins de Eclipse (Drag & Drop) y la calidad de los diagramas es excelente.

La herramienta esta disponible en dos modalidades:

- Embebida en Activiti Explorer (Portal de ejecución y administración de Activiti)
- Como aplicación web independiente

Es el mismo modelador, la diferencia básicamente esta en la forma en que se invoca al modelador (desde un acceso dentro del Explorer o directamente por su URL en un navegador Web). Igualmente, por las funcionalidades que el modelador provee esta pensado básicamente para funcionar integrado a Activiti Explorer.

El modelador soporta de forma nativa archivos XML de BPMN 2, permite trabajar con procesos almacenados en el repositorio por defecto de Activiti Explorer o importar un archivo BPMN2 en formato XML. A la hora de importar un archivo es importante tener en cuenta que Modeler básicamente soporta los elementos que el motor de workflow maneja, por lo tanto, si el diagrama contiene otros elementos, los mismos será ignorados.

También permite crear un nuevo proceso y salvar su contenido a archivo XML de BPMN2. No posee funcionalidades que permitan exportar un modelo a archivo PDF por ejemplo, o a cualquier formato de imagen.

En la Figura 22 se presenta un diagrama con la arquitectura de Activiti Modeler.

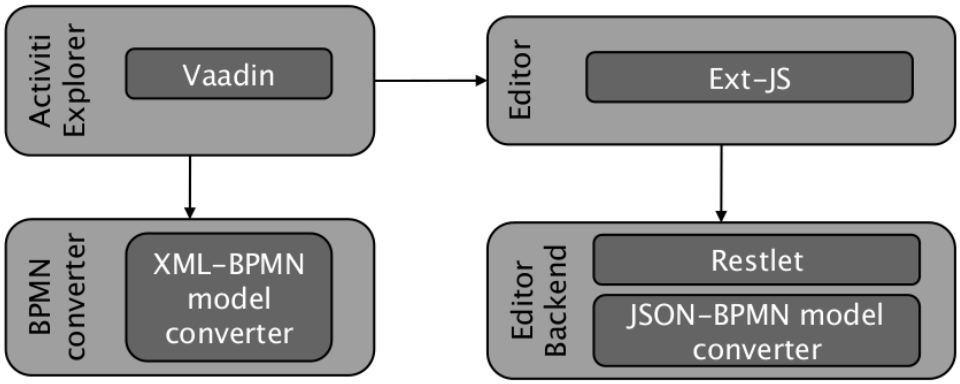


Figura 22: Arquitectura Activiti Modeler de [ActInAc12]

Como se puede apreciar en la Figura 22, Activiti Modeler esta compuesto por estos cuatro módulos:

Activiti Explorer: Esta construido sobre el framework de desarrollo Vaadin. Explorer es la herramienta que provee Activiti como consola de administración de procesos. Activiti Modeler esta integrado a esta consola y desde ahí es que se dispara la ejecución del modelador de procesos.

Editor: El editor gráfico de procesos es una versión Open Source de Kiss BPM. Hasta la versión 5.11 Activiti incorporaba un fork de Signavio Core Components como editor de procesos, desde la versión 5.11 se cambió a este nuevo modelador. Se trato de obtener mas información sobre KISS pero lo cierto es que no existe demasiada documentación técnica al respecto. Igualmente KISS al igual que Signavio Core Components esta construido sobre Oryx por lo tanto comparten básicamente el mismo código fuente. A partir de la versión 5.17.0 se substituyo EXT-JS por Angular JS. Este cambio es sumamente atractivo, ya que el principal problema de Signavio Core Components es que esta construido sobre Ext-JS 2.3, que es una versión obsoleta desde hace varios años.

Se trato de obtener mas información sobre KISS pero lo cierto es que no existe demasiada documentación técnica al respecto. Igualmente, KISS al igual que Signavio Core Components esta construido sobre Oryx por lo tanto comparten básicamente el mismo código fuente.

Los otros dos módulos (BPMN Converter y Editor BackEnd) básicamente proveen la lógica y funcionalidades necesarias de validación y conversión que permiten trabajar con los modelos BPMN2 en formato XML y JSON. Adicionalmente el repositorio de modelos es persistido en la base de datos del motor. La Figura 23 presenta el diagrama definido para la persistencia.

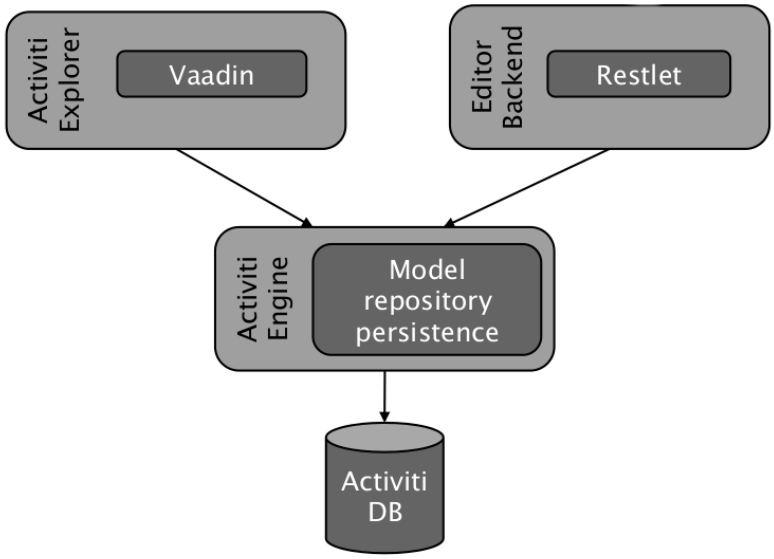


Figura 23: Persistencia en Activiti de [ActInAc12]

En la Figura 23 se muestra como están armadas las funcionalidades de persistencia del Modeler. Por defecto, al igual que el portal de ejecución Activiti Explorer, el Modeler viene configurado con H2 Database Engine ejecutando en memoria (Implica que se pierde la información de los procesos creados con Modeler cuando se da de baja el servicio).

Tanto Activiti Explorer como el Editor comparten la API de persistencia de modelos proporcionada por el engine de Activiti (Motor de workflow). Es posible modificar el comportamiento del engine para que utilice otros motores de base de datos, como por ejemplo MySql, Postgres, entre otros.

5.2 Arquitectura

La arquitectura del prototipo viene dada por la arquitectura definida por Activiti, es decir, se trato de respetar la arquitectura base de los componentes a modificar introduciendo la menor cantidad de cambios posibles con el objetivo de acoplar el modulo de BPS a la suite sin interferir con ninguna de las funcionalidades ya disponibles.

Como se menciono anteriormente, el prototipo consta de 2 partes, el motor de simulación y la herramienta visual de simulación de procesos. El simulador se construyo a partir del motor de workflow de Activiti y se trato de respetar la API original con la idea que continué siendo familiar a los usuarios que ya la conocen, se agregaron nuevas funcionalidades y se modificaron existentes para dar soporte a la simulación.

En esta parte del prototipo fue donde se introdujeron la mayor cantidad de cambios al componente de base, ya que la idea es acoplar el simulador a la suite y seguir manteniendo el motor de workflow original de Activiti, por lo que se disponía de una mayor libertad a la hora de diseñar e implementar. En la Figura 24 se pueden apreciar los componentes originales de la suite indicando en rojo el impacto de los cambios realizados.

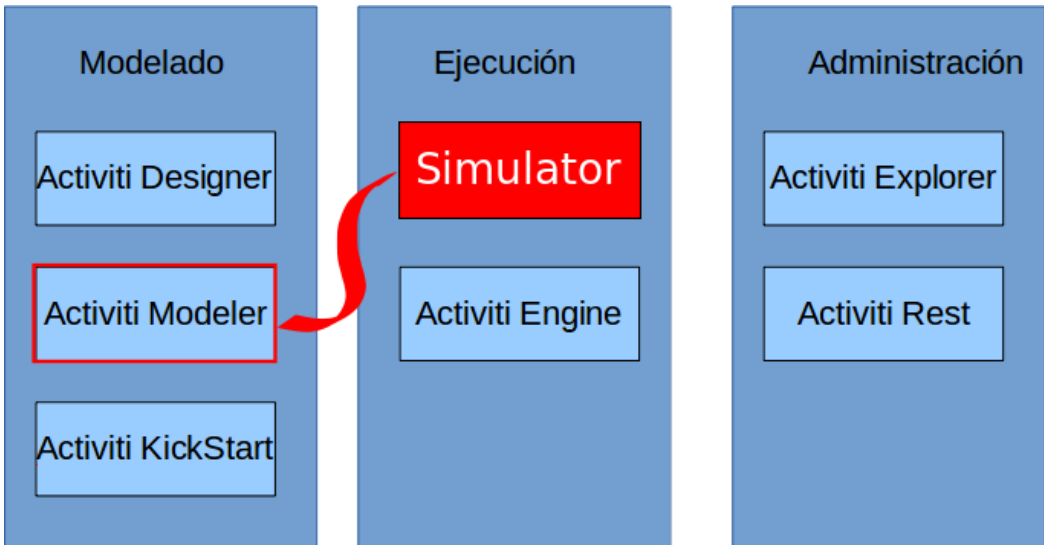


Figura 24: Impacto del simulador en los módulos de Activiti de [Activiti15]

En la Figura 24 se presenta el simulador del motor de procesos de forma separada, aunque cabe destacar que el simulador podría perfectamente actuar como motor de procesos ya que los cambios realizados no afectaron su comportamiento con el archivo de configuración correcto.

En la Figura 25 se presenta diagrama de los componentes mas relevantes tanto del simulador como del modelador.

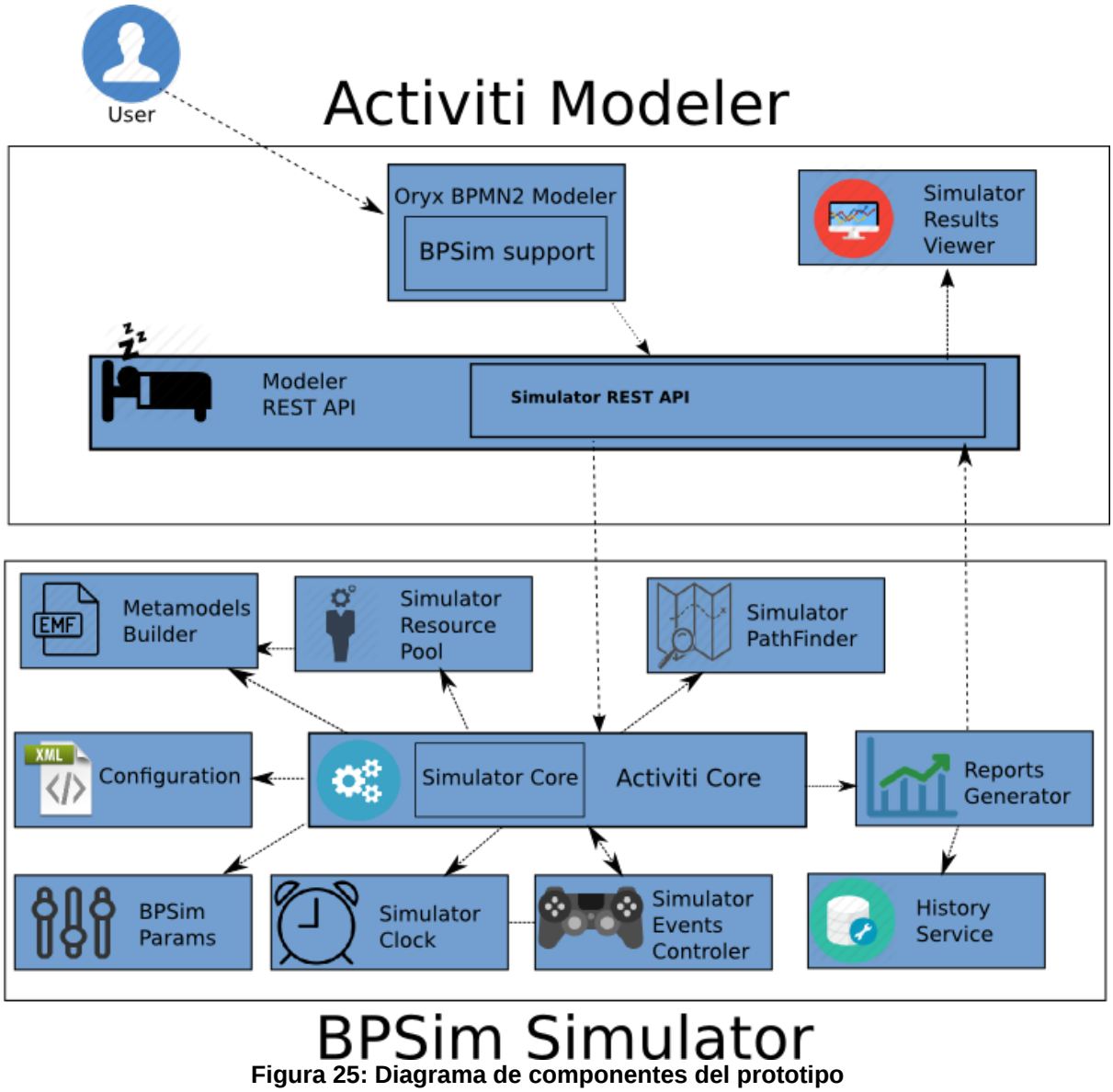


Figura 25: Diagrama de componentes del prototipo

5.2.1 MODELADOR

A continuación se describen los módulos que se presentan en la Figura 25 para el Modelador de procesos.

Oryx BPMN 2 Modeler: Este módulo corresponde a la parte visual del modelador de procesos. Fue necesario modificarlo y extenderlo para agregar el soporte a los elementos de simulación.

Simulator results viewer: Este módulo es el responsable de procesar el resultado devuelto por el simulador y mostrarlo de forma gráfica y amigable.

Modeler Rest API: Este módulo viene por defecto con Activiti y es el encargado de proveer servicios Web REST para comunicar el modelador con el motor de procesos. Fue necesario incorporar nuevos servicios REST específicos para comunicar al modelador con el motor de simulación.

5.2.2 SIMULADOR

Metamodels Builder: Este módulo es el encargado de transformar los modelos de BPMN 2 como de BPSim desde XML a JSON y viceversa.

Simulator Resource Pool: Este módulo es el encargado de brindar las funcionalidades de pool de recursos necesarias para simular la perspectiva recursos de BPSim.

Simulator Pathfinder: Este módulo es el responsable de identificar previa ejecución los diferentes caminos a simular y su probabilidad de ocurrencia.

Configuration: Archivo de configuración de Activiti especialmente configurado para el simulador de procesos. Desde este punto se puede modificar el comportamiento del simulador, por ejemplo substituir la base de datos en memoria por Mysql o Postgres.

Simulator Core: Este módulo es el corazón del simulador. Esencialmente es el core de Activiti modificado para poder simular procesos de negocio.

Report Generator: Este módulo es el responsable de recopilar y retornar los resultados de la ejecución de un escenario de simulación.

BPSim Params: Este módulo es el responsable de interpretar los diferentes tipos de parámetros especificados en el XML de BPSim, ya sean del tipo constante o probabilístico y devolver el valor correspondiente para ser utilizado en el proceso de simulación.

Simulator clock: Este módulo es el encargado de proporcionar el tiempo al motor de simulación logrando que la simulación sea independiente del pasaje real del tiempo.

Simulator events controller: Este módulo tiene la responsabilidad de procesar los eventos internos del simulador que van ocurriendo acorde avanza la simulación además de invocar a los servicios correspondientes para realizar los cálculos asociados a las diferentes perspectivas de BPSim.

History Service: Este servicio facilita el acceso a la información del resultado de la simulación almacenado en la base de datos.

En el siguiente apartado se brinda una descripción mas detallada de los componentes mas relevantes descriptos anteriormente tanto del simulador como del modelador.

5.3 Componentes del simulador

A Continuación se describen desde el punto de vista de diseño los componentes mas relevantes asociados al modelador de procesos de negocio.

5.3.1 PATHFINDER

Este módulo es el encargado de determinar todos los posibles caminos o paths en el diagrama BPMN2 a simular. Una vez determinados los caminos, el simulador se encarga de ejecutarlos de forma secuencial la cantidad de veces que se haya determinado.

Para explicar su funcionamiento supongamos que tenemos el proceso presentado en la Figura 26. Como se mencionó anteriormente, esta parametrizado de tal forma que se definieron 10 instancias a simular, además se definió la probabilidad de ejecución de cada uno de los flujos de secuencia que salen del nodo de decisión, determinando los posibles caminos.

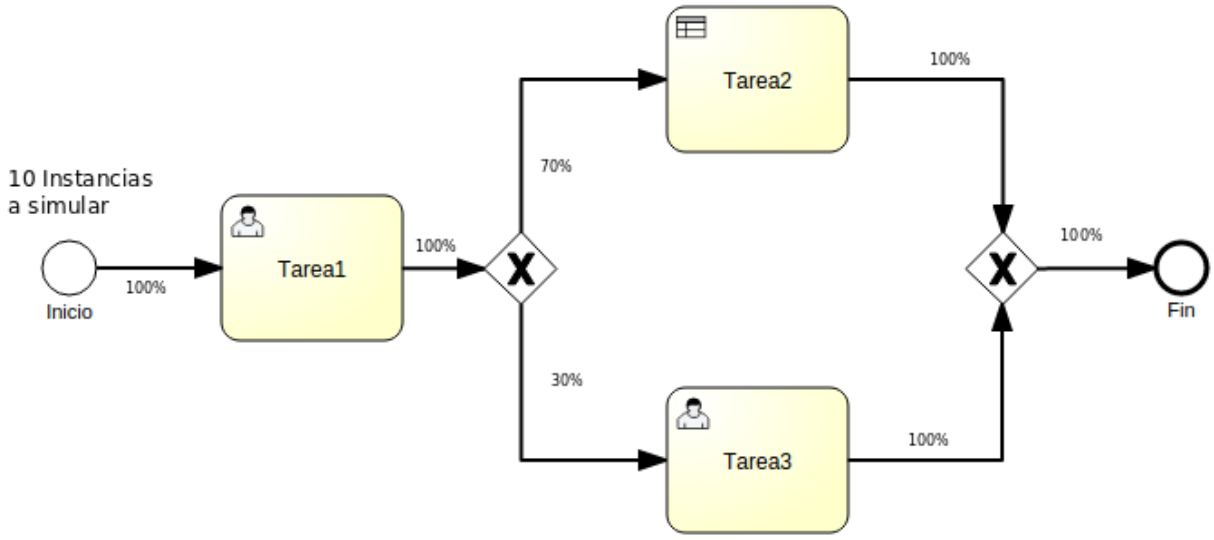


Figura 26: PathFinder - Ejemplo de funcionamiento

Una vez iniciada la simulación el módulo PathFinder recorre recursivamente el metamodelo de BPMN2 y determina todos los posibles caminos de ejecución y la cantidad de veces que se debe simular. En el ejemplo planteado existen únicamente 2 caminos posibles: el que ejecuta las tareas 1 y 2 y el que ejecuta las tareas 1 y 3.

Este primer path o camino deberá ser ejecutando un total de 7 veces. En este caso el porcentaje de probabilidad de ejecución establecido a la salida del nodo de decisión (gateway exclusivo) determina la cantidad de veces que se ejecutará ese camino.

El segundo path o camino deberá ser ejecutado un total de 3 veces. Al igual que en el caso anterior el porcentaje establecido a la salida del gateway determina la cantidad de veces a ejecutar este path.

La complejidad del PathFinder va creciendo a medida que crece la complejidad de los procesos a simular, por ejemplo si se incluyen subprocesos, eventos intermedios, excepciones, bucles, etc.

En la Figura 27 se presenta diagrama de las principales clases involucradas en el módulo PathFinder expresado en UML.

No se muestra el contenido de las clases que aparecen en el diagrama por problemas de espacio. Algunas de estas clases contienen muchos atributos y métodos que hacen que el diagrama se torne considerablemente grande, complicando su legibilidad.

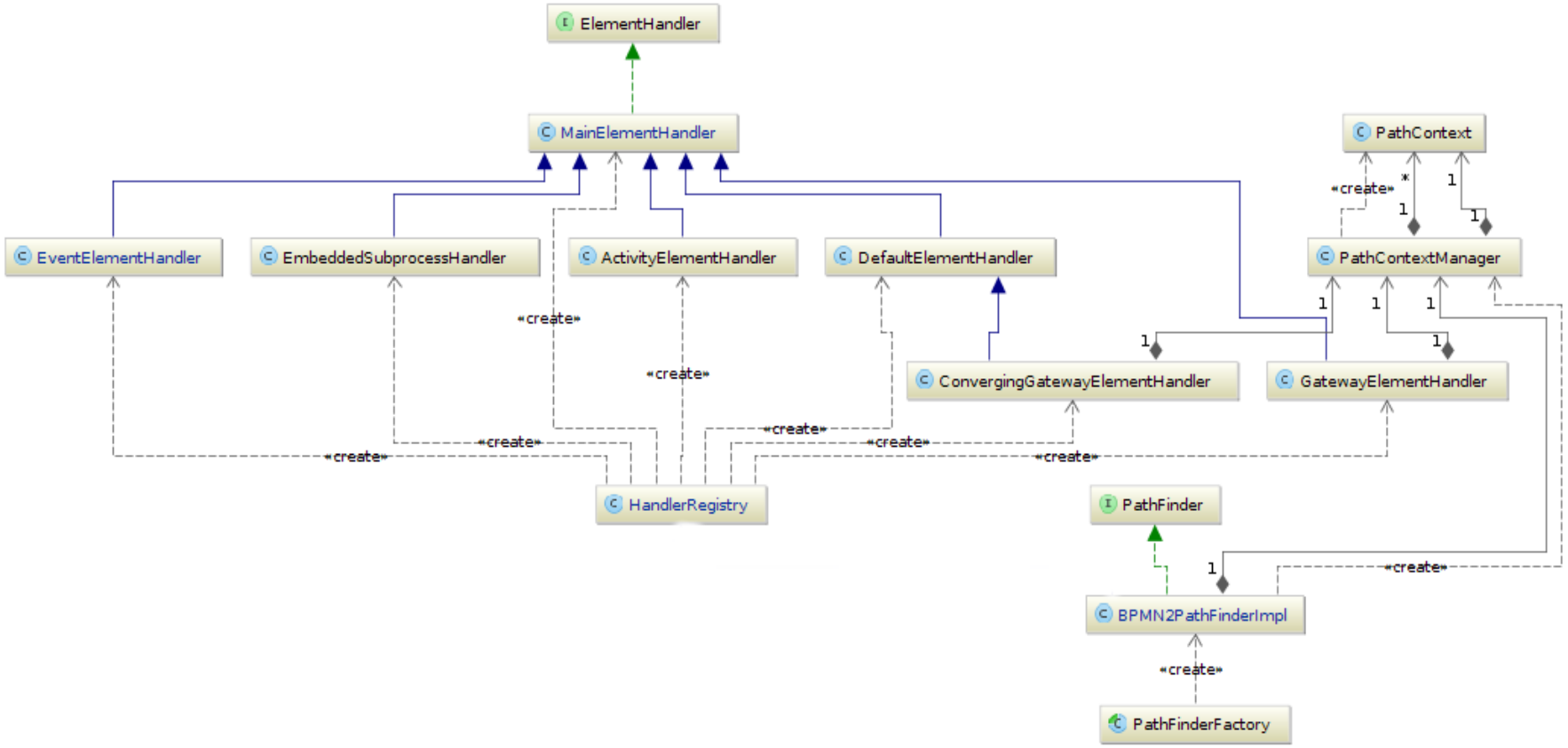


Figura 27: Diagrama de clases UML - PathFinder

A continuación se describen las clases principales del módulo PathFinder.

PathFinderFactory: Esta clase cumple la función de factory y se encarga de crear una instancia del PathFinder de BPMN2.

BPMN2PathFinderImpl: Esta clase es la que “conoce” como implementar la búsqueda de paths en un modelo BPMN2. Se encarga de implementar los métodos de la interfaz PathFinder y realizar la búsqueda recursiva sobre los elementos del metamodelo para construir los diferentes caminos.

HandlerRegistry: Esta clase es la que se encarga de proporcionar el handler adecuado para los diferentes elementos del modelo.
En el diagrama se pueden apreciar 6 handler diferentes:

- **EventElementHandler:** encargado de manejar los elementos correspondiente a eventos.
- **EmbeddedSubprocessHandler:** encargado de manejar los subprocesos.
- **ActivityElementHandler:** encargado de manejar los elementos correspondiente a los diferentes tipos de actividades (Tareas de Usuario, tareas de script, etc.).
- **ConverginGatewayElementHandler:** encargado de manejar los gateways convergentes.
- **GatewayElementHandler:** encargado de manejar los gateways divergentes.
- **DefaultElementHanler:** se utiliza básicamente para seguir navegando recursivamente sobre los elementos del modelo e invocar el handler adecuado llegado el momento.

El resultado de la ejecución del PathFinder es una lista de elementos del tipo PathContext donde cada uno de estos elementos contiene la secuencia de elementos de BPMN2 que conforman cada uno de los caminos de ejecución encontrados.

5.3.2 SIMULATOR CORE

Este módulo es el corazón del simulador y básicamente esta compuesto por el motor de workflow de Activiti mas clases agregadas que por ejemplo se encargan de proveer el punto de acceso al simulador, de la coordinación entre los diferentes módulos, de la generación de los resultados, etc.

En este módulo encontramos el punto de entrada al simulador, la clase SimulatorEngine. Esta clase expone las funcionalidades necesarias para su invocación desde linea de comando o desde la herramienta de modelado. Además es la encargada de orquestar la interacción entre los diferentes módulos necesarios para llegar a ejecutar un proceso en el core y procesar los resultados de dicha simulación.

En la Figura 28 se presenta diagrama UML con las clases mas relevantes del módulo asociadas a la ejecución de una simulación.

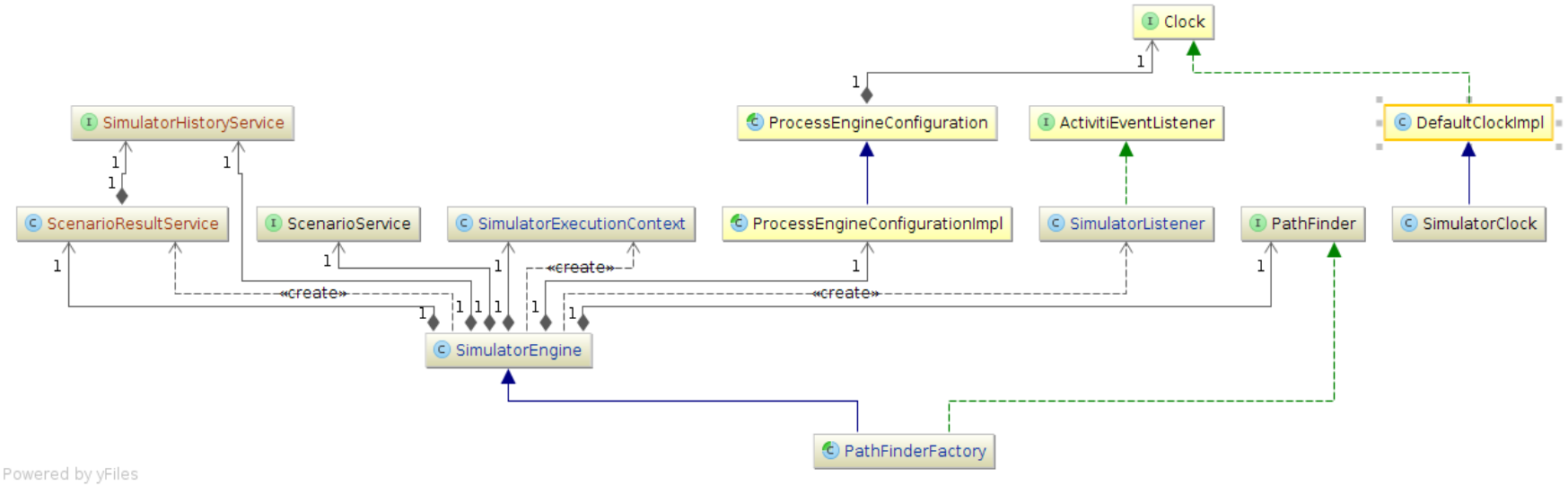


Figura 28: Diagrama de clases UML - SimulatorEngine

La clase SimulatorEngine contiene o referencia de alguna manera a las siguiente módulos/clases:

- **PathFinder:** ya se explico con anterioridad el funcionamiento de este módulo, lo relevante para este caso es que el PathFinder le provee al motor de simulación los diferentes paths de simulación y la cantidad de veces que se debe de simular cada path.
- **SimulatorExecutionContext:** esta clase contiene todos los datos que deben estar disponibles durante toda la ejecución de la simulación para uso de los diferentes módulos, como por ejemplo, identificadores internos de proceso, escenario, unidad de moneda y de tiempo por defecto, fecha y hora de inicio de la simulación, un puntero al estado de inicio, el path en ejecución, etc.. Esta clase vendría a ser para el simulador lo que es el scope de sesión a una aplicación Web.
- **ScenarioService:** Este es el servicio encargado de proveer las funcionalidades necesarias para manejar el escenario a simular. La mas destacada de estas funcionalidades es proveer acceso a los servicios encargados de manejar las diferentes perspectivas de la simulación, ejemplo tiempo, costos, recursos, etc.
- **ScenarioResultService:** Este servicio es el encargado de tomar los datos del escenario simulado y construir la información que se presentará como resultado de la simulación.
- **SimulatorHistoryService:** Este servicio es el que provee acceso a los logs de ejecución del simulador. Recordar que al simular un proceso en la configuración por defecto se generan los logs de ejecución tradicionales de Activiti en una base de datos en memoria.
- **ProcessEngineConfiguration:** Esta clase es la que provee la configuración por defecto del simulador. En esta clase se configura el comportamiento interno del simulador/ motor de workflow. El simulador ya provee una instancia de esta clase llamada **StandAloneInMemSimulatorEngineConfiguration** ya configurada con todo lo necesario para poder simular un proceso.
- **SimulatorListener:** Es la encargada de escuchar determinados eventos que ocurren antes, durante y después de la ejecución de una actividad a simular. En cada uno de estos eventos se realizan determinadas acciones, como cálculos de costos, asignación de recursos, modificaciones al reloj de simulación, etc. Esta clase pertenece al módulo Events Controller, por lo que su funcionamiento se explicara en mayor detalle en el apartado correspondiente.
- **Simulator clock:** Esta clase es una extensión del reloj por defecto que provee Activiti y es la encargada de manejar el tiempo durante una simulación. El tiempo de una simulación es independiente del tiempo real, por ejemplo, en unos segundos que dura una simulación se puede estar representando días, meses o años de ejecución de un proceso.

En la Figura 29 se presenta diagrama de secuencia UML ejemplificando una ejecución de un escenario de simulación.

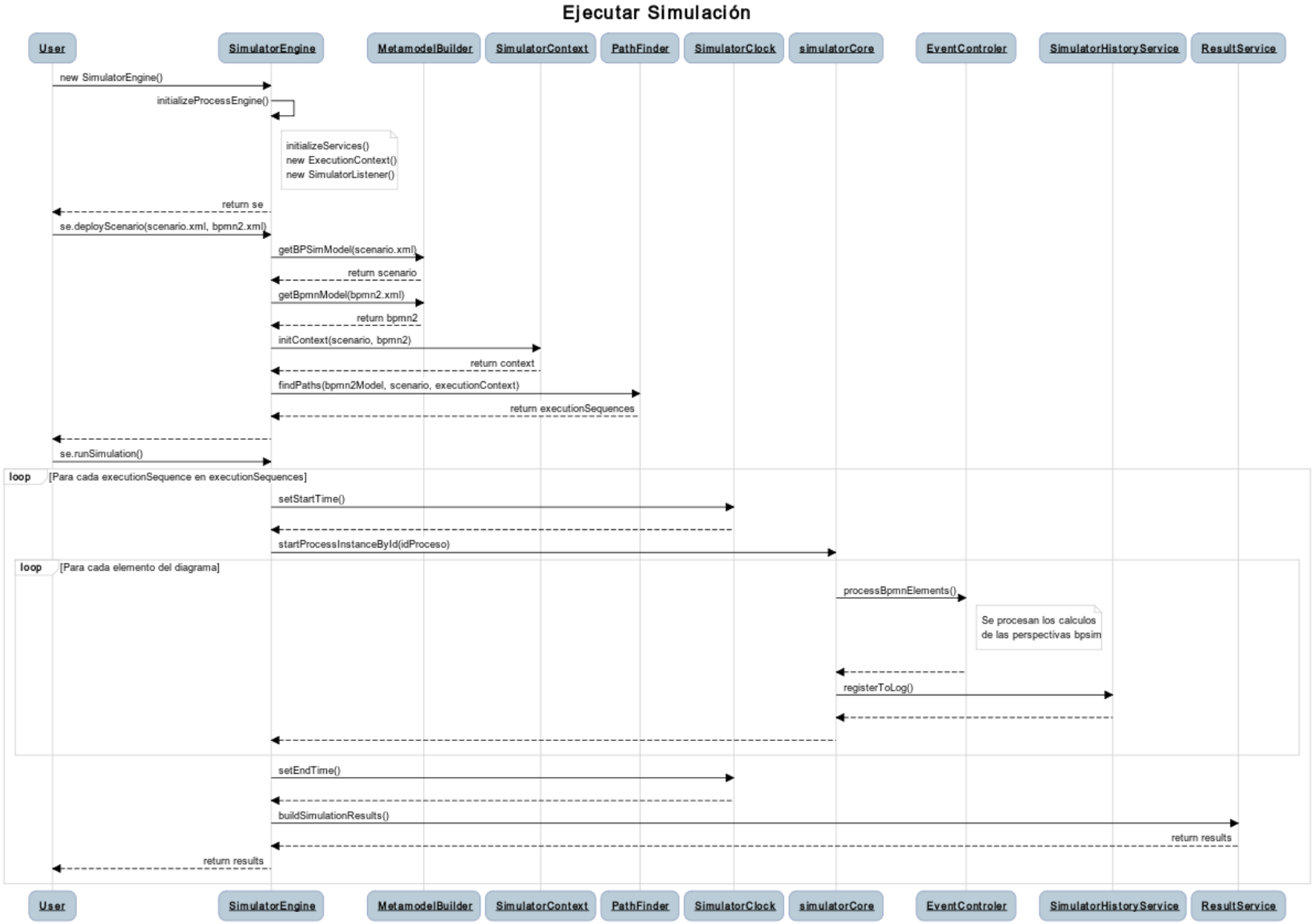


Figura 29: Diagrama UML de secuencia asociado a la ejecución de una simulación

5.3.3 SIMULATOR CLOCK

Como ya se menciono con anterioridad, el reloj de la simulación no mantiene relación alguna con la noción de tiempo real, este reloj se utiliza para simular el pasaje del tiempo durante una simulación. El motor de workflow Activiti ya viene por defecto con su implementación de reloj que permite trabajar de forma independiente del tipo real, aunque por defecto este concuerda con el pasaje tradicional del tiempo. En la Figura 30 podemos apreciar el diagrama UML de clases simplificado con los principales objetos involucrados en la funcionalidad de reloj.

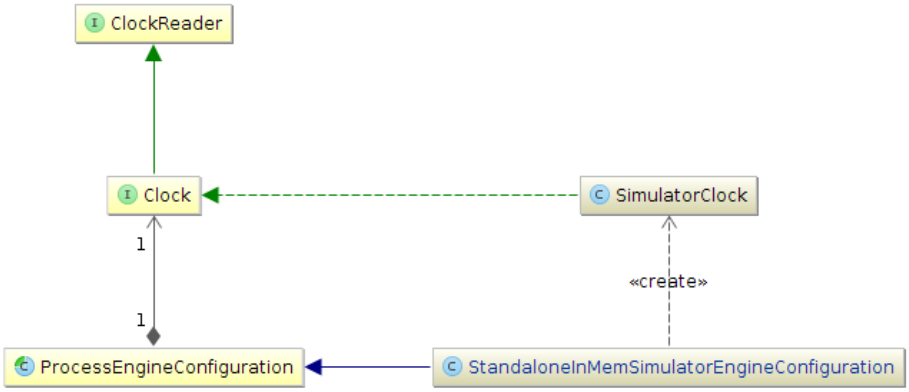


Figura 30: Diagrama de clases UML - Clock

Como se muestra en la Figura 30 podemos apreciar las principales clases involucradas en el reloj, las clases de la izquierda (ClockReader, Clock y ProcessEngineConfiguration) son parte de la implementación por defecto del reloj de Activiti, y las clases de la derecha (SimulatorClock y StandAloneInMemSimulatorEngineConfiguration) son clases provistas por el simulador que modifican y extienden el comportamiento provisto por Activiti. En la Figura 31 podemos apreciar los métodos expuestos por el reloj.

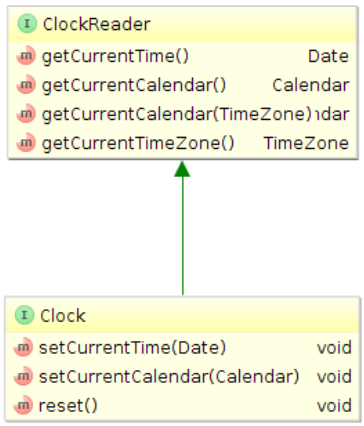


Figura 31: Diagrama detallado - Clock

5.3.4 BPSIM PARAMS

Este módulo tiene la responsabilidad de procesar los parámetros definidos en el escenario de simulación y devolver al motor de simulación el valor correspondiente. Los diferentes atributos pertenecientes a las perspectivas de BPSim soportadas por el simulador están asociadas a un parámetro y este parámetro debe de proporcionar un valor por defecto al simulador.

BPSim define tres tipos de parámetros:

- Constant parameters
- Expression parameters
- Distribution parameters

El segundo tipo de parámetros (Expression parameters) solo se utiliza desde la perspectiva Propiedades, por lo que el simulador no provee soporte para este tipo de parámetros debido a que la implementación de esta perspectiva se planteó como trabajo a futuro. Si se soporta parámetros del primer y del tercer tipo.

Parámetros constantes

Como su nombre lo define, este tipo de parámetro se caracteriza porque su valor es siempre el mismo. BPSim propone los siguientes tipos de parámetros constantes:

- **BooleanParameter:** Representa un valor booleano (True o False).
- **FloatingParameter:** Representa un valor en punto flotante.
- **NumericParameter:** Representa un valor entero.
- **DurationParameter:** Permite representar una duración de tiempo.
- **StringParameter:** Permite representar una cadena de caracteres.
- DateTimeParameter: Permite representar un instante en el tiempo.

En el caso de **DurationParameter** y **DateTimeParameter**, su valor debe estar expresado en el formato ISO 8601 [ISO8601]. En el caso de **NumericParameter** y **FloatingParameter** además de especificar el valor se debe especificar la unidad de tiempo (TimeUnit) o la moneda (CurrencyUnit) de ser necesario. De no especificarse estas propiedades se toma el valor de los parámetros del escenario. El simulador soporta todos los tipos de parámetros propuestos por BPSim en el caso de parámetros constantes.

Parámetros del tipo distribución

Este tipo de parámetro se caracteriza porque su valor puede variar en función del tiempo acorde a un determinado tipo de distribución estadística dada. En la Figura 32 se presentan los tipos de distribuciones soportadas por BPSim.

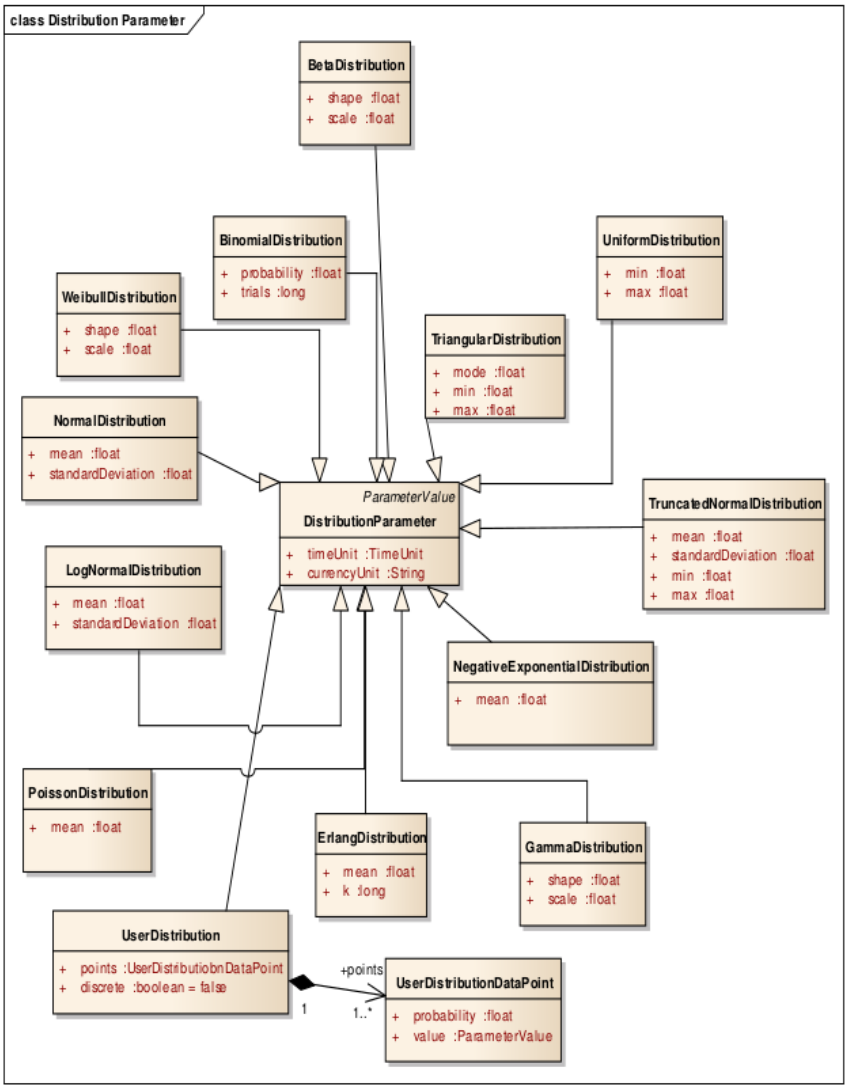


Figura 32: Distribuciones estadísticas soportadas por BPSim

De todas las distribuciones propuestas se decidió implementar cuatro de ellas, Distribución normal (NormalDistribution), Distribución normal trunca (TruncatedNormalDistribution), Distribución poisson (PoissonDistribution), Distribución uniforme (UniformDistribution).

En la Figura 33 se presenta diagrama UML de clases donde se puede apreciar la implementación de los diferentes tipos de parámetros utilizados desde la perspectiva tiempo.

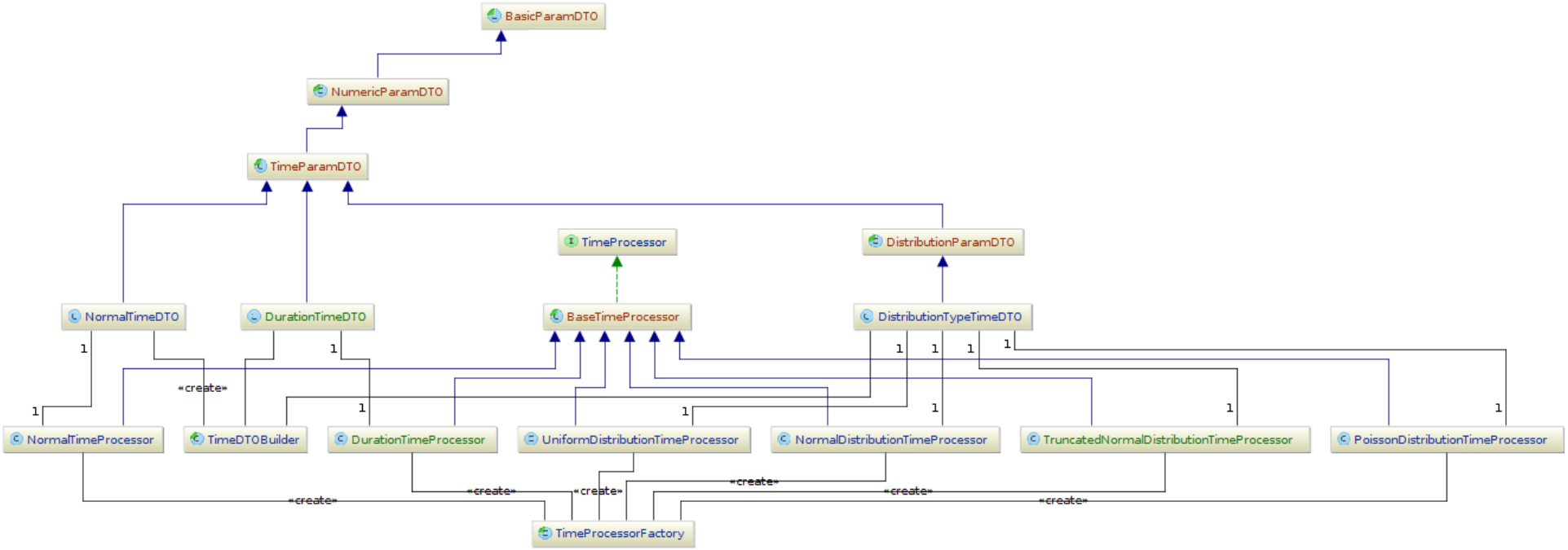


Figura 33: Diagrama de clases UML - Parámetros perspectiva Tiempo

En la Figura 33 se puede apreciar los diferentes tipos de parámetros (Constantes o estadísticos) implementados en el simulador y utilizados en este caso por la perspectiva tiempo. Se utilizó esta perspectiva como ejemplo debido a que es la que presenta mayor variedad de tipos de parámetros de todas las perspectivas implementadas.

La clase **TimeProcessorFactory** es la encargada de devolver la implementación del tipo de parámetro correspondiente según lo que se haya parametrizado en el escenario a simular. Esta clase devuelve un elemento del tipo **TimeProcesor** correspondiente el tipo de parámetro a utilizar, por ejemplo, **NormalDistributionTimeProcessor** si se trata de un parámetro estadístico implementado con un tipo de distribución normal o **DurationTimeProcessor** si se trata de un parámetro constante del tipo duración expresado en ISO 8601. La clase **TimeDTOBuilder** es la encargada de crear la estructura de DTOs [GAMMA94] correspondiente para soportar el tipo de parámetro a utilizar.

5.3.5 EVENTS CONTROLLER

Este módulo tiene la responsabilidad de procesar los eventos internos del simulador que van ocurriendo acorde avanza la simulación además de invocar a los servicios correspondientes para realizar los cálculos asociados a las diferentes perspectivas de BPSim.

Activiti introdujo a partir de la versión 5.15 un mecanismo de eventos que permite notificar al desarrollador cuando ocurren determinados eventos en el motor. Si se tiene interés de realizar determinada acción asociada a un determinado evento es necesario registrar un listener para ese evento en particular.

Los listener se registran utilizando el servicio **RuntimeService**, se puede registrar listener para cada evento en particular o un único listener al cual se le va a notificar cuando ocurra cualquier evento.

Los eventos procesados por la clase **SimulatorListener** corresponden a eventos asociados a diferentes tipos de actividades soportadas por el simulador. En esta clase además se procesan las perspectivas de costo, tiempo y recursos. En la Figura 34 se pueda apreciar en que orden acontecen los eventos asociados a una actividad.

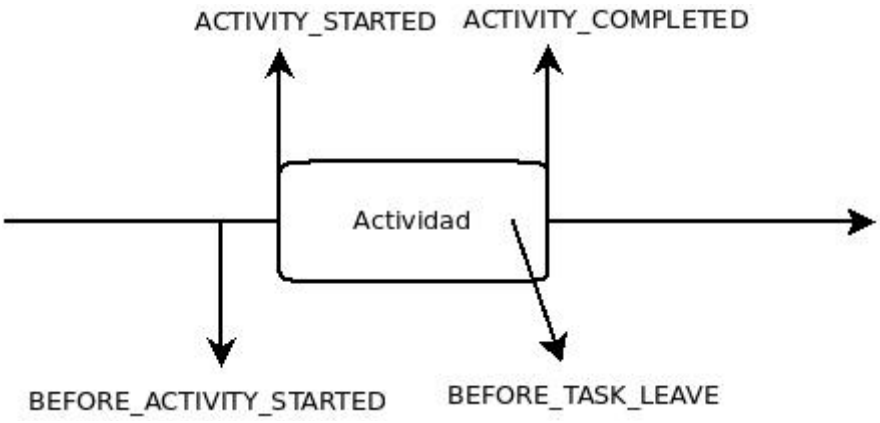


Figura 34: Orden de ejecución de eventos

En la Figura 34 se puede ver la secuencia de eventos en la ejecución de un proceso en el motor.
Para el manejo de eventos Activiti implementa el patrón de diseño “Asynchronous Dispatcher”. Básicamente es una modificación del patrón Observer pero para funcionar en ambientes concurrentes. El momento de disparo de estos eventos hace alusión a su nombre, por ejemplo, el evento ACTIVITY_STARTED se dispara cada vez que el token arriba a la actividad en cuestión, lo mismo pasa con ACTIVITY_COMPLETED, este evento se dispara inmediatamente después que el token deja la actividad.

Fue necesario agregar los eventos marcados como agregados “SI” en la Tabla 19 para poder capturar dos momentos en la ejecución de una actividad que por temas de diseño interno de Activiti no era posible capturarlos en los eventos que ofrece por defecto. El primer evento (BEFORE_ACTIVITY_STARTED) se ejecuta inmediatamente antes que la actividad comience (Es decir, antes que se ejecute ACTIVITY_STARTED), el problema es que cuando se dispara este último previamente se escribe en los logs de ejecución el timestamp en el cual se dio inicio a la actividad, por lo que cuando se disparo el evento y el desarrollador toma el control, estos valores ya están persistidos en la base de datos. Como en determinados casos era necesario realizar cálculos y actualizar el reloj previo a que estos valores sean persistidos, fue necesario incorporar

este evento para poder tomar el control antes de que se escriban los logs de ejecución.

El segundo evento (BEFORE_TASK_LEAVE) fue necesario incorporarlo debido a que para las tareas de usuario el proceso de tomar la tarea y completarla naturalmente se realiza entre los eventos ACTIVITY_STARTED y ACTIVITY_COMPETED, pero por defecto Activiti no provee ningún evento que ocurra entre estos dos.

Todos los eventos de las actividades se procesan de igual forma, a excepción de las tareas de usuario, estas se procesan de forma particular, debido a la necesidad de tener que simular la intervención del usuario.

En la Tabla 19 se presenta la lista de eventos soportados por Activiti y que fueron de utilidad para la simulación. Para la implementación del simulador fue necesario modificar el comportamiento de Activiti y dar soporte también a dos eventos nuevos (Agregados = SI).

Tabla 19: Eventos de Activiti

Evento	Descripción	¿Agregado?
ENGINE_CLOSED	El proceso asociado al contexto de ejecución fue cerrado. Cuando esto sucede no se permite realizar mas llamadas a las APIS de Activiti asociadas a un proceso.	NO
ACTIVITY_STARTED	Este evento se dispara cuando una actividad comenzó a ser ejecutada	NO
ACTIVITY_COMPLETED	Este evento se dispara cuando una actividad es completada con éxito	NO
BEFORE_ACTIVITY_STARTED	Este evento se dispara inmediatamente antes de que una actividad comience a ser ejecutada	SI
BEFORE_TASK_LEAVE	Este evento esta asociado a una actividad del tipo TASK y se dispara antes de que esta sea finalizada.	SI

Descripción de Eventos

BEFORE_ACTIVITY_STARTED: En este evento se realizan los cálculos asociados a la perspectiva tiempo y costo del subtipo de actividad TASK. De la perspectiva tiempo se calculan los siguientes valores asociados a los tiempos presentados anteriormente:

- waitTime
- transferTime
- queueTime
- lagDuration (waitTime + transferTime + queueTime)

Desde la interfaz gráfica del simulador solo es posible parametrizar el parámetro waitTime, por lo que lagDuration va a ser igual a waitTime.

De la perspectiva costo se calcula el costo del parámetro lagDuration, teniendo en cuenta lo que se menciono anteriormente, es lo mismo decir que se calcula el costo del parámetro waitTime.

ACTIVITY_STARTED:
En este evento al igual que el anterior se realizan cálculos asociado a las perspectiva tiempo y

costo del subtipo de actividad TASK. En la perspectiva tiempo se calculan los siguientes valores:

- processingTime
- setupTime
- validationTime
- reworkTime
- duration (processingTime + setupTime + validationTime + reworkTime)

Desde la interfaz gráfica del simulador solo es posible parametrizar el parámetro processingTime, por lo que duration va a ser igual a processingTime.

De la perspectiva costo se calcula el costo del parámetro duration, teniendo en cuenta lo que se menciono anteriormente, es lo mismo decir que se calcula el costo del parámetro processingTime.

BEFORE_TASK_LEAVE: Este evento se dispara únicamente para el subtipo de actividad TASK. Además solo se utiliza para procesar un subtipo especial de TASK, las tareas de usuario. En este evento se realizan cálculos asociados a la perspectiva recursos. De esta perspectiva se calcula el tiempo de espera por los recursos solicitados (resourceWaitTime).

En este evento también se ejecuta la lógica necesaria para tomar y finalizar la tarea de usuario utilizando la API de Activiti. Estas acciones deben realizarse para que se registren los eventos correspondientes en el log de ejecución de Activiti.

Actualización del reloj de simulación

En este módulo también se actualiza el reloj de simulación acorde a los valores de la perspectiva tiempo calculados anteriormente. La actualización del reloj se realiza en eventos diferentes dependiendo si se trata de una tarea de usuario o de cualquier otro tipo de actividad.

En la Figura 35 se muestra como se actualiza el reloj cuando se simula una actividad.

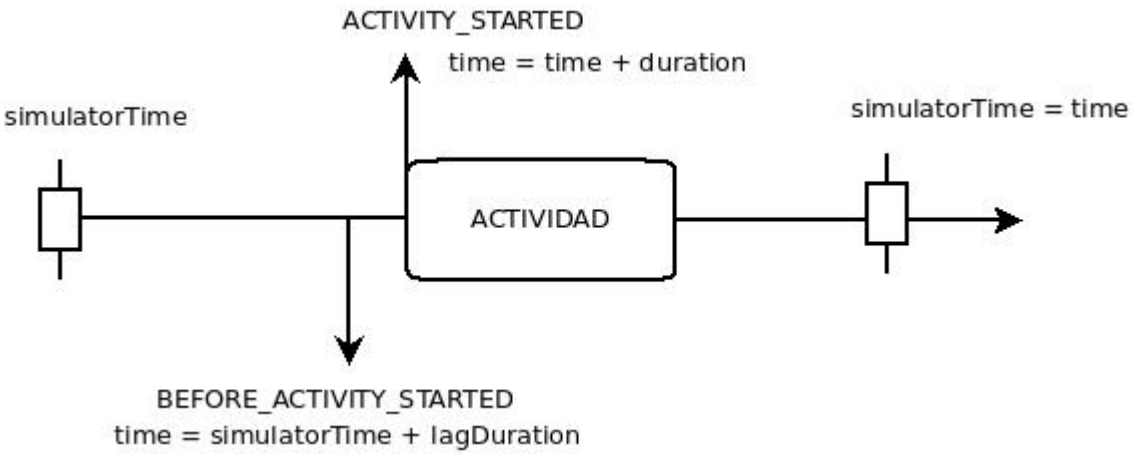


Figura 35: Actualización del reloj de simulación - Actividad

En el evento BEFORE_ACTIVITY_STARTED se calcula el tiempo de espera antes de comenzar la actividad, ese tiempo de espera se almacena en la variable lagDuration, antes de comenzar la

actividad el tiempo de espera es sumado al tiempo del reloj de simulación, logrando así adelantar el reloj de simulación como si realmente hubiera transcurrido ese tiempo.

En el evento ACTIVITY_STARTED se calcula el tiempo de procesamiento de la actividad, ese tiempo se almacena en la variable duration, y luego se suma al tiempo del reloj de simulación. Una vez finalizada la actividad el reloj de simulación se le sumo el tiempo de espera y de procesamiento de la actividad.

En la Figura 36 podemos apreciar como se actualiza el reloj cuando se simula una tarea de usuario.

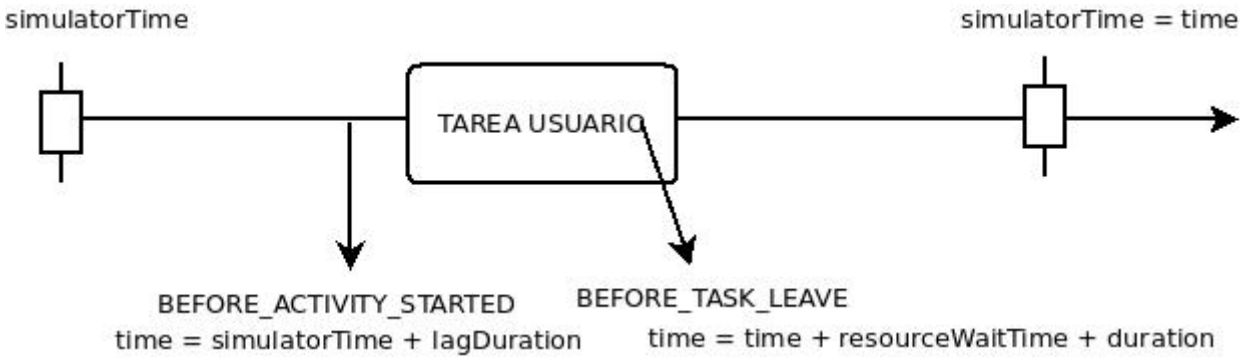


Figura 36: Actualización del reloj de simulación - Tarea

La razón porque en la tareas de usuario se realiza en un evento diferente al resto de las actividades es porque en el evento BEFORE_TASK_LEAVE es donde la tarea es tomada y asignada a un recurso del pool de recursos (En caso de que se le haya asignado recurso en la parametrización del escenario), ahí se calculan el tiempo de procesamiento de la tarea y se le suma el tiempo de espera por el recurso asignado.
En el caso de las tareas de usuario, la única diferencia que presenta en el cálculo de la duración de la tareas es que al tiempo de procesamiento se le suma el tiempo de espera por el recurso asignado.

5.3.6 RESOURCE POOL

Este módulo es el encargado de proveer las funcionalidades para el manejo del pool de recursos.
Los recursos son un concepto importante en todo BPS, pueden ser seres humanos, equipamiento como máquinas o vehículos, o inclusive dinero.

Los recursos generalmente son clasificados de diversas formas (renovable, no renovable, bióticos y abióticos, etc.). Mas allá de la clasificación todos ellos comparten al menos estas dos propiedades, son de disponibilidad limitada y tienen potencial de agotamiento o consumo. Es decir, un recurso para que sea recurso debe de ser finito y se tiene que poder consumir de alguna forma.

Un pool de recursos es un conjunto de recursos (que comparten las mismas características) inicializados previamente que se mantienen disponibles para su uso cuando sea solicitado. Estos recursos generalmente se crean o inicializan en un determinado momento en lugar de ser asignados y destruidos bajo demanda.

Los recursos del pool son consumidos cuando sea necesario y una vez utilizados son devueltos al pool. Cuando un proceso solicita un recurso que no se encuentra disponible, deberá esperar hasta el momento que este sea devuelto, hasta ese momento, el proceso queda bloqueado a la espera de su liberación.

Los pools de recursos pueden ser sofisticados y complejos, por ejemplo permitir definir calendarios de turnos para los recursos, obtener estos recursos de una consulta a un servicio de LDAP, etc.

En la Figura 37 podemos apreciar el diagrama UML de clases del diseño del pool de recursos utilizada en el simulador.

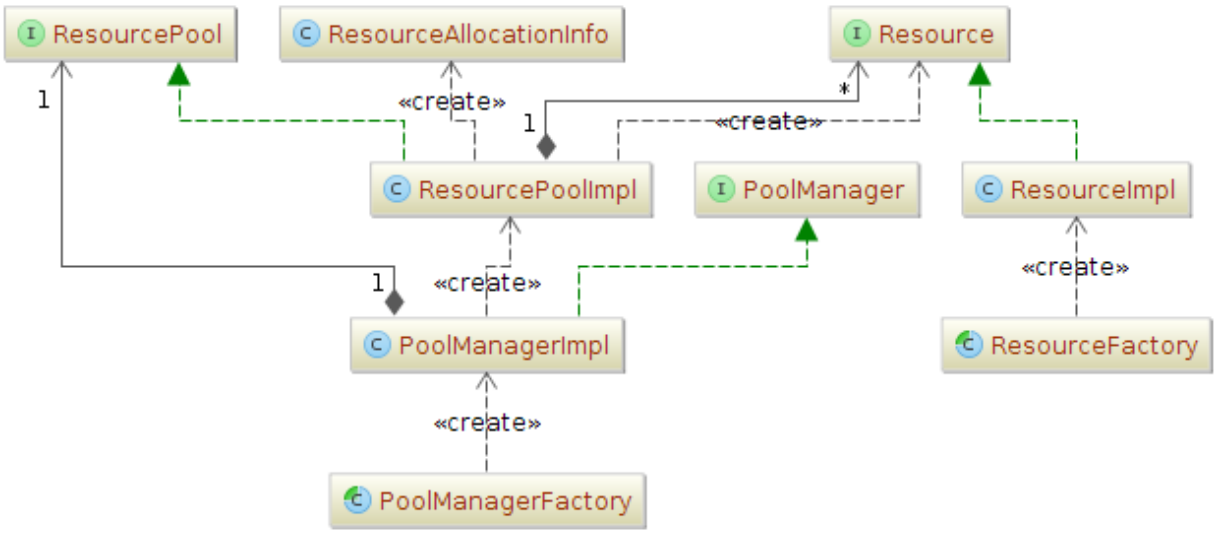


Figura 37: Diagrama UML de clases simplificado - Pool de Recursos

En la Figura 37 podemos apreciar las clases e interfaces involucradas en el manejo de recursos, de las clases que allí se mencionan podemos destacar tres conceptos, PoolManager, ResourcePool y Resource. Resource representa a un recurso en particular, por ejemplo, un enfermero, un desarrollador, ResourcePool representa a un conjunto de esos recursos y PoolManager representa a un conjunto de pools de recursos. En el proceso de simulación cuando se necesita asignar un recurso o varios recursos a una tarea, el simulador le pide al PoolManager el pool correspondiente al tipo de recurso a asignar, luego le pide al pool la cantidad de recursos necesarios para asignarlos a la tarea.

En la Figura 38 podemos apreciar una simplificación del diagrama de clases para una mejor comprensión de los conceptos utilizados.

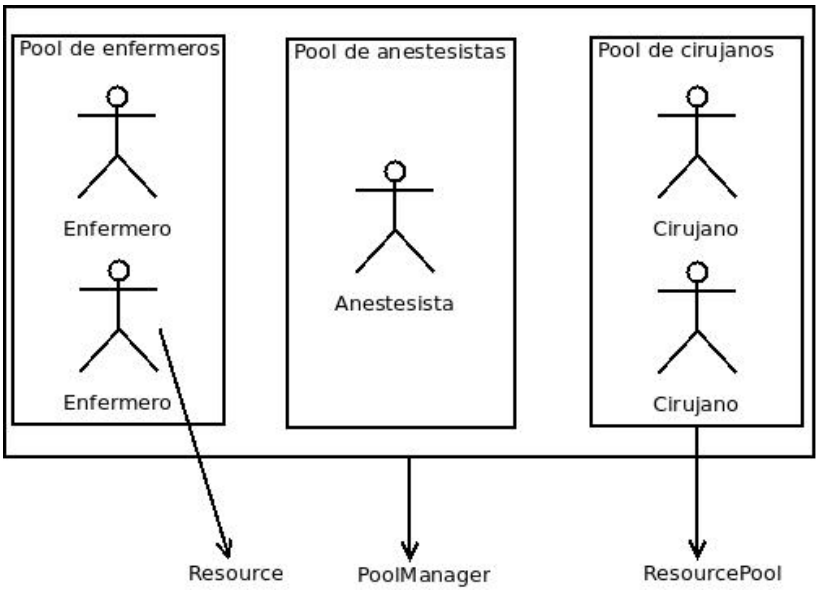


Figura 38: Diagrama ejemplo - Pool de recursos

Como se aprecia en la Figura 38, en el nivel mas básico tenemos los recursos (Resource), estos se agrupan lógicamente en pools de recursos (ResourcePool) y estos a su vez se agrupan todos en un único contenedor llamado PoolManager.

Cuando se ejecuta el simulador uno de los pasos previos antes de comenzar con la simulación es obtener la información de los recursos. Esta información se obtiene de tanto del archivo XML de BPMN2 como del XML de BPSim. Con la información contenida en ambos archivos XML al momento de ejecutar un escenario de simulación se construyen los pools de recursos y el PoolManager de recursos.

En la Figura 39 podemos ver un diagrama UML de secuencia donde se puede apreciar la instancia de creación del PoolManager.

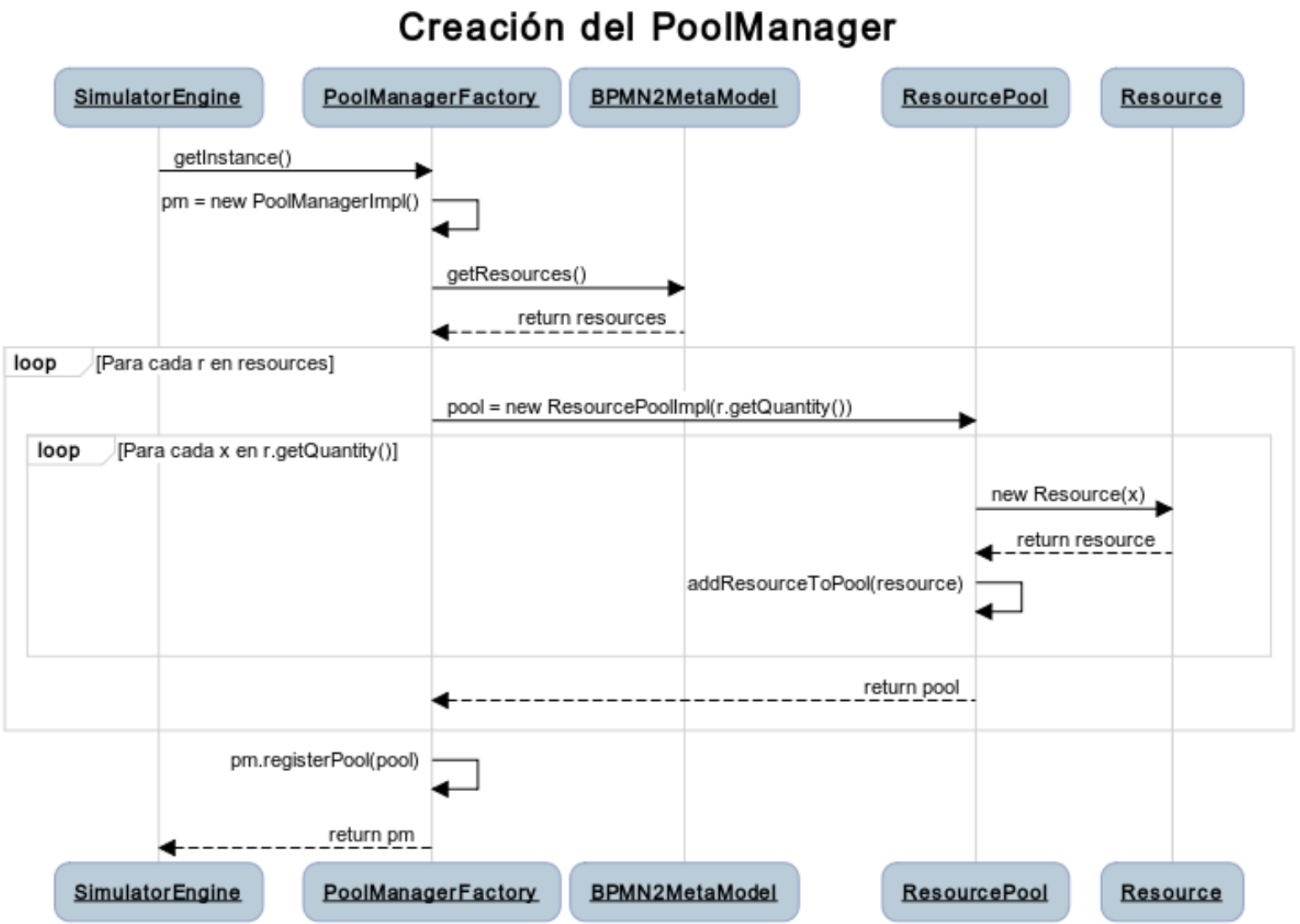


Figura 39: Diagrama de secuencia - Creación del PoolManager

5.3.7 HISTORY SERVICE

La API de Activiti provee por defecto una serie de servicios que exponen las funcionalidades del motor, dentro de esa API se encuentra el servicio History Service. Este servicio es el encargado de suministrar información tanto de lo que ha pasado con ejecuciones anteriores como del estado actual de las instancias del proceso.

En este trabajo de tesis se extendió el comportamiento por defecto de este servicio para brindar acceso al histórico de la simulación de la misma forma que se accede al resto de los datos históricos de Activiti, siempre tratando de respetar el diseño original, de tal forma que se pueda acceder a los datos de la simulación como si fueran parte nativa del motor de workflow.

En la Figura 40 se presenta diagrama UML de clases donde se puede apreciar el punto de extensión al servicio HistoryService.

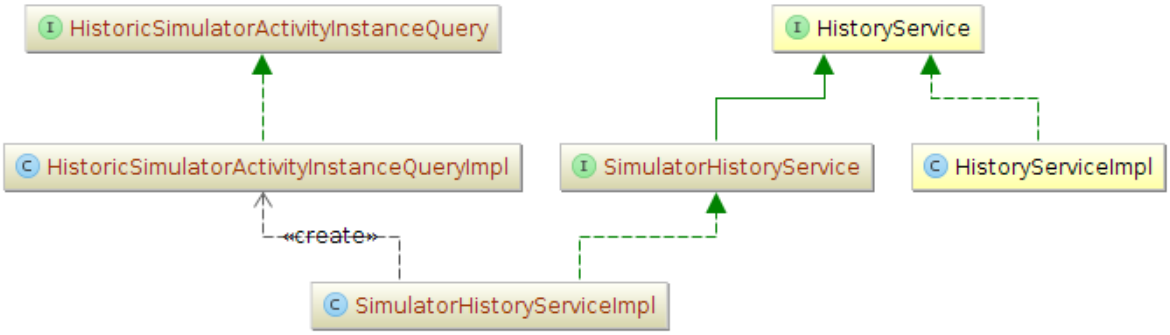


Figura 40: Diagrama de clases UML - HistoryService

La clase **SimulatorHistoryService**, que extiende del servicio **HistoryService** es la encargada de complementar este servicio con las funcionalidades necesarias para el simulador. Esta clase agrega un único método que se puede apreciar en la Figura 41.

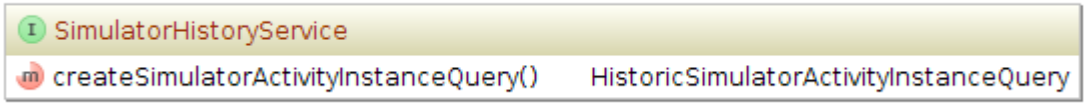


Figura 41: Nuevo método - SimulatorHistoryService

El servicio **SimulatorHistoryService** es utilizado por el módulo Report Generator para presentar los resultados de la simulación al usuario en la forma de reportes predefinidos, como se describe en la próxima sección.

5.3.8 REPORT GENERATOR

Este módulo es el encargado de recopilar la información post ejecución de simulación de una o varias instancias de procesos y retornarla como resultado de la simulación. Hace un uso intensivo

del **SimulatorHistoryService** para obtener la información histórica almacenada tanto de los procesos como de las actividades que lo componen. En la Figura 42 se puede apreciar el diagrama UML de clases que componen este módulo.

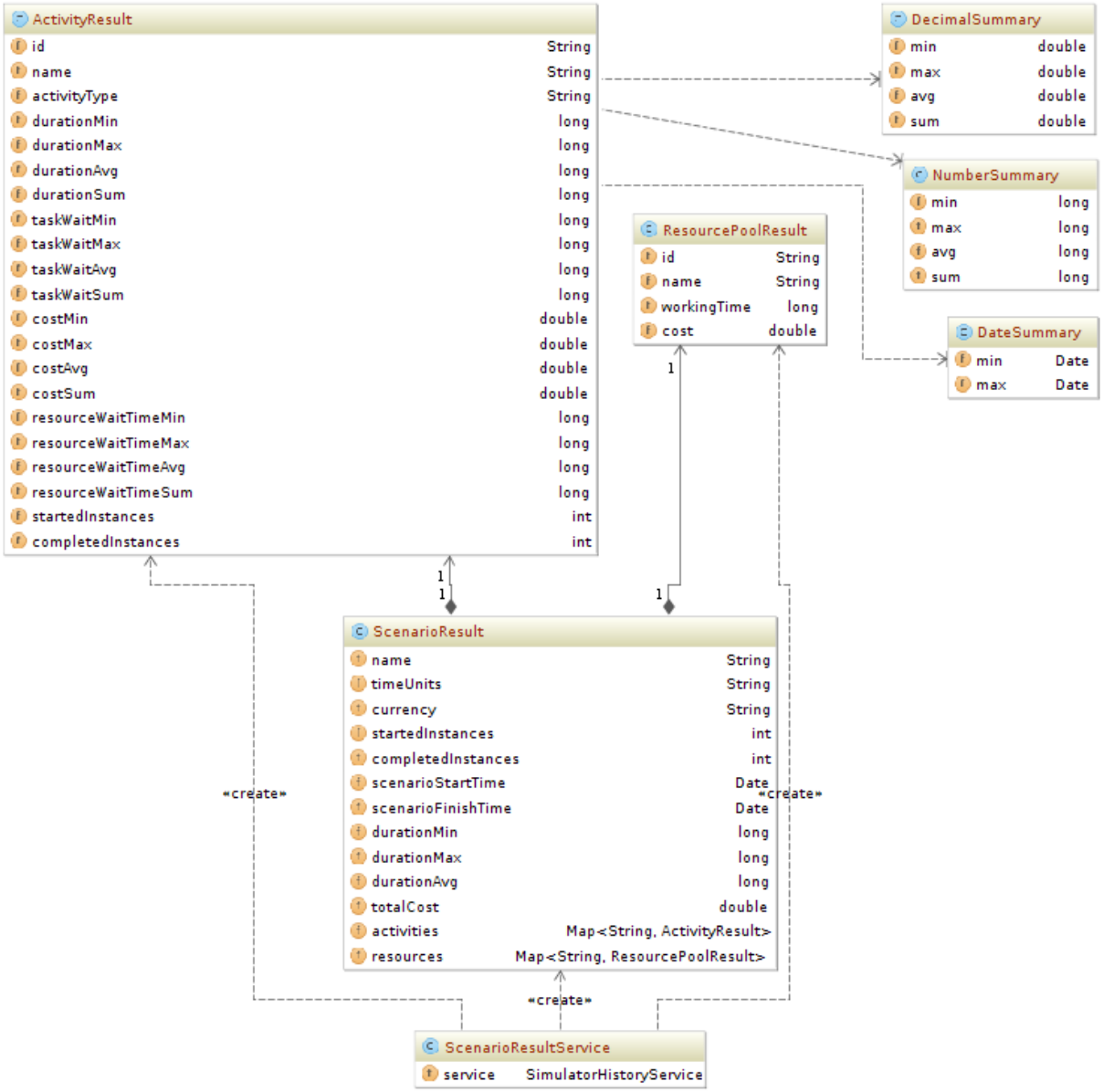


Figura 42: Diagrama de clases UML - Report Generator

El punto de acceso a este módulo es la clase **ScenarioResultService**. Una vez finalizada la simulación del escenario, el motor invoca al servicio ScenarioResultService para que retorne los resultados de la simulación. Como se puede apreciar en la Figura 42, el resultado esta compuesto básicamente por tres clases POJO Java:

- **ScenarioResult:** En esta clase se presentan los datos asociados al escenario de simulación, como ser cantidad de instancias simuladas, instancias completadas, tiempo de duración de la simulación, etc.-
- **ActivityResult:** En esta clase se almacena el resultado de una actividad en particular, y se

pueden encontrar indicadores como tiempos máximos, mínimos y promedios de duración, de espera, etc.

- **ResourcePoolResult:** En esta clase se almacenan los indicadores asociados a la utilización de los diferentes pools de recursos, específicamente el tiempo de utilización.

Las clases **DecimalSummary**, **NumberSummary** y **DataSummary** son clases utilitarias que calculan máximos, mínimos y promedios dependiendo del tipo de dato que corresponda.

5.4 Componentes de la herramienta de modelado

Como se mencionó previamente, para la interfaz gráfica del simulador se extendió el modelador web de Activiti para poder incluir las funcionalidades de simulación propuestas en el trabajo de tesis.

Se extendió el modelador en dos puntos, uno a nivel de servicios REST ofrecidos y otro a nivel de interfaz gráfica. A nivel de servicios se incorporaron las funcionalidades para poder ejecutar una simulación, de transformación de modelos BPSim y BPMN2.0 en formato JSON a XML. Por el lado de la interfaz gráfica fue necesario incorporar las funcionalidades para la visualización de resultados de la simulación (Simulator Result Viewer), y extender las funcionalidades básicas de Signavio Core Components (Oryx) mediante su mecanismo de plugins para incorporar los datos de simulación. A continuación se describen cada uno de estos tres módulos.

5.4.1 SIMULATOR EXTENSIONS PLUGINS

El modelador de Activiti esta construido sobre Signavio Core Components y este a su vez esta construido sobre Oryx [Oryx06], es un proyecto Open Source (Proyecto académico impulsado principalmente por el grupo de investigación de Tecnología de Procesos de Negocio en el Hasso Plattner-Institute) cuyo producto es una herramienta Web que permite modelar procesos BPMN. El grupo de investigación detrás de esta herramienta fue liderado por el profesor Mathias Weske, referente mundial en lo que refiere a proceso de negocio.

Actualmente el soporte y mantenimiento de la herramienta a sido discontinuado, los desarrolladores principales detrás de ella fundaron Signavio en el año 2009. Muchas de las empresas que hoy ofrecen modeladores Web BPMN 2 en realidad ofrecen una integración con una versión OEM de Signavio. Un ejemplo de esto es Activiti. En la Figura 43 se presenta la arquitectura de Oryx.

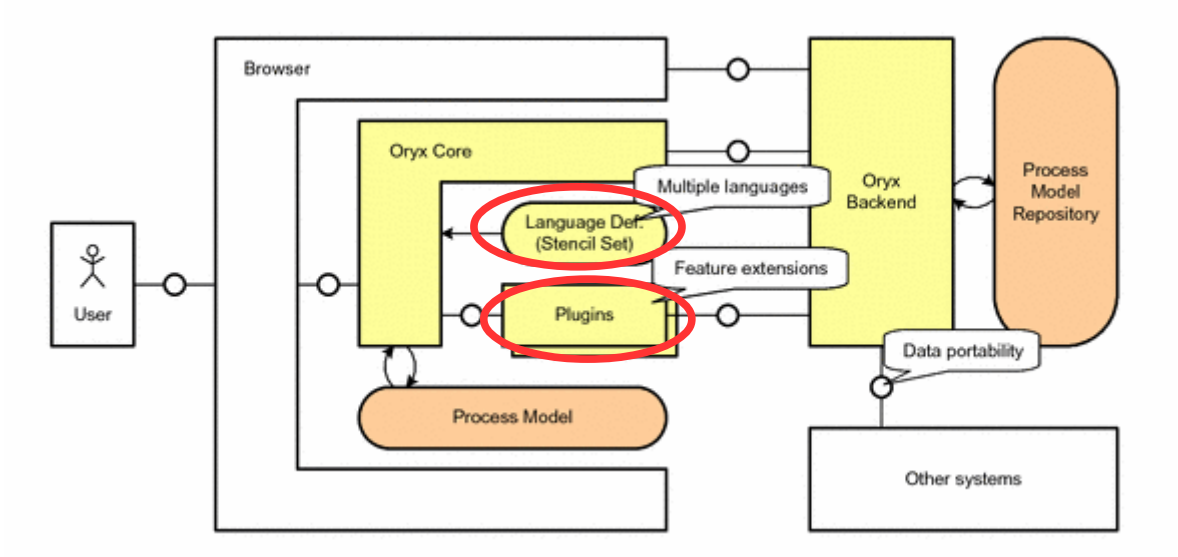


Figura 43: Arquitectura de Oryx de [DUMAS08]

De los módulos presentados en la Figura 43, para poder agregar las funciones de simulación fue necesario modificar o extender 2 módulos:

- Language Definition (Stencil Set)
- Plugins

Stencil Set

Este módulo de Oryx es el que permite definir el lenguaje que se desea modelar. Originalmente esta definición se hacía a través de RDF [RDF15], actualmente esta definición se realiza en lenguaje JSON.

Como se muestra en la Figura 44, algunos elementos de la definición del stencil refieren a los elementos de BPMN 2.0 presentados en el capítulo de Estado del Arte. Específicamente podemos apreciar la definición de la propiedad “Id”, que es la que permite referenciar los elementos BPMN 2.0 desde el escenario de simulación BPSim. Para dar soporte a las propiedades de BPSim fue necesario extender este Stencil Set incorporando las propiedades de simulación soportadas para cada elemento de BPMN 2.0.

```
{
  "title" : "BPMN 2.0",
  "namespace" : "http://b3mn.org/stencilset/bpmn2.0#",
  "description" : "This is the BPMN 2.0 stencil set specification.",
  "propertyPackages" : [ {
    "name" : "elementbase",
    "properties" : [ {
      "id" : "overrideid",
      "type" : "String",
      "title" : "Id",
      "value" : "",
      "description" : "Unique identifier of the element.",
      "popular" : true
    } ]
  }, {
    "name" : "baseattributes",
    "properties" : [ {
      "id" : "name",
      "type" : "String",
      "title" : "Name",
      "value" : "",
      "description" : "The descriptive name of the BPMN element.",
      "popular" : true,
      "refToView" : "text_name"
    }, {
      "id" : "documentation",
      "type" : "Text",
      "title" : "Documentation",
      "value" : "",
      "description" : "The descriptive name of the BPMN element.",
      "popular" : true
    } ]
  }, {
    "name" : "processbase",
    "properties" : [ {
      "id" : "process_id",
      "type" : "String",
      "title" : "Process identifier",
      "value" : "process",
      "description" : "Unique identifier of the process definition.",
      "popular" : true
    }, {
      "id" : "process_author",
      "type" : "String",
      "title" : "Process author",
      "value" : "",
      "description" : "Author of the process definition.",
      "popular" : false
```

Figura 44: Oryx Stencil Set para BPMN 2

Extensión del módulo Plugins

Además de extender el Stencil Set fue necesario modificar código del modelador para adaptarlo por ejemplo a la visualización de las propiedades de simulación en la paleta de propiedades de los elementos del modelo. Las funcionalidades nuevas siempre que fue posible se incorporaron mediante plugins, evitando de esta forma modificar el código original.

5.4.2 REST SERVICES

Activiti por defecto ya implementa servicios REST que brindan funcionalidades del Activiti engine al modelador, como ser:

- Carga de archivos :
  - Carga del archivo Stencil Set a utilizar
  - Cargar la página que contiene el editor (editor.html)

    - Carga del archivo de definición de plugins (plugins.xml)
- Transformación de un modelo BPMN 2.0 a JSON
- Salvar un modelo

Las funcionalidades que están implementadas como servicios REST son las que interactúan con el motor de procesos y/o con archivos de configuración. Fue necesario anexar servicios para poder cubrir los requerimientos planteados por el simulador. A los servicios antes mencionados se anexaron los siguientes:

- Convertidor de formatos
- Simulación de escenario

Convertidor de formatos

Ese servicio se encarga de convertir del modelo interno de Oryx en formato JSON a BPMN 2 y BPSim en formato XML, para poder mostrar en pantalla tanto el código del proceso como del escenario a simular.

Simulación de escenario

Este servicio REST se encarga de ejecutar un escenario parametrizado en la herramienta en el motor de simulación y de devolver los resultados. Los resultados se devuelven en formato JSON y se le pasan al módulo “Simulator Results Viewer” para su presentación.

5.4.3 SIMULATOR RESULTS VIEWER

Este componente se encarga de recibir los resultados de simular un escenario en formato JSON y presentarlo al usuario de una manera amigable, donde se pueden apreciar indicadores relevantes al proceso simulado, como ser, tiempo de ejecución y de espera de cada actividad simulada, tiempo de uso y de espera de los recursos, costos, etc.

6 Implementación del prototipo

En esta sección se presentan los detalles de implementación del prototipo, explicando en profundidad los módulos que lo componen y presentando diferentes diagramas UML que faciliten su comprensión. La implementación del prototipo se puede separar en dos, la implementación del motor de simulación y la herramienta Web de modelado y simulación de procesos.

6.1 Dependencias y versiones

Al comenzar el prototipo se clono el repositorio de Activiti en GitHub [GHUB15] a la última versión estable disponible (5.15.1). Este repositorio contiene 21 proyectos eclipse “mavenizados” relacionados entre si. A continuación, en la Figura 45 se puede apreciar una captura de pantalla del IDE Eclipse donde se pueden apreciar los proyectos.

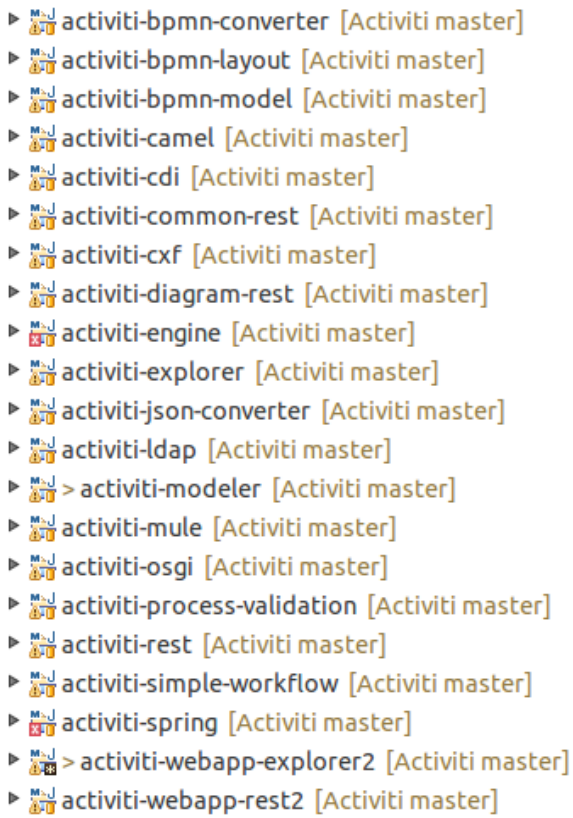


Figura 45: Estructura del proyecto Activiti

El trabajo realizado se dividió en dos “proyectos” diferentes. El primero fue la creación de un proyecto nuevo bajo el nombre de activiti-simulator. Este fue desarrollado íntegramente utilizando el IDE IntelliJ IDEA de JetBrains por un tema de gusto personal. Las dependencias de este proyecto son las siguientes:

- **activiti-engine 5.15.1:** Dependencia asociada al motor de workflow de Activiti.

- **H2 database 1.3.172:** Motor de base de datos en memoria utilizado por defecto para almacenar los logs de ejecución del simulador así como los resultados de la simulación.
- **Metamodelo BPSim 6.0.1 Final:** Metamodelo BPSim creado por la gente de JBPM utilizando Eclipse EMF.
- **Ical4j 1.0.5.2:** Framework que permite trabajar con calendarios en formato ICalendar (RFC 5545,5545 y 6868). BPSim utiliza este estándar para expresar parámetros del tipo calendario.
- **Junit 4.11:** Framework utilizado para construir la suite de test del simulador.
- **Commons-io 1.3.2: Librería** Apache commons-io utilizada para el manejo de archivos, específicamente para leer archivos XML de BPMN2 y BPSim para alimentar al simulador.
- **Commons-math 2.0:** Librería Apache commons-math utilizada para la generación de números pseudoaleatorios y como proveedor de distribuciones estadísticas.

La otra parte del proyecto constó de la modificación del código fuente de los siguientes proyectos de Activiti para incluir las funcionalidades de simulación al modelador de procesos:

- **activiti-webapp-explorer2:** Proyecto que alberga el código fuente del Explorer, dentro de este proyecto se encuentra el código de Activiti Modeler.
- **Activiti-modeler:** Este proyecto contiene el código de los servicios web utilizados por Modeler.

Los 19 restantes proyectos no se modificaron, pero si fue necesario bajar su código fuente por razones de dependencia para poder compilar y generar el WAR correspondiente a Activiti Explorer. Solamente se modificó el archivo POM (Project Object Model) del proyecto ROOT para incluir como dependencia al JAR del proyecto activiti-simulator.

6.2 Componentes del motor de simulación

A continuación en la Figura 46 se muestran los componentes mas relevantes del motor así como su distribución para la ejecución del mismo con Activiti BPMS, también se pueden apreciar los puntos de extensión a los componentes originales de Activiti.

Como se puede apreciar se distinguen dos tipos de componentes, los de color blanco son los componentes originales provistos por Activiti, los componentes en color gris son los componentes que agrega el simulador, ya sea que extienden a un componente original o que incorporan nuevas funcionalidades.

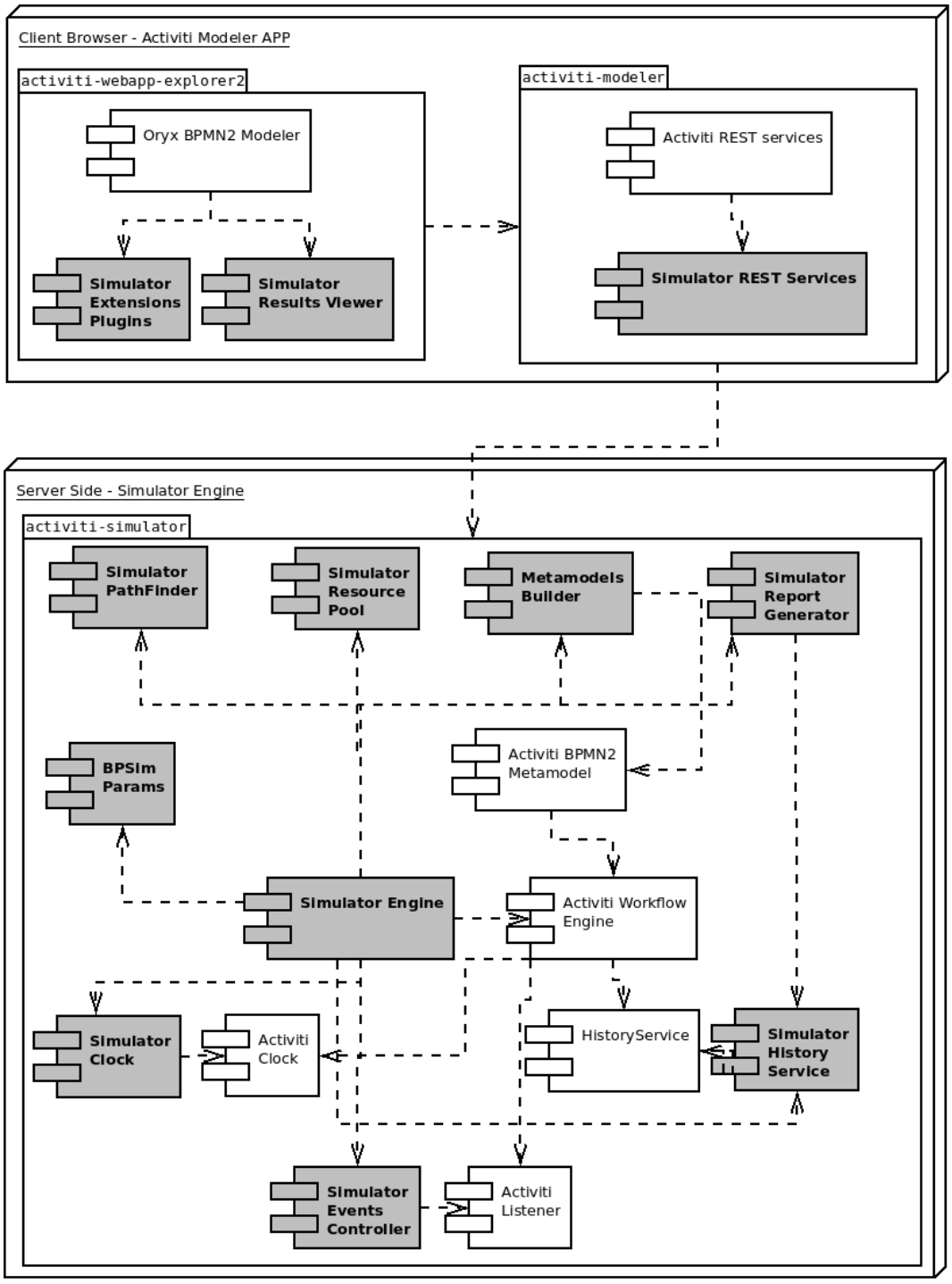


Figura 46: Diagrama UML de componentes del Simulador

A continuación se brindan detalles de los aspectos relevantes a la implementación de algunos de los componentes provistos por el simulador (Los componentes de color gris).

6.2.1 METAMODELS BUILDER

Este no es en realidad un módulo, sino que son funcionalidades básicas implementadas en clases utilitarias para el manejo de los metamodelos de BPSim y de BPMN2. Activiti ya provee una implementación propia del metamodelo de BPMN2 que fue reutilizada y extendida por el simulador.
Para la implementación del metamodelo de BPSim se reutilizó una implementación realizada por JBoss y utilizada en el simulador de JBPM 5. Esta implementación esta realizada con Eclipse EMF y se conoce con el nombre de “jbpm-bpmn2-emfextmodel”.

El simulador puede ser ejecutado en 2 contextos diferentes, de forma standalone (aplicación Java que puede ser ejecutada por linea de comando) o desde la herramienta de modelado.

Cuando se ejecuta una simulación desde la herramienta de modelado se invoca al simulador y este recibe como parámetros ambos metamodelos ya instanciados, por lo que previamente tuvieron que ser convertidos desde el formato que se utiliza como base para los mismos (JavaScript Object Notation, JSON [JSON15]). Esta conversión se realiza en el módulo que en la imagen anterior esta representado con el nombre “JSON to XML”.

JSON es el formato por defecto que utiliza ORYX y por ende la herramienta de modelado de Activiti para representar la metadata del diagrama modelado. Partiendo de este punto, todas las propiedades de simulación que se agregaron a la herramienta se implementaron respetando el diseño ya existente. EL resultado obtenido al modelar un proceso en la herramienta es un metamodelo interno representado en JSON, luego ese metamodelo es procesado y se convierte a las respectivas representaciones en Java tanto para BPMN2 como para BPSim.

Las funcionalidades de conversión de JSON a BPMN2 ya estaban disponibles en las clases utilitarias de Activiti, lo que si fue necesario fue la implementación de las funcionalidades de conversión de JSON al metamodelo de BPSim. Para realizar esta conversión se utilizó la librería Jackson [JACK15] en su versión 1.9.9.

Cuando se ejecuta el motor de simulación de forma standalone este puede recibir como parámetros el PATH a los archivos XML donde se encuentra el proceso a simular en BPMN2 y el escenario de simulación. En este caso los archivos XML son validados y convertidos a sus metamodelos correspondientes en Java.

6.2.2 SIMULATOR ENGINE

Al implementar el punto de acceso al simulador se tuvo como principal objetivo la simplicidad, por lo que ejecutar una simulación se transforma en una tarea sencilla.

A continuación, en la Figura 47 se presenta un extracto de código Java donde se puede apreciar una invocación al motor de simulación.

```java
/**
 * Created by Juan Angel Larrayoz on 12/06/15.
 */
public class PruebaSimulador {

    public static void main(String[] args) {
        SimulatorEngine se = new SimulatorEngine();
        se.deployScenario("default", "bpSim/procesoComplejoCostScenario1.xml",
                "bpSim/procesoComplejo.bpmn20.xml");
        ScenarioResult result = se.runSimulation();
        se.close();
    }
}
```

Figura 47: Ejemplo código de invocación al simulador

En la Figura 47 se puede apreciar el código para la invocación del simulador desde una aplicación java standalone (por línea de comandos). En la primera instrucción que encontramos dentro del método main se puede apreciar una la creación de una instancia del simulador:

SimulatorEngine se = new SimulatorEngine();

Cuando se crea una nueva instancia del simulador internamente acontece lo siguiente:

- Se crea una instancia del motor de simulación en base a la configuración descripta en el “archivo activitiSimulation.cfg.xml”.
- Se crea una instancia de alguno de los servicios básicos como ser el HistoryService, ResultService, etc.
- Se crea el contexto de simulación y se registra los listeners correspondiente al motor de simulación.

La segunda instrucción de código es la responsable de cargar en el contexto correspondiente el escenario de BPSim a simular y el proceso de negocio correspondiente.

se.DeployScenario(scenarioId, bpSimScenarioPath, bpDiagramPath);

El método deployScenario recibe tres parámetros para su ejecución:

- **scenarioId:** Identificador del escenario a simular. Se debe especificar el id debido a que un archivo de BPSim (**bpSimScenarioPath**) podría contener mas de un escenario a simular.
- **BpSimScenarioPath:** Path al archivo XML que contiene el escenario de BPSim a simular.
- **BpDiagramPath:** Path al archivo XML que contiene el proceso de negocio en BPMN 2.0 a simular. El archivo debe de tener extensión “.bpmn20.xml” para que se pueda interpretar, esta es una restricción de Activiti que se traslado al simulador.

La tercera instrucción de código invoca al método “runSimulation” que es el responsable de finalmente ejecutar la simulación y devolver el resultado del escenario simulado. El resultado de la simulación es devuelto en un objeto del tipo “ScenarioResult”.

Finalmente en la cuarta instrucción se finaliza la simulación. Esta instrucción se encarga de quitar de memoria todas las intancias de servicios, contextos, etc. creadas en la primera instrucción.

6.2.3 SIMULATOR CLOCK

El tiempo se manipula básicamente de dos lugares diferentes, al comenzar la simulación y acorde

van sucediendo los eventos a simular. En el primer caso se hace desde la clase **SimulatorEngine**, cuando se ejecuta el método “runSimulation**”** como se puede apreciar en la Figura 48.

```
public ScenarioResult runSimulation(){
    Set<String> simulatedProcess = new HashSet<~>();

    //Obtiene la fecha de inicio del escenario de simulación y la almacena en el contexto
    Date simDate = executionContext.getStartTime();
    if (simDate == null){
        Date aux = new Date();
        long timeAux = aux.getTime() % 1000;
        Calendar c   = Calendar.getInstance();
        c.setTimeInMillis(aux.getTime() - timeAux);
        simDate = c.getTime();
        executionContext.setStartTime(simDate);
    }
```

Figura 48: Modificación del reloj al ejecutar simulación

Una vez iniciada la simulación, luego de haber cargado el contexto con los datos necesarios se modifica el tiempo de inicio de la simulación. Este tiempo puede estar dado por el parámetro **startTime** de BPSim o si este no fue parametrizado, se configura el reloj de simulación con la fecha y hora del momento de la ejecución. A cada nueva instancia de simulación que se va a ejecutar, se le suma al **simDate** (Valor actual del reloj) el tiempo establecido en el parámetro **InterTriggerTimer** del evento start del proceso como se puede apreciar en la Figura 49.

```
ControlParamService cps =  ControlParamServiceFactory.getInstance(executionContext);
interval = cps.getInterTriggerTimer(executionContext.getStartEventId());

if (interval > 0 ) {
    processEngineConfiguration.getClock().setCurrentTime(SimulatorDateUtils.addMilis(simDate, interval));
    simDate = processEngineConfiguration.getClock().getCurrentTime();
}

executionContext.finishSimulationInstance(false);
//printResults();
simulatedProcess.add(executionContext.getProcessInstanceId());
executionContext.setProcessInstanceId(null);
```

Figura 49: Incremento del intervalo de tiempo al simular una instancia

En el primer recuadro rojo se puede apreciar como se obtiene el valor del parámetro **InterTriggerTimer.** En el segundo recuadro rojo podemos ver como se le suma al tiempo actual de simulación (simDate) el valor del intervalo de tiempo.

El otro punto de manipulación del reloj es en la clase **SimulatorListener,** donde se modifica acorde a la perspectiva tiempo de los elementos a simular. Cabe destacar que la perspectiva tiempo solo afecta a los elementos de BPMN2 del tipo Actividad, y en el caso del simulador específicamente a los del subtipo tarea y subproceso.

6.2.4 BPSIM PARAMS

Para la implementación en el simulador de las diferentes distribuciones estadísticas se utilizó la librería “Apache Commons Math” [ACM15]. Esta librería provee soporte matemático y estadístico

a la mayoría de los problemas comunes que se pueden encontrar a la hora de desarrollar un programa en el lenguaje Java. También se utilizo esta librería como generador de números pseudoaleatorios.

En la Figura 50 se puede apreciar el código Java de la clase que implementa el tipo de distribución normal.

```
public class NormalDistributionTimeProcessor extends BaseTimeProcessor {
    private static RandomData generator = new RandomDataImpl();
    private DistributionTypeTimeDTO dto;

    public NormalDistributionTimeProcessor(DistributionTypeTimeDTO dto, String baseTimeUnit){
        super(dto.getCalendar(), baseTimeUnit);
        this.dto = dto;
    }

    @Override
    public long returnTime() {
        TimeUnit localTimeUnit;
        long mean;
        long deviation;

        if (dto.getTimeUnit() != null){
            localTimeUnit = SimulatorDateUtils.getTimeUnitFromString(dto.getTimeUnit());
        }
        else{
            localTimeUnit = SimulatorDateUtils.getTimeUnitFromString(baseTimeUnit);
        }

        mean = timeUnit.convert((long) dto.getMean(), localTimeUnit);
        deviation = timeUnit.convert((long) dto.getStandardDeviation(), localTimeUnit);

        if (deviation > 0){
            return  (long) generator.nextGaussian(mean, deviation);
        }
        else{
            return 0;
        }
    }
```

Figura 50: Implementación de parámetro del tipo distribución normal

En el primer recuadro rojo se puede apreciar la creación de generador de datos estadísticos (**RandomData**). Esta clase contiene por defecto la implementación de un generador de números pseudoaleatorios (clase **RandomGenerator**) y de los tipos de distribuciones utilizadas en el simulador. En el segundo recuadro rojo se puede ver la invocación al tipo de distribución normal con los parámetros obtenidos del escenario a simular (media y desviación).

6.2.5 EVENTS CONTROLLER

Como se menciono anteriormente en la misma sección pero del capítulo de diseño fue necesario incorporar a los eventos que Activiti ofrecía por defecto dos nuevos eventos, **BEFORE_ACTIVITY_STARTED**, **BEFORE_TASK_LEAVE.** Para disparar cada uno de estos eventos en el momento oportuno fue necesario identificar y modificar en el código de Activiti el punto de extensión correspondiente.

En la Figura 51 se puede apreciar el punto de extensión donde se lanza el evento **BEFORE_ACTIVITY_STARTED.**

```
public class SimulatorActivityInstanceStartHandler extends ActivityInstanceStartHandler {

    public void notify(DelegateExecution execution) {

        try {

            if (Context.getProcessEngineConfiguration() != null &&
                    Context.getProcessEngineConfiguration().getEventDispatcher().isEnabled()) {

                ExecutionEntity entity = (ExecutionEntity) execution;

                CustomActivityEvent newEvent = new CustomActivityEvent(ActivitiEventType.CUSTOM,
                        SimulatorCustomEventType.BEFORE_ACTIVITY_STARTED);

                newEvent.setActivityId(entity.getActivityId());
                newEvent.setExecutionId(entity.getId());
                newEvent.setProcessDefinitionId(entity.getProcessDefinitionId());
                newEvent.setProcessInstanceId(entity.getProcessInstanceId());

                Context.getProcessEngineConfiguration().getEventDispatcher().dispatchEvent(newEvent);

            }

        } catch (RuntimeException e) {
            throw e;
        }
        Context.getCommandContext().getHistoryManager().recordActivityStart((ExecutionEntity) execution);
    }
```

Figura 51: Implementación del evento BEFORE_ACTIVITY_STARTED

Como se aprecia en la Figura 51, fue necesario implementar una clase nueva, **SimulatorActivityInstancesStartHandler** que hereda de la clase **ActivityInstanceStartHandler**. En esta última se identificó el momento exacto antes de grabar en los logs de ejecución el inicio de actividad (Código enmarcado en rojo) y se inserto el código correspondiente al lanzamiento del evento.

En la Figura 52 se puede apreciar el punto de extensión donde se lanza el evento **BEFORE_TASK_LEAVE.**

```
public UserTaskActivityBehavior createUserTaskActivityBehavior(UserTask userTask, TaskDefinition taskDefinition) {
    return new UserTaskActivityBehavior(taskDefinition){

        public void execute(ActivityExecution execution) throws Exception {

            ...

            //handleAssignments(task, execution);

            // All properties set, now firing 'create' event
            task.fireEvent(TaskListener.EVENTNAME_CREATE);


            //Fire CUSTOM event before task auto leve
            CustomActivityEvent newEvent = new CustomActivityEvent(ActivitiEventType.CUSTOM,
                    SimulatorCustomEventType.BEFORE_TASK_LEAVE);
            newEvent.setActivityId(task.getId());
            newEvent.setExecutionId(task.getExecutionId());
            newEvent.setProcessDefinitionId(task.getProcessDefinitionId());
            newEvent.setProcessInstanceId(task.getProcessInstanceId());
            newEvent.setTaskId(taskDefinition.getKey());
            Context.getProcessEngineConfiguration().getEventDispatcher().dispatchEvent(newEvent);
        }
```

Figura 52: Implementación del evento BEFORE_TASK_LEAVE

En el recuadro rojo se puede apreciar el código correspondiente al lanzamiento del evento.

6.2.6 RESOURCE POOL

Para poder implementar la perspectiva de recursos fue necesario realizar modificaciones al metamodelo de BPMN 2.0 propuesto por Activiti. Para la definición de recursos BPSim plantea la utilización del tag BPMN2 **<resource>**, el cual no esta soportado por Activiti.

La definición del pool de recursos se hace en base a información que se encuentra en el archivo XML de BPMN2 y del archivo XML que contiene al escenario BPSim a simular. En el archivo XML de BPMN2 se encuentra la definición de los diferentes pools de recursos (mediante el uso del tag resource) así como la asignación especifica de recursos a cada una de las tareas de usuario del proceso de negocio. Para la asignación de recursos a una tarea se respetó la implementación propuesta por Activiti. La Figura 53 presenta un extracto de XML de un archivo BPMN2 donde se puede apreciar la parametrización de 3 pools de recursos, resourceCirujanos, Anestesistas y Enfermeros.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<definitions xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL" xmlns:acti
             xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI"
             xmlns:omgdc="http://www.omg.org/spec/DD/20100524/DC" xmlns:omg
             xmlns:tns="http://www.activiti.org/test" xmlns:xsd="http://www
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" expressi
             id="m1400340027818" name="" targetNamespace="http://www.activi
             typeLanguage="http://www.w3.org/2001/XMLSchema">

    <resource name="resourceCirujanos" id="resourceCirujanos"/>
    <resource name="Anestesistas" id="Anestesistas"/>
    <resource name="Enfermeros" id="Enfermeros"/>

    <process id="pruebaBPSim"  name="pruebaBPSim" isExecutable="true">
        <userTask id="tarea"  name="tarea" activiti:candidateUsers="desarro
            <incoming>_C75217CC-1054-4CB0-B54F-80C27F163796</incoming>
            <outgoing>_654F25A8-FF0E-453A-9D51-5AD87B5EC910</outgoing>
```

Figura 53: Extracto XML - Parametrización de pools

En la Figura 54 se puede apreciar la información complementaria para la definición de pools de recursos en el archivo XML de BPSim. Se puede apreciar la cantidad de recursos asignados a cada uno de los pools definidos en la Figura 53.

```
<?xml version="1.0" encoding="UTF-8"?>
<bpsim:BPSimData xmlns:bpsim="http://www.bpsim.org/schemas/1.0">
  <bpsim:Scenario id="58ead37d-1fe5-4c58-b9de-b73854352158">
    <bpsim:ScenarioParameters baseCurrencyUnit="US$" baseTimeUnit="min">
      <bpsim:Start>
        <bpsim:DateTimeParameter value="2014-01-01T00:00:00"/>
      </bpsim:Start>
      <bpsim:Duration>
        <bpsim:DurationParameter value="PT10H"/>
      </bpsim:Duration>
    </bpsim:ScenarioParameters>
    <bpsim:ElementParameters elementRef="resourceCirujanos">
      <bpsim:ResourceParameters>
        <bpsim:Quantity>
          <bpsim:NumericParameter value="2"/>
        </bpsim:Quantity>
      </bpsim:ResourceParameters>
    </bpsim:ElementParameters>
    <bpsim:ElementParameters elementRef="Anestesistas">
      <bpsim:ResourceParameters>
        <bpsim:Quantity>
          <bpsim:NumericParameter value="1"/>
        </bpsim:Quantity>
      </bpsim:ResourceParameters>
    </bpsim:ElementParameters>
    <bpsim:ElementParameters elementRef="Enfermeros">
      <bpsim:ResourceParameters>
        <bpsim:Quantity>
          <bpsim:NumericParameter value="5"/>
        </bpsim:Quantity>
      </bpsim:ResourceParameters>
```

Figura 54: BPSim - Definición de pools de recursos

Con la información contenida en ambos archivos XML al momento de ejecutar un escenario de simulación se construyen los pools de recursos o el PoolManager de recursos.

Las estructuras desarrolladas para el manejo de recursos necesarios para la simulación tiene una limitante, solo permiten asignar un tipo de recursos a una tarea a simular. Debido a la complejidad que conlleva tener que sincronizar disponibilidad entre diferentes pools de recursos se decidió que la sincronización de recursos disponibles solo se realizaría a nivel de un mismo pool, pero no a nivel de diferentes pools. Queda planteado como trabajo a futuro poder levantar esta limitante.

Ejemplos de Funcionamiento del pool de recursos

A continuación se explica con un poco mas de detalle el funcionamiento del pool de recursos con un ejemplo práctico. Supongamos que disponemos de un pool con un único recurso, y este esta asignado a las dos tareas del proceso que se muestra en la Figura 55.

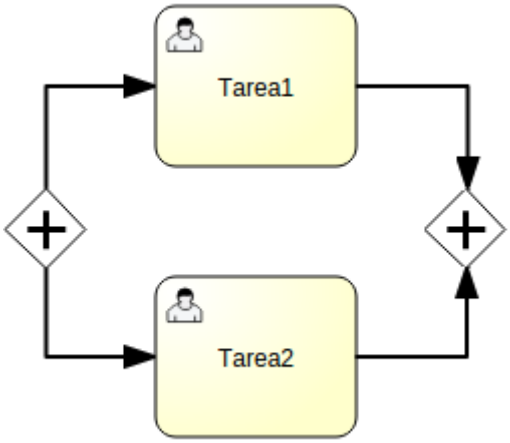


Figura 55: Proceso de ejemplo - Pool de Recursos

Cuando la ejecución del proceso llega a la compuerta paralela se ejecutarán las tareas indicadas en paralelo. Veremos el caso en que se ejecuta primero la tarea 1 y luego la tarea 2.

Al ejecutar la tarea1 se le pide al pool de recursos asignar un recurso a la tarea, que será utilizado por el período definido desde el tiempo que marca el reloj de la simulación hasta una duración determinada que viene dada generalmente por el valor del parámetro processingTime de la tarea.

ResourceAllocationInfo info = pool.allocate(simDate, duration, resourceAmount);

simDate: Reloj actual de la simulación
duration: Durante que tiempo se va a asignar el recurso o los recursos
resourceAmount: cantidad de recursos a asignar

Cuando se llegue a la ejecución de la tarea2, como ambas tienen la misma hora de comienzo (y ningún tiempo de espera definido), cuando se invoque a la función de asignación como se hizo en el caso de la tarea 1, esta arrojará como resultado que tarea2 deberá esperar X tiempo para comenzar debido a que el recurso solicitado no está disponible en ese momento. Los tiempos de espera por los recursos también se registran y se utilizan en los cálculos de tiempos generales de la simulación

6.2.7 HISTORY SERVICE

La información histórica de los procesos se almacena en la base de datos de Activiti en las siguientes tablas que se presentan en la Figura 56.

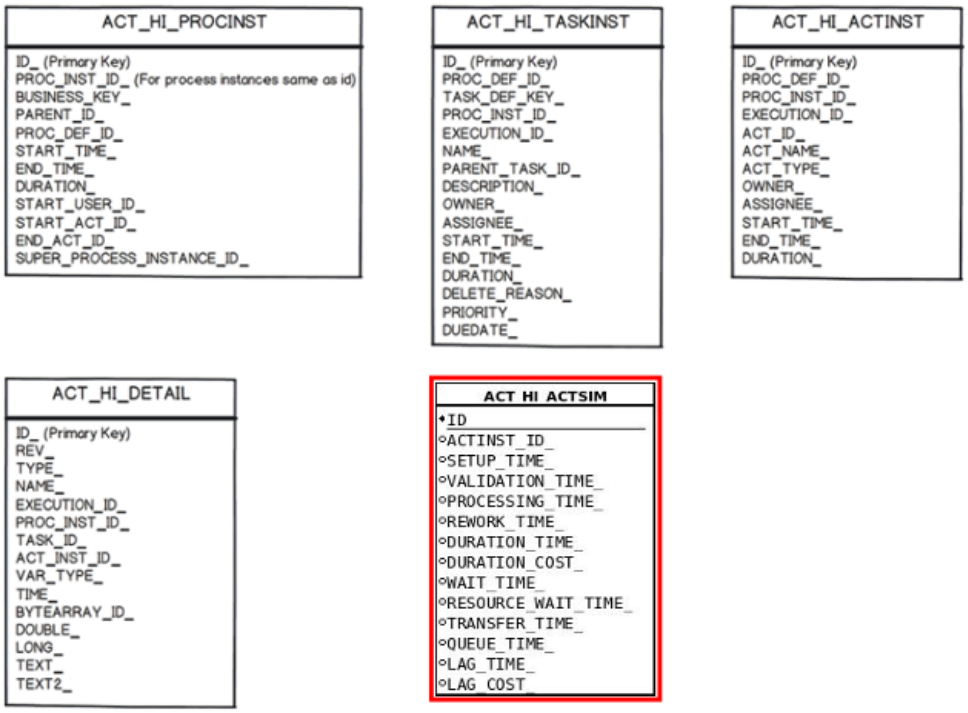


Figura 56: Diagrama de tablas - Histórico de Activiti

Como se puede apreciar en la Figura 56, existen cuatro tablas principales asociadas al registro de históricos de la ejecución que son: **ACT_HI_PROCINST** asociada a los procesos, **ACT_HI_ACTINST** asociada a las actividades en general, y **ACT_HI_TASKINST** asociada a las tareas de usuario. Además la tabla **ACT_HI_DETAIL** permite guardar información adicional. A continuación se describen brevemente los elementos principales en cada una.

ACT_HI_PROCINST: En esta tabla se registran algunas métricas interesantes asociadas a una instancia de proceso en particular, como por ejemplo, tiempo de inicio, fin y duración. Estos valores permiten calcular el tiempo mínimo, máximo y promedio de duración de un proceso.

ACT_HI_TASKINST: Esta tabla almacena información similar a la anterior con la diferencia que lo hace para tareas de usuario.

ACT_HI_ACTINST: Esta tabla contiene información de cada actividad que es ejecutada como parte de una instancia de proceso en particular. Contiene información de tiempo de inicio, fin, duración de cada actividad ejecutada por el motor.

ACT_HI_DETAIL: Cuando se cambia la configuración del motor de workflow y se aumenta el nivel de registro de información a auditoría o full se guarda la información contenida en las variables del proceso.

Para poder guardar la información asociada a la simulación de un escenario fue necesario modificar el modelo de datos propuesto por Activiti y agregar una nueva tabla llamada **ACT_HI_ACTSIM;** esta tabla contiene la información relacionada a la simulación de cada una de las actividades pertenecientes a la instancia de proceso ejecutada en el simulador. Esta tabla se

utiliza como complemento de la tabla **ACT_HI_ACTINST**.

Activiti utiliza para el manejo de la persistencia el framework mybatis [Batis15]. Este framework de persistencia tiene como objetivo eliminar mucho del código JDBC involucrado en el manejo de base de datos. Simplificando se podría decir que es una solución intermedia entre JDBC y el uso de un ORM (Object-Relational mapping, [ORM15]) como Hibernate [Hib15].

Al incorporar la tabla ACT_HI_ACTSIM fue necesario proveer a Activiti con los scripts para crear y eliminar la tabla. Dentro del código del simulador podemos encontrar los siguientes archivos:

- **activiti.h2.create.historySimulator.sql:** Contiene el script de creación de la tabla
- **activiti.h2.drop.historySimulator.sql:** Contiene el script para el drop de la tabla.

6.3 Componentes de la herramienta de modelado

Como se mencionó previamente, para la interfaz gráfica del simulador se extendió el modelador web de Activiti. En el diagrama de la Figura 57, se muestra la definición de tres puntos de extensión para poder incluir en el modelador de Activiti las funcionalidades de simulación propuestas en el trabajo de tesis.

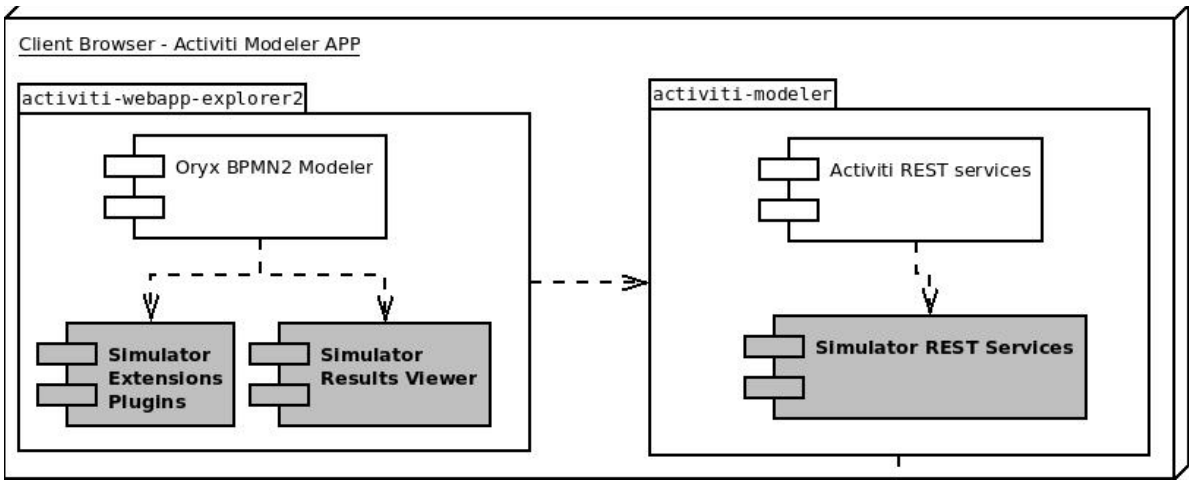


Figura 57: Diagrama de componentes herramienta de modelado

6.3.1 SIMULATOR EXTENSIONS PLUGINS

Oryx provee por defecto un mecanismo de extensión mediante la implementación de plugins. Aprovechando este punto de extensión alguna de las funcionalidades del simulador se incorporaron por esta vía, como por ejemplo el tab donde se muestran los resultados de la simulación, los iconos de la barra del menú donde se puede ejecutar un escenario de simulación, la definición del pool de recursos, etc. En la Figura 58 se muestra un ejemplo de la construcción básica de un plugin de Oryx.

```
if (!ORYX.Plugins)
    ORYX.Plugins = new Object();


ORYX.Plugins.SimulationResults = Clazz.extend({

    facade: undefined,

    construct: function(facade){
        this.facade = facade;

        this.facade.registerOnEvent(ORYX.CONFIG.EVENT_SIMULATION_SHOW_RESULTS, this.showSimulationResults.bind(this));
    },
    showSimulationResults: function(options) {
        Ext.getCmp('maintabs').setActiveTab(1);
    }
});
```

Figura 58: Código de ejemplo plugin de Oryx

Como se menciono anteriormente, alguna de las funcionalidades nuevas se implementaron mediante plugins, por lo que no fue necesario modificar código existente del modelador. Para los casos que no fue posible implementar la funcionalidad mediante plugins fue necesario modificar código del modelador. Un ejemplo de esto es la definición de las propiedades para la simulación, estas se visualizan agrupadas y separadas del resto de las propiedades. En la Figura 59 se puede apreciar como se ven las propiedades de simulación en la herramienta.

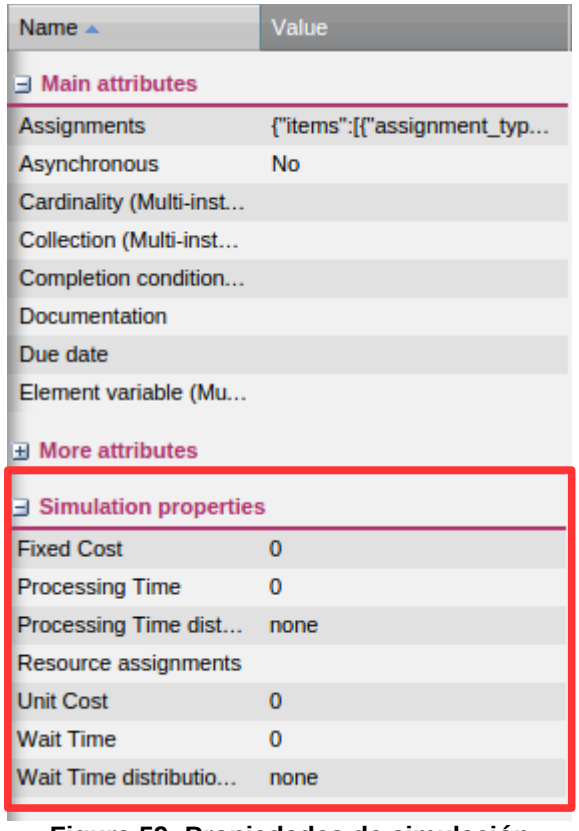


Figura 59: Propiedades de simulación

Como se muestra en la figura 59 recuadrado en color rojo, las propiedades de simulación se integran a las propiedades incluidas en el modelador para los elementos existentes. Para lograr este comportamiento fue necesario modificar el código de la paleta de propiedades del modelador.

En este caso se pueden apreciar propiedades de simulación asociadas a una tarea de usuario, donde es posible parametrizar atributos de la perspectiva costos (costo fijo y unitario, Fixed Cost, Unit Cost) y atributos de la perspectiva tiempo (tiempo de procesamiento, tiempo de espera, Processing Time, Wait Time).

6.3.2 STENCIL SET

Como se mencionó anteriormente en el capítulo de diseño, mediante el Stencil Set se define el lenguaje que se desea modelar. Activiti por defecto provee el stencil necesario para modelar proceso de negocio en BPMN 2.0. Para brindar soporte a BPSim fue necesario modificar el stencil provisto por defecto e incorporar las propiedades de simulación definidas en el capítulo de Análisis de Requerimientos.

En la Figura 60 se muestra un extracto del Stencil Set BPMN 2.0 asociado a una tarea de usuario.

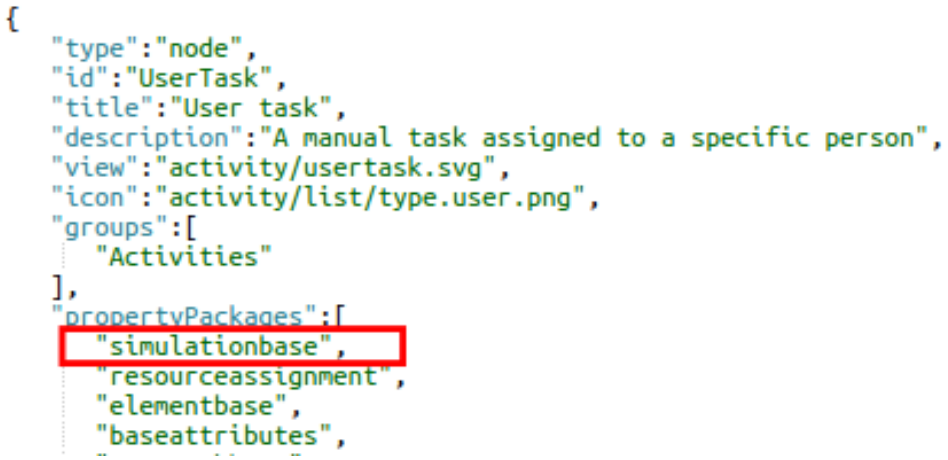


Figura 60: Extracto del Stencil Set BPMN 2 asociado a una tarea de usuario

Se destaca en rojo el punto de extensión con las propiedades de simulación. En este caso las propiedades de simulación están definidas en un paquete con el nombre “simulationbase”. En la Figura 61 se presenta la definición de dicho paquete.

```
{
    "name" : "simulationbase",
    "properties" : [ {
      "id" : "waittime",
      "type" : "String",
      "title" : "Wait Time",
      "value" : "0",
      "description" : "The time between the Successor being allocated and it being
                       actually started",
      "simulation" : true,
      "fordistribution":"nonewt"
    },{
        "id":"waitTimeMin",
        "type":"Float",
        "title":"Wait Time (min)",
        "value":"0",
        "description":"",
        "simulation" : true,
        "fordistribution":"uniformwt|truncated_normalwt"
    },
    {
        "id":"waitTimeMax",
        "type":"Float",
        "title":"Wait Time (max)",
        "value":"0",
        "description":"",
        "simulation" : true,
        "fordistribution":"uniformwt|truncated_normalwt"
    },
    {
        "id":"waitTimeSD",
        "type":"Float",
        "title":"Wait Time SD",
        "value":"0",
        "description":"Wait time Standard Deviation.",
        "simulation" : true,
        "fordistribution":"normalwt|truncated_normalwt"
    },
```

Figura 61: Definiciópn del paquete simulationbase del Stencil Set para BPSim

En la Figura 61 se puede apreciar la definición de la propiedad “waitTime” asociada a la perspectiva tiempo de BPSim. El atributo “fordistribution” determina a que tipo de distribución esta asociado cada uno de los atributos definidos.

6.3.3 REST SERVICES

Activiti encapsula en el proyecto “activiti-modeler” los Web Services REST responsables de la interacción entre el modelador y el repositorio de procesos. Estos servicios están implementados mediante el framework Restlet [RLT15]. Para la implementación del simulador se incorporaron a este proyecto dos nuevos servicios REST, el servicio responsable de la conversión de formato de modelos (JSON a BPMN2 y JSON a BPSim) y el servicio de simulación.

En la Figura 62 se puede apreciar el punto de extensión donde se definen los Web Services necesarios para el simulador.

```
public class ModelerServicesInit {

  public static void attachResources(Router router) {
    router.attach("/model/{modelId}/json", ModelEditorJsonRestResource.class);
    router.attach("/model/{modelId}/save", ModelSaveRestResource.class);

    router.attach("/editor", EditorRestResource.class);
    router.attach("/editor/plugins", PluginRestResource.class);
    router.attach("/editor/stencilset", StencilsetRestResource.class);

    //Add rest services needed for simulator
    //JSON to BPMN2
    router.attach("/model/{modelId}/xml", ModelEditorXmlRestResource.class);
    //JSON to BPSim
    router.attach("/model/bpSim", ModelEditorBpSimRestResource.class);
    //Simulate scenario
    router.attach("/model/runSimulation", SimulatorRestResource.class);

  }
}
```

Figura 62: Punto de extensión Web Services Simulador

En la Figura 62 se puede apreciar la definición de tres servicios, el primero implementado en la clase “ModelEditorXmlRestResource” responsable de convertir el modelo interno de Oryx a BMPN 2.0 en XML. El segundo servicio implementado en la clase “ModelEditorBpSimRestResource” realiza la misma transformación pero asociado a BPSim, y el tercer servicio implementado en la clase “SimulatorRestResource” se encarga de ejecutar el escenario de simulación.

7 Caso de estudio

En este capitulo se expone un caso práctico donde apreciar el funcionamiento del prototipo desarrollado en este trabajo de tesis. A continuación se describirá el caso de estudio seleccionado, se mostrará como configurar/parametrizar el simulador y se presentarán los resultados de la simulación.

7.1 Introducción

Al momento de seleccionar un caso de estudio para desarrollar en el trabajo de tesis se opto por considerar uno de los tres casos expuestos en la documentación oficial de BPSim. Esta documentación esta compuesta por dos documentos, el primero donde se detalla la especificación del estándar y el segundo conocido como guía del desarrollador/implementador.

En este último documento se da una explicación práctica de como usar BPSim para la simulación de procesos. Se exponen tres casos de estudio, donde se exploran las diferentes perspectivas soportadas por el estándar.

Para esta tesis se selecciono el segundo caso expuesto en el documento, descripto con el nombre de “Home Loan” o “Préstamo Hipotecario”. Este caso era el que mejor se adaptaba a las funcionalidades soportadas por el simulador, de esa forma, se pudo trabajar en el realizando solo mínimas modificaciones al diagrama de procesos BPMN 2.0 original.

7.2 Proceso “Préstamo Hipotecario”

En la Figura 63 se puede apreciar el diagrama BPMN 2.0 original. A continuación se describen los principales aspectos del proceso de negocio, incluyendo participantes, flujo de control, etc.

Actores Involucrados: Agente de prestamos, prestatario o solicitante del préstamo, investigador de títulos, escribano.

Disparador del proceso: El prestatario solicita un préstamo hipotecario completando una solicitud y entregándosela al agente de prestamos.

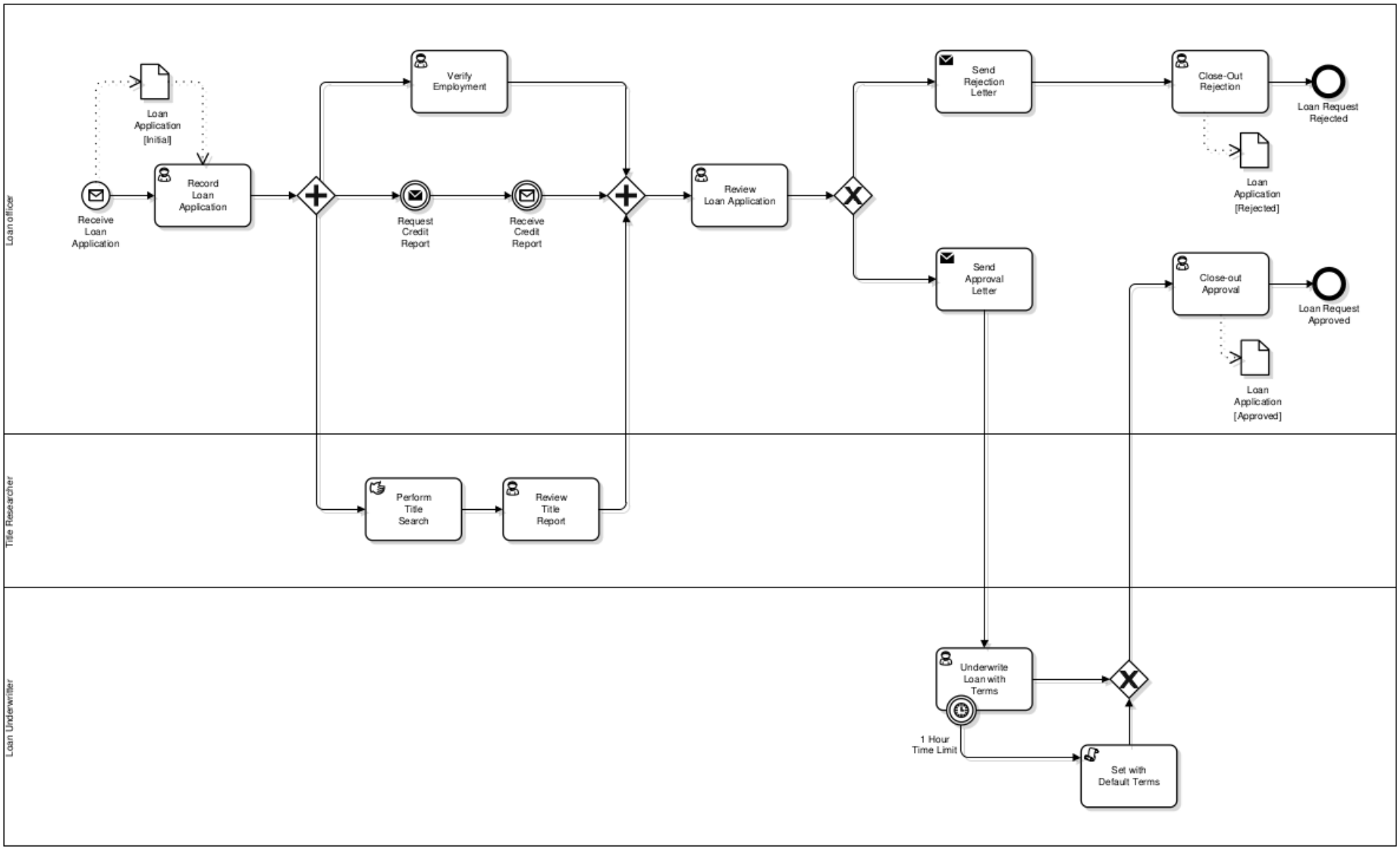


Figura 63: Caso de Estudio - Diagrama BPMN 2 Original de [BPSimImp14]

7.2.1 DESCRIPCIÓN DEL PROCESO

Este proceso tiene como participantes principales una entidad de préstamos y el cliente, de la primera se muestran los roles participantes y el cliente es un participante externo que no esta incluido en el modelo. El proceso comienza cuando el agente de prestamos recibe la solicitud de préstamo completada anteriormente por el prestatario y la ingresa al sistema de prestamos, luego un agente de prestamos verifica la información laboral del solicitante, registrando en el sistema el resultado de su investigación.

En paralelo se solicita y se recibe informe crediticio de tres agencias diferentes de forma consolidada. También en paralelo un investigador de títulos busca en los registros correspondientes a nivel nacional por los bienes involucrados verificando que estos estén libres de gravámenes.

Luego de revisado los antecedentes laborales y analizado la situación crediticia del cliente un agente de prestamos reúne y revisa el expediente (solicitud de préstamo + informe crediticio + estudio de títulos) para aprobar o rechazar la solicitud, si se rechaza, un agente de prestamos envía un aviso de rechazo al solicitante y luego cierra el expediente. En caso de que se apruebe, un agente de prestamos envía un aviso al solicitante y pasa el expediente al escribano para que este redacte el contrato y lo retorne al agente de prestamos, si la redacción demora mas de una hora, se da al solicitante un contrato estándar para que firme. Finalmente el agente de prestamos cierra el expediente aprobado.

7.3 Parametrización original propuesta por BPSim

En esta sección se presenta la parametrización original propuesta por BPSim. El escenario de simulación propuesto en la guía del implementador de BPSim no cubre todas las perspectivas soportadas, solo aplica a las perspectivas control y tiempo.

7.3.1 PARÁMETROS DEL ESCENARIO

En la Tabla 20 se presentan los parámetros globales definidos para el escenario a simular.

Tabla 20: Parámetros del escenario original

Parámetro	Valor
Time unit	Minutes
Duration	40 hs
Replications	3

7.3.2 DISPARADORES DEL PROCESO

En la Tabla 21 se muestra la configuración del parámetro interTriggerTimer para el nodo de inicio.

Tabla 21: Valores nodo de inicio original

Parámetro InterTriggerTimer	Valores
Mínimo	10
Máximo	30
Media	16

7.3.3 DURACIÓN DE ACTIVIDADES

Según la documentación de BPSim la duración de las actividades (parámetro processingTime) se debe modelar utilizando parámetros del tipo distribución normal trunca, por lo que para la simulación de las actividades del proceso se definen los valores que se presentan en la Tabla 22.

Tabla 22: Parámetros duración de actividades

Actividad	Tipo de actividad	Media (minutos)	Desviación (minutos)
Record Loan Application	Tarea de usuario	20	1
Verify employment	Tarea de usuario	30	4
Perform title search	Tarea manual	60	2
Review title report	Tarea de usuario	20	2
Review loan application	Tarea de usuario	30	4
Close out rejection	Tarea de usuario	5	0,25
Close out approval	Tarea de usuario	10	0,25
Underwrite loan with terms	Tarea de usuario	50	10

Para el tipo de distribución normal se debe de proporcionar dos valores adicionales, la cota mínima y la cota máxima. En este caso la cota mínima propuesta de 0 minutos y la cota máxima es de 1000. Se aclara que el valor 0 para la cota mínima es solo para que el resultado de aplicar la distribución no sea negativo, que un valor distinto de 0 sería mas apropiado.

En el caso de las tareas automáticas (envío de mail y tarea de script) la duración será constante, con los valores que se presentan en la Tabla 23.

Tabla 23: Duración actividades automáticas

Actividad	Tipo de actividad	Duración (minutos)
Send rejection letter	Tarea envío de mail	1
Send approval letter	Tarea envío de mail	1
Set default terms	Tarea de script	1

7.3.4 PROBABILIDAD EN PUNTOS DE DECISIÓN

En el diagrama de proceso existe un solo gateway de decisión donde se evalúa si se aprueba o no el préstamo. En la documentación de BPSim se detalla que 8 de cada 30 préstamos solicitados son aprobados, es decir, un 27% de las solicitudes son aprobadas. Esto define los valores de los dos caminos de salida de la decisión que se presentan en la Tabla 24.

Tabla 24: Probabilidad en puntos de decisión

Flujo	Probabilidad (%)
Aprobado	27
Desaprobado	73

7.4 Escenario a simular

A continuación se describe en detalle la parametrización del escenario a simular y los cambios realizados sobre el escenario original. Esencialmente estos cambios responden a funcionalidades no soportadas por el simulador y a simplificaciones realizadas sobre el diagrama original que no afectan la ejecución/simulación del proceso.

7.4.1 CAMBIOS EN LOS ELEMENTOS DEL DIAGRAMA

A continuación se presentan los cambios realizados en los elementos que componen el diagrama original.

Pools y lanes

Estos elementos son meramente descriptivos y no influyen en el comportamiento de un proceso de negocio al momento de ejecutarlo en Activiti y tampoco influyen al momento de simularlo. Por esta razón y con el objetivo de simplificar el proceso a simular en todo los aspectos que sea posible se decidió eliminarlos.

Estados de recepción y envío de mensajes

En la documentación original no se plantea ningún tipo de parametrización para estos estados y teniendo en cuenta que Activiti originalmente no soporta el evento de BPMN 2.0 “Intermediate message throwing event” (por lo tanto el simulador tampoco lo soporta) se decidió remover estos dos estados del modelo a simular.

Timers

El modelo de proceso original presenta un timer del tipo “Boundary TimerEvent” asociado a la tarea de usuario “Underwrite Loan with Terms”. Este tipo de eventos no esta soportado por el simulador, por lo que fue necesario removerlo del modelo a simular. Para no perder riqueza en la diversidad de elementos propuestos por el modelo original se agregó un timer del tipo intermedio (Intermediate catching timer event). Si bien son elementos diferentes que cambian en cierta medida el comportamiento del modelo original, igualmente se incluyo con el objetivo de contar con un evento del tipo timer a la hora de simular.

Tarea de script “Set with Default Terms”

Al dejar fuera del modelo a simular el timer del tipo “Boundary TimerEvent asociado a la tarea “Underwrite loan with Terms” esta tarea deja de tener sentido (ya que se disparaba solo cuando el tiempo de procesamiento de la tarea asociada llegaba a 60 minutos). Si bien el simulador soporta tareas del tipo script se opto por eliminarla, ya que no aportaba a la simulación sin el evento timer que se describe en el punto anterior.

7.4.2 PARÁMETROS DEL ESCENARIO

En la Tabla 25 se presentan los parámetros definidos para el escenario de simulación a ejecutar.

Tabla 25: Parámetros del escenario

Parámetro	Valor
Time unit	Minutes
Duration	0

El parámetro Replications no esta soportado por el simulador, por lo que no se tendrá en cuenta para la parametrización del escenario. El simulador para poder funcionar necesita que obligatoriamente se defina el parámetro triggerCount para el nodo de inicio. Este parámetro define la cantidad de instancias del proceso a simular. Es obligatorio definirlo porque en base a esta parámetro se determina la cantidad de instancias a simular de cada uno de los paths identificados por el módulo Pathfiner.

En base a esta necesidad y considerando que cortar la simulación luego de transcurridas 40 hs como se propone en el escenario original no aporta valor, se optó por configurar el valor del parámetro Duration a 0, y configurando el parámetro triggerCount del nodo inicio en 120.

7.4.3 DISPARADORES DEL PROCESO

El ejemplo de BPSim plantea que para una jornada laboral de 8 horas se reciben 30 solicitudes de prestamos, es decir, una cada 16 minutos. También se plantea definir el parámetro InterTriggerTimer del evento de inicio mediante el uso de una distribución del tipo triangular con los valores que fueron presentados en la Tabla 11.

Cabe destacar que el simulador no soporta parámetros de tipo distribución para este parámetro, no por una limitación del prototipo, sino porque si nos referimos a la documentación de BPSim este parámetro debería ser del tipo NumericParameter, FloatingParameter o DurationParameter; Todos estos tipos de parámetros son subtipos de ConstantParameter, uno de los 4 subtipos de ParameterValue disponibles, dentro de los cuales también se encuentra DistributionParameter, al cual pertenece el tipo de parámetro distribución triangular. Como el simulador se construyó respetando lo descripto en el documento de especificación para el parámetro InterTriggerTimer solo se aceptan valores constante (ConstantParameter y sus derivados). Por lo tanto el valor para este parámetro se configuro en 16 minutos.

7.4.4 DURACIÓN Y COSTOS DE ACTIVIDADES

Según la documentación de BPSim la duración de las actividades (parámetro processingTime) se debe modelar utilizando parámetros del tipo distribución normal trunca. El problema que presenta

modelar la duración utilizando un parámetro del tipo distribución es que el valor del parámetro va a ser aleatorio y no permite realizar un test unitario para validar el escenario de simulación en su totalidad (las duración de las actividades y los cálculos que se realizan a partir de estos valores pasan a ser aleatorios).

Como se mencionó anteriormente, se están simulando 120 instancias del proceso, por lo que validar el resultado del simulador de forma manual es sumamente complejo. Con el objetivo de poder validar el resultado del escenario simulado de forma automática se diseño un test unitario que se incluye como parte de la suite de test del simulador (CasoDeEstudioTest.java) que valida los resultados del escenario del caso de estudio.

Para poder desarrollar el test fue necesario cambiar el tipo de parámetro propuesto (Distribución normal trunca) por parámetros del tipo constante.

Como se mencionó anteriormente, el escenario original no evalúa la perspectiva costos. Para poder evaluar esta perspectiva se optó por definir un costo arbitrario a las diferentes actividades propuestas. En la Tabla 26 se puede apreciar los valores asociados a la duración y costos de las actividades del modelo a simular.

Tabla 26: Parámetros duración y costos de actividades

Actividad	Tipo de actividad	Duración (minutos)	Costo Fijo	Costo Unitario
Record Loan Application	Tarea de usuario	20	5	0
Verify employment	Tarea de usuario	30	20	0
Perform title search	Tarea manual	60	40	0
Review title report	Tarea de usuario	20	0	2
Review loan application	Tarea de usuario	30	25	2
Close out rejection	Tarea de usuario	5	0	3
Close out approval	Tarea de usuario	10	0	3
Underwrite loan with terms	Tarea de usuario	50	20	0

En el caso de las tareas automáticas (envío de mail y tarea de script) se respetó la parametrización propuesta para el escenario original. en la Tabla 27 se presentan los valores utilizados.

Tabla 27: Duración actividades automáticas

Actividad	Tipo de actividad	Duración (minutos)
Send rejection letter	Tarea envío de mail	1
Send approval letter	Tarea envío de mail	1
Set default terms	Tarea de script	1

7.4.5 PROBABILIDAD EN PUNTOS DE DECISIÓN

En esta punto se respecto la parametrización propuesta por el escenario original. Los valores utilizados se pueden apreciar en la Tabla 28.

Tabla 28: Probabilidad puntos de decisión

Flujo	Probabilidad (%)
Aprobado	27
Desaprobado	73

7.4.6 RECURSOS

Esta perspectiva no es analizada en el ejemplo propuesto por BPSim, por lo que en este caso para poder obtener información de los diferentes pool de recursos la siguiente asignación de recursos fue propuesta como parte de los agregados al caso de estudio, con lo valores que se muestran en las Tablas 29 y 30.

Pool de recursos

Como se menciono anteriormente, se definieron tres pools de recursos, Agente de prestamos, Investigador de títulos y Escribano. En la Tabla 29 se especifica la cantidad de recursos asignada a cada pool.

Tabla 29: Parámetros Pool de Recursos

Nombre del Pool	Cantidad de recursos
Agente de prestamos	2
Investigador de títulos	2
Escribano	1

Asignación de recursos

Como se puede apreciar en el modelo del proceso presentado en la Figura 63, las tareas se ubican en los roles que tienen las responsabilidades de realizarlas, por lo que la asignación de recursos a tareas queda definida como se muestra en la Tabla 30.

Tabla 30: Asignación de recursos a las tareas

Tarea	Pool	Cantidad
Record Loan Application	Agente de prestamos	2
Verify employment	Agente de prestamos	2
Review title report	Investigador de títulos	2
Review Loan Application	Agente de prestamos	2
Underwrite loan with terms	Escribano	1
Close out rejection	Agente de prestamos	2
Close out approval	Agente de prestamos	2

7.5 Simulación

En la sección anterior se presento el proceso a simular y la parametrización de los diferentes elementos para poder analizar los resultados desde diferentes perspectivas (tiempo, control, costos y recursos). En este punto se presenta paso a paso como parametrizar el escenario directamente en la herramienta de modelado y simulación desarrollada en esta tesis. En la Figura 64 se presenta el modelo de proceso a simular (Modelado con Activiti Modeler) incorporando los cambios planteados en la sección anterior.

Como se puede apreciar, se eliminaron los eventos de envío y recepción de mensajes (Request Credit Report y Recive Credit Report), se eliminaron pools y lanes, se eliminó la tarea de script (Set with Default Terms) y el evento intermedio del tipo timer asociado a la tarea “Underwrite Loan with Terms”. También se pude apreciar el nuevo evento “timerUnderWrite” del tipo “intermediate timer catching event”.

En la siguiente sección se muestra un resumen de la parametrización realizada en la herramienta de simulación de los parámetros definidos anteriormente.

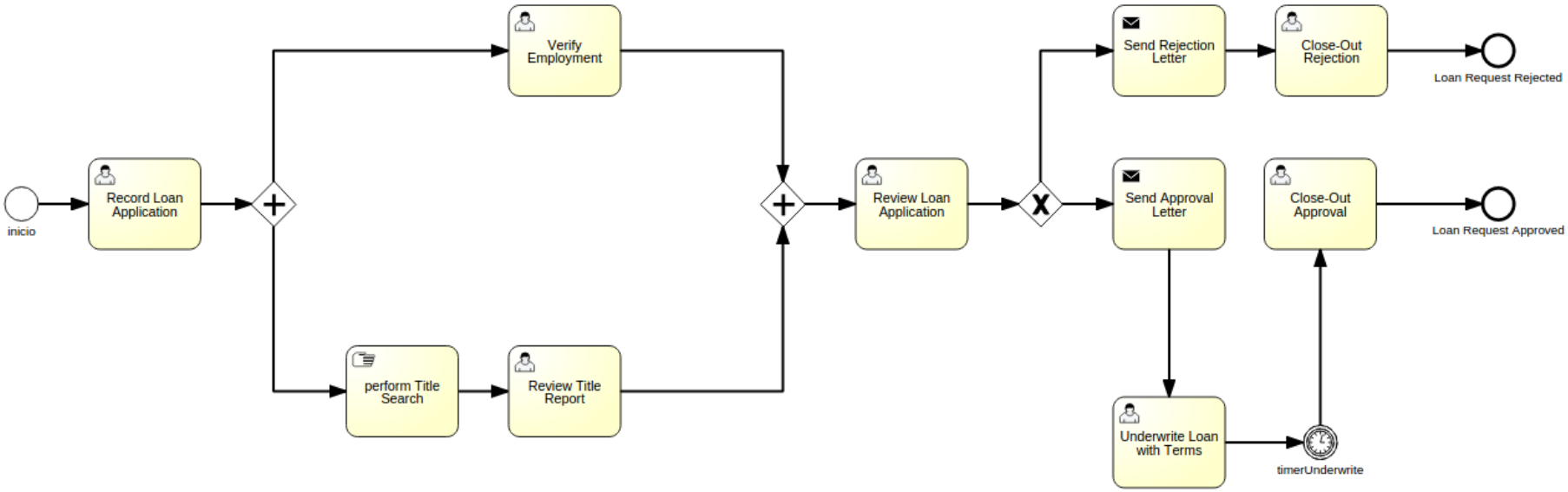


Figura 64: Caso de Uso - Diagrama de proceso en Activiti

7.5.1 PARÁMETROS DEL ESCENARIO

En la Figura 65 se presenta captura de pantalla de la herramienta de simulación. En este caso se puede apreciar la parametrización a nivel de escenario, donde se define por ejemplo la moneda y la unidad de tiempo por defecto, nombre descripción e identificador del escenario y la fecha y hora de comienzo de la simulación.

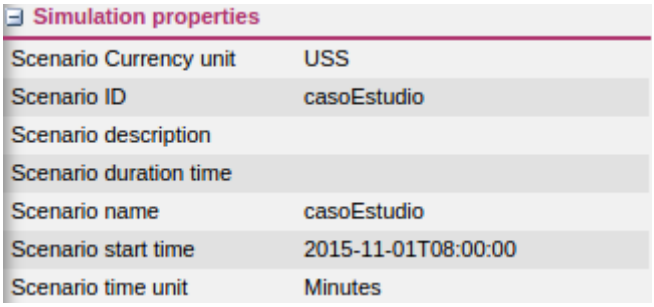


Figura 65: Simulador - Parámetros del escenario

7.5.2 DISPARADORES DEL PROCESO

En la Figura 66 se muestra la parametrización asociada al evento de inicio con los valores definidos y presentados anteriormente.

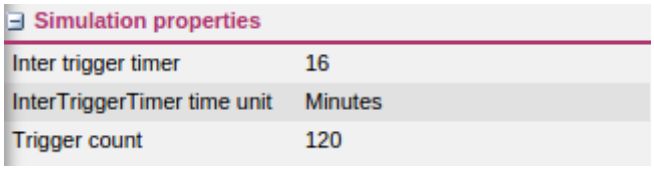


Figura 66: Simulador - Disparadores del proceso

Como se observa en la Figura 66, en la simulación se ejecutarán 120 instancias separadas en el tiempo por 16 minutos.

7.5.3 DURACIÓN DE ACTIVIDADES

En la Figura 67 se presenta como ejemplo la parametrización de la actividad Record Loan Application.

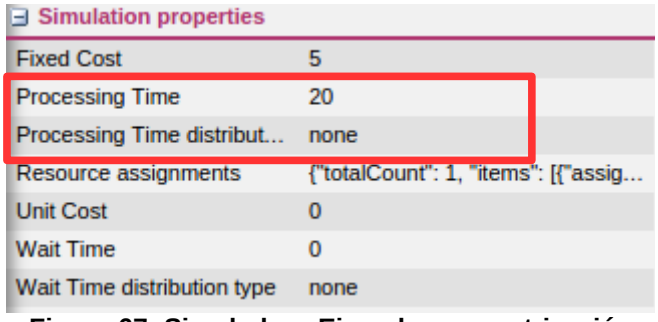


Figura 67: Simulador - Ejemplo parametrización Actividad

Como se puede apreciar en la Figura 67, el recuadrado en rojo muestra los parámetros asociados a la duración (Processing Time) de la tarea. En esta caso se utilizo un parámetro del tipo constante con el valor 20. La unidad de tiempo utilizada es la definida como global a nivel del escenario, en este caso, minutos.

7.5.4 PROBABILIDAD EN PUNTOS DE DECISIÓN

En el proceso a simular existe un solo punto de decisión donde se determina si se aprueba o no el préstamo. En la Figura 68 se puede apreciar la parametrización de los dos flujos de secuencia.

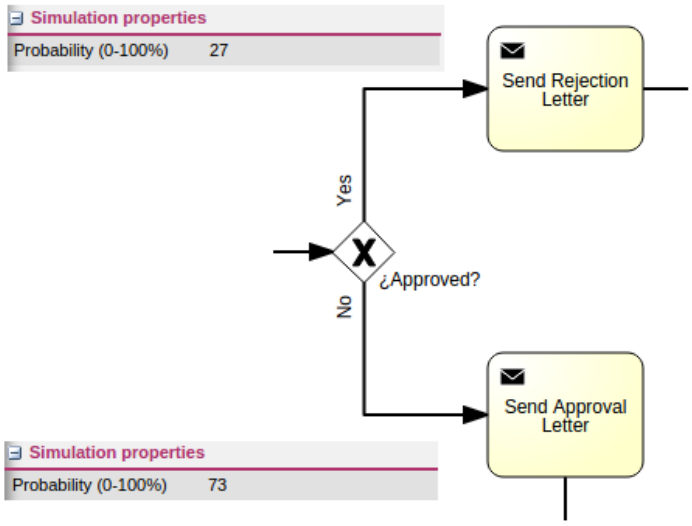


Figura 68: Simulador - Puntos de decisión

7.5.5 DEFINICIÓN DE POOLS Y ASIGNACIÓN DE RECURSOS

En la Figura 69 se puede apreciar la definición de los diferentes pools de recursos, con su respectivo nombre y cantidad de recursos.

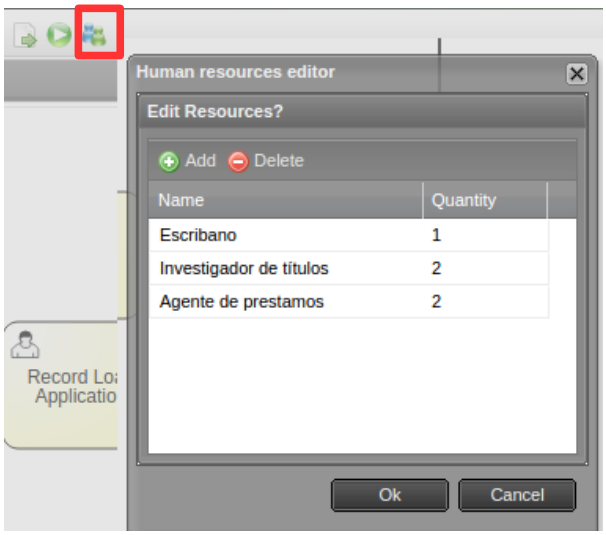


Figura 69: Simulador - Pool de Recursos

En la Figura 70 se muestra como ejemplo la asignación de recursos para la tarea Verify Employment.

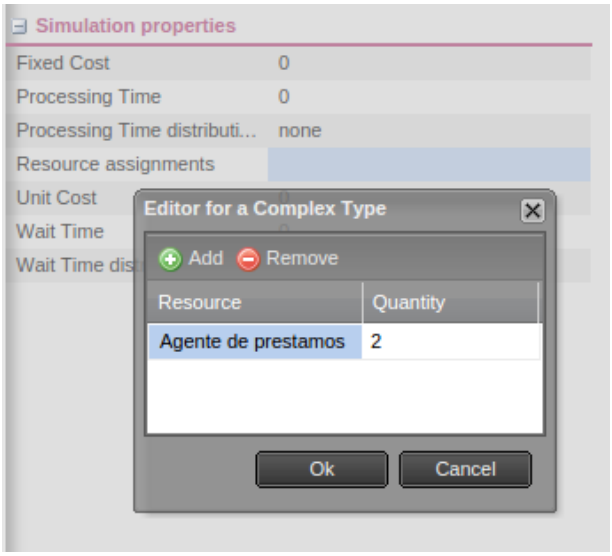


Figura 70: Simulador - Asignación de recursos

7.5.6 COSTOS

En la Figura 71 se presenta la parametrización asociada a costos de la tarea Review loan application.

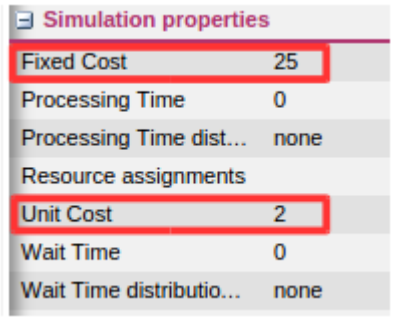


Figura 71: Simulador - Costos

7.5.7 RESULTADOS

En esta sección se presentan los resultados de simular el escenario descripto en la sección anterior. Los resultados se presentan a cuatro niveles, un primer nivel global con datos resumidos sobre el escenario de simulación; un segundo nivel con el desglose de las actividades simuladas detallando por ejemplo duraciones, costos, tiempo de espera por recursos, etc., un tercer nivel donde se presenta la carga de trabajo (cantidad de tiempo trabajado) por cada pool de recursos definido y finalmente un cuarto nivel donde se presentan los caminos (paths) identificados al momento de la simulación.

Nivel 1 - Datos generales del escenario

En la Figura 72 se presentan los resultados resumidos del escenario simulado.

Nombre	Valor
Nombre	casoEstudio
Unidad de Tiempo	minutos
Moneda	USS
Procesos iniciados	120
Procesos completados	120
Fecha de Inicio	01/11/2015 08:00:00
Fecha de Fin	09/11/2015 03:04:00
Duración mínima	136
Duración máxima	9320
Duración promedio	4710.86
Costo total	23800

Figura 72: Resultados resumidos del escenario de simulación

En la tabla presentada en la Figura 72 se pueden apreciar datos que fueron parametrizados previamente en el escenario (Por ejemplo, moneda, unidad de tiempo) y otros datos que surgen como resultado de haber simulado el escenario (por ejemplo fecha de fin, duraciones y costos). Los valores de duración y costos se expresan en las unidades definidas para el escenario, en esta caso minutos y dolares (U$S).

Se puede apreciar que se simularon 120 instancias, toda ellas se completaron con éxito, el costo total fue de 23800 U$S. En la Tabla 31 se muestran los resultados asociados a la duración convertidos a una unidad que facilite su interpretación.

Tabla 31: Duración escenario caso de estudio

Atributo	Valor
Duración mínima	2 horas 16 minutos
Duración máxima	6 días 11 horas y 20 minutos.
Duración promedio	3 días 6 horas y 31 minutos aprox.

Nivel 2 – Resumen de actividades

En la Figura 73 se presenta los resultados de la simulación asociado a actividades y timers. No aparecen en esta tabla estado de inicio y fin del proceso.

nombre	Tipo de actividad	Duración mínima	Duración máxima	Duración promedio	Duración total	Costo mínimo	Costo máximo	Costo promedio	Costo total	Tiempo de espera por recursos mínimo	Tiempo de espera por recursos máximo	Tiempo de espera por recursos promedio	Tiempo de espera por recursos total	Instancias iniciadas	instancias completadas
Record Loan Application	userTask	20	9124	4574.18	548902	5	5	5	600	0	9104	4554,18	546502	120	120
perform Title Search	manualTask	60	60	60	720	40	40	40	4800	0	0	0	0	120	120
Verify Employment	userTask	30	78	40,2	4824	20	20	20	2400	0	48	10,2	1224	120	120
Review Title Report	userTask	20	20	20	2400	40	40	40	4800	0	0	0	0	120	120
Review Loan Application	userTask	30	60	35,75	4290	85	85	85	10200	0	30	5,75	690	120	120
Send Rejection Letter	serviceTask	1	1	1	88	0	0	0	0	0	0	0	0	88	88
Close-Out Rejection	userTask	5	5	5	440	3	3	3	264	0	0	0	0	88	88
Send Approval Letter	serviceTask	1	1	1	32	0	0	0	0	0	0	0	0	32	32
timerUnderwrite	intermediateTimer	1	1	1	32	0	0	0	0	0	0	0	0	32	32
Underwrite Loan with Terms	userTask	50	50	50	1600	20	20	20	640	0	0	0	0	32	32
Close-Out Approval	userTask	10	10	10	320	3	3	3	96	0	0	0	0	32	32

Figura 73: Resultado de actividades y timers del escenario de simulación

En la tabla presentada en la Figura 73 se puede apreciar valores mínimos, máximos, promedio y total para los siguientes ítems:

- Duración
- Costo
- Tiempo de espera por recursos

El total de cada ítem representa la suma del valor individual de todas las instancias simuladas. El tiempo de espera por recursos es el tiempo que una tarea tiene que esperar por los recursos necesarios hasta que estos son liberados. Analizando brevemente los valores de la Figura 73 se puede apreciar claramente que la tarea que mas tiempo demoro en completarse fue “Record Loan Application”, pero también se puede ver que es la que tiene mas tiempo de espera por liberación de recursos.

Nivel 3 – Resumen de uso de recursos

En este nivel se presenta la información de utilización de los pools de recursos. En la Figura 74 se presenta el resultado de uso de los recursos asociado al caso de estudio.

Pool de recursos	Tiempo de uso	% uso
Escribanos	1600	14,25
Investigador de Titulos	4800	42,76
agenteDePrestamos	20720	184,6

Figura 74: Utilización de los recursos

Los valores de tiempo de uso que se aprecian en la Figura 74 están datos en la unidad de tiempo definida para el escenario de simulación, en este caso, minutos. Se puede apreciar claramente que el tiempo de utilización de los recursos del pool “agenteDePrestamos” es sumamente elevado en comparación con los otros pools. Si expresamos ese valor de una forma más amigable, podemos ver que el tiempo de uso excede los 14 días.

Si tenemos en cuenta que la duración del escenario (fecha de fin – fecha de inicio) es de 11224 minutos, podemos decir que los recursos de los pools de escribanos e Investigador de Títulos tienen tiempo ocioso (14,25% del total del tiempo para el pool de “Escribanos” y 42,76% para el pool “Investigador de Títulos”), en cambio los recursos del pool agenteDePrestamos están muy sobrecargados (184,6% del total del tiempo). Si el caso de estudio fuera un proceso de la vida real, esta sería una situación a prestar atención, porque los recursos de este pool están actuando de cuello de botella y afectan el desempeño de todo el proceso.

Nivel 4 – Paths simulados

En este nivel se presentan los caminos (paths) identificados y ejecutados por el simulador. En la Figura 75 se pueden apreciar los dos paths identificados para el caso de estudio.

Path identificado	Instancias	Elementos
Path 1	32	inicio
		Record Loan Application
		Verify Employment
		perform Title Search
		Review Title Report
		Review Loan Application
		Send Rejection Letter
		Close-Out Rejection
		Loan Request Rejected
Path2	88	inicio
		Record Loan Application
		Verify Employment
		perform Title Search
		Review Title Report
		Review Loan Application
		Send Approval Letter
		timerUnderwrite
		Underwrite Loan with Terms
		Close-Out Approval
		Loan Request Approved

Figura 75: Paths identificados en caso de estudio

Para cada path se detallan los elementos que lo componen. Solo se muestran los elementos a los cuales se les haya asignado id en el modelador, por esa razón en la Figura 75 no aparecen las compuertas ni paralelas ni de decisión.

8 Conclusiones y trabajo a futuro

En esta sección del documento se presentarán las conclusiones derivadas del trabajo realizado, así como ideas y sugerencias que se podrían “desarrollar/tener en cuenta” en trabajos futuros asociados a la simulación de procesos de negocio.

8.1 Conclusiones

Durante este trabajo de tesis se investigaron los conceptos y requerimientos asociados a la simulación de proceso de negocio, desarrollando un motor y herramienta visual de de simulación totalmente funcional. Se toma como punto de partida una da las suites BPMS Open Source mas reconocidas en la comunidad como es Activiti y se la complementa con funcionalidades de BPS, tratando entre otras cosas de llevar a dicho proyecto un paso mas cerca de ser una solución completa de BPMS. Además como usuario activo de todo tipo de software Open Source de hace mas de 10 años espero con este prototipo aportar mi granito de arena y contribuir al desarrollo de esta comunidad.

Como parte de la tesis se investigó al inicio un nuevo estándar de simulación en incipiente desarrollo, el estándar BPSim que promete sentar nuevas bases en lo que a simulación de procesos se refiere. BPSim comenzó a gestarse en manos de la WFMC en el año 2011, viendo la luz su primera versión en el año 2013.

En este sentido, era un riesgo embarcarse en un trabajo de tesis que se centraría en la aplicación de un estándar cuyo documento de especificación oficial consta apenas de 35 páginas. Adicionalmente, existían (y existen) muy pocas herramientas que implementaran esta nueva especificación (tres o cuatro en ese entonces) y de esas herramientas, solo una era Open Source.

Si bien el riesgo era grande fue uno de los principales atractivos a la hora de escoger el tema de la tesis, poder investigar y trabajar en un campo en el cual recién se están dando los primeros pasos y las oportunidades de realizar aportes son mayores. Otro de los grandes motivadores fue el hecho de tener que tomar como base software ya existente de clase mundial para analizarlo, entenderlo y modificarlo sin degradar su calidad. Tener la posibilidad de modificar una de las dos suite BPMS reconocidas a nivel mundial dentro del software Open Source fue sumamente motivador y desafiante.

Los objetivos definidos para la tesis en la sección 1.2 del documento fueron alcanzados, se logró comprender los conceptos involucrados en la simulación de procesos de negocio y su adecuación al estándar BPSim y como se integra este con el estándar BPMN 2.0, también se logró plasmar estos conceptos en el desarrollo del prototipo realizado.

Con los resultados obtenidos del capítulo dos, “Estado del arte” y el capítulo tres “Herramientas evaluadas de soporte”, se cumplió con los objetivos específicos **OE1** y **OE2**. Estos capítulos presentan un resumen del análisis realizado de BPSim y BPMN 2.0 de forma separada y los mecanismos de integración propuestos por BPSim.

El capitulo tres presenta el trabajo de relevamiento de mas de 20 herramientas propietarias y Open Source relacionadas al modelado y simulación de procesos de negocio, el cual permitió

dilucidar claramente cual es el grado de adopción de BPSim a nivel comercial. Además este trabajo también sirvió como paso fundamental para la selección de las herramientas base sobre las cuales finalmente se construyo el prototipo. Durante estos capítulo se comenzó con el análisis de la integración propuesta por BPSim con el metamodelo de BPMN 2.0, pero fue recién con la culminación del análisis y diseño del prototipo que se pudo cumplir cabalmente con el objetivo **OE3**.

Los capítulos cuatro, cinco y seis permitieron cumplir con el objetivo **OE4**. Durante estos capítulos se analizó en detalle el problema, se diseño una solución que cumpliera con los requerimientos establecidos y finalmente se implemento. En el capítulo siete se puede apreciar el prototipo en funcionamiento mediante la ejecución de un caso de estudio tomado de la guía para el implementador oficial de BPSim. Durante el trabajo surgieron muchas ideas y mejoras que lamentablemente no fue posible tenerlas en cuenta, pero si se dejarán planteadas como posibles trabajos a futuro. Estas ideas quedaron plasmadas en la sección “Trabajos a Futuro” del este capítulo.

El prototipo desarrollado provee el mismo nivel de funcionalidades provistas por el resto de herramientas de simulación BPSim analizadas al momento de comenzar la tesis, por lo que si bien es un prototipo inicial constituye una buena base para continuar el desarrollo por parte de la comunidad. Adicionalmente se debería realizar también una etapa de testing un poco mas exhaustiva que la realizada en el trabajo de tesis.

La idea básica detrás de la simulación de procesos de negocio es simple, se busca adquirir conocimiento y alcanzar decisiones estudiadas con respecto a un sistema del mundo real (el negocio). Muchas veces estudiar este sistema no es una tarea fácil, por lo que estudiar el sistema directamente no es una opción (por razones de costos, tiempos, etc.). En estos casos procedemos indirectamente creando y estudiando otra entidad (el modelo de simulación) que es suficientemente similar al sistema real y nos permite extrapolar el conocimiento adquirido sobre el modelo.

La simulación de procesos de negocio es cada vez mas común en el mundo de los negocios y las herramientas de BPS proporcionan un camino válido para analizar y comprender el comportamiento de un sistema. Es sumamente importante que este tipo de herramienta estén construidas sobre estándares que permitan seguir evolucionando en conjunto y no a base de esfuerzos individuales. BPSim es una apuesta compartida de muchas empresas (algunas de ellas referentes del mercado actualmente) que ya poseen herramientas del tipo BPS para evolucionar hacia la interoperabilidad y alinearse a BPMN 2, el estándar para modelado de procesos mas difundido actualmente. Creemos fehacientemente que este trabajo de tesis logra aportar su granito de arena en pos de lograr este objetivo, difundir y adoptar el estándar BPSim.

8.2 Trabajos a Futuro

A continuación se describen las lineas de trabajo que fueron surgiendo durante la tesis. Sería sumamente interesante poder abordar a futuro estos tópicos con la finalidad de complementar el prototipo desarrollado.

- **Parámetros históricos:** Un punto interesante soportado por BPSim es el de poder definir valores para un parámetro basados en datos históricos. Cuando se quiere simular un proceso ya existente para optimizar alguna variable en particular lo mas acertado sería

poder utilizar datos históricos de simulaciones anteriores que puedan ayudar a determinar el comportamiento real de esa variable, en vez de tener que usar una distribución que emule ese comportamiento.

- **Animación del proceso al simular:** Una funcionalidad muy interesante que poseen algunos de los simuladores analizados en el trabajo de tesis (por ejemplo Bizagi) es la de poder ver mediante una animación en tiempo real la ejecución de las diferentes instancias simuladas sobre el modelo de proceso asociado. Esta funcionalidad no afecta en nada al simulador en si, solo a la herramienta gráfica de simulación. Es un plus muy interesante de investigar ya que esta funcionalidad hace que la herramienta se distinga de la media del mercado.

- **Ampliar soporte a elementos BPMN2:** Durante el diseño del simulador se decidió dar soporte a los elementos mas utilizados del estándar BPMN 2, los que integran el nivel 1 de conformidad. Activiti brinda soporte a otros elementos mas complejos, como pueden ser gateway inclusivos y de eventos, diferentes tipos de eventos que no sean timers, sub procesos expandidos, transacciones, etc.. Sería interesante poder dar soporte al resto de elementos BPMN2 que soporta Activiti.

- **Ampliar soporte de BPSim:** Al igual que pasa con el soporte de elementos a de BPMN 2, el prototipo no soporta todos los parámetros posibles ni perspectivas propuestas por BPSim. Sería deseable poder incorporar a corto plazo las perspectivas de prioridad y propiedades, así como los parámetros de las perspectivas costo, tiempo, recursos y control que no fueron implementados.

- **Mejora en el pool de recursos:** El pool de recursos desarrollado cumple las necesidades básicas planteadas para el prototipo. Existe por ejemplo la limitante de poder asignar a una tarea recursos de un solo tipo, los recursos están disponibles 24 hs, poder diferenciar entre recursos pertenecientes a un mismo pool, por ejemplo, poder decir que de 2 recursos definidos en el pool A, el recurso A1 es 30% mas productivo que el recurso A2, y tener en cuenta el impacto que tiene este factor en la tarea asignada.

- **Definición de calendarios de trabajo:** Como se menciono anteriormente asociado a los recursos, disponibilidad 24 hs, esto mismo pasa para a nivel de proceso. Se hace necesario a futuro poder definir un calendario de trabajo tanto para los recursos como para el proceso en si, don de puede estipular horarios de trabajo, feriados, etc.. Para lograr mejores resultados y poder simular escenarios cada vez mas complejos se hace necesario incorporar esta funcionalidad.

- **Comparación de escenarios:** Las funcionalidades actuales no permiten la posibilidad de comparar el resultado de la simulación de dos escenarios por ejemplo, tampoco existe la posibilidad de imprimir los resultados o descargarlos a una planilla de cálculo. Estas funcionalidades serían un plus interesante para un usuario final a la hora de analizar y comparar resultados de una simulación.

- **Herramienta gráfica independiente de la herramienta BPMS:** Sería un plus mas que atractivo poder diseñar una herramienta independiente de la suite a Activiti para la simulación de procesos. Esta herramienta permitiría simular, no modelar procesos. Además sería desarrollada con tecnologías mas modernas que la del modelador de Activiti y permitiría por ejemplo simular procesos realizados en otros modeladores previa conversión a BPMN 2 genérico o a BPMN2 de Activiti.

9 Bibliografía

Aalst11: Van der Aalst, Process Mining, 2011
ACM15: https://commons.apache.org/proper/commons-math/, Apache Commons Math, ,
ActInAc12: Tijs Rademaker, Joram Barrez, Activiti in Action, 2012
Activiti15: http://www.activiti.org/components.html, Activiti Web Page, 2015,
Apache04: Apache Software Foundation, Apache License 2.0, 2004,
Barrez2013: Joram Barrez (Activiti core developer), http://www.jorambarrez.be/blog/2013/03/18/a-sad-day-for-open-source-camunda-decides-to-fork-activiti/, 2013,
Batis15: http://mybatis.org/mybatis-3/es/, Mybatis Official Web Site, ,
BIMP15: http://qbp-simulator.cloudapp.net/, BIMP Official Web Site, 2015,
BPMaN12: Jan Mendling, Matthias Weidlich, Business Process Model And Notation - 4th International Workshop, 2012
BPMN15: www.bpmn.org, Oficial Web Site, 2015,
BPMNSpec11: OMG, BPMN Standard Specification, 2011
BPSim2013: www.bpsim.org, Business Process Simulation Interchange Standard, 2013,
BPSimImp14: WfMC, BPSim Implementer's Guide, 2014
BPSimSpec13: WfMC - BPSWG, Business Process Simulation Specification, 2013
DUMAS08: Marlon Dumas, MAnfred Reichert, Business Process Management, 6th International Conference, Milan, Italy, 2008
ESTOC15: https://es.wikipedia.org/wiki/Estoc%C3%A1stico, , ,
GAMMA94: Erich Gamma, Richard Helm, Design Patterns, Elements of Reusable Object-Oriented Software, 1994
GHUB15: https://es.wikipedia.org/wiki/GitHub, , 2015,
Hib15: http://hibernate.org/orm/, Hibernate Official Web Site, ,
ISO8601: https://es.wikipedia.org/wiki/ISO_8601, Norma ISO 8601, ,
JACK15: https://github.com/FasterXML/jackson, Jackson Official Web Site, ,
JSON15: http://www.json.org/, JSON Web Site, 2015,
ORM15: https://es.wikipedia.org/wiki/Mapeo_objeto-relacional, , ,
Oryx06: http://bpt.hpi.uni-potsdam.de/Oryx/, Oryx - Hasso-Platter-Institute, 2006,
P&K05: Bernd Page y Wolfgang Kreutzer, The Java Simulation Handbook, 2005
PSEU15: https://es.wikipedia.org/wiki/N%C3%BAmero_pseudoaleatorio, Números pseudoaleatorios, ,
RDF15: http://www.w3.org/RDF/, RDF - Resource Description Framework, 2015,
RLT15: http://restlet.com/projects/restlet-framework/, Restlet Official Web Site, 2015,
SHAN75: Robert Shannon, Systems Simulation: The art and science, 1975
VenLock15: https://en.wikipedia.org/wiki/Vendor_lock-in, Wikipedia, 2015
VERM09: Naresh Verma, Business Process Management: Profiting from Process, 2009
WES07: Mathias Weske, Business Process Managment: Concepts, Languages, Architectures, 2007
WES12: Mathias Weske, Business Process Managment: Concepts, Languages, Architectures, Second Edition, 2012
WfMC15: http://www.wfmc.org/, Official Web Site, 2015,
WfMC99: WfMC, Terminology & Glossary, 1999