Trabajo Fin de Grado
Grado en Ingeniería Informática

Minería de procesos para la mejora de la seguridad en sistemas de información Web

Process Mining to improve Security in Web Information Systems

Autor

Raúl Piracés Alastuey

Directoras

Raquel Trillo Lado
Simona Bernardi

Universidad de Zaragoza
Escuela de Ingeniería y Arquitectura

Junio 2016

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

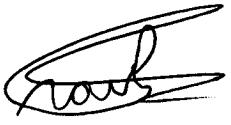
D./Dª. Raúl Piracés Alastuey,

con nº de DNI 73026929C en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado, (Título del Trabajo) Minería de procesos para la mejora de la seguridad de sistemas de información Web,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, a 17 de Junio de 2016



Fdo: Raúl Piracés Alastuey

Agradecimientos

Quisiera empezar, agradeciendo a Raquel y Simona, las directoras de mi proyecto, su ayuda, dedicación y asesoramiento a lo largo de este proyecto, ya que además de permitirme la realización del mismo, me han permitido aprender nuevos conocimientos y aspectos desconocidos hasta el momento para mí, en este último año de grado. También quisiera agradecer a Abel Gómez y José Javier Merseguer, su ayuda, consejo y revisión en el proyecto, lo que me ha ayudado a mejorar el mismo y los conocimientos sobre los que se basa. Además quisiera citar al grupo de investigación de "Sistemas de Información Distribuidos", por su prestación y ayuda en todo momento.

Por otra parte, quiero darles las gracias a mis compañeros y amigos de clase, con los que he pasado muchos momentos buenos y superado siempre otros. Con todos ellos, sin lugar a dudas, he aprendido y he disfrutado a lo largo de estos cuatro años. No quiero olvidar tampoco a mis amigos de toda la vida, que siempre han estado animándome y ayudándome en todo momento.

Por último, y no por ello menos importante, agradecer enormemente a mis padres y hermana, su apoyo incondicional, ánimos, paciencia y ayuda que me han sabido prestar durante todos estos años, sobre todo en los malos y más estresantes momentos. Sin ellos, no hubiese podido llegar hasta aquí y alcanzar mi meta propuesta.

Por ello, quisiera dedicar a todos los citados anteriormente, el esfuerzo y trabajo que he puesto en este proyecto.

¡Muchas gracias a todos!

Minería de procesos para la mejora de la seguridad en sistemas de información Web

Resumen

El uso de sistemas de información Web para automatizar la gestión de información y datos digitales está ampliamente extendido en la gran mayoría de entidades y organizaciones de la sociedad desarrollada. Además, gran parte de estos sistemas constituyen un activo fundamental y clave en diversos ámbitos. Por ello debe asegurarse su correcto funcionamiento e impedirse que sean atacados.

Tradicionalmente, los sistemas de información Web se han protegido mediante medidas preventivas y reactivas. Las medidas preventivas se basan en el análisis de patrones de ataques existentes y documentados, y el establecimiento de reglas en sistemas de gestión de eventos y seguridad de la información (*Security Information and Event Management*, SIEM) que eviten las condiciones que permitan dichos ataques. Las medidas reactivas no impiden que se produzcan ataques y actúan cuando se detecta un comportamiento anómalo. El principal inconveniente del uso de medidas preventivas es que no permiten analizar y evitar ataques no conocidos (*"zero-day attacks"*). Por otro lado, las medidas reactivas se activan una vez ha ocurrido una intrusión en el sistema, esto puede implicar que el ataque se detecte demasiado tarde, cuando el daño producido por dicho ataque sea irreparable o reparable a un alto coste.

En este trabajo se define un método para la detección de nuevas amenazas en sistemas de información Web antes de que estas se materialicen y produzcan algún tipo de daño. Esta metodología se basa en técnicas de minería de procesos (*Process Mining*) y la creación semiautomática de modelos formales a partir de los diagramas elaborados durante las fases de diseño y desarrollo del sistema que se desea proteger y a partir de los ficheros *logs* del sistema en producción. Además se presentan los resultados experimentales obtenidos aplicando el método propuesto y las herramientas de soporte al sistema de información Web empleado por el grupo de investigación Sistemas de Información Distribuidos para la gestión y publicación de artículos.

Por último, señalar que durante el desarrollo de este trabajo final de grado (TFG) se ha elaborado un artículo de investigación, que ha sido aceptado para su publicación en la IV edición del Congreso Nacional de I+D en Defensa y Seguridad (DESE I+D 2016).

Índice

Índice de figuras

Índice de cuadros

1. Introducción

Los objetivos principales de este proyecto, son analizar y diseñar métodos basados en *minería de procesos* [1] para mejorar la seguridad de sistemas de información Web, y validar mediante pruebas en dos sistemas de información Web en funcionamiento los métodos propuestos.

1.1. Planteamiento del problema

Dada la importancia actual de los sistemas de información Web, y su valor para las organizaciones, se plantea un método para mejorar la seguridad de estos. En concreto, a lo largo de este Trabajo de Fin de Grado (TFG), se desarrolla y valida un método para tratar de detectar y gestionar anomalías y ataques en sistemas de información Web. El método está compuesto de los siguientes pasos o etapas: 1) monitorización de los sistemas a proteger, 2) obtención de modelos formales del comportamiento esperado del sistema (*modelo estándar o normativo*) y del comportamiento en producción del mismo (*modelo operativo o productivo*), y 3) análisis y comparación de ambos modelos para la detección de diferencias. Cabe aclarar y mencionar, que el modelo normativo es aquel que representa el comportamiento "conocido" o "esperado" del sistema; mientras que el modelo operativo representa el comportamiento del sistema en ejecución; y que los modelos formales empleados en este TFG son modelos en redes de Petri [2].

Respecto al modelo normativo del sistema, este se obtiene a partir de la especificación de diseño del mismo en el Lenguaje de Modelado Unificado (UML) [3], a través de una transformación "modelo-a-modelo" (M2M) [4], de UML a redes de Petri. Por otra parte, el modelo operativo del sistema, se obtiene de los *logs* del mismo, aplicando técnicas de minería de procesos (en particular, algoritmos de *process discovery* o descubrimiento de procesos).

Mediante el análisis y estudio de los modelos, se obtienen una serie de patrones de ataque o amenazas potenciales, así como comportamientos benignos no considerados en el desarrollo del sistema (nuevas tendencias de uso). Tras la obtención de estos patrones, se introducen en los sistemas de monitorización reglas y acciones para mitigar y/o anular los efectos de las potenciales amenazas que pueden llegar a materializarse en futuros ataques (de manera semi-automática). En concreto, se aplican técnicas de minería de procesos que permiten averiguar y comprobar el nivel de ajuste/conformidad de los modelos respecto a los *logs*, con métricas que reflejan este ajuste. Estas técnicas también permiten la detección de desviaciones de trazas en los *logs* que pueden representar anomalías (ataques) o, simplemente, comportamientos no contemplados en el diseño del sistema.

1.2. Motivación

Hoy en día, la mayor parte de entidades y organizaciones emplean sistemas de información Web para automatizar la gestión de información y datos que generan y recopilan de numerosas fuentes heterogéneas, tanto internas como externas. Por ello, estos sistemas de información se han conver-

tido en un activo fundamental y clave que debe ser protegido. Debido a esto, en este proyecto se plantea el desarrollo de un método para la detección y gestión de amenazas persistentes avanzadas en sistemas de información Web para tratar de garantizar la seguridad de dichos sistemas. Para ello, se plantean técnicas, herramientas y métodos de monitorización, procesado y análisis de *logs* del sistema (en el contexto de la minería de procesos).

1.3. Descripción del entorno de aplicación

El entorno de aplicación del método desarrollado en este proyecto, son los sistemas de información Web correspondientes al "Grupo de I+D en Computación Distribuida (DisCo)" [5] y el correspondiente a la biblioteca digital del "Grupo de Sistemas de Información Distribuidos (SID)" [6]. Estos sistemas de información considerados, constituyen un activo fundamental y clave para cada grupo de investigación, por lo que debe ser protegido y asegurado convenientemente.

Actualmente, ambos sistemas, constan de un sistema de seguridad básico y una serie de copias de seguridad, que ante caso de pérdida de información podrían ser utilizadas para recuperar el estado del sistema y su información correspondiente. No obstante, el método desarrollado en este TFG ha sido aplicado en estos sistemas, con los siguientes fines: mejorar su seguridad (por ejemplo, detectando y gestionando amenazas persistentes avanzadas) y validar el método propuesto. Tal y como se explicará en las conclusiones del proyecto, tras aplicar el método desarrollado, ambos sistemas constan de un nivel de seguridad superior.

1.4. Objetivos

El principal objetivo de este proyecto es el desarrollo de un método para la detección y gestión de amenazas persistentes avanzadas en sistemas de información Web para tratar de garantizar la seguridad de dichos sistemas. Por ello, en este proyecto se monitorizan y analizan dos sistemas de información Web (citados en la descripción del entorno de aplicación). Para alcanzar este objetivo global se han planteado los siguientes sub-objetivos:

- **Objetivo 1:** Monitorización y control de los sistemas.
- **Objetivo 2:** Definición de los modelos de comportamiento de los sistemas actuales mediante UML.
- **Objetivo 3:** Obtención de modelos normativos a partir de las especificaciones UML de los sistemas.
- **Objetivo 4:** Obtención de modelos operativos a partir de los *logs* de los sistemas.
- **Objetivo 5:** Análisis de los modelos obtenidos para la detección de anomalías o comportamientos no esperados en el sistema.

1.5. Metodología de trabajo

La metodología aplicada para la planificación y elaboración del presente trabajo, se ha basado en el marco de desarrollo ágil *Scrum*[1]. Esto ha permitido una estrategia de desarrollo incremental y cíclica, en lugar de la planificación y ejecución completa de cada una de las diferentes fases de desarrollo (análisis, diseño, implementación y pruebas) de forma secuencial. Por último, cabe destacar que esta metodología prima la calidad del resultado y el conocimiento tácito, más que la calidad de los procesos empleados. En la Figura 1, se muestra un diagrama del proceso cíclico empleado en el desarrollo del TFG.

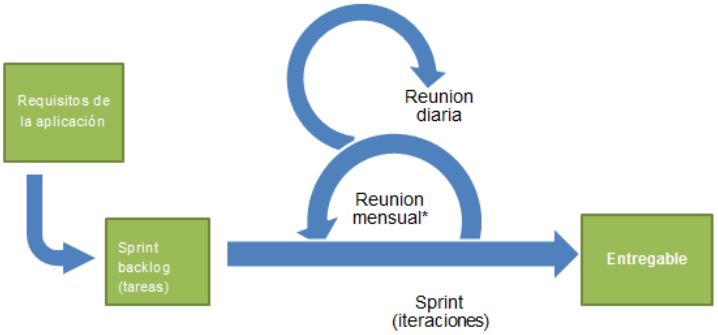


Figura 1: Marco de trabajo SCRUM[2]

.

1.6. Planificación temporal y cronograma

Para el desarrollo del proyecto se ha apostado por la consecución de las siguientes tareas (en orden cronológico):

1. Implantación de las herramientas de monitorización de los sistemas de información Web a analizar.
2. Definición y/o actualización de los modelos de comportamiento estándar de los sistemas de información Web a analizar mediante UML.
3. Obtención de modelos de análisis formal a partir de los diagramas UML.
4. Obtención de modelos de comportamiento operativo de los sistemas y su análisis y estudio frente a los modelos esperados (o normativos).
5. Elaboración de la memoria de trabajo y preparación de la defensa del proyecto.

En la Figura 2 se listan estas tareas junto con su duración, fecha de inicio y fecha de finalización.

[1]Scrum: https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software), accedida el 4 de Junio de 2016.
[2]Imagen obtenida desde: https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software), accedida el 4 de Junio de 2016.

Nombre	Fecha de fin	Fecha de inic...
Formación y documentación mediante bibliografía recome...	13/02/16	7/02/16
Implantación de herramientas de monitorización de los sis...	7/03/16	14/02/16
Desarrollo de diagramas UML del sistema	21/03/16	8/03/16
Obtención de modelos de análisis formal a partir de UML	21/04/16	22/03/16
Obtención de modelos de comportamiento operativo y an...	1/06/16	22/04/16
Documentación	1/06/16	14/02/16
Documentación de tareas realizadas	1/06/16	14/02/16
Elaboración de la memoria de trabajo y preparación de la...	24/06/16	2/06/16

Figura 2: Lista de tareas realizadas correspondientes al cronograma

.

A su vez, en la Figura 3 se muestra el diagrama de Gantt correspondiente al desarrollo del proyecto. En el diagrama se refleja la planificación temporal de las fases del trabajo del proyecto. No obstante, una planificación más detallada, junto con una estimación de los costes del proyecto se encuentra disponible en el anexo D.

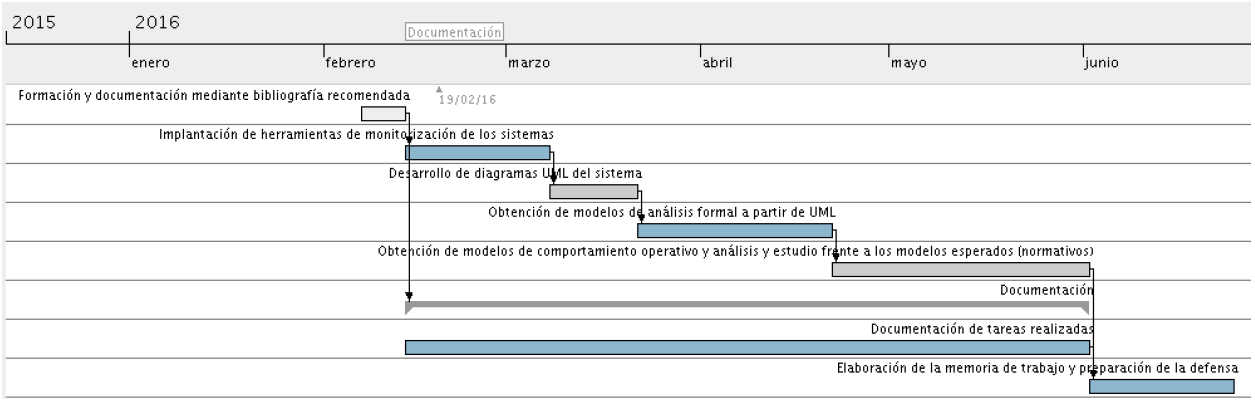


Figura 3: Diagrama de Gantt del proyecto realizado

.

1.7. Estructura del documento

Este documento consta de un total de seis capítulos en los que se describe el trabajo realizado. En el Capítulo 2 se describe el contexto técnico del proyecto, destacando conceptos clave y tecnologías utilizadas; en el Capítulo 3 se analizan los principales avances en minería de procesos y en concreto en la minería de procesos orientada a la seguridad. La descripción en detalle de la aproximación propuesta se encuentra en el Capítulo 4, mientras que los experimentos para validar la propuesta con los sistemas analizados se presentan en el Capítulo 5. Por último, en el Capítulo 6 se presentan las conclusiones y valoraciones generales sobre el proyecto.

Se incluyen también cuatro anexos que proporcionan: una introducción a la minería de procesos,

una descripción de la herramienta utilizada en la conversión entre modelos, una descripción de los diagramas UML realizados para el principal sistema analizado, y la planificación temporal del proyecto (incluyendo una estimación del coste del mismo).

2. Contexto técnico

En esta sección, se describen varios conceptos acerca de las tecnologías y herramientas utilizadas en el desarrollo de este proyecto. En primer lugar, se procede a explicar algunos conceptos básicos para la comprensión de la aproximación desarrollada en el proyecto, teniendo en cuenta el contexto en el que se ha trabajado. Posteriormente, se procede a describir las tecnologías utilizadas y el uso específico dado a cada una de ellas en el proyecto.

2.1. Disciplinas

A continuación, se procede a describir conceptos y disciplinas clave para el entendimiento del contexto en el que se desarrolla el proyecto presentado. En primer lugar se introducen conceptos claves sobre la minería de procesos brevemente (para mayor profundidad, consúltese el Anexo A). A continuación, se introducen conceptos clave sobre las transformaciones de modelos UML en Redes de Petri.

2.1.1. Minería de procesos

La minería de procesos [1], se establece sobre dos pilares clave: la minería de datos y el modelado de procesos de negocio y su análisis.

La *minería de datos* se define en [7] como "el análisis de (habitualmente grandes) conjuntos de datos para la búsqueda de relaciones insospechadas y para resumir los datos mediante vías novedosas que son entendibles y útiles para el dueño de los datos". Los datos de entrada son tradicionalmente proporcionados en forma de "tablas" y la salida puede variar entre reglas, estructuras arbóreas, grafos, ecuaciones, patrones, *clusters*, etc. La minería de datos, constituye uno de los pilares clave y fundamentales de la minería de procesos, ya que algunas técnicas de la minería de procesos están construidas sobre estas técnicas, y la minería de datos puede ser útil para evaluar resultados de este tipo de minería. Por último, la gran cantidad de datos disponibles y su tamaño, hacen de la minería de datos un campo muy popular y útil a la hora de tratar con datos y obtener información de utilidad sobre los mismos.

Por otra parte, *el modelado de procesos y análisis de procesos de negocio*, juega un papel clave en la gestión de operaciones, y en particular en la investigación sobre operaciones, una rama de la ciencia de gestión que depende en gran medida del modelado. Los modelos son utilizados para razonar acerca de los procesos (rediseño) y para la toma de decisiones "dentro de los procesos" (planificación y control). Por último, un modelo de procesos puede ser usado para analizar responsabilidades, analizar el cumplimiento de ciertas acciones, predecir el rendimiento mediante la simulación y otras tareas varias como por ejemplo el análisis de amenazas (tema tratado en este proyecto).

La minería de procesos permite el análisis de procesos de negocios, basándose en los *logs* de eventos producidos por el sistema a analizar. En más detalle, la minería de procesos utiliza algoritmos

especializados correspondientes a la minería de datos para extraer conocimiento sobre los eventos a partir de los *logs* que produce el sistema a analizar; y tiene como objetivo el análisis y entendimiento de los procesos. La minería de procesos provee técnicas y herramientas para: 1) el descubrimiento de procesos (*process discovery*), 2) la realización de chequeo o comprobaciones de conformidad (*conformance*), y 3) la mejora de los procesos (*enhancement*) partiendo de la base de los *logs* de eventos. Estas tareas propias de la minería de procesos se reflejan en la Figura 4.

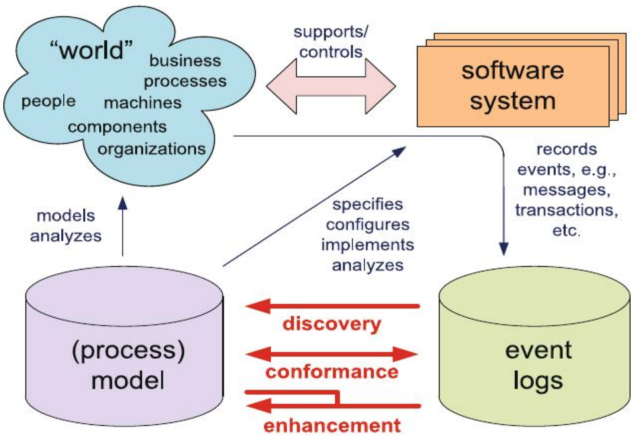


Figura 4: Tres tipos de minería de procesos: *discovery, conformance* y *enhancement* [1].

En este proyecto se ha empleado la herramienta ProM (descrita en la sección 2.2) para llevar a cabo tareas de descubrimiento de procesos y comprobaciones/chequeos de conformidad.

2.1.2. Ingeniería del software dirigida por modelos

La Ingeniería del Software dirigida por modelos [8], en inglés *Model Driven Software Engineering* (MDSE), es una metodología de desarrollo de software que se centra en la creación y explotación de modelos. Existen diferentes propuestas en el contexto de MDSE [9, 10]. No obstante, en todas ellas se tiene como objetivo principal, la utilización de los modelos a lo largo de todo el ciclo de vida del software, no solo para documentar y especificar los componentes software, sino también para generar de forma automática el código de los mismos. A modo de ejemplo, según el enfoque *Model Driven Software Design* (MDSD) [9] el 60-80 % del código se genera de forma automática a partir de los modelos; mientras que según el enfoque DSM (*"Domain-Specific Modeling"*) [10] la generación del código es total, y mediante el enfoque MDA (*"Model-driven Architecture"*) [8] esta generación también es parcial.

Además del código, la MDSE contempla también la transformación de modelos expresados en un lenguaje en modelos expresados en otros lenguajes. Por ejemplo, las transformaciones "modelo-a-modelo" de UML a modelos formales que sirven, en las etapas de verificación/validación, para asegurar que el sistema diseñado/implementado cumple con los requisitos funcionales y no funcionales (prestaciones, fiabilidad, seguridad, etc.) requeridos.

En este tipo de metodología, se le atribuye a los modelos el papel principal de todo el proceso,

frente a las propuestas tradicionales (basadas en componentes tecnológicos) [4]. Por otra parte, este tipo de metodologías persigue elevar el nivel de abstracción en el desarrollo de software, convirtiendo a los modelos y a las transformaciones entre ellos en los principales artefactos de todas las fases del proceso de desarrollo de software [11]. Además se considera que estas aproximaciones aumentan la productividad mediante la maximización de la compatibilidad entre los sistemas, la simplificación del proceso de diseño, y la promoción de la comunicación entre los individuos que trabajan en el sistema.

En conclusión, la MDSE hace referencia al uso sistemático de modelos, como los elementos principales en la ingeniería de software, durante el ciclo de vida completo.

En este proyecto se ha empleado MDSE, y en concreto transformaciones de modelos UML a modelos de Redes de Petri para la obtención de modelos normativos de los sistemas de información Web a analizar.

Lenguaje de modelado UML

UML (Lenguaje Unificado de Modelado)[3], es el lenguaje de modelado de sistemas software más conocido y utilizado en la actualidad. Además el *Object Management Group*[4] (OMG) lo ha promovido y convertido en estándar porque abre una vía para la visualización del diseño de un sistema. Este lenguaje de modelado consta de diferentes diagramas para realizar los modelos del sistema software desde diferentes puntos de vista. Fundamentalmente, existen dos tipos de diagramas: *estructurales o estáticos* y de *comportamiento o dinámicos*. Los diagramas estáticos (diagrama de clases, componentes, despliegue, etc.) describen la estructura de un determinado sistema; mientras que los diagramas dinámicos (diagramas de casos de uso, secuencia, actividad, etc.) describen el comportamiento del sistema, es decir, las funcionalidades del mismo.

En resumen, UML es un lenguaje estándar gráfico para visualizar, especificar, construir y documentar un sistema que permite describir todos los comportamientos y estructuras de un sistema dado, incluyendo aspectos conceptuales.

En el caso de este proyecto, se han utilizado fundamentalmente diagramas UML de actividad para realizar su transformación "modelo-a-modelo" a Redes de Petri. No obstante, se han usado también otros diagramas, tanto estáticos como dinámicos, como por ejemplo diagramas de clase, de secuencia y de casos de uso para comprender los sistemas de información Web analizados en profundidad y construir modelos normativos de mayor calidad.

Lenguaje de modelado formal de Red de Petri

Las Redes de Petri [2] son un lenguaje formal de modelado, introducido por C.A. Petri en su

[3]Especificación de UML del grupo OMG: http://www.omg.org/spec/UML/
[4]Object Management Group: http://www.omg.org/

tesis doctoral en 1962. Las Redes de Petri permiten modelar una gran variedad de sistemas (no sólo sistemas software), caracterizados por situaciones de concurrencia y/o conflictos entre actividades o eventos. Aunque la notación gráfica es intuitiva y simple (una Red de Petri, es un grafo bipartito consistente de "lugares" y transiciones), las Redes de Petri son ejecutables y permiten múltiples técnicas de análisis que no son triviales.

La estructura de la red es estática (como se puede observar en el ejemplo de Red de Petri de la Figura 5). Sin embargo, su comportamiento es gobernado por las reglas de activación y disparo de las transiciones dinámicamente. De esta forma, una transición está activada si tiene un número suficiente de marcas en sus lugares de entrada (al menos tantas como las multiplicidades asociadas a los arcos de entrada correspondientes). Una transición activada puede dispararse y su disparo provoca un cambio de estado de la red. El nuevo marcado (o estado de la red) se obtiene restando de los lugares de entrada de la transición, tantas marcas como las multiplicidades de los arcos de entrada correspondientes, y añadiendo a los lugares de salida de la transición tantas marcas como las multiplicidades de los arcos de salida correspondientes indiquen.

Por último, cabe destacar que para las Redes de Petri marcadas se puede establecer un grafo de alcanzabilidad (que es un sistema de transiciones).

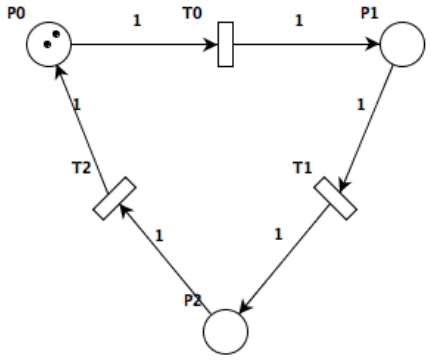


Figura 5: Ejemplo de modelo formal de Red de Petri.

Tal y como se ha comentado previamente, las Redes se Petri se usan en este proyecto para la construcción de modelos normativos y de operativos de los sistemas de información Web a analizar.

2.2. Tecnologías utilizadas

En esta sub-sección, se procede a describir todas las tecnologías usadas en el desarrollo del proyecto, así como cual ha sido su uso dentro de él y para que objetivos se ha utilizado cada una. En primer lugar se describen las tecnologías y herramientas usadas para la monitorización y control de los sistemas analizados. A continuación se describen las herramientas usadas en el preprocesado de los *logs* obtenidos en la monitorización, y la definición de los modelos normativos de los sistemas mediante UML. Por último, se presentan las herramientas para realizar las transformaciones de modelos UML a modelos de análisis formal, y las herramientas para la obtención, análisis y estudio

de los modelos de comportamiento operativo de los sistemas a analizar.

2.2.1. Zabbix

Zabbix [12] es un software empresarial (de código abierto) de monitorización para Redes y aplicaciones. Esta diseñado para monitorizar y "rastrear" el estado de multitud de servicios de red, servidores y otro hardware de red. Entre sus opciones de monitorización disponibles, caben destacar comprobaciones simples de disponibilidad de máquinas en red, monitorización mediante "agentes" que recogen información general del sistema, y otras alternativas más personalizadas para comprobaciones usando mecanismos más completos.

En el proyecto, Zabbix se ha utilizado como herramienta de monitorización para los sistemas tratados, para comprobar su estado y otras características básicas. Por otro lado, también se ha usado para la gestión de mitigación y/o anulación de amenazas o posibles ataques identificados durante los experimentos realizados. Por último, el uso de Zabbix se ha complementado con el agente de monitorización Zorka[5], el cual permite obtener información más detallada sobre servidores Web y otros tipos de peticiones y respuestas vía Web.

2.2.2. Wireshark

Wireshark [13], es un analizador de protocolos utilizado para realizar análisis y solucionar problemas en Redes de comunicaciones, y para desarrollar software y protocolos. En el proyecto, Wireshark se ha considerado como herramienta alternativa para generar los *logs* de eventos HTTP, necesarios para algunos pasos del método propuesto, cuando no se disponga de estos *logs* (por ejemplo, cuando no se tenga acceso al servidor Web o no disponga de un sistema de *logs*).

2.2.3. Sistema de *logs* de Apache Tomcat

Apache Tomcat [14], un servidor Web (de código abierto) conforme a las especificaciones de Java EE [15]. El funcionamiento de este servidor ha sido analizado y estudiado en profundidad porque los sistemas de información Web usados para validar el método de trabajo propuesto en este TFG. se encuentran instalados en este servidor. Se ha utilizado el servidor Web y su sistema de *logs* para las fases de monitorización y control del sistema analizado.

2.2.4. Papyrus (Eclipse)

Papyrus [16] (incluido en el proyecto Eclipse) es una herramienta utilizada para el desarrollo de diagramas UML (aunque también da soporte a otros lenguajes). Esta herramienta es extensible y simple, y consta de una gran variedad de *plugins*. Se ha empleado para crear los diagramas de los sistemas de información Web analizados.

[5]Zorka Monitoring Agent: http://zorka.io/

A pesar de que con esta herramienta se crearon numerosos tipos de diagramas UML que facilitaron la comprensión del funcionamiento del sistema de información Web analizado, debido a las restricciones posteriores de las transformaciones entre modelos no se emplearon todos ellos para la construcción de modelos normativos.

2.2.5. DICE Simulation (Eclipse)

La herramienta/*plugin* DICE Simulation [17] (extensión del proyecto Eclipse, y parte clave del proyecto DICE [18]), es una herramienta para la transformación de diagramas de actividad (*Activity Diagrams* –AD-, en inglés) en modelos formales de Redes de Petri, que además permite realizar simulaciones sobre el modelo formal obtenido. En el proyecto, se ha utilizado para la transformación entre modelos, en concreto, para la transformación de los diagramas de actividad UML a modelos formales de Redes de Petri[6].

2.2.6. ePNK (Eclipse)

La herramienta/*plugin* ePNK [19] (extensión del proyecto Eclipse), permite la visualización, creación y edición manual de forma simple de Redes de Petri. En el proyecto se ha utilizado para la visualización de los modelos formales obtenidos en la fase de conversión, y su posterior edición para mejorar la visualización o eliminar algunos errores, provocados por la herramienta de transformación entre modelos (DICE Simulation).

2.2.7. Python

Python [20], es un lenguaje de *scripting* interpretado que ha sido utilizado para el desarrollo de las herramientas auxiliares necesarias para el pre-procesado de los *logs* y su transformación a formato *Comma-Separated Values* (CSV). Esta tarea, constituye un paso clave para la realización de minería de procesos sobre los eventos correspondientes al comportamiento operativo de los sistemas analizados. Se ha escogido este lenguaje debido a la existencia de numerosas librerías que facilitaban las tareas que se requerían, su simplicidad y por ser un lenguaje multiplataforma.

2.2.8. The Process Mining Toolkit (ProM)

The Process Mining Toolkit (ProM) [21], es una herramienta para la realización de todo tipo de tareas relacionadas con la minería de procesos en general. ProM consta de una gran variedad de *plugins* para la realización de todo tipo de tareas de minería de procesos en diferentes contextos. Esta herramienta, ha sido de gran utilidad para todas las tareas relacionadas con la minería de procesos en el proyecto, en concreto para las tareas de análisis, estudio y comparación entre modelos normativos y operativos.

[6]Más información en el anexo B.

3. Estado del Arte

En esta sección se analizan trabajos relacionados con la temática tratada en este proyecto: la minería de procesos, la ingeniería de software dirigida por modelos, y la minería de datos.

3.1. Minería de procesos

Aunque las tareas basadas en la minería de procesos se han empleado principalmente en el contexto de los procesos de negocio, trabajos recientes aplican esta disciplina a otros dominios diferentes, como por ejemplo los sistemas software (el caso de este proyecto), en concreto, para aplicaciones software intensivas en datos.

En el trabajo [22] se describe una aproximación para el descubrimiento de procesos (*process discovery*) para la extracción de modelos de referencia de un sistema de seguridad marítimo desde los *logs* del sistema.

Por otra parte, la utilización de técnicas de minería de procesos para el análisis de aplicaciones software intensivas en datos, supone hoy en día un reto (debido al gran volumen de datos a manejar). Existen algunas aproximaciones para tratar con *logs* de gran tamaño: como por ejemplo, el trabajo [23]. Este trabajo propone técnicas de descubrimiento de modelos de procesos basadas en la tecnología *"Hadoop MapReduce"* [24]. Otra propuesta similar está disponible en [25], donde se describe una aproximación sistemática para la evaluación del rendimiento de un sistema de gestión de operaciones en puertos marítimos. Esta aproximación emplea minería de procesos y técnicas de la ingeniería del software dirigida por modelos (MDSE).

A continuación, en la siguiente sub-sección, se revisan las contribuciones que emplean la minería de procesos en el contexto de la seguridad de sistemas de información y software en general.

3.2. Minería de procesos para la seguridad

Este proyecto, se sitúa en el contexto de la minería de procesos orientada a la seguridad. En la actualidad, existen varios proyectos relacionados con la minería de procesos en el ámbito de la seguridad en general. Por ello se está avanzando en este tema de investigación.

Entre los trabajos desarrollados en este ámbito, caben destacar el trabajo [26], donde se propone una aproximación para el control de fugas de información en procesos de negocio, proporcionando teoría, experimentación (datos) y herramientas para la gestión de la misma (que puede ser utilizada en otros contextos). Por otra parte, la contribución [27] está más relacionada con el tema tratado en este proyecto y refleja la importancia, y potencial, de proyectos como el desarrollado en este trabajo. En particular, en [27] se propone el empleo/uso de las técnicas de minería de procesos para la auditoría de seguridad de los procesos de negocio. En concreto, se utiliza el descubrimiento de procesos (*process discovery*) para hallar, desde los *logs*, un modelo mediante el cual realizar una

auditoría de seguridad. A su vez, las contribuciones [28] y [29], abordan la detección de anomalías en trazas en *logs* de sistemas de información, proporcionando algoritmos para la detección de estas "desviaciones".

Respecto a la disponibilidad de herramientas para la realización de tareas de este tipo, en la actualidad existen varias. En [30] se propone una herramienta orientada a la seguridad para analizar eventos de *logs* procedentes de diversos procesos.

Una vez analizado el estado del arte relacionado con el proyecto que se aborda basándose en él, se propone una aproximación para la mejora de la seguridad de los sistemas de información Web.

4. Aproximación propuesta

La aproximación propuesta en este proyecto, tiene como objetivo mejorar la seguridad de los sistemas de información Web, mediante la minería de procesos. En concreto, se busca proponer un método para la gestión, detección y monitorización (incluyendo mitigación o anulación) de las *Amenazas Persistentes Avanzadas* (APT).

Una APT es un ataque multi-etapa donde las etapas son las siguientes: (1) reconocimiento, a través de ingeniería social; (2) obtención del acceso al sistema "víctima"; (3.a) exploración, expansión del acceso y control; (3.b) recogida y extracción de la información; y (3.c) encubrimiento de las trazas. Destacar que las etapas 3a, 3b y 3c son concurrentes, y se materializan una vez que la intrusión ha tenido éxito. Por ello, estas APT requieren un alto grado de cobertura durante un largo periodo de tiempo.

La aproximación propuesta en este trabajo se centra en la etapa de la obtención del acceso al sistema víctima (etapa 2), además de las etapas 3a y 3b, cuando se trabaja con sistemas de información Web. Esta aproximación (esquematizada en la Figura 6) plantea una serie de cuatro fases semi-automáticas: monitorización, especificación UML del comportamiento conocido del sistema, transformación de modelos y minería. Se parte de una definición actualizada de las especificaciones UML del comportamiento conocido o esperado del sistema. A continuación, a partir de las especificaciones UML y los *logs* que incluyen trazas de ejecución del sistema, se hallan las posibles trazas de ataques/amenazas contra el sistema analizado.

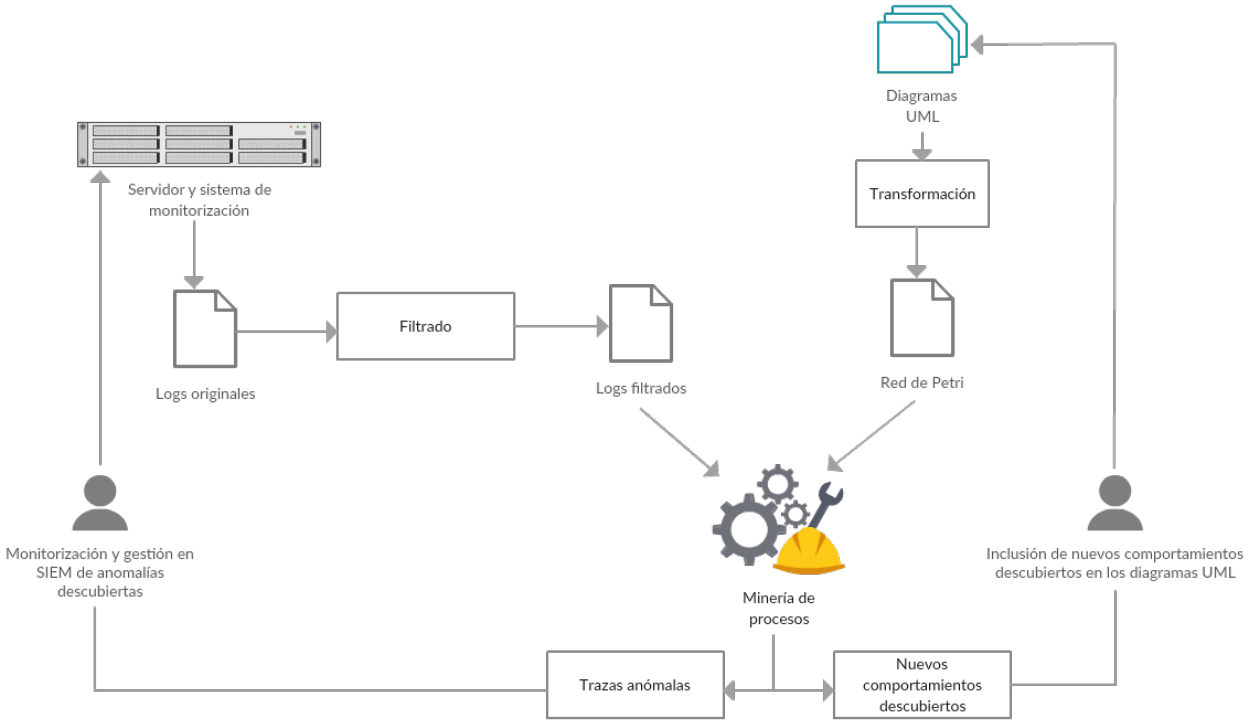


Figura 6: Esquema global de la aproximación planteada.

A continuación, se procede a explicar en detalle las cuatro fases a seguir para la aplicación de la aproximación propuesta.

4.1. Monitorización y control de los sistemas analizados

En primer lugar, la aproximación propuesta, establece la monitorización y control de los sistemas que vayan a ser analizados. Los sistemas encargados de estas tareas de monitorización y control, deben de proporcionar un conjunto de *logs* que contengan las trazas de ejecución del sistema (en particular, qué acciones se están ejecutando en el sistema y cómo se está utilizando). Estos *logs* constituyen los *datos en bruto* de entrada para la obtención de un modelo operativo del sistema mediante el uso de técnicas de descubrimiento de procesos y, en la mayoría de casos, deben de ser preprocesados antes de procederse al análisis de sus datos para, entre otros, obtener *logs de eventos* en formato CSV [31] (que es el formato que emplea ProM, la herramienta de minería de procesos usada en este TFG). Además, es necesario tener en cuenta que los *logs* de eventos, en el contexto de la minería de procesos, se caracterizan por tres atributos claves: (1) un identificador de caso, (2) un identificador de evento y (3) una marca temporal. El identificador de caso permite identificar las trazas de ejecución, cada traza incluye uno o más acontecimientos de eventos; el identificador de evento representa un tipo de evento y la marca temporal representa el momento en que acontece un evento.

La marca temporal para un evento, es fácilmente obtenible y habitualmente está disponible en los *logs*, sin embargo, el identificador de caso y de evento no suelen ser proporcionados por los sistemas de monitorización citados. En el caso de peticiones o eventos HTTP, se considera el evento como la petición en sí realizada, pudiendo contar a su vez con datos sobre direcciones IP del usuario, identificadores de sesiones y otros, que pueden ser de utilidad para asignar un identificador de caso a cada evento.

Aunque se disponga del identificador de evento, todavía es necesario el identificador de caso. Por ello, se han planteado dos posibles soluciones para la asignación de identificadores de caso a cada evento: la primera aproximación se basa en el identificador de sesión del usuario; y la segunda se basa en la sesión del usuario en conjunto con el caso de uso (es decir, la funcionalidad) que está ejecutando el sistema. Para cada una de las soluciones se propone una o varias heurísticas, que se describen y analizan a continuación.

4.1.1. Identificador de sesión como identificador de caso

En el caso de contar con identificadores de sesiones en los *logs* obtenidos del comportamiento operativo del sistema, se puede considerar la sesión como identificador de caso (para los eventos de una traza completa). Cada sesión, agrupa los eventos correspondientes a la ejecución de todos los casos de uso realizados por un mismo usuario en un periodo temporal de interacción con el sistema. Si, por otra parte, no se dispusiese de este identificador de sesión, puede realizarse un preprocesado

de los *logs* para la obtención del mismo. Para la obtención de este identificador de sesión, se propone una heurística, similar a la utilizada por "Google Analytics"[7] que permite identificar las sesiones. Esta heurística determina que dos eventos pertenecen a la misma sesión de usuario si:

1. Ambos eventos proceden de la misma IP de origen y no han transcurrido más de 30 minutos entre ambos (tiempo ajustable).
2. Entre ambos eventos (procedentes de una misma IP de origen[8]), no se procede a un cambio de usuario, o no se realiza un proceso de autenticación de usuario.

En cualquier otro caso, se considera que se trata de sesiones diferentes. Estas reglas pueden ser ajustadas para el sistema tratado en concreto, para obtener o identificar de forma exacta una sesión de usuario. No obstante, mediante el uso de esta heurística de sesión, se puede llegar a identificar sesiones con un porcentaje de exactitud muy alto y adecuado para las tareas que se deben realizar. Cabe destacar que será necesario que los *logs* contengan una marca temporal en cada registro y una IP que permitan aplicar las reglas propuestas anteriormente. Para validar la heurística se han realizado pruebas aplicando la misma a diferentes conjuntos de *logs* y comparando los identificadores de caso obtenidos con los identificadores de sesión reales (se disponían de los mismos). Los resultados de comparación muestran que el porcentaje de exactitud de identificación correcta de sesiones es siempre del 100 %[9]. Este porcentaje tan alto y exacto, se debe a que el aspecto clave (que es el tiempo de expiración de la sesión), es conocido (por defecto en Tomcat, 30 minutos).

4.1.2. Identificador de caso de uso como identificador de caso

Los casos de uso del sistema de información a tratar también se han considerado como posibles identificadores de caso. Para ello, también se requiere definir reglas para desarrollar heurísticas que permitan identificar casos de uso dentro de sesiones de usuario identificadas previamente. En Ingeniería del Software, un caso de uso de "nivel usuario" representa una funcionalidad del sistema software e incluye un conjunto de escenarios de interacción (alternativos de éxito y de fracaso) entre el usuario y el sistema, que se llevan a cabo para perseguir el objetivo del usuario (indicado por el nombre del caso de uso).

En este caso, la heurística está estrechamente relacionada con la aplicación Web que se está analizando, ya que los casos de uso son específicos de cada aplicación. Aun así, una buena heurística para identificar casos de uso sobre los *logs* de una aplicación Web, es usar el *"referer"* (página web desde donde se solicita la petición) de los *logs* e intentar establecer políticas de cada caso de uso y secuencias de eventos. Las cadenas de peticiones pertenecerían a casos de uso diferentes, según se establezcan en los diagramas UML de actividad definidos para la aplicación Web que se está analizando. Por lo tanto, de esta manera, se conseguiría identificar casos de uso por sesión de

[7] Google. (2016). *How a session is defined in Analytics.* Accedido el 31 de Mayo, 2016, desde https://support.google.com/analytics/answer/2731565?hl=en

[8] Se considera que se pueden entrelazar eventos de diferentes IP de origen.

[9] Resultados de validación disponibles en: http://sid.cps.unizar.es/PMS/tests.html

usuario. Combinando la heurística anterior con la actual, se consigue el identificador de caso ("case id"), necesario para la tarea de minería de procesos.

Aunque se utilice el campo correspondiente a la cabecera *"referer"*, este puede ser manipulado y no representar la realidad. Por este motivo, se plantean dos heurísticas de casos de uso (con diferentes variaciones), las cuales permiten determinar qué casos de uso se dan en una determinada sesión de usuario. A continuación, se procede a la explicación de estas heurísticas y sus posibles variaciones.

Heurística de puntos de entrada (H1)

Esta heurística, se basa en la hipótesis de que cada caso de uso de la aplicación Web, tiene un punto de entrada bien definido y único; es decir, cada caso de uso de la aplicación comienza por un evento representativo y único, que no puede ocurrir en otros casos de uso (lo cual es bastante común). De esta manera, se consigue identificar con facilidad el inicio de un determinado caso de uso, el cual a su vez representa el fin del caso de uso anterior. Por ello, hasta que no se identifique/ocurra un evento representativo de caso de uso, se mantiene el caso de uso anterior (o se establece el correspondiente si se trata del inicio de una sesión de usuario / interacción con la aplicación Web).

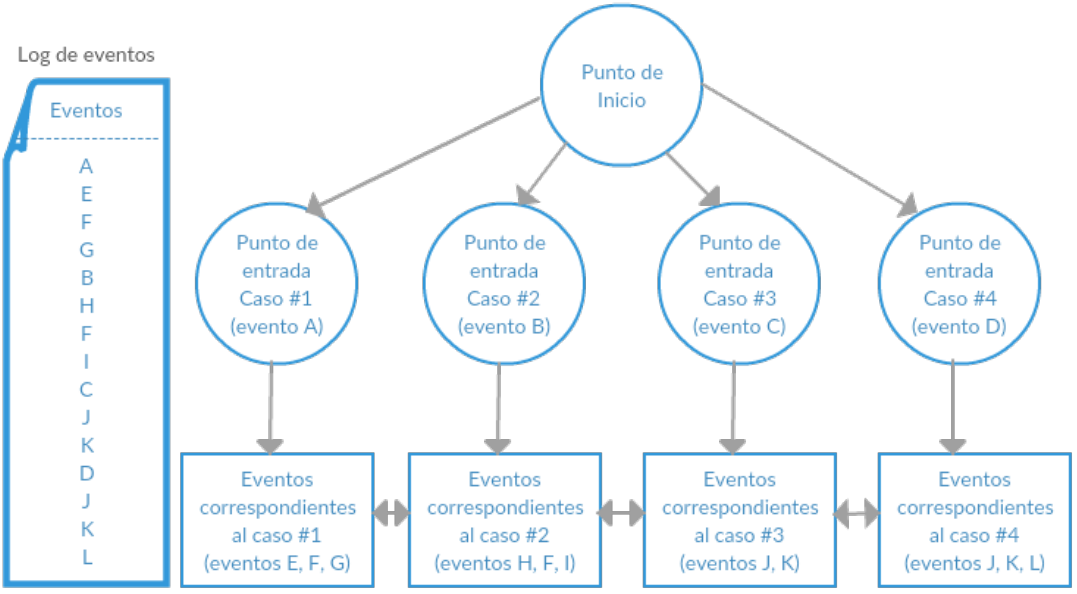


Figura 7: Esquema de la heurística de puntos de entrada (H1) descrita.

La Figura 7 muestra un ejemplo de aplicación de la heurística con cuatro casos de uso y un conjunto de eventos (A-L), ordenados alfabéticamente. La heurística identifica los casos de la siguiente manera:

- Se comienza por el evento A, punto de entrada del caso 1, por lo que este evento y sucesivos

(E, F, G) se etiquetan como correspondientes a este caso de uso (a no ser que alguno de estos casos sea el punto de entrada de otro caso de uso).

- Al llegar al evento B, se comienza a etiquetar el mismo y sucesivos como correspondientes al caso 2. Destacar que aunque se dan algunos eventos compartidos con otros casos de uso, al no tratarse estos de un punto de entrada identificado, siguen siendo etiquetados según el caso de uso actual.
- Por último, este proceso se repite para el caso de uso 3 y 4, cuando se procesan los eventos C y D, respectivamente; identificando un total de cuatro trazas con sus eventos correspondientes (a pesar del aspecto de compartición de eventos en diferentes casos de uso).

Un aspecto de esta heurística, relacionado con la seguridad ante posibles ataques o peticiones erróneas, es la consideración de que un caso de uso puede verse terminado antes de la aparición de otro "punto de entrada". Por lo tanto, se consideran una serie de páginas y peticiones/eventos posibles dentro de un determinado caso de uso. Considerando esta posibilidad, se puede identificar cuando un caso de uso termina de forma "anómala" (p. ej., cuando se realiza un evento que no pertenece a la serie de páginas), antes de que ocurra el evento de entrada a otro caso de uso. Estas "anomalías" se consideran como tales y se registran convenientemente.

Heurística de caminos (H2)

Esta heurística, se basa en la suposición de que en una sesión de usuario, la traza de ejecución correspondiente incluye eventos que pueden pertenecer a varios casos de uso. En cada sesión de usuario se asigna, a cada evento, los posibles casos de uso que se pueden estar ejecutando. Cuando finaliza la sesión, se comprueban las probabilidades de que caso de uso se ha ejecutado, considerando la asociación *evento-caso de uso* para cada evento y el número de apariciones del caso de uso en la traza.

De esta manera, se identifican los caminos más largos de eventos correspondientes a un determinado caso de uso y se le asigna, a ese "camino" de eventos, el caso de uso correspondiente (pudiendo una sesión constar de varios caminos, varios casos de uso).

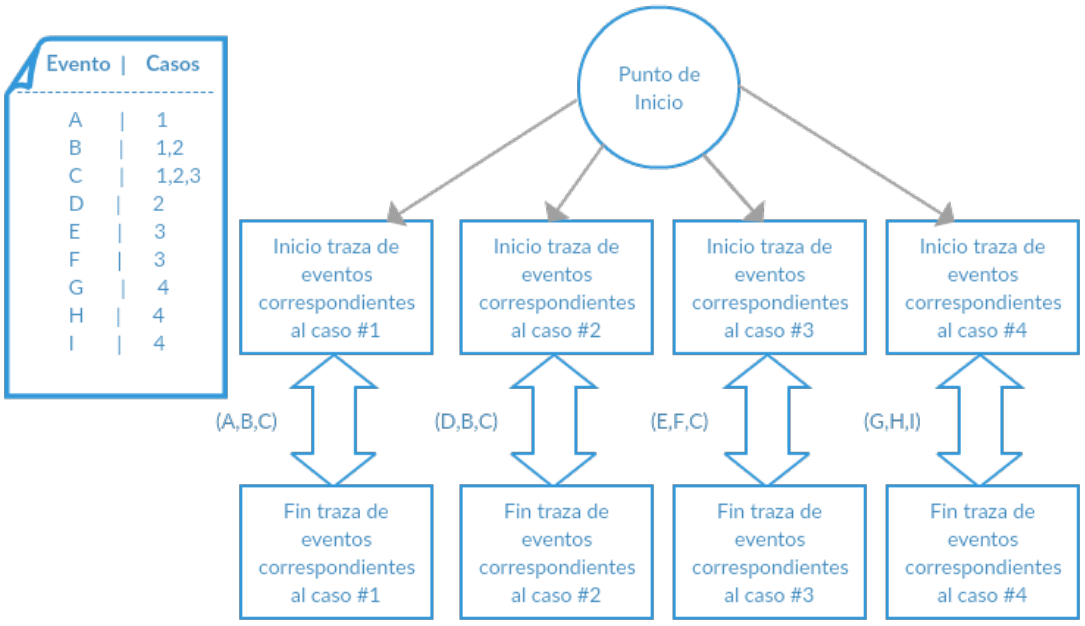


Figura 8: Esquema de la heurística de caminos (H2).

La Figura 8 muestra un ejemplo de aplicación de la heurística a una traza de ejecución correspondiente a una sesión de usuario:

- Se capturan la completitud de eventos de las diferentes trazas por orden, pero se desconoce a qué caso de uso pertenece cada evento, aunque la longitud máxima de la traza (según la figura), es de tres eventos (longitud del "camino" más largo). Este primer paso, permite asignar las etiquetas a cada evento que indican los casos de uso correspondientes.
- Posteriormente, mediante los tres primeros eventos, se comprueba cuál es el "camino más largo", es decir, el caso de uso que se identifique con el mayor número de eventos posibles de los seleccionados. En el primer caso, con los eventos (A,B,C), la heurística los etiquetaría como correspondientes al caso 1, ya que corresponden los tres eventos, por encima del caso 2 aunque corresponden dos de ellos.
- Por último, se itera mediante este procedimiento, identificando los "caminos más largos" y etiquetándolos con el número de caso de uso correspondiente (caso de uso con el que se identifican más eventos).

Considerando los valores de parámetros en cada petición HTTP (además del recurso o página), se consigue una gran exactitud de identificación de casos de uso ejecutados en una determinada sesión de usuario. A su vez, se pueden asignar diferentes pesos para cada evento (si resulta más o menos significativo) para calcular las probabilidades con mayor exactitud y conseguir mejores resultados.

4.1.3. Comparación de heurísticas

La gráfica en la Figura 9 muestra el porcentaje de "acierto" de las heurísticas descritas con anterioridad, en un conjunto de *logs* de pruebas para determinar cual de ellas usar.

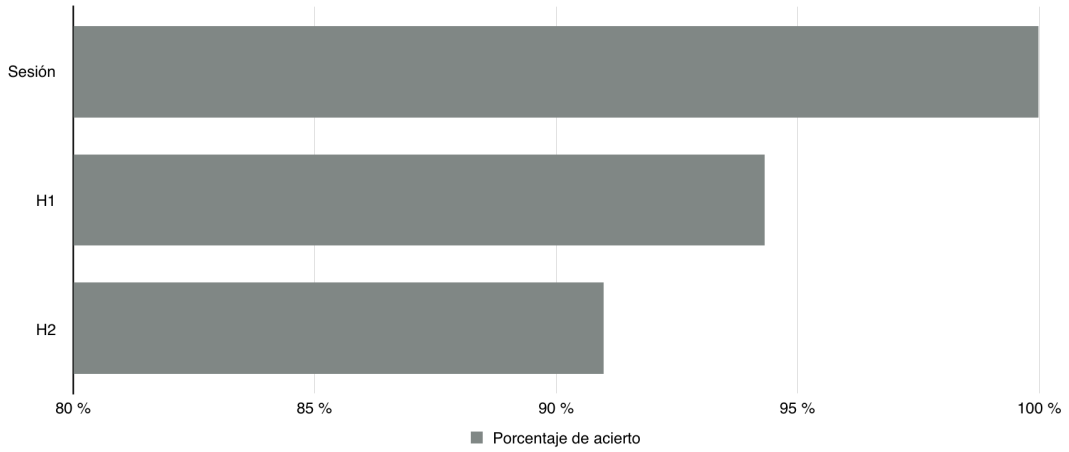


Figura 9: Gráfico de barras comparativo entre heurísticas.

Para más detalles sobre el experimento y las métricas obtenidas ver http://sid.cps.unizar.es/PMS/tests.html. En este caso, se observa que la heurística H1 es más fiable que la heurística H2.

Por otra parte, también se puede considerar el orden en el que las páginas o eventos de un caso de uso deben darse. Esta característica se puede aplicar para detectar con mayor exactitud los casos de uso e identificar posibles trazas de eventos "anómalas". No obstante, mediante la aplicación de minería de procesos, estas trazas pueden ser identificadas y procesadas. Por lo tanto, la consideración del orden de los eventos a la hora de aplicar este tipo de heurísticas en este contexto, no se considera necesaria, aunque si interesante y útil en algunos casos.

Las anteriores heurísticas se pueden ver modificadas y mejoradas observando los *logs* de la aplicación Web y la utilización "real" de la misma (por parte de los usuarios), que puede diferir de la idea inicial supuesta sobre el sistema desarrollado; es decir se pueden identificar patrones de comportamiento validos por parte de los usuarios en la aplicación Web, y se pueden retro-alimentar estas heurísticas propuestas, consiguiendo mejores resultados. Observando estos comportamientos no conocidos a priori y añadiéndolos como casos de uso en la heurística propuesta, se consigue una mayor exactitud y una probabilidad mayor de identificación correcta de casos de uso. Por este motivo, la heurística está sujeta a cambios (adiciones de casos de uso) observados en la utilización de la aplicación Web en un entorno real. Estos casos de uso observados, pueden ir (por ejemplo) desde comportamientos de *bots* de motores de búsqueda, hasta accesos directos proporcionados por diferentes fuentes y no considerados.

4.2. Especificación UML del comportamiento conocido del sistema

En la segunda fase de la aproximación, se define el modelo de comportamiento "conocido" del sistema. Dado que estamos considerando sistemas ya operativos, es posible obtener de forma automática una especificación UML del mismo a partir de su implementación, mediante la ingeniería inversa. Por otra parte, si los modelos ya están disponibles, se pueden actualizar las especificaciones UML de diseño del sistema.

Las especificaciones UML tienen que incluir diagramas de comportamiento, como, por ejemplo, diagramas de actividad o de secuencia, para poder generar de forma automática un modelo (formal) normativo. La elección de elaboración de diagramas de actividad mediante UML y no otro tipo de diagramas, reside en la facilidad de "producción" de los mismos y su fácil modificación y comprensión por la mayoría de personas con conocimientos básicos de ingeniería de software (o ingeniería dirigida por modelos, *MDE*), además de que se puede disponer en un principio de ellos si el sistema ha sido desarrollado según las convenciones sugeridas por la ingeniería del software. En esta aproximación se consideran los diagramas de actividad, ya que son fáciles de entender y producir.

En un principio, los diagramas de actividad deben reflejar el comportamiento del sistema al mismo nivel del que se disponen los *logs* operativos del mismo. Es decir, las acciones incluidas en un diagrama de actividad tienen que tener una correspondencia con los eventos incluidos en las trazas de los *logs*. Cabe destacar, que la especificación UML del comportamiento conocido del sistema esta sujeta a modificaciones futuras, debido a la retroalimentación ofrecida por las siguientes fases de la aproximación. Esto se debe, a que se pueden detectar patrones de comportamiento operativo del sistema "benignos", realizados por los usuarios y no considerados en la fase de diseño del sistema (nuevas tendencias de uso que pueden ser confundidas en primera instancia con patrones de ataques o posibles amenazas).

4.3. Definición de transformaciones necesarias para la construcción de modelos de análisis formal a partir de los diagramas UML anotados

Una vez definida la especificación UML del comportamiento esperado del sistema, se puede proceder a su conversión a un modelo formal, el cual pueda ser usado en tareas correspondientes a la minería de procesos. En la aproximación propuesta, se utiliza la transformación de diagramas UML a redes de Petri implementada en la herramienta "DICE Simulation" [17] (actualmente en fase de desarrollo en el contexto del proyecto Europeo "DICE" [18]).

"DICE Simulation" es capaz de transformar uno o varios diagramas de actividad UML a Redes de Petri (RdP), pudiendo especificar ciertas opciones y pudiendo simular la misma en un entorno en el cual se puede obtener propiedades de rendimiento de la misma. No obstante, esta aproximación se centra solamente en la conversión/transformación, siempre y cuando el modelo RdP resultante cumpla con la especificación PNML [32], la cual permite que dicha red pueda ser utilizada por

herramientas externas. La Figura 10 muestra un ejemplo de la Red de Petri obtenida (abajo) con la herramienta, a partir del diagrama de actividad UML (arriba). Más información sobre la herramienta de conversión, su funcionamiento y uso, puede ser encontrada en el anexo B.

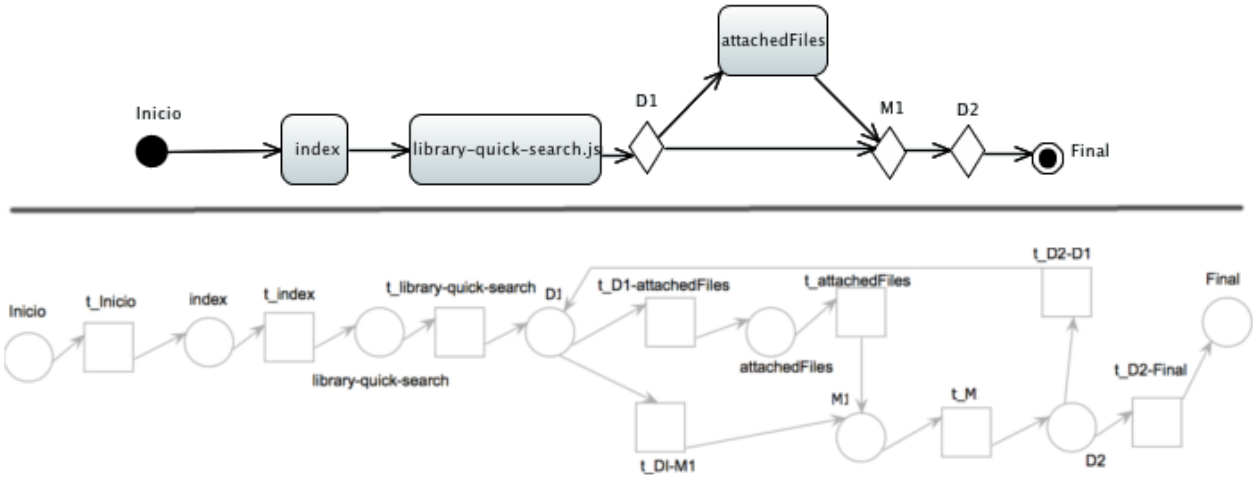


Figura 10: Ejemplo de transformación desde diagrama de actividad UML a Red de Petri.

El fichero PNML, es el que define el modelo formal (según los diagramas UML) y es utilizado en posteriores pasos para el estudio y análisis del comportamiento operativo de los sistemas frente a este modelo normativo.

4.4. Uso de técnicas de minería de procesos para la identificación de desviaciones de comportamiento en el sistema

En la última fase de la aproximación (véase Figura 6), se utilizan técnicas de minería de procesos con el doble objetivo de descubrir comportamientos anómalos (es decir, posibles ataques o amenazas) y nuevos comportamientos no considerados en el modelo normativo. En concreto, las técnicas de minería de procesos que se utilizan en esta fase son: 1) el análisis de conformidad y prestaciones (*"conformance & performance checking"*), y 2) el descubrimiento de procesos (*"process discovery"*).

La primera técnica consiste en reproducir las trazas de los *logs* en el modelo normativo obtenido en la fase anterior de la aproximación. Mediante el análisis de conformidad se identifican las trazas que no encajan en el modelo normativo. Además, con el análisis de prestaciones se obtienen métricas (p. ej., el número de veces que se verifica un evento en la unidad de tiempo o *throughput*) útiles a la hora de estudiar la posible aparición de APT. Por ejemplo, un *throughput* de los eventos de un proceso de validación de usuario mayor que el de los eventos del proceso de aceptación del mismo, puede indicar un posible ataque de credenciales contra el sistema de autenticación.

La segunda técnica permite obtener de forma automática un modelo formal (p. ej., Redes de Petri) operativo a partir de los *logs*. Las desviaciones de comportamiento se identifican en este caso mediante un análisis comparativo entre el modelo operativo y el modelo normativo.

4.4.1. Gestión y control de las amenazas o patrones de ataques detectados

Por último, y una vez analizados y estudiados los posibles patrones, estos deben ser mitigados o anulados (o simplemente monitorizados según el caso). Para ello, se pueden establecer reglas en sistemas de gestión de eventos y seguridad de la información (Security Information and Event Management, SIEM), mediante las cuales se consiga la detección y mitigación y/o anulación de posibles amenazas persistentes avanzadas, con el fin de protegerse ante las mismas antes de que se materialicen y afecten (provocando daños) al sistema. Estas reglas, se basan en los resultados obtenidos en el análisis y comparación de diferencias entre modelos normativo y operativo del sistema.

Cabe destacar, que se puede hacer uso de los SIEM o de los servidores Web utilizados en el sistema, en el caso de que estos tengan las capacidades necesarias para identificar, mitigar y/o anular estas amenazas o ataques.

5. Experimentación de la aproximación propuesta

Respecto a la experimentación con la aproximación propuesta, descrita en el capítulo 4, se ha realizado su aplicación, evaluación y validación en dos sistemas: el correspondiente al "Grupo de I+D en Computación Distribuida (DisCo)" [5] y el correspondiente a la biblioteca digital del "Grupo de Sistemas de Información Distribuidos (SID)" [6].

5.1. Introducción sobre los sistemas analizados

Respecto al primer sistema, cabe destacar que al ser de naturaleza completamente estática (solo dispone de páginas HTML estáticas), la experimentación se ha basado únicamente en el análisis de número de peticiones a diferentes páginas HTML y la inspección de peticiones anómalas (que no se corresponden con el sistema), como por ejemplo una petición a una página inexistente. La aproximación propuesta, en su aplicación para este sistema, no produce resultados de interés, puesto que el sistema no está sujeto a amenazas persistentes avanzadas o ataques complejos. Por otra parte, es posible comprobar (mediante la aproximación descrita) indicios de ataques de denegación de servicio, y acciones de ataque automatizadas (sin éxito, por la naturaleza del sistema).

Debido a la naturaleza del sistema "DisCo", la mayor parte de experimentación de la aproximación propuesta, recae sobre el segundo sistema de información Web de la biblioteca digital del grupo SID. La flexibilidad y dinamismo, características de este sistema, lo hacen óptimo para la aplicación de la aproximación propuesta. Esta flexibilidad y dinamismo se debe a que se tratan de páginas Web dinámicas, con un amplio abanico de peticiones posibles así como de respuestas, con sus correspondientes puntos débiles y su mayor exposición a ataques o amenazas.

5.2. Arquitectura del principal sistema analizado

La arquitectura del sistema "Biblioteca Digital del grupo SID" (representada en la Figura 11) se concreta en un servidor Apache Tomcat, desarrollado en Java (JEE). Esta es una arquitectura de tres capas donde las capas/niveles de presentación y lógica están desarrolladas en el servidor 1 (Apache Tomcat) y la capa/nivel de datos en el servidor 2 (*BBDD MySQL*). La *BBDD MySQL* almacena toda la información correspondiente a autores, publicaciones, fechas de publicación, citaciones y otra información como enlaces a ficheros adjuntos, imágenes, presentaciones, etc. No obstante, también se dispone de un sistema de ficheros en el que se organizan, en diferentes directorios, archivos adjuntos correspondientes a cada publicación.

El sistema tiene un mecanismo de autenticación sencillo (gestionado por el mismo servidor Tomcat, sin cifrar), para tratar de evitar que usuarios no autorizados realicen acciones como la inserción, edición o borrado de publicaciones. A su vez, el sistema tiene funcionalidades y páginas Web accesibles a todo el público general, como son la búsqueda de información relativa a publicaciones y la citación de las mismas.

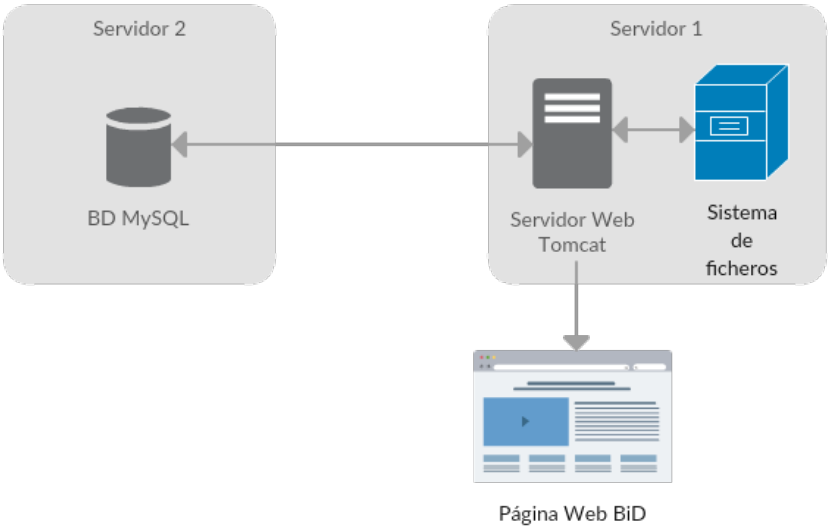


Figura 11: Esquema de la arquitectura del principal sistema analizado.

Los principales componentes de interés para el análisis del sistema, son las páginas y documentos (estáticos y dinámicos) que lo componen. Esto se debe a que los *logs* disponibles del comportamiento operativo del sistema, incluyen las peticiones a nivel HTTP realizadas a estos. A partir de estos componentes y la interacción entre los usuarios y el sistema, se realizarán los modelos y el análisis del comportamiento operativo para la detección de eventos anómalos.

5.3. Pruebas realizadas

Para este sistema en concreto, se han seguido todas las fases de la aproximación propuesta. Su monitorización y control se ha realizado mediante el sistema de *logs* de Apache Tomcat y Zabbix (con posterior preprocesado con una herramienta auxiliar propia desarrollada en Python[10]). La especificación UML del sistema se ha creado mediante el plugin de Eclipse Papyrus. La obtención automática del modelo normativo en Redes de Petri a partir de la especificación UML, se ha obtenido mediante el plugin de Eclipse "DICE Simulation"; y por último, la obtención de modelos de comportamiento operativo del sistema y su análisis frente al modelo normativo, mediante el *framework* ProM.

Para el análisis de este sistema, se disponen de un total de 4 años de *logs* de eventos acumulados (desde el inicio de su funcionamiento). Estos *logs* del mismo, contienen trazas no relevantes para el análisis correspondiente (proceso de desarrollo y puesta en producción). Estas trazas se han filtrado para que no influyesen en las pruebas realizadas. Debido a que no se disponía de una especificación UML del sistema, se crearon todos los diagramas UML correspondientes al sistema, a partir del código fuente (también disponible) y realizando un proceso de ingeniería inversa. En concreto, se han definido diagramas de casos de uso, de clase, de secuencia y de actividad[11]. Además, a partir

[10]Herramienta disponible en: https://github.com/piraces/Tomcat-Logs-Utilities
[11]Disponibles en: http://sid.cps.unizar.es/PMS/resources.html

de los diagramas de actividad se ha generado el modelo normativo en Redes de Petri[12]. Los demás diagramas, no han sido transformados pero han sido de gran utilidad para adquirir un mayor conocimiento del sistema y de los *logs* generados por el mismo.

5.3.1. Análisis de conformidad y rendimiento

Una vez obtenido el modelo formal normativo del sistema (modelo en Redes de Petri), se pudieron realizar pruebas de análisis de conformidad y rendimiento (*"conformance & performance checking"*), para determinar si los eventos del sistema recogidos en los *logs*, pueden ser reproducidos por el modelo.

Mediante la técnica de análisis de conformidad, fue posible evaluar y validar los diferentes diagramas de actividad UML desarrollados, ya que esta técnica permite obtener un porcentaje de *fitness* o exactitud. Este porcentaje mide la similitud entre el comportamiento operativo del sistema y el normativo. Además, también permite observar las trazas en los *logs* que se corresponden con secuencias de disparo de las transiciones en el modelo en Redes de Petri y las que no (identificando posibles trazas de eventos de ataques/amenazas)[13]. De hecho, se analizaron en detalle las trazas no coincidentes con el modelo normativo, para determinar si se correspondían a un ataque o amenaza, o si se trataba de un comportamiento benigno que no se ha considerado en el desarrollo del sistema (hábitos nuevos de usuarios, atajos encontrados por usuarios no peligrosos, y otro tipo de trazas habituales no consideradas en la fase del diseño del sistema).

5.3.2. Descubrimiento de procesos

Aunque mediante la técnica anterior, es posible detectar las trazas anómalas en el comportamiento operativo del sistema, se decidió también utilizar los mecanismos de descubrimiento de procesos (*"process discovery"*) de ProM con los *logs* de la Biblioteca Digital del grupo SID. De hecho, mediante esta técnica se compararon los modelos de comportamiento operativo y normativo del sistema (de forma manual).

Existen varias técnicas de descubrimiento de procesos que producen modelos formales diferentes, como RdP o *"Fuzzy Causal Nets"*. En los experimentos se emplearon las dos y cabe destacar que con el uso de *"Fuzzy Causal Nets"*, se obtuvieron resultados más fáciles de interpretar gracias a que con el uso de la herramienta ProM, se pueden llevar a cabo animaciones temporales de la red causal generada, visualizando todo el flujo de eventos, así como trazas concretas. A modo de ejemplo, en la Figura 12, se puede observar un ataque real realizado contra el sistema. En ese ataque se intentaba buscar (de forma automática, como indican los tiempos entre peticiones) un fichero vulnerable correspondiente al *script* "Uploadify"[14]. Actualmente se conocen varias vulnerabilidades de carácter grave de este *script* como por ejemplo la ejecución arbitraria de código o la subida de

[12]Recursos disponibles en: http://sid.cps.unizar.es/PMS/resources.html
[13]Ejemplo de trazas de ataques identificadas: http://sid.cps.unizar.es/PMS/resources/images/trazasAtaques.png
[14]Uploadify (página oficial): http://www.uploadify.com/

ficheros arbitrarios[15].

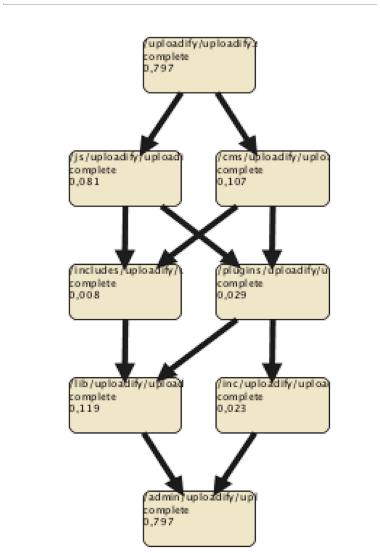


Figura 12: Ejemplo de traza de ataque representada mediante un modelo formal *"Fuzzy Causal Net"*.

En resumen, mediante el descubrimiento de procesos (*"process discovery"*), se consigue una comparación entre los dos modelos formales, que permite identificar los patrones representados en el modelo operativo que no están presentes en el modelo normativo. En los experimentos realizados, en ocasiones las trazas anómalas son fácilmente identificables ya que aparecen representadas como redes separadas (en lugar de una única red). Sin embargo, los patrones de ataque más complejos se dan en la red que representa el comportamiento principal del sistema. No obstante, también se pueden identificar fácilmente considerando los eventos propios del sistema, reflejados en el modelo formal normativo.

5.4. Resultados

Con el método propuesto, se ha conseguido obtener varias trazas de ataques simples contra el sistema de información Web analizado, así como información sobre amenazas potenciales y otros comportamientos anómalos. Gracias a este resultado, se han conseguido bloquear todo tipo de ataques identificados en las fases de análisis y estudio de los modelos del sistema. En concreto, se ha conseguido evitar "cuellos de botella" en ciertos procesos del sistema, y se han eliminado: *scripts* maliciosos en busca de vulnerabilidades; ataques de fuerza bruta (contra sistemas de autenticación de usuario) y amenazas potencialmente peligrosas contra el sistema de inserción y edición de publicaciones. Además, se ha evitado y reducido el uso del sistema por parte de usuarios maliciosos, consiguiendo la identificación de los mismos y aplicando reglas de bloqueo y reporte correspondientes.

[15]Lista de vulnerabilidades reconocidas oficialmente del *script*: https://web.nvd.nist.gov/view/vuln/search-results?query=uploadify&search_type=all&cves=on

Respecto a las llamadas Amenazas Persistentes Avanzadas (APT), se han llegado a detectar (y bloquear satisfactoriamente) trazas correspondientes a las mismas como el intento de edición de publicaciones por parte de usuarios, que no son autores ni tienen permiso para realización de dichas ediciones. A su vez, muchos intentos de inserción y ataques contra el sistema de lectura/escritura de la BBDD y el sistema de ficheros de publicaciones, han sido detectados y controlados antes de su ejecución.

Por otra parte, se detectaron gran cantidad *scripts* dedicados a la búsqueda de paneles de administración y otras páginas correspondientes al CMS WordPress (en busca de puntos de entrada que atacar, sin éxito obviamente). Esto se puede deber a que varios de los investigadores del grupo constan de páginas personales y *blogs* realizados mediante WordPress, lo que puede llevar a los atacantes a pensar que la página principal del grupo también puede basarse en el mismo CMS, e investigarlo para la detección de posibles puntos débiles de entrada. Otro tipo de ataques detectados, son los de búsqueda y ataque contra puntos de subida de ficheros al sistema (páginas de *"upload"*), con el fin de subir ficheros maliciosos propios (como *shells* en PHP), además de para otros CMS concretos (y vulnerabilidades conocidas de los mismos).

Por otro lado, se han descubierto nuevos patrones de comportamiento (benignos) por parte de los usuarios del sistema. Estos nuevos patrones de comportamiento, se corresponden con la visualización y búsqueda de publicaciones (según el autor) en páginas externas, por parte de los investigadores. Esto se debe a que en las páginas Web propias de ciertos investigadores del grupo, estos realizan peticiones directas a los servicios descritos anteriormente, así lo que en un principio se consideraba una posible amenaza, ya que es de gran complejidad realizar una búsqueda o filtrar publicaciones sin utilizar la interfaz correspondiente, una vez analizado y estudiado, se consideró como comportamiento normal en varias páginas[16]. Por ello fue añadido al modelo normativo del sistema (para no considerarlo como un comportamiento anómalo de nuevo).

Por último, se han detectado vulnerabilidades en el código del sistema de información web citado, que estaban siendo aprovechadas por ciertos usuarios maliciosos para "escalar privilegios" y realizar ciertas acciones "no relacionadas con el sistema" en el servidor, así como provocar una gran carga al mismo e intentar provocar su denegación de servicio.

5.5. Gestión de ataques y amenazas

Para la gestión de ataques y amenazas tras los resultados obtenidos (descritos con anterioridad), se ha hecho uso de la herramienta Zabbix, junto con el agente de monitorización Zorka (para obtención de datos específicos del servidor). Con estas herramientas, ha sido posible la gestión, mitigación y anulación de amenazas posibles antes de que se lleguen a materializar. En concreto, mediante el agente de monitorización "especializado" Zorka, se ha obtenido toda la información necesaria del servidor Web (peticiones a nivel HTTP y métricas). Posteriormente a partir de estas métricas, se han definido los *triggers* necesarios en Zabbix para que se activen cuando se den las

[16]Ejemplos: http://sid.cps.unizar.es/AndroidSemantic/publications.html; https://webdiis.unizar.es/~mena/

acciones o eventos anómalos. Cuando estos *triggers* se disparan (ante un evento anómalo dado), se realiza una acción dependiendo de el nivel de gravedad de dicho evento. Esta acción varía y puede ser desde una simple notificación al administrador vía email, hasta el bloqueo permanente por dirección IP.

Por último señalar que la recogida de información por parte de los agentes, se realiza en tiempo real, pudiendo de esta manera también, reaccionar en tiempo real y tomar las acciones pertinentes ante ataques o amenazas en el justo momento en el que estas ocurren.

5.6. Validación

Varias técnicas de validación de los métodos aplicados en las diferentes fases de la aproximación propuesta han sido aplicados, con el fin de comprobar que cumplen con los requisitos establecidos en un comienzo. A continuación, se proceden a describir los procesos de validación realizados en cada fase de la experimentación con la aproximación propuesta[17]:

- En la fase de monitorización y control del sistema, en relación a las diferentes heurísticas propuestas, se han aplicado técnicas para comprobar su tasa de rendimiento o exactitud. En el caso de la heurística para el identificador de sesión, la citada exactitud es siempre del 100 % en todas las pruebas realizadas (con ajustes propios del sistema conocidos). En el caso de las heurísticas de caso de uso, estas no pueden ser validadas de una forma directa, aunque se han aplicado diversos métodos automáticos y manuales para comprobar la exactitud con un conjunto pequeño de *logs*. En esos casos se han obtenido resultados también superiores al 90 % de exactitud en el caso de la heurística de "caminos", y superiores al 94 % en el caso de la heurística de "puntos de entrada".

- En la fase de definición o actualización de los modelos de comportamiento esperado (o normativo) de los sistemas mediante UML y la transformación al correspondiente modelo formal, se han verificado conjuntamente los procesos de *"process discovery"* y *"conformance & performance checking"*. En concreto, se ha empleado una métrica de similitud del comportamiento operativo del sistema (*fitness*), que permite comprobar la validez de los diagramas UML (y las reglas de transformación) y de su correcta transformación (modelo normativo). Cabe destacar que se han ajustado ambos aspectos, hasta conseguir una medida de *fitness* superior al 90 % (considerando los distintos tipos de heurísticas de sesión y de casos de uso), sin ataques. Considerando ataques, estos porcentajes de validación son en el peor caso (heurísticas para casos de uso) del 85 %. En el caso de heurísticas de sesión, el porcentaje sube al 95 %. En la figura 13, se puede observar la gráfica comparativa de porcentajes de validación.

- En la fase de obtención de modelos de comportamiento operativo de los sistemas (mediante *"process discovery"*) y su análisis y estudio frente al modelo normativo esperado, las validaciones se han realizado mediante análisis exhaustivo. Al determinarse que las trazas no

[17] Ficheros de validación disponibles en: http://sid.cps.unizar.es/PMS/tests.html

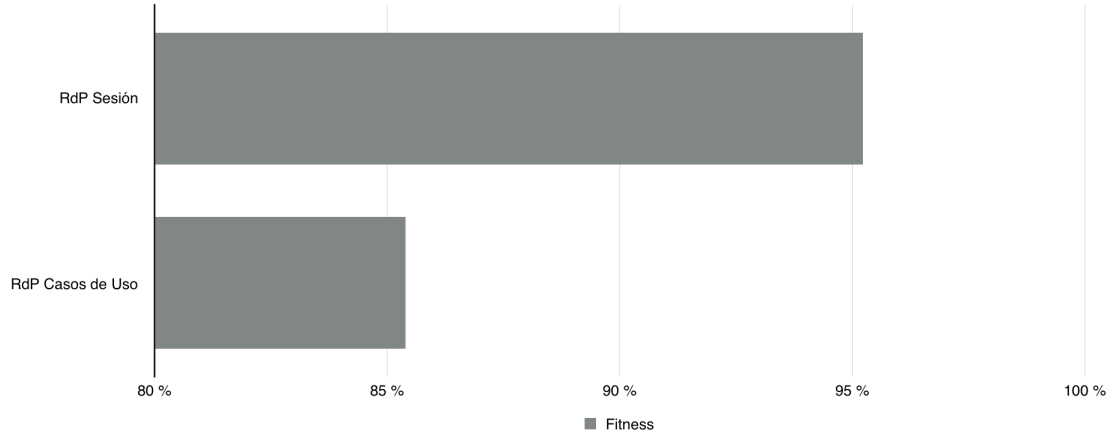


Figura 13: Comparación entre el porcentaje de *fitness* de las diferentes RdP.

coincidentes con el modelo normativo del sistema se correspondían con ataques o amenazas, o con comportamientos benignos no considerados en el desarrollo del sistema, se puede considerar que el resultado es satisfactorio.

Mediante estos métodos de validación, se consigue una comprobación íntegra de la aproximación propuesta, y más en concreto su aplicación al sistema concreto analizado; puesto que se asegura que todos los resultados obtenidos en las diferentes fases son válidos y que no se están dando errores. Por todo ello, se puede concluir que el método propuesto es válido.

6. Conclusiones

En este TFG se ha propuesto una aproximación novedosa para la mejora de la seguridad en sistemas de información Web, basada en la minería de procesos y la ingeniería del software dirigida por modelos. La aproximación propuesta se ha aplicado a dos casos de estudio: el sistema correspondiente a la "biblioteca digital SID" y el correspondiente al grupo de investigación "DisCo". Su aplicación al sistema "biblioteca digital SID" ha demostrado su viabilidad y eficacia para la detección y prevención de amenazas persistentes avanzadas (APT) en sistemas de información Web.

Por último, señalar que con los resultados de este TFG se ha escrito un artículo que ha sido aceptado para su publicación en el congreso DESEI+D 2016[18]; y que mediante la difusión a la comunidad científica, y la retroalimentación obtenida de la misma, se han abierto nuevas líneas de investigación, para mejorar la aproximación propuesta.

6.1. Problemas encontrados

Al tratarse de un tema de investigación, y debido al poco trabajo dado en el contexto tratado, se han dado varios problemas en todas las fases de la aproximación presentada. Esto se debe en gran medida, a que las principales herramientas de minería de procesos, son herramientas sencillas en proceso de desarrollo y aún no se encuentran en una fase estable. El principal *framework* de minería de procesos utilizado, ProM, aún siendo el más popular y usado en este contexto, tiene muchas limitaciones y fallos en los *plugins* necesarios para la aplicación de esta aproximación. No obstante, los problemas con ProM se han solventado buscando alternativas para diferentes técnicas y *plugins*. De forma análoga, la herramienta de conversión entre diagramas de actividad UML y modelos formales de redes de Petri, al estar en una fase de desarrollo temprana y sujeta a modificaciones en un futuro, consta de varios problemas, los cuales son descritos con más detalle en el anexo B.

Por otro lado ha resultado costoso adaptar el conocimiento disponible sobre minería de procesos, a un sistema muy flexible y dinámico como es la biblioteca digital del grupo SID. Se ha tenido que realizar una experimentación intensiva y multitud de pruebas de validación y comprobación en todas las fases de la aproximación. También debido a este aspecto, se han tenido que desarrollar las heurísticas propuestas en la fase de "Monitorización y control de los sistemas analizados", ya que algunos de los datos necesarios para la minería de procesos, no suelen estar disponibles en los casos de uso reales.

Por último, relacionado con lo anteriormente descrito, las técnicas, métodos y herramientas para tratar con grandes conjuntos de datos en tareas de minería de procesos, no son muy avanzadas (aunque existen aproximaciones y soluciones para algunos casos [23]). Esto provoca problemas de escalabilidad y obtención de resultados en un periodo corto de tiempo con grandes conjuntos de *logs* (el caso del principal sistema analizado).

[18] IV Congreso Nacional de I+D en Defensa y Seguridad: http://deseid.org/

6.2. Trabajo futuro

El trabajo futuro considerado para la aproximación propuesta se basa principalmente en su completa automatización. Aunque algunas de las fases son automáticas y son realizables en un periodo corto de tiempo, existen fases en las que se requiere de trabajo y supervisión manual. Esta característica de automatización adquiere más importancia teniendo en cuenta que el desarrollo (o actualización) de los modelos normativos del sistema, así como reflejar los resultados obtenidos por retroalimentación de las posteriores fases en esta, es una tarea manual. A su vez, el estudio y análisis, así como la obtención del modelo formal del comportamiento operativo del sistema, son tareas que requieren de actuación humana. Esto es debido en gran parte a que el *framework* utilizado para la fase anterior (además de su mal funcionamiento en algunos casos) no permite la interacción mediante herramientas externas para la realización de estas tareas.

Por último, cabe destacar la gran complejidad del proceso y la necesidad de revisión manual para la validación de los resultados. Es por esto que la ampliación de la aproximación propuesta para automatizar de manera total la misma es de gran complejidad. No obstante, las tareas de transformación entre modelos, obtención de resultados y gestión/detección de ataques y amenazas (una vez descubiertas) se realizan de manera totalmente automática, y en el caso de monitorización en tiempo real. Considerando la automatización total de la aproximación, se podría llegar a conseguir una aproximación exhaustiva y potente para la mejora de la seguridad de sistemas de información Web y la detección de amenazas persistentes avanzadas que se puedan dar en la misma (alcanzando un rango de ataques/amenazas mayor que los SIEM tradicionales).

Por otra parte, también se considera la mejora de las diferentes herramientas de transformación entre modelos, y las relacionadas con las diferentes tareas de la minería de procesos, las cuales se encuentran en una fase de desarrollo temprana.

Mediante la mejora de estos aspectos, y considerando la aplicación de la aproximación propuesta a otros sistemas o servicios (no solo a sistemas de información Web), se alcanzaría un gran avance en la investigación en el contexto en el que se basa el presente trabajo.

6.3. Opinión personal

Mi opinión personal sobre el proyecto realizado, es, en general, muy positiva. Mediante el desarrollo de este proyecto, he podido adquirir conocimiento y experiencia sobre la minería de procesos, la cual no es impartida en el grado, pero puede resultar de gran utilidad e interés. A su vez, y debido a la naturaleza de la minería de procesos, se ha podido adquirir un conocimiento más en profundidad sobre la minería de datos, los procesos de negocio, el modelado de sistemas y la ingeniería dirigida por modelos (la cual es interesante en el contexto de la ingeniería del software).

Por otra parte, al tratarse de un trabajo principalmente de investigación, he podido descubrir y experimentar algunas de las tareas relacionadas con el ámbito de la investigación, el cual tampoco se refleja de gran manera en el grado. La realización de esta investigación y la redacción del artículo

vinculado a este trabajo, me ha permitido conocer en mayor medida el ámbito investigador, y todas las herramientas, métodos y metodologías que se siguen en el mismo. Como ejemplo, se halla el aprendizaje de la redacción de un artículo de investigación, la utilización de herramientas del ámbito académico, el lenguaje LaTeX, las plataformas de publicación de artículos académicos, la búsqueda de artículos relacionados con un determinado tema, la correcta referencia de los mismos, etc.

Por último, considero que mediante este trabajo he conseguido adquirir conocimientos en nuevas áreas del ámbito de la ingeniería informática, que no hubiese sido posible adquirirlas en el ámbito de las asignaturas del grado, y que con el desarrollo del proyecto se puede conseguir. Lo más interesante en mi opinión de la realización de un trabajo de fin de grado, es el poder profundizar en áreas de interés propio y llegar más lejos de lo impartido y aprendido en el grado, poniendo interés en una determinada área que pueda estar relacionada con tu futuro.

A. Introducción a la minería de procesos

En este anexo, se presenta una introducción a la minería de procesos. La información correspondiente a este anexo ha sido obtenida en [1]. Entre las diferentes sub-secciones de este anexo, se describen todos los apartados de importancia de la minería de procesos, obteniendo una clara idea principal de lo que trata, cuales son sus objetivos y otras características.

A.1. Introducción

Los sistemas de información se están volviendo cada vez más entrelazados con los procesos operacionales que los mismos soportan. Como resultado, multitud de eventos son almacenados por los sistemas de información actuales. Aun así, las organizaciones constan de grandes problemas a la hora de extraer valor de estos datos. El objetivo principal de la minería de procesos es usar los datos de eventos almacenados para extraer información relacionadas con los procesos, como por ejemplo, descubrir automáticamente un modelo de proceso observando los eventos almacenados por un determinado sistema. La importancia de la minería de datos, está estrechamente relacionada con el gran crecimiento actual de datos sobre determinados eventos disponibles, y la limitación de las aproximaciones clásicas para la gestión de procesos de negocio. A su vez, la minería de procesos juega un papel de gran importancia en el cumplimiento de objetivos establecidos por las tendencias contemporáneas de gestión (como "SOX" y "Six Sigma"). Por todo lo anteriormente descrito, la minería de procesos es un tema actual de gran importancia, estrechamente relacionado con la minería de datos y la gestión de procesos de negocio.

En los últimos años, el crecimiento de los datos generados y almacenados por los diferentes sistemas de información, ha crecido exponencialmente. Debido a esto, y aun con todos los avances tecnológicos, las personas y las organizaciones dependen cada vez más de dispositivos de computación y fuentes de información en internet. Las cuales a su vez, constan de problemas para trabajar con cantidades de datos tan grandes. Por ello, uno de los principales objetivos de las organizaciones a día de hoy, recae en la extracción de información y valor de los datos almacenados en sus sistemas de información. Aún así, la importancia de los sistemas de información no se refleja únicamente en el gran crecimiento de datos disponibles, sino también en el papel que estos sistemas juegan en los procesos de negocio actuales. El reto entonces recae en explotar los datos sobre eventos de una manera en la que estos, adquieran cierto significado (de valor), esto es el objetivo que persigue la minería de datos. La minería de procesos, complementa las aproximaciones existentes como BPM (*"Business Process Management"*). Sobre esto, además, existen ciertas limitaciones de modelado (de aproximaciones existentes) como pueden ser las Redes de Petri, BPMN u otros. Algunos de estos lenguajes para modelar procesos, pueden ofrecer una vista limitada sobre los mismos, o carecen de notación para expresar ciertos aspectos importantes sobre los modelos dados. El principal problema existente es el alineamiento entre el modelo representado y la realidad, ya que el modelo puede idealizar el proceso representado, abstraerse o generalizar, y no representar con fidelidad la realidad (lo que puede llevar a tomar decisiones erróneas).

La minería de procesos, intenta conseguir (debido a los problemas presentados) relacionar los eventos a los modelos de procesos, consiguiendo una manera mejor de descubrir procesos, y mejorar los modelos existentes (evaluándolos y mejorándolos). La minería de procesos es una disciplina de investigación relativamente "joven", la cual se sitúa entre el aprendizaje automático y la minería de datos por un lado y el modelado de procesos y análisis en el otro lado. La idea de la minería de procesos es descubrir, monitorizar y mejorar los procesos reales, extrayendo conocimiento de los *logs* de eventos fácilmente disponibles en los sistemas actuales (ofreciendo la posibilidad de "cerrar" el ciclo BPM tradicional). Existen a su vez, tres actividades principales de minería de procesos (que permiten analizar el sistema desde diferentes perspectivas): *"discovery"* (desde eventos hasta un modelo sin usar información a priori), *"conformance"* (comprobar que el modelo se ajusta a los *logs* y viceversa), y *"enhancement"* (extender o mejorar el modelo de proceso existente).

Por todo lo anterior, la minería de procesos se sitúa como una potente herramienta dentro de BPM y también en el contexto de la inteligencia de negocios (BI).

A.2. Proceso de modelado y análisis

La plétora de notaciones para el modelado de procesos disponibles hoy en día, ilustran la relevancia del modelado de procesos. Existen multitud de organizaciones que operan a un nivel BPM de gran madurez, que usan modelos que pueden ser analizados y usados para representar los procesos operacionales. Aun así, la mayoría de estos modelos no se basan en un riguroso análisis de los datos existentes sobre el proceso (son hechos a mano). Además de esto, existen ciertas limitaciones de las aproximaciones clásicas, motivando de esta manera la necesidad de la aplicación de la minería de procesos.

Actualmente, los procesos de negocio son cada vez más complejos, dependiendo en gran medida en los sistemas de información, y pueden abarcar múltiples organizaciones. Por lo tanto, el modelado de procesos se ha convertido en un aspecto de extrema importancia. Los modelos de procesos, asisten en manejar la complejidad proporcionando visión y documentación de procedimientos. Como resultado, los modelos de procesos son ampliamente usados en las organizaciones actuales, debido a que su productividad puede ser notablemente incrementada (además de otros aspectos), existiendo algunos tipos de organizaciones las cuales confían fuertemente en el modelado. Estos modelos son usados para razonar sobre procesos (rediseño) y para realizar decisiones en los procesos (planificación y control). Aún así, la creación de modelos es una difícil tarea, muy propensa a errores, entre los que se incluyen: describir una versión idealizada de la realidad, la inhabilidad de capturar el comportamiento humano de manera adecuada, modelar en un nivel de abstracción equivocado, y otros varios a los que se tienen que enfrentar las organizaciones en el modelado. Estos errores, pueden llevar a un modelo inadecuado, que a su vez desembocan en malas conclusiones. La minería de procesos permite la extracción de modelos basados en hechos. Es más, la minería de procesos no se enfoca a crear un único modelo del proceso. En su lugar, provee varias vistas sobre la misma realidad en diferentes niveles de abstracción.

Desarrollar un modelo de proceso no es una tarea sencilla, y no es fácil hacer buenos modelos (sin embargo es una cuestión de gran importancia). Afortunadamente, la minería de procesos puede facilitar la construcción de mejores modelos en menos modelos. Algunos algoritmos de descubrimiento de procesos, como el algoritmo α, pueden generar automáticamente modelos de procesos. Como se ha indicado con anterioridad, existen varias notaciones para el modelado de procesos. Afortunadamente, la traducción entre diferentes notaciones puede realizarse de manera automática y sin grandes complicaciones (aunque se debe elegir siempre la mejor notación según el contexto dado). Este modelado de procesos y sus diferentes notaciones, se centra en la perspectiva de "flujo de control" de los procesos. El objetivo de un modelo de proceso es decidir cuales actividades necesitan ser ejecutadas y en que orden. Las actividades pueden ser secuenciales, opcionales o concurrentes, y la ejecución repetida de la misma actividad puede ser posible.

La notación mas básica para el modelado de procesos es un sistema de transiciones. Un sistema de transiciones consiste de estados y transiciones. Existe un estado inicial y un estado final (o varios), con varios posibles estados intermedios, todos con una etiqueta única. En esta notación, las transiciones se modelan con arcos, etiquetadas con el nombre de la actividad (pueden no ser únicas). De esta manera, dado un sistema de transiciones, se puede razonar acerca de su comportamiento. A su vez, existen varios problemas que se pueden detectar en un sistema de transición, como son los llamados *"deadlocks"* y *"livelocks"*. Cualquier modelo de proceso con semántica ejecutable puede ser traducido en un sistema de transiciones. Además, las nociones básicas de estado y transición, se encuentran en lenguajes de modelado de más alto nivel como redes de Petri, BPMN, y diagramas de actividad UML.

A continuación, se presentan los lenguajes de modelado más utilizados para el modelado de procesos.

- **Redes de Petri:** Las redes de Petri [2] son un lenguaje formal de modelado, introducido por C.A. Petri en su tesis doctoral en 1962. Las redes de Petri permiten modelar una gran variedad de sistemas (no sólo sistemas software), caracterizados por situaciones de concurrencia y/o conflictos entre actividades/eventos. Aunque la notación gráfica es intuitiva y simple, las redes de Petri son ejecutables y múltiples técnicas de análisis pueden ser usadas para analizarlas. Más en concreto, una red de Petri, es un grafo bipartito consistente de "lugares" y transiciones.

  La estructura de la red es estática, aunque su comportamiento dinámico es gobernado por las reglas de activación y disparo de las transiciones. De esta forma, una transición está activada si tiene un número suficiente de marcas en sus lugares de entrada (al menos tantas como las multiplicidades asociadas a los arcos de entrada correspondientes). Una transición activada puede disparar y su disparo provoca un cambio de estado. Por lo tanto, el nuevo marcado se obtiene restando de los lugares de entrada de la transición, tantas marcas como las multiplicidades de los arcos de entrada correspondientes, y añadiendo a los lugares de salida de la transición tantas marcas como las multiplicidades de los arcos de salida correspondientes.

  Por último, cabe destacar, que para las redes de Petri (marcadas) se puede establecer un

grafo de alcanzabilidad (que es un sistema de transiciones).

- **Workflow Nets:** es una subclase de las redes de Petri (WF-nets). Una red de flujo de trabajo (WF-net), es una red Petri con un lugar fuente marcado que representa el estado de inicio del proceso, y un lugar "sumidero" que representa la finalización del proceso. Es más, todos los nodos están en un camino desde la fuente hasta el sumidero. Son relevantes para el modelado de procesos de negocio, ya que tienen un inicio y final bien definido. Esta notación, también es la representación natural para la minería de procesos. Cabe destacar que no toda WF-net representa un proceso correcto (por ej., puede contener bloqueos). Como las redes de Petri, también consta de ciertas propiedades que se deben tener en cuenta en el modelado.

- **YAWL (Yet Another Workflow Language):** YAWL es un lenguaje de modelado de flujos de trabajo y un sistema de flujos de trabajo de código abierto. Realizando un análisis sistemático de notaciones ya existentes, se desarrolló este lenguaje, con unos patrones que cubren todas las perspectivas de los flujos de trabajo. El objetivo de YAWL es ofrecer soporte directo para varios patrones mientras se mantiene el lenguaje simple. Puede verse como una referencia de implementación de los patrones más importantes, en lo que a flujo de trabajo se refiere.

- **Business Process Modeling Notation (BPMN):** recientemente se ha caracterizado como uno de los lenguajes para el modelado de procesos de negocios más ampliamente utilizado. Es soportado por muchas herramientas y ha sido estandarizado por la OMG. Son básicamente diagramas de actividad UML estereotipados, donde las actividades atómicas son llamadas tareas. Consta de un conjunto de elementos notacionales amplio (más de 50), de los que se utilizan unos pocos en la realidad. Por lo tanto, su notación y las diferencias con las anteriores notaciones son aspectos importantes a considerar a la hora de elegir esta notación para modelar procesos.

- **Event-Driven Process Chains (EPC):** proveen de una notación clásica para modelar procesos de negocio. Básicamente, EPCs cubren un subconjunto limitado de BPMN y YAWL, usando una notación gráfica dedicada. EPC no ofrece una semántica clara y tampoco referencias sobre implementación, lo que introduce varios problemas. A pesar de estos problemas y su notación, los conceptos clave son muy similares a otros variados lenguajes.

- **Causal Nets:** son una representación adaptada hacia la minería de procesos. Una red causal es un grafo donde los nodos representan actividades y los arcos representan dependencias causales. Cada actividad consta de un conjunto de posibles enlaces de entrada y enlaces de salida (la lógica de rutas es únicamente representada por estos enlaces). Como otras notaciones explicadas con anterioridad, consta de una serie de propiedades que se deben de considerar en el modelado.

Respecto al análisis de procesos y sus modelos correspondientes, se debe de saber como explotar los datos de eventos para la realización de estas tareas. Existen dos aproximaciones convencionales

para el análisis basado en modelos: análisis de verificación y análisis de rendimiento (*"performance"*). El primero se preocupa de la corrección de un sistema o proceso, mientras que el segundo, se centra en tiempos de flujo, tiempos de espera, utilización y niveles de servicio. La verificación es un criterio de corrección que puede ser comprobado usando técnicas de verificación (comparación entre dos modelos, comprobación de propiedades, etc.). Respecto al rendimiento, se identifican tres dimensiones: tiempo, coste y calidad (con diferentes indicadores posibles). Existen varias limitaciones en ambos tipos de análisis de modelos ya que confían fuertemente en la disponibilidad y buena calidad de los modelos.

A.3. Minería de datos

La minería de procesos se constituye sobre dos pilares fundamentales: el modelado y análisis de modelos, y la minería de datos. Por lo tanto, es de gran importancia conocer el campo de la minería de datos y sus diferentes aproximaciones, debido a varias razones. En primer lugar, algunas técnicas de minería de procesos, se construyen sobre técnicas clásicas de minería de datos. En segundo lugar, ideas procedentes del campo de la minería de datos, son usadas para la evaluación de los resultados de la minería de procesos. Aunque la minería de datos tiene poca utilidad para ciertas tareas de la minería de procesos, un conocimiento básico de la minería de datos es de gran ayuda para comprender las diferentes técnicas de minería de procesos.

La minería de datos, es definida como el análisis de (habitualmente grandes) conjuntos de datos para encontrar relaciones insospechadas y para resumir los datos en formas novedosas, que son entendibles y útiles para el "dueño" de los datos. La popularidad de este campo, viene motivada básicamente por los mismos motivos que la minería de procesos, descritos con anterioridad. Para las diferentes técnicas de la minería de datos, es importante comprender el formato de los conjuntos de datos y entender como se pueden clasificar sus diferentes atributos e instancias posibles. Respecto las las diferentes técnicas de minería de datos, estas pueden ser clasificadas en dos principales categorías: aprendizaje supervisado y aprendizaje no supervisado.

En la categoría de aprendizaje supervisado se encuentran las tareas de clasificación y regresión. El aprendizaje supervisado asume datos etiquetados, es decir, existe una variable de respuesta que etiqueta cada instancia del conjunto de datos, siendo el resto variables "predictoras". El interés reside en explicar la variable de respuesta en términos de las variables "predictoras". Las técnicas de clasificación, asumen una variable de respuesta categórica y el objetivo es clasificar las diferentes instancias basándose en las variables "predictoras". Una técnica de clasificación clásica es la construcción y uso de árboles de decisión. En cambio, las técnicas de regresión asumen una variable de respuesta numérica, siendo el objetivo encontrar una función que ajuste los datos con el mínimo error. La técnica de regresión más usada habitualmente, es la regresión lineal.

En la categoría de aprendizaje no supervisado se encuentran las tareas de *clustering*/agrupamiento y descubrimiento de patrones. El aprendizaje no supervisado asume datos no etiquetados, es decir, las variables no se dividen en variables de respuesta y predictoras. Los algoritmos de *clustering* exa-

minan los datos para buscar grupos de instancias similares. De manera diferente a la clasificación, el foco no se centra en una variable de respuesta sino en la instancia como un todo. Algunas técnicas conocidas para la tarea de *clustering*, son el *clustering "k-means"* y el agrupamiento jerárquico. Respecto a la tarea de descubrimiento de patrones, existen muchas técnicas para ello, siendo el objetivo encontrar reglas de forma "Si X entonces Y" donde X e Y, se relacionan con los valores de diferentes variables. La técnica más conocida es la minería de reglas de asociación, donde el algoritmo más utilizado es el algoritmo *Apriori*. Los resultados de la minería de datos, en general, pueden ser descriptivos y predictivos.

Respecto al descubrimiento de patrones, además de las reglas de asociación, muchos otros patrones que pueden ser descubiertos. Un ejemplo es la minería de patrones secuenciales y de episodios, además de otras de gran relevancia para la minería de procesos. La minería de secuencias intenta superar el problema de no considerar el orden de los eventos (que sufre el algoritmo Apriori), analizando las secuencias de los conjuntos de instancias. El objetivo es encontrar secuencias frecuentes definidas por un patrón. Una secuencia es frecuente si el patrón esta contenido en una proporción predefinida de las secuencias en el conjunto de datos. De esta manera, se puede considerar que una secuencia es frecuente por su soporte (si es mayor que un determinado valor), o según otras medidas de la misma forma (como la confianza). Cabe considerar que si una secuencia es frecuente, entonces sus sub-secuencias son también frecuentes.

Por otra parte, la minería de episodios, trata del descubrimiento de episodios frecuentes. Este problema, de similar manera a la minería de secuencias, puede ser resuelto con una aproximación similar al algoritmo Apriori. En este caso, una ventana deslizante es usada para analizar con que frecuencia un episodio esta apareciendo. Un episodio define un orden parcial. El objetivo es descubrir episodios frecuentes. La entrada para la minería de episodios es una secuencia de tiempo. Esta secuencia consiste en puntos temporales discretos, y en algunos puntos del tiempo un evento ocurre (que consta de un tipo y una marca temporal). Un determinado episodio ocurre en una ventana temporal si el orden parcial esta "embedido" en ella. Un episodio está descrito por un grafo dirigido acíclico, donde los nodos representan tipos de eventos y los arcos definen un orden parcial. La minería de episodios y secuencias pueden ser vistas como variantes de el aprendizaje de reglas de asociación. Debido a que tienen en cuenta el orden de eventos, están relacionados con el descubrimiento de procesos, aunque existen varias diferencias con los algoritmos de minería de procesos.

En la minería de datos y aprendizaje automático, muchas otras técnicas han sido desarrolladas para analizar secuencias de eventos (además de las descritas). Sus aplicaciones tienen lugar en la minería de texto, bioinformática, reconocimiento de voz, analítica Web y muchas otras. Ejemplos de técnicas utilizadas para este propósito son las redes neuronales y los modelos ocultos de Markov.

Aunque estas técnicas de minería de datos descritas hasta ahora, pueden ser explotadas para la minería de procesos, no pueden ser usadas para tareas importantes de la minería de procesos como el descubrimiento de procesos, comprobación de la conformidad, y mejora de procesos. Como ocurre en la minería de datos, no es trivial analizar la calidad de los resultados de la minería de

procesos. Debido a esto, se puede obtener beneficio de las experiencias obtenidas en el campo de la minería de datos. Por lo tanto, es importante considerar las técnicas de validación y evaluación desarrollados para los algoritmos descritos con anterioridad.

Respecto a este aspecto, se puede medir el rendimiento de un determinado clasificador, el cual es relevante para juzgar lo fidedigno que es el clasificador resultante y para compararlo con diferentes aproximaciones. Aunque surge una complicación, debido a que solo se puede juzgar el rendimiento basándose en las instancias "vistas" o disponibles, mientras que el objetivo es también predecir buenas clasificaciones para instancias "no vistas". Para esta tarea, se pueden utilizar matrices de confusión, para visualizar los resultados de la clasificación, además de métricas basadas en la matriz de confusión (precisión, *"recall"*, *"F1 score"*, etc.). Debido a la complicación descrita anteriormente, el más obvio criterio para estimar el rendimiento de un clasificador es su precisión predictiva sobre instancias "no vistas", aunque su número puede ser muy grande (sino infinito), por lo tanto se necesita una estimación basada en un conjuntos de datos de test. Esto se refiere comúnmente a *"cross-validation"*. Básicamente los datos se dividen en un conjunto de entrenamiento (para aprender el modelo) y otro de test (para evaluar el modelo con instancias "no vistas"). De esta forma, evaluando el modelo con el conjunto de test se obtiene un indicador de rendimiento. Esta técnica no está solamente limitada a la clasificación, sino que se puede usar para cualquier técnica de minería de datos. Otra posibilidad es el uso de *"k-fold cross-validation"*. Esta aproximación es usada cuando hay relativamente pocas instancias en el conjunto de datos o cuando el indicador de rendimiento está definido sobre un conjunto de instancias antes que en una sola instancia. En esta aproximación, se divide el conjunto de datos en k partes equitativas, entonces k test son realizados. En cada test, uno de los subconjuntos de datos se utiliza como el conjunto de test, donde el resto de subconjuntos (k-1) sirven como conjuntos de entrenamiento (en conjunto), y estos subconjuntos van rotando y cambiando.

Evaluar la calidad de los resultados de minería de datos no es nada trivial. De esta forma, existen complicaciones adicionales que son también relevantes para la minería de procesos. Existen "prejuicios" representacionales, de aprendizaje e inductivas (que pueden no ser perjudiciales). Estos "prejuicios" deben ser tomados en cuenta y solucionados. A su vez, pueden existir otros tipos de problemas variados.

A.4. Obtención de datos

La minería de procesos resulta una tarea imposible sin los *logs* de eventos apropiados. Es de gran importancia saber la información que debe estar presente en estos *logs* de eventos. Aun así, dependiendo de la técnica de minería de procesos utilizada, los requerimientos pueden variar.

El reto reside en extraer dichos datos desde una gran variedad de fuentes de datos. Cuando se deben juntar y extraer datos, la sintaxis y la semántica juegan un papel muy importante. Por otra parte, dependiendo de las cuestiones para las que se busca una respuesta, se necesitan diferentes vistas sobre los datos disponibles. Habitualmente los datos proceden de fuentes operacionales, donde

mediante un proceso de ETL (*Extract, Transform and Load*) pueden ser cargados a un almacén de datos (opcionalmente) o extraídos directamente. Posteriormente, estos datos se transforman a un *log* de eventos en un formato estándar (XES, MXML o similar), formando un *log* de eventos sin filtrar. Por último, estos *logs* de eventos son filtrados para ser utilizados para los diferentes tipos de la minería de procesos (*discovery, conformance y enhancement*), proporcionando modelos de procesos y diferentes respuestas.

Respecto a los *logs* de eventos, es importante que cada evento haga referencia a una sola instancia de un proceso (también denominado caso). A su vez, es importante que un *log* de eventos contenga datos relacionados con un solo proceso. También se puede asumir que los eventos pueden estar relacionados a alguna actividad. Estas asunciones son naturales en el contexto de la minería de procesos. Por todo esto, las columnas de identificador de caso y actividad de los *logs* de datos, constituyen el mínimo para la minería de procesos. Además, los eventos dentro de un caso deben de estar ordenados. Sin información de orden, es imposible descubrir dependencias causales en los modelos de procesos. A su vez, estos *logs* pueden contener otro tipo de atributos que den mayor información para el análisis (marcas temporales, recursos, costes, etc.). Por último, estos *logs* de eventos, pueden ser filtrados y "adaptados" para mejorar el posterior análisis y su uso en los diferentes tipos de minería de procesos.

Continuando con los *logs* de eventos, como se ha descrito anteriormente, estos se deben especificar en un formato estándar para poder ser utilizados posteriormente de manera correcta. Hasta hace poco, el estándar de facto para almacenar e intercambiar *logs* de eventos era MXML (*Mining eXtensible Markup Language*). Mediante MXML, es posible almacenar *logs* de eventos usando una sintaxis basada en XML. Sin embargo, de este estándar surgieron muchas extensiones para interpretar atributos de cierta manera. Estas extensiones funcionan correctamente en la práctica, pero la gran variedad de ellas llevó al desarrollo de su estándar sucesor: XES (*eXtensible Event Stream*). XES se basa en muchas experiencias prácticas con MXML, haciendo el formato de XES menos restrictivo y verdaderamente extensible, lo que hizo que sustituyese a MXML. Aunque mediante XES, se ofrece una sintaxis concreta y un formato objetivo bien definido, la extracción de *logs* de eventos puede ser muy problemático. Los problemas principales de este proceso de extracción, son cinco: la correlación entre eventos, las marcas de tiempo de los eventos, las instantáneas (eventos no completos), la determinación del alcance de los *logs* de eventos y la granularidad de los eventos.

Para la realización de la minería de procesos, los eventos necesitan estar relacionados con casos. Todas las actividades en un modelo de proceso convencional corresponden a cambios de estados de tal caso. Se suele referir a estos modelos de procesos como modelos planos. Respecto a esto, es importante saber que los procesos reales no son "planos". Por este motivo, es de gran importancia saber como representar la realidad en *logs* de eventos (como modelos "planos").

Como la mayoría de técnicas de minería de procesos requieren "aplanar" los datos, se propone la siguiente aproximación (como la más adecuada):

- Crear un almacén de datos orientado a procesos, conteniendo información sobre eventos rele-

vantes.

- Dependiendo de las cuestiones, definir una vista apropiada.
- Utilizar un corte de dos dimensiones para aplicar una gran variedad de técnicas de minería de procesos. Si es necesario filtrándolo, continuar extrayendo, filtrando y minando hasta que las diferentes cuestiones sean resueltas.

A.5. Descubrimiento de procesos

El descubrimiento de procesos es una de las tareas de la minería de procesos más complejas. Basándose en el *log* de eventos, un modelo de proceso es construido capturando así el comportamiento visto en el *log*. Para esta tarea, se describe el algoritmo α. Este algoritmo ilustra las ideas generales usadas por varios algoritmos de minería de procesos, entendiendo la noción del descubrimiento de procesos. Este tipo de tareas se centran en el tipo de minería de procesos *"discovery"* y la perspectiva de control de flujo (combinación conocida como descubrimiento de procesos). En concreto, un algoritmo de descubrimiento de procesos, es una función que mapea un *log* de eventos a un modelo de proceso de tal manera que el modelo es "representativo" para el comportamiento observado en el *log*. El objetivo, es que el modelo obtenido pueda reproducir el *log* de eventos (todas sus trazas). Se trata de una técnica *"Play-In"*, que construye un modelo a partir de un log de eventos. En general, el modelo de proceso descubierto, debe cumplir cuatro criterios de calidad: *fitness* (modelo permite el comportamiento del *log*), precisión (no permite comportamiento diferente a lo observado en el *log*), generalización (modelo generaliza el comportamiento de ejemplo observado en el *log*), y simplicidad (el modelo debe ser todo lo simple posible).

El algoritmo α es un ejemplo de descubrimiento de procesos descrito anteriormente, es decir, dado un *log* de eventos "simple" produce una red de Petri que (habitualmente) puede reproducir el comportamiento del *log*. El algoritmo α es uno de los primeros algoritmos para el descubrimiento de procesos que puede tratar de manera adecuada la concurrencia. Aun así, este algoritmo no debe ser visto como una técnica muy práctica de minería de procesos, ya que tiene problemas con "ruido", comportamiento no frecuente o incompleto, y construcción de rutas complejas. El algoritmo α es simple y muchas de sus ideas han sido introducidas en otras técnicas mas complejas y robustas.

La entrada para el algoritmo α es un *log* de eventos simple. Las actividades correspondientes a este *log* de eventos, corresponden a las transiciones en la red de Petri descubierta. De esta manera, la salida de este algoritmo es una red de Petri "marcada". Entrando en el funcionamiento del algoritmo, este escanea el *log* de eventos en busca de patrones particulares. Sobre esto, se distinguen cuatro relaciones de orden (basadas en *logs*), que aspiran a capturar patrones relevantes en el *log*: consecuencia directa ($>$), causalidad ($\rightarrow$), consecuencia indirecta ($\#$), y concurrencia ($\|$). Estas cuatro relaciones de orden, se utilizan para el descubrimiento de patrones en el modelo de proceso correspondiente. Este algoritmo, es capaz de obtener redes "WF" (*WF-nets*) a partir de un *log* dado.

Aun así, aunque este algoritmo puede descubrir una gran clase de *"WF-nets"* si se asume que el *log* es completo respecto a la relación de orden (antes nombrada) de consecuencia directa, puede tener varias limitaciones y problemas. Incluso si se asume que el *log* es completo, el algoritmo α tiene algunos problemas. Existen muchas *"WF-nets"* diferentes pero con trazas equivalentes, lo que puede llevar al algoritmo a generar una red innecesariamente compleja. A su vez, este algoritmo tiene problemas para lidiar con bucles cortes (de longitud uno o dos), descubrimiento de dependencias no locales, y otras. Otra limitación se debe a que las frecuencias no se tienen en cuenta, siendo el algoritmo muy sensitivo al ruido y la incompletitud. Debido a esto, existen variaciones y mejoras del algoritmo α, para lidiar con bucles y otras cuestiones. Por último, el algoritmo α puede ser fácilmente adaptado para tener el ciclo de vida transaccional en cuenta.

Respecto a este y otros algoritmos, se puede crear un entorno experimental para comprobar los algoritmos de descubrimiento de procesos en los cuales se asume que el modelo original se conoce. Utilizando como punto de partida un modelo de proceso dado, basándose en el, se pueden ejecutar varios experimentos de simulación y almacenar los eventos simulados en *logs*. Tras la obtención de este *log*, se descubre otro modelo de proceso con un algoritmo, y se comparan ambos modelos para comprobar si son equivalentes (y comprobar su funcionamiento). La comparación (para ver si son equivalentes) entre el modelo descubierto y el modelo original usado en el punto de partida, se conoce como el problema de redescubrimiento. A su vez, también se puede realizar esta comparación directamente entre *log* y modelo. El algoritmo α es uno de los primeros algoritmos para el descubrimiento de procesos en capturar adecuadamente la concurrencia. Hoy en día, existen algoritmos mucho mejores que salvan las debilidades de el algoritmo α. Estos son variantes del algoritmo α o algoritmos que utilizan una aproximación completamente diferente. Para ello, es necesario identificar los retos principales a los que se tienen que enfrentar estos algoritmos, relacionados generalmente con errores representacionales o relacionados con el *log* de eventos de entrada (ruido o incompletitud).

Respecto a los errores representacionales, toda técnica de descubrimiento de procesos sufre de ellos. Aunque estas ayudan a limitar el espacio de búsqueda de candidatos de modelos disponibles y puede hacer los algoritmos más eficientes, pueden dar preferencia a modelos de un tipo particular (lo cual puede ser un problema). Por ello, las aproximaciones existentes pueden obtener beneficio seleccionando un error representacional más adecuado. Por ejemplo, el algoritmo α puede producir modelos con bloqueos (*"deadlocks"* o *"livelocks"*). Aquí, lo mas adecuado sería escoger un error representacional que se limitase a un espacio de búsqueda con modelos libres de estas anomalías. Desafortunadamente, actualmente, esto solo puede ser conseguido limitando severamente la expresividad del lenguaje de modelado, o usando técnicas de análisis que consuman más tiempo.

Otro aspecto, es que para descubrir un modelo de proceso ajustado, se asume que el *log* de eventos contiene un ejemplo representativo de comportamiento. Sin embargo, existen dos problemas o fenómenos que pueden hacer un *log* de eventos menos representativo para el proceso que se está estudiando: el ruido y la incompletitud. El ruido hace referencia a que el *log* de eventos contenga comportamiento extraño o poco frecuente, no representativo para el comportamiento

típico del proceso (*"outliers"*). El ruido no se debe confundir con eventos de *logs* incorrectos (estos deben de ser eliminados con anterioridad). Para la minería de procesos, es de gran importancia filtrar el ruido, y varias aproximaciones de descubrimiento de procesos se especializan en esto. Para esta tarea de filtrado de ruido, pueden utilizarse métricas como el soporte y la confianza. En el contexto de ruido, se considera el modelo 80/20, que se refiere al interés de que el modelo de proceso describa el 80 % del comportamiento observado en el *log*. Por la otra parte, la incompletitud hace referencia al caso en el que el *log* de eventos contenga demasiados pocos eventos para ser capaces de descubrir algunas de las estructuras subyacentes del flujo de control. En la minería de procesos, la noción de completitud es muy importante (está relacionada con el ruido). La completitud, más concretamente, hace referencia al problema de tener "muy pocos datos". Como en cualquier contexto de la minería de datos o aprendizaje automático, no se puede asumir haber visto todas las posibilidades en el conjunto de datos de entrenamiento (no se puede asumir que cada posible traza está presente en el *log*).

Todo lo descrito con anterioridad sobre la completitud y ruido, muestra la necesidad de realizar validación cruzada (*"cross-validation"*) mencionada con anterioridad. El *log* de eventos, puede ser dividido en un conjunto de entrenamiento y un conjunto de test (como se describe con anterioridad). De igual manera, y para el mismo propósito, se puede usar el *"k-fold cross-validation"*. Con estas técnicas de validación, se pueden disminuir o incluso solventar los problemas de completitud y ruido.

La completitud y el ruido se refieren a cualidades del *log* de eventos, y no dicen mucho de la calidad del modelo descubierto. Determinar la calidad de un resultado de minería de procesos es difícil y se caracteriza por varias dimensiones. Debido a esto, se establecen cuatro principales dimensiones de calidad: aptitud (*"fitness"*), simplicidad (*"simplicity"*), precisión, y generalización. Mediante estas cuatro dimensiones de calidad, se puede evaluar la calidad del modelo descubierto (aunque existen otras que pueden resultar de interés según el caso).

Estos ejemplos mostrados, ilustran que el descubrimiento de procesos no es un problema trivial, el cual requiere de técnicas de análisis sofisticadas. Estos problemas también se aplican a la minería de datos y el aprendizaje automático. Además, se dan problemas específicos de cada contexto. Todos estos problemas deben considerarse y resolverse para una buena realización de las tareas de minería de procesos.

B. Herramienta para la transformación de diagramas de actividad en UML, a un modelo formal de red de Petri

En este anexo, se describe la herramienta (utilizada en el proyecto) para la transformación de diagramas de actividad en UML, a un modelo formal de red de Petri. Esta herramienta, está actualmente en fase de desarrollo en el contexto del proyecto Europeo DICE [18].

B.1. Introducción

El proyecto DICE tiene como objetivo definir un marco impulsado por la calidad para el desarrollo de aplicaciones intensivas en datos (DIA), las cuales aprovechan las tecnologías relacionadas con Big Data.

Siguiendo el paradigma de ingeniería del software dirigida por modelos (MDSE), las aplicaciones se describen con el lenguaje UML, y las propiedades no funcionales (p. ej., tiempos de ejecución de tareas) de las "DIA" se especifican usando un perfil UML ("DICE profile"). Un conjunto de herramientas de simulación, análisis y optimización utilizan modelos UML anotados con el perfil DICE para obtener aplicaciones de alta calidad. Una de estas herramientas es la herramienta de simulación "DICE Simulation Tool" [17], que permite la evaluación de propiedades de prestaciones (rendimiento) y de fiabilidad. Esta herramienta en concreto, es la utilizada en este proyecto para generar de forma automática un modelo en red de Petri a partir de los diagramas de actividad UML.

B.2. Descripción

La herramienta "DICE Simulation Tool", componente clave del "DICE Framework", es capaz de simular el comportamiento de aplicaciones intensivas en datos (DIA) para evaluar su rendimiento con un modelo de red de Petri estocástico.

Entre las funciones de esta herramienta, se incluyen la capacidad de transformación de diagramas de actividad, secuencia y despliegue UML a modelos formales de redes de Petri estocásticos y la simulación de los modelos generados con herramientas externas para la obtención de métricas de prestaciones y fiabilidad (p. ej., tiempo de respuesta, utilización y disponibilidad de recursos).

A partir de un modelo UML anotado con el perfil DICE, la herramienta genera un modelo de configuración que permite al usuario asignar valores a los parámetros iniciales (especificados en las anotaciones). Posteriormente, para cada configuración de los parámetros, la herramienta genera un modelo de red de Petri estocástico en dos formatos: el formato estándar PNML y el formato propio de la herramienta externa "GreatSPN", utilizada para la simulación. Finalmente, la herramienta genera un informe con los resultados de la simulación.

En concreto, esta herramienta está implementada como *plugin* de Eclipse. Los diagramas UML

anotados con el perfil DICE se crean con los *plugins* "Papyrus" y "DICE-Profile". Los diagramas UML aceptados por la herramienta son diagramas de actividad, de secuencia y de despliegue.

Una vez modelados todos los diagramas de actividad del sistema o aplicación correspondiente, la herramienta es capaz de llevar a cabo una simulación de la misma para la obtención de datos de evaluación de rendimiento (con todos los pasos intermedios correspondientes).

B.3. Utilidad

La utilidad de esta herramienta en este proyecto es clara. Uno de los principales objetivos del proyecto es la realización/mejora de diagramas UML, para su posterior transformación a un modelo formal (como son las redes de Petri). Por lo tanto, el uso de esta herramienta permite de forma semi-automática la consecución de parte de este objetivo, que es la transformación entre modelos en sí. En este proyecto no se han utilizado las anotaciones del perfil DICE, puesto que el objetivo es obtener un modelo en red de Petri sin especificaciones estocásticas.

No obstante, aunque la herramienta permite la realización de tareas de simulación, estas no se han requerido para el desarrollo del trabajo, por lo que se ha prestado especial atención a la transformación intermedia desde UML a PNML (red de Petri). Además, aunque la transformación podría haberse realizado de manera manual (sin el uso de esta herramienta), resulta de gran utilidad y ahorra tiempos para la aplicación de la aproximación en sí al sistema que se quiera aplicar. Por último, en lugar de realizar una transformación de UML a red de Petri, cabe la posibilidad de elaborar una red de Petri de forma directa, pero puede resultar costoso y no tan simple como las otras posibilidades.

B.4. Uso en el proyecto

El uso en el proyecto, se ubica en la tarea definida en la propuesta como "Definición de transformaciones necesarias para la construcción de modelos de análisis formal a partir de los diagramas UML". Una vez desarrollados los diagramas UML correspondientes del sistema, se consiguen realizar las transformaciones necesarias, a un modelo de análisis formal (red de Petri), gracias a esta herramienta.

Así pues, se ha utilizado en gran medida para la realización de esta tarea en los sistemas tratados en este proyecto, tanto en el inicio, como después de modificar los diagramas de actividad UML del sistema, tras la retroalimentación obtenida con los experimentos correspondientes y el comportamiento operativo del sistema tratado.

B.5. Problemas relacionados

Dado que "DICE Simulation Tool" es una herramienta en proceso de desarrollo y no final, consta de problemas asociados a este proceso. Estos problemas, han tenido que ser resueltos en la

realización de los procesos y experimentos pertinentes del trabajo presentado.

Uno de los mayores problemas presentados en esta herramienta, es que está diseñada y pensada para la transformación y simulación sobre un solo diagrama de actividad o diagrama de secuencia (que representa un solo escenario de ejecución del sistema). No obstante, es capaz de combinar diagramas de actividad para su transformación y simulación, aunque con grandes limitaciones. Estas limitaciones se basan, sobre todo, en el proceso de transformación al modelo de red de Petri (PNML), en el que los nombres de identificadores de nodos en los diagramas de actividad, son mapeados directamente a nombres de transiciones en la red de Petri generada. Esto puede suponer un problema de conflictos entre nombres, provocando la imposibilidad de generar un modelo correcto. Aún así, teniendo en cuenta los nombres de nodos en los diagramas de actividad, se puede llegar a conseguir un modelo lo suficientemente representativo y correcto. No obstante, la generación de un solo diagrama de actividad representando todo el conjunto del sistema (en lugar de varios separados), asegura una mejor interpretación y transformación al modelo de red de Petri.

Por otra parte, la traducción de diagramas de actividad en UML a PNML (estándar para descripción de redes de Petri), puede resultar en el incumplimiento del estándar PNML (al considerar nombres de nodos como identificadores), lo que afecta posteriormente a herramientas que trabajen sobre este modelo (como ProM). Aun así, es posible comprobar y validar el modelo PNML resultante, resolviendo los conflictos pertinentes y asegurando una interpretación correcta en otras herramientas externas.

Por último, la herramienta puede en algunos casos generar redes de Petri con comportamiento ad-hoc para el análisis de prestaciones, pero no adecuado para el análisis con técnicas de minería de procesos. Para resolver y visualizar el correcto modelado de la red de Petri, se ha hecho uso de la herramienta ePNK [19], la cual permite editar y visualizar de manera simple e intuitiva la red de Petri resultante de la transformación descrita.

B.6. Problemas resueltos mediante esta herramienta

Mediante esta herramienta, se ha conseguido alcanzar uno de los sub-objetivos principales del proyecto, la transformación de diagramas UML del sistema a un modelo formal (red de Petri), el cual puede ser usado en tareas de minería de procesos, con el principal "framework" utilizado (ProM).

No obstante, si no hubiese sido posible realizar la transformación de los diagramas UML a un modelo formal de red de Petri, se podría haber elaborado la red de Petri directamente. Aún así, el proceso de elaboración de una red de Petri del comportamiento del sistema, puede ser más complejo que el desarrollo de los diagramas de actividad del mismo. Además se debe de tener en cuenta que muchos de los sistemas en los que se puede aplicar la aproximación descrita en esta memoria, pueden constar ya de los diagramas UML del sistema de la fase de diseño (si el sistema se ha desarrollado de una forma óptima y progresiva), siendo un gran avance para cumplir con los requerimientos de la aproximación presentada.

C. Diagramas UML realizados para el principal sistema analizado

En este anexo, se procede a mostrar los diferentes diagramas UML realizados en la aplicación de la aproximación propuesta en este trabajo, al principal sistema analizado (el correspondiente a la biblioteca digital del grupo de investigación SID).

Respecto al desarrollo de diagramas UML para este sistema, cabe destacar que se han realizado diagramas de actividad, de secuencia, de casos de uso, y de clases y paquetes. Cabe destacar que aunque solamente los diagramas de actividad han sido usados en la aplicación de la aproximación, debido a las limitaciones de la herramienta de conversión a modelos formales, los demás diagramas UML han sido realizados para obtener un mayor y más profundo conocimiento del sistema, y para plantear otras posibles aproximaciones que se podrían llevar a cabo, por ejemplo, con los diagramas de secuencia . A su vez, cabe destacar la gran complejidad para el desarrollo de los diferentes diagramas UML citados, ya que debido a su no disponibilidad, ha sido necesario realizar un procedimiento de ingeniería inversa del sistema analizado, el cual ha sido bastante costoso en tiempo, además de estar sujeto a varias validaciones sucesivas.

A continuación se procede a presentar las figuras correspondientes a los diagramas UML elaborados[19].

[19]Diagramas UML completos disponibles en: http://sid.cps.unizar.es/PMS/resources/pdf/

Diagrama de casos de uso

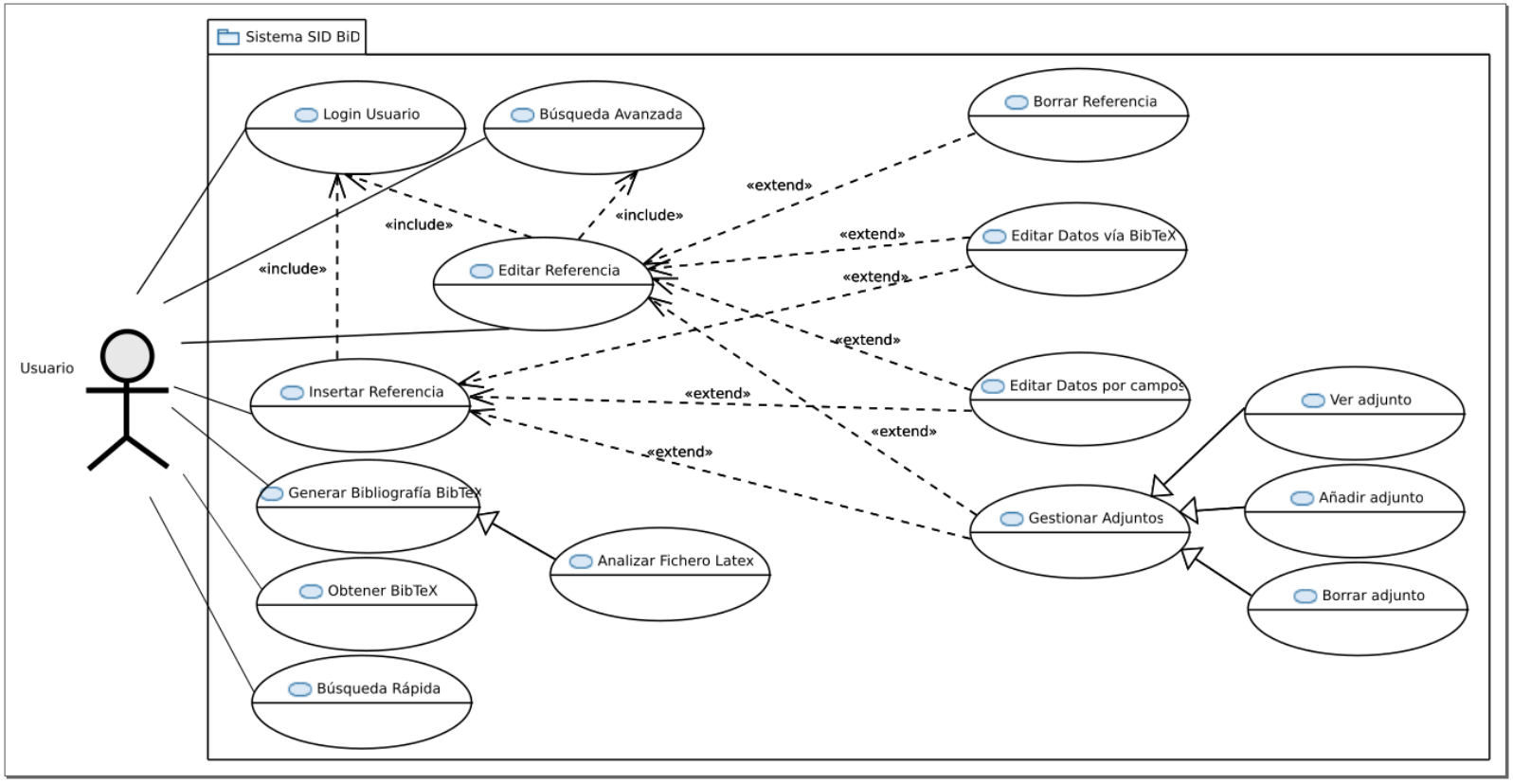


Figura 14: Diagrama de casos de uso del sistema.

Diagrama de clases del sistema 1 (Paquete Applet)

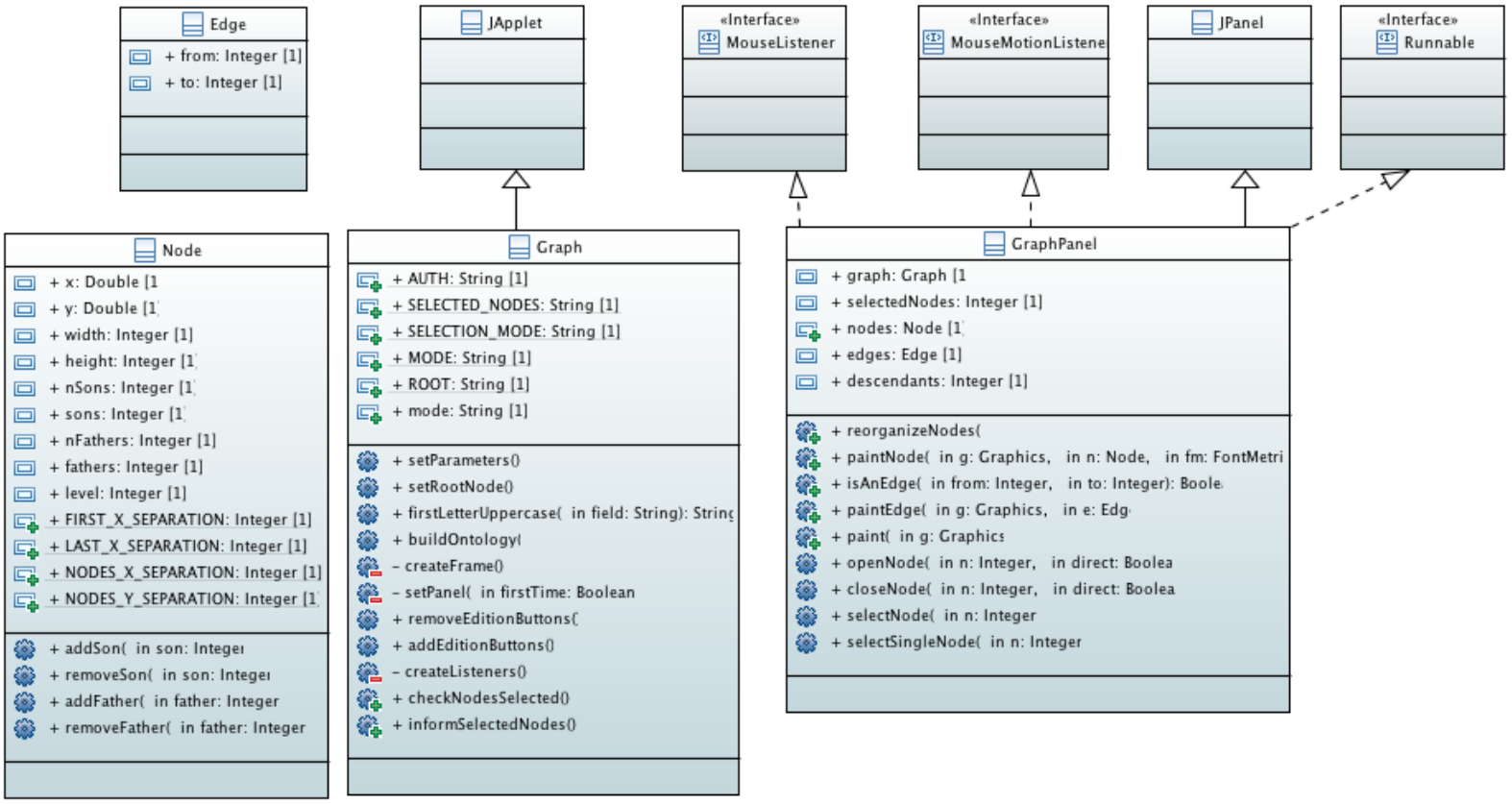


Figura 15: Diagrama de clases del sistema 1 (Paquete Applet).

Diagrama de clases del sistema 1 (Paquete Bean)

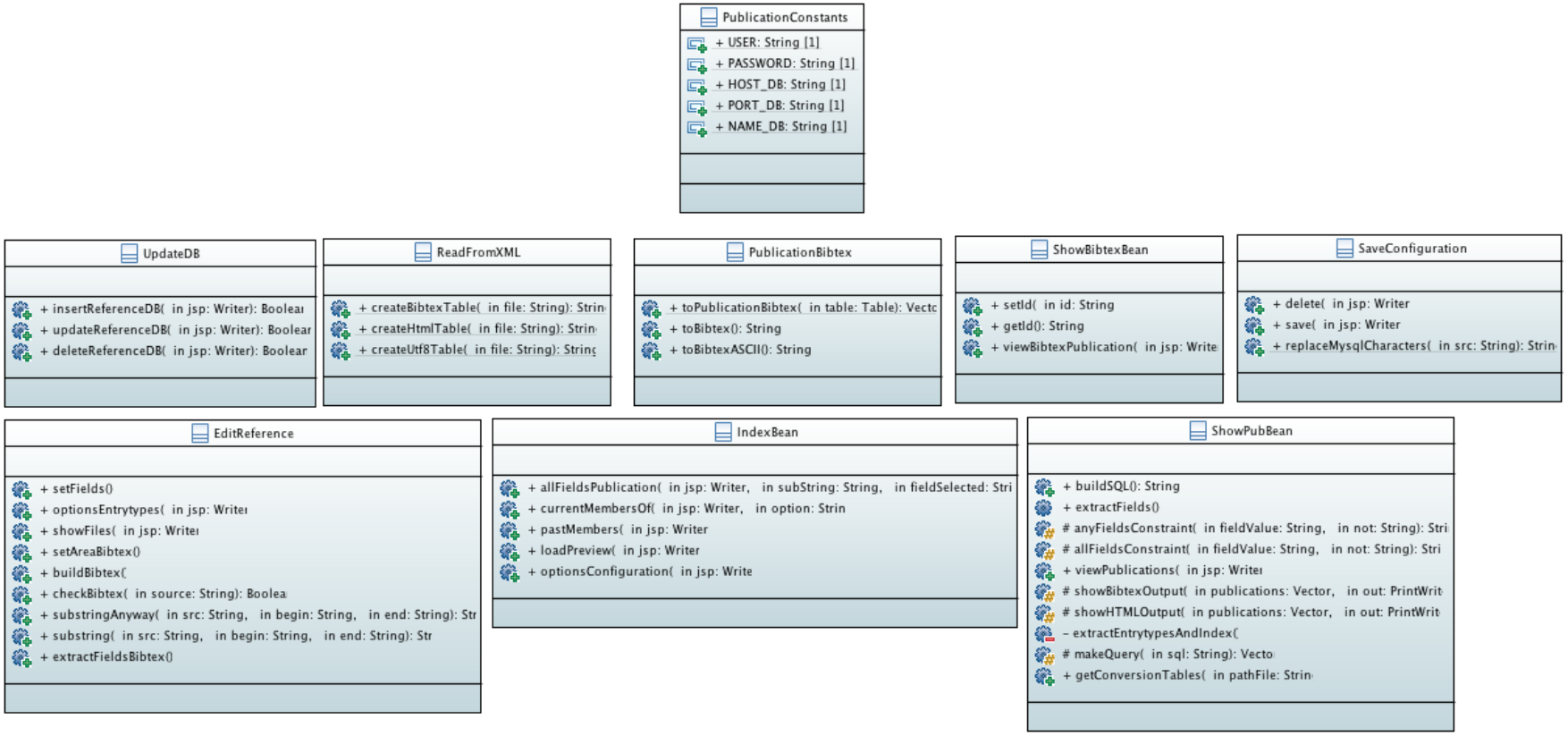


Figura 16: Diagrama de clases del sistema 2 (Paquete Bean).

Diagrama de clases del sistema 3 (Paquete BibTeX)

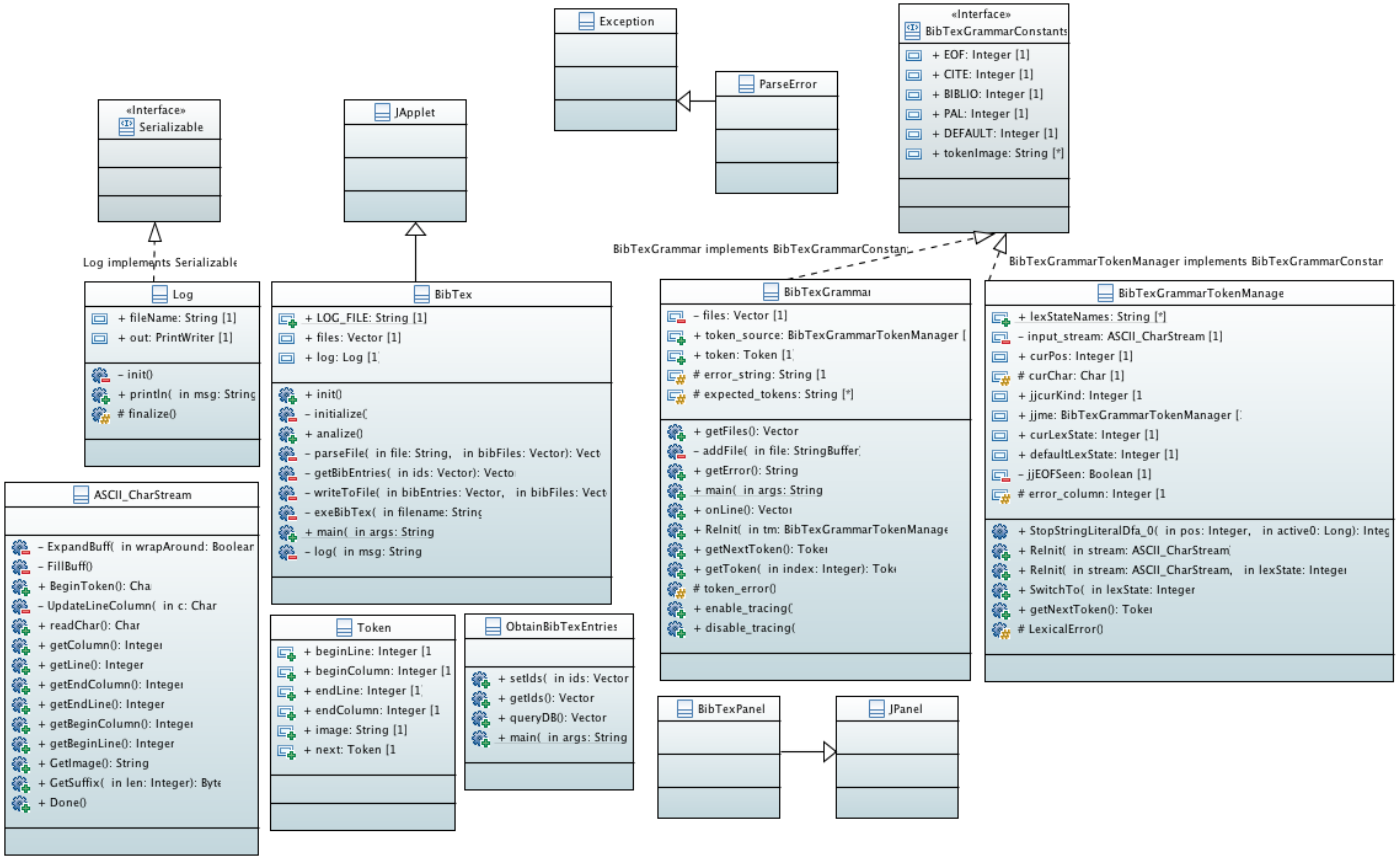


Figura 17: Diagrama de clases del sistema 3 (Paquete BibTeX).

Diagrama de secuencia (Búsqueda Avanzada)

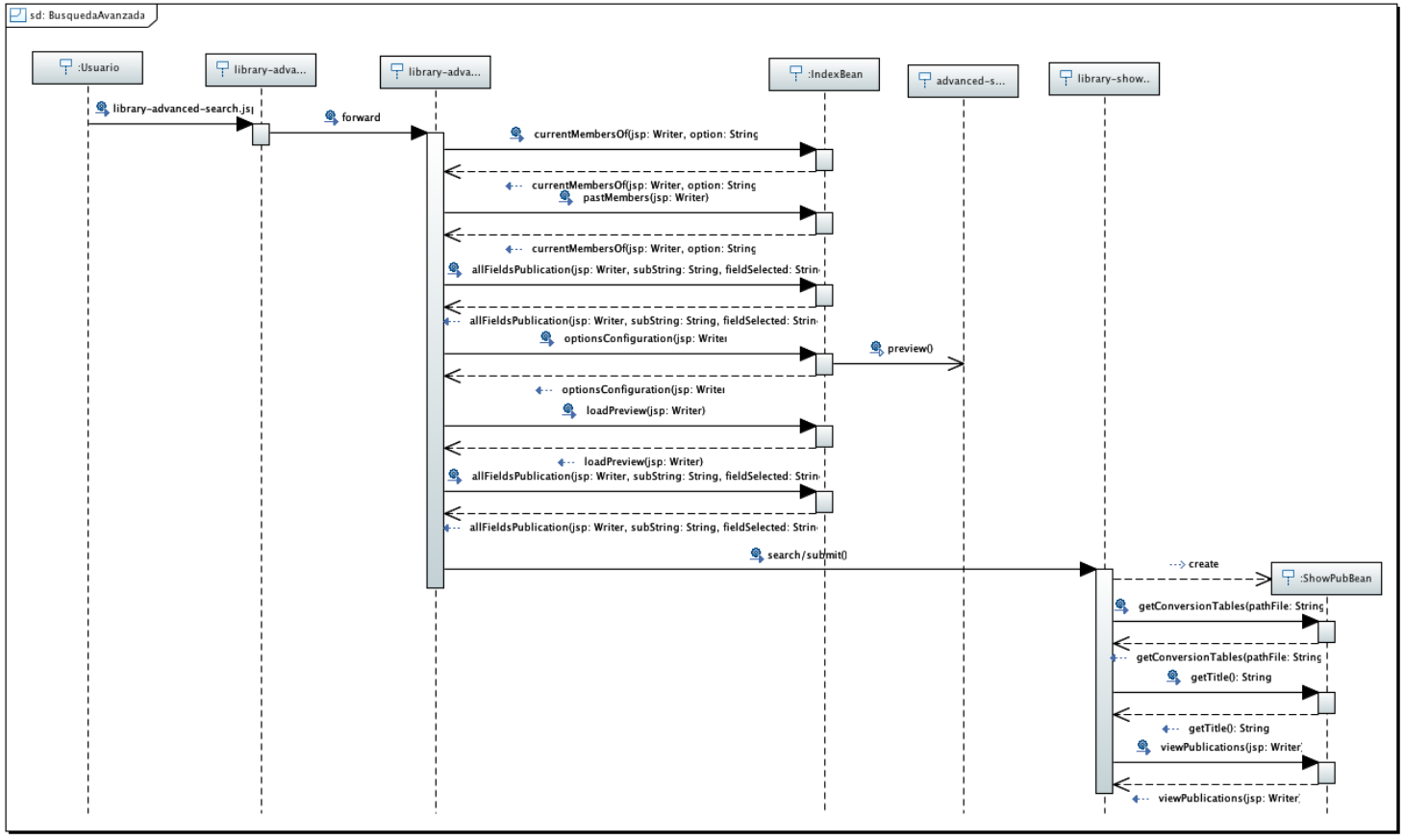


Figura 18: Diagrama de secuencia (Búsqueda Avanzada).

Diagrama de secuencia (Insertar Referencia)

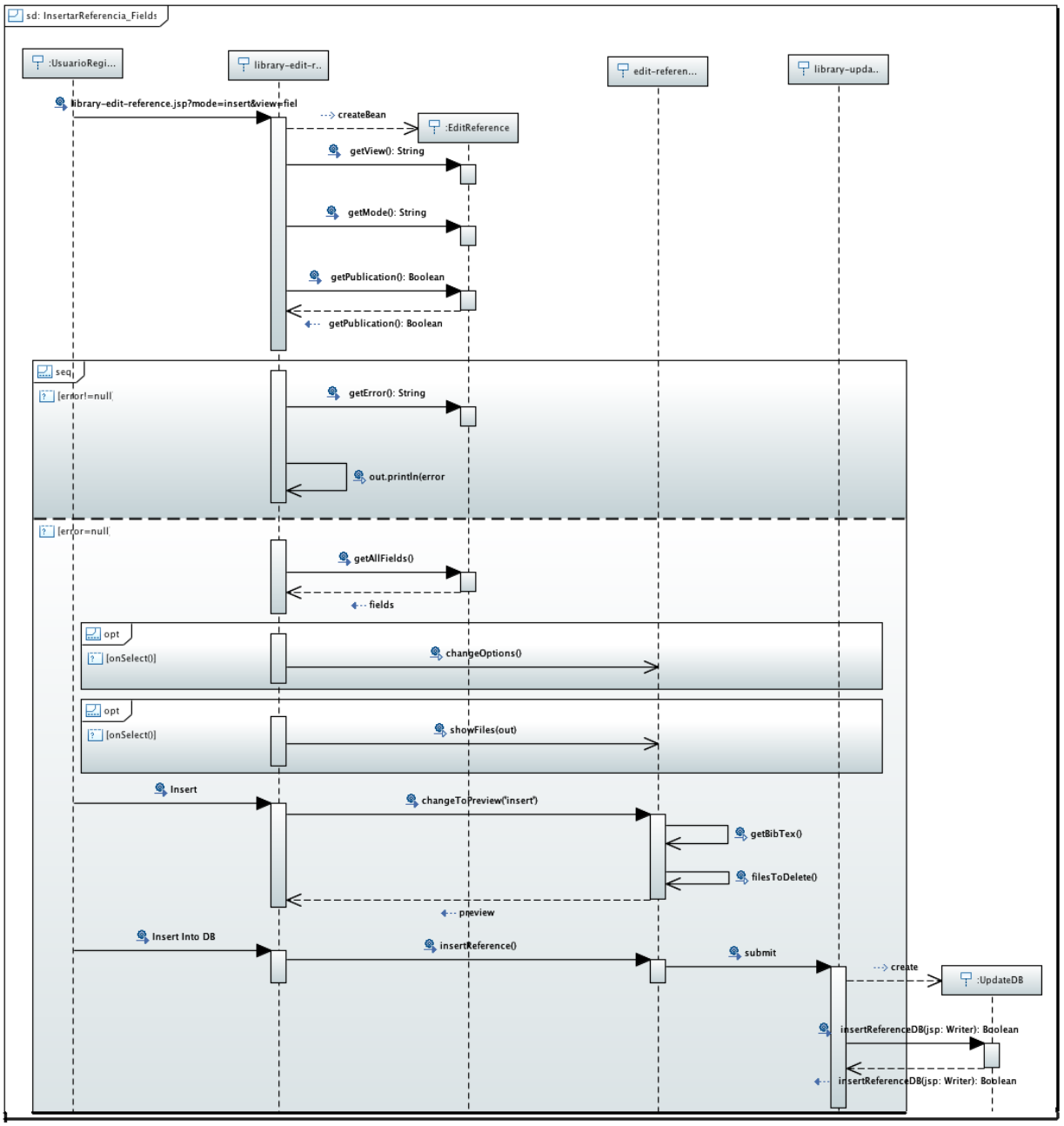


Figura 19: Diagrama de secuencia (Insertar Referencia).

Diagrama de secuencia (Obtener todos en formato BibTeX)

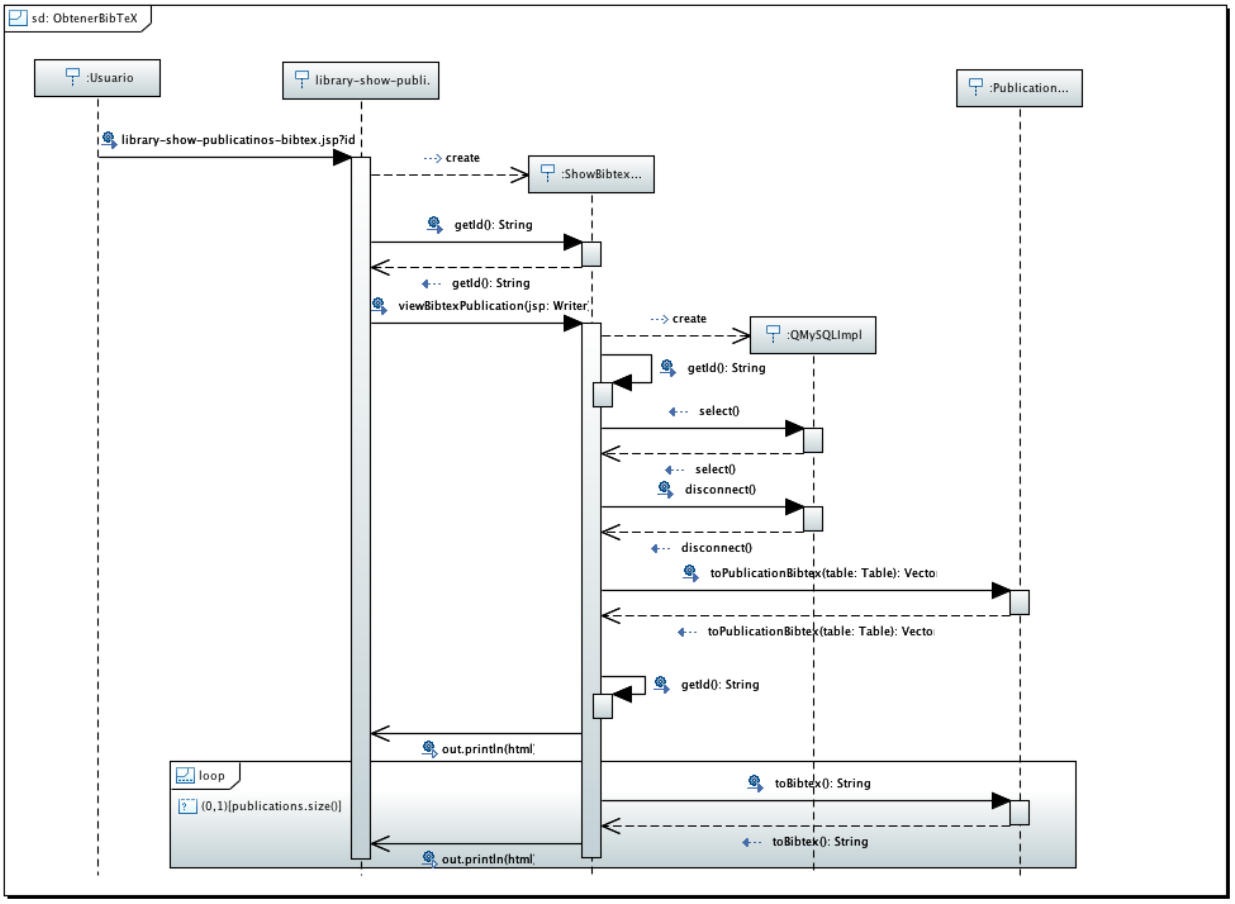


Figura 20: Diagrama de secuencia (Obtener todos en formato BibTeX).

Diagrama de actividad (Build BibTeX)

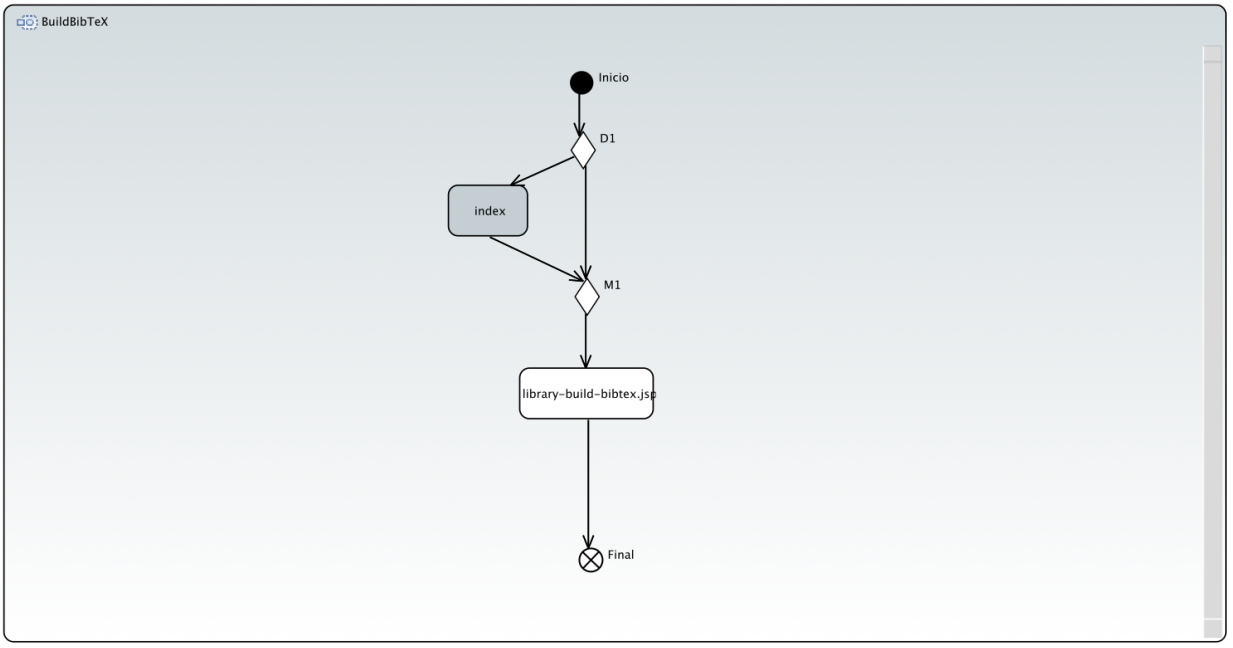


Figura 21: Diagrama de actividad (Build BibTeX).

Diagrama de actividad (Búsqueda Simple)

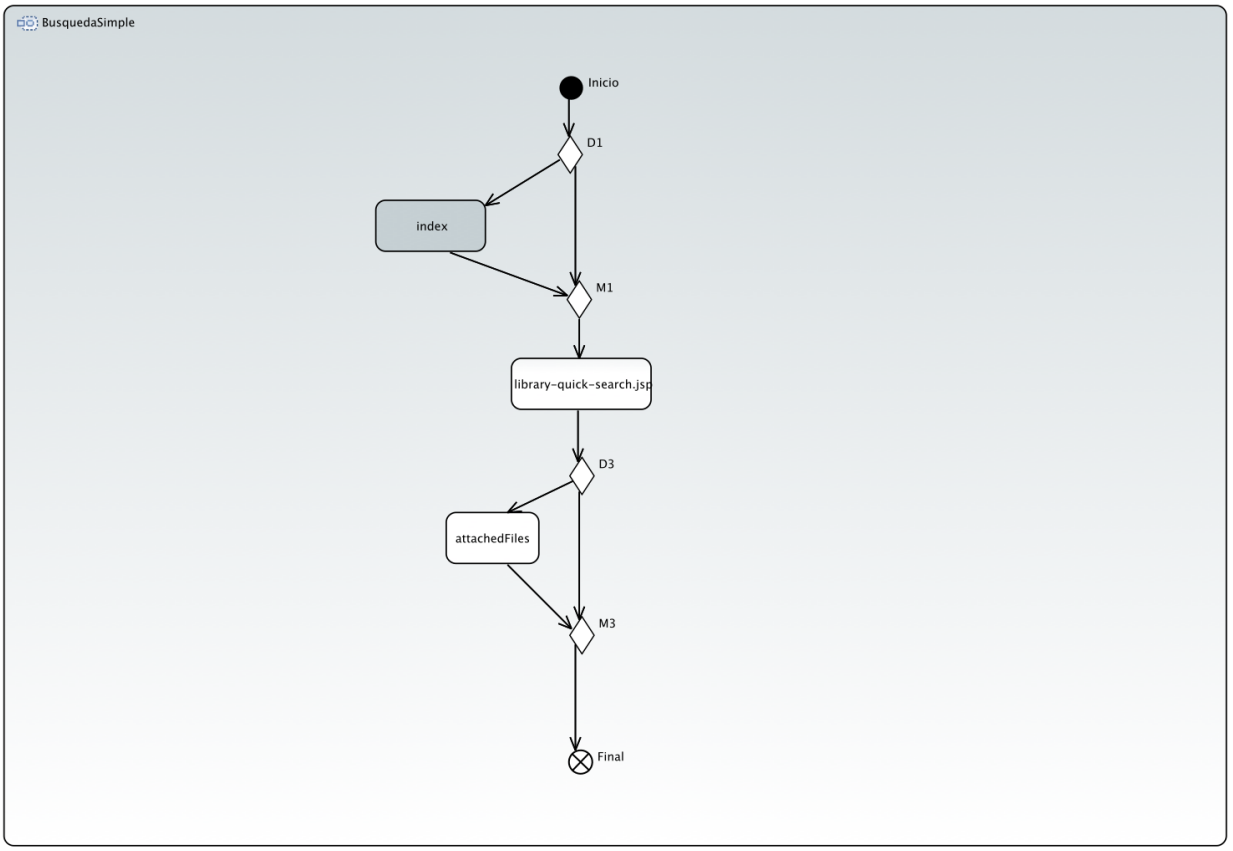


Figura 22: Diagrama de actividad (Búsqueda Simple).

Diagrama de actividad (Búsqueda Avanzada)

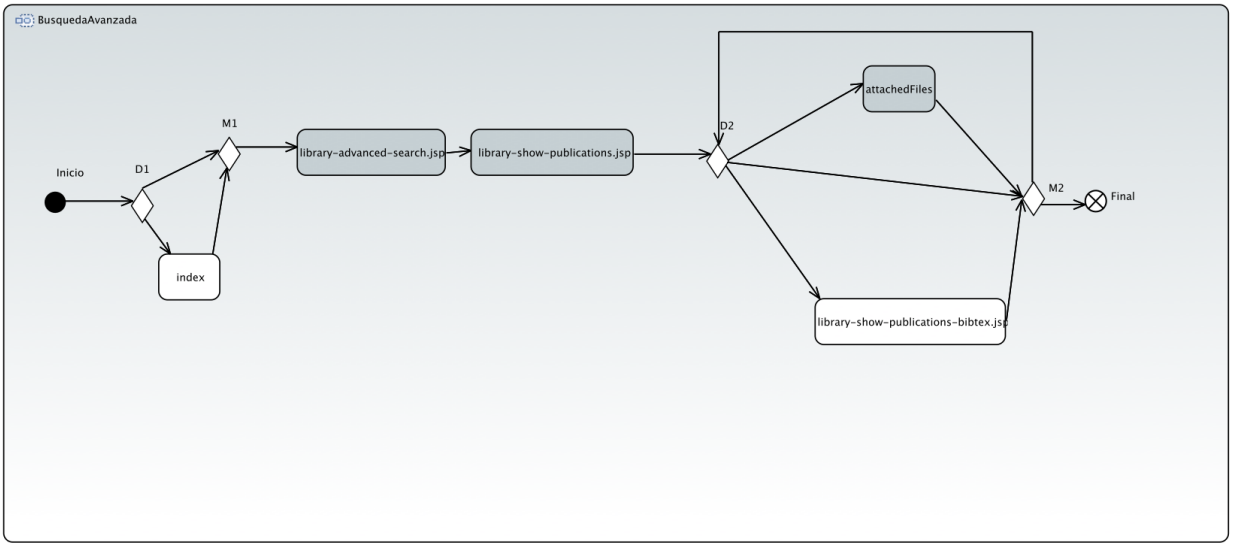


Figura 23: Diagrama de actividad (Búsqueda Avanzada).

Diagrama de actividad (Editar Publicación)

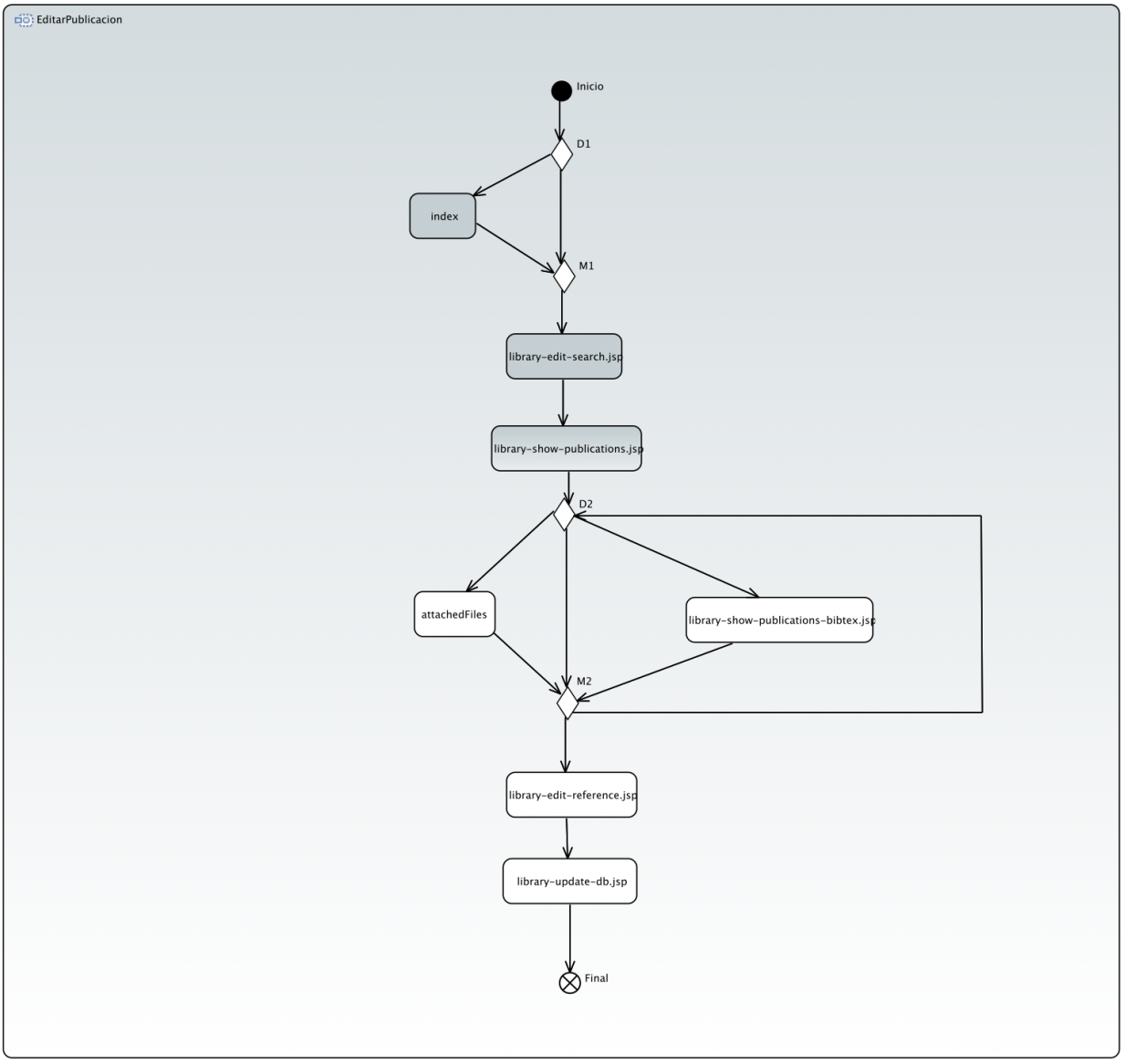


Figura 24: Diagrama de actividad (Editar Publicación).

Diagrama de actividad (Insertar Referencia)

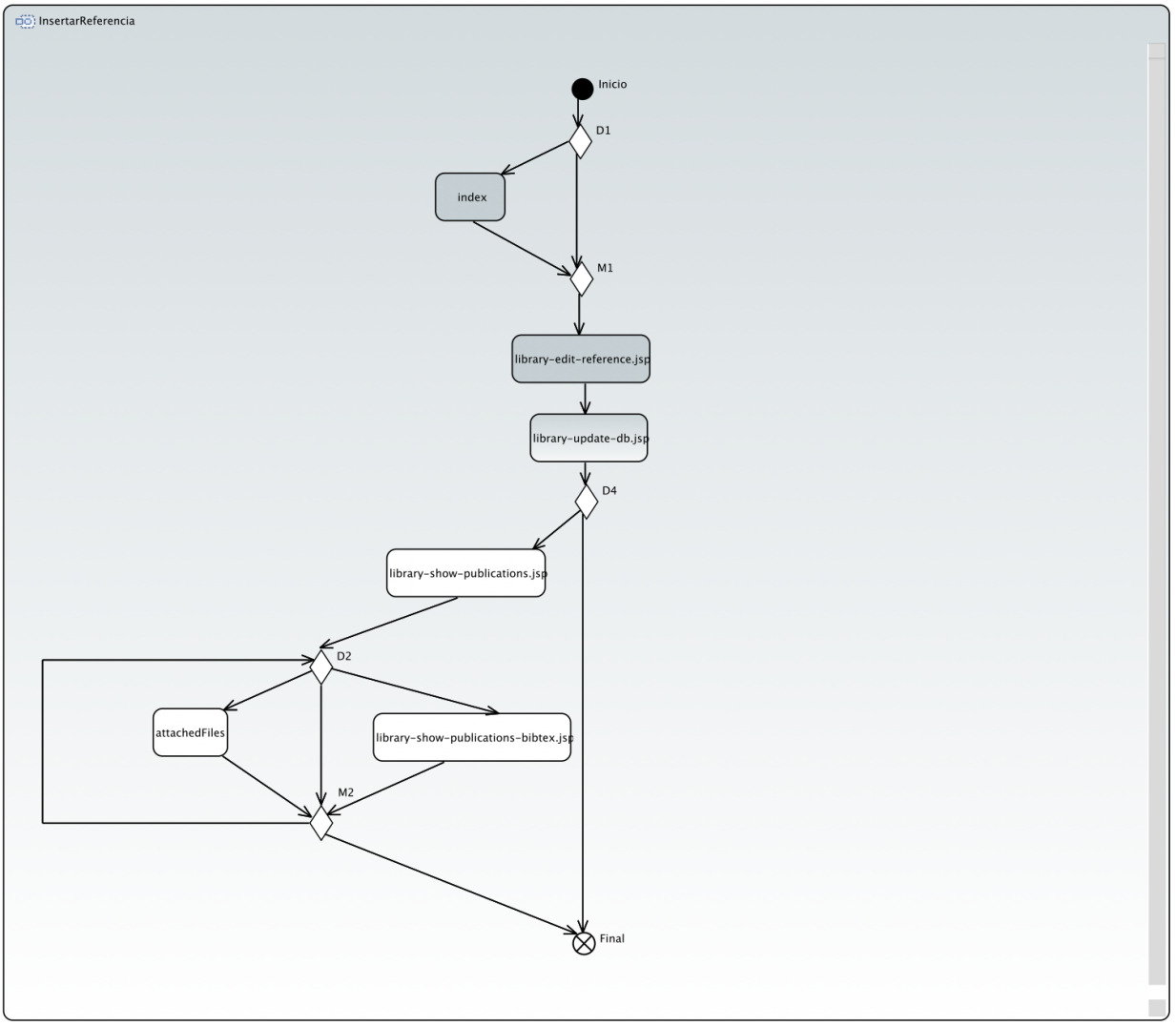


Figura 25: Diagrama de actividad (Insertar Referencia).

Diagrama de actividad (Obtener todas publicaciones en BibTeX)

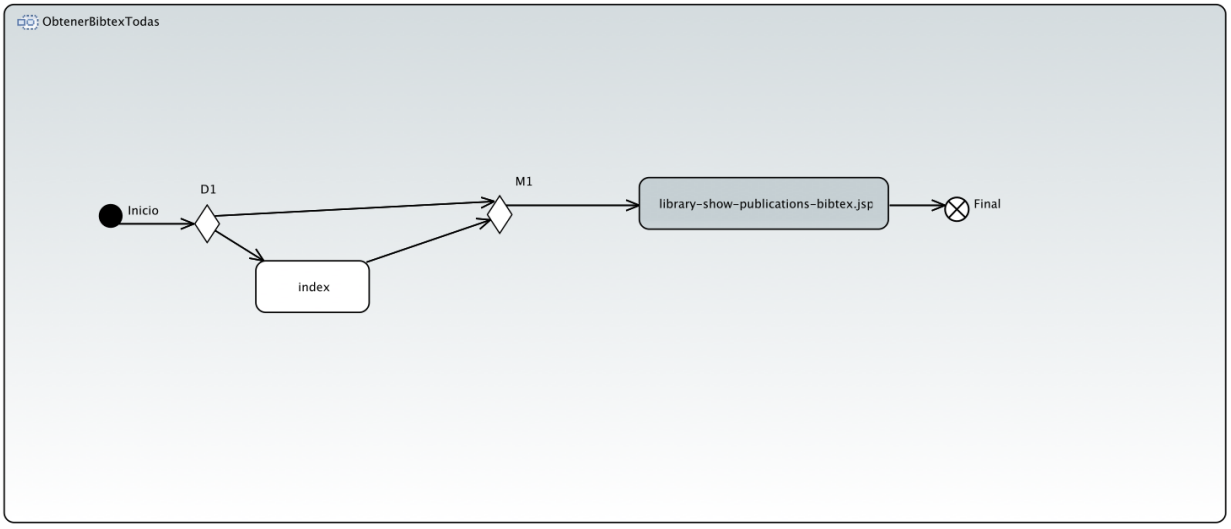


Figura 26: Diagrama de actividad (Obtener todas publicaciones en BibTeX).

Diagrama de actividad (Tareas Pendientes)

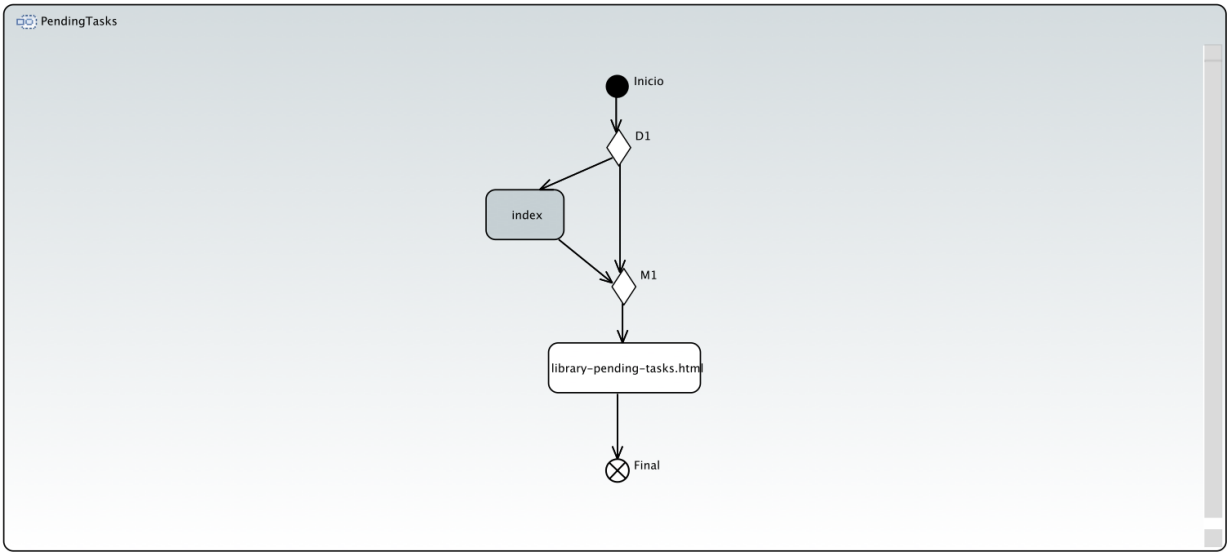


Figura 27: Diagrama de actividad (Tareas Pendientes).

Diagrama de actividad (Ayuda)

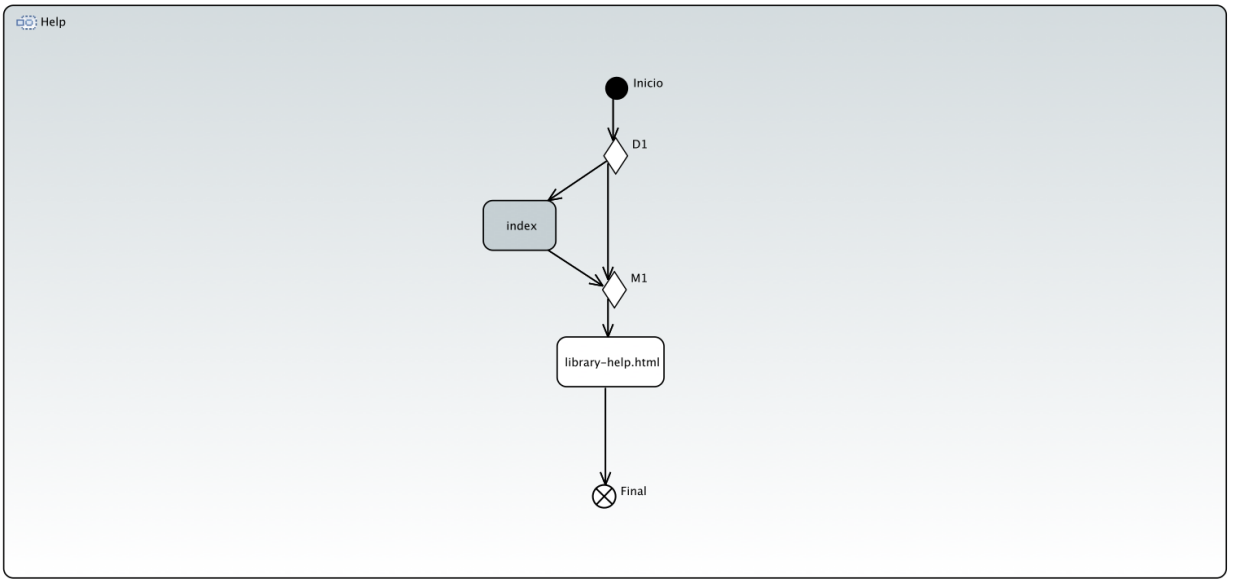


Figura 28: Diagrama de actividad (Ayuda).

Diagrama de actividad (Contenido Estático)

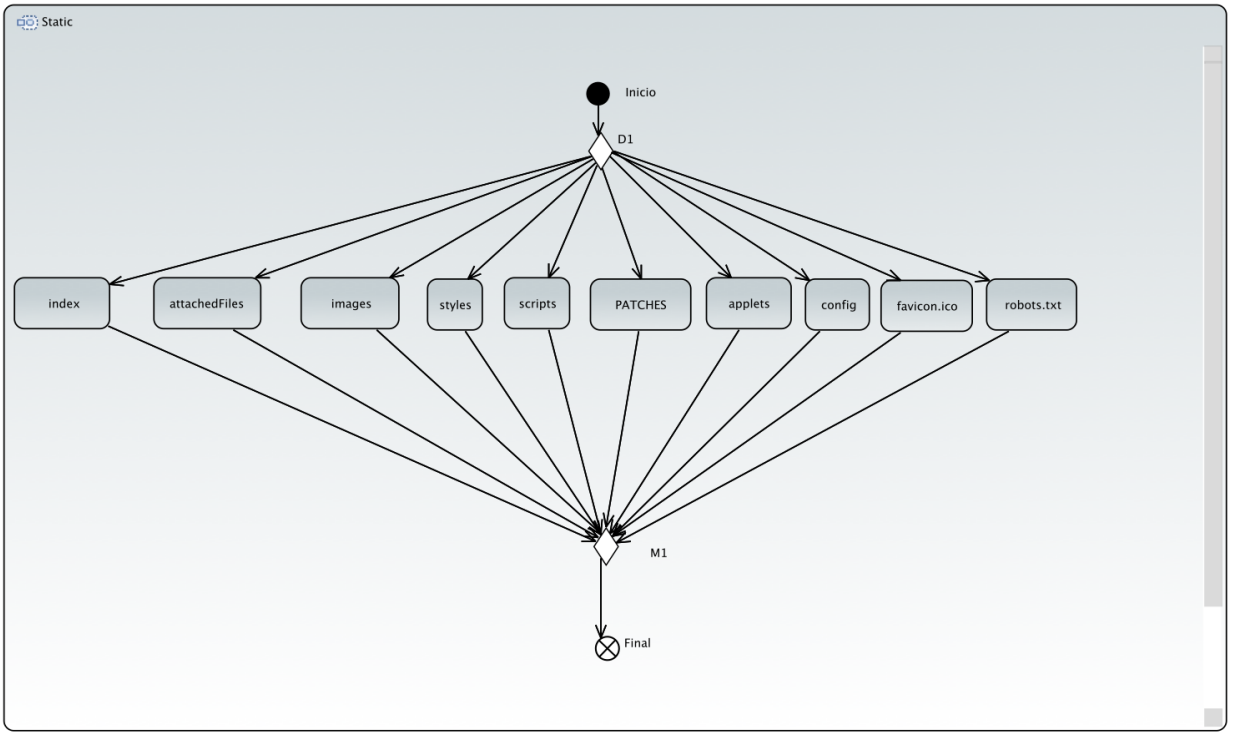


Figura 29: Diagrama de actividad (Contenido Estático).

D. Trabajo efectivamente realizado, horas invertidas y coste estimado del proyecto

En este anexo, se presenta el desglose del trabajo efectivamente realizado completo, siguiendo la metodología Scrum. Mediante la aplicación de esta metodología ágil, ha sido posible llevar a cabo un desarrollo del proyecto de manera incremental, con reuniones semanales, con la posibilidad de solapamiento entre tareas; que ha llevado a la validación y mejora progresiva de las tareas realizadas del proyecto durante el desarrollo del mismo. Además ha permitido ahorrar tiempo de trabajo y anticiparse a algunos problemas. En concreto, el proceso que se ha seguido es el siguiente:

- En primer lugar, se establecen los requisitos del trabajo que se va a realizar, analizando y estudiando el contexto del proyecto.
- En segundo lugar, se definen las tareas a desarrollar a lo largo de la duración del proyecto, dividiendo las mismas por partes, si se trata de tareas de larga duración (más de una semana).
- En tercer lugar, se producen entregables según las tareas definidas y se realizan reuniones semanales para evaluar los resultados obtenidos hasta el momento, y la calidad y validez de los mismos.
- Por último, este proceso se repite hasta la consecución total del proyecto.

Las tareas en concreto que se han realizado, se pueden observar en la siguiente tabla. Para cada tarea se indica su fecha de realización y el total de horas invertidas en ella:

Cuadro 1: Desglose de tareas, fechas y horas invertidas

Etiqueta	**Descripción**	**Fecha**	**Duración (H)**
Organización	Organización, gestión y preparación del trabajo.	2/8/2016	0,5000
Documentación	Lectura de bibliografía relacionada con la minería de procesos. Libro recomendado. Cinco primeros capítulos.	2/8/2016	5,0000
		2/9/2016	5,0000
		2/10/2016	5,0000
		2/11/2016	5,0000
		2/12/2016	5,0000
Instalación	Instalación de ProM, Papyrus, Sepia, SecSy, Anica, Wireshark. Además de realización de pruebas.	2/15/2016	8,0000
Diagramas UML	Creación de diagrama UML de casos de uso.	2/16/2016	2,0000

Instalación	Instalación de dos máquinas virtuales en VirtualBox (con Vagrant), con el SW de BiD. Tomcat 7, Java 6, MySQL.	2/17/2016	1,5000
Diagramas UML	Creación de diagramas UML de clase, actividad y secuencia.	2/17/2016	1,5000
		2/18/2016	4,0000
		2/19/2016	4,0000
		2/20/2016	3,0000
Instalación	Instalación en una MV con VirtualBox y Vagrant del SGBD MySQL del BiD a partir de un dump de la BBDD en producción. Conexión entre sistemas y comprobación de funcionamiento.	2/20/2016	4,5000
Memoria	Realización del anexo A. Resumen del libro Process Mining (primeros 5 capítulos).	2/24/2016	1,0000
		2/28/2016	4,2800
		2/29/2016	7,5000
Memoria	Creación de plantilla LaTeX para memoria y documentación.	2/29/2016	1,8000
Documentación	Lectura de normativa sobre el Trabajo de Fin de Grado.	3/1/2016	0,5000
Descubrimiento de procesos	Realización de tarea de descubrimiento de procesos con el algoritmo alfa, sobre logs de Tomcat (local).	3/1/2016	5,0000
Solución de problemas	Instalación de ProM en una MV con Windows 10 (ya que funciona mejor). A su vez se ha intentado, arreglar en Mac, mejorando ligeramente su funcionamiento.	3/3/2016	1,0000
Memoria	Transformar anexo del resumen de Process Mining de Word a LaTeX.	3/3/2016	1,0000
Seguridad	Escaneo de vulnerabilidadesen el sistema BiD.	3/6/2016	0,7300
Logs	Formateo de logs de Tomcat con ficheros de configuración y script realizado en Python.	3/7/2016	3,2330
Descubrimiento de procesos	Realización de pruebas de descubrimiento de procesos con el algoritmo alfa, sobre logs de Tomcat (local y formateados). Solución de problemas con Windows.	3/7/2016	1,6330
Diagramas UML	Creación de diagramas UML de clase, actividad y secuencia.	3/8/2016	6,5670

Análisis / Documentación	Documentación y análisis de Sepia, SecSy, Anica, Wireshark. Documentación y análisis de servidores J2EE.	3/9/2016	2,5500
Análisis / Documentación	Documentación y análisis de Sepia, SecSy, Anica, Wireshark.	3/10/2016	2,6330
		3/11/2016	0,5000
Análisis / Documentación	Documentación y análisis de servidores J2EE.	3/11/2016	0,8000
		3/12/2016	1,9500
Documentación / Definición	Definición de heurísticas para sesiones de usuario y casos de uso.	3/12/2016	1,2830
Descubrimiento de procesos	Realización de pruebas de descubrimiento de procesos con el algoritmo alfa, sobre logs de Tomcat (dados/remotos).	3/12/2016	1,0000
Logs / Desarrollo	Implementación de heurísticas de sesión de usuario y casos de uso sobre logs con herramienta en Python.	3/15/2016	2,7170
		3/16/2016	1,7830
		3/16/2016	2,8170
		3/21/2016	2,0000
Minería de procesos / Diagramas UML	Generación de diagramas UML de actividad, transformación a redes de Petri, y análisis de redes.	3/21/2016	8,5000
		3/28/2016	4,2500
Logs / Desarrollo	Implementación de heurísticas de casos de uso sobre logs con herramienta en Python.	3/29/2016	2,3500
Investigación / Desarrollo	Investigación obtención “Case-Id” elaborando ciertas heurísticas, analizando logs y comportamientos. Investigación y lectura de temas relacionados.	3/30/2016	5,2830
Instalación	Instalación de dos máquinas virtuales en VirtualBox, con el SW GreatSPN. Realización de pruebas y conexión.	3/31/2016	1,5000
Logs / Desarrollo	Filtrar parámetros URL de los logs los cuales no identifican casos de usos (para mejor RdP). Filtrar logs antiguos (construcción) no “útiles”.	4/4/2016	4,3000
Descubrimiento de procesos	Realización de pruebas de descubrimiento de procesos, sobre logs de Tomcat (dados/remotos).	4/5/2016	4,0000

Minería de procesos / Diagramas UML	Generación de diagramas UML de actividad, transformación a redes de Petri, y análisis de redes.	4/6/2016	2,5000
		4/12/2016	3,0300
Logs / Desarrollo	Filtrar parámetros URL de los logs los cuales no identifican casos de usos (para mejor RdP).	4/12/2016	0,7856
Minería de procesos / Conformance Checking	Realización de "replays" sobre la RdP generada, para calcular su "fitness" y similitud con la realidad.	4/12/2016	1,9390
Descubrimiento de procesos	Realización de pruebas de descubrimiento de procesos, sobre logs de Tomcat (dados/remotos).	4/12/2016	1,2320
Documentación / Investigación	Investigación sobre APT. Ejemplos, casos, soluciones, usos, etc.	4/12/2016	1,0000
Documentación / Definición	Documentación y definición de heurísticas para la identificación de casos de uso en logs (dentro de sesiones de usuario).	4/14/2016	1,0000
Documentación / Presentación	Presentación para reunión orientativa y obtención de retroalimentación. (Objetivo, problemas, realizado).	4/21/2016	2,0000
Descubrimiento de procesos	Realización de pruebas de descubrimiento de procesos, sobre logs de Tomcat (dados/remotos).	5/17/2016	5,0000
		5/18/2016	5,0000
Memoria	Estructuración de la memoría, con las recomendaciones de TFG.	5/20/2016	2,0000
Conformance & Performance checking	Realización de "conformance & performance checking" con RdP normativa y logs con casos de uso y sesión (incluyendo todo, ataques).	5/21/2016	8,0000
Conformance & Performance checking / Descubrimiento de procesos	Realización de conformance & performance checking con RdP normativa y logs con casos de uso y sesión (incluyendo todo, ataques). Además, se realizan pruebas con RdP operativa para obtener "Fuzzy Nets" y su animación.	5/22/2016	12,0000
Documentación	Documentación de todos los experimentos realizados en las anteriores tareas.	5/23/2016	5,0000

Memoria	Realización del anexo B. Resumen de utilización de herramienta de conversión de UML a RdP.	5/23/2016	4,0000
Seguridad / SIEM	Instalación de Zabbix en sistema de la MV y gestión y monitorización del sistema.	5/24/2016	10,0000
Artículo DESEID 2016	Creación de cuenta en Easychair e introducción del artículo.	5/25/2016	1,5000
Memoria	Desarrollo de la memoria del TFG	5/26/2016	5,0000
Artículo DESEID 2016	Desarrollo y estructura del artículo.	5/27/2016	6,8500
		5/30/2016	5,0000
	Desarrollo, corrección y envío del resumen.	5/30/2016	2,0000
Memoria	Desarrollo y estructura de la memoria.	5/31/2016	8,0000
		6/1/2016	3,0000
Experimentación y realización pag. Web	Descripción y disponibilidad de los resultados obtenidos y su validación. Realización de una página Web con todos los recursos obtenidos.	6/1/2016	5,2000
		6/2/2016	5,0000
Memoria	Desarrollo y estructura de la memoria.	6/2/2016	5,6330
		6/3/2016	6,0000
		6/4/2016	7,0000
		6/5/2016	5,0000
		6/6/2016	5,0000
		6/7/2016	8,0000
		6/8/2016	5,0000
		6/9/2016	5,0000
		6/15/2016	5,0000
		6/16/2016	8,0000
		6/17/2016	8,0000
		6/18/2016	8,0000
		6/19/2016	8,0000
		6/20/2016	8,0000
		6/21/2016	8,0000
		6/22/2016	8,0000

A modo de conclusión y resumen de lo presentado en el cuadro 1, cabe destacar las horas totales invertidas en la realización del proyecto, y el coste estimado del mismo. Estos aspectos se pueden ver reflejados en la siguiente tabla:

Cuadro 2: Horas invertidas y coste estimado de proyecto.

	Horas Invertidas	**Estimación coste de proyecto**
Completa realización del trabajo	374,1286	3.741,2860 € $\sim$ 5.611,9290 €

La estimación del coste de proyecto reflejada en el cuadro 2, se ha calculado considerando un coste de 10-15€ por hora trabajada (lo cual es el promedio actual para un desarrollador *junior*).

Referencias

[1] Van Der Aalst, W. (2011). *Process mining: discovery, conformance and enhancement of business processes.* Springer Science & Business Media.

[2] Silva, M. (1985). *Las Redes de Petri en la Informática y la Automática.* Editorial AC. Madrid.

[3] Uml, O. M. G. (2004). *2.0 Superstructure Specification.* OMG, Needham.

[4] Schmidt, D. C. (2006). *Model-driven engineering.* COMPUTER-IEEE COMPUTER SOCIETY-, 39(2), 25.

[5] DisCo. (2016). *Grupo de I+D en Computación Distribuida (DisCo).* Accedido el 31 de Mayo, 2016, desde http://www.discouz.es/

[6] SID Research Group. (2010). *Research group of distributed information systems (SID).* Accedido el 31 de Mayo, 2016, desde http://sid.cps.unizar.es/BiD/

[7] Hand, D. J., Mannila, H., & Smyth, P. (2001). *Principles of data mining.* MIT press.

[8] Object Management Group (OMG). *Model-driven Architecture - MDA*. OMG Initiative.

[9] Tolvanen, J. P., & Kelly, S. (2008). *Domain-Specific Modeling: Enabling Full Code Generation.* Wiley-IEEE Computer Society, 444, 231.

[10] Völter, M., Stahl, T., Bettin, J., Haase, A., Helsen, S., & Czarnecki, K. (2006). *Model-Driven Software Development: Technology. Engineering, Management.* Wiley, 5, 6.

[11] Balmelli, L., Brown, D., Cantor, M., & Mott, M. (2006). *Model-driven systems development.* IBM Systems Journal, 45(3), 569.

[12] Zabbix LLC. (2016). *Zabbix: The Enterprise-class Monitoring Solution for Everyone.* Accedido el 31 de Mayo, 2016, desde http://www.zabbix.com/

[13] Wireshark Foundation. *Wireshark.* Accedido el 3 de Junio, 2016, desde https://www.wireshark.org/

[14] Apache. *Apache Tomcat.* Accedido el 3 de Junio, 2016, desde https://tomcat.apache.org/

[15] Java, E. E. (2006). *Java EE at a Glance.* 2008-06-17. http://java.sun.com/javaee/index.jsp.

[16] The Eclipse Foundation. *Papyrus Modeling environment.* Accedido el 3 de Junio, 2016, desde https://eclipse.org/papyrus/

[17] Abel Gómez, Christophe Joubert, José Merseguer. (2016). *A Tool for Assessing Performance Requirements of Data-Intensive Applications.* XXIV Jornadas de Concurrencia y Sistemas Distribuidos (JCSD 2016).

[18] Casale, G., Ardagna, D., Artac, M., Barbier, F., Nitto, E. D., Henry, A., ... & Pérez, J. F. (2015, May). *DICE: quality-driven development of data-intensive cloud applications.* In Proceedings of the Seventh International Workshop on Modeling in Software Engineering (pp. 78-83). IEEE Press.

[19] Kindler, E. (2011). *The ePNK: an extensible Petri net tool for PNML.* In Applications and theory of Petri nets (pp. 318-327). Springer Berlin Heidelberg.

[20] Python Software Fundation. *Python Programming Language.* Accedido el 3 de Junio, 2016, desde https://www.python.org/

[21] Verbeek, H. M. W., Buijs, J. C. A. M., Van Dongen, B. F., & van der Aalst, W. M. (2010). *Prom 6: The process mining toolkit.* Proc. of BPM Demonstration Track, 615, 34-39.

[22] Maggi FM, Mooij AJ, van der Aalst WMP. (2013). *Analyzing vessel behavior using process mining.* Springer.

[23] Hernandez S, Zelst S, Ezpeleta J, van der Aalst W. (2015). *Handling big (ger) logs: Connecting ProM 6 to apache hadoop.* In Proceedings of the BPM2015 Demo Session, ser. CEUR Workshop Proceedings (Vol. 1418, pp. 80-84).

[24] Dittrich, J., & Quiané-Ruiz, J. A. (2012). *Efficient big data processing in Hadoop MapReduce.* Proceedings of the VLDB Endowment, 5(12), 2014-2015.

[25] S. Bernardi, J.I. Requeno, C. Joubert AR. (2016). *A Systematic Approach for Performance Evaluation using Process Mining: the POSIDONIA Operations Case Study.* In Proceedings of the 2nd International Workshop on Quality-aware DevOps (QUDOS).

[26] Accorsi, R., Lehmann, A., & Lohmann, N. (2015). *Information leak detection in business process models: Theory, application, and tool support.* Information Systems, 47, 244-257.

[27] Accorsi, R., Stocker, T., & Müller, G. (2013, March). *On the exploitation of process mining for security audits: the process discovery case.* In Proceedings of the 28th Annual ACM Symposium on Applied Computing (pp. 1462-1468). ACM.

[28] Bezerra, F., & Wainer, J. (2013). *Algorithms for anomaly detection of traces in logs of process aware information systems.* Information Systems, 38(1), 33-44.

[29] Bezerra, F., Wainer, J., & van der Aalst, W. M. (2009). *Anomaly detection using process mining.* In Enterprise, Business-Process and Information Systems Modeling (pp. 149-161). Springer Berlin Heidelberg.

[30] Stocker, T., & Accorsi, R. (2014). *SecSy: A Security-oriented Tool for Synthesizing Process Event Logs.* In BPM (Demos) (p. 71).

[31] Shafranovich, Y. (2005). *Common format and MIME type for Comma-Separated Values (CSV) files.*

[32] Hillah, L. M., Kordon, F., Petrucci, L., & Trèves, N. (2010). *PNML framework: An extendable reference implementation of the Petri Net Markup Language.* In Applications and Theory of Petri Nets (pp. 318-327). Springer Berlin Heidelberg.