Universidad de las Ciencias Informáticas

Facultad 3

Título: Componente para la transformación y extracción de trazas de Bosón para aplicar técnicas de Minería de procesos

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor(es): Pavel Eduardo Barzaga De la Paz

Tutor(es): Abraham Calás Torres

Junio del 2016

Declaración de autoría

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Pavel Eduardo Barzaga De la Paz	Abraham Calás Torres
_____	_____
Firma del Autor	Firma del Tutor

DATOS DE CONTACTO

Nombre y apellidos: Abraham Calás Torres

Correo: acalas@uci.cu

Síntesis del Tutor: Es un miembro activo del grupo de investigación de minería de procesos. Formó parte de una investigación para una tesis doctoral sobre un método para el diagnóstico de procesos de negocio a partir de registros de eventos con ruido y ausencia de información. Actualmente es tutor de dos tesis de grado y consultante en una, todas relacionadas con la extracción de trazas de procesos. Adicionalmente completó satisfactoriamente el curso online: Process Mining: Data Science in action, dictado por uno de los principales autores del área.

AGRADECIMIENTOS

Agradezco a mis padres Yenis y Pavel, a mi padrastro Eliéser y tío Libán, por todo lo que han sacrificado por mí, por brindarme siempre su apoyo y por convertirme en el hombre que soy gracias a sus actos y su ejemplo.

A mi enorme, gigantesca y cariñosa familia por siempre apoyarme y amarme, sus ánimos me impulsan a salir adelante.

A Madelín y a Rosalina porque más que amigas han sido también mis madres aquí en la Universidad.

A Kireida, María del Pilar, Thais, Liz Verónica, Claudia, Célida, Mireyita y Yanet por brindarme amor y cariño cuando más falta me hacía.

A mis mejores amigos Luis Fidel, Robertico, Tato, Antonio, El Chino, Leduán y Daniel, por siempre ayudarme incluso en mis momentos más insoportables.

Al grupo editorial de Dragones: Raynel, Polanco, Laynier, Odette, Celia, Ricardo, Abel, Lilian, Chirino y el gran Fabra, por haberme dado la oportunidad y la enorme responsabilidad de escribir para la comunidad, ustedes han sido también mi familia.

A mi tutor Abraham por su invaluable ayuda tanto en la investigación como en la programación.

Al personal del Departamento de desarrollo de Componentes, especialmente a René, Daniel Arturo, Julio César, Katia y Claudia por enseñarme a trabajar y apoyarme constantemente.

A Damián, por haberme ayudado con sus correcciones y sugerencias a mejorar mi investigación.

A todas las personas, presentes y ausentes, que siempre me apoyaron y me animaron a vencer la carrera.

DEDICATORIA

Dedico esta tesis principalmente a mis abuelos: Chela, Bebo, Melbys y Jorge. Ustedes siempre están presentes en mi corazón.

A mis padres y a mi familia, las personas que más me aman.

A mis niñas y mis hermanos del alma.

A todas mis amistades, tanto en casa como aquí en la Universidad.

Por último pero no menos importante, a Kireida.

RESUMEN

Bosón es una arquitectura de referencia creada para construir sistemas web de gestión. Estos tipos de sistemas generalmente almacenan trazas de las acciones ejecutadas por los usuarios. Para obtener información útil de las trazas se pueden aplicar técnicas y herramientas de Minería de procesos. La premisa para aplicar esta disciplina es la confección del registro de eventos, tarea que puede complejizarse en dependencia del tipo de sistema, como en el caso de Bosón que está enfocado hacia los datos. Por tanto el objetivo de la investigación es obtener un registro de eventos factible a partir de sistemas orientados a datos. Se realizó un estudio para determinar los eventos que componen el registro así como la información que deben contener los eventos. Como resultado, con la implementación de entidades y clases en el Componente de trazas de Bosón se realizó la captura de eventos y se confeccionó el registro. Se exportó el registro al estándar XES, reconocido por las principales técnicas y herramientas para realizar análisis de Minería de procesos. Con esta solución los sistemas construidos sobre Bosón cuentan con un instrumento para apoyarse en el proceso de la toma de decisiones.

Palabras clave: registro; evento; proceso; minería.

Abstract

Bosón is a reference architecture created to build web management systems. These types of systems generally store traces of actions carried out by users. For gathering useful information from traces, techniques and tools of processes mining can be applied. The premise for applying this discipline is the preparation of the event log, a task that can become complex depending on the type of system, as in the case of Bosón that is focused on the data. Therefore, the objective of the research is to obtain a feasible event log from data-oriented systems. A study was performed to determine the events to conform the log and the information to be contained within events. As a result, with the implementation of entities and classes in the Trace Component of Boson, event were captured and the log was created. The event log was exported to XES standard, recognized by the main techniques and tools for applying process mining. With this solution the systems built on Boson have a tool to support the process of decision making.

Keywords: *log; event; process; mining.*

INDICE

INDICE DE FIGURAS

INDICE DE TABLAS

INTRODUCCIÓN

Las organizaciones trabajan con grandes cantidades de datos. Los datos son valores básicos o hechos que están almacenados en bases de datos. Se piensa frecuentemente en los datos como sinónimo de información; sin embargo, la información consiste en datos que han sido organizados para ayudar a responder preguntas y resolver problemas. Un sistema de información se define como el software que ayuda a organizar y analizar los datos. Entonces su propósito es convertir los datos primarios en información útil que pueda ser empleada en la toma de decisiones en una organización (Stu15).

En el Centro de Informatización de Entidades de la Universidad de las Ciencias Informáticas (UCI) se desarrolla un marco de trabajo (*framework*) denominado Bosón. Este es una implementación de una arquitectura de referencia para construir sistemas de información orientados a la web, debido a que especifica las buenas prácticas para el desarrollo así como la definición de patrones y tecnologías a utilizar. Está formado por componentes desarrollados sobre Symfony 2 y estos responden a la mayoría de los requisitos tecnológicos de un amplio espectro de aplicaciones. Bosón se presenta como la solución que establecerá un formato de trabajo común entre las diversas soluciones de desarrollo, al aportarles componentes previamente construidos y listos para ser utilizados (Torres, 2014).

En un universo digital en constante expansión y cambio, multitudes de eventos son registrados por los sistemas de información modernos, sin embargo las organizaciones tienen problemas cuando quieren obtener valor de esos datos. La Minería de procesos emplea los datos para extraer información relacionada a los procesos, o sea, para descubrir automáticamente un modelo de procesos mediante la observación de los eventos registrados por un sistema empresarial. El objetivo de la Minería de procesos es descubrir, monitorizar y mejorar procesos reales (entiéndase procesos no asumidos) mediante la extracción del conocimiento de registros de eventos disponibles en los sistemas (Aalst, 2011). Su aplicación aprovecha los datos de eventos en una forma significativa, por ejemplo, para proveer un mejor entendimiento, identificar cuellos de botella, anticipar problemas, registrar violaciones de políticas, recomendar contramedidas, y simplificar procesos (Mining, 2011).

El punto de partida de la Minería de procesos es un registro de eventos. En los sistemas informáticos se confecciona el registro a partir de las trazas que generan y se convierte al estándar *eXtensible Event Stream* (XES) o Flujo Extensible de Eventos, desarrollado por el grupo de investigación de la Universidad de Eindhoven. Dicho estándar es comprendido por las

herramientas y técnicas de minería. Las dificultades para conseguir el registro surgen cuando se analiza el tipo de enfoque que posee el sistema informático. En el caso de Bosón se construyen sistemas orientados a los datos. Este enfoque facilita la ejecución de tareas específicas pero no tiene conocimiento del proceso al cual pertenecen dichas tareas (Dumas, y otros, 2005). La mayoría de estos sistemas almacenan sus datos sin una estructura o están esparcidos entre muchas tablas. En tales casos, los datos de los eventos existen pero se requiere algo de esfuerzo para obtenerlos (Aalst, 2011). En cambio, los sistemas orientados a los procesos (PAIS por sus siglas en inglés) son sistemas de software que gestionan y ejecutan procesos operacionales que involucran personas, aplicaciones y fuentes de información en las bases de los modelos de procesos (Dumas, y otros, 2005). Los PAIS generalmente producen trazas de gran calidad y completitud. Actualmente resulta muy costoso convertir a Bosón en un PAIS por tanto se debe hallar una manera de componer un registro de eventos desde el enfoque de datos.

Los eventos almacenados en el registro necesitan estar organizados y clasificados para ser interpretados por las diversas herramientas y algoritmos. Deben estar relacionados con un proceso, poseer la fecha y hora de su ocurrencia y un nombre o descripción. Esta información se obtiene a partir de las trazas de proceso, en el caso de Bosón se guardan trazas de acciones, excepciones y acceso a datos. Desafortunadamente ninguna brinda suficiente información sobre los eventos y sus datos están desorganizados. Tampoco existe una noción de proceso en las trazas, esta carencia implica que actualmente no se puede obtener un registro de eventos en Bosón.

Existen una serie de problemas relacionados con la extracción de trazas y atentan contra la calidad del registro de eventos. Entre ellos están la divergencia, convergencia, alta granularidad, errores en la fecha y hora o presencia de eventos incompletos. Si no se detectan y resuelven, influyen negativamente en los análisis que se ejecuten sobre el registro. Los modelos de proceso se tornan complejos innecesariamente y no concuerdan con la ejecución real de los casos. Esto implica un incorrecto descubrimiento del flujo de actividades en la organización y la información no es confiable para apoyar los procesos de toma de decisiones. Las técnicas de Minería de procesos se ven afectadas cuando deben manejar información incompleta (Bose, y otros, 2013). Estos problemas deben ser analizados para determinar cuántos están presentes en el caso de Bosón y cómo solucionarlos o minimizar su impacto, de manera que el registro de eventos obtenido sea lo más cercano a la realidad posible.

Por lo antes expuesto se plantea el siguiente **Problema a resolver:** Las trazas generadas por el marco de trabajo Bosón dificultan su transformación en registros de eventos factibles.

Se delimita como **Objeto de estudio** la Minería de procesos teniendo como **campo de acción** el Registro de trazas para la Minería de procesos.

Para solucionar los problemas detectados se plantea como **Objetivo General**: Desarrollar una solución informática que permita la extracción y transformación de las trazas registradas por Bosón para realizar análisis de los procesos ejecutados utilizando técnicas de Minería de procesos.

Para satisfacer el objetivo general se definieron los siguientes **Objetivos específicos**:

- Confeccionar el marco teórico conceptual de la investigación a partir de una búsqueda y revisión bibliográfica para definir los datos a registrar a partir de la definición de XES, sus extensiones y los posibles análisis de Minería de procesos a ejecutar.
- Realizar el análisis y diseño del componente para la transformación y extracción de las trazas registradas por Bosón.
- Desarrollar un componente para la transformación de trazas de proceso relacionados a los datos de los sistemas de información no orientados a procesos.
- Realizar la exportación de las trazas a formato XES.
- Validar la extracción de trazas mediante experimentación.
- Validar la solución propuesta mediante pruebas de caja blanca y caja negra.

Idea a defender

Si se desarrolla un componente para registrar y transformar las trazas generadas por el marco de trabajo Bosón se podrán aplicar técnicas de Minería de procesos sobre las mismas.

Posibles resultados

- Definición de trazas a registrar a partir de la definición de XES, sus extensiones y posibles análisis de Minería de procesos a ejecutar.
- Procedimiento para el registro de trazas de proceso en sistemas orientados a datos, construidos sobre Bosón.

El presente trabajo está estructurado en tres capítulos:

En el capítulo primero, “Fundamentación teórica”, se describen los principales conceptos teóricos del objeto de estudio. Se detalla el entorno donde se desarrolla la problemática y se analizan las características y ventajas de las soluciones implementadas. Se listan las herramientas y tecnologías para el desarrollo de la solución.

En el capítulo segundo, “Propuesta de solución” se describe detalladamente la solución a implementar. Se justifica la selección de patrones y modelos para el diseño de la solución. Se enuncian los requisitos funcionales y no funcionales.

En el capítulo tercero, “Implementación y pruebas”, se valida el diseño de la solución. Se describen los estándares de codificación utilizados, la nomenclatura del código y métodos implementados. Se analizan los resultados de las pruebas aplicadas para validar la investigación.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

En este capítulo se estudian los conceptos fundamentales relacionados con los registros de eventos. Se analizan los atributos necesarios para construir el registro, así como los problemas que presentan los datos. Se investigan las herramientas existentes para la confección de registros de eventos a partir de varias fuentes de datos. Finalmente se definen las herramientas y tecnologías para la confección de la solución.

1.1 Conceptos fundamentales

1.1.1 Minería de procesos

La Minería de procesos consiste en técnicas, herramientas y métodos para descubrir, monitorear y mejorar los procesos reales a través de la extracción de conocimiento de los registros de eventos, ampliamente disponibles en los actuales sistemas de información (Aalst, 2011).

El descubrimiento de procesos, el chequeo de conformidad y el mejoramiento de modelos, son los tres tipos básicos de Minería de procesos. El descubrimiento de procesos permite obtener un modelo de proceso a partir de un registro de eventos. A su vez el chequeo de conformidad analiza si la realidad (según consta en un registro de eventos) se ajusta al modelo y viceversa. Su objetivo es detectar las discrepancias y medir su gravedad. Por otra parte el mejoramiento de modelos permite extender o mejorar los procesos a partir de la información extraída de un registro de eventos. Por ejemplo, se pueden identificar cuellos de botella reproduciendo un registro de eventos en un modelo de proceso, mientras se examinan las marcas de tiempo (Mining, 2011). Para el caso de Bosón se configurarán los procesos para registrar las trazas con atributos que permiten obtener un registro de eventos sobre el cual aplicar los tipos de minería.

Se considera como proceso a un conjunto de actividades mutuamente relacionadas o que interactúan, las cuales transforman elementos de entrada en resultados. Un caso o instancia de proceso, es la entidad siendo ejecutada por el proceso que es analizado. Los eventos se relacionan con instancias del proceso. Una actividad se entiende como un paso bien definido en el proceso (Mining, 2011).

1.1.2 Registro de eventos

El registro de eventos es la colección utilizada como entrada para aplicar la Minería de procesos. Los eventos no necesitan ser almacenados en un archivo de registro por separado (por ejemplo, los eventos pueden estar dispersos en diferentes tablas de bases de datos). Los eventos se

corresponden con las acciones almacenadas en el registro como el inicio, conclusión o cancelación de una actividad para una instancia particular de un proceso (Mining, 2011).

Las técnicas de Minería de procesos requieren como entrada un registro de eventos “plano” similar al de la Tabla 1 Estructura básica del registro de eventos en los que cada uno tiene las siguientes propiedades:

- Identificador del caso: cada evento debe referirse a un caso (es decir, la instancia de proceso).
- Actividad: cada evento debe estar relacionado con una actividad.
- Marca de tiempo: son necesarias para determinar el orden de ocurrencia de los eventos, y vitales para medir el desempeño del caso o proceso.

Un conjunto de atributos de eventos opcionales pueden ser adicionados, por ejemplo:

- Recursos: la persona, equipo o componente de software que ejecuta el evento.
- Tipo: el tipo de transacción del evento (iniciado, completado, suspendido, reanudado).
- Costos: los costos asociados al evento.
- Cliente: información sobre la persona u organización que ejecuta o para quien se ejecuta el evento.

La Tabla 1 muestra los datos básicos de un registro de eventos para poder aplicar Minería de procesos.

Tabla 1 Estructura básica del registro de eventos

Caso	**Id**	**Nombre**	**Fecha**	**Recurso**	**Costo**
1	123	Agregar usuario	2015-04-23 16:44:50	Javier	20
1	124	Comprobar credenciales	2015-04-23 16:50:00	Javier	40
2	125	Agregar usuario	2015-04-24 08:30:25	Rosa	20
1	126	Proveer autorización	2015-04-24 09:40:32	Javier	20

2	127	Comprobar credenciales	2015-04-24 12:20:00	Rosa	40

1.1.3 Enfoques de los sistemas de información

Un sistema de información es un tipo particular de sistema de trabajo que utiliza la tecnología de información para captar, transmitir, almacenar, recuperar, manipular o mostrar información, apoyando así uno o más sistemas de trabajo. Esta definición utiliza dos términos clave: tecnologías de la información y sistema de trabajo. La tecnología de la información se define como “el hardware y el software utilizado para almacenar, recuperar y transferir información”, y un sistema de trabajo como “un sistema en el que los participantes humanos realizan un proceso de negocio utilizando la información, la tecnología y otros recursos para producir productos para los clientes internos” (Dumas, y otros, 2005).

La importancia de los sistemas de información no sólo se refleja en el crecimiento exponencial de los datos, sino también por el papel que estos sistemas desempeñan en los procesos de negocio de hoy en día, a medida que el universo digital y el universo físico se alinean cada vez más. Por ejemplo, el “estado de un banco” está determinado principalmente por los datos almacenados en el sistema de información del banco. Los datos registrados por los sistemas de información se pueden utilizar para proporcionar una mejor visión de los procesos reales, es decir, las desviaciones pueden ser analizadas y la calidad de los modelos se puede mejorar (Aalst, 2011).

Los sistemas de información tienen dos enfoques: consciente de los datos y consciente de los procesos. Desde finales del 1970 hasta principios de 1990, el centro de la atención en el área de sistemas de información fue hacia los datos. La atención se centró principalmente en el almacenamiento y recuperación de la información y, por lo tanto, los modelos de datos eran a menudo el punto de partida para el diseño de sistemas de información, mientras que los sistemas de gestión de bases de datos fueron considerados como el corazón de la infraestructura de tiempo de ejecución. Durante la década de 1990, una serie de tendencias paralelas cambió el enfoque hacia los procesos. Como resultado, un número cada vez mayor de los procesos de negocio ahora se lleva a cabo bajo la supervisión de los sistemas de información impulsados por modelos de procesos explícitos (Dumas, y otros, 2005).

Por otro lado los PAIS se utilizan principalmente para apoyar la ejecución de los procesos de negocio. Dentro de un PAIS el proceso se define como un modelo que se puede ver y cambiar. Esto permite la modificación de la definición del proceso sin la necesidad de modificar el código

de la aplicación. Los PAIS admiten la promulgación automática de los procesos soportados en una organización. La información puede ser dirigida automáticamente a los actores humanos o aplicaciones apropiadas.

Otros sistemas implementan “conciencia sobre procesos” al darle soporte desde dentro de su aplicación específica. Ejemplos de ello son las herramientas de colaboración, herramientas de gestión de proyectos y sistemas de manejo de caso. Otro tipo de PAIS comúnmente utilizado son los sistemas de Planificación de Recursos Empresariales (ERP) que apoyan todos los procesos de negocio de una empresa como la gestión de la cadena de suministro, los software para la administración de la relación con los clientes (CRM) y los recursos humanos (Aalst, 2011).

A pesar del enfoque que pueda tener un sistema de información, una tarea común es la de almacenar datos acerca de las actividades o procesos que se realizan en el área u organización que emplea el sistema. Estos datos tienen una importancia vital debido a su contenido y poseen un alto valor. Sin embargo al crear y registrar las trazas de estos datos, el enfoque influye en la cantidad y variedad de análisis que se pueden efectuar sobre ellos.

Vale aclarar que no necesariamente el sistema debe ser PAIS para generar trazas de gran completitud y organización. Todo depende de cómo se halle configurado el almacenamiento de las trazas y la cantidad de información que se recibe para confeccionarlas. Los PAIS incluyen a los sistemas que proporcionan una mayor flexibilidad o apoyan tareas específicas. Por ejemplo, pueden ser vistos como conscientes de los procesos sistemas más grandes de ERP (SAP, Oracle), CRM, sistemas basados en reglas, software de centro de llamadas, de alta gama de middleware (WebSphere), aunque no necesariamente controlan los procesos a través de algún motor de flujo de trabajo genérico. En su lugar, estos sistemas tienen en común que hay una noción proceso explícito y que el sistema de información es consciente de los procesos que soporta.

También un sistema de base de datos o programa de correo electrónico se pueden usar para ejecutar pasos en un proceso de negocio. Sin embargo, este tipo de herramientas de software no son “conscientes” de los procesos en que se utilizan. Por lo tanto, no participan activamente en la gestión y orquestación de los procesos para los que se utilizan (Aalst, 2011).

1.1.4 Estándar XES

No todos los sistemas de información registran los eventos en la forma deseada. La información sobre la relación entre los eventos y actividades o incluso entre las trazas y las instancias de proceso a menudo no se guarda. La razón principal de esto es que el registro de eventos no es

visto por los diseñadores de sistemas como una grabación de la ejecución de un proceso. En algunos sistemas el registro de eventos se utiliza para la depuración de errores encontrados, en otros los eventos tienen que ser derivados de los datos dispersos en varias tablas.

Aun cuando el registro de eventos (parcialmente) siga la estructura descrita el formato difiere entre sistemas. Antes de aplicar técnicas de análisis tales como la Minería de procesos, los registros deben ser convertidos a un formato estandarizado. Para ello el grupo de investigación de Arquitectura de Sistemas de Información en la Universidad Tecnológica de Eindhoven especificó el formato de registro de eventos XES (Buijs, 2010).

El estándar XES se emplea actualmente para representar la información de los registros de eventos, de tal forma que sea procesable por las herramientas para aplicar Minería de procesos. La Figura 1 muestra un ejemplo de un documento empleando XES.

```
<?xml version="1.0" encoding="UTF-8" ?>
<log xes.version="2.0" xes.features="arbitrary-depth" xmlns="http://www.xes-standard.org
    /">
    <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org/concept.
        xesext"/>
    <extension name="Time" prefix="time" uri="http://www.xes-standard.org/time.xesext"/>
    <global scope="trace">
        <string key="concept:name" value=""/>
    </global>
    <global scope="event">
        <string key="concept:name" value=""/>
        <date key="time:timestamp" value="1970-01-01T00:00:00.000+00:00"/>
        <string key="system" value=""/>
    </global>
    <classifier name="Activity" keys="concept:name"/>
    <classifier name="Another" keys="concept:name system"/>
    <float key="log attribute" value="2335.23"/>
    <trace>
        <string key="concept:name" value="Trace number one"/>
        <event>
            <string key="concept:name" value="Register client"/>
            <string key="system" value="alpha"/>
            <date key="time:timestamp" value="2009-11-25T14:12:45:000+02:00"/>
            <int key="attempt" value="23">
                <boolean key="tried hard" value="false"/>
            </int>
        </event>
        <event>
            <string key="concept:name" value="Mail rejection"/>
            <string key="system" value="beta"/>
            <date key="time:timestamp" value="2009-11-28T11:18:45:000+02:00"/>
        </event>
    </trace>
</log>
```

Figura 1 Registro de eventos en formato XES

El estándar define dos tipos de atributos. El primer tipo es para trazas y eventos, algunos de estos atributos son requeridos por algoritmos de Minería de procesos y la mayoría de los atributos de este tipo son definidos por las extensiones de XES. El segundo tipo de atributo son los atributos de datos, que almacenan información adicional acerca del objeto al cual se refiere la traza o actividad ejecutada. Algunos algoritmos requieren atributos específicos en el registro de eventos. Por otra parte no todos los atributos definidos por las extensiones estándar deben estar siempre presentes en el registro. Algunas veces la información requerida ni se encuentra en la fuente de datos (Peguero Álvarez, y otros, 2013).

La extensión de XES más importante es *concept*, la cual especifica un nombre para el registro de eventos, la traza y los eventos. Proporcionar nombres a cada elemento es fácil y muy informativo y por eso siempre deben ser provistos. Los nombres de las trazas incluirán identificadores únicos. La extensión *concept* define un atributo *instance* para eventos, este representa un identificador de la instancia de proceso que generó el evento.

Otra extensión importante es *time*. Esta especifica el atributo que representa el instante de ocurrencia del evento. La grabación del instante de tiempo es la clave para ordenar los eventos y permite realizar análisis de duración y desempeño. Es importante registrar el momento de la ejecución del evento con un alto nivel de detalle, preferiblemente hasta los segundos y en algunos casos los milisegundos. Especialmente en una base de datos operacional siendo consultada concurrentemente, muchos eventos pueden ser ejecutados dentro del mismo segundo, por esto es la necesidad de la precisión de milisegundos. Esto es necesario porque los eventos deben tener un único instante de tiempo dentro de la traza. Aunque la mayoría de los algoritmos pueden manejar instantes de tiempos iguales, los resultados mejoran si se brinda más precisión.

Los eventos son grabaciones atómicas y por eso no tienen duración. Como las actividades sí tienen duración, los eventos pueden ser de diferentes tipos, cada uno de estos grabando un estado de una actividad. Los tipos de eventos son proporcionados por la extensión *lifecycle*. Dos de los tipos de eventos más comunes son el *iniciado* y *completado* para indicar el inicio y la completitud respectivamente de una actividad. Contando con estos dos tipos de eventos es posible estimar tiempo de procesamiento y tiempo de espera en un análisis de desempeño, en la mayoría de los casos con usar el tipo *iniciado* y *completado* es suficiente. Si no existe información acerca del inicio y fin de una actividad se usa solamente el *completado* como tipo de evento.

Otra dimensión del análisis de Minería de procesos trata sobre la distribución del trabajo. Relacionando eventos, recursos o grupos de recursos se visualiza la distribución del trabajo entre diferentes unidades. Además se pueden construir redes sociales para indicar por ejemplo los actores del negocio que trabajaron juntos en algunos casos o grupos de actores desempeñando tareas similares. Para lograr un descubrimiento de estas redes, el actor o grupo que ejecutó el evento debe ser registrado con cada evento. Lo anterior se define como la extensión *organizational* que define tres diferentes atributos para eventos. Comúnmente se emplea el *resource* para recoger el nombre o identificador del actor que ejecutó el evento. Adicionalmente el rol del recurso se puede almacenar también con el atributo *role* así como el grupo al que el usuario pertenece, mediante el atributo *group*. La decisión de qué atributos usar depende de la información disponible y del nivel de detalle deseado. En algunos casos, los análisis sobre el rol o el grupo son necesarios (Günther, y otros, 2014).

Otra de las extensiones de XES a emplear es *micro*. Los actuales sistemas pueden definir y manejar actividades anidadas o sub procesos, y esta información debe ser representada en el registro de eventos. Para ello micro define un atributo *level* en correspondencia con el nivel de la actividad. Cuenta además con el atributo *parentId* para especificar en un evento el identificador de su padre, si tiene uno. Por último está el atributo *length* que brinda la cantidad de hijos si el evento en cuestión es padre (Group, 2016).

1.1.5 Arquitectura de referencia Bosón

El lenguaje de programación PHP es muy empleado para desarrollo de aplicaciones web de disímiles tipos, pero tenía como limitante que sus características no permitían utilizar las mejores prácticas y patrones comunes de programación y diseño de sitios web. A lo largo de cinco años evolucionó y aparecieron *frameworks* como Symfony, ZendFramework y Laravel que ayudan a los desarrolladores a convertir a PHP en un lenguaje donde aplicar las buenas prácticas. Aún con la existencia de estos *frameworks* no suplen todas las necesidades de los desarrolladores debido a que existen muchos requisitos que son propios de ciertos dominios (Torres, 2014).

Debido al amplio espectro de aplicaciones web que se desarrollan sobre dicho lenguaje, la Dirección General de Proyectos de la UCI le asignó la tarea al Departamento de Desarrollo de componentes del Centro de Informatización de Entidades de crear una arquitectura de referencia donde se especifiquen las buenas prácticas para el desarrollo de aplicaciones con PHP así como la definición de patrones y tecnologías a utilizar. De este modo será posible llegar a un punto común entre las diversas soluciones que hoy en día no se pueden integrar o reutilizar, aun cuando emplean el mismo lenguaje.

Bosón es una implementación de esta arquitectura de referencia y algunos de los componentes son los siguientes:

- Caché: permite utilizar varios mecanismos de almacenamientos para salvar temporalmente cualquier estructura de datos.
- Seguridad: redefine la seguridad de Symfony2 para gestionar la autenticación y la autorización. Utiliza el estándar SAML 2.0 para la comunicación, el patrón SSO para la autenticación y la gestión de usuarios basada en roles para la autorización.
- IUX: incluye todos los elementos necesarios para desarrollar con los componentes visuales de la Interfaz Única creada para los productos de la UCI.
- Excepciones: permite definir las excepciones del sistema en un fichero de configuración y lanzarlas sin necesidad de crear clases. También admite la internacionalización de las excepciones así como un log de ocurrencia de las mismas.
- Estructura y Composición (EC): permite definir de forma dinámica las estructuras y nomencladores que se necesiten en la aplicación a desarrollar.
- Trazas: permite almacenar en una base de datos las trazas de excepciones, acciones y accesos a datos. También permite registrar el rendimiento (tiempo de ejecución y memoria) de las trazas de acción.

1.2 Problemas en la confección del registro de eventos

1.2.1 Trazas de Bosón

En Bosón resulta complejo obtener un registro de eventos debido a la forma en que se guarda la información en sus trazas. Se registran tres tipos de trazas: acceso a datos, de acción y excepciones. Las trazas de datos mostradas en la Figura 2 brindan información sobre la tabla de la base de datos donde se ejecutó una operación de insertar, eliminar o actualizar.

Fecha	Hora	Usuario	Esquema	Tabla	Acción
2016-04-05	11:47	pebarzaga	public	seg_usuario	Update
2016-04-04	15:20	pebarzaga	public	seg_estado_sso	Delete
2016-04-04	15:14	pebarzaga	public	seg_usuario	Update
2016-04-04	14:47	pebarzaga	public	seg_estado_sso	Delete
2016-04-04	14:31	pebarzaga	public	seg_usuario	Update

Figura 2 Trazas de datos en Bosón

Las trazas de acciones mostradas en la Figura 3 indican el controlador o clase de PHP y el método o acción que se ejecutó en un momento. Si ocurre algún error las trazas de excepciones se encargan de recoger el tipo de excepción y el mensaje que devuelve.

Fecha	Hora	Usuario	Referencia	Controlador	Acción
2015-12-01	15:02	pebarzaga	Sandbox\SandboxBundle	TrazasController	trazasAccionesAction
2015-12-01	15:02	pebarzaga	Sandbox\SandboxBundle	TrazasController	trazasAccionesAction
2015-12-01	15:02	pebarzaga	Sandbox\SandboxBundle	TrazasController	trazasAccionesAction
2015-12-01	15:01	pebarzaga	Sandbox\SandboxBundle	TrazasController	trazasBaseDatosAction
2015-12-01	15:01	pebarzaga	Sandbox\SandboxBundle	TrazasController	trazasExcepcionesAction
2015-12-01	15:01	pebarzaga	Sandbox\SandboxBundle	TrazasController	trazasAccionesAction
2015-12-01	15:01	pebarzaga	Sandbox\SandboxBundle	TrazasController	trazasIndexAction
2015-12-01	15:01	pebarzaga	UCI\Boson\PortalBundle	DefaultController	getTilesAction
2015-12-01	15:01	pebarzaga	UCI\Boson\PortalBundle	DefaultController	getUserAction
2015-12-01	15:01	pebarzaga	UCI\Boson\PortalBundle	DefaultController	getUXParamsAction
2015-12-01	15:01	pebarzaga	UCI\Boson\PortalBundle	DefaultController	indexAction
2015-12-01	15:01	pebarzaga	AerialShip\SamlSPBundle	SecurityController	checkAction

Figura 3 Trazas de acción en Bosón

Todas ellas contienen la fecha y hora del evento y el usuario del sistema involucrado en la ejecución de una actividad. Estos valores se pueden emplear para conformar la marca de tiempo y definir atributos como el tipo de operación realizada o el recurso que la ejecutó.

El problema de las trazas radica en primer lugar en que los eventos no poseen un campo o atributo que los relacione con una instancia de proceso, generalmente se emplean identificadores o llaves foráneas para indicar dicha relación. En segundo lugar no se registra información adicional de estos eventos como costos de ejecución o recursos utilizados, información disponible en el resto de las tablas del sistema. Con estas limitantes actualmente no se puede aplicar alguna herramienta que permita obtener un registro de eventos.

1.2.2 Problema de identificación de eventos

Reconstruir un registro de eventos significa decidir cuándo inferir la existencia de un evento en la base de datos operacional y llenar cada uno de los atributos del evento con valores relevantes. Estos valores pueden ser extraídos de la base de datos o pueden ser especificados por el experto del dominio (Rodríguez, y otros, 2012).

Existen diversas formas de obtener los eventos según la fuente de datos, por ejemplo se aplican patrones de identificación, ordenamiento, asociación y correlación (Rodríguez, y otros, 2012). Para el caso de Bosón se emplea el modelo de eventos propuesto por Wil van der Aalst en su trabajo *Extracting Event Data from Databases to Unleash Process Mining*. Este modelo está definido de forma genérica para sistemas que emplean bases de datos como fuente de datos y se definen como eventos las acciones de insertar y eliminar objetos y sus relaciones, y las actualizaciones de los objetos, como se muestra en la Figura 4 (Aalst, 2011).

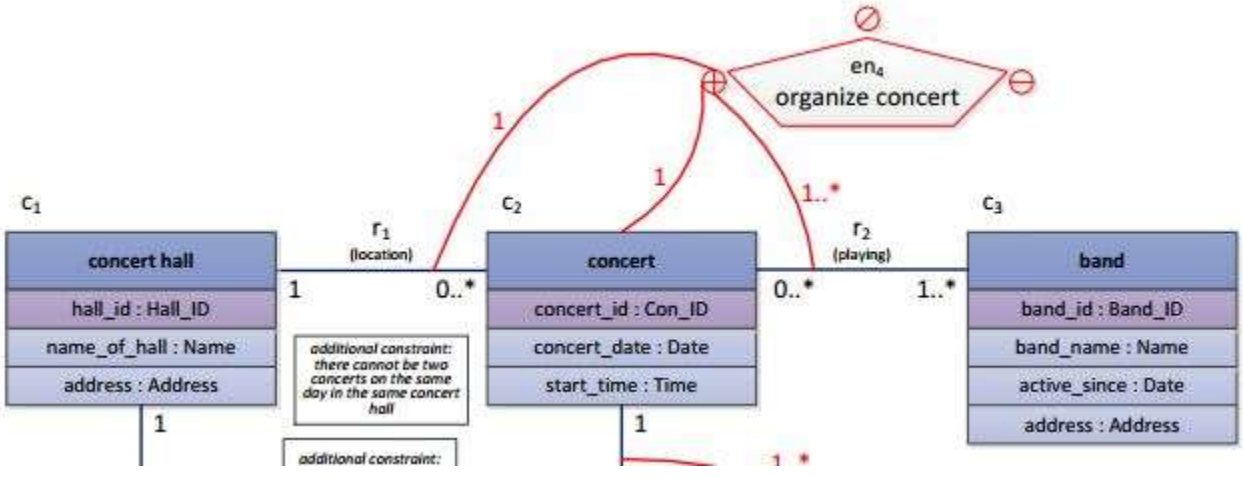

Figura 4 Modelo de eventos aplicado en un sistema

El símbolo ⊕ se corresponde con las inserciones, el símbolo ⊘ con las eliminaciones y el símbolo ⊖ con las actualizaciones. Las líneas rojas que van desde los símbolos hacia los elementos de la figura indican el tipo de operación que se está realizando en cada elemento, en el caso de la Figura 4 se definen como eventos las inserciones en la entidad *concert* y sus relaciones r1 y r2. Además el modelo brinda un conjunto de principios de confección del registro de eventos con el objetivo de identificar errores y obtener un buen punto de partida para aplicar la Minería de procesos.

Siguiendo el modelo de eventos de Wil van der Aalst, los eventos en Bosón se corresponden con las inserciones, actualizaciones y eliminaciones de datos en las tablas del negocio que se maneje. Por ejemplo en el caso de estudio “Gestionar sitio web”, mostrado en la Figura 5, la entidad Usuario es la instancia de proceso y los eventos son: insertar nuevos usuarios, contactos y publicaciones, eliminar y modificar publicaciones.

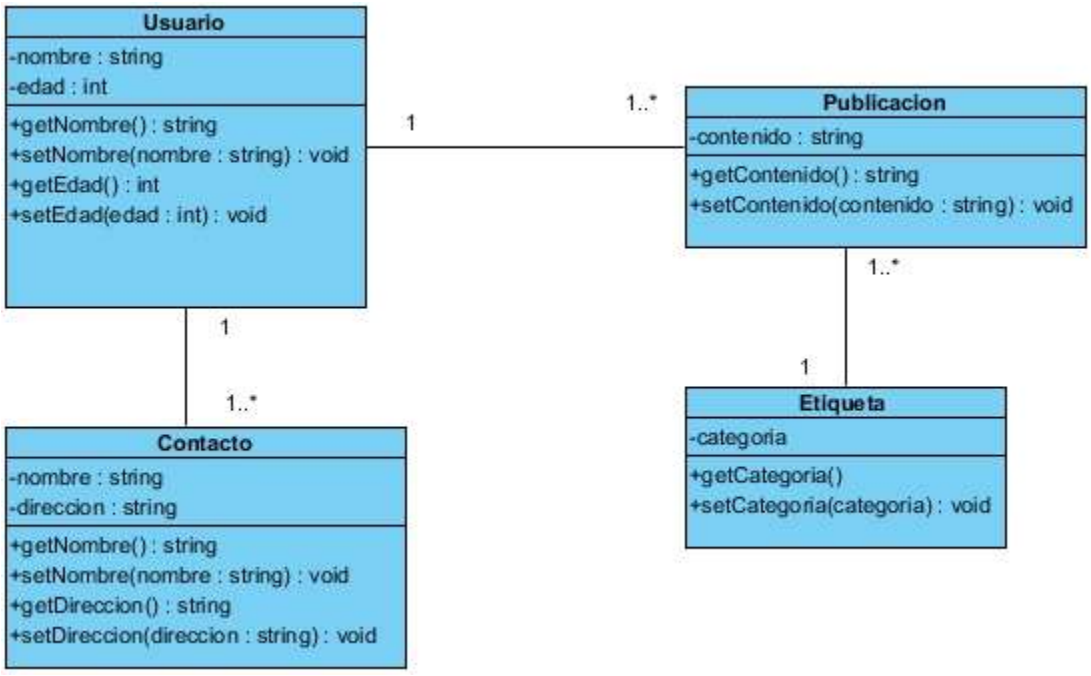

Figura 5 Caso de estudio

1.2.3 Problema de la convergencia y divergencia

Una de las propiedades de un registro de eventos es que cada evento tiene que estar relacionado a solo una instancia de un proceso. En la realidad esto no es siempre así, por lo que representa un problema en la conversión hacia registro de eventos (Buijs, 2010).

Como resultado de la convergencia y/o divergencia puede ocurrir que un algoritmo sea incapaz de utilizar un registro de eventos. Por ejemplo el algoritmo “Alpha miner” no es capaz de realizar Minería de procesos a un registro de eventos donde ocurran convergencia y divergencia.

La Figura 6 provee un ejemplo de tablas de una base de datos.

Order		
ID	**Value**	**Created On**
1	€ 100	01-01-2010 09:00
2	€ 100	13-01-2010 15:47
3	€ 100	05-02-2010 17:01
4	€ 100	13-03-2010 08:45

Payment		
ID	**Amount**	**Payed On**
10	€ 150	31-01-2010 18:00
11	€ 25	03-02-2010 11:12
12	€ 25	03-02-2010 15:14
13	€ 75	25-02-2010 12:55
14	€ 25	28-02-2010 16:03
15	€ 100	30-03-2010 08:46

Order × Payment	
Order ID	**Payment ID**
1	10
1	11
1	12
2	10
3	13
3	14
4	15

Figura 6 Entidades de ejemplo

Las órdenes(o pedidos) están en la tabla Orden. Cada orden tiene un identificador único almacenado en la columna ID Orden. Además, cada orden tiene un precio que necesita ser pagado almacenado en la columna precio. Los pagos se almacenan en la tabla Pago. Cada pago tiene un identificador único almacenado en la columna ID Pago. La cantidad pagada en cada pago se almacena en la columna Cantidad. Para relacionar las órdenes y los pagos existe la tabla Orden X Pago. No obstante, un pago no necesariamente tiene que pagar la orden completamente, pudiera pagar solamente una parte. Por ejemplo los pagos 13 y 14 pagan por la orden número 3 como se puede ver en la tabla Orden X Pago. Para complicar aún más las cosas un pago puede ser parte de dos órdenes. En el ejemplo el pago 10 paga por las órdenes 1 y 2. En este caso particular se puede deducir que 100 de los 150 pesos pagaron la orden 2 mientras

que los otros 50 pesos pagaron la orden 1. En general no se puede decidir cómo el pago se divide entre las órdenes correspondientes.

En este ejemplo se elige como instancia de proceso la orden siendo ID Orden el identificador de instancia de proceso. Los pagos son eventos que ocurren para pagar órdenes existentes.

El problema de la convergencia existe cuando una actividad se ejecuta en múltiples instancias de proceso a la vez. Esto se puede reconocer cuando hay una relación 1: N (uno a muchos) desde un evento hacia la instancia de proceso. Al ocurrir convergencia puede aparecer en el registro de eventos que un usuario estaba pagando dos órdenes a la vez (Buijs, 2010).

El mismo problema puede ocurrir a la inversa y se denomina divergencia. La divergencia existe cuando para una instancia de proceso la misma actividad es ejecutada múltiples veces. Esto se puede reconocer cuando existe una relación N: 1 (muchos a uno) desde los eventos hacia la instancia de proceso en una base de datos. Esto se puede observar en el ejemplo cuando la orden 3 se paga con el pago 13 y 14. También ocurre con la orden 1 que se paga con parte del pago 10 y el 11 y el 12. El efecto de divergencia en la Minería de procesos se magnifica en el modelo de proceso resultante. Los algoritmos para descubrir modelos de procesos especifican cada actividad solo una vez. Si un evento ocurre múltiples veces dentro de una traza, el algoritmo tratará de reutilizar la misma instancia de evento múltiples veces. Si no ocurren otros eventos entre múltiples ejecuciones de una actividad, esto resulta en ciclos en el modelo de proceso. No todos los algoritmos de descubrimiento de proceso pueden manejar ciclos de eventos. Sin embargo, si ocurren otros eventos, el modelo de proceso se volverá más complejo (Buijs, 2010).

1.2.4 Problemas de la confección del registro de eventos

Existen un conjunto de problemas independientes del enfoque, que están presentes en los registros de eventos extraídos de sistemas reales y están asociados a la herramienta empleada para conformar el registro, el algoritmo utilizado o las condiciones del negocio. Estos problemas deben ser considerados por los desarrolladores a la hora de crear el registro de eventos para que la información obtenida sea lo más completa posible (Bose, y otros, 2013):

- Granularidad de eventos: los eventos se almacena con bajos niveles de detalle o altos niveles de detalle en el mismo registro.
- Heterogeneidad de los casos: se refiere a instancias de procesos con muchos flujos alternos, o a los procesos que cambian constantemente en el tiempo.
- Datos voluminosos: los eventos de bajo nivel y cortos períodos de duración son registrados masivamente por algunos sistemas.

- Problemas relacionados con las marcas de tiempo: pueden ser incorrectas, estar mezcladas o no brindar suficiente información (faltan la hora o los minutos en que ocurrió un evento).
- Pérdida de datos: faltan valores de atributos, existe ausencia de eventos en las trazas, se registran eventos incompletos.
- Ambigüedad entre eventos: los eventos aparecen duplicados o las actividades se intercalan entre sí.
- Flexibilidad de los procesos: se refiere a los cambios momentáneos o permanentes en un proceso.
- Ruido: comportamientos raros, anormales o excepcionales en los procesos.
- Procesos agrupados: los procesos no se pueden ver como una entidad única, se diseñan y ejecutan como entidades modulares que poseen varios subprocesos agrupados.
- Alcance: en ocasiones no se determina correctamente el caso o el proceso a analizar, y se hace necesario tener conocimientos del dominio para hacer una selección adecuada.

Por las características de Bosón el problema de la marca de tiempo y el problema de la pérdida de datos son manejados porque siempre se registra la fecha y hora de ocurrencia de los eventos. Si se desconoce el valor de un atributo se le asigna un valor predeterminado. No se registran los eventos de bajo nivel. El ruido se analiza en el modelo de procesos obtenido a partir del registro de eventos, donde se detectan los comportamientos anómalos.

Desafortunadamente los procesos agrupados no se pueden analizar y/o modelar con los actuales enfoques de extracción de registros de eventos a partir de sistemas orientados a datos. Con Bosón no se resuelve actualmente la ambigüedad porque la marca de tiempo de las trazas no llega al nivel de detalles de segundos y milisegundos, por tanto dos eventos del mismo tipo ocurridos a la misma hora no se pueden diferenciar. Tampoco se resuelve la granularidad porque no se representan eventos compuestos.

1.3 Herramientas para gestionar el registro de eventos

Se analizaron y estudiaron diversas herramientas usadas para la obtención del registro de eventos, de ellas se identificaron cuatro en específico por sus características comunes y por cómo manejan los problemas antes mencionados.

1.3.1 Eventifier

Eventifier es una herramienta que ayuda en la reconstrucción de un log de eventos desde bases de datos operacionales donde existen evidencias de la ejecución de instancias de procesos. Primeramente, el experto en el dominio identifica los eventos en la base de datos operacional,

los ordena y le asocia datos. Para ello se apoya en el componente Extractor de eventos, que soporta estas actividades y ayuda al experto de forma interactiva. El resultado de este primer paso es una serie de eventos los cuales todavía no están relacionados. La correlación es dirigida por el componente Correlacionador, el cual ayuda al experto del dominio de manera interactiva a identificar los mejores atributos y condiciones para reconstruir trazas de proceso. El resultado del proceso completo es un registro de eventos listo para aplicársele Minería de procesos (Rodríguez, y otros, 2012).

Eventifier tiene como desventajas que no permite exportar a XES el registro de eventos obtenido, se limita al descubrimiento de los procesos y no maneja los problemas de convergencia, divergencia y heterogeneidad.

1.3.2 XESame

XESame es un mapeador de XES que provee una manera genérica para extraer un registro de eventos desde una fuente de datos. Una de las potencialidades de XESame es su facilidad de uso ya que no requiere habilidades de programación. XESame le permite a un experto en el dominio especificar cómo el registro de eventos debe ser extraído de la base de datos operacional de un sistema de información determinado y convertido a XES o MXML. El sistema brinda varias interfaces para la definición de la conversión en las cuales se pueden definir los tipos de gestores de bases de datos a usar como son PostgreSQL o Mysql. La herramienta permite establecer las relaciones entre las tablas de la base de datos y sus respectivos campos en el registro de eventos, teniendo en cuenta las extensiones de XES definidas. Una vez obtenido el registro de eventos en formato XES, se debería ser capaz de analizar este registro de todas las maneras posibles en el ámbito de Minería de procesos (Buijs, 2010).

La herramienta tiene como desventajas que se debe definir una conversión para poder mapear la base de datos y obtener los eventos, esta conversión requiere conocimientos de SQL para poder ejecutar las consultas que unen las tablas. Se le da tratamiento a la convergencia y divergencia seleccionando instancias de proceso de bajo nivel. Esta solución minimiza el impacto de los problemas pero no los resuelve eficazmente. XESame tampoco considera los eventos compuestos ni los reconoce.

1.3.3 Módulo para el registro y transformación de trazas de eventos al formato XES

Este módulo se desarrolló en la UCI y viene integrado en el *framework* Sauxe. Su caso era muy similar al de Bosón, las trazas no brindaban suficiente información así que se definieron nuevas tablas para guardar los datos de los eventos que ocurrían en el sistema. Con esta nueva forma

de organizar las trazas, se podía mapear la información y obtener un registro de eventos en formato XES válido para aplicarle Minería de procesos (Peguero Álvarez, y otros, 2013).

La principal desventaja de esta herramienta es que fue desarrollada sobre Sauxe y solo es aplicable para aquellos sistemas que se desarrollen con ese *framework*. Además, maneja la convergencia y divergencia de forma similar al XESame al seleccionar entidades de bajo nivel como instancias de proceso lo cual minimiza en cierta forma el impacto de los problemas pero no los resuelve.

1.3.4 XTract2 tool

Esta herramienta emplea un enfoque diferente al utilizado por otros trabajos. En lugar de obtener un registro de eventos y exportarlo a XES, obtiene el modelo de procesos usando un modelo de artefactos. Se mapea cada tabla de la base de datos como un artefacto y se establecen las relaciones entre ellos empleando puertos y canales de comunicación. El uso del modelo de artefactos disminuye en gran medida los problemas de la convergencia y la divergencia, elimina los eventos repetidos y permite representar los procesos relacionados entre sí, o sea aquellos que comparten actividades. Este último problema no había sido resuelto por otras investigaciones. Este trabajo permitió mejorar una herramienta existente para el modelado empleando artefactos y resolver muchos de los problemas que presenta este enfoque (Lu, 2013).

Como principal desventaja se tiene que la herramienta no se encuentra disponible aunque se espera que salga en la próxima versión del ProM, *framework* que agrupa un amplio espectro de herramientas y técnicas para aplicar Minería de procesos. Utilizar el modelo de artefactos en proyectos ágiles como Bosón resulta una desventaja; la complejidad del modelo requiere largos períodos para su comprensión y aplicación.

1.3.5 Resultados

La Tabla 2 y la Tabla 3 ilustran respectivamente las principales ventajas y desventajas identificadas en las herramientas estudiadas.

Tabla 2 Ventajas de las herramientas

Características	**Eventifier**	**XESame**	**Módulo de Sauxe**	**XTract2**
Identificar eventos	Logs de base de datos	Fuente de datos genérica	Trazas de Sauxe	Trazas del sistema
Identificar instancias	Patrones de identificación	Selección de instancia	Selección de instancia	Enfoque XTract

Correlacionar eventos con instancias	Patrones de correlación	Definición de correlación	Definición de correlación	Patrones de artefactos
Exportar a XES	No	Si	Si	Modelo de artefactos
Empleo de herramienta	Eventifier	XES Mapper	Módulo	XTract 2

Tabla 3 Dificultades de las herramientas

Dificultades	**Eventifier**	**XESame**	**Módulo de Sauxe**	**XTract2**
Procesamiento de la información	Análisis de logs de base de dato	Análisis de trazas existentes	Análisis de trazas existentes	Análisis de trazas existentes
Maneja convergencia	No	Selección de instancia de bajo nivel	Selección de instancia de bajo nivel	Si
Maneja divergencia	No	Selección de instancia de bajo nivel	Selección de instancia de bajo nivel	Si
Herramienta disponible	Si	Si	Contenida en Sauxe	No

A partir de este análisis se determina crear una solución para Bosón que agrupe las principales ventajas y buenas prácticas: definir eventos, definir procesos, correlacionar eventos a procesos, definir atributos adicionales y exportar a XES. Esta solución utiliza nuevas tablas para guardar la información sobre los eventos que ocurren en el sistema y realiza una configuración inicial de los procesos y sus actividades para que cada evento se asocie a una única instancia de proceso, minimizando la convergencia y divergencia. Además con la solución propuesta se intenta resolver los problemas de granularidad mediante el modelado de eventos compuestos.

1.4 Metodología, herramientas y tecnologías utilizadas

Para desarrollar la solución propuesta se emplean las herramientas y tecnologías definidas por el Departamento de Desarrollo de Componentes en la Vista de entorno de desarrollo tecnológico.

1.4.1 Metodología

Se emplea como metodología para el desarrollo la variación de la metodología *Agile Unified Process* (AUP) para los proyectos productivos de la UCI, en su fase de Ejecución.

1.4.2 Herramientas

- Symfony 2.3: Es un popular *framework* para desarrollar aplicaciones PHP. Se anunció por primera vez a principios de 2009 y supone un cambio radical tanto en arquitectura interna como en filosofía de trabajo respecto a sus versiones anteriores. Symfony2 ha sido ideado para exprimir al límite todas las nuevas características de PHP 5.3 y por eso es uno de los *frameworks* PHP con mejor rendimiento. Esta nueva versión mejora todos los aspectos de la versión original, al tiempo que mantiene una alta retro compatibilidad, salvo en el caso de los formularios (Eguiluz, 2013).
- PhpStorm 9.0.2: Es un Entorno de Desarrollo Integrado (IDE) de PHP compatible con PHP desde su versión 5.3 hasta la 7.0, proporciona prevención de errores, mejor autocompletado y refactorización de código, depuración de configuración cero, y un editor HTML, CSS, JavaScript. El IDE ofrece completamiento inteligente de código, resaltado de sintaxis, configuración extendida del formato de código, comprobación de errores sobre la marcha, plegado de código, soporta las mezclas de idiomas y más (Jet16).
- Visual Paradigm 5.0: Visual Paradigm es una herramienta de Ingeniería de Software Asistida por Computadora (CASE) que utilizando Lenguaje de Modelado Unificado (UML) como lenguaje de modelado, permite la captura, diseño, gestión y documentación de los artefactos generados durante el proceso de desarrollo de software (Vis16).

1.4.3 Tecnologías

- PHP 5.4: PHP (acrónimo recursivo para PHP: Hypertext Preprocessor) es un lenguaje de scripting de propósito general ampliamente utilizado que está especialmente indicado para el desarrollo web y puede ser embebido en páginas HTML. Lo que distingue a PHP es que el código se ejecuta en el servidor, lo que genera HTML que se envía al cliente (PHP16).
- Apache 2.0: Es un servidor web de libre acceso que se distribuye bajo una licencia de “código abierto”. La versión 2.0 se ejecuta en la mayoría de los sistemas operativos basados en UNIX (como Linux, Solaris, Digital UNIX y AIX), en otros sistemas POSIX derivados de UNIX (como Rhapsody, BeOS, y BS2000 / OSD), en AmigaOS, y en Windows 2000 y superiores (Sea16).
- AngularJS: Es un marco estructural para aplicaciones web dinámicas. Permite utilizar HTML como lenguaje de plantillas y extender la sintaxis de HTML para expresar los

componentes de la aplicación de forma clara y sucinta. Los enlaces de datos y las inyecciones de dependencia de Angular eliminan gran parte del código que de otro modo tendría que escribirse. Y todo sucede dentro del navegador, lo que lo convierte en un socio ideal con cualquier tecnología de servidor (Ang16).

- HTML 5: La última versión de HTML. El término representa dos conceptos diferentes: Se trata de una nueva versión de HTML, con nuevos elementos, atributos y comportamientos. Contiene un conjunto más amplio de tecnologías que permite a los sitios Web y a las aplicaciones ser más diversas y de gran alcance (Dev16).
- CSS 3: Es un lenguaje para definir el estilo o la apariencia de las páginas web, escritas con HTML o de los documentos XML. CSS se creó para separar el contenido de la forma, a la vez que permite a los diseñadores mantener un control mucho más preciso sobre la apariencia de las páginas (Des16).

Conclusiones del capítulo

El análisis realizado sobre el componente de trazas en Bosón y las extensiones de XES arrojó como resultado que la información almacenada actualmente no es lo suficientemente completa ni tiene una organización básica como para confeccionar un registro de eventos.

El enfoque de crear el registro de eventos partiendo de trazas guardadas sin configurar constituye una desventaja porque da lugar a la aparición de problemas que afectan el modelo de procesos. No manejar estos problemas implica obtener datos poco confiables que apoyen la toma de decisiones.

Las herramientas actuales para confeccionar un registro de eventos no cubren todos los problemas de Bosón pero brindan buenas prácticas para solucionarlos.

CAPÍTULO 2: PROPUESTA DE SOLUCIÓN

Introducción

En este capítulo se describe la propuesta de solución de forma detallada, explicando cómo se resuelven los problemas asociados a la confección del registro de eventos. Se identifican y describen los requisitos funcionales y no funcionales. Se presentan los patrones del diseño empleados en el desarrollo de la solución. Posteriormente se muestran los artefactos generados durante la fase de ejecución.

2.1 Modelo conceptual

Los principales conceptos y relaciones de la solución se muestran en la Figura 7. Bosón es un marco de trabajo donde se generan trazas y se gestionan procesos (en este caso, procesos de sistemas construidos con Bosón). El proceso es configurado por un experto del dominio. Las trazas contienen los eventos definidos en el negocio, y estos eventos se corresponden a procesos. En el registro de eventos se guarda la información de las trazas para luego ser exportado al estándar XES.

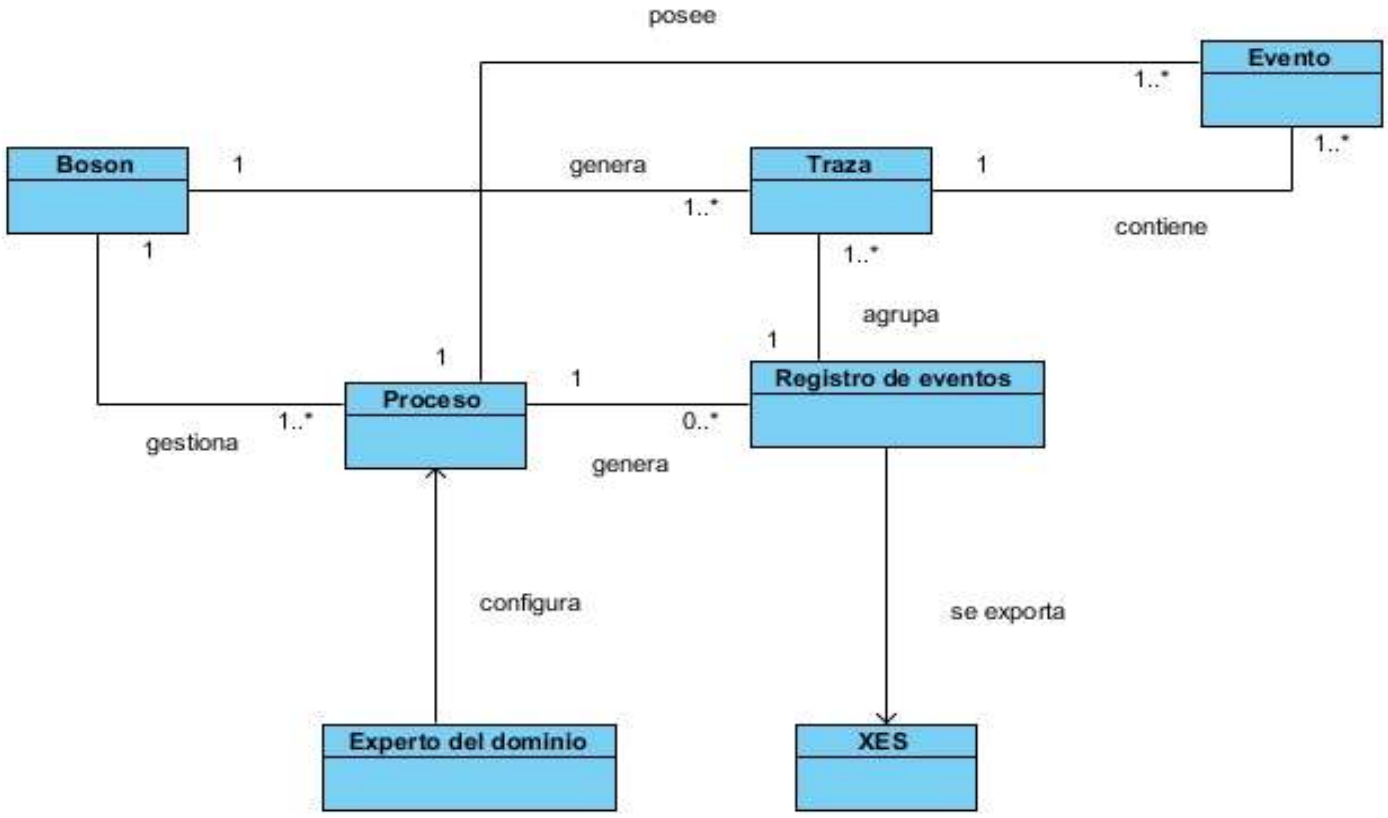

Figura 7 Modelo conceptual

2.2 Análisis del componente de trazas de Bosón

El componente de trazas cuenta con clases que “escuchan” (*listener* del inglés) y guardan los eventos en el momento exacto de su ejecución. Hay definida una clase *listener* para registrar los accesos a datos, las acciones y las excepciones. La clase *listener* de datos es la relevante para la solución porque captura la información de las tablas que están siendo modificadas cuando ocurre el evento. En la Figura 8 se ilustra la clase y a continuación se describen sus métodos.

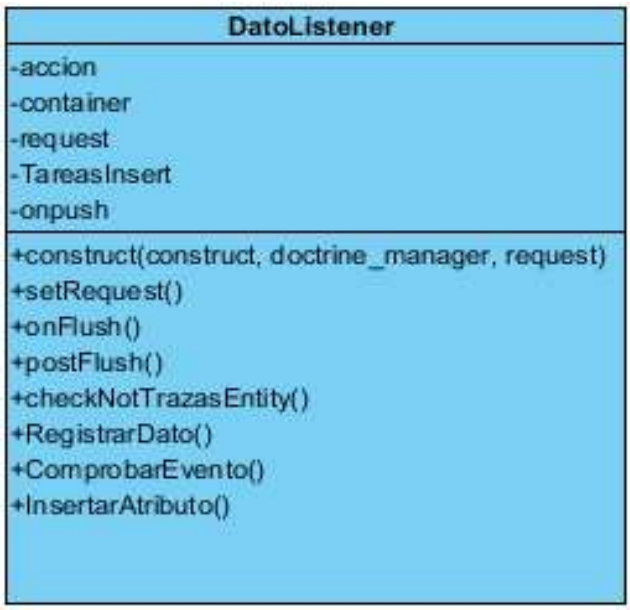

Figura 8 Clase listener para datos

La clase cuenta con un conjunto de atributos que permiten gestionar las peticiones web y acceder a servicios de otras clases. Los métodos onFlush y postFlush capturan los eventos de tipo inserción, actualización y eliminación. El método checkNotTrazasEntity distingue el evento si se ejecuta sobre las entidades del negocio o sobre las entidades que guardan la información de las trazas. El método RegistrarDato es el encargado de obtener y persistir la información del evento como traza. Los métodos ComprobarEvento e InsertarAtributo son nuevas funcionalidades agregadas, con el objetivo de comprobar el tipo de evento y guardar la información de la traza con el formato requerido para conformar el registro, además de los atributos adicionales del evento.

2.3 Propuesta de solución

El análisis de las soluciones existentes permite conocer cómo el efecto de confeccionar un registro de eventos a partir de trazas ya almacenadas resultaba una desventaja. Se deben realizar varios tipos de operaciones: mapeo de la base de datos, determinar eventos y procesos, correlacionarlos, definir conversiones y atributos adicionales en caso de ser necesario y exportar al estándar XES. Todas ellas consumen tiempo, recursos humanos y tecnológicos. Además, este

enfoque de trabajo da lugar a la ocurrencia de la convergencia, la divergencia, tener eventos duplicados, entre otros problemas. Tampoco se puede garantizar que el sistema guarde adecuadamente la información básica de los eventos de forma tal que se pueda confeccionar el registro.

En Bosón se decide realizar una pre configuración inicial donde se definen cuáles son los procesos a manejar en el sistema implementado, sus actividades y los atributos, así como las tablas que se corresponden con cada uno de estos elementos. De esta forma el sistema ya está listo para escuchar y registrar los eventos con los datos apropiados para aplicar Minería de procesos. Para ello se definen en Bosón instancias de proceso, eventos, atributos, la relación entre tablas y cómo se exporta a XES.

2.3.1 Selección de instancias de proceso

El primer paso en la configuración consiste en seleccionar las instancias de proceso. Cada vez que ocurra un evento en el sistema debe estar asociado a una única instancia. Elegir la instancia de proceso es complejo porque implica analizar cuáles son los objetos del negocio modelado y cómo se relacionan entre sí. Esta tarea se torna difícil en sistemas que manejen muchas entidades interrelacionadas. En un sitio web, por ejemplo, los usuarios pueden seleccionarse como instancia de proceso porque sobre ellos gira la lógica del negocio: escriben publicaciones o comentarios, crean contactos, comparten recursos.

La información de la instancia de proceso se guarda en la tabla pmProceso mostrada en la Figura 9.

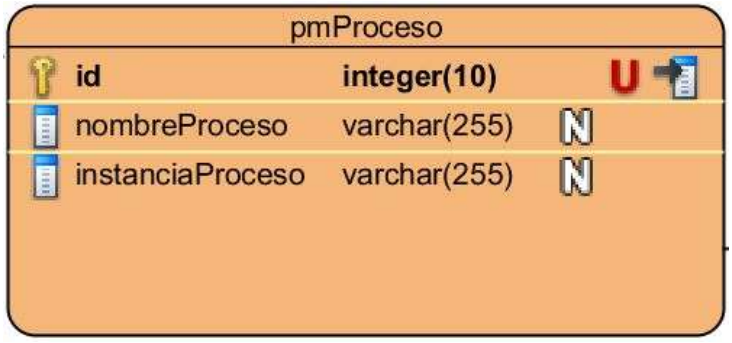

Figura 9 Entidad pmProceso

2.3.2 Selección de eventos

Los eventos se configuran como actividades relacionadas a un proceso, de ellas se guarda en la tabla pmActividad el nombre y la tabla sobre la cual se ejecuta la actividad, esta tabla puede ser la misma de la instancia de proceso u otra relacionada. Se registra además el identificador de la instancia, el tipo de actividad (insertar, eliminar o actualizar) que se relaciona con los tipos de

eventos definidos. Por último el nivel y si es una actividad hija el identificador de la actividad padre, como se muestra en la Figura 10.

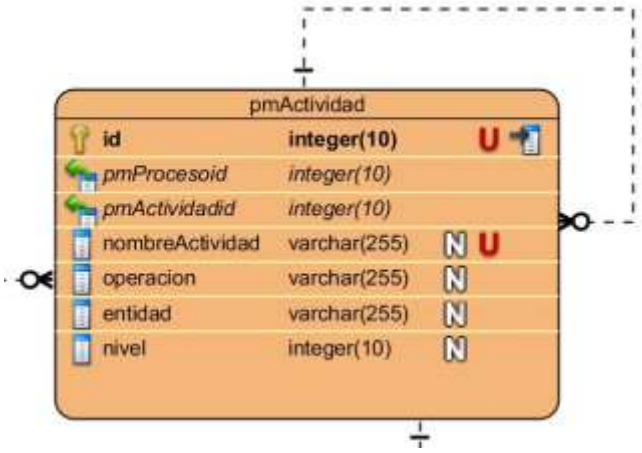

Figura 10 Entidad pmActividad

Se establece una relación de uno a muchos en la misma entidad y se emplean los valores del identificador de la actividad padre y el nivel precisamente para modelar los eventos compuestos. Estos eventos están constituidos por otros sub eventos, por ejemplo un evento A se compone de B, C y D. Al plantear una relación de tipo padre-hijo se modela la jerarquía de eventos propuesta por Reinhold Dunkl, con el objetivo de disminuir el efecto de la granularidad en los modelos de procesos y brindar información adicional acerca de los sub eventos que componen a un padre (Dunkl, 2013). La extensión Micro del estándar XES posee también campos que representan esta jerarquía, simplificando el modelo de procesos que se le muestra al usuario al emplear la herramienta ProM y permite revisar los eventos compuestos con mayor nivel de detalle (Group, 2016).

Aplicar el modelo jerárquico de Dunkl y aprovechar la extensión Micro permite representar en el registro los valores compuestos y minimizar el efecto del problema de granularidad.

Gracias a la clase *listener*, cuando ocurre un evento en Bosón se comprueba si se corresponde con alguna de las actividades definidas previamente en la tabla pmActividad. En caso de ser positiva la respuesta se guardan los datos del evento en la tabla pmRegistro mostrada en la Figura 11 a continuación. Sobre esta tabla se efectúan consultas posteriormente para extraer de ella los eventos que se deseen analizar en el registro.

Figura 11 Entidad pmRegistro donde se guardan los eventos

Cuando la clase listener captura el evento y desconoce el rol o el usuario les asigna valores por defecto: el usuario es “IS_AUTHENTICATED_ANONIMOUSLY” y el rol se representa con un espacio en blanco “”. Esta característica mencionada antes evita precisamente problemas de falta de datos o registrar eventos incompletos. Vale destacar además que los eventos poseen un valor de estado, que permite a las herramientas de minería calcular el tiempo en que se demora el evento en ejecutarse. Para el caso de Bosón el estado de los eventos simples es “completado”, y los compuestos poseen valor “iniciado” y “completado” según se vayan ejecutando sus hijos.

Gracias a la pre configuración establecida cada evento se registra como único y se corresponde con una instancia de proceso también única. De esta forma se resuelven los problemas de convergencia y divergencia al no permitir que un mismo evento ocurra en varias instancias, ni que para una instancia se ejecute el mismo tipo de evento más de una vez.

2.3.3 Atributos adicionales de los eventos

Los eventos necesitan una marca de tiempo, un identificador y una instancia de proceso para ser organizados y analizados. Esos valores son asignados al evento mediante los nuevos métodos de la clase DatoListener.

Los atributos adicionales ofrecen información extra como el costo de ejecutar una acción o qué recurso humano la ejecutó. El valor del atributo se obtiene de una o varias columnas de una tabla, siempre y cuando esa tabla se relacione con la tabla de la actividad o la tabla de la instancia

de proceso. Por ejemplo cuando se inserta una publicación en el sitio web puede resultar de interés conocer el nombre del usuario que la escribió o la etiqueta de la publicación (literatura, danza, música).

En Bosón estos atributos se definen como recursos adicionales y se configuran luego de las instancias y las actividades. Para cada actividad se pueden definir recursos. Del recurso se conoce la tabla y columna de la cual se extrae la información, como se muestra en la Figura 12.

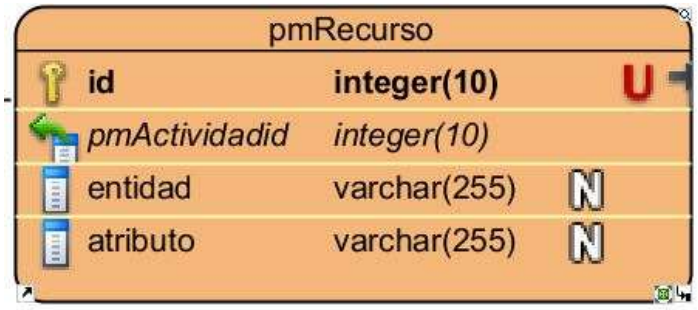

Figura 12 Entidad pmRecurso

Cuando ocurre un evento en el sistema y se comprueba si se corresponde con una actividad, también se buscan los recursos asociados a la actividad. Si existen se guardan en la tabla pmAtributo mostrada en la Figura 13 el identificador del evento y el recurso, el nombre del campo y el valor que tiene para la instancia de proceso del evento. De esta forma se asegura que el registro de eventos contará con información adicional acerca de las actividades que se ejecutan.

Figura 13 Entidad pmAtributo

2.3.4 Relación entre las entidades

Cuando se realiza la configuración se comprueba que existe una relación entre la tabla de la instancia de proceso y la tabla de la actividad, o la tabla del recurso. De lo contrario no habría forma de obtener los datos de esas tablas a partir de la instancia. Para resolver este problema en Bosón se implementa el algoritmo de Búsqueda a lo ancho (BFS por sus siglas en inglés). Dicho algoritmo actúa sobre un conjunto de elementos representándolos como nodos de un grafo

y a partir del nodo raíz busca sistemáticamente el camino mínimo hacia sus nodos hojas (Cormen, y otros, 2001). Un grafo de ejemplo es mostrado en la Figura 14.

En Bosón el algoritmo recibe como parámetros dos entidades y trata de hallar un camino entre ellas. Si lo encuentra significa que las entidades están relacionadas y desde una se puede acceder a la información de la otra. Tomando como ejemplo las tablas Usuario y Publicación, a través de la búsqueda se halla el camino mínimo entre estas entidades.

Figura 14 Base de datos mapeada como grafo por el algoritmo BFS

La implementación del algoritmo no solo se limita a buscar, el servicio brinda más opciones como obtener los caminos entre dos entidades o las consultas en DQL que se ejecutaron entre las entidades. El BFS es costoso computacionalmente comparado con otros algoritmos de búsqueda como Búsqueda en profundidad, pero para el caso de Bosón el consumo solo es alto en el momento de realizar la pre configuración. En este instante se buscan las entidades del negocio en un conjunto numeroso, pero con la obtención de los caminos mínimos entre ellas se reducen los tiempos de consultas porque ya la base de datos ha sido mapeada como un grafo cuyas conexiones son mínimas.

2.3.5 Generación del registro de eventos en XES

Como último paso en la propuesta de solución está la generación del registro de eventos y su exportación al estándar XES. Una vez que han sido almacenados los eventos y recursos en sus respectivas tablas, el usuario que desee realizar un análisis de los procesos en su sistema puede solicitar un registro de los eventos comprendidos en un rango de fecha y asociados a un proceso. Para ello se recogen en la interfaz de usuario del Componente de trazas los valores de la fecha

de inicio, fecha de fin y la instancia de proceso. Estos datos se envían a un controlador encargado de procesar la información y devolver una respuesta en dependencia de los resultados obtenidos.

Primeramente se debe comprobar que el rango de fechas sea válido, o sea que la fecha de fin sea mayor que la de inicio. Si las fechas son válidas se procede a buscar los eventos que coincidan con los criterios de fecha y proceso. Si se encuentran eventos se organizarán según la instancia de proceso a la que se relacionan y con ellos se construirá el archivo XES.

Para que el registro sea válido para algoritmos o técnicas de Minería de procesos debe referenciar las extensiones del estándar, contener una serie de atributos globales y definir clasificadores para las trazas y/o eventos. En el caso del registro a generar con Bosón las extensiones utilizadas son: Concept, Time, Lifecycle, Organizational y Micro. Los atributos globales son elementos comunes en todas las trazas o los eventos; para la solución se definen los siguientes atributos a nivel de eventos: nombre, marca de tiempo, usuario (que ejecutó el evento), rol del usuario, nivel del evento y estado. Los clasificadores asignan un identificador a los eventos para realizar comparaciones entre ellos (Günther, y otros, 2014), para la solución los eventos se clasifican por su nombre y estado.

Con esta información se confecciona el registro de eventos que recibe el usuario, y las herramientas de Minería de procesos pueden descubrir modelos de procesos o realizar otras actividades con el registro generado en Bosón.

2.4 Requisitos de software

Los requisitos de software definen funciones de un sistema y el comportamiento que debe tener el mismo. Permiten organizar y dirigir el trabajo hacia el cumplimiento de sus objetivos.

2.4.1 Requisitos funcionales

Los requisitos funcionales se encapsulan empleando las Historias de usuario por ser parte de la disciplina de modelado en la Metodología utilizada. En el caso de la solución propuesta se definieron 11 historias de usuario, de ellas se describen las más relevantes (*).

1. (*) Adicionar instancia de proceso
2. (*) Adicionar actividad
3. (*) Adicionar recurso
4. Buscar actividad
5. Eliminar actividad
6. Eliminar instancia de proceso
7. (*) Exportar eventos a formato XES

8. (*) Generar registro de eventos
9. Listar actividad
10. Listar evento
11. Listar instancia de proceso

Requisito funcional: Adicionar instancia de proceso

Número: 31	**Nombre del requisito:** Adicionar instancia de proceso
Programador: Pavel Eduardo Barzaga De La Paz	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 28 horas
Riesgo en Desarrollo: Poca experiencia de los estudiantes en las tecnologías de desarrollo del proyecto.	**Tiempo Real:** N/A
Descripción: Este requisito se encarga de adicionar una instancia de proceso al sistema, que es conjunto de actividades para realizar una tarea. Para ello se necesita: -El nombre del proceso -La tabla que será la instancia del proceso	

Requisito funcional: Adicionar actividad

Número: 22	**Nombre del requisito:** Adicionar actividad
Programador: Pavel Eduardo Barzaga De La Paz	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 28 horas
Riesgo en Desarrollo: Poca experiencia de los estudiantes en las tecnologías de desarrollo del proyecto.	**Tiempo Real:** N/A
Descripción: Este requisito se encarga de adicionar una actividad en el sistema, que son las acciones que se ejecutan dentro de un proceso, para ello necesita: -Nombre de la actividad -Entidad de la actividad -Tipo de operación -Proceso al que se relaciona	

Requisito funcional: Adicionar recurso

Número: 26	**Nombre del requisito:** Adicionar recurso
Programador: Pavel Eduardo Barzaga De La Paz	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 28 horas
Riesgo en Desarrollo: Poca experiencia de los estudiantes en las tecnologías de desarrollo del proyecto.	**Tiempo Real:** N/A
Descripción: Este requisito se encarga de registrar un recurso en el sistema, que es un valor asociado a una actividad, para ello necesita: -La actividad con la que se relaciona -La tabla de la que se extraerá el recurso -La columna de la tabla de la que se extraerá el recurso	

Requisito funcional: Exportar eventos a formato XES

Número: 27	**Nombre del requisito:** Exportar eventos a formato XES
Programador: Pavel Eduardo Barzaga De La Paz	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 28 horas
Riesgo en Desarrollo: Poca experiencia de los estudiantes en las tecnologías de desarrollo del proyecto.	**Tiempo Real:** N/A
Descripción: Este requisito se encarga de exportar al estándar XES los eventos, que son ocurrencias de actividades relacionadas a un proceso, para ello se debe definir un rango de fecha, insertando los valores: -Una fecha de inicio -Una fecha de fin -El identificador de la instancia de proceso	

Requisito funcional: Generar registro de eventos

Número: 32	**Nombre del requisito:** Generar registros de eventos
Programador: Pavel Eduardo Barzaga De La Paz	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 28 horas
Riesgo en Desarrollo: Poca experiencia de los estudiantes en las tecnologías de desarrollo del proyecto.	**Tiempo Real:** N/A
Descripción: Este requisito se encarga de registrar los eventos que ocurren en el sistema, los eventos se corresponden con la ejecución de actividades de un proceso.	

2.4.2 Requisitos no funcionales

Se corresponden con los requisitos definidos para Bosón en la Especificación de requisitos de software (Preval, 2015). A continuación se describen los requisitos y se especifica cuáles son contemplados en la solución.

Confiabilidad:

- El sistema concederá acceso a cada usuario autenticado solo a las funcionalidades que le estén permitidas de acuerdo a la configuración del sistema.
- El sistema registrará las trazas de las operaciones realizadas por los usuarios en cada momento.
- El sistema mostrará información detallada de los errores en el sistema solo en el entorno de desarrollo.

La confiabilidad se garantiza capturando y manejando los errores mediante las clases controladoras. Solamente el usuario con los privilegios adecuados puede realizar la configuración de los procesos y actividades.

Usabilidad:

- Los componentes visuales del sistema deben brindar al usuario una experiencia agradable y atractiva visualmente.
- La interfaz debe permitir la interacción táctil de los usuarios de dispositivos móviles.
- Permitir abrir varias aplicaciones a la vez en diferentes pestañas.

Eficiencia:

- El sistema no excede 1 segundo para efectuar acciones de salvar información (esta cifra no incluye los retardos por concepto de tráfico de red).
- El sistema no excede un 1 segundo de respuesta al efectuar acciones de cargar registros que conlleven la ejecución de algoritmos que consuman recursos y tiempo.

Mantenibilidad:

- El sistema almacena trazas de las excepciones ocurridas.
- El sistema almacena el registro de todo lo ocurrido durante las ejecuciones.

La mantenibilidad se garantiza registrando en el sistema no solo las trazas de los eventos, también las trazas de las acciones ejecutadas en el sistema.

Portabilidad:

- La aplicación se ejecuta en diferentes sistemas operativos.

- El sistema utiliza se conecta a múltiples gestores de base de datos.
- El sistema se integra con otras aplicaciones independientemente de la tecnología con la cual fue desarrollada.

La portabilidad se garantiza cuando el componente guarda las trazas independientemente del tipo de gestor de base de datos que se utiliza, además el componente es reutilizable y se integra con otros sistemas.

2.5 Modelo de diseño

El modelo de diseño se compone de una serie de productos de trabajo que sirven de guía para la implementación de la solución, como los diagramas de clase o el modelo de datos.

2.5.1 Patrón arquitectónico

La base de Bosón es el *framework* Symfony 2 por tanto utiliza el patrón Modelo Vista Controlador como patrón arquitectónico. Este es el encargado de separar los datos de la aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos (Eguiluz, 2013):

- Modelo: está compuesto por datos, reglas de negocio y las funcionalidades correspondientes para la comunicación con el encargado de persistir los datos Doctrine.
- Controlador: gestiona las entradas y las respuestas del sistema al usuario.
- Vista: muestra la información del modelo al usuario.

2.5.2 Patrones de diseño

En el diseño que propone este trabajo, se utilizan algunos patrones de diseño del Grupo de Cuatro (GoF por sus siglas en inglés) y Patrones Generales de Software para Asignación de Responsabilidades (GRASP por sus siglas en inglés) para solucionar y/o evitar diferentes problemas que pudieran surgir durante la implementación de la solución.

Patrones GRASP

- Experto: propone que la clase que contenga toda la información necesaria será la responsable de la creación de un objeto o la implementación de un método. El comportamiento se distribuye entre las que contienen la información requerida, siendo más fáciles de entender, mantener y ampliar, aumentando sus posibilidades de reutilización (Larman, 1999). Se observa el uso de este patrón en todas las clases a utilizar en la solución ya que cada clase conoce su información y es la encargada de implementar las funcionalidades que brindan información de las mismas. En la Figura 15 se muestra como ejemplo la entidad pmActividad la cual posee toda la información acerca de las actividades.

Figura 15 Clase pmActividad

- Controlador: el patrón controlador funciona como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la interfaz quien recibe los datos del usuario y los envía a las distintas clases según el método invocado (Larman, 1999). Como ejemplo se muestra en la Figura 16 la clase controladora pmProcesoController, encargada de llevar el control de todos los eventos relacionados con el negocio e implementa las funcionalidades que responden a las peticiones del usuario.

Figura 16 Clase controladora pmProcesoController

- Bajo acoplamiento: este patrón expresa que entre las clases deberán existir pocas ataduras, es decir, estas estarán lo menos relacionadas posible de forma tal que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases, incrementando la reutilización y disminuyendo la

dependencia entre las clases (Larman, 1999). El uso de este patrón se evidencia en la poca relación existente entre las clases que conforman el componente.

- Alta cohesión: propone que la información que almacena una clase debe de ser coherente y debe estar, en la medida de lo posible relacionada con la clase. Al realizar un cambio en una clase con alta cohesión, todos los métodos que pueden verse afectados, estarán a la vista, en el mismo archivo. Incrementa la claridad, la reutilización y la facilidad de comprensión del diseño (Larman, 1999). Ejemplo de esto es el controlador pmActividadController el cual al recibir un cambio en sus atributos requiere una restructuración de los métodos para su correcto funcionamiento.

Patrones GoF

- Proxy: cuando no se desea el acceso directo a un objeto sobre el que se va a aplicar determinada acción, este patrón propone la adición de un nivel que permita solamente el acceso al objeto a través de un objeto proxy sustituto, que será el responsable de controlar o mejorar el acceso al objeto real (Gamma, y otros, 2005). Este patrón se evidencia mediante el uso de Doctrine para el trabajo con la base de datos.

2.5.3 Diagramas de clases

Los diagramas de clases de diseño con estereotipos web ilustran las clases que componen la solución así como las relaciones entre ellas. A continuación se muestran todos los diagramas de la propuesta de solución, de la Figura 17 a la 19, y luego se describen las clases más importantes en ellos.

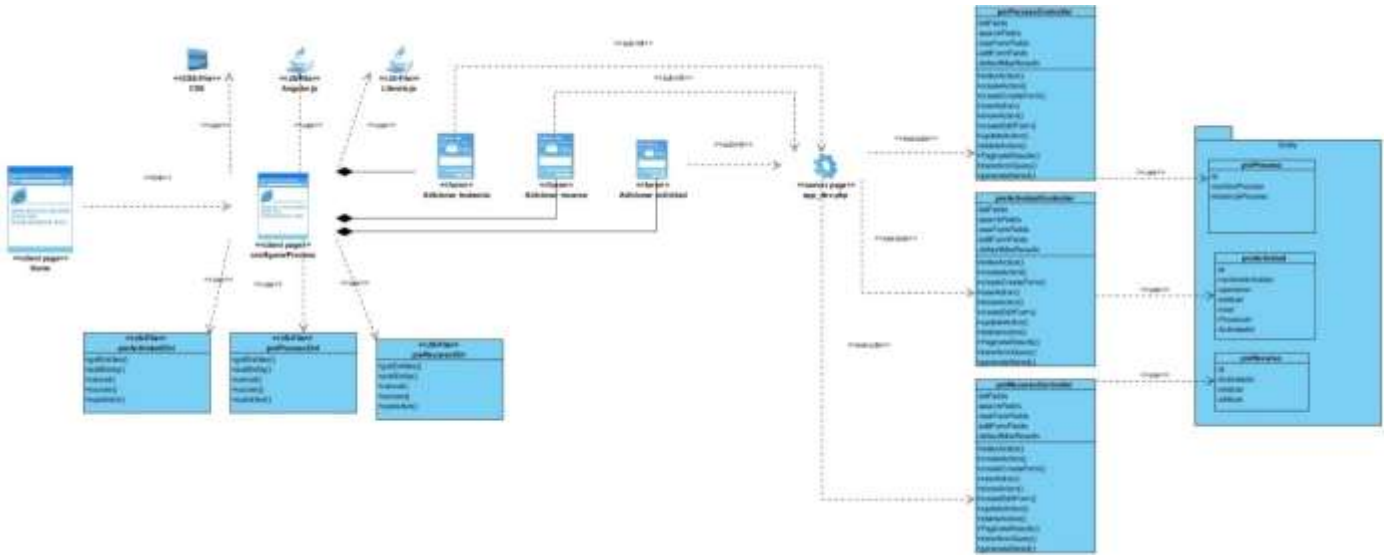

Figura 17 Diagrama de clases Configurar proceso

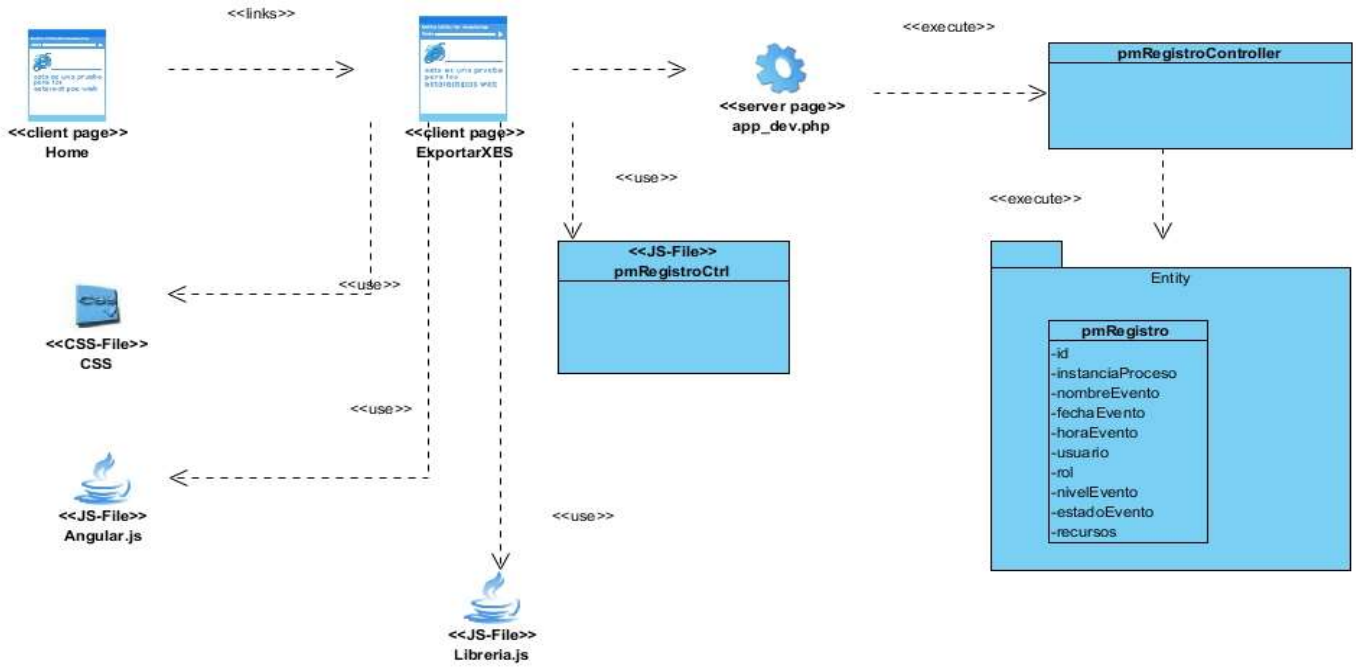

Figura 18 Diagrama de clases Inicio

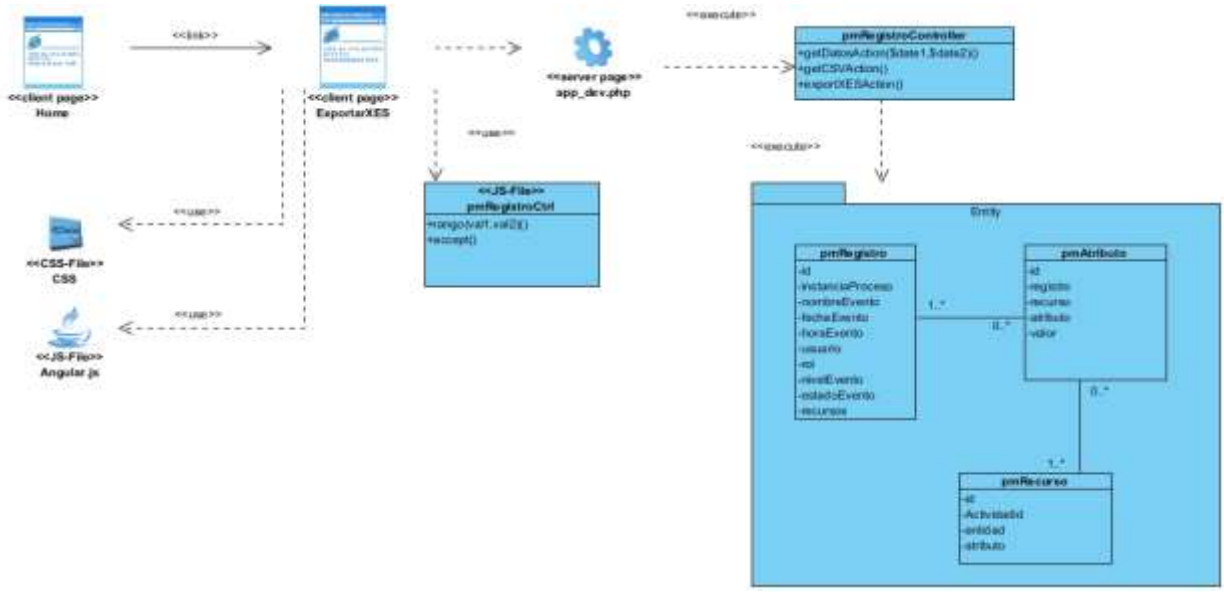

Figura 19 Diagrama de clases Exportar a XES

Descripción de las clases

- pmProcesoController: implementa los métodos para crear, listar y eliminar las instancias de proceso. Provee además de un servicio que busca y devuelve todas las instancias registradas en la base de datos y otro servicio que devuelve los namespaces de las instancias de proceso.
- pmActividadController: implementa los métodos para crear, actualizar, eliminar y listar las actividades. Brinda tres servicios: comprueba si dos entidades están relacionadas, devuelve los namespaces de las actividades y devuelve los datos de las actividades.
- pmRecursoController: implementa los métodos para insertar los recursos de las actividades y un servicio que devuelve los datos de los recursos registrados.
- pmRegistrocontroller: implementa el método para obtener los eventos a incluir en el registro de eventos, según un rango de fechas y una instancia de proceso. También implementa el método que exporta al estándar XES los eventos obtenidos anteriormente.
- pmRegistro: en esta entidad se almacenan los eventos que ocurren en el sistema con los datos necesarios para ser exportados a XES: nombre, instancia de proceso, proceso, marca de tiempo,usuario, rol, nivel y estado del evento.

2.5.5 Modelo de datos

En la Figura 20 se muestra el modelo de datos de la solución en el cual se muestran las tablas que se corresponden con las nuevas entidades de Bosón, y las relaciones entre ellas. Previamente se había descrito cada una de estas tablas con respecto a su función y los datos que almacenan.

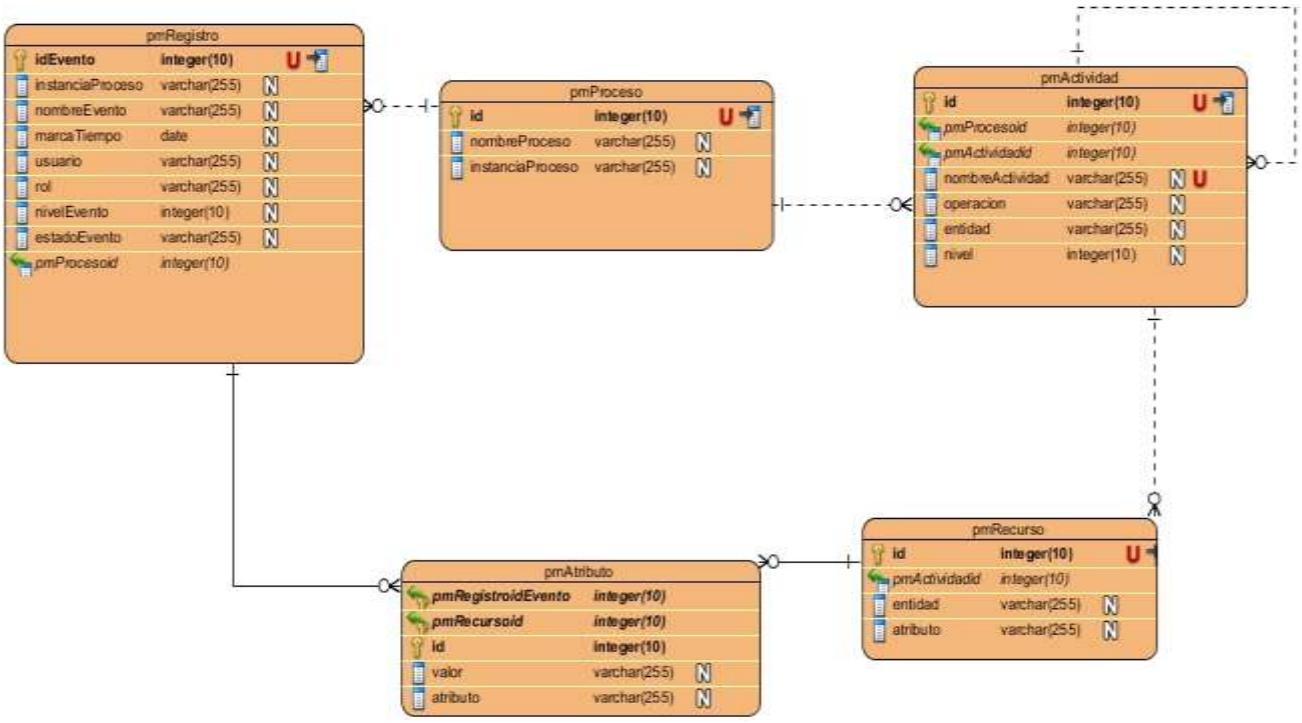

Figura 20 Modelo de datos

Conclusiones del capítulo

La definición de nuevas entidades en el componente de trazas de Bosón permitió establecer una forma organizada de registrar la información de los eventos en el sistema, con este nuevo formato se puede confeccionar un registro de eventos.

Se elaboraron además los productos de trabajo relacionados con la fase de ejecución, sirviendo estos como guía para la implementación del sistema.

La aplicación de patrones de diseño y arquitectónicos permitió brindar robustez al código y se proporcionó una estructura común conocida por los programadores. Dicha estructura facilita tareas como el mantenimiento y mejora la reusabilidad de la solución.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBAS

Introducción

En el presente capítulo se describe el modelo de implementación de la solución propuesta y el estándar de codificación utilizado. Se explican algunos algoritmos de interés para el desarrollo de la solución. Por último se realizan las pruebas para validar la solución.

3.1 Modelo de implementación

El modelo de implementación describe cómo los elementos del modelo de diseño se implementan en términos de componentes. Describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración disponibles en el entorno de implementación y en el lenguaje de programación utilizados; y cómo dependen los componentes unos de otros.

3.1.1 Diagrama de componentes

El diagrama de componentes mostrado en la Figura 21, representa la relación entre el componente de Trazas donde se implementa la solución y los componentes Doctrine y Symfony. Estos proveen los servicios para acceder a la base de datos y obtener los datos de los usuarios registrados en el sistema.

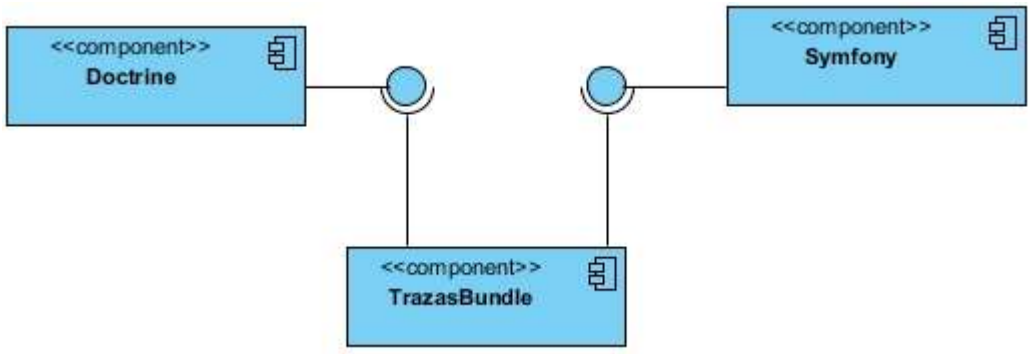

Figura 21 Diagrama de componentes

3.1.2 Algoritmos de interés

La relación entre las entidades y la obtención de los eventos constituyen dos de las funcionalidades claves en la solución. A continuación se describen con mayor nivel de detalle.

1. **Relación entre entidades**

Cuando se realiza la pre configuración en el sistema se selecciona la tabla sobre la que ocurre una acción. Si se desea insertar una actividad se comprueba que su tabla posee relación con la tabla de la instancia de proceso. Esta comprobación se realiza para determinar si desde la instancia se puede acceder a los valores y atributos de la actividad, de no ser posible no se

realiza la inserción. Para determinar esta relación se emplea el servicio Entities Graph el cual implementa el algoritmo BFS y otros métodos útiles.

Como primer paso, cuando se inserta una actividad o un recurso se capturan en el controlador de Javascript los namespaces de las entidades. A continuación estos datos son enviados mediante el protocolo HTTP al controlador del servidor encargado de manejar las actividades. Este posee el método que se encarga de devolver una respuesta en caso que las entidades se relacionen o no. En la función checkDataAction se realiza una llamada al servicio Entities Graph y se solicita el método canJoined para evaluar si existe relación o no entre las entidades. Este método emplea el algoritmo BFS, implementado dentro del servicio y mostrado en la Figura 22.

```
private function bfs($graph, $start, $end)
{
    $queue = new \SplQueue();
    $queue->enqueue($start);

    $visited = array_fill(0, count($graph), false);


    while ($queue->count() > 0) {
        $node = $queue->dequeue();
        # We've found what we want
        if ($node === $end) {
            return true;
        }
        foreach ($graph[$node] as $index => $neighbour) {
            if ($graph[$node][$index] != null && !$visited[$index]) {
                # Mark neighbour visited
                $visited[$index] = true;
                # Enqueue node
                $queue->enqueue($index);
            }
        };
    }
    return false;
}
```

Figura 22 Implementación del BFS

La función recibe como parámetros un grafo que contiene todas las tablas de la base de datos mapeadas como nodos; un nodo inicial que se corresponde con la tabla de la instancia de proceso y como nodo destino la tabla de la actividad que se quiere insertar. Desde el nodo inicial se recorren todas sus aristas para descubrir los nodos vecinos y se calcula la distancia hacia ellos. Esta distancia es el menor número de aristas desde el inicio hasta sus vecinos. Los nodos vecinos son marcados como visitados y se ubican en una cola. El algoritmo continúa recorriendo

los nodos en la cola hasta que uno de ellos coincida con el nodo destino, en caso de no hallar el destino significa que no se relaciona con el nodo inicio.

Según el resultado del algoritmo, el controlador en el servidor devuelve una respuesta al controlador de la vista y se le muestra un mensaje al usuario que está configurando el sistema, indicándole si la relación entre las tablas existe o no.

2. Confección del registro de eventos

Para obtener los eventos primeramente se selecciona un rango de fechas y el identificador del proceso a examinar. Las fechas indican el período de tiempo para el cual se observará el comportamiento del sistema. Estos valores se capturan en el controlador de la vista y son enviados al controlador del registro, el cual tiene implementado el servicio que devolverá los eventos.

Cuando los datos llegan al controlador se comprueban varias condiciones antes de obtener los eventos. Primeramente, que la fecha de inicio de búsqueda sea menor que la fecha de fin, de lo contrario se retorna un error indicando que las fechas son incorrectas. Si no hay problemas se buscan todos los eventos guardados en la tabla pmRegistro que estén comprendidos entre el rango de fechas. Si no se hallan eventos se le devuelve una respuesta al controlador de la vista indicando que no hay eventos entre las fechas señaladas. De lo contrario se organizan los eventos según el identificador de la instancia de proceso y se envían al método que construye el archivo en el formato XES.

Para confeccionar el archivo se sigue la estructura de los documentos creados en XES. Inicialmente se construye la cabecera con los valores de la versión y las características de XES. A continuación se insertan de las extensiones el nombre, el prefijo que utilizan y la URI donde se encuentran disponibles. En la Figura 23 se muestra la cabecera y una de las extensiones insertadas.

```
$rootNode = new \SimpleXMLElement(
    '<?xml version="1.0" encoding="UTF-8" ?><log xes.version="2.0" xes.features="arbitrary-depth"></log>'
);

$concept = $rootNode->addChild('extension');
$concept->addAttribute('name', 'Concept');
$concept->addAttribute('prefix', 'concept');
$concept->addAttribute('uri', 'http://www.xes-standard.org/concept.xesext');
```

Figura 23 Cabecera

Luego se insertan los atributos globales de los eventos. Poseen una clave que identifica la extensión a la cual referencia el atributo, y el valor por defecto que tendrá el atributo. En la Figura 24 se muestran los atributos del nombre y marca de tiempo del evento.

```
$string = $global->addChild('string');
$string->addAttribute('key', 'concept:name');
$string->addAttribute('value', '');

$tstamp = $global->addChild('date');
$tstamp->addAttribute('key', 'time:timestamp');
$tstamp->addAttribute('value', '1970-01-01T00:00:00.000+00:00');
```

Figura 24 Atributos globales

Después se insertan los clasificadores y sus dos atributos, el nombre del clasificador y los atributos del evento que conforman el clasificador. Para el caso de Bosón se había mencionado que los eventos se clasifican por sus nombres y estado, como se muestra en la Figura 25.

```
$clasifier1 = $rootNode->addChild('classifier');
$clasifier1->addAttribute('name', 'event');
$clasifier1->addAttribute('keys', 'concept:name lifecycle:complete');
```

Figura 25 Clasificadores

Por último, se recorren los eventos y se solicita la información necesaria a cada uno para otorgarle valor a sus atributos. En la Figura 26 se aprecia cómo los atributos de rol y usuario se reciben del evento almacenado en la variable.

```
$resource = $evento->addChild('string');
$resource->addAttribute('key', 'org:resource');
$resource->addAttribute('value', $d['usuario']);
$role = $evento->addChild('string');
$role->addAttribute('key', 'org:role');
$role->addAttribute('value', $d['rol']);
```

Figura 26 Atributos del evento: rol y usuario

Una vez conformado el archivo se exporta al formato .xes y el controlador del servidor envía una respuesta satisfactoria a la vista para informar al usuario que la operación fue un éxito.

3.1.3 Diagrama de despliegue

A continuación se muestra en la Figura 27 el diagrama de despliegue de la solución implementada. En la computadora cliente se accede al sistema mediante el protocolo HTTP. El sistema se halla instalado y configurado en un servidor web, y maneja las peticiones que realiza el cliente. La información del sistema, de las acciones ejecutadas y del negocio se halla almacenada en la base de datos.

Figura 27 Diagrama de despliegue

3.2 Estándares de codificación

Los estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. El uso de estándares de codificación permite lograr un código más legible y reutilizable, de tal forma que se pueda aumentar su mantenibilidad a lo largo del tiempo (Pérez Alfonso, 2012).

3.2.1 Nomenclaturas utilizadas

Para las nomenclaturas de las clases se emplea la notación CamelCase en su variante lowerCamelCase, que se aplica a frases o palabras compuestas, para esta variante la primera palabra siempre inicia en minúsculas.

- Nomenclatura de las clases: los nombres de las clases comienzan con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación lowerCamelCase. Ejemplo: pmActividad.
- Nomenclatura de los controladores: los controladores después del nombre llevan el sufijo Controller. Por ejemplo pmActividadController.
- Nomenclatura de los controladores de las vistas: después del nombre llevan el sufijo Ctrl. Ejemplo: pmRegistroCtrl.
- Nomenclatura de los servicios: después del nombre llevan el sufijo Svc. Ejemplo: pmRecursoSvc.
- Nomenclatura de las variables: El nombre a emplear para las variables se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se empleará notación lowerCamelCasing y comenzando con un prefijo según el tipo de datos. Ejemplo: $proceso y $nombreEvento.
- Nomenclatura de las funciones: El nombre a emplear para las funciones se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se empleará notación lowerCamelCase y en caso de ser una acción de la clase controladora se debe especificar el nombre de dicha acción en minúscula y seguido del sufijo Action. Ejemplo: getDatosAction.

3.2.2 Estilo de código

Como Bosón está implementado sobre Symfony2, se emplea el mismo estilo de código del *framework* base (Sym16).

- Añadir un solo espacio después de cada delimitador coma.
- Añadir un solo espacio alrededor de los operadores (==, &&, ...).
- Añadir una coma después de cada elemento del arreglo en un arreglo multilínea, incluso después del último.
- Añadir una línea en blanco antes de las declaraciones return, a menos que el valor devuelto solo sea dentro de un grupo de declaraciones (tal como una declaración if).
- Usar llaves para indicar la estructura del cuerpo de control, independientemente del número de declaraciones que contenga.
- Definir una clase por archivo. Esto no se aplica a las clases ayudante privadas, de las cuales no se tiene la intención de crear una instancia desde el exterior y por lo tanto no les preocupa la norma PSR-0.
- Declarar las propiedades de clase antes que los métodos.
- Declarar primero los métodos públicos, luego los protegidos y finalmente los privados.
- Usar paréntesis cuando se instancie una clase a pesar del número de argumentos que su constructor posea.

3.3 Métricas de diseño de software

Las métricas de diseño de software permiten medir de forma cuantitativa la calidad de los atributos internos del software. Esto proporciona una vía para evaluar la calidad durante el desarrollo del sistema. A nivel de componentes se concentran en las características internas de los componentes del software con medidas que ayudan a juzgar la calidad de un diseño a nivel de componente (Cuf12). Para el caso de Bosón se aplica Tamaño operacional de clases y Relación entre clases.

3.3.1 Tamaño operacional de las clases (TOC)

Está dado por el número de métodos u operaciones (de instancia privada y heredada) que están encapsulados dentro o por una clase. Evalúa los siguientes atributos de calidad mostrados en la Tabla 4.

Tabla 4 Atributos de calidad que evalúa TOC

Atributo	**Modo en que lo afecta**
Responsabilidad	Aumento del TOC provoca aumento de la responsabilidad asignada a la clase.

Complejidad de implementación	Aumento del TOC provoca aumento de la complejidad de implementación de la clase.
Reutilización	Aumento del TOC provoca disminución del grado de reutilización de la clase.

Para evaluar los atributos se emplean los siguientes criterios y categorías de evaluación presentes en la Tabla 5.

Tabla 5 Criterios y categorías de evaluación de TOC

Atributo	Categoría	Criterio
Responsabilidad	Baja	< =Prom
	Media	Entre Prom. y 2* Pom
	Alta	> 2* Prom
Complejidad de implementación	Baja	< =Prom
	Media	Entre Prom. y 2* Pom
	Alta	> 2* Prom
Reutilización	Baja	> 2* Prom
	Media	Entre Prom. y 2* Pom
	Alta	< =Prom

Después de aplicar la métrica a las principales clases del Componente de trazas con la solución implementada, se obtiene como resultado que un 57% de las clases poseen Responsabilidad y Complejidad bajas, y un 43% de las clases poseen Reutilización alta y media, como se aprecia de la Figura 28 a la Figura 30.

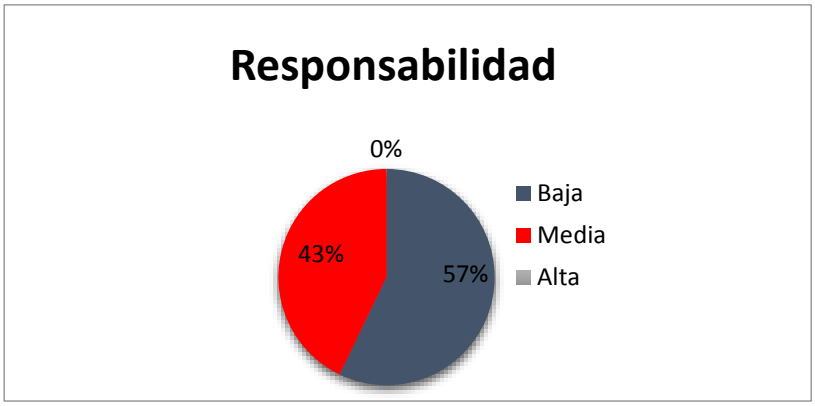

Figura 28 Resultados del análisis de responsabilidad

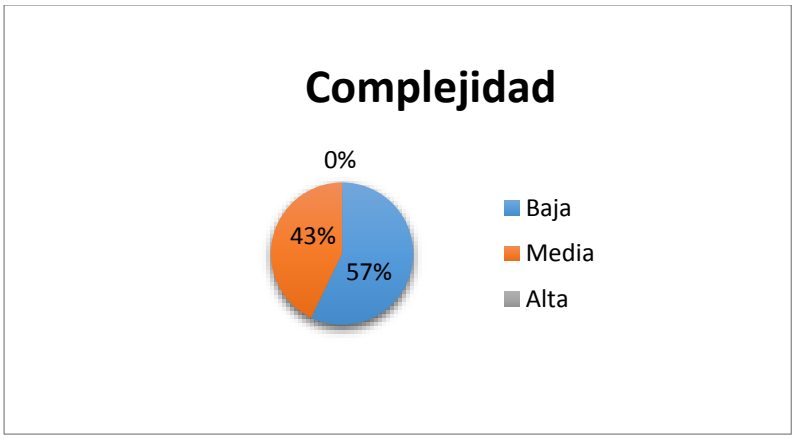

Figura 29 Resultados del análisis de complejidad

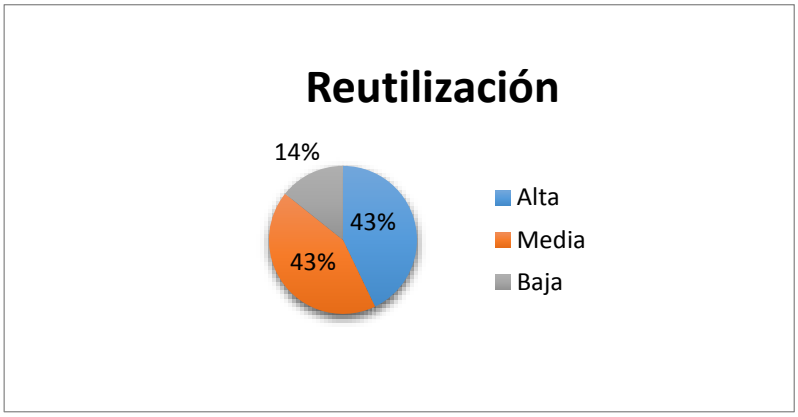

Figura 30 Resultados del análisis de reutilización

En la Tabla 6 se observan con mayor nivel de detalle los resultados de aplicar TOC.

Tabla 6 Resultados de aplicar TOC

Clase	**Cantidad de Procedimientos**	**Responsabilidad**	**Complejidad**	**Reutilización**
pmProcesoController	12	Baja	Baja	Alta
pmActividadController	15	Media	Media	Media
pmRegistroController	7	Baja	Baja	Alta
pmRecursoController	7	Baja	Baja	Alta
pmActividad	20	Media	Media	Media
pmProceso	12	Baja	Baja	Alta

pmRecurso	7	Baja	Baja	Alta
pmRegistro	21	Media	Media	Media
pmAtributo	9	Baja	Baja	Alta
pmProcesoCtrl	18	Media	Media	Media
pmActividadCtrl	22	Media	Media	Media
pmRecursoCtrl	18	Media	Media	Media
pmRegistroCtrl	9	Baja	Baja	Baja
DatoListener	9	Baja	Baja	Baja

3.3.2 Relaciones entre clases (RC)

Esta métrica se define con el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad presentes en la Tabla 7.

Tabla 7 Atributos de calidad evaluados por la métrica RC

Atributo	**Modo en que lo afecta**
Acoplamiento	Aumento de las relaciones entre clases provocan aumento del acoplamiento de la clase.
Complejidad de mantenimiento	Aumento de las relaciones entre clases provocan aumento de la complejidad de mantenimiento de la clase.
Reutilización	Aumento de las relaciones entre clases provocan disminución del grado de reutilización de la clase.
Cantidad de pruebas	Aumento de las relaciones entre clases provocan aumento de la cantidad de pruebas de unidad necesarias para probar una clase.

Para la evaluación de dichos atributos de calidad, se definen los siguientes criterios y categorías de evaluación mostrados en la Tabla 8:

Tabla 8 Criterios y categorías de evaluación de RC

Atributo	**Categoría**	**Criterio**
Acoplamiento	Ninguno	0

	Baja	1
	Media	2
	Alta	>2
Complejidad de mantenimiento	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Reutilización	Baja	Entre Promedio y 2*Promedio
	Media	>2*Promedio
	Alta	<=Promedio
Cantidad de pruebas	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio

Tras evaluar las relaciones de las clases en el Componente de Trazas con la solución implementada, se obtiene como resultado que un 64% de las clases posee Acoplamiento bajo, Complejidad de mantenimiento baja, Cantidad de pruebas baja y Reutilización alta como se muestra de la Figura 31 a la Figura 34.

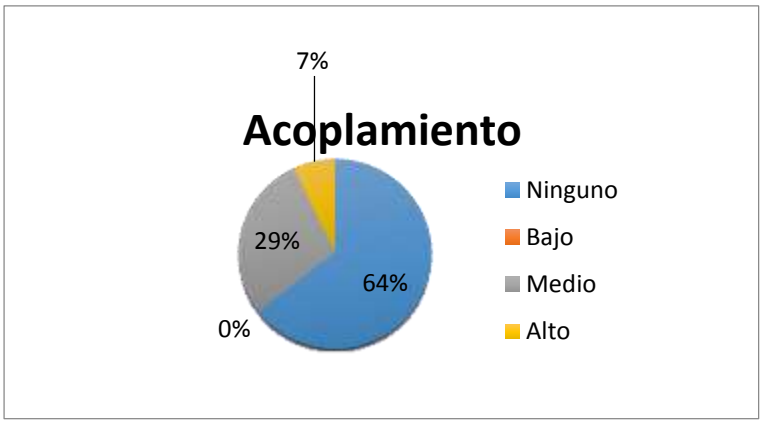

Figura 31 Resultado del análisis de acoplamiento

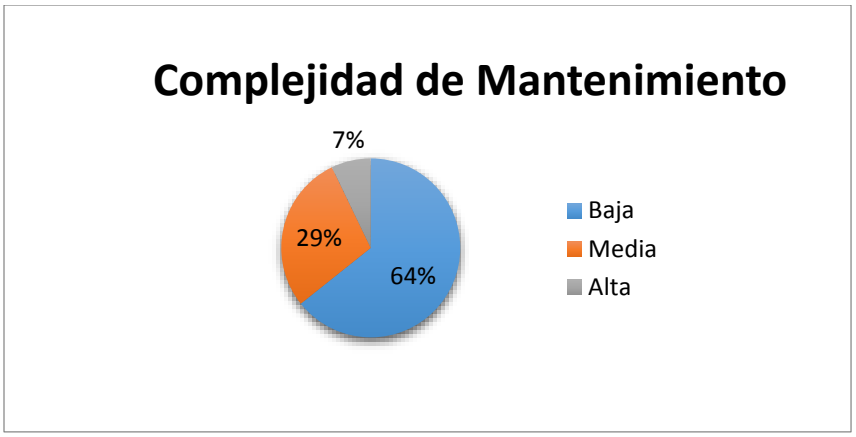

Figura 32 Resultado del análisis de complejidad de mantenimiento

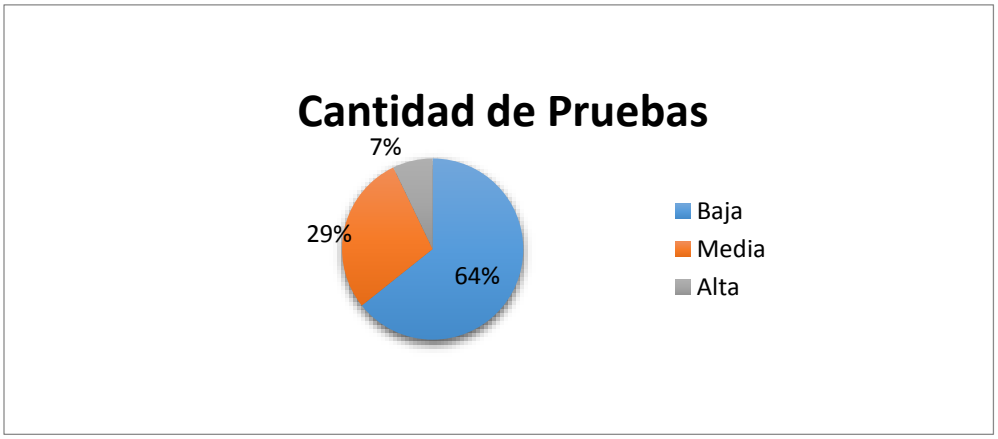

Figura 33 Resultado del análisis de cantidad de pruebas

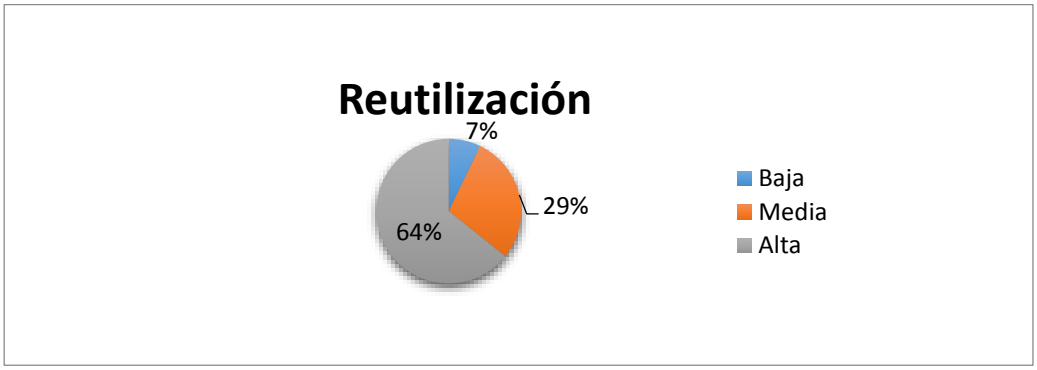

Figura 34 Resultado del análisis de reutilización

En la Tabla 9 se observan con mayor nivel de detalle los resultados de aplicar RC.

Tabla 9 Resultado de aplicar RC

Clase	**Cantidad de Relaciones de Uso**	**Acoplamiento**	**Complejidad de mantenimiento**	**Reutilización**	**Cantidad de Pruebas**
pmProcesoController	0	Ninguno	Baja	Alta	Baja
pmActividadController	0	Ninguno	Baja	Alta	Baja
pmRegistroController	0	Ninguno	Baja	Alta	Baja
pmRecursoController	0	Ninguno	Baja	Alta	Baja
pmActividad	3	Alto	Alta	Baja	Alta
pmProceso	2	Medio	Media	Media	Media
pmRecurso	2	Medio	Media	Media	Media
pmRegistro	2	Medio	Media	Media	Media
pmAtributo	2	Medio	Media	Media	Media
pmProcesoCtrl	0	Ninguno	Baja	Alta	Baja
pmActividadCtrl	0	Ninguno	Baja	Alta	Baja
pmRecursoCtrl	0	Ninguno	Baja	Alta	Baja
pmRegistroCtrl	0	Ninguno	Baja	Alta	Baja
DatoListener	0	Ninguno	Baja	Alta	Baja

3.3.3 Conclusiones de evaluar empleando las métricas

Como resultado de aplicar las métricas a la solución desarrollada se obtiene que las clases poseen bajos niveles de responsabilidad dentro del dominio de la solución y baja complejidad de implementación. Los niveles de acoplamiento también son bajos y significa que el grado de

interconexión entre clases es pequeño. Esta característica influye en la reutilización de la solución, a menores niveles de dependencia entre las clases mayor reutilización contándose en este caso con un alto nivel de reutilización. Una complejidad de mantenimiento baja indica que el grado de esfuerzo necesario para desarrollar un arreglo, una mejora o una rectificación de algún error será bajo y tendrá poco impacto en el sistema. El bajo nivel de la cantidad de pruebas indica que el esfuerzo para probar el componente mediante pruebas de calidad será bajo también.

3.4 Pruebas de caja blanca

La prueba de caja blanca es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Mediante estos métodos el ingeniero de software puede obtener casos de prueba que garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo; ejerciten todas las decisiones lógicas en sus vertientes verdaderas y falsas; ejecuten todos los bucles en sus límites y con sus limitaciones operacionales; y ejerciten las estructuras internas de datos para asegurar su validez (Pressman, 2005).

La técnica a aplicar es el camino básico, que permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y utilizar esta medida como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa (Pressman, 2005).

Para obtener el conjunto de caminos independientes se construye el grafo de flujo asociado a una función y se calcula su complejidad ciclomática, para este caso se analiza el método getDatosAction() encargado de obtener los eventos solicitados por el usuario, mostrado en las Figuras 35 y 36. Primeramente se enumeran las sentencias de código.

```
public function getDatosAction($date1, $date2, $idP)
{
    $em = $this->get('doctrine.orm.entity_manager');    1
    $qb = $em->createQueryBuilder();    2

    $d1 = strtotime($date1);
    $f1 = date("Y-m-d H:i:s", $d1);
    $d2 = strtotime($date2);
    $f2 = date("Y-m-d H:i:s", $d2);

    if($f1<$f2){    3

        $qb
            ->select('pmRegistro, partial proceso.{id}')
            ->from('TrazasBundle:pmRegistro', 'pmRegistro')
            ->where('pmRegistro.proceso = :id')
            ->andWhere('pmRegistro.marcaTiempo BETWEEN :fechaInicio AND :fechaFin')
            ->join('pmRegistro.proceso', 'proceso')
            ->setParameter('id', $idP)
            ->setParameter('fechaInicio', $f1)
            ->setParameter('fechaFin', $f2);
        $data = $qb->getQuery()->getArrayResult();    4

        if (count($data) > 0) {    5

            usort(    6
                $data,
                function ($item1, $item2) {
                    if ($item1['instanciaProceso'] == $item2['instanciaProceso']) {
                        return 0;
                    }
```

Figura 35 Fragmento del código del método getDatosAction

```
            $this->createXMLAction($data);  7

            $respuesta = "Eventos procesados";  8
            $codigo = 200;
        } else {
            $respuesta = "No se encontraron los eventos";  9
            $codigo = 500;
        }
    } else {
        $respuesta = "Ha introducido un rango de fechas incorrecto";  10
        $codigo = 500;
    }
    return new JsonResponse($respuesta, $codigo);  11
}
```

Figura 36 Fragmento del código del método getDatosAction

Una vez obtenidas las sentencias se construye el grafo, quedando de la forma mostrada en la Figura 37.

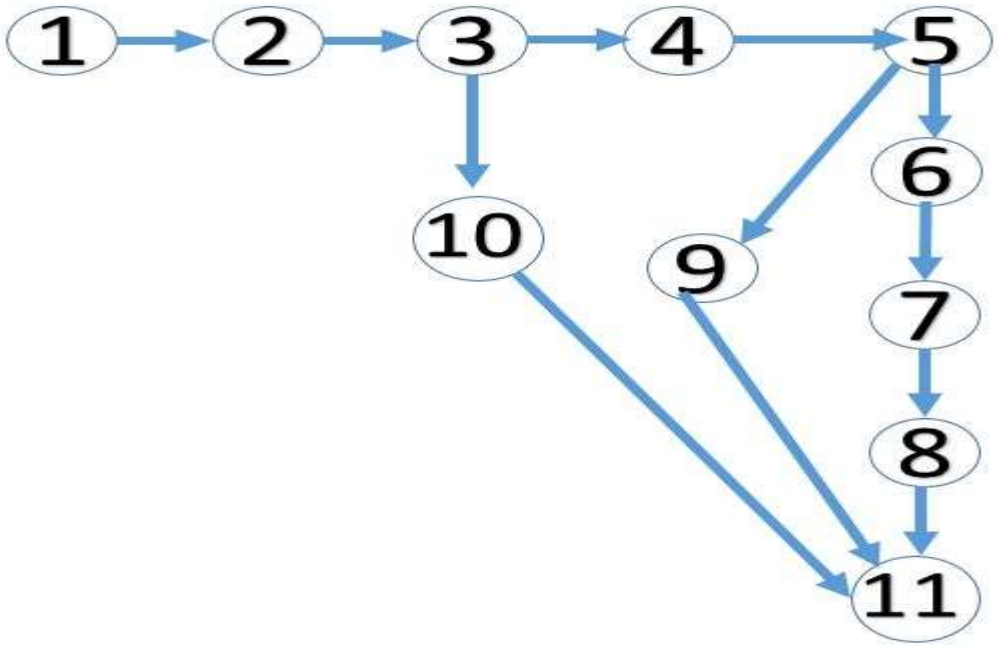

Figura 37 Grafo de flujo asociado a getDatosAction

La complejidad ciclomática es la métrica de software con que se define la cantidad de caminos independientes de cada una de las funcionalidades del programa y provee el límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez (Pressman, 2005).

Se calcula de la siguiente manera:

V(G)=E-N+2 donde E es el número de aristas y N los vértices

V(G)=12-11+2

V(G)=3

Para este caso se obtienen tres posibles caminos independientes y cantidad de pruebas que se deben realizar para comprobar que las sentencias se ejecutan al menos una vez.

Los caminos básicos son:

- Camino básico 1: 1, 2, 3, 4, 5, 6, 7, 8,11
- Camino básico 2: 1, 2, 3, 4, 5, 9,11
- Camino básico 3: 1, 2, 3, 10,11

Luego se elaboran los casos de prueba, quedando de la siguiente manera:

- Caso de prueba del camino 1

  Condición: Si se cumple $d1<$d2

  Resultado esperado: El sistema procede a buscar los eventos

- Caso de prueba del camino 2

  Condición: Si count($data) es vacío

  Resultado esperado: El sistema devuelve un mensaje indicando que no se encontraron los eventos en el rango de fechas

- Caso de prueba del camino 3

  Condición: Si no se cumple $d1<$d2

  Resultado esperado: El sistema devuelve un mensaje indicando que las fechas son incorrectas

Se ejecutan los casos de prueba para comparar los resultados obtenidos contra los esperados. Los resultados obtenidos coinciden con los esperados, por lo que se puede asegurar que todas las sentencias del método se han ejecutado al menos una vez.

3.5 Pruebas de caja negra

Las pruebas de caja negra se centran en los requisitos funcionales. O sea, la prueba de caja negra permite al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa (Pressman, 2005).

El tipo de prueba a aplicar es la partición equivalente. Es un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores (por ejemplo, proceso incorrecto de todos los datos de carácter) que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se

dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar (Pressman, 2005).

En el caso del Componente de trazas de Bosón se dieñaron 10 casos de prueba que pueden ser consultados en el Expediente de Desarrollo de Bosón 2.0. A continuación se muestran los resultados de las pruebas realizadas en la Tabla 10.

No conformidades	Documentación	Aplicación
Primera iteración	5	17
Segunda iteración	0	0

Tabla 10 No conformidades detectadas por iteración

El grafo de la Figura 38 ilustra estos valores.

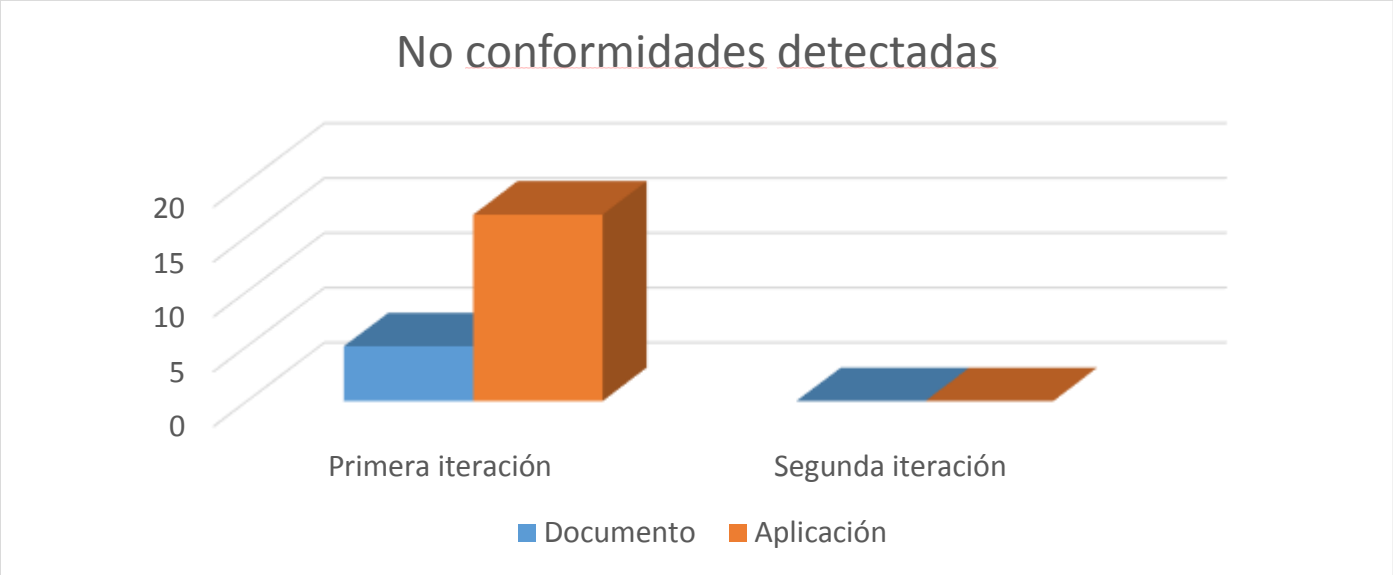

Figura 38 Gráfico de no conformidades por iteración

3.6 Aplicación de la solución en un caso de estudio

Para validar si el registro de eventos confeccionado es factible para aplicarle Minería de procesos, en el Componente de trazas de Bosón se diseña un caso de estudio que simula el comportamiento de los usuarios de un sitio web y las entidades que comúnmente interactúan. El proceso se llama “Gestionar sitio web” y comprende las siguientes actividades:

- Insertar usuario
- Insertar contactos
- Insertar publicación
- Modificar publicación
- Eliminar publicación

La Figura 39 muestra las entidades del proceso y las relaciones entre ellas.

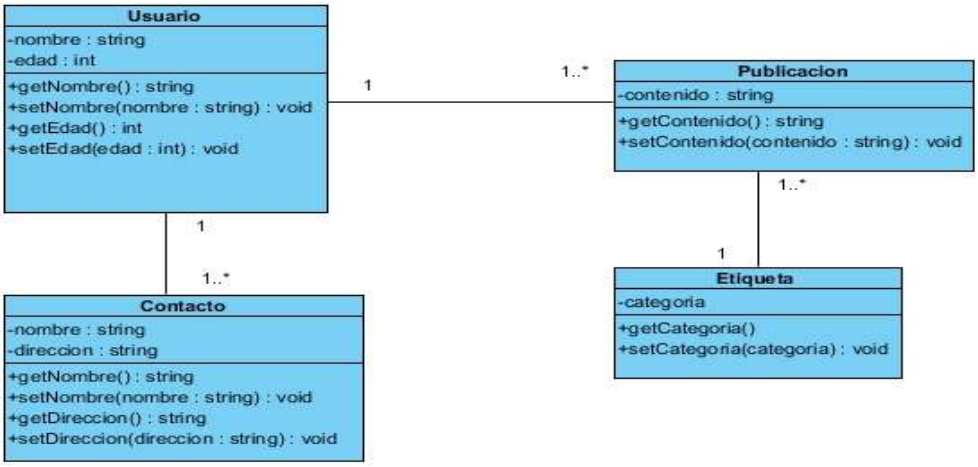

Figura 39 Entidades del proceso Gestionar sitio web

Posterior a la creación de las entidades se emplea la solución para efectuar la configuración inicial. La tabla Usuario se identifica como el objeto del negocio, constituyendo de esa manera la instancia del proceso. Las actividades se insertan y se procede a ejecutarlas para generar y registrar eventos. Una vez terminado el período de prueba y obtenido el registro de eventos, las herramientas de Minería de procesos deben descubrir el proceso desarrollado, con sus cinco instancias y los 43 eventos que se simularon.

El registro de eventos se analiza en el Prom. La herramienta identifica el proceso, las instancias y los eventos ocurridos (Figura 40), representa el flujo de eventos (Figura 41) y obtiene el modelo de procesos para el caso de estudio (Figura 42). Los resultados arrojados por este análisis coinciden con los resultados esperados, por tanto se afirma que el registro de eventos generado por el Componente de trazas es válido para aplicar Minería de procesos.

Figura 40 Información de los eventos descubiertos

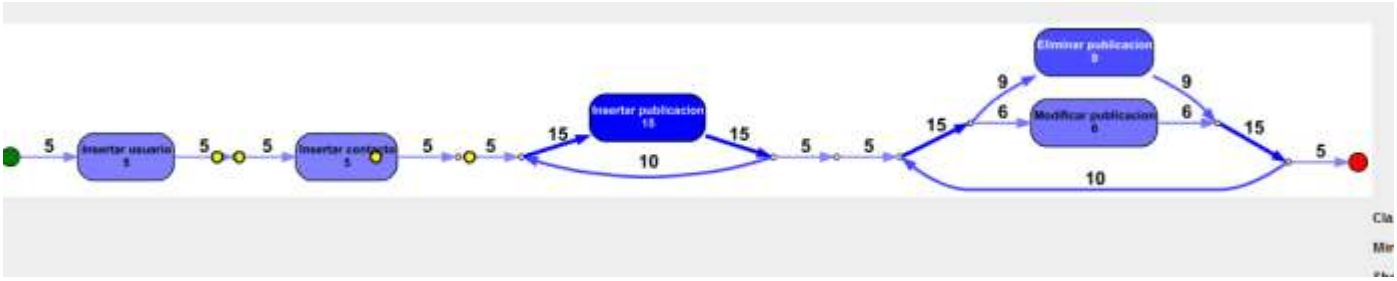

Figura 41 Flujo de eventos reconocido por la herramienta

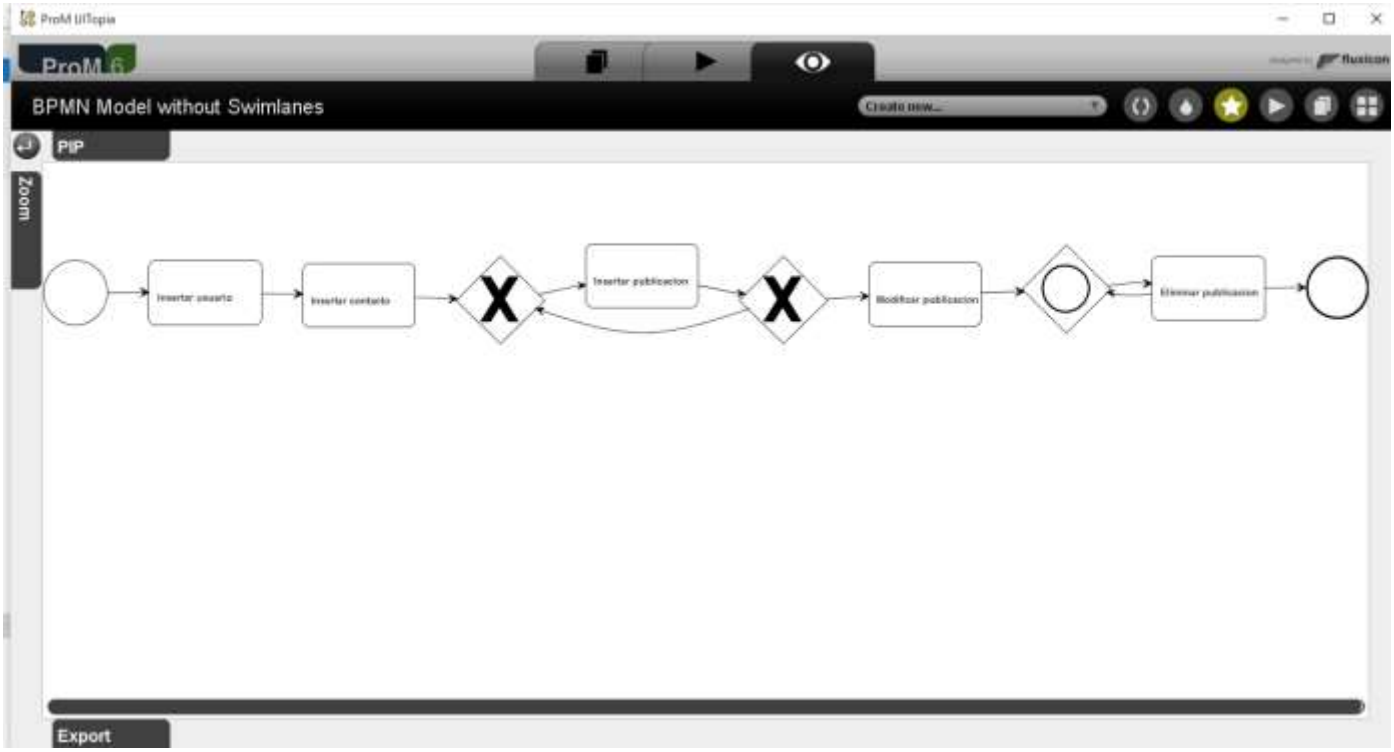

Figura 42 Modelo de proceso del caso de estudio "Gestionar sitio web"

Este resultado también evidencia el cumplimento de los principios planteados por Wil van der Aalst en su propuesta de modelo de eventos (Aalst, 2011):

- P4: Los valores de los atributos deberían ser lo más precisos posibles. Si el valor no tiene la precisión adecuada, esto debería ser indicado explícitamente.

Esto se aplica cuando se guarda la información de cada eventos, si se desconoce el valor de un atributo se le asigna un valor por defecto.

- P6: Los eventos deberían estar al menos parcialmente ordenados. Su orden puede ser almacenado explícitamente (empleando listas) o implícitamente (empleando marcas de tiempo).

En el caso de Bosón los eventos cuentan con la marca de tiempo y están agrupados por trazas que se corresponden con la instancia de proceso a la que están asociados.

- P7: De ser posible, almacenar también información transaccional del evento.

Se emplea la extensión Lifecycle del XES para representar el estado de los eventos iniciados y completados.

Conclusiones del capítulo

El empleo de métricas de software validó el diseño propuesto en el capítulo anterior y demostró que el componente cuenta con un diseño robusto.

El uso de estándares de codificación aseguró una homogeneidad en la nomenclatura de las clases y métodos facilitando su comprensión a otros programadores.

La práctica de pruebas de caja blanca y negra comprobó el cumplimiento de los requisitos funcionales y la correcta ejecución del código.

La modelación de un caso de estudio para un proceso con cinco instancias y 43 eventos generó un registro de eventos. Este registro fue analizado por las herramientas de Minería de procesos y devolvió los resultados esperados, validando la factibilidad del registro de eventos creado en el Componente de trazas de Bosón.

CONCLUSIONES GENERALES

Los enfoques de las soluciones existentes para obtener un registro de eventos a partir de sistemas orientados a datos se basaban en el análisis de trazas desorganizadas y sin los atributos necesarios. Para confeccionar adecuadamente un registro son obligatorios en las trazas los atributos: instancia de proceso, proceso, marca de tiempo y estado. En el *framework* Bosón se aplicó el enfoque de organizar primero la información a registrar y luego capturar los eventos.

Realizar una configuración inicial de las instancias de proceso, las actividades y los recursos reduce el efecto de los problemas asociados a la creación del registro de eventos.

El empleo de patrones de diseño e implementación y la validación empleando métricas y pruebas demostró que se obtuvo una solución adaptable, mantenible y reutilizable.

La aplicación de un caso de estudio aprovechando la solución generó un registro de eventos que las herramientas de Minería de procesos reconocieron y analizaron correctamente, alcanzado los resultados esperados. En consecuencia el registro es factible para ser empleado por diversas técnicas y algoritmos de la Minería de procesos.

La solución implementada permite a los sistemas que se desarrollen sobre Bosón analizar sus procesos y utilizar esta información en la toma de decisiones.

RECOMENDACIONES

Las trazas de acciones y excepciones en Bosón contienen información adicional acerca de los eventos que han ocurrido en el sistema. El modelo de eventos de Wil van der Aalst no contempla este tipo de operaciones en los sistemas que emplean bases de datos. Por tanto se recomienda expandir el modelo para registrar la información de las trazas de acciones y excepciones en los eventos.

Bibliografía

1. *Visual Paradigm.* [En línea] [Citado el: 23 de Febrero de 2016.] (http://www.visual-paradigm.com/product/vpuml/provides/).
2. *Symfony.com.* [En línea] SensioLabs. [Citado el: 1 de Junio de 2016.] http://symfony.com/doc/current/contributing/code/standards.html.
3. *Study.com.* [Online] [Cited: Noviembre 3, 2015.] http://study.com/academy/lesson/what-are-information-systems-definition-types-quiz.html.
4. *Search SOA.* [En línea] [Citado el: 3 de Febrero de 2016.] http://searchsoa.techtarget.com/definition/Apache.
5. *PHPnet.* [En línea] [Citado el: 3 de Febrero de 2016.] http://php.net/manual/en/intro-whatis.php.
6. *JetBrains.* [En línea] [Citado el: 3 de Febrero de 2016.] https://www.jetbrains.com/phpstorm/features/.
7. *Developer Mozilla.* [En línea] [Citado el: 3 de Febrero de 2016.] https://developer.mozilla.org/es/docs/HTML/HTML5.
8. *DesarrolloWeb.* [En línea] [Citado el: 3 de Febrero de 2016.] http://www.desarrolloweb.com/articulos/introduccion-css3.html.
9. *AngularJS.* [En línea] [Citado el: 3 de Marzo de 2016.] https://docs.angularjs.org/guide/introduction.
10. *Cufming Software.* [En línea] [Citado el: 4 de Abril de 2016.] https://cufmingsoftware.wordpress.com/estandares-de-diseno/.
11. **Aalst, Wil M. P. van der. 2011.** *Extracting Event Data from Databases to Unleash Process Mining.* Eindhoven : s.n., 2011.
12. **—. 2011.** *Process Mining Discovery, Conformance and Enhancement of Busines Processes.* s.l. : Springer, 2011. 978-3-642-19345-3.
13. **Bose, R.P. Jagadeesh Chandra, Mans, Ronny S. y van der Aalst, Wil M. P. 2013.** *Wanna Improve Process Mining Results? It's High Time We Consider Data Quality Issues Seriously.* Eindhoven : s.n., 2013.
14. **Buijs, Joos C. A. M. 2010.** *Mapping Data Sources to XES in a Generic Way.* Eindhoven : s.n., 2010.
15. **Cormen, Thomas H., y otros. 2001.** *Introduction to algorithms.* Massachusetts : s.n., 2001. 0-262-03293-7.
16. **Dumas, Marlon, van der Aalst, Wil M. P. y ter Hofstede, Arthur H. M. 2005.** *Process-Aware Information Systems Bridging People and Software Through Process Technology.* Hoboken, New Jersey : John Wiley & Sons, 2005. 978-0-471-66306-5.

17. **Dunkl, Reinhol. 2013.** *Data Improvement to Enable Process Mining on Integrated Non-log Data Sources.* Viena : s.n., 2013.

18. **Eguiluz, Javier. 2013.** *Desarrollo web ágil con symfony 2.* 2013.

19. **Gamma, Erich, y otros. 2005.** *Design Patterns: Elements of Reusable Object-Oriented Software.* 2005. ISBN 0-201-63361-2.

20. **Group, Eindhoven Research. 2016.** *Micro Event Extension.* Eindhoven : s.n., 2016.

21. **Günther, Christian W. y Verbeek, Eric. 2014.** *XES Standard Definition 2.0.* Eindhoven : s.n., 2014.

22. **Larman, Craig. 1999.** *UML y Patrones, Introducción al análisis y diseño orientado a objetos.* México : Prentice Hall, 1999.

23. **Lu, Xixi. 2013.** *Artifact-Centric Log Extraction and Process Discovery.* Eindhoven : s.n., 2013.

24. **Mining, IEEE Task Force on Process. 2011.** *Manifiesto sobre Minería de Procesos.* 2011.

25. **Peguero Álvarez, Héctor David y Torres Peregrino, Carlos Alberto. 2013.** *Módulo para el registro y transformación de trazas de eventos a formato XES.* La Habana : s.n., 2013.

26. **Pérez Alfonso, Damián. 2012.** *Normas y estándares de codificación. Universidad de las Ciencias Informáticas.* La Habana : s.n., 2012.

27. **Pressman, Roger S. 2005.** *Ingeniería del Software. Un enfoque práctico.* La Habana : Félix Varela, 2005.

28. **Preval, Katia Saria. 2015.** Excriba. [En línea] Octubre de 2015. [Citado el: 12 de Marzo de 2016.] https://excriba.prod.uci.cu/page/site/centro-ceige/document-details?nodeRef=workspace://SpacesStore/3f42be1e-b62a-4461-af09-fa0ab75047db .

29. **Rodríguez, Carlos, y otros. 2012.** *Eventifier: Extracting Process Execution Logs from Operational Databases.* Viena : s.n., 2012.

30. **Torres, Abraham Calas. 2014.** Excriba. *Ficha técnica del proyecto Boson.* [En línea] 2014. [Citado el: 3 de Noviembre de 2015.] https://excriba.prod.uci.cu/proxy/alfresco//api/node/content/workspace/SpacesStore/e5032d4e-1472-46bf-b554-e37016ed0009/CEIGE_Boson_Ficha%20tecnica%20del%20proyecto.odt?a=true.

31. **van der Aalst, Wil M. P. 2014.** *Extracting Event Data from Databases to Unleash Process Mining.* Eindhoven : s.n., 2014.