UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 2

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

COMPONENTE INFORMÁTICO DE EXTRACCIÓN DE REGISTROS DE EVENTOS PARA EL ANÁLISIS DE PROCESOS QUIRÚRGICOS BASADO EN UN ENFOQUE POR PROCESOS

AUTORES

Brian Pizzorno Suárez

Víctor Alejandro Roque Domínguez

TUTORES

MSc. José Felipe Ramírez Pérez

Ing. Abraham Calás Torres

La Habana, 21 de junio de 2016

"Año 58 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaramos ser los únicos autores del trabajo de diploma “Componente informático de extracción de registros de eventos para el análisis de procesos quirúrgicos basado en un enfoque por procesos” y autorizo al Centro de Informática Médica de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año 2016.

Brian Pizzorno Suárez

_____

Firma del Autor

Víctor Alejandro Roque Domínguez

_____

Firma del Autor

MSc. José Felipe Ramírez Pérez

_____

Firma del Tutor

Ing. Abraham Calás Torres

_____

Firma del Tutor

DATOS DE CONTACTO

MSc. José Felipe Ramírez: graduado de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas en el año 2012. Analista y desarrollador de componentes informáticos para la salud en áreas como la atención materno-infantil, medicina familiar, telemedicina, nefrología y el nivel hospitalario, en las que ha realizado varias publicaciones. Pertenece a la Sociedad Cubana de Informática Médica (SOCIM), a la Unión de Informáticos de Cuba (UIC) y a la Sociedad Cubana de Reconocimiento de Patrones (SCRP). Es miembro del grupo de investigación de minería de procesos. Es Máster en Informática Aplicada y profesor con categoría principal de Instructor. Actualmente es Jefe del Departamento de Desarrollo de Aplicaciones del Centro de Informática Médica (CESIM).

Correo electrónico: jframirez@uci.cu

Ing. Abraham Calás Torres: graduado de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas en el año 2013. Ubicado en el Centro de Informatización de Entidades, donde se desempeña como desarrollador en el Departamento de Desarrollo de Componentes. Es miembro del Grupo de Investigación de minería de procesos.

Correo electrónico: acalas@uci.cu

AGRADECIMIENTOS

Brian Pizzorno Suárez

Le agradezco a mi mamá (la mejor madre del mundo) por estar siempre apoyándome, preocupándose y confiando en mí, por su amor, su comprensión y sus consejos, gracias mami.

A mi esposa (lo mejor que me ha pasado en la vida) por estar siempre conmigo en las buenas y en las malas apoyándome, dándome su amor, gracias por confiar en mi cosita linda te amo con todo mi corazón.

A mi hermanito, porque es una de las personas que más quiero en esta vida, gracias titi.

A mis abuelos (Pipo y Mima), por siempre estar pendientes de mí en estos cinco años, los quiero.

A Osvaldo gracias por ser como un hijo para ti, para mi eres mi padre.

A toda mi familia, Susana, Jaciel, Elier, Osvaldito, por siempre brindarme su apoyo.

A mis suegros, que son mis segundos padres, gracias por todo su apoyo y preocupación en estos cinco años.

A mi cuñi y mi segunda abuela (Gladys) gracias por sus consejos, su apoyo y por siempre estar pendientes de mí.

A mi compañero de tesis, gracias por su amistad y por su ayuda incondicional, gracias por ser un verdadero amigo.

A todos mis amigos, en especial a Gilberto, Osmar, y Asley, por todos los momentos buenos que pasamos en esta etapa tan linda.

A mis tutores por toda su ayuda y preocupación.

A la gente del aula, gracias por ser mis compañeros, fueron 5 años inolvidables.

A todos los del grupo de investigación de Minería de procesos por su apoyo y su ayuda.

Agradecimientos

A mis profesores de estos 5 años
A todos de corazón gracias.

Víctor Alejandro Roque Domínguez

Le agradezco a mi mamá que me ama incondicionalmente sin importar nada. Gracias por ser tú quien me dio la vida.

A mis abuelos Mime, Pipe y Nena gracias por ser mi pedazo de cielo aquí en la tierra.

A mis tíos Ete, Ofo y Lily, Magdalena y William gracias por darme alegría y siempre estar pendientes de mí.

A Silvio que es mi tercer papá y como tal me ha cuidado en todos estos años de lucha.

A mi papá que aunque él diga que no está orgulloso de mí, yo sé que sí lo está. Papi te quiero.

A mis primos Maylin, Héctor Luis, Wilito y Magdamigdalis gracias por estar ahí para lo que necesite.

A mi compañero de tesis gracias por ser camarada, amigo, familia.

A mi tutor José Felipe por su paciencia y dedicación a nosotros, gracias.

A mi co-tutor Abraham gracias por ayudarnos. Es un honor poder contar con usted.

A Eddry y Dariel, que son mis hermanos en la UCI, gracias por cuidarme y quererme como su familia.

A todos mis amigos, en especial Gilberto, Osmar, Asley, Bienvenido, Manchón y Lourdes gracias por hacer de esta etapa de mi vida un momento inolvidable.

A todos los miembros de la brigada y a los que estuvieron alguna vez, gracias por haber formado parte de mi vida mucho he aprendido de ustedes.

A todos mis profesores, en especial Jussieini, José Nolberto, Roxana y Madelin gracias por marcar con sus acciones mi vida. Es una dicha haberlos conocidos.

A todos los que de una forma u otra contribuyeron para lograr mi sueño, gracias.

DEDICATORIA

Brian Pizzorno Suárez

Dedico el presente trabajo de diploma a mi mamá, a mi esposa, a mi hermano y a mi familia por todo su amor y su apoyo.

Víctor Alejandro Roque Domínguez

A Papá que solo por unos meses no pudiste verme graduado pero sé que donde estés te sentirás feliz. A Mami que es mi punto de apoyo y mi palanca. A todas las personas que ven, con esta tesis, sus sueños realizados.

RESUMEN

Los registros de eventos almacenan información relevante de los procesos en las instituciones. Permiten el análisis de dichos procesos para controlarlos, mejorarlos y descubrir sus flujos a partir de las trazas. Evidentemente, se guardan bajo una semántica bien definida y una calidad y seguridad determinada.

El Sistema de Información Hospitalaria XAVIA HIS del Centro de Informática Médica de la Universidad de las Ciencias Informáticas contiene un conjunto de módulos entre los que se encuentra Bloque Quirúrgico. Este sistema tiene implementado una bitácora como vía para recopilar información de la actividad de los usuarios y los procesos del sistema la que se almacena en forma de datos. Esta situación imposibilita el análisis de los procesos y, con ello, la posibilidad de ofrecer información de interés para apoyar la toma de decisiones.

La investigación presenta una solución para incluir al módulo Bloque Quirúrgico la construcción de trazas de procesos a partir de evidencias almacenadas en su base de datos y, conformar así, el registro de eventos. Esta es la base para, posteriormente, posibilitar el uso de técnicas de minería de proceso. En el trabajo se aplica la estrategia de extracción de trazas de procesos en sistema de información consciente de datos. La investigación culmina con la integración del componente al Sistema de Información Hospitalaria.

Palabras claves: estrategia de extracción de trazas, minería de procesos, registros de eventos, Sistema de Información Hospitalaria, trazas de proceso

ÍNDICE

ÍNDICE DE FIGURAS Y TABLAS

INTRODUCCIÓN

En la actualidad el entorno organizacional es competitivo y requiere de empresas que se adapten rápidamente a los mercados cambiantes, por tanto, se hace imperativo el control y mejoramiento de todos los procesos que se realizan dentro de ellas. La información se ha colocado en un lugar privilegiado como uno de los principales recursos que poseen las empresas. Los entes que se encargan de la toma de decisiones han comenzado a comprender que la información no es sólo un subproducto de la conducción empresarial, sino que a la vez alimenta a las organizaciones y puede ser uno de los tantos factores críticos para la determinación del éxito o fracaso de éstas.

Los sistemas de información juegan un papel importante dentro de las organizaciones, pues permiten el control y la ejecución de las tareas y actividades que se realizan dentro de cada área de la organización. Además estos sistemas tienen como objetivo recolectar, procesar, almacenar y distribuir datos en apoyo al control y la toma de decisiones. Además deberán contemplar el diseño de un sistema integrado que relacione las informaciones generales por las diversas aplicaciones funcionales de la empresa y que permita así mejorar los procesos de toma de decisiones (Muñoz, 2007).

Los sistemas de información se dividen en dos clasificaciones: los Sistemas de Información Conscientes de Datos (DAIS por sus siglas en inglés) y los Sistemas de Información Conscientes de Procesos (PAIS por sus siglas en inglés). Los DAIS no tienen nociones de procesos en sus trazas lo que conlleva a que las mismas contengan información insuficiente y de poca calidad (IEEE Task Force on Process Mining, 2011). Por otra parte los PAIS presentan varios beneficios entre los que se encuentran; una mayor eficiencia, mejor control del proceso, mejor servicio al cliente y la mejora de procesos de negocio, además de generar trazas de gran calidad y completitud (IEEE Task Force on Process Mining, 2011).

Dichos sistemas se encuentran vinculados a la mayoría de los sectores de la sociedad, incluyendo al sector de la salud, el cual tiene una alta vigencia en la calidad de vida de las personas. El sector de la salud en Cuba se evidencia mediante el Ministerio de Salud Pública (MINSAP), como organismo rector del Sistema Nacional de Salud (SNS), el cual es el

encargado de garantizar el acceso a servicios integrales de salud que se prestan en unidades ambulatorias y hospitalarias.

El SNS se divide en tres niveles de asistenciales: la Atención Primaria de Salud (APS), donde se solucionan aproximadamente el 80% de los problemas de salud y se ofrecen las acciones de promoción y prevención; este nivel se encamina sobre todo a la atención integral al paciente, a la familia y la comunidad y está basado en el programa del médico y enfermera de la familia (Domínguez et al, 2011). La Atención Secundaria que cubre cerca del 15%, en este nivel se ubica los hospitales y establecimientos donde se prestan servicios relacionados a la atención en medicina interna, pediatría, cirugía general y psiquiatría. Por último se encuentra la Atención Terciaria que solo abarca un 5%, este nivel se reserva para la atención de problemas poco prevalentes, se refiere a la atención de patologías complejas que requieren procedimientos especializados y de alta tecnología (Vacarezza et al, 2011).

Con el objetivo de informatizar los procesos del nivel secundario de salud en Cuba, en el Centro de Informática Médica (CESIM) de la Universidad de las Ciencias Informáticas (UCI), se desarrolló el Sistema de Información Hospitalaria (XAVIA HIS), el cual tiene implementado una bitácora, como vía para almacenar información de la actividad de los usuarios en el sistema, no obstante no es utilizada eficientemente para el análisis de los mismos.

Dicho sistema contiene un conjunto de módulos entre los que se encuentra Bloque Quirúrgico. Dentro de este se gestionan procesos de negocio como la Gestión de recursos entre los que se encuentran los quirófanos, los horarios de los cirujanos, los horarios de los anestesiólogos y los *kits* de cirugía, además se gestiona el plan quirúrgico, la intervención quirúrgica y las solicitudes.

La información que se maneja en el Bloque Quirúrgico se encuentra orientada a datos de las acciones de los usuarios en el sistema y no a los datos relacionados con la ejecución de los procesos. Esto trae como consecuencia que se dificulte la toma de decisiones pues no se tiene en cuenta el conocimiento obtenido de la ejecución real de los procesos de esta área para la planificación y gestión de recursos.

Esto supone un conjunto de deficiencias como:

- No se tiene en cuenta el consumo real de insumos para la configuración de los *kits* quirúrgicos por procedimiento. Esto puede ocasionar errores en la cantidad de productos o medicamentos de un determinado tipo, lo que puede ocasionar un gasto de dinero innecesario en productos que no se ajustan al consumo real de *kits* o a las necesidades de productos en el hospital.
- No se analizan los tiempos de duración real de los procedimientos quirúrgicos cuando se realiza la planificación. Puede darse el caso de que el tiempo que se destina para los procedimientos quirúrgicos pueden no estar acorde a lo que en realidad dura dicho procedimiento quirúrgico y eso trae conflictos con los horarios.
- No se realiza un seguimiento del comportamiento de los equipos de trabajo quirúrgicos respecto a la planificación inicial y planificación final del personal a operar. Puede ser que un equipo de trabajo que está realizando el seguimiento de un determinado caso no sea el que realice el procedimiento quirúrgico, esto puede afectar el resultado de dicho procedimiento.
- No se tiene en cuenta la cantidad de cirugías realizadas por servicios y especialistas. Imposibilita conocer cómo se comporta la relación entre cantidad de cirugías planificadas y la cantidad de cirugías que se realizaron en realidad, para identificar causas o motivos
- No se tiene en cuenta la procedencia de la solicitud de intervención quirúrgica. Imposibilita realizar un seguimiento del funcionamiento anómalo de los procesos y detectar cualquier desvío de recursos que pueda ocurrir.
- No se realizan análisis históricos o evaluaciones de tendencias de aplicación y cantidad de medicamentos y material de anestesia ante determinada dolencia, diagnóstico realizado o procedimiento quirúrgico practicado.

Por lo anteriormente planteado se identifica como **problema a resolver:** la forma en que se almacenan los datos en la bitácora del Sistema de Información Hospitalaria XAVIA HIS, imposibilita que el personal del servicio quirúrgico realice un análisis de los procesos quirúrgicos.

El problema está enmarcado en el **objeto de estudio**: enfoques basados en datos y procesos en los sistemas de información hospitalaria, centrado en el **campo de acción**: el enfoque basado en procesos en los procesos quirúrgicos.

Para solucionar el problema planteado, se define como **objetivo general**: desarrollar un componente informático basado en un enfoque por procesos, que permita al personal del servicio quirúrgico extraer registros de eventos para realizar análisis de los procesos desde el módulo Bloque Quirúrgico del Sistema de Información Hospitalaria.

Se proponen las siguientes tareas de la investigación:

1. Análisis de los sistemas informáticos existentes a nivel nacional e internacional, que generan registros de eventos a partir de una gestión basada en procesos, estableciendo similitudes con la investigación en curso.
2. Análisis de los procesos de negocio del módulo Bloque Quirúrgico del Sistema de Información Hospitalaria XAVIA HIS, para identificar los elementos relevantes objeto de análisis por el personal del servicio quirúrgico y administrativo.
3. Asimilación de las herramientas y tecnologías necesarias del Sistema de Información Hospitalaria, para el desarrollo del componente informático.
4. Implementación de un componente informático basado en un enfoque por procesos, desde el módulo Bloque Quirúrgico del Sistema de Información Hospitalaria XAVIA HIS, a partir de la creación de registros de eventos, que permita al personal del servicio quirúrgico realizar análisis de los procesos quirúrgicos.
5. Realización de las pruebas de software para validar el correcto funcionamiento del componente informático desarrollado.

Con el desarrollo de la presente investigación se pretende obtener un componente que permita la extracción de registros de eventos, de los procesos de negocio definidos en el módulo Bloque Quirúrgico. El componente permitirá disminuir el tiempo requerido en el análisis de los procesos de negocio del módulo Bloque Quirúrgico del Sistema de Información Hospitalaria XAVIA HIS y proveer un mayor apoyo a la toma de decisiones clínico-administrativas.

Los **métodos científicos** utilizados para desarrollar la investigación (Hernández, 2011) fueron:

Métodos teóricos:

- **Análisis histórico-lógico:** se utilizó para analizar el surgimiento, trayectoria y evolución de la minería de procesos para una mejor comprensión del objeto y campo de acción.
- **Analítico-sintético:** se puso de manifiesto realizando un estudio de las principales técnicas y herramientas existentes que son utilizadas en la aplicación de minería de procesos.
- **Inductivo-deductivo:** se utilizó en la aplicación de pruebas al sistema, llegando a conclusiones a partir de las respuestas proporcionadas por este.

Métodos empíricos:

- **Observación:** como instrumento para adquirir conocimiento sobre el campo de acción a través de la investigación directa de las herramientas.
- **Entrevista:** se realizaron entrevistas no estructuradas a los analistas de Sistema de Información Hospitalaria, con el objetivo de adquirir información para el desarrollo del componente.

El documento se encuentra estructurado en 3 capítulos, siendo estos:

El ***Capítulo 1. Fundamentación teórica de la investigación,*** explica los conceptos fundamentales sobre los diferentes elementos en los que se basan los autores de la presente investigación, que constituyen la base para el desarrollo del componente informático, para el análisis de procesos quirúrgicos basado en un enfoque por procesos, a partir de la creación de registros de eventos. Se describen además las tecnologías y herramientas propuestas para el desarrollo del componente propuesto.

El ***Capítulo 2. Propuesta de solución,*** describe la propuesta de solución, junto a la modelación del flujo de información del componente propuesto, se describe el procedimiento seguido para el desarrollo del componente así como los patrones usados en la solución.

Por último, en el ***Capítulo 3. Validación de la solución propuesta,*** se valida la aplicación obtenida y se realizan pruebas de software al componente desarrollado.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA DE LA INVESTIGACIÓN

Como su nombre lo indica, en este capítulo se tratan los conceptos fundamentales que apoyan el área del conocimiento sobre el cual versa. Estos son la base para el estudio del estado del arte sobre los sistemas informáticos que trabajan la extracción de registro de eventos. Aunque las herramientas a utilizar en el trabajo son adoptadas de la línea de desarrollo del sistema, se presenta un análisis de cada una de ellas así como de los lenguajes y tecnologías utilizadas.

1.1 Conceptos fundamentales

1.1.1 Procesos

Según la serie de normas internacionales ISO 9000 (2005) se define un proceso como "conjunto de actividades mutuamente relacionadas o que interactúan, las cuales transforman elementos de entrada en resultados".

El Sistema de Gestión de Calidad define proceso como un conjunto de tareas que conforman una serie de actividades, interrelacionadas entre sí, que transforman una entrada en una salida con valor añadido para el usuario.

De acuerdo al diccionario de la Real Academia Española, el concepto hace referencia a la acción de ir hacia adelante, al transcurso del tiempo, al conjunto de las fases sucesivas de un fenómeno natural o de una operación artificial.

Los autores de la presente investigación se basan para la realización de la misma, en la definición de proceso dada por la serie de normas internacionales ISO 9000 (2005).

1.1.2 Instancia de proceso

Una instancia de un proceso, en el ámbito de la Minería de procesos, hace referencia a la entidad, siendo ejecutada por el proceso que es analizado. Los eventos se refieren a instancias del proceso. A su vez, un evento expresa el estado de una actividad para una

instancia particular de un proceso, por ejemplo, el inicio, conclusión o cancelación de una actividad (IEEE Task Force on Process Mining, 2011).

1.1.3 Gestión basada en procesos

La serie de normas internacionales ISO 9001 (2008) promueve la adopción de un enfoque basado en procesos cuando se desarrolla, implementa y mejora la eficacia de un sistema de gestión de la calidad, para aumentar la satisfacción del cliente mediante el cumplimiento de sus requisitos. La gestión basada en procesos son un conjunto de actuaciones, decisiones, actividades y tareas que se encadenan de forma secuencial y ordenada para conseguir un resultado que satisfaga plenamente los requerimientos del cliente al que va dirigido (Blaya, 2006).

La gestión por procesos es una disciplina de gestión que ayuda a la dirección de la empresa a identificar, representar, diseñar, formalizar, controlar, mejorar y hacer más productivos los procesos de la organización para lograr la confianza del cliente (Carrasco, 2011).

En un análisis de (Mallar, 2010) se concluye que existen diversos motivos que mueven esta gestión dentro de una organización, entre los cuales se encuentran: extensión del programa institucional de calidad, cumplimiento de legislaciones vigentes, mejoramiento continuo, entender qué se está haciendo bien o mal a través de la comprensión de los procesos, así como la automatización y organización de los mismos.

1.1.4 Minería de procesos.

La minería de procesos enfoca sus esfuerzos en la extracción de información útil y relevante sobre los procesos de la organización, tomando como punto de partida los datos contenidos en los registros de eventos (Pérez et al, 2013). Existen tres tipos de técnicas de minería de procesos: el descubrimiento de proceso, la verificación de conformidad y el mejoramiento de modelos, como se muestra en la **Fig. 1**.

Figura 1. Representación de los tipos de técnicas de minería de procesos.
Fuente: van der Aalst, 2011.

El descubrimiento de procesos abarca el conjunto de técnicas para la obtención de un modelo de procesos partiendo de un registro de eventos. Para muchas organizaciones resulta sorprendente la capacidad de los algoritmos de descubrimiento para identificar los procesos reales partiendo únicamente de los registros de eventos asociados a su ejecución (Van der Aalst, 2011).

En el chequeo o verificación de conformidad se compara un modelo de proceso existente con un registro de eventos del mismo proceso. La verificación de conformidad puede ser usada para chequear si la realidad, tal como está almacenada en el registro de eventos, es equivalente al modelo y viceversa (Mining, 2011).

La mejora de modelos de procesos es extender o mejorar un modelo de proceso existente usando la información acerca del proceso real almacenada en algún registro de eventos.

1.1.5 Registro de eventos

Un registro de eventos es básicamente una tabla (Gehrke et al, 2013), que registra los eventos que se producen en un determinado proceso para un determinado caso (Buijs, 2010).

Esta colección de eventos es utilizada como entrada para la minería de procesos (Van Der Aalst et al. 2011).

El punto de partida de la minería de procesos es un registro de eventos. Todas las técnicas de minería de procesos asumen que es posible registrar eventos secuencialmente tal que cada evento se refiera a una actividad (i.e., un paso bien definido en algún proceso) y se relacione a un caso particular (i.e., una instancia de proceso). Los registros de eventos podrán almacenar información adicional acerca de los eventos. De hecho, siempre que sea posible, las técnicas de minería de procesos usan información extra, tales como el recurso (i.e., persona o dispositivo) que ejecuta o inicia la actividad, la marca de tiempo del evento, o elementos de datos registrados con el evento (e.g., el tamaño de un pedido) (Van Der Aalst et al, 2011).

Un registro de eventos puede ser visto como la secuencia de eventos E =<e1, e2, e3,…, em> donde:

ei=< id, tname, pname, piid, ts, pl>

ei es un evento de una instancia de un proceso, id es el identificador del evento, tname es el nombre de la tarea a que el evento está asociado, con pname siendo el nombre del tipo de proceso, con piid el identificador de la instancia del proceso, ts siendo la fecha del evento y pl siendo la “carga útil”.

Reconstruir un registro E de eventos ei 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. Específicamente para el atributo id, asignar un identificador a un evento significa reconocer la existencia de dicho evento. El valor de pname se puede obtener solamente del experto en el dominio que es quien sabe cuál es el proceso que se trata de identificar. El piid se utiliza para agrupar los eventos en instancias de procesos. El atributo ts se utiliza para ordenar cronológicamente los eventos siendo este un requerimiento esencial para el descubrimiento de procesos (Rodríguez, 2012).

Se le llama identificación de un evento a la asignación de valores a id, pname y tname, ordenamiento de eventos a la asignación de valores a ts, asociación de datos a la asignación de valores a pl y correlación a la asignación de valores a piid. Estas cuatro actividades juntas constituyen el proceso de configuración del registro de trazas de proceso (Rodríguez, 2012).

1.1.6 Formato de registro de eventos eXTensible Event Stream

Los registros de eventos contienen información sobre cómo los procesos se han desarrollado en los sistemas en ejecución. A medida que más y más sistemas capturan dicha información, hay una necesidad de ser capaz de transferir esta información de los sistemas en ejecución a un sitio donde la información puede ser analizada, ya sea automáticamente por el software de la campo de la inteligencia computacional, o manualmente (al menos en parte) usando dicho software (IEEE Computational Intelligence Society, 2016).

El formato XES es un estándar XML para los registros de eventos. El estándar ha sido adoptado por la IEEE Task Force on Process Mining como el formato de intercambio de registros de eventos por defecto.

Extensiones de XES

La extensión de XES más importante es la **concept** la cual especifica un nombre para el registro de eventos, la traza y los eventos. Proporcionando nombres a cada elemento es fácil y muy informativo y por eso siempre deben ser provistos. Los nombres de las trazas deberán incluir algún identificador único. Los nombres de los eventos deben proveer el nombre de la actividad ejecutada representada por el evento. La extensión **concept** define un atributo **instance** para eventos. Este atributo representa un identificador de la instancia de la actividad la cual generó el evento. Incluyendo este atributo en el registro de eventos permite enlazar los eventos con los registros en la fuente de datos para futuros análisis. También puede ayudar en el manejo de la convergencia como se explicó anteriormente (Peguero et al, 2015).

Otra extensión importante es la “time”. Esta extensión especifica el atributo que representa el instante de ocurrencia del evento. Mediante la grabación del instante de tiempo de ocurrencia

del evento, los eventos pueden ser ordenados. Además, esto permite realizar análisis de duración y desempeño (Peguero et al, 2015).

Al ejecutarse un evento es necesario que su registro recopile la mayor cantidad de información, con intervalos muy pequeños, al nivel de los segundos, incluso de los milisegundos si es posible. La decisión para determinar la frecuencia de captura de los datos del evento depende de cuántos de ellos y a qué velocidad ocurren. Por ejemplo: en una base de datos operacional, el número de eventos que pueden estarse ejecutando a la misma vez puede ser muy alto Es en ocasiones como estas donde se deciden intervalos de tiempo más pequeños para no perder información valiosa que puede ser crucial en el análisis de los procesos. Aunque la mayoría de los algoritmos pueden manejar instantes de tiempos iguales, si se brinda más precisión mejoran drásticamente.

En la mayoría de los casos con usar el tipo “iniciado” y “completado” es suficiente pero existen otros tipos de eventos. 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, la distribución del trabajo entre las diferentes unidades puede ser visualizada. Además, las redes sociales pueden ser construidas para indicar qué actores trabajaron juntos en algunos casos entregando el trabajo, etc. También se pueden detectar a un grupo de actores desempeñando tareas similares (Peguero et al, 2015).

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”. Esta extensión define tres diferentes atributos para eventos. El atributo más comúnmente usado es el “resource” el cual recoge el nombre o identificador del actor que ejecutó el evento. Adicionalmente el rol del recurso se puede almacenar también con el atributo “role”. En añadidura también se puede almacenar el grupo al que el usuario pertenece en el atributo “group”. La decisión de cuáles de los atributos usar depende de la información disponible y del nivel de detalle deseado. En alguno de los casos son necesarios los análisis sobre el rol o el grupo (Peguero et al, 2015).

La última de las cinco extensiones estándares de XES analizar es la “semantic”. Esta extensión agrega el atributo “modelReference” a todos los elementos en el registro de eventos. Dicho atributo se refiere a un modelo conceptual en una ontología (externa) la cual permite un entendimiento más a fondo de un evento. Otro ejemplo pudiese ser referirse a una ontología de usuarios en el atributo “resource” de la extensión “organizational” (Günther, 2009).

Pueden existir otras extensiones y atributos que definan los atributos relacionados a un proceso. Las cinco extensiones mencionadas anteriormente por otra parte, capturan los atributos usados frecuentemente. Al menos una parte de los atributos definidos por estas extensiones como “concept: name” y “time: timestamp", deben ser especificados en el registro de eventos para que la minería de procesos se pueda aplicar exitosamente. A continuación se muestra un ejemplo de un registro de eventos en formato XES (Peguero et al, 2015).

```
− <log xes.version="1.0" xes.features="nested-attributes" openxes.version="1.0RC7">
    <extension name="Lifecycle" prefix="lifecycle" uri="http://www.xes-standard.org/lifecycle.xesext"/>
    <extension name="Organizational" prefix="org" uri="http://www.xes-standard.org/org.xesext"/>
    <extension name="Time" prefix="time" uri="http://www.xes-standard.org/time.xesext"/>
    <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org/concept.xesext"/>
    <extension name="Semantic" prefix="semantic" uri="http://www.xes-standard.org/semantic.xesext"/>
  − <global scope="trace">
      <string key="concept:name" value="__INVALID__"/>
    </global>
  − <global scope="event">
      <string key="concept:name" value="__INVALID__"/>
      <string key="lifecycle:transition" value="complete"/>
    </global>
    <classifier name="MXML Legacy Classifier" keys="concept:name lifecycle:transition"/>
    <classifier name="Event Name" keys="concept:name"/>
    <classifier name="Resource" keys="org:resource"/>
    <string key="source" value="ProcessLogGenerator"/>
    <string key="concept:name" value="Process 3"/>
    <string key="lifecycle:model" value="standard"/>
  − <trace>
      <string key="concept:name" value="67"/>
      <string key="description" value="67"/>
    − <event>
        <string key="org:resource" value="Alain"/>
        <date key="time:timestamp" value="2010-04-20T15:31:00.000-04:00"/>
        <string key="concept:name" value="Tarea: Creación"/>
        <string key="description" value="Tarea: Creación"/>
        <string key="lifecycle:transition" value="complete"/>
      </event>
    − <event>
        <string key="org:resource" value="Alain"/>
        <date key="time:timestamp" value="2010-04-20T15:31:00.000-04:00"/>
        <string key="concept:name" value="Tarea: Creación"/>
        <string key="description" value="Tarea: Creación"/>
        <string key="lifecycle:transition" value="complete"/>
      </event>
    </trace>
  + <trace></trace>
  </log>
```

Figura 2. Registro de eventos en formato XES.

Fuente: elaboración propia.

1.1.7 Sistemas de información

Un sistema de información es un conjunto de personas, datos, procesos y tecnologías de la información que interactúan para recoger, procesar, almacenar y proveer información necesaria para el correcto funcionamiento de una organización (Fernandez, 2006).

Sistemas de Información Conscientes de Datos (DAIS)

Son sistemas que no tienen nociones de procesos en sus trazas lo que conlleva a que estos contengan información insuficiente y de poca calidad. Esto hace compleja o imposible la obtención de registros de eventos completos (IEEE Task Force on Process Mining, 2011).

Los elementos teóricos de la extracción de trazas de procesos desde DAIS son los siguientes:

Los sistemas de información almacenan todos los datos producidos en su base de datos operacional (OD por sus siglas en inglés). Las OD almacenan mayor cantidad de datos los cuales son más completos que los datos que se recogen en un registro de eventos, pero se distorsionan diferentes aspectos de los datos y se descuida la naturaleza basada en eventos de la ejecución de los procesos. Por esta razón el descubrimiento de procesos generalmente comienza a partir de un registro de eventos (Rodríguez, 2012).

Sistemas de Información Conscientes de Procesos (PAIS)

Las organizaciones están en constante búsqueda de mejoras en la eficiencia de sus procesos de negocio. Para ayudar a lograr estos objetivos, muchas organizaciones están recurriendo a los Sistemas de Información Conscientes de Procesos (PAIS) para configurar y controlar sus procesos (Dumas et al, 2005). Los principales beneficios que las organizaciones buscan mediante la implementación de soluciones de PAIS incluyen: una mayor eficiencia, mejor control del proceso, mejor servicio al cliente y la mejora de procesos de negocio. Además los sistemas PAIS son sistemas que generan trazas de gran calidad y completitud (IEEE Task Force on Process Mining, 2011).

1.2 Soluciones existentes a nivel nacional e internacional

1.2.1 Nivel internacional

A nivel internacional existen un conjunto de herramientas que permiten extraer registros de eventos, las cuales se describen a continuación:

XES Mapper (XESame): XESame proporciona una forma genérica para la extracción de un registro de eventos de alguna fuente de datos, y está diseñada para ser fácil de utilizar. Una fortaleza clave de XESame es que no se requieren conocimientos de programación. Toda la conversión de la fuente de datos de registro de eventos puede ser definida a través de la interfaz gráfica de usuario (Verbeek et al, 2011).

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, Mysql, etc. 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, debería ser capaz de analizar este registro de todas las maneras posibles en el ámbito de minería de procesos (Torres et al, 2013).

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 (Rodríguez et al 2015). Primeramente, el experto en el dominio identifica los eventos en la base de datos operacional (OD por sus siglas en inglés), los ordena y le asocia datos. Todas estas actividades están soportadas por el Extractor de eventos el cual ayuda al experto del dominio de una manera 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 procesos. El resultado del proceso completo es un registro de eventos listo para aplicársele minería de procesos (Rodríguez, 2012).

1.2.2 Nivel nacional

A nivel nacional, la situación es otra. Entre las aplicaciones informáticas que soportan registro de eventos están:

Componente para la extracción de registros de eventos en formato XES del sistema ZUN Suite: el componente fue desarrollado en el año 2015, se enfoca en el análisis de procesos de negocio, aplicando minería de procesos a partir de registros de eventos extraídos del sistema ZUN Suite (Gestión Hotelera). Como resultado se obtiene una descripción de los procesos de gestión hotelera cuyas ejecuciones son registradas en el sistema ZUN Suite. Además, un componente para el marco de trabajo de la minería de proceso ProM que permita la extracción de los registros de eventos, dicho componente disminuye el tiempo en el análisis de los procesos en las instalaciones hoteleras, lo cual puede redundar en una mejor atención a los clientes (Pérez et al, 2015). El Sistema ZUN Suite se encuentra estructurado por módulos y cada módulo tiene asociado una base de datos (GET). Para la obtención de los datos en las bases de datos los autores y realizadores del componente en cuestión desarrollaron un algoritmo para la obtención de los datos y la posterior extracción del registro de eventos.

Generador de registros de eventos para el análisis de procesos en un Sistema de Información Hospitalaria: el generador es un componente para generar registros de eventos del Sistema de Información Hospitalaria desarrollado por la Universidad de las Ciencias Informáticas en el año 2015, para el análisis de procesos hospitalarios, a partir de técnicas de inteligencia artificial. Las herramientas utilizadas para la implementación son de código abierto y cumplen con los paradigmas de Software Libre, entre estas, Eclipse como entorno de desarrollo, Java como lenguaje de programación y Visual Paradigm para el modelado de procesos.

El componente presenta limitantes que imposibilitan su utilización con el módulo Bloque Quirúrgico del Sistema de Información Hospitalaria XAVIA HIS, a pesar de estar desarrollado para el mismo sistema se encuentra desarrollado para el módulo Almacén, único módulo de

dicho sistema que se encuentra orientado a procesos, por lo que resulta imposible su utilización.

Componente de extracción de registros de eventos del Sistema de Gestión de Ensayos Clínicos (SIGEC): es un componente desarrollado en el año 2015, permite la extracción de registros de eventos en formato XES para la aplicación de técnicas de minería de procesos, que contribuye a potenciar el análisis de los procesos de ensayos clínicos del Centro de Inmunología Molecular. El mismo es una aplicación web desarrollada en java, que utilizó como entorno integrado de desarrollo JBoss Developer Studio y como gestor de base de datos PostgreSQL. Además el componente tiene como objetivo contribuir al análisis de los procesos de ensayos clínicos en a partir de la información almacenada en las trazas de procesos (Valdés et al, 2015).

El componente presenta varias desventajas, el mismo se encuentra muy ajustado al negocio debido a que las consultas que realiza están asociadas a la base de datos por lo que no se puede usar para otro proceso que no sea del SIGEC.

1.2.3 Resultado del análisis de las soluciones existentes

A continuación se presenta una tabla resumen que recoge algunos parámetros a tener en cuenta para realizar una comparación entre las herramientas útiles para la extracción de registro de eventos.

Tabla 1. Herramientas de extracción de registro de eventos.

Herramienta	Ajustado al dominio de aplicación	Extrae registro de eventos en sistemas DAIS	Requiere conocimientos de programación o base de datos.
XESame	No	Sí	Sí
Eventifier	No	Sí	Sí
Componente del SIGEC	Sí	Sí	No

Componente del sistema ZUN Suite	Sí	No	No
Generador de registros de eventos	Sí	No	No

Luego de haberse realizado el análisis comparativo entre las herramientas se arribó a las siguientes conclusiones:

XESame no se ajusta al negocio en cuestión ya que a pesar de no requerir conocimientos de programación, si requiere conocimientos de base de datos, por lo que imposibilita al personal quirúrgico su utilización.

Eventifier al igual que XESame requiere conocimientos de base de datos, además de requerir conocimientos de programación por lo que el personal quirúrgico ausente de dicho conocimiento le resulta imposible su utilización.

El componente para la extracción de registros de eventos en formato XES del sistema ZUN Suite presenta limitantes que imposibilitan su utilización con el Sistema de Información Hospitalaria XAVIA HIS, pues se encuentra orientado a procesos, además de encontrarse muy ajustado al negocio debido a las características que presentan las bases de datos con que cuenta el ZUN Suite, forzando incluso a los desarrolladores del componente a la creación de algoritmos específicos para la obtención de los datos necesarios para la extracción de registros de eventos.

El generador de registros de eventos para el análisis de procesos en un Sistema de Información Hospitalaria presenta limitantes que imposibilitan su utilización con el módulo Bloque Quirúrgico del Sistema de Información Hospitalaria XAVIA HIS, a pesar de estar desarrollado para el mismo sistema se encuentra desarrollado para el módulo Almacén, único módulo de dicho sistema que se encuentra orientado a procesos, por lo que resulta imposible su utilización.

El componente del CIGEC presenta varias desventajas, el mismo se encuentra muy ajustado al negocio debido a que las consultas que realiza están asociadas a la base de datos por lo que no se puede usar para otro proceso que no sea del SIGEC.

1.2.4 Características útiles encontradas después del análisis de las herramientas existentes

A pesar de las limitantes planteadas anteriormente para cada componente, se tomaron de estas herramientas algunos elementos o características, que brindaron un aporte significativo para el desarrollo del componente de extracción de registro de eventos en cuestión, estos son:

- Forma en que XESame representa la configuración de los procesos.
- Estrategia que utiliza Eventifier para la extracción de la información de los eventos.

1.3 Tecnologías y herramientas.

A continuación se describen las principales herramientas y tecnologías a utilizar en el proceso de desarrollo del componente propuesto.

1.3.1 Marcos de trabajo

Los marcos de trabajo (frameworks) se pueden considerar como soluciones completas que contemplan herramientas de apoyo a la construcción (ambiente de trabajo o desarrollo) y motores de ejecución (ambiente de ejecución). Estos aceleran el proceso de desarrollo, permiten reutilizar código existente y promover buenas prácticas de desarrollo, como el uso de patrones. (Zaninotto, 2008)

Jboss SEAM es un framework desarrollado por Jboss que tiene como objetivo facilitar el desarrollo de aplicaciones sobre todo aplicaciones web. Es una plataforma de desarrollo de código abierto. SEAM integra tecnologías como *JavaScript* asíncrono y XML (AJAX), *JSF*, *JPA*, *EJB* y *Business Process Management* (BPM) (Liu, 2011). Su función principal radica en eliminar la capa artificial que existe entre EJB 3.0 y JSF y provee un sistema de anotaciones

para integrar estos dos *frameworks*. Comparada con aplicaciones desarrolladas en otros *frameworks*, las aplicaciones SEAM son conceptualmente simples y requieren significativamente menos código (en Java y en XML) para obtener las mismas funcionalidades.

1.3.2 Lenguajes

Lenguaje Unificado de Modelado: UML 2.1

El Lenguaje Unificado de Modelado (UML, por sus siglas en inglés) es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema de software orientado a objetos. UML sirve para el modelado completo de sistemas complejos, tanto en el diseño de los sistemas de software como para la arquitectura hardware donde se ejecuten. Permite la representación conceptual y física de un sistema.

Cuenta con varios tipos de diagramas, los cuales muestran diferentes aspectos de lo que se quiere representar. Ofrece un estándar para describir un plano del sistema, incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables (Hernandez, 2012).

Lenguaje de programación: Java 1.7

Java es un lenguaje orientado a objetos de propósito general (Belmonte, 2005). El lenguaje es robusto y multiplataforma. Tiene muchas similitudes con el lenguaje C y C++. La principal característica de Java es la de ser un lenguaje compilado e interpretado. Todo programa en Java ha de compilarse y el código que se genera es interpretado por una máquina virtual.

1.3.3 Tecnologías

Para el desarrollo del componente, a continuación se propone un conjunto de tecnologías y herramientas de código abierto, las cuales cumplen las políticas de independencia tecnológica definidas en Cuba para la informatización de la sociedad.

Hibernate 3.3

Hibernate es un marco de trabajo de persistencia para Java de libre distribución que facilita el mapeo de atributos entre una base de datos relacional y el modelo de objetos de una aplicación. Además proporciona un potente lenguaje de consultas denominado Hibernate Query Language (HQL) (Scribd, 2012).

Java Platform Enterprise Edition (JavaEE) 5.0

JavaEE es una plataforma de programación para desarrollar y ejecutar *software* de aplicaciones en lenguaje de programación Java con arquitectura de N niveles distribuida (Franky, 2010). Se centra en hacer el desarrollo más fácil y se basa ampliamente en componentes de *software* modulares, siendo ejecutada sobre un servidor de aplicaciones.

Enterprise JavaBeans 3.0

Es un componente utilizado en Java que permite agrupar funcionalidades para formar parte de una aplicación, esto puede ser: un "*Java Bean*" agrupando información personal, datos sobre un pedimento, requerimientos de órdenes, entre otros. Permite realizar la administración automática de transacciones, seguridad, escalabilidad, concurrencia, distribución, acceso a ambientes portables y persistencia de datos. Incorpora el estándar JPA como el principal API de persistencia para aplicaciones EJB3. Su objetivo es simplificar el desarrollo de aplicaciones Java y estandarizar el API de persistencia para la plataforma Java. Forma parte de la especificación JEE 5 (Rondón, 2009).

Jboss Server 4.2

Jboss es un servidor de aplicaciones Java EE de código abierto implementado en Java puro y el más utilizado actualmente en el mercado. Puede ser utilizado en cualquier sistema operativo donde se encuentre la JVM. El proyecto fue adquirido por la empresa Red Hat y se nutre de una red mundial de colaboradores. Jboss implementa todo el paquete de servicios de J2EE y provee servicios extendidos de almacenamiento de datos en memoria y de manera persistente (REDHAT, 2010). Permite la integración de todas las tecnologías y herramientas utilizadas por Jboss SEAM.

1.3.4 Herramientas

Herramienta CASE Visual Paradigm 6.4

Visual Paradigm es una herramienta CASE (Ingeniería de Software Asistida por Computación). La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación (Pressman, 2002).

Visual Paradigm 6.4 es una herramienta de UML para el desarrollo de software de aplicación, diseñada para la ayuda del desarrollo del software. Es utilizada por distintos usuarios entre los que se incluyen ingenieros de software, analistas de sistemas, analistas de negocios, arquitectos y desarrolladores. Soporta el ciclo de vida completo del desarrollo de software. (Santiesteban et al., 2011) Esta herramienta permitió la modelación del componente utilizando como lenguaje UML a través de la creación de diagramas en un ambiente visual.

Entorno Integrado de Desarrollo Jboss Developer Studio 8.0

Un Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés) es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Los *IDEs* pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. Proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación (Alfaro, 2011).

JBoss Developer Studio proporciona un rendimiento superior para todo el ciclo de vida de desarrollo. Incluye un amplio conjunto de funciones de herramientas y soporte para múltiples modelos y marcos de programación, como Java ™ Enterprise Edition 6, RichFaces, JavaServer Faces (JSF), Enterprise JavaBeans (EJB), Java Persistence API (JPA) y Hibernate, HTML5, y muchas otras tecnologías populares.

PostgreSQL 9.4

Un Sistema de Gestión de Bases de Datos (SGBD) Consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a los mismos. El objetivo fundamental de un SGBD consiste en suministrar al usuario las herramientas que le permitan manipular, en términos abstractos, los datos, o sea, de forma que no le sea necesario conocer el modo de almacenamiento de los datos en la computadora, ni el método de acceso empleado.

Como Sistema Gestor de Base de Datos (SGBD) se utilizó PostgreSQL 9.4 que es un conjunto de programas no visibles al usuario final que se encargan de la privacidad, la integridad, la seguridad de los datos y la interacción con el sistema operativo (Oquendo, 2011).La versión 9.4 es la última estable del producto, que fue desarrollado por una comunidad conocida como PostgreSQL *Global Development Group* (PGDG). Por las características antes mencionadas y por ser de amplia utilización en nuestra universidad, se decidió utilizar este sistema.

Conclusiones del capítulo

- El análisis de la problemática y los conceptos necesarios para comprender el campo de acción de la investigación, permitió establecer la necesidad de obtener registros de eventos del módulo Bloque Quirúrgico en el Sistema de Información Hospitalaria XAVIA HIS.

- Se demuestra la imposibilidad de implantar una de las soluciones existentes en el mercado nacional e internacional por no considerar los elementos necesarios para el Sistema de Información Hospitalaria XAVIA HIS.

CAPÍTULO 2: PROPUESTA DE SOLUCIÓN

El objetivo fundamental del capítulo es describir la propuesta de solución, abordar los aspectos principales de la realización del componente informático de extracción de registros de eventos para el análisis de procesos quirúrgicos basado en un enfoque por procesos, describir el flujo de información y definir los patrones de diseño tenidos en cuenta para la realización del componente, además de mostrar el procedimiento seguido para el desarrollo del mismo.

2.1 Propuesta de solución

Los Sistemas de Información Hospitalarios almacenan datos de la ejecución de las actividades que se desarrollan dentro de una institución hospitalaria en su base de datos. Para utilizar los datos almacenados se propone el desarrollo de un componente informático de extracción de registro de eventos, representándose el flujo de información que lo caracteriza en la Figura 3. El mismo tiene como objetivo recolectar los datos de los eventos pertenecientes a la ejecución de los procesos configurados y estructurarlos jerárquicamente para luego proceder a organizarlas en un registro de eventos.

El componente ejecutará la tarea de generación del registro de eventos en tres pasos principales:

1. Carga la configuración de un proceso seleccionado desde un archivo en formato XML: luego de seleccionado el proceso se carga la configuración de este, construyendo los objetos que representan la jerarquía de etiquetas del archivo XML. Para finalizar se comprueba que esté bien configurado.
2. Una vez cargada y comprobada la configuración se realiza la extracción de los datos referentes a los eventos relacionados con la ejecución del proceso.
3. Por último se genera el registro de eventos en formato XES y es guardado en una dirección especificada.

Figura 3. Flujo de información del componente informático de extracción de registro de eventos.

Fuente: elaboración propia.

2.2 Descripción del componente informático

Como se menciona en el presente capítulo, la tarea de extracción de registro de eventos se desarrolla en tres pasos fundamentales: cargar la configuración de un proceso seleccionado, extraer los datos almacenados en la base de datos del sistema y generar el registro de eventos en formato XES. Para realizar cada uno de estos pasos se diseñan clases que tengan la responsabilidad de ejecutarlos.

La responsabilidad de guiar la tarea de extracción de registro de eventos se le asigna a la clase `ExtraccionControlador`, esta ejecuta la tarea como se muestra la tabla 1.

Tabla 2. Seudocódigo del algoritmo Generar registro de eventos.

Algoritmo	Generar registro de eventos
Entradas	Fecha de inicio y fin, proceso seleccionado
Salidas	Dirección del fichero en formato XES generado
Descripción	Permite la generación de un registro de evento luego de configurado el proceso en el sistema de información orientado a datos.

INICIO

1. Comprobar fechas
2. **Si** las fechas están correctas y hay un proceso seleccionado **entonces**
    2.1. *Cargar proceso*
    2.2. Generar ruta del registro de eventos
    2.3. **Si** se cargó el proceso y el registro de eventos no existe **entonces**
        2.3.1. *Construir registro de eventos*
        2.3.2. **Si** registro de eventos tiene trazas **entonces**
            2.3.2.1. Guardar registro de eventos
    2.4. **Sino**
        2.4.1. Almacenar mensajes de validación
        2.4.2. Cambiar valor de la ruta a vacío
3. Devolver ruta donde se guardó el fichero

FIN

La responsabilidad de guiar la tarea de carga de la configuración de los procesos es asignada a la clase `ConfiguracionControlador`, la cual es ejecutada como se describe a continuación.

Tabla 3. Seudocódigo del algoritmo Cargar proceso.

Algoritmo	Cargar proceso
Entradas	Proceso seleccionado
Salidas	Verdadero si se cargó correctamente el proceso o falso en caso contrario
Descripción	Permite cargar la configuración del proceso seleccionado y chequear esta configuración del proceso para posteriormente ser generado en un registro de eventos.

INICIO

1. **Si** hay un proceso seleccionado **entonces**
   1.1. Cargar configuración del proceso
   1.2. Chequear configuración del proceso
   1.3. **Si** hay errores registrados **entonces**
      1.3.1. Imprimir errores
      1.3.2. Devolver Falso
   1.4. Sino
      1.4.1. Devolver Verdadero
2. **Sino**
   2.1. Devolver Falso

FIN

La clase `ConfiguracionParser` tiene la responsabilidad de cargar la configuración del proceso seleccionado desde un archivo XML. A continuación se describe como realiza de manera general dicha tarea.

Tabla 4. Seudocódigo del algoritmo Cargar la configuración del proceso.

Algoritmo	Cargar la configuración del proceso

Entradas	Proceso seleccionado, fichero de configuración de los procesos
Salidas	Configuración cargada
Descripción	Permite cargar la configuración de un proceso previamente configurado, se convierte el XML de configuración en una estructura de objetos.

INICIO

1. Leer configuración del proceso
2. **Si** existe el proceso **entonces**
    2.1. Leer lista de extensiones
    2.2. **Si** lista de extensiones no es vacía **entonces**
        2.2.1. **Para** cada extensión
            2.2.1.1. Construir extensión
    2.3. Leer lista de globales
    2.4. **Si** lista de globales no es vacía **entonces**
        2.4.1. **Para** cada global
            2.4.1.1. Construir global
    2.5. Leer instancia de proceso
    2.6. **Si** instancia existe **entonces**
        2.6.1. Construir traza
    2.7. Leer lista de atributos
    2.8. **Si** lista de atributos no es vacía **entonces**
        2.8.1. **Para** cada atributo
            2.8.1.1. Construir atributo

FIN

La clase `ConfiguracionChecker` tiene la responsabilidad de comprobar que la configuración cargada del proceso esté correcta. A continuación se muestra de manera general cómo esta clase realiza la actividad.

Tabla 5. Seudocódigo del algoritmo Chequear configuración del proceso.

Algoritmo	Chequear configuración del proceso
Entradas	Configuración cargada del proceso
Salidas	Lista de errores
Descripción	Permite comprobar que el proceso seleccionado esté bien configurado. Este algoritmo se realiza siguiendo el patrón del visitante.

INICIO

1. Obtener lista de atributos
2. **Para** cada atributo
   2.1. Visitar atributo
3. Obtener lista de extensiones
4. **Para** cada extensión
   4.1. Visitar extensión
5. Obtener lista de globales
6. **Si** longitud de lista de globales mayor que 2 **entonces**
   6.1. Adicionar error “No puede configurarse más de dos globales”
7. **Sino**
   7.1. **Para** cada global
      7.1.1. Visitar global
8. Visitar traza

FIN

Por último la clase ProcesoConf tiene la responsabilidad de crear los elementos que conforman el registro de eventos en formato XES. La clase lo realiza como se describe a continuación.

Tabla 6. Seudocódigo del algoritmo Construir registro de eventos.

Algoritmo	Construir registro de eventos
Entradas	Fecha de inicio y fecha de fin
Salidas	Objeto *XLog*
Descripción	Permite la extracción de los datos de los eventos configurados en un rango de fecha determinado y estructurarlos en el formato XES

INICIO

1. Crear objeto log de tipo XLog
2. **Para** cada atributo
    2.1. Convertir atributo al formato XES
    2.2. **Si** atributo convertido no es nulo **entonces**
        2.2.1. Adicionar a log el atributo convertido
3. **Para** cada extensión
    3.1. Convertir extensión al formato XES
    3.2. **Si** extensión convertida no es nulo **entonces**
        3.2.1. Adicionar a log la extensión convertida
4. **Para** cada global
    4.1. Convertir global al formato XES
    4.2. **Si** global no es nulo **entonces**
        4.2.1. Adicionar a log el global convertido
5. Generar listado de trazas
6. **Si** listado de trazas no es nulo **entonces**
    6.1. **Para** cada traza del listado de trazas
        6.1.1. Adicionar traza a log
7. Devolver log

FIN

Mostrándolo desde otro punto de vista, en la tabla 6 se resumen, según el orden de ejecución, las actividades que se realizan para ejecutar la tarea de extracción.

Tabla 7. Actividades de la tares de extracción.

Actividad	**Clase responsable**
1. Generar el registro de eventos	ExtraccionControlador
1.1. Cargar el proceso	ConfiguracionControlador
1.1.1. Cargar la configuración del proceso	ConfiguracionParser
1.1.2. Chequear la configuración del proceso	ConfiguracionChecker
1.2. Construir el registro de eventos	ProcesoConf

2.2.1 Dependencias del componente

A continuación se enumeran las bibliotecas de las que depende el componente:

OpenXES 2.0: es una implementación de referencia del estándar XES para almacenar y gestionar los datos de registro de eventos. Los dominios de aplicación del estándar XES, y la biblioteca OpenXES, son múltiples. Incluyen, pero no se limitan a la supervisión de los sistemas de información conscientes de proceso, la minería y proceso de análisis general de procesos, y de minería de datos. Esta es liberada como de código abierto / software libre, bajo los términos de la GPL (LGPL) licencia GNU. Esta depende de la biblioteca Spex en su versión 1.0 y Google Guava en su versión 15.0.

Jdom 1.1.1: proporciona un medio robusto y ligero de leer y escribir datos XML. Proporciona un punto de entrada de bajo costo para el uso de XML. Está disponible bajo una licencia de código abierto de estilo Apache, con la cláusula de reconocimiento eliminado. Permite a los desarrolladores usar JDOM en la creación de nuevos productos sin que tengan que liberar sus propios productos como código abierto.

2.3 Patrones de diseño

Los patrones de diseño son soluciones a problemas repetidos en la construcción de software y en ocasiones pueden incluir sugerencias para aplicar estas soluciones en diversos entornos (Kaisler 2005). En el diseño de la presente investigación se utilizaron algunos patrones de diseño GRASP (traducido al español Patrones Generales de Software de Asignación de Responsabilidades), para solucionar y/o evitar diferentes problemas que pudieran aparecer durante la implementación de la solución.

GRASP

Experto: es el principio básico de asignación de responsabilidades. Nos indica, por ejemplo, que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. De este modo obtendremos un diseño con mayor cohesión y así la información se mantiene encapsulada (Almaguer, 2011). Se observa el uso de este patrón en la clase `ExtraccionControlador`.

Creador: el patrón creador nos ayuda a identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases. La nueva instancia deberá ser creada por la clase que: tiene la información necesaria para realizar la creación del objeto, usa directamente las instancias creadas del objeto, almacena o maneja varias instancias de la clase o contiene o agrega la clase. Una ventaja es el bajo acoplamiento, lo cual supone facilidad de mantenimiento y reutilización (Almaguer, 2011). El uso de este patrón se evidencia en las clases `ConfiguracionControlador`, `ConfiguracionParser`, `ConfiguracionChecker`, `ExtraccionControlador`.

Alta cohesión: los conceptos de cohesión y acoplamiento están íntimamente relacionados. Un mayor grado de cohesión implica un menor de acoplamiento. Plantea que la información que almacena una clase debe de ser coherente y debe estar (en la medida de lo posible) relacionada con la clase. (Almaguer, 2011)

La totalidad de las clases del componente evidencian el uso de este patrón, cada una contiene, principalmente, la información que le pertenece de forma coherente.

Bajo acoplamiento: es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases (Almaguer, 2011). El uso de este patrón se evidencia en la poca relación existente entre las clases que conforman los módulos.

Teniendo en cuenta que se implementaron clases altamente cohesionadas se garantizó el bajo acoplamiento de las mismas.

2.4 Estándares de codificación

La adopción de estándares de estilo y codificación son de vital importancia para asegurar la calidad del software. El uso de los mismos tiene innumerables ventajas tales como:

- Asegurar la legibilidad del código entre distintos programadores, facilitando la depuración del programa.
- Proveer una guía para el encargado de mantenimiento/actualización del sistema, con código claro y bien documentado.
- Facilitar la portabilidad entre plataformas y aplicaciones.

Un código fuente completo debe reflejar un estilo armonioso, como si todo el código fuera escrito por un único programador. Si bien los programadores deben implementar un estándar de forma prudente, este debe estar bien definido a nivel departamental, por tanto, al comenzar un proyecto de software es necesario establecer un estándar de codificación único para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada. Las convenciones de código o estándares de codificación son importantes para los programadores por un gran número de razones (Calleja, 2014):

- El 80% del coste del código de un programa va a su mantenimiento.
- Casi ningún software es mantenido toda su vida por el autor original.

- Las convenciones de código mejoran la lectura del software lo que permite entender código nuevo de manera más óptima y rápida.
- Si distribuyes tu código fuente como un producto, necesitas asegurarte de que está bien hecho y presentado como cualquier otro producto.

A continuación se presentan algunos de los estándares de codificación definidos y aplicados al sistema:

- Se debe utilizar como idioma el español, las palabras no se acentuarán.
- Todos los ficheros fuentes deben comenzar con un comentario en el que se lista el nombre de la clase, información de la versión, fecha y copyright.
- Las líneas en blanco mejoran la facilidad de lectura separando secciones de código que están lógicamente relacionadas. Se deben usar siempre dos líneas en blanco en las siguientes circunstancias:
    - Entre las secciones de un fichero fuente.
    - Entre las definiciones de clases e interfaces.
- Se debe usar siempre una línea en blanco en las siguientes circunstancias:
    - Entre métodos.
    - Entre las variables locales de un método y su primera sentencia.
    - Antes de un comentario de bloque o de un comentario de una línea.
    - Entre las distintas secciones lógicas de un método para facilitar la lectura.
- Se debe dar un espacio en blanco en la siguiente situación:
    - Entre una palabra clave del lenguaje y un paréntesis.
- Respecto a las normas de inicialización, declaración y colocación de variables, constantes, clases y métodos:
    - Todas las instancias y variables de clases o métodos empezarán con minúscula. Las palabras internas que lo forman, si son compuestas, empiezan con su primera letra en mayúsculas. Los nombres de variables no deben empezar con los caracteres guion bajo "_" o signo de peso "$", aunque ambos están permitidos por el lenguaje.

- Los nombres de variables de un solo carácter se deben evitar, excepto para variables índices temporales.
- Los nombres de las variables declaradas como constantes deben aparecer totalmente en mayúscula separando las palabras con un guion bajo ("_").
- Los nombres de las clases deben ser sustantivos, cuando son compuestos tendrán la primera letra de cada palabra que lo forma en mayúscula. Mantener los nombres de las clases simples y descriptivas. Usar palabras completas, evitar acrónimos y abreviaturas.
- Los métodos deben ser verbos, cuando son compuestos tendrán la primera letra en minúscula y la primera letra de las siguientes palabras que lo forman en mayúscula.

- Respecto a la identación[1] y longitud de la línea:
  - Se deben emplear cuatro espacios como unidad de identación. La construcción exacta de la identación (espacios en blanco contra tabuladores) no se especifica. Los tabuladores deben ser exactamente cada ocho espacios.
  - Evitar las líneas de más de ochenta caracteres, ya que no son manejadas bien por muchas terminales y herramientas.

A continuación se muestra un fragmento de código (Ver figura 4) en la que se evidencia el uso de los patrones anteriormente expuestos.

[1] Identación: Es un anglicismo (de la palabra inglesa indentation) de uso común en informática y significa mover un bloque de texto hacia la derecha insertando espacios o tabuladores para separarlo del texto adyacente, lo que en el ámbito de la imprenta se ha denominado siempre como sangrado o sangría.

```
56⊖    public void generarFicheroXES()
57             throws JDOMException, IOException, URISyntaxException{
58
59         String comprobarFech = comprobarFechas(fechaInicio, fechaFin);
60         if(comprobarFech == "" && proceso != "") {
61             XLogging.log("Inicio de la extracción.", Importance.INFO);
62             long start = System.currentTimeMillis();
63
64             boolean seCargo = configuracion.cargarProceso(proceso,
65                         entityManager);
66             File archivo = ExtraccionUtil.generarRuta(pathDirectory,
67                             proceso, fechaInicio, fechaFin);
68             if(seCargo && !archivo.exists()) {
69                 XLog l = configuracion.getProceso()
70                             .convertirElementoXES(fechaInicio, fechaFin,
71                                         entityManager);
72                 if(!l.isEmpty()) {
73                         ExtraccionUtil.guardarRegistroEventos(archivo, l);
74                 }
75             }
76             XLogging.log("Fin de la extracción."+" ("
77                         + (System.currentTimeMillis() - start) + " msec.)",
78                         Importance.INFO);
79         }
80         else {
81             mensajes += comprobarFech;
82             XLogging.log("Fin de la extracción.", Importance.WARNING);
83         }
84     }
```

Figura 4. Código perteneciente al método generarFicheroXES de la clase ExtraccionControlador. Fuente: elaboración propia.

2.5 Aplicación del componente informático en el Sistema de Información Hospitalaria XAVI HIS

Para aplicar el componente de extracción de registro de eventos desarrollado al Sistema de Información Hospitalaria XAIA HIS se siguieron una serie de pasos los cuales se enumeran a continuación:

1. Formular las preguntas que servirán de guía para el análisis de los procesos quirúrgicos.
2. Analizar los procesos quirúrgicos y seleccionar los datos a extraer para cada evento.
3. Configuración de los procesos.

2.5.1 Formular las preguntas que servirán de guía para el análisis de los procesos quirúrgicos

El módulo bloque quirúrgico del Sistema de Información Hospitalaria XAVIA HIS cuenta con los siguientes procesos de negocio:

- **Gestión de recursos:** se planifican recursos con el objetivo de garantizar que las intervenciones quirúrgicas se realicen con la calidad requerida. Los quirófanos, los equipos especiales, los dispositivos a implantar (en caso que se requiera), los kits de materiales quirúrgicos y de anestesia, el personal que llevará a cabo la intervención, son algunos de los recursos que se planifican con antelación en función de una intervención quirúrgica.

- **Gestión del plan quirúrgico:** tiene como objetivo la planificación de las intervenciones quirúrgicas para una fecha determinada. De acuerdo a su carácter las cirugías pueden ser electivas, ambulatorias o de emergencia. Las dos primeras pueden planificarse a mediano o largo plazo, provienen fundamentalmente de las áreas de consulta externa y hospitalización y se diferencian en que las electivas requieren hospitalización previa y posterior a la cirugía, la ambulatoria como no precisa de una recuperación compleja puede hacerse fuera del hospital. En el caso de las emergencias se requiere premura y entran por tanto directamente al plan quirúrgico del propio día, en ocasiones desplazando para otro día, cirugías programadas para la fecha.

  Las intervenciones planificadas se realizan para una fecha y pasan finalmente al plan quirúrgico un día antes de la fecha prevista, en caso que sea para los lunes se realiza los viernes. Los fines de semana está previsto que solo se realicen cirugías de

emergencia. Estas se aprueban o se rechazan y las aprobadas pasan al plan quirúrgico que es la principal y única entrada a los procesos asistenciales del área de bloque quirúrgico.

- **Realizar intervención quirúrgica:** es el proceso de mayor importancia en el módulo Bloque Quirúrgico pues se encarga de gestionar toda la información vinculada a la cirugía y al proceso de anestesia en el paciente. Genera importantes documentos clínicos que forman parte de la Historia Clínica Electrónica de un paciente, como son: La nota operatoria, preoperatorio, transoperatorio, posoperatorio y la evolución médica, esta última utilizada en la recuperación.

- **Gestión de solicitudes:** en el módulo Bloque Quirúrgico tienen lugar salidas a otras áreas del hospital propias de la interacción que origina el proceso de una cirugía. Por ejemplo se necesita realizar solicitudes de productos al almacén de Bloque Quirúrgico para que se garanticen los insumos mínimos indispensables configurados en los kits quirúrgicos para realizar el procedimiento. Es necesario además solicitar los medicamentos y anestésicos necesarios a la farmacia para la cirugía. También el paciente puede necesitar ser transfundido, por lo que se brinda la posibilidad de solicitar una transfusión de algún hemoderivado o de varios (incluyendo sangre total) al Banco de Sangre.

  Durante la cirugía el cirujano puede determinar si en el tejido sobre el cual realizó la intervención es necesario realizar una biopsia, la cual se envía a un Instituto de Anatomía Patológica en caso que el hospital no cuente con un departamento para estos fines. Por otra parte cuando un paciente entra por emergencia al área de Bloque quirúrgico no cuenta con servicio de hospitalización, por lo que se posibilita realizar desde esta propia área una orden de admisión.

- **Gestión de los reportes:** en el módulo Bloque quirúrgico se generan tres reportes que corresponden a salidas del sistema de acuerdo a procesos realizados en el flujo de información gestionada mediante el sistema. Permiten conocer las causas de la cancelación de intervenciones quirúrgicas tanto electivas como de emergencia en un

rango de fechas introducido por el usuario. Además se puede conocer el total de cirugías realizadas de acuerdo a los servicios y al carácter de las mismas en un rango de fechas especificado por el usuario.

Basándonos en el principio rector número dos de la minería de procesos que explica que la extracción de registros de eventos debería ser impulsada por preguntas que son la guía para seleccionar los atributos de los eventos a mostrar en el registro de eventos, se definieron para el análisis de la información contenida en los procesos de negocio, las siguientes preguntas que servirán como guía en el análisis descrito:

- ¿Cuál es la tendencia de solicitud de productos y consumo de Kits en función de determinado procedimiento quirúrgico?
- ¿Cómo se comportan los equipos de trabajo quirúrgicos respecto a la planificación inicial y planificación final de personal a operar?
- ¿Coincide el tiempo planificado con el tiempo que realmente dura la intervención quirúrgica?
- ¿Cómo se comportan las cirugías realizadas por servicios y especialistas?
- ¿Cuál es la procedencia de la solicitud de intervención quirúrgica?
- ¿Cómo se manifiestan las tendencias de aplicación y cantidad de medicamentos y material de anestesia ante determinada dolencia, diagnóstico realizado o procedimiento quirúrgico practicado?

2.5.2 Analizar los procesos quirúrgicos y seleccionar los datos a extraer por cada evento

Partiendo de las necesidades de análisis de información presentes en el módulo Bloque Quirúrgico del Sistema de Información Hospitalaria XAVIA HIS descritas en la introducción de dicho documento y para dar respuesta a las interrogantes planteadas, se identificaron dos procesos que engloban la información contenida en dichos procesos de negocio. Los procesos están compuestos por actividades, para cada actividad se muestra la tabla de la base de datos donde se encuentra la evidencia de su ejecución para una instancia de proceso

determinada. En la columna correspondiente a la tabla de la base de datos se representan los datos en la forma:<nombre de esquema>.<nombre de tabla>.

Tabla 8. Procesos definidos.

Proceso	**Instancia de proceso**	**Actividad**	**Tabla de la base de datos**
Atender paciente quirúrgico	solicitud_intervencion.id	Solicitar intervención quirúrgica	publico.solicitud_intervencion
		Crear preoperatorio	quirofano.preoperatorio
		Crear transoperatorio	quirofano.transoperatorio
		Crear nota operatoria	publico.nota_operatoria
		Crear postoperatorio	quirofano.pos_operatorio
Realizar transfusión de sangre		Solicitar transfusión de sangre	publico.solicitud_transfusion
		Aceptar solicitud de transfusión de sangre	publico.solicitud_transfusion
		Rechazar solicitud de transfusión de sangre	publico.solicitud_transfusion
		Realizar transfusión de sangre	publico.solicitud_transfusion
		Solicitar transfusión de sangre	publico.solicitud_intervencion

2.5.3 Configuración de los procesos

El archivo de configuración constituye la columna vertebral del componente pues es ahí donde se declaran los elementos necesarios para la extracción del registro de eventos. A continuación (Ver figura 5) se muestra una vista simple de la jerarquía del archivo en formato xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<procesos>
    <proceso>
        <extension/>
        <global>
            <atributo/>
        </global>
        <atributo/>
        <instancia>
            <atributo/>
            <evento>
                <atributo/>
            </evento>
        </instancia>
    </proceso>
</procesos>
```

Figura 5. Jerarquía del archivo en formato XML.

Fuente: elaboración propia.

A continuación se muestra las etiquetas que componen el archivo de configuración de procesos en formato XML:

- procesos: es la etiqueta raíz del archivo y se utiliza para agrupar la configuración de los distintos procesos. Esta puede contener más de una etiqueta proceso.
- proceso: esta etiqueta se utiliza para contener las etiquetas que son necesarias para la extracción del registro de eventos, además de poseer información adicional que

ofrezca una descripción sobre la configuración del registro. Puede contener más de una etiqueta extensión, global y atributo, pero solo una de instancia.

- extensión: es la etiqueta que describe las distintas extensiones que contiene el formato XES para la extracción de registro de eventos. En este caso se definen tres atributos para esta etiqueta:
  - nombre: sirve para especificar el nombre de la extensión.
  - prefix: representa el prefijo que se va a usar en los atributos que declaren tener esta extensión.
  - uri: este atributo representa la dirección del archivo de definición de la extensión.

En la figura 6 se muestra un ejemplo de configuración de la etiqueta extensión.

```
<extension nombre="Lifecycle" prefix="lifecycle" uri="http://www.xes-standard.org/lifecycle.xesext"/>
```

Figura 6. Configuración de la etiqueta extensión.
Fuente: elaboración propia.

- global: describe los atributos que deben estar contenidos en los elementos *event* o *trace* de un registro de eventos. Si esta etiqueta es declarada entonces cuando se extrae el registro de eventos los atributos que esta contienen son de obligatoria aparición en los elementos bajo el contexto que se ha especificado. Además esta etiqueta contiene el atributo *scope* que define a qué elementos del registro de eventos esta regla va a afectar.

```
<global scope="event">
    <atributo type="date" key="time:timestamp" valor=""/>
</global>
```

Figura 7. Ejemplo de configuración de la etiqueta global.
Fuente: elaboración propia.

- instancia: especifica dónde encontrar las instancias de ejecución del proceso configurado, además de contener las etiquetas *evento.* Cuenta con el atributo *query*,

que es con el que se especifica la consulta para seleccionar las instancias de procesos. Por otro lado puede contener atributos.

```
<instancia query="SELECT id FROM publico.solicitud_intervencion">
    <atributo/>
    <evento>
        <atributo/>
    </evento>
</instancia>
```

Figura 8. Ejemplo de configuración de la etiqueta instancia.
Fuente: elaboración propia.

- evento: con esta etiqueta se describe el elemento *event* de un registro de eventos. Esta contiene etiquetas *atributo* según se hayan declarado. Además se deben especificar los atributos de etiqueta:
    - campoInstancia: define el campo de la consulta a la base de datos del evento que contiene el identificador de la instancia de proceso.
    - query: este atributo contiene las tablas de la base de datos de dónde se va a extraer la información correspondiente al evento.

```
<evento campoInstancia="solicitud_intervencion.id" query="FROM publico.solicitud_intervencion">
    <atributo />
    <atributo />
</evento>
```

Figura 9. Ejemplo de configuración de la etiqueta evento.
Fuente: elaboración propia.

- atributo: esta etiqueta describe los datos que se van a extraer, para enriquecer a un evento. Se puede destacar que esta etiqueta puede ser hija de *proceso*, *instancia* y *evento*. Para lograr una correcta extracción de los datos de los eventos, se definieron los siguientes atributos que debe tener la etiqueta *atributo*:
    - type: este atributo indica el tipo de dato del atributo.
    - key: define la clave de la extensión que representa este atributo.

- esValor: indica si el valor del atributo es un dato ya predefinido o es un campo de la base de datos.
- compuesto: este atributo indica si un atributo es compuesto, es decir que involucra más de dos campos de la base de datos.
- valor: en este atributo se define el valor del atributo.

```
<atributo type="date" key="time:timestamp" esValor="false" compuesto="false" valor="solicitud_intervencion.fecha_solicitud"/>
<atributo type="string" key="lifecycle:transition" esValor="true" compuesto="false" valor="complete"/>
<atributo type="string" key="concept:name" esValor="true" compuesto="false" valor="Solicitar intervencion quirurgica"/>
```

Figura 10. Ejemplo de configuración de la etiqueta atributo.

Fuente: elaboración propia.

Luego de planteados los elementos de configuración de procesos se presenta un ejemplo general del archivo XML de configuración.

```
<?xml version="1.0" encoding="UTF-8"?>
<procesos>
    <proceso nombre="Atender paciente">
        <extension nombre="Concept" prefix="concept" uri="http://www.xes-standard.org/concept.xesext"/>
        <extension nombre="Time" prefix="time" uri="http://www.xes-standard.org/time.xesext"/>
        <extension nombre="Lifecycle" prefix="lifecycle" uri="http://www.xes-standard.org/lifecycle.xesext"/>
        <extension nombre="Org" prefix="org" uri="http://www.xes-standard.org/org.xesext"/>

        <global scope="event">
            <atributo type="date" key="time:timestamp" valor=""/>
        </global>

        <instancia query="SELECT id FROM publico.solicitud_intervencion WHERE fecha_solicitud BETWEEN ?1 AND ?2">
            <evento campoInstancia="solicitud_intervencion.id" query="FROM publico.solicitud_intervencion">
                <atributo type="date" key="time:timestamp" esValor="false" compuesto="false" valor="solicitud_intervencion.fecha_solicitud"/>
                <atributo type="string" key="lifecycle:transition" esValor="true" compuesto="false" valor="complete"/>
                <atributo type="string" key="concept:name" esValor="true" compuesto="false" valor="Solicitar intervencion quirurgica"/>
            </evento>
        </instancia>
    </proceso>
</procesos>
```

Figura 11. Ejemplo del archivo XML de configuración.

Fuente: elaboración propia.

Conclusiones del capítulo

- El desarrollo del componente de extracción de registros de eventos sirvió como punto de partida para la posterior aplicación de técnicas de minería de procesos.
- La descripción del procedimiento realizado para el desarrollo del componente propuesto permitió detallar cada una de las fases ejecutadas para la realización del mismo.
- El uso de los patrones de diseño, permitió organizar la implementación de la propuesta de solución.
- La implementación del componente descrito en el presente capítulo, demostró que es posible lograr una aplicación con las características que se determinaron previamente en la presente investigación, permitiendo generar registros de eventos para su posterior análisis aplicando técnicas de minería de procesos.

CAPÍTULO 3: VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

El objetivo del presente capítulo es validar la solución propuesta y mostrar los resultados arrojados por el componente de extracción de registros de eventos, a continuación se presentan los temas a tratar en el capítulo:

- Realización de pruebas de software.
- Realización de pruebas de rendimiento.
- Validación y comprobación del registro de eventos.

3.1 Realización de pruebas de software

El único instrumento adecuado para determinar el status de la calidad de un producto de software es el proceso de pruebas. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad, con el objetivo de medir el grado en que el software cumple con los requerimientos. El proceso de pruebas, sus objetivos y los métodos y técnicas usados se describen en el plan de prueba («Pruebas de software», 2005).

3.1.1 Pruebas funcionales

Se denominan pruebas funcionales a las pruebas de software que tienen por objetivo probar que los sistemas desarrollados, cumplan con las funciones específicas para los cuales han sido creados (Oré B., 2009).

3.1.2 Pruebas de caja blanca

Las pruebas de caja blanca son pruebas funcionales, estas son un método de diseño que usa la estructura de control descrita como parte del diseño al nivel de componentes para derivar los casos de prueba. Al emplear sus métodos, se puede garantizar que, al menos una vez, se ejecuten todas las rutas independientes del módulo. La técnica del camino básico permite

derivar casos de prueba a partir de un conjunto de caminos independientes por los cuales puede circular el flujo de control (Pressman, 2005).

Para obtener el conjunto de caminos independientes se construye el grafo de flujo asociado y se calcula su complejidad ciclomática. Para poder elaborar el grafo de flujo, primero se deben enumerar las sentencias del código:

A continuación se muestra un fragmento de código perteneciente a la clase desarrollada `ExtraccionRegistroEventosControlador`.

```
public String ge 1 rFicheroXES() thro 2 OMException, IOExcept 3
    if(!proceso.equals("") && !fechaFin.equals(null) && !fechaInicio.equals(null)){
        configuracion.setPathFicheroConf(pathConfig); 4
        boolean seCargo = configuracion.cargarProceso(proceso); 4
        String registro = ""; 4
        if(seCargo){ 4
            registro = configuracion.construirRegistroEventosXES(fechaInicio, fechaFin, entityManager); 5
            if(pathDirectory == null){ 5
                get(); 6
            }
            String ruta = pathDirectory.getAbsolutePath()+File.separatorChar; 7
            if(nombreFicheroXES == ""){ 7
                String nomb = ""; 8
                Calendar cal1 = Calendar.getInstance(); 8

                nomb = proceso+" "+cal1.get(Calendar.DATE)+"-"+cal1.get(Calendar.MONTH)
                        +"-"+cal1.get(Calendar.YEAR)+" "+cal1.get(Calendar.HOUR)
                        +","+cal1.get(Calendar.MINUTE)+","+cal1.get(Calendar.SECOND)
                        +"."+cal1.get(Calendar.MILLISECOND); 8
                ruta += nomb+".xes"; 8
            }
            else
                ruta += nombreFicheroXES+".xes"; 9
            File archivo = new File(ruta); 10
            BufferedWriter bw = new BufferedWriter(new FileWriter(archivo)); 10
            bw.write(registro); 10
            bw.close(); 10
            return ruta; 10
        }
    }
    return ""; 11
}
```

Figura 12. Código fuente del método generarFicheroXES.

Fuente: elaboración propia.

Posteriormente se debe proceder a la elaboración del grafo de flujo teniendo en cuenta dicha enumeración.

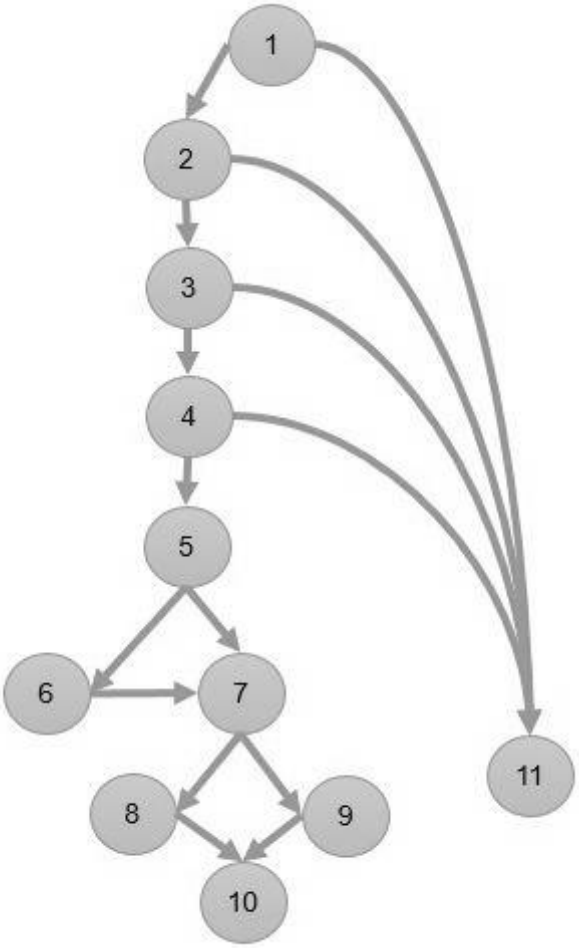

Figura 13. Grafo de flujo asociado a la funcionalidad generarFicheroXES.

Fuente: elaboración propia.

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).

La complejidad ciclomática se basa en la teoría gráfica y se calcula de dos maneras distintas, donde, para que el cálculo sea correcto, todas deben arrojar el mismo resultado:

1. El número de regiones corresponde a la complejidad ciclomática **"V (G)".**
   V (G) = R Donde R es la cantidad total de regiones.
   V (G) = 6
2. **V (G)** = E – N + 2 Donde E es el número de aristas y N número de nodos.
   V (G) = 15 – 11 + 2
   V (G) = 6

Teniendo en cuenta el valor obtenido de la complejidad de la funcionalidad generarFicheroXES, se obtuvo la cantidad de posibles caminos independientes y cantidad de pruebas que se deben realizar para comprobar que cada sentencia se ejecute al menos una vez.

El cálculo arrojó que V (G) = 4, por lo que los posibles caminos básicos son:

- Camino básico Nº 1: 1, 11
- Camino básico Nº 2: 1, 2,11
- Camino básico Nº 3: 1, 2, 3, 11
- Camino básico Nº 4: 1, 2, 3, 4,11
- Camino básico Nº 5: 1, 2, 3, 4, 5, 7, 9, 10
- Camino básico Nº 6: 1, 2, 3, 4, 5, 6, 7, 8, 10

Luego de elaborados el grafo de flujo y los caminos básicos a recorrer, se elaboran los casos de prueba correspondientes a cada camino con los que se garantizará que cada sentencia de código se ejecute al menos una vez en cada caso identificado.

Caso de prueba para camino básico Nº 1 (1, 11).

proceso == ""

resultado esperado: return ""

Caso de prueba para camino básico Nº 2 (1, 2, 11).

proceso != "" and fechaInicio == null
resultado esperado: return ""

Caso de prueba para camino básico Nº 3 (1, 2, 3, 11).

proceso != “” and fechaInicio !=null and fechaFin == null
resultado esperado: return “”

Caso de prueba para camino básico Nº 4 (1, 2, 3, 4, 11).

proceso != “” and fechaInicio !=null and fechaFin != null and seCargo == false
resultado esperado: return “”

Caso de prueba para camino básico Nº 4 (1, 2, 3, 4, 5, 7, 9, 10).

proceso != “” and fechaInicio !=null and fechaFin != null and seCargo = true and pathDirectory != null and nombreFicheroXES != “”
resultado esperado: return ruta

Caso de prueba para camino básico Nº 4 (1, 2, 3, 4, 5, 6, 7, 8, 10).

proceso != “” and fechaInicio !=null and fechaFin != null and seCargo = true and pathDirectory != null and nombreFicheroXES == “”
resultado esperado: return ruta

3.1.3 Resultado de las pruebas

En la siguiente tabla (ver Tabla 8) se recogen los datos de las no conformidades detectadas luego de realizar la prueba descrita.

Tabla 9. Tabla de las no conformidades detectadas al código.

Tipo de no conformidades	Cantidad de no conformidades
Primera Iteración	
Significativas	3
No significativas	1
Total	4

Segunda Iteración	
Significativas	0
No significativas	0
Total	0

A continuación se muestra un gráfico con el porciento que representan del total las no conformidades por cada una de las iteraciones clasificándolas en significativas o no significativas.

Figura 14. Por ciento de las no conformidades encontradas.

Fuente: elaboración propia.

3.2 Realización de pruebas de rendimiento

Las pruebas de rendimiento pueden servir para diferentes propósitos (Madeja, 2015):

- Demostrar que el sistema cumple los criterios de rendimiento.
- Comparar dos sistemas para encontrar cuál de ellos funciona mejor.

- Medir que partes del sistema o de carga de trabajo provocan que el conjunto rinda mal.

Para su diagnóstico, los ingenieros de software utilizan herramientas como pueden ser monitorizaciones que midan qué partes de un dispositivo o software contribuyen más al mal rendimiento o para establecer niveles (y umbrales) del mismo que mantenga un tiempo de respuesta aceptable (Madeja, 2015).

Para realizar una valoración de rendimiento para el componente de extracción de registro de eventos, es necesario definir un entorno de prueba. Se tiene una computadora con un procesador Intel(R) Celeron Dual Core CPU a 2.10 GHz de velocidad y 3GB de memoria RAM. La computadora tiene instalado el sistema operativo Windows 8. Para esto se generaron tres registros de eventos en diferentes rangos de fechas para el proceso Atender paciente quirúrgico, con el objetivo de ver el tiempo de generación de dichos registros de eventos (Ver tabla 9) y llegar a las siguiente conclusión.

Tabla 10: Tabla de las estadísticas de los registros de eventos generados.

Proceso	**Fecha Inicio**	**Fecha Fin**	**Cantidad de trazas**	**Cantidad de eventos**	**Tiempo de generación (s)**
Atender paciente quirúrgico	01/01/2013	31/01/2013	11	39	2,808
	01/01/2013	31/12/2013	853	1833	23,011
	01/01/2013	09/06/2016	2933	4231	54,366

El promedio del tiempo de generación fue de 26,72 segundos, para un promedio de eventos de 2034 y un promedio de trazas de 1266.

El análisis realizado a partir de los resultados obtenidos, permite garantizar que los tiempos de ejecución del componente desarrollado son relativamente pequeños si se tiene en cuenta las características de hardware de la computadora. Esto posibilita que el procesamiento de la

fuente de datos para obtener los resultados no afecte el rendimiento del Sistema de Información Hospitalaria XAVIA HIS.

3.3 Validación y comprobación del registro de eventos

Los registros de eventos que se obtuvieron a partir del componente desarrollado fueron validados mediante la comparación con los registros de eventos obtenidos con la herramienta XESame, lo cual demostró que ambos registran la misma información. A continuación se muestra un ejemplo de un registro de eventos que se obtuvo por el componente desarrollado.

```
<?xml version="1.0" encoding="UTF-8"?>
<log version="" features="">
        <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"/>
        <extension name="Lifecycle" prefix="lifecycle" uri="http://www.xes-standard.org/lifecycle.xesext"/>
        <extension name="Organizational" prefix="org" uri="http://www.xes-standard.org/org.xesext"/>
    <trace>
        <event>
            <date key="time:timestamp" value="2015-01-08" />
            <string key="lifecycle:transition" value="complete" />
            <string key="concept:name" value="Solicitar intervencion quirurgica" />
        </event>
    </trace>
    <trace>
        <event>
            <date key="time:timestamp" value="2015-01-01" />
            <string key="lifecycle:transition" value="complete" />
            <string key="concept:name" value="Solicitar intervencion quirurgica" />
        </event>
    </trace>
    <trace>
        <event>
            <date key="time:timestamp" value="2015-01-02" />
            <string key="lifecycle:transition" value="complete" />
            <string key="concept:name" value="Solicitar intervencion quirurgica" />
        </event>
    </trace>
    <trace>
        <event>
            <date key="time:timestamp" value="2015-01-02" />
            <string key="lifecycle:transition" value="complete" />
            <string key="concept:name" value="Solicitar intervencion quirurgica" />
        </event>
    </trace>
```

Figura 1. Registro de eventos generado por el componente desarrollado.
Fuente: elaboración propia.

Además se muestra un registro de eventos obtenido por la herramienta XESame.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- This file has been generated with the OpenXES library. It conforms -->
<!-- to the XML serialization of the XES standard for log storage and -->
<!-- management. -->
<!-- XES standard version: 1.0 -->
<!-- OpenXES library version: 1.0RC7 -->
<!-- OpenXES is available from http://www.openxes.org/ -->
<log xes.version="1.0" xes.features="nested-attributes" openxes.version="1.0RC7">
    <extension name="Organizational" prefix="org" uri="http://www.xes-standard.org/org.xesext"/>
    <extension name="Time" prefix="time" uri="http://www.xes-standard.org/time.xesext"/>
    <extension name="Lifecycle" prefix="lifecycle" uri="http://www.xes-standard.org/lifecycle.xesext"/>
    <extension name="Semantic" prefix="semantic" uri="http://www.xes-standard.org/semantic.xesext"/>
    <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org/concept.xesext"/>
    <global scope="event">
        <string key="concept:name" value="UNKNOWN"/>
        <string key="lifecycle:transition" value="UNKNOWN"/>
        <date key="time:timestamp" value="1970-01-01T00:00:00+01:00"/>
    </global>
    <classifier name="Activity classifier" keys="&apos;concept:name lifecycle:transition&apos;"/>
    <classifier name="Resource classifier" keys="org:resource"/>
    <trace>
        <event>
            <string key="concept:instance" value="10010000000000004395"/>
            <string key="concept:name" value="Crear solicitud de intervencion"/>
            <string key="lifecycle:transition" value="complete"/>
            <date key="time:timestamp" value="2015-01-01T00:00:00+01:00"/>
        </event>
    </trace>
    <trace>
        <event>
            <string key="concept:instance" value="10010000000000004396"/>
            <string key="concept:name" value="Crear solicitud de intervencion"/>
            <string key="lifecycle:transition" value="complete"/>
            <date key="time:timestamp" value="2015-01-02T00:00:00+01:00"/>
        </event>
    </trace>
```

Figura 16. Registro de eventos generado por la herramienta XESame.

Fuente: XESame.

Los registros de eventos generados tanto por la herramienta XESame como por el componente desarrollado se configuraron según los eventos y atributos definidos del proceso atender paciente quirúrgico, en el rango de fecha del 1 de enero del 2015 hasta el 31 de enero del 2015.

Después de generados los registros de eventos se utilizó la herramienta ProM para aplicarle técnicas de descubrimiento y comparar la similitud entre los registros de eventos obtenidos y ver si el registro de eventos generado por el componente es aceptado por esta herramienta. Se puede observar que la herramienta muestra el mismo resultado para ambos casos, el cual quedaría como se muestra a continuación (Ver figura 17).

Figura 17. Ejemplo en la herramienta ProM para el componente desarrollado y el XESame.
Fuente: ProM.

Con este análisis se llegó a la conclusión que ambos registros de eventos muestran la misma cantidad de casos (11), la misma cantidad de eventos (39), igual número de clases de eventos (6), al igual que tipos de eventos (1) y originadores (13), lo que demuestra la similitud existente entre los registros de eventos generados por ambas herramientas.

3.3.1 Análisis de los resultados obtenidos por el componente informático desarrollado

Para demostrar la utilidad de los resultados obtenidos por el componente informático desarrollado, se tomó como ejemplo un registro de eventos generado en el rango de fecha del 1 de enero de 2013 al 31 de enero de 2013 al proceso atender paciente quirúrgico, para aplicarle las técnicas de descubrimiento de modelo *Heuristic Miner* y *Fuzzy Miner*. Esta prueba tiene como objetivo comparar el resultado de las técnicas antes mencionadas, con el modelo de cómo debe ejecutarse el proceso (Ver figura 18), y verificar si se puede descubrir el modelo de proceso a partir de los datos almacenados en el registro de eventos generado.

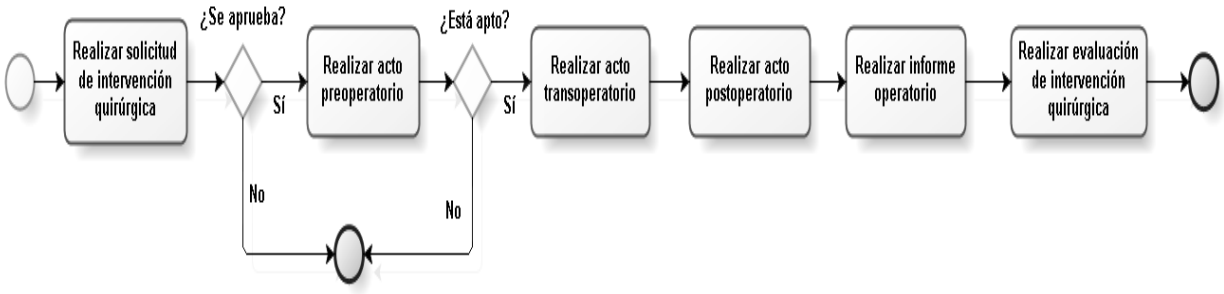

Figura 18. Modelo de cómo debe ejecutarse el proceso atender paciente quirúrgico.
Fuente: elaboración propia.

A continuación se muestra los modelos de descubrimiento generados por las técnicas con los parámetros especificados.

Fuzzy Miner: el algoritmo está basado en medidas de significación y correlación para visualizar el comportamiento en los registros de eventos. Es aconsejable utilizar esta técnica cuando tenga los datos de registros complejos y no estructurados, o cuando se quiere simplificar el modelo de una manera interactiva. Con su aplicación se obtiene un modelo basado en gráficos, capaz de proporcionar una vista de alto nivel de un proceso, con la abstracción de los detalles no deseados (van der Aalst et al, 2007). El resultado de aplicar el plugin Fuzzy Miner al registro de eventos se muestra a continuación:

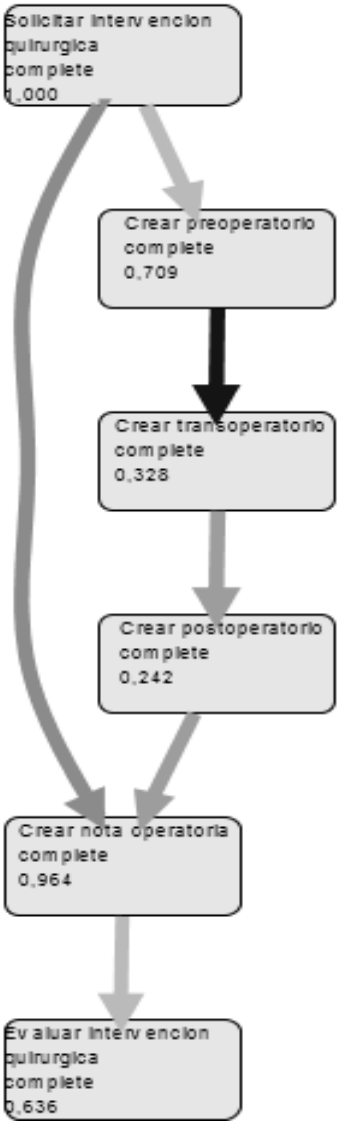

Figura 19. Modelo de descubrimiento de la técnica *Fuzzy Miner.*

Fuente: elaboración propia.

***Heuristic Miner*:** es un plugin de la herramienta ProM basado en el algoritmo del mismo nombre y es un plugin que presenta una alta robustez ante el ruido y las excepciones7 debido a que se basa en la frecuencia de patrones lo que hace posible centrarse en el comportamiento principal en el registro de eventos (Weijters et al, 2009). El resultado de aplicar el plugin Heuristic Miner al registro de eventos se muestra a continuación:

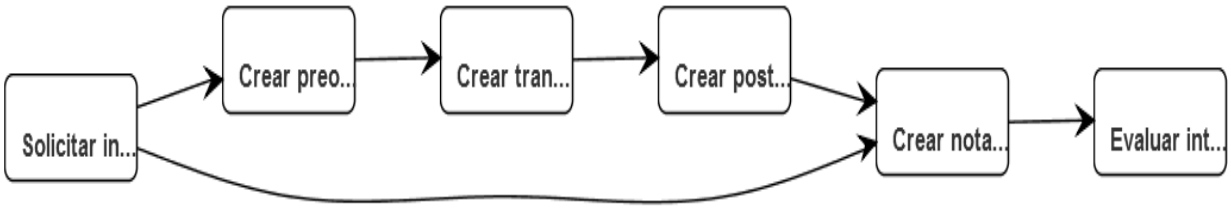

Figura 20. Modelo de descubrimiento de la técnica *Heuristic Miner.*
Fuente: elaboración propia.

A partir de los modelos de procesos descubiertos se puede concluir que; de los datos almacenados en el registro de eventos se logra descubrir el flujo completo de actividades del proceso atender paciente quirúrgico. Además se observa que no siempre se realiza el flujo completo de sus actividades, evidenciándose en los modelos descubiertos por los *plugins* (Figura 19 y Figura 20), cuando pasa de la actividad solicitar intervención quirúrgica a la actividad crear nota operatoria, obviando las actividades entre ellas.

Conclusiones del capítulo

- La ejecución de las pruebas funcionales al componente desarrollado permitió depurar su código fuente, eliminando los errores que atentaban contra su funcionamiento.
- Los resultados arrojados luego de aplicar las pruebas de rendimiento al componente desarrollado demuestran que el mismo ejecuta las tareas de extracción en un tiempo adecuado.
- Se comprueba que el registro de eventos generado por el componente contiene la información necesaria para aplicar técnicas de minería de procesos.

CONCLUSIONES

Con la realización de la presente investigación se ha cumplido con el objetivo general propuesto, así como con las tareas de la investigación, arribándose a las siguientes conclusiones:

- El análisis realizado a las principales soluciones existentes tanto a nivel nacional como internacional, evidenció que presentan limitantes las cuales impiden su utilización en la solución al problema, por lo que resulta necesario la realización de un componente para la extracción de registros de eventos.
- El análisis de los procesos de negocio del módulo Bloque Quirúrgico del Sistema de Información Hospitalaria XAVIA HIS y de su bitácora, demostró que no es utilizada eficientemente en el análisis de los procesos quirúrgicos.
- La asimilación de las herramientas y tecnologías propuestas posibilitó el correcto desarrollo, desde su concepción técnica, del componente informático de extracción de registros de eventos.
- El desarrollo del componente informático, basado en un enfoque por procesos, permitirá al personal del servicio quirúrgico la extracción de registros de eventos, para realizar análisis de los procesos desde el módulo Bloque Quirúrgico del Sistema de Información Hospitalaria XAVIA HIS.
- Las pruebas de software desarrolladas permitieron validar el correcto funcionamiento del componente informático implementado para el análisis de procesos quirúrgicos y demostraron que los resultados obtenidos tras su ejecución son correctos.

RECOMENDACIONES

Con el fin de contribuir a un mayor desarrollo de la presente investigación se realizan las siguientes recomendaciones:

- Aplicar técnicas de minería de procesos que complementen las potencialidades del componente de captura de registro de eventos.
- Aplicar el componente al resto de los módulos del Sistema de Información Hospitalaria XABIA HIS que tengan un enfoque consciente de datos con vistas a propiciar el tratamiento de los eventos de cada módulo.

REFERENCIAS BIBLIOGRÁFICAS

ADÁN, V.G. 2011. Pruebas de caja negra. Globe Testing. [En línea] 2011. [Citado el: 28 de marzo de 2016.]. Disponible en: http://www.globetesting.com/2012/08/pruebas-de-caja-negra/.

Alfaro, F.M. 2011. Instituto Nacional de Estadística e Informática. [En línea] 2011. [Citado el: 28 de marzo de 2016.]. Disponible en: http://www.inei.gob.pe/biblioineipub/bancopub/Inf/Lib5103/Libro.pdf .

Almaguer, Norge. 2011. Documento normativo de Arquitectura de Software. 2011.

Belmonte Fernandez, Oscar. 2005. Introducción al lenguaje de programación Java: Una guía básica. 2005.

Blaya, Inmaculada. 2006. Gestión por Procesos. s.l. : Oficina de Gestión y Control de la Calidad, 2006.

Bravo Carrasco, Juan. 2011. Gestión por procesos. 2011. pág. 204.790.

Buijs, J. 2010. Mapping data sources to xes in a generic way. Maters Thesis. 2010.

BURBECK, S. 2016. Applications Programming in Smalltalk-80(TM), How to use Model-View-Controller (MVC). [En línea] 2016. [Citado el: 02 de 02 de 2016.]. Disponible en: http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html. .

Calleja, M.A. 2014. Estándares de codificación. [En línea] 2014. [Citado el: 22 de mayo de 2016.]. Disponible en: http://www.cisiad.uned.es/carmen/estilo-codificacion.pdf..

Carrasco, Juan Bravo. 2011. Gestión por procesos. 2011.

Cerrano Gómez, Lupita y Ortíz Pimiento, Néstor Raúl. 2012. Caracterización del nivel de desempeño en la gestión por procesos en ips- clínicas y hospitales de bucaramanga y AMB. Medellín : Dyna, 2012. ISSN 0012-7353.

DEACON, J. 2012. Model-View-Controller (MVC) Architecture. 2012.

Domínguez, Emma y Zacca, Eduardo. 2011. Sistema de salud de Cuba. 2011.

Dumas, M. 2005. On the Suitability of BPMN for Bussines Process Modelling. 2005.

Faces, Java Server. 2012. Introducción a JSF. [En línea] 2012. [Citado el: 30 de abril de 2016.]. Disponible en: http://www.desarrolloweb.com/articulos/2380.php .

Fernandez Alarcon, Vincenc. 2006. Desarrollo de sistemas de información: Una metodologia basada en un modelado. 2006.

Fernández García, Ricardo . 2010. Sistema de la gestión de la calidad, ambiente y prevención de riesgos. s.l. : Club universitario Casa del Libro, 2010.

Franky, C. 2010. ConstruColectiva: Guía metodológica para la gestión de proyectos de software basados en metodologías ágiles, utilizando ambientes de desarrollo colaborativo. Caso de estudio: GForge. Undergraduate final project, Pontificia Universidad Javeriana. Available in http://pegasus. javeriana. edu. co/~ CIS0910IS05.

Gherke, N. y Werner, M. 2013. Process mining. Das Wirtschaftswachstum. 2013. págs. 934-943.

Günther, Christian w. 2009. XES Extensible Event Stream standard definition. 2009.

Hernandez Orallo, Enrique. 2012. Lenguaje Unificado de Modelado UML. España : s.n., 2012.

IEEE COMPUTATIONAL INTELLIGENCE SOCIETY. 2016. Draft Standard for XES - eXtensible Event Stream - for achieving interoperability in event logs and event streams. New York : s.n., 2016.

Jacobson, Ivar , Grady , Booch y Rumbaugh, James. 2009. El Proceso Unificado de Desarrollode Software. España : s.n., 2009.

Madeja. 2015. Buenas prácticas en el diseño de pruebas de rendimiento. Marco de Desarrollo de la Junta de Andalucía. [En línea] 2015. [Citado el: 25 de mayo de 2016.]. Disponible en: http://www.juntadeandalucia.es/servicios/madeja/contenido/libro-pautas/75.

Mallar, M.A. 2010. La gestión por procesos: un enfoque de gestión eficiente: Visión de futuro. 2010.

Mining, IEEE Task Force On Process. 2011. Manifiesto sobre Minería de Proceso. [En línea] 2011. [Citado el: 10 de marzo de 2016.]. Disponible en: http://www.win.tue.nl/ieeetfpm/lib/exe/fetch.php?media=shared:pmm .

Mining, IEEE Task Force On Process. 2011. Manifiesto sobre Minería de Proceso. 2011.

Mora, Francisco. 2006. UML: Lenguaje Unificado de Modelado. . Universidad de Alicante-DCCIA, España : s.n., 2006.

Muñoz Cañabate, Antonio. 2007. Sistemas de informacion en las empresas. 2007.

Oquendo, Y.Y. 2011. Desarrollo de un componente de Transmisión de Audio y Video para el Sistema de Teleconsulta. La Habana : s.n., 2011.

Oré B., Alexander . 2009. Pruebas Funcionales. [En línea] 2009. [Citado el: 29 de marzo de 2016.]. Disponible en: http://www.calidadysoftware.com/testing/pruebas_funcionales.php.

Orellana, Arturo, Larrea Armenteros, Osvaldo Ulises y Perez Alfonso, Damian. 2015. Generador de registros de eventos para el análisis de procesos en un sistema de información hospitalaria. Habana : s.n., 2015.

Peguero Alvarez, Hector, Saria, Katia y Torres Peregrino, Carlos Alberto. 2015. Módulo para el registro y transformación de trazas de eventos a formato XES. Habana : s.n., 2015.

Pérez Alfonso, Damián , Yzquierdo Herrera, Raykenler y Llanes Hernández, Alkaid Cruz . 2012. Minería de proceso en el entorno cubano. oportunidades y retos. La Habana, Cuba : s.n., 2012.

Perez Alfonso, Damian, Batista Corella, Aliuska y Aguilera Martinez, Yordany. 2015. Componente para la extracción de registros de eventos en formato XES del sistema ZUN Suite. Habana : s.n., 2015.

Pérez Alfonso, Damián, y otros. 2013. Utilización de técnicas de minería de proceso en el entorno empresarial cubano. Ciudad de La Habana, Cuba : s.n., 2013.

Pérez Alfonso, Damian, Yzquierdo Herrera, R. y Lazo Cortés, M. 2013. Utilización de técnicas de minería de proceso en el entorno empresarial cubano. Ciudad de la Habana : s.n., 2013.

PRESSMAN, R. 2002. Ingeniería del software, un enfoque práctico. 2002.

Pressman, Roger.S. 2005. Ingeniería del Software. 2005.

2005. Pruebas de software. [En línea] 2005. [Citado el: 28 de marzo de 2016.]. Disponible en: http://pruebasdesoftware.com/laspruebasdesoftware.htm..

REDHAT. 2010. Jboss Enterprise Application Platform 4.2. [En línea] 2010. [Citado el: 15 de mayo de 2016.]. Disponible en: https://access.redhat.com/documentation/enUS/JBoss_Enterprise_Application_Platform/4.2/html/Getting_Started_Guide/index.html. .

Rodriguez, Carlos, y otros. 2012. Eventifier:Extracting Process Execution Logs from Operational Databases. Italia : s.n., 2012.

Rondón, L. 2009. JAVA J2EE. JPA - Java Persistence API. [En línea] 3 de abril de 2009. [Citado el: 18 de mayo de 2016.]. Disponible en: http://www.luchorondon.blogspot.com/2009/04/jpa-java-persistence-api.html. .

Santiesteban Peréz, Irina Ivis, y otros. 2011. Desarrollo de funcionalidades que faciliten al docente su preparación y el control del aprendizaje de los estudiantes en la plataforma educativa Zera. 2011.

Scribd. 2012. [En línea] 2012. [Citado el: 21 de mayo de 2016.]. Disponible en: http://es.scribd.com/doc/99836434/Persistencia-Hibernate.

SOMMERVILLE, I. 2005. Ingeniería del software. Madrid : ISBN , 2005. 84-7829-074-5.

Standard, An Introduction to the XES. 2014. Flux Capacitor. [En línea] 2014 [Citado el: 15 de febrero de 2016.]. Disponible en: https://fluxicon.com/blog/2010/09/intro-to-xes/. .

UNESCO. 2005. Hacia las sociedades del conocimiento. s.l. : Ediciones UNESCO, 2005. ISBN 92-3-304000-3.

Vacarezza, Mariela, y otros. 2011. Niveles de atención, de prevención y atención primaria de la salud. 2011.

Valdés Medina, Lilian y Ramayo Mendez, Lian Jaime. 2015. Componente para la extracción de registros de eventos en formato XES del Sistema de Gestión de Ensayos Clínicos (SIGEC). Habana : s.n., 2015.

van der Aalst, W. 2011. Process Mining. Discovery, Conformance and Enhancement of Business Process. London New York: Springer : s.n., 2011.

van der Aalst, W.M.P. y Günther, Christian. 2007. Fuzzy Mining – Adaptive Process Simplification Based on Multi-perspective Metrics. Business Process Management. Alemania : Springer- Berlin Heidelberg, 2007. ISBN 978-3-540-75182-3..

Verbeek, H.M.W, Buijs, B.F y Van der Aals. 2011. XES, XESame, and ProM 6. 2011.

Weijters, A.J. M.M., van der Aalts, W. M.P. y alves de Medeiros, A. K. . 2009. Process Mining with the HeuristicsMiner Algorithm. Eindhoven : Eindhoven University of Technology, 2009. NL-5600 MB.

ANEXOS

Anexo 1. Entrevista

1. ¿En que lenguaje esta programado el componente?
2. ¿Está embebido el código dentro del sistema?
3. ¿Cuáles son los pasos que sigue para extraer registros de eventos?
4. ¿En que formato(s) de registro de eventos extrae la información?
5. ¿Se puede reutilizar el componente en otro sistema?
6. ¿Extrae registro de eventos en sistemas DAIS?

La entrevista favoreció y esclareció a los autores de la presente investigación, sobre el componente de extracción de registro de eventos desarrollado para el módulo Almacén del Sistema de Información Hospitalaria XAVIA HIS abordado en el documento.

Perfil del entrevistado

El entrevistado fue el MSc. Arturo Orellana por ser conocedor y creador del componente de extracción de registro de eventos para el módulo Almacén del Sistema de Información Hospitalaria XAVIA HIS, de forma que nos proporcionara información relevante acerca del tema.