Universidad de las Ciencias Informáticas

Facultad 3

Título: Módulo para el registro y transformación de trazas de eventos a formato XES.

Trabajo de Diploma para optar por el título de
Ingeniero Informático

Autor(es): Carlos Alberto Torres Peregrino
Héctor David Peguero Alvarez

Tutor(es): Damián Pérez Alfonso
René Rodrigo Bauta Camejo

Junio del 2013

1
2
3
4

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Héctor David Peguero Alvarez

_____

Firma del Autor

Carlos Alberto Torres Peregrino

_____

Firma del Autor

Damián Pérez Alfonso

_____

Firma del Tutor

René Rodrigo Bauta Camejo

_____

Firma del Tutor

DATOS DE CONTACTO

Tutor: Ing. Damián Pérez Alfonso

Edad: 26 años.

Ciudadanía: cubano.

Institución: Universidad de las Ciencias Informáticas (UCI).

Título: Ingeniero en Ciencias Informáticas.

Categoría docente: Instructor.

E-mail: dalfonso@uci.cu.

Ingeniero en Ciencias Informáticas, graduado en 2010 en la Universidad de las Ciencias Informáticas. Labora desde hace 3 años en un departamento de desarrollo tecnológico para sistemas de gestión empresarial y sus investigaciones se concentran en el área de minería de proceso. Se desempeña como profesor de inteligencia artificial y programación en la Universidad de Ciencias Informáticas.

Tutor: Ing. René Bauta Camejo

Edad: 28 años.

Ciudadanía: cubano.

Institución: Universidad de las Ciencias Informáticas (UCI).

Título: Ingeniero en Ciencias Informáticas.

Categoría docente: Instructor.

E-mail: rrbauta@uci.cu.

Ingeniero en Ciencias Informáticas, graduado en 2009 en la Universidad de las Ciencias Informáticas. Instructor. Cuatro años de experiencia en el desarrollo de software. Cuatro años de graduado.

Agradecimientos

[Insertar agradecimientos (opcional)]

Dedicatoria

De Héctor:

A mamá por ser adivina y estar ahí siempre para mí.

A mamí por quererme tanto y apoyarme siempre.

A abuelo porque siempre me está y estará cuidando.

A ñañí por obligarme a esforzarme.

Resumen

La Minería de Proceso es una disciplina de investigación que proporciona técnicas para descubrir, monitorear y mejorar los procesos en una variedad de dominios de aplicación. Las organizaciones dedican mucho tiempo y esfuerzo a analizar sus procesos en aras de mejorar su desempeño organizacional y operacional. Las trazas que registra el marco de trabajo Sauxe son generadas por las aplicaciones adscritas al mismo. Estas trazas carecen de nociones de procesos impidiendo algunas actividades como la realización de auditorías de seguridad y la creación de modelos de procesos utilizando técnicas de Minería de Proceso. Esta investigación se enfoca en la creación de módulos para el registro y transformación de trazas de eventos a formato XES[1] (por sus siglas en inglés e**X**tensible **E**vent **S**tream).

Con esta solución se pretende lograr la generación de trazas en Sauxe de alta calidad para que de esta forma sean aptas para su uso en estudios de Minería de Proceso y su exportación al formato XES.

PALABRAS CLAVE

Minería de Proceso, nociones de proceso, proceso, traza, XES.

[1] **XES:** Estándar XML para registros de eventos en Minería de Proceso.

Tabla de contenido

Índice de figuras

Índice de Tablas

INTRODUCCIÓN

En el mundo empresarial actual, altamente competitivo y en rápida evolución, la mera supervivencia depende de la capacidad de una empresa para reconocer cambios, desafíos del mercado y responder a ellos con rapidez y precisión. La Inteligencia de negocios aumenta considerablemente la agilidad empresarial, porque ofrece a grupos estratégicos dentro y fuera de la organización un punto de vista específico de los datos corporativos que se requieren para obtener el éxito (Microsoft 2004).

Las organizaciones y empresas poseen y generan diariamente un gran cúmulo de datos imposibles de analizar a simple vista. Mucho esfuerzo es dedicado por las mismas en el análisis y la mejora de procesos para optimizar su desempeño organizacional y operacional. Analizar procesos consume mucho tiempo, involucra a muchas personas y es caro (Van Der Heijden, 2012).

El análisis de forma manual del desempeño de los procesos depende en gran medida de la existencia de modelos de procesos de alta calidad. Cuando los modelos de procesos y la realidad tienen poco en común, los análisis basados en modelos no tienen mucho sentido. Frecuentemente hay una falta de alineación entre los modelos hechos manualmente y la realidad. La Minería de Proceso se centra en resolver estos problemas realizando una conexión directa entre los modelos y los datos de eventos de los procesos (W. M. P. Van der Aalst 2011). El área de investigación de la Minería de Proceso proporciona técnicas para descubrir, monitorear y mejorar los procesos en una variedad de dominios de aplicación (Van Der Heijden, 2012).

El punto de partida para cualquier actividad de Minería de Proceso son los eventos registrados. La calidad de un resultado de Minería de Proceso en gran medida depende de la entrada (IEEE Task Force on Process Mining 2011). Para dar formato a un log de registro de eventos el cual se utiliza como fuente de datos en los análisis de Minería de Proceso, se definieron en un principio estándares como el Mining eXtensible Markup Language **(**MXML). Sin embargo, un nuevo formato surge en el 2009 llamado eXtensible Event Stream (XES) desarrollado por la IEEE Task Force on Process Mining (equipo de trabajo de la IEEE para la Minería de Proceso) para servir como un estándar en el registro de datos de eventos. Este formato de registro de eventos soluciona algunos problemas encontrados durante el uso de MXML para el registro de ejecuciones de procesos en tiempo real y permite una mayor flexibilidad (J.C.A.M. Buijs, 2010).

Las trazas registradas (registro de eventos) deben tener una cantidad de información mínima para que la aplicación de técnicas de Minería de Proceso sea viable y se obtenga un resultado útil. Los sistemas de información conscientes de los datos (DAIS por sus siglas en inglés) no tienen nociones de procesos en sus trazas lo que conlleva a que las mismas contengan información insuficiente y de poca calidad.

Esto hace compleja o imposible la obtención de registros de eventos completos. Por otra parte, los sistemas de información conscientes de los procesos (PAIS por sus siglas en inglés) generan trazas de gran calidad y completitud (IEEE Task Force on Process Mining 2011). La obtención de registros de eventos desde DAIS ha sido abordada por Joos Buijs con el desarrollo de la herramienta XESAME y por Carlos Rodríguez y colegas con la herramienta Eventifier.

Sauxe es una base tecnológica desarrollada en Cuba, específicamente en la Universidad de Ciencias Informáticas en el centro CEIGE (Centro de informatización de la gestión de entidades) para el desarrollo de aplicaciones web de gestión. Sauxe es utilizado en varios proyectos de la UCI y entidades desarrolladoras de software del país. El principal sistema que utiliza este marco de trabajo es Cedrux, solución genérica para la gestión de entidades. Las trazas que registra Sauxe son generadas por las aplicaciones que soporta. Los tipos de trazas definidos en Sauxe son: trazas de URL, autenticación, cierre de sesión, acción, integración, rendimiento, datos y excepción de integración. Las aplicaciones adscritas a Sauxe están orientadas a datos, provocando que sus trazas no contengan nociones de procesos. Sauxe posee un motor de flujo de trabajo con el cual se pueden ejecutar algunos procesos. Éste actualmente no registra ningún tipo de trazas y no es aplicable a todas las soluciones.

El marco de trabajo Sauxe posee un módulo para la gestión de trazas. Este módulo registra varios tipos de trazas pero ninguna de estas contiene la información suficiente para realizársele análisis utilizando técnicas de Minería de Proceso. Adicionalmente, los datos almacenados en la base de datos operacional de Sauxe poseen una alta variabilidad. Esto trae consigo que las trazas en ocasiones referencien datos inexistentes o diferentes de cuando se originó la traza.

Por lo antes expuesto se plantea el siguiente **Problema a resolver:**
¿Cómo garantizar la transformación de las trazas registradas por el marco de trabajo Sauxe para realizar análisis de los procesos ejecutados utilizando técnicas de Minería de Proceso?

Se delimita un **Objeto de estudio:** Minería de Proceso, específicamente en el **campo de acción** del: Registro de trazas para la Minería de Proceso.

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

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

- Confeccionar el marco teórico conceptual de la investigación a partir de una búsqueda y revisión bibliográfica para definir los datos a registrar a partir de la definición de XES, sus extensiones y los posibles análisis de Minería de Proceso a ejecutar.
- Realizar el análisis y diseño del componente para la transformación de las trazas registradas por el marco de trabajo Sauxe.
- Desarrollar los módulos para la transformación de trazas de proceso relacionados a los datos de los procesos que no se informatizan utilizando el motor de flujo de trabajo.
- Realizar la exportación de las trazas de proceso a formato XES.
- Validar la solución propuesta mediante pruebas de caja blanca y caja negra.

Idea a defender

Si se desarrolla un módulo que permita la transformación de las trazas registradas por el marco de trabajo Sauxe se podrán realizar análisis de los procesos ejecutados utilizando técnicas de Minería de Proceso.

Posibles resultados:

- Definición de trazas a registrar a partir de la definición de XES, sus extensiones y posibles análisis de Minería de Proceso a ejecutar.
- Procedimiento para el registro de trazas de proceso en sistemas orientados a datos, ajustado al marco de trabajo Sauxe.
- Módulo de registro y transformación de las trazas de sistemas desarrollados con el marco de trabajo Sauxe, hacia el formato XES, para el análisis de los procesos ejecutados mediante la aplicación de técnicas de Minería de Proceso.

El presente trabajo está estructurado en **tres** capítulos:

En el capítulo **primero**, “Fundamentación teórica”, se hace referencia a los principales conceptos teóricos existentes dentro del objeto de estudio, se describe el entorno en el que se desarrolla la problemática, se analizan las posibles soluciones, características y viabilidad. Se proponen la utilización de una serie de herramientas y tecnologías definidas por el centro a emplear en el desarrollo de la solución.

En el capítulo **segundo**, “Propuesta de solución”, se describen los requisitos funcionales y no funcionales de la solución desarrollada, su integración con componentes de Sauxe, el patrón arquitectónico utilizado y los patrones de diseño usados en la implementación y módulos de Sauxe relevantes para la implementación.

En el capítulo **tercero**, “Implementación y pruebas”, se valida el diseño mediante la aplicación de las métricas Tamaño operacional de clase y Relaciones entre clases. Se describen los estándares de codificación utilizados en la nomenclatura del código y se valida la propuesta de solución mediante pruebas de caja blanca y caja negra. Adicionalmente, se detalla la validación de la Idea a defender de la investigación mediante la aplicación a un caso de estudio.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

En este capítulo se hace un estudio de los conceptos fundamentales relacionados con la extracción de trazas de proceso en sistemas de información. Adicionalmente se detallan los aportes teóricos en el tema de los autores que han abordado este tema. Se analizan las herramientas existentes para la extracción de trazas de proceso generadas por sistemas de información conscientes de los datos (DAIS por sus siglas en inglés). Finalmente, describen las herramientas y tecnologías utilizadas para el desarrollo de la solución.

1.1. Conceptos Fundamentales

Un **Sistema de Información** es un tipo particular de sistema de trabajo que usa tecnología de la información para captar, transmitir, almacenar, recuperar, manipular o exhibir información soportando a uno o más sistemas de trabajo distintos. A su vez, **Sistema de Trabajo** es un sistema en el cual los participantes humanos realizan un proceso de negocio utilizando información, tecnología y otros recursos para producir productos para clientes internos. Los sistemas de información se dividen en dos ramas, sistemas de información conscientes de los datos y sistemas de información conscientes de los procesos. Los sistemas de información conscientes de los procesos (PAIS por sus siglas en inglés) son softwares que gestionan y ejecutan procesos operacionales involucrando personas, aplicaciones y/o fuentes de información basados en modelos de procesos (Dumas, Wil M. van der Aalst, Hofstede, 2005).

Se considera como **proceso** a un conjunto de actividades mutuamente relacionadas o que interactúan, las cuales transforman elementos de entrada en resultados (ISO 2000). Una **Actividad** es un paso bien definido en el proceso (IEEE Task Force on Process Mining 2011). Una actividad está compuesta por tareas y se entiende como **tarea** a una actividad atómica que está incluida dentro de un proceso. Por su parte un **Sub-Proceso** es un proceso que está incluido en otro proceso (OMG 2006).

Una **Instancia de un Proceso,** en el ámbito de la Minería de Proceso, 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). El **registro de eventos** es la colección de eventos utilizados como entrada para la Minería de Proceso. Los eventos no necesitan ser almacenados en un archivo de registro por separado (por ejemplo, los eventos pueden estar dispersos en diferentes tablas de bases de datos).

1.1.1. Minería de Proceso

Se entiende por **Minería de Proceso** a la disciplina de investigación que permite descubrir, monitorear y mejorar los procesos reales a través de la extracción de conocimiento de los registros de eventos ampliamente disponibles en los actuales sistemas de información (IEEE Task Force on Process Mining, 2011).

El **Descubrimiento de procesos**, la **Verificación de conformidad** y el **Mejoramiento de modelos,** son los tres tipos básicos de Minería de Proceso. El descubrimiento de procesos permite obtener un modelo de proceso a partir de un registro de eventos. A su vez la Verificación de Conformidad analiza si la realidad (según consta en un registro de eventos) se ajusta al modelo y viceversa. Su objetivo es detectar las discrepancias y medir su gravedad. Por otra parte el Mejoramiento de Modelos permite extender o mejorar a partir de la información extraída de un registro de eventos. Por ejemplo, se pueden identificar cuellos de botella reproduciendo un registro de eventos en un modelo de proceso, mientras se examinan las marcas de tiempo (IEEE Task Force on Process Mining, 2011).

A continuación se muestran algunos de los casos de éxitos en el tema de Minería de Proceso más recientes:

- ✓ **Desempeño de la Minería de Proceso desde registro de eventos**

En este trabajo, se analiza un registro de eventos de un proceso de la vida real, tomado de una entidad financiera holandesa, utilizando técnicas de Minería de Proceso. En particular, se utilizó la técnica de alineación para obtener información sobre el flujo de control y el rendimiento de la ejecución del proceso. Se demuestra que las alineaciones entre los registros de eventos y modelos de procesos descubiertos con algoritmos de descubrimiento de procesos, revelan conocimientos sobre las desviaciones que se producen con frecuencia y cómo tales ideas pueden ser explotadas para reparar los modelos de procesos originales, reflejando mejor la realidad. Además, se demuestra que los alineamientos pueden aprovecharse más para obtener información sobre el rendimiento. Todos los análisis de este trabajo se llevan a cabo mediante plug-ins de la aplicación de código abierto para la Minería de Proceso ProM (ADRIANSYAH, A. y BUIJS, J., 2012).

- ✓ **La Minería de Proceso aplicada al reto de la Inteligencia de Procesos de Negocio**

En este artículo a partir de un registro de eventos de la vida real, tomado de una entidad financiera holandesa, se analiza dicho registro usando técnicas de Minería de Proceso. El registro contiene eventos relacionados con las solicitudes de crédito/sobregiro de los clientes. Se propone una descomposición jerárquica del registro en subconjuntos homogéneos de casos basados en características tales como la

decisión final, la oferta y la sospecha de fraude. Estos subconjuntos se utilizan para descubrir ideas interesantes. El registro de eventos en su totalidad y en los subconjuntos homogéneos, se analizan utilizando diversas técnicas de Minería de Proceso. En concreto, se analiza el registro de eventos (a) en la perspectiva de los recursos y la influencia de los recursos en la ejecución / tiempos de respuesta de las actividades, (b) en el punto de vista de flujo de control y (c) para el diagnóstico de procesos. Un plug-in especializado para ProM que fue creado para este desafío, permite un análisis exhaustivo desde la perspectiva de los recursos. Para el análisis de control de flujo y diagnóstico del proceso, se usaron plug-ins pre-existentes de ProM. Se muestra una evaluación donde la mezcla de técnicas es capaz de descubrir muchos hallazgos interesantes y podría ser utilizado para mejorar el proceso de manejo de solicitud de préstamo / sobregiro subyacente (Bose, R. P. J. C. y Van Der Aalst, W. M. P., 2012).

- ✓ **Minería de Proceso basado en reglas**

La abundancia de datos de eventos disponibles, procedentes de los sistemas de información conscientes de procesos, crea oportunidades para las aplicaciones de gestión de riesgo empresarial en la intersección de los campos de investigación: Negocio y administración, Inteligencia artificial y Representación del conocimiento empresarial. Este documento propone un enfoque de Minería de Proceso basado en reglas para hacer frente a la incertidumbre y el riesgo. La aplicabilidad del enfoque se demuestra usando el proceso de actualización y depuración de un proveedor de servicios de la seguridad social (Caron, Filip, Vanthienen, Jan y Baesens, Bart, 2012).

1.1.2. Registros de eventos o trazas de proceso

Para formalizar la estructura de los registros de eventos a utilizar en la Minería de Proceso se han definido dos estándares: MXML y XES. **MXML** es un formato basado en XML para el intercambio de registros de eventos. MXML fue el primer estándar que surgió en el 2003 y fue adoptado por la herramienta de Minería de Proceso ProM. MXML establece una notación estándar para almacenar fechas, recursos y tipos de transacciones. XES en el 2010 reemplaza a MXML como el nuevo formato para Minería de Proceso independiente de la herramienta (IEEE Task Force on Process Mining 2011). Es un estándar basado en experiencias prácticas de MXML, menos restrictivo y verdaderamente extensible. Su principal propósito es ofrecer un formato de intercambio de registros de eventos entre herramientas y dominios de aplicaciones (Christian W. Günther 2009).

Para asegurar un análisis de Minería de Proceso exitoso, además del formato de almacenamiento del registro de eventos se debe garantizar su **calidad**. La misma se define a partir de tres aspectos fundamentales: confiabilidad, completitud y seguridad. La **confiabilidad** consiste en que los eventos deben ser confiables, es decir, debería ser seguro asumir que los eventos registrados realmente

ocurrieron y que los atributos de los eventos son correctos. La **completitud** se relaciona a que los registros de eventos deberían ser completos, dado un determinado contexto, no puede faltar ningún evento. Además, cualquier evento registrado debe tener una semántica bien definida. Por otra parte si los datos de eventos son **seguros** si se tienen en cuenta consideraciones de privacidad y seguridad al registrar los eventos (IEEE Task Force on Process Mining 2011).

Niveles de madurez

La combinación de los tres aspectos de calidad se refleja en los **niveles de madurez** definidos por la IEEE para los registros de eventos:

Nivel-1: Los eventos se registran manualmente, los eventos registrados podrían no corresponder a la realidad y podrían faltar eventos.

Nivel-2: Los eventos se registran automáticamente, no se sigue un enfoque sistemático para decidir qué eventos se registran, podrían faltar eventos o estos podrían no registrarse correctamente.

Nivel-3: Los eventos se registran automáticamente, pero no se sigue un enfoque sistemático para registrar los eventos. Hay un nivel de garantía de que los eventos registrados calzan con la realidad (el registro de eventos es confiable pero no necesariamente completo).

Nivel-4: Los eventos se registran automáticamente y de manera sistemática y confiable, a diferencia del nivel 3 da soporte de manera explícita a nociones tales como instancia de proceso (caso) y actividad.

Nivel-5: El registro de eventos es de excelente calidad (confiable y completo) y los eventos están bien definidos. Los eventos se registran de manera automática, sistemática y segura. Se toman en cuenta adecuadamente consideraciones acerca de la privacidad y la seguridad (IEEE Task Force on Process Mining 2011).

1.2. Elementos Teóricos de la extracción de trazas de proceso desde DAIS

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 Carlos, 2012).

Un registro de eventos puede ser visto como la secuencia de eventos $E = <e_1, e_2, e_3, \ldots, e_m>$ donde:

$e_i =<$ id, tname, pname, piid, ts, pl$>$

e_i 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 e_i 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 Carlos, 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 Carlos, 2012).

1.2.1. Patrones de identificación de eventos

Estos patrones ayudan en la identificación de eventos de la base de datos operacional. En estos patrones se asume que la existencia de una fila en una relación R indica la presencia de un evento (Rodríguez Carlos, 2012) Estos eventos se expresan con la función:

Identificar (R, pname, tname) $\rightarrow e^0 =<$ id, pname, −, tname, −, t>

Donde pname y tname son definidos por el experto del dominio y t es la tupla en R que originó e^0. En concreto se depende de los siguientes 3 patrones para la identificación de eventos:

Patrón de un evento y una fila, (Figura 1(A))

En este patrón cada fila en una relación R indica la existencia de un evento

Patrón de múltiples eventos y una fila (Figura 1(B))

Una tupla A en R puede evidenciar la existencia de más de un evento. Esto se manifiesta cuando diferentes valores de atributos A_i de R indican diferentes eventos potenciales.

Patrón de un evento y múltiples filas (Figura 1(C))

Múltiples filas en una relación R indican la presencia de solo un evento

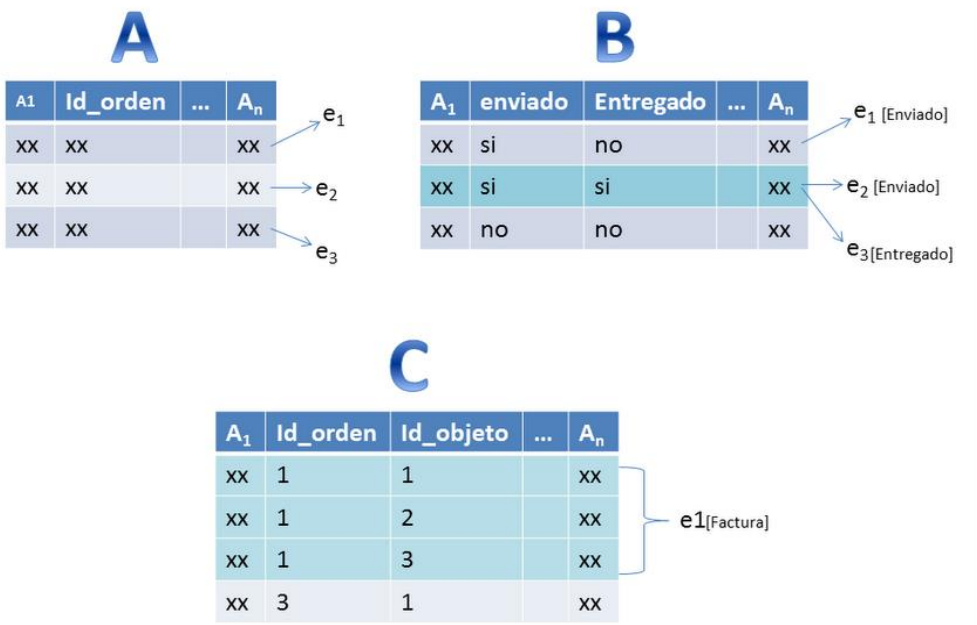


Figura 1. Tipos de patrones de identificación de eventos: (A) Patrón de un evento y una fila, (B) Patrón de múltiples eventos y una fila y (C) Patrón de un evento y múltiples filas.

1.2.2. Patrón de ordenamiento de eventos

Este patrón tiene como objetivo derivar el orden de los eventos a partir de fechas almacenadas en la base de datos operacional y se representa:

Orden (e^0) → e^1 = <id, pname, −, tname, ts, t>

Donde e^1 es el resultado de asignar un valor de fecha a ts. Si solo se encuentra una fecha es usada directamente. Si existe más de una posible fecha en pl, el experto del dominio decide cuál es la que representa mejor el tiempo de ejecución de la tarea (Rodríguez Carlos, 2012).

1.2.3. Patrón de asociación de datos

Este patrón tiene como objetivo seleccionar qué datos guardar en pl. En los patrones anteriores simplemente se seleccionaba la fila completa t como “carga útil” del evento. Mientras que en este patrón se selecciona cuáles de los atributos de t son realmente relevantes. Se asume que todos los datos necesarios ya están dentro de t, no se necesita consultar ninguna tabla adicionar de la OD para llenar pl con datos relevantes. Esto se debe a que en el paso de identificación de eventos las tablas necesarias fueron unidas y t contiene todos los datos potencialmente relevantes. El patrón de asociación de datos se representa:

Obtener_datos (e^1) → e^2 = <id, pname, −, tname, ts, pl>

Donde e^1 es como se definió anteriormente y pl es la nueva "carga útil" creado mediante la proyección de atributos de t. En ausencia de conocimientos sobre la OD por el experto del dominio, la heurística que se aplica es copiar todos los atributos de t para pl, excepto fechas y atributos autoincrementables, los cuales no se pueden usar para la correlación. El experto del dominio puede por supuesto seleccionar cuáles atributos incluir y cuáles excluir (Rodríguez Carlos, 2012).

1.2.4. Patrones de correlación de eventos

El objetivo de la correlación de eventos es agrupar los eventos en instancias de procesos los cuales son la base para el descubrimiento de procesos. Como se explicó anteriormente, se asume que después de asociar la "carga útil" final a los eventos, toda la información que se necesita para correlacionar está presente en el "carga útil" pl de los eventos. En la práctica, correlacionar eventos en las trazas significa descubrir la función matemática sobre los atributos de pl que dice si un evento pertenece a una instancia de un proceso, identificado por el piid de salida de la función. Este paso se representa de la siguiente manera:

Correlación (e^2) → e = <id, pname, piid, tname, ts, pl>

Donde e^2 es como está definido anteriormente y **e** es la versión final del evento descubierto de la OD con el atributo piid llenado con un identificador de instancia de proceso acertado al cual el evento pertenece (Rodríguez Carlos, 2012).

La definición de la obtención de un registro de eventos parte de la selección de la instancia de proceso, ya que esta determina el ámbito del proceso a investigar. Después de la selección de la instancia del proceso a la cual los eventos deben estar relacionados, la selección y especificación de estos eventos puede comenzar. La selección de los eventos a incluir va a depender del enfoque del proyecto de minería a realizar.

Al ser los eventos grabaciones atómicas de actividades ejecutadas, estos no tienen duración. Para almacenar los diferentes estados que una actividad puede tener, cada evento debe ser de un cierto tipo. Los tipos de eventos más usados son el "iniciado" y el "completado" para indicar el inicio y la completitud de una actividad respectivamente. Es posible almacenar eventos de un solo tipo el cual generalmente es "completado". Para ciertos tipos de análisis es imperativo adicionar el tipo "iniciado" como cuando es necesario medir el tiempo de procesamiento. El tipo de evento es esencial para mediciones de desempeño. Los tipos de eventos "iniciado" y "completado" se usan para distinguir entre el tiempo de espera y tiempo de procesamiento. El **tiempo de espera** o tiempo de cola, es el tiempo entre el fin de

una actividad y el inicio de la siguiente. El **tiempo de procesamiento** es el tiempo requerido para desempeñar una actividad. Por lo anteriormente planteado adicionar estos tipos de eventos puede ser muy útil en el análisis de un registro de eventos y permitir nuevos métodos de análisis.

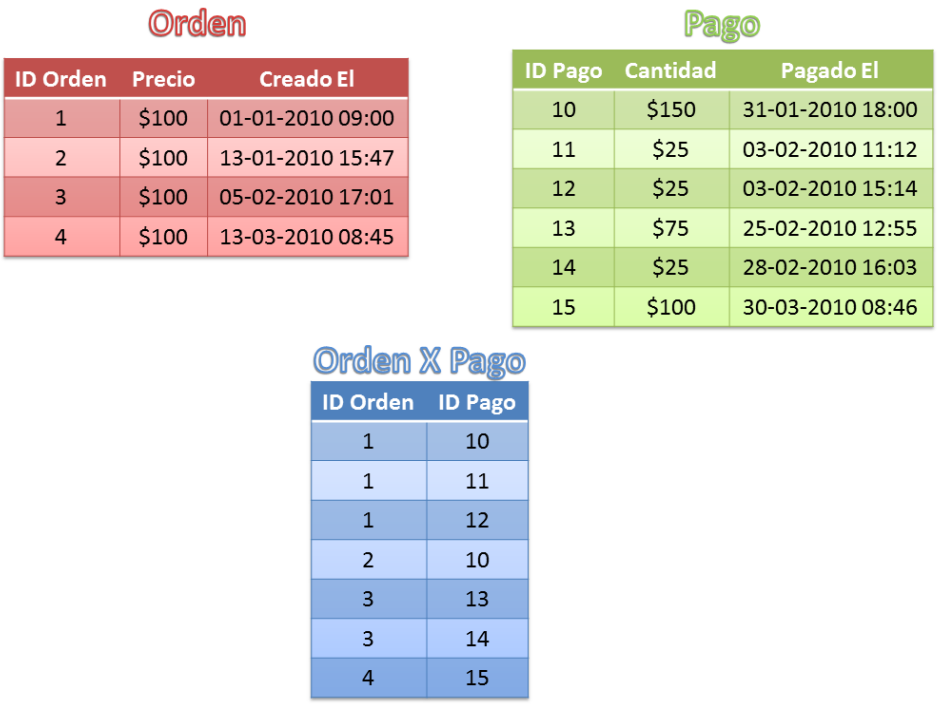
Otro de los aspectos en la selección de eventos es la elección de definir cada evento de manera separada, reutilizar algún tipo de registro de eventos o ambas. La última opción es común en muchos sistemas pues estos generalmente tienen una especie de registro de eventos existente. Frecuentemente el formato de dicho registro no es adecuado para Minería de Proceso. Un ejemplo de esto puede ser que algunos eventos no estén relacionados a una única instancia de proceso. También se puede dar el caso de que falte cierta información. Por otra parte estos registros proveen información importante sobre qué actividades son ejecutadas. Es posible extraer directamente una parte de este registro, enriquecerlo con la información faltante y convertirlo a un registro de eventos más adecuado para la Minería de Proceso. Otra opción es examinar el registro para tener una idea de qué actividades se registran en el sistema. Al haber hecho esto se puede especificar cada actividad de manera separada. La desventaja de especificar cada actividad por separado, es que posiblemente que no todos los eventos del sistema sean incluidos. Algunos eventos pueden ser accidentalmente pasados por alto o simplemente ser desconocidos por eso pueden no aparecer en el registro de eventos. Las fechas en la fuente de datos proveen buenas pistas para el almacenamiento de eventos. La ventaja de especificar cada evento por separado es que el nivel de detalle puede mantenerse consistente. En la realidad la definición de eventos es impulsada por la información presente en la fuente de datos (J.C.A.M. Buijs, 2010).

1.2.5. Problema de Convergencia y Divergencia

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

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

La **Figura 2** provee un ejemplo demostrativo de Convergencia y Divergencia:



Orden

ID Orden	Precio	Creado El
1	$100	01-01-2010 09:00
2	$100	13-01-2010 15:47
3	$100	05-02-2010 17:01
4	$100	13-03-2010 08:45

Pago

ID Pago	Cantidad	Pagado El
10	$150	31-01-2010 18:00
11	$25	03-02-2010 11:12
12	$25	03-02-2010 15:14
13	$75	25-02-2010 12:55
14	$25	28-02-2010 16:03
15	$100	30-03-2010 08:46

Orden X Pago

ID Orden	ID Pago
1	10
1	11
1	12
2	10
3	13
3	14
4	15

Figura 2. Convergencia y Divergencia.

Las órdenes(o pedidos) están en la tabla Orden. Cada orden tiene un identificador único almacenado en la columna **ID Orden**. Además, cada orden tiene un precio que necesita ser pagado almacenado en la columna precio. Los pagos se almacenan en la tabla **Pago**. Cada pago tiene un identificador único almacenado en la columna **ID Pago**. La cantidad pagada en cada pago se almacena en la columna **Cantidad**. Para relacionar las órdenes y los pagos existe la tabla **Orden X Pago**. No obstante, un pago no necesariamente tiene que pagar la orden completamente, pudiera pagar solamente una parte. Por ejemplo los pagos 13 y 14 pagan por la orden número 3 como se puede ver en la tabla **Orden X Pago**. Para complicar aún más las cosas un pago puede ser parte de dos órdenes. En el ejemplo el pago 10 paga por las órdenes 1 y 2. En este caso particular se puede deducir que 100 de los 150 pesos pagaron la orden 2 mientras que los otros 50 pesos pagaron la orden 1. En general no se puede decidir cómo el pago se divide entre las órdenes correspondientes.

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

Convergencia

El problema generado debido a que un evento se relaciona a múltiples instancias de procesos se llama convergencia. La **convergencia** existe cuando una actividad se ejecuta en múltiples instancias de proceso a la vez. Esto se puede reconocer cuando hay una relación de 1: N desde un evento hacia la

instancia de proceso. Al ocurrir convergencia puede aparecer en el registro de eventos que un usuario estaba pagando dos órdenes a la vez (J.C.A.M. Buijs, 2010).

Divergencia

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

Solución a Convergencia y Divergencia

Para resolver el problema de convergencia y/o divergencia se puede cambiar la representación de la instancia de proceso en la conversión. En el ejemplo anterior se pudiera cambiar el identificador de instancia de proceso desde el **ID Orden** de la tabla **Orden X Pago** hacia la fila completa (ID Orden e ID Pago) de la tabla **Orden X Pago**. Esto resolvería el problema de divergencia pues solo un pago estaría relacionado a cada “ID Orden X ID Pago”. El problema de la convergencia todavía permanecería ya que el pago 10 siempre pagaría al menos 2 “ID Orden X ID Pago”, uno para la orden 1 y uno para la orden 2. Por esto la convergencia y la divergencia no siempre pueden ser resueltas.

La convergencia y la divergencia siempre deben ser tenidas en cuenta durante la fase de Minería de Proceso. Cabe notar que la convergencia puede ser reconocida por algoritmos de Minería de Proceso si se utiliza el atributo **instance** de la extensión **concept** de **XES**. Mediante la comparación de eventos usando el identificador de instancia de actividad estos eventos pueden ser reconocidos y tratados de manera diferente. Desafortunadamente ninguno de los algoritmos actuales utiliza esta facilidad (J.C.A.M. Buijs, 2010).

1.3. Atributos del registro de eventos y extensiones de XES

El registro de eventos contiene un registro de elementos como su raíz el cual contiene todas las trazas. Este registro de elementos también puede contener atributos. Como el registro de elementos solo se crea una vez, el impacto de incluir muchos atributos en el registro es mínimo (J.C.A.M. Buijs, 2010). Por

otra parte es de importancia incluir información relevante describiendo el contenido del registro de eventos y su origen. Los siguientes atributos deben ser tomados en consideración para su adición en el registro de elementos:

Nombre de Proceso El nombre del proceso al cual el registro le graba su ejecución.

Fuente de Datos Una descripción del Sistema de Información del cual se extrae el registro de eventos.

Organización Fuente El nombre de la organización que provee los datos.

Descripción Una breve descripción del contenido del registro de eventos.

Versión Un identificador para diferenciar versiones de registros de eventos.

Autor Nombre y detalles de contacto del que definió la conversión.

Proyecto de Minería de Proceso Una referencia del Proyecto de Minería de Proceso o el propósito del registro de eventos.

Extensiones de XES

Para las trazas y eventos pueden ser definidos dos tipos de atributos. El primer tipo son esos atributos que pueden ser especificados para trazas y eventos. Algunos de estos atributos son requeridos por algoritmos de Minería de Proceso. La mayoría de los atributos de este tipo son definidos por las cinco extensiones estándar de XES. El segundo tipo de atributo son los atributos de datos. Los atributos de datos almacenan información adicional acerca del objeto al cual la traza se refiere o la actividad ejecutada. Algunos algoritmos requieren atributos específicos en el registro de eventos. Por otra parte no todos los atributos definidos por las extensiones estándar deben estar siempre presentes en el registro. Algunas veces la información requerida ni se encuentra en la fuente de datos.

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.

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. Es importante lograr recoger la mayor cantidad de información acerca del momento de la ejecución del evento. Preferiblemente al nivel de segundos y en algunos casos de milisegundos. Especialmente en

una base de datos operacional siendo consultada concurrentemente, muchos eventos pueden ser ejecutados dentro del mismo segundo, por esto es la necesidad de la precisión de milisegundos. Esto es necesario pues es muy importante que los eventos tengan un único instante de tiempo dentro de la traza. Aunque la mayoría de los algoritmos pueden manejar instantes de tiempos iguales, los resultados si se brinda más precisión, mejoran drásticamente.

Como se discutió anteriormente, los eventos son grabaciones atómicas y por eso no tienen duración. Como las actividades sí tienen duración, los eventos pueden ser de diferentes tipos, cada uno de estos grabando un diferente estado de una actividad. Los tipos de eventos son proporcionados por la extensión "lifecycle". Dos de los tipos de eventos más comunes son el "iniciado" y "completado". Estos tipos de eventos indican el inicio y la completitud respectivamente de una actividad. Contando con estos dos tipos de eventos es posible estimar tiempo de procesamiento y tiempo de espera en un análisis de desempeño. En la mayoría de los casos con usar el tipo "iniciado" y "completado" es suficiente 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 Proceso 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. 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 ejecuto 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.

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 un proceso. Las cinco extensiones discutidas anteriormente por otra parte, capturan los atributos más frecuentemente usados. 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 Proceso se pueda aplicar exitosamente.

A continuación se muestra un ejemplo sencillo de un registro de eventos en formato XES.

```
− <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 3. Registro de eventos en formato XES.

1.4. Herramientas para la extracción de trazas de los sistemas de información

A pesar de descuidar la naturaleza basada en eventos de los datos, si se pueden relacionar los atributos de los eventos a campos en la base de datos operacional, siempre se podrán realizar estudios utilizando técnicas de minería de proceso. Las herramientas estudiadas para la extracción de trazas de proceso de los DAIS fueron XESame y Eventifier. Estas herramientas se describen a continuación:

1.4.1. XES Mapper (XESame)

XESame es un mapeador de XES que provee una manera genérica para extraer un registro de eventos desde una fuente de datos. Una de las potencialidades de XESame es su facilidad de uso ya que no requiere habilidades de programación. XESame le permite a un experto en el dominio especificar cómo

el registro de eventos debe ser extraído de la base de datos operacional de un sistema de información determinado y convertido a XES o MXML. El sistema brinda varias interfaces para la definición de la conversión en las cuales se pueden definir los tipos de gestores de bases de datos a usar como son PostgreSQL, 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, se debería ser capaz de analizar este registro de todas las maneras posibles en el ámbito de Minería de Proceso (J.C.A.M. Buijs, 2010).

Usando XESame se pueden especificar cada evento y cada actividad de manera independiente. El registro, sus trazas y eventos poseen atributos para almacenar información. Los atributos pueden contener otros atributos anidados para almacenar información más detallada. Además de seleccionar cuáles de los atributos han de ser incluidos en el registro de eventos, también se pueden seleccionar qué trazas o eventos se han de incluir. Esto se puede lograr limitando el número de trazas a extraer y/o el número de eventos. XESame posee ciertas limitaciones, pues al estar desarrollada en Java para su conexión a base de datos requiere un driver ODBC. El driver ODBC tiene conflictos con el uso de funciones SQL en la definición de la conversión. Otra limitación es que Microsoft no ha desarrollado una versión de los drivers ODBC para Windows en 64-bit lo que trae consigo que solo se pueda utilizar la aplicación en modo 32-bit acarreando el problema de que se podrán utilizar un máximo de solo 4gb de memoria RAM lo que podría en un caso dado imposibilitar la conversión. XESame no detecta la convergencia ni la divergencia entre las trazas y los eventos lo que pudiera prevenir al usuario de cometer errores en la fase de análisis (J.C.A.M. Buijs, 2010). La principal desventaja de XESame es que solo es aprovechable si existe alguna evidencia de la ejecución de un proceso.

Esta solución no es aplicable ya que Sauxe es un marco de trabajo para aplicaciones de gestión. Por las características de estas aplicaciones existe la necesidad de sobrescribir y eliminar datos. Al sobrescribirse los datos se crean pérdidas y/o variación en los mismos.

Por ejemplo, en el sistema de seguridad si se elimina un usuario, todos los eventos que hayan sido realizados por ese usuario no podrán ser almacenados en el registro de eventos debido a la inexistencia de dicho usuario.

1.4.2. Eventifier

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

datos. 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 proceso. El resultado del proceso completo es un registro de eventos listo para aplicársele Minería de Proceso (Rodríguez Carlos, 2012).

Las principales desventajas de Eventifier son las siguientes: la herramienta solo es aprovechable si existe una ejecución previa los procesos. La misma se limita al descubrimiento de procesos. En adición, esta herramienta es una aplicación desktop desarrollada en Java por lo que necesita una máquina virtual de Java para su ejecución.

Resultados

Después de haberse analizado las dos herramientas se llega a la conclusión que solo son aprovechables si existe evidencia en la base de datos operacional de la ejecución de al menos un proceso. Esto limita la usabilidad en sistemas de información con aplicaciones de gestión. Además, no tienen en cuenta la convergencia ni divergencia entre las trazas, creando posibilidades de errores en el registro de eventos.

1.5. Herramientas y Tecnologías utilizadas

Para el modelado del negocio y la implementación de la solución, se utilizaron las tecnologías que brindan soporte a Sauxe y herramientas definidas por el CEIGE para el desarrollo de software de gestión. Estas son:

Herramientas

Apache 2.2.9.

Apache es un servidor de aplicaciones cuya adaptabilidad, robustez y estabilidad lo han hecho popular desde 1996. Proporciona un servidor seguro, eficiente y extensible que provee servicios HTTP (traducido al español Protocolo de Transferencia de Hipertexto) en sincronía con los estándares HTTP actuales. Es una tecnología gratuita de código abierto (The Apache Software Foundation 2011).

Netbeans7.1

NetBeans es un IDE (traducido al español Entorno de Desarrollo Integrado) libre y de código abierto, que con variados módulos brinda las herramientas necesarias para el desarrollo de aplicaciones profesionales de escritorio, web y aplicaciones móviles con la plataforma Java, así como con PHP y otras (Oracle Corporation and its affiliates 2012).

Visual Paradigm 8.0

Visual Paradigm es una herramienta CASE (traducido al español Ingeniería de Software Asistida por Computadora) que utilizando UML (traducido al español Lenguaje de Modelado Unificado) como lenguaje de modelado, permite la captura, diseño, gestión y documentación de los artefactos generados durante el proceso de desarrollo de software (Visual Paradigm 2011).

Tecnologías

PostgreSQL 8.3.8

PostgreSQL es un sistema de base de datos objeto-relacional de almacenamiento y manipulación de datos muy común. Permite la creación de tipos propios. Incorpora una estructura de datos array. Incorpora funciones de diversa índole, manejo de fechas, geométricas, orientadas a operaciones con redes, entre otras. Permite la declaración de funciones propias, así como la definición de disparadores. Soporta el uso de índices, reglas y vistas. Incluye herencia entre tablas (aunque no entre objetos), por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales. Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos (ANON, 2010).

Marco de trabajo Sauxe 2.3

Sauxe es un marco de trabajo que contiene un conjunto de componentes reutilizables. Provee la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo (Baryolo Oiner, Silega Nemuris, 2008).

ExtJS 2.2

ExtJS es una librería JavaScript ligera y de alto rendimiento, construida para el desarrollo veloz de aplicaciones Web, compatibles con la mayoría de navegadores. Utiliza técnicas como Ajax, DHTML y manipulación del DOM. ExtJS incluye un conjunto de controles para el desarrollo de interfaces en los desarrollos web (ANON, 2011).

PHP5.2.6

PHP es un lenguaje de scripting de propósito general ampliamente utilizado que es especialmente adecuado para el desarrollo web y puede ser embebido en páginas HTML. La mayoría de su sintaxis es similar a C, Java y Perl. Su meta es permitir escribir a los creadores de páginas web, páginas dinámicas de una manera rápida y fácil (ANON, 2012).

Doctrine 0.9

Doctrine es un mapeador de objeto relacional (ORM por sus siglas en inglés) para PHP 5.2.3 que se encuentra en la parte superior de una capa poderosa de abstracción de base de datos (DBAL por sus siglas en inglés). Una de sus principales características es la opción de escribir las consultas de base de datos en un dialecto orientado a objetos de propiedad SQL llamado Doctrine Query Language (DQL), lo que proporciona a los desarrolladores una alternativa a SQL, que mantiene la flexibilidad sin necesidad de duplicar el código innecesario (ANON. 2012).

Conclusiones del Capítulo

El análisis realizado al marco de trabajo Sauxe y de las extensiones de XES arrojó como resultado que la información que almacena su sistema de auditoría no es suficiente para realizar estudios utilizando técnicas de Minería de Proceso.

El estudio de las herramientas para extracción de trazas de proceso a partir de bases de datos operacionales, permitió identificar debilidades de las mismas pues estas dependen de la evidencia de la ejecución previa de los procesos. Esta evidencia por su parte, puede ser inconsistente ya que los datos en ocasiones son eliminados o sufren variaciones debido a la naturaleza de las bases de datos operacionales.

CAPÍTULO 2: PROPUESTA DE SOLUCIÓN

Introducción

En este capítulo se describe la propuesta de solución denominada Módulo para el registro y transformación de trazas de eventos a formato XES. Para lograr un mejor entendimiento de la propuesta de solución se presenta el modelo conceptual. Adicionalmente, se detalla la propuesta de solución. Se identifican y describen los requisitos funcionales y no funcionales. Se presenta el patrón arquitectónico y los patrones del diseño empleados en el desarrollo de la solución. Posteriormente se muestran los artefactos generados durante la fase de análisis y diseño.

2.1. Modelo conceptual

Los principales conceptos y relaciones de la solución se presentan en la Figura 4. El presente modelo conceptual ilustra el funcionamiento del registro de trazas una vez implementado el Módulo para el registro y transformación de trazas de eventos a formato XES. Sauxe es un marco de trabajo el cual posee subsistemas. Las aplicaciones de estos subsistemas generan trazas. Adicionalmente, Sauxe tiene definido procesos de negocio los cuales poseen eventos asociados. Estas trazas generadas contienen los eventos definidos en los procesos de negocio. Un proceso puede generar registros de eventos los cuales agrupan trazas de un proceso y son exportados al formato XES, mediante el uso de sus extensiones. El experto de dominio es quien configura los procesos delimitando los datos a capturar.

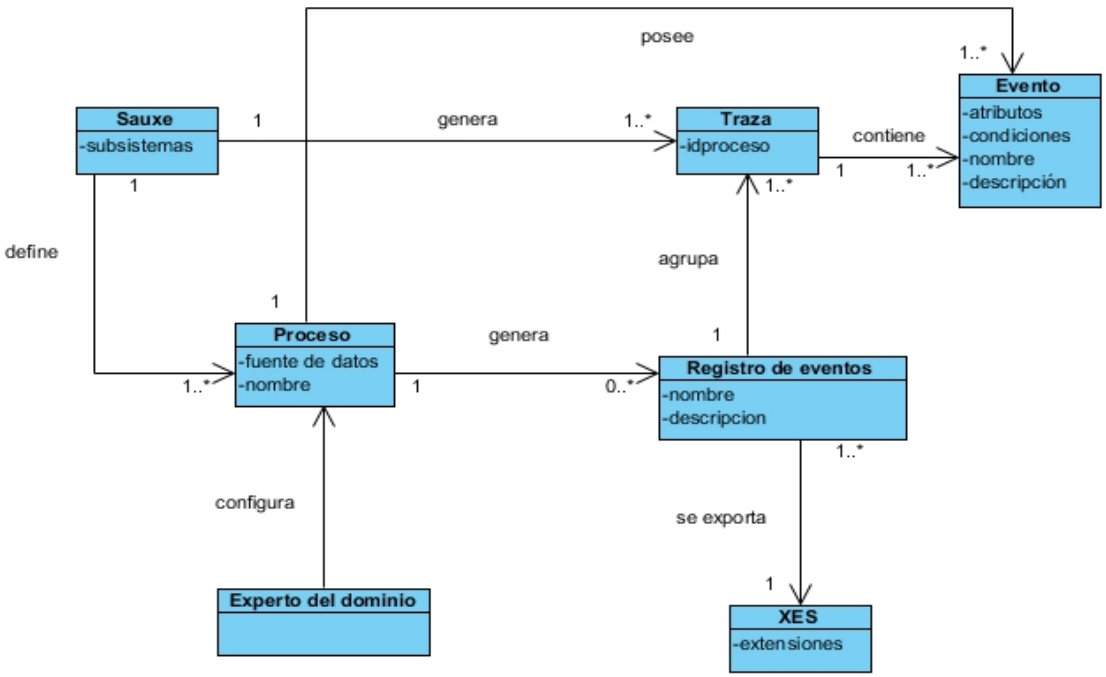


Figura 4. Modelo conceptual.

2.2. Módulos de Sauxe relevantes para la solución

En aras de hacer un mejor uso de los patrones de diseño, se realizó un estudio de las aplicaciones de Sauxe en busca de código reutilizable. Dicho estudio identificó módulos de Sauxe útiles para la implementación de la solución. Los módulos identificados se describen a continuación:

2.2.1. Historiales de datos

El módulo de Historiales permite crear tablas espejo de tablas específicas en la base de datos de aplicación de Sauxe. Esta réplica captura todos los datos de la original. Además, la misma registra la operación realizada sobre los datos (insertar, modificar o eliminar) y el instante de tiempo de dicha operación. Estas tablas espejo persisten en el tiempo y no sufren variaciones en sus datos. El uso del módulo de Historiales permite superar una de las principales deficiencias identificadas en las herramientas XESame y Eventifier. Una réplica de una tabla no sufre variaciones ni pérdidas de los datos. Una modificación o eliminación de datos en la tabla original se traduce como otra adición en su respectiva tabla espejo, reflejándose la acción realizada sobre los datos en el campo operación. En la Figura 5 se muestra una imagen de una tabla y una réplica de la misma utilizando la solución de Historiales.

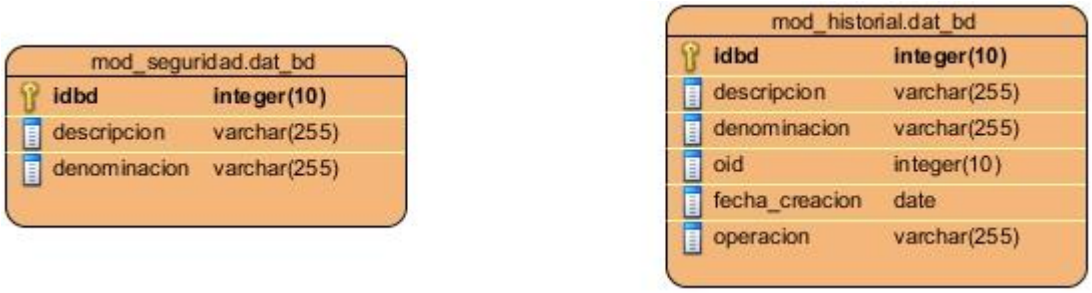


Figura 5. Historial de datos.

2.2.2. Trazas de datos

Las trazas y las trazas de datos almacenan un gran volumen de la información relacionada al usuario que utiliza Sauxe, así como particularidades de los datos manejados respectivamente. Entre trazas y trazas de datos, existe una relación de herencia. Esta herencia facilita la obtención de información de las trazas de datos mediante el identificador de la traza. A continuación se muestran unas imágenes de la información que recogen las trazas y las trazas de datos.

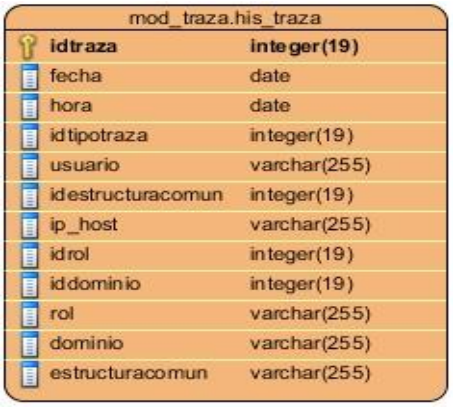


Figura 6. Trazas.

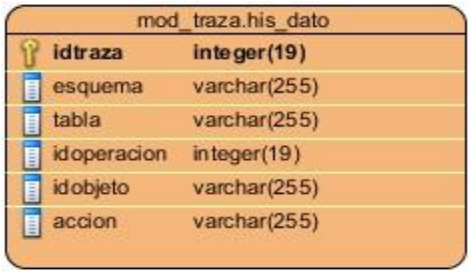


Figura 7. Trazas de datos.

2.3. Propuesta de solución

Como se explicó en el capítulo anterior, las trazas de proceso son el tipo de traza que posee la información suficiente para realizar estudios utilizando técnicas de Minería de Proceso. La solución se enmarca en la configuración del registro de trazas de proceso por un experto del dominio a partir de las trazas que generan las aplicaciones de Sauxe y su exportación como registro de eventos en formato XES. Para lograr lo anteriormente planteado se decidió hacer uso de la solución de Historiales de datos de Sauxe, así como de las trazas de datos. El uso del módulo de Historiales resuelve el problema de la pérdida y/o variación de los datos tan comunes en las OD. La otra parte necesaria de los datos para la configuración del registro de trazas de proceso, se encuentra en las trazas de datos. La información contenida en las trazas, es referente al usuario que realizó los cambios y a las particularidades de los datos alterados. Una vez almacenada toda la información necesaria se procede, a petición del usuario, a la generación de las trazas de proceso. Las trazas de proceso pueden ser exportadas como registro de eventos al formato XES para la realización de futuros análisis utilizando técnicas de Minería de Proceso.

Para el desarrollo de la solución se determinó la implementación de cuatro módulos. Un módulo para la gestión de las conexiones, dos para cubrir el proceso de configuración del registro de trazas de proceso

y uno para la generación de trazas de proceso y administración de los procesos. Además, se incluyó una funcionalidad adicional en el Módulo de gestión de trazas para la exportación de trazas de proceso. Dichos módulos se describen a continuación:

Módulo primero: Gestión de la conexión

Gestiona las conexiones a base de datos definidas exclusivamente para la definición y configuración de un proceso. Dichas conexiones pueden ser independientes para cada proceso o pueden ser compartidas entre varios procesos.

Módulo segundo: Definición de proceso

Permite la definición de un proceso. Esta definición consiste en la selección de una o más tablas asociadas a la ejecución del proceso, la definición del nombre (único para cada proceso), fuente de datos, descripción del proceso y la selección de la tabla principal del proceso la cual debe poseer el identificador de instancia de proceso. Este módulo permite adicionalmente, definir eventos con su nombre y descripción asociados a un proceso determinado.

Módulo tercero: Configuración de proceso

Este módulo permite la configuración de los datos necesarios para generar las trazas de proceso de uno previamente definido. La configuración de un proceso consiste en la asociación de los atributos de cada uno de los eventos a columnas de tablas de bases de datos. Estas asociaciones pueden estar bajo ciertas condiciones especificadas. Los atributos de los eventos son las extensiones definidas por el estándar XES anteriormente.

Módulo cuarto: Administración de proceso

Administra los procesos previamente definidos y configurados. La administración de un proceso está compuesta por la validación, activación o desactivación de un proceso y la generación de trazas de proceso. La validación de un proceso consiste en verificar que todos los atributos de todos los eventos tengan columnas asociadas y que las condiciones especificadas para todos los eventos posean tipos de datos compatibles. La validación de un proceso es un requisito previo para la activación del mismo. La activación de un proceso consiste en la creación de historiales de las tablas cuyas columnas forman parte de los atributos de los eventos y/o las condiciones de los mismos, la activación de las trazas de datos y el registro de la activación del proceso. La desactivación de un proceso consiste en el registro de la desactivación del mismo. En el caso de la desactivación se determinó no eliminar los historiales creados ni desactivar las trazas de datos, debido a que ambas soluciones pueden ser usadas por terceros. La generación de trazas de proceso consiste en construir dichas trazas con los datos de los historiales en conjunto con las trazas de datos teniendo en cuenta el rango de tiempo en que el proceso ha estado activo. Una vez que el proceso ha estado activo al menos una vez, cualquier cambio en la configuración o definición (excluyendo nombre, descripción y fuente de datos) del mismo se refleja como

una nueva versión del proceso. Si un proceso recibe cambios en su configuración, este se invalida y desactiva automáticamente en caso de estarlo. Un proceso se invalida y desactiva automáticamente si recibe cambios en su definición que no incluyan el nombre del proceso, descripción y/o fuente de datos del mismo.

Funcionalidad añadida al Módulo de gestión de trazas: Exportar Trazas de proceso a formato XES

Esta funcionalidad permite exportar las trazas de proceso a formato XES previamente generadas por el Módulo de administración de proceso o las generadas por el motor de flujo de trabajo de Sauxe cuando este se encuentre operacional. La exportación permite especificar el proceso al que se le quiere exportar las trazas, el intervalo de tiempo de las trazas a exportar y la versión del proceso a exportar.

2.4. Requisitos de software

El proceso de desarrollo de software comprende en sus etapas tempranas la definición de tareas orientadas a captar las necesidades o características para satisfacer el sistema que se vaya a crear o modificar. Como resultado de esta captación, se obtendrán los requisitos con los que deberá cumplir la solución (Pressman, 2005).

2.4.1. Requisitos funcionales

El proceso de captura de requisitos utilizando los métodos **entrevista con el cliente**, **tormenta de ideas** y **observación de sistemas semejantes**, identificó 31 requisitos funcionales (RF). Estos requisitos fueron agrupados por el criterio de **agrupación modular** el cual permite detectar el alcance de cada requisito, delimitando las partes del sistema a las que afecta.

A continuación se muestran los requisitos funcionales identificados y se describen los de mayor relevancia (*) para la solución. Para consultar el resto de las descripciones funcionales véase el Expediente de proyecto de Sauxe.

RF1: Gestionar conexión	RF2: Definir proceso	RF3: Configurar proceso	RF4: Administrar proceso	RF5: Exportar trazas de proceso
RF1.1 Listar conexiones.	RF2.1 Listar procesos.	RF3.1 Listar procesos y eventos.	RF4.1 Listar procesos.	RF5.1 Listar procesos.
RF1.2 Adicionar conexión.	RF2.2 Adicionar proceso.(*)	RF3.2 Agregar condiciones a los eventos.	RF4.2 Activar proceso.(*)	RF5.2 Exportar trazas de proceso.
RF1.3 Modificar conexión.	RF2.3 Modificar proceso.(*)	RF3.3 Listar condiciones de los eventos.	RF4.3 Desactivar proceso.	
RF1.4 Eliminar conexión.	RF2.4 Eliminar proceso.	RF3.4 Adicionar configuración de eventos.	RF4.4 Validar proceso.(*)	
RF1.5 Probar conexión.	RF2.5 Buscar conexión.	RF3.5 Modificar configuración de eventos.	RF4.5 Generar trazas de proceso.(*)	
	RF2.6 Listar esquemas.			
	RF2.7 Buscar esquema.			
	RF2.8 Listar tablas.			
	RF2.9 Buscar tabla.			
	RF2.10 Listar eventos.			
	RF2.11 Adicionar evento.			
	RF2.12 Modificar evento.			
	RF2.13 Eliminar evento.			
	RF2.14 Listar conexiones.			

Figura 8. Listado de requisitos funcionales.

RF2: Definir proceso

RF2.1 Adicionar proceso

Tabla 1. RF2.1 Adicionar proceso.

Precondiciones	– Debe existir al menos una conexión disponible.
Flujo de eventos	
Flujo básico 1 Adicionar proceso.	
1	Se muestra un listado de las conexiones disponibles.
2	Se selecciona una conexión de las existentes.
3	Se presiona el botón **Siguiente**.
4	Se muestra un listado de los esquemas asociados a la conexión seleccionada.
5	Se selecciona uno o varios esquemas.
6	Se presiona el botón **Siguiente.**
7	Se muestra un listado de las tablas asociados a los esquemas seleccionados.
8	Se selecciona una o varias tablas.
9	Se presiona el botón **Siguiente.**
10	Se muestra una interfaz dividida en dos campos, en la parte izquierda se insertan los datos del nuevo proceso y en la derecha se muestra un listado de tablas, las cuales una de ellas contendrá el identificador de instancia de proceso.
11	Se insertan los datos en la parte izquierda de la interfaz en los campos: – Nombre del proceso. – Fuente de datos. – Descripción.
12	Se validan los datos (Ver validación 1).
13	Se selecciona una instancia de proceso en la parte derecha de la interfaz.
14	Se presiona el botón **Adicionar**.
15	Se muestra un mensaje de información: **“Proceso Adicionado satisfactoriamente”.**
16	Concluye el requisito.
Pos-condiciones	
1	Se adiciona un proceso en el sistema.
Flujos alternativos	
Flujo alternativo 14.a Adicionar proceso dejando campos vacíos.	

1	Se muestra un mensaje de error: **“Por favor verifique, existen campos vacíos”.**
2	Volver al paso 11 del flujo básico.
Pos-condiciones	
1	N/A
Flujo alternativo 14.b Adicionar un proceso con valores incorrectos.	
1	Se muestra un mensaje de error: **“Por favor verifique, existen campos con valores incorrectos”.**
2	Volver al paso 11 del flujo básico.
Pos-condiciones	
1	N/A
Flujo alternativo 4.c Adicionar un proceso con un nombre ya existente.	
1	Se muestra un mensaje de error: **“Error, existe un proceso con el mismo nombre”.**
2	Volver al paso 11 del flujo básico.
Pos-condiciones	
1	N/A
Flujo alternativo *.a El usuario cancela la acción.	
1	Concluye el requisito.
Pos-condiciones	
1	No se registran los datos.
Validaciones	
Se validan los datos según lo establecido en el Modelo conceptual Módulo para el registro y transformación de trazas de eventos a formato XES.	
Conceptos	N/A
Requisitos especiales	N/A
Asuntos pendientes	N/A

Prototipo elemental de interfaz gráfica de usuario.

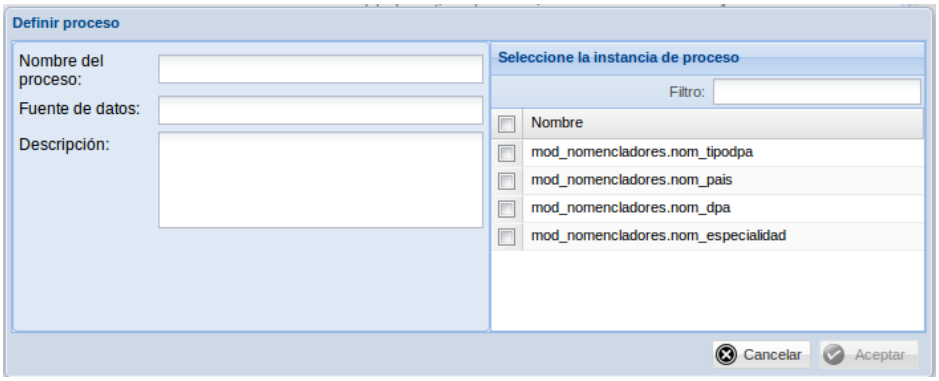


Figura 9. Interfaz gráfica del requisito Adicionar proceso.

RF2.2 Modificar proceso

Tabla 2. RF2.2 Adicionar proceso.

Precondiciones	– Debe existir al menos una conexión disponible. – Debe existir al menos un proceso definido en el sistema.
Flujo de eventos	
Flujo básico 1 Modificar proceso.	
1	Se selecciona un proceso de los listados.
2	Se presiona el botón **Modificar proceso.**
3	Se muestra un listado de las conexiones disponibles y la conexión del proceso seleccionada.
4	Se selecciona otra conexión de las existentes si se desea.
5	Se presiona el botón **Siguiente**.
6	Se muestra un listado de los esquemas asociados a la conexión seleccionada.
7	Se selecciona o deselecciona los esquemas deseados.
8	Se presiona el botón **Siguiente.**
9	Se muestra un listado de las tablas asociados a los esquemas seleccionados.
10	Se selecciona o deselecciona una o varias tablas.
11	Se presiona el botón **Siguiente.**
12	Se muestra una interfaz dividida en dos campos, en la parte izquierda se modifican los datos del proceso seleccionado y en la derecha se muestra un listado de tablas, las cuales una de ellas contendrá el identificador de instancia de proceso.
13	Se modifican los datos en la parte izquierda de la interfaz en los campos:

	– Nombre del proceso. – Fuente de datos. – Descripción.
14	Se validan los datos (Ver validación 1).
15	Se cambia de instancia de proceso en la parte derecha de la interfaz si se desea.
16	Se presiona el botón **Modificar**.
17	Se muestra un mensaje de información: **"Proceso Modificado satisfactoriamente".**
18	Concluye el requisito.
Pos-condiciones	
1	Se modifica un proceso en el sistema.
2	El proceso se desactiva en caso de estar activado si se modificaron valores además de: – Nombre del proceso. – Descripción.
3	El proceso se desvalida en caso de estar validado si se modificaron valores además de: – Nombre del proceso. – Descripción.
4	Se crea una nueva versión del proceso si se modificaron valores además de: – Nombre del proceso. – Descripción.
Flujos alternativos	
Flujo alternativo 14.a Modificar un proceso dejando campos vacíos.	
1	Se muestra un mensaje de error: **"Por favor verifique, existen campos vacíos".**
2	Volver al paso 13 del flujo básico.
Pos-condiciones	
2	N/A
Flujo alternativo 14.b Modificar un proceso con valores incorrectos.	
1	Se muestra un mensaje de error: **"Por favor verifique, existen campos con valores incorrectos".**
2	Volver al paso 13 del flujo básico.

Pos-condiciones	
1	N/A
Flujo alternativo 4.c Modificar un proceso con un nombre ya existente.	
1	Se muestra un mensaje de error: **"Error, existe un proceso con el mismo nombre".**
2	Volver al paso 13 del flujo básico.
Pos-condiciones	
1	N/A
Flujo alternativo *.a El usuario cancela la acción	
1	Concluye el requisito.
Pos-condiciones	
1	No se registran los datos.
Validaciones	
1	Se validan los datos según lo establecido en el Modelo conceptual Módulo para el registro y transformación de trazas de eventos a formato XES.
Conceptos	N/A
Requisitos especiales	N/A
Asuntos pendientes	N/A

Prototipo elemental de interfaz gráfica de usuario.

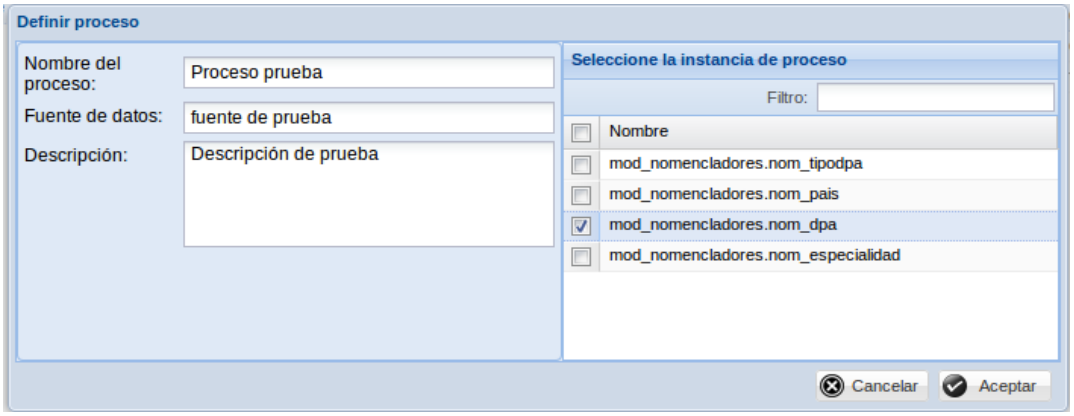


Figura 10. Interfaz gráfica del requisito Modificar proceso.

RF4 Administrar proceso.

RF4.2 Activar proceso.

Tabla 3. RF4.2 Activar proceso.

Precondiciones	– Debe existir al menos un proceso validado en el sistema.
Flujo de eventos	
Flujo básico Activar proceso.	
1	Se selecciona un proceso.
2	Se presiona el botón **Activar Proceso**.
3	Se muestra un mensaje de información: **“El proceso se ha activado satisfactoriamente”.**
4	Concluye el requisito.
Pos-condiciones	
1	Se registra la activación de un proceso en el sistema.
2	Se crean historiales de las tablas del proceso si no están creadas.
3	Se activan las trazas de datos si están desactivadas.
Flujos alternativos	
Flujo alternativo 2.a Activar un proceso que no está validado.	
1	Se muestra un mensaje de error: **“El proceso no está validado”.**
2	Volver al paso 1 del flujo básico.
Pos-condiciones	
1	N/A
Flujo alternativo 2.b Activar un proceso ya activo o activado.	
1	Se muestra un mensaje de error: **“El proceso ya está activado”.**
2	Volver al paso 1 del flujo básico.
Pos-condiciones	
1	N/A
Conceptos	N/A
Requisitos especiales	N/A
Asuntos pendientes	N/A

RF4.4 Validar proceso.

Tabla 4. RF4.4 Validar proceso.

Precondiciones	– Deben existir al menos un proceso configurado en el sistema.
Flujo de eventos	
Flujo básico Validar proceso.	
1	Se selecciona un proceso.
2	Se presiona el botón **Validar Proceso**.
3	Se muestra un mensaje de información: **“El proceso se ha validado correctamente”.**
4	Concluye el requisito.
Pos-condiciones	
4	Se valida un proceso en el sistema.
5	Se actualizan las tablas del proceso.
Flujos alternativos	
Flujo alternativo 2.a Validar un proceso con errores en su configuración.	
1	Se muestra un mensaje de error: **“El proceso tiene errores en su configuración. No se puede validar”.**
2	Volver al paso 1 del flujo básico.
Pos-condiciones	
1	N/A
Flujo alternativo 2.b Validar un proceso con la configuración incompleta.	
1	Se muestra un mensaje de error: **“La configuración del proceso está incompleta. Necesita completarla”.**
2	Volver al paso 1 del flujo básico.
Pos-condiciones	
1	N/A
Flujo alternativo 2.c Validar un proceso con la configuración incorrecta.	
1	Se muestra un mensaje de error: **“La configuración del proceso está incorrecta. Algunas condiciones poseen tipos de datos incompatibles”.**
2	Volver al paso 1 del flujo básico.
Pos-condiciones	

1	N/A
Flujo alternativo 2.d Validar un proceso ya validado.	
1	Se muestra un mensaje de error: **"El proceso ya está validado".**
2	Volver al paso 1 del flujo básico.
Pos-condiciones	
1	N/A
Conceptos	N/A
Requisitos especiales	N/A
Asuntos pendientes	N/A

RF4.5 Generar trazas de proceso.

Tabla 5. RF4.5 Generar trazas de proceso.

Precondiciones	– Deben existir al menos un proceso activado en el sistema.
Flujo de eventos	
Flujo básico Generar trazas de proceso.	
1	Se selecciona un proceso.
2	Se presiona el botón **Generar trazas de proceso**.
3	Se muestra un mensaje de información: **"Se han generado las trazas de proceso correctamente".**
4	Concluye el requisito.
Pos-condiciones	
1	Se generan trazas de proceso.
Flujos alternativos	
Flujo alternativo 2.a Seleccionar un proceso que no está activo.	
1	Se muestra un mensaje de error: **"El proceso no está activo".**
2	Volver al paso 1 del flujo básico.
Pos-condiciones	
3	N/A

Asuntos pendientes	N/A

2.4.2. Requisitos no funcionales

Dado que la solución se utilizará internamente en el marco de trabajo Sauxe, no se han definido requisitos no funcionales (RnF) adicionales a los que Sauxe ya posee.

Usabilidad

RnF 1: el sistema podrá ser usado por cualquier persona que posea conocimientos básicos en el manejo de la computadora.

Eficiencia

RnF 2: los tiempos de respuesta y velocidad de procesamiento de la información serán rápidos, no mayores de 5 segundos para las actualizaciones y 20 para las recuperaciones.

Soporte

RnF 3: la aplicación contará antes de su puesta en marcha con un período de pruebas, se le dará mantenimiento, configuración y se brindará el servicio de instalación.

RnF 4: los desarrolladores deben utilizar para el desarrollo de la solución el estándar de codificación elaborado en el Departamento de Tecnología.

Restricciones de diseño

RnF 5: el lenguaje de programación a utilizar para desarrollar el sistema debe ser PHP para la lógica del negocio y ExtJS para la capa de presentación.

RnF 6: las herramientas a utilizar para desarrollar el sistema deben ser PostgreSQL (desde 8.3 hasta 9.1) como gestor de base de datos y Pgadmin III como cliente, Doctrine para la capa de acceso a datos y ZendExt Framework para la lógica del negocio.

RnF 7: el sistema debe ser multiplataforma, haciendo énfasis en Linux y Windows.

Ambiente.

- Para que el sistema funcione es necesario garantizar los siguientes requisitos de software:

Para el cliente:

RnF 8: Navegador Mozilla Firefox.

RnF 9: Sistema operativo Windows xp o superior o Linux.

- **Para el servidor:**

RnF 10: Sistema operativo Windows Advancer Server (2000 o superior) o Linux en cualquiera de sus distribuciones.

RnF 11: un servidor Apache 2.0 o superior con módulo PHP 5.0 disponible, este debe estar configurado con la extensión “pgsql” incluida.

RnF 12: un servidor de base de datos PostgreSQL 8.0 o superior.

- Para que el sistema funcione es necesario garantizar los siguientes requisitos de hardware:

Para el servidor:

RnF 13: requerimientos mínimos: Procesador Pentium III a 1GHz de velocidad de procesamiento y 1Gb de memoria RAM.

RnF 14: al menos 40Gb de espacio libre en disco duro.

RnF 15: Tarjeta de red.

- **Para el cliente:**

RnF 16: requerimientos mínimos: Procesador Pentium II a 133Mhz con 128 Mb de memoria RAM.

RnF 17: tarjeta de red.

2.5. Modelo de Diseño

Uno de los artefactos generados en la fase de análisis y diseño es el modelo de diseño. Este artefacto está compuesto por los diagramas de clases y diagramas de secuencias los cuales serán la guía para la implementación de la solución.

2.5.1. Estilo Arquitectónico

El estilo arquitectónico definido por el Departamento de Tecnología, donde se desarrolla el marco de trabajo Sauxe, es el **Estilo N capas**. Se identificaron cuatro niveles o capas fundamentales. La base tecnológica Sauxe cuenta con la capa de presentación, control, acceso a datos y base de datos (Baryolo, 2010).

- **Capa de presentación:** esta es la capa encargada de elaborar y mostrar las interfaces de los usuarios. La capa de presentación está compuesta principalmente por el marco de trabajo ExtJS y centra su desarrollo en tres componentes fundamentales archivos JS, archivos CSS, páginas clientes y páginas servidoras (Baryolo, 2010).
- **Capa de control o negocio:** en esta capa se encuentra ubicado el marco de trabajo Zend y se implementa el patrón Modelo-Vista-Controlador (Baryolo, 2010).
- **Capa de acceso a datos:** está ubicado el marco de trabajo Doctrine, como Object Relation Manager (ORM) para la comunicación con el servidor de datos mediante el protocolo PDO[2] (Baryolo, 2010).

[2] Protocolo de conexión a base de datos basado en objetos.

- **Capa datos: e**n esta capa se encuentran ubicado los servidores de base de datos y los archivos XML donde se almacena la información de la aplicación (Baryolo, 2010).

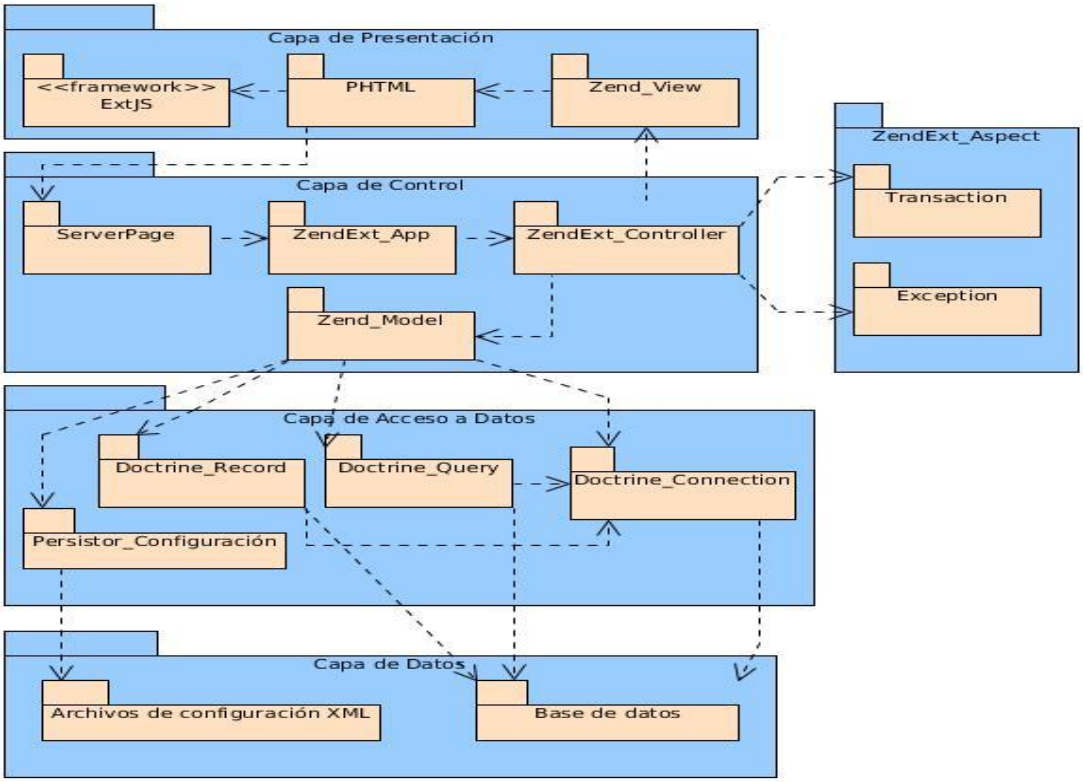


Figura 11. Arquitectura de Sauxe.

2.5.2. Patrón Arquitectónico

El patrón arquitectónico Modelo-Vista-Controlador (MVC) se emplea en la capa de control y es el encargado de separar los datos de la aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos:

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

2.5.3. 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 que propone este trabajo, se utilizaron algunos patrones de diseño GoF (traducido al español Grupo de Cuatro) y 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: propone que la clase que contenga toda la información necesaria, será la responsable de la creación de un objeto o la implementación de un método. El comportamiento se distribuye entre las que contienen la información requerida, siendo más fáciles de entender, mantener y ampliar, aumentando sus posibilidades de reutilización. Se observa el uso de este patrón en todas las clases a utilizar en la solución ya que cada clase conoce su información y es la encargada de implementar las funcionalidades que brindan información de las mismas.

Controlador: el patrón controlador funciona como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la interfaz quien recibe los datos del usuario y los envía a las distintas clases según el método invocado. Ejemplo de esto la clase controladora ProcessAdministrationController se encarga de llevar el control de todos los eventos relacionados con el negocio e implementa las funcionalidades que dan respuesta a las peticiones del usuario.

Bajo acoplamiento: este patrón expresa que entre las clases deberán existir pocas ataduras, es decir, estas estarán lo menos relacionadas posible, de forma tal que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases, incrementando la reutilización y disminuyendo la dependencia entre las clases. El uso de este patrón se evidencia en la poca relación existente entre las clases que conforman los módulos.

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

GoF

Proxy: cuando no se desea el acceso directo a un objeto sobre el que se va a aplicar determinada acción, este patrón propone la adición de un nivel que permita solamente el acceso al objeto a través de

un objeto proxy sustituto, que será el responsable de controlar o mejorar el acceso al objeto real. Este patrón se evidencia mediante el uso de Doctrine para el trabajo con la base de datos.

El uso de estos patrones garantizó asignar a cada clase la responsabilidad que le corresponde, obtener el menor número de relaciones y dependencias entre clases y aumentar las posibilidades de reusabilidad de las mismas. Todos estos elementos ayudaron a la confección de un diseño de la solución de gran claridad y entendimiento.

2.5.4. Diagramas de clases

Otro de los artefactos a generar a tono con el modelo de desarrollo del CEIGE es el diagrama de clases del diseño con estereotipos web. Estos diagramas representan las clases de la solución, así como las principales relaciones que existen entre las clases. A continuación se muestra el diagrama de clase de la agrupación de requisitos Gestionar conexión. Para consultar el resto de los diagramas véase el Expediente de proyecto de Sauxe.

RF1. Gestionar conexión.

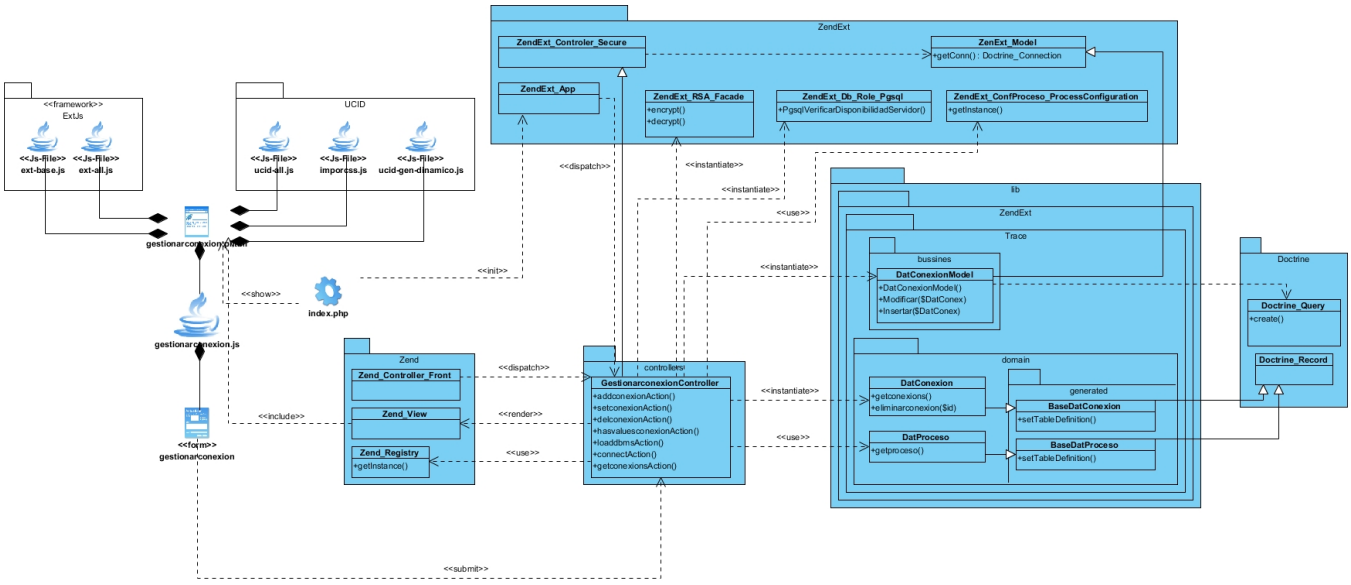


Figura 12. Diagrama de clase Gestionar conexión.

Descripción de las clases comunes para los diagramas de clases del diseño con estereotipos web.

***** Se refiere al nombre de las clases de los diferentes diagramas. Ejemplo *.js hace referencia a todas las clases con extensión .js de los diagramas.

Tabla 6. Descripción de las clases del diseño.

Clase	*Descripción*
***.js**	Es el JavaScript que se encarga de crear todas las interfaces que serán usadas por las diferentes funcionalidades.
***.phtml**	Es el HTML se incluyen todos los JavaScript para mostrar al usuario las interfaces que serán utilizadas en las diferentes funcionalidades.
index.php	Se encarga de determinar cuál es la clase controladora que le corresponde a cada página cliente.
<<Framework>> ExtJS	Librería JavaScript que utiliza la clase gestaccion.phtml para realizar las validaciones en los datos introducidos por el usuario al igual que para crear las interfaces.
Paquete UCID	Paquete JavaScript que utiliza la clase gestaccion.phtml para realizar las validaciones en los datos introducidos por el usuario al igual que para crear las interfaces.
***Controller**	Es la clase que gestiona la lógica de negocio y se encarga de capturar los parámetros que le son enviados desde la interfaz y devolver a las interfaces la información solicitada por el usuario.
***Model**	Es la clase encargada de procesar el negocio e interactuar con los modelos.
Base*	Es la clase base la cual tiene mapeado todos los campos de una tabla en la base de datos.
ZendExt_Controler_Secure	Clase padre del framework Zend de la cual extienden las clases controladoras.
ZendExt_RSA_Facade	Clase que ofrece Zend Framework. Se encarga de la seguridad cuando se trabaja con contraseñas.
ZendExt_Db_Role_Pgsql	Clase de Zend Framework que permite realizar transacciones a la base de datos.
ZenExt_Model	Entre las características de esta clase se pueden mencionar que obtiene las conexiones activas en el sistema cuando se trata de acceder a una clase determinada del modelo.
Doctrine_Query	Permite la creación de consultas hacia la base de datos para obtener los datos.

Doctrine_Record	Esta clase crea referencias de un objeto almacenado en una tabla de la base de datos, puede ser utilizada para conocer si un usuario puede modificar dicho objeto a través de los métodos mostrados en el diagrama.
Zend_View	Esta clase trabaja con la visión del patrón modelo-vista-controlador. Existe para ayudar a mantener el script de la vista separado del modelo y scripts del controlador.

2.5.5. Diagramas de secuencia

Como parte del diseño de la solución y a tono con el modelo de desarrollo del CEIGE se crearon diagramas de secuencias para lograr un mayor entendimiento de los métodos y su interacción con las clases de la solución.

A continuación se muestran los diagramas de secuencia de algunos requisitos de la solución. Para consultar el resto de los diagramas de secuencia véase el Expediente de proyecto de Sauxe.

RF1.2 Adicionar conexión.

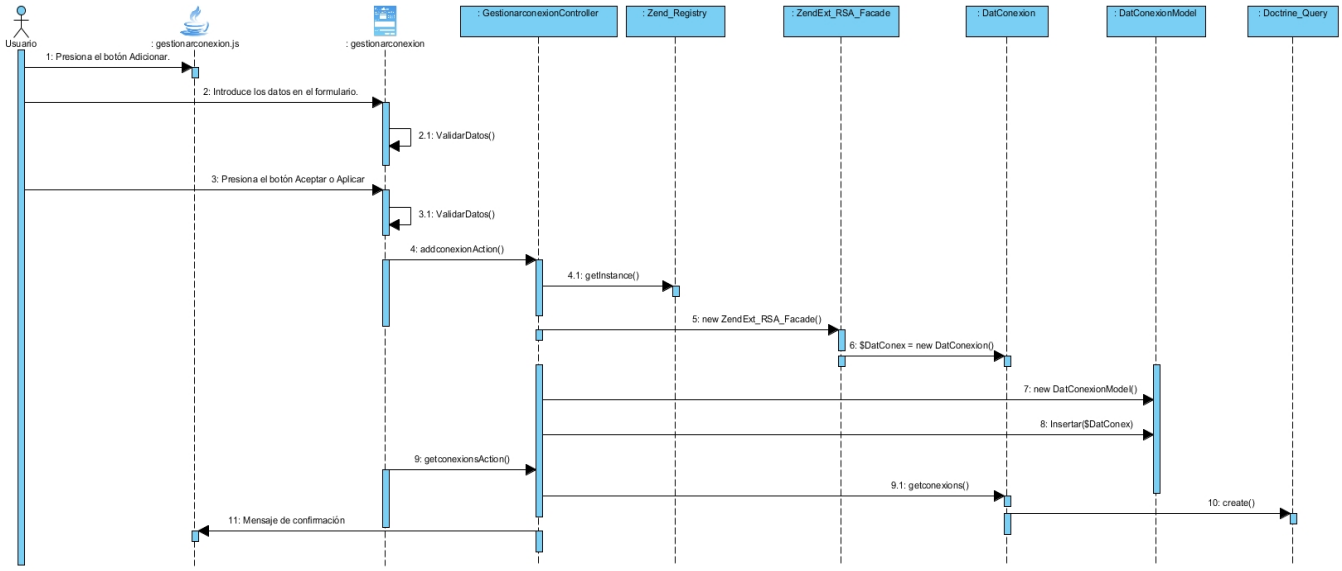


Figura 13. Diagrama de secuencia del RF1.2 Adicionar conexión.

RF4.3 Desactivar proceso.

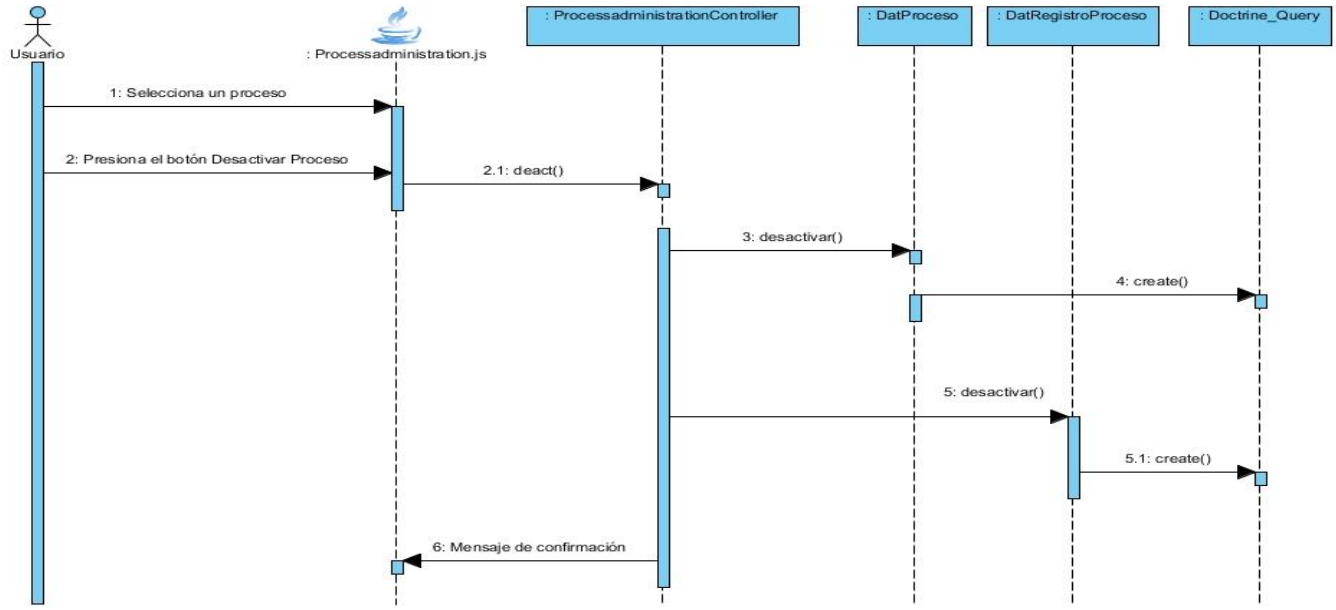


Figura 14. Diagrama de secuencia RF4.3 Desactivar proceso.

2.5.6. Modelo de datos

Otro de los artefactos generados en la fase de análisis y diseño de acuerdo con el modelo de desarrollo del centro es el modelo de datos. El mismo tienen el propósito de garantizar que los datos persistentes sean almacenados coherente, eficazmente y definir el comportamiento que debe ser implementado en la base de datos (Pressman, 2005). En la Figura 15 se observa el modelo de datos de la solución en el cual se ilustran las tablas de base de datos utilizadas y las relaciones entre las mismas.

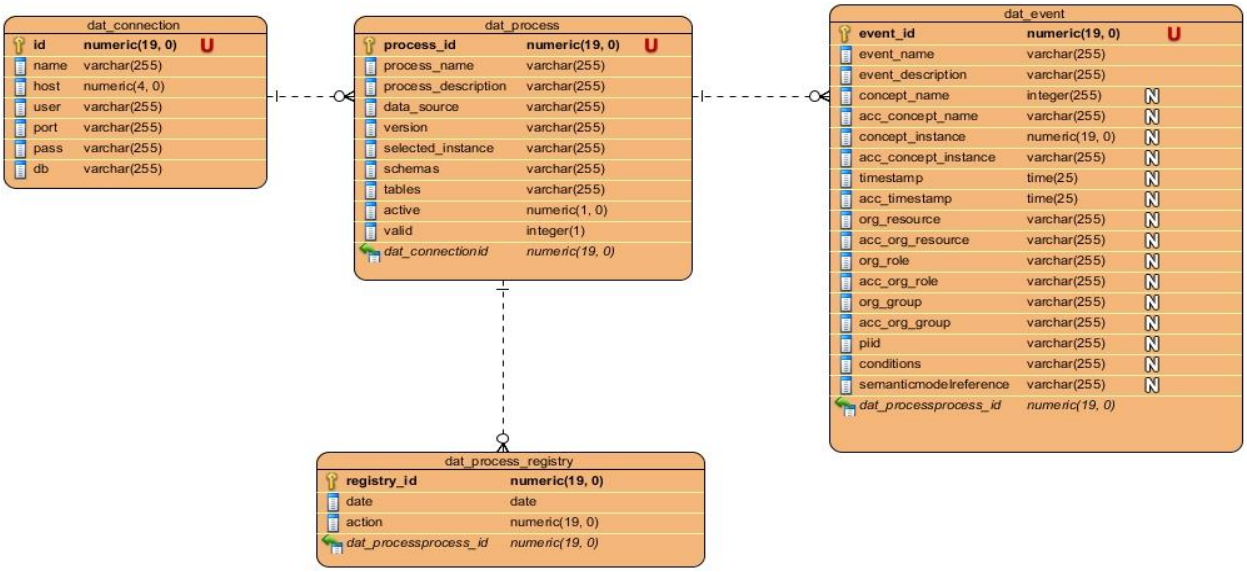


Figura 15. Modelo de datos.

dat_connection: contiene toda la información relacionada con las conexiones a usar para la gestión de los procesos.
dat_process: contiene toda la información de los procesos definidos, así como su estado de activación y validez.
dat_event: contiene toda la información relacionada a los eventos de los procesos definidos.
dat_process_registry: esta tabla almacena la información concerniente a la activación y desactivación de los procesos, así como los intervalos de tiempo de los mismos.

Conclusiones del Capítulo

La funcionalidad para exportación de trazas de proceso hacia el formato XES permitió que el módulo de gestión de trazas de Sauxe fuese capaz de generar registros de eventos en formato XES aptos para estudios de Minería de Proceso.
La definición de los datos a registrar a partir de las extensiones de XES aseguró la completitud de la información necesaria para la creación de trazas de proceso a generar.
Se elaboraron los artefactos definidos en el modelo de desarrollo del CEIGE relacionados con la fase de análisis y diseño que sirvieron de guía para la implementación del sistema.
El uso de los patrones de diseño permitió ganar en reutilización y brindó robustez al diseño de la solución.
La utilización de patrones de diseño facilita el mantenimiento y extensión por otros programadores debido a que proporcionan una estructura común conocida por todos los programadores.
La integración con otros módulos de Sauxe facilitó la reutilización de código, repercutiendo positivamente en el ahorro de tiempo y esfuerzo dedicado, así como una garantía de la calidad en las funcionalidades integradas.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBAS

INTRODUCCIÓN

En el presente capítulo se define el modelo de implementación para la creación de los módulos especificados en el capítulo anterior. Se describe el estándar de codificación utilizado para garantizar un correcto entendimiento del código. Se explican algunos algoritmos de interés especial para el desarrollo de la solución. Adicionalmente se realizan pruebas y validaciones certificando la calidad de la solución.

3.1. Modelo de implementación

El modelo de implementación describe cómo los elementos del modelo de diseño se implementan en términos de componentes. Describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje de programación utilizados; y cómo dependen los componentes unos de otros (Acuña, 2013).

3.1.1. Diagrama de componentes

El diagrama de componentes representa la estructura física del sistema, su agrupación por paquetes y muestra las dependencias entre estos.

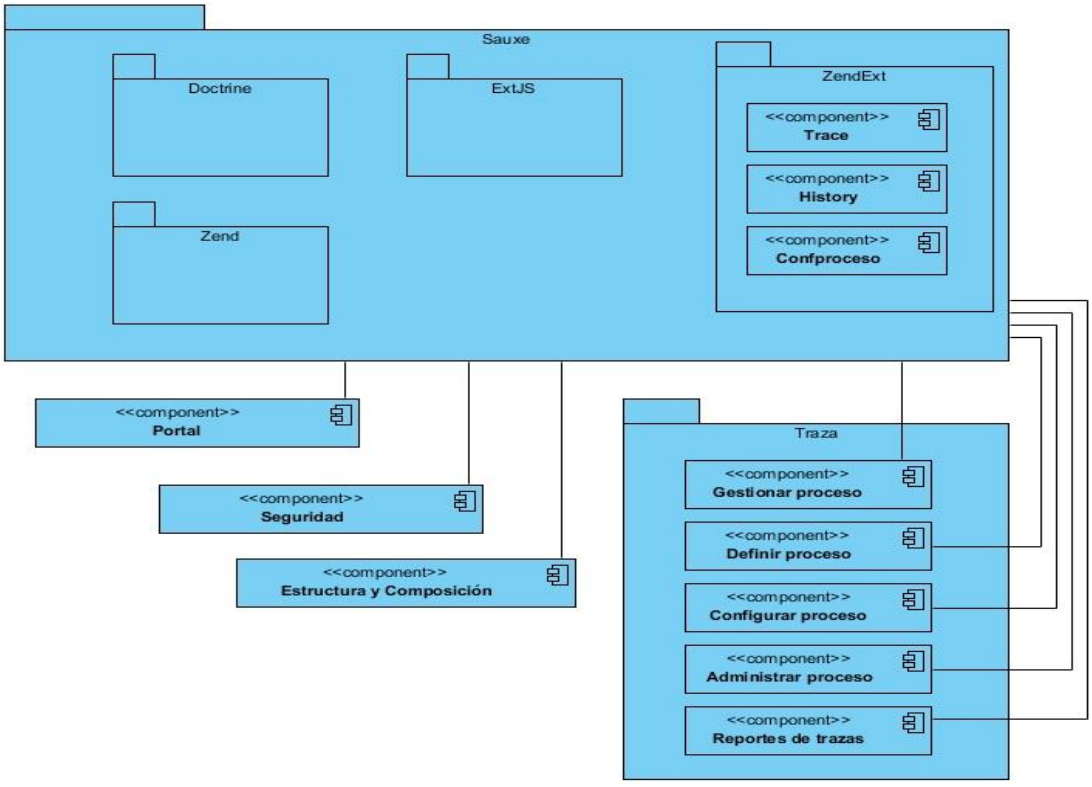


Figura 16. Diagrama de componentes.

3.1.2. Algoritmos de interés

La activación de un proceso y la generación de trazas de proceso constituyen el núcleo de la solución. Estas funcionalidades del módulo de administración de procesos permiten la interacción con otros sistemas de Sauxe y la creación de trazas de proceso aptas para estudios de Minería de Proceso. A continuación se describen estas funcionalidades con un mayor nivel de detalle.

Activar Proceso

El algoritmo de activar proceso primeramente verifica si el proceso ha sufrido modificaciones que requieran un cambio de versión del proceso. De ser necesario se procede a incrementar la versión del proceso en uno como se muestra en la Figura 17**.**

```
function activateprocessAction() {
    $id = $this->getRequest()->getPost('id');

            $proceso = DatProcess::activarProceso($id);
                    if($proceso->modificarversion==1){
                    $proceso->version=($proceso->version+1);
                    $proceso->modificarversion=0;
                    $m = new DatProcessModel();
                    $m->Modificar($proceso);
                        }
```

Figura 17. Fragmento del código activateprocessAction()(1).

En la Figura 18 se observa que posteriormente se verifica que el proceso esté validado pero no activado. En caso de no ser así se muestra un mensaje con la explicación pertinente de por qué no se activó el proceso. Si el proceso está validado pero no activado, se procede a activar las trazas de datos y se obtienen todas las tablas que se seleccionaron al momento de la definición del proceso, así como las tablas ya en uso por la solución de Historiales. Las tablas del proceso se preparan de tal manera que queden de la forma "**NombreEsquema.NombreTabla**" en un arreglo llamado ($tablasProceso). Las tablas de Historiales se almacenan en un arreglo llamado ($TProcesoH).

```
if ($validado == 1 && $activado == 0) {

    $this->actTrazaDatos();
    $tablasProceso = DatProcess::gettables($id);
    $tablasProceso = $tablasProceso[0]['tablas'];
    $tablasHistorial = $this->_history->gethistorial();
    $TProcesoH = array();
    $tablasProceso = str_replace('{', '', $tablasProceso);
    $tablasProceso = str_replace('}', '', $tablasProceso);
    $tablasProceso = split(',', $tablasProceso);
    $cont = 0;
    $aCrear = array();
    foreach ($tablasHistorial as $value) {
        $TProcesoH[$cont] = $value['esquema'] . "." . $value['tabla'];
        $cont++;
    }
```

Figura 18. Fragmento del código activateprocessAction()(2).

En la Figura 19 se aprecia que una vez obtenidas las tablas de Historiales y las seleccionadas del proceso, se procede a verificar qué tablas del proceso no poseen historiales creados. Estas tablas se almacenan en el arreglo ($diferencia). Posteriormente se almacenan en el arreglo ($Parahistorial) las tablas a crear por la solución de historiales en el formato requerido.

```
$diferencia = array_diff($tablasProceso, $TProcesoH);

$cont = 0;
foreach ($diferencia as $value => $key) {
    $aCrear[$cont] = split('\.', $key);
    $cont++;
} $cont = 0;

$Parahistorial = array();
foreach ($aCrear as $key) {
    $Parahistorial[$cont]->table_name = $key[1];
    $Parahistorial[$cont]->table_schema = $key[0];
    $cont++;
}
```

Figura 19. Fragmento del código activateprocessAction()(3).

Se captura el nombre de usuario de aplicación desde la configuración de Sauxe y se procede a la generación de las tablas historiales, así como se registra la activación del proceso en cuestión. Consecutivamente a esto se muestra un mensaje de notificación de la activación del proceso.

```
$user = $this->global->Perfil->usuario;
$this->_history->CreateHistorial($Parahistorial, $user);
$this->activarRegistroProceso($id);
echo json_encode(0);
```

Figura 20. Fragmento del código activateprocessAction()(4).

Generar trazas de proceso

Para generar trazas de proceso primeramente se crea una instancia de la clase “ZendExt_ConfProceso_ProcessConfiguration” la cual posee todos los métodos necesarios para la creación de un script SQL capaz de generar trazas de proceso. Se capturan los id de todos los eventos del proceso seleccionado en el arreglo ($events). Se hace uso del método **precreatesql** recibiendo como parámetro el id de un evento del proceso tantas veces como eventos tiene el arreglo ($events). En caso de haber algún error se muestra un mensaje de notificación.

```
public function generateprocesstracesAction() {
    $idp = $this->_request->getPost('idp');
    $ProcessConfiguration = ZendExt_ConfProceso_ProcessConfiguration::getInstance();
    $events = DatEvento::getidevent($idp);
    foreach ($events as $event){
        if(!($ProcessConfiguration->precreatesql($event['idevento'], $idp))){
            echo json_encode(0);
            return;
        }

    }
    echo json_encode(1);
    return;
}
```

Figura 21. Método generateprocesstracesAction().

El método **precreatesql** se encarga de la creación del script SQL el cual generará las trazas de proceso para cada rango de tiempo en que el proceso ha estado activo. Este método primeramente agrega los datos relevantes del negocio especificados (si existen) asociados al evento en cuestión. En la variable ($Db_Concrete) guarda una instancia de la solución de Historiales para ejecutar consultas SQL. En la variable ($fechas) almacena todas las fechas de activación y desactivación que posea el proceso al cual se le están generando trazas de proceso.

```
function precreatesql($idevent, $idp) {
    $this->addpl($idevent);
    $sql = "select mod_traza.dat_registro_proceso.fecha from mod_traza.dat_registro_proceso  where
    mod_traza.dat_registro_proceso.id_proceso=$idp;";
    $Db_Concrete = ZendExt_History_Db_Singleton::getInstance();
    $fechas = $Db_Concrete->query($sql);
    $count = count($fechas);
```

Figura 22. Fragmento del código precreatesql($idevent, $idp)(1).

Posteriormente en la Figura 23 se procede a la creación de trazas de proceso con el método **createsql** para cada rango de tiempo en que el proceso estuvo activo. De existir algún error el sistema muestra un mensaje de notificación.

```
    $this->fechadeactivacion = "";
    $this->fechadedesactivacion = "";
    for ($index = 0; $index < $count; $index++) {
        $this->fechadeactivacion = $fechas[$index]['fecha'];
        $this->fechadedesactivacion = $fechas[$index + 1]['fecha'];
        if ($this->createsql($idevent, $idp) == false)
            return FALSE;
        $index+=1;
    }
    return TRUE;
}
```

Figura 23. Fragmento del código precreatesql($idevent, $idp)(2).

3.1.3. Diagrama de despliegue

El diagrama de despliegue se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes (nodos). Los nodos son objetos físicos que existen en tiempo de ejecución y que representan algún tipo de recurso computacional, también pueden ser dispositivos del sistema.

Figura 24. Diagrama de despliegue.

3.2. Estándares de codificación

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

Para el desarrollo del componente administración de conexiones se utilizarán algunos de los estándares de codificación y normas propuestos como parte de la Línea de arquitectura determinada para el desarrollo del ERP Cuba (Pérez, 2012).

Los estándares utilizados en la codificación fueron los siguientes:

- **Notación PascalCasing:** En este caso los identificadores y nombres de las variables, métodos y funciones están compuestos por múltiples palabras juntas, iniciando cada palabra con letra mayúscula.
- **Notación CamelCasing:** Es parecido al PascalCasing con la excepción que la letra inicial del identificador no debe estar en mayúscula.

Nomenclatura de las clases

Los nombres de las clases comienzan con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación PascalCasing. Con sólo leerlo se reconoce el propósito de la misma.

Ejemplo: DatConnection.

Nomenclatura según el tipo de clases

1. **Clase controladora.**

Las clases controladoras después del nombre llevan la palabra: “Controller”.
Ejemplo: processaministrationController.

2. **Clases de los modelos.**

- **Business (Negocio).**

Las clases que se encuentran dentro de Business después del nombre llevan la palabra: “Model”.
Ejemplo: DatProcessModel

- **Domain (Dominio).**

Las clases que se encuentran dentro de Domain el nombre que reciben es el de la tabla en la Base de Datos.
Ejemplo: DatProcess.

- **Generated (Dominio bases)**

Las clases que se encuentran dentro de Generated el nombre comienza con la palabra: “Base” y seguido el nombre de la tabla en la Base de Datos.
Ejemplo: BaseDatProcess.

Nomenclatura de las funciones

El nombre a emplear para las funciones se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se empleará notación CamelCasing y en caso de ser una acción de la clase controladora se debe especificar el nombre de dicha acción en minúscula y seguido del sufijo “Action”.
Ejemplo: modificargestorAction.

Nomenclatura de las variables

El nombre a emplear para las variables se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se empleará notación CamelCasing y comenzando con un prefijo según el tipo de datos.
Ejemplo: $proceso.

Normas de comentariado

Se debe comentar todo lo que se haga dentro del desarrollo, establecer las pautas que conlleven a lograr un código más legible y reutilizable y así se pueda aumentar su mantenimiento a lo largo del tiempo.

Nomenclatura de los comentarios.

Los comentarios deben ser lo bastante claros y precisos de forma tal que se entienda el propósito de lo que se está desarrollando.

Estilo del código

En la implementación cuando se escriba una sentencia en php la forma de utilizar los tabs del mismo es la siguiente:

```php
<?php
//código
?>
```

Figura 25. Estilo del código.

- **Sangría o Indexado**

La política de sangría a utilizar en la implementación es por **tab**.

Ejemplo:

```php
<?php
/**
 * Indentation
 */
class Example {
    var $theInt = 1;
    function foo($a, $b) {
    switch ( $a) {
    case 0 :
    $Other->doFoo ();
    break;
    default :
    $Other->doBaz ();
    }
    }
    function bar($v) {
    for($i = 0; $i < 10; $i ++) {
    $v->add ( $i );
    }
    }
}
?>
```

Figura 26.Estilo del código: Sangría o Indexado.

- **Brazas o llaves.**

En la declaración de clases o interfaces, métodos, bloques y switch, la apertura de llaves se hace en la misma línea.

Ejemplo:

```php
<?php
/**
 * Braces
 */
interface EmptyInterface {
}

class Example {
    function bar($p) {
        for($i = 0; $i < 10; $i ++) {
        }
        switch ( $p) {
            case 0 :
                $fField->set ( 0 );
            break;
            case 1 :
                {
                    break;
                }
            default :
                $fField->reset ();
        }
    }
}
?>
```

Figura 27. Estilo del código: Brazas o llaves.

3.3. Métricas de Software

La evaluación de un producto, mediante métricas, es un aspecto fundamental a tener en cuenta; ya que, aunque las métricas del producto el software no suelen ser absolutas, brindan la posibilidad de evaluar la calidad a partir de varias reglas definidas claramente. Permiten detectar y corregir los posibles problemas que se puedan presentar durante el proceso de desarrollo y no después de terminado el producto.

La IEEE define las métricas como: *"una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado"* (IEEE, 2007).

3.3.1. Tamaño Operacional de Clases (TOC).

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

Tabla 7. Atributos de calidad evaluados por la métrica TOC.

Atributo de Calidad	*Modo en que lo afecta*
Responsabilidad	Aumento del TOC provoca aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	Aumento del TOC provoca aumento de la complejidad de implementación de la clase.
Reutilización	Aumento del TOC provoca disminución del grado de reutilización de la clase.

Para la evaluación de dichos atributos de calidad, se definieron los siguientes criterios y categorías de evaluación:

Tabla 8. Criterios de evaluación de la métrica TOC.

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

	Alta	<=Promedio

Resultados obtenidos de la aplicación de la métrica TOC.

Evaluando la cantidad de operaciones de cada una de las clases con que cuenta el Módulo para el registro y transformación de trazas de eventos a formato XES, según los criterios establecidos en la tabla 8. Se obtuvo que un 67% de las clases cuentan con Responsabilidad y Complejidad: baja y Reutilización: alta. Los resultados obtenidos de estas pruebas, sugieren que el uso de patrones tuvo un impacto moderado en la calidad del diseño de la solución.

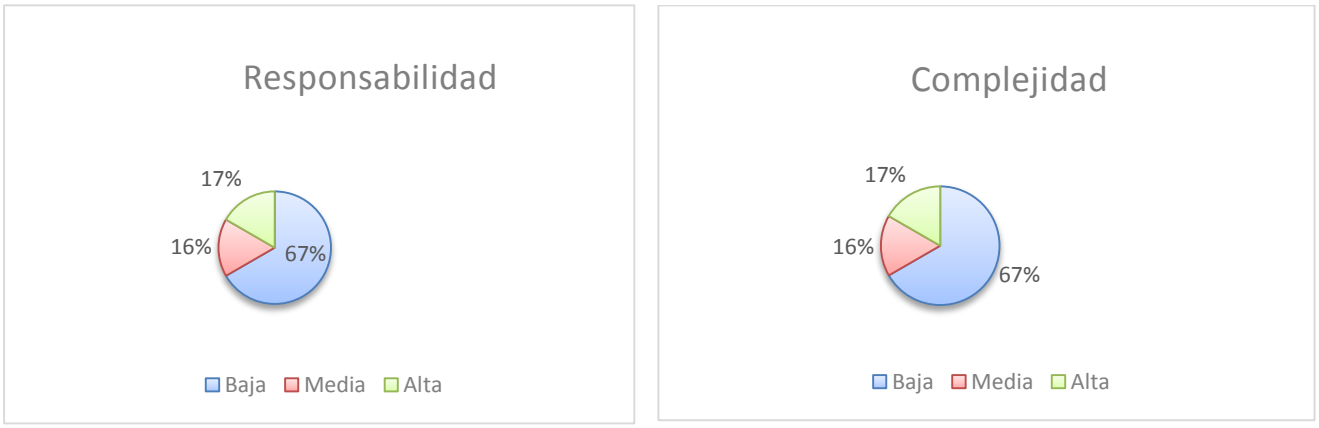


Figura 28. Resultados de la evaluación de la métrica TOC para los atributos Responsabilidad y Complejidad.

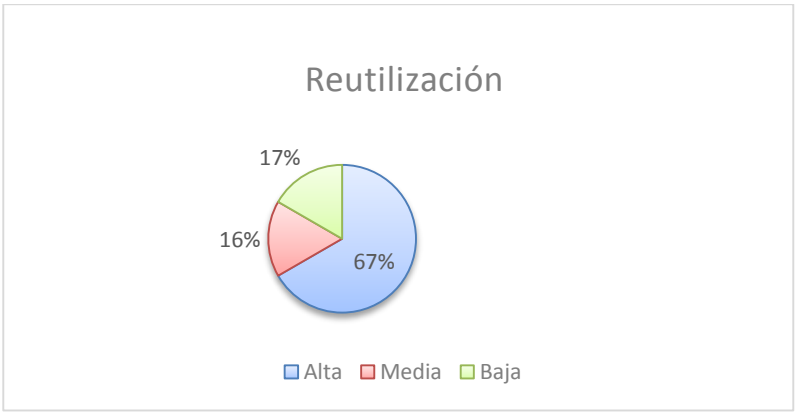


Figura 29. Resultados de la evaluación de la métrica TOC para el atributo de calidad Reutilización.

En el siguiente instrumento de evaluación de la métrica TOC se puede observar con un mayor nivel de detalle la aplicación de la métrica.

Tabla 9. Instrumento de evaluación de la métrica Relaciones entre clases.

Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización

ConnectionManagerController	21	Alta	Alta	Baja
ProcessDefinitionController	36	Alta	Alta	Baja
ProcessadministrationController	16	Alta	Alta	Baja
ProcessconfigurationController	9	Media	Media	Media
DatProcessModel	3	Baja	Baja	Alta
DatProcessRegistryModel	3	Baja	Baja	Alta
DatEventModel	3	Baja	Baja	Alta
DatConnectionModel	3	Baja	Baja	Alta
HisProcesoModel	3	Baja	Baja	Alta
HisTrazaModel	3	Baja	Baja	Alta
HistoryModel	3	Baja	Baja	Alta
DatEvent	11	Media	Media	Media
DatProcess	10	Media	Media	Media
DatProcessRegistry	3	Baja	Baja	Alta
DatConnection	**3**	Baja	Baja	Alta
HisProceso	**3**	Baja	Baja	Alta
HisTraza	**3**	Baja	Baja	Alta
History	**1**	Baja	Baja	Alta

3.3.2. Relación entre Clases (RC)

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

Tabla 10. Atributos de calidad evaluados por la métrica Relaciones entre clases.

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

Para la evaluación de dichos atributos de calidad, se definieron los siguientes criterios y categorías de evaluación:

Tabla 11. Criterios de evaluación para la métrica RC.

Atributo	*Categoría*	*Criterio*
Acoplamiento	Ninguno	0
	Baja	1
	Media	2
	Alta	>2
Complejidad de mantenimiento	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Reutilización	Baja	>2*Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	<=Promedio
Cantidad de pruebas	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio

Resultados obtenidos de la aplicación de la métrica RC

Evaluando la cantidad de relaciones entre clases con que cuenta el Módulo para el registro y transformación de trazas de eventos a formato XES, según los criterios establecidos en la Tabla 11, se obtuvo que un 67% de las clases cuentan con baja Complejidad de mantenimiento, Cantidad de pruebas y alta Reutilización. Por otra parte un 45% de las clases cuentan con bajo Acoplamiento.

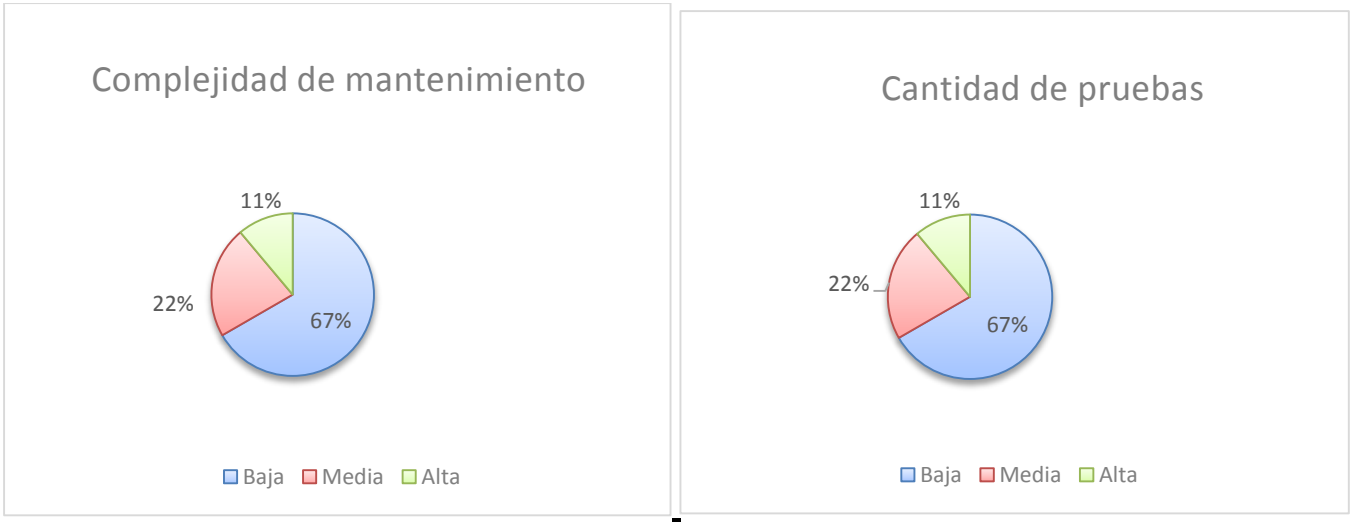


Figura 30. Resultados de la evaluación de los atributos de la métrica RC.

Tabla 12 Atributos de calidad evaluados por la métrica Relaciones entre clases.

Clase	Cantidad de relaciones de uso	Acoplamiento	Complejidad Mant.	Reutilización	Cantidad de pruebas
ConnectionManagerController	3	Alta	Media	Media	Media
ProcessDefinitionController	7	Alta	Alta	Baja	Alta
ProcessadministrationController	5	Alta	Alta	Baja	Alta
ProcessconfigurationController	3	Alta	Media	Media	Media
DatProcessModel	2	Media	Baja	Alta	Baja
DatProcessRegistryModel	2	Media	Baja	Alta	Baja
DatEventModel	2	Media	Baja	Alta	Baja
DatConnectionModel	1	Baja	Baja	Alta	Baja
HisProcesoModel	1	Baja	Baja	Alta	Baja
HisTrazaModel	1	Baja	Baja	Alta	Baja
HistoryModel	1	Baja	Baja	Alta	Baja
DatEvent	2	Media	Baja	Alta	Baja
DatProcess	4	Alta	Media	Media	Media
DatProcessRegistry	4	Alta	Media	Media	Media
DatConnection	1	Baja	Baja	Alta	Baja
HisProceso	1	Baja	Baja	Alta	Baja

HisTraza	1	Baja	Baja	Alta	Baja

3.4. Pruebas de caja blanca

Las pruebas de caja blanca 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 (Figura 32) asociado y se calcula su complejidad ciclomática. Para poder elaborar el Grafo de Flujo, primero se deben enumerar las sentencias del código:

```
function actprocessAction() {
    $id = $this->getRequest()->getPost('id');   1
            $proceso = DatProcess::activarProceso($id);
                    if($proceso->modificarversion==1){  2
                    $proceso->version=($proceso->version+1); 3
                    $proceso->modificarversion=0;
                    $m = new DatProcessModel();
                    $m->Modificar($proceso);
                        }
    $validado = DatProcess::getvalidado($id); 4
    $activado = DatProcess::getactivado($id);
    $activado = $activado[0]["activado"];
    $validado = $validado[0]["validado"];
    if ($validado == 1 && $activado == 0) { 5
        $this->actTrazaDatos(); 6
        $tablasProceso = DatProcess::gettables($id);
        $tablasProceso = $tablasProceso[0]['tablas'];
        $tablasHistorial = $this->_history->gethistorial();
        $TProcesoH = array();
        $tablasProceso = str_replace('{', '', $tablasProceso);
        $tablasProceso = str_replace('}', '', $tablasProceso);
        $tablasProceso = split(',', $tablasProceso);
        $cont = 0;
        $aCrear = array();
        foreach ($tablasHistorial as $value) { 7
            $TProcesoH[$cont] = $value['esquema'] . "." . $value['tabla']; 8
```

```
            $cont++;
        }
        $diferencia = array_diff($tablasProceso, $TProcesoH);9
        $cont = 0;
        foreach ($diferencia as $value => $key) {10
            $aCrear[$cont] = split('\.', $key);11
            $cont++;
        }
        $cont = 0;12
        $Parahistorial = array();
        foreach ($aCrear as $key) {13
            $Parahistorial[$cont]->table_name = $key[1];14
            $Parahistorial[$cont]->table_schema = $key[0];
            $cont++;
        }
        $user = $this->global->Perfil->usuario;15
        $this->_history->CreateHistorial($Parahistorial, $user);
        $this->activarRegistroProceso($id);
        echo json_encode(0);
    }
    else {
        if ($validado == 0)16
            echo json_encode(1);17
        else if ($activado == 1)18
            echo json_encode(2);19
    }20
}
```

Figura 31. Código fuente de la funcionalidad Activar proceso.

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

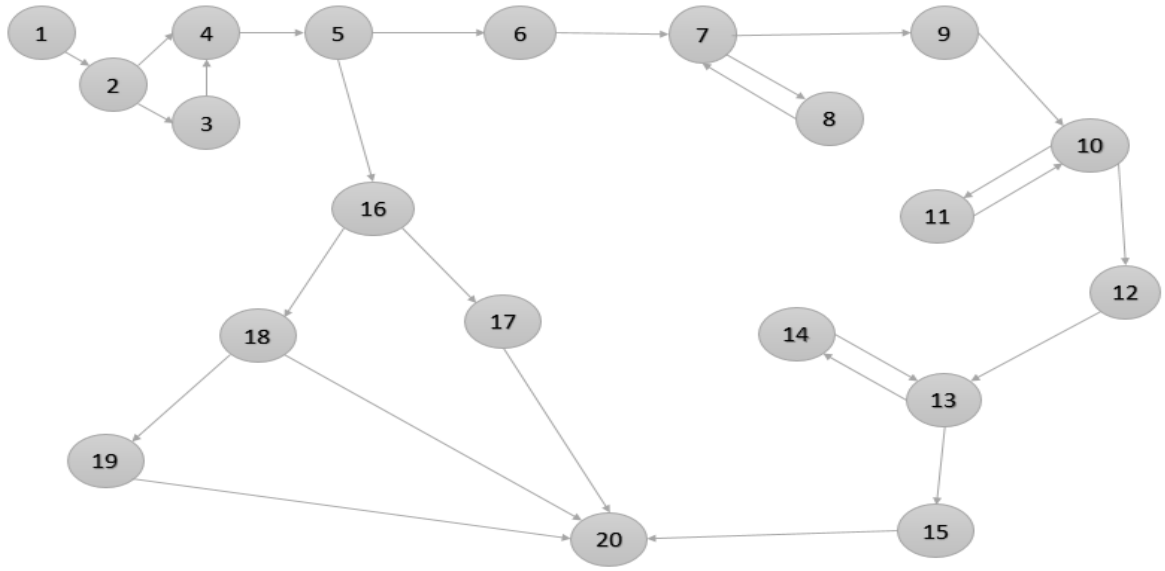


Figura 32 Grafo de flujo asociado a la funcionalidad Activar proceso.

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 tres 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) = 8
2. **V (G)** = **E** – **N** + 2 Donde **E** es el número de aristas y **N** número de nodos.
   V (G) = 26 – 20 + 2
   V (G) = 8
3. **V (G)** = **P** + 1 Donde **P** es el número de nodos predicados.
   V (G) = 7 + 1
   V (G) = 8

Teniendo en cuenta el valor obtenido de la complejidad de la funcionalidad Activar proceso, 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)** = 8, por lo que los posibles caminos básicos son:

- Camino básico Nº 1: 1, 2, 4, 5, 16, 17,20.
- Camino básico Nº 2: 1, 2, 4, 5, 16, 18,20.
- Camino básico Nº 3: 1, 2, 3, 4, 5, 16, 17,20.
- Camino básico Nº 4: 1, 2, 4, 5, 16, 18, 19,20.
- Camino básico Nº 5: 1,2,4,5,6,7,9,10,12,13,15,20.
- Camino básico Nº 6: 1,2,4,5,6,7,8,7,9,10,12,13,15,20.
- Camino básico Nº 7: 1,2,4,5,6.7.9,10,11,10,12,13,15,20.
- Camino básico Nº 8: 1,2,4,5,6.7.9,10,12,13,14,13,15,20.

Derivación de casos de prueba para la funcionalidad: **activateprocessAction.**
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, 2, 4, 5, 16, 17, 20).
Si $proceso->modificarversion es diferente de 1, $validado = 0.

Caso de prueba para camino básico Nº 2 (1, 2, 4, 5, 16, 18, 20).
Si $proceso->modificarversion es diferente de 1, $validado =1 y $activado >1.

Caso de prueba para camino básico Nº 3 (1, 2, 3, 4, 5, 16, 17, 20).
Si $proceso->modificarversion = 1, $validado =0.

Caso de prueba para camino básico Nº 4 (1, 2, 4, 5, 16, 18, 19, 20).
Si $proceso->modificarversion es diferente de 1, $activado =1.

Caso de prueba para camino básico Nº 5 (1, 2, 4, 5, 6, 7, 9, 10, 12, 13, 15, 20).
Si $proceso->modificarversion es diferente de 1, $activado=0, $validado =1, $tablasHistorial esté vacío, $diferencia esté vacío y $acrear esté vacío.

Caso de prueba para camino básico Nº 6 (1, 2, 4, 5, 6, 7, 8, 7, 9, 10, 12, 13, 15, 20).
Si $proceso->modificarversion es diferente de 1, $activado=0, $validado =1, $tablashistorial tenga elementos, $diferencia esté vacío y $acrear esté vacío.

Caso de prueba para camino básico Nº 7 (1, 2, 4, 5, 6. 7. 9, 10, 11, 10, 12, 13, 15, 20).
Si $proceso->modificarversion es diferente de 1, $activado=0, $validado =1, $tablasHistorial esté vacío, $diferencia tenga elementos y $acrear esté vacío.

Caso de prueba para camino básico Nº 8 (1, 2, 4, 5, 6. 7. 9, 10, 12, 13, 14, 13, 15, 20).
Si $proceso->modificarversion es diferente de 1, $activado=0, $validado =1, $tablasHistorial esté vacío, $diferencia esté vacío y $acrear tenga elementos.

3.5. Pruebas de caja negra

A los módulos de la solución se le realizaron pruebas funcionales o de caja negra. Estas pruebas permiten obtener conjuntos de entrada que ejerciten los requisitos funcionales del software, complementándose con las pruebas de caja blanca, obteniendo errores en las siguientes categorías (Pressman, 2005):

- Funciones incorrectas o inexistentes.
- Errores en la interfaz.
- Errores en la estructura de datos.
- Rendimiento.

- Inicialización y terminación.

En el presente trabajo se utilizará la técnica de partición de equivalencia pues esta técnica es una de las más efectivas, permite examinar los valores válidos e inválidos de las entradas existentes en el software. Además, descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de pruebas que descubran clases de errores, reduciendo así el número de clases de prueba que hay que desarrollar.

Para aplicar esta técnica se confeccionaron los diseños de caso de prueba para cada una de las funcionalidades del componente los cuales pueden ser consultados en el Expediente de Proyecto de Sauxe. Para comprobar la calidad de los módulos para el registro y transformación de trazas de eventos a formato XES se realizaron dos iteraciones de revisiones por el Departamento de Tecnología a la aplicación.

Resultados de las pruebas

En la siguiente tabla (ver Tabla 12) se recogen los datos de las no conformidades detectadas en cada una de las pruebas realizadas por el Departamento de Calidad en la documentación y la aplicación.

Tabla 13. Tabla de las No conformidades detectadas en la documentación y la aplicación por iteraciones. Fuente: Elaboración propia.

Tipo de no Conformidades	*Documentación*	*Aplicación*
Primera Iteración		
Significativas	0	2
No significativas	5	25
Total	**5**	**27**
Segunda Iteración		
Significativas	0	0
No significativas	0	0
Total	**0**	**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.

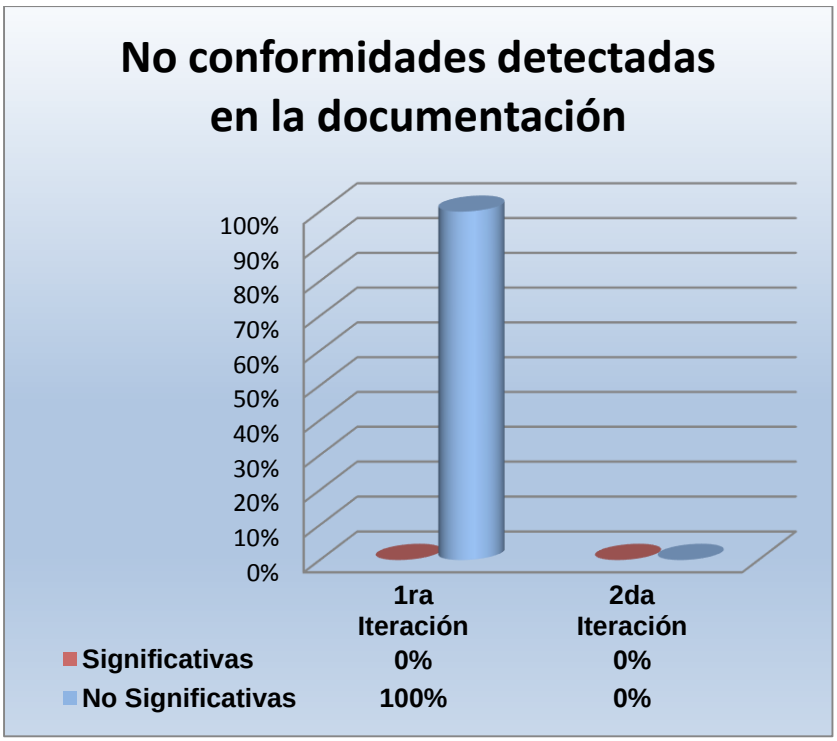


Figura 33. Por ciento que representan las no conformidades del total en la documentación por cada una de las iteraciones.

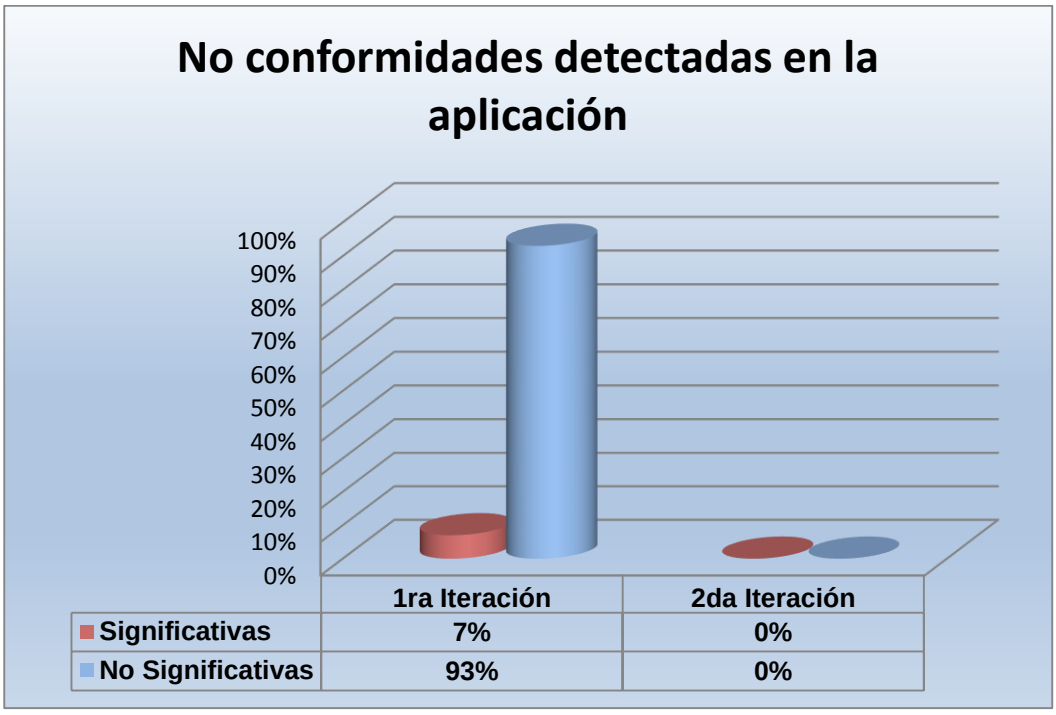


Figura 34. Por ciento que representan las no conformidades del total en la aplicación por cada una de las iteraciones.

3.6. Aplicación en caso de estudio

Sauxe posee un módulo para la gestión de trazas el cual captura trazas de acción, integración, rendimiento, datos, excepciones y excepciones de integración. Ninguno de los tipos de trazas de Sauxe están orientados a procesos lo que dificulta la identificación de un caso o instancia de un proceso. Sauxe posee una funcionalidad para la exportación de trazas a formato XML. Esta funcionalidad no está sujeta a ningún estándar XML, lo que imposibilita el uso de las trazas exportadas para estudios en Minería de Proceso. Por estas razones antes de la creación del Módulo para el registro y transformación de trazas de eventos a formato XES, no era posible realizar estudios de Minería de Proceso a las trazas de Sauxe.

El caso de estudio consiste en la prueba del **Módulo para el registro y transformación de trazas de eventos a formato XES como solución al problema científico de la investigación.**

El módulo Definir proceso, permite adicionar, modificar y eliminar tanto eventos como procesos del sistema. Cabe notar que los eventos están asociados siempre a un proceso.

Mediante este sencillo caso de estudio la investigación se propone probar la utilidad del registro de eventos exportado al formato XES en la herramienta más usada del área de investigación de Minería de Proceso **ProM 6.2** (véase Figura 35)**.**

Figura 35. ProM 6.2.

Con la herramienta **ProM 6.2** se creó un modelo de proceso utilizando las trazas de proceso del módulo de definición de proceso generadas por el **Módulo para el registro y transformación de trazas de eventos a formato XES**.

En la Figura 36 se muestra una vista del módulo de definición de proceso.

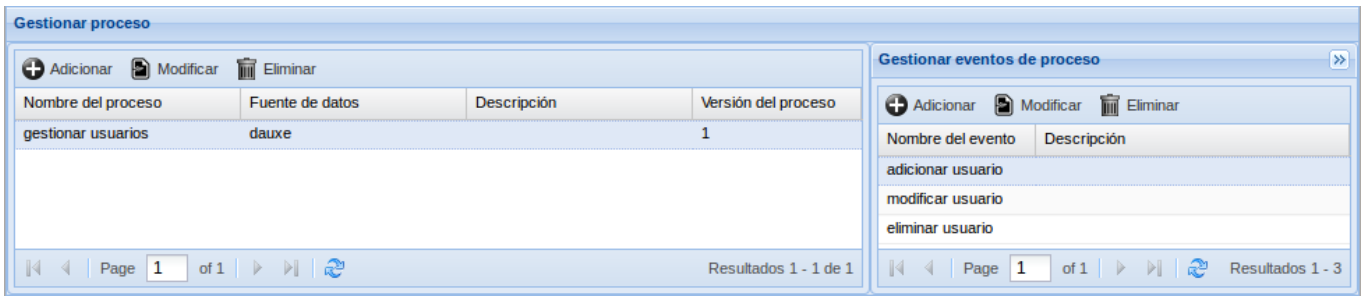


Figura 36. Módulo de definición de proceso.

3.6.1. Situación después de la existencia del Módulo para el registro y transformación de trazas de eventos a formato XES

Una vez definido, configurado, validado y activado el proceso, se procedió a un período de pruebas donde un total de cinco usuarios interactuaron con el módulo de definición de proceso creando, modificando y eliminando procesos y eventos a discreción. En el momento en que culminó la etapa de pruebas, se generaron trazas de proceso y las mismas fueron exportadas a un registro de eventos en el formato XES.

Este registro de eventos fue importado por la herramienta ProM 6.2. El hecho de que la herramienta ProM importara satisfactoriamente el registro de eventos en formato XES, implica que dicho registro posee una estructura sintácticamente correcta. Una vez importado, se aplicó el algoritmo “GeneticMiner”. Como resultado de esta aplicación se obtuvo el modelo de proceso asociado a las trazas del registro de eventos generadas por la solución. Al generarse el modelo de proceso satisfactoriamente, se demuestra la calidad de las trazas de procesos contenidas en el registro de eventos importado. Este modelo de proceso se muestra a continuación en la Figura 37.

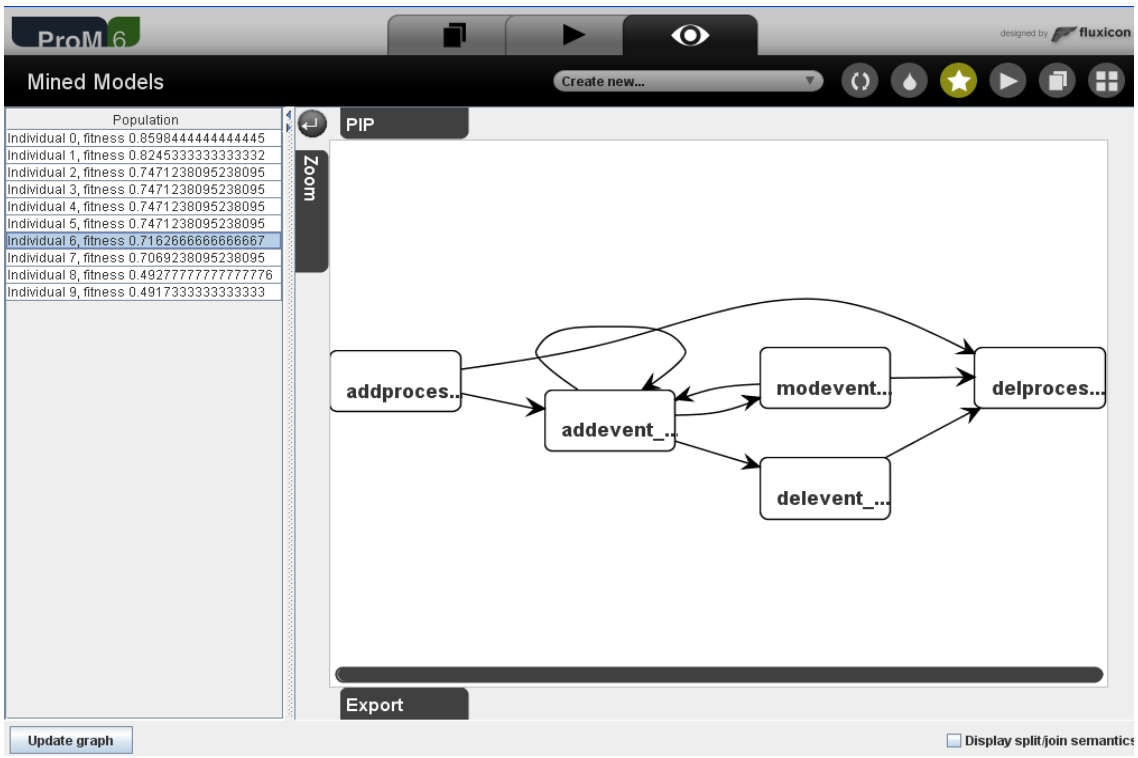


Figura 37. Modelo de proceso del módulo de definir proceso.

3.6.2. Conclusiones de la validación del módulo mediante un caso de estudio

Como se observó en la Figura 37**,** una vez terminado el módulo, se exportó un registro de eventos en formato XES. Este registro de eventos al ser importado por la herramienta ProM 6.2 generó un modelo de proceso el cual muestra las ejecuciones de 10 instancias de procesos del módulo Definir proceso organizadas cronológicamente y con los eventos ocurridos por cada instancia de proceso.

Conclusiones del capítulo

La validación realizada al diseño propuesto en el capítulo anterior mediante el empleo de métricas permitió demostrar que el componente cuenta con un diseño robusto.

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

Las pruebas de caja blanca y caja negra realizadas a la solución demostraron que esta respondía correctamente a los requisitos identificados a partir de las necesidades del cliente. Al finalizar estas pruebas el Departamento de Tecnología emitió un aval que certifica estos resultados.

El uso de un algoritmo de Minería de Proceso logró validar el objetivo de la investigación y así certificar la calidad del registro de eventos exportado por el Módulo para el registro y transformación de trazas de eventos a formato XES.

Conclusiones generales

Los sistemas estudiados para extracción de trazas de proceso al depender de la evidencia de la ejecución de procesos se les dificulta la completitud de las trazas debido a la variación de los datos.

A partir del estudio realizado de las trazas generadas por Sauxe se detectó que las mismas no contienen la información necesaria para realizar estudios utilizando técnicas de Minería de Proceso.

Esta solución informática garantiza que los datos persistan en el tiempo a partir de los módulos implementados para la configuración del registro de las trazas de proceso. Esto asegura la transformación de las trazas registradas por el marco de trabajo Sauxe a trazas de proceso y su exportación a formato XES.

La validación realizada al diseño propuesto en el capítulo dos mediante el empleo de métricas permitió demostrar que el módulo cuenta con un diseño robusto.

Las pruebas de caja blanca y caja negra realizadas a la solución demostraron que esta respondía correctamente a los requisitos identificados a partir de las necesidades del cliente. Al finalizar estas pruebas el Departamento de Tecnología emitió un aval que certifica estos resultados.

La validación del objetivo de la investigación mediante un caso de estudio permitió utilizar un algoritmo de Minería de Proceso y así certificar que el registro de eventos exportado por el Módulo para el registro y transformación de trazas de eventos a formato XES estaba estructurado correctamente y contenía la información necesaria para realizar estudios de Minería de Proceso.

RECOMENDACIONES

- Mejorar el tratamiento de las condiciones en la configuración de los eventos de un proceso.
- Desarrollar una configuración para la generación periódica de las trazas de proceso de manera automática.

BIBLIOGRAFÍA

1-Aida Sacerio Martínez, 2010. *Análisis, diseño e implementación del Framework de registro de eventos y su herramienta de seguimiento.* S.l.: s.n.

2-ANON, 2011. Industry Leading JavaScript Framework for Building Desktop Web Apps | Sencha Ext JS | Products | Sencha. En: [online]. [Accedido 29 enero 2013 a]. Disponible desde: http://www.sencha.com/products/extjs.

3-ANON, 2012. Introduction — Doctrine 1.2.4 documentation. En: [online]. [Accedido 29 enero 2013 b]. Disponible desde: http://docs.doctrine-project.org/projects/doctrine1/en/latest/en/manual/introduction.html#what-is-doctrine.

4-ANON, 2013. PHP: Hypertext Preprocessor. En: [online]. [Accedido 29 enero 2013 c]. Disponible desde: http://php.net/.

5-ANON, 2010. PostgreSQL: The world's most advanced open source database. En: [online]. [Accedido 29 enero 2013 d]. Disponible desde: http://www.postgresql.org/.

5- Günther Christian w., 2009. *XES Extensible Event Stream standard definition*. S.l.: s.n.

6-Dumas, Marlon, AALST, Wil M. van der y HOFSTEDE, Arthur H. ter, 2005. *Process Aware Information Systems: Bridging People and Software Through Process Technology*. 1. S.l.: Wiley-Interscience. ISBN 0471663069.

7-IEEE Task Force On Process Mining, 2011. *Manifiesto sobre Minería de Proceso* [online]. S.l.: s.n. Disponible desde: http://www.win.tue.nl/ieeetfpm/lib/exe/fetch.php?media=shared:pmm-spanish-v1.pdf.

8-ISO, 2000. *Norma Internacional ISO 9000-2000*. 2000. S.l.: s.n.

9-J.C.A.M. Buijs, 2010. *Mapping Data Sources to XES in a Generic Way* [online]. Master Thesis. Eindhoven, The Netherlands: Technische Universiteit Eindhoven. [Accedido 16 marzo 2012]. Disponible desde: http://www.processmining.org/_media/xesame/xesma_thesis_final.pdf.

10-Microsoft, 2004. *Guía de estratégica de business intelligence*. 2004. S.l.: s.n.

11- Baryolo Oiner Gómez y Nemuris Silega Martínez, 2008. *Plantilla Registro de la Propiedad Intelectual*. Digital. S.l.: Universidad de las Ciencias Informáticas.

12-OMG, 2006. *Business Process Modeling Notation Specification*. 2006. S.l.: s.n.

13-Oracle Corporation And Its Affiliates, 2011. NetBeans IDE - Features. En: *NetBeans IDE* [online]. 2011. [Accedido 5 marzo 2012]. Disponible desde: http://netbeans.org/features/index.html.

14-The Apache Software Foundation, 2011. Apache. En: *The Apache HTTP Server Project* [online]. 2011. [Accedido 16 enero 2012]. Disponible desde: http://httpd.apache.org/.

15-Van Der Aalst, W. M. P., 2011. *Process Mining: Discovery, Conformance and Enhancement of Business Processes* [online]. S.l.: Springer. [Accedido 29 enero 2013]. Disponible desde:

Bibliografía

http://www.google.com/books?hl=en&lr=&id=I1KOAfiqfxYC&oi=fnd&pg=PR4&dq=Discovery,+Conformance+and+Enhancement+of+Business+Processes&ots=K-kS69vVBL&sig=WLc3SuDDHQYJbX4dZpraXFzsxGI.

16-Van Der Heijden, THC, 2012. *Process Mining Project Methodology: Developing a General Approach to Apply Process Mining in Practice* [online]. Master of Science in Operations Management and Logistics. Netherlands: TUE. School of Industrial Engineering. [Accedido 20 octubre 2012]. Disponible desde: http://alexandria.tue.nl/extra2/afstversl/tm/Van_der_Heijden_2012.pdf.

17-VISUAL PARADIMG, 2011. Visual modeling tool for building enterprise applications. En: *Visual Paradimg website* [online]. 2011. [Accedido 5 marzo 2012]. Disponible desde: http://www.visual-paradigm.com/product/vpuml/provides/.

18-Pressman, Roger S., 2002. Ingeniería del software. McGrawHill. : s.n., 2002.

19- Kaisler, Stephen H., 2005. Software Paradigms. New Jersey: John Wiley & Sons, Inc.

20- Acuña, K. B., 2013. eumed.net. Retrieved 04 30, 2013, from Brito Acuña, Karenny. eumed.net. [En línea] [Citado el: 9 de Abril de 2012.] http://www.eumed.net/libros/2009c/584/RUP%20Diseno%20e%20implementacion%20del%20sistema.htm.

21- Pérez Alfonso, Damián, 2012. Normas y estándares de codificación. Universidad de las Ciencias Informáticas. 2012. Normas y estándares de Codificación de Sauxe.

22- IEEE, 2007. Introducción a la Ingeniería de Requisitos. Ingeniería de software. [Digital] 2007.

23- Pressman, Roger, 2005. Ingeniería Software. Un enfoque práctico. La Habana: Félix Varela, 2005.

24-Baryolo, Oiner Gómez, 2010. Solución Informática de Autorización. La Habana: Universidad de las Ciencias Informáticas, 2010.

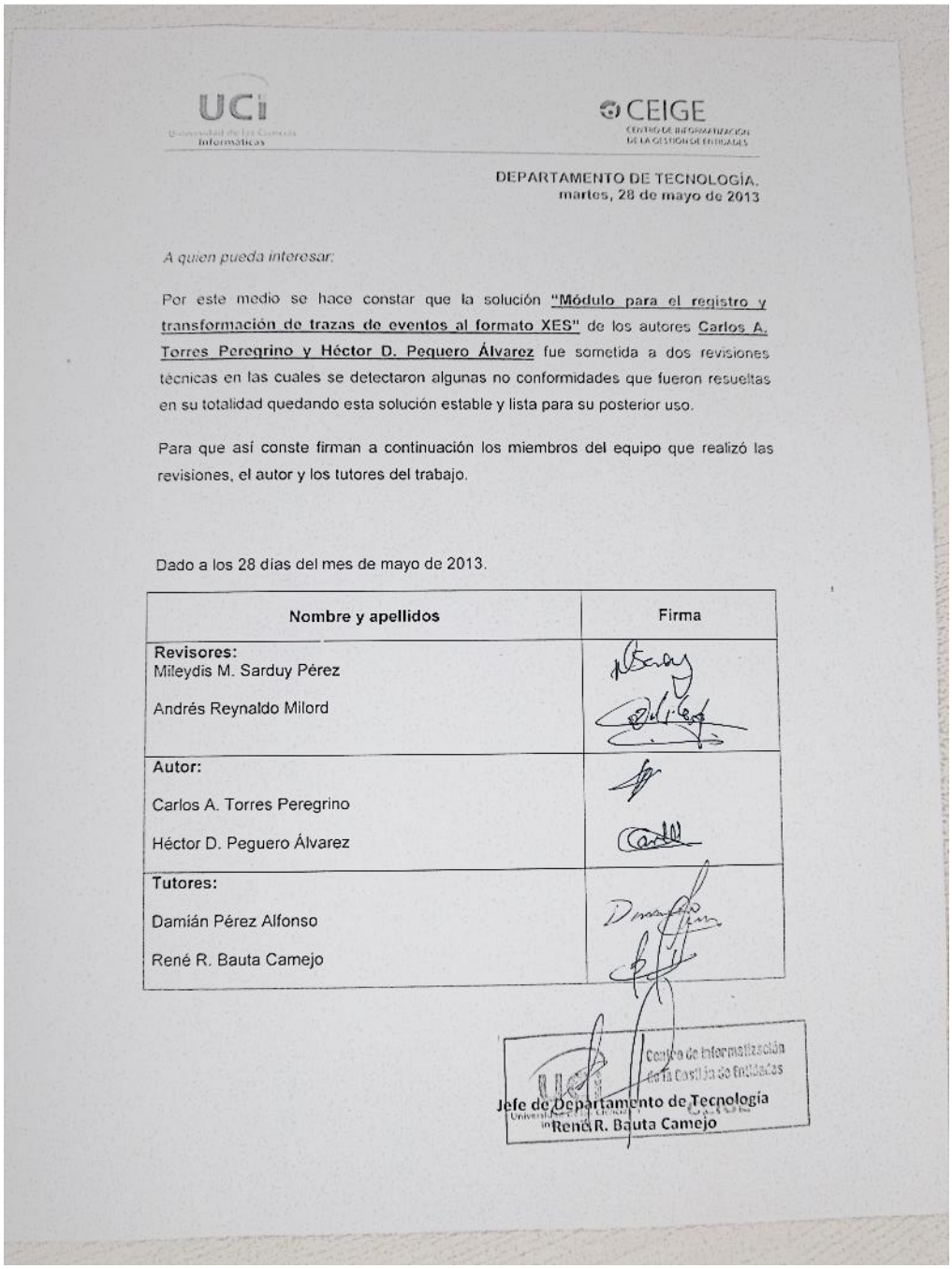
25- Rodríguez Carlos,Engel Robert, Kostoska Galena, Florian Daniel, Casati Fabio y Aimar Marco, 2012. Eventifier: Extracting Process Execution Logs from Operational Databases

26-Adriansyah, A. y Buijs, J., 2012. Mining Process Performance from Event Logs. The BPI Challenge 2012 Case Study. En: *BPM Conference* [online]. S.l.: s.n. 2012.

27-Bose, R. P. J. C. y Van Der Aalst, W. M. P., 2012. *Process Mining Applied to the BPI Challenge 2012: Divide and Conquer While Discerning Resources*. S.l. Technical Report BPM Center BPM-12-16, bpmcenter. org.

28-Caron, Filip, Vanthienen, Jan y Baesens, Bart, 2012. Rule-Based Business Process Mining: Applications for Management. S.l.: Springer Berlin Heidelberg. Advances in Intelligent Systems and Computing. pp. 273-282.

ANEXOS



UCI
Universidad de las Ciencias Informáticas

CEIGE
CENTRO DE INFORMATIZACIÓN DE LA GESTIÓN DE ENTIDADES

DEPARTAMENTO DE TECNOLOGÍA.
martes, 28 de mayo de 2013

A quien pueda interesar:

Por este medio se hace constar que la solución **"Módulo para el registro y transformación de trazas de eventos al formato XES"** de los autores **Carlos A. Torres Peregrino y Héctor D. Peguero Álvarez** fue sometida a dos revisiones técnicas en las cuales se detectaron algunas no conformidades que fueron resueltas en su totalidad quedando esta solución estable y lista para su posterior uso.

Para que así conste firman a continuación los miembros del equipo que realizó las revisiones, el autor y los tutores del trabajo.

Dado a los 28 días del mes de mayo de 2013.

Nombre y apellidos	Firma
Revisores: Mileydis M. Sarduy Pérez Andrés Reynaldo Milord	
Autor: Carlos A. Torres Peregrino Héctor D. Peguero Álvarez	
Tutores: Damián Pérez Alfonso René R. Bauta Camejo	

Centro de Informatización de la Gestión de Entidades
Jefe de Departamento de Tecnología
René R. Bauta Camejo

Peña Tecnológica
... donde convergen personas y tecnologías

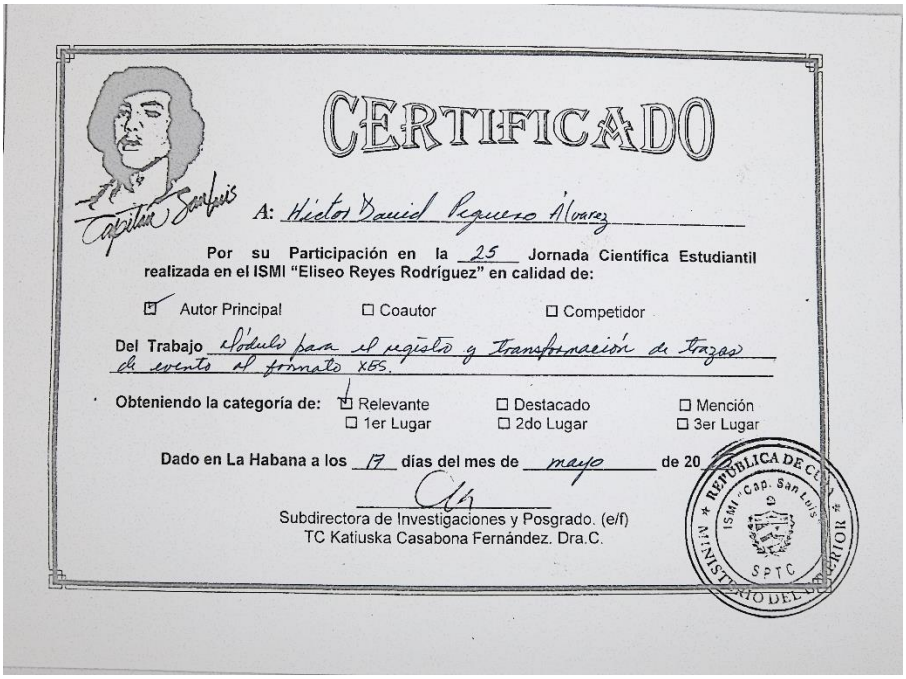
CERTIFICADO

A: Módulo para el registro y transformación de trazas de evento al formato XES

Por: Haber obtenido la categoría DESTACADO

Entregado en la Universidad de la Ciencias Informáticas, La Habana, a los 10 días del mes de abril del año 2013.

Ing. Héctor Ochil Pérez
Presidente del Comité Organizador del Evento
Presidente del Consejo Científico DTS-MININT



CERTIFICADO

A: Héctor Daniel Peguero Álvarez

Por su Participación en la 25 Jornada Científica Estudiantil realizada en el ISMI "Eliseo Reyes Rodríguez" en calidad de:

☑ Autor Principal ☐ Coautor ☐ Competidor

Del Trabajo Módulo para el registro y transformación de trazas de evento al formato XES.

Obteniendo la categoría de: ☑ Relevante ☐ Destacado ☐ Mención
☐ 1er Lugar ☐ 2do Lugar ☐ 3er Lugar

Dado en La Habana a los 17 días del mes de mayo de 20

Subdirectora de Investigaciones y Posgrado. (e/f)
TC Katiuska Casabona Fernández. Dra.C.

REPÚBLICA DE CUBA * ISMI - Cap. San Luis * SPTC * MINISTERIO DEL INTERIOR