UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS (UCI)

Módulo de Odoo para la extracción y transformación de trazas de registro de eventos para la Minería de Procesos

Autor:

Luis Ángel Mojena Román

Tutores:

Ing. Abraham Calás Torres

Ing. Leduán Flores Riera

Ing. Yaniris Blanco Zamora

Tesis presentada en opción al título de
Ingeniero en Ciencias Informáticas

18 de junio de 2018

DECLARACIÓN JURADA DE AUTORÍA

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

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

Autor: _____ Tutor: _____

“Any fool can write code that a computer can understand.
Good programmers write code that humans can understand.”
Martin Fowler, 2008.

RESUMEN

Odoo es una plataforma para el desarrollo y puesta en marcha de Sistemas de Planificación de Recursos Empresariales. Estos tipos de sistemas generalmente almacenan trazas de las acciones ejecutadas por los usuarios, para obtener información útil de las trazas se pueden aplicar técnicas y herramientas de Minería de procesos. La premisa para aplicar esta disciplina es la confección de un registro de eventos, tarea que puede complejizarse en dependencia del tipo de sistema, como en el caso de Odoo que está enfocado hacia los datos. Por tanto, el objetivo de la investigación es obtener un registro de eventos factible para las herramientas de Minería de Procesos, a partir de las trazas que brinda Odoo. Se realizó un estudio para determinar los eventos que componen el registro, así como la información que deben contener los eventos. Como resultado se obtiene un módulo para Odoo del cual se derivan: el mecanismo para la captura de eventos, la exportación del registro de eventos, los algoritmos y la estructura de datos para el trabajo con las relaciones de las entidades. El registro se exporta al estándar XES, reconocido por las principales técnicas y herramientas para realizar análisis de Minería de procesos. Con esta solución los sistemas construidos sobre Odoo contarían con un instrumento para apoyar el proceso de toma de decisiones en sus empresas.

Palabras clave: evento, minería, módulo, Odoo, proceso, registro

ABSTRACT

Odoo is a platform for the development and implementation of Enterprise Resource Planning Systems. These types of systems usually store traces of the actions executed by the users. To obtain useful information from the traces can be applied techniques and tools of Process Mining. The premise to apply this discipline is the preparation of an event log, a task that can be more complex depending on the type of system, as in the case of Odoo, which is a data-aware system. Therefore, the goal of the research is to obtain a feasible event log from the traces provided by Odoo. A study was carried out to determine the events that constitute the registry, as well as the attributes that events must contain. As results we have a module for the Odoo platform from which derive the mechanism for capturing and exporting event logs, the algorithms and the data structurefor working with entities relationships. The event log is exported to the XES standard, recognized by the main techniques and tools to perform process mining analysis. With this solution, the systems built on Odoo would have an instrument to support the decision-making process on their enterprises.

Keywords: event, log, mining, module, Odoo, process

Dedicatoria

Le dedico los resultados y el esfuerzo de esta investigación a mis padres, a mi hermano, a mis sobrinos y a mi novia. En general a toda mi familia, que es lo más valioso que tengo en esta vida.

Agradecimientos

Quisiera agradecer:

A mi madre, gracias por la ternura, el amor y el cuidado que sólo una madre devota sabe dar. Gracias por las noches sin dormir cuando enfermo, por la preocupación, por el trabajo que dí cuando era pequeño. Y por tu amor incondicional y siempre eterno.

A mi padre, gracias por enseñarme a ser hombre y a vivir por mis principios, gracias por todo el apoyo, la fé y el cariño. Sé que puedo contar con ustedes para lo que sea. Le doy gracias a Dios porque ustedes son mis padres.

También quiero agradecer:

A mi hermanito de 36 años y mis sobrinos por pintar mis días de color con sus travesuras. A veces siento que tengo 4 hermanos pequeños.

A mis abuelos por ser la sabiduría que guía mis pasos. Ya sea desde el cielo o en la tierra.

A toda mi familia que de una forma u otra sé que velan por mi bienestar y me quieren mucho.

A todos mis buenos amigos: Alejandro, Jesús, Leduan, Pavel y Rosalina que todos ellos contribuyeron con un granito o un saco de arena a mi tesis.

A mis tutores: Leduan (¿tu aquí otra vez?), Abraham y Yaniris por ayudarme a explotar mi potencial al máximo de mis capacidades.

A todo el antiguo y presente personal del Departamento de Desarrollo de Componentes, por hacerme sentir como uno más de esa pequeña familia, en especial a Claudia, Arturo, Danielito y Julio.

A todos los profesores que me han dado clases a lo largo de los años en esta universidad, gracias a todos por permitirme aprender de ustedes.

A todas las personas aquí presentes y ausentes que siempre me animaron y ayudaron a lo largo de mi carrera.

Y por último pero no menos importante, a mi novia Daymari, gracias amor por tu enorme dedicación y tu paciencia infinita.

Luis Ángel Mojena Román

Contenidos

Lista de Figuras

Índice de Algoritmos

Índice de Tablas

Abreviaturas

UCI	**U**niversidad de las **C**iencias **I**nformáticas
CEIGE	**Ce**ntro de **I**nformatización de **E**ntidades
ERP	**E**nterprise **R**esource **P**lanning (Planificación de Recursos Empresariales)
WfMSs	**W**orkflow **M**anagement **S**ystems (Sistemas de Administración de Flujos de Trabajo)
B2B	**B**usiness **T**o **B**usiness (Empresa a Empresa)
CRM	**C**lient**R**elationship **M**anagement (Administración de Relaciones con el Cliente)
BAM	**B**usiness **A**ctivity **M**onitoring (Monitoreo de la Actividad Empresarial)
BOM	**B**usiness **O**perations **M**anagement (Gestión de Operaciones Comerciales)
BPI	**B**usiness **P**rocess **I**nteligence (Inteligencia de Procesos de Negocios)
XES	e**X**tensible **E**vent **S**tream (Flujo de Eventos Extensible)
PAIS	**P**rocess **A**ware **I**nformation **S**ystem (Sistema de Información Consciente de los Procesos)
MXML	**M**agic e**X**tensible **M**arkup **L**anguage (Lenguaje de Marcado Extensible Mágico)
BBFS	**B**idirectional **B**reath **F**irst **S**earch
PHP	**P**HP **H**ypertext **P**rocessor (Procesador de Hipertexto PHP)
AUP	**A**gile **U**nified **P**rocess (Proceso Ágil Unificado)
MVC	**M**odel **V**iew **C**ontroller (Modelo Vista Controlador)
GRASP	**G**eneral **R**esponsability **A**ssigment **S**oftware **P**atterns
ORM	**O**bject **R**elational **M**apper
CRUD	**C**reate **R**etrieve **U**pdate **D**elete
CGI	**C**ommon **G**ateway **I**nterface
HTTP	**H**yper**T**ext **T**ransfer **P**rotocol (Protocolo de Transferencia de Hipertextos)

Introducción

En la actualidad los sistemas de Planificación de Recursos Empresariales(ERP) son un componente importante para el desarrollo e informatización de las empresas. Estos son sistemas de negocio que integran y coordinan los datos de la empresa combinándose en un sistema que soporta las necesidades de toda la organización. Los sistemas ERP están diseñados para mejorar todos los aspectos de los procesos claves, tales como compras, contabilidad, fabricación y ventas, tomando procesos y funciones previamente disociados y apoyados por varios sistemas heredados o sistemas de negocios antiguos, independientes y dispares, integrándolos y combinándolos [1].

Los sistemas ERP soportan y brindan apoyo a los procesos empresariales en la mayoría de sus subsistemas, dígase: capital humano, contabilidad, finanzas, compras y ventas. Dichos procesos se pueden beneficiar de la aplicación de técnicas de Minería de Procesos. La Minería de Procesos emplea los datos para extraer información relacionada a los procesos, o sea, para descubrir automáticamente un modelo de procesos mediante la observación de los eventos registrados por un sistema empresarial. El objetivo de la Minería de Procesos es descubrir, monitorizar y mejorar procesos reales (entiéndase procesos no asumidos) mediante la extracción del conocimiento de registros de eventos disponibles en los sistemas [2].

Hoy en día, muchos sistemas de información empresariales almacenan eventos relevantes en alguna forma estructurada. Por ejemplo Sistemas de Administración de Flujos de Trabajo (WfMSs) típicamente registran el inicio y final de las actividades [3]. Sistemas ERP como SAP guardan todas las transacciones, por ejemplo, usuarios llenando formularios y cambiando documentos. Sistemas Empresa-a-Empresa (B2B) registran el intercambio de mensajes con terceras partes. Los paquetes de centros de llamadas y los sistemas CRM de uso general son ejemplos de sistemas que registran las interacciones con los clientes. Estos ejemplos muestran que muchos sistemas tienen algún tipo de registro de eventos a menudo denominado historial, pista de auditoría o registro de transacciones [4–7]. Todos estos ejemplos pueden ser utilizados como base para la conformación de un registro de eventos que se usa para la aplicación de técnicas de Minería de Procesos.

Además de la disponibilidad de los registros de eventos, existe un mayor interés en el monitoreo de los procesos empresariales. La nueva política económica definida en los lineamientos del partido y el gobierno cubano, y el mayor énfasis en la gobernabilidad corporativa, están obligando a las organizaciones a seguir más de cerca sus actividades empresariales [8]. En estos momentos existe una presión constante para mejorar el rendimiento y la eficiencia de los procesos de negocio. Esto requiere más instalaciones de monitoreo de grano fino como lo ilustran nuevas áreas de investigación, por ejemplo: Monitoreo de la Actividad Empresarial (BAM), la Gestión de Operaciones Comerciales (BOM) y la Inteligencia de Procesos de Negocio (BPI) [9].

En los sistemas de información que no exportan sus eventos, se desechan las potencialidades de la Minería de Procesos. Un ejemplo de lo anterior es la habilidad para descubrir violaciones, esta característica permite verificar si la ejecución de un determinado proceso se ajusta a una guía (descripción, regla o ley) lo cuál es una

característica deseable en un sistema ERP. Esto permite que el sistema cumpla con los requisitos especificados en la guía y el flujo de trabajo se ejecute en la forma definida. Las guías pueden ser documentos legales y es obligatorio que se implementen los flujos de trabajos en la forma en que están descritos. Otras potencialidades de la Minería de Procesos que se pueden explotar mediante el registro de eventos son: descubrir desviaciones en la ejecución de los procesos, cuellos de botella y redes de interacción entre los usuarios del sistema. Además, la predicción de casos relevantes y la recomendación de acciones basadas en historial. Estos análisis no son posibles si no se tiene un mecanismo que provea las trazas de los procesos que se ejecutan en el sistema.

La calidad de un resultado de Minería de Procesos en gran medida depende de la entrada. Para que un registro de eventos posea la calidad requerida, hay que tratar los datos del registro como ciudadanos de primera clase, para esto Van der Aalst argumenta: Un registro de eventos debe ser *confiable*, *completo*, debe tener una *semántica* bien definida y debe ser *seguro* [9]. La baja calidad de estos criterios pueden conducir a problemas que deben ser analizados para determinar cómo solucionarlos o minimizar su impacto, de forma tal que el registro de eventos obtenido cumpla con los criterios de calidad.

Entre los módulos de Odoo no se encontró en el repositorio oficial uno que permita la generación de trazas de procesos que cumpla con los criterios de calidad expuestos. Odoo no está concebido para implementar modelos de procesos de forma explícita y no posee un motor de flujo de trabajo, esto provoca que se dificulte la aplicación de técnicas de Minería de Procesos en los sistemas desarrollados sobre esta plataforma.

Por lo antes expuesto se plantea el siguiente **Problema a resolver:** *Las trazas generadas por el marco de trabajo Odoo dificultan la realización del análisis de los procesos que se ejecutan sobre esta plataforma.* Se delimita como **Objeto de estudio** la *Minería de Procesos* teniendo como **Campo de acción** *el registro de trazas para la Minería de Procesos*.

Para solucionar el problema detectado se plantea como **Objetivo general:** Desarrollar un módulo que permita la extracción de las trazas registradas por Odoo y su transformación al formato XES. Para satisfacer el objetivo general propuesto se definieron los siguientes **Objetivos específicos:**

- Confeccionar el marco teórico conceptual de la investigación a partir de una revisión bibliográfica para definir el proceso de configuración, así como los datos a registrar a partir de la definición de XES (Flujo de Eventos Extensible), sus extensiones y clasificadores.
- Realizar el análisis y diseño del módulo para la transformación y extracción de las trazas registradas en Odoo.
- Desarrollar un módulo para la transformación de trazas de proceso relacionados a los datos de Odoo.
- Validar la solución propuesta mediante pruebas de software y un caso de estudio.

Se define como **Idea a Defender:** El desarrollo de un módulo para la extracción de las trazas registradas por Odoo y su transformación al formato XES permitirá la realización del análisis de los procesos que se ejecutan sobre esta plataforma.

Métodos Científicos: [10]

- Métodos Lógicos
  - Métodos Lógicos de Soporte: Contribuyen a la obtención de conocimiento [10, 11].
    - Modelación: Se crean abstracciones con el propósito de explicar la realidad. Este método se utiliza en la confección de diagramas y modelos [10, 11].
    - Analítico - Sintético: División del todo en partes con el objetivo de estudiarlas y analizarlas, luego de su descomposición y análisis se hace una reunión de todos los elementos en una nueva totalidad. Se realiza sobre los resultados del análisis. Este método se utiliza en el análisis del marco teórico de la Minería de Procesos y sus conceptos [10, 11].
- Empíricos
  - Cuantitativos: Tratan de establecer relaciones causales que supongan una explicación del fenómeno [10, 11].
    - Métodos Observacionales
      - Caso de Estudio: Estudia un fenómeno en su contexto natural. Se aplica en la validación de la solución propuesta [10, 11].

El presente trabajo está estructurado en 3 capítulos. En el Capítulo 1, “Fundamentación teórica”, se describen los principales conceptos del objeto de estudio que se abordan en la presente investigación. Se brinda un mayor detalle del contexto en el cuál se desarrolla la problemática y se analizan las características, ventajas y desventajas de las soluciones implementadas. Además se mencionan las herramientas y tecnologías utilizadas en el desarrollo de la solución.

En el Capítulo 2, “Propuesta de solución”, se describe la solución a implementar. Se justifica la selección de patrones y modelos para el diseño de la solución. Se enuncian los requisitos funcionales y no funcionales.

En el Capítulo 3, “Implementación y pruebas”, se valida el diseño de la solución. Se describen los estándares de codificación utilizados, la nomenclatura del código y métodos implementados. Se aplica un caso de estudio para la validación de la solución propuesta. Se analizan los resultados de las validaciones aplicadas a la investigación.

Capítulo 1

Fundamentación teórica

Introducción

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

1.1. Conceptos fundamentales

A continuación se describen los conceptos fundamentales tratados en la presente investigación.

1.1.1. Minería de Procesos

La Minería de procesos consiste en técnicas, herramientas y métodos para descubrir, monitorear y mejorar los procesos reales a través de la extracción de conocimiento de los registros de eventos, ampliamente disponibles en los actuales sistemas de información [9].

El descubrimiento de procesos, el chequeo de conformidad y el mejoramiento de modelos, son los tres tipos de Minería de procesos. El descubrimiento de procesos permite obtener un modelo de proceso a partir de un registro de eventos. A su vez el chequeo de conformidad analiza si la realidad (según consta en un registro de eventos) se ajusta al modelo y viceversa. Su objetivo es detectar las discrepancias y medir su gravedad. Por otra parte el mejoramiento de modelos permite extender o mejorar los procesos a partir de la información extraída de un registro de eventos. Por ejemplo, se pueden identificar cuellos de botella reproduciendo un registro de eventos en un modelo de proceso, mientras se examinan las marcas de tiempo [9]. Para el caso de Odoo se configurarán los procesos para registrar las trazas con atributos que permitan obtener un registro de eventos sobre el cuál aplicar los distintos tipos de minería de procesos.

Se considera un proceso a un conjunto de actividades mutuamente relacionadas o que interactúan entre sí, las cuales transforman entradas en resultados. Un caso o instancia de proceso, es la instanciación (o ejecución) de un conjunto de actividades que pertenecen todas a un proceso [9]. Luego, los eventos son las instancias de dichas actividades, por ende una instancia de proceso está formada por eventos. La relación entre los conceptos expuestos[1] es muy parecida a la de clase y objeto en el paradigma orientado a objetos.

[1] Tomando el proceso y las actividades el rol de clase, y la instancia de proceso y los eventos, el rol de objetos respectivamente

TABLA 1.1: Ejemplo de datos de un registro de eventos

Caso	Id	Nombre	Fecha	Recurso	Costo
1	123	Agregar usuario	2015-04-23 16:44:50	Javier	20
1	124	Comprobar credenciales	2015-04-23 16:50:00	Javier	40
2	125	Agregar usuario	2015-04-24 08:30:25	Rosa	20
1	126	Proveer autorización	2015-04-24 09:40:32	Javier	20
2	127	Comprobar credenciales	2015-04-24 12:20:00	Rosa	40

1.1.2. Registro de eventos

El registro de eventos es la colección utilizada como entrada para aplicar la Minería de procesos. Los eventos no necesitan ser almacenados en un archivo de registro por separado (por ejemplo, los eventos pueden estar dispersos en diferentes tablas de bases de datos). Los eventos se corresponden con las acciones almacenadas en el registro como el inicio, conclusión o cancelación de una actividad para una instancia particular de un proceso [9].

Las técnicas de Minería de Procesos requieren como entrada un registro de eventos "plano" similar al de la Tabla 1.1 en la que cada uno tiene las siguientes propiedades:

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

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

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

1.1.3. Enfoques de los sistemas de información

Un sistema de información es “un tipo particular de sistema de trabajo que utiliza la tecnología de información para captar, transmitir, almacenar, recuperar, manipular o mostrar información, apoyando así uno o más sistemas de trabajo” [12]. Esta definición utiliza dos términos clave: tecnologías de la información y sistema de trabajo. La tecnología de la información se define como “el hardware y el software utilizado para almacenar, recuperar y transferir información” [12], un sistema de trabajo como “un sistema en el que los participantes humanos realizan un proceso de negocio utilizando la información, la tecnología y otros recursos para producir productos para los clientes internos” [13].

La importancia de los sistemas de información no sólo se refleja en el crecimiento exponencial de los datos, sino también por el papel que estos sistemas desempeñan en los procesos de negocio de hoy en día, a medida que el universo digital y el universo físico se alinean cada vez más. Por ejemplo, el “estado de un banco” está determinado principalmente por los datos almacenados en el sistema de información del banco. Los datos registrados por los sistemas de información se pueden utilizar para proporcionar una mejor visión de los procesos reales, es decir, las desviaciones pueden ser analizadas y la calidad de los modelos se puede mejorar [9].

Los sistemas de información, de acuerdo a la forma en que implementan y definen sus funcionalidades, tienen varios enfoques, dos de ellos son el consciente de los datos y el consciente de los procesos. Desde finales de 1970 hasta principios de 1990, el centro de atención en el área de sistemas de información fue hacia los datos, principalmente en el almacenamiento y recuperación de la información y, por lo tanto, los modelos de datos eran a menudo el punto de partida para el diseño de sistemas de información, mientras que los sistemas de gestión de bases de datos fueron considerados como el corazón de la infraestructura de tiempo de ejecución. Durante la década de 1990, una serie de tendencias paralelas cambió el enfoque hacia los procesos. Como resultado, un número cada vez mayor de los procesos de negocio ahora se llevan a cabo bajo la supervisión de los sistemas de información impulsados por modelos de procesos explícitos o PAIS (*Process Aware Information Systems*)[13]. Los PAIS se utilizan principalmente para apoyar la ejecución de los procesos de negocio. Dentro de un PAIS el proceso se define como un modelo que se puede ver y cambiar. Esto permite la modificación de la definición del proceso sin la necesidad de modificar el código de la aplicación. Los PAIS admiten la promulgación automática de los procesos soportados en una organización. La información puede ser dirigida automáticamente a los actores humanos o aplicaciones apropiadas [13].

Otros sistemas de información implementan “conciencia sobre procesos” al darle soporte desde dentro de su aplicación específica. Ejemplos de ello son las herramientas de colaboración, herramientas de gestión de proyectos y sistemas de manejo de caso. Otro tipo de PAIS utilizado son los sistemas de Planificación de Recursos Empresariales (ERP) que apoyan todos los procesos de negocio de una empresa como la gestión de

la cadena de suministro, los software para la administración de la relación con los clientes (CRM) y los recursos humanos [9].

Una tarea común en un sistema de información es la de almacenar datos acerca de las actividades y procesos que se realizan en el área u organización que emplea el sistema. Estos datos son importantes y ayudan a la toma de decisiones en la organización. Sin embargo al registrar las trazas de estos datos, el enfoque influye en cómo se manipulan estos datos, afectando la gama de análisis que se pueden efectuar sobre ellos.

Un elemento clave a aclarar es que no necesariamente un sistema debe ser PAIS para generar trazas de gran completitud y organización, esto depende de cómo se configure el almacenamiento de las trazas y la cantidad de información que se recibe para confeccionarlas. Los PAIS incluyen a los sistemas que proporcionan una mayor flexibilidad o apoyan tareas específicas[13]. Por ejemplo, pueden ser vistos como conscientes de los procesos sistemas más grandes de ERP (SAP, Oracle), CRM, sistemas basados en reglas, software de centro de llamadas, de alta gama de middleware (WebSphere), aunque no necesariamente controlan los procesos a través de algún motor de flujo de trabajo genérico, en su lugar, estos sistemas tienen en común que hay una noción acerca de un proceso explícito y que el sistema de información es consciente de los procesos que soporta[9].

También un sistema de base de datos o programa de correo electrónico se pueden usar para ejecutar pasos en un proceso de negocio. Sin embargo, este tipo de herramientas de software no son “conscientes” de los procesos en que se utilizan. Por lo tanto, no participan activamente en la gestión y orquestación de los procesos para los que se utilizan [9].

1.1.4. Estándar XES

Aun cuando el registro de eventos (parcialmente) siga la estructura descrita anteriormente el formato difiere entre sistemas. Antes de aplicar técnicas de análisis tales como la Minería de Procesos, los registros deben ser convertidos a un formato estandarizado. Para ello el grupo de investigación de Arquitectura de Sistemas de Información en la Universidad Tecnológica de Eindhoven especificó el formato de registro de eventos XES [14].

El estándar XES se emplea actualmente para representar la información de los registros de eventos, de tal forma que sea procesable por las herramientas para aplicar Minería de Procesos. La Figura 1.1 muestra un ejemplo de un documento empleando XES.

El estándar define dos tipos de atributos, el primer tipo es para trazas y eventos. Algunos de estos atributos son requeridos por algoritmos de Minería de procesos y la mayoría de los atributos de este tipo son definidos por las extensiones de XES. El segundo tipo de atributo son los atributos de datos, que almacenan información

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- This file has been generated with the OpenXES library. It conforms -->
<!-- to the XML serialization of the XES standard for log storage and -->
<!-- management. -->
<!-- XES standard version: 1.0 -->
<!-- OpenXES library version: 1.0RC7 -->
<!-- OpenXES is available from http://www.openxes.org/ -->
<log xes.version="1.0" xes.features="nested-attributes" openxes.version="1.0RC7" xmlns="http://www.xes-standard.org/">
    <extension name="Metadata_Time" prefix="meta_time" uri="http://www.xes-standard.org/meta_time.xesext"/>
    <extension name="Lifecycle" prefix="lifecycle" uri="http://www.xes-standard.org/lifecycle.xesext"/>
    <extension name="Metadata_Lifecycle" prefix="meta_life" uri="http://www.xes-standard.org/meta_life.xesext"/>
    <extension name="Organizational" prefix="org" uri="http://www.xes-standard.org/org.xesext"/>
    <extension name="Metadata_Organizational" prefix="meta_org" uri="http://www.xes-standard.org/meta_org.xesext"/>
    <extension name="Time" prefix="time" uri="http://www.xes-standard.org/time.xesext"/>
    <extension name="Metadata_Concept" prefix="meta_concept" uri="http://www.xes-standard.org/meta_concept.xesext"/>
    <extension name="3TU metadata" prefix="meta_3TU" uri="http://www.xes-standard.org/meta_3TU.xesext"/>
    <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org/concept.xesext"/>
    <extension name="General metadata" prefix="meta_general" uri="http://www.xes-standard.org/meta_general.xesext"/>
    <extension name="Semantic" prefix="semantic" uri="http://www.xes-standard.org/semantic.xesext"/>
    <global scope="trace">
        <date key="REG_DATE" value="1970-01-01T00:00:00.000+01:00"/>
        <string key="AMOUNT_REQ" value="UNKNOWN"/>
        <string key="concept:name" value="UNKNOWN"/>
    </global>
    <global scope="event">
        <date key="time:timestamp" value="1970-01-01T00:00:00.000+01:00"/>
        <string key="lifecycle:transition" value="UNKNOWN"/>
        <string key="concept:name" value="UNKNOWN"/>
    </global>
    <classifier name="Activity classifier" keys="concept:name lifecycle:transition"/>
    <classifier name="Resource classifier" keys="org:resource"/>
    <trace>
        <date key="REG_DATE" value="2011-10-01T16:39:55.798+02:00"/>
        <string key="concept:name" value="173772"/>
        <string key="AMOUNT_REQ" value="3000"/>
        <event>
            <string key="org:resource" value="112"/>
            <string key="lifecycle:transition" value="COMPLETE"/>
            <string key="concept:name" value="A_SUBMITTED"/>
            <date key="time:timestamp" value="2011-10-01T16:39:55.798+02:00"/>
        </event>
        <event>
            <string key="org:resource" value="112"/>
            <string key="lifecycle:transition" value="COMPLETE"/>
            <string key="concept:name" value="A_PARTLYSUBMITTED"/>
            <date key="time:timestamp" value="2011-10-01T16:39:55.898+02:00"/>
        </event>
        <event>
            <string key="org:resource" value="112"/>
            <string key="lifecycle:transition" value="COMPLETE"/>
            <string key="concept:name" value="A_DECLINED"/>
            <date key="time:timestamp" value="2011-10-01T16:40:31.075+02:00"/>
        </event>
    </trace>
```

FIGURA 1.1: Registro de eventos en formato XES

Fuente: https://www.researchgate.net/publication/304550918_Improve_Your_Business_through_Process_Mining/figures

adicional acerca del objeto al cual se refiere la traza o actividad ejecutada. Algunos algoritmos requieren atributos específicos en el registro de eventos. Por otra parte no todos los atributos definidos por las extensiones estándar deben estar siempre presentes en el registro, a veces la información requerida ni se encuentra en la fuente de datos [15].

La extensión de XES más importante es `concept`, la cual especifica un nombre para el registro, las trazas y los eventos. Proporcionar nombres a cada elemento es la única forma que tienen los algoritmos para diferenciar los eventos, por eso siempre deben ser provistos. Los nombres de las trazas incluirán identificadores únicos. La extensión `concept` define un atributo *instance* para eventos, este representa un identificador de la instancia de proceso que generó el evento [14].

Otra extensión es `time` [14]. Esta especifica el atributo que representa el instante de ocurrencia del evento. La grabación del instante de tiempo es la clave para ordenar los eventos y permite realizar análisis de duración y desempeño. Es importante registrar el momento de la ejecución del evento con un alto nivel de detalle,

preferiblemente hasta los segundos y en algunos casos los milisegundos[2]. Esto es necesario porque los eventos deben tener un único instante de tiempo dentro de la traza. Aunque la mayoría de los algoritmos pueden manejar instantes de tiempos iguales, los resultados mejoran si se brinda más precisión [14].

Los eventos son grabaciones atómicas y por eso no tienen duración. Como las actividades sí tienen duración, los eventos pueden ser de diferentes tipos, cada uno de estos grabando un estado de una actividad. Los tipos de eventos son proporcionados por la extensión `lifecycle` [14].

Dos de los tipos de eventos más comunes son el iniciado y completado para indicar el inicio y la completitud respectivamente de una actividad. Contando con estos dos tipos de eventos es posible estimar tiempo de procesamiento y tiempo de espera en un análisis de desempeño, en la mayoría de los casos con usar el tipo iniciado y completado es suficiente. Si no existe información acerca del inicio y fin de una actividad se usa solamente el completado como tipo de evento [14].

Otra dimensión del análisis de Minería de procesos trata sobre la distribución del trabajo. Relacionando eventos, recursos o grupos de recursos se visualiza la distribución del trabajo entre diferentes unidades. Además se pueden construir redes sociales para indicar por ejemplo los actores del negocio que trabajaron juntos en algunos casos o grupos de actores desempeñando tareas similares [14].

Para lograr un descubrimiento de estas redes, el actor o grupo que ejecutó el evento debe ser registrado con cada evento. Lo anterior se define como la extensión `organizational` que define tres diferentes atributos para eventos. Comúnmente se emplea el `resource` para recoger el nombre o identificador del actor que ejecutó el evento. Adicionalmente el rol del recurso se puede almacenar también con el atributo `role` así como el grupo al que el usuario pertenece, mediante el atributo `group` [14]. La decisión de qué atributos usar depende de la información disponible y del nivel de detalle deseado. En algunos casos, los análisis sobre el rol o el grupo son necesarios [16].

Otra de las extensiones de XES a emplear es `micro`. Los actuales sistemas pueden definir y manejar actividades anidadas o subprocesos, y esta información debe ser representada en el registro de eventos. Para ello `micro` define un atributo `level` en correspondencia con el nivel de la actividad. Cuenta además con el atributo `parentId` para especificar en un evento el identificador de su padre, si tiene uno. Por último está el atributo `length` que brinda la cantidad de hijos si el evento en cuestión es padre [17].

[2]La granularidad depende del tipo análisis que se desee ejecutar, los eventos de grano fino a veces contienen información que es irrelevante para los análisis

1.1.5. Odoo

De los ERP que existen en el mercado los que más destacan son SAP, Dynamics, Deltek Vision, ePROMISE y Odoo [18]. De ellos se selecciona Odoo para enmarcar la investigación debido a que actualmente es la propuesta que adopta la UCI como plataforma para la informatización de la empresa cubana. Además es software libre y esto brinda una serie de características deseables como flexibilidad de implementación, es ajustable al modelo económico de Cuba y contribuye a la soberanía tecnológica del país.

En el sitio web de la compañía se explica:

> "Odoo es un conjunto de aplicaciones empresariales de código abierto que cubren todas las necesidades de su empresa: CRM, comercio electrónico, contabilidad, inventario, punto de venta, gestión de proyectos, etc."[19]

1.2. Problemas en la confección del registro de eventos

En las siguientes secciones se introducen los problemas relativos a la confección de registros de eventos que se abordan en la presente investigación.

1.2.1. Trazas en Odoo

En Odoo resulta complejo obtener un registro de eventos debido a que no almacena ningún tipo de traza en un formato compatible con XES. Existe un módulo de Odoo llamado Auditlog que "... permite al administrador registrar las operaciones del usuario realizadas en modelos de datos como crear, leer, escribir y eliminar" [20] pero este módulo captura eventos para todos los usuarios sin distinción, no permite capturar los eventos en tiempo real[3] que es necesario para obtener la marca de tiempo y tampoco asocia los eventos a una misma instancia , elementos que influyen en la calidad del registro de eventos.

1.2.2. Problema de identificación de eventos

Este es el problema principal a resolver cuando se está en la fase de extracción y registro de eventos de la Minería de Procesos, ya que si no se identifican los eventos correctamente se afecta toda la salida del proceso de minado [2]. Reconstruir un registro de eventos significa decidir cuándo inferir la existencia de un evento en la base de datos operacional y llenar cada uno de los atributos del evento con valores relevantes. Estos valores pueden ser extraídos de la base de datos o pueden ser especificados por el experto del dominio [21].

[3] Que un evento sea capturado en tiempo real significa que el evento se registra inmediatamente luego de creado, para que su marca de tiempo sea lo más fidedigna posible

Existen diversas formas de obtener los eventos según la fuente de datos, por ejemplo se aplican patrones de identificación, ordenamiento, asociación y correlación [21].

Para el caso del módulo de Odoo se emplea el modelo de eventos propuesto por Wil van der Aalst en su trabajo “Extracting Event Data from Databases to Unleash Process Mining”. Este modelo está definido de forma genérica para sistemas que emplean bases de datos como fuente de datos y se definen como eventos las acciones de insertar y eliminar objetos y sus relaciones, y las actualizaciones de los objetos, como se muestra en la Figura 1.2. [22].

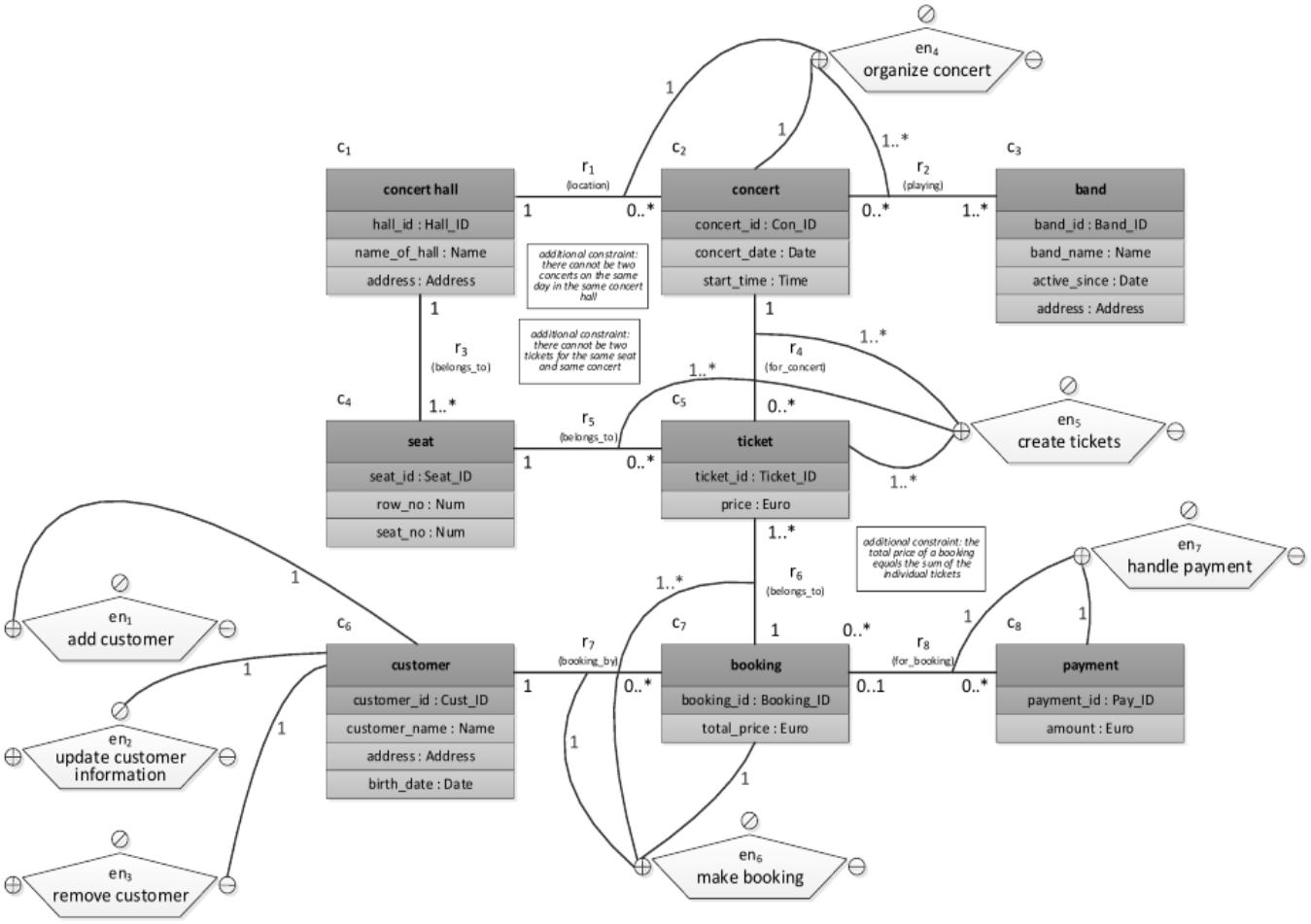

FIGURA 1.2: Modelo de eventos aplicado en un sistema

Fuente: Extracting Event Data from Databases to Unleash Process Mining [22]

El símbolo ⊕ se corresponde con las inserciones, el símbolo ⊘ con las eliminaciones y el símbolo ⊖ con las actualizaciones. Las líneas rojas que van desde los símbolos hacia los elementos de la figura indican el tipo de operación que se está realizando en cada elemento, en el caso de la Figura 1.2 se definen como eventos las inserciones en la entidad `concert` y sus relaciones `r1` y `r2`. Además el modelo brinda un conjunto de principios de confección del registro de eventos con el objetivo de identificar errores y obtener un buen punto de partida para aplicar la Minería de procesos. Siguiendo el modelo de eventos de Wil van der Aalst, los eventos en Odoo

se corresponden con las inserciones, actualizaciones y eliminaciones de datos en las tablas del negocio que se maneje.

1.2.3. Problema de la convergencia y divergencia

Una de las propiedades de un registro de eventos es que cada evento tiene que estar relacionado a solo una instancia de un proceso. En la realidad esto no es siempre así, por lo que representa un problema en la conversión hacia registro de eventos [14].

Como resultado de la convergencia y/o divergencia puede ocurrir que un algoritmo sea incapaz de utilizar un registro de eventos. Por ejemplo el algoritmo "Alpha miner" no es capaz de realizar Minería de procesos a un registro de eventos donde ocurran convergencia y divergencia, lo cuál limita la gama de análisis que se pueden realizar sobre el registro de eventos y en el caso que el algoritmo no pueda detectar la existencia de estos dos problemas, el resultado del análisis puede ser erróneo. [14]

La Figura 1.3 provee un ejemplo de convergencia y divergencia entre la traza de la Orden (Order) y el evento Pago (Payment)

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

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

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

FIGURA 1.3: Ejemplo de convergencia y divergencia entre traza y evento

Fuente: Mapping Data Sources to XES in a Generic Way [14]

Las órdenes(o pedidos) están en la tabla Orden, los pagos se almacenan en la tabla 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 mayor complicación, 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. [14]

El problema de la convergencia existe cuando una actividad se ejecuta en múltiples instancias de proceso a la vez. Esto se puede reconocer cuando hay una relación 1: N (uno a muchos) desde un evento hacia la instancia de proceso. En el ejemplo antes descrito, la convergencia se manifiesta en el caso del pago con identificador 10, que aparece en las trazas de las órdenes 1 y 2. Desde el punto de vista de un algoritmo de minería de procesos parecería que un mismo usuario estaba ejecutando dos actividades de pago a la vez.[14]

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

1.2.4. Problema de la confección del registro de eventos

Existen un conjunto de problemas independientes del enfoque, que están presentes en los registros de eventos extraídos de sistemas reales y están asociados a la herramienta empleada para conformar el registro, el algoritmo utilizado o las condiciones del negocio. Estos problemas deben ser considerados por los desarrolladores a la hora de crear el registro de eventos para que la información obtenida sea lo más completa posible: [23]

- Granularidad de eventos: los eventos se almacenan con bajos niveles de detalle o altos niveles de detalle en el mismo registro.
- Heterogeneidad de los casos: se refiere a instancias de procesos con muchos flujos alternos, o a los procesos que cambian constantemente en el tiempo.

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

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

Desafortunadamente los procesos agrupados no se pueden analizar y/o modelar con los actuales enfoques de extracción de registros de eventos a partir de sistemas orientados a datos. Con Odoo se resuelve la ambigüedad porque la marca de tiempo de las trazas alcanza el nivel de detalles de segundos y milisegundos, por tanto dos eventos del mismo tipo ocurridos a la misma hora se pueden diferenciar. En el caso de la granularidad no es posible resolver este problema porque el modelo no contempla la representación de eventos compuestos.

1.3. Herramientas para gestionar el registro de eventos

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

1.3.1. Eventifier

Eventifier es una herramienta que ayuda en la reconstrucción de un log de eventos desde bases de datos operacionales donde existen evidencias de la ejecución de instancias de procesos. Primeramente, el experto en el dominio identifica los eventos en la base de datos operacional, los ordena y le asocia datos. Para ello se apoya en el componente Extractor de eventos, que soporta estas actividades y ayuda al experto de forma interactiva. El resultado de este prime 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 la aplicación de Minería de procesos [21].

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

1.3.2. XESame

XESame es un mapeador de XES que provee una manera genérica para extraer un registro de eventos desde una fuente de datos. Una de las potencialidades de XESame es su facilidad de uso ya que no requiere habilidades de programación. XESame le permite a un experto en el dominio especificar cómo el registro de eventos debe ser extraído de la base de datos operacional de un sistema de información determinado y convertido a XES o MXML. El sistema brinda varias interfaces para la definición de la conversión en las cuales se pueden definir los tipos de gestores de bases de datos a usar como son PostgreSQL o Mysql. La herramienta permite establecer las relaciones entre las tablas de la base de datos y sus respectivos campos en el registro de eventos, teniendo en cuenta las extensiones de XES definidas. Una vez obtenido el registro de eventos en formato XES, se debería ser capaz de analizar este registro de todas las maneras posibles en el ámbito de Minería de procesos [14].

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

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

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

Con esta nueva forma de organizar las trazas, se podía mapear la información y obtener un registro de eventos en formato XES válido para aplicarle Minería de procesos [15]. Además el framework se desarrolló en PHP, haciendo difícil la integración de dicho módulo dentro de la plataforma Odoo.

1.3.4. XTract2 tool

Esta herramienta emplea un enfoque diferente al utilizado por otros trabajos. En lugar de obtener un registro de eventos y exportarlo a XES, obtiene el modelo de procesos usando un modelo de artefactos. Se mapea cada tabla de la base de datos como un artefacto y se establecen las relaciones entre ellos empleando puertos y canales de comunicación. El uso del modelo de artefactos disminuye en gran medida los problemas de la convergencia y la divergencia, elimina los eventos repetidos y permite representar los procesos relacionados entre sí, o sea aquellos que comparten actividades. Este último problema no había sido resuelto por otras investigaciones. Este trabajo permitió mejorar una herramienta existente para el modelado empleando artefactos y resolver muchos de los problemas que presenta este enfoque [24].

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

1.3.5. Resultados

La Tabla 1.2 y la Tabla 1.3 ilustran respectivamente las principales ventajas y desventajas identificadas en las herramientas estudiadas.

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

TABLA 1.2: Ventajas de las herramientas

Criterios	**Eventifier**	**XESame**	**Módulo de SAUXE**	**XTract2**
Identificar eventos	Logs de base de datos	Fuente de datos genérica	Trazas de Sauxe	Trazas del sistema
Identificar instancias	Patrones de identificación	Selección de instancia	Selección de instancia	Enfoque XTract
Correlacionar eventos con instancias	Patrones de correlación	Definición de correlación	Definición de correlación	Patrones de artefactos
Exportar a XES	No	Si	Si	Modelo de Artefactos
Empleo de herramienta	Eventifier	XES Mapper	Módulo	Xtract2

TABLA 1.3: Carencias de las herramientas

Criterios	**Eventifier**	**XESame**	**Módulo de SAUXE**	**XTract2**
Origen de la información	Logs de bases de datos	Trazas existentes	Trazas existentes	Trazas existentes
Maneja convergencia	No	Selección de instancia de bajo nivel	Selección de instancia de bajo nivel	Si
Maneja divergencia	No	Selección de instancia de bajo nivel	Selección de instancia de bajo nivel	Si
Herramienta disponible	Si	Si	Contenida en SAUXE	No

1.4. Metodología, herramientas y tecnologías usadas

En el desarrollo de la solución propuesta se emplean las herramientas y tecnologías necesarias para la construcción de módulos en la plataforma Odoo y algunos específicos dedicados a la obtención de los eventos en tiempo real.

1.4.1. Metodología

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

1.4.2. Herramientas y tecnologías

* Pip v9.0.1 (Anexo A.1.1)

* Postgresql v9.5 (Anexo A.1.2)
* PyCharm Community Edition v2016.3.1 (Anexo A.1.3)
* Visual Paradigm Community Edition v15 (Anexo A.1.4)
* Odoo v10 (Anexo A.1.5)
* Odoo Connector v1.0.0 (Anexo A.1.6)
* Python v2.7 (Anexo A.1.7)
* Core Filing XML Schema Validator v1.2.0.r278285 (Anexo A.1.8)

Conclusiones del capítulo

La confección del marco teórico conceptual y la revisión bibliográfica permitieron tomar elementos para la definición del proceso de configuración para la captura y extracción de registros de eventos.

El análisis realizado sobre el componente de trazas en Odoo y las extensiones de XES arrojó como resultado que los datos almacenados actualmente no son completos ni tienen la organización básica como para confeccionar un registro de eventos de acuerdo con el estándar XES.

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

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

Capítulo 2

Propuesta de solución

Introducción

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

2.1. Análisis del módulo

El módulo de Odoo se auxilia de un módulo de terceros para la captura de los eventos en tiempo real llamado Odoo Connector (Anexo A.1.6), este módulo define una clase que "escucha" (*listener* del inglés) los eventos producidos y los registra, es una implementación a nivel de modelo del patrón Observer [25–27]. Dicho módulo instala automáticamente otro módulo denominado *component_event* que se encarga de redefinir la clase base de los modelos *models.Base* para habilitar la captura de datos en tiempo real.

Una vez que el módulo *component_event* es instalado la clase base de los modelos *models.Base* se redefine con la estructura mostrada en la Figura 2.1. Los métodos *create*, *write* y *unlink* son homólogos a las operaciones insertar, actualizar y eliminar en una base de datos relacional. Con este diseño cualquier modelo dentro de la plataforma Odoo puede ser configurado para escuchar sus eventos. En el caso de los métodos *create* y *write*, el parámetro *vals* son los valores actuales del objeto que está siendo creado o actualizado.

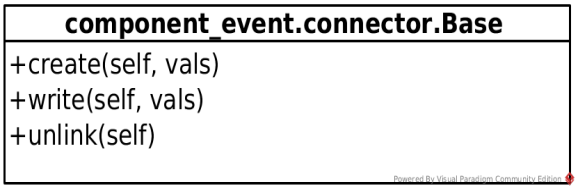

FIGURA 2.1: Clase listener para eventos

Fuente: Elaboración propia

2.2. Propuesta de solución

En la propuesta de solución se configura inicialmente cuáles son los procesos a manejar en el sistema implementado, sus actividades y los atributos, así como las tablas que se corresponden con cada uno de estos elementos. De esta forma el sistema ya está listo para escuchar y registrar los eventos con los datos apropiados para aplicar Minería de Procesos. Para ello se definen en Odoo instancias de proceso, eventos, atributos, la relación entre tablas y cómo se exportan a XES cada uno de estos procesos.

La solución está definida como una serie de pasos de configuración que hay que realizar en el sistema para poder exportar los registros de eventos, cada uno de estos pasos se detalla y fundamenta en las siguientes secciones:

2.2.1. Paso 1: Selección de instancias de proceso

El primer paso en la configuración consiste en seleccionar las instancias de proceso. Cada vez que ocurra un evento en el sistema debe estar asociado a una única instancia. Elegir la instancia de proceso es complejo porque implica analizar cuáles son los objetos del negocio modelado y cómo se relacionan entre sí. Esta tarea se torna difícil en sistemas que manejen muchas entidades interrelacionadas, debido a que una mayor cantidad de entidades y relaciones aumenta la complejidad del modelo que debe analizar el experto del dominio.

En un sitio web de bloggers, por ejemplo, los usuarios pueden seleccionarse como instancia de proceso porque sobre ellos gira toda la lógica del negocio y ejecutan acciones como: escribir publicaciones, escribir comentarios, crear contacto, compartir recurso, el punto es que todas estas acciones son realizadas por usuarios del sitio y sin ellos, dichas actividades no tendrían lógica o representarían una ambigüedad en el sistema.

En la solución que aquí se propone, la información de la instancia de proceso se guarda en la tabla PMProceso, la cuál se muestra en la Figura 2.2. Cada uno de los campos en la tabla y su descripción se relaciona en la Tabla 2.1.

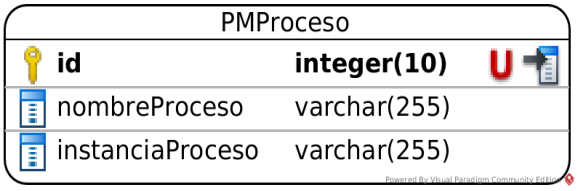

FIGURA 2.2: Entidad PMProceso

Fuente: Elaboración propia

TABLA 2.1: Descripción de campos de la tabla PMProceso

Campo	Descripción
id	Identificador del proceso
nombreProceso	Nombre del proceso
instanciaProceso	Nombre de la tabla que representa la instancia de proceso

2.2.2. Paso 2: Selección de eventos

Los eventos son capturados a partir de las actividades relacionadas a un proceso, de ellas se guarda en la tabla PMActividad el nombre y la tabla sobre la cual se ejecuta la actividad, esta tabla puede ser la misma de la instancia de proceso u otra relacionada. Se registra además el identificador de la instancia, el tipo de actividad (insertar, eliminar o actualizar) que se relaciona con los tipos de eventos definidos. Por último el nivel y si es una actividad hija el identificador de la actividad padre, como se muestra en la Figura 2.3.

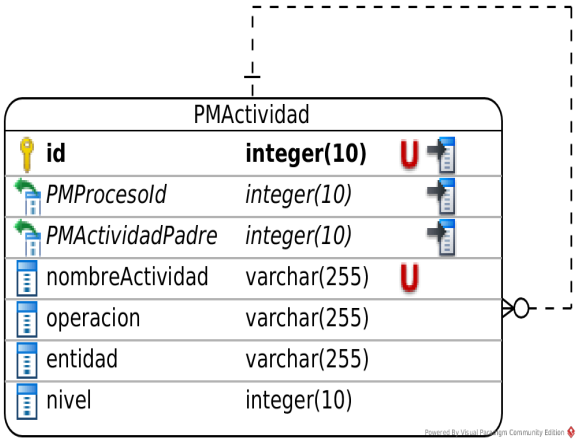

FIGURA 2.3: Entidad PMActividad

Fuente: Elaboración propia

Se establece una relación de uno a muchos con la misma entidad y se emplean los valores del identificador de la actividad padre y el nivel para modelar los eventos compuestos. Estos eventos están constituidos por otros sub eventos, por ejemplo un evento A se compone de B, C y D. Al plantear una relación de tipo padre-hijo se modela la jerarquía de eventos propuesta por Reinhold Dunkl, con el objetivo de disminuir el efecto de la granularidad en los modelos de procesos y brindar información adicional acerca de los sub eventos que componen a un padre [28].

La extensión *micro* del estándar XES posee también campos que representan esta jerarquía, simplificando el modelo de procesos que se le muestra al usuario al emplear la herramienta ProM y permite revisar los eventos compuestos con mayor nivel de detalle [17]. Aplicar el modelo jerárquico de Dunkl y aprovechar la extensión *micro* permite representar en el registro los valores compuestos y minimizar el efecto del problema de granularidad.

El módulo Odoo Connector permite que cuando ocurre un evento en Odoo se comprueba si se corresponde con alguna de las actividades definidas previamente en la tabla PMActividad. En caso de ser positiva la respuesta se guardan los datos del evento en la tabla PMRegistro mostrada en la Figura 2.4 a continuación. Sobre esta tabla se efectúan consultas posteriormente para extraer de ella los eventos que se deseen analizar en el registro.

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

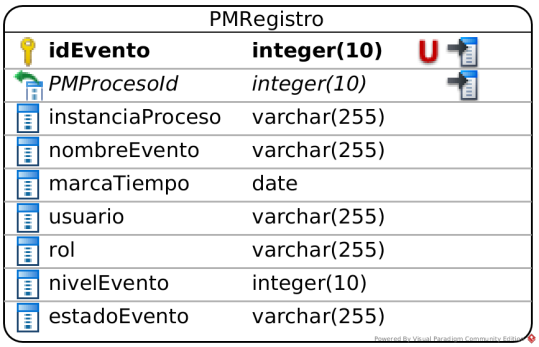

FIGURA 2.4: Entidad PMRegistro

Fuente: Elaboración propia

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

2.2.3. Paso 3: Selección de atributos adicionales de los eventos

Los eventos necesitan una marca de tiempo, un identificador y una instancia de proceso para ser organizados y analizados. Esos valores son asignados al evento mediante funciones implementadas que toman la información de cuándo el proceso se ejecutó, almacenando el momento exacto hasta los segundos y utilizando un algoritmo diseñado para la obtención de la instancia de proceso, los detalles del algoritmo en cuanto al chequeo de cardinalidad se encuentran en el Anexo B.1.1

Los atributos adicionales ofrecen información extra como el costo de ejecutar una acción o qué recurso humano la ejecutó. El valor del atributo se obtiene de una o varias columnas de una tabla, para este caso también se desarrolló un algoritmo[1] que toma los atributos adicionales de tablas en las que sus relaciones formen un camino entre la tabla de la actividad y la tabla del recurso que se pide y no específicamente una relación en la que el camino sea de longitud 1.

En Odoo estos atributos se definen como recursos adicionales y se configuran luego de las instancias y las actividades. Para cada actividad se pueden definir recursos. Del recurso se conoce la tabla y columna de la cual se extrae la información, como se muestra en la Figura 2.5.

Cuando ocurre un evento en el sistema y se comprueba si se corresponde con una actividad, se buscan los recursos asociados a la actividad. Si existen se guardan en la tabla PMAtributo mostrada en la Figura 2.6 el identificador del evento y el recurso, el nombre del campo y el valor que tiene para la instancia de proceso del

[1] Para mayor información refiérase al Anexo B.1.2

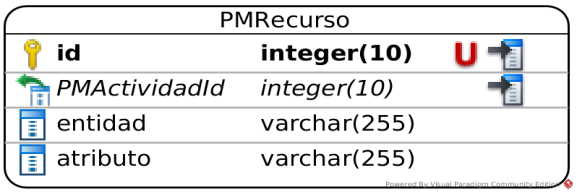

FIGURA 2.5: Entidad PMRecurso

Fuente: Elaboración propia

evento. De esta forma se asegura que el registro de eventos contará con información adicional acerca de las actividades que se ejecutan.

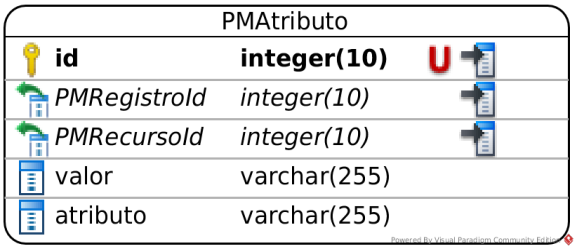

FIGURA 2.6: Entidad PMAtributo

Fuente: Elaboración propia

2.2.4. Paso 4: Determinar relaciones entre las entidades

Cuando se realiza la configuración se comprueba que existe una relación entre la tabla de la instancia de proceso y la tabla de la actividad, o la tabla del recurso. De lo contrario no habría forma de obtener los datos de esas tablas a partir de la instancia. Para resolver este problema en Odoo se implementa el algoritmo de Búsqueda a lo ancho Bidireccional (BBFS por sus siglas en inglés). Dicho algoritmo actúa sobre un conjunto de modelos de base de datos representándolos como nodos de un grafo y partiendo de ambos nodos origen y destino busca sistemáticamente hasta que encuentra un punto en común para ambos recorridos o el nodo en el lado opuesto de la búsqueda [29–31].

Un grafo de ejemplo es mostrado en la Figura 2.7. En Odoo el algoritmo recibe como parámetros dos entidades y trata de hallar un camino entre ellas. Si lo encuentra significa que las entidades están relacionadas y desde una se puede acceder a la información de la otra. Tomando como ejemplo las tablas Usuario y Publicación, a través de la búsqueda se halla el camino mínimo entre estas entidades, almacenando en caché el origen y destino de la búsqueda utilizando la técnica de *memoization*, de esta forma se recuperan los caminos la próxima vez con una operación que teóricamente demora $O(1)$[32].

El BFS bidireccional es costoso computacionalmente comparado con otros algoritmos de búsqueda en grafos como Búsqueda en profundidad[33], pero para el caso de Odoo el consumo solo es alto en el momento de

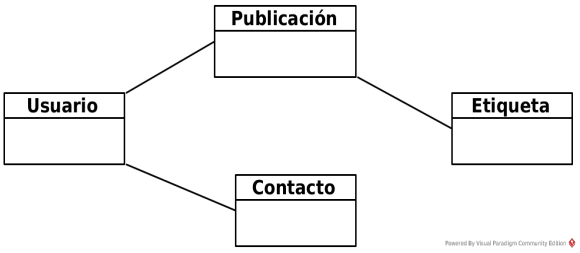

FIGURA 2.7: Representación del grafo en recorrido BFS de las tablas de la base de datos

Fuente: Elaboración propia

realizar la pre configuración. En este instante se buscan las entidades del negocio, pero con la obtención de los caminos mínimos entre ellas se reducen los tiempos de consultas porque ya la base de datos ha sido mapeada como un grafo cuyas conexiones son mínimas. Además de que los tiempos se amortizan aún más cuando se utiliza la técnica de *memoization*[2] como se explicó brevemente en el párrafo anterior.

2.2.5. Paso 5: Generación del registro de eventos en formato XES

Como último paso en la propuesta de solución está la generación del registro de eventos y su exportación al estándar XES. Una vez que han sido almacenados los eventos y recursos en sus respectivas tablas, el usuario que desee realizar un análisis de los procesos en su sistema puede solicitar un registro de los eventos asociados a un proceso. Para ello se recoge en la interfaz de usuario del módulo en cuestión el proceso al cuál se le desea exportar los datos a XES. Estos datos se envían a un controlador encargado de procesar la información y devolver una respuesta en dependencia del proceso seleccionado. Se procede a buscar los eventos que coincidan con los criterios de proceso. Si se encuentran eventos se organizarán según la instancia de proceso a la que se relacionan y con ellos se construirá el archivo XES.

Para que el registro sea válido en la aplicación de algoritmos o técnicas de Minería de Procesos debe referenciar las extensiones del estándar, contener una serie de atributos globales y definir clasificadores para las trazas y/o eventos.

En el caso del registro a generar con Odoo las extensiones utilizadas son: *Concept*, *Time*, *Lifecycle*, *Organizational* y *Micro*. Los atributos globales son elementos comunes en todas las trazas o los eventos; para la solución se definen los siguientes atributos a nivel de eventos: nombre, marca de tiempo, usuario (que ejecutó el evento), rol del usuario, nivel del evento y estado. Los clasificadores asignan un identificador a los eventos para realizar comparaciones entre ellos[16], para la solución los eventos se clasifican por su nombre y estado.

[2]Para una mayor información en cómo se utilizó esta técnica revise el Anexo B.1.3

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

2.3. Requisitos de software

La especificación de requisitos de software es una descripción completa del comportamiento del sistema que se va a desarrollar. Incluye un conjunto de casos de uso que describe todas las interacciones que tendrán los usuarios con el software. Los casos de uso también son conocidos como requisitos funcionales.

Además de los casos de uso, la especificación de requisitos de software también contiene requisitos no funcionales. Los requisitos no funcionales son requisitos que imponen restricciones en el diseño o la implementación. Está dirigida tanto al cliente como al equipo de desarrollo. El lenguaje utilizado para su redacción debe ser informal, de forma que sea comprensible para todas las partes involucradas en el desarrollo [34, 35].

2.3.1. Requisitos funcionales

Los requisitos funcionales se encapsulan empleando las Historias de usuario por ser parte de la disciplina de modelado en la Metodología utilizada. En el caso de la solución propuesta se definieron 10 historias de usuario, de ellas se describen las más relevantes (*).

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

2.3.1.1. Requisito Funcional: Adicionar Instancia de Proceso

Número: 1	**Nombre del Requisito**: Adicionar Instancia de Proceso
Programador: Luis Angel Mojena Román	**Iteración Asignada**: 1
Prioridad: Media	**Tiempo estimado**: 20 horas
Riesgo en Desarrollo: Poca experiencia del estudiante en la tecnología de la plataforma Odoo	**Tiempo Real**: N/A
Cuando se adiciona una instancia de proceso, se comienzan a escuchar los cambios que ocurren en las tablas de todas las actividades asociadas a la instancia de procesos. El usuario debe seleccionar la entidad de la instancia de procesos de una lista que se le provee[3]. Para efectuar la acción se necesita que el usuario provea la siguiente información: • Nombre del Proceso. • Entidad de la Instancia de Proceso[4].	

2.3.1.2. Requisito Funcional: Adicionar Actividad

Número: 2	**Nombre del Requisito**: Adicionar Actividad
Programador: Luis Angel Mojena Román	**Iteración Asignada**: 1
Prioridad: Media	**Tiempo estimado**: 20 horas
Riesgo en Desarrollo: Poca experiencia del estudiante en la tecnología de la plataforma Odoo	**Tiempo Real**: N/A
Al adicionar una actividad hay que tener en cuenta el proceso al cuál pertenece y la operación asociada a la actividad,así como la entidad que representa la actividad. El usuario debe seleccionar la entidad de una lista que se provee y el proceso también. Los procesos serán una lista de nombres de procesos que ya se hayan adicionado al sistema. Para efectuar la acción se necesita que el usuario provea la siguiente información: • Nombre de la Actividad • Entidad de la actividad[5]. • Tipo de operación[6]. • Proceso al que se relaciona.	

[3] Dicha lista debe contener todos los nombres de las tablas del sistema excepto los de la propia solución y se aplica para todos los campos de selección de entidades

[4] La tabla que representa la instancia de proceso

[5] La tabla que representa el objeto central de la actividad

[6] Las operaciones en la base de datos (INSERTAR, ACTUALIZAR, ELIMINAR)

2.3.1.3. Requisito Funcional: Adicionar Recurso

Número: 3	**Nombre del Requisito**: Adicionar Recurso
Programador: Luis Angel Mojena Román	**Iteración Asignada**: 1
Prioridad: Media	**Tiempo estimado**: 20 horas
Riesgo en Desarrollo: Poca experiencia del estudiante en la tecnología de la plataforma Odoo	**Tiempo Real**: N/A
Para adicionar un recurso se debe tener en cuenta que hay que obtener los nombres de las columnas de la entidad que se seleccione como entidad del recurso. La selección de la actividad, la entidad y la columna se debe hacer mediante una lista con datos que se le proveen al usuario. Para efectuar la acción se necesita que el usuario provea la siguiente información: • La actividad con que se relaciona. • La entidad del recurso[7]. • La columna de la entidad del recurso[8].	

2.3.1.4. Requisito Funcional: Exportar eventos a formato XES

Número: 7	**Nombre del Requisito**: Exportar eventos a formato XES
Programador: Luis Angel Mojena Román	**Iteración Asignada**: 4
Prioridad: Media	**Tiempo estimado**: 27 horas
Riesgo en Desarrollo: Poca experiencia del estudiante en la tecnología de la plataforma Odoo	**Tiempo Real**: N/A
Se deben extraer cada uno de los casos del proceso con sus respectivos eventos y construir el fichero XES, primero con el preámbulo del estándar y luego definiendo los atributos globales para casos y para eventos. Luego se construyen usando la información existente en la tabla PMRegistro, de la cuál cada tupla es un evento que corresponde a un caso relacionado con la instacia de proceso seleccionada. Para poder seleccionar la instancia de proceso, se le debe mostrar al usuario una lista de los procesos que hay en el sistema. Para ejecutar la acción se necesitan los siguientes datos: • El identificador de la instancia de proceso[9].	

[7] La tabla que representa al recurso
[8] La columna de la tabla que representa al recurso
[9] El nombre del proceso que se desea exportar

2.3.1.5. Requisito Funcional: Generar Registro de Eventos

Número: 8	**Nombre del Requisito**: Generar Registro de eventos
Programador: Luis Angel Mojena Román	**Iteración Asignada**: 4
Prioridad: Media	**Tiempo estimado**: 35 horas
Riesgo en Desarrollo: Poca experiencia del estudiante en la tecnología de la plataforma Odoo	**Tiempo Real**: N/A
Cada vez que una operación ocurre a nivel de base de datos, ya sea un insertar, eliminar o actualizar, esta se debe de registrar en la tabla PMRegistro almacenando siempre la siguiente información: * Tipo de Operación[10] * Actividad * Proceso * Marca de tiempo	

2.3.2. Requisitos no funcionales

Se corresponden con los requisitos definidos para Odoo en su versión 10. A continuación se describen los requisitos y se especifica cuáles son contemplados en la solución.

* Confiabilidad

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

  La confiabilidad se garantiza capturando y manejando los errores mediante los controladores. Solamente un usuario con los privilegios adecuados puede realizar la configuración de los procesos y actividades.

* Usabilidad

  - Los componentes visuales del sistema deben brindar al usuario una experiencia agradable y atractiva visualmente.
  - Permitir abrir varias aplicaciones a la vez en diferentes pestañas.

[10] Se refiere al tipo de operación realizada en la tabla de la actividad (insertar, eliminar o actualizar)

* Eficiencia

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

* Portabilidad

  - La aplicación se ejecuta en diferentes sistemas operativos.
  - El sistema se conecta a múltiples gestores de base de datos.

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

2.4. Modelo de diseño

El diseño de software es el proceso mediante el cual un agente crea una especificación de un artefacto de software, destinado a lograr objetivos, utilizando un conjunto de componentes primitivos y sujeto a restricciones. El diseño del software puede referirse a "toda la actividad involucrada en conceptualizar, enmarcar, implementar, encargar y finalmente modificar sistemas complejos" o "la actividad siguiente a la especificación de requisitos y antes de la programación, como en un proceso de ingeniería de software estilizado" [36, 37].

2.4.1. Patrón arquitectónico

El sistema que brinda soporte al módulo es la plataforma Odoo, este por tanto utiliza el patrón Modelo Vista Controlador como patrón arquitectónico. Es el encargado de separar los datos de la aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos. Odoo sigue la semántica del patrón MVC de la siguiente forma:[38]

* **Modelo**: Tablas de la base de datos.
* **Vista**: Vistas definidas en formato XML.
* **Controlador**: Los objetos de la plataforma.

2.4.2. Patrones de diseño

En el diseño que propone este trabajo, se utilizan algunos patrones de diseño del Grupo de Cuatro (GoF por sus siglas en inglés) y Patrones Generales de Software para Asignación de Responsabilidades (GRASP por sus siglas en inglés) para solucionar y/o evitar diferentes problemas que pudieran surgir durante la implementación de la solución. [39]

∗ Patrones GRASP

- **Experto**: propone que la clase que contenga toda la información necesaria será la responsable de la creación de un objeto o la implementación de un método. El comportamiento se distribuye entre las que contienen la información requerida, siendo más fáciles de entender, mantener y ampliar, aumentando sus posibilidades de reutilización [39, 40]. Se observa el uso de este patrón en todas las clases a utilizar en la solución ya que cada clase conoce su información y es la encargada de implementar las funcionalidades que brindan información de las mismas. En la Figura 2.8 se muestra como ejemplo la entidad PMActividad la cual posee toda la información acerca de las actividades.

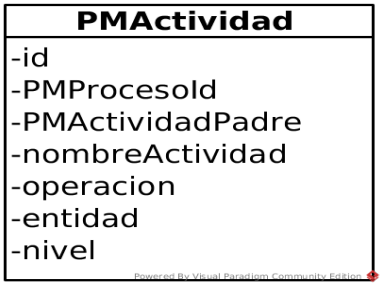

FIGURA 2.8: Clase PMActividad

Fuente: Elaboración propia

- **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 [39, 40]. Como ejemplo se muestra en la Figura 2.9 la clase controladora Pmining, encargada de exportar en formato XES los id de los procesos que se seleccionen.
- **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 [39, 40]. El uso de este patrón se evidencia en la poca relación existente entre las clases que conforman el módulo.

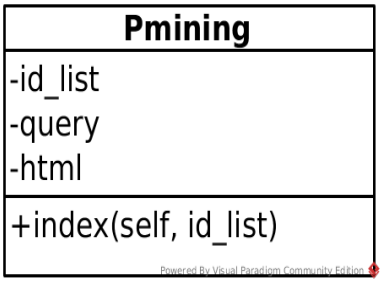

FIGURA 2.9: Clase Pmining

Fuente: Elaboración propia

- **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 [39, 40]. Ejemplo de esto es la clase *RelationsManagerGraph* la cual al recibir un cambio en sus atributos requiere una restructuración de los métodos para su correcto funcionamiento.

∗ Patrones GOF

- **Proxy**: cuando no se desea el acceso directo a un objeto sobre el que se va a aplicar determinada acción, este patrón propone la adición de un nivel que permita solamente el acceso al objeto a través de un objeto proxy sustituto, que será el responsable de controlar o mejorar el acceso al objeto real[26, 39, 41, 42]. Este patrón se evidencia mediante el uso del ORM (*Object Relational Mapper*) de Odoo para el trabajo con la base de datos.
- **Singleton**: el patrón singleton es un patrón de diseño de software que restringe la creación de instancias de una clase a un objeto. Esto es útil cuando se necesita exactamente un objeto para coordinar acciones en todo el sistema[26, 42, 43]. Este patrón se manifiesta en la clase *RelationsManagerGraph* la cuál controla el grafo de relaciones de las tablas de la base de datos en toda la plataforma Odoo.

2.4.3. Modelo de datos

La mayoría de los sistemas de información no registran eventos explícitamente. Solo los PAIS almacenan datos de eventos con los atributos mostrados en la Tabla 1.1. Para crear un registro de eventos, a menudo se necesita recopilar datos de diferentes fuentes de datos donde los eventos existen solo de manera implícita. Para la mayoría de los proyectos de Minería de Procesos , los datos de eventos deben extraerse de las bases de datos convencionales.

Las herramientas como XESame [44] y ProMimport [45] proporcionan cierto soporte, pero aún así los registros de eventos deben construirse consultando la base de datos y convirtiendo los registros de la base de datos (fila en tablas) en eventos. Además, las "tablas regulares" en una base de datos solo proporcionan el estado actual del sistema de información y puede ser imposible ver cuándo se creó o actualizó un registro. Además, los registros eliminados son generalmente invisibles. [11]

Tomando el punto de vista de que la base de datos refleja el estado actual de uno o más procesos, es necesario un modelo de datos que funcione como respaldo a los pasos de configuración. Luego de una revisión se decide optar por el modelo de eventos propuesto por Will M.P. van der Aalst en su artículo *"Extracting Event Data from Databases to Unleash Process Mining"* el cuál se encuentra reflejado en la Figura 2.10 una vez fue convertido a modelo de datos.

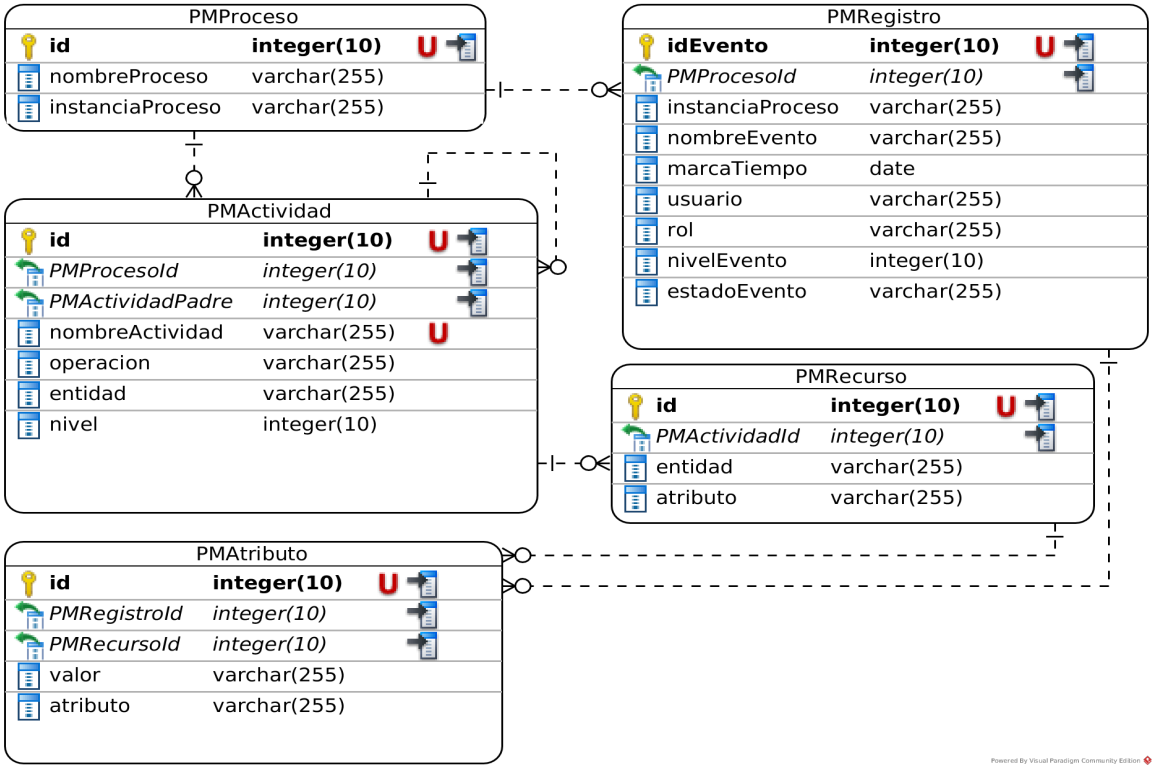

FIGURA 2.10: Modelo de datos

Fuente: Elaboración propia

Este modelo de datos propone cinco entidades para almacenar los casos, los eventos y sus atributos asociados, de forma tal que al momento de exportar los datos como un fichero en formato XES, se encuentre toda la información necesaria en la base de datos para generar un registro de eventos lo más completo y fidedigno posible de acuerdo con los parámetros de configuración. A continuación se describen cada una de las entidades y relaciones definidas en el modelo de datos.

[11] Cada vez más, los sistemas marcan los objetos eliminados como no relevantes (lo que se denomina eliminación suave) en lugar de eliminarlos. De esta forma, se pueden reconstruir todos los estados intermedios de la base de datos. Además, marcar objetos como eliminados en lugar de eliminarlos por completo de la base de datos suele ser más natural, por ejemplo, los conciertos no se eliminan: se cancelan, los empleados no se eliminan, se despiden, etc.

* **Entidad PMPRoceso:** Se encarga de almacenar la información referente a los procesos que se configuran en el sistema. De ellos se almacena el nombre del proceso y el nombre de la tabla que se selecciona como instancia de proceso.

TABLA 2.2: Relaciones de la entidad PMProceso

Entidad relacionada	Tipo de Relación	Propósito
PMActividad	$1..*$	Define que un proceso contiene una o varias actividades
PMRegistro	$1..*$	Define que a un proceso están asociados varios registros, uno por cada evento que se ejecute de las actividades del proceso

* **Entidad PMActividad:** Se encarga de almacenar la información referente a las actividades que se configuran en el sistema. De ellas se almacena el nombre de la actividad, la operación que se ejecutó (insertar, actualizar, eliminar), en el campo entidad se almacena el nombre de la tabla que se selecciona como objeto del negocio central para la actividad y el nivel de la actividad, que permite saber si tiene actividades hijas, además de las llaves foráneas de PMProceso[12] y PMActividad.

TABLA 2.3: Relaciones de la entidad PMActividad

Entidad relacionada	Tipo de Relación	Propósito
PMActividad	$1..*$	Define que una actividad puede o no contener actividades hijas en forma de subactividades
PMRecurso	$1..*$	Define que una actividad puede tener asociados ninguno o varios recursos. Los recursos son campos de la tabla de actividad o de tablas con las que ésta está relacionada que por su importancia para el análisis se deben incluir en el registro de eventos

* **Entidad PMRegistro:** Se encarga de almacenar la información de los eventos que se ejecutan en el sistema. De ellos se almacena el nombre de la instancia de proceso, el nombre del evento ejecutado, la marca de tiempo del momento en que se ejecutó el evento, así como el nombre del usuario y el rol que lo ejecutó, además de la llave foránea de PMProceso[12]. El nivel y el estado del evento son autogenerados basado en configuraciones.

* **Entidad PMRecurso:** Se encarga de almacenar los recursos que se definen para las actividades. De estos se almacena la entidad de la cuál se van a extraer y el nombre del atributo que se va insertar en el registro, además de la llave foránea de PMActividad[13]

[12] La naturaleza de la relación con PMProceso ya se explicó en la Tabla 2.2
[13] La naturaleza de la relación con PMActividad ya se explicó en la Tabla 2.3

TABLA 2.4: Relaciones de la entidad PMRegistro

Entidad relacionada	Tipo de Relación	Propósito
PMAtributo	$1..*$	Define que un registro puede tener asociados uno o varios atributos

TABLA 2.5: Relaciones de la entidad PMRecurso

Entidad relacionada	Tipo de Relación	Propósito
PMAtributo	$1..*$	Define que un recurso puede tener asociados uno o varios atributos

* **Entidad PMAtributo:** Se encarga de almacenar los atributos que se definen para los registros y en el caso de las actividades en forma de recursos. De los atributos se almacena el nombre del atributo y su valor, además de las llaves foráneas de PMRegistro[14] y PMRecurso[15]

Conclusiones del capítulo

La definición de nuevas entidades en la solución propuesta como resultado del análisis permitió establecer una forma organizada de registrar los datos de los eventos del sistema, permitiendo con este nuevo formato confeccionar un registro de eventos.

La aplicación de patrones de diseño y arquitectónicos permitió brindar robustez al código y se proporcionó una estructura que facilita tareas como el mantenimiento y la reusabilidad de la solución.

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

Los algoritmos basados en grafos junto con la técnica de *memoization* permitieron un aumento en la velocidad en los algoritmos de búsqueda de relaciones entre las tablas.

[14] La naturaleza de la relación con PMRegistro ya se explicó en la Tabla 2.4
[15] La naturaleza de la relación con PMRecurso ya se explicó en la Tabla 2.5

Capítulo 3

Implementación y Pruebas

Introducción

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

3.1. Modelo de implementación

El modelo de implementación describe cómo los elementos del modelo de diseño se implementan en términos de componentes. Describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración disponibles en el entorno de implementación y en el lenguaje de programación utilizados; y cómo dependen los componentes unos de otros [46].

3.1.1. Diagrama de componentes

El diagrama de componentes mostrado en la Figura 3.1, representa la relación entre el módulo donde se implementa la solución y los componentes *Odoo Connector* y *Odoo Framework*. En el caso de *Odoo Framework*, provee todas las funcionalidades básicas del marco de trabajo y *Odoo Connector* provee los mecanismos para capturar los eventos.

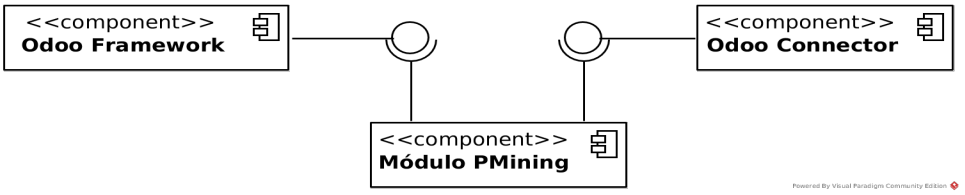

FIGURA 3.1: Diagrama de componentes

Fuente: Elaboración propia

3.1.2. Algoritmos utilizados

La búsqueda de las relaciones entre las tablas de la base de datos y la obtención de los eventos constituyen dos de las funcionalidades claves en la solución. A continuación se describen con mayor nivel de detalle.

1. **Búsqueda de relación entre tablas**

   Cuando se realiza la pre configuración en el sistema se selecciona la tabla sobre la que ocurre una acción. Si se desea insertar una actividad se comprueba que su tabla posee relación con la tabla de la instancia de proceso. Esta comprobación se realiza para determinar si desde la tabla de la instancia se

puede acceder a los valores y atributos de la actividad, de no ser posible no se realiza la inserción. Para determinar esta relación se emplea el servicio *RelationsManagerGraph* el cual implementa el algoritmo BBFS y otros métodos útiles.

Como primer paso, cuando se inserta una actividad o un recurso se capturan en el controlador los nombres de las tablas seleccionadas. A continuación estos datos son enviados mediante el protocolo HTTP al controlador del servidor encargado de manejar las actividades. Este posee el método que se encarga de devolver una respuesta en caso que las entidades se relacionen o no. En la función *create* o *write* del modelo PMActividad se realiza una llamada al servicio *RelationsManagerGraph* y se solicita el método are_related para evaluar si existe relación o no entre las entidades. Este método emplea el algoritmo BBFS, implementado dentro del servicio y mostrado en el Algoritmo 3.1.

```
def bbfs(origen, destino):
    if origen not in self.G or destino not in self.G:
        msg = 'Cualquiera de origen o destino no existen en el grafo.'
        raise exceptions.NodeNotFound(msg.format(source, target))

    resultados = _bidireccional_pred_suc(self.G, origen, destino)
    pred, suc, nodo = results

    # construir camino con base pred+nodo+suc
    camino = []
    while nodo is not None:
        camino.append(nodo)
        nodo = pre[nodo]
    camino.reverse()
    nodo = suc[camino[-1]]
    while nodo is not None:
        camino.append(nodo)
        nodo = suc[nodo]
    return camino
```

ALGORITMO 3.1: BBFS

La función recibe como parámetros un nodo origen y un nodo destino, que representan la tabla origen y la tabla destino, luego el algoritmo se encarga de buscar en el grafo si existe un camino que enlace estas dos tablas. Primero el algoritmo verifica que ambos nodos existan en el grafo, luego hace una búsqueda de los nodos predecesores y sucesores a lo ancho a partir de los nodos destino y origen, respectivamente. Luego se construye el camino utilizando como nexo los sucesores y antecesores progresivos de los nodos destino y origen [30]. El algoritmo devuelve una lista con los nodos intermediarios entre el origen y

el destino. Si la lista es vacía, no existe un camino entre los nodos, en caso contrario sí existe un camino. En cualquier caso se notifica al usuario si el camino existe o no.

2. **Confección del registro de eventos.**

   Para obtener los eventos primeramente se selecciona el identificador del proceso a examinar. Esta información se captura en el controlador de la vista y es enviada al controlador del registro, el cual tiene implementado el servicio que devolverá los eventos. Primero se buscan todos los eventos guardados en la tabla PMRegistro que pertenezcan a la instancia de proceso seleccionada. Si no se hallan eventos se le devuelve una respuesta al controlador de la vista indicando que no hay eventos para la instancia de proceso seleccionada. De lo contrario se organizan los eventos según el identificador de la instancia de proceso y se envían al método que construye el archivo en el formato XES. Para confeccionar el archivo se sigue la estructura de los documentos creados en XES. Inicialmente se construye la cabecera con los valores de la versión y las extensiones que se utilizan en el archivo. A continuación se insertan de las extensiones el nombre, el prefijo que utilizan y la URI donde se encuentran disponibles. En el Algoritmo B.6 se muestra el algoritmo que construye la cabecera y el método que se encarga de adicionar las extensiones por defecto que se incluirán en el archivo XES.

```
def __init__(self):
    self.CREATOR = "Python XES v1.2"
    self.log = et.Element("log")
    self.log.set("xes.version", "1.0")
    self.log.set("xmlns", "http://www.xes-standard.org")
    self.log.set("xes.creator", self.CREATOR)

def add_default_extensions(self):
    self.extensions = [
        Extension(name="Concept",
            prefix="concept",
            uri="http://www.xes-standard.org/concept.xesext"),
        Extension(name="Lifecycle",
            prefix="lifecycle",
            uri="http://www.xes-standard.org/lifecycle.xesext"),
        Extension(name="Time",
            prefix="time",
            uri="http://www.xes-standard.org/time.xesext"),
        Extension(name="Organizational",
            prefix="org",
            uri="http://www.xes-standard.org/org.xesext")
    ]
```

ALGORITMO 3.2: Creación de cabecera y extensiones por defecto

3.1.3. Diagrama de despliegue

A continuación se muestra en la Figura 3.2 el diagrama de despliegue de la solución implementada. En la computadora cliente se accede al sistema mediante el protocolo HTTP. El sistema se halla instalado y configurado en un servidor web, y maneja las peticiones que realiza el cliente. La información del sistema, de las acciones ejecutadas y del negocio se halla almacenada en la base de datos.

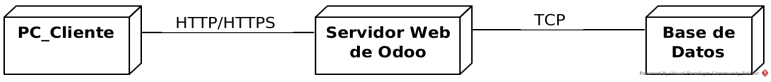

FIGURA 3.2: Diagrama de despliegue

Fuente: Elaboración propia

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 [47]. Las nomenclaturas utilizadas y el estilo de código es el especificado por la guía de estilo PEP8, que es el estándar que se sigue en el lenguaje Python.

3.2.1. Nomenclaturas utilizadas

Nombres de paquetes y módulos

Los módulos deben tener un nombre corto y en minúscula. Se separan con guiones bajos en lugar de espacios. Los paquetes en Python también deberían tener un nombre corto y en minúscula.[48].

Nombres de clases

Los nombres de clases deben utilizar la convención “CapWords” (palabras que comienzan con mayúsculas). Clases para uso interno tienen un guión bajo como prefijo [48].

Nombres de excepciones

Debido a que las excepciones deben ser clases, se aplica la convención anterior.Se debe usar un sufijo “Error” en los nombres de excepciones (en caso que corresponda a un error) [48].

Nombres de funciones

Las funciones deben ser en minúscula, con las palabras separadas por un guión bajo [48].

Argumentos de funciones y métodos

Siempre usar self para el primer argumento de los métodos de instancia. Siempre usar cls para el primer argumento de los métodos de clase. Agregar un guión bajo como sufijo antes de usar una abreviación u ortografía incorrecta [48].

Constantes

Las constantes definidas a nivel módulo, escritas con todas las letras en mayúscula y con guiones bajos separando palabras. Por ejemplo, MAX_OVERFLOW y TOTAL [48].

3.2.2. Estilo de código

En el estilo de código también se utilizan las convenciones de código especificadas en el PEP 8. Las más importantes que se usaron en el desarrollo del módulo son las siguientes:

* Usa 4 (cuatro) espacios por indentación [48].
* El paréntesis / corchete / llave que cierre una asignación debe estar alineado con el primer carácter que no sea un espacio en blanco o puede ser alineado con el carácter inicial de la primera línea [48].
* Limita todas las líneas a un máximo de 79 caracteres [48].
* Separa funciones de alto nivel y definiciones de clase con dos líneas en blanco [48].
* Definiciones de métodos dentro de una clase son separadas por una línea en blanco [48].
* Líneas en blanco adicionales pueden ser utilizadas (escasamente) para separar grupos de funciones relacionadas. Se pueden omitir entre un grupo (de funciones) de una línea relacionadas (por ejemplo, un conjunto de implementaciones ficticias) [48].
* Usa líneas en blanco en funciones, escasamente, para indicar secciones lógicas [48].
* El código en el núcleo de la distribución de Python siempre debería utilizar la codificación ASCII o Latin-1 (alias ISO-8859-1). Para Python 3.0 y en adelante, UTF-8 es preferible ante Latin-1 [48].
* Archivos usando ASCII no deberían tener una “coding cookie” (especificación de la codificación al comienzo del archivo); de lo contrario, usar \x, \u o \U es la manera preferida para incluir caracteres que no correspondan a dicha codificación en cadenas (strings) [48].
* Las importaciones deben estar en líneas separadas [48].

* Las importaciones siempre se colocan al comienzo del archivo, simplemente luego de cualquier comentario o documentación del módulo, y antes de globales y constantes [48].

* Las importaciones deben estar agrupadas en el siguiente orden:[48]

    1. Importaciones de la librería estándar
    2. Importaciones terceras relacionadas
    3. Importaciones locales de la aplicación / librería

* Siempre rodea estos operadores binarios con un espacio en cada lado: asignación (=), asignación de aumentación (+=, -=, etc.), comparaciones (==, ¡, ¿, !=, ¡¿, ¡=, ¿=, in, not in, is, is not), “Booleans” (and, or, not) [48].

3.3. Métricas de diseño de software

Las métricas de diseño de software permiten medir de forma cuantitativa la calidad de los atributos internos del software. Esto proporciona una vía para evaluar la calidad durante el desarrollo del sistema. A nivel de componentes se concentran en las características internas de los componentes del software con medidas que ayudan a juzgar la calidad del diseño [49]. En el caso del módulo de Odoo que se desarrolla se aplica Tamaño Operacional de Clases (WMC) y Relación entre Clases (CBO). [50–52]

3.3.1. Tamaño operacional de las clases (TOC)

Está dado por el número de métodos u operaciones (de instancia privada y heredada) que están encapsulados dentro o por una clase. Evalúa los siguientes atributos de calidad mostrados en la Tabla 3.1 [49, 53, 54].

TABLA 3.1: Atributos de calidad que evalúa TOC

Atributo	**Efecto en el diseño**
Responsabilidad	Aumento del TOC provoca aumento de la responsabilidad asignada a la clase.
Complejidad de Implementación	Aumento del TOC provoca aumento de la complejidad de implementación de la clase.
Reutilización	Aumento del TOC provoca disminución del grado de reutilización de la clase.

Para evaluar los atributos se emplean los siguientes criterios y categorías de evaluación presentes en la Tabla 3.2 [49]

TABLA 3.2: Criterios y Categorías de evaluación de TOC

Atributo	**Categoría**	**Criterio**
Responsabilidad	Baja	$<= Prom$
	Media	Entre $[Prom; 2 * Prom]$
	Alta	$> 2 * Prom$
Complejidad de Implementación	Baja	$<= Prom$
	Media	Entre $[Prom; 2 * Prom]$
	Alta	$> 2 * Prom$
Reutilización	Baja	$> 2 * Prom$
	Media	Entre $[Prom; 2 * Prom]$
	Alta	$<= Prom$

Luego de aplicar la métrica a las principales clases del módulo, se obtiene como resultado que un 65.22 % de las clases poseen Responsabilidad y Complejidad bajas, y un 17.39 % de las clases poseen Reutilización alta y media, como se aprecia en las figuras 3.3, 3.4 y 3.5.

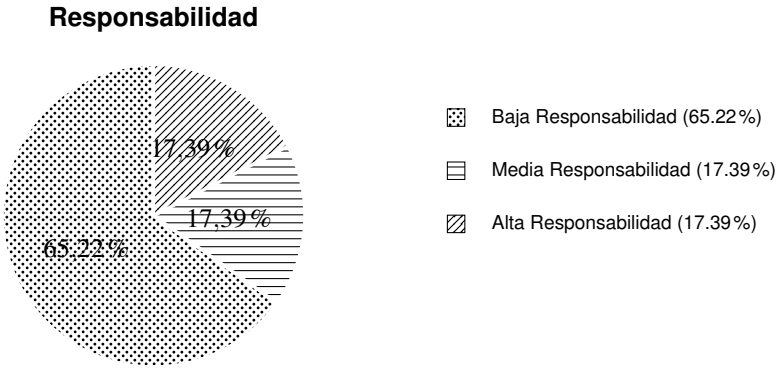

FIGURA 3.3: Resultados del análisis de responsabilidad

Fuente: Elaboración propia

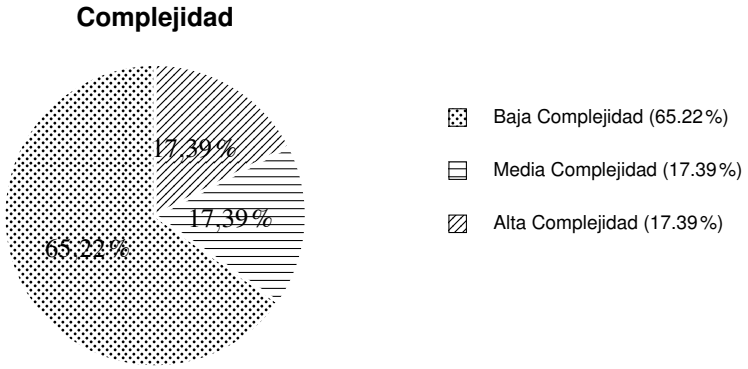

FIGURA 3.4: Resultados del análisis de complejidad

Fuente: Elaboración propia

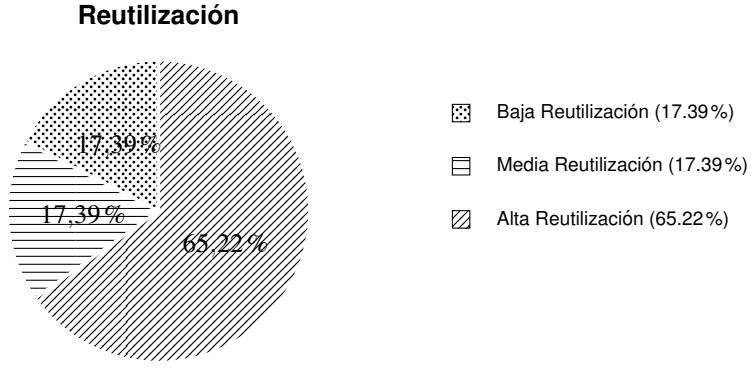

FIGURA 3.5: Resultados del análisis de reutilización

Fuente: Elaboración propia

En la Tabla 3.3 se observa con mayor nivel de detalle los resultados de aplicar la métrica TOC.

TABLA 3.3: Resultados de aplicar TOC

Clase	**Cantidad de procedimientos**	**Responsabilidad**	**Complejidad**	**Reutilización**
RelationTypes	0	Baja	Baja	Alta
RelationsManagerGraph	11	Alta	Alta	Baja
Log	11	Alta	Alta	Baja
Event	4	Media	Media	Media
Attribute	3	Media	Media	Media
Trace	5	Media	Media	Media
Extension	2	Baja	Baja	Alta
Classifier	2	Baja	Baja	Alta
XESFileBuilder	6	Alta	Alta	Baja
EventTracker	9	Alta	Alta	Baja
GeneralException	0	Baja	Baja	Alta
MisconfigurationException	1	Baja	Baja	Alta
MisconfigurationError	0	Baja	Baja	Alta
NoPathBetweenModels	1	Baja	Baja	Alta
NoSuchRelationException	1	Baja	Baja	Alta
Pmining	1	Baja	Baja	Alta
PMProcess	1	Baja	Baja	Alta
PMActivity	1	Baja	Baja	Alta
PMRegister	0	Baja	Baja	Alta
PMResource	1	Baja	Baja	Alta
PMAttribute	1	Baja	Baja	Alta
PMModel	1	Baja	Baja	Alta
Base	3	Media	Media	Media

3.3.2. Relaciones entre clases (RC)

Esta métrica se define con el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad presentes en la Tabla 3.4 [49, 53, 54].

Para la evaluación de dichos atributos de calidad, se definen los siguientes criterios y categorías de evaluación mostrados en la Tabla 3.5:

Tras evaluar las relaciones de las clases en el módulo, se obtiene como resultado que un 65.22% de las clases posee complejidad de mantenimiento baja, cantidad de pruebas baja y reutilización alta y un 56.22% posee un bajo acoplamiento como se muestra en las figuras 3.6, 3.7, 3.8 y 3.9.

TABLA 3.4: Atributos de calidad evaluados por la métrica RC

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

TABLA 3.5: Criterios y Categorías de evaluación de RC

Atributo	**Categoría**	**Criterio**
Acoplamiento	Ninguno	0
	Baja	1
	Media	2
	Alta	> 2
Complejidad de Mantenimiento	Baja	$<= Prom$
	Media	Entre $[Prom; 2 * Prom]$
	Alta	$> 2 * Prom$
Reutilización	Baja	$> 2 * Prom$
	Media	Entre $[Prom; 2 * Prom]$
	Alta	$<= Prom$
Cantidad de Pruebas	Baja	$<= Prom$
	Media	Entre $[Prom; 2 * Prom]$
	Alta	$> 2 * Prom$

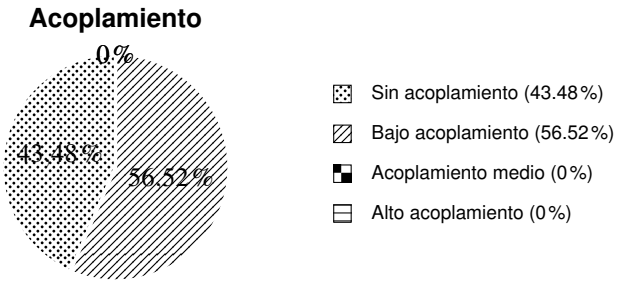

FIGURA 3.6: Resultados del análisis de acoplamiento

Fuente: Elaboración propia

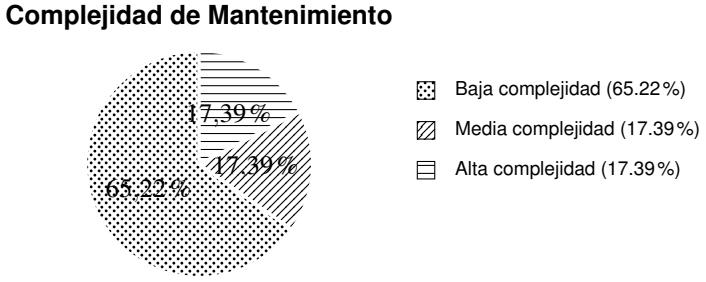

FIGURA 3.7: Resultados del análisis de complejidad de mantenimiento

Fuente: Elaboración propia

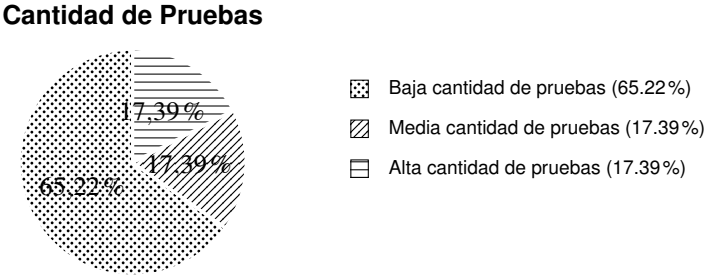

FIGURA 3.8: Resultados del análisis de cantidad de pruebas

Fuente: Elaboración propia

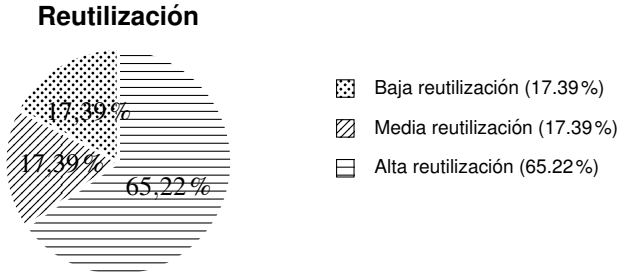

FIGURA 3.9: Resultados del análisis de reutilización

Fuente: Elaboración propia

En la Tabla 3.6 se observa con mayor detalle los resultados de aplicar RC. [55]

TABLA 3.6: Resultados de aplicar RC

Clase	Cantidad de procedimientos	Complejidad de Mantenimiento	Cantidad de Pruebas	Reutilización	Acoplamiento
RelationTypes	0	Baja	Baja	Alta	Baja
RelationsManagerGraph	11	Alta	Alta	Baja	Ninguno
Log	11	Alta	Alta	Baja	Ninguno
Event	4	Media	Media	Media	Ninguno
Attribute	3	Media	Media	Media	Ninguno
Trace	5	Media	Media	Media	Ninguno
Extension	2	Baja	Baja	Alta	Ninguno
Classifier	2	Baja	Baja	Alta	Ninguno
XESFileBuilder	6	Alta	Alta	Baja	Ninguno
EventTracker	9	Alta	Alta	Baja	Ninguno
GeneralException	0	Baja	Baja	Alta	Baja
MisconfigurationException	1	Baja	Baja	Alta	Baja
MisconfigurationError	0	Baja	Baja	Alta	Ninguno
NoPathBetweenModels	1	Baja	Baja	Alta	Baja
NoSuchRelationException	1	Baja	Baja	Alta	Baja
Pmining	1	Baja	Baja	Alta	Baja
PMProcess	1	Baja	Baja	Alta	Baja
PMActivity	1	Baja	Baja	Alta	Baja
PMRegister	0	Baja	Baja	Alta	Baja
PMResource	1	Baja	Baja	Alta	Baja
PMAttribute	1	Baja	Baja	Alta	Baja
PMModel	1	Baja	Baja	Alta	Baja
Base	3	Media	Media	Media	Baja

3.3.3. Resultados de la aplicación de las métricas

Como resultado de aplicar las métricas a la solución desarrollada se obtiene que las clases poseen bajos niveles de responsabilidad dentro del dominio de la solución y baja complejidad de implementación. Los niveles de acoplamiento también son bajos y significa que el grado de interconexión entre clases es pequeño. Esta característica influye en la reutilización de la solución, a menores niveles de dependencia entre las clases mayor reutilización siendo este el caso. Una complejidad de mantenimiento baja indica que el grado de esfuerzo necesario para desarrollar un arreglo, una mejora o una rectificación de algún error será bajo y tendrá poco impacto en el sistema. El bajo nivel de la cantidad de pruebas indica que el esfuerzo para probar el componente mediante pruebas de calidad será bajo también.

3.4. Pruebas de caja blanca

La prueba de caja blanca es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Mediante estos métodos el ingeniero de software puede

obtener casos de prueba que garanticen que: se ejercita por lo menos una vez todos los caminos independientes de cada módulo; ejerciten todas las decisiones lógicas en sus vertientes verdaderas y falsas; ejecuten todos los bucles en sus límites y con sus limitaciones operacionales; y ejerciten las estructuras internas de datos para asegurar su validez [56].

La técnica a aplicar es el camino básico, que permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y utilizar esta medida como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa [56].

Para obtener el conjunto de caminos independientes se construye el grafo de flujo asociado a una función y se calcula su complejidad ciclomática, para este caso se analiza el método *get_resources()* encargado de obtener los recursos que el usuario configura para las actividades.

```
def get_resources(attribute_value, path, resource_model, attribute_name):
    if isinstance(attribute_value, list): (1)
        if isinstance(attribute_value[0], resource_model): (2)
            return attribute_value (3)
        else: (4)
            resources = [] (5)
            for value in attribute_value: (6)
                _object = traverse_path(value, path, resource_model, attribute_name)
                if isinstance(_object, list): (7)
                    resources.extend(_object) (8)
                else: (9)
                    resources.append(_object) (10)
            return resources (11)
    else: (12)
        data = traverse_path(attribute_value, path, resource_model, attribute_name) (13)
        return extract_resources(data, resource_model, attribute_name) \ (16)
(17) (14) if isinstance(data, list) > 0 else getattr(data, attribute_name) (15)
```

ALGORITMO 3.3: Obtención de los recursos de las actividades

Una vez obtenidas las sentencias se construye el grafo, quedando en la forma mostrada en la Figura 3.10

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. Provee el límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez [56].

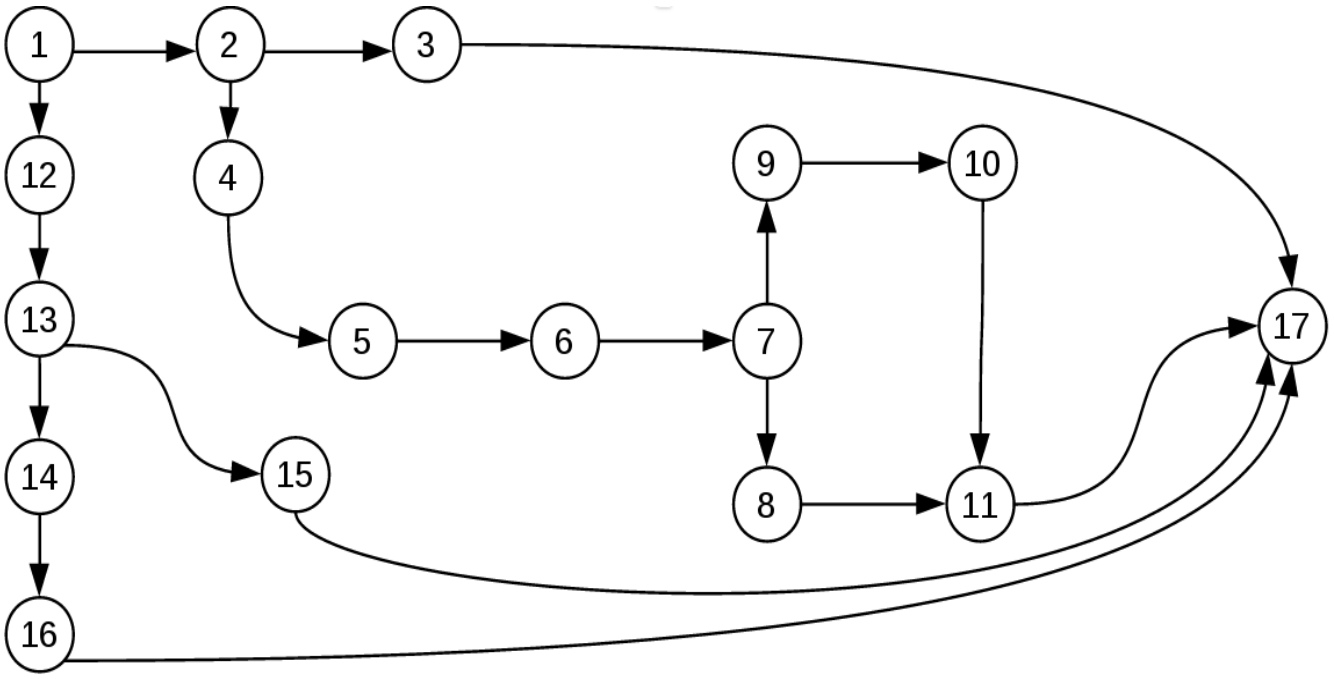

FIGURA 3.10: Grafo construido a partir del flujo del método

Fuente: Elaboración propia

Se calcula de la siguiente forma:

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

$V(G) = 20 - 17 + 2$

$V(G) = 5$

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

* Camino básico 1: 1, 12, 13, 14, 16, 17.
* Camino básico 2: 1, 12, 13, 15, 17.
* Camino básico 3: 1, 2, 4, 5, 6, 7, 8, 11, 17.
* Camino básico 4: 1, 2, 4, 5, 6, 7, 9, 10, 11, 17.
* Camino básico 5: 1, 2, 3, 17.

Luego se elaboran los casos de prueba para su aplicación en forma de pruebas unitarias, quedando de la siguiente manera:

* Caso de prueba del camino 1

  Condición: `isinstace(attribute_value, list) == False && isinstace(attribute_value, list) ==`

`True`

Resultado esperado: El sistema busca los caminos según el valor actual hasta que lo encuentra y devuelve los recursos asociados.

* Caso de prueba del camino 2

  Condición:`isinstace(attribute_value, list) == False && isinstace(attribute_value, list) == False`

  Resultado esperado: El sistema retorna el recurso asociado al valor actual.

* Caso de prueba del camino 3

  Condición: `isinstace(attribute_value, list) == True && isinstace(attribute_value[0], resource_model) == False && isinstace(_object, list) == True`

  Resultado esperado: El sistema verifica que tiene como objeto una lista de recursos y busca los caminos para cada uno de ellos, luego encuentra otra lista de recursos y fusiona las listas para luego retornar la lista fusionada.

* Caso de prueba del camino 4

  Condición: `isinstace(attribute_value, list) == True && isinstace(attribute_value[0], resource_model) == False && isinstace(_object, list) == False`

  Resultado esperado: El sistema verifica que tiene como objeto un solo recurso y busca los caminos para el recurso y los adiciona a una lista que luego retorna.

* Caso de prueba del camino 5

  Condición: `isinstace(attribute_value, list) == True && isinstace(attribute_value[0], resource_model) == True`

  Resultado esperado: El sistema detecta que esta trabajando con una lista y que su primer elemento es instancia del recurso que se busca, entonces lo devuelve como recurso.

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

3.5. Pruebas de caja negra

Las pruebas de caja negra se centran en los requisitos funcionales. O sea, la prueba de caja negra permite al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa [56].

La técnica de prueba a aplicar es la partición equivalente. Esta técnica perteneciente al método de caja negra divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores (por ejemplo, proceso incorrecto de

todos los datos de carácter) que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar [56].

En el caso de la solución propuesta se diseñaron 9 casos de prueba, un caso de prueba por requisito funcional exceptuando el caso del requisito funcional “Generar registro de eventos, debido a que este no posee una interfaz gráfica”. A continuación se muestran los resultados de las pruebas realizadas en la Tabla 3.7. Las pruebas fueron realizadas por el grupo de calidad del centro CEIGE.

TABLA 3.7: No conformidades detectadas por iteración

No conformidades	**Aplicación**
Primera Iteración	0

El gráfico de la Figura 3.11 ilustra estos valores:

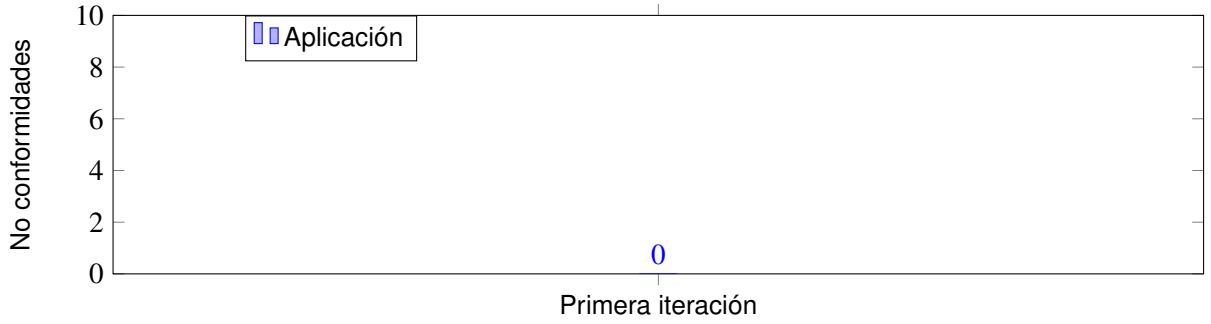

FIGURA 3.11: No conformidades detectadas por iteración (gráfica)

Fuente: Elaboración propia

3.6. Aplicación de la solución en un caso de estudio

Para validar si el registro de eventos confeccionado es factible para aplicarle Minería de Procesos, en el módulo que contempla la solución se diseña un caso de estudio que simula el comportamiento de un administrador de Odoo con la gestión de los usuarios. El proceso se llama “Gestionar Usuario” y comprende las siguientes actividades:

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

El diseño del caso de estudio consta de 21 casos y 82 eventos. Los eventos se ejecutaron de forma aleatoria y no se ejecutó ningún evento “Asignar Cuenta” debido a que requiere que exista una pasarela de pago y

TABLA 3.8: Actividades del caso de estudio y su equivalente representación en el modelo de Van der Aalst

Actividades	**Operación**	**Tabla**
Insertar usuario	INSERT - ⊕	User
Eliminar usuario	DELETE - ⊘	User
Modificar usuario	UPDATE - ⊖	User
Asignar empresa	UPDATE - ⊖	Company
Asignar cuenta	INSERT - ⊕	Account
Crear contacto	INSERT - ⊕	Contact
Eliminar contacto	DELETE - ⊘	Contact
Modificar contacto	UPDATE - ⊖	Contact
Asignar teléfono	INSERT - ⊕	Phone
Asignar email	INSERT - ⊕	Email

sincronización de banca configurada en el sistema. La única restricción que se documenta en la ayuda del módulo es que para adicionar una cuenta, el usuario debe tener un correo electrónico definido.

Luego de la aplicación del caso de estudio a la solución la herramienta ProM detecta los elementos definidos en la configuración de la solución correctamente. Se detectan los 21 casos y los 82 eventos ejecutados en el sistema (Figura 3.12 y Figura 3.13). En la Figura 3.14 se puede ver la salida del algoritmo *Inductive Visual Miner* que confirma que las actividades referentes al usuario se pueden realizar en cualquier orden. Por otra parte para adicionar una cuenta, primero hay que asignar un email al usuario. Los resultados arrojados por este análisis coinciden con los resultados esperados, por tanto, se afirma que el registro de eventos generado por el Componente de trazas es válido para aplicar Minería de Procesos.

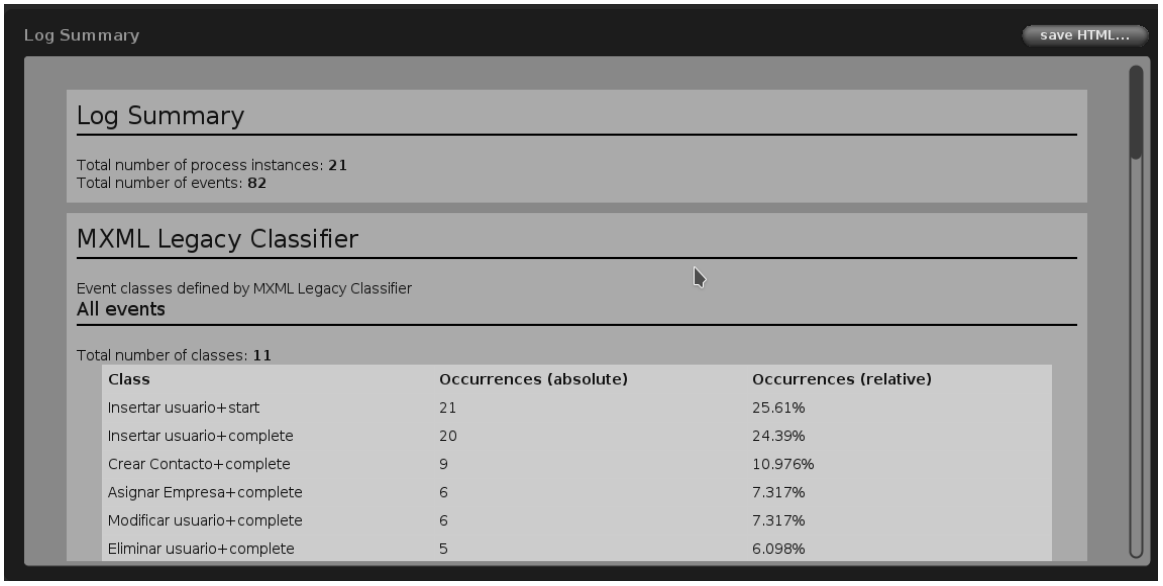

FIGURA 3.12: Información de los eventos descubiertos

Fuente: Elaboración propia

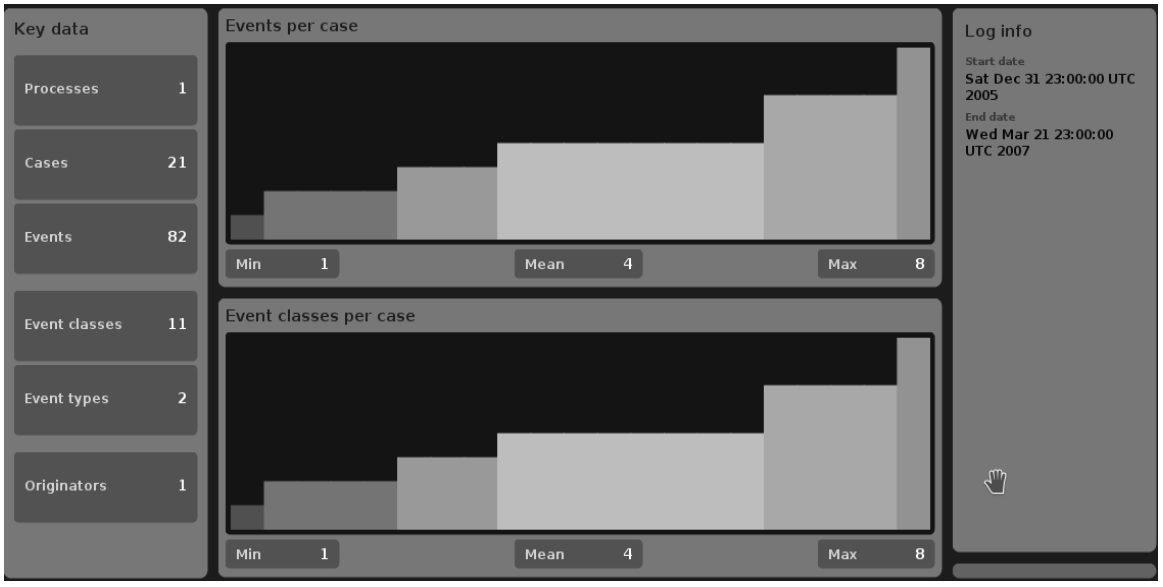

FIGURA 3.13: Información de los eventos descubiertos(gráfica).

Fuente: Elaboración propia

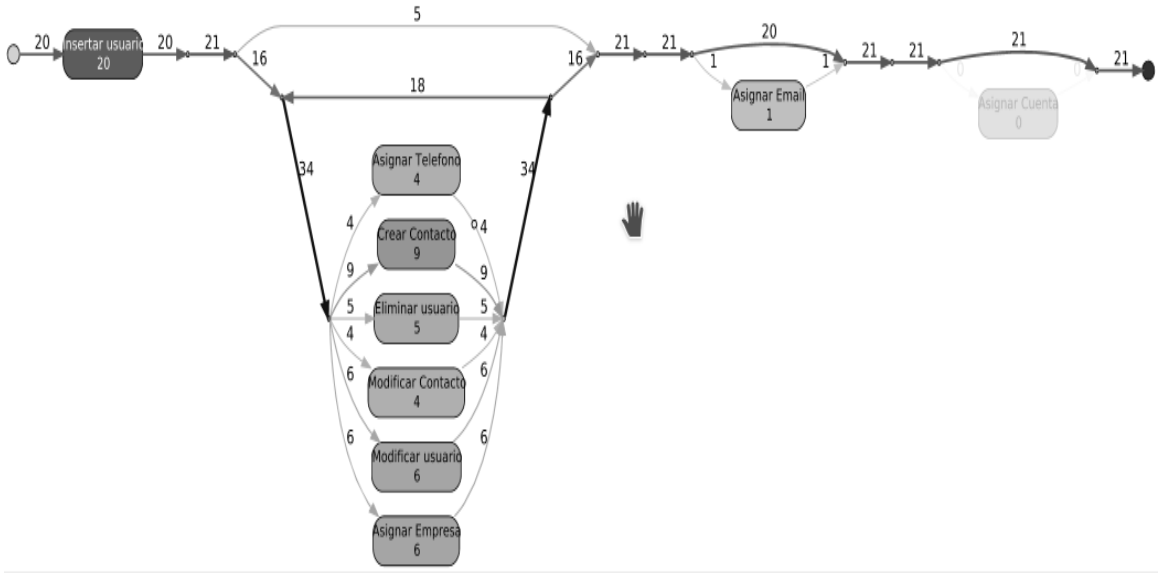

FIGURA 3.14: Flujo de eventos reconocido por la herramienta.

Fuente: Elaboración propia

Este resultado también evidencia el cumplimento de los principios planteados por Wil van der Aalst en su propuesta de modelo de eventos [22]:

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

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

* P6: Los eventos deberían estar al menos parcialmente ordenados.

  En el caso de la solución propuesta los eventos cuentan con la marca de tiempo y están agrupados por trazas que se corresponden con la instancia de proceso a la que están asociados. Su orden puede ser almacenado explícitamente (empleando listas) o implícitamente (empleando marcas de tiempo).

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

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

3.7. Validación de la sintaxis del registro de eventos

Una de las características que debe poseer un registro de eventos factible para la Minería de Procesos es que debe ser sintáctica y semánticamente correcto. En la presente investigación no se trata el problema de la semántica, pero sí el de la sintáctica. Existe una herramienta en línea desarrollada por Fluxicon la compañía que desarrolló la herramienta ProM que permite validar si un archivo XML sigue un esquema de definición[57, 58]. Luego de la ejecución del caso de estudio, el registro de eventos dado como salida, fue sometido a la evaluación de la herramienta y dió como resultado que el registro analizado (según la definición de esquema para un registro de eventos) era válido, el proceso se evidencia en la Figura 3.15 y la Figura 3.16.

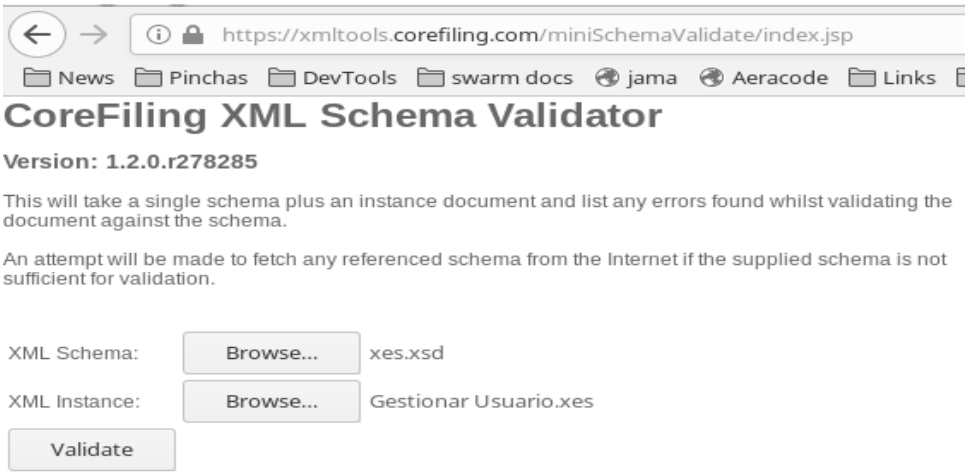

FIGURA 3.15: Ficheros cargados en la herramienta antes de validar

Fuente: Elaboración propia

FIGURA 3.16: Resultado de la herramienta

Fuente: Elaboración propia

Conclusiones del capítulo

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

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

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

La modelación de un caso de estudio y su aplicación generó un registro de eventos que fue analizado por las herramientas de Minería de Procesos obteniendo los resultados esperados.

Conclusiones

1. Los enfoques encontrados en la revisión bibliográfica de las soluciones existentes para obtener un registro de eventos a partir de sistemas orientados a datos, permitió definir un proceso de configuración acorde con los datos, extensiones y clasificadores definidos en el estándar XES.

2. La aplicación de patrones de diseño y arquitectónicos permitieron desarrollar una solución adaptable, mantenible y reutilizable.

3. La aplicación de métricas del diseño y las pruebas de software realizadas demostraron que se obtuvo una solución libre de errores y validó el diseño creado para la solución.

4. La aplicación de un caso de estudio aprovechando la solución generó un registro de eventos que las herramientas de Minería de Procesos reconocieron y analizaron correctamente, alcanzado los resultados esperados.

Recomendaciones

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

El modelo para el soporte de la jerarquía entre actividades padres e hijas de Reinhold Dunkl ya está definido e implementado en la aplicación, pero se necesita un algoritmo que pueda utilizar este modelo para obtener las actividades compuestas y conocer cuándo estas finalizan y comienzan. Por ello, se recomienda extender la aplicación con el desarrollo de un algoritmo que cumpla este cometido.

Referencias bibliográficas

[1] Marianne Bradford. *Modern ERP: select, implement, and use today's advanced business systems*. Lulu. com, 2015.

[2] Van Der Aalst. Process mining discovery, conformance and enhancement of business processes, 2011.

[3] Wil Van Der Aalst and Kees Max Van Hee. *Workflow management: models, methods, and systems*. MIT press, 2004.

[4] Wil MP Van der Aalst, Boudewijn F van Dongen, Joachim Herbst, Laura Maruster, Guido Schimm, and Anton JMM Weijters. Workflow mining: A survey of issues and approaches. *Data & knowledge engineering*, 47(2):237–267, 2003.

[5] Rakesh Agrawal, Dimitrios Gunopulos, and Frank Leymann. Mining process models from workflow logs. *Advances in Database Technology—EDBT'98*, pages 467–483, 1998.

[6] Daniela Grigori, Fabio Casati, Umeshwar Dayal, and Ming-Chien Shan. Improving business process quality through exception understanding, prediction, and prevention. In *VLDB*, volume 1, pages 159–168, 2001.

[7] Mehmet Sayal, Fabio Casati, Umeshwar Dayal, and Ming-Chien Shan. Business process cockpit. In *Proceedings of the 28th international conference on Very Large Data Bases*, pages 880–883. VLDB Endowment, 2002.

[8] Thomas Hoffman. Sarbanes-oxley sparks forensics apps interest: vendors offer monitoring tools to help identify incidents of financial fraud. *ComputerWorld*, 38:14–14, 2004.

[9] Wil Van Der Aalst, Arya Adriansyah, Ana Karla Alves De Medeiros, Franco Arcieri, Thomas Baier, Tobias Blickle, Jagadeesh Chandra Bose, Peter van den Brand, Ronald Brandtjen, Joos Buijs, et al. Process mining manifesto. In *International Conference on Business Process Management*, pages 169–194. Springer, 2011.

[10] Roberto Hernández Sampieri, Carlos Fernández Collado, and Pilar Baptista Lucio. Metodología de la investigación. sexta edición. editorial mc graw hill. méxico. 2014• hernández, r. *Metodología de la Investigación. 6a Edición, Mc Graw Hill, México*, 2014.

[11] Graciela Elisa Barchini. Métodos i+ d de la informática. *Revista de Informática Educativa y Medios Audiovisuales*, 2(5):16–24, 2005.

[12] Alter Steven. Information systems: A management perspective, 2000.

[13] Marlon Dumas, Wil M Van der Aalst, and Arthur H Ter Hofstede. *Process-aware information systems: bridging people and software through process technology*. John Wiley & Sons, 2005.

[14] JCAM Buijs. Mapping data sources to xes in a generic way. *Maters Thesis*, 2010.

[15] Carlos Alberto Peguero Álvarez, Héctor David y Torres Peregrino. Módulo para el registro y transformación de trazas de eventos a formato xes. 2013.

[16] Christian W Günther and Eric Verbeek. Xes standard definition version 2.0. *Eindhoven University of Technology*, 2014.

[17] Eindhoven Research Group. Micro event extension. *Eindhoven University of Technology*, 2016.

[18] Best ERP Software | 2017 Reviews of the Most Popular Systems. URL http://www.capterra.com/enterprise-resource-planning-software/?utf8=%E2%9C%93&review_stars=4&users=&feature%5B0%5D=18559&feature%5B1%5D=18563&feature%5B2%5D=18568&feature%5B5%5D=18569&feature%5B6%5D=18571&feature%5B9%5D=18579&feature%5B10%5D=18582&commit=Filter+Results.

[19] Open source erp and crm, . URL https://www.odoo.com/.

[20] Odoo Community Association (OCA) ABF OSIELL.

[21] Carlos Rodrıguez, Robert Engel, Galena Kostoska, Florian Daniel, Fabio Casati, Marco Aimar, et al. Eventifier: Extracting process execution logs from operational databases. In *Demonstration Track of BPM Conference, CEUR-WS*, pages 17–22. Citeseer, 2012.

[22] Wil MP Van der Aalst. Extracting event data from databases to unleash process mining. In *BPM-Driving innovation in a digital world*, pages 105–128. Springer, 2015.

[23] RS Mans and WMP van der Aalst. Wanna improve process mining results. *It's high time we consider data quality issues seriously. BPMcenter. org*, 2013.

[24] Xixi Lu. Artifact-centric log extraction and process discovery. *Unpublished master's thesis, Eindhoven University of Technology*, 2013.

[25] Alan Dennis, Barbara Haley Wixom, and David Tegarden. *Systems analysis and design: An object-oriented approach with UML*. John wiley & sons, 2015.

[26] Erich Gamma. *Design patterns: elements of reusable object-oriented software*. Pearson Education India, 1995.

[27] connector/component_event at 10.0 · OCA/connector, . URL https://github.com/OCA/connector/tree/10.0/component_event.

[28] Reinhold Dunkl. Data improvement to enable process mining on integrated non-log data sources. In *International Conference on Computer Aided Systems Theory*, pages 491–498. Springer, 2013.

[29] Thomas H Cormen. *Introduction to algorithms*. MIT press, 2009.

[30] Rong Zhou and Eric A Hansen. Breadth-first heuristic search. *Artificial Intelligence*, 170(4-5):385–408, 2006.

[31] Andrew V Goldberg and Chris Harrelson. Computing the shortest path: A search meets graph theory. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 156–165. Society for Industrial and Applied Mathematics, 2005.

[32] Sparsh Mittal. A survey of techniques for approximate computing. *ACM Computing Surveys (CSUR)*, 48 (4):62, 2016.

[33] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

[34] Pierre Bourque, Richard E Fairley, et al. *Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0*. IEEE Computer Society Press, 2014.

[35] IEEE Standards Coordinating Committee et al. Ieee standard glossary of software engineering terminology (ieee std 610.12-1990). los alamitos. *CA: IEEE Computer Society*, 169, 1990.

[36] Paul Ralph and Yair Wand. A proposal for a formal definition of the design concept. In *Design requirements engineering: A ten-year perspective*, pages 103–136. Springer, 2009.

[37] Peter Freeman and David Hart. A science of design for software-intensive systems. *Communications of the ACM*, 47(8):19–21, 2004.

[38] Architecture — OpenERP Server Developers Documentation 7.0b documentation, . URL `http://odoo-docs.readthedocs.io/en/latest/02_architecture.html`.

[39] Apostolos Ampatzoglou, Sofia Charalampidou, and Ioannis Stamelos. Research state of the art on gof design patterns: A mapping study. *Journal of Systems and Software*, 86(7):1945–1964, 2013.

[40] Craig Larman. *Applying UML and Patterns: An Introduction to Object Oriented Analysis and Design and Interative Development*. Pearson Education India, 2012.

[41] Mawal Ali and Mahmoud O Elish. A comparative literature survey of design patterns impact on software quality. In *Information Science and Applications (ICISA), 2013 International Conference on*, pages 1–7. IEEE, 2013.

[42] Apostolos Ampatzoglou, Alexander Chatzigeorgiou, Sofia Charalampidou, and Paris Avgeriou. The effect of gof design patterns on stability: a case study. *IEEE Transactions on Software Engineering*, 41(8): 781–802, 2015.

[43] Adel Noureddine and Ajitha Rajan. Optimising energy consumption of design patterns. In *Proceedings of the 37th International Conference on Software Engineering-Volume 2*, pages 623–626. IEEE Press, 2015.

[44] HMW Verbeek, Joos CAM Buijs, Boudewijn F Van Dongen, and Wil MP Van Der Aalst. Xes, xesame, and prom 6. In *Forum at the Conference on Advanced Information Systems Engineering (CAiSE)*, pages 60–75. Springer, 2010.

[45] Christian W Günther and Wil MP van der Aalst. A generic import framework for process event logs. In *International Conference on Business Process Management*, pages 81–92. Springer, 2006.

[46] Ivar Jacobson. *The unified software development process*. Pearson Education India, 1999.

[47] Damián Pérez Alfonso. Normas y estándares de codificación, 2012.

[48] PEP 8 – Style Guide for Python Code | Python.org. URL `https://www.python.org/dev/peps/pep-0008/`.

[49] Unidad 5 | IngSoftwareII-CUFM. URL `https://cufmingsoftware.wordpress.com/estandares-de-diseno/`.

[50] Shyam R Chidamber and Chris F Kemerer. *Towards a metrics suite for object oriented design*, volume 26. ACM, 1991.

[51] Shyam R Chidamber and Chris F Kemerer. A metrics suite for object oriented design. *IEEE Transactions on software engineering*, 20(6):476–493, 1994.

[52] Victor R Basili, Lionel C. Briand, and Walcélio L Melo. A validation of object-oriented design metrics as quality indicators. *IEEE Transactions on software engineering*, 22(10):751–761, 1996.

[53] Gagandeep Singh. Metrics for measuring the quality of object-oriented software. *ACM SIGSOFT Software Engineering Notes*, 38(5):1–5, 2013.

[54] Elvira Maria Arvanitou, Apostolos Ampatzoglou, Alexander Chatzigeorgiou, Matthias Galster, and Paris Avgeriou. A mapping study on design-time quality attributes and metrics. *Journal of Systems and Software*, 127:52–77, 2017.

[55] Brij Mohan Goel and Pradeep Kumar Bhatia. Analysis of reusability of object-oriented systems using object-oriented metrics. *ACM SIGSOFT Software Engineering Notes*, 38(4):1–5, 2013.

[56] Roger S Pressman. Software engineering, a practical approach. *McGraw-Hill*, 1997.

[57] Definition xsd (xml schema definition), June 2014. URL `http://searchsoa.techtarget.com/definition/XSD`.

[58] Corefiling xml schema validator, January 2016. URL https://xmltools.corefiling.com/miniSchemaValidate/index.jsp.

[59] pip — pip 9.0.3 documentation, . URL https://pip.pypa.io/en/stable/.

[60] PostgreSQL: About, . URL https://www.postgresql.org/about/.

[61] Features - PyCharm, . URL https://www.jetbrains.com/pycharm/features/.

[62] About Visual Paradigm, . URL https://www.visual-paradigm.com/aboutus/.

[63] odoo/odoo: Odoo. Open Source Apps To Grow Your Business., . URL https://github.com/odoo/odoo.

[64] Odoo Connector — Connector documentation, . URL http://odoo-connector.com/.

[65] Dave Kuhlman. *A python book: Beginning python, advanced python, and python exercises (rev1.3a)*. December 2013.

Anexos

Anexo A: Descripción de Herramientas y Tecnologías

A.1.1. Pip. v9.0.1

Pip es un sistema de gestión de paquetes utilizado para instalar y gestionar paquetes de software escritos en Python. [59]

A.1.2. PostgreSQL. v9.5

PostgreSQL es un poderoso sistema de base de datos relacional de objetos de código abierto. Tiene más de 15 años de desarrollo activo y una arquitectura comprobada que le ha valido una sólida reputación de fiabilidad, integridad de datos y corrección. Se ejecuta en todos los principales sistemas operativos, incluidos Linux, UNIX (AIX, BSD, HP-UX, macOS, Solaris) y Windows. Es totalmente compatible con ACID, tiene soporte completo para claves externas, combinaciones, vistas, disparadores y procedimientos almacenados (en varios idiomas). Incluye la mayoría de los tipos de datos SQL: 2008, incluidos INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL y TIMESTAMP. También es compatible con el almacenamiento de objetos grandes binarios, incluyendo imágenes, sonidos o video. Tiene interfaces de programación nativas para C / C ++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, entre otros, y documentación excepcional. [60]

A.1.3. Pycharm Community Edition. v2016.3.1

PyCharm proporciona completado inteligente de códigos, inspecciones de códigos, resaltado de errores sobre la marcha y soluciones rápidas, junto con refactorizaciones automáticas de códigos y capacidades de navegación avanzadas. La enorme colección de herramientas de PyCharm incluye un depurador y corrector de prueba integrado; Python Profiler; una terminal incorporada; integración con VCS principal y herramientas de base de datos incorporadas; capacidades de desarrollo remoto con intérpretes remotos; un terminal ssh integrado; e integración con Docker y Vagrant. Además de Python, PyCharm brinda soporte de primera clase para varios frameworks de desarrollo web de Python, lenguajes de plantillas específicos, JavaScript, CoffeeScript, TypeScript, HTML / CSS, AngularJS, Node.js y más. PyCharm se integra con IPython Notebook, tiene una consola interactiva de Python y es compatible con Anaconda, así como con múltiples paquetes científicos, incluidos Matplotlib y NumPy. [61]

A.1.4. Visual Paradigm. v15

Visual Paradigm permite que su equipo administre la complejidad de la transformación empresarial para hacer frente a los mercados, las tecnologías y los requisitos regulatorios que cambian rápidamente. Es una solución

de ventanilla única ideal para la planificación de la arquitectura empresarial y la transformación empresarial, la gestión de proyectos y el desarrollo de software ágil, para que su empresa pueda mantener el control y fomentar el crecimiento. [62]

A.1.5. Odoo. v10

Odoo es un conjunto de aplicaciones empresariales de código abierto basadas en la web. Las principales aplicaciones de Odoo incluyen un CRM de código abierto, creador de sitios web, comercio electrónico, gestión de almacenes, gestión de proyectos, facturación y contabilidad, puntos de venta, recursos humanos, marketing, fabricación y gestión de compras. Las aplicaciones Odoo se pueden utilizar como aplicaciones independientes, pero también se integran perfectamente para que pueda obtener un ERP de código abierto con todas las funciones cuando instala varias aplicaciones. [63]

A.1.6. Odoo Connector v1.0.0

Odoo Connector es un potente marco para desarrollar cualquier tipo de conector bidireccional entre Odoo (Open Source ERP) y cualquier otro software o servicio. Este complemento Odoo tiene un núcleo modular y genérico, con la posibilidad de ampliarse con módulos adicionales para nuevas características o personalizaciones. [64]

A.1.7. Python. v2.7

Python es un lenguaje de programación de alto nivel interpretado para programación de propósito general. Creado por Guido van Rossum y lanzado por primera vez en 1991, Python tiene una filosofía de diseño que enfatiza la legibilidad del código, notable a través de su uso significativo de espacios en blanco. Proporciona construcciones que permiten una programación clara en escalas pequeñas y grandes. [65]

A.1.8. Core Filing XML Schema Validator. v1.2.0.r278285

Herramienta en línea desarrollada por Fluxicon la compañía que desarrolló la herramienta ProM. Permite validar si un archivo XML sigue un esquema de definición. La herramienta valida el fichero utilizando un validador de esquema XSD según definición de la W3C. El estándar especifica cómo describir formalmente los elementos en un fichero con formato XML. Chequea si cada uno de los elementos en el fichero XML se adhiere a la definición[57, 58].

Anexo B: Algoritmos usados en la solución

B.1.1. Algoritmo para la obtención de la instancia de proceso

```
def get_process_instance_id_from_activity(env, activity, values, _id, record):
    process_instance_table = activity.process.instance
    process_instance_model = env[process_instance_table]

    entity_name = activity.entity
    entity_model = env[entity_name]

    # check first if the activity table is the process instance table
    if process_instance_model == entity_model:
        return _id
    g = RELATIONS_GRAPH
    path = g.get_process_to_activity_path(process_instance_table, entity_name)
    path.reverse()
    _object = entity_model.search([('id', '=', _id)])[0]
    i = 0

    while not isinstance(_object, type(process_instance_model)) and i < len(path) - 1:
        from_model = path[i]
        to_model = path[i + 1]
        field = g.edge[from_model][to_model]['field']
        print field
        print type(field)
        if isinstance(field, list):
            field = [x for x in field if x['type'] == RelationTypes.many_to_one]
            field = field[0]
        lookup_field = field
        _object = getattr(_object, lookup_field)
        i += 1
    return _object.id
```

ALGORITMO B.4: Algoritmo para la obtención de la instancia de proceso

B.1.2. Algoritmo para la obtención de atributos

```
def get_resources(attribute_value, path, resource_model, attribute_name):
   if isinstance(attribute_value, list):
      if isinstance(attribute_value[0], resource_model):
         return attribute_value
      else:
         resources = []
         for value in attribute_value:
            _object = traverse_path(value, path, resource_model, attribute_name)
            if isinstance(_object, list):
               resources.extend(_object)
            else:
               resources.append(_object)
         return resources
   else:
      data = traverse_path(attribute_value, path, resource_model, attribute_name)
      return extract_resources(data, resource_model, attribute_name) \
         if isinstance(data, list) > 0 else getattr(data, attribute_name)
```

ALGORITMO B.5: Algoritmo para la obtención de atributos

B.1.3. Algoritmo de la técnica de memoization

```
def get_shortest_path(self, origin_model, destiny_model):
   try:
      path = PATH_CACHE[origin_model][destiny_model]
      return path
   except KeyError:
      path = nx.shortest_path(self._graph, origin_model, destiny_model)
      PATH_CACHE[origin_model][destiny_model] = path
      return path
```

ALGORITMO B.6: Creación de cabecera y extensiones por defecto