Proyecto de Grado

Facultad de Ingeniería, UdelaR

Calidad de datos en logs de eventos para minería de procesos de negocios

Montevideo, 2022

Alumnos:
Francisco Betancor
Federico Pérez

Tutores:
Andrea Delgado
Adriana Marotta

Resumen del trabajo

La Calidad de Datos (CD) es un elemento clave en un proyecto de Ciencia de Datos (DS) para garantizar que sus resultados proveen información consistente y confiable.

Tanto la minería de procesos como la minería de datos, como parte de DS, operan sobre grandes conjuntos de datos pertenecientes a la organización, llevando a cabo el esfuerzo de análisis.

En la minería de procesos, la entrada básica es el log de eventos, que incluye las instancias de ejecución de un proceso, con sus eventos (actividades) ordenados, fechas de ejecución, responsables, entre otros datos. Estos datos son utilizados para descubrir modelos de procesos, chequear conformidad entre modelos existentes y datos de su ejecución, y extensión de modelos con información de la operativa diaria.

Por otro lado, en la minería de datos se trabaja con datos organizacionales relacionados al dominio de la organización, como pueden ser clientes, ventas, pacientes, entre otros. El objetivo de esta disciplina es descubrir patrones y relaciones e información no conocida sobre los datos analizados.

Este manejo separado de los datos impide a las organizaciones tener una visión completa de su operacion diaria y evaluacion correspondiente, probablemente ocultando informacion util para mejorar sus procesos. A pesar de que existen varios enfoques de CD y modelos para datos organizacionales, y algunas propuestas de CD para datos de procesos de negocios, ninguno de ellos toma una visión integrada sobre los procesos y los datos organizacionales.

En un trabajo previo se trabajó sobre la integración de datos de procesos y organizacionales, para realizar análisis más completos sobre la ejecución de procesos y datos. A partir de tal trabajo surge este proyecyo, donde se presenta un modelo de calidad de datos llamado Business Process and Organizational Data Quality Model (BPODQM), el cual define específicamente dimensiones, factores y métricas para la evaluación de calidad de datos de procesos y organizacionales, para así en forma previa a realizar cualquier análisis, poder detectar problemas claves en los conjuntos de datos asociados a los procesos.

A partir de esto se realizó el diseño, la implementación y documentación de un plug-in para la herramienta de minería de procesos ProM, el cual permite aplicar el modelo de calidad definido a un log de eventos que contenga datos de un proceso y datos organizacionales integrados.

Se llevó a cabo un caso de estudio sobre un log de eventos con datos basados en un proceso de negocio real, al cual se le aplicó el modelo de calidad propuesto y se analizaron los problemas de calidad del mismo.

Palabras clave: Modelo de Calidad de Datos. Minería de procesos y Minería de datos. Data Science. Datos de procesos y datos organizacionales integrados.

Tabla de contenido

1. Introducción

La calidad de datos (CD) es un elemento clave en cualquier proyecto de Data Science (DS) para garantizar que sus resultados brinden información consistente y confiable. Tanto la minería de procesos como la minería de datos, como parte de DS, operan sobre grandes conjuntos de datos pertenecientes a la organización, llevando a cabo el esfuerzo de análisis. En el primer caso, los datos representan la ejecución diaria de procesos de negocios (BPs) en la organización, tales como procesos de ventas o procesos de salud, y en el segundo caso, los datos corresponden a datos organizacionales relacionados al dominio de la organizacion, como pueden ser clientes, ventas, pacientes, entre otros. Un modelo de CD define los aspectos de calidad a medir y sobre qué datos se aplican estas mediciones.

En este proyecto se presenta un modelo de calidad de datos llamado Business Process and Organizational Data Quality Model (BPODQM), el cual define específicamente dimensiones, factores y métricas para la evaluación de calidad de datos de procesos y organizacionales, para asi poder detectar problemas claves en los conjuntos de datos asociados a los procesos.

1.1. Motivación

La necesidad de explotar los datos disponibles en las organizaciones se ha incrementado considerablemente en los últimos años, debido a la continua generación de datos de fuentes diferentes e interconectadas. La complejidad de los sistemas socio-técnicos que conectan personas, objetos, datos y BPs, integrando tecnologías y elementos heterogéneos, también aumenta la complejidad de integrar y generar conjuntos de datos útiles, así como su gestión. Al mismo tiempo, para que los datos puedan ser explotados generando beneficios, es necesario conocer su calidad y mejorarla.

Aunque existen varios enfoques y modelos de CD para datos organizacionales, y algunas propuestas de CD para datos de BP, ninguno de ellos toma una visión integrada sobre los procesos y los datos organizacionales. En trabajos anteriores en el Instituto de Computación se ha abordado este problema definiendo un framework integrado [1], trabajando con procesos integrados y datos organizacionales [2], así como identificando la necesidad de aplicar un modelo CD sobre los datos antes de que se pueda llevar a cabo el esfuerzo de la minería.

1.2. Objetivos

El objetivo principal de este proyecto es definir un modelo de calidad para datos integrados de procesos y organizacionales que permita analizar la calidad de los datos de un log de eventos que soporte datos de procesos y organizacionales integrados.

Para llevarlo a cabo se analizarán tanto distintas propuestas existentes de modelos de CD, como conceptos de calidad de datos relacionados a log de eventos de BP (O1). En base a esto se definirán las distintas dimensiones, factores, métricas y métodos que construyen a la propuesta de modelo de CD para log de eventos con estas

características (O2).

A partir de esto se requiere implementar un prototipo de plug-in para la herramienta de gestión de procesos ProM [3], el cual pueda aplicar el modelo de CD a un log de eventos de entrada (O3). Luego, se espera realizar un caso de estudio tomando como entrada un log de eventos de un proceso real que maneje datos del proceso y organizacionales integrados (O4).

El proyecto de grado se enmarca en el proyecto de investigación "Minería de procesos y datos para la mejora de procesos en las organizaciones", financiado por Comisión Sectorial de Investigación Científica (CSIC), UdelaR y con participación de AGESIC. Como complemento a este documento, se publicó un artículo en [4], donde se muestra una aproximación al problema que se resuelve en este proyecto.

1.3. Estructura del documento

El informe consta de 7 capítulos, en el capitulo 2 "Marco Conceptual" se presentan conceptos básicos de procesos de negocio, log de eventos, calidad de datos y modelos de calidad de datos, así como herramientas y propuestas existentes.

En el capítulo 3 "Análisis del Problema" se presenta el contexto de este trabajo. Primero se describe el problema planteado y luego la solución propuesta a dicho problema. A continuación se describe el BP "Student Mobility", el cual es un proceso real del cual se cuenta con una implementación y generación de log de eventos extendido con datos integrados, que se utilizará como caso de estudio.

En el capítulo 4 "Modelo de Calidad BPODQM" se presenta lo que es el foco principal de este trabajo, el modelo de CD para BP y datos organizacionales BPODQM. Se realiza una descripción general del modelo, en donde se define el formato de los datos obtenidos junto con las granularidades correspondientes y luego se presenta la estructura del modelo. Además se describe de manera concisa las dimensiones, factores y métricas que lo componen.

El capítulo 5 "Desarrollo del Prototipo" se divide en dos partes. Primero se realiza un análisis de lo que se intenta resolver con esta herramienta y se presenta la arquitectura de la misma. Luego se describe la implementación del plug-in desarrollado en ProM, mostrando como se estructura y las decisiones que se tomaron para llevarlo a cabo.

En el capítulo 6 "Caso de Estudio" se realiza un caso de estudio, y se presenta un análisis con los datos obtenidos.

En el capítulo 7 "Conclusiones y Trabajos futuros" se presentan las conclusiones acerca del trabajo, cuales fueron los logros obtenidos, las limitaciones identificadas, y posibles trabajos a futuro que serían de interés abordar.

2. Marco Conceptual

En este capítulo se presentan los conceptos necesarios para entender el marco conceptual del proyecto. Se introducen conceptos básicos de procesos de negocio, log de eventos, calidad de datos y calidad de datos en log de eventos.

2.1. Procesos de Negocio

En esta sección se presentan conceptos asociados a procesos de negocio. Primero se describe la gestión de procesos de negocio, también conocida como Business Process Management. Luego se presenta la disciplina de Minería de Procesos. Por último se describe el concepto de Log de Eventos.

2.1.1. Business Process Management (BPM)

"Un Proceso de Negocio es un conjunto de actividades realizadas en coordinación en un entorno organizacional y técnico, para alcanzar un objetivo del negocio." [5]

"(BPM) incluye conceptos, métodos y técnicas para apoyar el diseño, administración, configuración, Ejecución y análisis de procesos de negocios."[5]

En otras palabras, BPM es la disciplina que se encarga de gestionar el ciclo de vida de los procesos de negocios.

El ciclo de vida de un proceso de negocio consiste en cuatro etapas relacionadas entre sí y estructuradas de forma cíclica; Diseño y Análisis, Configuración, Ejecución y Evaluación [5]. El ciclo de vida de un proceso de negocio se puede ver en la Figura 1.

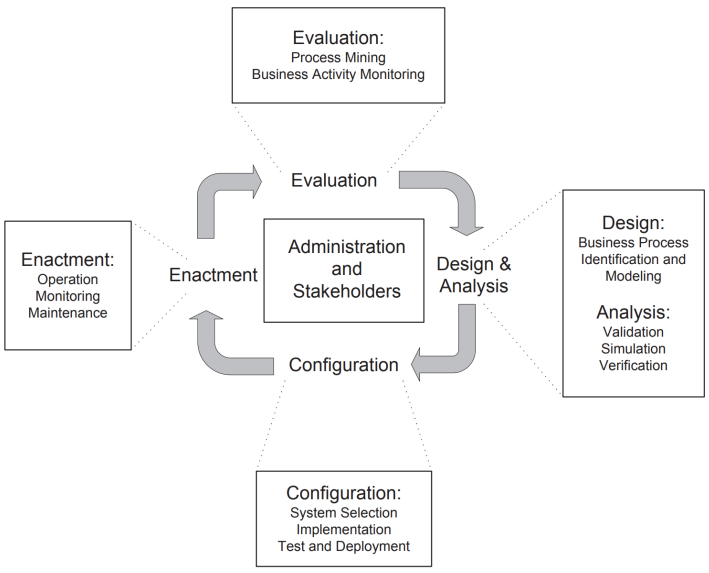


Figura 1: Ciclo de vida de un BP, extraído de [5]

La actividad principal de la etapa de Diseño y Análisis es el Modelado del proceso

de negocio mediante un lenguaje estandarizado que explicite todas las actividades de un proceso de negocio y las restricciones de ejecución entre ellas. Esta actividad es esencial realizarla antes de comenzar la implementación, ya que permite realizar análisis y reflexiones que pueden terminar en posibles re-diseños del modelo.

A lo largo del tiempo han existido diversos lenguajes o notaciones para el modelado de procesos de negocio. Todos los lenguajes existentes son útiles para distintas necesidades, uno de los lenguajes más utilizados en la comunidad de procesos es el estándar Business Process Model and Notation 2.0 (BPMN 2.0) [6].

Un ejemplo de proceso de negocio es la Figura 2, donde la ejecución del proceso comienza con un evento de inicio, a continuación una actividad *Place Order*, luego se abre una compuerta de tipo AND, donde el flujo de ejecución del proceso continua en dos ramas en paralelo. En una rama, se ejecutan las actividades *Receive Invoice* y *Settle Invoice* respectivamente. En la otra rama, se ejecuta la actividad *Receive Products*. Cuando ambas ramas son completadas se cierra la compuerta AND y el proceso finaliza con un evento de fin.

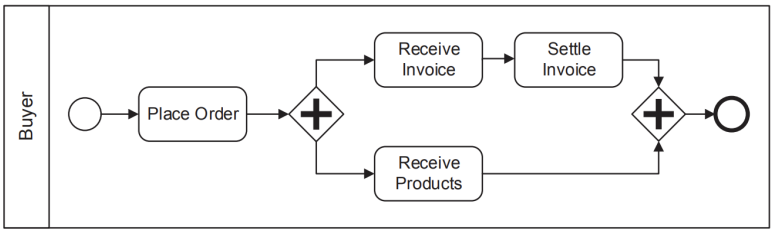


Figura 2: Proceso de Negocio, extraído de [5]

Los procesos de negocios se pueden separar en dos tipos; Orquestación y Colaboración. Una Orquestación es un proceso el cual su flujo de ejecución es controlado por una única organización. Es decir, cada tarea o actividad de la ejecución de dicho proceso es interna a la organización. [5] Un ejemplo de Orquestación es el proceso de la figura 2.

En cambio una Colaboración es un proceso colaborativo en el que participan dos o mas organizaciones, y su flujo de ejecución es compartido entre estas organizaciones mediante intercambio de mensajes. [5]

Un concepto importante en los procesos de negocios es el flujo de secuencia. El flujo de secuencia conecta los objetos de un proceso y define el orden de flujo entre ellos. Cada actividad de un proceso puede tener uno o mas flujos de entrada y uno o mas flujos de salida, además un proceso tiene que tener por lo menos un evento de inicio y un evento de fin. El flujo de secuencia determina los distintos posibles caminos en el modelo que puede tener una instancia del proceso. [5]

Una vez diseñado y verificado el modelo del proceso de negocio, se pasa a la etapa de configuración. En esta etapa se traduce el modelo (en caso de ser necesario), se completa y se implementa con el fin de convertirlo en un ejecutable.

Esta implementación consta de actividades como realizar la interacción con cada usuario del sistema (crear roles, implementar formularios, desplegar mensajes, etc), configurar las integraciones con sistemas externos, configurar la base de datos empresarial, etc. Luego de realizadas dichas actividades se procede a generar el ejecutable del proceso de negocio.[5]

Una vez obtenido el ejecutable del proceso, se pasa a la etapa de ejecución. En esta etapa ya se cuenta con la ejecución en tiempo real del proceso, donde se lleva a cabo el monitoreo de las instancias del proceso para así saber el estado de ejecución de cada una de ellas.

Durante la etapa de Ejecución se ejecutan eventos en los cuales se recolectan datos de valor, los cuales se conocen como event logs. Los event logs son registros de eventos que ocurren durante la ejecución de un proceso de negocio. Esta información es de relevante importancia para la etapa de Evaluación.[5]

En las etapas de Configuración y Ejecución es indispensable contar con un herramienta de gestión de procesos de negocio. Este tipo de herramientas son conocidas como BPMS (Business Process Management System).

En la etapa de evaluación se toman los datos recolectados en la parte de ejecución para evaluar y mejorar el proceso de negocio. El objetivo de esto es evaluar la calidad de dicho proceso y su adecuación con el entorno. Este objetivo se lleva a cabo mediante monitoreo activo y el uso de técnicas de minería de procesos. El monitoreo activo de un proceso de negocio es una actividad en la que se pueden detectar problemas de performance, como por ejemplo que ciertas actividades demoran demasiado tiempo en ejecutarse.[5]

2.1.2. Minería de Procesos

Como es mencionado anteriormente los procesos son una parte integral de la actualidad, cumpliendo funcionalidades internas en empresas, a lo largo del mundo. Si bien hay muchos sistemas disponibles para apoyar la ejecución de estos procesos, las prácticas actuales para monitorizar y analizar esta ejecución en la realidad organizacional aún es algo que no tiene la suficiente madurez. [7]

"Es una disciplina de investigación relativamente joven que se ubica entre el aprendizaje automático y la minería de datos, por un lado, y el modelado y análisis de procesos, por otro lado." [7]

"La idea de la minería de procesos es descubrir, monitorear y mejorar procesos reales extrayendo conocimiento de los logs de eventos disponibles en los sistemas actuales." [7]

La minería de procesos brinda información sobre varias perspectivas, como la perspectiva del proceso, el rendimiento, los datos y la perspectiva de la organización. Su objetivo es descubrir, monitorizar y mejorar procesos reales mediante la extracción de conocimiento de los logs de eventos disponibles en los sistemas actuales. [7]

La mayoría de los Sistemas de Información (SI) almacenan la información de una forma no estructurada. Los datos de los eventos se encuentran dispersos en muchas tablas o deben extraerse de los subsistemas para intercambiar mensajes. En estos casos, existen event logs, pero se necesitan algunos esfuerzos extras para extraerlos. La extracción de datos es una parte integral de cualquier esfuerzo de minería de procesos. [7]

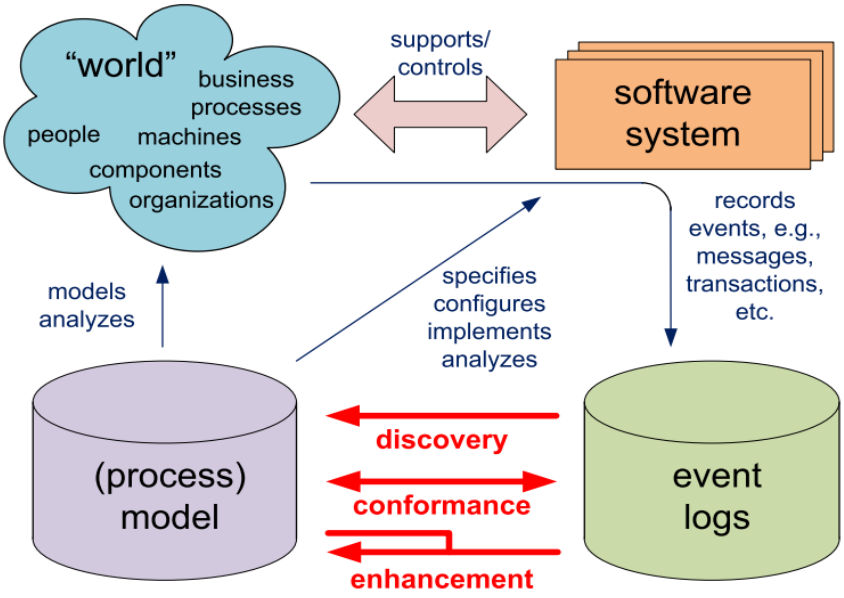


Figura 3: Tipos de minerías apartir de event logs, extraído de [7].

En la figura 3 se observa que se pueden realizar tres tipos de minerías de procesos utilizando los event logs [7]:

- Descubrimiento(discovery): toma un registro de eventos y produce un modelo sin usar ninguna información a priori.
- Conformidad(conformance): Un modelo de proceso existente se compara con un registro de eventos del mismo proceso. Se puede usar para verificar si la realidad, es tal como se registra en el log, se ajusta al modelo y viceversa. Puede ser util para descubrir, por ejemplo, posibles casos de fraude.
- Extensión(enhancement): Se extiende o mejora un modelo de proceso existente utilizando información sobre el proceso real registrado en algún registro de eventos. A diferencia de Conformidad que mide la alineacion entre la realidad y el modelo, Extensión tiene como objetivo modificar el modelo a priori. Uno de los casos en los que se aplica este tipo de minieria de procesos es la modificacion de un modelo para adaptarlo de mejor forma a la realidad.

La Minería de Procesos se puede utilizar para identificar distintos cuellos de botella en los procesos. Hay cuatro perspectivas mineras diferentes [7]:

- Flujo de control: se centra en el orden de las actividades del proceso.
- Organizacional: se centra en la información de recursos en el registro.
- Del caso: se centra en las propiedades de los casos.

- Temporal: se centra en el tiempo y la frecuencia de los eventos.

Poniendo en practica las distintas perspectivas en conjunto a los distintos tipos de minería de procesos se pueden realizar distintos análisis teniendo como objetivo descubrir nuevos conocimientos sobre el estado actual de un proceso dentro de una organización. Y de esta forma se puede mejorar en los lugares donde el trabajo no va según lo planeado o como se esperaba, de modo que se pueda mejorar el rendimiento. Este tipo de información es realmente valiosa para las organizaciones, especialmente si estos problemas se pueden solucionar, y de esta forma mejorar los procesos existentes.

2.1.3. Log de Eventos

Un log de eventos es básicamente un registro de las actividades realizadas en la ejecución de un proceso. Contiene información acerca de eventos correspondientes a una actividad y a una instancia del proceso, conocida también como caso. Una actividad pueden ser una tarea, operación, acción, la cual es ejecutada en un caso.

Típicamente los eventos tienen un timestamp que indica la fecha y hora en que ocurrió el evento. Además un evento puede contener información sobre la persona que ejecuto o inicio el evento.

En la Figura 4 se muestra la estructura de los logs de eventos. Se puede observar que:

- Un proceso consiste de casos.
- Un caso consiste de eventos, tal que cada evento solo se relaciona con un caso.
- Los eventos dentro de un caso están ordenados.
- Los eventos pueden tener atributos. Como pueden ser el timestamp, la persona involucrada, la actividad, etc.

Los eventos para un caso son representados como una secuencia de eventos únicos, la cual es conocida como traza. Es decir que una traza es una secuencia de eventos tales que cada ejecución de un evento aparece solo una vez.

Un log de eventos es un conjunto de casos tales que cada caso aparece como máximo una vez en todo el log. Todo caso y evento debe estar representado con un único identificador. Esto es de mucha utilidad ya que nos permite hacer referencia a un evento o un caso en particular.

El formato definido para los logs de eventos es el Extensible Event Stream que es un estándar basado en XML [8]. El mismo provee un formato genérico para el intercambio de data entre herramientas, y está principalmente enfocado en Process Mining.

En la Figura 5 se muestra un diagrama UML que describe el meta modelo de XES. Como se puede observar en el meta-modelo, el estándar XES se compone de la jerarquía proceso, traza, evento, al igual que un log de eventos.

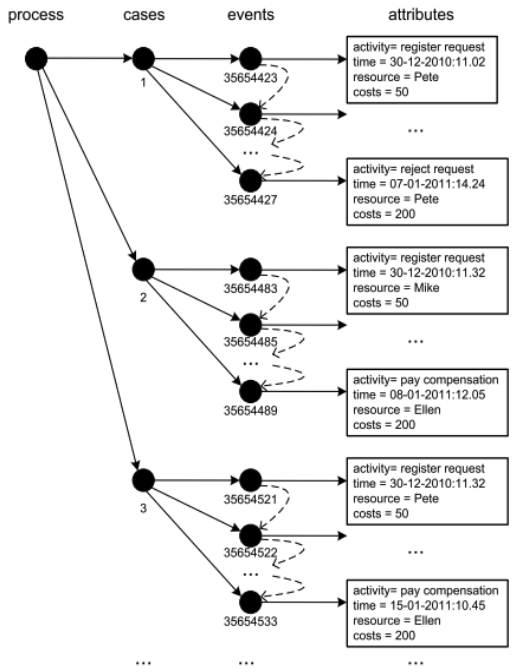


Figura 4: Estructura de un log de eventos, extraído de [7].

También se agrega el concepto de extensión de un log. El uso de extensiones permite introducir atributos que son de ayuda para el análisis de log de eventos. Además también permite la definición de atributos para un dominio de aplicación especifico. XES provee un conjunto de extensiones estándar, pero también es posible definir nuevas extensiones para el mismo.

2.1.4. Herramientas

En ésta sección se presentaran herramientas que son de utilidad en BPM y Minería de Procesos.

BPMS

Los BPMS son sistemas constituidos por un conjunto de herramientas que permiten la configuración y ejecución de procesos de negocios. Existen al día de hoy varios sistemas para gestionar los estándares de modelado de procesos de negocios, los cuales son XPDL, BPEL y el ya mencionado BPMN.

"Un BPMS es un sistema de software genérico que está impulsado por representaciones de procesos explícitos para coordinar la implementación de los procesos de negocio." [5]

A partir del 2011, con la liberación de BPMN 2.0, surgen un tipo de BPMS dedicado a trabajar con procesos modelados con el estándar BPMN 2.0, estos BPMS son conocidos como motores BPMN2. Un motor BPMN es capaz de interpretar un modelo de un proceso con notación BPMN 2.0, y además es capaz de ejecutarlo.

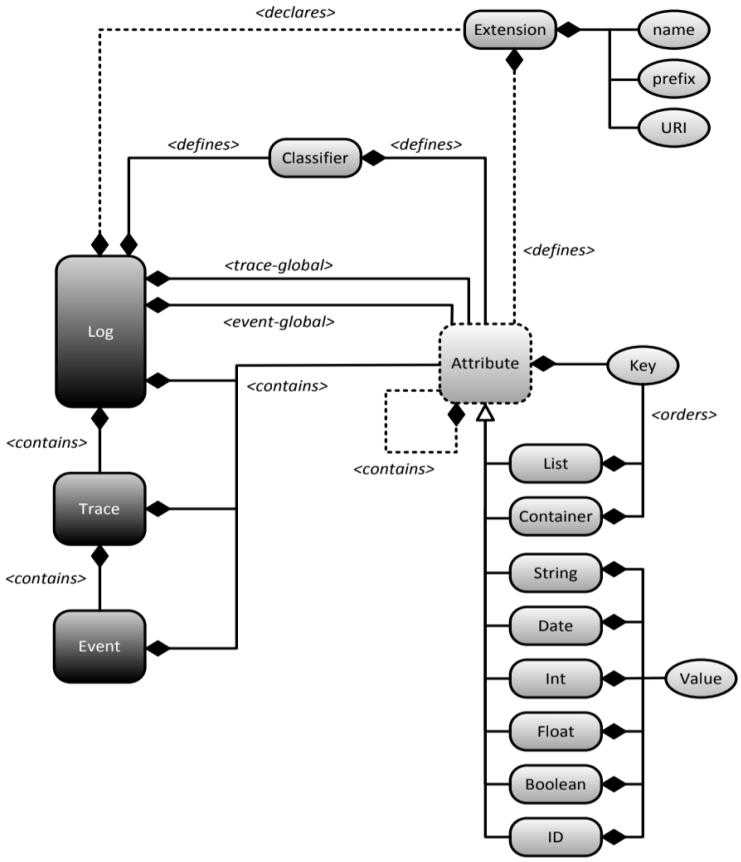


Figura 5: Meta-modelo de XES, extraído de [8].

Minería de Procesos

Existen varias herramientas para trabajar con minería de procesos, una de las cuales es ProM. Esta herramienta, la cual está implementada en Java, es de código abierto y se puede descargar de forma gratuita. La misma proporciona una plataforma para usuarios y desarrolladores de algoritmos de minería de procesos donde se busca la facilidad tanto a la hora de usar como al momento de extender la herramienta.

Posee una amplia variedad de plug-ins de diferentes técnicas de minerías de procesos. Los cuales se pueden clasificar en seis tipos diferentes [9]:

- Plug-ins de minería: implementan un algoritmo de minería, que obtiene como resultado un modelo de proceso.
- Plug-ins de análisis: implementan una propiedad de análisis en resultados de minería anteriores.
- Plug-ins de transformación: implementa transformaciones entre diferentes formatos de datos.
- Plug-ins de visualización: implementa una visualización para diferentes formatos de datos.
- Plug-ins de exportación.
- Plug-ins de importación.

2.2. Calidad de datos

Los datos representan objetos del mundo real en un formato que puede ser:

- Almacenado, recuperado y elaborado por un procedimiento de software.
- Comunicado a través de una red.

Para muchas organizaciones, su actividad principal es procesar datos. Muchas decisiones del negocio se toman en base al procesamiento de los datos, por lo que los mismos tienen un gran valor para las organizaciones. Por esto la calidad de los datos juega un rol fundamental en el desarrollo de la organización. Una buena gestión de la calidad de los datos de una organización facilita el procesamiento de los mismos para luego tomar decisiones óptimas.

Calidad de datos es un concepto muy amplio, que implica muchos aspectos, problemas y desafíos. Existen muchos problemas de calidad los cuales traen graves consecuencias, directas y a mediano/largo plazo, en la eficiencia y efectividad de las organizaciones.

Algunos ejemplos de problemas de calidad son: Datos incorrectos, datos inconsistentes con la realidad, datos inconsistentes entre sí, datos desactualizados, información incompleta, datos poco confiables debido a su fuente, datos difíciles de acceder.

La Gestión de la Calidad de Datos es la tarea de medir, analizar, mejorar y controlar, entre otras, los distintos aspectos de calidad de los datos que son de interés para un escenario específico. Esta tarea implica un conjunto de metodologías, técnicas y herramientas para manejar la calidad de los datos, en una organización o en varias que están cooperando. [10]

En los SI la gestión de calidad se centra en un Modelo de Calidad de Datos, como se puede ver en la Figura 6. Este modelo es el que guía la gestión de la calidad para un conjunto de datos específico. Se necesita definir un modelo de calidad para abordar la calidad de los datos. El mismo debe ser adecuado a las necesidades y prioridades de los consumidores.

2.2.1. Modelo de Calidad de Datos

Un Modelo de Calidad de datos define qué dimensiones de calidad se consideran, sobre que datos se aplican y como se miden. Dado que la calidad de datos se representa por un conjunto de dimensiones que abordan diferentes aspectos de los datos, es posible definir una jerarquía de conceptos de calidad.

En la figura 7 podemos ver la jerarquía de los conceptos de calidad. Donde una **dimensión de calidad** de datos es un conjunto de factores de calidad que tienen el mismo propósito. De estos **factores**, cada uno representa un aspecto particular de una dimensión de calidad. Las **métricas** son instrumentos que definen una forma de medir un factor de calidad, y se pueden usar varias para un mismo factor.

Para definir una métrica se debe tener en cuenta la semántica, la unidad de medición

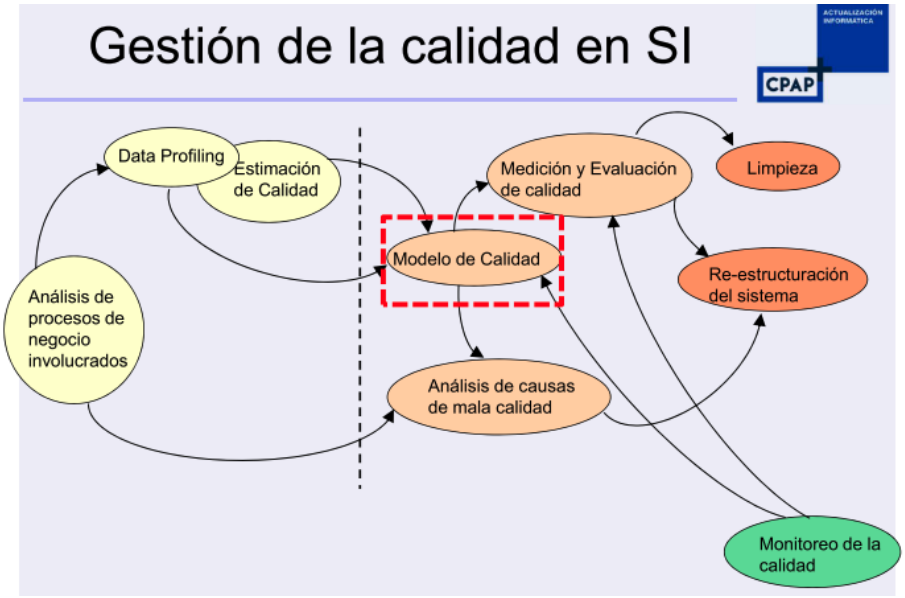


Figura 6: Gestión de calidad, extraído de [11].

y la granularidad que tendrá la medida. La granularidad de la medición es la unidad básica de información a la que asociamos las medidas. En bases de datos relacionales las granularidades típicas son: celda, tupla, conjunto de celdas, columna, tabla, grupo de tablas o base de datos.

Una **Métrica Genérica** representa una familia de métricas con características similares. Su objetivo es guiar, agilizar y uniformizar la creación de nuevas métricas, brindando definiciones generales que pueden ser refinadas por métricas más específicas. El refinamiento se realiza estableciendo valores para las Propiedades de Configuración que define la métrica genérica, así como especificando una semántica más concreta. Como resultado se obtienen las métricas específicas las cuales pueden refinar métricas genéricas para abordar requerimientos más específicos.

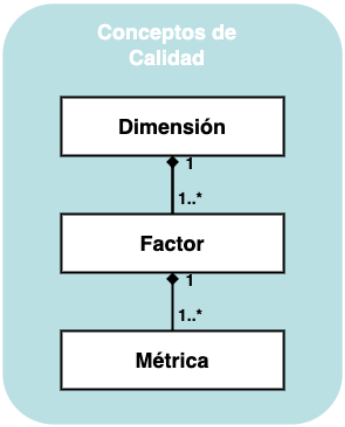


Figura 7: Jerarquía de conceptos de calidad, extraído de [11].

Una **Métrica especifica** define la aplicación de una métrica a un dato o conjunto de datos en particular. La misma se define por un nombre y un conjunto de datos.

El concepto de **agregaciones** también juega un papel clave en la calidad de datos, se basa en los resultados obtenidos de las métricas definidas. A partir de estos se pueden realizar agregaciones de dos tipos:

- Basándose en los factores donde se puede obtener una medida general de la calidad de los datos que fueron evaluados, de forma de establecer un peso a los factores, dependiendo de la relevancia que tienen en la dimensión.
- A partir de una medida a un determinado nivel de granularidad, obtener la medida a un nivel de granularidad menor.

A lo largo del tiempo han surgido muchas propuestas con distintos enfoques sobre las dimensiones de calidad de datos para los SI. La propuesta más reciente la presenta el estándar ISO/IEC 25012 [12], el cual surge en el 2008 pero tuvo su última revisión en 2019. Este estándar define 18 dimensiones de calidad de datos y las caracteriza considerando dos puntos de vista: propias del sistema y dependientes del sistema.

A continuación se describen algunas dimensiones de calidad de datos, junto a un ejemplo de factor para cada una de ellas [10]:

- **Exactitud**: Indica qué tan precisos, validos y libres de errores están los datos.
  - Factor Precisión: Captura el grado de detalle que posee un dato que lo hace útil para un determinado uso o que permite discriminarlo de otros datos que no son exactamente iguales.

    Un posible ejemplo es:
    En un sistema donde se almacenen distancias entre ciudades. Si se almacenan dichas distancias en kilómetros, el factor precisión no será el mismo que si se almacenan en metros.

- **Completitud**: Indica si el SI contiene toda la información de interés.
  - Factor Cobertura: Captura la proporción entre la cantidad de entidades existentes en una determinada colección de datos, y el total de entidades que deberían existir en dicha colección. La cobertura varía si se utiliza la Asunción de Mundo Cerrado, según la cual una colección de datos debería contener todas las entidades de un tipo, o si se utiliza la Asunción de Mundo Abierto, según la cual una colección de datos puede ser una representación parcial de las entidades del mundo real.

    Un posible ejemplo es:
    Cuando se necesita tomar un taxi al día de hoy en general se utiliza alguna aplicación que permita solicitarlo. Los viajes que se realizan mediante pedido de la aplicación quedan registrados y estos datos pueden ser utilizados como forma de estudio. por ejemplo para que rango de horas la demanda aumenta (Asunción de Mundo Cerrado). Pero existe la posibilidad de que para solicitar un taxi no uses ninguna aplicación y simplemente esperes a cruzar alguno en la calle que te encontras, por tanto esos viajes pueden ser registrados en la aplicacion o no (dependiendo de que el chofer se tome la molestia de ingresarlo. Si el chofer no ingresa

esos viajes nos encontramos con viajes que no se encuentran registrados (Asunción de Mundo Abierto).

- **Frescura**: Indica qué tan vigentes son los datos.
  - Factor Actualidad: Captura el tiempo de demora entre un cambio en el mundo real y la correspondiente actualización de los datos.

    Un posible ejemplo es:
    En un sistema donde se registran periódicamente el valor de distintas monedas, se encuentra que el valor del dolar por ejemplo no cambió en sucesivas consultas diarias, por lo que se determina que esos valores están desactualizados.

- **Consistencia**: Indica si los datos satisfacen las reglas definidas sobre los mismos, es decir, si no hay contradicciones entre los datos.
  - Factor Integridad Intra-Entidad: Captura la satisfacción de reglas entre atributos de una misma entidad.

    Un posible ejemplo es:
    En una tabla llamada Domicilios se guardan los datos de direcciones de puerta de Montevideo, se encuentra una tupla con atributos nombreCalle=«Andes» y numero-Puerta=«9052». El problema consiste en que no existe el numero de puerta 9052 en la calle Andes en Montevideo, por lo tanto hay un error en alguno de los dos atributos.

- **Unicidad**: Mide la unicidad de los datos, que los mismos no se encuentren representados mas de una vez.
  - Factor No-Contradicción: Captura el grado de repetición de una misma instancia de entidad del mundo real que es representada con datos contradictorios.

    Un posible ejemplo es:
    En una empresa de ventas se busca unificar los datos de los clientes que poseen. Para poder realizarlo, se utiliza el registro de las compras realizadas por la web y las compras realizadas en el local, las cuales se registran en un sistema local. En el registro de la unificación se generó una única tabla de clientes en base a su documento de identidad. Existen dudas de la confiabilidad de ese dato en los sistemas de origen, ya que se encontraron entidades duplicadas con contradicciones como las siguientes:
    1. t1:(documento:4568721-1, nombre:«Emanuel», apellido:«Martinez», direccion:«Maldonado 2060», telefono: 096873341, profesion:«estudiante»)
    2. t2:(documento:4568721-0, nombre«Manuel», apellido:«Martinez», direccion:«Maldonado 2060», telefono: 092246342, profesion:«albañil»)

Propuesta existente de modelo de calidad

Un ejemplo de modelo de calidad de datos existente es el 'Framework para la Gestión de la Calidad de Datos en Gobierno Digital" [10] (FGCDGD) de AGESIC. Este modelo de calidad fue construido para datos organizacionales. Esto significa que no

tiene en cuenta datos de procesos asociados ni del log de eventos de sus ejecuciones, únicamente tiene en cuenta los datos de la bases organizacionales.

El trabajo FGCDGD tiene como objetivo principal contribuir a la sistematización de la gestión de la calidad de datos en organizaciones vinculadas al gobierno digital en Uruguay, buscando mejorar la calidad de los datos que se generan y/o utilizan en el Estado Uruguayo. Mediante la aplicación del framework se busca obtener evidencia de mayor calidad para la toma de decisiones estratégicas, que permitan brindar mejores servicios y productos al ciudadano de forma más eficiente.

En la figura 8 se presenta la jerarquía que tiene el Modelo de Referencia para Calidad de Datos, el cual sigue una jerarquía Dimensión/Factor/Métrica como fue ilustrado anteriormente en la figura 7. Ésta jerarquía utilizada es de gran ayuda porque presenta una organización de los elementos de calidad del modelo, que facilita el entendimiento del mismo.

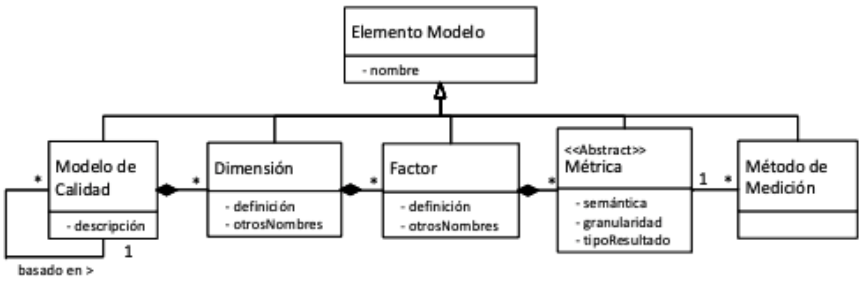


Figura 8: Jerarquía del Modelo de Calidad de FGCDGD, extraído de [10].

Se puede apreciar que cada Modelo de Calidad está compuesto por un conjunto de dimensiones, las cuales representan las cuestiones de calidad de datos relevantes. Cada dimensión está compuesta por un conjunto de factores y cada factor por un conjunto de métricas, y cada métrica por un conjunto de métodos de medición.

Este modelo define elementos de calidad muy interesantes, donde algunos de ellos serán tenidos en cuenta para nuestra propuesta que sera presentada en el capitulo 4 “Modelo de Calidad de Datos”.

Las dimensiones que define este trabajo se pueden observar en la tabla 1.

Tabla 1: Dimensiones del FGCDGD [10]

Dimension	**Descripción**
Exactitud	Proximidad entre un valor de datos v y un valor de datos v', considerado como la representación correcta del fenómeno del mundo real que v intenta representar.
Consistencia	Captura la violación de las reglas semánticas definidas sobre un conjunto de entidades de negocio o de sus atributos.
Completitud	Captura la medida en que los datos son de la amplitud, profundidad y alcance suficientes para una determinada tarea.
Unicidad	Captura el grado en el que un dato del mundo real es representado en forma única.
Frescura	Captura la rapidez con la que los cambios en el mundo real son reflejados en la actualización de los datos.

2.2.2. Calidad de log de eventos

En general cuando se trabaja en minería de procesos no se hace foco en la calidad de los datos del log de eventos, por lo que hay pocos trabajos que aborden este tema. En esta sección se presentan posibles errores en la calidad de log de eventos y pautas que tienen como fin destacar los problemas relacionados a la entrada en la minería de procesos.

Problemas en calidad de log de eventos

Luego de tener los logs de eventos se puede analizar la calidad de los mismos, para detectar problemas y poder corregirlos antes de usar los datos como input de la minería de procesos. Dentro de la calidad los posibles problemas se pueden agrupar en cuatro las categorías [7]:

- Datos perdidos: puede ocurrir que a la hora del armado del log se tenga un problema que ocasione que por ejemplo un dato que sea obligatorio el log de eventos no lo tenga entre sus atributos obtenidos.
- Datos incorrectos: Esto se debe a que los datos obtenidos a diferencia del tipo anterior si estan en el log, pero los datos no son los que verdaderamente se deberían haber registrado.
- Datos imprecisos: En este caso puede suceder cuando los datos no se expresan de la forma esperada, sino que se expresan con menos detalle del necesario, perdiendo validez como dato, al no poder ser interpretado con la granularidad esperada.
- Datos irrelevantes: estos datos son datos que sin ningún mecanismo de conversión o interpretación no sean útiles, pero si se les aplica pueden llegar a generar datos provechosos.

A partir de estas cuatro categorías de errores en el registro de logs de eventos se pueden obtener 27 clases diferentes de problemas de calidad dependiendo de la granularidad a la que nos refiramos en los datos.

Cada uno de los problemas de calidad de la figura 9, se identifica de la forma 'IX' donde X es un numero del 1 al 27. De igual forma se debe aclarar que existen combinaciones en la tabla que no son aplicables entre la categoría de error y cierta granularidad del log.

	Caso	Evento	Perteneciente	Atributo de Caso	Posición	Nombre de Actividad	Timestamp	Recurso	Atributo de Evento
Datos perdidos	I1	I2	I3	I4	I5	I6	I7	I8	I9
Datos incorrectos	I10	I11	I12	I13	I14	I15	I16	I17	I18
Datos imprecisos			I19	I20	I21	I22	I23	I24	I25
Datos irrelevantes	I26	I27							

Figura 9: Problemas principales de calidad del registro de eventos, extraído de [13].

Dentro de los posibles problemas en la calidad los mas frecuentes en la practica son los siguientes [13]:

- I2 se efectúa con granularidad de evento, puede ocurrir cuando en una secuencia de eventos se omite por parte del log el registro de eventos intermedios, puede generar que se infieran relaciones inexistentes, o con información parcial de la realidad.
- I16 se da a nivel del timestamp. El timestamp es muy importante para poder ordenar y/o filtrar los logs. En el primer problema se proporciona timestamp pero la marca de tiempo registrada de al menos alguno de los eventos en el registro no se corresponde con el tiempo real en el que ocurrieron los eventos.
- I23 también se da a nivel del timestamp. Como se menciono anteriormente el timestamp es vital tanto para el orden como para el filtrado de logs. En este caso el problema se da cuando las marcas de tiempo se proveen pero no son del nivel de abstracción adecuado.
- I22 que se da en el nombre de la actividad. Este problema corresponde al escenario en el que los nombres de las actividades son demasiado generales. Como resultado, dentro de un seguimiento puede haber varios eventos con el mismo nombre de actividad y no se pueden diferenciar e identificar cual es el objetivo de la actividad.
- I27 se produce en granularidad de evento, puede ocurrir que el registro de eventos no sea relevante en el estudio al cual es enfocado. En este caso conlleva que se deba filtrar eventos para evitar los que no son relevantes.

Pautas para evitar los problemas en la calidad de log de eventos

En el articulo se definen 12 pautas, las cuales tienen como objetivo señalar los problemas relacionados con la entrada de la minería de procesos [14]. De todas las pautas, consideramos interesante mencionar las siguientes:

- GL1 - Los nombres de referencia y atributo deben tener una semántica clara, es decir, deben tener el mismo significado para todas las personas involucradas en la creación y análisis de datos de eventos.
- GL4 - Los valores de los atributos deben ser lo más precisos posible. Si el valor no tiene la precisión deseada, esto debe indicarse explícitamente. Por ejemplo,

si para algunos eventos solo se conoce la fecha pero no la marca de tiempo exacta, entonces esto debe indicarse explícitamente.

- GL5 - Incertidumbre respecto a la ocurrencia del evento o su referencia. Por ejemplo, debido a errores de comunicación, algunos valores pueden ser menos fiables de lo habitual.
- GL7 - Si es posible, también almacene información transaccional sobre el evento. Tener eventos de inicio y finalización permite calcular la duración de las actividades. Se recomienda almacenar referencias de actividad para poder relacionar eventos pertenecientes a la misma instancia de actividad.
- GL11 - No elimine eventos y asegure la procedencia, debido a que la reproducibilidad es clave para la minería de procesos.
- GL12 - Garantice la privacidad sin perder correlaciones significativas. Los datos sensibles o privados deben eliminarse lo antes posible (es decir, antes del análisis). Sin embargo, si es posible, se debe evitar eliminar las correlaciones. El hash puede ser una herramienta poderosa en el intercambio entre privacidad y análisis.

Propuesta existente de modelo de calidad en log de eventos

Una ejemplo de modelo de calidad en log de eventos es la Tesis de Maestría "Evaluating Quality of Event Data within Event Logs: An Extensible Framework" [9].

Esta Tesis de Maestría, persigue el objetivo de cuantificar los aspectos de calidad de los datos de los eventos. Esto significa que debería ser posible dar una indicación de los aspectos de calidad, qué tan buenos son para un registro de eventos específico, y por qué este es el caso.

Sin embargo en este trabajo el conocimiento del dominio no tiene importancia (a diferencia del anterior trabajo presentado que el dominio de los datos con los que se trabaja es conocido) para los aspectos de calidad, lo que hace que estas dimensiones de calidad sean relevantes para cualquier conjunto de datos de eventos. Por tal motivo la interpretación del usuario es importante. Existen casos donde lo que generalmente se acepta como un "mal" escenario, no siempre es un mal escenario para situaciones específicas. Significa que su interpretación de la situación también debe tenerse en cuenta.

En este trabajo se define un Modelo de Calidad de Datos enfocado a datos en log de eventos como se muestra en la figura 10, el cual también define dimensiones de calidad muy interesantes. El problema principal en este Modelo de CD es que no sigue ninguna jerarquía en su estructura, en cambio define únicamente dimensiones y métodos de medición de las mismas, donde varias de estas dimensiones se podrían agrupar ya que apuntan a aspectos de calidad en común.

Figura 10: Jerarquía del Modelo de Calidad de "Evaluating Quality of Event Data within Event Logs: An Extensible Framework", extraído de [9].

Las dimensiones que define dicha Tesis de Maestría se pueden observar en la tabla 2.

Tabla 2: Dimensiones de la Tesis de Maestría [9].

Dimension	**Descripción**
Completeness	Qué tan completos son los datos, incluyendo la mayor posible información.
Uniqueness/Duplicates	El número de atributos cuyo valor ocurre solo una vez.
Timeliness	Qué tan actual son los datos, y en que marco de tiempo se espera que estén.
Validity	Validez de los datos conforme a la sintaxis de su definición.
Accuracy	Dado un dato obtenido, qué tan cerca esta del valor real del mismo.
Consistency	Cuanto más consistente son los datos, más ausencia de diferencias se podrá encontrar.
Believability/Credibility	La confianza que tienen los usarios sobre los datos.
Relevancy	La importancia de los datos.
Security/Confidentiality	Qué tan protegidos están los datos.
Complexity	El grado de extensión de la estructura del proceso. Cuanto más extensa es la estructura del proceso, más compleja es.
Coherence	La interconexión lógica entre los datos.
Representation/Format	La medida en que los datos se representan de forma compacta y se presentan en el mismo formato.

3. Análisis del Problema

El desafío que se afronta en este trabajo surge de un problema base que ha sido afrontado por un proyecto de investigación de mayor dimensión. Este proyecto se encuentra en el marco del proyecto de investigación "Minería de procesos y datos para la mejora de procesos en las organizaciones", financiado por CSIC, UdelaR y con participación de AGESIC.

En este capítulo se presenta el contexto de este trabajo. Primero se describe el problema general, en donde se hace una introducción al mismo y luego se describe la solución que ha sido propuesta por parte del proyecto CSIC. A continuación se describe el BP "Student Mobility", el cual es un proceso real que cuenta con una implementación y generación de log de eventos extendido con datos integrados, que se utilizará como caso de estudio. Por ultimo se presenta la extensión del formato de log de eventos XES definida para representar los datos integrados en un log de eventos.

3.1. Descripción del problema

Los sistemas organizacionales suelen estar implementados en diversas tecnologías y plataformas, los cuales se comunican entre sí utilizando bases de datos centralizadas, exponiendo servicios web que pueden ser invocados por otros sistemas, entre otros.

Cuando un BP interactúa con diversos sistemas implementados en otras tecnologías, sucede que los datos generados por el proceso estén distribuidos en estos sistemas y se dificulta el análisis de los mismos.

En la figura 11 se puede observar un BP integrado con datos organizacionales el cual está modelado en un BPMS, y que a su vez interactúa con varios componentes y Sistemas de Información externos. En este tipo de BP se debe tener en cuenta al menos dos orígenes de datos, los cuales son la base de datos del motor que ejecuta el BP y la base de datos organizacionales. [2]

El motor cuenta con su propia base de datos en la cual almacena elementos asociados a las trazas de ejecución, pero además algunos de estos elementos de información son almacenados en la base de datos organizacional. Estas fuentes de datos no son automáticamente asociadas, por lo que surge el desafío de asociar los registros de la base de datos del motor con sus respectivos en la base de datos organizacional para poder contar con datos integrados cuando nos encontramos en la etapa de Process Mining.

Para lidiar con el problema de la integración de los datos del proceso con los datos organizacionales se ha propuesto una solución, la cual se puede ver en la figura 12. Esta solución abarca tres perspectivas: procesamiento de los datos, modelo integrado y data warehouse. [2]

El procesamiento de los datos consiste en cargar los datos de la base del proceso y de la base organizacional a un metamodelo de procesos y datos definido. Luego

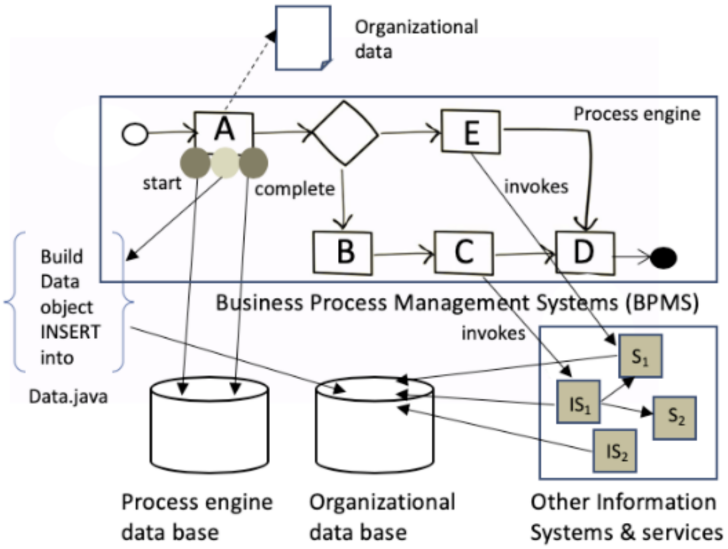


Figura 11: Integración de un proceso con datos organizacionales, extraido de [2].

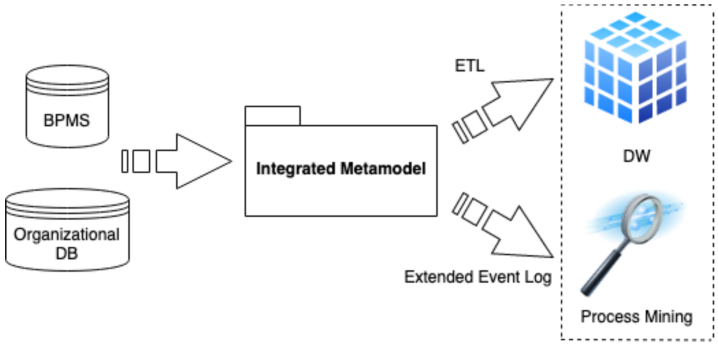


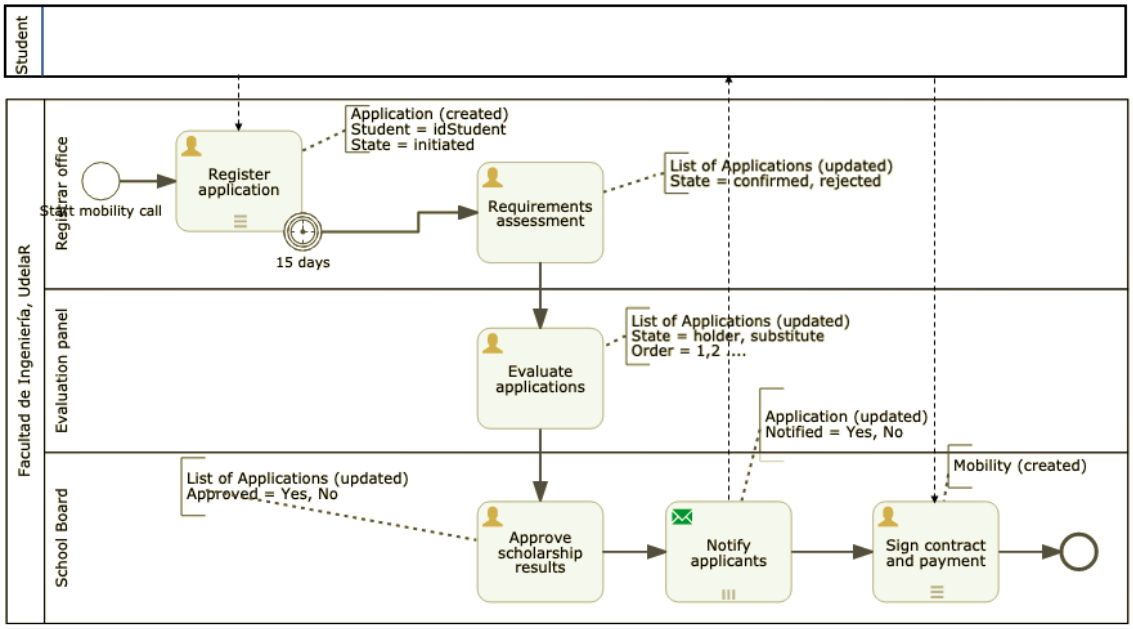
Figura 12: Modelo integrador, extraído de [2].

mediante un algoritmo de matcheo se integran los datos y se genera el log de eventos extendido en formato XES. Todo este procesamiento se realiza de forma automatizada. [15]

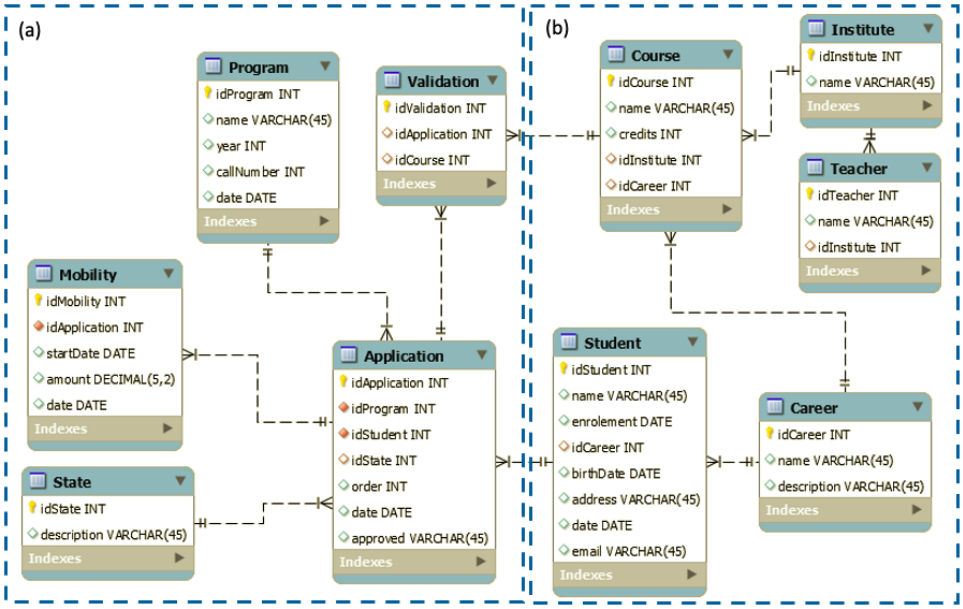
3.2. Modelo del BP y los datos organizacionales

Para este trabajo se ha tomado un BP real el cual será utilizado para realizar un caso de estudio en el capitulo 6. Este BP es llamado "Movilidad estudiantil" (Students Mobility) [2] y describe la evaluación de programas de intercambio para estudiantes de nuestra universidad. El BP se encarga de atender las aplicaciones de estudiantes para luego, después de realizar algunos controles y evaluaciones, asignar becas para los mismos. En la figura 13a se muestra el BP modelado con BPMN 2.0, y en la figura 13b se muestra el modelo de datos organizacional soportado por el BP.

La figura 13a describe el flujo de ejecución del BP *Students Mobility*, el cual comienza cuando la Oficina de Registro recibe las solicitudes para abrir los programas de intercambio de estudiantes, y los registra en la tarea "Register application". Después de 15 días cierra el período de solicitudes, y en la tarea "Requirement assessment"



(a) Proceso de negocio Students Mobility extraído de [2]



(b) Modelo de datos soportado por el proceso Students Mobility extraído de [2]

Figura 13: Pruba de concepto de Students Mobility

se chequean las solicitudes para ver si cumplen con los requerimientos del programa de intercambio. Luego, las solicitudes que cumplen con los requerimientos del programa son enviados una evaluación efectuada por la tarea “Evaluate applications”, donde se ordena la lista de solicitantes y se seleccionan tanto los titulares como los suplentes. Finalemente la tarea “Approve scholarship results” toma la lista ordenada de solicitantes y aprueba los resultados, luego se notifica a los solicitantes mediante la tarea “Notify applicants” y en la tarea “Sign contract and payment” es donde los titulares firman el contrato y reciben el monto de dinero correspondiente por la beca.

Por otro lado, en el BP se puede observar que cada tarea tiene un comentario aso-

ciado a los datos organizacionales que son impactados por la misma. En la tarea "Register application" se accede a la tabla *Application* (del modelo de datos organizacionales descrito en la figura 13b) para poder insertar una nueva solicitud de un estudiante identificado por *idStudent* y el estado *initiated*.

En la tarea "Requirements assessment" se recibe la lista de solicitudes de la base de datos de la organización, y se le actualiza a cada elemento de la lista con el resultado confirmado o rechazado. En el resto de las tareas se manipula la lista de solicitudes con sus correspondientes actualizaciones y cada solicitud cuando se notifica al solicitante los resultados. En la última tarea se crea un nuevo registro en la tabla de Movilidad donde se incluye la fecha de inicio de la movilidad y el monto que se le otorga.

3.3. Log de eventos extendido

Para poder obtener un log de eventos que represente la integración de los datos como se define en el metamodelo, en el proyecto de investigación CSIC se ha definido una extensión para XES, llamada "Organizational Data" (orgdata).

Esta extensión asocia los datos del BP con los datos organizacionales tal cual como se define en el metamodelo. Para esto se agregan dos listas de atributos a cada evento del log: *varlist* y *entlist*. La lista *varlist* contiene las variables que fueron definidas durante la ejecución del evento por parte del BPMS, lo cual se corresponde al cuadrante *instancias de procesos* del metamodelo. Mientras que la lista *entlist* contiene las entidades y atributos que fueron manipulados en el evento, lo cual se corresponde al cuadrante *instancias de datos*. Cada entidad provee además de su nombre, una lista de atributos.

Cada attributo contiene su nombre (*attname*), tipo (*valueType*) y valor de ese momento de la ejecución (*attValue*). Además si un atributo esta asociado a una variable del BP, se agrega una referencia hacia dicha variable (*refVariable*). En la Figura 14 se muestra un extracto de un log extendido el cual utiliza la extension *orgdata*. Este ejemplo se extrajo del BP "Students Mobility" descrito anteriormente.

```
<?xml version="1.0" encoding="UTF-8"?>
<log xmlns="http://your_namespace"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://your_namespace"
        xes.version="1.0" xes.features="nested-attributes" >
        <extension name="Organizational" prefix="org" uri="http://www.xes-standard.org/org.xesext"/>
        <extension name="Time" prefix="time" uri="http://www.xes-standard.org/time.xesext"/>
        <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org/concept.xesext"/>
        <extension name="OrganizationalData" prefix="orgdata" uri="http://www.xes-standard.org/orgdata.xesext"/>
        <string key="concept:name" value="Mobility"/>
        <trace>
                <string key="id" value="773783"/>
        <string key="concept:name" value="773783"/>
        <event>
                <string key="concept:name" value="Register application"/>
                <string key="lifecycle:transition" value="Start"/>
                <date key="time:timestamp" value="2020-11-16T09:36:33.784-0300"/>
                <string key="org:role" value="Jorge"/>
                <string key="org:resource" value="5394935"/>
                <string key="orgdata:elemType" value="UserTask"/>
                <list key="orgdata:varlist">
                        <variables>
                                <string key="concept:varname" value="studentid">
                                        <string key="orgdata:varValue" value="89964588"/>
                                        <string key="orgdata:valueType" value="string"/>
                                        <date key="time:timestamp" value="2020-11-16T09:36:50.554-0300"/>
                                </string>
                        </variables>
                </list>
                <list key="orgdata:entlist">
                    <entities>
                        <string key="concept:entname" value="application">
                                <list key="orgdata:attlist">
                                     <attributes>
                                        <string key="concept:attname" value="idapplication">
                                                <string key="orgdata:attValue" value="1592"/>
                                                <string key="orgdata:valueType" value="int4"/>
                                                <date key="time:timestamp" value="2020-11-16T09:36:50.863-0300"/>
                                        </string>
                                        <string key="concept:attname" value="idstate">
                                                <string key="orgdata:attValue" value="1"/>
                                                <string key="orgdata:valueType" value="int4"/>
                                                <date key="time:timestamp" value="2020-11-16T09:36:50.863-0300"/>
                                        </string>
                                        <string key="concept:attname" value="idprogram">
                                                <string key="orgdata:attValue" value="263"/>
                                                <string key="orgdata:valueType" value="int4"/>
                                                <date key="time:timestamp" value="2020-11-16T09:36:50.863-0300"/>
                                        </string>
                                        <string key="concept:attname" value="idstudent">
                                                <string key="orgdata:attValue" value="89964588"/>
                                                <string key="orgdata:valueType" value="int4"/>
                                                <date key="time:timestamp" value="2020-11-16T09:36:50.863-0300"/>
                                                <string key="orgdata:refVariable" value="studentid"/>
                                        </string>
                                        <string key="concept:attname" value="notified">
                                                <string key="orgdata:attValue" value="NULL"/>
                                                <string key="orgdata:valueType" value="int4"/>
                                                <date key="time:timestamp" value="2020-11-16T09:36:50.863-0300"/>
                                        </string>
                                     </attributes>
                                </list>
                        </string>
                        .....
                    </entities>
                </list>
        </event>
        ....
    </trace>
    .....
</log>
```

Figura 14: Log de eventos extendido en formato XES, extraído de [2]

En este ejemplo, la actividad que se muestra corresponde a la tarea "Register application". La variable manejada por la ejecución del BP es *studentid*, la cual se corresponde al attribute de la base organizacional *idstudent* de la entidad *Application*. Se puede notar que el valor de los dos elementos es el mismo. Esta actividad se corresponde con el ejemplo mencionado en la sección anterior en el cual la tarea "Register application" inserta un nuevo registro en la tabla *Application* de la base de datos organizacional, para cada estudiante, en el programa de intercambio. Por simplicidad, en este ejemplo se omiten otras variables, entidades y atributos relacionados a la tarea "Register application".

4. Modelo de calidad BPODQM

En este capítulo se presenta el foco principal de este trabajo: el modelo de CD para BP y datos organizacionales BPODQM. Para la construcción de este modelo se tomó como referencia las dos propuestas mencionadas en la secciones 2.2.1 y 2.2.2, los cuales son "Framework para la Gestión de la Calidad de Datos en Gobierno Digital" [10] (FGCDGD) de AGESIC y la Tesis de Maestría "Evaluating Quality of Event Data within Event Logs: An Extensible Framework"[9]. A estos dos trabajos se le sumaron varias entrevistas y seminarios con el grupo de investigadores del proyecto, en donde se fue construyendo el modelo propuesto.

4.1. Definición del modelo de calidad

En esta sección se presenta el modelo BPODQM. Por un lado, en la primer sección se realiza una descripción general del modelo, en donde se define el formato de los datos obtenidos junto con las granularidades correspondientes, y luego se presenta la estructura del modelo. Por otro lado, en la segunda sección se describe de manera detallada las dimensiones, factores y métricas que lo componen.

4.1.1. Descripción

Los datos de entrada al modelo BPODQM provienen de un log de eventos extendido y cumplen con el formato estándar XES [8]. El log de eventos extendido que se muestra en la Figura 14 es un ejemplo de datos de entrada para este modelo.

Para definir un modelo de CD, es necesario definir las diferentes granularidades para identificar los grupos de datos sobre los que se aplicarán las métricas. Estas granularidades son las siguientes:

- **valor de atributo**. Refiere a un valor particular de un atributo. Por ejemplo, en la figura 14, el valor "Register application" asociado al atributo nombre de la tarea.
- **atributo**. Refiere al conjunto de valores correspondientes a la misma key. Por ejemplo, en la figura 14, todos los valores que aparecen para la key "concept:name".
- **evento**. Corresponde a todos los datos incluidos en un evento. Por ejemplo, en la figura 14, todos los datos incluidos en el evento llamado "Register application".
- **case** Corresponde a todos los datos incluidos en una traza. Por ejemplo, en la figura 14, todos los datos incluidos en la traza identificada como "73783".
- **log**. Corresponde a un log entero.

En la construcción del modelo de calidad BPODQM hemos utilizado la jerarquía de conceptos de calidad Dimensión-Factor-Métrica, la cual fue descrita en el capítulo 2. Este modelo se compone de 7 dimensiones de calidad, las cuales se componen de factores y métricas. A continuación se muestra la estructura del modelo BPODQM.

Para una mejor visualización, el modelo se presenta en 2 imágenes, las figuras 15 y 16. La métrica que tiene un tick en verde, significa que la misma se ha implementa-

do, cuando en cambio tiene una cruz en rojo significa que no se llegó a implementar en el plug-in que será presentado en el capitulo 5.

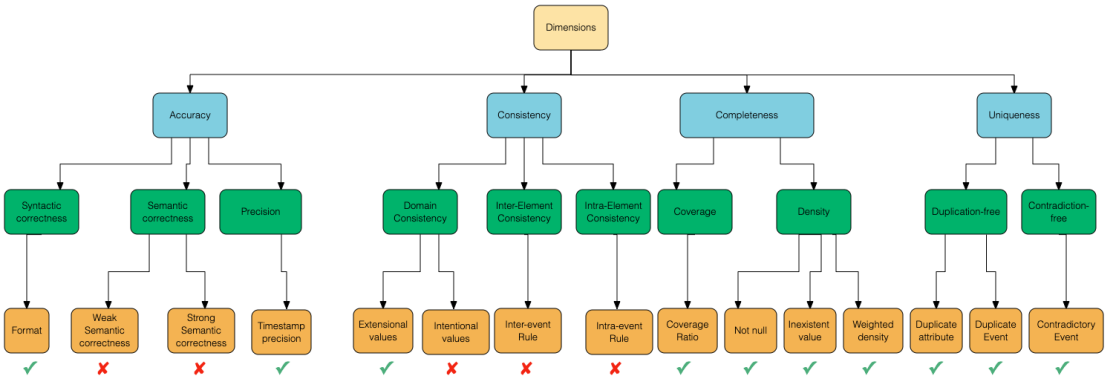


Figura 15: Modelo de Calidad de Datos 1.º parte.

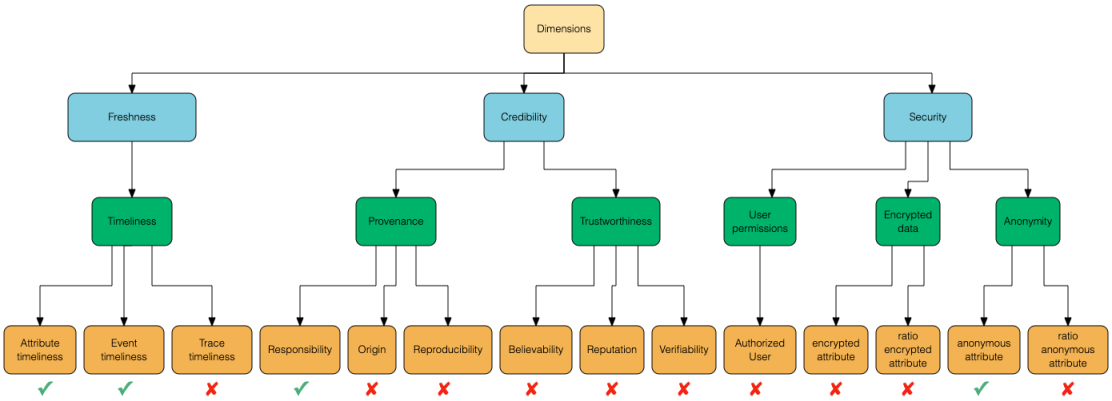


Figura 16: Modelo de Calidad de Datos 2.º parte.

En la Tabla 3 se describe las granularidades que le corresponden a cada una de las métricas del modelo. Para cada factor se indica en base a qué fue inspirada su definición (o en su defecto definida en este trabajo) mediante un punto de color. Las referencias significan lo siguiente:

●: Proviene del Framework para la Gestión de la Calidad de Datos en Gobierno Digital (en adelante, FGCDGD) [10].
●: Proviene de la Tesis de Maestría de calidad de datos en log de eventos (en adelante, Tesis de Maestría) [9].
●: Agregado en este trabajo.

Tabla 3: Dimensiones, factores, metricas y granularidad

Dimension	Factor	Metric	Granularity
Accuracy	Syntactic Accuracy	Format	attribute value
	Semantic Accuracy	Weak Semantic Accuracy	event
		Strong Semantic Accuracy	event
	Precision	Timestamp precision	attribute value
Consistency	Domain Consistency	Extensional Values	attribute value
		Intensional Values	attribute value
	Inter-element Consistency	Inter-event Rule	activity
	Intra-element Consistency	Intra-event Rule	event
Completenes	Coverage	Coverage Ratio	trace
	Density	Not Null	attribute value
		Inexistent Value	event
		Weighted Density	event
Uniqueness	Duplication-free	Duplicate Attribute	attribute value
		Duplicate Event	event
	Contradiction-free	Contradictory Event	event
Freshness	Timeliness	Attribute Timeliness	attribute value
		Event Timeliness	event
		Trace Timeliness	trace
Credibility	Provenance	Responsibility	log
		Origin	event
		Reproducibility	log
	Trustworthiness	Believability	attribute value
		Reputation	attribute value
		Verifiability	attribute value
Security	User Permissions	Authorized User	event
	Encrypted Data	Encrypted Attribute	attribute value
		Ratio Encrypted Att	event
	Anonymity	Anonymous Attribute	attribute value
		Ratio Anonymous Att	event

4.1.2. Descripción detallada

Se presenta y describe el modelo propuesto, con sus dimensiones, factores y métricas que lo componen. Para cada dimensión y factor se brinda una explicación del motivo por el que fue incluido en el modelo.

Como comentario general se definieron métricas específicas exclusivamente a las métricas genéricas que se abordan en la implementación. La definición de las mismas se inspiró en [16] en conjunto con los trabajos mencionados anteriormente.

Accuracy

Es muy importante que los datos con los que trabajemos se correspondan con el valor de los datos durante la ejecución del BP. En otras palabras los datos registrados en el log deben coincidir con su verdadero valor en el mundo real. Para asegurarnos de esto se define la dimensión “Accuracy”, la cual mide la proximidad entre un valor

de datos v y un valor de datos v', considerado como la representación correcta del fenómeno del mundo real que v intenta representar. Para definir esta dimensión nos hemos basado en la dimensión "Accuracy" de la Tesis de Maestría y la dimensión "Exactitud" del FGCDGD.

Por un lado, nos ha parecido que la dimensión definida en la Tesis de Maestría era correcta pero insuficiente debido a como está estructurado su modelo de calidad, donde existe una jerarquía y se trata como un único aspecto a medir dentro de la exactitud. Por otro lado, la dimensión que define el FGCDGD agrega factores que nos resultaron muy interesantes y que podían ser aplicados a datos provenientes de logs de eventos. Estos factores son "Correctitud Sintáctica", "Correctitud Semántica" y "Precisión", de los cuales "Correctitud Semántica" coincide con la definición de "Accuracy" de la Tesis de Maestría.

A partir de estos elementos se han definido tres factores para la dimensión "Accuracy": "Syntactic accuracy", "Semantic accuracy" y "Precision". Los factores "Syntactic accuracy" y "Semantic accuracy" agregan la necesidad de tener información previa, ya que para el primero se necesita saber cual es el dominio del atributo a medir, y para el segundo se necesita saber cual es el valor verdadero del mismo.

En la tabla 4 se describe la dimensión "Accuracy", y a continuación se describen sus factores asociados: "Syntactic accuracy", "Semantic accuracy", y "Precision".

Tabla 4: Dimension Accuracy

Dimensión Accuracy	
Definición	Proximidad entre un valor de datos v y un valor de datos v', considerado como la representación correcta del fenómeno del mundo real que v intenta representar.
Otros nombres	Exactitud, Correctitud.
Factores	Syntactic accuracy Semantic accuracy Precision

En la tabla 5 se define el factor Syntactic accuracy y sus métricas genéricas y específicas.

En la tabla 6 se define el factor Semantic accuracy y sus métricas genéricas y específicas.

Tabla 5: Factor Syntactic accuracy

Factor Syntactic accuracy	
Definición	Proximidad entre el valor v de un atributo y los elementos del dominio de definición de dicho atributo.
Otros nombres	Correctitud Sintáctica.
Métricas genéricas	**Format** – **Granularidad:** valor de atributo – **Definicion:** Indica si el valor de un atributo cumple con el formato definido para ese atributo según algún estándar.
Métricas específicas	**timestampFormat(date): bool** Retorna verdadero si el timestamp de un objeto del log extendido esta en el formato yyyy-MM-dd'T'HH:mm:ss. **attributeFormat(attribute, format): bool** Retorna verdadero si el atributo 'attribute' esta en el formato 'format'.

Tabla 6: Factor Semantic accuracy

Factor Semantic accuracy	
Definición	Proximidad entre el valor v de un atributo y su verdadero valor v'.
Otros nombres	Correctitud Semántica.
Métricas genéricas	**Weak Semantic accuracy:** – **Granularidad:** evento – **Definicion:** Evalúa si una instancia de un atributo, que no forma parte de la identificación de la entidad a la que pertenece, existe dentro de un referencial de valores posibles de ese atributo. **Strong Semantic accuracy:** – **Granularidad:** evento – **Definicion:** Evalúa si una instancia de un atributo, que forma parte de la identificación de la entidad a la que pertenece, existe dentro de un referencial de valores posibles de ese atributo.
Métricas específicas	**weakSemanticAccuracy(event, attributeSet): attributeSet** Dado un evento y el conjunto de atributos que lo identifica, retorna conjunto de atributos no pertenecientes a la identificación cuyos valores no pertenecen al dominio de su tipo definido. **strongSemanticAccuracy(event, attributeSet): attributeSet** Dado un evento y el conjunto de atributos que lo identifica, retorna conjunto de atributos pertenecientes a la identificación cuyos valores no pertenecen al dominio de su tipo definido.

En la tabla 7 se define el factor Precision y sus métricas genéricas y específicas.

Tabla 7: Factor Precision

Factor Precision	
Definición	Captura el grado de detalle que posee un dato que lo hace útil para un determinado uso o que permite discriminarlo de otros datos que no son exactamente iguales.
Otros nombres	Precisión.
Métricas genéricas	**Timestamp precision:** – **Granularidad:** valor de atributo – **Definicion:** Captura el nivel de detalle del valor de un atributo timestamp.
Métricas específicas	**timestampPrecision(attribute):bool** Retorna verdadero si el timestamp del atributo 'attribute' tiene el nivel de detalle yyyy-MM-dd'T'HH:mm:ss.

Consistency

La necesidad de que los datos sean consistentes es grande a la hora de trabajar con los mismos, por esto nos ha parecido que es importante contar con una dimensión que mida esta consistencia. Esta consistencia puede estar relacionada tanto al dominio de un atributo como a dependencias entre distintos atributos. Fue por esto que se ha definido la dimensión "Consistency", la cual captura la violación de las reglas semánticas definidas sobre un conjunto de eventos o de sus atributos. Para definir esta dimensión nos hemos basado en la dimensión "Consistency" de la Tesis de Maestría y la dimensión "Consistencia" del FGCDGD.

El FGCDGD define tres factores para esta dimensión: "Integridad de dominio", "Integridad Inter-entidad", e "Integridad Intra-entidad". Mientras que la Tesis de Maestría define la dimensión "Consistency" de tal forma que se corresponde con el factor "Integridad de dominio".

A partir de esto se han definido los tres factores que componen esta dimensión: "Inter-Element Consistency", "Intra-Element Consistency" y "Domain consistency". Para medir estos tres factores es necesario tener información como input por parte del usuario, de otra manera serían imposibles de medir. Por un lado para el factor "Domain consistency" es necesario saber el dominio del atributo a medir, por otro lado para los factores "Inter-Element Consistency" e "Intra-Element Consistency" se necesitan reglas de consistencia entre atributos de distintos eventos.

En la tabla 8 se describe la dimensión "Consistency" y a continuación se describen sus factores asociados: Domain Consistency, Inter-Element Consistency e Intra-Element Consistency.

Tabla 8: Dimensión Consistency

Dimensión Consistency	
Definición	Captura la violación de las reglas semánticas definidas sobre un conjunto de eventos o de sus atributos.
Otros nombres	Consistencia, Cohesion, Coherencia.
Factores	Domain Consistency Inter-Element Consistency Intra-Element Consistency

En la tabla 9 se define el factor Domain Consistency y sus métricas genéricas y específicas.

Tabla 9: Factor Domain Consistency

Factor Domain Consistency	
Definición	Captura la satisfacción de reglas sobre los valores posibles que puede tomar un atributo.
Otros nombres	Integridad de dominio.
Métricas genéricas	**Extensional values:** – **Granularidad:** valor de atributo – **Definicion:** Indica si el valor de un atributo se encuentra dentro de un dominio definido por extensión. **Intentional values:** – **Granularidad:** valor de atributo – **Definicion:** Indica si el valor de un atributo se encuentra dentro de un dominio definido por comprensión, el cual puede estar dado por una propiedad que cumplen los elementos de ese dominio o por el tipo de dato conocido de ese dominio.
Métricas específicas	**extensionalValueAttribute(attribute, dom): bool** Retorna verdadero si el tipo de un atributo se encuentra dentro del dominio definido por el parámetro dom, el cual es una lista de valores. **intentionalValueAttribute(attribute, dom): bool** Retorna verdadero si el tipo de un atributo se encuentra dentro del dominio definido por el parámetro dom, el cual es una regla de negocio.

En la tabla 10 se define el factor Inter-Element Consistency y sus métricas genéricas.

Tabla 10: Factor Inter-Element Consistency

Factor Inter-Element Consistency	
Definición	Captura la satisfacción de reglas entre atributos de diferentes eventos.
Otros nombres	Integridad Inter-entidad, Integridad Inter-relacion.
Métricas genéricas	**Consistency Rule Inter-Event:** – **Granularidad:** traza – **Definicion:** Regla de inclusión o expresión condicional entre atributos de diferentes eventos. Ejemplo, un atributo de un evento A, determine el valor de un atributo de evento B.

En la tabla 11 se define el factor Intra-Element Consistency y sus métricas genéricas.

Tabla 11: Factor Intra-Element Consistency

Factor Intra-Element Consistency	
Definición	Captura la satisfacción de reglas entre atributos de un mismo evento.
Otros nombres	Integridad Intra-entidad.
Métricas genéricas	**Consistency Rule Intra-Event:** – **Granularidad:** evento – **Definicion:** Reglas de dependencia de atributos y unicidad de atributos.

Completeness

Otro aspecto de calidad importante es la completitud de los datos. Suelen haber ciertos sistemas de información en los cuales hay datos importantes que no se encuentran en el log, como también se suelen almacenar datos nulos. En determinados dominios se entiende que una densidad grande de valores nulos o inexistentes afectan mucho a la calidad de los mismos, y dificulta la minería.

Para poder medir esto se definió la dimensión “Completeness”, que captura la medida en que los datos son de la amplitud, profundidad y alcance suficientes para una determinada tarea. Para definir esta dimensión nos hemos basado en la dimensión “Completeness” de la Tesis de Maestría y la dimensión “Completitud” del FGCDGD. Este último agrega dos factores los cuales se adaptan bien a nuestro modelo, los mismos son “Cobertura” y “Densidad”.

Dado esto, se han definido los factores para nuestra dimensión: “Coverage” y “Density”. El primero apunta a detectar trazas que sobresalen del resto en cuanto a su baja cantidad de eventos, mientras que el segundo se dedica a cuantificar datos incompletos. Para el factor “Coverage” se necesita como input por parte del usuario

un umbral que indique la mínima cantidad de eventos en una traza.

En la tabla 12 se describe la dimensión "Completeness" y a continuación se describen sus factores asociados: Coverage, Density.

Tabla 12: Dimensión Completeness

Dimensión Completeness	
Definición	Captura la medida en que los datos son de la amplitud, profundidad y alcance suficientes para una determinada tarea.
Otros nombres	Completitud
Factores	Coverage Density

En la tabla 13 se define el factor Coverage y sus métricas genéricas y específicas.

Tabla 13: Factor Coverage

Factor Coverage	
Definición	Captura la proporción entre la cantidad de eventos existentes en una determinada traza, y el total de eventos que deberían existir en dicha traza.
Otros nombres	Cobertura
Métricas genéricas	**Coverage Ratio:** – **Granularidad:** traza – **Definicion:** Proporción entre la cantidad de eventos en una traza y el número total de eventos en una traza de referencia del log extendido.
Métricas específicas	**coverageWarn(threshold): traceSet** Dado un porcentaje, retorna las trazas que poseen menor porcentaje de eventos que el mismo.

En la tabla 14 se define el factor Density y sus métricas genéricas y específicas.

Tabla 14: Factor Density

Factor Density	
Definición	Captura la proporción entre la cantidad de instancias de atributo con valores no nulos y el total de instancias de dicho atributo.
Otros nombres	Densidad.
Métricas genéricas	**Not null:** – **Granularidad:** valor de atributo – **Definicion:** Indica si una instancia de atributo tiene un valor no nulo. **Inexistent value:** – **Granularidad:** evento – **Definicion:** Indica si en un elemento se detecta la inexistencia de al menos uno de sus atributos. **Weighted density:** – **Granularidad:** evento – **Definicion:** Aplica un cálculo sobre un conjunto de atributos, multiplicando el resultado de cada atributo por un coeficiente entre 0 y 1 cuya suma sea igual a 1.
Métricas específicas	**notNullAttribute(attribute):bool** Indica si el atributo 'attribute' es nulo. **inexistentAttribute(attribute):bool** Indica si el atributo 'attribute' no existe en un evento. **weightedDensityNotNullElement(attributeSet):float** Se aplica un cálculo sobre el conjunto de atributos 'attributeSet' de los elementos del log extendido, evaluando para cada uno si es nulo o no. Se multiplica el porcentaje de nulidad de cada atributo por un coeficiente entre 0 y 1 cuya suma sea igual a 1. A mayor gravedad de tener un nulo en un atributo, más cercano a 1 será el coeficiente ingresado por el usuario para ese atributo. **weightedDensityNotFoundElement(attributeSet):float** Se aplica un cálculo sobre el conjunto de atributos 'attributeSet' de los elementos del log extendido, evaluando para cada uno si se encuentra o no. Se multiplica el porcentaje de nulidad de cada atributo por un coeficiente entre 0 y 1 cuya suma sea igual a 1. A mayor gravedad de tener un nulo en un atributo, más cercano a 1 será el coeficiente ingresado por el usuario para ese atributo.

Uniqueness

La unicidad de los datos es otro aspecto importante en la calidad, se debe evitar que hayan datos duplicados o contradictorios si se quiere realizar un posterior análisis. Para esto se ha definido la dimensión "Uniqueness", la cual captura el grado en el que un dato del mundo real es representado en forma única. Para esta definición nos hemos basado en la dimension "Uniqueness / Duplicates" de la Tesis de Maestría y en la dimensión "Unicidad" del FGCDGD.

Se han definido dos factores para esta dimensión: "Duplication-free" y "Contradiction-free". El primero de ellos captura datos duplicados, como por ejemplo atributos duplicados dentro de un evento o eventos duplicados. Mientras que el segundo apunta a capturar eventos contradictorios, por ejemplo dos eventos que tienen el mismo identificador pero que difieren en el valor de algún atributo.

En la tabla 15 se describe la dimensión "Uniqueness" y a continuación se describen sus factores asociados: Duplication-Free y Contradiction-Free.

Tabla 15: Dimensión Uniqueness

Dimensión Uniqueness	
Definición	Captura el grado en el que un dato del mundo real es representado en forma única.
Otros nombres	Unicidad.
Factores	Duplication-Free Contradiction-Free

En la tabla 16 se define el factor Duplication-Free y sus métricas genéricas y específicas.

En la tabla 17 se define el factor Contradiction-Free y sus métricas genéricas y específicas.

Tabla 16: Factor Duplication-Free

Factor Duplication-Free	
Definición	Captura el grado de duplicación de un mismo dato.
Otros nombres	No duplicación.
Métricas genéricas	**Duplicate attribute:** – **Granularidad:** valor de atributo – **Definicion:** Indica si una instancia de atributo tiene el mismo valor que otra instancia del mismo atributo. **Duplicate Event:** – **Granularidad:** event – **Definicion:** Indica si existe, para una instancia de evento, al menos otra instancia más que representa el mismo objeto del mundo real, con los mismos datos o algún dato faltante.
Métricas específicas	**duplicateAttribute(event): attributeSet** Dado un evento, retorna el conjunto de atributos con más de una instancia de dicho atributo con el mismo valor. **duplicateEvent(event, trace): bool** Dado una traza y un evento de la misma, retorna si en dicha traza existe mas de una instancia de dicho evento con el mismo identificador.

Tabla 17: Factor Contradiction-Free

Factor Contradiction-Free	
Definición	Captura el grado de duplicación (o repetición) de una misma instancia de evento que es representada con datos contradictorios.
Otros nombres	No contradicción.
Métricas genéricas	**Contradictory Event:** – **Granularidad:** event – **Definicion:** Verifica si existen eventos contradictorios dentro de una traza.
Métricas específicas	**contradictoryEvents(trace): eventSet** Dado una traza se devuelven los eventos contradictorios dentro de la misma.

Freshness

Otro aspecto importante es la frescura de los datos, es necesario que los datos del log sean lo mas actualizados posible con respecto a los datos del mundo real. Para esto se ha definido la dimensión "Freshness", la cual captura que los datos registrados por una entidad se encuentren dentro de un rango de tiempo especificado. Se tomaron como referencia la dimensión "Timeliness" de la Tesis de Maestría y la dimensión "Frescura" del FGCDGD.

Se ha definido un único factor para esta dimensión: "Timeliness". Este factor se encarga de chequear que un dato del log no tenga una diferencia de tiempo significativa con respecto al dato del mundo real, y este chequeo se realiza a nivel de traza, evento y atributo. Para que este chequeo sea posible se necesita un input por parte del usuario que indique un rango de tiempo máximo aceptado.

En la tabla 18 se describe la dimensión "Freshness" y a continuación se describe su factor asociado Timeliness.

Tabla 18: Dimensión Freshness

Dimensión Freshness	
Definición	Captura que los datos registrados por una entidad se encuentren dentro de un rango de tiempo especificado.
Otros nombres	Frescura, Exactitud temporal.
Factores	Timeliness

En la tabla 19 se define el factor Timeliness y sus métricas genéricas y específicas.

Tabla 19: Factor Timeliness

Factor Timeliness	
Definición	Captura que los datos registrados por un elemento del log extendido se encuentren dentro de un rango de tiempo, especificado por el usuario o en su defecto su máximo rango posible.
Otros nombres	Oportunidad.
Métricas genéricas	**Attribute Timeliness:** – **Granularidad:** valor de atributo – **Definicion:** Indica si el rango de ejecución de una instancia de atributo se corresponde con el de la instancia del evento al que pertenece. **Event Timeliness:** – **Granularidad:** evento – **Definicion:** Indica si el rango de ejecución de una instancia de evento está incluido dentro del rango de ejecución de la instancia de traza a la que pertenece. **Trace Timeliness:** – **Granularidad:** traza – **Definicion:** Retorna verdadero si el rango de ejecución de una instancia de traza está incluido dentro del rango de ejecución del log extendido.
Métricas específicas	**attributeTimelinessTimestamp(time):bool** Retorna verdadero si el timestamp de un atributo proveniente de la base de datos de negocio referenciado a un atributo del evento al que pertenece se encuentra comprendido (con una tolerancia 'time') en el timestamp del evento. **eventTimelinessTimestamp(event):bool** Dado un evento se indica si el timestamp del evento se encuentra comprendido en el timestamp de la traza a la que pertenece. **traceTimelinessTimestamp(trace):bool** Dada una traza retorna verdadero si el timestamp de la traza se encuentra comprendido en el timestamp del log extendido al que pertenece.

Credibility

En algunos sistemas la credibilidad de los datos puede ser un aspecto importante y crítico. ¿Es confiable la fuente de donde provienen los datos?, ¿La persona que ejecutó la tarea tiene buena reputación?, ¿El valor de un dato registrado es exacto?. Varias preguntas se nos pueden ocurrir relacionadas con la credibilidad de los datos, cuyas respuestas pueden ser de mucha utilidad para el usuario. Por esto se ha definido la dimensión "Credibility", la cual captura el grado de credibilidad y confianza de los datos registrados.

La Tesis de Maestría no profundiza mucho en este aspecto de calidad, mientras que en el FGCDGD no se tiene en cuenta, por lo que para la definición de esta dimensión se tuvo que profundizar en las tareas investigación, a partir de lo planteado en [16], y discusión del equipo. Se definieron dos factores para esta dimensión: "Provenance" y "Trustworthiness". El factor "Provenance" se enfoca en la credibilidad de la procedencia de los datos, mientras que "Trustworthiness" apunta a la confianza sobre los datos.

En la tabla 20 se describe la dimensión "Credibility" y a continuación se describen sus factores asociados: Provenance y Trustworthiness.

Tabla 20: Dimensión Credibility

Dimensión Credibility	
Definición	Captura el grado de credibilidad y confianza de los datos registrados.
Otros nombres	Credibilidad.
Factores	Provenance Trustworthiness

En la tabla 21 se define el factor Provenance y sus métricas genéricas y específicas.

En la tabla 22 se define el factor Trustworthiness y sus métricas genéricas.

Tabla 21: Factor Provenance

Factor Provenance	
Definición	Captura la credibilidad sobre la procedencia de los datos.
Otros nombres	Procedencia.
Métricas genéricas	**Responsibility:** – **Granularidad:** log – **Definicion:** Mide la credibilidad en las personas que participaron de la manipulación de los datos . **Origin:** – **Granularidad:** evento – **Definicion:** Mide la credibilidad del origen del evento. **Reproducibility:** – **Granularidad:** log – **Definicion:** Indica qué tan reproducible es el workflow. Se chequea contra un input de datos, por ejemplo, una lista que diga el evento A y el evento B no pueden ocurrir en una misma traza.
Métricas específicas	**responsibilityEvents([events], [participants]): [events]** El usuario debe ingresar el conjunto de eventos a los que desea chequear que no participen los participantes (también ingresados por el usuario). Retorna eventos, en los cuales participan particioantes en conflicto con el criterio establecido. **reproducibilityEvents(event,[events]): bool** El usuario debe ingresar un evento y los eventos que necesariamente deban ocurrir previamente al mismo. Retorna verdadero si se cumplen los criterios.

Tabla 22: Factor Trustworthiness

Factor Trustworthiness	
Definición	Captura el grado de confianza sobre los datos registrados.
Otros nombres	Fidedigno.
Métricas genéricas	**Believability:** – **Granularidad:** valor de atributo – **Definicion:** Dado un dato, esta métrica mide el grado de creencia de que el dato sea verdadero. **Reputation:** – **Granularidad:** valor de atributo – **Definicion:** Determina la integridad de la fuente de un dato registrado. **Verifiability:** – **Granularidad:** valor de atributo – **Definicion:** Captura el grado en que un conjunto de datos registrados se considera exacto.

Security

La seguridad en los sistemas de información es un área que ha tomado mucho protagonismo en los últimos tiempos. Es muy importante que un sistema sea seguro, y sobretodo es indispensable en los sistemas que manipulan datos sensibles y críticos. Para poder medir la seguridad de un BP integrado con datos organizacionales se ha definido la dimensión "Security" en la cual se captura el grado de seguridad de los eventos y atributos de los mismos.

Esta dimensión no es profundizada en la Tesis de Maestría, tampoco en el FGCDGD, y es una área que no está muy desarrollada en el mundo de la calidad de datos en procesos de negocios. Por lo tanto la definición, al igual que "Credibility", ha llevado más dedicación que las demás dimensiones.

Se han definido tres factores para esta dimensión: "User permissions", "Encrypted data" y "Anonimity". El factor "User permissions" apunta a los permisos de los actores que realizan las tareas del proceso. Por otro lado, el factor "Encrypted data" se enfoca en los datos del log que deben estar encriptados. Por ultimo, el factor "Anonimity" verifica aquellos datos que tienen que estar anonimizados.

En la tabla 23 se describe la dimensión "Security" y a continuación se describen sus factores asociados: User permissions, Encrypted data y Anonymity.

Tabla 23: Dimensión Security

Dimensión Security	
Definición	Captura el grado de seguridad de los eventos y atributos de los mismos.
Otros nombres	Seguridad.
Factores	User permissions Encrypted data Anonymity

En la tabla 24 se define el factor User Permissions y sus métricas genéricas y específicas.

Tabla 24: Factor User Permissions

Factor User Permissions	
Definición	Indica si las acciones las realizan los actores que tengan los permisos.
Otros nombres	Permisos de usuario.
Métricas genéricas	**Authorized User:** – **Granularidad:** evento – **Definicion:** Mide que la persona que realizó una acción sobre un evento tiene los permisos correspondientes a dicha acción.
Métricas específicas	**authorizedUserPermissions([participants],event): [participants]** El usuario debe ingresar la lista de participantes autorizados a ejecutar un evento en especifico. Retorna los participantes ajenos al conjunto de autorizados del evento.

En la tabla 25 se define el factor Encrypted Data y sus métricas genéricas.

Tabla 25: Factor Encrypted Data

Factor Encrypted Data	
Definición	Captura el grado de datos encriptados de un conjunto de atributos de un evento.
Otros nombres	Datos encriptados.
Métricas genéricas	**Encrypted attribute:** – **Granularidad:** valor de atributo – **Definicion:** Mide que la información de un atributo de evento que deba estar encriptado lo esté. **Ratio encrypted attribute(attributeSet): float** – **Granularidad:** evento – **Definicion:** Retorna el porcentaje de atributos con sus valores encriptados, en el total de los atributos indicados como encriptados.

En la tabla 26 se define el factor Anonymity y sus métricas genéricas y específicas.

Tabla 26: Factor Anonymity

Factor Anonymity	
Definición	Captura el grado de anonimización de un conjunto de atributos de un evento.
Otros nombres	Anonimato.
Métricas genéricas	**Anonymous attribute:** – **Granularidad:** valor de atributo – **Definicion:** Indica que la información de un atributo que deba estar anónima lo esté. **Ratio anonymous attribute:** – **Granularidad:** evento – **Definicion:** Indica el porcentaje de información de atributos que deba estar anónima de un evento lo estén.
Métricas específicas	**anonymousAttribute(attribute, formato): [attributes]** El usuario debe ingresar una expresión regular con el patrón de un atributo que deba permanecer anónimo. Por ejemplo con la siguiente expresion regular "`^user[0-9]{3}$`" se chequea que el valor sea del estilo: "user101". **ratioAnonymousAttribute(attributeSet): float** Retorna el porcentaje de atributos con sus valores anónimos, en el total de los atributos indicados como anónimos.

Comentarios generales de la definición del modelo

- En las definiciones de las primeras cinco dimensiones, con sus respectivos factores y métricas en algunas se tomó como referencia una de las dos propuestas mencionadas en la secciones 2.2.1 y 2.2.2 y en otras se tomaron ambas.

- Para la definición de las últimas dos dimensiones con sus respectivos factores y métricas el esfuerzo fue mayor, ya que no se contaba con trabajos previos en los que se puedan tomar como inspiración.

- Se definieron métricas específicas exclusivamente a las métricas genéricas que se abordan en la implementación.

5. Desarrollo del Prototipo

En la sección anterior se presentó en detalle el modelo de calidad BPODQM para datos integrados de procesos y organizacionales, con la definición de sus dimensiones, factores, métricas y métodos de medición como se describe en el capítulo 4. A partir de esto, surge la necesidad de diseñar e implementar un prototipo de herramienta que pueda llevar este modelo a la práctica tomando como input un log de eventos extendido.

Este capítulo cuenta con dos secciones. En la primer sección haremos un breve análisis de la herramienta y luego veremos algunos problemas de diseño a los que nos hemos enfrentado. En la segunda sección describiremos la etapa de implementación del prototipo de herramienta.

5.1. Análisis y Diseño de la herramienta

En esta sección se pondrá foco en el trabajo previo a la implementación de la herramienta. Se realizará un análisis de lo que se intenta resolver con esta herramienta. Luego se hará énfasis en el diseño de la misma.

ProM

Como se ha visto en el capitulo 2, ProM es un framework extensible que soporta una amplia variedad de técnicas de minería de procesos de negocio mediante el uso de plug-ins. Este framework provee una aplicación de escritorio en la cual están disponibles una gran cantidad de plug-ins implementados por desarrolladores de la comunidad.

ProM es de código abierto y es uno de los mas usados por la comunidad en el área de procesos de negocios. Por estos motivos se ha elegido esta herramienta para implementar un plug-in que resuelva nuestras necesidades.

La arquitectura de ProM se puede observar en la figura 17. ProM soporta diferentes sistemas, formatos de archivos, algoritmos de minería, y tecnicas de análisis. Es posible agregar nuevos plug-ins de minería sin necesidad de cambiar el framework. [17]

Cabe mencionar que este diagrama de arquitectura fue creado en la versión 5.2 de ProM. El artículo “ProM 6: The Process Mining Toolkit” [18] describe las mejoras de la versión 6.0 en adelante.

Problema

La herramienta a implementar debe aplicar el modelo de calidad de datos definido, a un log de eventos extendido. Esto implica que dado un log de eventos de entrada, la herramienta tiene que ser capaz de analizar el log identificando sus trazas, eventos y atributos, para así poder aplicar sobre estos elementos los métodos definidos por las métricas del modelo.

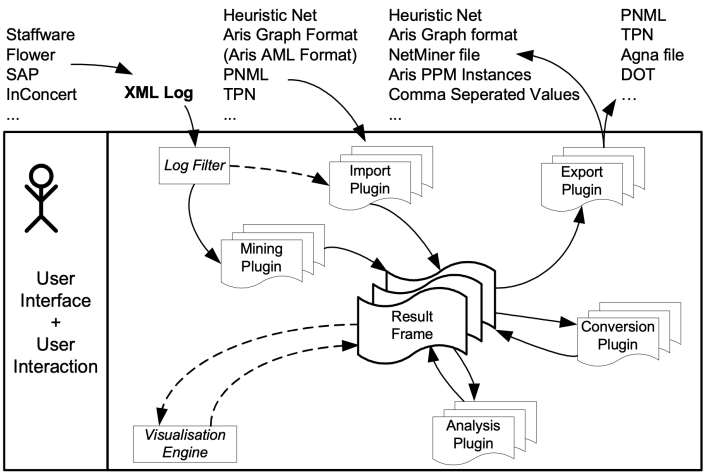


Figura 17: Arquitectura de ProM, extraído de [17].

Se debe brindar una interfaz gráfica amigable en la cual el usuario pueda aplicar distintas métricas de calidad a los datos de entrada, y proveer en detalle los resultados de las mismas. Estos resultados deben poder ser exportados a un archivo para su posterior análisis.

Requerimientos

Los principales requerimientos que la herramienta debe cumplir se listan a continuación:

- Ingresar parámetros de entrada para las métricas que lo requieran.
- Ejecutar el modelo de calidad para un log de eventos extendido.
- Visualizar los resultados de las métricas.
- Exportar los resultados a un archivo CSV.
- La herramienta debe ser extensible, de manera que sea fácil agregar nuevos items al modelo de calidad.

Parámetros de entrada

Uno de los requerimientos principales de la herramienta es la posibilidad de ingresar parámetros de entrada para las métricas que así lo requieren. Esto nos ha llevado a tener que tomar una decisión de diseño, la cual consiste en: cuándo y cómo pedir estos parámetros de entrada.
Para esto se tuvo en cuenta dos opciones:

- Pedir los parámetros durante la visualización de las métricas.
  Esta solución consiste en pedir los parámetros para una métrica cuando se accede a la misma a través de la interfaz gráfica del plug-in. La ventaja que nos brinda esta opción es que no es necesario ingresar parámetros para todas las métricas que lo requieran, sino que el usuario ingresa los parámetros únicamente para aquellas métricas que quiere visualizar sus resultados. Esto hace que la funcionalidad sea amigable y fácil de usar. La gran desventaja que tiene esta opción es que cada vez que se ingresen parámetros para una métrica se

necesita recorrer el log entero para así aplicar las mismas, lo cual lleva a un problema de performance importante.
- Pedir los parámetros antes de la ejecución de la visualización de las métricas. En este caso se piden los parámetros cuando se inicia el plug-in, recorriendo una vez sola el log, para los casos donde se le permite seleccionar de datos que existen en el log ingresado. La ventaja de esta solución es que es óptima debido a que el log se recorre una vez extra para poder aplicar las métricas. Por otro lado, la desventaja que nos presenta esta solución es que el usuario debe ingresar los parámetros para todas las métricas que lo requieran, inmediatamente al iniciarse la ejecución del plug-in. Esta desventaja obliga a que el usuario sepa de antemano cuales métricas quiere visualizar sus resultados y cuales son los parámetros para las mismas.

Luego de analizar las ventajas y desventajas de las dos posibles soluciones, se tomó la decisión de ir por la solución de pedir los parámetros antes de la ejecución de la visualización de las métricas, debido a que consideramos que la performance de la ejecución del plug-in es un factor importante.

Extensibilidad

Otro requerimiento importante a tener en cuenta consiste en que la herramienta debe ser extensible. Esto quiere decir que si en un futuro se quiere implementar nuevas métricas de calidad, se tiene que poder hacer de forma sencilla de manera que la dificultad solo se centre en los cálculos de la métrica y no en la integración.

Para cumplir este requisito lo que se decidió fue crear una interfaz genérica de modo que cada métrica implemente la misma. Esta interfaz define un conjunto de métodos que sirven para calcular la calidad de un dato para todas las granularidades posibles y todas las métricas deben implementar estos métodos.
Además se mantiene una estructura de datos con las instancias de todas las métricas, de forma de recorrer esta estructura e ir calculando la calidad de un dato para cada métrica.

De esta forma la integración de una nueva métrica consistirá de solamente tres (o cuatro) sencillos pasos:

1. Crear una clase que represente a la métrica nueva e implemente la interfaz mencionada anteriormente.

2. Inicializar la métrica al inicio de la ejecución del plug-in.

3. Agregar la instancia de la métrica a la estructura de datos que contiene todas las instancias.

4. En caso de que la métrica necesite input, se debe agregar una pestaña al panel donde se introducen los inputs. La pestaña debe contener la descripción de la métrica y los campos necesarios para procesar el valor de entrada.

Arquitectura

En la figura 18 se describe el diagrama de componentes del sistema, y se describe función que cumple cada uno de ellos.

El componente principal de esta arquitectura es la *Lógica de negocio*, la cual ocurre todo dentro del framework de ProM. Este componente se puede dividir en dos partes que lo componen, la "Ejecución" y la "Visualización de resultados".

Por un lado el módulo que se encarga de la ejecución realiza dos tareas fundamentales. La primera se basa en procesar las entradas que provee el usuario para aquellas métricas que requieran parámetros de entrada. Estos parámetros de entrada se utilizan luego en la segunda tarea, la cual consiste en procesar el log de eventos que se recibió como entrada al plug-in. En este procesamiento se recorren las trazas y eventos del log para así poder ir aplicando las métricas del modelo de calidad a los datos.

Por otro lado está el módulo que se encarga de la visualización de los resultados, en el cual para cada métrica se muestra en detalle el resultado de aplicar dicha métrica de calidad a los datos del log de eventos.

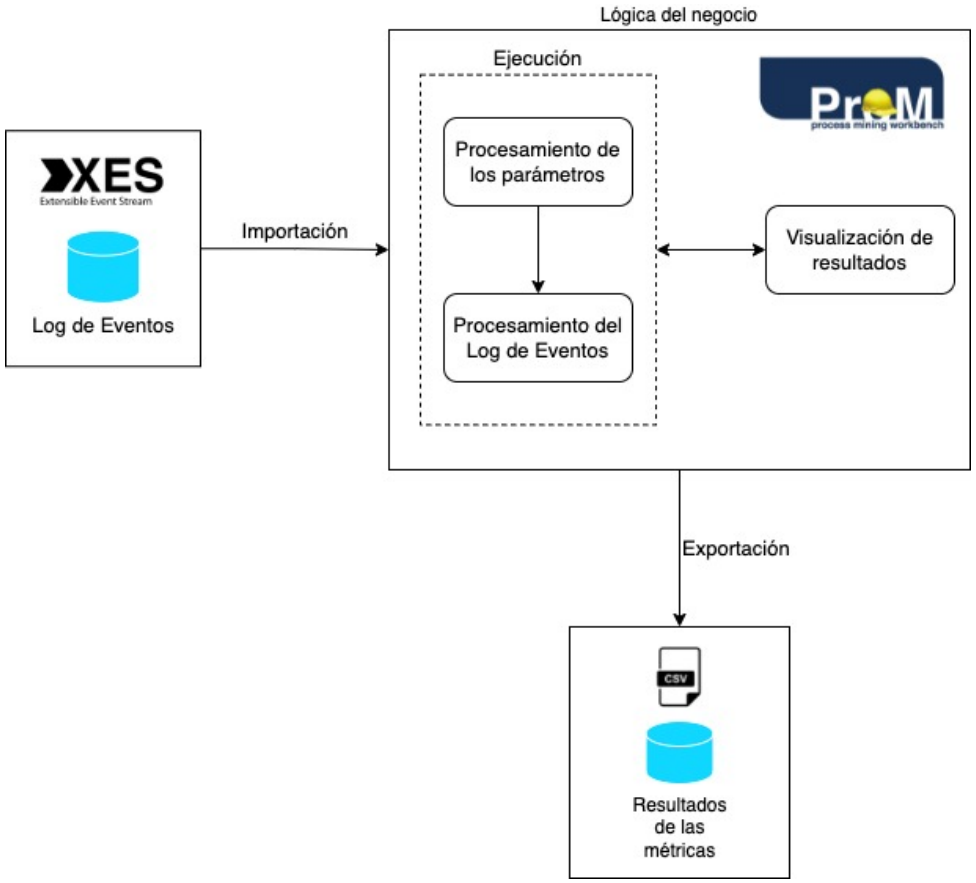


Figura 18: Arquitectura del plug-in

Por último, se provee un modulo para poder exportar los resultados obtenidos en la ejecución a un archivo con formato csv.

5.2. Implementación

En esta sección se presenta la implementación de la herramienta. Se implementó un plug-in para el framework ProM, utilizando el lenguaje de programación Java.

5.2.1. Librería OpenXES

El meta-modelo XES fue diseñado para ser independiente de cualquier implementación, y la librería OpenXES es una implementación que se toma de referencia que fue diseñada para llevar a cabo los siguientes objetivos [19]:

- Cumplir totalmente con el estándar XES en todos los aspectos.
- Fácil de integrar y ser sencillo de usar por parte de los desarrolladores.
- Proporcionar el máximo rendimiento para la gestión y el almacenamiento de datos de registro de eventos.
- Servir como implementación de referencia clara y comprensible para otras implementaciones de la norma.

En el resto de esta sección, se describe la implementación OpenXES del estándar XES con más detalle. La misma sirve como guía de alto nivel para los desarrolladores que buscan para implementar sistemas que requieren almacenamiento, administración, serialización o análisis de logs de eventos.

Jerarquía de tipos de modelos

Los elementos de jerarquía de tipo de modelo reales del estándar XES también están definidos por interfaces que se pueden encontrar en el paquete *org.deckfour.xes.model*:

- **XElement:** La interfaz XElement se amplía con todos los elementos de la jerarquía de tipo de modelo XES en OpenXES. Define que cada elemento necesita tener un XID.
- **XLog:** La interfaz XLog amplía la interfaz XElement. Además, amplía la interfaz genérica Set<XTrace>para acceder y modificar el conjunto de elementos traza contenidos.
- **XTrace:** La interfaz XTrace amplía la interfaz XElement. Además, amplía la interfaz genérica List<XEvent>para acceder y modificar la lista ordenada de elementos de registro contenidos.
- **XEvent:** La interfaz XEvent amplía la interfaz XElement.

Los ID son una parte integral del estándar XES, ya que son atributos obligatorios para cada elemento de la jerarquía del tipo de modelo. En OpenXES, la gestión de ID se implementa según el estándar XES y se representa en el paquete *org.deckfour.xes.id*:

- La clase **XID** encapsula los ID de forma transparente y proporciona herramientas para leerlos desde su representación de cadenas y flujos de datos.
- La clase **XIDFactory** proporciona medios para generar ID únicos, basándose en el estado actual del sistema. Los métodos de fábrica proporcionados por esta clase son la forma recomendada de crear instancias de XID.

Cada elemento de la jerarquía de tipo de modelo XES puede equiparse con atributos, incluso los propios atributos. Por lo tanto, la gestión de atributos también tiene un papel importante en la implementación de OpenXES. Las interfaces importantes para esta tarea se encuentran en el paquete *org.deckfour.xes.model*, aunque pueden existir varias implementaciones.

XAttribute: la interfaz XAttribute define el esqueleto básico para los atributos en OpenXES, incluido el acceso a la clave y la extensión de los atributos (si está disponible). El tipo de un atributo, así como el acceso al valor de un atributo, lo proporcionan las subinterfaces fuertemente tipadas de XAttribute de la siguiente manera:

- XAttributeLiteral
- XAttributeBoolean
- XAttributeContinuous
- XAttributeDiscrete
- XAttributeTimestamp
- XAttributeDuration
- XAttributeID
- XAttributeCollection
- XAttributeList
- XAttributeContainer

XAttributeMap: la interfaz XAttributeMap define un contenedor para atributos. No es deseable adjuntar atributos directamente a sus respectivos elementos de jerarquía de tipo de modelo, tanto por razones de claridad como de eficiencia de implementación. Cada objeto que puede tomar atributos los almacena y los administra dentro de una instancia de esta interfaz.

XAttributable: la interfaz XAttributable define las capacidades de los elementos de jerarquía de tipo de modelo, que se pueden equipar con atributos.

Extensions

OpenXES proporciona implementaciones convenientes para las extensiones estándar definidas para XES. Estas implementaciones se realizan como clases singleton y se pueden encontrar en el paquete *org.deckfour.xes.extension.std*.

Las implementaciones de extensión estándar proporcionan métodos para acceder y modificar atributos definidos por la extensión respectiva de una manera conveniente y fuertemente tipada. Actualmente, el siguiente conjunto de extensiones estándar está implementado en OpenXES:

- XConceptExtension
- XIdentityExtension
- XLifecycleExtension
- XOrganizationalExtension
- XSemanticExtension

- XTimeExtension
- XCostExtension

5.2.2. Estructura del plug-in

Al implementar el plug-in un factor muy importante fue su estructura. Tener una estructura bien definida facilita a todo aquel que quiera entender el código y funcionamiento, como también para el que desea escribir código.
Uno de los objetivos del plug-in fue lograr que sea de fácil entendimiento para que se pueda llevar a cabo la extensión de nuevas implementaciones de métricas.

La estructuracion del plug-in se realizó mediante el uso de distintos *packages*. Cada *package* contiene diferentes clases Java que se relacionan entre si con otros *packages*. A continuación se presenta una descripción general de estos *packages*, sus clases y se indica a que modulo pertenecen de la figura 18:

- Constructors: Este *package* contiene las clases que representan la salida del plug-in. Estas clases contienen métodos constructores para la salida y brindan métodos para objetos específicos. Pertenece al módulo *Visualización de resultados*.
- Dataqualityaspects: este *package* contiene cada verificación de calidad de métrica, y el nombre de la clase indica qué aspecto de calidad se está verificando en esa clase. Pertenece al módulo *Procesamiento del Log de Eventos*.
- Helpers: este *package* contiene funciones que son de utilidad para diversas métricas de calidad, evitando duplicación de código. Pertenece al módulo *Procesamiento del Log de Eventos*.
- Interfaces: este *package* contiene todas las interfaces que se han creado para el plug-in. La interfaz que se utiliza es la interfaz de control de calidad. Esto define, para cada clase de aspecto de calidad de datos, los métodos que son comunes entre ellos y que son utilizados por cada una de las métricas en sus ejecuciones. Pertenece al módulo *Procesamiento del Log de Eventos*.
- Listcreators: estas son las clases que contienen listas específicas que se requieren para controles de calidad. Algunos controles de calidad requieren una lista con información específica en lugar de solo un valor de atributo específico, de ahí la necesidad de crear y completar diferentes listas. Dichas listas pueden ser utilizadas para mas de una métrica lo cual hace mas eficiente la ejecución del plug-in. Pertenece al módulo *Procesamiento del Log de Eventos*.
- Models: Este *package* contiene una representación del modelo de calidad para la interfaz de usuario, permitiendo instanciar métricas y factores del mismo. Pertenece al módulo *Visualización de los resultados*.
- Plug-ins: El *package* contiene el marco principal del plug-in (con todas las iteraciones necesarias que se deben realizar en el log ingresado) y la exportación de los resultados obtenidos a csv. Este *package* participa en los módulos *Procesamiento de los parámetros*, *Procesamiento del Log de Eventos* y *Exportación de los resultados*.
- Visualizer: este *package* contiene el visualizador del plug-in con todas las propiedades aplicadas al mismo. Pertenece al módulo *Visualización de los resultados*

Inputs del usuario

Durante la implementación de las distintas métricas de calidad presentadas el objetivo fue que las mismas sean implementadas de la forma mas genérica posible. Esto quiere decir que el resultado de aplicar una métrica de calidad sea independiente al log de eventos que se le ingresa como entrada al plug-in.

Para un conjunto de métricas de calidad se las pudo implementar de forma genérica, pero al intentar implementar otro conjunto de métricas surgió la dificultad de que estas dependen fuertemente del contexto del log de eventos que se ingrese.

Para resolver el conjunto de métricas que tienen una dependencia fuerte del contexto del log de eventos, se le brinda al usuario antes de realizar el calculo de las métricas de calidad distintos input donde el usuario puede ingresar datos los cuales seran utilizados para realizar los calculos.

Los tipos de input puede ser desde una lista de posibles valores que puede tener un atributo, expresiones regulares que deban cumplir los atributos de determinado evento, porcentajes de ponderación de un atributo, etc.

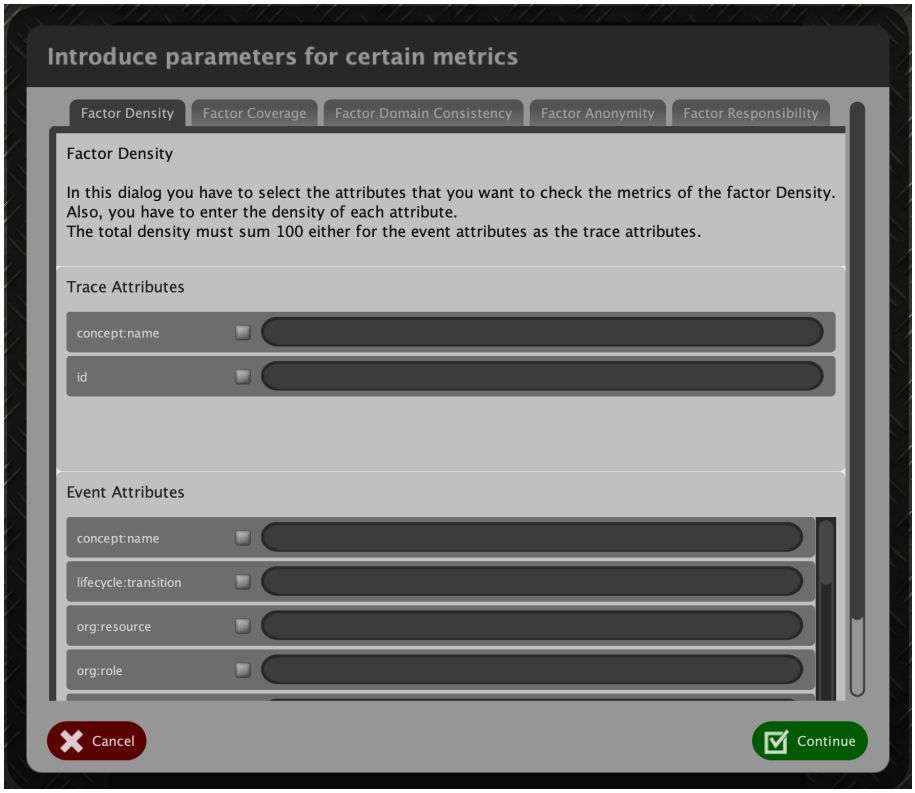


Figura 19: Inputs del plug-in

En la Figura 19 se puede apreciar los distintos input que se le presentan al usuario para que seleccione e ingrese los que desee. Además se aprecian que estan agrupados por factor, y se les brinda una descripción.

Interfaz

Debido al gran tamaño que puede llegar a tener un log de eventos, era necesario encontrar una solución que tuviera esto en cuenta y lograra eficiencia al analizar el log con las múltiples métricas implementadas. No era una opción posible el recorrer el log en cada métrica que se debía analizar, era necesario poder analizarlo con la menor cantidad de recorridas.

Al realizar esta implementación nos enfrentamos a diversas dificultades. Dependiendo de la métrica a analizar, se puede necesitar datos que son propios de la traza, del evento o de un atributo en particular. Estos datos pueden llegar a tener el mismo nombre a pesar de ser de distinta jerarquía a nivel estructural del log, lo cual le agregaba cierta dificultad al momento de almacenar y trabajar con estos datos en el calculo de ciertas métricas.

Para lograr esto se creó una clase *<interface>*, la cual cada una de las métricas de calidad heredan de ella. En la figura 20 se puede observar el diagrama de clases asociado a la clase que consta de los siguientes métodos:

- initialize();
- checkQuality(XLog eventlog);
- checkQuality(XLog eventlog, XTrace trace);
- checkQuality(XLog eventlog, XTrace trace, XAttribute att);
- checkQuality(XLog eventlog, XTrace trace, XEvent event);
- checkQuality(Xlog eventlog, XTrace trace, XEvent event, XAttribute att);
- checkClear(CentralRegistryInterface list);
- getResult();

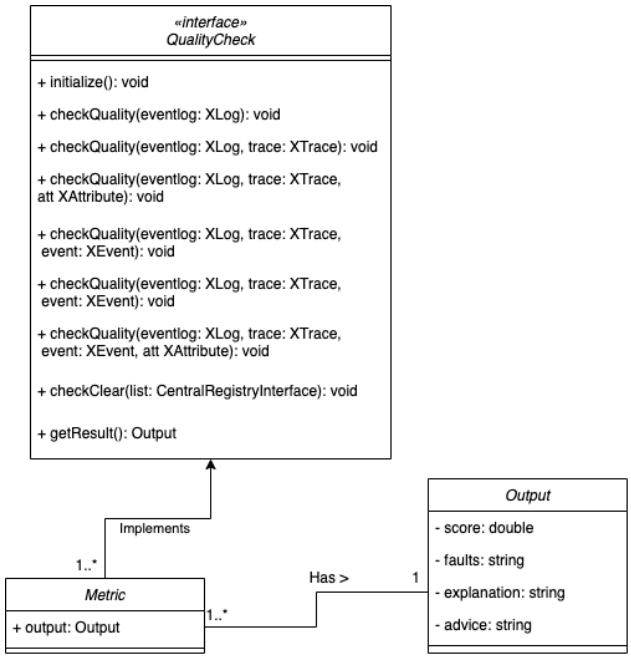


Figura 20: Diagrama de clases asociado a la interfaz

Gracias a las funciones provistas en la interface, para cada nivel se puede verificar

la calidad si la métrica de calidad lo necesita. En el caso de que una métrica de calidad no utilice algún nivel, el método se mantiene vació. Los niveles pueden ser log, traza, atributo de traza, evento, atributo de evento. Al ejecutarse de esta forma, solo necesitamos pasar por el log de eventos dos veces, logrando así la eficiencia deseada.

Cada verificación de métrica de calidad de datos de eventos se coloca en una lista. Cada vez que se solicita un método para el control de calidad, lo hacemos para cada métrica de calidad presente en la lista. Para esto se recorre la lista de métricas de calidad y llamando al método especificado. A continuación se realiza un punteo de los pasos que se llevan a cabo al recorrer el log, con sus trazas, eventos y los atributos correspondientes a cada nivel:

1. Lo primero es recibir como entrada un log de eventos (puede ser un log extendido como se menciono en la subsección 3.3 o no) en formato XES como entrada.
2. Se reciben los input ingresados por el usuario que serán utilizados para el calculo de las diferentes métricas de calidad. Previamente se obtienen los nombres de los atributos organizacionales, que se muestran como opción al usuario.
3. Se llama al método checkQuality(XLog eventlog) para cada métrica de calidad.
4. Se completan listas específicas con información del log si es necesario.
5. Se recorren todos las trazas del log de eventos.
6. Se llama al método checkQuality(XLog eventlog, Xtrace, trace) para cada métrica de calidad.
7. Se completan listas específicas con información de la traza si es necesario.
8. Se recorren todos los atributos de la traza actual.
9. Se llama al método checkQuality(Xlog eventlog, Xtrace trace, XAttribute att) para cada métrica de calidad.
10. Se completan listas específicas con atributos de información de trazas si es necesario.
11. Se recorren todos los eventos de la traza actual.
12. Se llama al método checkQuality(Xlog eventlog, Xtrace trace, XEvent event) para cada métrica de calidad.
13. Se completan listas específicas con información del evento si es necesario.
14. Se recorren todos los atributos del evento actual.
15. Se llama al método checkQuality(Xlog eventlog, Xtrace trace, XEvent event, XAttribute att) para cada métrica de calidad.
16. Se completan listas específicas con atributos de información de eventos si es necesario.

17. Luego de terminar cada uno de los recorridos del log, se invoca al método checkClear(CentralRegistryInterface list) para cada métrica de calidad. Esto se hace al final porque todas las listas necesarias para las implementaciones de las métricas de calidad se han completado durante las recorridas del log de eventos. En esta función es donde se realiza la mayoría de las implementación del calculo de la métrica de calidad.

18. Se crea la salida para el plug-in y se coloca en la interfaz de usuario.

Esta lista de pasos se muestra visualmente en la Figura 21. Al realizar este tipo de implementación, solo se recorre el log de eventos dos veces. Esta es la solución que se ha realizado para mejorar la eficiencia del plug-in, al no querer que el tiempo de ejecución pueda llegar a ser extremadamente extenso.

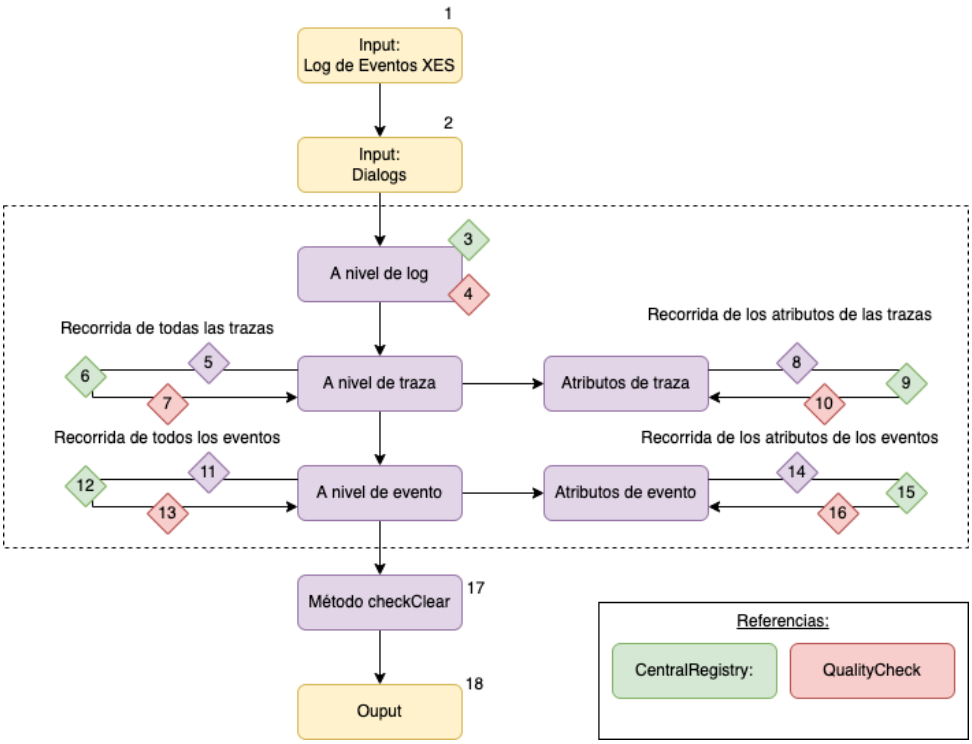


Figura 21: Etapas del plug-in

Visualización

Para cada métrica de calidad se necesita definir algún tipo de salida que el visualizador de ProM pueda leer. Para poder lograr que todas las métricas tengan la misma forma de visualización, se creó una *interfaz*, y de esta forma lograr definir una visualización estructurada. La interfaz definida tiene los siguientes atributos:

- **Name:** El nombre de la métrica de calidad.
- **Description:** La descripción de la métrica de calidad
- **Score:** la puntuación que devuelve la ejecución de la métrica de calidad.
- **Explanation:** Una explicación sobre cómo funciona el sistema de puntuación para la métrica de calidad.

- **Error (o Warning):** Se indica en un comentario global los errores cometidos en el log para dicha métrica de calidad, en caso de que hayan.
- **Details:** Detalla los errores que presenta el log para esa métrica de calidad.

De esta manera, como la visualización es uniforme, es sencilla la comprensión de los datos obtenidos para cada una de las métricas aplicadas al log ingresado.

Exportación

Luego de que el usuario ingrese el log de eventos, e ingrese los datos a los input que se esperaban para realizar cálculos para las distintas métricas de calidad, puede visualizar los resultados en la interfaz de usuario.

Sin embargo, luego de un análisis del log por diversas métricas de calidad, las cuales algunas de ellas recibieron un input de usuario en el instante previo a su ejecución, los resultados son mostrados únicamente mientras se mantiene la sesión de ProM activa.

Para poder visualizar los resultados se deberán ingresar los mismos input, y el mismo log de eventos, el cual puede tener una gran dimensión, lo cual haría que recalcular las métricas de calidad no sea práctico.

De esta problemática surge la posibilidad brindada al usuario de que pueda descargar los resultados de las métricas, con sus faults y score correspondientes a un archivo csv.

La organización del archivo es de la forma:
- Nombre de la métrica
- Para cada atributo al que fue aplicada la métrica de calidad (en caso de ser más de uno):
  - Score
  - Error o Warning (en caso de tener)

5.2.3. Interfaz de Usuario

Luego de que el log de eventos es cargado y se decide ejecutar el plug-in al usuario se le muestran distintos pestañas con campos en los que puede ingresar un input el cual sera usado para las métricas de cierta dimensión. Como se puede apreciar en la Figura 19 en cada pestaña se le brinda una explicación al usuario de como debe ser ingresado el input y para que métrica será utilizado.

Esto se debe a que existen métricas que dependen de un dato ingresado por el usuario para poder ser calculadas, en caso de que no se les ingrese no serán calculadas.

Luego de que el usuario haya ingresado los datos deseados para los cálculos de las métricas, se le muestra una visualización de lo obtenido tras la ejecución con el log de eventos que fue cargado.

En la figura 22 se puede apreciar en la parte superior el nombre del plug-in y por debajo una breve descripción. En el costado izquierdo se ve un panel, donde el titulo

te indica que selecciones una dimensión, y por debajo se lista cada una de las dimensiones en distintos botones. Luego de seleccionar una dimensión se listan debajo los factores correspondientes a dicha dimensión bajo el titulo de que se debe seleccionar un factor. Por último se debe seleccionar una métrica de ese factor seleccionado, las cuales también se listan por debajo, bajo el titulo de que se debe seleccionar una métrica.

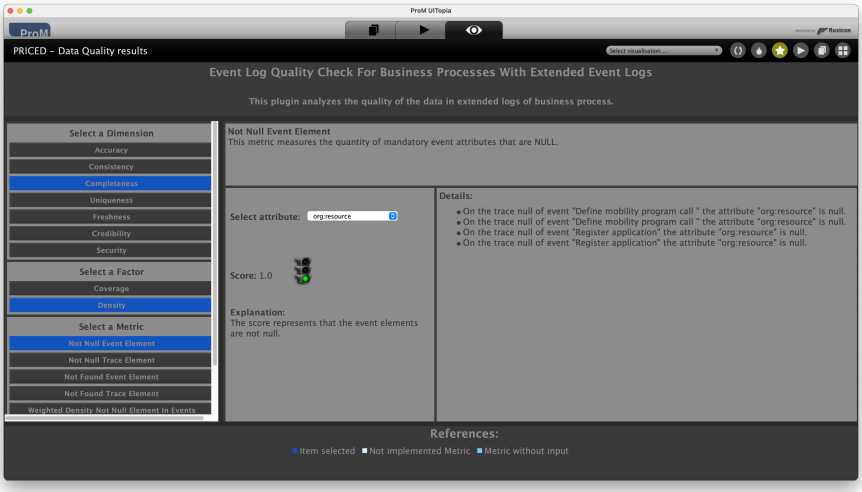


Figura 22: Interfaz del plug-in

En el camino en el que se va a elegir que métrica se desea ver los resultados que le asigna el plug-in, pueden pasar 3 cosas, la métrica deseada puede no estar implementada (como se mencionará en la subsección 5.2.5), estar implementada pero que necesite un input de usuario (que no fue ingresado) para ser analizada, o simplemente estar implementada y ejecutada correctamente.

Para distinguir los distintos casos los botones en las métricas tienen un color que las identifica, y el significado se puede ver en la parte inferior de la vista, donde se encuentran las Referencias.

Luego de elegida la métrica, en el rectángulo de fondo gris claro se divide en 3 partes:

- La superior muestra el nombre de la métrica elegida y una descripción de la misma.
- En la división izquierda,
  - Se muestra un combo donde el usuario puede seleccionar sobre cual de esos atributos desea ver el análisis (en el caso de las métricas que fueron analizadas para más de un atributo).
  - Por debajo se puede apreciar el score asignado, y a su lado un semáforo. El color del semáforo es verde si el score es superior a 0.8 y menor o igual a 1, es amarillo si es mayor a 0.6 y menor o igual a 0.8, o rojo si es menor a 0.6.
  - La explicación de lo que representa el score. Donde el score cuanto mas cercano sea a 1 menos errores tiene, y por el contrario cuanto mas cercano

a 0 tiene mas errores de calidad detectados en la métrica.
    - Si el semáforo es amarillo se muestra "Warning" y un mensaje que explica porque es un análisis que presenta una advertencia, si es rojo se muestra "Error" y un mensaje que explica porque es un análisis que es considerado como error, en cambio si es verde no se muestra nada.
- En la subsección derecha se muestra los datos que no cumplieron con lo solicitado en la métrica de calidad.

Por último y no menos importante se provee al usuario la posibilidad de que todas las métricas calculadas junto a sus scores, y detalles sean exportadas en un archivo CSV. Esto lo consideramos importante, porque si el usuario necesita hacer un buen análisis de los datos, de no poder exportar debería mantener a ProM en dicha salida y analizar todo en el momento. Gracias a poder exportar los datos los mismos pueden ser enviados a distintos colegas o analizados posteriormente.

Para poder exportarlos se debe hacer clic como se indica en la Figura 23 y luego como se puede ver en la Figura 24 se debe seleccionar el archivo cuyo tipo sea OutputDefinition y luego hacer clic en "Export to disk". Se le debe ingresar un nombre al archivo y una ruta donde desea ser almacenado, y se termina de realizar la descarga.

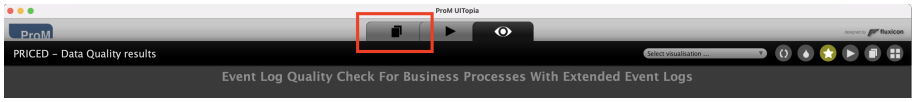


Figura 23: Exportación a csv

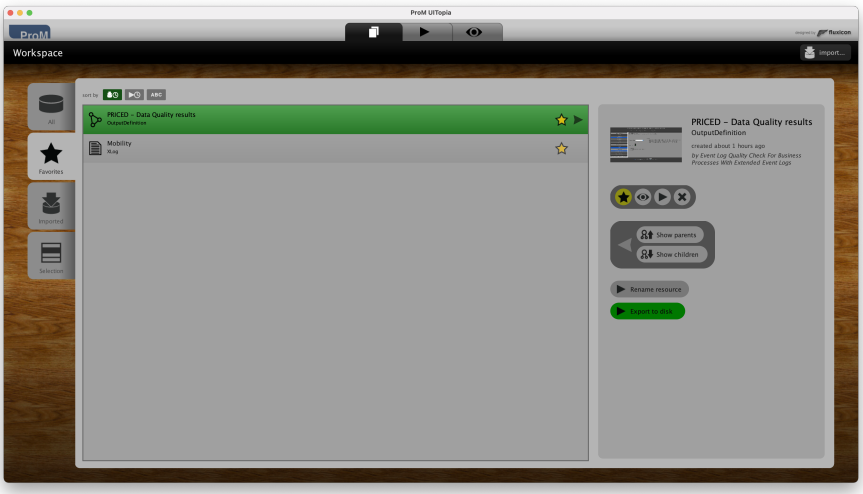


Figura 24: Exportación a csv

5.2.4. Métricas de Calidad implementadas

En el transcurso del proyecto, se consideró implementar la mayor cantidad de métricas de calidad de datos. A pesar de esto, no se implementaron en su totalidad las métricas del modelo propuesto en el proyecto. La no implementación en su totalidad

se debe a muchos factores, pero como principal razón es el tiempo que se disponía, ya que el objetivo central del proyecto era la definición del modelo presentado en el capitulo 4.

A su vez, se buscó que el plug-in y su implementación se enfocara en la calidad del mismo, donde las métricas funcionen de forma correcta. Teniendo en cuenta estos criterios el objetivo planteado fue implementar al menos una métrica por dimensión del modelo. En la figuras 15 y 16 se puede apreciar cuales son las métricas que fueron implementadas y cuales no se llegaron a implementar.

El objetivo planteado fue alcanzado con éxito, incluso si se observa las dimensiones Completeness y Uniqueness se implementaron la totalidad de las métricas. En el capitulo 4 se da una descripción de la implementación de las métricas.

5.2.5. Métricas de Calidad no implementadas

Como se mencionó en la subsección 5.2.4 el objetivo planteado fue implementar al menos una métrica por dimensión del modelo, por tanto hubo métricas las cuales no se llegaron a implementar por distintos motivos, los cuales serán explicados.

La métrica *Trace timeliness (Timeliness)* no se llevó a cabo la implementación debido a que ProM en su normal funcionamiento ya se encarga de corregir este problema de calidad por si solo.

De forma común, un motivo por el cual no se implementaron la totalidad de las métricas del modelo presentado en el capitulo 4 es que ya se habían implementado al menos una métrica de la dimensión a la cual pertenece la métrica no implementada y el tiempo que conllevaba su implementación. Las siguientes métricas se corresponden únicamente a esta razón:

- Weak Semantic correctness (Accuracy)
- Strong Semantic correctness (Accuracy)
- Authorized User (Security)
- Ratio Anonymous Attribute (Security)

No se logro encontrar una forma sencilla para que el usuario defina e ingrese en el plug-in una propiedad o (patrón de encriptamiento) que deba cumplir cierto atributo de los elementos del log:

- Intentional Values (Consistency)
- Reproducibility (Credibility)
- Encrypted Attribute (Security)

La métrica *Ratio Encrypted Attribute (Security)* no se pudo implementar por depender directamente de la anteriormente mencionada *Encrypted Attribute (Security)*.

Por último para las siguientes métricas genéricas no se logró definir una métrica especifica que se adecúe a una situación genérica. La definición en cada caso depende fuertemente del contexto del log ingresado, dado que en cada contexto particular puede tomar relevancia un atributo distinto y una forma particular de medición:

- Inter-event Rule (Consistency)
- Intra-event Rule (Consistency)
- Origin (Credibility)
- Believability (Credibility)
- Reputation (Credibility)
- Verifiability (Credibility)

5.2.6. Guía para uso del plug-in

Para usar el plug-in se necesita ingresar un log de eventos (puede ser un log de eventos extendido como fue explicado en la subsección 3.3) en formato XES. En el caso de que los datos a ingresar no tengan un formato XES, ProM brinda complementos los cuales se pueden utilizar para hacer una conversión del archivo.

Luego de que se importa el log en ProM se puede ejecutar el plug-in. Para esto se puede realizar una búsqueda por el nombre, o simplemente buscarlo en la lista de plug-ins, como se aprecia en la Figura 25.

Figura 25: Busqueda del plug-in

Como se vio en 5.2.3 después que se selecciona, el plug-in presenta unas vistas donde se le ofrece al usuario que ingrese distintos input los cuales serán usados para el análisis de ciertas métricas de calidad. En el caso de que el usuario no ingrese ningún input que sea necesario para el cálculo de una métrica especifica, la misma no sera calculada, y esto se vera reflejado en la vista luego de la ejecución.

Luego el plug-in procesa el log de eventos seleccionado y al recorrerlo como se explico en la subsección 5.2.2 evalúa el score para cada una de las posibles métricas

de calidad. Se muestra una barra de progreso, que indica qué tan avanzado está el plug-in con la ejecución, como se aprecia en la Figura 26.

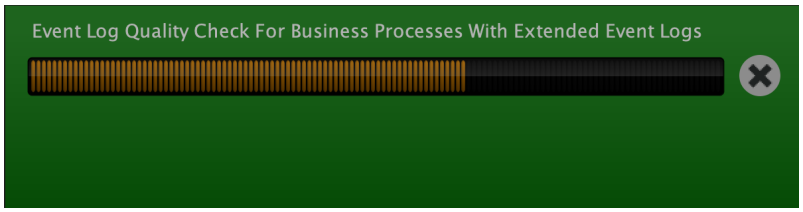


Figura 26: Barra de progreso

Al terminar la barra de progreso, se puede apreciar el visualizador de ProM, donde se ven las distintas dimensiones, las cuales contienen sus factores, y ellos sus propias métricas. El análisis de ésta visualización, los detalles y aspectos caracteristicos de cada metrica fueron analizados en la subsección 5.2.3.

6. Caso de Estudio

En este capítulo se presenta un ejemplo práctico para mostrar el plug-in desarrollado aplicado a un caso de estudio concreto y que se puedan materializar los conceptos planteados previamente. En las próximas secciones se describen los datos de entrada y el dominio de donde provienen. Se realizó un análisis de los resultados obtenidos, y por último se presenta una conclusión del caso de estudio planteado.

6.1. Datos de entrada

En ésta sección se brinda una explicación de la estructura del log de datos a utilizar, luego se describe como se preparó el log de eventos para este caso de estudio, y por ultimo se explica la configuración de parámetros que fue utilizada para el análisis.

6.1.1. Descripción del log de eventos

Como se mencionó en el capítulo 3 para este caso de estudio se utilizó un BP real llamado "Students Mobility" descrito en la figura 13a, el cual esta integrado con un modelo de datos organizacionales descrito en la figura 13b. A partir de la ejecución real de este BP y mediante el uso del modelo integrador descrito en la figura 12 se obtiene el log de eventos en formato XES[8], el cual utilizó como dato de entrada para este caso de estudio. En la figura 14 se puede apreciar un pequeño extracto del log.

Este log de eventos utiliza cinco extensiones: "Organizational", "Time", "Concept", "Lifecycle" y "OrganizationalData", donde las cuatro primeras pertenecen al estándar de XES, mientras que la última ha sido definida en el marco del proyecto CSIC y es definida en el capítulo 3

La extensión "Organizational" suele ser utilizada en dominios donde los eventos pueden ser causados por seres humanos, los cuales son parte de una organización. En este caso dicha extensión se utiliza para identificar el actor que causó el evento y su posición en la organización. Por otro lado, la extensión "Time" es utilizada para registrar la fecha y el tiempo exacto cuando ocurre un evento o traza, lo cual es conocido como "timestamp", y se registra en formato UTC. Luego, la extensión "Concept" es utilizada para hacer referencia al nombre por el cual se conoce el evento o traza. Mientras que la extensión "Lifecycle" es utilizada para especificar la transición del ciclo de vida de la actividad representada por el evento. Por último, como se vio anteriormente, la extensión "OrganizationalData" es utilizada para integrar los datos del proceso con los organizacionales.

Este log de eventos de gran tamaño contiene 100 trazas correspondientes a 100 instancias de ejecución del BP "Student Mobility". Además el log contiene un total de 22412 eventos, lo cual indica que cada instancia ejecuta un promedio de 224 eventos.

Cada traza contiene, además de un identificador y un nombre, todos los eventos que se ejecutaron en esa instancia correspondiente a la traza. Entre los atributos de un evento encontramos por un lado algunos definidos por las tres primeras extensiones mencionadas anteriormente, los cuales son:

- “concept:name”: Contiene el nombre del evento.
- “lifecycle:transition”: Contiene la transición del ciclo de vida del evento.
- “time:timestamp”: Contiene el timestamp correspondiente al momento de ejecución del evento.
- “org:role”: Contiene el rol del actor que ejecutó el evento, dentro de la estructura organizacional.
- “org:resource”: Contiene el nombre, o identificador, del actor que ejecutó el evento.

Por otro lado se encuentran los atributos correspondiente a la nueva extensión “OrganizationalData”. Estos atributos son:

- “orgdata:elemType”: Contiene el tipo de elemento asociado al evento, de forma de indicar si es una tarea de usuario, de servicio, mensaje, etc.
- “orgdata:varlist”: Contiene la lista de variables definidas a nivel de proceso en el evento.
- “orgdata:entlist”: Contiene la lista de entidades y atributos de la base de datos que fueron impactados en la ejecución del evento.

6.1.2. Preparación del Log

Luego de realizar una ejecución del BPODQM tomando como entrada el log de eventos descrito anteriormente, se obtuvieron resultados muy buenos, por lo que no se detectaron problemas de calidad considerables. Entonces, para poder realizar este caso de estudio se tuvo que realizar modificaciones en el log de entrada, para así poder obtener resultados útiles para su posterior análisis. A continuación describiremos estas modificaciones.

Para obtener resultados del factor “Density” de la dimensión “Completeness” se realizaron las siguientes modificaciones. Por un lado, para provocar que las métricas “Inexistent value” y “Weighted density” arrojen resultados negativos, se borró el atributo “id” a 30 trazas, y el atributo “concept:name” a otras 30 trazas. Luego en 401 eventos se borró el atributo “org:role” y en 100 se borró el atributo “orgdata:date” de algunas entidades de la base de datos organizacional. Por otro lado, en las siguientes 40 trazas, se le asignó el valor null al atributo “concept:name” de las primeras 20 trazas y al atributo “id” de las 20 trazas restantes. Luego se le asignó el valor null al atributo “org:role” y “orgdata:date” de 400 y 600 eventos respectivamente. Estas modificaciones impactan en los resultados de las métricas “Not null” y “Weighted density”.

Luego, para obtener resultados del factor “Anonimity” de la dimensión “Security” se modificó el valor del atributo “idProgram” perteneciente a la entidad “Application” de la base de datos organizacional en 75 oportunidades. Este cambio impacta a la métrica “Anonymous attribute” y consistió en agregarle dígitos al valor del atributo para que el mismo contenga más de tres dígitos. Este cambio se debe a que previamente decidimos que el atributo “idProgram” debe contener números de tres dígitos, lo cual se va a imponer con una expresión regular como parámetro de entrada.

Se duplicaron 240 eventos para poder obtener resultados del factor “Duplication Free”, esto impacta a la métrica “Duplicate Event”. Por ultimo se duplicaron 3064 atributos organizacionales, esto impacta a la métrica “Duplicate Attribute”. Estas modificaciones provocan que la cantidad de eventos en el log incremente de 22412 a 22652 y la cantidad de atributos de eventos en el log de 278049 a 281113.

Para el resto de las métricas no ha sido necesario realizar modificaciones en el log, ya que muchas de ellas dependen de parámetros de entrada. La configuración de los parámetros de entrada la veremos en la siguiente subsección.

6.1.3. Configuración de parámetros de entrada

Como se ha mencionado anteriormente, varias de las métricas implementadas necesitan de parámetros de entrada para poder ser ejecutadas. Esto se debe a que algunos aspectos de calidad a medir dependen de factores externos, como el dominio de donde provienen los datos, las necesidades de los usuarios, criticidad de algunos datos, etc. Por este motivo se definió una configuración de parámetros para tener resultados que puedan ser considerados para el análisis, lo cual se describe a continuación.

Empezando con la métrica “Extensional values” del factor “Domain Consistency”, vemos que la misma necesita recibir como parámetro un dominio definido por extensión, es decir una lista de los posibles valores de un atributo separados por coma. En este caso se decidió medir el atributo “org:resource” pasándole como dominio la lista de valores [Carlos, Jorge, Elena, Andrea]. Esto quiere decir que para cada atributo “org:resource” de un evento, su valor debe estar contenido en la lista de entrada.

Siguiendo con el factor “Density”, sus tres métricas “Not null”, “Inexistent value” y “Weighted density” necesitan parámetros de entrada. Para las dos primeras es necesario indicar a qué atributos se quiere medir la calidad, mientras que para la última se necesita además el peso de cada atributo de los elegidos anteriormente. Entonces, para coincidir con las modificaciones que se realizaron en la subsección anterior sobre el log de eventos, se eligió los atributos “id” y “concept:name” para las trazas, mientras que para los eventos se eligió “org:role” y “orgdata:date”. Con respecto a los pesos, para los atributos de traza se le asignó 70 % al atributo “id” y 30 % al atributo “concept:name”. Estos pesos se deben a que se consideró que es mas importante la presencia del atributo “id”, ya que sería crítico que una traza no cuente con el mismo. Mientras que para los atributos de eventos, se le asignó igual peso a cada atributo, es decir 50 % al atributo “org:role” y 50 % al atributo “orgdata:date”, ya que se consideró que ambos tienen la misma importancia.

Luego tenemos la métrica “Coverage Ratio” del factor “Coverage”, la cual necesita como entrada un umbral que indica el menor porcentaje de eventos aceptable para una traza, tomando este porcentaje sobre el total de eventos en el log. Dado que el log contiene 100 trazas se decidió que sería razonable que el umbral sea algo cercano al 1 %, por lo cual dejando un margen se ingresó un 0,9 %, donde se tuvo en cuenta que no todas las trazas contienen la misma cantidad de eventos y por lo tanto van a haber trazas que estén por encima del umbral y otras por debajo.

Continuando con los parámetros nos encontramos con la métrica "Attribute Timeliness" del factor "Timeliness". Esta métrica necesita un parámetro que indique la tolerancia mínima en: la diferencia de tiempo entre el timestamp de un atributo de la base organizacional con el timestamp de su correspondiente variable en el proceso. El valor ingresado para este input fue 1000 milisegundos.

Luego se tiene el atributo "Responsibility" del factor "Provenance", el cual necesita como entrada una lista de eventos y una lista negra de participantes, los cuales no pueden ejecutar ninguno de los eventos de la primer lista. Para esto se ingresó la lista de eventos [Register application, Approve scholarship results] y la lista de participantes [Luis, Edinson, Raul], la cual es un subconjunto del total de los participantes que se encuentran en el log. Esto significa que se considera como inválido cada evento con nombre "Register application" o "Approve scholarship results" y que tenga a "Luis", "Edinson" o "Raul" como valor del atributo "org:resource".

Por último contamos con la métrica "Anonymous attribute" del factor "Anonymity", la cual necesita recibir un atributo a ser medido y una expresión regular que determine el formato que debe cumplir el valor de este atributo. Dado que en algunos eventos se modificó el atributo "idProgram" de la base de datos organizacional para que tengan mas de cuatro dígitos, se decidió que el formato que debe seguir el valor de este atributo consiste en cifras de tres dígitos. Por lo tanto el parámetro de entrada será la expresión regular "`^[0-9]{3}$`".

6.2. Análisis de los resultados obtenidos

En ésta sección se analizan los resultados que fueron obtenidos luego de la ejecución del plug-in desarrollado tomando como entrada el caso de estudio presentado. En la tabla 27 se presentan las métricas a analizar con el score obtenido para cada una de ellas. A continuación una breve justificación de cada uno de dichos valores.

Tabla 27: Métricas analizadas y scores correspondientes

Dimensión	Factor	Métrica	Score
Accuracy	Syntactic accuracy	Format	1.00
	Precision	Timestamp precision	1.00
Consistency	Domain Consistency	Extensional Values	0.47
Completeness	Density	Not Null Trace Element(concept:name)	0.80
		Not Null Trace Element(id)	0.80
		Weighted Density Not Null In Traces	0.80
		Not Null Event Element(org:role)	0.98
		Not Null Event Element(orgdata:date)	0.94
		Weighted Density Not Null In Events	0.96
		Inexistent Event Element(org:role)	0.98
		Inexistent Event Element(orgdata:date)	0.55
		Weighted Density Inexistent In Events	0.77
		Inexistent Trace Element(concept:name)	0.70
		Inexistent Trace Element(id)	0.70
		Weighted Density Inexistent In Traces	0.70
	Coverage	Coverage Ratio	0.81
Uniqueness	Duplication-free	Duplicate Event	0.99
		Duplicate Attribute	0.99
	Contradiction-Free	Contradictory Event	0.58
Freshness	Timeliness	Attribute Timeliness	0.98
Credibility	Provenance	Responsibility	0.61
Security	Anonymity	Anonymous Attribute	0.99

Format

Para la evaluación de ésta métrica, se decidió probar modificar el formato a atributos "timestamp". La herramienta ProM al importar el log, como existían atributos los cuales no cumplían con el formato de timestamp, mostraba un error y no permitía seguir la ejecución.

Esto hizo que la medición de la métrica no se pueda llevar a cabo como se esperaba, y el score obtenido es 1,00.

Timestamp precision

Para la evaluación de ésta métrica, se decidió probar sacándole precisión a atributos "timestamp". Sacarle precisión fue eliminando el valor que tenia para los milisegundos por ejemplo.

La herramienta ProM al importar el log, a los atributos que se les saco precisión, les agregaba con 0 las cifras faltantes, logrando que cada atributo modificado vuelva a tener milisegundos.

Esto hizo que la medición de la métrica no se pueda llevar a cabo como se esperaba, y el score obtenido es 1,00.

Extensional Values(org:resource)

En ésta métrica como se menciono anteriormente se ingresó una lista de valores posibles para el atributo "org:resource". Se obtuvo un total de 11950 org:resource donde el valor no estuvo contenido dentro de esos posibles valores ingresados.
Los valores ingresados que no se encontraban en la lista [Carlos, Jorge, Elena, Andrea] con sus ocurrencias fueron los siguientes:

(null - 5, Luis - 3522, Edinson - 3062, admin - 68, Adriana - 390, Alberto - 264, Alejandro - 392, Daniel - 480, Flavia - 332, Gonzalo - 362, Juan - 292, Julio - 554, Laura - 370, Libertad - 541, Maria - 326, Raul - 374, Roberto - 282, Stefania - 334).

De esta forma el score obtenido de la métrica es: 0,47.

Not Null Trace Element(concept:name)

Para la evaluación de ésta métrica como se mencionó anteriormente se modificó el log de las ultimas 40 trazas. Dentro de las las 40 trazas, las primeras 20 se les modificó el valor del atributo "concept:name", por tanto 20 trazas de 100 tienen un valor null en el atributo a medir.

El score obtenido para ésta métrica es 0,80.

Not Null Trace Element(id)

Para la evaluación de ésta métrica como se mencionó anteriormente se modificó el log de las ultimas 40 trazas. Dentro de las las 40 trazas, de esas 40 trazas, las ultimas 20 se modifico el atributo "id", por tanto 20 trazas de 100 tienen un valor null en el atributo a medir.

El score obtenido para ésta métrica es 0,80.

Weighted Density Not Null Element In Traces(concept:name=30,id=70)

Para la evaluación de ésta métrica como se mencionó anteriormente se modificó el valor a "null" al atributo "id" en 20 trazas del log, de un total de 100 trazas, y tambien se modificó el valor al atributo "concept:name" en 20 trazas del log.

Las trazas que cumplieron con tener el atributo "concept:name" son 80, y el input ingresado por el usuario para este atributo es 30, Por lo tanto se realiza 80 * 30. A esta cuenta se la divide entre 100, ya que el score que se maneja en el trabajo se

encuentra en el rango [0,1].

Las trazas que cumplieron con tener el atributo “id” son 80, y el input ingresado por el usuario para este atributo es 70, Por lo tanto se realiza 80 * 70. A esta cuenta también se la debe dividir entre 100.

Por tanto para calcular el score de la métrica se realiza una sumatoria de ambas cuentas presentadas anteriormente. En este caso el calculo del score es:

(((80 * 30)/100) + ((80 * 70)/100)) = 0,80.

Not Null Event Element(org:role)

Para la evaluación de ésta métrica como se mencionó anteriormente se modificó el valor al atributo “org:role” en 400 oportunidades, de un total de 22251 oportunidades que el atributo se encuentra en el log.

La cantidad de eventos que se modificó del total se corresponde aproximadamente al 1,77 % del total de eventos del log, por tanto aproximadamente un 98 % de eventos cumplieron con la condición de la métrica. El score obtenido de la métrica es 0,98.

Not Null Event Element(orgdata:date)

Para la evaluación de ésta métrica como se mencionó anteriormente se modifico el valor al atributo “orgdata:date” en 600 oportunidades, de un total de 10026 oportunidades que el atributo se encuentra en el log.

La cantidad de eventos que se modifico del total se corresponde aproximadamente al 6 % del total de eventos del log, por tanto aproximadamente un 94 % de eventos cumplieron con la condición de la métrica.

El score obtenido de la métrica es 0,94.

Weighted Density Not Null Element In Events(org:role=50,orgdata:date=50)

Para la evaluación de ésta métrica como se mencionó anteriormente se modificó el valor a “null” al atributo “org:role” en 400 oportunidades, de un total de 22251 oportunidades del atributo en el log, y tambien se modificó el valor al atributo “orgdata:date” en 600 oportunidades, de un total de 10026 oportunidades del atributo en el log.

La cantidades de oportunidades que cumplieron con tener el atributo “org:role” son (1-(400/22251)), y el input ingresado por el usuario para este atributo es 50, Por lo tanto se realiza (1-(400/22251)) * 50. A esta cuenta se la divide entre 100, ya que

el score que se maneja en el trabajo se encuentra en el rango [0,1].

La cantidades de oportunidades que cumplieron con tener el atributo “orgdata:date” son (1-(600/10026)), y el input ingresado por el usuario para este atributo es 50, Por lo tanto se realiza (1-(600/10026)) * 50. A esta cuenta también se la debe dividir entre 100.

Por tanto para calcular el score de la métrica se realiza una sumatoria de ambas cuentas presentadas anteriormente. En este caso el calculo del score es:

((((1-(400/22251)) * 50)/100) + (((1-(600/10026)) * 50)/100)) = 0,96.

Inexistent Event Element(org:role)

Para la evaluación de ésta métrica como se mencionó anteriormente se elimino al atributo “org:role” en 401 eventos del log, de un total de 22652 eventos.

La cantidad de eventos que se modificó del total se corresponde aproximadamente al 1,77 % del total de eventos del log, por tanto aproximadamente un 98 % de eventos cumplieron con la condición de la métrica. El score obtenido de la métrica es 0,98.

Inexistent Event Element(orgdata:date)

Para la evaluación de ésta métrica como se mencionó anteriormente se eliminó al atributo “orgdata:date” en 100 eventos del log.

En este caso se puede apreciar que la cantidad de eventos que no poseen este atributo es 10202, un número bastante superior a los 100 que fueron eliminados en las modificaciones realizadas.

Siendo 22652 eventos pertenecientes al log, la cantidad de eventos que no tienen el atributo del total se corresponde aproximadamente al 45 % del total de eventos del log, por tanto aproximadamente un 55 % de eventos cumplieron con la condición de la métrica. El score obtenido de la métrica es 0,55.

Weighted Density Inexistent Element In Events(org:role=50,orgdata:date=50)

Para la evaluación de ésta métrica como se mencionó anteriormente no se encontró al atributo “org:role” en 401 eventos del log, y no se encontró el atributo “orgdata:date” en 10202 eventos del log.

Los eventos que cumplieron con tener el atributo “org:role” son (1-(401/22652)), y el input ingresado por el usuario para este atributo es 50, Por lo tanto se realiza (1-(401/22652))) * 50. A esta cuenta se la divide entre 100, ya que

el score que se maneja en el trabajo se encuentra en el rango [0,1].

Los eventos que cumplieron con tener el atributo "orgdata:date" son (1-(10202/22652)), y el input ingresado por el usuario para este atributo es 50, Por lo tanto se realiza (1-(10202/22652)) * 50. A esta cuenta también se la debe dividir entre 100.

Por tanto para calcular el score de la métrica se realiza una sumatoria de ambas cuentas presentadas anteriormente. En este caso el calculo del score es:

((((1-(401/22652)) * 50)/100) + (((1-(10202/22652)) * 50)/100)) = 0,77.

Inexistent Trace Element(concept:name)

Para la evaluación de ésta métrica como se mencionó anteriormente se eliminó al atributo "concept:name" en 30 trazas del log, de un total de 100 trazas.

La cantidad de trazas que se modificó del total se corresponde al 30 % del total de trazas del log, por tanto la cantidad de trazas que cumplieron con la condición de la métrica son el 70 %. El score obtenido de la métrica es 0,70.

Inexistent Trace Element(id)

Para la evaluación de ésta métrica como se mencionó anteriormente se elimino al atributo "id" en 30 trazas del log, de un total de 100 trazas.

La cantidad de trazas que se modifico del total se corresponde al 30 % del total de trazas del log, por tanto la cantidad de trazas que cumplieron con la condición de la métrica son el 70 %. El score obtenido de la métrica es 0,70.

Weighted Density Inexistent Element In Traces(concept:name=30,id=70)

Para la evaluación de ésta métrica como se mencionó anteriormente se eliminó al atributo "id" en 30 trazas del log, de un total de 100 trazas, y no se pudo eliminar el atributo "concept:name" para ninguna traza.

Las trazas que cumplieron con tener el atributo "concept:name" son 70, y el input ingresado por el usuario para este atributo es 30, Por lo tanto se realiza 70 * 30. A esta cuenta se la divide entre 100, ya que el score que se maneja en el trabajo se encuentra en el rango [0,1].

Las trazas que cumplieron con tener el atributo "id" son 70, y el input ingresado por el usuario para este atributo es 70, Por lo tanto se realiza 70 * 70. A esta cuenta también se la debe dividir entre 100.

Por tanto para calcular el score de la métrica se realiza una sumatoria de ambas cuentas presentadas anteriormente. En este caso el calculo del score es:

(((70 * 30)/100) + ((70 * 70)/100)) = 0,70.

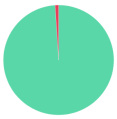
Coverage Ratio

Para la evaluación de ésta métrica como se mencionó anteriormente se ingresó el valor 0,9 %, dejando cierto margen a las trazas que sean muy cercanas a 1 % por debajo. Donde las trazas que no cumplieron con la condición su id es:

(1027474, 1039729, 1039728, 1292576, 1115135, 1127711, 1157883, 1256932, 1260627, 1267277, 1269387, 1277566, 1290492, 1298787. Además 5, tenían id null).

Un total de 19 trazas que no cumplieron la condición, de un total de 100 trazas. Como es de esperarse, el score obtenido de la métrica es: 0,81.

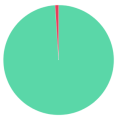
Duplicate Event



Para la evaluación de ésta métrica como se mencionó anteriormente se duplicaron 240 eventos en el log. Teniendo en cuenta que el total de eventos del log es 22652, la cantidad de eventos duplicados es aproximadamente un 1 % del total.

Por tanto el score obtenido de la metrica es: 0,99.

Duplicate Attribute



Para la evaluación de ésta métrica como se mencionó anteriormente se duplicaron 3064 atributos de eventos en el log. Teniendo en cuenta que el total de atributos de eventos del log es 281113, la cantidad de eventos duplicados es aproximadamente un 1 % del total.

Por tanto el score obtenido de la metrica es: 0,99.

Contradictory Event

Para la evaluación de ésta métrica se tomó como "key" a los valores de los atributos ["concept:name","lifecycle:transition","org:resource", "org:varlist"]. La cantidad de eventos que compartían el valor de estos atributos, pero diferían en el valor del resto 9574 eventos de un total de 22652.

La cantidad de eventos contradictorios es aproximadamente el 42 % del total de eventos del log, por tanto aproximadamente un 58 % de eventos no son contradictorios.

El score obtenido para ésta métrica es 0,58.

Attribute Timeliness

Para la evaluación de ésta métrica como se mencionó anteriormente se ingresó como input 1000 milisegundos. Se obtuvieron 693 atributos los cuales superan ese tiempo, de un total de 34432 atributos. La proporción de atributos que superan el tiempo es aproximadamente 2 % del total.

Por tanto el score obtenido de la métrica es 0,98.

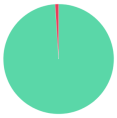
Responsibility (org:resource)

Para la evaluación de ésta métrica como se mencionó anteriormente se ingresó como input la lista de eventos [Register application, Approve scholarship results] y la lista de participantes que no deben participar en alguno de estos eventos: [Luis, Edinson, Raul].

De un total de 12401 eventos que tienen el atributo "concept:name" con valor "Register application" o "Approve scholarship results", en 4856 los participantes son alguno de la lista de participantes ingresada. Esto es un 39 % aproximadamente.

Por tanto el score obtenido de la métrica es: 0,61.

Anonymous Attribute(orgdata:idprogram)



Para la evaluación de ésta métrica como se mencionó anteriormente se modificó el valor al atributo "idprogram" en 75 oportunidades, de un total de 10162 apariciones del atributo. La proporción de instancias del atributos modificadas sobre el total se aproxima al 1 %.

Por tanto el score obtenido de la métrica es: 0,99.

6.3. Conclusiones del Caso de Estudio

Al finalizar el análisis del caso de estudio planteado se pueden obtener las siguientes conclusiones:

- ProM dificultó la realización de dos métricas: "Format" y "Timestamp precision". Con la primera ProM mostraba un error a la hora de cargar el log, impidiendo ejecutarlo, debido a que no permitía eliminar un atributo de tipo "timestamp". La segunda ProM se encargaba de completar los milisegundos que se le borraban a atributos "timestamp". Por tanto en ambas métricas obtuvimos un score 1,0.
- El log extendido de eventos que se usó, se modificó sin ningún criterio en particular, simplemente teniendo como objetivo comprobar que las métricas implementadas cumplieran con lo esperado.
- Cabe destacar que los resultados obtenidos para las métricas que necesitan inputs de usuario, dependen fuertemente de lo ingresado en los mismos. Por lo tanto para las mismas se realizo un análisis del log y se ingresaron datos que tuvieran una cantidad relevante de errores.
- Para el caso particular de la métrica "Contradictory Event" no se obtuvo un score alto, esto se debe a que depende exclusivamente de la key seleccionada mediante los input ingresados por el usuario al ejecutar el plug-in.
- Para el caso particular de la métrica "Responsibility" una posible mejora sería solicitar con un input los eventos que se desean analizar, y una lista específica de usuarios que no puedan participar para cada uno de los eventos. Actualmente se solicita una única lista de participantes que no puede participar en ninguno de los eventos que se ingresan como input.
- La decisión de como manejar los atributos organizacionales se puede mejorar. Actualmente se almacenan para luego ser utilizados de la forma "orgdata:atributo_organizacional", una posible mejora es almacenarlos de la forma "orgdata:entidad:atributo_organizacional". Se evidencio con las metricas del tipo "Inexistent Value", debido a que al eliminar un atributo por ejemplo en un entidad de atributos organizacionales de un evento puede que si exista para otra entidad y la métrica no detecte la inexistencia.

7. Conclusiones y trabajos futuros

En ésta sección se presentan las conclusiones que se obtuvieron a partir del trabajo realizado, y posibles mejoras detectadas, interesantes a realizarse en una linea de trabajo a futuro.

7.1. Conclusiones

La calidad de los datos a lo largo del tiempo ha sido un gran problema, especialmente dentro de las organizaciones. Este trabajo ha presentado diferentes dimensiones/factores/métricas de la calidad de los datos, donde cada una de estos elementos han sido interpretado hacia los datos de log de eventos, por lo cual los convierte en un tema relevante para el área de calidad de datos en minería de procesos.

El objetivo principal de este proyecto era definir un modelo de calidad de datos integrados de procesos y organizacionales que permita analizar la calidad de los datos de un log de eventos que soporte datos de procesos y organizacionales integrados. Analizando la literatura relevante al tema (O1) existen distintas formas de definir la calidad de datos, en este trabajo se propuso un modelo jerárquico (O2) de forma de agrupar y medir diversos aspectos de calidad. En el log de eventos extendido se incluyen datos del proceso de negocio, así como datos organizacionales del mismo. Por tanto el análisis de los datos fue extensivo a los datos organizacionales, integrados en la extensión del log con la que se trabajo.

Actualmente el análisis de la calidad de log de eventos no es un área en la cual existan muchos trabajos de análisis, por tanto presenta un desafío. Incluso si se tiene en cuenta trabajos de análisis aplicados a la extensión del log de eventos que se utiliza en este trabajo no existen antecedentes, lo cual implica un desafío mayor.

En la definición de los elementos del modelo de calidad, se los planteo de forma genérica, donde las definiciones no fueran influenciadas por el contexto del log de eventos ingresado, permitiendo un mayor alcance de posibles interesados en utilizarlo. El estudio de la literatura señala que existen múltiples formas de analizar la calidad de datos, en el trabajo propuesto se fue por el camino de lograr un modelo genérico, donde se lograra integrar las distintas visiones de mediciones y sin orientarlas a un sector en particular.

Además de definir el modelo de calidad de datos, se lo llevo a la implementación en ProM (O3), para que el usuario pueda hacer uso del modelo definido y aprovechar de las dimensiones, factores y métricas definidas e implementadas, logrando que no deba realizar el análisis de las mismas de forma manual.

El plug-in para cada una de las métricas brinda un detallado de el análisis realizado, donde se incluye la descripción de la métrica, un desglose de los elementos del log que no cumplieron lo esperado, y una puntuación. Se llevo a cabo un caso de estudio (O4) con un log extendido real. Analizar y ser critico con el resultado obtenido para cada una de las métricas es una tarea muy importante para el usuario. El usuario debe ser capaz de determinar si en el contexto en el que se encuentra son relevantes

los errores o sugerencias que le pueda indicar la salida del plug-in.

Sin dudas que lograr que las métricas de calidad sean genéricas no es una tarea sencilla, y lleva un gran análisis del usuario objetivo al que se apunta, que es lo mas genérico posible. Cada log de eventos es diferente, pero esto no debe ser una barrera para que el modelo de calidad de datos y el plug-in en este caso puedan ser aplicados.

La interpretación del usuario y el conocimiento del dominio son dos factores principales y muy relevantes parar lograr una evaluación certera de la calidad de los datos. La forma que se encontró para que el usuario pueda acortar la brecha entre un modelo de calidad genérico y un log de eventos especifico aplicado a un contexto particular es permitirle ingresar datos que serán tenidos en cuenta a la hora de realizar el análisis.

7.2. Trabajos Futuros

Se podría extender el modelo en nuevas dimensiones/factores/métricas. A nivel del modelo no se necesita una gran curva de aprendizaje para poder lograrlo. Esto se debe al modelo jerárquico el cual permite enlazar de forma sencilla un nuevo elemento. A nivel del plug-in la estructura del mismo fue realizada de tal manera que sea extensible, de forma que agregar un elemento no sería de gran impacto en cuanto a tiempo de configuración.

Algunas de las métricas genéricas definidas, se las dejó sin al menos una métrica especifica, este trabajo es un gran desafío el cual le daría un mayor valor agregado. Además, diversas métricas planteadas en el modelo de calidad no se llegaron a implementar en el plug-in por una cuestión de tiempos, que las mismas sean implementadas enriquecerían aún mas el trabajo realizado.

En la definición de métricas especificas de calidad para una métrica genérica, se definió en todos los casos a lo sumo una única métrica especifica, y esa métrica fue la que se tuvo en cuenta a la hora de la implementación. Sabido es que existen diferentes formas de especificar y medir un aspecto de calidad. Como trabajo a futuro se puede plantear para el caso que se crea conveniente definir mas de una métrica especifica, para que el usuario tenga la libertad de optar por la cual crea que se adecua mas a su contexto de análisis.

La exportación de los datos obtenidos luego de la ejecución se puede dar en otro formato, el cual pueda ser utilizado como parámetro de entrada para ejecutar otro plug-in en ProM o para realizar algún tipo de trabajo en un ambiente externo.

Luego por último, mejorar la interfaz de usuario la cual se le ofrece al usuario, tanto para el ingreso de parámetros o la presentación de los resultados obtenidos siempre es una tarea que se puede mejorar a gusto.

Referencias

[1] A. Delgado, A. Marotta, L. González, L. Tansini, and D. Calegari, "Towards a data science framework integrating process and data mining for organizational improvement," in *15th Int. Conf. on Software Technologies, ICSOFT 2020*. ScitePress, 2020.

[2] A. Delgado and D. Calegari, "Towards a unified vision of business process and organizational data," in *XLVI Latin American Computing Conference, CLEI 2020*. IEEE, 2020, p. To appear.

[3] O. Source, "Prom - framework for process mining," 2006.

[4] A. Paiva, A. Cavalli, P. Martins, and R. Pérez-Castillo, *Quality of Information and Communications Technology: 14th International Conference, QUATIC 2021, Algarve, Portugal, September 8–11, 2021, Proceedings*, ser. Communications in Computer and Information Science. Springer International Publishing, 2021.

[5] M. Weske, *BPM - Concepts, Languages, Architectures, 3rd Ed.* Springer, 2019.

[6] R. Dijkman and P. Van Gorp, "Bpmn 2.0 execution semantics formalized as graph rewrite rules," in *Business Process Modeling Notation*, J. Mendling, M. Weidlich, and M. Weske, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 16–30.

[7] W. M. P. van der Aalst, *Process Mining - Data Science in Action, 2nd Edition*. Springer, 2016.

[8] C. W. Günther and E. Verbeek, "Ieee standard for extensible event stream (xes) for achieving interoperability in event logs and event streams," *IEEE Std 1849-2016*, pp. 1–50, 2016.

[9] R. Verhulst, "Evaluating quality of event data within event logs:an extensible framework," Master's thesis, Eindhoven University of Technology, 2016.

[10] AGESIC, "Framework para la gestión de la calidad de datos en gobierno digital," in *Framework para la Gestión de la Calidad de Datos en Gobierno Digital*, 2019.

[11] F. de Ingenieria UDELAR, "Curso: Calidad de datos," 2020.

[12] *ISO/IEC 25012:2008. Software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Data quality model*, 2008.

[13] R. Bose, R. Mans, and W. Aalst, "Wanna Improve Process Mining Results? It's High Time We Consider Data Quality Issues Seriously," in *IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2013)*, B. Hammer, Z. Zhou, L. Wang, and N. Chawla, Eds. Singapore: IEEE, 2013.

[14] W. Aalst, "Extracting event data from databases to unleash process mining," in *BPM: Driving Innovation in a Digital World*, J. vom Brocke and T. Schmiedel, Eds. Springer-Verlag, Berlin, 2015.

[15] A. Delgado, D. Calegari, A. Artus, and A. Borges, "Integration of business process data and organizational data for evidence-based business intelligence," *CLEI electronic journal*, vol. 24, no. 2, 2021.

[16] C. Batini and M. Scannapieco, *Data and Information Quality - Dimensions, Principles and Techniques*, ser. Data-Centric Systems and Applications. Springer, 2016.

[17] W. Aalst, F. Burstein, and C. Holsapple, *Decision Support Based on Process Mining*, 01 2008, vol. 4295, pp. 637–657.

[18] E. Verbeek, J. C. A. M. Buijs, B. F. van Dongen, and W. M. van der Aalst, "Prom 6: The process mining toolkit," *European Journal of Operational Research*, 2010.

[19] C. Günther and E. Verbee, *OpenXES Developer Guide*, ser. University of Technology Eindhoven, 2014.