A Reference Data Model to Specify Event Logs for Big Data Pipeline Discovery

Dario Benvenuti[1], Andrea Marrella[1], Jacopo Rossi[1],
Nikolay Nikolov[2], Dumitru Roman[2], Ahmet Soylu[3], and Fernando Perales[4]

[1] *Sapienza Università di Roma, Rome, Italy*
{d.benvenuti,marrella,j.rossi}@diag.uniroma1.it
[2] *SINTEF AS, Oslo, Norway*
{nikolay.nikolov,dumitru.roman}@sintef.no
[3] *Oslo Metropolitan University, Oslo, Norway*
ahmetsoy@oslomet.no
[4] *JOT Internet Media, Madrid, Spain*
fernando.perales@jot-im.com

Abstract. State-of-the-art approaches for managing Big Data pipelines assume their anatomy is known by design and expressed through ad-hoc Domain-Specific Languages (DSLs), with insufficient knowledge of the dark data involved in the pipeline execution. Dark data is data that organizations acquire during regular business activities but is not used to derive insights or for decision-making. The recent literature on Big Data processing agrees that a new breed of Big Data pipeline discovery (BDPD) solutions can mitigate this issue by solely analyzing the event log that keeps track of pipeline executions over time. Relying on well-established process mining techniques, BDPD can reveal fact-based insights into how data pipelines transpire and access dark data. However, to date, a standard format to specify the concept of Big Data pipeline execution in an event log does not exist, making it challenging to apply process mining to achieve the BDPD task. To address this issue, in this paper we formalize a universally applicable reference data model to conceptualize the core properties and attributes of a data pipeline execution. We provide an implementation of the model as an extension to the XES interchange standard for event logs, demonstrate its practical applicability in a use case involving a data pipeline for managing digital marketing campaigns, and evaluate its effectiveness in uncovering dark data manipulated during several pipeline executions.

Keywords: Big Data Pipeline Discovery (BDPD) · Big Data Pipeline · Reference Data Model · Process Mining · Event Log · Dark Data · XES

1 Introduction

In the current era of Big Data and Internet-of-Things (IoT), we are witnessing the transformation of traditional working domains into new challenging cyber-physical environments (e.g., smart manufacturing) characterized by the availability of a large variety of sensors that monitor the evolution of several real-world

Pre-print copy of the manuscript published by Springer
(available at: https://link.springer.com/chapter/10.1007/978-3-031-41623-1_3)
identified by doi: 10.1007/978-3-031-41623-1_3

objects of interest and produce a considerable amount of data. Nonetheless, many data are stored for compliance purposes only but not turned into value, thus becoming *dark data*. Gartner defines dark data as the information assets organizations collect, process and store during regular business activities, but generally fail to use for other purposes (e.g., analytics, business relationships and direct monetizing).[5] Examples range from server log files, which can give clues related to the workers' habits while executing their tasks, to geolocation data that could reveal traffic patterns. Nowadays, storing and securing dark data usually entails more expenses and risks than the potential return profit [10,30].

The recent literature on Big Data processing agrees that discovering and interpreting the *Big Data pipelines* that run within the organization workflow is essential to valorise dark data for insights and decision-making [8,25]. Big data pipelines are composite steps for processing data with non-trivial properties, referred to as the Vs of Big Data (e.g., volume, velocity, etc.) [22]. They: *(i)* ingest raw data from disparate sources; *(ii)* process such data in the computing continuum, which offers on-demand resource provisioning through a fluid ecosystem integrating Cloud, Fog, and Edge technologies; and *(iii)* move it toward the data consumers, which undertake further transformations, visualizations, etc.

State-of-the-art approaches for managing Big Data pipelines work assuming their anatomy is known by design and expressed using one of the many available domain-specific languages (DSLs) [19].

To tackle this issue, in the context of the recently funded EU H2020 DataCloud project[6], one of the main targets is to realize a new breed of *Big Data pipeline discovery* (BDPD) solutions to infer the structure and behavior of a data pipeline by solely analyzing the *event log* that keeps track of its past executions. Relying on the similarity among the concepts of "data pipeline" and "business process", one of the project's vision is to leverage and customize well-established *process mining* techniques to reveal fact-based insights into how data pipelines transpire and access dark data [4]. However, traditional event logs used for process mining are limited in scope [2]. They include attributes tailored to recording sequence-flow details of process execution (e.g., timestamp and completion of activities, etc.), thus neglecting any data- and technological-related aspects needed to perform BDPD. In addition, to date, a standard format to specify the concept of Big Data pipeline execution in an event log does not exist, making it challenging to apply process mining techniques to achieve the BDPD task. This leads to the following research questions:

- **RQ1**: Which attributes are required in an event log to keep track of data pipeline executions and properly perform BDPD?
- **RQ2**: Which process mining techniques can be exploited to uncover and valorize dark data manipulated during data pipeline executions?
- **RQ3**: Does process mining provide an effective way to perform BDPD and uncover dark data in real-world data pipeline executions?

[5] `https://www.gartner.com/en/information-technology/glossary/dark-data`
[6] `https://cordis.europa.eu/project/id/101016835`

In answering these questions, in this paper, we: (i) formalize a universally applicable reference data model to conceptualize the core properties of a data pipeline execution; (ii) implement the model as an extension to the XES[7] interchange standard for event logs; (iii) demonstrate its practical applicability in a use case involving a real-world data pipeline for managing digital marketing campaigns; and (iv) evaluate the effectiveness of (some) process mining techniques in uncovering relevant dark data manipulated during pipeline executions.

The rest of the paper is organized as follows. Section 2 describes our research methodology, based on design science principles. Section 3 presents the relevant background on event logs and data pipelines, together with a concrete use case. The reference data model, its underlying design concepts, and an accompanying interchange format, are presented in Section 4. Section 5 demonstrates the practical applicability of the reference model in the use case. Section 6 evaluates the effectiveness of applying process mining over the reference model to perform BPDB and uncover dark data. Finally, Section 7 draws conclusions, discusses the limitations of this work, and traces future work.

2 Research Methodology

Our research methodology is inspired to the Design Science approach described by Johannesson and Perjons in [12]. The methodology is applied in five distinct sequential phases: problem formulation and objectives, requirements definition, design and development, demonstration and evaluation.

Problem Formulation and Objectives. In this phase, which is addressed in Section 1, we first specify the research problem to be tackled, i.e., *realizing a BDPD solution to identify and take advantage of the dark data accessed during a data pipeline execution*. In Section 3, we justify its significance in the Big Data processing field trough a motivating use case. Then, we elaborate on three research questions, i.e., RQ1, RQ2 and RQ3, to guide our research toward defining an artefact to solve the problem. A *reference data model* to specify a data pipeline and its core properties, and its implementation as an extension to the XES standard for event logs, represent such an artefact, which opens the possibility of applying process mining techniques to perform BDPD.

Requirements Definition. The second phase consists of eliciting the requirements for the outlined artefact. In Section 3, after providing the required background concepts on data pipelines and event logs, we discuss the main findings of our previous work [19]. In [19], we analyzed the literature on Big Data pipeline modeling to extract *three requirements* that guided us to formalize a novel DSL (called DC-DSL) toward a standardized representation of the structure of a data pipeline. However, while pipeline modeling through DSLs represents, by nature, a "subjective" and static view of reality, BDPD is "instance-driven", i.e., it targets extracting concrete pipeline execution data from event logs. In this direction, we rely on the main concepts defined in DC-DSL and its requirements to build the

[7] `https://xes-standard.org/`

skeleton of our reference data model, and we augment it through the key process mining notions of "event" and "trace".

Design. Based on the analysis of the background and the requirements, in the third phase we make design decisions explicit, discussing the reference data model and describing its main features in Section 4. Moreover, we present in detail an implementation of the model as an extension to the XES interchange standard for event logs, which enables us to answer RQ1.

Demonstration. In the fourth phase, to answer RQ2, we demonstrate the practical applicability of the reference model in the use case. Specifically, we show in Section 5 how the targeted use of process mining techniques can support uncovering and understanding the dark data accessed during many executions of a real-world data pipeline for managing digital marketing campaigns.

Evaluation. Finally, to answer RQ3, in Section 6 we perform a preliminary evaluation involving 10 expert users from research institutions and companies engaged in Big Data pipeline management activities. The aim is to assess the effectiveness of applying process mining techniques over the reference model to untangle the relevant dark data manipulated by the use case data pipeline.

3 Background

3.1 Process Mining and Event Logs

Process mining [1] is a family of data analysis techniques that enable decision-makers to discover flowchart models from event data [5], compare expected and actual behaviours [7], and enhance models. It focuses on the real execution of processes, as reflected by the footprint of reality logged by the information systems (ISs) of an organization. The starting point is an *event log*, which is analysed to extract insights and recurrent patterns about how processes are executed. Event logs consist of *traces* that each correspond to one process instance. Each trace contains a sequence of *events* that occurred during the execution of the process instance. Events are related to a particular step in a process with an activity label, a timestamp, and a trace identifier.

To enable the exchange of event logs between different ISs, the process mining community has developed an interchange standard that defines the structure and general contents of event logs. Since 2016, the official IEEE standard for storing, exchanging and analysing event logs is XES (eXtensible Event Stream) [3]. In XES, event logs are organized in a three-level hierarchy of log, trace, and event objects, with a minimal set of explicitly defined attributes on each of the levels. The standard is designed to allow for additional attributes to extend its scope. Some relevant extensions to XES were proposed to support communications [13], privacy-preserving data transmission [24], and uncertain data management [21].

3.2 Big Data Pipelines

The concept of Big Data pipeline can be traced back to 2012 [23], where data pipelines are described as a "*mechanism to decompose complex analyses of large*

data sets into a series of simpler tasks". Over the years, many definitions of a data pipeline were provided. Among the most relevant, in [20], the authors refer to a data pipeline as the "*path through which Big Data is transmitted, stored, processed and analyzed*". In [18], a data pipeline is defined as "*a complex chain of interconnected activities from data generation through data reception, where the output of one activity becomes the input of the next one*".

While the literature lacks a rigorous specification of the concept of data pipeline, some common features that are related to it can be identified:

- A data pipeline consists of chains of processing elements that manipulate and interact with data sets;
- The outcome of a processing element of a data pipeline will be the input of the next element in the pipeline;
- Each processing element of a data pipeline interacts with data sets considered as "big", i.e., with at least one of the Vs dimensions that is verified to hold.

Looking at the above characteristics, it is evident that many similarities exist between the concepts of "data pipeline" and "business process". With the main difference that any step of a data pipeline is thought to manipulate some data. Conversely, processes include activities that do not necessarily interact with any kind of data [1]. Nonetheless, since BDPD resembles the discovery of processes, as both require an event log to enact the discovery task, it is worth employing process mining techniques to support the development of novel BDPD solutions. To achieve this objective, a reference model that formalizes the main properties of a data pipeline and an extension of the XES standard for event logs is required to capture the data and technological aspects related to a data pipeline execution.

3.3 Big Data Pipeline Specification trough DSLs

The literature on Big Data processing has proposed several ad-hoc DSLs for specifying the structure of a data pipeline in graphical format or as an XML file [19]. DSLs are specification languages targeted to describe a specific application domain. This is in contrast to a general-purpose language (GPL), which is broadly applicable across domains. Compared to GPLs, DSLs cannot cover all aspects of a given problem due to their limited scope. Still, they fill this gap with improved expressiveness, offering better domain-specificity and significantly improving collaboration between domain experts and developers [17].

In our previous work [19], we analyzed the literature on Big Data processing to categorize the existing DSLs for modeling data pipelines based on their expressiveness. We found that the majority of DSLs: (i) propose similar constructs having different semantics to specify a data pipeline; and (ii) are often characterized by an ambiguous semantics, which can hardly be formalized and does not enable the application of any reasoning technique. Driven by this analysis, in [19] we derived three requirements to build a novel DSL that integrates and formalizes the main concepts underlying the structure of a data pipeline:

1. The DSL must include a pipeline definition mechanism with a clear separation between design and run-time aspects and not limited to a specific technology stack, application domain or ad-hoc processing models;
2. The DSL must include a run-time support that considers pipelines as separate units, rather than a single unit, for individual pipeline steps;
3. The DSL must include an enactment approach with run-time driven execution and support for race-condition-free parallel branches.

The above requirements were realized through DC-DSL (DC stands for DataCloud), which enables different stakeholders to create Big Data pipelines exploiting containerization and orchestration technologies. These are required concepts to allow data pipeline execution on the resources available in the Computing Continuum. Since DC-DSL is *event log agnostic*, in Section 4, we show how we used it to build the skeleton of our reference data model, which - in contrast - will be aware of the key process mining concepts "event" and "trace". In addition, the reference model keeps track of many execution parameters used to monitor a data pipeline execution. They are typically recorded by ISs in different data sources and neglected by DC-DSL, thus becoming dark data.

3.4 Use Case

Let us consider the real-world case of a Big Data pipeline targeting higher mobile business revenues in smart marketing campaigns. This use case is offered by one of the small-medium enterprises involved in the H2020 DataCloud project. To discover the structure of the use case pipeline, we relied on the interview-driven methodology defined in our previous work [6], which allowed us to specify various simulation scenarios to frame the boundaries of all possible pipeline executions. Then, we generated a simulated event log in the traditional XES format using the Simio[8] tool, obtaining 10,000 execution traces (the log is available for testing at: `https://dx.doi.org/10.5281/zenodo.7387553`) compliant with the simulation scenarios. In Fig. 1, it is shown the Directly-Follow Graph (DFG) representing the pipeline structure, discovered by feeding a process discovery tool (we used Disco by Fluxicon[9]) with the

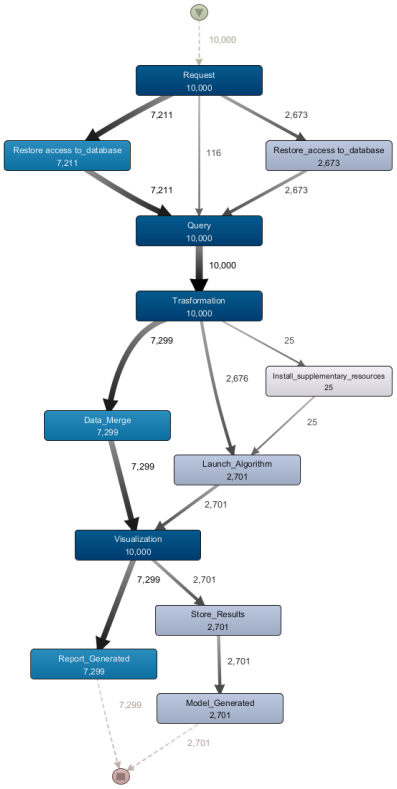

Fig. 1. DFG of the use case pipeline.

[8] `https://www.simio.com/`
[9] `https://fluxicon.com/disco/`

simulated event log. The pipeline is triggered when the system receives a request to model a new marketing campaign or to report on how an already existing one is performing. In both cases, the first two steps of the pipeline are querying the required data and applying specific transformations to it. If the request is for a report, there is a need to merge the queried data. On the other hand, for a model request, an algorithm to compute it is launched, and the results are stored in dedicated databases. Finally, the pipeline ends with either the report or the model being generated. By applying traditional process mining techniques, only sequence-flow details of the pipelines and related metrics can be obtained. This information can be used to grasp insights on the workflow running behind the data pipeline (e.g., how steps are sequenced, branching probabilities, potential bottlenecks, and other issues in the process flow). While useful, they do not allow us to infer further details from different perspectives, e.g., the flow of data accessed and manipulated during pipeline execution, the technologies used to process the Big Data in each pipeline step, etc. In a nutshell, *there are relevant execution data that could be easily captured by any logging system during pipeline execution, but are lost during the analysis, thus becoming dark data.* The first step to enable process mining techniques accessing and elaborating such data is to capture them in the event log, as shown in the next sections.

4 Reference Model and XES Extension for Event Logs

In this section, we present our reference data model to capture data pipeline executions by analyzing its UML class diagram, which is shown in Fig. 2, and explaining how it relates to the concept of event log. Then, Section 4.2 examines how to extend the XES standard to capture the properties defined in the model.

4.1 Reference Data Model

We start by looking at the class ***Big Data Pipeline***, which has an *ID*, a *Name*, and a *Communication Medium* (e.g., a message queue) on which data flows. Each Big Data Pipeline needs to have at least one ***Step*** by definition, and a Step belongs to only one pipeline. As can be seen from its attributes, a Step has an *ID*, a *Name*, and operates on a *Continuum Layer* (e.g., edge, fog or cloud) and has a *Type* depending on the computed data transformation (e.g., it can be a data consumer, a data producer, or both). Finally, a Step needs to have at least a ***Data Source***. A Data Source has an *ID*, a *Name*, a *Type*. The latter specifies if it is used as input, output, or both, and can be characterized by how it relates to the Vs of Big Data, e.g., by looking at its *Volume*. We specialized Data Source by highlighting ***Data Streams*** which are data sources with a certain *Velocity*, and we acknowledge that this class can be further specialized to include all the different Vs that can be appropriated to the context in which the model will be used. Finally, a Data Source needs to be used by at least one Step. A Step is made up by at least one ***Step Phase***, which is the core component of the reference model. A Step Phase belongs to only one Step and has an *ID*, a *Name*,

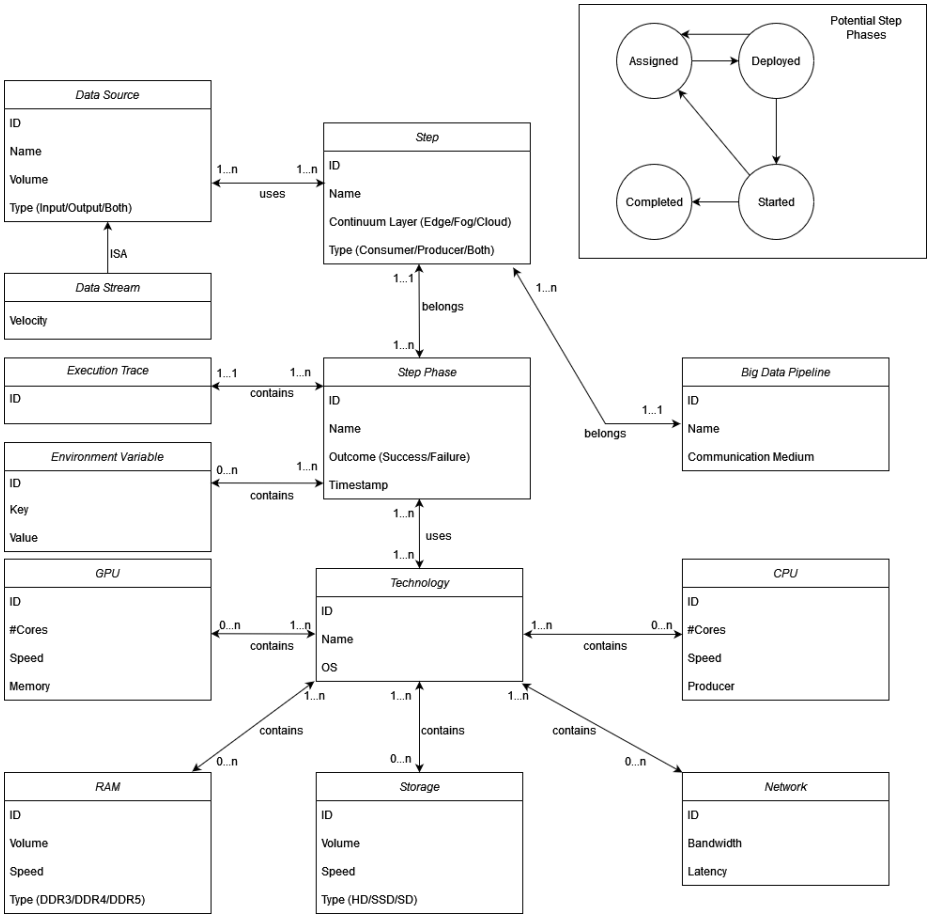

Fig. 2. The UML class diagram for the reference data model.

an *Outcome* (either success or failure), and a *Timestamp*. We highlight four potential step phases that are commonly used to describe the life cycle of the steps of a data pipeline, *Assigned*, *Deployed*, *Started* and *Completed*. Still, we acknowledge that this could vary depending on the context. A Step Phase taps at least from one ***Technology***, which has an *ID*, a *Name*, and an *Operating System* (OS). We consider of interest only Technologies used by at least one Step Phase. Detailed technological information can be expressed using the ***GPU***, ***CPU***, ***RAM***, ***Storage*** and ***Network*** classes, which contain an *ID* and a series of self-explanatory attributes related to the specific class. We consider of interest only CPUs, GPUs, RAMs, Storages, and Networks that are related to at least one Technology. Optionally, a Step Phase can be associated with ***Environment Variables***. They consist of simple pairs of *Key* and *Value* attributes, which can be used to describe the Step Phase domain-specific properties. An Environment

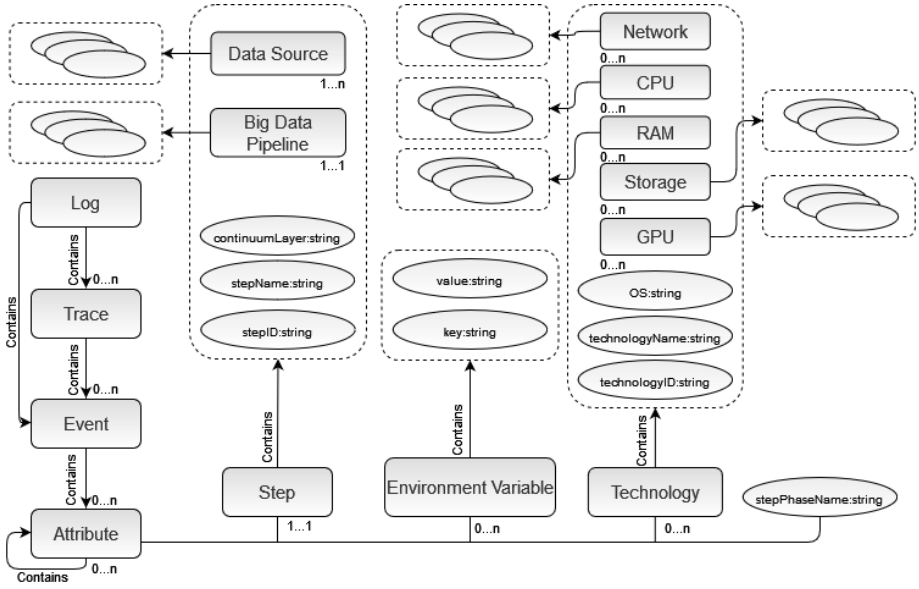

Fig. 3. Extension of XES derived from the proposed reference model. The nested attributes at the level of each Container attribute (e.g., Data Source or Big Data Pipeline) are omitted to avoid overwhelming the diagram.

Variable is of interest only if it is associated with at least one Step Phase. Finally, we model an ***Execution Trace*** with its *ID*. Each Execution Trace needs to contain at least one Step Phase, and each Step Phase is contained only in one Execution Trace. We point out that *Data Source* and *Environment Variable* are derived directly from DC-DSL, while *Technology*, and each class in relationship with it, has been customized by interpreting the requirements details defined for DC-DSL. It is worth noticing that while we based our reference model on some of the core concepts of DC-DSL, its structure has been iteratively evaluated with the domain experts of the business cases involved in the DataCloud project.

The main connection between the proposed UML class diagram and the concept of event log is the tight relation between pipeline steps and business process activities. Indeed, traditional business processes, whose executions are responsible for generating event data, consist of activities, while data pipelines contain a sequence of steps. Thus, a pipeline's step can be seen as an activity that performs some data transformation. We exploited this notion in the reference model by associating each Step with the information about its data sources and the computing layer (i.e., Cloud, Edge, or Fog) on which the transformation is applied. Even if steps are the core elements of a data pipeline, we decided to further split a single step into different phases. In this way, when it comes to runtime, information about when a step enters one of the phases of its lifecycle can be exploited for analysis purposes. Hence, an event log following the proposed

Table 1. Attributes at the Event and Trace levels in the XES extension.

Attribute Level	*Key*	*Type*
Trace	ID	Trace ID
Event	Step Phase ID	string
Event	Step Phase Name	string
Event	Outcame	boolean
Event	Timestamp	Date
Event	Step	Container
Event	Environment Variable	Container
Trace	Big Data Pipeline	Container
Event	Data Source	Container
Event	Technology	Container
Event	CPU	Container
Event	GPU	Container
Event	RAM	Container
Event	Storage	Container
Event	Network	Container

model will have multiple entries for the same pipeline step, one for each of its phases.

4.2 Extending XES

Fig. 3 describes an extension of the XES standard able to represent Big Data pipelines as formalized in the reference data model. As reported in the latest XES Standard Definition, the concepts of *log*, *trace*, and *event* contain no information, but they only define the structure of event data. Thus, information in an event log should be stored in *attributes*. All attributes in XES have a string-based key. Logs, traces, and events each contain an arbitrary number of attributes. There are six types of elementary attributes, each defined by the type of data value they represent, and two complex types: *List* and *Container*. These attributes hold any number (may be empty) of child attributes. The value of a List/Container attribute is derived from the values of its child attributes. Only in the case of the List, child attributes are ordered and their keys need not be unique.

For this reason, except for Execution Trace and Step Phase classes that match with the concept of *Trace* and *Event* in XES, we translated any other class included in the UML diagram of the reference model in a *Container* attribute. At the same time, we exploited the *List* attribute to represent relations between classes. The child attributes of any reference model class have been represented with one of the elementary attribute types.

In Table 1 we show the definition of the main attributes in the XES extension, at the *Event* and *Trace* levels. In Table 2, it is shown how we translated relationships between classes through lists at the levels of one of the container of the classes participating in the relationship. The lists highlighted with *(*)* should contain at least one item. Finally, we translated each attribute of the UML classes into Attributes at the level of each respective Container. For the sake of space, we do not show here all the details of the XES extension, whose specification

Table 2. List attributes in the XES extension to represent relations between classes.

Attribute Level	*Key*	*Type*
Event	Environment Variables	list[Environment Variable]
Event	Technologies	list[Technology](*)
Technology	CPUs	list[CPU](*)
Technology	GPUs	list[GPU](*)
Technology	RAMs	list[RAM](*)
Technology	Storages	list[Storage](*)
Technology	Networks	list[Network](*)
Event	Step	Step
Step	Big Data Pipeline	Big Data Pipeline
Step	Data Sources	list[Data Sources](*)

(obtained through the above considerations) is straightforward. Nonetheless, it is worth noticing that its definition represents our answer to **RQ1**.

5 Demonstration

To tackle **RQ2**, in this section we discuss how process mining techniques can be applied over our XES extension to: (i) get more detailed process-centric information on the use case pipeline of Section 3.4; (ii) obtain data-centric insights about its execution and untangle dark data accessed by the pipeline to generate new potential business value.

First, the XES extension works when an IS is recording events at the step phase level. This enables us to extract more insights into the executed steps of the big data pipeline using traditional process discovery techniques. Indeed, by applying one of the process discovery techniques available in [5] over the XES extension, we can: (i) get the duration of individual step phases and more accurate measures of the overall duration of steps; (ii) get more detailed information about which step phases are causing bottlenecks or other performance issues; (iii) have an idea of how the execution of pipeline steps is distributed on the computing continuum; and (iv) understand how different step phases interleave.

For example, by analyzing the DFG in Fig. 1, we notice that sometimes after the *Transformation* step there might be the need of installing supplementary resources. The usage of the XES extension would allow traditional process mining tools like Disco to get more detailed information about this issue, such as understanding: (i) in which phase of the *Transformation* step these additional resources are needed; (ii) how it relates to the outcome of the current step phase; (iii) in which continuum layer the associated computations are taking place, and (iv) if the remote (hardware) resources are needed to consume or produce data within the pipeline execution. Even more important, with a complete knowledge of the resources employed in each step phase it may support a fast understanding of the technological threshold that was exceeded during the *Transformation* step, which led to the need of installing supplementary resources.

Second, when both the DC-DSL representation of a data pipeline and its past executions recorded into an event log are available, we can apply tradi-

tional conformance checking techniques [7] to detect discrepancies (e.g., steps missing in some pipeline execution or performed in a wrong order, etc.) between the expected pipeline behavior and its observed executions. In addition, trace alignment [14] can further support experts to associate a severity to such discrepancies suggesting a repair strategy to fix them. For example, in the presence of an event log represented through our XES extension, we can catch the exact moment in the range of a step life-cycle where a deviation occurs. By looking at the data pipeline model of Fig. 2, there could be execution traces in which the "Query" step starts before the "Transformation" step but is completed after it, generating a misalignment.

Third, to obtain further insights from the event log other than the traditional sequence-flow based information and give value to the dark data involved in many pipeline executions, we can leverage some recent process mining solutions that exploit database (DB) theory to carry on analysis on the process behavior recorded over a relational DB. A couple of approaches specifying DB schemas fully compatible with the XES standard have been proposed to store log data [9,29], and they can be easily extended to accommodate the additions provided by our XES extension. Overall, our idea is to build patterns of SQL queries to discover relations between the steps of a data pipeline that would be not made explicit by relying on traditional process discovery techniques. In this direction, we present some interesting query performed on a DB whose schema perfectly matches the UML class diagram of the reference model in Fig. 2.

For example, to discover potential dark data sources, we could search for all the data sources that appear in the log as the output of some steps but are never used as input, as it is expressed in the following query (Query 1):

```
SELECT ds.name as DataSourceName
FROM DataSource ds
WHERE NOT EXISTS(
    SELECT * FROM stepUsesDataSource suds
    WHERE suds.dataSourceName = ds.name
    AND (ds.type = 'Input' OR ds.type = 'Both'))
```

Similarly, we may be interested in knowing the amount of dark data that the pipeline is producing on the cloud. Since dark data are known to be a potential risk for organizations, having them on the cloud can increase the probability of attackers that exploit them. To get an idea of this measure, we can run the following query (Query 2):

```
SELECT  ds.name as DataSourceName, ds.volume as DataSourceVolume
FROM    Step s, DataSource ds, StepUsesDataSource suds
WHERE   s.continuumLayer = 'Cloud' AND s.name = suds.stepName AND
        ds.name = suds.dataSourceName
AND NOT EXISTS(
        SELECT * FROM StepUsesDataSource sudsrc
        WHERE   sudsrc.dataSourceName = ds.name
        AND (ds.type = 'Input' OR ds.type = 'Both'))
```

Moreover, if we perform one of the queries available in the literature [26,27] implementing process discovery over an event log stored as a relational DB, we can translate the DFG of Fig. 1 in a DB view *DFG(StepPhaseID1, StepPhaseID2)*, in which each record describes pairs of subsequent step phases in the

DFG. Coming back to our use case pipeline, we can rely on such a view to understand if the need to install supplementary resources is associated with the outcome of the phases of the step *Transformation*. To this aim, we can run the following query (Query 3), which is searching for the outcome of all the step phases belonging to the step *Transformation* followed by a step phase belonging to the step *InstallSupplementaryResources*:

```
SELECT  sp1.name as StepPhaseName, sp1.outcome as StepPhaseOutcome
FROM    Step s1, Step s2, StepPhase sp1, StepPhase sp2,
        StepPhaseBelongsToStep spbts, DFG d
WHERE   (d.stepPhaseID1 = sp1.ID
    AND sp1.ID = spbts.StepPhaseID
    AND spbts.StepID = s1.ID
    AND s1.name = 'Transform')
AND     (d.stepPhaseID2 = sp2.ID
    AND sp2.ID = spbts.StepPhaseID
    AND spbts.StepID = s2.ID
    AND s2.name = 'InstallSupplementaryResources')
```

In a similar fashion we can obtain the resource threshold that is triggering this issue, by looking at the technologies used by step phases belonging to the step *Transformation* and followed by a step phase belonging to the step *InstallSupplementaryResources*, as expressed in the following query (Query 4):

```
SELECT  t.ID as TechnologyID, t.name as TechnologyName
FROM    Technology t, StepPhaseUsesTechnology sputc,
        Step s1, Step s2, StepPhase sp1, StepPhase sp2,
        StepPhaseBelongsToStep spbts, DFG d
WHERE   (d.stepPhaseID1 = sp1.ID
    AND sp1.ID = spbts.StepPhaseID
    AND spbts.StepID = s1.ID
    AND s1.name = 'Transform')
AND     (d.stepPhaseID2 = sp2.ID
    AND sp2.ID = spbts.StepPhaseID
    AND spbts.StepID = s2.ID
    AND s2.name = 'InstallSupplementaryResources')
AND     sp1.ID = sputc.StepPhaseID
    AND t.ID = sputc.TechnologyID
```

Once we have the IDs and names of those technologies, we can fetch more details about GPU, CPU, RAM, Storage or Network by performing simple queries, which are omitted here for the sake of brevity.

Finally, in many cases, dark data derives from domain-dependent values that are stored in the log but can not be exploited by general-purpose process discovery techniques. To address this issue, we can structure generalized queries that can be instantiated on a domain basis to infer insights on those values which would have not been used in a traditional setting. An example of such a query (Query 5) is the one in which we fetch all the steps containing at least a step phase with a specific pair of key and value between its environment variables:

```
SELECT  ev.key as Key, ev.value as Value, s.name as StepName
FROM    Step s, StepPhase sp, EnvironmentVariable ev,
        StepPhaseBelongsToStep spbts,
        StepPhaseContainsEnvironmentVariable spcev
WHERE   spbts.StepID = s.ID AND spbts.StepPhaseID = sp.ID
    AND sp.ID = spcev.StepPhaseID AND spcev.EnvironmentVariableID =
        ev.ID
    AND ev.key = 'domain-specific-key'
    AND ev.value = 'domain-specific-value'
```

6 Preliminary Evaluation

To address **RQ3**, we performed a test involving 10 Big Data pipeline management user experts. We aimed to understand if the use of targeted process mining techniques enabled us to perform BDPD effectively over an event log expressed through our XES extension. Specifically, starting from the event log employed in the real-world use case of Section 3.4, suitably augmented to accommodate the new concepts introduced in the XES extension, we presented to the users the results of the application of process mining techniques over the event log, as discussed in Section 5 (thus, related to process discovery, conformance checking and querying mechanisms). For each suggested application of process mining, we administered the users a questionnaire asking to rate: (i) the perceived effectiveness of the adopted solution in performing BDPD and uncovering dark data, and (ii) the complexity of extracting dark data from pipeline executions without the support of process mining. Questions are rated with a 4-point average scale structured as follows: 1 ("None"), 2 ("Low"), 3 ("Moderate"), 4 ("High"). Each user was allowed to add textual feedback to explain a score better. The average score for any question is reported in Table 3.

Table 3. Questionnaire results.

Process Mining solution	Effectiveness in performing BDPD and uncovering dark data (1-4)	Complexity of extracting dark data without BDPD (1-4)
Process Discovery	3.4	3.4
Conformance Checking	3.8	3.2
Query 1	3	3
Query 2	3	3.6
Query 3	3.8	3.6
Query 4	3.6	3.6
Query 5	3.6	3.6

The majority of the users (70%) were selected from the business case partners of the H2020 DataCloud project, which focus their business on managing big data pipelines targeting reduced live streaming production costs of sports events, trustworthy eHealth patient data management, and analytics in Industry 4.0 manufacturing. The remaining 30% of users are academics engaged in research activities related to data pipeline definition, deployment, and adaptation.

The questionnaire results clearly outline that: (i) process mining solutions represent an effective means to perform BDPD towards data pipeline analysis and dark data extraction, and (ii) it is extremely complex to obtain the same findings shown in Section 5 by employing traditional Big Data processing solutions. These conclusions are enforced by the fact that all the scores range from 3 to 4. It is worth noticing that, in their textual feedback, many users pointed out that similar results to process discovery and querying mechanisms could be obtained through an extensive analysis of the system logs recorded during pipeline executions by the IS of a company, which are often scattered among different

data sources. The users also recognized this is a time-consuming and error-prone activity requiring manual and trial-and-error testing over the logs. On the other hand, the users confirmed that the same results obtained by applying Conformance Checking in Section 5 can not be straightforwardly emulated using traditional Big Data processing techniques, unless ad-hoc programming scripts are developed.

7 Concluding Remarks

In the era of Big Data, the application of process-oriented solutions to deal with issues requiring data awareness is increasing [11,15,16]. In this context, the analysis of Big Data pipelines and the realization of novel techniques for efficiently managing their life-cycle is considered as a relevant research challenge in the field of Big Data processing. In this context, a BDPD solution that provides a precise knowledge of the characteristics of the processing steps performed during a pipeline execution (e.g., CPU usage, resource consumption, size of the involved data, etc.) can be used for better scheduling the available cloud resources, enabling smart load-balancing and memory management decisions before the execution of a data pipeline. The discovery activity is also crucial to interpret bottlenecks, inefficiencies and risks hidden behind the complexity of data pipelines, which prevent or delay their proper enactment.

To realize this vision, in this paper we have formalized a universally applicable reference data model to conceptualize the core concepts and properties of a Big Data pipeline execution. We provided an implementation of the model as an extension to the XES interchange standard for event logs and demonstrated its practical applicability in a concrete use case data pipeline.

Our reference model can contribute to the Big Data pipeline field by providing a common application-independent conceptual framework for capturing data pipeline executions. Moreover, by achieving the objectives of this research, we envision a relevant impact also for process mining future developments. Indeed, differently from the existing process mining techniques, we aim at discovering models of data pipelines that not only include the traditional sequence-flow constructs, but that embed information about the performance of the observed pipelines and the data manipulated by any step of the pipelines. This result would allow us to push forward the research on the discovery of data-aware and object-aware business processes, which is currently at an early stage [28].

As an immediate future work, we aim to validate the model's practical applicability and the effectiveness of process mining to perform BDPD and uncover dark data against the strong selection of complementary business cases involved in the H2020 DataCloud project. This will enable us to mitigate the rather preliminary evaluation presented in this paper, which limits the generalizability of our findings on the effectiveness of process mining (cf. RQ3) only to the data pipeline analyzed in the use case.

Acknowledgments This work is supported by the H2020 project DataCloud (Grant number 101016835), and the Sapienza project DISPIPE.

References

1. van der Aalst, W.M.P.: Process Mining: Data Science in Action. Springer (2016)
2. Abb, L., Rehse, J.: A Reference Data Model for Process-Related User Interaction Logs. In: 20th Int. Conf. on Business Process Management (BPM 2022) (2022)
3. Acampora, G., Vitiello, A., Di Stefano, B., van der Aalst, W.M.P., Günther, C., Verbeek, E.: IEEE 1849: The XES Standard. IEEE Comp. Int. Mag. (2017)
4. Agostinelli, S., Benvenuti, D., De Luzi, F., Marrella, A.: Big Data Pipeline Discovery through Process Mining: Challenges and Research Directions. In: 1st Italian Forum on Business Process Management, co-located with BPM 2021 (2021)
5. Augusto, A., Conforti, R., Dumas, M., La Rosa, M., Maggi, F.M., Marrella, A., Mecella, M., Soo, A.: Automated Discovery of Process Models from Event Logs: Review and Benchmark. IEEE TKDE **31**(4), 686–705 (2018)
6. Benvenuti, D., Falleroni, L., Marrella, A., Perales, F.: An interactive approach to support event log generation for data pipeline discovery. In: 2022 IEEE 46th Annual Computers, Software, and Applications Conf. (COMPSAC'22) (2022)
7. Carmona, J., van Dongen, B.F., Weidlich, M.: Conformance checking: foundations, milestones and challenges. In: Process Mining Handbook, pp. 155–190 (2022)
8. Corallo, A., Crespino, A.M., Vecchio, V.D., Lazoi, M., Marra, M.: Understanding and Defining Dark Data for the Manufacturing Industry. IEEE Transactions on Engineering Management **70**(2) (2021)
9. van Dongen, B.F., Shabani, S.: Relational XES: Data Management for Process Mining. In: CAiSE Forum 2015. pp. 169–176 (2015)
10. Gimpel, G.: Bringing Dark Data into the Light: Illuminating Existing IoT Data Lost within your Organization. Business Horizons **63**(4), 519–530 (2020)
11. Humayoun, S.R., Catarci, T., de Leoni, M., Marrella, A., Mecella, M., Bortenschlager, M., Steinmann, R.: Designing mobile systems in highly dynamic scenarios: The WORKPAD methodology. Knowledge, Tech. & Policy **22**, 25–43 (2009)
12. Johannesson, P., Perjons, E.: An Introduction to Design Science. Springer (2014)
13. Leemans, M., Liu, C.: XES Software Telemetry Extension. XES W. Group (2017)
14. de Leoni, M., Marrella, A.: Aligning Real Process Executions and Prescriptive Process Models through Automated Planning. Exp. Syst. App. **82** (2017)
15. Marrella, A., Mecella, M., Russo, A.: Collaboration on-the-field: Suggestions and beyond. In: 8th Int. Conf. on Information Systems for Crisis Response and Management (ISCRAM'11) (2011)
16. Marrella, A., Mecella, M., Russo, A.: Featuring Automatic Adaptivity through Workflow Enactment and Planning. In: 7th Int. Conf. on Coll. Comp.: Networking, Applications and Worksharing (CollaborateCom'11). pp. 372–381. IEEE (2011)
17. Mernik, M., Heering, J., Sloane, A.M.: When and how to develop domain-specific languages. ACM Comput. Surv. **37**(4), 316–344 (2005)
18. Munappy, A.R., Bosch, J., Olsson, H.H.: Data pipeline management in practice: Challenges and opportunities. In: Int. Conf. on Product-Focused Software Process Improvement (PROFES'20). pp. 168–184 (2020)
19. Nikolov, N., Dessalk, Y.D., Khan, A.Q., Soylu, A., Matskin, M., Payberah, A.H., Roman, D.: Conceptualization and scalable execution of big data workflows using domain-specific languages and software containers. Internet of Things **16** (2021)
20. Oleghe, O., Salonitis, K.: A framework for designing data pipelines for manufacturing systems. Procedia CIRP **93**, 724–729 (2020)
21. Pegoraro, M., Uysal, M.S., van der Aalst, W.M.: An XES extension for uncertain event data. arXiv preprint 2204.04135 (2022)

22. Plale, B., Kouper, I.: The Centrality of Data: Data Lifecycle and Data Pipelines. In: Data Analytics for Int. Transportation Syst. Elsevier (2017)
23. Rabl, T., Jacobsen, H.A.: Big Data Generation. In: Specifying Big Data Benchmarks, pp. 20–27. Springer (2012)
24. Rafiei, M., van der Aalst, W.M.: Privacy-Preserving Data Publishing in Process Mining. In: BPM forum 2020. pp. 122–138 (2020)
25. Roman, D., Prodan, R., Nikolov, N., Soylu, A., Matskin, M., Marrella, A., et al.: Big Data Pipelines on the Computing Continuum: Tapping the Dark Data. Computer **55**(11), 74–84 (2022)
26. Schönig, S., Rogge-Solti, A., Cabanillas, C., Jablonski, S., Mendling, J.: Efficient and Customisable Declarative Process Mining with SQL. In: 28th Int. Conf. on Advanced Information Systems Engineering (CAiSE'16) (2016)
27. Schönig, S.: SQL Queries for Declarative Process Mining on Event Logs of Relational Databases. arXiv preprint 1512.00196 (2015)
28. Steinau, S., Marrella, A., Andrews, K., Leotta, F., Mecella, M., Reichert, M.: DALEC: a framework for the systematic evaluation of data-centric approaches to process management software. Software & Systems Modeling **18**(4) (2019)
29. Syamsiyah, A., van Dongen, B.F., van der Aalst, W.M.P.: DB-XES: enabling process discovery in the large. In: Sixth Int. Symp. on Data-Driven Process Discvery and Analysis SIMPDA'16. vol. 1757, pp. 63–77 (2016)
30. Teymourlouei, H., Jackson, L.: Dark Data: Managing Cybersecurity Challenges and Generating Benefits. In: Advances in Parallel & Distributed Processing, and Applications, pp. 91–104. Springer (2021)