DIS-PIPE: A Tool for Data Pipeline Discovery

Simone Agostinelli[1,†], Dario Benvenuti[1,†], Andrea Marrella[1,†] and Jacopo Rossi[1,*,†]

[1]*Sapienza Universitá di Roma - Dipartimento di Ingegneria Informatica Automatica e Gestionale Antonio Ruberti*

Abstract

This paper presents DIS-PIPE, a software tool that leverages well-established process mining techniques to tackle the Data Pipeline Discovery (DPD) task. Data pipelines are composite steps that move data from disparate sources to some data consumers. While data travels through the pipeline, it can undergo various transformations processed by computational platforms. In this context, DPD targets learning the structure and behavior of a data pipeline from an event log that keeps track of its past executions, uncovering, to some extent, specific execution-related dark data whose knowledge is critical to improving the quality of pipeline modeling. DIS-PIPE has been designed, implemented, and validated in the H2020 European project DataCloud context, and is able to interpret XES logs enriched with information to capture the core concepts of data pipelines.

Keywords

Data Pipeline Discovery (DPD), Data Pipeline, Process Mining, Event Log, Dark Data, DataCloud, XES

Metadata description	Value
Tool name	DIS-PIPE
Current version	1.0
Legal code license	Apache 2.0
Languages, tools and services used	Python, Flask , Java, PostgreSQL, C, HTML, Javascript, CSS
Supported operating environment	GNU/Linux, Docker
Download/Demo URL	https://github.com/bpm-diag/DIS-PIPE
Documentation URL	https://github.com/bpm-diag/DIS-PIPE/blob/main/README.md
Source code repository	https://github.com/bpm-diag/DIS-PIPE
Screencast video	https://github.com/bpm-diag/DIS-PIPE/tree/main/video

1. Introduction

With the recent developments of the Internet of Things and cloud-based technologies, massive amounts of data are generated by heterogeneous sources and stored through dedicated cloud solutions. Many data are stored for compliance purposes only but not turned into value, thus becoming *dark data*, which may emerge at various stages of data processing and are considered untapped [1]. A typical example is the server *log files*, which can give clues related to the *steps enacted for data processing*, such as the performance of the cloud storage systems in processing data, previously unnoticed traffic patterns, etc.

ICPM Doctoral Consortium and Demo Track 2024, October 14-18, 2024, Kongens Lyngby, Denmark

*Corresponding author.

†These authors contributed equally.

agostinelli@diag.uniroma1.it (S. Agostinelli); d.benvenuti@diag.uniroma1.it (D. Benvenuti); marrella@diag.uniroma1.it (A. Marrella); j.rossi@diag.uniroma1.it (J. Rossi)

The literature on data processing agrees that *discovering* and *interpreting* the *data pipelines* that run within the organization's workflow is essential to *uncovering* and *valorizing* such dark data for insights and decision-making [2]. Data pipelines are composite steps for: *(i)* ingesting raw data from disparate sources; *(ii)* processing such data with the support of (cloud-based) computational platforms to ensure appropriate data integration; and *(iii)* moving it toward the data consumers, who undertake further transformations and visualizations.

State-of-the-art approaches for managing data pipelines assume their anatomy is known by design and expressed using one of the many available domain-specific languages (DSLs) [3]. Nonetheless, modeling a data pipeline from scratch means having complete knowledge of the pipeline behavior and its interactions with the data sources and computational platforms at the outset. This is recognized as a time-consuming and error-prone task since the behavior of the data pipeline strongly depends on certain technological details that will be known only during its execution, making this kind of data as "dark" at design time.

To mitigate this issue, we present DIS-PIPE, a software tool that leverages well-established process mining techniques to tackle the Data Pipeline Discovery (DPD) task. DIS-PIPE has been designed, implemented, and validated in the context of the H2020 European project DataCloud.[1] According to [4], DPD targets *learning the structure and behavior of a data pipeline from an event log that keeps track of its past executions, uncovering, to some extent, specific execution-related dark data and, consequently, aiming to enhance the overall quality of pipeline modeling.*

However, traditional event logs used for process mining are limited in scope when used in fields far from business process management [5]. 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 DPD. For this reason, in DIS-PIPE we rely on an extension to the XES interchange standard, formalized through a reference data model that represents data pipelines' core concepts and execution properties unambiguously, e.g., CPU usage, resource consumption, etc.; see [4] for more details. This enables DIS-PIPE to reveal fact-based insights into how a data pipeline transpires and interacts with dark data by interpreting the event log that records its execution details.

The rest of the paper is organized as follows. To show the relevance of DPD, Section 2 introduces a use case involving a data pipeline for managing digital marketing campaigns. Section 3 presents the tool architecture and the technical aspects and functionalities of DIS-PIPE. Section 4 describes the tool's maturity and its practical applicability. Finally, Section 5 concludes the paper by discussing the significance of DIS-PIPE to the process mining field.

2. Use case

Let us consider a real-world use case offered by one of the enterprises involved in the DataCloud project. The pipeline, which targets higher mobile business revenues in smart marketing campaigns, is triggered when (1) the system receives a request to model a new marketing campaign or report on how an existing one is performing. In both cases, the first two steps of the pipeline are: (2) querying the required data and (3) applying specific transformations to it. Sometimes, after the *Transformation* step there might be the need of installing supplementary

[1] https://cordis.europa.eu/project/id/101016835

resources (4). Then, if the request is for a report, the queried data must then be merged (5a). On the other hand, for a model request, an algorithm to compute it is launched, and the results are stored in dedicated databases (5b). Finally, the pipeline ends with the visualization of the results (6) and either the report (7a) or the model (7b) 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 into the workflow behind the data pipeline (e.g., how steps are sequenced, branching probabilities, etc.). While useful, they do not allow us to infer further details from different perspectives, e.g., the flow of data accessed and manipulated during the pipeline execution, the technologies used to process data, etc. In a nutshell, *there are relevant execution data that any logging system could easily capture during pipeline execution but are lost during the analysis, thus becoming dark data.* For instance, in this use case, there might be an interest in analyzing the data sources that appear in the log and determining the amount of data the pipeline produces on the cloud. Indeed, having such data on the cloud can increase the probability of attackers to exploit them, becoming a potential risk for the organization. The XES event log underlying the behavior of the presented data pipeline is available for testing at: https://doi.org/10.5281/zenodo.13619148.

3. DIS-PIPE Architecture

The software architecture devised for DIS-PIPE is shown in Fig. 1. DIS-PIPE includes a frontend with a GUI written in HTML, Javascript and CSS, which enables us to import XES event logs represented as described in [4]. A backend developed using Python, Java, PostgreSQL and C, integrates with a *Flask* web application to receive user inputs and provide discovery results. Specifically, the following functionalities are provided:

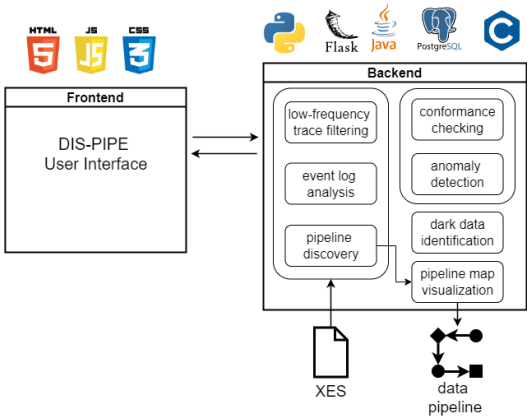
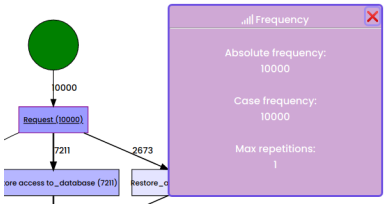


Figure 1: DIS-PIPE software architecture

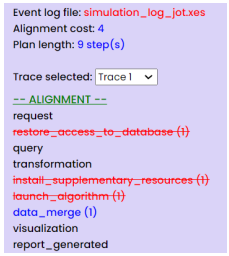
Pipeline discovery: it uses the *PM4Py* [6] library for process discovery to find a suitable pipeline model that describes the order of pipeline steps as recorded into a given event log.

Pipeline map visualization: it provides a flowchart view of a discovered pipeline using a Directly-Follows Graph (DFG). Each node is a pipeline step, and the edges represent the connections between them. The DFG is enriched with specific metrics (cf. Fig. 2(a)) that are displayed through two visualizations: *frequency* (with the parts of the pipeline that were executed most frequently) and *performance* (with the run-time duration details).

Low-frequency trace filtering: Two sliders can be used to apply filters to remove potential outliers and control the level of details shown in the DFG model. Adjusting the *step sliders* from 100% to 0% reduces the number



(a) Pipeline Step Information



(b) Anomaly Detection

Step	key	value
Install_supplementary_resources	DataSourceName	errors-source
Query	DataSourceName	Queries-source
Restore_access_to_database	DataSourceName	DB-errors-source
Transformation	DataSourceName	Transformed-data

(c) Result of Query 1

dsname.key	dsname.value	dsvolume.key	dsvolume.value
DataSourceName	DB-errors-source	DataSourceVolume	10000

(d) Result of Query 2

Figure 2: DIS-PIPE interface

of pipeline steps in the DFG only on the most frequent ones. By calibrating the *path slider* from 100% to 0%, we can focus only on the most dominant paths of the data pipeline.

Event log analysis: It enables inspecting the details of the pipeline execution traces in the log to analyze the full history of the recorded traces, including how the traces are categorized into variants having the same step sequence. A relevant functionality consists of iteratively applying specific filters on the log to exclude non-relevant traces and re-discover the pipeline model starting from the filtered log for a focused analysis. Four filters were developed, concerning (i) timeframe, (ii) performance, (iii) log attributes and (iv) compliance to declarative constraints.

Conformance checking and Anomaly detection: DIS-PIPE allows comparing an event log (i.e., the observed behavior of multiple pipeline executions) against the structure of a data pipeline previously specified with a DSL. The aim is to detect the misalignments between the observed and the expected behaviour of a data pipeline and understand their nature. We provide a scalable approach that relies on automated planning in AI [7], and a grounding algorithm to build planning domains and problems to solve the trace alignment task [8]. The planning system embedded in DIS-PIPE is the state-of-the-art planner Fast-Downward[2]. Its adoption to achieve conformance checking is proven to be effective in the case of models and event logs of remarkable size [7]. For anomaly detection, the results of the conformance checking are visualized on the pipeline's DFG model, highlighting deviations in red. DIS-PIPE can identify pipeline steps that were executed but not allowed by the DSL model, and steps that should have been executed but were missing from the trace (cf. Fig. 2(b)).

Dark data identification: To obtain further insights other than the traditional sequence flow-based information and give value to the dark data involved in the pipeline executions, we exploit database (DB) theory to analyze the data pipeline behavior recorded over a relational DB. To this end, we employed the procedure detailed in our previous work [9] to translate a XES file, extended with the reference model for conceptualizing the data pipelines' core concepts [4], to a relational DB for enabling log querying via SQL. In DIS-PIPE, the log translation was performed by creating a PostgreSQL database. The database tables were structured to have the following columns: *trace_id*, *event_name*, *time*, *key*, and *value*. Specifically, *trace_id* refers to the ID of a log trace, *event_name* indicates the name of a single pipeline step, *time* refers to the timestamp, and *key* and *value* indicate the name of a specific log parameter and its value.

[2]https://www.fast-downward.org/

The latter two columns allow for considering all log parameters, including those technical ones related to the pipeline executions as formalized in the reference model [4]. With this structure, DIS-PIPE enables the writing of personalized SQL queries to discover relationships between the steps of a data pipeline that traditional process discovery techniques may not make explicit. For example, if we consider the use case of Section 2, we can write a query using a dedicated text box visual component to understand if there is a need to install supplementary resources after the *Transformation* step of the data pipeline. Similarly, we can obtain the resource threshold that is triggering this issue, by looking at the technologies used during the *Transformation* step. If a query is considered relevant, DIS-PIPE enables us to save it for being updated and re-executed without redefining it from scratch. For example, to uncover potential dark data sources involved in the use case pipeline, DIS-PIPE allows searching for all the data sources that appear in the log as the output of some pipeline steps but are never used as input. As shown in Fig. 2(c), four different sources remained untapped by the company. It is also possible to determine the amount of dark data produced on the cloud. As shown in Fig. 2(d), only one source is present on the cloud, encompassing 10,000 exposed files. For the sake of space, we let the readers refer to [4] for more examples of queries and their structure.

4. Maturity of the tool

DIS-PIPE was iteratively evaluated and improved during the DataCloud project. Two significant validations were performed in the last year of the project before releasing the current version of the tool. First, during ICPM 2023 in Rome, a dedicated event was organized to disseminate the project results and validate the functionalities developed for DIS-PIPE. Specifically, 30 users belonging to three categories (i.e., researchers in process mining, data scientists working in companies, and PhD Students in Engineering in Computer Science) were involved in the validation. Each user was asked to perform 6 different scenarios including various tasks covering each tool's functionality. Finally, users completed a questionnaire to assess whether the outcome of each functionality was understandable and helpful for performing DPD. The results, measured on a four-point Likert scale (from *1-strongly disagree* to *4-strongly agree*), showed a median score of 3.6, indicating that users considered effective the functionalities of DIS-PIPE for DPD.

We also evaluated the usability of the tool through a SUS (System Usability Scale) test that involved 16 Big Data analytics experts from the business case partners in the project and 10 MsC students knowledgeable in using process mining tools, such as Fluxicon Disco or Celonis. Users were initially instructed on the DIS-PIPE functionalities through a short training session, followed by an interactive session with the tool and the administration of the 10-item SUS questionnaire, which rates responses on a five-point Likert scale from *1-strongly disagree* to *5-strongly agree*. The overall SUS score (cf. [10] for instructions on how to compute it), considered a reliable measure of perceived usability and user experience, was benchmarked against literature standards to determine the tool's usability. The SUS score was 79.0, corresponding to an A- rank according to the selected benchmark [10], indicating usability between *very good* and *excellent*.

5. Conclusion

DIS-PIPE provides an innovative contribution to DPD by enabling the discovery of the structure and behaviour of a data pipeline directly from an event log, without the need to predefine the pipeline's anatomy, as is often required by many state-of-the-art approaches [3]. Unlike the traditional process mining tools, DIS-PIPE can interpret XES logs by implementing the reference model for data pipelines described in [4]. This ability, combined with the formalization of the event log as a relational DB and the targeted use of well-established DB techniques to perform log querying [9], allows us to uncover dark data recorded in the log that process mining techniques overlook. We believe that DIS-PIPE not only support data scientists in enhancing the modeling stage of a data pipeline, but also represents a concrete and successful application of process mining techniques to address a critical and timely challenge in data processing.

Acknowledgments. This work has been supported by the H2020 project DataCloud (Grant number 101016835), the Sapienza projects DISPIPE and FOND-AIBPM, and the PNRR MUR project PE0000013-FAIR. Jacopo Rossi is supported by Thales Alenia Space and Regione Lazio, through the fellowships 35757-22066DP000000041-A0627S0031 *Advanced Software Based on Cloud Computing and Machine Learning for Space Systems.*

References

[1] G. Gimpel, Bringing Dark Data into the Light: Illuminating Existing IoT Data Lost within your Organization, Business Horizons 63 (2020).

[2] D. Roman, R. Prodan, N. Nikolov, A. Soylu, M. Matskin, A. Marrella, D. Kimovski, B. Elvesæter, A. Simonet-Boulogne, G. Ledakis, H. Song, F. Leotta, E. Kharlamov, Big Data Pipelines on the Computing Continuum: Tapping the Dark Data, Computer 55 (2022).

[3] N. Nikolov, Y. D. Dessalk, A. Q. Khan, A. Soylu, M. Matskin, A. H. Payberah, D. Roman, Conceptualization and Scalable Execution of Big Data Workflows using Domain-specific Languages and Software Containers, Internet of Things 16 (2021).

[4] D. Benvenuti, A. Marrella, J. Rossi, N. Nikolov, D. Roman, A. Soylu, F. Perales, A Reference Data Model to Specify Event Logs for Big Data Pipeline Discovery, in: BPM'23, 2023.

[5] L. Abb, J. Rehse, A Reference Data Model for Process-Related User Interaction Logs, in: BPM'22, Springer, 2022.

[6] A. Berti, S. van Zelst, D. Schuster, PM4Py: a Process Mining Library for Python, Software Impacts 17 (2023).

[7] A. Marrella, Automated Planning for Business Process Management, Journal on Data Semantics 8 (2019).

[8] G. Acitelli, M. Angelini, S. Bonomi, F. M. Maggi, A. Marrella, A. Palma, Context-Aware Trace Alignment with Automated Planning, in: ICPM '22, IEEE, 2022.

[9] F. Riva, D. Benvenuti, F. M. Maggi, A. Marrella, M. Montali, An SQL-Based Declarative Process Mining Framework for Analyzing Process Data Stored in Relational Databases, in: BPM'23, 2023.

[10] J. Sauro, J. R. Lewis, Quantifying the User Experience: Practical Statistics for User Research, Morgan Kaufmann, 2016.