Enabling Process Mining in Aircraft Manufactures: Extracting Event Logs and Discovering Processes from Complex Data

Álvaro Valencia-Parra[1], Belén Ramos-Gutiérrez[1], Ángel Jesús Varela-Vaca[1], María Teresa Gómez-López[1], and Antonio García Bernal[2]

[1] IDEA Research Group, Dpto. Lenguajes y Sistemas Informáticos, Universidad de Sevilla, Spain http://www.idea.us.es/
{avalencia,brgutierrez,ajvarela,maytegomez}@us.es
[2] Airbus Defence & Space, http://www.airbus.com/
antonio.garcia.bernal@airbus.com

Abstract. Process mining is employed by organizations to completely understand and improve their processes and to detect possible deviations from expected behavior. Process discovery uses event logs as input data, which describe the times of the actions that occur the traces. Currently, Internet-of-Things environments generate massive distributed and not always structured data, which brings about new complex scenarios since data must first be transformed in order to be handled by process mining tools. This paper shows the success case of application of a solution that permits the transformation of complex semi-structured data of an assembly-aircraft process in order to create event logs that can be managed by the process mining paradigm. A Domain-Specific Language and a prototype have been implemented to facilitate the extraction of data into the unified traces of an event log. The implementation performed has been applied within a project in the aeronautic industry, and promising results have been obtained of the log extraction for the discovery of processes and the resulting improvement of the assembly-aircraft process.

Keywords: Process Mining · Event log · IoT · Complex data structure · Domain-Specific Language

1 Introduction

Process mining facilitates the understanding of business processes of organizations. This technique usually involves [14] the analysis of process event logs and the discovery of the process models behind them. There exist several approaches to discover them [4–6].

One of the challenges in process mining is related to *the ability to extract suitable events* [3]. Moreover, the way to obtain event logs produced by the systems is becoming more and more complex. The systems tend to produce massive, distributed, heterogeneous, and complex data in new Internet-of-Things (IoT) environments, and derive chaotic combinations of elements [13,18] without

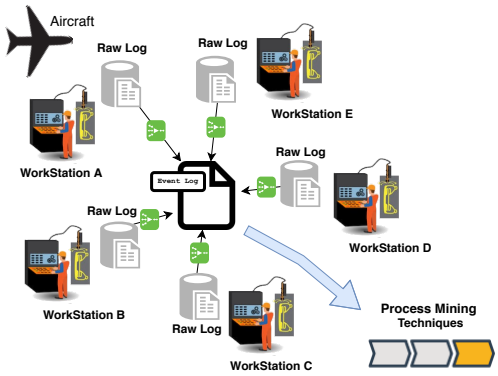

Fig. 1. Example of IoT Scenario.

structured schema. It brings about an extra level of complexity to be managed by the current process mining solutions that involve the use of event logs because they are mainly focused on structured homogeneous data.

The main issue tackled in this paper concerns how complex data structures, typically generated in IoT environments, can be transformed to create an event log that could be managed by the existing tools in process mining. Figure 1 shows a scenario of log extraction in the aeronautic industry, based on the data produced by workstations in an aircraft-assembly process. In this scenario, several workstations perform tests and produce logs of their execution. The information, which follows a complex data structure, is stored in a NoSQL database (i.e., MongoDB). The challenges are thus aggregating such complex data and extracting event logs in XES format from them. Then, process discovery techniques can be applied.

Hence, the objective of this paper is to develop an approach that enables the extraction of event logs in XES format from complex data. In addition, this solution is integrated in a tool so that non-expert users can benefit from this solution.

2 Situation faced

Challenging Scenario

Nowadays, IoT environments are becoming highly relevant for organizations [12] due to the necessity of monitoring their own operations. The challenge is how to manage and analyze the recovered information in order to be useful. The project *Clean Sky 2: A-24 One step beyond on automated testing technologies*, developed in collaboration with Airbus Space & Defence, inspired the desire to improve data extraction in IoT scenarios for the creation of event logs. The example is based on the assembly and testing process of aircraft as previously shown in Figure 1.

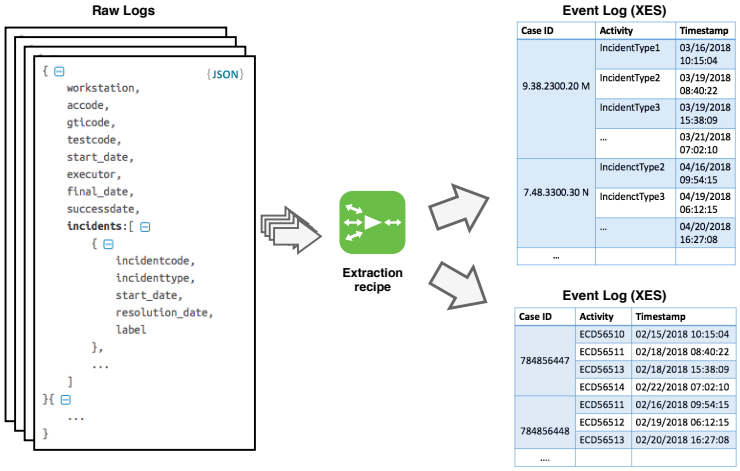

Fig. 2. Scenario of data extraction.

This describes a complex process that is executed at several locations without guidance from a Business Process Management System (BPMS) [8].

In this context, the main process in the organization is focused on the assembly of the aircraft in accordance with the design of an assembly chain formed of various workstations. Each workstation in the factory executes its own separate tests depending on the equipment that it has available, as described in the following:

1. An aircraft with an identifier (`accode`) is located at a `workstation`.
2. In each workstation, a set of test instructions must be carried out (`gticode`).
3. Each set of test instructions can be executed several times until it is validated by a designated operator (`executor`), and it begins and finishes at a specific moment (`start_date`, `final_date`) and `successdate`.
4. Each execution might either finish successfully or generate incidents. Those include an `incidentcode`, `incidenttype`, `start_date`, `resolution_date`, and `label`.

The workstations are physically distributed and generate complex data structures depending on the tests executed thereon. In order to generate logs that can be consumed by process-mining techniques, an extraction from the structure presented in Figure 2 must be defined. The left side of the figure represents a set of datasets formed of complex structures (lists and nested structures). However, the event logs must fit the schema presented in the right-hand side of the figure.

By using this data, several trace logs can be generated, which will depend on the type of process that must be discovered, for example, the evolution of the aircraft per workstation, the evolution of tests, or the life-cycle of incidents. These different perspectives (points of view) of the data can be used as the input of the process-discovery techniques to ascertain the actual process and to enable the analysis and improvement of the processes of assembly and testing.

Problems to be faced

The main problem to be faced in this scenario is the lack of proposals to extract event logs from complex semi-structured data. The majority of the approaches that focus on the extraction of event logs have been developed using relational databases as a starting point. The main idea in these cases is that events leave footprints by changing the underlying databases that are registered in the redo logs of the database system [2, 15–17]. This idea has been supported by tools, such as XESame and PromImport plugin[3] [10, 20], that transform relational database information into XES event logs. This data extraction from relational databases has also been analysed from the perspective of ontology-based data access [6,7] by using metamodels as an intermediary [9], and has been supported by the *onProm* tool[4].

Nonetheless, as the scenario of IoT confirms, relational databases are not considered as the unique source of events. In [11], an approach is presented that shows how data is extracted from bug report histories, which are XML, JSON, or log files, although nested and complex structures are excluded. This shows that the need to extract event logs from unstructured sources is a real issue, but there is an absence of frameworks that facilitate it. In the same study, a quasi-manual mapping of the attributes of the report and those of the process is carried out. Then, they are stored in a relational database from which the log of events will be extracted. In other contexts, such as CRM (Customer Relationship Management), the possibility of transforming unstructured data into a log of events in XES log format [1] has also been studied (e.g., [5]). In that case, the proposal involves a framework to discover business process models from semi-structured data which applies various pre-processing steps to CRM activities and then uses Latent Dirichlet Allocation (LDA) to classify and label all activities automatically, by constructing a log of XES events.

To the best of our knowledge, our proposal constitutes the first approach for the construction of XES event logs from complex semi-structured data, abstracting non-expert users from the definition of complex transformations required to obtain a XES event logs.

3 Action taken

First of all, the characteristics of the dataset (i.e., raw logs) for the real scenario are depicted. In order to tackle the problems of event log extraction, we have developed a Domain-Specific Language (DSL) [19] to enable XES event log extraction from complex semi-structured data. It fulfills the main objective, since it abstract users from the necessity of transforming complex data with nested and recursive structures when generating event logs in XES format. In short, the proposal is based on the specification of the paths to the attributes to be employed as *case_id*, *activity_id*, *timestamp* and other optional parameters of

[3] PromImport plugin: http://www.promtools.org/promimport/
[4] onProm tool: https://onprom.inf.unibz.it/

XES event logs. The underlying framework then infers the transformations to be performed in order to reach the desired schema, in this case, a XES event log with the attributes indicated by the user.

Understanding the data

In order to measure the complexity of the scenario, it is necessary to ascertain the characteristics of the datasets (i.e., raw logs) used as the input of our approach. As shown in Table 1, the dataset contains 9367 sets of tests, provided by 52 workstations. In these workstations, 15 aircraft have been tested, with a total number of $1,110$ tests (i.e., GTI). The tests can be executed more than once for each aircraft. This is the reason why there are 9397 different executions of these tests. These repetitions occur when a test execution is unsuccessful, thereby causing, an incident, hence there are $6,049$ incidents in the raw log. In order to complement this information, some statistical data has been extracted (for instance, the average of GTI per aircraft) that provide a greater level of detail. We would like to point out that the dataset presented is a subset of the real data obtained in the project *Clean Sky 2*. Our dataset is just a small part of the large amount of information generated, is made up of 9397 raw logs, filtered and modified.

Taking into account the IoT scenario, each workstation periodically provides a raw log on the execution of a test on a particular aircraft, which may also contain incidents.

Table 1. Characteristics of the raw log [5] .

Description	Values
Number of Aircraft	15
Number of Workstations	52
Number of GTIs	1,110
Number of Executions	9,397
Number of Incidents	3,049

Illustration of event log extraction

Example 1.1 illustrates a use of the DSL. It represents *the process of the assembly aircraft for each of the workstations where the tests are evaluated.* The aircraft production follows a process where each part thereof is assembled and tested in accordance with the design of the engineers. Each step of the assembly process is performed in a workstation, where the resulting event log is formed of a set of traces with the following information:

[5] Due to confidentiality reasons, the data shown regarding incidents, time, dates have been manipulated and anonymised to avoid showing real data and the values only represent a subset of the real data.

- ***case_ id***: *accode* (i.e., the aircraft identifier).
- ***activity_ id***: *workstation*. Several tests can be executed in the same workstation, and therefore if more than one consecutive register of the testing of an aircraft is found in the same workstation, then only the first registration is considered in the creation of the event log.
- ***timestamp***: *start_ date* (i.e., the date when the aircraft passed through the workstation).

Example 1.1: Piece of extraction code

```
extract(
 define trace id("accode"),
 define trace event(
  activity = "workstation",
  criteria =
   orderBy(t"start_date" -> toDate("MM/dd/yyyy HH:mm:ss")),
   timestamp = t"start_date" -> toDate("MM/dd/yyyy HH:mm:ss")
 )
) from ("mongodb://mongo-instance:27017/db/logs")
```

The event log generated by the piece of code in *Example 1.1* is shown in *Example 1.2*.

Example 1.2: Example of generated event log

```
<log xes.version="1.0"
  xes.features="nested-attributes"
  openxes.version="1.0RC7">
  <trace>
    <string key="concept:name" value="7122693173813"/>
    <event>
      <string key="concept:name" value="ECD56510"/>
      <date key="time:timestamp"
        value="2016-10-07T12:11:15.000+02:00"/>
    </event>
    [...]
  </trace>
  [...]
</log>
```

4 Results achieved

This section describes how the obtained results help improve the process in the assembly-aircraft process. A tool implemented to validate the results is also present.

Extraction of event logs

First of all, a set of test cases are outlined bellow. The results are discussed in next subsections.

- **Test Case A. Process of the aircraft according to the workstation that executes the test.** This test case has been described in Section 3 (cf. Example 1.1).

– **Test Case B. Process of the aircraft according to the GTI execution.** During the aircraft assembly process, a set of test instructions must be carried out. In this test case, the processes of the aircraft according to the tests applied to them (*gticode*) are extracted. Hence, the resulting event log has the form:

  - ***case_ id***: *accode* (i.e., the aircraft identifier).
  - ***activity_ id***: *gticode*. If an aircraft passes the same test more than once, then it is only considered the first time that it passed.
  - ***timestamp***: *start_ date* (i.e., the first time the test was applied to the aircraft).

  The piece of code which enables this test case to be carried out is shown in Example 1.3.

Example 1.3: Test Case B

```
extract(
 define trace id("accode"),
 define trace event(
  activity = "gticode",
  criteria =
   orderBy(t"start_date" -> toDate("MM/dd/yyyy HH:mm:ss")),
  timestamp = t"start_date" -> toDate("MM/dd/yyyy HH:mm:ss")
 )
) from ("mongodb://mongo-instance:27017/db/logs")
```

– **Test Case C. Process of the incidents according to the type of incident.** As explained in Section 2, the tests (*gticode*) could report a set of incidents with an (*incidenttype*). The resulting event log must be in the following form:

  - ***case_ id***: *gticode* (i.e., the test identifier).
  - ***activity_ id***: *incidents.incidenttype*. If a type of incident is produced more than once in a test, then it is only considered the first time it was produced.
  - ***timestamp***: *start_ date* (i.e., the first time this type of incident occurred).

  The piece of code which corresponds to this extraction is shown in Example 1.4.

Example 1.4: Test Case C

```
extract(
 define trace id("gticode"),
 define trace event(
  activity = "incident.incidenttype",
  criteria =
   orderBy(t"start_date" -> toDate("MM/dd/yyyy HH:mm:ss")),
  timestamp = t"start_date" -> toDate("MM/dd/yyyy HH:mm:ss")
 )
) from ("mongodb://mongo-instance:27017/db/logs")
```

Analysis of the extracted event logs

Once the event logs (i.e., XES files) are obtained, it is possible to perform certain studies on the data that they contain and discover the processes. Next, the logs retrieved after each extraction are explained in a more precise manner. Table 2 describes the features attained for each example of data extraction. A process-discovery tool, Disco[6], has been employed to analyse these logs.

Table 2. Extracted logs features [7].

Description	Test Case A	Test Case B	Test Case C
CaseId	15	15	673
Events	369	5771	1777
Activities	52	1110	10
Median case duration	75.8 days	50.2 days	0 miliseconds
Mean case duration	76.1 days	67.1 days	20.8 days
Activities minimal frequency	1	1	16
Activities median frequency	6	3	108
Activities mean frequency	7.1	5.2	177.7
Activities maximal frequency	15	15	424
Activities frequency standard deviation	3.45	4.58	144.3

The resulting processes discovered are illustrated in Figure 3. These processes help in the understanding and identification of certain information that could not be analysed manually. For instance, the process for *Test Case A* helps towards the understanding of the flow of the aircraft through the different workstations in the factory. We can observe how certain workstations can perform tests in parallel with other workstations. *Test Case C* can help in the understanding of the flow of the incidents, by showing how the incidents evolve.

Effects of resulted action taken

The event logs and consequently discovered processes helped to improve the industry in the understanding and knowledge about the aircraft-assembly process:

Test Case A. In some cases an aircraft was in two workstations simultaneously, because for some tests it is easier to move the station than to move the aircraft. Regarding workstations, there are workstations with assumable transition values between them (hours or days) but also workstations in which the time in which the aircraft is at, is always 0 seconds, thus, we can assume that they are intermediate states in which nothing is actually done. On the other hand, there are workstations in which the aircraft has only been once and in almost all but one workstation the aircraft has been 0ms, in one of them (55

[6] Disco by Fluxicon: https://fluxicon.com/disco/

[7] Due to confidentiality reasons, the data shown regarding incidents, time, dates have been manipulated and anonymised to avoid showing real data and the values only represent a subset of the real data.

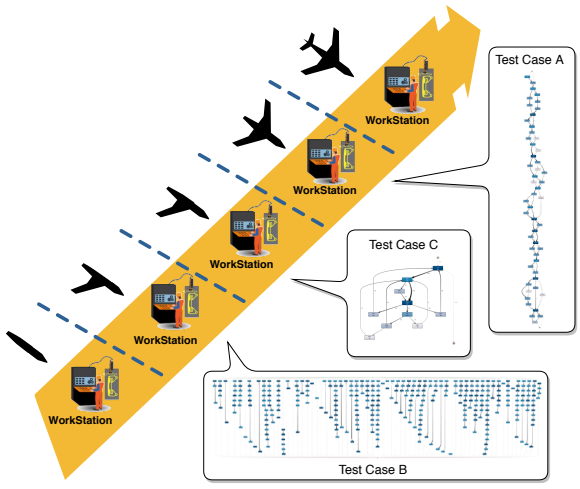

Fig. 3. Result of the process discovery for the extraction of test case A, B and C.

days). Knowing the average time invested in each workstation and the workstations through which the aircraft must pass for aircraft of the same type, it is possible for experts to know if deviations are occurring and the current state against what would be desired.

Test Case B. This case helped experts to know the sequence of tests that must be carried out and the time that must be spent in each one. Finally, it allows experts to make decisions on the improvements that could be applied to the tests performed on the workstations.

Test Case C. The discovery of the incident process by type makes it easier for experts to know which types of incidents are most likely to occur after another incident occurred. The discovery of the incident process by type helps know the average time that is likely to be invested in the resolution of the same incident and hence know how this will affect the entire assembly and testing process. The most frequent type of test is in all cases the first incident and also the last one. Other particular aspects are detected such as after the incident of shift change of operators, if an incident occurs, which occurs in more than 30 percent of cases, it is always of two possible types: *Aircraft Configuration Failure* or *Test Edition Failure*.

Tool and Technical Details

The implemented tool is based on three main components: (1) the DSL language, which enables the specification of the recipes of extraction from NoSQL databases, such as MongoDB, and the generation of event logs in XES format, (2)

a prototype of application (cf., Figure 4) in which transformation can be defined in a user-friendly way, and; (3) a connector which allows the generated XES logs to automatically feed the ProM^{TM}[8] tool for the discovery of process models. All the resources that have been employed in this work are freely available at [19]. The raw log used in the paper has been stored in a MongoDB database and the extraction recipes have been implemented in the DSL language, based on Scala, and developed using a model-driven engineering approach, which enables integration with other platforms. The current implementation of the connector uses only *Inductive Miner* [18]. However, in order to illustrate the discovery process, other tools, such as Disco from Fluxicon, have also been employed.

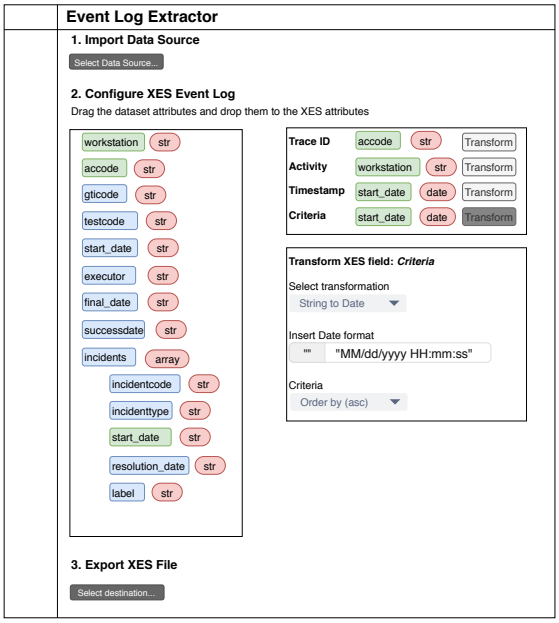

Fig. 4. Prototype of UI for transformation.

5 Lessons learned

This paper presented an industrial approach to cover the extraction of data from complex semi-unstructured databases for the generation of an event log that can be used by process-mining tools. The solution includes a DSL and a prototype that enable non-expert users to describe how event logs can be extracted from NoSQL sources. The solution has been applied to a real case study based on the aircraft-assembly process, thereby showing the applicability of our proposal to a real scenario after the discovery processes using the extracted data. The

[8] ProM: http://www.processmining.org/prom/start

proposal has been also supported by the implementation of an instrument which is connected to process-mining tools.

The following important lessons have been learned in the development of the approach: *i*) **extraction of event logs in IoT scenarios.** The complexity of the data in IoT scenarios increases the complexity in the extraction of event logs that require a intensive analysis; *ii*) **massive data production and processing.** An environment where data is continuously being generated could require the integration with Big Data architectures; *iii*) **the selection of case analysis useful for the organization.** The selection of the case analysis and the activity filtering criteria can influence in the results of the selected case, therefore, it is not always an easy task since it involves a great pre-analysis of the data and a huge understanding of the organization, and ; *iv*) **risk factors of discovery spaghetti-like processes.** Depending on the data and the case of study the discovery process can retrieve a spaghetti-like process which increase the complexity of analysis of the discovery process.

Acknowledgement

This work has been partially funded by the Ministry of Science and Technology of Spain through ECLIPSE (RTI2018-094283-B-C33), the Junta de Andalucia via the PIRAMIDE and METAMORFOSIS projects, and the European Regional Development Fund (ERDF/FEDER). The authors would like to thank the Cátedra de Telefónica "Inteligencia en la Red" of the Universidad de Sevilla for its support.

References

1. IEEE Standard for eXtensible Event Stream (XES) for Achieving Interoperability in Event Logs and Event Streams. IEEE Std 1849-2016 pp. 1–50 (Nov 2016). https://doi.org/10.1109/IEEESTD.2016.7740858
2. van der Aalst, W.M.P.: Extracting event data from databases to unleash process mining. In: BPM - Driving Innovation in a Digital World, pp. 105–128 (2015)
3. van der Aalst, W.M.P.: Process Mining - Data Science in Action, Second Edition. Springer (2016)
4. van der Aalst, W.M.P., Weijters, T., Maruster, L.: Workflow mining: Discovering process models from event logs. IEEE Trans. Knowl. Data Eng. **16**(9), 1128–1142 (2004)
5. Banziger, R., Basukoski, A., Chaussalet, T.J.: Discovering business processes in CRM systems by leveraging unstructured text data. In: 20th IEEE International Conference on High Performance Computing and Communications; Exeter, United Kingdom, June 28-30, 2018. pp. 1571–1577 (2018)
6. Calvanese, D., Kalayci, T.E., Montali, M., Santoso, A.: OBDA for log extraction in process mining. In: Reasoning Web. Semantic Interoperability on the Web - 13th International Summer School 2017, London, UK, July 7-11, 2017, Tutorial Lectures. pp. 292–345 (2017)
7. Calvanese, D., Montali, M., Syamsiyah, A., van der Aalst, W.M.P.: Ontology-driven extraction of event logs from relational databases. In: Business Process Management Workshops - BPM 2015, 13th International Workshops, Innsbruck, Austria, August 31 - September 3, 2015, Revised Papers. pp. 140–153 (2015)

8. Dumas, M., Rosa, M.L., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management, Second Edition. Springer (2018). https://doi.org/10.1007/978-3-662-56509-4, https://doi.org/10.1007/978-3-662-56509-4
9. Gómez-López, M.T., Reina-Quintero, A.M., Parody, L., Pérez-Álvarez, J.M., Reichert, M.: An architecture for querying business process, business process instances, and business data models. In: Business Process Management Workshops
10. Günther, C.W., van der Aalst, W.M.P.: A generic import framework for process event logs. In: Business Process Management Workshops, BPM 2006 International Workshops, BPD, BPI, ENEI, GPWW, DPM, semantics4ws, Vienna, Austria, September 4-7, 2006, Proceedings. pp. 81–92 (2006)
11. Gupta, M., Sureka, A.: Nirikshan: mining bug report history for discovering process maps, inefficiencies and inconsistencies. In: 7th India Software Engineering Conference, Chennai, ISEC '14, Chennai, India - February 19 - 21, 2014. pp. 1:1–1:10 (2014)
12. Lee, I., Lee, K.: The Internet of Things (IoT): Applications, investments, and challenges for enterprises. Business Horizons **58**(4), 431–440 (2015). https://doi.org/10.1016/j.bushor.2015.03.008
13. Lira, R., Salas-Morales, J., de la Fuente, R., Fuentes, R., Sepúlveda, M., Arias, M., Herskovic, V., Munoz-Gama, J.: Tailored process feedback through process mining for surgical procedures in medical training: The central venous catheter case. In: Daniel, F., Sheng, Q.Z., Motahari, H. (eds.) Business Process Management Workshops. pp. 163–174. Springer International Publishing, Cham (2019)
14. de Murillas, E.G.L.: Process mining on databases: Extracting event data from real-life data sources (2019)
15. González López de Murillas, E., Reijers, H.A., van der Aalst, W.M.P.: Connecting databases with process mining: a meta model and toolset. Software & Systems Modeling (2018)
16. de Murillas, E.G.L., van der Aalst, W.M.P., Reijers, H.A.: Process mining on databases: Unearthing historical data from redo logs. In: Business Process Management - 13th International Conference, BPM 2015, Innsbruck, Austria, August 31 - September 3, 2015, Proceedings. pp. 367–385 (2015)
17. de Murillas, E.G.L., Reijers, H.A., van der Aalst, W.M.P.: Connecting databases with process mining: A meta model and toolset. In: Enterprise, Business-Process and Information Systems Modeling - 17th International Conference, BPMDS 2016, Ljubljana, Slovenia, June 13-14, 2016, Proceedings. pp. 231–249 (2016)
18. Tax, N., Sidorova, N., van der Aalst, W.M.P.: Discovering more precise process models from event logs by filtering out chaotic activities. J. Intell. Inf. Syst. **52**(1), 107–139 (2019)
19. Álvaro Valencia-Parra, Ramos-Gutiérrez, B., Ángel Jesús Varela-Vaca, Gómez-López, M.T.: (2019), http://www.idea.us.es/bpm2019/
20. Verbeek, H.M.W., Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P.: XES, XESame, and ProM 6. In: Information Systems Evolution - CAiSE Forum 2010, Hammamet, Tunisia, June 7-9, 2010, Selected Extended Papers. pp. 60–75 (2010)