Enabling Process Mining in Airbus Manufacturing

Extracting Event Logs and Discovering Processes from Complex Data

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

1 Introduction

Process mining helps organizations understand their business processes. It usually involves analyzing process event logs and discovering the process models behind them (de Murillas, 2019). Several approaches can be used to discover them (Banziger, Basukoski, & Chaussalet, 2019; Calvanese, Kalayci, Montali, & Santoso, 2017; Van Der Aalst, Weijters, & Maruster, 2004).

One of the challenges in process mining is related to *the ability to extract suitable events* (van der Aalst, 2016). Moreover, obtaining event logs produced by the systems is becoming more complex, as the systems tend to produce massive, distributed, heterogeneous, and complex data in new Internet-of-Things (IoT) and Cyber-Physical Systems (CPS) environments and to derive chaotic combinations of elements (Lira et al., 2019; Tax, Sidorova, & van der Aalst, 2019) without structured schema. These issues bring about an extra level of complexity to be managed by the current process mining solutions that involve the use of event logs because they focus primarily on structured homogeneous data.

The main issue this chapter addresses concerns how the complex data structures that are typically generated in IoT environments can be transformed to create an event log that the existing process-mining tools can manage. In this regard, the chapter provides a rich case on how to put process mining to practice in order to

Á. Valencia-Parra (✉) · B. Ramos-Gutiérrez · Á. J. Varela-Vaca · M. T. Gómez-López
Universidad de Sevilla, Seville, Spain
e-mail: avalencia@us.es; brgutierrez@us.es; ajvarela@us.es; maytegomez@us.es

A. García Bernal
Airbus Defence and Space, Seville, Spain
e-mail: antonio.garcia.bernal@airbus.com

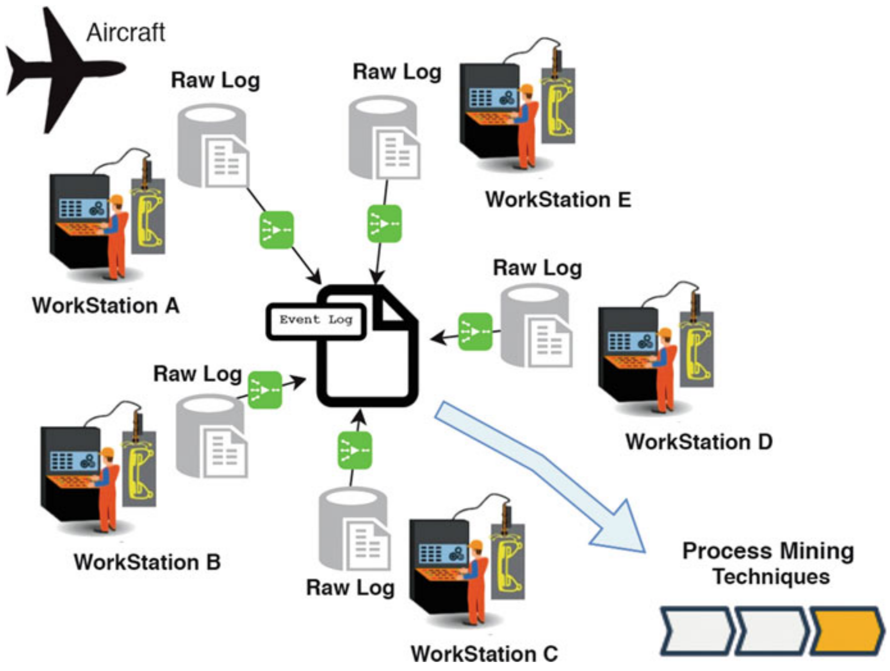


Fig. 1 Example of IoT scenario

deliver the strategy in a real-world organizational context (vom Brocke, Mendling, & Rosemann, 2021). Figure 1 shows a scenario of log extraction in the aeronautic industry that is 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 involve aggregating such complex data and extracting from them event logs in XES format, at which time process discovery techniques can be applied.

The objective of this work is to develop an approach that enables the extraction of event logs in XES format from complex data and to integrate this solution into a tool that non-experts can use to can benefit from the solution.

2 Situation Faced

This section describes the situation faced by the company.

2.1 Challenging Scenario

IoT and CPS environments are becoming highly relevant to organizations' (Lee & Lee, 2015) efforts to monitor their operations. However, a challenge concerns how to manage and analyze the recovered information so it is useful. The project *Clean Sky 2: A-24 One step beyond on automated testing technologies*, which was developed in collaboration with *Airbus Space & Defence*, inspired the notion of improving data extraction in IoT and CPS scenarios for the creation of event logs. The example is based on the assembly and testing processes of aircraft. Figure 1 illustrates a complex process that is executed in several of Airbus Space and Defence's locations without guidance from a Business Process Management System (BPMS) (Dumas, La Rosa, Mendling, & Reijers, 2013).

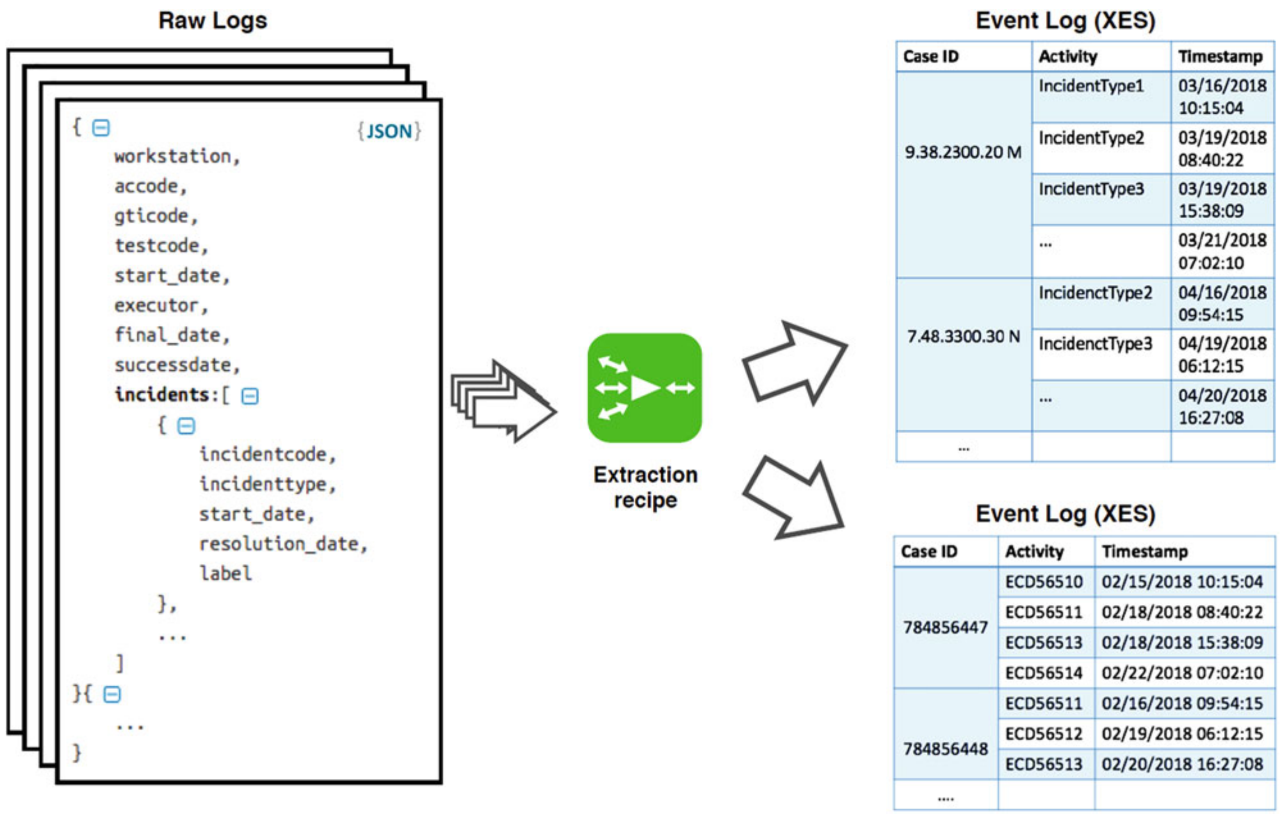
In this context, the main process in the organization is the assembly and testing of aircraft in accordance with an assembly chain of workstations. In this process, an aircraft with an identifier (*accode*) is located at a *workstation*, where a set of instructions is carried out (*gticode*). The instructions are validated by an operator (*executor*), and they begin and finish at a specific time (*start_date*, *final_date*). The moment when the test succeeds is specified (*successdate*). Each instruction might generate a set of incidents (*incidentcode*, *incidenttype*, *start_date*, *resolution_date*, and *label*).

The workstations, which are physically distributed, generate complex data structures depending on the tests that are executed. To generate logs that can be consumed by process-mining techniques, extraction of the logs from the structure in Fig. 2 must be defined. The figure shows a set of datasets formed of complex structures (lists and nested structures), but the event logs must fit the schema that is also shown in the figure.

Using this data, several trace logs can be generated that depend on the type of process that must be discovered, such as the evolution of the aircraft per workstation, the evolution of tests, or the life-cycle of incidents. These perspectives of the data can be used as the input from the process-discovery techniques to ascertain the actual process and to enable the analysis and improvement of the processes of assembly and testing.

2.2 Problems to Be Faced

The main problem to be faced in this scenario is the lack of a reliable approach to extracting event logs from complex semi-structured data. Many of the extant approaches were 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 database system's redo logs (de Murillas, 2019; de Murillas, van Der Aalst, & Reijers, 2015; van der Aalst, 2015). This approach has



```
{ ⊟
    workstation,
    accode,
    gticode,
    testcode,
    start_date,
    executor,
    final_date,
    successdate,
    incidents:[ ⊟
        { ⊟
            incidentcode,
            incidenttype,
            start_date,
            resolution_date,
            label
        },
        ...
    ]
}{ ⊟
    ...
}
```

Case ID	Activity	Timestamp
9.38.2300.20 M	IncidentType1	03/16/2018 10:15:04
	IncidentType2	03/19/2018 08:40:22
	IncidentType3	03/19/2018 15:38:09
	...	03/21/2018 07:02:10
7.48.3300.30 N	IncidenctType2	04/16/2018 09:54:15
	IncidenctType3	04/19/2018 06:12:15
	...	04/20/2018 16:27:08
...		

Case ID	Activity	Timestamp
784856447	ECD56510	02/15/2018 10:15:04
	ECD56511	02/18/2018 08:40:22
	ECD56513	02/18/2018 15:38:09
	ECD56514	02/22/2018 07:02:10
784856448	ECD56511	02/16/2018 09:54:15
	ECD56512	02/19/2018 06:12:15
	ECD56513	02/20/2018 16:27:08
....		

Fig. 2 Scenario of event log extraction

been supported by tools like XESame and PromImport plugin[1] (Günther & Van Der Aalst, 2006; Verbeek, Buijs, van Dongen, & van der Aalst, 2011), which transform information from relational databases into XES event logs. Data extraction from relational databases has been analyzed in general contexts that are not oriented toward process mining (Valencia-Parra, Varela-Vaca, López, & Ceravolo, 2019) but also from the perspective of ontology-based data access (Calvanese et al., 2017; Calvanese, Montali, Syamsiyah, & van der Aalst, 2016) by using metamodels as intermediaries (Gómez-López, Reina Quintero, Parody Núñez, Pérez Álvarez, & Reichert, 2018), supported by the *onProm* tool.[2]

Nonetheless, as the IoT scenario confirms, relational databases are not considered unique sources of events. Gupta and Sureka (2014) present an approach that shows how data is extracted from bug report histories, which are XML, JSON, or log files, although nested and complex structures are excluded. That work shows that the need to extract event logs from unstructured sources is a real issue, but there are no frameworks that facilitate it. In the same study, quasi-manual mapping of the report's attributes and those of the process is carried out, and these attributes are stored in a relational database from which the log of events will be extracted. In the context of Customer Relationship Management (CRM), Banziger et al. (2019) studied the possibility of transforming unstructured data into a log of events in XES log format (IEEE Computational Intelligence Society, 2016). In that case, the proposal involved a framework with which to discover business process models from semi-structured data that applies various preprocessing 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.

We conclude that extracting event logs in a usable format for process-discovery purposes is a key step in the context of business process management and that such extraction is not properly supported in IoT contexts with semi-structured data. As described in *The Six Core Elements of Business Process Management* (Rosemann & vom Brocke, 2010), among the core elements in this framework are the methods and information technology. While the first is intended to provide tools and techniques to enable activities along the process life-cycle, the second provides technical solutions for this purpose. The proposal described in this chapter is intended to help in the *process design and modeling* stage. Our proposal will help organizations by providing a methodological approach for extracting event logs from semi-structured data sources so they can discover the processes and inefficiencies in their manufacturing. This methodology is supported by an IT solution.

[1]PromImport plugin: http://www.promtools.org/promimport/

[2]onProm tool: https://onprom.inf.unibz.it/

Table 1 Characteristics of the raw log

Description	Values
Number of aircraft	15
Number of workstations	52
Number of GTIs	1110
Number of executions	9397
Number of incidents	3049

3 Action Taken

The first action was to depict the characteristics of the dataset (i.e., raw logs) for the real scenario. To address the problems of event log extraction, we developed a domain-specific language (DSL)[3] to enable XES event log extraction from complex semi-structured data. The DSL fulfills the main objective since it removes the need to transform complex data with nested and recursive structures when generating event logs in XES format. In short, the proposal is based on specifying the paths to the attributes to be employed as *case_id*, *activity_id*, *timestamp* and other optional parameters of XES event logs. The underlying framework then infers the transformations to be performed so as to reach the desired schema, in this case, an XES event log with the attributes indicated by the user.

3.1 Understanding the Data

To measure the scenario's complexity, the characteristics of the datasets (i.e., raw logs) that are used as the input in our approach must be ascertained. As shown in Table 1, the dataset contains *9367* sets of tests, provided by *52* workstations. *Fifteen* aircraft were tested in these workstations, with a total of *1110* tests (i.e., GTI). The tests can be executed more than once for each aircraft, which is why there are *9397* executions of these tests. These repetitions occur when a test execution is unsuccessful, thereby causing an incident, so there are *6049* incidents in the raw log. Some statistical data has been extracted (e.g., the average of GTI per aircraft) to complement this information and provide a greater level of detail. The dataset we use here is a subset of the real data obtained in the project *Clean Sky 2*; our dataset is a small part of the information generated and is made up of *9397* raw logs, filtered and modified.

Considering the IoT scenario, each workstation periodically provides a raw log of the execution of a test on an aircraft, which may also contain incidents.

[3]Our tool can be found at https://github.com/IDEA-Research-Group/ELE

3.2 Example of Event Log Extraction

Example 1.1 illustrates a use of the DSL in *the process of the assembly of aircraft for each of the workstations in which the tests are evaluated*. The aircraft production follows a process in which each part of the aircraft is assembled and tested in accordance with the engineers' design. Each step of the assembly process is performed in a workstation, where the resulting event log is made up of a set of traces with the following information: (i) ***case_id***: *accode*, the aircraft identifier; (ii) ***activity_id***: *workstation*, the workstation in which the test was conducted; and (iii) ***timestamp***: *start_date*, the date when the aircraft passed through the workstation.

Example 1.1 Piece of code for extracting event logs in XES format.

```
  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")
```

4 Results Achieved

This section describes how the results we obtained help to improve the aircraft-assembly process.

4.1 Extraction of Event Logs

A set of test cases is outlined first, followed by the results are discussed in the next subsections.

Test Case A. Process according to the workstation that executes the test This test case has been described in Sect. 3.1 (cf. Example 1.1).

Test Case B. Process according to the GTI execution A set of test instructions must be carried out during the aircraft-assembly process. In this test case, the processes related to the tests applied to the aircraft (*gticode*) are extracted. Hence, the resulting event log has the form: (i) ***case_id***: *accode*, the aircraft identifier;

(ii) ***activity_id***: *gticode*; and (iii) ***timestamp***: *start_date*, the first time the test was applied to the aircraft.

The piece of code that allows this test case to be carried out is shown in Example 1.2.

Example 1.2 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")
```

Example 1.3 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")
```

Test Case C. Process according to the type of incident If the tests (*gticode*) report a set of incidents with an (*incidenttype*), the resulting event log must be in the following form: (i) ***case_id***: *gticode*, the test identifier; (ii) ***activity_id***: *incidents. incidenttype*; and (iii) ***timestamp***: *start_date*, the first time this type of incident occurred.

The piece of code that corresponds to this extraction is shown in Example 1.3.

4.2 Analysis of the Extracted Event Logs

Once the event logs (i.e., XES files) are obtained, certain studies on the data that they contain can be performed to discover the processes. Table 2 describes the features

Table 2 Extracted logs features

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 ms
Mean case duration	76.1 days	67.1 days	20.8 days
Activities' minimal frequency	1	1	16
Activities' maximal frequency	15	15	424
Activities' median frequency	6	3	108
Activities' mean frequency	7.1	5.2	177.7
Activities' standard deviation	3.45	4.58	144.3

attained for each example of data extraction. A process-discovery tool, Disco,[4] was employed to analyze these logs.

The resulting processes discovered are shown in Fig. 3. These processes help to clarify and identify information that could not be analyzed manually. For instance, the process for *Test Case A* helps to clarify the flow of the aircraft through the various workstations, so we can see how certain workstations can perform tests in parallel with other workstations. *Test Case C* helps to clarify the flow of the incidents by showing how the incidents evolve.

4.3 Effects of Actions Taken

The event logs and the discovered process help to clarify 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. Some workstations have assumable transition values between them (hours or days) while in others the time the aircraft is there is always 0 s. Therefore, we can assume that they are intermediate states in which nothing is done. There are also workstations in which the aircraft has been only once or not at all. The time spent in the workstations ranges from 0 ms to 55 days. Knowing the average time aircraft spend in each workstation and which workstations aircraft of the same type must pass through, experts can know whether deviations are occurring and compare the current state to the desired state.

Test Case B This case helped experts to see the sequence of tests that must be carried out and the time that must be spent on each one. It also allowed the experts to

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

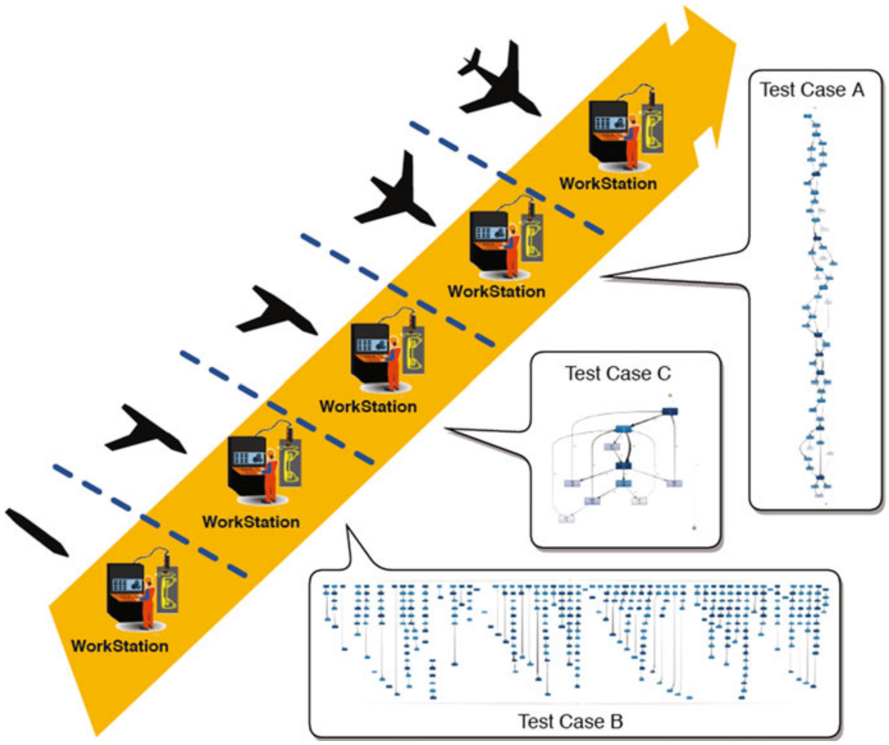


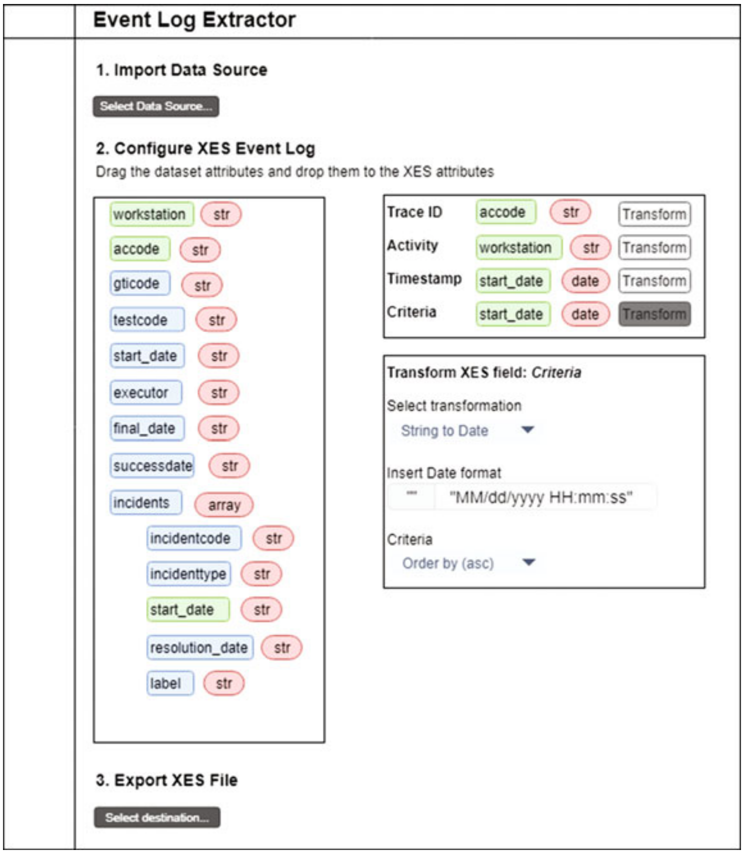
Fig. 3 Result of the process discovery for the extraction of each test case

make decisions about improvements that could be made to the tests performed at the workstations.

Test Case C The Discovery of the incident process by type helps experts see which types of incidents are most likely to occur after another incident occurred. Discovery of the incident process by type helps them see the average time that is likely to be required to resolve the same incident and how it will affect the entire assembly and testing process. The most frequent type of test is on the first incident and the last one. Other issues are detected, such as incidents that occur after a shift change of operators, which occurs in more than 30% of cases. These incidents are always of two: *Aircraft Configuration Failure* or *Test Edition Failure*.

4.4 Tool and Technical Details

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


Event Log Extractor

1. Import Data Source

Select Data Source...

2. Configure XES Event Log

Drag the dataset attributes and drop them to the XES attributes

workstation	str
accode	str
gticode	str
testcode	str
start_date	str
executor	str
final_date	str
successdate	str
incidents	array
incidentcode	str
incidenttype	str
start_date	str
resolution_date	str
label	str

Trace ID	accode	str	Transform
Activity	workstation	str	Transform
Timestamp	start_date	date	Transform
Criteria	start_date	date	Transform

Transform XES field: *Criteria*

Select transformation: String to Date

Insert Date format: "" "MM/dd/yyyy HH:mm:ss"

Criteria: Order by (asc)

3. Export XES File

Select destination...

Fig. 4 Prototype of UI for transformation

application (cf., Fig. 4) in which the transformation can be defined in a user-friendly way; and (3) a connector that allows the generated XES logs to feed the ProM™ (see footnote 2) tool automatically during the discovery of process models. All the resources that have been were employed in this work are freely available (see footnote 3). The raw log used in the paper was stored in a MongoDB database, and the extraction recipes were implemented in the DSL language, based on Scala, and developed using a model-driven engineering approach. The current implementation of the connector uses only *Inductive Miner* (Tax et al., 2019), but other tools, such as Disco from Fluxicon, have also been employed to illustrate the discovery process.

5 Lessons Learned

This chapter presents an industrial approach to 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 extract event logs from NoSQL sources. The solution was applied to a real case study based on the aircraft-assembly process, thereby showing the applicability of our proposal to a real scenario after using the extracted data for process discovery. The proposed approach was also supported by the implementation of an instrument that is connected to process-mining tools.

The proposed approach contributes to the need for methods and IT solutions for process design and modeling. Both the methodology, which consists of matching the dataset attributes with an XES event log template, and the IT solution, which frees users from dealing with flattening and aggregating operations, fulfill the objectives of these areas in *The Six Core Elements* framework (Rosemann & vom Brocke, 2010).

Four lessons have been learned in the development of the approach:

1. *Extraction of event logs in IoT and CPS scenarios*. The complexity of the data in IoT scenarios increases the complexity in the extraction of event logs that require intensive analysis.
2. *Massive data production and processing*. An environment in which data is continuously being generated could require integration with Big Data architectures.
3. *Selection of case analyses that are useful to the organization*. The selection of the case analysis and the activity-filtering criteria can influence the results of the selected case, so it is not always an easy task since it involves considerable pre-analysis of the data and a significant understanding of the organization.
4. *Risk factors of discovery of spaghetti-like processes*. Depending on the data and the case, the discovery process can retrieve a spaghetti-like process that increases the complexity of analyzing the discovery process.

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

References

Banziger, R., Basukoski, A., & Chaussalet, T. (2019). Discovering business processes in CRM systems by leveraging unstructured text data. In *Proceedings – HPCC/SmartCity/DSS 2018* (pp. 1571–1577). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/HPCC/SmartCity/DSS.2018.00257

Calvanese, D., Kalayci, T. E., Montali, M., & Santoso, A. (2017). OBDA for log extraction in process mining. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Springer (pp. 292–345). https://doi.org/10.1007/978-3-319-61033-7_9

Calvanese, D., Montali, M., Syamsiyah, A., & van der Aalst, W. M. P. (2016). Ontology-driven extraction of event logs from relational databases. In *Lecture Notes in Business Information Processing* (pp. 140–153). Springer. https://doi.org/10.1007/978-3-319-42887-1_12

de Murillas, E. G. L. (2019). *Process mining on databases: Extracting event data from real-life data sources*. Eindhoven: Technische Universiteit Eindhoven.

de Murillas, E. G. L., van Der Aalst, W. M. P., & Reijers, H. A. (2015). Process mining on databases: Unearthing historical data from redo logs. In *Lecture Notes in Computer Science* (pp. 367–385). New York: Springer. https://doi.org/10.1007/978-3-319-23063-4_25

Dumas, M., La Rosa, M., Mendling, J., & Reijers, H. (2013). *Fundamentals of business process management, fundamentals of business process management*. Berlin: Springer. https://doi.org/10.1007/978-3-642-33143-5.

Gómez-López, M. T., Reina Quintero, A. M., Parody Núñez, M. L., Pérez Álvarez, J. M., & Reichert, M. (2018). An architecture for querying business process, business process instances, and business data models. In *Lecture Notes in Business Information Processing* (pp. 757–769). Springer. https://doi.org/10.1007/978-3-319-74030-0_60

Günther, C. W., & Van Der Aalst, W. M. P. (2006). A generic import framework for process event logs. In *Lecture Notes in Computer Science* (pp. 81–92). Springer. https://doi.org/10.1007/11837862_10

Gupta, M., & Sureka, A. (2014). Nirikshan: Mining bug report history for discovering process maps, inefficiencies and inconsistencies. In *ACM International Conference Proceeding Series*. Association for Computing Machinery. https://doi.org/10.1145/2590748.2590749

IEEE Computational Intelligence Society. (2016). Standard for eXtensible Event Stream (XES) for achieving interoperability in event logs and event streams. In *Proceedings of the IEEE* (pp. 1–50). https://doi.org/10.1109/IEEESTD.2016.7740858

Lee, I., & Lee, K. (2015). The Internet of Things (IoT): Applications, investments, and challenges for enterprises. *Business Horizons, 58*(4), 431–440. https://doi.org/10.1016/j.bushor.2015.03.008.

Lira, R., Salas-Morales, J., Leiva, L., de la Fuente, R., Fuentes, R., & Delfino, A., et al. (2019). Tailored process feedback through process mining for surgical procedures in medical training: The central venous catheter case. In *Lecture Notes in Business Information Processing* (pp. 163–174). Springer. https://doi.org/10.1007/978-3-030-11641-5_13

Rosemann, M., & vom Brocke, J. (2010). The six core elements of business process management. In *Handbook on business process management* (pp. 105–122). https://doi.org/10.1007/978-3-642-00416-2

Tax, N., Sidorova, N., & van der Aalst, W. M. P. (2019). Discovering more precise process models from event logs by filtering out chaotic activities. *Journal of Intelligent Information Systems, 52* (1), 107–139. https://doi.org/10.1007/s10844-018-0507-6.

Valencia-Parra, Á., Varela-Vaca, Á. J., López, M. T. G., & Ceravolo, P. (2019). CHAMALEON: Framework to improve Data Wrangling with Complex Data. In *ICIS 2019 proceedings*. Retrieved November 14, 2019, from https://aisel.aisnet.org/icis2019/data_science/data_science/16

van der Aalst, W. M. P. (2015). *Extracting event data from databases to unleash process mining* (pp. 105–128). Cham: Springer. https://doi.org/10.1007/978-3-319-14430-6_8.

van der Aalst, W. (2016). Data science in action. In *Process mining* (pp. 3–23). Berlin: Springer. https://doi.org/10.1007/978-3-662-49851-4_1

Van Der Aalst, W., Weijters, T., & Maruster, L. (2004). Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering, 16*(9), 1128–1142. https://doi.org/10.1109/TKDE.2004.47.

Verbeek, H. M. W., Buijs, J. C. A. M., van Dongen, B. F., & van der Aalst, W. M. P. (2011). XES, XESame, and ProM 6. In *Lecture notes in business information processing* (pp. 60–75). New York: Springer. https://doi.org/10.1007/978-3-642-17722-4_5.

vom Brocke, J., Mendling, J., & Rosemann, M. (2021). Planning and scoping business process management projects and programs with the BPM Billboard. In J. vom Brocke, J. Mendling, & M. Rosemann (Eds.), *BPM cases. Digital innovation and business transformation in practice* (Vol. 2). New York: Springer.