ADVANCED REVIEW

Extraction, Correlation, and Abstraction of Event Data for Process Mining

Kiarash Diba[1] | **Kimon Batoulis**[1] | **Matthias Weidlich**[2] | **Mathias Weske**[1]

[1]Hasso Plattner Institute, University of Potsdam, Potsdam, Germany

[2]Humboldt-Universität zu Berlin, Berlin, Germany

Correspondence
Kiarash Diba, Business Process Technology, Hasso Plattner Institute, University of Potsdam , Potsdam, Germany
Email: kiarash.diba@hpi.de

Process mining provides a rich set of techniques to discover valuable knowledge on business processes based on data that was recorded in different types of information systems. It enables analysis of end-to-end processes to facilitate process re-engineering and process improvement. Process mining techniques rely on the availability of data in the form of event logs. In order to enable process mining in diverse environments, the recorded data needs to be located and transformed to event logs. The journey from raw data to event logs suitable for process mining can be addressed by a variety of methods and techniques, which are the focus of this article. In particular, proposed techniques in the literature to support the creation of event logs from raw data are reviewed and classified. This includes techniques for identification and extraction of the required event data from diverse sources as well as their correlation and abstraction.

KEYWORDS
process mining, event data extraction, event correlation, event abstraction

1 | INTRODUCTION

Business process management is concerned with designing, monitoring, executing, and evaluating business processes in organizations. The main artifact is a process model, which is an abstract representation of a business process (Weske, 2019). While research in business process management has for a long time centered around process models and

languages to express them, recently, the focus has shifted to process mining. Process mining uses execution data in the form of events, recorded during process executions, which may be exploited in several ways (van der Aalst, 2016). Process mining subsumes automatic discovery of process models based on recorded behavior of the process. This recorded behavior can be compared to designed models to find process deviations and their root causes (Carmona et al., 2018). A plethora of process mining techniques have been developed in recent years; the increasing uptake in industry shows that valuable insights across different business domains are provided by process mining.

Despite the maturity of the individual process mining techniques, in process mining projects considerable resources have to be allocated for the extraction and preparation of event data, before the actual analysis can even start. This is also indicated in the process mining manifesto (van der Aalst et al., 2012) which states that "finding, merging, and cleaning event data" remains a challenge for application of process mining techniques.

In complex application scenarios in large companies, the data required for process mining resides in various databases and information systems used by the organization. Most of these do not record the execution data in a process-centric way, so that the data is not immediately ready for process mining. Examples of these information systems include Enterprise Resource Planning systems, Customer Relationship Management systems, and other legacy systems. Data in these systems can be in various forms and formats, so that it incurs a substantial effort to locate and transform these event data to the event log format required by process mining techniques.

Techniques for event log preparation can be organized in three groups: techniques for event data extraction, correlation, and abstraction. Event data extraction deals with techniques to identify data elements that characterize events from heterogeneous data sources. Event correlation helps us to group the data elements that relate to a single process instance. Finally, event abstraction looks at the mapping of data elements to events that correspond to activity executions in a business process. Techniques in this area assign semantics to sets of data elements by defining how they can jointly be interpreted as the execution of a business process activity.

These event log preparation techniques are sometimes overlapping in their scope and may also adopt similar models and algorithms. Moreover, the discussed techniques partially incorporate common pre-processing techniques, such as filtering of data elements. However, we perceive the distinction of techniques for event data extraction, correlation, and abstraction to be suitable to give an overview of the field as these are key steps for building meaningful event logs.

The paper is structured as follows. Background and terminology are provided in Section 2. Section 3 focuses on the extraction of event data from a variety of data sources. The challenge of correlating extracted data elements is discussed in Section 4. Section 5 elaborates techniques for the abstraction of data elements to events that denote activity executions, before Section 6 concludes the paper.

2 | BACKGROUND

This section elaborates on the problems that emerge when striving for the preparation of an event log. Event log elements are explained with regard to process elements, thereby connecting concepts of the world of data to the process world. An example is provided, before an essential event log format and related standards are discussed.

2.1 | Terminology

Process mining provides a rich set of techniques and algorithms for process discovery, conformance checking, and enhancement (van der Aalst, 2016). Here, process discovery aims at the creation of a process model automatically

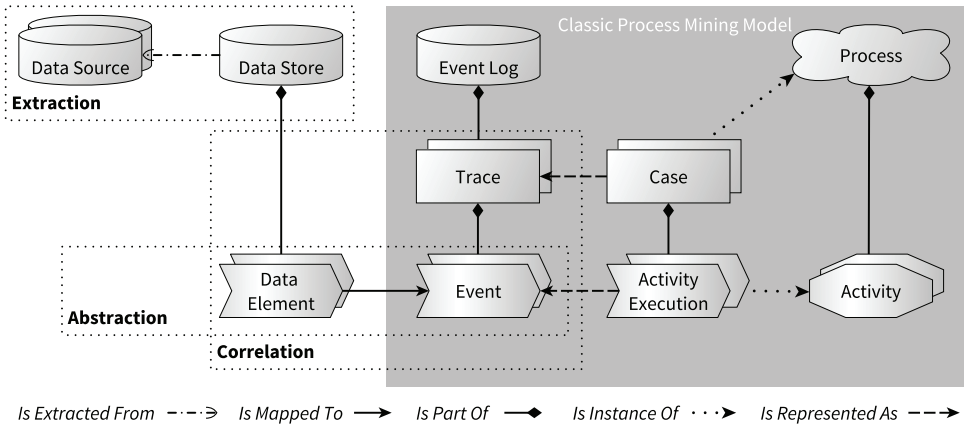

FIGURE 1 Overview of the main concepts related to data and events in process mining, adapted from (Carmona et al., 2018).

from the data recorded during process execution (Augusto et al., 2019). Conformance checking, in turn, takes as input the recorded data and a process model, assessing their consistency and providing diagnostic results on their deviations (Carmona et al., 2018). Process enhancement typically enriches a given process model based on the recorded data (de Leoni and Mannhardt, 2019; Depaire and Martin, 2019), thereby providing a more complete process representation as the basis for quantitative or qualitative analysis. For all these use cases, however, most algorithms operate on event logs, consisting of a sequential list (aka traces) of events. In fact, it is the core assumption of most process mining techniques that events are provided in an event log format that is suitable for processing.

Investigating real-world application scenarios of process mining, the availability of suitable event logs is one of the main obstacles for widespread adoption of process mining technology. In practice, a typical process mining project spends considerable resources in event data extraction, correlation, and abstraction, before process mining activities can even be started.

To provide a conceptual basis of the topics addressed in this paper, Figure 1 defines the main concepts related to data and events in process mining. Studying concrete business processes in organizational and technical environments, we first note that there are many instances of such a process, which are called cases. For instance, each case in an ordering process deals with a specific customer and the articles ordered. Hence, a case is an instance of the process, as represented by the dotted arc in Figure 1. The activities performed in the context of a case are called activity executions, and each activity execution is part of a case. Activity executions are represented by events, and events participate in traces. A trace is a sequence of all events that belong to one case and a log contains a set of traces for a specific process.

In many real-world scenarios, however, events and traces are not readily available. Before a process mining technique can be applied, event logs have to be developed from existing data sources. Aspects related to data and their relationship to events are represented by the concepts of a data source, data store, and data element. Data elements that are relevant for the project at hand need to be extracted from one or multiple data sources; this information is stored in data stores. Data elements can be regarded as tuples in a relational database, rows in a spreadsheet, or any other data format.

Event data are data elements that are recorded as a result of occurrence of events. Yet, to use this data for

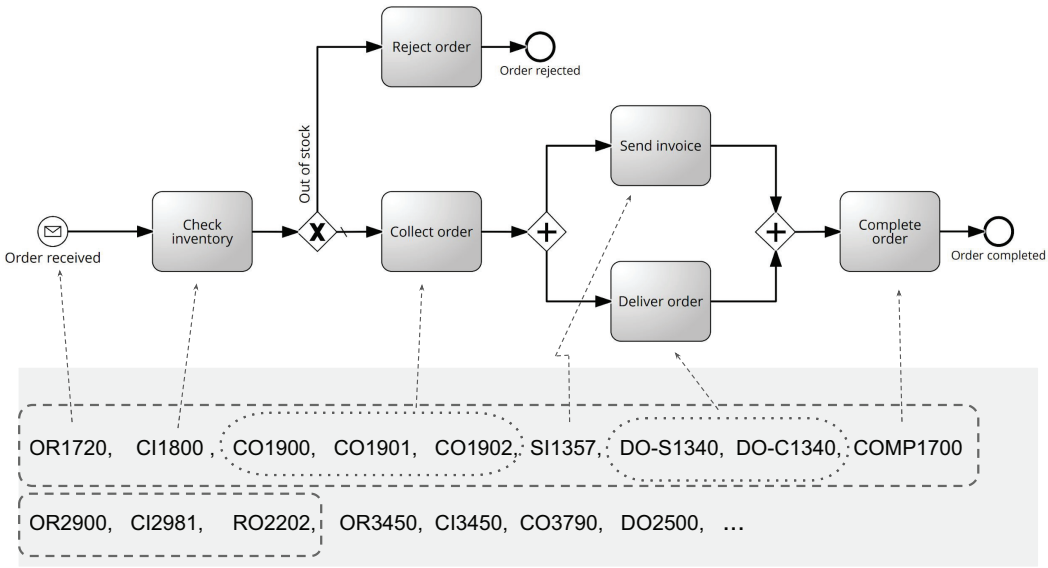

FIGURE 2 A simple order handling process model and a collection of event data extracted for this process. The dashed rectangles indicate which elements are correlated as they refer to the same case. The dotted ovals illustrate the abstraction of sets of event data to events that denote activity executions, while the respective activities are highlighted by the the dotted arrows.

process mining, relevant data elements have to be extracted and event data need to be identified. They then need to be correlated to their respective cases. These data elements might be recorded in detail and thus be low-level (several data elements are recorded which together reflect the execution of a single activity). Therefore, mapping them to such events that, by their definition in the process mining context, denote activity executions requires an abstraction level. Event abstraction, therefore, aggregates low-level event data elements into higher level events that represent the execution of activities. However, note that in some cases, the extracted data elements may already be at the desired level of abstraction, such that they can be used for process mining right away. To summarize, techniques to extract, correlate, and abstract event data are required, which are characterized as follows:

Event Data Extraction is the derivation of data elements, jointly captured in a data store, from data sources. The derived event data elements are used to define events and, thereby, traces.

Event Correlation groups event data that belong to specific cases. The event data that can be correlated to a case will be used in the definition of the events of the same trace.

Event Abstraction maps event data to events representing activity executions. As such, it enables the interpretation of these data elements in terms of specific activity executions.

For each type of technique, Section 3, Section 4, and Section 5 will later give an overview of the existing approaches. In doing so, the relevant literature is summarized, pointing out the expected inputs and outputs of the different approaches, the utilized pieces of information (such as domain knowledge), as well as their basic assumptions and limitations. Each section will close with a table that summarizes the approaches, thereby giving a concise overview of the state-of-the-art of the respective solutions.

Order id	Event id	Activity	Timestamp
1	OR1720	Order received	02/01/2019:09.00
1	CI1800	Check inventory	02/01/2019:09.10
1	CO19000	Collect order	02/01/2019:09.30
1	SI1357	Send invoice	02/01/2019:09.50
1	DO1340	Deliver order	02/01/2019:10.30
1	COMP1700	Complete order	02/01/2019:11.15
2	OR2900	Order received	02/01/2019:10.05
2	CI2981	Check inventory	02/01/2019:10.15
2	RO2202	Reject order	02/01/2019:10.45

TABLE 1 An excerpt of an event log for the order handling process.

2.2 | Illustrating Example

To illustrate the need for techniques to support the creation of event logs, consider the order handling process of a supplier company shown in Figure 2. After receiving an order, the inventory is checked for availability of the ordered goods. If the goods are out of stock, the order is rejected and the process terminates. Otherwise, the order is collected before it is sent for delivery and, concurrently, an invoice is sent to customer. Finally, the order is completed.

The process model is now related to the concepts introduced in Figure 1. The process shown consists of activities. A specific instance of the process (an individual order in this example) is a case. Each performed activity – like checking the inventory for a specific order – is an activity execution.

To analyze the concrete behavior of the business process on the basis of the data generated during process execution, process mining techniques can be used. These techniques require an event log as input, in which events reflect the activity executions. The events are correlated based on the case, for which they indicate an activity execution, so that they form a trace. An event log is a collection of traces (multiple executions of the processes for different cases) recorded in a specific period of time. Table 1 shows an excerpt of an event log for the order handling process.

Depending on the information systems used to support the order handling process, data might not be readily available in the format of Table 1. It can be in multiple formats, and it might reside in different data sources. Regardless of the information systems, the execution of activities leave a digital footprint behind. In typical application scenarios, however, this data needs to be located and transformed into an event log, before process mining becomes applicable.

To develop an event log as shown in Table 1, relevant event data needs to be obtained. That is, the respective data elements need to be located in data sources and extracted into a data store. Techniques to support this extraction step are reviewed in Section 3.

Once data elements have been extracted, they need to be correlated according to cases. For example, the data elements OR1720 and CI1800 have to be analyzed to identify that they both relate to the same case, i.e., the processing of the same order. For elements OR2900 and CI2981, however, it must be established they belong to a different case. In general, correlation is not a trivial task. For example, the order id attribute in Table 1 may not be available in the data. We review techniques to correlate data elements in Section 4.

While mapping relevant data elements to events, further issues may stem from the granularity at which the data is recorded. Potentially, data elements need to be grouped and abstracted to reflect an event in the process mining

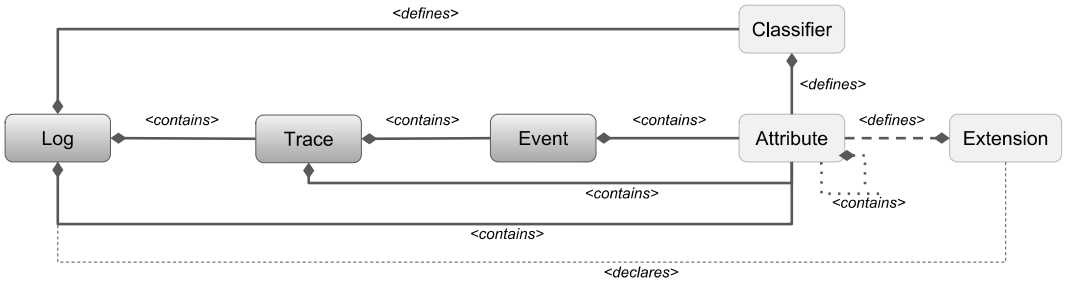

FIGURE 3 Main elements of the XES metamodel (Günther and Verbeek, 2014).

sense, i.e., an execution of a specific activity. In our example, multiple data elements are recorded for the *Collect order* activity, which together reflect the execution of the respective activity. The problem of event abstraction is discussed in Section 5.

2.3 | Standards for Event Logs

The input for process mining techniques is commonly an event log. As detailed above, an event log is a collection of traces and each trace is a collection of ordered events. Each event in a trace indicates the execution of one activity for a case of the process, and each trace contains all the events recorded sequentially for one single case.

To unify the input format of process mining, a standard XML-based format, MXML (Mining XML) was introduced (van Dongen and van der Aalst, 2005). To facilitate mapping between logs of different systems to MXML a meta model was also provided. An MXML log named a *WorkflowLog* consists of a *Process* element, a *Data* element used for storing additional attributes, and a *Source* element containing information about the information system used for recording data. The process element is composed of several *ProcessInstance* entries which contain several *AuditTrailEntry* elements. Each AuditTrailEntry is an event and contains a *WorkflowModelElement*, *EventType*, and a number of optional attributes such as timestamp and originator of the event. The WorkflowModelElement is the activity in the process which the event refers to and the EventType reflects the type of the event based on the transactional model of activity life-cycle (scheduled, started, completed, etc.). While the terminology adopted by MXML differs from the one introduced here, we note that it supports the classic process mining model given in Figure 1.

The MXML format had been widely used as the input of process mining techniques before the new standard, XES (eXtensible Event Stream) was introduced (IEEE (XES) Working Group, 2016; Verbeek et al., 2010). This new standard was developed to address a few problems encountered with MXML logs (e.g. unclear semantic of additional attributes). The new format was adopted by the IEEE Task Force on Process Mining as the standard format for event logs. Figure 3 shows the XES meta model. In an XES event log, *log* is the root element containing several *traces*. each trace consist of several *events*. Log, traces and events have a number of *attributes*. The semantic of an attribute is defined by its *extension*. There are a number of standard extensions for common attributes such as timestamp, life-cycle, and resource. Additional extensions and their semantics can be defined by the user in specific applications. A *classifier* can be used to assign identity to each event (e.g. by a combination of event name and timestamp). XES is the current standard for process mining event logs, and many tools and techniques require an XES event log as input. OpenXES [1] is the reference implementation of XES as a Java library. We note that XES conforms to the classic process

[1] http://www.xes-standard.org/openxes/start

mining model in Figure 1.

3 | EVENT DATA EXTRACTION

The data that is required for process mining can reside in a variety of sources. Depending on the information systems and the databases in use, this data might have different formats and characteristics. It is often the case that valuable data resides in data sources such as relational databases where it is not recorded with a process in mind and therefore, it is not process-centric (i.e. events and traces are not explicitly recorded). The challenge is to identify relevant event data among a pool of data in these systems, extract and transform them to the process-centric event log format required by process mining techniques. This requires a great amount of domain knowledge and it mostly involves manual and ad-hoc solutions. This transformation gives rise to a number of challenges which affect the quality of the resulting event logs. Specifically, the following challenges exist:

Challenge 1: Finding and locating the relevant event data;

Challenge 2: Creating a large number of queries to extract the data, which requires knowledge of the underlying database;

Challenge 3: Historical data might not be recorded explicitly;

Challenge 4: Dealing with one-to-many and many-to-many relationships between data elements. Forcing the data to event logs with a flat structure and a clear case notion, causes the problem of data convergence (where one event relates to multiple cases) and divergence (multiple records of the same event relate to one case);

Challenge 5: Avoiding the loss of information on the data perspective (data objects, classes, and relationships). In the remainder of this section we discuss approaches focusing on one or more of the above-mentioned challenges. First, the works focusing mainly on identifying and extracting event data (challenges 1 and 2) are introduced (Section 3.1). Then, the works tackling the lack of recorded history (challenge 3) are presented (Section 3.2), before discussing the approaches with the main focus on the challenges of data relationships and the data perspective (challenges 4 and 5) (Section 3.3).

3.1 | Event Data Identification and Extraction

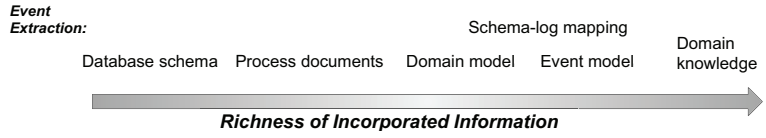

FIGURE 4 The spectrum of incorporated information for event extraction techniques

Figure 4 provides a brief overview on the information used by the approaches to assist the identification and extraction of event data from databases. Information on the data sources, relevant events, and the domain of interest facilitates the identification and extraction of event data from non-event-based data sources. A *database schema* describes how data elements are stored and how they are related to each other. It helps in identifying event data by acquiring a better understanding of the database and its structure. *Process-related documents* and policies which give information on the processes in organizations, if available, can be exploited as a source of information about the underlying process, data classes and elements involved in the context of the process under analysis. Using a *domain*

model that provides information on which classes of data are involved in the domain or the process under analysis, further assists identification of the relevant event data to be extracted. Defining an *event model*, which identifies types of events and their relation to the classes in the domain model, on top of the domain model facilitates event data identification even more. These concepts together with a *mapping* specification to the underlying database facilitates the data extraction. In addition, *domain knowledge* is usually required to define these concepts and to identify the relevant event data in the database.

Jans and Soffer (2017) specify a set of decisions to be made, based on the domain knowledge, when creating an event log from raw data. Based on the three main elements of an XES event log—process, trace, and event—these decisions are classified into three categories: (i) Selecting the process to analyze, (ii) Creating a specific view on the process (selecting the process instance), and (iii) Selecting the relevant events. Based on these decisions a procedure to provide guidance for decision making is suggested as follows. *Set business goals*, *collect information on the data source*, *select process instance and its level of granularity* (e.g. order vs order line) *based on business goals*, *select relevant events*, and *list relevant attributes*. A variety of tools and techniques have been proposed in recent years in academic literature to support this extraction phase towards a more generic and automated solution.

3.1.1 | The Process Lexicon

To support finding and locating (Challenge 1) and extracting relevant event data that match the process under analysis and its related activities, Wang et al. (2012) introduce an approach based on text mining of organization-specific **process documents** and industry standard reference models. Using text mining on documents such as process models, process documents, and policy manuals, they identify a *process lexicon* containing a dictionary of process components such as activities and data items. Based on this lexicon, tables and attributes of databases are scored and ranked for their relevance to the process lexicon using a similarity function. This ranked data source is now called a *process-aware database* which is used by the domain expert to extract the relevant data. The approach can be helpful to domain experts, when relevant documents are available and the taxonomy used in the database matches the process taxonomy. The approach is improved and evaluated in (Li et al., 2015).

3.1.2 | ProM Import Framework

Günther and van der Aalst (2006a) developed one of the first approaches towards a generic and automated solution for transforming event logs of a variety of process-aware information systems (PAISs) to the (by the time) standard format, MXML. ProM Import Framework is implemented in ProM process mining framework[2] in an extensible plug-in architecture allowing development of adapters to any PAIS. The tool supports the transformation of event logs generated by a number of well-known workflow systems of that time (e.g. FLOW*er*, Staffware, etc.) and allows for extensions to other PAISs through the plug-in architecture. Although the tool provides an automated environment for converting the event logs of PAISs to MXML, it is limited to a number of PAISs and new plug-ins must be written (in Java) for system not directly supported. This requires the analyst to have programming skills and detailed knowledge of the system. In addition, restricting the framework to PAISs, the assumption is that events are explicitly recorded by the source system. Therefore, the extraction of event data scattered through various sources and tables in a non-process-aware system is not supported.

[2] http://www.promtools.org/doku.php

3.1.3 | XESame, EVS Model Builder, and Eventifier

To eliminate restrictions and the programming need posed by the ProM Import Framework and to account for the newly adapted event log format XES, XESame was developed (Verbeek et al., 2010; Buijs, 2010). XESame lifts the assumption made by ProM Import Framework that some sort of event log already exists, thereby facilitating the event data extraction from databases (Challenge 1 and 2). XESame provides a graphical interface for domain experts to create a *conversion definition* consisting of a **domain model** and a **mapping** of database tables to the event log requirements. Afterwards, SQL queries are generated based on the specified mapping, event data is extracted from the data source, associated to traces, and the event log is created. The tool has been proved successful in many applications. However, the extensive effort and domain knowledge required for identification of relevant tables and attributes, and the definition of the mapping for conversion poses a challenge for large scale extractions.

One of the most common type of systems on which the application of process mining can be valuable is ERP systems, such as SAP, Oracle ERP, and Microsoft Dynamic. The wide usage and their support for end-to-end operational processes make them a potential source for process mining.

EVS Model Builder was developed by Ingvaldsen and Gulla (2007) to support the extraction of SAP transaction data. The extraction consists of four phases: First, the user has to provide a **meta-description** (in form of a UML class diagram) consisting of business objects (business entities e.g., user, sales order), event descriptions, and their relation based on the process to be analyzed. This meta-description assists in locating the relevant data. Then, the system extracts business objects and their relations by constructing SQL statements and executing them on the database. Afterwards, data elements and their relation to business objects are extracted. Finally, the process instances (cases) are identified by following dependency relationships of data elements and the data is transformed to an MXML event log. The meta-description is similar to the **mapping** concept in XESame. Although their approach provides automated SQL query generation and format conversion, defining the meta-description and locating data demands considerable effort and domain knowledge. However, they argue that this meta-description can be reused when the structure of SAP databases is consistent.

In (Rodrıguez et al., 2012) the identification and extraction of event data from operational databases is called *eventification*, and the tool supporting this process *eventifier*. Eventification consists of event identification (identifying and extracting event data in the database), event ordering, data association, and correlation. They assume that the process and activities which events relate to are defined by domain experts. For identifying event data in the database, they assume that a record in a relation between tables reflects an event, and introduce three *event identification patterns*. Single row, single event, where each record corresponds to exactly one event; single row, multiple events, where one record is the evidence for the occurrence of more than one event; multiple row, single event, where multiple records together reflect the occurrence of one event. The suitable pattern for the analysis should be selected carefully by a domain expert. Afterwards, events are ordered based on the timestamps found in the database. In the data association phase the attributes required are added to events. Afterwards, they correlate events through relations indicated in the attributes added in the data association part. One major assumption here is that all information needed for correlation is present in the attributes. Finally, the event collection is transformed to XES using OpenXES libraries.

3.1.4 | Ontology-Based Data Access Approaches

Ontology-Based Data Access (OBDA) approaches aim at extracting data using a high level conceptual view of the domain of interest expressed as an ontology, and linking this view to the **database schema**. Calvanese et al. (2016)

propose an approach for the extraction and creation of event logs for process mining based on OBDA. The conceptual schema of the domain of interest, the **domain model**, is defined as a UML class diagram by the user. It is then translated to and expressed in an ontology language, as the domain ontology. This domain ontology is then annotated by a domain expert based on the **event log ontology** which is constructed based on the XES metamodel. Afterwards, data is extracted automatically, by connecting the data from databases to the event log ontology. Decoupling the *design phase*, in which the user enriches the domain ontology with regard to the elements of the event log ontology, from the *data access phase*, where OBDA techniques are used to automatically extract data from the database to an XES event log, allows the user to focus only on the domain ontology and its annotation rather than the database schema and the actual extraction. This approach facilitates multi-perspective process mining due to the fact that different views on the data can be taken easily by adjusting the annotations of the domain ontology.

Onprom (Calvanese et al., 2017c,b) is a toolchain and data extraction methodology that builds on the idea of OBDA event data extraction. The methodology starts with the conceptual modeling step. In this step, two conceptual models need to be created, a **conceptual data model**, and a **mapping specification**. The conceptual data model or the domain ontology is created by the domain expert using a UML class diagram. The mapping specification connects the database schema to the domain ontology. These models form the *OBDA system* which allows abstracting from details of the information system and its database structure. The next step is to enrich the conceptual data model with **event data annotations** consisting of *case*, *event*, and *attribute annotation*. These annotations specify the class containing the case notion, occurrences of specific types of events from classes and relevant attributes for events, and provide the basis for extraction and correlation of event data to form event logs. In the third step event data is automatically extracted and an event log is created using the combination of mappings and annotation leveraging the OBDA technology and methods.

The Onprom toolchain has been implemented as a set of ProM plug-ins with a UML editor, annotation editor and a log extractor to create XES event logs from relational databases. Their approach provides a conceptual view that helps understanding the information system conceptually and enables automatic extraction of event data. The approach has been compared to a general ETL-like method in a context of a case study. For the ETL method, extensive knowledge of the underlying database was needed and creating views on the database was labor and time intensive and error-prone, and creating additional views required a complete iteration of the extraction and preparation effort. Their approach however, decouples the domain ontology from details of the information system, and by leveraging OBDA, reduces the effort needed. It also facilitates creating multiple views only by adjusting annotations of the domain ontology. However, the first and second step of the methodology are manual and require effort and knowledge of the domain and process to create and annotate the domain ontology manually. Additionally, the mapping specification is either created manually or semi-automatically using the OBDA framework *ontop* (Calvanese et al., 2017a). They also rely on ontop for the creation of the mapping between data and XES elements.

While XES is the standard for event logs and is supported by most of the process mining tools and techniques, it poses a number of restrictions regarding the structure and the type of event data. A number of alternative formats have been proposed which are discussed in the last part of this chapter. For the OBDA approach to lift the restriction of the method to XES format and to provide a more general approach regardless of the event log format used, in (Calvanese et al., 2018) the authors propose a second conceptual level called the *upper schema* for their OBDA framework. In case of process mining, this upper schema replaces the XES schema. This allows the analyst to define the upper schema in their desired format. The resulting framework named *2OBDA* extends the previous works by adding this second conceptual layer and a *conceptual transformation* of the domain ontology to the upper schema. This transformation is an ontology-to-ontology mapping which is performed in a similar way to mappings in previous approach. The tool is extended with a *transformation rule generator* which automatically maps the two conceptual models based

on the annotated domain ontology with regard to the upper ontology.

3.2 | Redo Logs for Event Extraction

It can be the case that historical data is not recorded in the database and the data is overwritten when a transaction is performed on the database (Challenge 3). In this case the data in the database only reflects the current state of the system and does not provide any historical data on the changes and the steps of processes.

van der Aalst (2015) proposes an approach to identify and extract event data from databases in which historical data is not recorded. Building on the idea that current values of database tables reflect the current state of the information systems and processes, it is suggested that changes in the database refer to events that change the state of the system or the process. Based on the assumption that these changes are recorded in the **redo logs** of databases, these logs can be used to identify events. Redo logs of relational databases therefore, can provide useful information on the occurrence of events. The approach starts by defining an **event model** which relates database changes to events. This event model enriches the **data model** of the database with possible event notations. After such an event model is provided (either by a domain expert or extracted and constructed from the redo logs) event data needs to be extracted and converted to an event log format with a clear case notion. This is done by *Scoping*, *binding*, and *classifying* event data and process instances.

Scoping selects the relevant event data for the purpose of analysis. The selection can be done based on the names of the events, a time period, or classes containing these events. Binding correlates events to process instances. This is done by manually identifying relevant tables and relations of the database for the selected case notion. Classifying is used to create multiple logs for different process models to facilitate process comparison. The approach conceptually defines the idea of creating an event model which relates databases to events using changes recorded in redo logs of such databases, which provides the conceptual basis for development of methods to facilitate event data extraction from relational databases.

The work by de Murillas et al. (2015) builds on this idea and enhances the conceptual approach with concrete techniques and a prototype implementation and evaluation. First, they extract relevant event data from redo logs, transforming and abstracting its records to events. Related attributes of events are transformed directly from redo log records or extracted from affected tables in the database. After creating a set of events, these events need to be correlated to form traces. Here, a **data model** is created from the database with the use of queries on tables, fields and primary and foreign keys in the **database schema**. This data model is then used to identify relations and correlating events based on the defined case notion for the trace. Their approach has been implemented as a proof of concept and evaluated for an Oracle DBMS. However, they argue that the approach is generic and can be extended to any database technology which guarantees the availability of redo logs with clear timestamps. One of the issues reported for their approach is the duplication of same events into different traces in the generated event log as a result of the algorithm used. This may cause problems for process mining techniques and leads to false statistics.

in (de Murillas et al., 2017b) the authors compared redo log event extraction techniques with other techniques such as XESame which they refer to as traditional approaches. It is indicated that redo log approaches allow more automation and generalization compared to traditional approaches, demanding less domain knowledge for selection of relevant data and less dependency on a specific database. In addition, data deleted from the database can be tracked and included in the analysis. The main challenge faced by these approaches is however, that special database privileges needed to configure and access redo logs which might not be available in every set up. Moreover, redo logs should be enabled in the database and archived (redo logs are overwritten by the DBMS when redo log files are full, however they can be archived to address this issue). These assumptions might not hold on many real life cases.

Nonetheless, if redo logs are available and configured correctly the approach proved to yield valuable results.

3.3 | Data- and Process-Centric Models

XES event logs are process-oriented and they restrict the data to fit this process-oriented frame. As mentioned before, this can lead to problems of data convergence and divergence, and loosing information on the data perspective. New models have been proposed for storing event data aiming to preserve the object-centric nature of data, and avoiding data convergence and divergence (Challenge 4 and 5), while facilitating multi-perspective process mining.

3.3.1 | Artifact-Centric Approaches

As mentioned before, ERP systems, such as SAP, Oracle ERP, and Microsoft Dynamic, provide valuable sources for potential process mining analysis. However, despite their implicit support for processes, their data is normally stored in object-centric relational databases. In SAP data objects usually refer to instances of certain business objects such as orders, invoices, users, etc. There are one-to-many and many-to-many relations between objects, and there is not a clear case notion relating these objects. Flattening data from these sources to an event log format can cause problems of convergence and divergence.

Artifact-centric approaches (Hull, 2008; Fahland et al., 2011; Popova et al., 2015) attempt to adjust the process view of event logs to that of underlying databases. Instead of forcing the data residing in different tables of ERP systems to be transformed into a process-oriented view with a clear case notion, they preserve the object-centric nature by considering artifacts (Business objects in ERP systems) as the case identifier. This way, flattening the event log to a single view with one case notion can be avoided.

Nooijen et al. (2012) propose the discovery of artifact life-cycle models for each artifact in an ERP system instead of a traditional process model. These life-cycle models describe the evolution of one business object throughout a specified time span. In order to discover these life-cycle models, three elements have to be discovered. The artifacts involved in the process, the schema for each artifact in the database (i.e. their attributes and relations), and finally the life-cycle model for each artifact. They start by (re)discovering the **database schema** (in case the available schema is incomplete, they suggest to use schema extraction techniques (Zhang et al., 2010; Ahmadi et al., 2009)). Then, they cluster the schema to discover one schema for each artifact using the k-means clustering technique. Next, a **schema-log mapping** for each artifact is automatically discovered to map the schema to event log specifications. Afterwards, they exploit XESame (Verbeek et al., 2010) to extract the event data and create one event log for each artifact (multiple logs are created). This is done in a way that events within each artifact log are correlated to the respective artifact. Finally, existing discovery algorithms are used to discover one life-cycle model for each artifact.

Their approach automates various steps of the log extraction. Specially, the schema-log mapping phase automates the first step of XESame, mapping the database schema to event log schema. This enables them to leverage XESame for extracting data in an automated way. Additionally, their approach is generic and can be applied to any ERP system in which the timestamp of event data is recorded in a separate column. However, the approach is subject to a number of limitations. The schema extraction step can take a long time to execute as key discovery is an NP-complete problem. Besides, identifying value of k for clustering is still manual and difficult to select the right value to discover the right amount of artifacts. Including domain knowledge in the selection of k and the discovery of schema-log mapping can improve their approach. in addition, their approach only discovers individual life-cycle models for each artifact and neglects the interaction and relation between artifacts. These interactions can be a crucial factor to discover end-to-end meaningful process models.

Lu et al. (2015) addresses these problems in their approach by discovering complete artifact-centric process models including artifacts life-cycle and interaction between them. They argue that by indicating one-to-many and many-to-many relations in the database as interactions, the problems of convergence and divergence can be totally overcome. First, the life-cycle of each artifact, and then interactions between them are discovered. The approach is capable of discovering the interaction between two artifacts. Discovering interaction of multiple artifacts is not supported. Besides, identification of artifacts needs to be done manually by domain expert. After these artifacts are known, the discovered process model can correctly capture the behavior of object-centric systems avoiding convergence and divergence. However, the discovered model can be complex and hard to interpret. Besides, conformance checking and performance analysis on these models have not been studied.

Pajić and Bečejski-Vujaklija (2016) develop a metamodel for artifact-centric approaches. The metamodel connects concepts of the artifact-centric approach and XES metamodel illustrating the relationship of business objects, artifacts, their artifact schema, direct and indirect relations, and links to XES elements, traces, and events. The metamodel is intended to facilitate understanding and adoption of these approaches in practice.

3.3.2 | OpenSLEX

de Murillas et al. (2018) propose the OpenSLEX metamodel which combines the elements of an event log with object-related elements to store data in a way that provides sufficient information on processes, data types and relations. It integrates the data and process view into one storage. The proposed metamodel is compatible with XES to enable transformation to and from XES logs without data loss. Multiple event logs can be stored sharing events to avoid duplication. The meta model contains six elements, three for data view and three for process view. The data view contains data model (database schema), object (unique entities of data), and version (the values of attributes of an object in a certain time window). The process side consists of process, instance (case), and event (referring to high-level events reflecting activity executions). These two views are connected through version and event, in a way that changes of versions are tracked as events.

The meta model has been implemented as a library, OpenSLEX, in Java and can be accessed in the same way as XES, and it is stored in SQLite files. Additionally, the procedure of extracting, transforming and querying data has been implemented in RapidProM (Mans et al., 2014) [3]. Using adapters for each database, data can be extracted and populated to OpenSLEX. A limitation is that an ad-hoc adapter needs to be written for every data schema, and a general method independent of the underlying data source is missing. However, after the adapters are built, using operators of RapidProM, logs with additional perspectives can be created automatically. Adapters for a number of systems including Oracle redo logs, and SAP change tables are available.

OpenSLEX provides a technology to store data for process mining in a way to keep the balance between process and data view. Multiple event logs can then be created (in XES) based on the chosen view (case notion) to be processed by process mining techniques. Their approach is comparable to works of data warehousing (Niedrite et al., 2007; Neumuth et al., 2008; Eder et al., 2002; Zur Muehlen, 2001) and process cubes (van der Aalst, 2013; Vogelgesang and Appelrath, 2015; Bolt and van der Aalst, 2015; Vogelgesang and Appelrath, 2016; Vogelgesang et al., 2016) which allow storing multidimensional data in a process-oriented way and enable analyzing data from multiple aspects using slice and dice operations. However, OpenSLEX offers more generalization and independence from a specific process compared to these works. in (de Murillas et al., 2017a) they apply their method for extraction and storage of event data from audit trails of hospital information systems and show that due to independent from a single case notion at the time of extraction, data loss is avoided and creation of different views at the time of analysis is facilitated.

[3] http://www.rapidprom.org/

In (de Murillas, 2019), the approach for extracting data and building event logs has been extended and improved. In addition, the techniques to create event logs are applied on a Hospital Information System (HIS) in six steps. *Data exploration*, where the underlying database is analyzed and information on its size, existing tables, and relations is collected. *Data schema discovery*, in which the complete database schema is discovered, including primary and foreign key relations that might have only been created in the application layer. *Data extraction*, that populates OpenSLEX by data objects and object versions. *Event data discovery*, to identify and extract event data from multiple tables. This is done by automatically identifying timestamp columns and using them as event data. *Case notion discovery*, which discovers and recommends possible case notion based on the data schema, for correlation. *Event log building*, to construct multiple event logs based on the extracted event data and different case notions, in order to create multiple views on the database and processes. The techniques for the above steps are implemented as a python library called *eddytools*. The approach however, is semi-automatic as the domain knowledge is exploited to analyze and refine the output of each step, and provide more information for achieving meaningful event logs.

3.3.3 | XOC Event Log Format

Li et al. (2018b) developed the eXtensible Object-Centric (XOC) event log format, as an alternative to XES. It lifts the assumption of existence of a case notion in the log to avoid flattening the data. As a result, there is no trace which events are grouped into it. Instead, events are related to each other based on objects (data elements) and their relation. XOC logs contain five elements for each event. event identifier, event type, object references (modified objects by the event), object model (objects and relations), and event ordering. In order to extract event data, domain knowledge is required to identify event types to link events to database changes. *XOC Log Generator* is a plug-in in ProM which automatically extracts XOC logs from redo logs or change tables of databases, after relevant event types are specified using domain knowledge.

Based on XOC logs, data-aware models in a novel modelling notation called OCBC (Object-Centric Behavioral Constraint) can be discovered. These models combine the data model, consisting of data object classes, and relations, with the process model, including activities and behavioral constraints expressed in a declarative model (Declare (van der Aalst et al., 2009)). Declarative models are constraint-based, that instead of explicitly modelling the ordering between activities, define constraints which implicitly imply possible ordering of activities in the model. These OCBC models can then be exploited for conformance checking to reveal deviations with regard to the related data which could not be detected previously. These OCBC models discovered from XOC logs include relationships between objects capturing one-to-many and many-to-many relations. However, they can be difficult to interpret as they do not have a process instance notion, and have constructs different than the procedural modelling notations (e.g. BPMN).

TABLE 2 Overview of event data extraction approaches (ordered based on their main focus)

Authors (Year)	Approach	Main Focus	Input	Output	Limitations
Wang et al. (2012); Li et al. (2015)	text mining on process documents	event identification	process documents	ranked tables of databases	requires manual annotation and validation of meaningful text components
Günther and van der Aalst (2006)	ProM Import Framework	event log conversion	PAIS logs	MXML	limited to a number of PAISs
Ingvaldsen and Gulla (2007)	EVS Model Builder	event log extraction and conversion of SAP	user defined meta-description (mapping)	MXML	knowledge and time intensive conversion and mapping definition
Verbeek et al. (2010); Buijs (2010)	XESame	event log extraction (query generation) and conversion	domain model + mapping definition	XES	Knowledge and time intensive conversion and mapping definition
Rodrıguez et al. (2012)	eventifier	event identification and extraction	records of table joins are considered as events	XES	naive assumption on event identification
Calvanese et al. (2016, 2017c,b)	OBDA-based (onprom)	event log extraction	domain model + event annotation + mapping	XES	manual and time intensive mapping definition
van der Aalst (2015)	redo-log-based	databases with no historical data	redo-logs of databases + data model + event model	event logs	no implementation
de Murillas et al. (2015, 2017b)	redo-log-based	databases with no historical data	redo-logs of databases + data model	XES	duplicate events across traces, accessibility and availability of redo-logs
Nooijen et al. (2012)	artifact-centric	dealing with many-to-many relations	database schema + schema log mapping (automatic)	XES (for each artifact)	limited to discovering artifact life-cycle models neglecting interaction between artifacts
Lu et al.(2015)	artifact-centric	Dealing with one-to-many and many-to-many relations	database schema + schema log mapping (semi-automatic)	XES	limited to discovering interaction between two artifacts
de Murillas et al. (2018)	OpenSLEX	preserving data perspective, and extracting logs with different views	data model	XES	ad-hoc adapters required for new systems
Li et al. (2018b)	XOC log creation	dealing with many-to-many relations, and preserving data perspective	redo logs/change tables	XOC	limited to discovery of OCBC process models

Table 2 provides an overview of the extraction approaches. Despite the efforts for automating extraction of event data from databases, no fully automated approach exists to date. Although, many approaches automate the generation of queries, most approaches rely on domain knowledge to identify event data in the database. Several approaches also attempt to mitigate the problem caused by many-to-many relations and propose novel approaches and modeling notations to achieve this. However, overcoming this issue, while discovering an end-to-end process model in the conventional process modeling notations such as BPMN or Petri net remains a challenge.

4 | EVENT CORRELATION

Event correlation aims at associating event data extracted from data sources to cases of a business process. Thereby, the vast majority of approaches pursuing this endeavor agree on the assumption that each event will be correlated to a case, but there may be differences regarding the information exploited to achieve the correlation. For example, only information about the event name may be used (Ferreira and Gillblad, 2009), but also additional event attribute values (such as timestamps) (Motahari-Nezhad et al., 2011), and even the corresponding process model (Bose et al., 2013) are consulted. This is illustrated in Figure 5. Therefore, in the following we will describe event correlation approaches from the literature and emphasize what event attribute information they rely on, and whether or not they assume an existing process model whose instances the events are to be correlated with. Also, limitations of the approaches are mentioned.

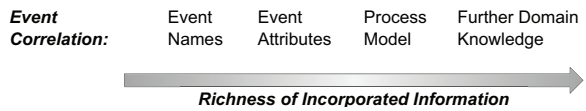

FIGURE 5 The spectrum of event correlation techniques.

A probabilistic approach to correlate events with process instances as well as discovering a process model is described in (Ferreira and Gillblad, 2009) based on Markov chains and an expectation-maximization technique (Dempster et al., 1977). This approach only requires the ordered event log with **event names** as input. From that information, it estimates an initial transition probability matrix that gives for each pair of events a probability that one follows the other. Given that matrix, it can assign an event to a process instance by choosing the instance whose last event is most likely to be followed by the current event. Note that it may also be the case that the current event is most likely to start a new instance or terminate a running instance. After all events have been assigned to a process instance, the estimate of the transition probability matrix can be improved, and the events can be reassigned based on the improved matrix. This procedure is iterated until the matrix and the event assignment do not change anymore. The approach, however, is unable to deal with existence of loops and parallelism.

Walicki and Ferreira (2011) provide a sequence partitioning approach to derive a set of partitions that represents the minimum cover of the unlabeled log. The input to that approach is a stream of events missing case identifiers which are then clustered into traces by applying sequence partitioning. Their method only requires the event names and orderings. Again, it is unable to deal with loops and parallelism.

Pourmirza et al. (2015) tackle the correlation problem by simply mining models from logs without relying on case identifiers. Therefore, even after a model was discovered from the log, there will still be no information about which event in the log belongs to which process instance. This approach only relies on event names and timestamps in the log, no additional event attributes are required. The idea is based on *orchestration graphs*, as for example used by the process mining tool Disco (Günther and Rozinat, 2012). Those graphs consist of nodes that correspond to events in the log and directed edges between those nodes. Each node is associated with the number of times the corresponding event was observed in the event log. In the same way each edge is annotated with the number of times the corresponding transition was taken. Finally, the number of incoming and outgoing transition counts of a node must be equal to the count that is associated with that node. Based on those constraints, several orchestration graphs can be generated from the log. The generated orchestration graphs are assigned a score based on two metrics. On the one hand, a precede/succeed score indicates how many times an event has occurred before another event in

the log. If this score is high, it is likely that there is a corresponding edge in the process model. On the other hand, a duration score indicates the time difference between two events. If this score is low, it is also likely that there is an edge between the two events in the model. In the end, integer linear programming is used to find the optimal orchestration graph. Still, loops are not supported.

In (Motahari-Nezhad et al., 2011) the authors propose an approach to associate events from different data sources belonging to the same case based on *correlation conditions*. Such conditions are formulated in reference to the **attribute values** of the events. In simple cases one can compose conditions from identifier attributes of the events, for instance, events may have the same *OrderID*. However, also more complex correlation conditions are conceivable that consist of conjunctions and/or disjunctions of attribute values. An example for this is given in (Engel et al., 2016, 2013) where authors correlate EDI messages using their *OrderID* and *ItemID*. However, those approaches do rely on a conceptual diagram, such as an entity-relationship model, to link events from different data sources. Similarly, (Burattin and Vigo, 2011), relate different activities to the same process instance via the identification of relations among their attributes. Such relations are simply established by matching the attribute values of the different activities. This method highly relies on user-defined parameters, though, for instance, the number of events in a case.

Motahari-Nezhad et al. (2011) furthermore suggest a method to automatically derive those conditions on the basis of their "interestingness". For example, an attribute is *not* interesting for correlation if it is unique for every event since that would lead to one process instance per event. Similarly, attributes with very small domains such as Booleans will not be particularly useful for correlation either. Hence, only attributes that have a good ratio of distinct values across the event log are considered for correlation conditions. Also, the correlation condition should not cause the log to be partitioned into very few long instances or into a very high number of short instances. Relying on such heuristics different types of log partitions (i.e., event correlations) can be proposed, that may then be validated by a domain expert.

In (Beheshti et al., 2011) that approach is further developed with an extension of the query language SPARQL so as to partition events in logs into groups and paths, in order to facilitate the discovery of event correlations.

Another extension of the above technique is proposed by Pérez-Castillo et al. (2014), in which authors also discover correlation sets over attributes from events. This work, however, makes use of additional external knowledge, namely the source code, which is extended in order to collect events during their execution, together with potential event correlation attributes, defined by domain experts. Afterwards, the approach of Motahari-Nezhad et al. (2011) is applied to correlate events with process instances. Obviously, a limitation of this technique is that it requires access to the source code for event logging.

Rozsnyai et al. (2011) determine correlation candidates by calculating various statistics on the event attribute values. Those statistics include, for instance, the size of the attribute's domain, a count for how many times each value of the domain is encountered in the data, and the total number of instances in which the attribute occurs. Based on such statistics, correlation pairs are determined automatically, ranked by a confidence value. For example, if an attribute value occurs more than a certain number of times, it is unlikely that it can be used for correlation. Note that this is similar to the ratio of distinct values determined by the approach of Motahari-Nezhad et al. (2011).

In (Reguieg et al., 2012) and (Reguieg et al., 2015) a MapReduce-based approach is described that is supposed to discover correlation conditions from big process data. The authors also intend to automatically discover atomic and conjunctive/disjunctive correlation conditions, using the same metrics as Motahari-Nezhad et al. (2011). However, they make this approach more efficient by generating all possible candidate conditions and partitioning them over the network. This way, all potential correlation conditions can be processed in parallel. This, however, results in a huge amount of network traffic. Therefore, the idea of parallel processing of potential correlation conditions was further continued by Cheng et al. (2017), including a filtering step in the procedure to prune a large number of uninteresting

rules. Therefore, not all potential correlation conditions are examined but only those that fulfill certain criteria will be sent over the network for parallel processing.

Abbad Andaloussi et al. (2018) assume that the case id is a hidden attribute inside the log. After filtering out attributes that have a low probability to be the case id because they contain many distinct values (such as an event id), for each of the remaining attributes they assume that it is indeed the case id and discover a process model based on this assumption. The attribute that yields the best model according to the known control-flow quality dimensions of process mining is then suggested as the case id. Due to the iterative nature of this approach, with large event logs there is a possibility of memory overhead. They propose to use a sample of the log to avoid this issue. However, the selection of an optimal sample size is a challenge. Their evaluation shows that in case of small sample size the accuracy drops and wrong case id candidates might receive good scores. They suggest, however, as future work that heuristics or domain knowledge can be used to filter out unlikely candidate attributes.

In (Bose et al., 2013) authors correlate events relying on both attribute values from an event log and a **process model** represented in the declarative process modeling language *Declare* (van der Aalst et al., 2009). Two events are considered to be correlated if they, for example, share common data objects of the process or if they are executed by the same resource. In that way, correlation candidates are determined, which are then filtered with the help of the temporal logic constraints on the corresponding events formulated in the Declare model: If a fulfilled constraint of the Declare model can be observed in a trace and additionally the potential correlation is true in that trace, then the correlation's significance is increased. This approach actually does not produce a correlated event log. Rather, it enhances an existing Declare map.

In (Bayomie et al., 2016b) the authors correlate events to cases using the names and completion timestamps of the events. Furthermore, they employ the corresponding process model containing activities that have the same name as the events, and heuristic information about the execution duration of the activities in the model. From the process model the authors are able to compute a behavioral profile (Weidlich et al., 2011) that defines for every pair of activities in the model whether they occur in sequence, in parallel, in reverse order, exclusively, or in no order. On the basis of the behavioral profiles and the execution times, the authors can then label the event log with case identifiers. For example, from the behavioral profile they can derive which events can and cannot appear in the same case, and in which order. Further, from the completion timestamps and the average execution durations, they can refine the correlation, by excluding events from a case whose average duration would contradict the timestamps of the case. In (Bayomie et al., 2016a) the approach is extended with a pre-processing and a filtering step to account for cyclic process models containing loops. However, in this approach the quality of the output is sensitive to its input. For example if the process model contains activities or activity relations different than in the log, the output labeled log will contain noise or missing events.

In (Mannhardt et al., 2015) the authors enrich existing event logs with supplementary events from other data sources. While the existing event log contains case identifiers, the events from the additional sources do not, and need to be correlated with the existing events. To do so, the approach requires as another input, the process model corresponding to the existing event log, which should be specified as a data Petri net (DPN) (de Leoni et al., 2014). DPNs are an extension of Petri nets with transitions that can read and write variables. Therefore, transitions may have expressions over the variables attached to them (called guards) that must be fulfilled in order for the transition to be enabled. Also, the transitions may update the variables. Given a process model specified as a DPN and the corresponding event log, the authors compute alignments (Mannhardt et al., 2016) between those two artifacts. An alignment essentially matches each event of a trace with its corresponding activity in the process model. In this way it is possible to find out if there are activities in the model that were not recorded in the event log. The objective is then to find those missing events in the additional event sources. For each missing event three steps are executed: First, all

the events that correspond to the missing event are searched for in the additional sources. Second, the found event set is filtered for events that have compatible timestamps with the events in the trace under investigation. Third, the filtered event set is further filtered by excluding events that do not conform to the data requirements specified by the DPN together with the trace under investigation. This should give a small set of candidate events that may be correlated with the trace missing that event. In general, in the model-based approaches the accuracy and reliability of final result depend on the quality and fitness of the model. If the model contains many different behavior or activities from the event log the approaches may produce noisy or incorrect logs with missing events.

Data correlation is also a topic in the context of discovering artifact life-cycles, such as in (Nooijen et al., 2012; Popova et al., 2015; Lu et al., 2015). The life-cycles of artifacts are assumed to be related to potentially multiple process instances, such that the discovered processes are called *artifact-centric*. The main challenge of such approaches is the discovery of the artifacts around which the process is centered. The events can then be correlated to an artifact via primary key-foreign key relationships. In this paper, however, we consider an activity-centric setting as described in Section 2.1 and Section 2.1, requiring different approaches to correlation since the events cannot be related to a central artifact.

Li et al. (2018a) correlate events extracted from service-oriented systems (such as ERP and CRM systems) via object paths. The events are extracted from various data sources, for example, from the redo logs of the databases, which essentially record the history of database changes, also referring to the objects that were changed. Since objects in the database are related to each other via primary/foreign keys, the authors are able to correlate events of the redo logs by means of so called objects paths. An object path is a sequence of objects in which adjacent objects are related to each other. Therefore, if redo events refer to the objects in the same object path, they must be related to each other. The assumption here is that the events extracted from the data sources can always be related to some database object. Note that in this case events are not further grouped into traces. The method described in (de Murillas et al., 2015; de Murillas, 2019), however, allows to generate different sets of cases by defining particular case notions on the basis of data relations in the data model. A case notion defines which events should be part of the same trace by determining database objects that all the events must refer to directly or indirectly via primary/foreign keys relations.

Table 3 summarizes the approaches described in this section. It not only lists which pieces of information of the spectrum of Figure 5 they include, but also specifies their outputs and limitations. In summary, event correlation approaches that are able to deal with real-world data and processes make strong assumptions about the availability of extra information in addition to raw events. For example, the data must come from databases with an underlying data model, or an existing process model must be available that perfectly fits the data. This cannot be guaranteed in every situation such that those approaches lack the flexibility to be universally applicable.

TABLE 3 Overview of event correlation approaches (ordered based on the incorporated information as outlined in Figure 5)

Authors (Year)	Event Names	Event Attributes	Process Model	Further Domain Knowledge	Output	Limitations
Ferreira and Gillblad (2009)	yes	no	no	no	correlated events	no support for parallelism/loops
Walicki and Ferreira (2011)	yes	no	no	no	correlated events	no support for parallelism/loops
Pourmirza et al. (2015)	yes	yes (timestamp)	no	no	process model	no support for loops
Motahari-Nezhad et al. (2011)	yes	yes	no	conceptual diagram	correlated events	requires conceptual diagram
Engel et al. (2013, 2016)	yes	yes	no	no	correlated events	limited to EDI message events
Burattin and Vigo (2011)	yes	yes	no	no	correlated events	dependent on user-defined parameters
Beheshti et al. (2011)	yes	yes	no	conceptual diagram	correlated events	requires conceptual diagram
Pérez-Castillo et al. (2014)	yes	yes	no	source code	correlated events	requires source code
Rozsnyai et al. (2011)	yes	yes	no	no	correlated events	requires large data set
Reguieg et al. (2012, 2015)	yes	yes	no	no	correlated events	high network traffic
Cheng et al. (2017)	yes	yes	no	no	correlated events	requires event data partitioning
Abbad Andaloussi et al. (2018)	yes	yes	no	no	correlated events	memory overhead possibility and sampling bias
Bose et al. (2013)	yes	yes	Declare	no	enhanced Declare map	no event log output
Bayomie et al. (2016a, 2016b)	yes	yes (timestamp)	Petri net	event data constraints	correlated events	sensitive to the quality of its input (e.g. a fitting process model and accurate activity durations)
Mannhardt et al. (2015)	yes	yes	data Petri net	no	extended event log	assumes existence of a main event log with case identifier
de Murillas et al. (2015)	yes	yes	no	data schema	correlated events	extracted events must be related to database objects
Li et al. (2018a)	yes	yes	no	objects paths	correlated events	extracted events must be related to database objects

5 | EVENT ABSTRACTION

Techniques for event abstraction aim to bridge differences in the granularity at which data is recorded and at which it shall be analyzed. In Figure 1, this step can be seen as the transition from data elements that reside in a data store to the events that represent the execution of an activity and are correlated to traces in an event log. As such, event abstraction can be seen as a way to assign semantics to data elements, by identifying which data elements jointly describe the execution of an activity. Note though, that in the literature, data elements are often referred to as low-level events, whereas the events that represent activity executions are called high-level events.

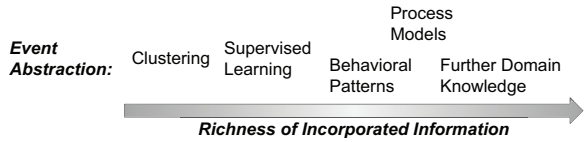

FIGURE 6 The spectrum of event abstraction techniques.

A general overview of the spectrum of approaches for event abstraction is given in Figure 6. We note that approaches differ in the richness of the information that is considered as a starting point.

A first angle to approach the abstraction of data elements to events is to exploit techniques for **clustering**. Clustering may simply be applied to all attribute values of data elements (Brzychczy and Trzcionkowska, 2018). Yet, for real-world data, this can be expected to only support a semi-automated process to abstraction, as expert input on relevant attributes and the exact cluster definitions is needed. Moreover, it was shown by Günther and van der Aalst (2006b) that clustering may in particular exploit the proximity of temporal information that is assigned to data elements. Here, the underlying assumption is that only data elements that are assigned timestamps which are close to each other, qualify for being aggregated into a single event. Specifically, the approach considers the notion of a sliding window over the timestamps of data elements to identify initial clusters for abstraction. The resulting clusters are then refined based on further attribute values, e.g., by requiring the data elements to have the same values for attributes that indicate originators or cases. Also, overlap among clusters is resolved, so that, eventually, one event is derived per cluster. Note that similar ideas of aggregating elements based on their proximity may also be employed on the model level (Günther and van der Aalst, 2007), i.e., a model is discovered first from low-level data elements and clustering is then applied for the vertices of the resulting graph.

Clustering of data elements has also been used in the context of feature engineering for predictive models, see van der Aalst et al. (2011); Folino et al. (2014, 2015). When aiming to learn a predictive model from historical data recorded for a process, the construction of features from sets of data elements can be seen as a form of event abstraction. However, the goal of this abstraction is not the precise determination of an event that denotes an activity execution, but rather the construction of features with high predictive power.

Data elements may also be clustered based on linguistic analysis of the labels assigned to them. Based on the observation that process-related data is often referring to actions that are applied to objects, the identification of these actions and objects (i.e., the verbs and nouns in labels) using techniques of Natural Language Processing (NLP) provides an angle to construct clusters (Richetti et al., 2014). Elements are grouped based on semantic relations, such as hypernymy, by employing common measures for the semantic distance between terms.

Beyond clustering, techniques for **supervised learning** may be used for event abstraction. Specifically, Tax et al. (2016a) and Tax et al. (2016b) showed how to phrase event abstraction as a sequence labeling problem. Then, features are defined that relate to some ordering of data elements, organizational information attached to them, or temporal

aspects. Given some labelled training data, Conditional Random Fields are learned to obtain a probabilistic mapping of data elements to events, i.e., to activity executions is learned. A similar idea is followed by Fazzinga et al. (2018b), who suggest to train a Hidden Markov Model where the hidden states are events that denote activity executions, while the observations are given by the data elements.

Another set of techniques proposes event abstraction based on **behavioral patterns**. Assuming that data elements are ordered (e.g., by a timestamp) and partitioned into traces (e.g., by a case attribute), the regularities in sequences of data elements may be analyzed to identify candidates for abstraction. Bose and van der Aalst (2009) provide a taxonomy of behavioral patterns for event abstraction. For instance, they define patterns based on the repetition of (approximate) sub-sequences of data elements, which may originate from choices or concurrency as part of loops in a process. Such patterns may be detected within the sequence of data elements that belong to a single case, or among those of different cases. In any case, the detection of these patterns, along with an evaluation of their significance, facilitates event abstraction: The respective pattern occurrences denote events and represent the execution of an activity. Various extensions of this general idea have been proposed. In particular, the relations between patterns, i.e., whether one pattern covers another pattern, as well as measures to assess the frequency and significance of patterns have been proposed (Li et al., 2011; Bose et al., 2011). Based thereon, fine-granular control on the selection of patterns to use for event abstraction is achieved.

A richer notion of patterns has been proposed by Mannhardt et al. (2018) for event abstraction. That is, activity patterns capture common behavioural structures that denote activity executions. Unlike the aforementioned patterns, activity patterns are semantically richer as they are defined using common process modelling approaches. Assuming that data elements are ordered and partitioned according to cases, activity patterns may specify not only sequential ordering of data elements, but also exclusive choices, concurrency, and repetitions. From a set of these activity patterns, a so-called abstraction model is then derived by defining further control-flow between them. Sequences of data elements may then be aligned with this abstraction model by constructing a matching between the data elements and the activity patterns. The elements that are matched to the same pattern are then subject to abstraction and yield a single event, which represents the execution of an activity or a transition in the life-cycle of transition execution (e.g., its start or end). While this approach enables event abstraction based on activity patterns that have been manually defined by domain experts, it may also be combined with approaches that discover frequent process model fragments (Mannhardt and Tax, 2017). This way the above idea of frequent sub-sequence mining for event abstraction (Bose and van der Aalst, 2009) is lifted to the richer notion of activity patterns. Moreover, it was argued that supervised techniques for pattern-detection shall be combined with unsupervised ones (e.g., based on clustering as mentioned above) when aiming at the identification of patterns among data elements (Lu et al., 2017), thereby achieving a more interactive approach of event abstraction.

Instead of going the path via model synthesis, a **process model** may also directly be used for event abstraction. Baier et al. (2014, 2018) show that such a process model, along with accompanying documentation, provides a starting point for abstraction based on textual analysis. That is, data elements are correlated with the activities of a process model based on the similarity of their assigned labels, which potentially involves linguistic analysis using techniques for NLP as mentioned above. Moreover, context information, e.g., in terms of ordering constraints that are imposed between the data elements, is exploited to reduce the number of possible matches. While these approaches support event abstraction, they cannot be completely automated though and require manual intervention. Neglecting textual information and focusing on ordering constraints, Fazzinga et al. (2018a) presented an approach to relate data elements to (a set of) process models. They show how to assign probabilities to different interpretations of data elements in terms of activity executions (of potentially even different processes).

Model-based approaches may also be combined with an analysis of patterns among the timestamps, and thus,

ordering of data elements. In Ferreira et al. (2014), for instance, event abstraction was phrased as a mapping problem between sets of data elements and events that represent executions of activities in a model. Construction of such a mapping shall not only consider the control-flow constraints imposed by the model, but be based on common patterns among the data elements, assuming that these patterns indicate a meaningful scope for abstraction.

Lastly, comprehensive **domain knowledge** on the context of process execution provides a valuable source of information for event abstraction techniques. A prime example here is the approach proposed by Senderovich et al. (2016). It first relates data elements that have been recorded for different resources. Based on temporal and spatial information assigned to the data elements, intervals of interaction between resources are determined. These interactions are then matched to events, i.e., activity executions with the help of domain knowledge, e.g., on common execution times of certain activities (and, thus, lengths of the respective intervals), the involvement of certain types of resources in activities, or the ordering of several different activities for specific resources. As such, domain knowledge provides a set of constraints to optimize a matching between sets of data elements and events.

TABLE 4 Overview of event abstraction approaches (ordered by the richness of the incorporated information as outlined in Figure 6)

Authors (Year)	Approach	Input	Main Assumption/Limitation
Brzychczy and Trzcionkowska (2018)	clustering based on attribute values	relational data	requires domain knowledge on relevant attributes, semi-automated construction of clusters
Günther and van der Aalst (2006)	clustering based on temporal proximity and attribute values	timestamped data elements	temporal information needs to indicate abstraction
van der Aalst et al. (2011); Folino et al. (2014,2015)	learning of compound features that induce abstraction	data that is labelled for a prediction task	tailored to the setting of predictive analytics
Richetti et al. (2014)	group elements based on linguistic relations and textual, semantic distances	textual data	data carries labels that are amendable to linguistic analysis
Tax et al. (2016b); Tax et al. (2016a)	supervised sequence labeling to learn a function that maps elements to events	data elements labelled with events	multivariate data that enables the derivation of useful features
Fazzinga et al. (2018b)	training of a Hidden Markov Model to derive events	labelled data elements	abstraction must follow a state-based, Markovian model
Bose and van der Aalst (2009); Li et al.(2011); Bose et al. (2012)	identify frequent sequential patterns for abstraction	sequential data	assumption of events materializing as repetitive patterns
Mannhardt et al. (2018); Mannhardt and Tax (2017)	abstraction based on predefined or mined activity patterns	activity patterns or labelled data	predefined patterns are assumed to be given or assumed to be frequent for mining
Baier et al. (2014,2018)	abstraction based on textual matching of event data and process model	process models and textual event data	assumes a strong link between the model and data, as well as user input
Fazzinga et al. (2018a)	abstraction using matching of process models and event data based on ordering constraints	process models and sequential data	ordering is assumed to characterize abstraction
Ferreira et al. (2014)	combined model-based abstraction with temporal analysis of data	timestamped data and process model	strong link between model and data is assumed
Senderovich et al.(2016)	identify events based on spatial data	spatial data of resources and domain knowledge on possible activities	tailored to data of real-time locating systems
Leonardi et al. (2017)	abstraction based on a given domain ontology	mapping of data elements to ontology	domain ontology and contextual rules need to be known

In the same vein, it was suggested to first map data elements to elements of a formal ontology that describes the domain (Leonardi et al., 2017). Based thereon, rules to select among different interpretations, i.e., different abstractions, of these data elements are exploited. Those encode context, for instance, as ordering constraints for activity executions. In a final step, data elements are abstracted into a single event, if they are assigned consecutive timestamps and if the rules selected the same abstractions for them.

Recently, Koschmider et al. (2018) argued that such contextual information provide a rich basis for the event abstraction. In particular, a classification of context factors to consider has been developed, featuring personal and social context, task context, environmental context, and spatial-temporal context.

Table 4 provides an overview of the event abstraction approaches. The approaches range from using clustering techniques to behavioral patterns, or even existing process models. As can be observed from the table, the approaches operate on different inputs, make different assumptions, and are exposed to certain limitations. Depending on the problem settings and available information, certain approaches become relevant.

6 | CONCLUSION

Creating event logs from data residing in various information systems, enables the application of process mining to extract meaningful knowledge and insights on the behavior of these systems and the underlying processes. By discovering and analyzing end-to-end processes, this knowledge can trigger substantial improvement in the design and performance of systems and business processes. In an event log, events are recorded reflecting the execution of activities, and are related to specific instances of running processes, and thereby grouped into traces. However, in most systems, data is not recorded in the suitable way for process mining. Therefore, to extract and transform this data from a variety of data sources with different characteristics, to the event log format required by process mining techniques, several challenges have to be overcome. This article provides a comprehensive review of the techniques introduced in the academic literature to support the three steps towards the creation of event logs from raw data, in order to present the state-of-the-art, and encourage future work in this crucial and challenging phase of process mining. These three steps are, *event data extraction*, *correlation*, *and abstraction*.

Databases of most information systems such as ERP or legacy information systems are not process-aware and do not record events explicitly. Therefore, techniques need to be developed to identify relevant event data in these databases, extract and transform them. These techniques normally involve various manual steps and rely on the knowledge of the database and the domain. To support the domain expert, to automate this process as much as possible, and to decouple the database knowledge and the domain knowledge, several event data identification and extraction techniques have been developed. A number of these techniques rely on defining a domain model, which represents the domain of interest, an event model, which enriches the domain model by event information, and a mapping specification, connecting these models to event data and databases.

The extracted event data needs to be correlated to specific instances of processes and grouped into respective traces to enable the analysis of end-to-end process executions. This correlation is not trivial when a clear notion of a process instance is not available or events come from sources in which correlation is non-existent. Correlation techniques have been proposed depending on the available information, ranging from event attributes, available process models, or relations between data.

Events in event logs also need to relate to activity executions. Therefore, data extracted from data sources need to be mapped to respective activities. In case low-level events are recorded (for example, by sensors, devices , X-ray machines), they need to be grouped and abstracted to a higher level of granularity to reflect activity executions.

Abstraction techniques range from clustering, supervised learning techniques to behavioral patterns. Abstraction techniques bridge the granularity gap between recorded data and that of required by process mining.

The approaches to date are subject to a number of limitation. Often they make strong assumptions or depend on the domain expert to provide the necessary information. In the context of event data identification and extraction, in spite of various works, a fully automatic approach is missing and most approaches rely on domain knowledge. The distinct characteristics of various existing data sources is one of the main obstacles for an automatic generic approach. Studying and analyzing different types of system and their characteristics can lay the foundation for the design of a generic approach. Besides, a number of problems caused by the nature of data storage, in which the required data is stored and relates to other data objects remains unsolved. Many-to-many relationships cause the problem of convergence and divergence in event logs. Although, approaches preventing these problems have been developed, the problem is not fully avoided, or new and often complicated process modellings languages are used. Preventing this issues for more conventional process models requires further investigation. Moreover, events might be recorded in different levels (parent-child relations between tables e.g. order vs. order line). This can lead to incorrect result (incorrect sequence relations) if they are mixed in an event log. New approaches capable of handling this problem are required (e.g. Multi-level process mining).

Event correlation approaches often make strong assumptions about the availability of extra information in addition to raw events, to guide the correlation. Domain knowledge is also required for a number of these approaches. Besides, many of the approaches have problems with scalability when it comes to large scale projects. In terms of event abstraction more empirical evaluations and comparative studies will be beneficial to evaluate and compare the performance of the different approaches. The potential of using approaches from the area of Complex Event Processing (CEP) (Soffer et al., 2018) as a basis for abstraction of event data is an interesting area for further future work. One interesting direction for future work on event log creation is studying the possibility of combining a few of the approaches from different areas to create an end-to-end and generic solution for event log creation from raw data of different data sources. A number of the approaches can potentially be combined leveraging the ideas and techniques developed separately, in an integrated manner.

Acknowledgements

We thank Felix Wolff for his support in collecting and summarizing the literature referenced in this review.

references

van der Aalst, W. M. (2013) Process cubes: Slicing, dicing, rolling up and drilling down event data for process mining. In *Asia-Pacific Conference on Business Process Management*, 1–22. Springer.

— (2015) Extracting event data from databases to unleash process mining. In *BPM-Driving innovation in a digital world*, 105–128. Springer.

— (2016) *Process Mining - Data Science in Action, Second Edition*. Springer. URL: `https://doi.org/10.1007/978-3-662-49851-4`.

van der Aalst, W. M., Adriansyah, A., de Medeiros, A. K. A., Arcieri, F., Baier, T., Blickle, T., Bose, J. C., van den Brand, P., Brandtjen, R., Buijs, J., Burattin, A., Carmona, J., Castellanos, M., Claes, J., Cook, J., Costantini, N., Curbera, F., Damiani, E., de Leoni, M., Delias, P., van Dongen, B. F., Dumas, M., Dustdar, S., Fahland, D., Ferreira, D. R., Gaaloul, W., van Geffen, F., Goel, S., Günther, C., Guzzo, A., Harmon, P., ter Hofstede, A., Hoogland, J., Ingvaldsen, J. E., Kato, K., Kuhn, R., Kumar, A., La Rosa, M., Maggi, F., Malerba, D., Mans, R. S., Manuel, A., McCreesh, M., Mello, P., Mendling, J., Montali, M., Motahari-Nezhad, H. R., zur Muehlen, M., Munoz-Gama, J., Pontieri, L., Ribeiro, J., Rozinat, A., Seguel Pérez, H., Seguel Pérez, R.,

Sepúlveda, M., Sinur, J., Soffer, P., Song, M., Sperduti, A., Stilo, G., Stoel, C., Swenson, K., Talamo, M., Tan, W., Turner, C., Vanthienen, J., Varvaressos, G., Verbeek, E., Verdonk, M., Vigo, R., Wang, J., Weber, B., Weidlich, M., Weijters, T., Wen, L., Westergaard, M. and Wynn, M. (2012) Process mining manifesto. In *Business Process Management Workshops* (eds. F. Daniel, K. Barkaoui and S. Dustdar), 169–194. Berlin, Heidelberg: Springer Berlin Heidelberg.

van der Aalst, W. M., Pesic, M. and Schonenberg, H. (2009) Declarative workflows: Balancing between flexibility and support. *Computer Science - Research and Development*, **23**, 99–113. URL: https://doi.org/10.1007/s00450-009-0057-9.

van der Aalst, W. M., Schonenberg, M. H. and Song, M. (2011) Time prediction based on process mining. *Information systems*, **36**, 450–475.

Abbad Andaloussi, A., Burattin, A. and Weber, B. (2018) Toward an automated labeling of event log attributes. In *Enterprise, Business-Process and Information Systems Modeling* (eds. J. Gulden, I. Reinhartz-Berger, R. Schmidt, S. Guerreiro, W. Guédria and P. Bera), 82–96. Cham: Springer International Publishing.

Ahmadi, B., Hadjieleftheriou, M., Seidl, T., Srivastava, D. and Venkatasubramanian, S. (2009) Type-based categorization of relational attributes. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, 84–95. ACM.

Augusto, A., Conforti, R., Dumas, M., Rosa, M. L., Maggi, F. M., Marrella, A., Mecella, M. and Soo, A. (2019) Automated discovery of process models from event logs: Review and benchmark. *IEEE Transactions on Knowledge and Data Engineering*, **31**, 686–705. URL: https://doi.org/10.1109/TKDE.2018.2841877.

Baier, T., Di Ciccio, C., Mendling, J. and Weske, M. (2018) Matching events and activities by integrating behavioral aspects and label analysis. *Software & Systems Modeling*, **17**, 573–598. URL: https://doi.org/10.1007/s10270-017-0603-z.

Baier, T., Mendling, J. and Weske, M. (2014) Bridging abstraction layers in process mining. *Information Systems*, **46**, 123–139.

Bayomie, D., Awad, A. and Ezat, E. (2016a) Correlating unlabeled events from cyclic business processes execution. In *International Conference on Advanced Information Systems Engineering*, 274–289. Springer.

Bayomie, D., Helal, I. M. A., Awad, A., Ezat, E. and ElBastawissi, A. (2016b) Deducing case ids for unlabeled event logs. In *Business Process Management Workshops* (eds. M. Reichert and H. A. Reijers), 242–254. Cham: Springer International Publishing.

Beheshti, S.-M.-R., Benatallah, B., Motahari-Nezhad, H. R. and Sakr, S. (2011) A query language for analyzing business processes execution. In *International Conference on Business Process Management* (eds. S. Rinderle-Ma, F. Toumani and K. Wolf), 281–297. Berlin, Heidelberg: Springer Berlin Heidelberg.

Bolt, A. and van der Aalst, W. M. (2015) Multidimensional process mining using process cubes. In *International Conference on Enterprise, Business-Process and Information Systems Modeling*, 102–116. Springer.

Bose, R. J. C. and van der Aalst, W. M. (2009) Abstractions in process mining: A taxonomy of patterns. In *International Conference on Business Process Management*, 159–175. Springer.

Bose, R. J. C., Verbeek, E. H. and van der Aalst, W. M. (2011) Discovering hierarchical process models using prom. In *International Conference on Advanced Information Systems Engineering*, 33–48. Springer.

Bose, R. P. J. C., Maggi, F. M. and van der Aalst, W. M. (2013) Enhancing declare maps based on event correlations. In *Business Process Management* (eds. F. Daniel, J. Wang and B. Weber), 97–112. Berlin, Heidelberg: Springer Berlin Heidelberg.

Brzychczy, E. and Trzcionkowska, A. (2018) Process-oriented approach for analysis of sensor data from longwall monitoring system. In *International Conference on Intelligent Systems in Production Engineering and Maintenance*, 611–621. Springer.

Buijs, J. (2010) *Mapping data sources to xes in a generic way*. Master's thesis, Eindhoven University of Technology.

Burattin, A. and Vigo, R. (2011) A framework for semi-automated process instance discovery from decorative attributes. In *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2011, part of the IEEE Symposium Series on Computational Intelligence 2011, April 11-15, 2011, Paris, France*, 176–183.

Calvanese, D., Cogrel, B., Komla-Ebri, S., Kontchakov, R., Lanti, D., Rezk, M., Rodriguez-Muro, M. and Xiao, G. (2017a) Ontop: Answering sparql queries over relational databases. *Semantic Web*, **8**, 471–487.

Calvanese, D., Kalayci, T. E., Montali, M. and Santoso, A. (2017b) Obda for log extraction in process mining. In *Reasoning Web International Summer School*, 292–345. Springer.

Calvanese, D., Kalayci, T. E., Montali, M., Santoso, A. and van der Aalst, W. M. (2018) Conceptual schema transformation in ontology-based data access. In *European Knowledge Acquisition Workshop*, 50–67. Springer.

Calvanese, D., Kalayci, T. E., Montali, M. and Tinella, S. (2017c) Ontology-based data access for extracting event logs from legacy data: the onprom tool and methodology. In *International Conference on Business Information Systems*, 220–236. Springer.

Calvanese, D., Montali, M., Syamsiyah, A. and van der Aalst, W. M. (2016) Ontology-driven extraction of event logs from relational databases. In *Proc. of the 11th Int. Workshop on Business Process Intelligence (BPI 2015)*, vol. 256 of *Lecture Notes in Business Information Processing*, 140–153. Springer.

Carmona, J., van Dongen, B. F., Solti, A. and Weidlich, M. (2018) *Conformance Checking - Relating Processes and Models*. Springer. URL: https://doi.org/10.1007/978-3-319-99414-7.

Cheng, L., van Dongen, B. F. and van der Aalst, W. M. (2017) Efficient event correlation over distributed systems. In *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, CCGrid '17, 1–10. Piscataway, NJ, USA: IEEE Press. URL: https://doi.org/10.1109/CCGRID.2017.94.

Dempster, A. P., Laird, N. M. and Rubin, D. B. (1977) Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, **39**, 1–38.

Depaire, B. and Martin, N. (2019) Data-driven process simulation. In Sakr and Zomaya (2019). URL: https://doi.org/10.1007/978-3-319-63962-8_102-1.

van Dongen, B. F. and van der Aalst, W. M. (2005) A meta model for process mining data. *EMOI-INTEROP*, **160**, 30.

Eder, J., Olivotto, G. E. and Gruber, W. (2002) A data warehouse for workflow logs. In *International Conference on Engineering and Employment of Cooperative Information Systems*, 1–15. Springer.

Engel, R., Bose, R. J. C., Pichler, C., Zapletal, M. and Werthner, H. (2013) Ediminer: A toolset for process mining from edi messages. In *CAiSE Forum*, 146–153. Citeseer.

Engel, R., Krathu, W., Zapletal, M., Pichler, C., Bose, R. P. J. C., van der Aalst, W. M., Werthner, H. and Huemer, C. (2016) Analyzing inter-organizational business processes. *Information Systems and e-Business Management*, **14**, 577–612. URL: https://doi.org/10.1007/s10257-015-0295-2.

Fahland, D., de Leoni, M., van Dongen, B. F. and van der Aalst, W. M. (2011) Behavioral conformance of artifact-centric process models. In *International Conference on Business Information Systems*, 37–49. Springer.

Fazzinga, B., Flesca, S., Furfaro, F., Masciari, E. and Pontieri, L. (2018a) Efficiently interpreting traces of low level events in business process logs. *Information Systems*, **73**, 1–24.

Fazzinga, B., Flesca, S., Furfaro, F. and Pontieri, L. (2018b) Process discovery from low-level event logs. In *International Conference on Advanced Information Systems Engineering*, 257–273. Springer.

Ferreira, D. R. and Gillblad, D. (2009) Discovering process models from unlabelled event logs. In *International Conference on Business Process Management*, 143–158. Springer.

Ferreira, D. R., Szimanski, F. and Ralha, C. G. (2014) Improving process models by mining mappings of low-level events to high-level activities. *Journal of Intelligent Information Systems*, **43**, 379–407. URL: https://doi.org/10.1007/s10844-014-0327-2.

Folino, F., Guarascio, M. and Pontieri, L. (2014) Mining predictive process models out of low-level multidimensional logs. In *International conference on advanced information systems engineering*, 533–547. Springer.

— (2015) Mining multi-variant process models from low-level logs. In *International Conference on Business Information Systems*, 165–177. Springer.

Günther, C. W. and van der Aalst, W. M. (2006a) A generic import framework for process event logs. In *International Conference on Business Process Management*, 81–92. Springer.

— (2006b) *Mining activity clusters from low-level event logs*. BETA publicatie : working papers. Eindhoven University of Technology.

— (2007) Fuzzy mining – adaptive process simplification based on multi-perspective metrics. In *International conference on business process management* (eds. G. Alonso, P. Dadam and M. Rosemann), 328–343. Berlin, Heidelberg: Springer Berlin Heidelberg.

Günther, C. W. and Rozinat, A. (2012) Disco: Discover your processes. *BPM (Demos)*, **940**, 40–44.

Günther, C. W. and Verbeek, E. (2014) Xes standard definition. *Fluxicon Process Laboratories (November 2009)*.

Hull, R. (2008) Artifact-centric business process models: Brief survey of research results and challenges. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, 1152–1163. Springer.

IEEE (XES) Working Group (2016) IEEE Standard for eXtensible Event Stream (XES) for Achieving Interoperability in Event Logs and Event Streams. *IEEE Std 1849-2016*, 1–50.

Ingvaldsen, J. E. and Gulla, J. A. (2007) Preprocessing support for large scale process mining of sap transactions. In *International Conference on Business process management*, 30–41. Springer.

Jans, M. and Soffer, P. (2017) From relational database to event log: decisions with quality impact. In *International Conference on Business Process Management*, 588–599. Springer.

Koschmider, A., Mannhardt, F. and Heuser, T. (2018) On the contextualization of event-activity mappings. In *International Conference on Business Process Management*, 445–457. Springer.

Leonardi, G., Striani, M., Quaglini, S., Cavallini, A. and Montani, S. (2017) Towards semantic process mining through knowledge-based trace abstraction. In *International Symposium on Data-Driven Process Discovery and Analysis*, 45–64. Springer.

de Leoni, M. and Mannhardt, F. (2019) Decision discovery in business processes. In Sakr and Zomaya (2019). URL: https://doi.org/10.1007/978-3-319-63962-8_96-1.

de Leoni, M., Munoz-Gama, J., Carmona, J. and van der Aalst, W. M. (2014) Decomposing alignment-based conformance checking of data-aware process models. In *On the Move to Meaningful Internet Systems: OTM 2014 Conferences* (eds. R. Meersman, H. Panetto, T. Dillon, M. Missikoff, L. Liu, O. Pastor, A. Cuzzocrea and T. Sellis), 3–20. Berlin, Heidelberg: Springer Berlin Heidelberg.

Li, G., de Carvalho, R. M. and van der Aalst, W. M. (2018a) Configurable event correlation for process discovery from object-centric event data. In *2018 IEEE International Conference on Web Services (ICWS)*, vol. 00, 203–210. URL: doi.ieeecomputersociety.org/10.1109/ICWS.2018.00033.

Li, G., de Murillas, E. G. L., de Carvalho, R. M. and van der Aalst, W. M. (2018b) Extracting object-centric event logs to support process mining on databases. In *International Conference on Advanced Information Systems Engineering*, 182–199. Springer.

Li, J., Bose, R. P. J. C. and van der Aalst, W. M. (2011) Mining context-dependent and interactive business process maps using execution patterns. In *Business Process Management Workshops* (eds. M. zur Muehlen and J. Su), 109–121. Berlin, Heidelberg: Springer Berlin Heidelberg.

Li, J., Wang, H. J. and Bai, X. (2015) An intelligent approach to data extraction and task identification for process mining. *Information Systems Frontiers*, **17**, 1195–1208.

Lu, X., Fahland, D., Andrews, R., Suriadi, S., Wynn, M. T., ter Hofstede, A. H. and van der Aalst, W. M. (2017) Semi-supervised log pattern detection and exploration using event concurrence and contextual information. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, 154–174. Springer.

Lu, X., Nagelkerke, M., van de Wiel, D. and Fahland, D. (2015) Discovering interacting artifacts from erp systems. *IEEE Transactions on Services Computing*, **8**, 861–873.

Mannhardt, F., de Leoni, M. and Reijers, H. A. (2015) Extending process logs with events from supplementary sources. In *Business Process Management Workshops* (eds. F. Fournier and J. Mendling), 235–247. Cham: Springer International Publishing.

Mannhardt, F., de Leoni, M., Reijers, H. A. and van der Aalst, W. M. (2016) Balanced multi-perspective checking of process conformance. *Computing*, **98**, 407–437. URL: `https://doi.org/10.1007/s00607-015-0441-1`.

Mannhardt, F., de Leoni, M., Reijers, H. A., van der Aalst, W. M. and Toussaint, P. J. (2018) Guided process discovery–a pattern-based approach. *Information Systems*, **76**, 1–18.

Mannhardt, F. and Tax, N. (2017) Unsupervised event abstraction using pattern abstraction and local process models. *arXiv preprint arXiv:1704.03520*.

Mans, R., van der Aalst, W. M. and Verbeek, H. E. (2014) Supporting process mining workflows with rapidprom. In *BPM (Demos)*, 56.

Motahari-Nezhad, H. R., Saint-Paul, R., Casati, F. and Benatallah, B. (2011) Event correlation for process discovery from web service interaction logs. *The VLDB Journal—The International Journal on Very Large Data Bases*, **20**, 417–444.

de Murillas, E. G. L. (2019) *Process mining on databases: extracting event data from real-life data sources*. Ph.D. thesis, Eindhoven University of Technology.

de Murillas, E. G. L., van der Aalst, W. M. and Reijers, H. A. (2015) Process mining on databases: Unearthing historical data from redo logs. In *International Conference on Business Process Management*, 367–385. Springer.

de Murillas, E. G. L., Helm, E., Reijers, H. A. and Küng, J. (2017a) Audit trails in openslex: Paving the road for process mining in healthcare. In *International Conference on Information Technology in Bio-and Medical Informatics*, 82–91. Springer.

de Murillas, E. G. L., Hoogendoorn, G. and Reijers, H. A. (2017b) Redo log process mining in real life: Data challenges & opportunities. In *International Conference on Business Process Management*, 573–587. Springer.

de Murillas, E. G. L., Reijers, H. A. and van der Aalst, W. M. (2018) Connecting databases with process mining: a meta model and toolset. *Software & Systems Modeling*, 1–39.

Neumuth, T., Mansmann, S., Scholl, M. H. and Burgert, O. (2008) Data warehousing technology for surgical workflow analysis. In *2008 21st IEEE International Symposium on Computer-Based Medical Systems*, 230–235. IEEE.

Niedrite, L., Solodovnikova, D., Treimanis, M. and Niedritis, A. (2007) Goal-driven design of a data warehouse based business process analysis system. In *Proceedings of the 6th Conference on 6th WSEAS Int. Conf. on Artificial Intelligence, Knowledge Engineering and Data Bases*, 243–249.

Nooijen, E. H., van Dongen, B. F. and Fahland, D. (2012) Automatic discovery of data-centric and artifact-centric processes. In *International Conference on Business Process Management*, 316–327. Springer.

Pajić, A. and Bečejski-Vujaklija, D. (2016) Metamodel of the artifact-centric approach to event log extraction from erp systems. *International Journal of Decision Support System Technology (IJDSST)*, **8**, 18–28.

Pérez-Castillo, R., Weber, B., de Guzmán, I. G.-R., Piattini, M. and Pinggera, J. (2014) Assessing event correlation in non-process-aware information systems. *Software & Systems Modeling*, **13**, 1117–1139.

Popova, V., Fahland, D. and Dumas, M. (2015) Artifact lifecycle discovery. *International Journal of Cooperative Information Systems*, **24**, 1550001.

Pourmirza, S., Dijkman, R. and Grefen, P. (2015) Correlation mining: mining process orchestrations without case identifiers. In *International Conference on Service-Oriented Computing*, 237–252. Springer.

Reguieg, H., Benatallah, B., Nezhad, H. R. M. and Toumani, F. (2015) Event correlation analytics: scaling process mining using mapreduce-aware event correlation discovery techniques. *IEEE Transactions on Services Computing*, **8**, 847–860.

Reguieg, H., Toumani, F., Motahari-Nezhad, H. R. and Benatallah, B. (2012) Using mapreduce to scale events correlation discovery for business processes mining. In *International Conference on Business Process Management*, 279–284. Springer.

Richetti, P. H. P., Baião, F. A. and Santoro, F. M. (2014) Declarative process mining: Reducing discovered models complexity by pre-processing event logs. In *International Conference on Business Process Management* (eds. S. Sadiq, P. Soffer and H. Völzer), 400–407. Cham: Springperez2014assessinger International Publishing.

Rodrıguez, C., Engel, R., Kostoska, G., Daniel, F., Casati, F. and Aimar, M. (2012) Eventifier: Extracting process execution logs from operational databases. In *Demonstration Track of BPM Conference, CEUR-WS*, 17–22. Citeseer.

Rozsnyai, S., Slominski, A. and Lakshmanan, G. T. (2011) Discovering event correlation rules for semi-structured business processes. In *Proceedings of the 5th ACM International Conference on Distributed Event-based System*, DEBS '11, 75–86. New York, NY, USA: ACM. URL: `http://doi.acm.org/10.1145/2002259.2002272`.

Sakr, S. and Zomaya, A. Y. (eds.) (2019) *Encyclopedia of Big Data Technologies*. Springer. URL: `https://doi.org/10.1007/978-3-319-63962-8`.

Senderovich, A., Rogge-Solti, A., Gal, A., Mendling, J. and Mandelbaum, A. (2016) The road from sensor data to process instances via interaction mining. In *International Conference on Advanced Information Systems Engineering*, 257–273. Springer.

Soffer, P., Hinze, A., Koschmider, A., Ziekow, H., Di Ciccio, C., Koldehofe, B., Kopp, O., Jacobsen, A., Sürmeli, J. and Song, W. (2018) From event streams to process models and back: Challenges and opportunities. *Information Systems*. URL: `http://www.sciencedirect.com/science/article/pii/S0306437917300145`.

Tax, N., Sidorova, N., Haakma, R. and van der Aalst, W. M. (2016a) Event abstraction for process mining using supervised learning techniques. In *Proceedings of SAI Intelligent Systems Conference*, 251–269. Springer.

— (2016b) Mining process model descriptions of daily life through event abstraction. In *Proceedings of SAI Intelligent Systems Conference*, 83–104. Springer.

Verbeek, H., Buijs, J. C., van Dongen, B. F. and van der Aalst, W. M. (2010) Xes, xesame, and prom 6. In *Forum at the Conference on Advanced Information Systems Engineering (CAiSE)*, 60–75. Springer.

Vogelgesang, T. and Appelrath, H.-J. (2015) A relational data warehouse for multidimensional process mining. In *International Symposium on Data-Driven Process Discovery and Analysis*, 155–184. Springer.

— (2016) Pmcube: a data-warehouse-based approach for multidimensional process mining. In *International Conference on Business Process Management*, 167–178. Springer.

Vogelgesang, T., Kaes, G., Rinderle-Ma, S. and Appelrath, H.-J. (2016) Multidimensional process mining: questions, requirements, and limitations. In *CAISE 2016 Forum*, 169–176.

Walicki, M. and Ferreira, D. R. (2011) Sequence partitioning for process mining with unlabeled event logs. *Data & Knowledge Engineering*, **70**, 821–841.

Wang, H. J., Li, J. and Bai, X. (2012) Towards an intelligent approach to extracting data for process mining. In *SIGBPS Workshop on Business Processes and Services (BPS'12)*, 108.

Weidlich, M., Mendling, J. and Weske, M. (2011) Efficient consistency measurement based on behavioral profiles of process models. *IEEE Transactions on Software Engineering*, **37**, 410–429.

Weske, M. (2019) *Business Process Management - Concepts, Languages, Architectures, Third Edition*. Springer. URL: https://doi.org/10.1007/978-3-662-59432-2.

Zhang, M., Hadjieleftheriou, M., Ooi, B. C., Procopiuc, C. M. and Srivastava, D. (2010) On multi-column foreign key discovery. *Proceedings of the VLDB Endowment*, **3**, 805–814.

Zur Muehlen, M. (2001) Process-driven management information systems combining data warehouses and workflow technology. In *Proceedings of the International Conference on Electronic Commerce Research (ICECR-4)*, 550–566.