Granular Computing (2021) 6:719–736
https://doi.org/10.1007/s41066-020-00226-2

ORIGINAL PAPER

Event abstraction in process mining: literature review and taxonomy

Sebastiaan J. van Zelst[1,2] · Felix Mannhardt[3,4] · Massimiliano de Leoni[5] · Agnes Koschmider[6]

Received: 15 October 2019 / Accepted: 25 April 2020 / Published online: 27 May 2020

Abstract
The execution of processes in companies generates traces of event data, stored in the underlying information system(s), capturing the actual execution of the process. Analyzing event data, i.e., the focus of process mining, yields a detailed understanding of the process, e.g., we are able to discover the control flow of the process and detect compliance and performance issues. Most process mining techniques assume that the event data are of the same and/or appropriate level of granularity. However, in practice, the data are extracted from different systems, e.g., systems for customer relationship management, Enterprise Resource Planning, etc., record the events at different granularity levels. Hence, pre-processing techniques that allow us to abstract event data into the right level of granularity are vital for the successful application of process mining. In this paper, we present a literature study, in which we assess the state-of-the-art in the application of such event abstraction techniques in the field of process mining. The survey is accompanied by a taxonomy of the existing approaches, which we exploit to highlight interesting novel directions.

Keywords Granular computing · Process mining · Sequential data · Label refinement · Event abstraction

1 Introduction

In modern organizations, the execution of business processes is often supported by different information systems. Organizations aim to improve the understandability of their core processes, since this yields improved process performance from different perspectives: reduced lead time, higher revenue, higher customer satisfaction, better compliance with internal/external regulations, etc. *Process mining* provides several techniques to extract actionable knowledge and insights of a process, on the basis of historical execution data (van der Aalst 2016). Within the realm of process mining, *process discovery algorithms* are able to translate the captured event data into a process model, in a (semi)automated fashion. Also, *conformance checking algorithms* allow us to compute whether or not the execution of the process, as recorded in the event data, is in line with a reference model. Furthermore, several techniques exist that allow us to compute insights in the performance of the process, perform root-cause analyses, correlate behavior with different KPIs, improve processes and its models, etc., see, e.g., van der Aalst (2016).

The majority of the available process mining techniques assume that event data are captured on the same level of granularity. However, often, multiple dedicated information systems are used within a company that support different aspects of the business, e.g., customer relationship management (CRM), enterprise resource planning (ERP), etc. Typically, these systems track the different activities executed, i.e., *events*, in the context of the process. However, these systems capture the concept of activities (and

✉ Sebastiaan J. van Zelst
sebastiaan.van.zelst@fit.fraunhofer.de

Felix Mannhardt
felix.mannhardt@sintef.no

Massimiliano de Leoni
deleoni@math.unipd.it

Agnes Koschmider
ak@informatik.uni-kiel.de

1 Fraunhofer Institute for Applied Information Technology, Sankt Augustin, Germany

2 RWTH Aachen University, Aachen, Germany

3 SINTEF Digital, Trondheim, Norway

4 NTNU Norwegian University of Science and Technology, Trondheim, Norway

5 University of Padua, Padua, Italy

6 Kiel University, Kiel, Germany

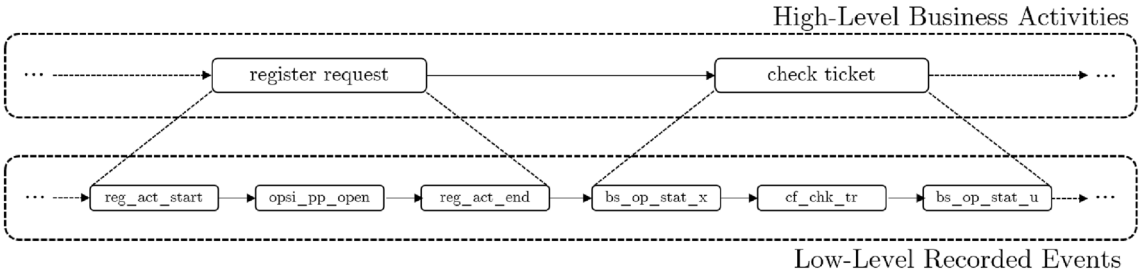


Fig. 1 Example visualization of logging at different granularity levels versus the business activity level. Multiple recorded events constitute a high-level business process activity, e.g., the event sequence ⟨`reg_act_start`, `opsi_pp_open`, `reg_act_end`⟩ corresponds to *register request*

hence events) differently, which might not all be at the same level of abstraction, i.e., the level of granularity might be too fine. Also, different systems within the same organization might not agree on their granularity level, i.e., joining the event data requires us to first ensure the same granularity. As an example, consider Fig. 1, in which we depict a simplified example of such a scenario. In the figure, sequences of *low-level recorded events* correspond to one *high-level business activity*. For example, the first three events, i.e., `reg_act_start`, `opsi_pp_open` and `reg_act_end`, together refer to the high-level *register request* activity. Observe that, the example is oversimplified, i.e., as indicated, different (information) systems might record events at different levels of granularity.

The presence of mixed/overly fine granularity in event logging causes problems in the direct application of process mining techniques. For example, in the context of process discovery, the presence of events in the data leads to the discovery of process models that are often of high complexity, i.e., no longer human interpretable. As an example of such data, consider *click-stream data*. When applying automated process discovery algorithms directly on the raw logged click-stream data, we obtain a process model such as the model depicted in Fig. 2a. Clearly, such a complex process model, severely prevents us from achieving the overall goal of process mining, i.e., an improved understanding of the process. The main reason for obtaining such a high-complex model is the fine granularity at which the user clicks are logged by the server. In this case, it is necessary to group those low-level events to high-level concepts, to discover the actual tasks performed by the users. For example, in de Leoni and Dundar (2020), the authors propose an abstraction technique that allows discovering the process model depicted in Fig. 2b, when applied on the same event data. Clearly, the complexity in this model is greatly reduced, which enables us to gain more insights into the high-level behavior of the users.

As a naive solution, one might be tempted to just focus on the most frequent events, independently of the granularity. However, this typically leads to process models that are severely under-fitting with respect to the data. Hence, we advocate the need of more advanced ways of abstracting fine-granular events into higher level concepts. In process mining, a large share of research has already been conducted on the problem of event log abstraction. However, a concise overview and categorization of these works is lacking. Hence, in this paper, we present a literature survey of event abstraction techniques developed in the context of process mining. We rely on a taxonomy, derived from the existing literature, to provide an overview of event abstraction techniques, which additionally allows us to identify interesting directions for future work.

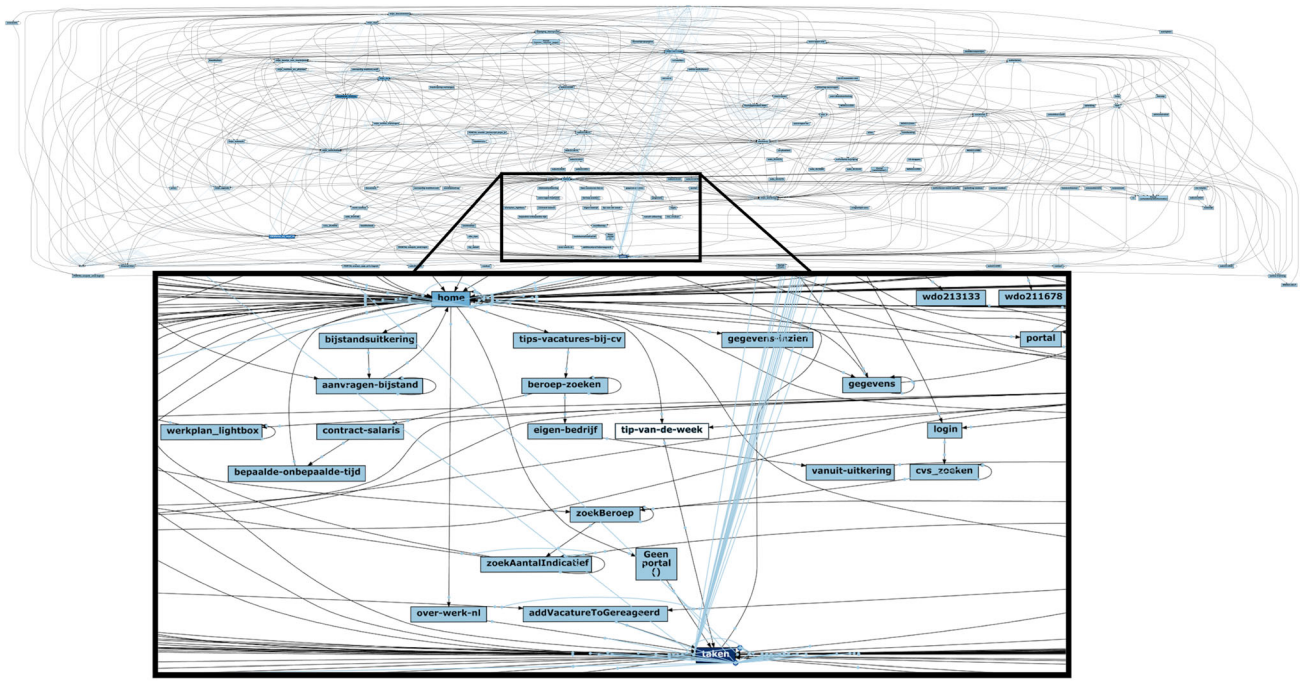
The remainder of this article is structured as follows. In Sect. 2, we introduce the field of process mining. In Sect. 3, we present a newly developed taxonomy for event abstraction techniques in the field of process mining. In Sect. 4, we discuss the existing techniques designed for event abstraction, along the lines of the defined taxonomy. In Sect. 5, we discuss the techniques covered in this work compared to related domains, and, we highlight interesting directions for future work based on underexposed taxonomy dimensions and recent developments in process mining. Section 6 concludes this paper.

2 Process mining

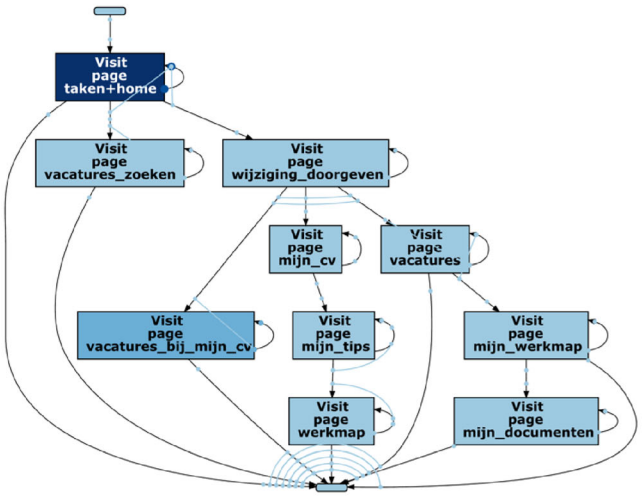
The techniques that we report on in this paper are intended to abstract *event logs*, with the ultimate purpose to achieve better results when subsequently applying process mining techniques. This section introduces event data, i.e., as stored in event logs. Subsequently, we provide an overview of the different aspects of process mining. Finally, we discuss the problems and challenges posed by mixed-granular/low-level event data, in the context of process mining.

2.1 Event logs

Event logs, e.g., Table 1, are the primary input data for any process mining analysis. In Table 1, each row represents an



(a)



(b)

Fig. 2 Two process models, automatically discovered, i.e., using process discovery algorithms, on the basis of logged clickstream data (de Leoni and Dundar 2020). Figure 2a is discovered using the raw logged data, Fig. 2b is discovered using the same data, yet, abstracted to a more coarse-grained view of the data

event, which captures a specific execution of an activity within a *case* (i.e., an instance of the process). Columns are the attributes associated with the events. Mandatory attributes are the *Case Identifier*, which allows identifying to which case the event refers, the *Timestamp* when the event occurred, and the *Activity* that was executed. Other typical attributes are the *Resource* that executed the activity and *Transactional information* indicating the activity's state. Using transactional information one can determine the duration of the activities: The execution of activity *check ticket* for case 12374 started at 12.12 on 30-7-2019, and completed at 14.42, and hence, took 150 min. Other transactional states, e.g., *scheduled*,*active*, etc. exist as

Table 1 An example event log, adapted from van der Aalst (2016), describing behavior related to a compensation request process for concert tickets

Case id	Timestamp	Activity	Resource	Transactional	Cost	···
⋮	⋮	⋮	⋮	⋮	⋮	···
12373	30-7-2019 11.02	Register request	Barbara	Start	50	···
12373	30-7-2019 11.12	Register request	Barbara	Complete	50	···
12374	30-7-2019 11.32	Register request	Jan	Start	50	···
12374	30-7-2019 11.44	Register request	Jan	Complete	50	···
12373	30-7-2019 12.12	Check ticket	Hajo	Start	100	···
12374	30-7-2019 14.16	Examine casually	Jorge	Start	400	···
12375	30-7-2019 14.32	Register request	Josep	Start	50	···
12374	30-7-2019 14.16	Examine casually	Jorge	Complete	400	···
12373	30-7-2019 14.42	Check ticket	Hajo	Complete	100	···
12375	30-7-2019 14.32	Register request	Josep	Complete	50	···
12375	30-7-2019 15.42	Examine thoroughly	Marlon	Start	600	···
12373	03-8-2019 11.18	Examine thoroughly	Barbara	Start	600	···
12375	03-8-2019 12.42	Examine thoroughly	Marlon	Complete	600	···
12373	03-8-2019 15.18	Examine thoroughly	Barbara	Complete	600	···
⋮	⋮	⋮	⋮	⋮	⋮	···

well. Additional process-specific attributes might also be present in an event log, e.g., the *Cost* attribute in Table 1.

2.1.1 Events versus activities

The event log in Table 1 highlights the difference between *events* and *activities*. Activities are the work packages that are instantiated within instances of the process. The history of execution of these activities is stored within event logs for a posteriori analysis. Events represent the records within these event logs. Note that, the execution of one activity in a process instance can be reflected by multiple events, e.g., when event logs record both the start and completion of an activity. The *business activities* (i.e., the concepts known at business level) can differ significantly from the corresponding events stored in the event log, since the underlying system might not operate on the same conceptual level.

2.1.2 Instances versus classes

Aside from the differences between events and activities, we also differentiate between the *instance* and *class* level for activities as well as events. An activity class describes an activity that may potentially be executed for some instance of a process. An activity instance describes the actual execution of an activity of some class. For example, *register request* is an activity that is potentially executed. As such, it represents an *activity class*. However, the effective execution of such an activity (class) is referred to as an *instance* of the activity class. Similarly, an *event class*, describes a potentially observable entity related to the execution of an activity. Likewise, an *event instance* describes the actual recording of an event. For example, the first two event instances listed in Table 1 refer to the event classes that describe starting and completing the *register request* activity, i.e., they are both recorded in the context of the same *register request* activity instance. The number of event classes per activity class, i.e., in terms of observable event instances, depends on the detail at which a system actually keeps track of activity execution.

2.2 Process models

Aside from event logs, another typical input of process mining techniques is a process model. A process model is a *description* of how a process ought to be executed. Compared with a textual description in natural language, a process model encodes (or should encode) the set of allowed executions in a more formal way. Using a formal process modeling notation has several advantages, i.e., it allows us to assess various quality properties of the model, e.g., the absence of deadlocks. It also allows software tools to automatically reason about the modeled process behavior.

Several modeling notations have been put forward (-van der Aalst 2016). However, the current de facto standard in industry is the Business Process Model and Notation (BPMN) formalism (Chinosi and Trombetta 2012), which mediates between being mathematically grounded and human interpretable. Figure 3 illustrates a simple model describing the same process behavior as captured in Table 1. The model describes a few fundamental constructs, often

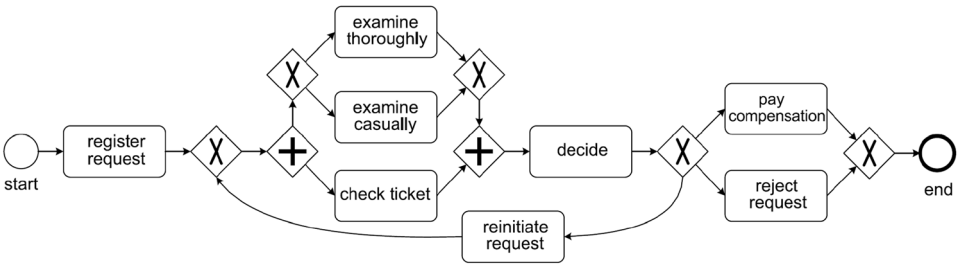


Fig. 3 Example process model in BPMN notation, adopted from van der Aalst (2016). The model describes process behavior in terms of the same high-level business activities as presented in the event log in Table 1

used when modeling a process. The circle with label "start" represents the starting point of the process, whereas the circle with label "end" represents the end point. Each rounded rectangle represents an activity that we are able to execute within the process, e.g., the *decide* activity. The arcs represent the general behavioral flow of the model, e.g., the first activity is *register request*.

The diamond-shaped operators represent behavioral control-flow constructs. The diamonds containing a $\times$-symbol represent an *exclusive choice*, i.e., either one of the outgoing/incoming arcs is followed. The diamonds containing a $+$-symbol represent *concurrency*: all outgoing/incoming arc (and branches) are followed, and, executed completely independently/concurrently. The $+$-symbol with multiple incoming arcs are synchronization points. We refer the reader to Chinosi and Trombetta (2012) for a more elaborate introduction.

The $+$-construct is an important feature present in most process modeling notations that are used in industrial settings. For instance, a block of 10 activities modeled concurrently, yields a total of $10!=3,628,800$ different ways to schedule those activities. When modeling the process as an *automaton*, the resulting automaton is impossible to be interpreted by humans, i.e., not allowing us to achieve the goal of process mining: understanding the process.

2.3 Process discovery and conformance checking

Process mining largely focuses its attention on *process discovery* and *conformance checking*. We briefly introduce these subfields here.

2.3.1 Process discovery

Here, the goal is to automatically discover a process model, on the basis of the event log. In Augusto et al. (2019), the authors report an up-to-date survey of the major, current process-discovery techniques. Process discovery algorithms are typically able to discover a process model such as the one depicted in Fig. 3, using event logs as presented in Table 1. The quality of the discovered model depends on whether or not the input event log contains sufficient behavior and a limited amount of noise.

2.3.2 Conformance checking

Here, the starting point is a process model that describes how the process ought to be executed. Such a process model, i.e., a reference model, is either designed by a human process analyst, or, is the result of the application of a process discovery algorithm on an event log. Conformance checking techniques/algorithms enable verifying to what degree the execution of a process, i.e., as captured in an event log, conforms with respect to the reference model (Carmona et al. 2018). Reconsider the process model depicted in Fig. 3, and a trace of process behavior of the form $\langle$*register request*, *reinitiate request*, *check ticket*, *decide*, $\ldots\rangle$. Conformance checking techniques are able to pinpoint that, e.g., the 2nd event of the trace of behavior was wrongly executed, i.e., according to the model, it is not possible to execute *reinitiate request* directly after the *register request* activity.

2.4 Mixed granular and fine granular events in process mining

Most process mining techniques assumes that event logs record events at the right level of granularity, i.e., matching the concepts known at business level, e.g., Table 1. However, as indicated, this is often not the case. As an example, it could be the case that the execution of a real process, e.g., behaving as modeled in Fig. 3, generates the event log in Table 2. This event log is clearly not matching the business concepts in the model. For example, the first three events in Table 2, `reg_act_start`, `opsi_pp_open` and `reg_act_end` correspond to activity *register request* executed by Barbara.

Several challenges exist when applying process mining algorithms on the basis of fine-granular event logs, i.e., event logs such as exemplified in Table 2:

Process Discovery	Fine-granular events typically generate complex process models, e.g., in Fig. 2a. As indicated, these models lose their purpose to convey actionable knowledge of the process being analyzed.

Table 2 Example event log describing behavior related to a compensation request process for concert tickets (i.e., similar to Table 1), depicting how the events are typically logged in an information system

Case id	Event id	Timestamp	Event Description	Resource	Cost	⋯
⋮	⋮	⋮	⋮	⋮	⋮	⋯
12373	45632X1i	20193007T1102	reg_act_start	null	15	⋯
12373	45632X2i	20193007T1102	opsi_pp_open	112AA12	null	⋯
12373	45632X3i	20193007T1102	reg_act_end	null	35	⋯
12374	45633X1i	20193007T1132	reg_act_start	null	15	⋯
12374	45633X2i	20193007T1132	opsi_pp_open	1333A27	null	⋯
12374	45633X3i	20193007T1132	reg_act_end	null	35	⋯
12373	45634YZ1	20193007T1212	bs_op_stat_x	null	null	⋯
12373	45634YZ2	20193007T1212	ch_chk_tr	0093B74	100	⋯
12373	45634YZ3	20193007T1212	bs_op_stat_u	null	null	⋯
⋮	⋮	⋮	⋮	⋮	⋮	⋯

Conformance Checking In conformance checking, one aims to explain the observed behavior in an event log in terms of a reference model. Clearly, if there is no exact match between the captured events and the modeled activities, conformance checking is not properly applicable.

The example fine-granular event log in Table 2 illustrates some additional challenges. For example, the costs recorded for the first event of Table 1 is built up out of two logged cost data attributes on the lower level, i.e., of event id `45632X1i` and `45632X3i`. Furthermore, a mapping of the resources tracked in the event data, e.g., `112AA12` corresponding to *Barbara*, is performed as well. However, in the remainder of this paper, we are primarily concerned with translating observed sequences of fine-granular events to sequences of coarse-granular events, i.e., events describing activities at the business level.

3 Event abstraction in process mining

To structure the literature review on event abstraction, we first present the notion of event abstraction itself. In Sect. 3.2, we present the methodology followed in taxonomy construction. In Sect. 3.3, we present and discuss a taxonomy of event abstraction techniques in process mining. In Sect. 4, we discuss existing work on event abstraction along the lines of the taxonomy presented here.

3.1 Event abstraction

We assume that the techniques considered in this paper translate (multiple) instances of fine-granular events into instances of coarse-granular events, i.e., representing a level of detail that is closer, or equal to, the level of detail at which one aims to analyze the process. Our work is focused on event abstraction methods addressing the mapping from fine-granular events to coarse-granular events and, optionally, their connection to activity instances.

Typically, given a sequence of fine-granular events σ, after applying an event abstraction technique α, we obtain a sequence σ' of events, i.e., $\alpha(\sigma)=\sigma'$, at a coarser granularity level. Typically, we have $|\alpha(\sigma)| \leq |\sigma|$, i.e., $\alpha(\sigma)$ is intended to be significantly shorter/less complex than σ. We assume σ to be a discrete sequence of events, however, not all techniques covered in this study work on discrete event sequences, i.e., some techniques transform continuous data to coarse granular events. Nonetheless, most techniques, i.e., potentially after applying some pre-processing on their primary output, yield or provide means to obtain, a discrete sequence of coarse-granular events, on the basis of a given discrete sequence of fine-granular events.

Event abstraction as defined and considered in this paper can be seen in the broader spectrum of a larger hierarchy linking observations from the physical world to meaningful activity instances. Consider Fig. 4, in which we sketch this hierarchy (Koschmider et al. 2018). Within the figure, the cloud represents observations in the physical world, translated to "low-level events," i.e., *fine-granular events* in our context. Subsequences of said low-level events are subsequently translated into "high-level events," i.e., *coarse-granular events*, by means of applying abstraction and/or aggregation techniques, i.e., *event abstraction* in our context. In some cases, even the activity instances, i.e., as described by the high-level events, are not directly related to one-another. In such cases, additional correlation techniques need to be used as well, i.e., to determine which

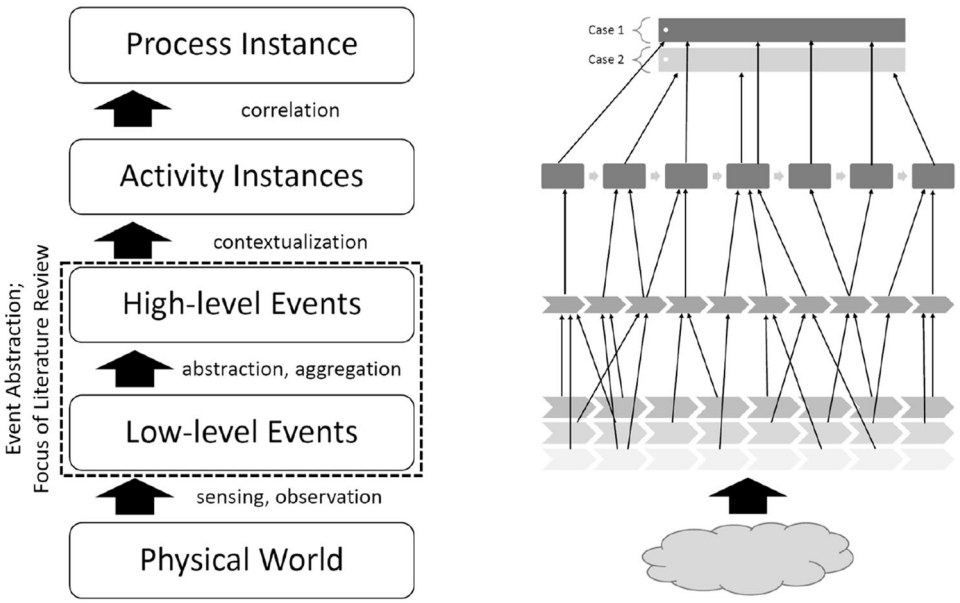


Fig. 4 Hierarchy of mappings (Koschmider et al. 2018); from raw data captured in the physical world, to activity instances that can be correlated to process instances

activity instances belong to the same process instance. However, in this paper, we do not consider such event correlation challenges.

3.2 Taxonomy model and construction strategy

In this section, we briefly detail on the methodology followed to construct the taxonomy presented. We adopt the model of taxonomies as defined in Nickerson et al. (2010), i.e., a taxonomy T consists of a collection of n dimensions $D_1, \ldots D_n$. A dimension D_i consists of $k_i \geq 2$ characteristics $C_{i,1}, \ldots, C_{i,k_i}$. These characteristics, ideally, are mutually exclusive and collectively exhaustive. According to Nickerson et al. (2010), taxonomy construction is either *inductive* or *deductive*. In an inductive approach, the authors propose a set of dimensions based on their understanding of the field and classify existing work accordingly. In a deductive approach, the set of dimensions is derived from existing work in the field. Here, we primarily follow a deductive approach, however, in some cases, the taxonomy is extended based on domain knowledge of the authors. The study inclusion criteria within taxonomy construction and the corresponding literature survey are presented in Sect. 4.1.

3.3 A taxonomy of event abstraction methods

In this section, we present a taxonomy that helps in classifying event abstraction methods in the field of process mining. Based on the literature assessed, we identify 7 different dimensions: *supervision strategy*, *fine-granular event interleaving*, *probabilistic nature of the outcome*, *data nature*, *use of alternative perspectives*, *event class/activity class relation* and *event instance/activity instance relation*. In the upcoming sections, we explain each dimension in detail and discuss the different categories of each dimension.

3.3.1 Supervision strategy

In line with common machine learning terminology, we identify two main categories of supervision strategy used by event abstraction techniques: *supervised* and *unsupervised*. In a supervised scenario, the event abstraction technique expects some form of additional input, e.g., a labeled event log, a reference model, a set of possible activities, etc., that is used to guide the event abstraction algorithm in order to translate the observed fine-granular events. In case an algorithm is unsupervised, no form of additional input is required for the algorithm to translate the fine-granular events into events of coarser granularity. We do not distinguish between weak/strong forms of supervision.

3.3.2 Fine-granular event interleaving

The dimension of *fine-granular event interleaving* refers to the capability of event abstraction techniques to handle true concurrency on the higher granular level. This implies that a technique that is able to handle true concurrency is able to map a group of fine-granular events, which potentially occur interleaved with other fine-granular events, to the same coarse-granular event instance. Consider Fig. 5, in which we visualize this concept in terms of the running example. Within this dimension, we identify two main categories, i.e., *strictly sequential-* and *interleaved*

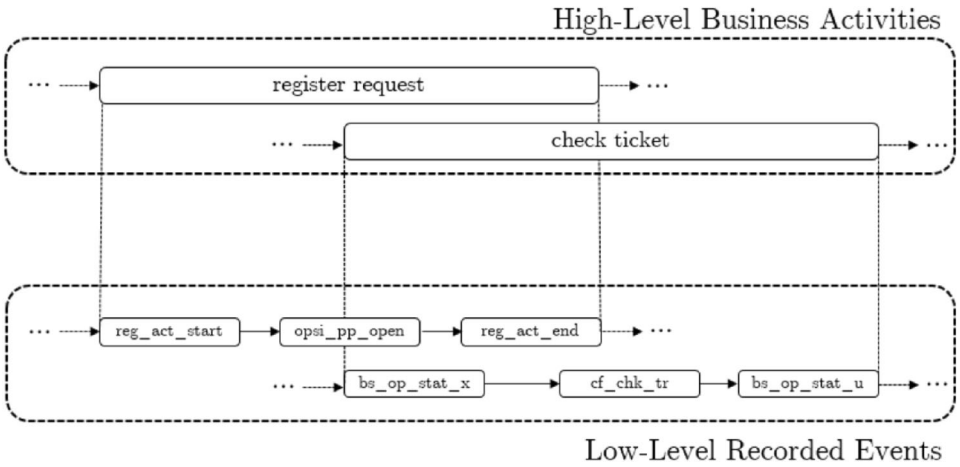


Fig. 5 Example of fine-granular events which are interleaving with each other. Some abstraction techniques support this, whereas other techniques only support strictly sequential mapping, cf. Fig. 1

techniques. Strictly sequential techniques only allow us to map strict sub-sequences of fine-granular events to a coarse-granular event, e.g., as examplified in Fig. 1. Interleaved techniques allow for event interleaving, e.g., as exemplified in Fig. 5. Interleaved techniques, by definition, support the strictly sequential case. However, as interleaved techniques allow us to express true concurrency on the high level, we still differentiate between these two categories.

3.3.3 Probabilistic nature of outcome

Some of the approaches that handle the mixed granularity problem, internally use probabilistic models. In some cases, the usage of an internal probabilistic model is reflected in the output. Rather than returning a sequence of coarse-granular events, these algorithms return a more probabilistic result, e.g., a sequence of probability distributions over high-level events. As such, we distinguish purely *deterministic* approaches versus *probabilistic* approaches. Here, the probabilistic nature is defined on the output of the event abstraction technique. In some cases, techniques use probabilistic models internally, yet, the resulting outcome is always a sequence of coarse-granular events. Hence, in such a case, we categorize the technique as being deterministic.

3.3.4 Data nature

We assume the event abstraction techniques to translate discrete sequences of fine-granular events into sequences of coarse-granular events. However, some work assumes continuous data to originate from the execution of a process, which is aimed to be translated into sequences of discrete, coarse-granular events. For example, consider the analysis of human behavior in processes by means of analyzing motion sensor data. Whereas this refers to the first mapping challenge in the lower-left part of Fig. 4, the results are coarse-granular events. Hence, we distinguish between techniques that work on *discrete* event data and techniques that are able to handle *continuous/sensor* data.

3.3.5 Alternative perspectives

In Tables 1 and 2, we depict additional data attributes for events, e.g., the associated resources and costs of the executed activities. Some event abstraction techniques explicitly exploit such additional information for the purpose of event abstraction. In terms of the taxonomy, we define a strictly binary distinction between event abstraction techniques, i.e., a technique either exploits additional information, or not. In case a technique does not exploit any additional information, we assume the technique to just consider the event data from a *control-flow perspective*, i.e., solely considering/using the captured sequences of fine-granular events. For the techniques that exploit additional perspectives, we further identify the following non-exclusive perspectives:

- *Time*; Event time-stamps are omnipresent in event data and allow us to order the observed events. However, some event abstraction techniques, apart from using sequential ordering of events, explicitly exploit timing information in the event data, e.g., exploiting periods of process inactivity to determine the stop criterion of a high-level activity instance.
- *Resources*; Another data attribute that is often present relates to resource information, e.g., which worker caused the low-level event to be recorded. Some techniques exploit this information to classify membership relationship of low-granular events to higher-level activity instances.
- *Additional Available Event Payload (AAEP)*; Some techniques allow for expressing behavioral patterns in

terms of fine-granular events, which relate to higher-level concepts. In some cases, these patterns include constraints on arbitrary event data attributes, e.g., activity costs. This extends the capabilities of methods to correlate events with activity instances, by using additional contextual information.

3.3.6 Event class/activity class relationship

As explained in Sect. 2.1, we differentiate between events and activities. Typically, an event abstraction technique translates sequences of fine-granular events to coarse-granular events, which, in turn, represent instances of high-level activities. Some techniques only allow us to map one event class to one activity class, e.g., in terms of Fig. 5, such techniques only allow us to map *any observed instance of* event class `reg_act_start` to coarse-granular event instances representing the higher level *register request* activity class. Hence, the mapping is fixed on a class level. Other techniques are less restrictive, i.e., they allow a single event class to be related to multiple high-level activity classes. This is sometimes referred to as *shared functionality* (Baier et al. 2014). We distinguish between 1:1, *n*:1 and *n*:*m* mappings. Observe that, these mappings are not necessarily mutually exclusive, i.e., *n*:*m* includes both 1:*n* and 1:1. However, due to the explicit difference in expressive power between the different types of mappings, we identify them as separate categories.

3.3.7 Event instance/activity instance relation

In line with the event class/activity class relationship, we define a similar dimension for the actual instances of events/activities. We distinguish between techniques that allow us to map a fine-granular event instance to multiple coarse-granular event/activity instances, and techniques which do not allow for this. We distinguish between 1:1, *n*:1 and *n*:*m* mappings. Observe that, in case of 1:1 mapping, we do not effectively apply event abstraction, i.e., each observed event directly corresponds to an executed activity/coarse-granular event. As such, after applying a technique describing a 1:1 mapping, the level of granularity of the data remains unchanged.

4 Literature review

Having defined the taxonomy dimensions, we now discuss the relevant literature on event abstraction techniques in the domain of process mining. In Sect. 4.1, we briefly outline the data collection and study inclusion strategies followed. In Sect. 4.2, we discuss the relevant work along the different dimensions identified in the taxonomy. For each dimension, we discuss what category is most commonly represented, and, highlight noteworthy alternative approaches. A schematic overview and classification of the literature, covered by this review, is presented in Table 3.

4.1 Methodology

In this section, we describe the methodology adopted during the conduction of the literature review. We first highlight the data collection phase, after which we briefly mention inclusion criteria for the selected studies.

4.1.1 Data collection

As we are primarily interested in studies that originate from the area of process mining, i.e., assuming that the data is collected during the execution of a process, we used the following query to identify appropriate publications:

```
(‘‘event abstraction’’ OR ‘‘event and activity abstraction’’ OR
‘‘event and activity matching’’ OR ‘‘low-level event log’’ OR
‘‘unsupervised event abstraction’’ OR ‘‘supervised event abstraction’’)
AND ‘‘process mining’’.
```

To find synonyms for the term *event abstraction* we checked all citations of papers that are related to process mining, having this term in the title. We always checked the first 100 hits of each query and searched the databases *SpringerLink*, *ScienceDirect* and the *ACM Digital Library* for related literature. When there were more hits, we checked them as long as no matching publication was found on three consecutive result pages. We also used Google Scholar to find appropriate literature by browsing the citations of related publications already found in the scientific databases. Secondly, we conducted a backward search to find more appropriate publications cited in papers of the first search round. The result list was cross-checked with two commonly known publications on event abstraction in process mining (Baier et al. 2014; Tax et al. 2016). Moreover, we added six papers, which were not found in the literature search, but were known to the

Table 3 Classification of work (sorted chronologically) in the field of event abstraction, along the taxonomy dimensions identified in Sect. 3.3

Article	Year	Dimension										
		Supervision		Interleaving		Probability		Data		Alt. Perspectives	EC:AC	EI:AI
		Un.	*Su.*	*Se.*	*In.*	*Pr.*	*De.*	*Di.*	*Co.*			
Bose and van der Aalst (2009)	2011	✓		✓			✓	✓		None	n:1	n:1
Günther et al. (2009)	2009	✓			✓		✓	✓		None	n:1	n:1
Ferreira et al. (2013b)	2013		✓	✓			✓	✓		None	n:1	n:1
Baier et al. (2014)	2014		✓		✓		✓	✓		AAEP	n:m	n:1
Folino et al. (2015)	2015	✓		✓			✓	✓		AAEP[a]	n:1	n:1
van Eck et al. (2016)	2016		✓[b]	✓			✓		✓	None	n:m	n:1
Senderovich et al. (2016)	2016		✓		✓		✓	✓		Resource	n:m	n:1
Begicheva and Lomazov (2017)	2017		✓		✓		✓	✓		None	n:1	n:1
Leonardi et al. (2017)	2017		✓		✓		✓	✓		Time	n:m	n:1
Mannhardt and Tax (2017)	2017	✓			✓		✓	✓		None	n:m	n:1
Sánchez-Charles et al. (2017)[c]	2017	✓			✓		✓	✓		None	n:1	n:1
Alharbi et al. (2018)	2018	✓		✓			✓	✓		None	n:m	n:1
Baier et al. (2018)	2018		✓		✓		✓	✓		None	n:1	n:1
Bernard and Andritsos (2018)	2018		✓		✓		✓	✓		None	n:m[d]	n:1
Fazzinga et al. (2018a)	2018		✓		✓	✓		✓		Time	n:m	n:1
Fazzinga et al. (2018b)	2018		✓	✓		✓		✓		None	n:m	n:1[e]
Mannhardt et al. (2018b)	2018		✓		✓		✓	✓		AAEP	n:m	n:1
Tax et al. (2018)	2018		✓	✓			✓	✓		Time, Resource	n:m	n:1
Rehse and Fettke (2019)	2019	✓		✓			✓	✓		None	n:1	n:1
Tello et al. (2019)	2019		✓	✓			✓	✓		Time	n:m	n:1
de Leoni and Dundar (2020)	2020		✓		✓		✓	✓		None	n:m	n:1

For some works, we combine initial and extended work. We refer to the year of publication of the latest work

[a]The approach does solely use the data attributes attached to events and does not make use of the control-flow perspective.

[b]The approach uses a provided window size to create segments of sensor data. However, a clustering algorithm (unsupervised) is used to merge similar clusters. The identified clusters need to be labeled by a human expert.

[c]The technique presented is inspired by event names/classes captured in natural language, yet, is generally applicable.

[d]Examples provided in the paper do not show the n:m relation, however, such relation is theoretically possible.

[e]The n:1 mapping of event instances to activity instances is only sketched in this work

authors. Eventually, we ended up with 28 relevant publications. Related publications were found in the following databases: SpringerLink (65%), ScienceDirect (5%) and ACM Digital Library (36%). The 28 publications were published in the following years: 2009 (7.14%), 2011 (3.5%), 2012 (3.5%), 2013 (7.14%), 2014 (3.5%), 2015 (10.71%), 2016 (14.3%), 2017 (14.3%), 2018 (21.4%), 2019 (10.71%), 2020 (3.5%). These numbers indicate an increasing interest in the topic.

4.1.2 Study inclusion

Some of the event abstraction techniques developed in the context of process mining bare great similarity compared to techniques developed in other areas, e.g., the area of complex event processing (CEP). In Sect. 5, we discuss these similarities in more depth. However, in process mining, a key assumption is the existence of some underlying process, for which multiple instances are executed. In the majority of pattern detection methods originating from different domains, such assumption is not met. Hence, direct adoption of such methods in the context of process mining potentially leads to combining inter-case events into the same pattern. Therefore, as a criterion for inclusion within this paper, we require the work described in a paper to explicitly assume the existence of a process. Furthermore, we filtered the resulting list and included publications according to the following criteria.

- We only include work if it was not superseded by a more recent revised version in which case we only considered the latest publication.

- We do not include work if the publication did not address an underlying model. For example, in Baier et al. (2015), a method is proposed to find the best matching activity name (represented by a coarse-granular event) to observed fine-granular events. Hence, the method assumes that the events observed are in fact at the same level of granularity as the activities in which the process model is expressed.
- We do not include work if the presented approach allows to only translate an event instance to exclusively one activity/high-level event instance, i.e., a 1:1 mapping on the event instance/activity instance dimension, which is not suitable for event abstraction in process mining.
- We exclude work describing a high-level application of event abstraction, i.e., not a specific approach, e.g., Brzychczy and Trzcionkowska (2019).
- We exclude publications primarily focusing on event correlation, e.g., Montahari-Nezhad et al. (2011) and Pérez-Castillo et al. (2012);
- We excluded techniques that are described incompletely and/or techniques that potentially could be used for event abstraction, yet, a detailed study has not been performed. For example, in Folino et al. (2014), event abstraction is discussed in the context of process performance prediction, however, it is unclear if the approach can provide a coarse-granular event log for use in generic process mining tasks. In Richetti et al. (2014), event abstraction using semantic similarity of fine-granular events is proposed, yet, the actual abstraction phase is not described. Similarly, in Nguyen et al. (2019), the authors propose to automatically detect *stages* of a process, yet, focus on applying process discovery on sublogs created for said stages, at the same granularity level as the initial input.

After applying study exclusion (including removal of work that is superseded by follow-up work), 21 articles are considered in the review.

4.2 Review

In this section, we discuss the identified event abstraction techniques in the area of process mining, along the lines of the dimensions of the deduced taxonomy. The supervision dimension is the most variable dimension, i.e., techniques differ greatly in this domain. Hence, we highlight all works in this section in detail, in a chronological fashion per supervision type, i.e., unsupervised and supervised. For the other dimensions, the variability of the different approaches is less prominent, hence, for these dimensions, we briefly discuss what is the most common category among the different works identified, and, we highlight interesting applications, i.e., techniques falling into a less common category.

4.2.1 Supervision strategy

Whereas the majority of techniques uses some form of supervision strategy to group fine-granular events into coarse-granular events, the first two event abstraction techniques identified (chronologically) work in an unsupervised manner. Hence, we first turn our focus toward techniques implementing an unsupervised strategy, after which we turn our focus to supervised techniques. We start each paragraph with a summary of the different techniques, after which we present each technique covered in the study in more detail.

4.2.2 Unsupervised techniques

The unsupervised techniques all aim to group the fine-granular events on the basis of *strong re-occurrence of patterns*. In some cases, these patterns are simply repeating (strict) sub-sequences. In other cases, these patterns are more advanced, e.g., imperative process models that describe sub-sequences of fine-granular events. In some cases, the techniques cluster (sometimes iterative) patterns into new coarser-grained events.

In Bose and van der Aalst (2009), the authors consider the notion of coherent sub-sequences of behavior. However, no clusters are discovered in this techniques, i.e., repeating local execution patterns, e.g., tandem arrays or maximal repeats, are discovered and then abstracted to higher-level activities. In Günther et al. (2009), the authors propose to learn coherent sub-sequences of event instances, i.e., referred to as *trace segmentation*, which are used to create coarse-granular events. Each event and/or trace segment is assigned to a cluster. Internally, a hierarchy of clusters is constructed, s.t., in the end, all events belong to the root cluster of the hierarchy. In Folino et al. (2015), the authors propose an unsupervised event abstraction techniques, which is subsequently followed by a trace-level clustering. Predictive clustering trees are used to cluster low-level events, i.e., considered as being sets of attributes, into hierarchical clusters so that each cluster corresponds to an activity type for which the low-level events are recorded. Clustering is based on repeatedly checking whether the workflow schema (process model) extracted, when applying the clustering, improves two quality measures defined in the paper.

In Mannhardt and Tax (2017), more advanced methods to find patterns are used. In this work, frequent local process models, i.e., process models that describe just a fraction of an observed fine-granular trace, are discovered and subsequently used as a basis for abstraction. In Sánchez-

Charles et al. (2017), the authors propose to use *word-embedding* techniques to map subsequences of fine-granular events into coarse-granular events. The authors propose to do so, in order to group events that have a semantically similar name. In Alharbi et al. (2018), the authors propose to learn high-level activities by discovering hidden Markov models. The number of states of these models is computed using log likelihood, after which the Viterbi algorithm (Forney 1973) is applied to extract sequences of states corresponding to coarse-granular events. Finally, in Rehse and Fettke (2019), the authors propose to compute the "spatial proximity" between fine-granular activities, i.e., essentially grouping events that often co-occur. Subsequently, a hierarchical clustering of events is computed, which forms the basis for coarse-granular event recognition.

4.2.3 Supervised techniques

The supervised event abstraction techniques generally use three different types of supervision artefacts. Some use a time interval to determine groups of fine-granular events. Each group typically relates to a distinct higher-level activity. A fairly limited number of techniques assumes process analysts to select a small portion of event data on which they provide a manual mapping from fine-granular events to higher-level activities. The mapping is then used as a knowledge base to reason on the large portion of event data. The majority of the techniques assumes some form of reference model, i.e., ranging from loosely specified domain knowledge to more advanced/formal models. Typically, these models are used as a basis to apply matching of fine-granular patterns.

In Ferreira et al. (2013a), the authors expect the user to provide a *macro-model* as an input, next to a sequence of fine-granular events. The macro-model is represented by a Markov model, expressed over the high-level activity classes. As such, the user of the technique is expected to know the basic high-level activities and their relationship. In Baier et al. (2014), the authors propose to use domain knowledge to guide the event abstraction algorithm. In particular, the technique expects an input process model together with a description, which is transformed into an annotated process model. The annotated process model, in turn, is used to define event-activity relations for the input event log, i.e., the event log on which the abstraction is applied.

In van Eck et al. (2016), the authors translate continuous sensor data into higher-level events. The only input required is a minimal window size, in which the input sensor data needs to be segmented. Clustering is subsequently used to group segments, and, based on cluster properties, generate implicit labels for the observed events. Optionally, such implicit labels can be refined by a domain expert. Similarly, Senderovich et al. (2016) propose an event abstraction technique for discrete sensor data. The sensor data (location data) is translated to a collection of interactions. Given an interaction pattern, sets of interactions are grouped that match the given pattern. Additional process knowledge, e.g., a person cannot be in two locations in the same time, is encoded in an ILP, which is used to translate the observed interactions into high-level activity instances.

In Begicheva and Lomazov (2017), the authors assume that each coarse-granular activity class is represented by the execution of a sub-process, tracked at the fine-granular level. Hence, such a mapping needs to be initially given, in order to effectively apply the event abstraction. The fine-granular events are replaced by corresponding coarse-granular activities, which are further translated into a process model. In Leonardi et al. (2017), the authors use domain knowledge, in the form of *ontologies*, to translate fine-granular events into coarse-granular events. The fine-granular events are mapped to the ground terms of such ontologies, after which a rule-base is exploited to learn higher-level concepts. In Baier et al. (2018), the authors propose to match fine-granular events to an existing process model of the whole process, by solving constraint satisfaction problems that are built based on sets of declarative constraints discovered on both the process model and the event log. Further disambiguation of the matches is performed by integration of domain knowledge, i.e., by consultation of human experts.

In Fazzinga et al. (2018a), the authors abstract fine-granular events on the basis of a given high-level process model. Each possible interpretation of the fine-granular trace, in terms of coarse-granular events, is computed with an associated probability distribution. In Fazzinga et al. (2018b), the authors use a hidden Markov model that models the generation of low-level events. The frequency of precedence relations between high-level activities are inferred. To do so, the authors use candidate mappings, based on activity dependencies. In Mannhardt et al. (2018b), the authors propose to capture behavioral patterns as *Data Petri nets*, i.e., given by the domain expert. The technique subsequently finds these behavioral patterns in the fine-grained data, and, replaces them with a coarse-granular activity instance. In Bernard and Andritsos (2018), the author propose to use *process trees*, i.e., a subclass of Petri nets, to transform fine granular event sequences onto coarser-level activity instances. In particular, the authors propose to use event abstraction in the domain of customer journey mapping.

[1] Note that, in some cases, these internal representations are used as a supervision model, yet, this is not always the case.

In Tax et al. (2018), the authors propose to use a supervised learning approach on the basis of *conditional random fields*. The approach expects a set of annotated traces in which each fine-granular event instance has a corresponding coarse-granular event instance. On the basis of the annotated traces, a conditional random field is trained, which is used on a collection of unlabeled fine-grained events to apply event abstraction. In Tello et al. (2019), the authors propose a framework that first aims to distinguish segments in traces. For such a segment separation, an optimization approach is used that requires a "segmentation list" for the separation problem. The segmentation list, essentially, is a set of relatively small examples. Next, the separated traces are provided as an input to a clustering algorithm that is used to create high-level activity labels. Finally, in de Leoni and Dundar (2020), the authors propose to create "sessions" of fine-granular events. These sessions correspond to time periods, which are defined on a minimum time of inactivity, i.e., if no events are observed for at least a time-period Δ, a new sessions starts.

4.2.4 Fine-granular event interleaving

In terms of fine-granular event interleaving, we observe that the distribution of the relevant work is relatively mixed. Typically, work that supports interleaving within the fine-granular event representation uses some form of internal model that is able to capture concurrency, in order to represent the higher level events.[1] For example, Petri nets (Murata 1989) are used internally to map the fine-granular events into coarse-granular events (Mannhardt and Tax 2017; Mannhardt et al. 2018b). Also, more declarative process modeling languages are used for this purpose (Baier et al. 2018). Similarly, techniques that use ontologies (Leonardi et al. 2017) are able to use an underlying engine to map fine-granular events in an interleaved manner. In case of strictly sequential techniques, it is harder to pinpoint a commonly used internal model type. For example, in Tax et al. (2018), the authors propose to use linear-chain conditional random fields (CRFs) to represent sequences of sensor data. Likewise, in Alharbi et al. (2018) and Fazzinga et al. (2018b), the authors propose to use hidden Markov chains as an internal representation.

4.2.5 Probabilistic nature of outcome

The vast majority of the work results in a deterministic output, i.e., a sequence of coarse-granular events. Notable exceptions to the standardized outcome specification are Tax et al. (2016), Fazzinga et al. (2015, 2018a, b). Tax et al. (2016) return an estimated Gaussian mixture model (GMM) for each tuple of lifecycle steps for a certain activity (start and complete). The estimation is based on the time differences between start and complete lifecycle steps for an activity. In Fazzinga et al. (2018b), the authors propose to use a hidden Markov model that models the generation of fine-granular events, which is used to infer the frequency of precedence relations, rather than coarse-granular events. In Fazzinga et al. (2015, 2018a), the authors aim to find all possible interpretations of the fine-granular trace, yet, finally returning a compact representation of these interpretations.

4.2.6 Data nature

Clearly, most techniques covered in this work are using discrete data. However, van Eck et al. (2016) is a notable exception, i.e., the only work applicable on continuous data. In van Eck et al. (2016), the authors propose to apply process mining on sensor measurement data. The stream of sensor data, which might be multivariate, is first segmented using a given window size. Subsequently, for each segment, relevant features are calculated, which are used to cluster the different segments. These clusters need to be labeled by domain experts. Interestingly, the output of this first step is a sequence of potentially (yet not necessarily) fine-granular events, i.e., any of the techniques working on discrete data are applicable on the output of this step. Other work exists which covers the application of process mining on continuous/sensor data as well (Brzychczy and Trzcionkowska 2019). However, this work does not present a generic method and/or technique for the purpose of event abstraction, i.e., translation of the sensor signal is performed manually. Another notable work is Senderovich et al. (2016), which propose to translate interactions, e.g., nurses providing a treatment to a patient, into coarse-granular events. In this regard, the work is not directly applicable on arbitrary fine-granular events, yet, relevant in the context of event abstraction. Finally, in Sánchez-Charles et al. (2017), the authors propose a technique inspired by event classes in natural language, i.e., event log complexity is reduced by means of learning word embeddings.

4.2.7 Alternative perspectives

Interestingly, the vast majority of the techniques covered only uses the control-flow perspective, i.e., solely the sequence of events is considered, despite the opportunity to leverage additional information recorded as event payload. Using additional information that encodes the context in which an event is recorded may help to map events to the

[1] Note that, in some cases, these internal representations are used as a supervision model, yet, this is not always the case.

correct activity instance. Baier et al. (2014) were the first to allow the specification of rules over additional data attributes. Here, so-called *attribute conditions* are defined over the event payload of surrounding events, which are then combined with rules based on the control-flow perspective and allow to restrict the number of events that may be matched to a specific activity instance. However, in case, multiple conditions match a single event instance, all matches are considered valid. A different approach is taken in Folino et al. (2015). The authors propose to first cluster the event log, while disregarding the control-flow perspective by considering an event as a set of data attributes. The data perspective takes precedence on the decision which events to cluster together, and, control flow is only implicitly taken into account through the clustering. As an improvement over the work of Baier et al. (2014), the approach presented in Mannhardt et al. (2018b) allows to exploit arbitrary data by using Petri nets with variables and data expressions as the internal model to represent patterns of fine-granular events. Here, the ambiguity of multiple conflicting rules being activated for the same fine-granular event is resolved by solving a global optimization problem, i.e., at the cost of higher computational complexity. Several other methods have been exploiting specific perspectives, e.g., the *time* and *resource* perspective. For example, in Senderovich et al. (2016), the discovered mappings are based on interactions between resources, i.e., heavily relying on the resource perspective. In Leonardi et al. (2017), the authors incorporate the time perspective in an ontology-based approach. The conditional random fields used in Tax et al. (2016) include both the time and resource perspective. Notably, the work in Fazzinga et al. (2018a) supports the time dimension in combination with providing a probabilistic output.

4.2.8 Relation between event classes and activity classes

Concerning the relationship between event classes and activity classes, most recent works support the full *n:m* mapping, i.e., the challenge posed by shared functionality, first raised in Baier et al. (2014). A single event class may, in these works, relate to different activity classes depending on the context is solved. This is in contrast to early work that does not allow for such ambiguity in the input data, i.e., a certain event class is always linked to the same higher level activity, i.e., a *n:1* mapping is used.

4.2.9 Relation between event instances and activity instances

Opposed to the class relationship between events and activities, on an instance level, we observe that almost all techniques are limited to an *n:1* relationship model. This means that, whereas event classes may be shared between activities, individual events cannot be part of more than one activity instance. This is restriction limits certain scenarios, e.g., when the same sensor is triggered by two different activities at the same time. A notable exception is Tax et al. (2016), where, due to its probabilistic nature, i.e., as described previously, an event instance is able to (partially) belong to multiple high-level concepts. However, this is not extensively evaluated nor further discussed in the paper.

5 Discussion

In this section, we discuss different fields and/or groups of techniques that bear similarities with the techniques presented in this paper. We discuss *granular computing*, *complex event processing*, *activity recognition* and *(process mining specific) data pre-processing techniques*. Furthermore, we present interesting unexplored dimensions as well as potential novel dimensions to be considered.

5.1 Related fields of study

In this section, we discuss the similarities of event abstraction in process mining with respect to several related fields.

5.1.1 Granular computing

The event abstraction techniques covered in this literature survey clearly follow the basic conceptual principles of granular computing (Pedrycz 2001; Bargiela and Pedrycz 2016). In particular, the works considered in this paper can be regarded as abstraction mechanisms that reduce the conceptual burden of understanding the information carried by the data, i.e., as exemplified by Fig. 2. The majority of the work considered here translates data at the information-granule level (fine-grained events) to the symbolic level (coarse-grained events).

Despite the clear relationship between granular computing and event abstraction, typical models used by techniques in the domain of granular computing are not necessarily adopted in state-of-the-art event abstraction techniques. Concepts originating from the domain of fuzzy logic are often applied in granular computing, yet, have not at all been considered in event abstraction. For example, (Chiang et al. 2018) use fuzzy Petri nets (Looney 1988), in order to analyze electroencephalogram (EEG) signals for the purpose of sleep quality measurement. Similarly, in Liu and Zhang (2018); Liu et al. (2018) principles of fuzzy logic are used for classification

problems in cases where class labels are not necessarily mutually exclusive.

5.1.2 Complex event processing (CEP)

Methods based on complex event processing (Cugola and Margara 2012) typically assume a stream of events, over which queries are evaluated. When a query is matched a high-level activity is detected. Traditionally, CEP does not consider the notion of process instance (i.e., a case) and in case of overlapping queries (e.g., shared functionalities or n:m relations on a event/activity type level) both high-level activities are detected. There are some applications using the CEP paradigm in a business process context, e.g., CEP is used for process monitoring in Oliveira et al. (2013) and Bülow et al. (2014). In Halle and Varvaressos (2014), a CEP notation is formalized such that it could be used as a basis for event abstraction. The application of CEP to business processes and process mining is elaborated more extensively in Soffer et al. (2019). The authors explicitly list event abstraction as a use case and issue to be worked on.

5.1.3 Activity recognition (ARC)

Many *activity recognition methods*, i.e., focusing on translating low-level sensor data from ambient/wearable sensors (e.g., vision-based or motion-based signals) to human activities, have been proposed (Mannhardt et al. 2018a; Abdallah et al. 2018). Often, predictive or intervention-related task scheduling, i.e., based on the recognised activity, is the main goal. A recent overview focused on using these techniques for process mining in industrial environments can be found in Mannhardt et al. (2018a). Broader surveys of the field are published in Aggarwal and Ryoo (2011); Abdallah et al. (2018).

5.1.4 Data pre-processing

Recently, some authors have considered data pre-processing algorithms, specifically designed for process mining (Conforti et al. 2017; Sani et al. 2017, 2018; Chapela-Campa et al. 2019; Sun et al. 2019). These techniques do not aim to alter the granularity level of the data. Rather, the techniques aim to find (in)frequent patterns in the event data, which they decide to either keep, remove, replace, etc. The main challenge in these techniques is the existence of concurrency in the process under study. Concurrent scheduling of activities yields a large number of different orderings in which the corresponding events can be observed. As such, it is difficult to find common patterns. Nonetheless, the application of the aforementioned pre-processing techniques typically leads in more precise process models, i.e., more restrictive process models. The pre-processing techniques are expected to perform bad when the event data are very fine grained, i.e., combined with concurrency among these fine-grained events. Hence, a combination of data pre-processing techniques with event abstraction techniques may lead to better results.

5.2 Future directions

Based on the conducted literature review, some dimensions identified in the taxonomy, turn out to be underexposed. Secondly, some recent developments in the area of process mining indicate possibilities for work on novel dimensions. In this section, we briefly discuss these aspects.

5.2.1 Supervision

Our analysis of the state-of-the-art has illustrated that the majority of existing techniques is supervised, i.e., requiring some domain knowledge to be fed in. Just a small portion is unsupervised. Clearly, the use of domain knowledge enables us to generate more accurate abstractions, but is not always easy to provide, i.e., multiple employees are responsible for sub-parts of a process, process awareness/knowledge is limited/biased. Hence, there seems to be room for novel work in this category, where, process analysts are facilitated to provide some, possibly limited, domain knowledge, e.g., via visual analytics (de Leoni and Dundar 2020).

5.2.2 Probabilistic output

The majority of the techniques results in a discrete output. Whereas there is some work in the area of probabilistic output models, it is questionable whether such output is desired, i.e., on needs to translate it in order to use it. An advantage of probabilistic approaches is the relative ease of "playing" with different abstractions of the same data. Different abstractions specifically come into play when moving toward *n:m* relations between event and activities on the class and instance level. For example, when using a local (Baier et al. 2014) or even a global (Mannhardt et al. 2018a) search method for solving event abstraction in such *n:m* scenarios there is rarely only one clear optimal solution, yet, most methods only return a single result. Even when methods provide a probabilistic interpretation, the utility of such result is low, i.e., most process mining methods require a deterministic event log as input. An exception to this is Pegoraro and van der Aalst (2019), which assumes that an event log contains quantified uncertainty.

5.2.3 Input data

Interestingly, only one of the reviewed works combines the notion of continuous data sources with the notion of process instances. Clearly, a lot of work exists that considers discretization of continuous data/signals. It remains interesting to study whether such techniques are easily translated to work on data that assume the execution of an underlying process. Furthermore, the use of additional data attributes is relatively underexposed. In some cases, certain data attributes logged during the execution of fine-grained activities can aid in mapping onto coarsely-grained concepts. A difficulty when considering arbitrary rules over data combined with control flow is that the computational complexity of abstraction methods often increases considerably.

5.2.4 Streaming data

Within process mining, the notion of online analysis and event streams have gained some attention over recent years, e.g., consider (Burattin 2019; van Zelst et al. 2018). As CEP techniques are generally defined on the basis of online streams of events, an investigation of the applicability of these techniques in an online process mining setting seems promising. Particularly, challenging in online event abstraction is the fact that the knowledge we have of a process instances, i.e., the activities that have been executed, changes over the course of the analysis. As such, the mappings detected by an online event abstraction technique potentially change over time.

5.3 Drawbacks and limitations of event abstraction

The techniques covered in this paper show a wide variety in terms of supervision strategy (including the type of supervision artefact when using a supervised approach). Similarly, the techniques differ greatly on support for interleaving among the fine-granular events. For other categories, we observe less differences, yet, variety still exists. This variety can be seen as a limitation of event abstraction, i.e., it signifies the different perspectives one can take in order to effectively abstract the event data. Therefore, it is likely that we will not observe the development of highly generic, domain independent event abstraction techniques. Rather, it is to be expected that the majority of novel event abstraction techniques are tailored toward specific application domains and/or assumptions posed on the input data.

In case of using an unsupervised approach, there is no guarantee that the patterns discovered rightly correspond to high-level activity instances. On the one hand, some supervised techniques cope with this problem, i.e., by using domain knowledge, the semantic meaning of certain subsequences of fine-granular events is more clear. However, on the other hand, it is questionable whether the type of supervision artifact is available in practice. Specifically, when applying the event abstraction techniques in the context of discovery, a process model accurately describing the process is typically unknown.

6 Conclusion

The widespread application of process mining in industry has unraveled new challenging problems, one of which is mixed granular event collection. Mixed-granular event data hampers the applicability of process mining, i.e., discovered models tend to be too complex and conceal the true business-level process logic. Furthermore, conformance checking techniques are not applicable, in case, there is a mismatch in granularity between the given reference model and the collected event data. In this paper, we present the results of a systematic literature review, covering work that abstracts events into a higher level of granularity. On the basis of the relevant literature, we proposed a taxonomy along the lines of which we classify and discuss the related works in this domain. On the basis of our literature review, we observe ample opportunities for work that does not use any form of supervision in the event abstraction. Similarly, techniques that work on continuous data are of interest. Finally, we observe that the recent interest in online process mining is an interesting future application domain for event abstraction techniques as well.

Acknowledgements Open Access funding provided by Projekt DEAL.

References

Abdallah ZS, Gaber MM, Srinivasan B, Krishnaswamy S (2018) Activity recognition with evolving data streams. ACM Comput Surv 51(4):1–36

Aggarwal J, Ryoo M (2011) Human activity analysis. ACM Comput Surv 43(3):1–43

Alharbi A, Bulpitt A, Johnson OA (2018) Towards unsupervised detection of process models in healthcare. Stud Health Technol Inform 247:381–385

Augusto A, Conforti R, Dumas M, Rosa ML, Maggi FM, Marrella A, Mecella M, Soo A (2019) Automated discovery of process models from event logs: review and benchmark. IEEE Trans Knowl Data Eng 31(4):686–705

Baier T, Mendling J, Weske M (2014) Bridging abstraction layers in process mining. Inf Syst 46:123–139

Baier T, Di Ciccio C, Mendling J, Weske M (2018) Matching events and activities by integrating behavioral aspects and label analysis. Softw Syst Model 17(2):573–598

Baier T, Rogge-Solti A, Mendling J, Weske M (2015) Matching of events and activities: an approach based on behavioral constraint satisfaction. In: Proceedings of the 30th annual ACM symposium on applied computing, Salamanca, Spain, 13–17 April 2015, pp 1225–1230

Bargiela A, Pedrycz W (2016) Granular computing. In: Handbook on computational intelligence: volume 1: fuzzy logic, systems, artificial neural networks, and learning systems. World Scientific, pp 43–66

Begicheva AA, Lomazov IA (2017) Discovering high-level process models from event logs. Model Anal Inf Syst 24(2):125–140

Bernard G, Andritsos P (2018) CJM-ab: abstracting customer journey maps using process mining. In: Information systems in the big data era—CAiSE Forum 2018, Tallinn, Estonia, 11–15 June 2018. Proceedings, pp 49–56

Bose RPJC, van der Aalst WMP (2009) Abstractions in process mining: a taxonomy of patterns. In: Business process management, 7th international conference, BPM 2009, Ulm, Germany, 8–10 Sept 2009. Proceedings, pp 159–175

Brzychczy E, Trzcionkowska A (2019) Process-oriented approach for analysis of sensor data from longwall monitoring system

Bülow S, Backmann M, Herzberg N, Hille T, Meyer A, Ulm B, Wong TY, Weske M (2014) Monitoring of business processes with complex event processing. In: Business process management workshops. Springer International Publishing, pp 277–290

Burattin A (2019) Streaming process discovery and conformance checking. In: Encyclopedia of big data technologies

Carmona J, van Dongen BF, Solti A, Weidlich M (2018) Conformance checking—relating processes and models. Springer, Berlin

Chapela-Campa D, Mucientes M, Lama M (2019) Simplification of complex process models by abstracting infrequent behaviour. In: Service-oriented computing—17th international conference, ICSOC 2019, Toulouse, France, 28–31 Oct 2019. Proceedings, pp 415–430

Chiang HS, Chen MY, Wu ZW (2018) Applying fuzzy petri nets for evaluating the impact of bedtime behaviors on sleep quality. Granul Comput 3(4):321–332

Chinosi M, Trombetta A (2012) BPMN: an introduction to the standard. Comput Stand Interfaces 34(1):124–134

Conforti R, Rosa ML, ter Hofstede AHM (2017) Filtering out infrequent behavior from business process event logs. IEEE Trans Knowl Data Eng 29(2):300–314

Cugola G, Margara A (2012) Processing flows of information: from data stream to complex event processing. ACM Comput Surv 44(3):15

de Leoni M, Dundar S (2020) Event-log abstraction using batch session identification and clustering. In: Proceedings of the 35th ACM/SIGAPP symposium on applied computing (SAC 2020)

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

Fazzinga B, Flesca S, Furfaro F, Masciari E, Pontieri L (2015) A probabilistic unified framework for event abstraction and process detection from log data. In: Proceedings of the 23th OTM confederated international conference on cooperative information systems, vol 9415. Springer, LNCS, pp 320–328

Fazzinga B, Flesca S, Furfaro F, Pontieri L (2018b) Process discovery from low-level event logs. In: Advanced information systems engineering—30th international conference, CAiSE 2018, Tallinn, Estonia, 11–15 June 2018, Proceedings, Lecture Notes in Computer Science, vol 10816. Springer, pp 257–273

Ferreira DR, Szimanski F, Ralha CG (2013a) Mining the low-level behaviour of agents in high-level business processes. Int J Bus Process Integr Management 8 6(2):146–166

Ferreira DR, Szimanski F, Ralha CG (2013b) Mining the low-level behaviour of agents in high-level business processes. IJBPIM 6(2):146–166

Folino F, Guarascio M, Pontieri L (2014) Mining predictive process models out of low-level multidimensional logs. In: Advanced information systems engineering—26th international conference, CAiSE 2014, Thessaloniki, Greece, 16–20 June 2014. Proceedings, pp 533–547

Folino F, Guarascio M, Pontieri L (2015) Mining multi-variant process models from low-level logs. In: Business information systems. Springer International Publishing, pp 165–177

Forney GD (1973) The viterbi algorithm. Proc IEEE 61(3):268–278

Günther CW, Rozinat A, van der Aalst WMP (2009) Activity mining by global trace segmentation. In: Business process management workshops, BPM 2009 international workshops, Ulm, Germany, 7 Sept 2009. Revised Papers, pp 128–139

Halle S, Varvaressos S (2014) A formalization of complex event stream processing. In: 2014 IEEE 18th international enterprise distributed object computing conference. IEEE

Koschmider A, Mannhardt F, Heuser T (2018) On the contextualization of event-activity mappings. In: Business process management workshops, LNBIP, vol 342. Springer, pp 445–457

Leonardi G, Striani M, Quaglini S, Cavallini A, Montani S (2017) Towards semantic process mining through knowledge-based trace abstraction. In: Data-driven process discovery and analysis—7th IFIP WG 2.6 international symposium, SIMPDA 2017, Neuchatel, Switzerland, 6–8 Dec 2017. Revised Selected Papers, pp 45–64

Liu H, Zhang L (2018) Fuzzy rule-based systems for recognition-intensive classification in granular computing context. Granul Comput 3(4):355–365

Liu H, Cocea M, Ding W (2018) Multi-task learning for intelligent data processing in granular computing context. Granul Comput 3(3):257–273

Looney CG (1988) Fuzzy petri nets for rule-based decisionmaking. IEEE Trans Syst Man Cybern 18(1):178–183

Mannhardt F, de Leoni M, Reijers HA, van der Aalst WMP, Toussaint PJ (2018b) Guided process discovery—a pattern-based approach. Inf Syst 76:1–18

Mannhardt F, Bovo R, Oliveira MF, Julier S (2018a) A taxonomy for combining activity recognition and process discovery in industrial environments. In: Intelligent data engineering and automated learning—IDEAL 2018, LNCS, vol 11315. Springer, pp 84–93

Mannhardt F, Tax N (2017) Unsupervised event abstraction using pattern abstraction and local process models. In: Joint proceedings of (EMISA) co-located with the 29th international conference on advanced information systems engineering 2017 (CAiSE 2017), Essen, Germany, 12–13 June 2017, pp 55–63

Montahari-Nezhad H, Saint-Paul R, Casati F, Benatallah B (2011) Event correlation for process discovery from web service interaction logs. VLBD J 20(3):417–444

Murata T (1989) Petri nets: properties, analysis and applications. Proc IEEE 77(4):541–580

Nguyen H, Dumas M, ter Hofstede AHM, Rosa ML, Maggi FM (2019) Stage-based discovery of business process models from event logs. Inf Syst 84:214–237

Nickerson RC, Muntermann J, Varshney U (2010) Taxonomy development in information systems: a literature survey and problem statement. In: Sustainable IT collaboration around the globe. 16th Americas conference on information systems, AMCIS 2010, Lima, Peru, 12–15 Aug 2010, p 125

Oliveira CAL, Silva NC, Sabat CL, Lima RMF (2013) Reducing the gap between business and information systems through complex event processing. Comput Inform 32(2):225–250

Pedrycz W (2001) Granular computing: an introduction. In: Proceedings joint 9th IFSA world congress and 20th NAFIPS international conference (Cat. No. 01TH8569), vol 3, pp 1349–1354

Pegoraro M, van der Aalst WMP (2019) Mining uncertain event data in process mining. In: ICPM. IEEE, pp 89–96

Pérez-Castillo R, Weber B, Guzmán I, Piattini M, Pinggera J (2012) Assessing event correlation in non-process-aware information systems. Softw Syst Model 13:1–23

Rehse JR, Fettke P (2019) Clustering business process activities for identifying reference model components. In: Business process management workshops—BPM 2018 international workshops, Sydney, NSW, Australia, 9–14 Sept 2018. Springer International Publishing, Revised Papers, pp 5–17

Richetti PHP, Baião FA, Santoro FM (2014) Declarative process mining: reducing discovered models complexity by pre-processing event logs. In: Business process management—12th international conference, BPM 2014, Haifa, Israel, 7–11 Sept 2014. Proceedings, pp 400–407

Sánchez-Charles D, Carmona J, Muntés-Mulero V, Solé M (2017) Reducing event variability in logs by clustering of word embeddings. In: Business process management workshops—BPM 2017 international workshops, Barcelona, Spain, 10–11 Sept 2017, Revised Papers, pp 191–203

Sani MF, van Zelst SJ, van der Aalst WMP (2017) Improving process discovery results by filtering outliers using conditional behavioural probabilities. In: Business process management workshops—BPM 2017 international workshops, Barcelona, Spain, 10–11 Sept 2017, Revised Papers, pp 216–229

Sani MF, van Zelst SJ, van der Aalst WMP (2018) Repairing outlier behaviour in event logs using contextual behaviour. Enterp Model Inf Syst Archit 14:5:1–5:24

Senderovich A, Rogge-Solti A, Gal A, Mendling J, Mandelbaum A (2016) The ROAD from sensor data to process instances via interaction mining. In: Advanced information systems engineering—28th international conference, CAiSE 2016, Ljubljana, Slovenia, 13–17 June 2016. Proceedings, pp 257–273

Soffer P, Hinze A, Koschmider A, Ziekow H, Ciccio CD, Koldehofe B, Kopp O, Jacobsen H, Sürmeli J, Song W (2019) From event streams to process models and back: challenges and opportunities. Inf Syst 81:181–200

Sun X, Hou W, Yu D, Wang J, Pan J (2019) Filtering out noise logs for process modelling based on event dependency. In: 2019 IEEE international conference on web services, ICWS 2019, Milan, Italy, 8–13 July 2019, pp 388–392

Tax N, Sidorova N, Haakma R, van der Aalst WMP (2018) Mining process model descriptions of daily life through event abstraction. In: Bi Y, Kapoor S, Bhatia R (eds) Intelligent systems and applications. Springer, Cham, pp 83–104

Tax N, Sidorova N, Haakma R, van der Aalst WMP (2016) Event abstraction for process mining using supervised learning techniques. In: Proceedings of SAI intelligent systems conference (IntelliSys) 2016. Springer, pp 251–269

Tello G, Gianini G, Mizouni R, Damiani E (2019) Machine learning-based framework for log-lifting in business process mining applications. In: business process management—17th international conference, BPM 2019, Vienna, Austria, 1–6 Sept 2019, Proceedings, pp 232–249

van der Aalst WMP (2016) Process mining—data science in action, 2nd edn. Springer, Berlin

van Zelst SJ, van Dongen BF, van der Aalst WMP (2018) Event stream-based process discovery using abstract representations. Knowl Inf Syst 54(2):407–435

van Eck ML, Sidorova N, van der Aalst WMP (2016) Enabling process mining on sensor data from smart products. In: Tenth IEEE international conference on research challenges in information science, RCIS 2016, Grenoble, France, 1–3 June 2016, pp 1–12