# Navigating Event Abstraction in Process Mining: A Comprehensive Analysis of Sub-problems, Data, and Process Characteristic Considerations

Jungeun Lim and Minseok Song(✉)

Department of Industrial and Management Engineering, Pohang University of Science and Technology, Pohang, South Korea
{je5719,mssong}@postech.ac.kr

**Abstract.** Process mining technologies assume event logs with an appropriate level of granularity, but many systems generate low-level event logs, resulting in complex process models. Event abstraction addresses this issue by transforming low-level event logs into abstracted event logs, enabling the derivation of business-level process models. However, practitioners often struggle to choose suitable event abstraction methods. This is primarily due to the lack of comparative studies that analyze the differences between methods and the insufficient information regarding the data and relevant process characteristics to be considered. This study conducts a comprehensive literature review on event abstraction to overcome these challenges. The review focuses on summarizing specific sub-problems in event abstraction, identifying types of data that can be utilized, and highlighting important process characteristics that should be considered. The insights and guidance provided by this review will be valuable to practitioners seeking to select and implement effective event abstraction techniques.

**Keywords:** Event Abstraction · Hierarchical Process Model · Literature Review

## 1 Introduction

The field of process mining deals with analyzing actual business processes by utilizing event logs [1]. Typically, event logs capture system-level activities rather than easily understandable business-level activities. This poses challenges for process discovery and modeling, as the detailed low-level activities may lead to complex process models that fail to capture the higher-level structure of the process. Event abstraction serves as a solution to address these challenges by identifying and abstracting low-level activities associated with the same high-level activity [2, 3].

While various event abstraction methods have been proposed in research studies, practitioners face challenges in selecting appropriate strategies for their specific situations and understanding the characteristics of their data for event abstraction. Additionally, there is no list of process characteristics to consider in event abstraction and comparative studies to guide the selection of abstraction methods based on process characteristics.

In recent years, several relevant literature reviews have been conducted on event abstraction. Diba et al. [4] classified 20 event abstraction studies into clustering, supervised learning, behavioral patterns, and process model-based approach, focusing on technology. The data used in each study and assumptions for the application of technology were also explained, but they were not systematically organized. Marin-Castro & Tello-Leal [5] briefly introduced nine studies, categorizing them as unsupervised and supervised approaches. Zelst et al. [6] conducted an in-depth literature review on event abstraction, presenting seven taxonomies based on existing studies and providing valuable insights for understanding event abstraction from multiple perspectives.

However, existing literature reviews have not fully addressed the aforementioned challenges. Therefore, this study aims to present a conceptual framework that facilitates the understanding and comparison of event abstraction studies for practitioners. This paper is organized as follows. Section 2 defines the sub-problems of event abstraction, and Sect. 3 summarizes the data used in event abstraction research. Section 4 introduces methods for solving each sub-problem, and Sect. 5 defines process characteristics and conducts an analysis to show the importance of considering process characteristics in event abstraction. Finally, we conclude this paper in Sect. 6.

## 2 Event Abstraction Process

Typically, event abstraction involves several sub-problems that vary depending on specific conditions. This section presents an overview of the event abstraction process described in the literature, along with the individual problems that constitute this process. (For detailed information on the methods proposed to solve each sub-problem, please refer to Sect. 5.) The event abstraction process was designed by reviewing the existing literature that was covered in three review papers related to event abstraction published within five years. In total, 39 papers were considered, and after the abstract screening, 36 papers were selected.

The event abstraction process is summarized and organized like the BPMN model in Fig. 1. If a high-level process model, which represents the flow of activities, is available, solving the process model-based event abstraction problem is sufficient to obtain an abstracted event log. However, additional steps are required in cases where a high-level process model is not accessible. The first step, preprocessing, is optional and applied when the event cycle is too short. The subsequent step is to map an event class to an activity class (EC-AC) or an event instance to an activity class (EI-AC). Solving the EI-AC problem is particularly useful when different activity classes can be associated with the same event classes. Finally, the problem of mapping event instances to activity instances (EI-AI) is addressed to obtain an abstracted event log.

**Process Model-Based Event Abstraction.** This approach is employed when a high-level process model is available. In this method, studies [3, 7–9] focus on determining the activity class and activity instance of each event instance by considering the control-flow of activities within the high-level process model. If you performed process model-based event abstraction, there is no need to go through additional steps.

**Preprocessing.** The preprocessing step aims to distinguish event instances. In the literature [10, 11], this step was necessary because data collected from sensors or the web

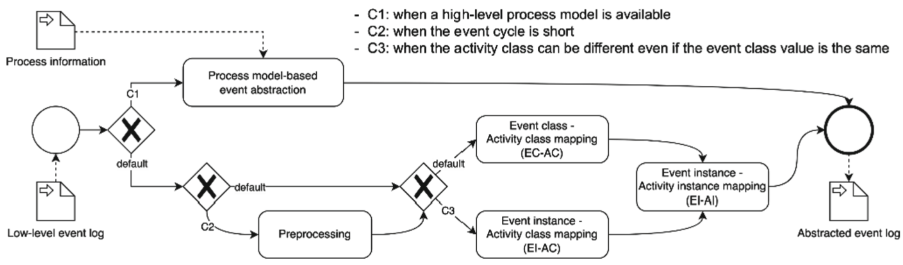
- C1: when a high-level process model is available
- C2: when the event cycle is short
- C3: when the activity class can be different even if the event class value is the same
Process information
Process model-based event abstraction
C1
default
default
C2
Preprocessing
default
C3
Event class - Activity class mapping (EC-AC)
Event instance - Activity instance mapping (EI-AI)
Event instance - Activity class mapping (EI-AC)
Low-level event log
Abstracted event log

**Fig. 1.** Event abstraction process with a BPMN-like notation.

often have very short intervals, unlike typical event logs. In such cases, multiple data points are treated together as a single event instance to enable subsequent analysis.

**Event Class – Activity Class Mapping.** The EC-AC problem [12–21] aims to find the mapping rule $\varphi : ec \rightarrow ac$ to identify event classes belonging to the same activity class. An event class *ec* is an attribute representing the event type, such as the event name, when an event is a low-level task. An activity class *ac* is an attribute representing the type of activity, when an activity is a high-level task, consisting of a low-level subprocess. The output of this problem in the literature shows either an n:1 or an n:m relationship between event classes and activity classes. In Fig. 2 (a), a conceptual example is provided: regardless of the attribute values, the activity class is determined solely based on the event class.

**Event Instance – Activity Class Mapping.** The EI-AC problem [3, 14, 19, 22–31] focuses on finding a mapping rule $\rho : ei \rightarrow ac$. An event instance *ei* is the occurrence of an event and it can be represented by a tuple of attribute values. The attributes representing the event instance do not include the activity class and activity instance identifier. This problem aims to identify patterns of event attributes that belong to the same activity class. The literature demonstrates the output of this problem as an n:1 relationship between attribute tuples and activity classes. In Fig. 2 (b), a conceptual example is provided: the activity class is determined by considering the event class and other attribute values.

**Event Instance – Activity Instance Mapping.** The EI-AI problem [3, 7–12, 14–28, 30–33] deals with finding a mapping rule $\psi : ei \rightarrow ai$. An activity instance *ai* is a trace of event instances, and attributes representing the event instance include the activity class. Even if multiple event instances share the same activity class values, they may either belong to a single activity instance or multiple activity instances created by performing the same activity multiple times. EI-AI addresses this situation. The literature presents the output of this problem as a 1:1 or an n:1 relationship between attribute tuples and activity instances. In Fig. 2 (c), a conceptual example is provided: the activity instance is determined by considering the event class and other attribute values, including the activity class, which can be derived from either EC-AC or EI-AC mapping results.

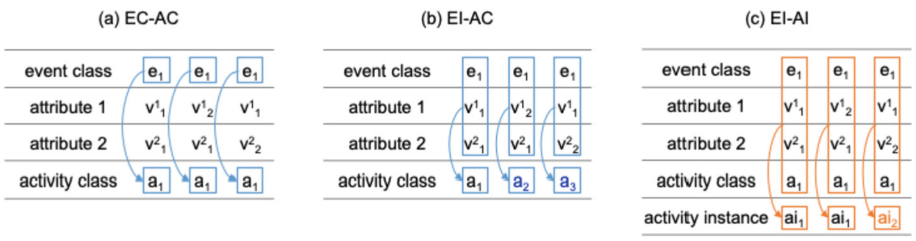

**Fig. 2.** Event log examples describing EC-AC, EI-AC, and EI-AI problems.

## 3 Data for Event Abstraction

The data used for event abstraction can be classified into three categories: event class sequence, event attributes, and activity-related data.

### 3.1 Event Class Sequence

The event class sequence represents a simple event log that considers only the class of events and their chronological order based on timestamps. This data is typically used to analyze the relationship between events, such as directly followed relations and causal relations. Alternatively, the data is transformed into position values, which indicate the relative or absolute position of an event within traces.

Several methods exist for calculating the position value of an event [12, 13, 23]. In the works of Günther et al. [12] and Rehse & Fettke [13], the position value is determined by measuring the distance between two events in the event sequence for all event class pairs. Folino et al. [23] expressed the position of an event as a vector with three values: 1) the distance of the event from the first event in the trace, 2) the length of the trace to which the event belongs compared to the longest trace in the event log, and 3) the distance of the event from the last event in the trace relative to the length of the longest trace.

### 3.2 Event Attributes

There have been studies using various event attributes. We collected all the attributes from the literature and classified them into four categories: event class attributes, event instance attributes, case attributes, and global attributes (Table 1).

Event class attributes are specific to the event class itself. Examples of event class attributes include the event name and event description [14–16]. Event instance attributes can vary across different event instances, even if they share the same event class. These attributes include resource, time, status, group, and others [3, 10, 11, 14, 15, 22, 23, 26, 27, 32]. Case attributes relate to the specific case or context in which the event occurs [14, 22, 27]. They provide information about the case at the time the event is performed, such as previous tasks performed and the condition of the case. Lastly, global attributes represent external factors that can dynamically change [14]. These attributes capture information that may not be directly associated with the event or case, but can still influence the event, such as case priority or temperature.

**Table 1.** Four categories of event attributes and their examples.

| Event class attributes | event name, event description |
|---|---|
| Event instance attributes | group, role, resource, time, location, status, support team, functional division, organization line, organization country, activity class, sub-object |
| Case attributes | preceding events, succeeding events, case's clinical conditions |
| Global attributes | case priority |

### 3.3 Activity-Related Data

Unlike event sequences and event attributes, activity-related data is often challenging to obtain. This data is typically held by organizations to support the business processes, or it may need to be created based on expert experience. Examples of activity-related data include the high-level process model, low-level subprocess model, activity set, and activity description [7, 8, 14, 15, 26, 27, 32, 34]. As mentioned earlier, when a high-level process model is available, event abstraction can be achieved through process model-based event abstraction. Other activity-related data serves as additional information for solving the EC-AC, EI-AC, and EI-AI problems.

## 4 Methods for Solving Event Abstraction Sub-problems

In this section, we introduce several studies with data and methods for each sub-problem. We categorize the methods into three classes: heuristic, unsupervised, and supervised approaches. The heuristic approach comprises methods that define abstraction rules based on domain knowledge. These methods rely on expert insights and predefined rules to guide the event abstraction process. The unsupervised approach encompasses methods where abstraction rules are derived from learning the event log without utilizing activity-related data. Lastly, the supervised approach includes methods that are similar to the unsupervised approach, but with the additional utilization of activity-related data.

### 4.1 Process Model-Based Event Abstraction Methods

The methods to solve process model-based event abstraction problems were primarily categorized as a supervised approach since a process model is a prerequisite for this problem. Mannhardt et al. [3] and Mannhardt & Tax [7] aimed to determine the best activity classes and activity instance identifiers that can effectively explain a given event sequence. These studies utilized both the high-level process model and the low-level subprocess model. Trace alignment techniques were employed to align the event sequence with potential event sequences derived from the models. On the other hand, Ferreira [34] solely relied on the high-level process model. They utilized the EM-algorithm to estimate the optimal alignment between a given event sequence and the high-level process model.

### 4.2 Preprocessing Methods

There were two studies that performed preprocessing: one study utilized data obtained from sensors [11], and the other study utilized data obtained from the web [10]. Both studies used the naive solution for preprocessing. Eck et al. [11], the user set the duration of each event instance, taking into consideration the sensor's data collection interval. Data points were then classified into event instances based on this duration. Leoni & Dündar [10] set a threshold for the time interval between data points, and data points were divided into different event instances when the time interval exceeded the threshold.

### 4.3 Event Class – Activity Class Mapping Methods

**Unsupervised Approach.** Clustering techniques were commonly used in the unsupervised approach to group event classes with similar properties into the same activity class [12, 13, 16]. Various combinations of data, including event position value, event class attributes, and event instance attributes were utilized to represent the properties of each event class. Another technique used in the unsupervised approach was inductive mining [17]. This algorithm was applied to the event class sequence, resulting in the identification of control-flow relationships between event classes in the form of a process tree. Events belonging to the same activity were then defined by grouping the sub-tree within the process tree.

**Supervised Approach.** Biar et al. [14, 15] employed a technique that is applicable when activity-related data is available. They utilized the name and descriptions of events and activities to calculate their similarity. By identifying pairs with high similarity, they established mappings between events and activities. Additionally, Biar et al. [15] utilized declarative mining, which involved deriving declarative rules from low-level event logs and simulated event logs generated using a high-level process model. These rules helped filter out EC-AC combinations that were considered impossible.

### 4.4 Event Instance – Activity Class Mapping Methods

**Heuristic Approach.** In studies employing the heuristic approach, domain experts defined criteria that event attributes must satisfy in order to map an event instance to a specific activity class [8, 14, 22]. For example, the resource attribute can be used as a mapping condition where event instances with the same resource value are mapped to the same activity class, regardless of other attribute values such as event class.

**Unsupervised Approach.** Fazzinga et al. [25] and Alharbi et al. [24] utilized Hidden Markov Models (HMM). They considered the activity class as a hidden state and, using the sequence of event classes as input data, learned the transition probability between activities and the emission probability of events associated with each activity. Based on the learned transition and emission probability, the activity class values that maximize the probability of the event class sequence were assigned to each event instance.

Folino et al. [23] employed a predictive clustering tree where event instance attributes served as predictors (x) and the relative positions of the events were the target variable (y). This study falls under the unsupervised approach since the relative position value used as the target variable is not the ground truth.

**Supervised Approach.** Tax et al. [26] addressed the problem by learning a conditional random field (CRF) model using event attributes as predictors (x) and activity classes as the target variable (y). Senderovich et al. [27] utilized the integer linear programming (ILP) by incorporating domain knowledge, such as the activity class set, characteristics of event instances that cannot be mapped to a single activity class, and prior relationships between activities, as constraints. Tello et al. [32] utilized a list of candidate low-level subsequences of activities as additional information. This list was used to calculate the probability of a direct-following relation between event activities. The calculated probability values were then used to divide the event sequence into a set of subsequences, selecting the result with the highest probability. The resulting subsequences were represented using event attributes, and finally, the subsequences were mapped to activity classes using the multivariate time series clustering method.

### 4.5 Event Instance – Activity Instance Mapping Methods

Naive approaches were most used for solving the EI-AI. In some studies, continuous event instances with the same activity class value were treated as a single activity instance [11, 12, 17, 24, 26]. Otherwise, studies considered each event instance as a separate activity instance [16, 23, 25, 27].

**Heuristic Approach.** Heuristic approaches were also widely adopted, where the user directly set the conditions that event instances must satisfy to be assigned to an activity instance. Studies introduced several parameters that can be used to set the condition, including the maximum number of event instances that can be assigned to a single activity instance [14, 15]; the maximum duration of an activity instance [10, 11, 14, 15]; the maximum number of event instances with different activity instance IDs occurring between event instances with the same activity instance ID (i.e., the maximum number of interleaved activity instances) [22]; and the maximum duration of the interleaved activity instances [22].

## 5 Process Property Consideration

Studies tried to perform event abstraction in consideration of characteristics such as interleaving and repeated use of activity in the process. However, there is no systematic summary of the characteristics of the process to be considered, making it difficult to compare studies. In this section, we organize the characteristics of the process and evaluate some methods with some process characteristics to present the need to consider process characteristics.

### 5.1 Process Properties

The first process property is the structure of a process. The high-level process and low-level subprocess are composed of a combination of sequential, exclusive, parallel, and redo structures (Fig. 3). Depending on the combination of these structures, the event

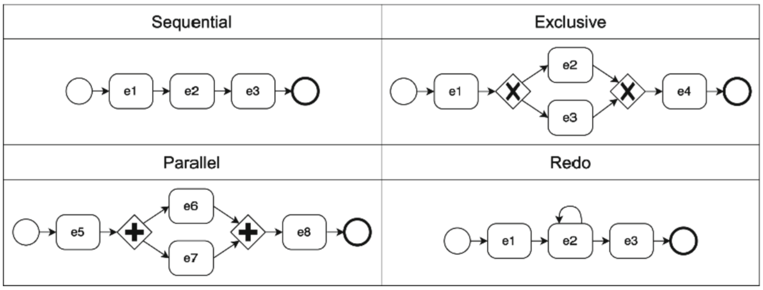

**Fig. 3.** Basic process structures of processes.

abstraction result may be affected. For example, if the high-level process includes a parallel structure, events required to perform parallel activities can be mixed in order, making it challenging to derive abstraction rules from the event log.

Another important process characteristic is the relationship between low-level and high-level activities (EC:AC). A low-level activity can be associated with only one high-level activity (n:1) or multiple high-level activities (n:m). This relationship between activities can significantly impact the performance of event abstraction, even when the process structure remains the same.

## 5.2 Analysis Method

We evaluated methods proposed in previous studies [12, 13, 23] that address the EC-AC problem through clustering based on event position values. To generate the data for evaluation, we created virtual process models with all possible process structure combinations where EC:AC = n:1. In total, we generated 16 virtual process models (SS, SE, SP, SR, ES, EE, EP, ER, PS, PE, PP, PR, RS, RE, RP, and RR: the first character represents the high-level structure with its first alphabet, and the second character represents the low-level structure with its first alphabet). We unified the structures of low-level subprocess models for simplicity. For example, the virtual process model PR is like Fig. 4.

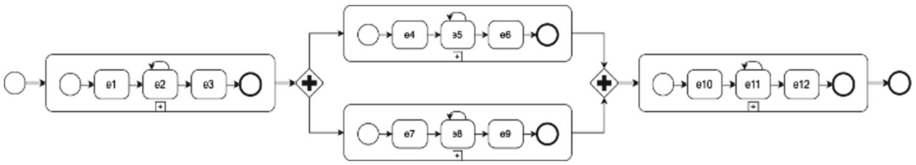

**Fig. 4.** Virtual process model consisting of a parallel structure at the high-level and a redo structure at the low-level.

We generated 1000 traces for each virtual process model by simulation. The position values were calculated following the formula provided in the literature, and we applied agglomerative clustering. We evaluated the method as effective if events belonging to the same activity were successfully grouped into one cluster. We conducted a qualitative analysis using the agglomerative clustering dendrogram and did not evaluate the result precisely based on strict thresholds. It is important to note that we only assessed the activity corresponding to the selected structure. For example, in the case of the model shown in Fig. 4, we evaluated the clustering result of (e4, e5, e6) and (e7, e8, e9), while (e1, e2, e3) and (e10, e11, e12) were not assessed.

### 5.3 Analysis Result

Across all three methods, the combinations of SS, SE, and any combinations involving high-level parallel structures (i.e., PS, PE, PP, and PR) did not yield satisfactory clustering results. In contrast, when both the high-level and low-level structures were redo structures, all three methods showed promising outcomes. Table 2 presents the results for the remaining combinations. Among these combinations, [12] and [13] consistently yielded similar results, except for the SP. The method of [23] did not work well in many combinations compared to the other two studies but performed well in the case of SR, where the other two studies struggled. Although this analysis is based on a limited dataset from a simplified process model, it shows the importance of considering process characteristics in event abstraction.

**Table 2.** Mapping results for each process structure combination.

| | SP | SR | ES | EE | EP | ER | RS | RE | RP | RR |
|---|---|---|---|---|---|---|---|---|---|---|
| [12] | o | x | o | x | o | o | o | o | o | o |
| [13] | x | x | o | o | o | o | o | o | o | o |
| [23] | o | o | x | x | x | x | x | x | x | o |

## 6 Discussion and Conclusion

Event abstraction plays a crucial role in converting low-level event logs, which hinders effective business process analysis. Therefore, it is a vital problem that must be addressed to achieve business process intelligence. In this study, we provide a comprehensive summary of the event abstraction process, sub-problems, data, and process characteristic considerations to facilitate the understanding and practical application of event abstraction studies. The insights presented in this study can be leveraged and further developed in several ways.

First, combining diverse methods to solve sub-problems allows us to explore possibilities for obtaining better event abstraction outcomes. Each method handles different process characteristics, so combining them holds the potential for improving event abstraction results. Furthermore, there is an opportunity to automate event abstraction. Optimal results can be derived by automatically selecting the most suitable event abstraction process and method for a given dataset. The evaluation method for event abstraction plays a crucial role in achieving automated outcomes. Although this aspect is not covered in this study, it is necessary for further investigation to enable automated event abstraction.

Second, comparing data and techniques across different approaches can lead to new research directions. For example, the heuristic approach, which defines rules based on diverse data, can inspire investigations into unsupervised or supervised approaches.

Third, regarding the process characteristics, this study conducted only small-scale experiments that show the necessity of considering these factors. To facilitate the comparison and evaluation of event abstraction methods, generating datasets with the suggested process characteristic outlined in this study, varying in size and conditions would be beneficial.

Finally, the recent review included in this study is from 2021, as the literature was selected from existing review papers. We need to analyze more recent literature using the framework introduced in this research.

**Acknowledgment.** This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2023-2018-0-01441) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation) and by a grant of the Korea Health Technology R&D Project through the Korea Health Industry Development Institute (KHIDI), funded by the Ministry of Health & Welfare, Republic of Korea (grant number: HI23C0061).

## References

1. Van der Aalst, W.: Process mining: data science in action (2016)
2. Li, C.Y., Van Zelst, S.J., Van Der Aalst, W.M.P.: An activity instance based hierarchical framework for event abstraction. In: La Rosa, M., Loos, P., Pastor, O. (eds.) Business Process Management. BPM 2016. LNCS, vol. 9850, pp. 160–167. Springer, Cham (2021). https://doi.org/10.1007/978-3-319-45348-4_8
3. Mannhardt, F., de Leoni, M., Reijers, H.A., van der Aalst, W.M.P., Toussaint, P.J.: From low-level events to activities - a pattern-based approach. In: La Rosa, M., Loos, P., Pastor, O. (eds.) Business Process Management. BPM 2016. LNCS, vol. 9850, pp. 125–141. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45348-4_8
4. Diba, K., Batoulis, K., Weidlich, M., Weske, M.: Extraction, correlation, and abstraction of event data for process mining. Wiley Interdiscip. Rev. Data Min. Knowl. Discov. **10**, 1–24 (2020)
5. Marin-Castro, H.M., Tello-Leal, E.: Event log preprocessing for process mining: a review. Appl. Sci. (Switzerland). **11**, 1–29 (2021)
6. van Zelst, S.J., Mannhardt, F., de Leoni, M., Koschmider, A.: Event abstraction in process mining: literature review and taxonomy. Granul. Comput. **6**, 719–736 (2021)
7. Mannhardt, F., Tax, N.: Unsupervised event abstraction using pattern abstraction and local process models. arXiv preprint arXiv:1704.03520 (2017)
8. Mannhardt, F., de Leoni, M., Reijers, H.A., van der Aalst, W.M.P., Toussaint, P.J.: Guided process discovery – a pattern-based approach. Inf. Syst. **76**, 1–18 (2018)
9. Ferreira, D.R., Szimanski, F., Ralha, C.G.: Improving process models by mining mappings of low-level events to high-level activities. J. Intell. Inf. Syst. **43**, 379–407 (2014)

10. De Leoni, M., Dündar, S.: Event-log abstraction using batch session identification and clustering. In: Proceedings of the ACM Symposium on Applied Computing, pp. 36–44 (2020)
11. Van Eck, M.L., Sidorova, N., Van Der Aalst, W.M.P.: Enabling process mining on sensor data from smart products. In: Proceedings - International Conference on Research Challenges in Information Science. 2016-Augus, (2016)
12. Günther, C.W., Rozinat, A., Van Der Aalst, W.M.P.: Activity mining by global trace segmentation. In: Daniel, F., Sheng, Q., Motahari, H. (eds.) Business Process Management Workshops. BPM 2018. LNBIP, vol. 342, pp. 128–139. Springer, Cham (2010). https://doi.org/10.1007/978-3-030-11641-5_1
13. Rehse, J.-R., Fettke, P.: Clustering business process activities for identifying reference model components. In: Daniel, F., Sheng, Q., Motahari, H. (eds.) Business Process Management Workshops. BPM 2018. LNBIP, vol. 342, pp. 5–17. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-11641-5_1
14. Baier, T., Mendling, J., Weske, M.: Bridging abstraction layers in process mining. Inf. Syst. **46**, 123–139 (2014)
15. Baier, T., Di Ciccio, C., Mendling, J., Weske, M.: Matching events and activities by integrating behavioral aspects and label analysis. Softw. Syst. Model. **17**, 573–598 (2018)
16. Sánchez-Charles, D., Carmona, J., Muntés-Mulero, V., Solé, M.: Reducing event variability in logs by clustering of word embeddings. In: Teniente, E., Weidlich, M. (eds.) Business Process Management Workshops. BPM 2017. LNBIP, vol. 308. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-74030-0_14
17. Andritsos, P.: CJM-ab : Abstracting Customer Journey, vol. 1, pp. 49–56 (2018)
18. Richetti, P.H.P., Baião, F.A., Santoro, F.M.: Declarative process mining: reducing discovered models complexity by pre-processing event logs. In: Sadiq, S., Soffer, P., Völzer, H. (eds.) Business Process Management. BPM 2014. LNCS, vol. 8659, pp. 400–407. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10172-9_28
19. Brzychczy, E., Trzcionkowska, A.: Process-oriented approach for analysis of sensor data from longwall monitoring system. Adv. Intell. Syst. Comput. **835**, 611–621 (2019)
20. Van Dongen, B.F., Adriansyah, A.: Process mining: fuzzy clustering and performance visualization. In: Rinderle-Ma, S., Sadiq, S., Leymann, F. (eds.) Business Process Management Workshops. BPM 2009. LNBIP, vol. 43, PP. 158–169. Springer, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12186-9_15
21. Begicheva, A.A., Lomazov, I.A.: Discovering high-level process models from event logs. Model. Anal. Inf. Syst. **24**, 125–140 (2017)
22. Leonardi, G., Striani, M., Quaglini, S., Cavallini, A., Montani, S.B.: Towards semantic process mining through knowledge-based trace abstraction. In: Ceravolo, P., van Keulen, M., Stoffel, K. (eds.) Data-Driven Process Discovery and Analysis. SIMPDA 2017. LNBIP, vol. 340, pp. 45–64. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-11638-5_3
23. Folino, F., Guarascio, M., Pontieri, L.: Mining multi-variant process models from low-level logs. In: Abramowicz, W. (eds.) Business Information Systems. BIS 2015. LNBIP, vol. 208, pp. 165–177. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-19027-3_14
24. Alharbi, A., Bulpitt, A., Johnson, O.: Towards unsupervised detection of process models in healthcare. In: Ugon, A., Karlsson, D., Dlein, gunnar O., and Moen, A. (eds.) Building Continents of Knowledge in Oceans of Data: The Future of Co-Create eHealth, pp. 381–385. IOS Press (2018)
25. Fazzinga, B., Flesca, S., Furfaro, F., Pontieri, L.: Process discovery from low-level event logs. In: Krogstie, J., Reijers, H. (eds.) Advanced Information Systems Engineering. CAiSE 2018. LNCS, vol. 10816, PP. 257–273. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91563-0_16
26. Tax, N., Sidorova, N., Haakma, R., van der Aalst, W.: Mining process model descriptions of daily life through event abstraction. Stud. Comput. Intell. **751**, 83–104 (2018)

27. Senderovich, A., Rogge-Solti, A., Gal, A., Mendling, J., Mandelbaum, A.: The ROAD from sensor data to process instances via interaction mining. In: Nurcan, S., Soffer, P., Bajec, M., Eder, J. (eds.) Advanced Information Systems Engineering. CAiSE 2016. LNCS, vol. 9694, pp. 257–273. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39696-5_16
28. Fazzinga, B., Flesca, S., Furfaro, F., Masciari, E., Pontieri, L.: Efficiently interpreting traces of low level events in business process logs. Inf. Syst. **73**, 1–24 (2018)
29. Bose, R.P.J.C., Maggi, F.M., Van Der Aalst, W.M.P.: Enhancing declare maps based on event correlations. In: Daniel, F., Wang, J., Weber, B. (eds.) Business Process Management. LNCS, vol. 8094, pp. 97–112. Springer, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40176-3_9
30. Li, J., Bose, R.P.J.C., Van Der Aalst, W.M.P.: Mining context-dependent and interactive business process maps using execution patterns. In: Zur Muehlen, M., Su, J. (eds.) Business Process Management Workshops. BPM 2010. LNBIP, vol. 66, pp. 109–121. Springer, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20511-8_10
31. Folino, F., Guarascio, M., Pontieri, L.: Mining predictive process models out of low-level multidimensional logs. In: Jarke, M., et al. (eds.) Advanced Information Systems Engineering. CAiSE 2014. LNCS, vol. 8484, pp. 533–547. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07881-6_36
32. Tello, G., Gianini, G., Mizouni, R., Damiani, E.: Machine learning-based framework for log-lifting in business process. In: Hildebrandt, T., van Dongen, B., Röglinger, M., Mendling, J. (eds.) Business Process Management. BPM 2019. LNCS, vol. 11675. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26619-6_16
33. Sun, Y., Bauer, B.: A graph and trace clustering-based approach for abstracting mined business process models. In: ICEIS 2016 - Proceedings of the 18th International Conference on Enterprise Information Systems, vol. 1, pp. 63–74 (2016)
34. Ferreira, D.R., Szimanski, F., Ralha, C.G.: Mining the low-level behaviour of agents in high-level business processes. Int. J. Bus. Process. Integr. Manag. **6**, 146–166 (2013)