Computers in Industry 137 (2022) 103612

Contents lists available at ScienceDirect

Computers in Industry

journal homepage: www.elsevier.com/locate/compind

Utilizing domain knowledge in data-driven process discovery: A literature review

Daniel Schuster[a,b,*], Sebastiaan J. van Zelst[a,b], Wil M.P. van der Aalst[a,b]

[a] Fraunhofer FIT, Process Mining Research Group, Schloss Birlinghoven, Sankt Augustin 53757, North Rhine-Westphalia, Germany
[b] RWTH Aachen University, Chair of Process and Data Science, Ahornstraße 55, Aachen 52074, North Rhine-Westphalia, Germany

A R T I C L E I N F O

Article history:
Received 1 September 2021
Received in revised form 4 December 2021
Accepted 17 January 2022
Available online xxxx

Keywords:
Process mining
Process discovery
Process models
Human-in-the-loop
Hybrid intelligence

A B S T R A C T

Process mining aims to improve operational processes in a data-driven manner. To this end, process mining offers methods and techniques for systematically analyzing event data. These data are generated during the execution of processes and stored in organizations' information systems. Process discovery, a key discipline in process mining, comprises techniques used to (automatically) learn a process model from event data. However, existing algorithms typically provide low-quality models from real-life event data due to data-quality issues and incompletely captured process behavior. Automated filtering of event data is valuable in obtaining better process models. At the same time, it is often too rigorous, i.e., it also removes valuable and correct data. In many cases, prior knowledge about the process under investigation can be additionally used for process discovery besides event data. Therefore, a new family of discovery algorithms has been developed that utilizes domain knowledge about the process in addition to event data. To organize this research, we present a literature review of process discovery approaches exploiting domain knowledge. We define a taxonomy that systematically classifies and compares existing approaches. Finally, we identify remaining challenges for future work.

Contents

* Corresponding author at: Fraunhofer FIT, Process Mining Research Group, Schloss Birlinghoven, Sankt Augustin 53757, North Rhine-Westphalia, Germany.
E-mail address: daniel.schuster@fit.fraunhofer.de (D. Schuster).

https://doi.org/10.1016/j.compind.2022.103612
0166-3615/© 2022 The Author(s). Published by Elsevier B.V.
CC_BY_4.0

1. Introduction

The execution of processes–ranging from business to production processes–is often supported by information systems. These systems record the processes' execution in great detail, e.g., which process activities were executed, their order, their duration, and which resources were involved. These records are referred to as *event data*. To improve an organization's processes, the analysis of event data provides valuable opportunities. *Process mining* (van der Aalst, 2016) offers techniques, tools, and methods for systematically analyzing event data to gain insights into the execution of processes. These insights are used to *optimize the processes*, i.e., the central objective of process mining.

Process discovery (van der Aalst, 2018) is a key discipline in process mining. Starting from event data, a discovery algorithm learns a process model describing the process as captured in the provided event data. Fig. 1 illustrates the general architecture of conventional discovery algorithms. Automated, data-driven process discovery assumes that event data are the most objective representation of a process. However, event data often have quality issues (Martin et al., 2021, Table 4), such as incorrect logged process behavior, incomplete process behavior resulting from ongoing process executions while the event data was extracted, and missing process behavior that actually occurred. Apart from data-quality issues, concurrency in many processes often yields incomplete event data because not every possible scheduling of the concurrent process activities is captured. As such, process discovery from event data can be seen as an *unsupervised learning* task.

Many process discovery algorithms have been developed. Most of them work fully-automated, i.e., they take event data as input, optionally allow to set configuration parameters, and return a process model, cf. Fig. 1. We refer to such algorithms as *conventional* process discovery algorithms. However, the process models discovered by conventional algorithms are often of low quality because of the aforementioned data-quality issues (Suriadi et al., 2017). Moreover, it is difficult for a user to choose a suitable process discovery algorithm for specific event data because each algorithm works differently and, thus, may produce different results. Therefore, in-depth

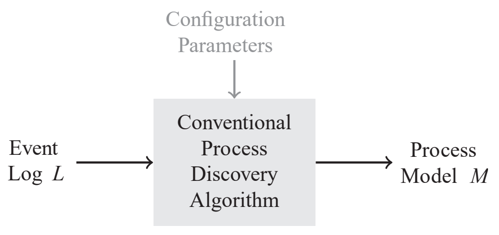


Fig. 1. Schematic visualization of the general architecture of conventional process discovery algorithms. From event data that describe the execution of a process, a process model is automatically learned using a process discovery algorithm. Limited interaction is possible: either filtering the input or adjusting configuration parameters.

knowledge of the discovery algorithms is generally necessary to select a suitable algorithm for a given use case. This practical problem, for instance, led to research on recommender systems supporting the selection of an appropriate discovery algorithm for given event data (Ribeiro et al., 2014; Pérez-Alfonso et al., 2015).

The circumstances above led to a new family of process discovery algorithms: algorithms that utilize *domain knowledge* in addition to event data. In this paper, we refer to domain knowledge as any prior knowledge about the process to be discovered besides event data. Domain knowledge is often available alongside existing event data, such as documents describing the process and the knowledge of process participants about the process. The utilization of domain knowledge in process discovery has various advantages. For instance, it helps: to reduce the negative impact of quality issues in the event data on the process model to be discovered, to overcome limitations when only incomplete process behavior is available, and to discover specific patterns that most conventional discovery algorithms cannot learn, e.g., long-term dependencies and process models with duplicate activity labels. Moreover, first studies indicate that exploiting domain knowledge within process discovery outperforms conventional discovery algorithms (Dixit et al., 2018b; Benevento et al., 2019).

Existing domain-knowledge-utilizing discovery approaches range from fully-automated approaches, which additionally use domain knowledge along with event data as input, to truly interactive/incremental approaches where a user plays a central role in the discovery process and actively provides domain knowledge to the algorithm during discovery. Fig. 2 visualizes various opportunities where domain knowledge can be utilized in the context of process discovery. For example, domain knowledge can be utilized in the event data pre-processing phase, e.g., to label observed process behavior, i.e., positive behavior should be incorporated in a process model, and negative behavior should not be incorporated. Thus, process discovery is transformed into a *supervised* learning task. Domain knowledge can also be used as an additional input of the discovery algorithm. For example, a handcrafted model describing parts of the model to be discovered could be provided to the discovery algorithm as a starting point next to event data. Also, during the execution of the discovery algorithm, i.e., during the learning of a process model, domain knowledge can be utilized, e.g., a user has to actively give feedback to the discovery algorithm on intermediate discovered process models and next steps. These examples illustrate that domain-knowledge-utilizing process discovery algorithms are much more heterogeneous compared to conventional process discovery from a user's perspective.

The field of domain knowledge utilization in process discovery has become increasingly attractive in recent years (cf. Fig. 3). This

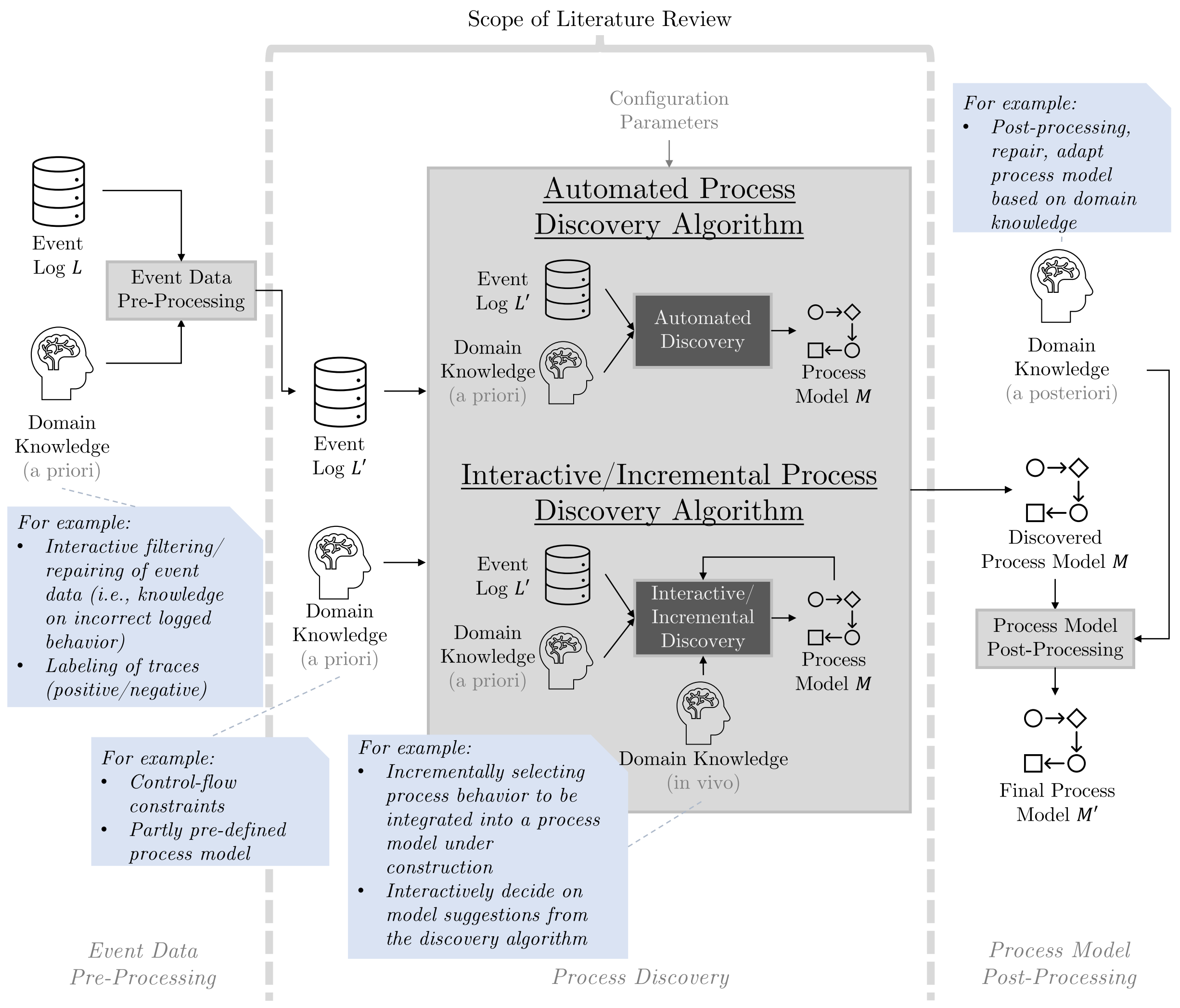
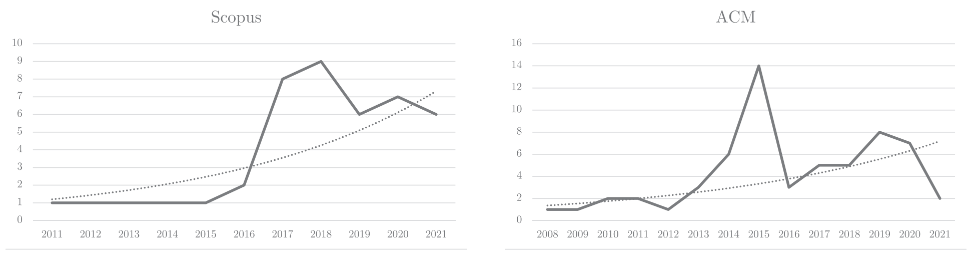

Scope of Literature Review
Event Log L
Event Data Pre-Processing
Domain Knowledge (a priori)
For example:
• Interactive filtering/ repairing of event data (i.e., knowledge on incorrect logged behavior)
• Labeling of traces (positive/negative)
Event Log L'
Domain Knowledge (a priori)
For example:
• Control-flow constraints
• Partly pre-defined process model
Configuration Parameters
Automated Process Discovery Algorithm
Event Log L'
Domain Knowledge (a priori)
Automated Discovery
Process Model M
Interactive/Incremental Process Discovery Algorithm
Event Log L'
Domain Knowledge (a priori)
Interactive/ Incremental Discovery
Process Model M
Domain Knowledge (in vivo)
For example:
• Incrementally selecting process behavior to be integrated into a process model under construction
• Interactively decide on model suggestions from the discovery algorithm
For example:
• Post-processing, repair, adapt process model based on domain knowledge
Domain Knowledge (a posteriori)
Discovered Process Model M
Process Model Post-Processing
Final Process Model M'
Event Data Pre-Processing
Process Discovery
Process Model Post-Processing

Fig. 2. Overview of various opportunities where domain knowledge about the process to be discovered can be utilized in the different phases of process discovery. Note that we distinguish two types of domain-knowledge-utilizing discovery algorithms: *automated* discovery algorithms and *interactive/incremental* discovery algorithms, allowing the user to dynamically provide domain knowledge based on intermediate results during the discovery.



(a) Scopus (`https://www.scopus.com`) (b) ACM (`https://dl.acm.org`)

Fig. 3. Growing interest in domain-knowledge-utilizing process discovery. Published documents by year matching the search term `("process mining" AND ("process discovery" OR "model repair") AND ("interactive" OR "incremental" OR "domain knowledge" OR "prior knowledge" OR "hybrid intelligence" OR "human-in-the-loop"))` using Scopus and ACM on August 18, 2021.

paper, therefore, presents a literature review of existing domain-knowledge-utilizing discovery approaches to answer the research question:

RQ1. What is the current state of research in process discovery that uses domain knowledge in addition to event data?Further, this literature review addresses the following research questions about domain-knowledge-utilizing process discovery approaches.

RQ1.1. At which point is domain knowledge utilized/provided?

RQ1.2. What types of domain knowledge are used, and how is domain knowledge specified?

RQ1.3. Which process model formalisms are used?

RQ1.4. Do the domain-knowledge-utilizing process discovery algorithms exhaust the full possibilities of a process model formalism?

RQ1.5. How are the algorithms realized in software tools?

In answering these research questions, we aim to achieve three main research objectives: (1) providing an overview of existing domain-knowledge-utilizing discovery approaches, (2) developing a taxonomy for said approaches, and (3) identifying open challenges and opportunities for future work. This paper presents the first literature review on this process discovery family to the best of our knowledge. In total, we identified thirteen discovery approaches. We propose a taxonomy for domain-knowledge-utilizing discovery approaches to systematically categorize and compare the identified approaches. Finally, we identify ten open challenges, highlighting the need for further research.

The remainder of this paper is organized as follows. Section 2 presents related work. Section 3 introduces basic concepts of process mining. Section 4 introduces a taxonomy for domain-knowledge-utilizing process discovery approaches. Section 5 provides an overview and systematic comparison of existing approaches. Section 6 discusses the adoption of domain-knowledge-utilizing process discovery in industry. Section 7 highlights open challenges for future work. Finally, Section 8 concludes this paper.

2. Related work

The field of process mining (van der Aalst, 2016) is a largely growing discipline, cf. (Reinkemeyer, 2020, Fig. 21.1). Various research areas have emerged over the years within process mining, for instance, process discovery (van der Aalst, 2010), conformance checking (Carmona et al., 2018; Dunzer et al., 2019), prediction (Tax et al., 2020), and event abstraction (van Zelst et al., 2021). There are reviews of the various research findings for each of the previously exemplified areas to summarize the state of knowledge and identify remaining challenges. In the remainder of this section, we focus on process discovery.

Many algorithms exist within conventional process discovery (cf. Fig. 1). A recent overview is presented in (Augusto et al., 2018), which systematically compares 35 conventional process discovery algorithms. Further, the authors benchmark a subset of the identified approaches using real-life event data. In (van Dongen et al., 2009), a review of conventional process discovery algorithms that are based on Petri nets is given. In (van der Aalst, 2018), a brief history of process discovery is presented, and general, underlying concepts of process discovery algorithms are presented. A further overview of process discovery algorithms can be found in (van der Aalst, 2016).

Domain-knowledge-utilizing process discovery techniques have recently increased in interest, cf. Fig. 3. The usage of prior knowledge about the process under investigation and the user's involvement in the discovery process is gaining attention in process mining. For example, in (Reinkemeyer, 2020, Chapter 21), the challenge of combining process modeling with process discovery is presented. The authors propose *hybrid models*, a mix of hand-crafted and automatically discovered process models, as one of the main challenges in process mining. The idea of hybrid models corresponds to the more general idea of hybrid intelligence, i.e., combining human and machine intelligence. According to (Kerremans et al., 2021, 38ff.), hybrid models are a promising approach ***for the further development of process mining.*** In (Benevento et al., 2019; Dixit et al., 2018b), use case studies in the healthcare domain are presented, showing that a specific domain-knowledge-utilizing approach outperforms conventional process discovery. The above references show that domain-knowledge-utilizing process discovery is relevant with high future potential. However, there is no literature review on domain-knowledge-utilizing process discovery to the best of our knowledge.

3. Background

This section outlines the main concepts within process mining. First, we introduce event data and process models. Finally, we present process discovery and model repair that can also be used for interactive process discovery.

3.1. Event data

Executing operational processes generates data in the information systems involved. Such data, capturing the execution of processes, are referred to as *event data*. Table 1 shows an example of an event log, i.e., a collection of event data, from a travel permit administration process, adopted from (van Dongen, 2020). Each row corresponds to an event capturing the execution of a process activity for a specific process instance, i.e., a *case*. Each column corresponds to an *event attribute*. For example, the first event indicates that the activity *Permit submitted* was executed for the process instance identified by the case-id 72861 on January 11, 2017 at 21:55:13 by an

Table 1
Example of an event log where each row corresponds to a single event. Events are ordered by case-id. The shown event data describes an administrative process of handling travel permits, adopted from (van Dongen, 2020). Note that, in general, events usually contain more attributes than shown here.

Case-ID	Activity	Timestamp	Resource	Resource Role	…
⋮	⋮	⋮	⋮	⋮	…
72861	Permit submitted	11.01.2017 21:55:13	Employee	Staff member	…
72861	Permit final approved	11.01.2017 21:55:18	Supervisor	Staff member	…
72861	Start trip	12.01.2017 00:00:00	Employee	Staff member	…
72861	End trip	20.01.2017 00:00:00	Employee	Staff member	…
72861	Declaration submitted	22.02.2017 18:08:07	Employee	Staff member	…
72861	Declaration final approved	22.02.2017 18:14:32	Supervisor	Staff member	…
72861	Request payment	03.03.2017 14:51:42	undefined	System	…
72861	Payment handled	06.03.2017 17:31:08	undefined	System	…
72379	Permit submitted	12.01.2017 09:17:36	Employee	Staff member	…
⋮	⋮	⋮	⋮	⋮	…

Employee who belongs to the group *Staff member*. Many process mining techniques, e.g., discovery algorithms, analyze event data to generate insights into the actual process.

We refer to the sequence of executed activities for a specific case as a *trace*. Note that each event relates to precisely one case. For instance, the trace for case 72861 is < *Permit submitted* , *Permit final approved*, *Start trip*, *End trip*, *Declaration submitted*, *Declaration final approved*, *Request payment*, *Payment handled* > . An event log typically consists of multiple cases, i.e., multiple recorded executions of a process. Events within a case are typically ordered based on their execution timestamp. We refer to unique sequences of executed activities as *trace variants*.

3.2. Process models

Process models allow us to specify the execution of processes graphically. Many formalisms exist, e.g., Business Process Modeling Notation (BPMN) (Chinosi and Trombetta, 2012), Petri nets (van der Aalst, 1998), and process trees (Leemans, 2017). In Fig. 4, we present three different process model formalisms describing the same process. Although all models in Fig. 4 describe the same control flow of activities, the expressiveness of the three formalisms generally differs. For example, the BPMN model specifies that a reminder is sent after 48 h if no declaration is submitted. Further, the model indicates which resources are involved in executing the different process activities. Such information cannot be modeled in the other two formalisms.

Most process model formalisms used in process discovery focus on the *control flow perspective*, i.e., which activities are present in a process and how they relate to each other. Next to the control flow perspective, some process model formalisms include a *data perspective* or an *organizational/resource perspective* (Dumas et al., 2013). For instance, BPMN provides modeling elements to specify the resources involved in executing an activity and, further, provides elements to model the flow of data artifacts. However, most process discovery algorithms only discover the control flow perspective.

Next to the process model formalism and its graphical representation, many process discovery algorithms restrict the class of discoverable models within a given formalism. Therefore, these algorithms constrain the chosen process model formalism, e.g., requiring specific structural properties, prohibiting activity duplicates and silent/unobservable activities in the discovered process models. For example, various important sub-classes of Petri nets play an important role in process discovery, e.g., Workflow nets (WF-net), sound WF-nets, free-choice WF-nets, and block-structured WF-nets (van der Aalst, 1996; Leemans, 2017). A detailed overview of different target process model classes is out of this paper's scope; we refer to (van der Aalst, 2016) for an overview. However, it is essential to know that the target model class of process discovery algorithms is of great importance because specific control flow patterns cannot be represented in every process model class. Therefore, we refer to limitations imposed by choice of a particular process model formalism and other possible restrictions made by the discovery algorithm on that formalism as *representational bias* (van der Aalst, 2016).

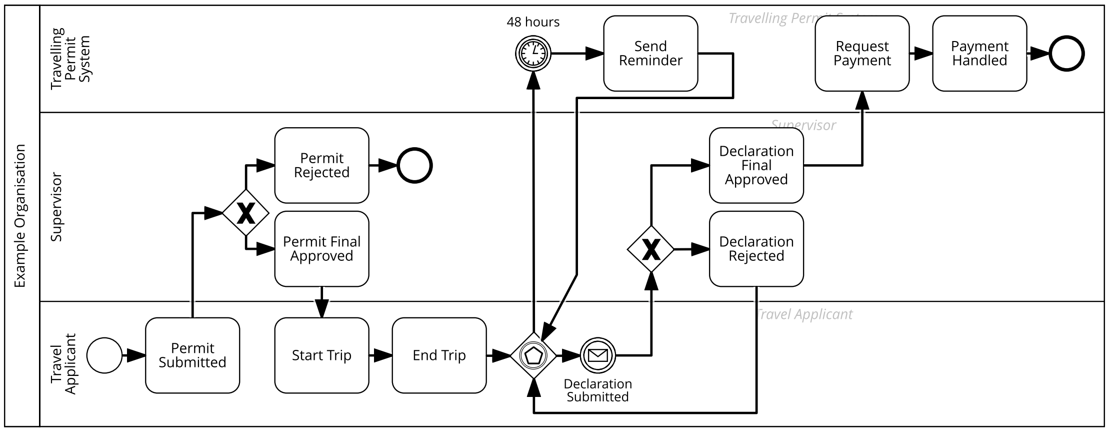
3.3. Process discovery

In business process management (BPM), process discovery is considered the collection of information about a process and the transformation of said information into a process model, which describes the process as it is executed in reality (Dumas et al., 2013). Thereby, techniques like document analysis, process observations, interviews and workshops with process stakeholders are applied to discover a process model. However, process discovery in the context of process mining, however, follows an entirely data-driven approach. Event data are considered a digital imprint of a process, and process discovery algorithms learn a process model from these event data, i.e., historical data on the executions of a process. In the remainder of this paper, we focus on process discovery as considered in process mining.

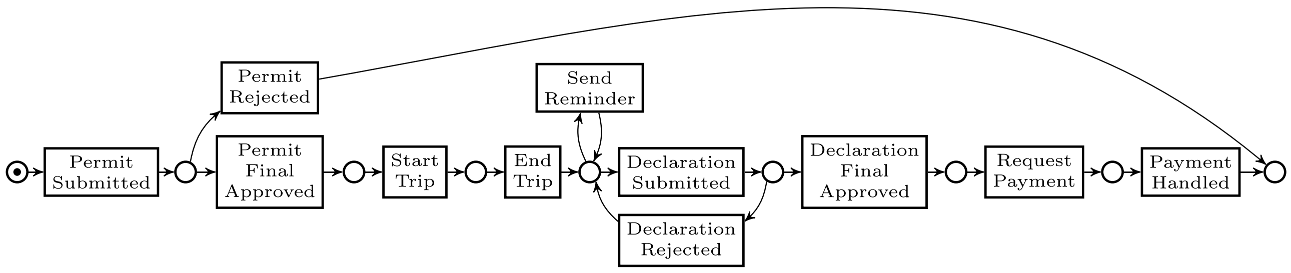
Most existing process discovery approaches are fully automated. Event data are fed into a discovery algorithm, and a process model is returned. Apart from adjusting configuration parameters of the discovery algorithm, no other input besides the event data is used. The configuration parameters of an algorithm can be used, e.g., to adjust the complexity of the resulting process model or to automatically filter the event data (e.g., filtering infrequent behavior). Apart from setting the configuration parameters, no interaction is possible. The actual process discovery phase, i.e., learning a model, is fully automated, cf. Fig. 1. We call this described family of process discovery approaches *conventional process discovery*.

3.4. Conformance checking

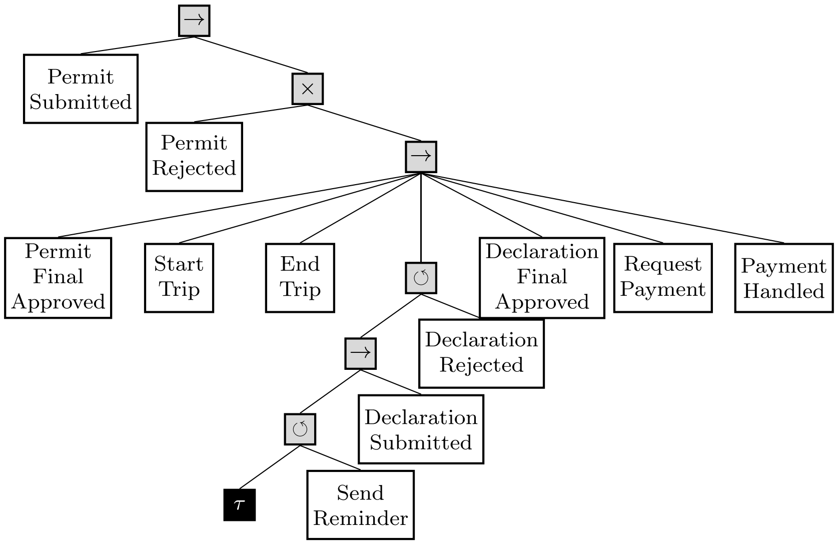
Conformance checking (Carmona et al., 2018) deals with comparing process models and event data. The purpose of conformance checking techniques is to quantify the extent to which observed process behavior, i.e., event data, conforms or does not conform to modeled process behavior, i.e., process models. In general, conformance checking is a separate research area within process mining. Nevertheless, conformance checking techniques are essential for domain-knowledge-utilizing process discovery. For example, in an interactive discovery phase, i.e., a user discovers a process model jointly with an algorithm, feedback on the current preliminary process model's conformance with the event data is of great importance.



(a) Example BPMN model. We refer to [19] for an introduction to BPMN



(b) Example Petri net with visualized initial marking. We refer to [1] for an introduction to Petri nets



(c) Example process tree. We refer to [1] for an introduction to process trees

Fig. 4. Three process models describing the same travel permit process, i.e., the same control flow of activities, in different process model formalisms, i.e., BPMN, Petri nets, and process trees. The process starts with submitting a permit, which is either rejected or approved. If it is approved, the trip takes place, i.e., *Start Trip* and *End Trip*. Next, a reminder is optionally sent multiple times. Once a declaration is submitted, it is either rejected, in this case a declaration has to be submitted again, or it is approved. Finally, payment related activities are executed. The BPMN model (a) additionally contains resource information and models that a reminder is sent 48 h after a declaration has not been submitted.

3.5. *Process model repair*

The area of process model repair (Fahland and van der Aalst, 2015), which is considered a separate research area, can be positioned between *process discovery* and *conformance checking* (Carmona et al., 2018). Process model repair techniques use conformance checking techniques to compare a given process model to given event data. If the model does not fully reflect the behavior recorded in the event data, repair the given model. Consider Fig. 5, in which we depict a schematic overview of process model repair. A process model M and event data are the input for model repair approaches. If the recorded process behavior as captured in the event data conforms to the process model M, there is nothing to repair, and M is returned, i.e., $M' = M$. Suppose the recorded process behavior is

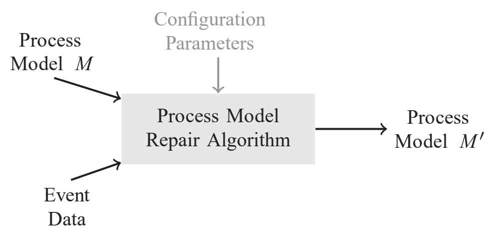


Fig. 5. Schematic visualization of process model repair. A process model M and event data are assumed as input. If all the recorded process behavior (i.e., event data) conforms to M, M does not get modified, i.e., $M = M'$. If there is process behavior recorded in the event data that is not included in model M, the process discovery approach repairs M such that the repaired model M' reflects all process behavior from the event data.

not fully supported by the process model M. In that case, M is altered such that all the recorded process behavior is supported and M' is as similar as possible compared to the original model M. Especially the latter goal, i.e., the repaired model should be as similar as possible to the original model, is important because it distinguishes process model repair from process discovery (besides the different inputs). Assuming this goal would not exist, one could apply process discovery on the given event data using an algorithm that guarantees to return a process model that describes all given behavior.

Although model repair techniques are intended to post-process a process model, they can also be used incrementally to learn a process model. Compared to Fig. 2, repair techniques can be considered an interactive/incremental process discovery algorithm. Given an initial model (a priori domain knowledge), missing process behavior can be added to the initial process model gradually (in vivo, user-selected behavior) or at once.

4. Distinguishing features for domain-knowledge-utilizing process discovery

To structure the field of domain-knowledge-utilizing process discovery, this section presents distinguishing features that allow us to systematically compare and categorize the different approaches. The remainder of this section is structured like follows. First, we explain how we derived the specific distinguishing features in Section 4.1. Subsequently, we present the specific derived features and corresponding characteristics. Finally, we present dependencies between the different features and characteristics.

4.1. Identification and construction strategy

This section highlights the construction of the proposed distinguishing features and their corresponding characteristics. We followed the definition of a taxonomy from Nickerson et al. (2010). The authors define a taxonomy as a set of dimensions $D_1, \ldots, D_n$. Each dimension D_i for $i \in \{1, \ldots, n\}$ consists of $k_i \geq 2$ potential characteristics. Further, when applying this taxonomy to a specific object, for each dimension exactly one potential characteristic applies, i.e., mutually exclusive and collectively exhaustive (Nickerson et al., 2010). However, for this literature review, we modified this definition by omitting mutual exclusiveness. Hence, we allow an object, i.e., a domain-knowledge-utilizing process discovery approach, to have more than one potential characteristic for a given dimension. In the following, we refer to the dimensions as features to differentiate from (Nickerson et al., 2010).

The goal of the distinguishing features is to answer the research questions posed in Section 1. The proposed distinguishing features are the result of various discussions within the author team and with other peers. The proposed characteristics for each distinguishing feature are derived from the analysis of the identified papers.

4.2. Distinguishing features

In this section, we present eight distinguishing features with corresponding characteristics. Fig. 6 summarizes the identified features. Note that we have identified first-level and second-level features. For instance, if an approach uses *explicit domain knowledge* (feature: domain knowledge type), we are further interested in whether the used *specification formalism* is *imperative* or *declarative* (Fig. 6).

4.2.1. Timing of domain knowledge provision

As shown in Fig. 2, domain knowledge can be provided and utilized at various times during process discovery. For instance, domain knowledge can be an additional input next to event data, domain knowledge can be utilized during an interactive discovery phase, and domain knowledge can be used after the discovery of a model to post-process it. We are therefore interested in the timing of domain knowledge provision.

We identify three characteristics regarding when domain knowledge is assumed to exist and be used: *a priori*, *in vivo*, and *a posteriori*. We categorize an approach as a priori if domain knowledge is an input next to event data, i.e., domain knowledge is required before the actual discovery phase starts. *In vivo* describes approaches that allow for domain knowledge injection during the discovery phase, e.g., by incorporating user feedback. An approach utilizes domain knowledge *a posteriori* if after the actual discovery phase domain knowledge is used, e.g., domain knowledge is used to post-process a discovered process model.

4.2.2. Domain knowledge type

Domain knowledge is a broad concept and can exist in many different forms. Therefore, we are interested in the type of domain knowledge assumed by the domain-knowledge-utilizing process discovery algorithms.

In this paper, we distinguish between *explicit* domain knowledge and domain knowledge in terms of *user feedback* to the algorithm. We consider explicit domain knowledge as domain knowledge which is specified in a formalism and fed into the discovery algorithm and, thus, the user explicitly provides it to the algorithm. For instance, precedence constraints between process activities and initial process models, which serve as a starting point for the discovery algorithm, are considered explicit domain knowledge. In contrast, user feedback is any decision that a user makes based on options provided by a discovery algorithm. An example of user feedback is an algorithm providing intermediate process models to the user in an ongoing discovery phase. The user must decide which process model to continue using. Compared to explicit domain knowledge, discovery algorithms actively request user feedback.

Note that the domain knowledge type and the timing of domain knowledge utilization are independent features. For instance, explicit domain knowledge can be incrementally provided by the user to the discovery algorithm in vivo, i.e., during the discovery phase. Alternatively, user feedback can be requested in vivo.

We are further interested in the domain knowledge specification formalism if an approach utilizes explicit domain knowledge. We distinguish *imperative* and *declarative* specification formalisms. Imperative formalisms follow a closed-world assumption, i.e., only behavior explicitly described in the formalism is allowed. In contrast, declarative formalisms follow an open-world assumption, i.e., all behavior is allowed that satisfies given restrictions specified in the formalism.

4.2.3. Degree of interactivity

Domain-knowledge-utilizing process discovery algorithms have different approaches regarding user involvement during the discovery phase. The spectrum ranges from completely automated

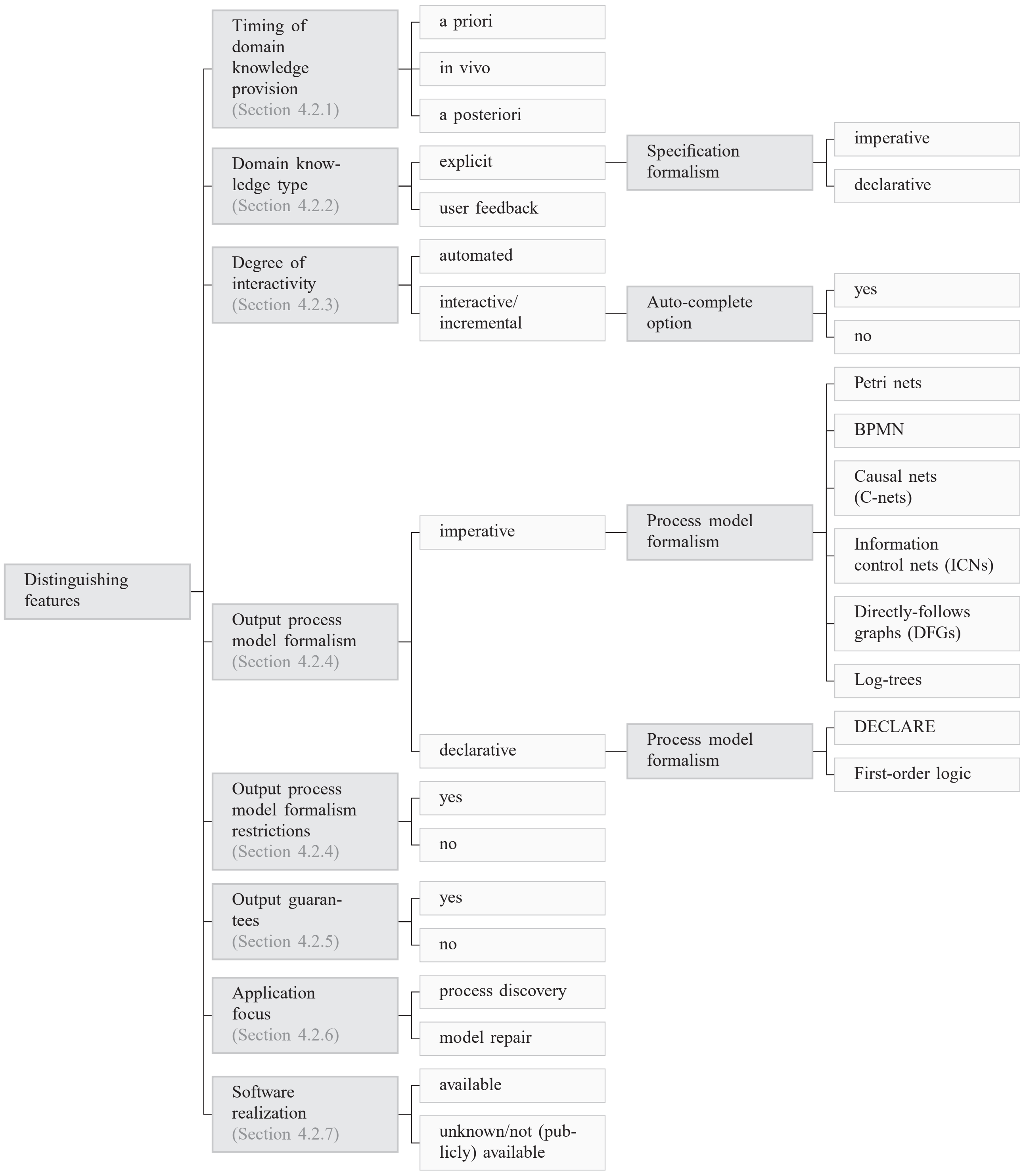

Distinguishing features
Timing of domain knowledge provision (Section 4.2.1)
a priori
in vivo
a posteriori
Domain knowledge type (Section 4.2.2)
explicit
Specification formalism
imperative
declarative
user feedback
Degree of interactivity (Section 4.2.3)
automated
interactive/incremental
Auto-complete option
yes
no
Output process model formalism (Section 4.2.4)
imperative
Process model formalism
Petri nets
BPMN
Causal nets (C-nets)
Information control nets (ICNs)
Directly-follows graphs (DFGs)
Log-trees
declarative
Process model formalism
DECLARE
First-order logic
Output process model formalism restrictions (Section 4.2.4)
yes
no
Output guarantees (Section 4.2.5)
yes
no
Application focus (Section 4.2.6)
process discovery
model repair
Software realization (Section 4.2.7)
available
unknown/not (publicly) available

Fig. 6. Overview of the identified distinguishing features (gray filled boxes) and their characteristics (light gray filled boxes).

algorithms, i.e., no user interaction is possible during the discovery phase, to algorithms in which the user has a central role in an interactive discovery phase and makes essential decisions regarding the process model to be discovered. According to Fig. 2, we distinguish two levels of interactivity: *automated* and *interactive/incremental*. We categorize approaches that do not support any option to inject domain knowledge, e.g., through user feedback, during the discovery phase as *automated*. For instance, approaches that use explicit domain knowledge and event data as input but do not offer user interaction are categorized as *automated* (cf. Fig. 2). Note that, by definition, automated approaches do not use domain knowledge in vivo.

In contrast, we categorize approaches that allow injecting domain knowledge during the actual discovery phase, e.g., through user feedback, as *interactive/incremental*. We further distinguish interactive/incremental approaches by the existence of an *auto-*

complete option. An auto-complete option ensures that the algorithm is able to autonomously learn a process model if the user decides to stop providing feedback to the algorithm.

4.2.4. *Output process model formalism*

The used process model formalism and potential restrictions of the formalism made by the discovery algorithm are of great importance when comparing different process discovery approaches because formalisms differ in their expressiveness. Reconsider the term representational bias, introduced in Section 3.3. The choice of a discovery algorithm on a particular formalism for the discovered process models and potential restrictions of the chosen formalism always implies a representative bias since every formalism has restrictions and limitations on which process behavior can be specified.

We are therefore interested in the formalisms used. First, we distinguish between *imperative* and *declarative* process model formalisms, similar to Section 4.2.2. Thus, we further distinguish the exact formalism used, i.e., *Petri nets*, *BPMN*, *causal nets (C-nets)*, *information control nets*, *directly-follows graphs (DFGs)*, log-trees, *DECLARE*, or *first-order logic*. We are further interested in whether discovery approaches constrain the chosen output formalism, as such constraints limit the expressiveness of a process model formalism. For instance, some process discovery algorithms cannot discover models that have the same activity label twice in different places in the model, i.e., *duplicate labels*.

4.2.5. *Output guarantees*

Guarantees regarding the discovered model are of great importance when comparing process discovery approaches in general. We are interested in whether the approaches can guarantee specific properties of the discovered process model regarding the given event data and the given domain knowledge. For example, if full replay fitness is ensured, i.e., all selected process behavior from the event data is represented in the discovered model. Alternatively, if it is always guaranteed that domain knowledge about the process being discovered is fully integrated into the resulting model, i.e., the domain knowledge is fully reflected by the model. However, guarantees can also refer to process models and their properties, e.g., whether a Petri net represents a *sound* Workflow net. Soundness (van der Aalst et al., 2011) is a favorable property of Workflow nets, e.g., ensuring that from any state of the model, there is always the possibility to complete the process and that there are no dead transitions within the model. We refer to (van der Aalst, 2016, 2011) for further information on soundness. In summary, such guarantees are important for the application of process discovery approaches because they make the discovery approaches reliable with respect to the output.

4.2.6. *Application focus*

As mentioned in Section 3.5, model repair techniques can also be used to discover a process model incrementally. Therefore, we are interested in the intended application focus of an algorithm. The application focus describes whether an approach has been intended for *process discovery* or *model repair* use cases. Starting from an initial process model, a user can gradually select process behavior that is not yet supported by the process model and apply a model repair approach. The resulting process model can then be reused in the next iterative execution in which further process behavior is added.

4.2.7. *Software realization*

The software support, especially for domain-knowledge-utilizing process discovery approaches, is of great importance. For example, interactive process discovery algorithms, i.e., human-in-the-loop process discovery, must communicate intermediate results, design decisions, or questions to the user, for which the algorithm requests feedback. Another example is the elicitation and specification of domain knowledge where the user needs support from software. For instance, supporting the user in specifying constraints in a particular formalism and verifying that the specified domain knowledge is free of contradictions. Therefore, we are interested in the software realization of the compared approaches.

Besides the existence of an implementation of an approach, we are interested in whether the software provides a graphical user interface (GUI) and how the user interaction is realized therein. For instance, how does the tool support the user to specify explicit domain knowledge, or how is the feedback loop from the user to the algorithm designed. Furthermore, we are interested in the technical realization of the software.

Table 2
Overview of mutual exclusiveness regarding characteristics for each feature.

Feature	Mutual Exclusive
Timing of domain knowledge provision	
Domain Knowledge Type	
Specification formalism (per single type)	✓
Degree of Interactivity	✓
Auto-complete option	✓
Output Process Model Formalism	✓
Process model formalism	✓
Output Guarantees	✓
Application Focus	✓
Software Realization	✓

4.3. *Dependencies among characteristics and features*

As discussed in Section 4.1, we do not generally assume mutual exclusiveness for the characteristics per feature. Therefore, we provide in Table 2 an overview for which feature mutual exclusiveness applies. For instance, the feature “Timing of domain knowledge provision” is not mutually exclusive; thus, a domain-knowledge-utilizing approach can be assigned various characteristic values for this feature (cf. Fig. 6).

Further, there exist two dependencies between certain features and characteristics. Thus, not all theoretically potential feature/characteristic combinations exist. In the following, we explain these two dependencies.

1. If the “Application focus” of an approach is model repair, the approach requires a process model as input, i.e., explicit domain knowledge. Hence, the approach’s “Domain knowledge type” is explicit. Note that the approach can additionally use user feedback because the characteristics of the feature “Domain knowledge type” are not mutually exclusive.
2. If the “Degree of interactivity” of an approach is automated, the “Timing of domain knowledge provision” feature cannot be in vivo.

5. Literature review

Based on the identified distinguishing features, this section provides a detailed overview of various process discovery algorithms which utilize domain knowledge. The remainder of this section is structured as follows. In Section 5.1, we present the design of our literature review, including the literature search process and the criteria we applied to assess if the literature is within the scope of this review. Next, Section 5.2 briefly and individually describes the identified discovery algorithms and their core idea regarding the utilization of domain knowledge. Thereby, we classify each approach based on the distinguishing features defined in Section 4. Finally, Section 5.3 discusses and compares the approaches and their characteristics per distinguishing feature.

5.1. Design

According to the diversity in the design of a literature review, we present our concrete approach in this section to ensure transparency regarding our design decisions. In this literature review, we primarily followed the guidelines given in Brocke et al. (2009) and Webster and Watson (2002). According to Brocke et al. (2009), the process of literature reviewing can be split up into five phases: 1) definition of the review scope, 2) conceptualization of the topic, 3) literature search and evaluation, 4) literature analysis and synthesis, and 5) research agenda. The outcome of the first phase, the definition of review scope, was presented in Section 1. We will further discuss the scope in more detail in this section. The outcomes of the second phase, conceptualization of topic, were presented in Section 3. Further, we presented distinguishing features in Section 4. The third phase, literature search, and evaluation of the found literature, i.e., filtering literature outside the scope of a literature review, is a crucial phase, according to Brocke et al. (2009). Thus, we exhaustively explain our approach regarding the third phase in this section. Phases four and five, the analysis of the identified literature, and the development of a research agenda will be presented in Sections 5.2, 5.3, and 7.

To identify relevant literature, we performed a keyword search on scientific databases. First, we queried the databases: Scopus,[1] ACM Digital Library,[2] and SpringerLink.[3] All databases index a plethora of journals, book series, conference proceedings, and workshop proceedings. For instance, Scopus indexes over 39,000 journals and conference proceedings in total.[4] For each database, we used the same logical combination of keywords. For example, we used the search term `TITLE-ABS-KEY("process mining" AND ("process discovery" OR "model repair") AND ("interactive" OR "domain knowledge" OR "hybrid intelligence" OR "human-in-the-loop"))` in Scopus. Note that the exact query syntax may differ depending on the database. Table 3 shows in the column *Hits, i.e.,* the number of literature found matching the search query for each database. Note that we do not further limit the search results, e.g., we do not apply a time filter or exclude workshop publications.

Given the search results, we evaluated the literature based on three knockout criteria according to the scope outlined in Section 1. First, we focus on approaches whose primary focus is process discovery. Second, we exclude work that solely provides conceptual ideas without providing a specific algorithm and use case studies applying process discovery. For instance, in Kindler et al. (2006), a general framework to incrementally discover process models from document and version management systems is presented, and in Hammori et al. (2004), requirements for an interactive workflow mining system are described.

Third, we focus on approaches that utilize domain knowledge in the process discovery phase (cf. Fig. 2). Thus, we exclude approaches that utilize domain knowledge solely in the event data pre-processing phase because these approaches address event data pre-processing in a general manner. Event data pre-processing is required/beneficial for any applied process mining technique, and it is considered a separate research area. For instance, we exclude domain-knowledge-utilizing filtering techniques (Sadeghianasl et al., 2020; Martin et al., 2019; Sani et al., 2019; Gschwandtner et al., 2014), label splitting techniques (Lu et al., 2016), and event abstraction techniques (van Zelst et al., 2021). Further, we exclude domain-knowledge-utilizing post-processing approaches for process models, i.e., approaches only applied after the actual discovery phase. However, as discussed in Section 4.2.6, process model repair techniques can also be used to discover a process model incrementally. Thus, we consider process model repair techniques in this literature review. It should be noted that we do not, though, provide a complete overview of all existing approaches to process model repair unless the techniques differ according to the identified distinguishing characteristic regarding use in the context of process discovery. Furthermore, from the point of view of process discovery, a detailed comparison of model repair techniques makes little sense since the mode of operation to discover a model is the same. Analogously, we exclude pure post-processing approaches. Moreover, we exclude approaches that use only domain knowledge and no event data to discover a process model. For example, in (Friedrich et al., 2011), the authors describe an approach that learns a model based only on natural language descriptions of a process and does not use any event data. Apart from the listed restrictions, we include all approaches that are intended to discover a process model and utilize domain knowledge in addition to event data.

Table 3 shows the number of found publications for each database and the number of remaining publications after gradually applying the three knockout criteria.[5] After removing duplicates, we identified 12 relevant approaches from the database queries. In addition to the conducted database queries, we applied backward search for all 12 identified publications (Brocke et al., 2009). Thus, we evaluated all references from each of the 12 identified publications. After removing duplicates, the 12 identified papers cite in total 270 publications. From these 270 publications, 10 fulfill all three criteria. We found one publication/approach among the ten identified publications we did not identify before with the database searches. Thus, we finally identified 13 domain-knowledge-utilizing discovery approaches that are included in this literature review. An overview of the identified approaches is presented in Table 4.

5.2. Descriptions of identified approaches

This section briefly describes each identified domain-knowledge-utilizing approach to provide an overview of how the approaches work and, in particular, how domain knowledge is utilized. Following the order in Table 4, we individually present the approaches. Note that in Section 5.3, we discuss and compare the different approaches based on the identified distinguishing features, cf. Fig. 6.

5.2.1. Approach A1

Goedertier et al. (2009) present an *automated* approach that utilizes a priori domain knowledge. The approach allows providing prior knowledge about which process activities are executed in parallel and which are sequentially executed. This information is utilized to enrich the given event log with missing behavior because event logs are often incomplete and do not contain all possible scheduling sequences of parallel behavior. For instance, assume there are n atomic process activities executed in parallel. In this case, there exist $n!$ many sequential executions of these n activities. Any missing sequential execution in the event log is automatically added based on the provided domain knowledge. Based on the domain knowledge, the approach learns a Petri net that may contain, for example, loops, duplicate labels, and non-free choice constructs.

5.2.2. Approach A2

Maggi et al. (2011) propose an *automated* approach to learn a process model, i.e., a DECLARE model graphically represented as a ConDec model (Pesic and van der Aalst, 2006). DECLARE consists of

[1] http://scopus.com
[2] https://dl.acm.org/
[3] https://link.springer.com/
[4] https://www.scopus.com/sources.uri (Accessed 11/25/2021).

[5] Detailed information on the search terms used, the literature found, and the application of the three criteria can be found in the supplementary material.

Table 3
Overview of the literature search process. First, we queried the given databases. We identified 12 publications matching the scope of this literature review. Starting from these 12 publications, we conducted backward search that resulted in 10 identified publications. The backward search identified a new previously unidentified approach. Thus, we identified in total 13 publications/approaches.

		Number of publications satisfying		
Source	Hits	Criterion 1.	Criteria 1. and 2.	Criteria 1., 2., and 3.
Scopus	43	20	13	9
ACM Digital Library	60	26	11	2
SpringerLink	408	79	63	10
Σ				12 (duplicates removed)
Backward Search (given the 12 identified publications)	270			10
Σ				**13** (duplicates removed)

Table 4
Overview of the identified approaches ordered by publication year. Note that when an approach is refined/extended over multiple publications, we list the year of the most recent publication.

Approach ID	First author	Ref.	Year	Description
A1	Goedertier et al.	Goedertier et al. (2009)	2009	Section 5.2.1
A2	Maggi et al.	Maggi et al. (2011)	2011	Section 5.2.2
A3	Rembert et al.	Rembert et al. (2013)	2013	Section 5.2.3
A4	Yahya et al.	Yahya et al. (2013)	2013	Section 5.2.4
A5	Dixit et al.	Dixit et al. (2015)	2015	Section 5.2.5
A6	Greco et al.	Greco et al. (2015, 2012)	2015	Section 5.2.6
A7	Fahland et al.	Fahland and van der Aalst (2015, 2012)	2015	Section 5.2.7
A8	Armas-Cervantes et al.	Armas-Cervantes et al. (2017a)	2017	Section 5.2.8
A9	Canensi et al.	Canensi et al. (2017)	2017	Section 5.2.9
A10	Dixit et al.	Dixit et al. (2018b)	2018	Section 5.2.10
A11	Yürek et al.	Yürek et al. (2018)	2018	Section 5.2.11
A12	Ferilli et al.	Ferilli and Esposito (2013), Ferilli (2020)	2020	Section 5.2.12
A13	Schuster et al.	Schuster et al. (2020)	2020	Section 5.2.13

behavioral templates to represent specific relationships between process activities. The user restricts (a priori domain knowledge) the algorithm to a desired subset of templates that the algorithm is allowed to use, i.e., which templates appear in the resulting process model. The approach automatically learns constraints based on the selected templates and generates a process model out of this. Furthermore, the approach guarantees that the given event data fits the discovered constraints.

5.2.3. *Approach A3*

Rembert et al. (2013) propose an *automated* discovery approach that utilizes *a priori, explicit* domain knowledge in the form of an augmented Information Control Net (ICN), i.e., an *imperative* formalism. An ICN represents relationships between process activities. In an extended ICN, the user assigns belief values (between 0 and 1) to the relationships, expressing the user's belief in the given dependency between the corresponding activities. Based on the event data and the augmented ICN, a process model is automatically learned and presented as an ICN. The authors explicitly highlight cases where event data contain uncertainties or where important but infrequent process behavior is present in the event data as a primary application of their approach.

5.2.4. *Approach A4*

Yahya et al. (2013) present an *automated* approach that utilizes a priori domain knowledge. A user can specify the following relationships between activities: causal, unrelated, parallel. Additionally, start and end activities can be specified. Based on the notion of activity proximity, which reflects the relations between activities as recorded in the event log, and the specified domain knowledge, a process model is learned, i.e., a directed graph, showing the directly-follows-relation between activities.

5.2.5. *Approach A5*

Dixit et al. (2015) present an *automated* approach. The approach uses an initial process model, event data, and user-specified constraints, i.e., DECLARE constraints, as input. Thus, the approach uses domain knowledge *a priori*. Given the initial process model, a brute force method is applied to generate different models by randomly applying edit operations. Next, the resulting models are evaluated based on the event data—standard quality measures are calculated, i.e., replay fitness, precision, generalization, and simplicity—based on the number of applied edit operations and the number of satisfied user-specified constraints. For an introduction to the standard quality measures, we refer to (van der Aalst, 2016, Chapter 6.4.3). These six measures are used to create a Pareto front of the best process models. Compared to Fig. 2, the approach returns multiple process models after the discovery phase. Finally, the obtained selection of process models is presented to the user, who must select the final process model. We categorize this *user feedback* as *a posteriori*.

5.2.6. *Approach A6*

Greco et al. (2015, 2012) propose an *automated* process discovery approach that utilizes *explicit* domain knowledge in the form of precedence constraints, i.e., a *declarative* formalism. These precedence constraints specify the dependencies among the process activities and are an additional input next to event data, i.e., *a priori*. Process models are represented as extended *C-nets*. The approach guarantees that the resulting C-net describes the behavior in the event data and fulfills the given precedence constraints; otherwise, no model is returned. The authors explicitly highlight the log completeness problem, i.e., when event data does not include all possible executions of the actual process to be discovered, as a suited use case of their approach.

5.2.7. *Approach A7*

Fahland and van der Aalst (2015, 2012) introduce the field of process *model repair* and present a process model repair approach. Given a process model and an event log containing traces that the model does not support, the repair approach alters the process model such that the model accepts all traces in the event log and is as similar—both from a language and a structural point of view—as possible to the original model. Therefore, we categorize this repair approach as *interactive/incremental*, i.e., starting from an initial process model (*explicit*, *a priori* domain knowledge), we repair the process model by incrementally adding trace variants to it (*explicit*, *in vivo* domain knowledge). The repair approach generally works on Petri nets, i.e., no restriction on a specific subclass. Moreover, an *auto-complete option* is theoretically given by automatically adding all remaining, non-fitting trace variants from a given log.

5.2.8. *Approach A8*

Armas-Cervantes et al. (2017a) propose a model repair approach, which we categorize as *interactive/incremental* like A7. Compared to A7, this approach relies on *user feedback*. The approach uses *BPMN* as process model formalism. By applying conformance checking techniques to a given event log and BPMN model, mismatches between the model and the data are detected and visualized in a BPMN model editor. These visualizations of discrepancies between the model and the log are an essential feature of the approach. Next to visualizing the discrepancies, the approach also visualizes repair proposals in the model. Based on the visual feedback, the user can then manually repair the process model or apply the suggested repair. As a possible disadvantage, the authors mention the potential effort for the user if there are many discrepancies between the log and the model.

5.2.9. *Approach A9*

Canensi et al. (2017) introduce an *interactive/incremental* process discovery approach tailored for medical process mining use cases. Initially, the approach automatically discovers a process model, i.e., a log-tree, using an existing, non-domain-knowledge-utilizing algorithm (Bottrighi et al., 2016). The discovered log-tree describes the entire event log, i.e., no automatic filtering is applied. Note that a log-tree can contain the same activity label various times because it encodes all traces from the event log in a tree structure. Given the discovered log-tree, a user can specify *a posteriori* subgraphs and provide these as domain knowledge to the approach. Given the subgraphs, the approach identifies all occurrences of these subgraphs in the log-tree and highlight them to the user. Based on *user feedback*, i.e., a user selects all or a subset of the identified subgraphs in the log-tree, the approach merges the selected subgraphs to obtain a simplified log-tree. However, note that this merging might result in a log-tree that describes additional behavior that is not present in the initially discovered log-tree.

5.2.10. *Approach A10*

Dixit et al. (2018b) propose an *interactive/incremental* process discovery approach where the user models the process model in an interactive editor with support of the underlying algorithm, which recommends modeling options based on the provided event log. Starting from an initial model that does not contain any activity from the event data, the user gradually extends the model by adding new elements. Thus, the approach uses *implicit* domain knowledge *in vivo*. The process model formalism used is free-choice (Desel and Esparza, 2005) workflow nets, a subclass of *Petri nets*. The approach guarantees that the Petri net under construction remains sound, a favorable property of Petri nets, by restricting the edit operations to the application of synthesis rules (Desel and Esparza, 2005). During editing the net, the user gets constant feedback and support from the tool regarding positioning new elements and relations between process activities. Additionally, the tool offers an *auto-complete option*.

5.2.11. *Approach A11*

Yürek et al. (2018) present an *interactive/incremental* discovery approach that uses explicit domain knowledge *in vivo*. First, the approach automatically discovers a DFG from the provided event data. Afterward, a user can explicitly change the model by aggregating, deleting, and adding activities to the DFG. However, the actual change of the DFG is processed by the algorithm and the user only specifies where/how/what should be changed, e.g., between process activity *a* and *b* process activity *x* should be executed. Then, the algorithm modifies the underlying data model accordingly and discovers a new process model. This procedure can be executed iteratively, cf. (Yürek et al., 2018, Fig. 3).

5.2.12. *Approach A12*

Ferilli and Esposito (2013) and Ferilli (2020) propose an *interactive/incremental* process discovery approach. First-Order Logic (FOL) (*declarative*) is the formalism used to represent process behavior. Starting from an initial model, i.e., a set of formulae that might also be empty, new process behavior can be added incrementally by the user. The approach uses *explicit* domain knowledge a priori (initial model) and in vivo (incremental adding of process behavior). The approach also learns information on the process beyond the control flow. For instance, the process model, i.e., FOL formulae, also includes information on resources involved in executing process activities. As for the other incremental approaches, the approach offers an *auto-complete option*.

5.2.13. *Approach A13*

Schuster et al. (2020) propose an *interactive/incremental* process discovery approach where a user incrementally selects process behavior not yet supported by a process model under construction. Starting from an initial process model (i.e., *explicit* and *a priori* domain knowledge), a user incrementally selects a trace variant not supported by the current process model yet. We categorize these incrementally selected trace variants as *explicit* and *in vivo* domain knowledge. The selected trace variant, potential previously selected trace variants, and the current process model are fed into the incremental discovery approach. The approach alters the process model such that the resulting model supports the previously added traces and the selected trace variant. The returned process model is then used as input in the next iteration, where the user adds the next trace variant to the model under construction. The approach uses *process trees* as a process model formalism and guarantees that the incrementally selected trace variants fit the resulting model. Further, the approach offers an *auto-complete* option by simply adding all the behavior of an event log to the model.

5.3. *Discussion*

This section compares the different approaches and discusses the different characteristics per distinguishing feature. A compact overview of this comparison can be found in Table 5, in which we list the different characteristics per distinguishing feature for each approach. Table 6 provides an overview of the number of approaches that have been categorized according to the different characteristics of the distinguishing features. Subsequently, each feature is discussed individually, following the order given in Fig. 6.

5.3.1. *Timing of domain knowledge provision*

Comparing the timing of domain knowledge provision, we observe that ten approaches assume domain knowledge to be provided *a priori*, six approaches utilize domain knowledge *in vivo*, and two approaches utilize domain knowledge *a posteriori*, i.e., after the

Table 5
Classification and overview of process discovery approaches utilizing domain knowledge and/or user feedback (La Rosa et al., 2011; Armas-Cervantes et al., 2017b; Dixit et al., 2018a; Schuster et al., 2021a).

		Domain knowledge		Output				
Approach	Degree of Interactivity	Timing of provision	Type	Process model formalism	Process model formalism restrictions	Guarantees	Application focus	Software realization
A1	automated	a priori	explicit (declarative) (parallel executed process activities)	imperative (Petri nets)	no	no	process discovery	available (ProM plugin)
A2	automated	a priori	explicit (declarative) (allowed DECLARE templates)	declarative (DECLARE)	no	yes (event data fits the discovered DECLARE model)	process discovery	available (ProM plugin)
A3	automated	a priori	explicit (imperative) (augmented ICN)	imperative (ICN)	no	no (resulting model contains only statistically significant behavior)	process discovery	unknown/not (publicly) available
A4	automated	a priori	explicit (declarative) (declarative constraints specifying the relationship between activities and specification of potential start and end activities)	imperative (DFG)	no	no	process discovery	available (ProM plugin called *Proximity Miner*)
A5	interactive/ incremental	(1) a priori & (2) a posteriori	(1) explicit (declarative) (DECLARE constraints) & (2) user feedback (selecting finally a process model from a set of result candidates)	imperative (Petri nets)	yes (process trees)	no (resulting process models candidates are randomly generated)	process discovery	unknown/not (publicly) available
A6	automated	a priori	explicit (declarative) (precedence constraints over process activities)	imperative (C-nets/ extended C-nets)	no	yes (event data and the discovered process model satisfy the precedence constraints)	process discovery	available (ProM plugin)
A7	interactive/ incremental (auto-complete option)	(1) a priori & (2) in vivo	(1) explicit (imperative) (initial process model) & (2) explicit (imperative) (traces incrementally selected by the user)	imperative (Petri nets)	no	yes (incrementally added traces fit the resulting model)	model repair	available (ProM plugin)
A8	interactive/ incremental (auto-complete option)	(1) a priori & (2) in vivo	(1) explicit (imperative) (initial process model) & (2) explicit (imperative) (traces incrementally selected by the user) & (2) user feedback (accepting/modifying proposed repair)	imperative (BPMN)	no (Note: focus only on the control flow perspective)	(yes) (incrementally added traces fit the resulting model if the user always accepts the proposed changes from the approach)	model repair	available (stand-alone tool called *Apromore* [55, 56])
A9	interactive/ incremental	a posteriori	explicit (imperative) (subgraphs contained in the log-tree) & user feedback (selection of detected subgraphs in the process model)	imperative (log-tree)	no	yes (discovered model describes all behavior from the log)	process discovery	unknown/not (publicly) available
A10	interactive/ incremental (auto-complete option)	in vivo	user feedback (supported by suggestions from the algorithm, a user creates the process model in an editor)	imperative (Petri nets)	yes (free-choice WF-nets)	yes (discovered model is always sound)	process discovery	available (ProM plugin, called *Prodigy* [57])
A11	interactive/incremental (auto-complete option)	in vivo	explicit (imperative) (merging, deletion, adding requests of activities from the user)	imperative (DFG)	no	no	process discovery	unknown/not (publicly) available
A12	interactive/ incremental (auto-complete option)	(1) a priori & (2) in vivo	(1) explicit (imperative) (initial process model) & (2) explicit (imperative) (traces incrementally selected by the user)	declarative (First-order logic formulae)	yes (Datalog Horn clauses)	yes (incrementally added traces fit the resulting model, i.e., FOL formulae)	process discovery	unknown/not (publicly) available
A13	interactive/ incremental (auto-complete option)	(1) a priori & (2) in vivo	(1) explicit (imperative) (initial process model) & (2) explicit (user incrementally selects traces)	imperative (Petri nets)	yes (process trees)	yes (incrementally added traces fit the resulting model)	process discovery	available (stand-alone tool called *Cortado* [58])

actual discovery phase. Note that the feature timing of domain knowledge provision is not mutually exclusive (cf. Table 2), e.g., an approach can utilize domain knowledge a priori and in vivo. Further note that we have excluded approaches that focus solely on post-processing process models, cf. Section 5.

Interestingly, only six approaches support *in vivo* utilization of domain knowledge, i.e., these allow the user to interact with the algorithm during the actual discovery. Three out of these six approaches (i.e., A7, A12, and A13) follow an incremental discovery approach, i.e., traces are incrementally selected by the user and added to a process model under construction by the algorithm. Two other approaches (i.e., A8, A10) assume that the user is actively providing feedback by either modeling or repairing the process model guided by suggestions from the algorithm. Although A11 also allows editing an initially discovered process model, the algorithm does not provide the user with any modeling suggestions compared to the previous two approaches. This observation shows that current algorithms are very limited in terms of the variety of in vivo utilization of domain knowledge and that many possibilities of in vivo utilization of domain knowledge have not yet been explored.

5.3.2. *Domain knowledge type*

Twelve out of thirteen approaches utilize explicit domain knowledge, and only four , i.e., A5, A8, A9, and A10, utilize user feedback. Only A10 solely utilizes user feedback. Note that the feature domain knowledge type is not mutually exclusive, cf. Table 2. Further note that the automated approaches utilize only explicit domain knowledge since these approaches do not offer user interaction, as discussed in Section 4.3.

The process discovery algorithms incrementally adding traces to a process model under construction (i.e., A7, A8, A10, and A12) use an initial model a priori, explicit domain knowledge, and user-selected traces in vivo. The user-selected traces are explicit domain knowledge within the discovery phase.

The remaining approaches that utilize explicit domain knowledge (i.e., A1-A6, A9 and A11) use declarative as well as imperative formalisms. We observe the following formalisms to represent explicit domain knowledge: control flow constraints (e.g., precedence constraints and DECLARE constraints), initial process models, augmented ICNs, and incrementally selected traces. Precedence constraints and DECLARE constraints are both declarative formalisms.

Table 6
Overview of the number of approaches assigned to the individual characteristics, cf. Fig. 6. We marked features that are not mutually exclusive with *, cf. Table 2. For the calculation of the relative numbers of the characteristics of the second level, we have taken the absolute number of the characteristics of the corresponding first level as a basis.

Distinguishing feature **2nd level distinguishing feature**	**Characteristic**	**#Approaches**
Timing of domain knowledge*	a priori	10 (≈ 77%)
	in vivo	6 (≈46%)
	a posteriori	2 (≈15%)
Domain knowledge type*	explicit	12 (≈92%)
Specification formalism	imperative	7 (≈58%)
	declarative	5 (≈42%)
	user feedback	4 (≈ 31%)
Degree of interactivity	automated	5 (≈38%)
	interactive/incremental	8 (≈62%)
Auto complete option	yes	6 (≙75%)
	no	2 (≙25%)
Output process model formalism	imperative	11 (≈85%)
Process model formalism	Petri nets	5 (≈45%)
	BPMN	1 (≈9%)
	Causal nets (C-nets)	1 (≈9%)
	Information control nets (ICNs)	1 (≈9%)
	Directly -follows graphs (DFGs)	2 (≈18%)
	Log-trees	1 (≈9%)
	declarative	2 (≈15%)
Process model formalism	DECLARE	1 (≙50%)
	First-order logic	1 (≙50%)
Output process model formalism restrictions	yes	4 (≈31%)
	no	9 (≈69%)
Output guarantees	yes	8 (≈62%)
	no	5 (≈38%)
Application focus	process discovery	11 (≈85%)
	model repair	2 (≈15%)
Software realization	available	8 (≈62%)
	unknown/ not (publicly) available	5 (≈38%)

However, the DECLARE formalism is more expressive than precedence constraints because DECLARE contains precedence constraints and offers further constraints, e.g., the DECLARE formalism enables the user to specify the exact number of times a process activity must occur. Augmented ICNs are an imperative formalism modeling the control flow of process activities. However, in practical use, it might be difficult for the user to assign a value between 0 and 1 to the different control flow relations, which expresses the certainty of the user for to the relation.

Comparing the utilized user feedback across the approaches (i.e., A5, A8, A9, and A10), we observe the following interaction options: selecting a final model out of multiple candidates (i.e., A5), modeling/repairing a model based on suggestions from the algorithm in an editor (i.e., A10/A8), and selecting identical process model parts that are to be merged (i.e., A9).

5.3.3. *Degree of interactivity*

Comparing the degree of interactivity, we observe that five approaches (i.e., A1-A4 and A6) are fully-automated. The remaining approaches are classified as interactive/incremental, i.e., they support in vivo provision of domain knowledge or some form of user feedback. All interactive/incremental approaches, except for A5 and A9, offer an auto-complete option, i.e., these approaches can also be used in an automated way without mandatory domain knowledge provision. Note that to both approaches, i.e., A5 and A9, an auto-complete option could be easily added. In the case of A5, the approach would have to decide at the end which process model to return based on some predefined criteria instead of leaving this decision to the user. Similarly, A9 could simply merge all detected identical process model parts.

5.3.4. *Output process model formalism and restrictions*

Comparing the used process model formalisms, we see that a wide range of different formalisms is used. We observe that only two approaches use declarative formalisms to represent the discovered model, while the other approaches use imperative formalisms. However, most of the used process model formalisms are not common in industrial applications, for instance, process trees, C-nets, and ICNs.

The most common formalism used by the identified approaches is Petri nets. Except for one approach (i.e., A7), approaches using Petri nets focus on a subclass, i.e., either process trees or free-choice workflow nets. Process trees are widely used in state-of-the-art conventional process discovery approaches, e.g., the Inductive Miner (Leemans et al., 2013) and the Evolutionary Tree Miner (Buijs et al., 2014). However, it is important to note that the expressiveness of process trees is limited, e.g., long-term dependencies—a choice at the beginning of a process influences a choice later in the process—cannot be modeled. On the other hand, process trees guarantee favorable behavioral characteristics, e.g., they are deadlock-free, i.e., a process tree cannot be in a state where no more activity can be executed although the end has not yet been reached. Therefore, process trees are an important subclass of Petri nets for process mining use cases but have inherent limitations. Similarly, according to (van der Aalst, 2016, cf. Chapter 6.4), free-choice workflow nets cannot reflect all behavior from real-life processes. A4 and A11 use a simple graph representation, i.e., a DFG, as process model formalism. Although DFGs are widely used in industrial applications of process mining, this formalism has low expressiveness compared to the other imperative formalisms observed. For example, control flow operators such as choices and parallel joins/splits cannot be modeled because DFGs represent only process activities and their potentially directly following process activities. van der Aalst (2019) extensively explains the use of DFGs and the pitfalls of deriving misleading diagnoses from DFGs.

5.3.5. *Output guarantees*

We observe that eight approaches provide guarantees concerning the discovered process model, and five approaches do not. Most approaches guarantee that the initially given or incrementally selected event data fit the discovered process model, i.e., the recorded process behavior in the event data conforms to the discovered process model. Thereby, it is important to distinguish between approaches guaranteeing that the entire given event data are fitting (i.e., A2, A6, A9) and approaches (i.e., A7, A8, A12, and A13), allowing the user to interactively select the desired process behavior to be included in the process model under construction. If there is no option for the user to interactively influence the process model during the process discovery phase, e.g., by selecting desired process behavior from the event data to be included in the model or by editing the process model under construction, data-quality is even more critical. All process behavior captured in the given event data—including noise, incomplete behavior, etc.–will be represented in the discovered process model if the approach guarantees full fitness. Therefore, the quality of the input event data is critical to the successful application of the techniques above.

Interestingly, only approach A2 provides guarantees regarding the *a priori*, *explicit* domain knowledge, and the resulting process model. The approach guarantees that the resulting model fits the given event data and the precedence constraints between process activities given as explicit domain knowledge. This leads to an interesting design decision. Suppose domain knowledge and event data have conflicting information about the process to be discovered. In that case, the approach must decide, i.e., either return nothing because no solution satisfies both domain knowledge and event data, favor domain knowledge over event data, or vice versa. For instance, the user could model the precedence constraint specifying process activity a is always executed before b; however, we observe the exact opposite in the event data. A6 is designed to not return a model unless both event data and domain knowledge can be fully represented in a process model. Although we have argued that guarantees regarding the discovered process model are of great importance, the above example shows that such guarantees can also lead to challenges. For example, approach A3 offers no guarantees; however, it may still be useful in practice not to always favor domain knowledge over event data, or vice versa, but to focus on the statistically significant behavior of event data and domain knowledge instead.

5.3.6. *Application focus*

We observe that eleven out of thirteen approaches have been mainly designed for process discovery by comparing the application focus. Nevertheless, we identified two model repair approaches that can also be used to discover a process model incrementally, as described in Section 3.5. Although this observation is not surprising, the two process model repair techniques demonstrate the link from model repair to interactive and incremental process discovery.

5.3.7. *Software realization*

Regarding software support, we note that eight approaches have been realized in software. Of these, six approaches are available as a plugin within the process mining toolkit *ProM* (Verbeek et al., 2010).[6] ProM is an open-source software tool for process mining that provides general process mining functionality and offers the possibility to distribute algorithms and approaches via a plugin manager. A13 has been implemented in a standalone software tool called *Cortado* (Schuster et al., 2021a),[7] which is designed explicitly for incremental process discovery. A8 has been integrated into the tool called *Apromore* (La Rosa et al., 2011).[8] Apromore is a general process mining software tool that includes various process mining functionality next to the process model repair functionality. For five approaches, no software realization is (publicly) available, or the status is unknown to us. All software realizations offer a graphical user interface, allowing users to apply the approach easily.

6. Adoption in industry

In this section, we briefly focus on the adoption of process mining, especially of domain-knowledge-utilizing process discovery approaches, in industry. In general, process mining technologies are used in a wide range of industrial sectors, for instance, manufacturing (Rozinat et al., 2009; Park et al., 2015), healthcare (Rojas et al., 2016; Yang and Su, 2014), education (Bogarín et al., 2018) and auditing (Jans et al., 2013; van der Aalst et al., 2010). We refer to Reinkemeyer (2020) for various examples of process mining use cases at various organizations, e.g., Siemens, Uber, BMW, and Bosch. Further use case studies can be found in (Thiede et al., 2018).

The overall market for process mining is growing. The market size growth of 2021 is estimated to be 70%, and in 2022 the market growth rate is estimated to be between 40% and 50% (Kerremans et al., 2021). These growth rates indicate the increasing relevance of process mining technology in industry. According to Kerremans et al. (2021), more than 40 commercial process mining tools are available from various vendors, including Apromore, Celonis, IBM, SAP, and UiPath.

A comparison of different process mining tools is outside the scope of this literature review. In general, the existing commercial tools offer different process mining technologies, but often there is a focus on the functionality offered (Kerremans et al., 2021). According to Martin et al. (2021, Table 4), existing solutions often do not cover a wide range of process mining functionalities.

Looking at process discovery functionality in commercial tools, we observe that many existing tools offer limited process discovery functionality. From our perspective, many of the existing conventional process discovery algorithms developed in academia (Augusto et al., 2018; van Dongen et al., 2009) are not used in commercial tools. Often, commercial process mining tools offer only simplified process discovery techniques that discover a Directly-Follows Graph (DFG) as a process model. However, DFGs lack advanced control flow structures, and their expressiveness is significantly limited compared to other process model formalisms, like Petri nets and BPMN. We refer to van der Aalst (2019) for an extensive discussion on the limitations of DFGs. Nevertheless, process discovery is a common use case for process mining in industry (Kerremans et al., 2021).

The development of domain-knowledge-utilizing process discovery approaches is still at an early stage. Few approaches exist compared to conventional process discovery. From the 13 identified approaches in this literature review, eight are realized in software tools. Six out of these eight approaches are available as a plugin within ProM (Verbeek et al., 2010), an open-source process mining tool developed by academia. Although ProM is a great success in academia and offers a variety of process mining functionality, it is hardly used in industry according to our knowledge due to professional support, scalability, and ease of use. Two out of the eight approaches that have been realized in software tools are available in standalone tools. Approach A13 (cf. Table 5) is realized in a tool that is developed by academia and is in an early stage of development.

[6] https://www.promtools.org/

[7] https://cortado.fit.fraunhofer.de/

[8] https://apromore.org/

Thus, this approach is not used in industry at this moment. Approach A8 (cf. Table 5) is the only approach identified in this literature review realized in a commercial, open-source process mining tool. In addition, we are not aware of any other identified domain-knowledge-utilizing process discovery approaches that commercial vendors have adopted in their software tools.

In short, domain-knowledge-utilizing process discovery is at an early stage from both an academic and an industrial perspective. Nevertheless, process discovery is a central use case of process mining in industry (Kerremans et al., 2021). Given the challenges, opportunities, and future directions of process mining presented in Section 2, e.g., hybrid intelligence and hybrid models, domain-knowledge-utilizing process discovery techniques have a great potential to become a valuable asset within the set of process mining techniques for industry.

7. Challenges for future work

This section lists the main challenges and directions for future work and the design of domain-knowledge-utilizing process discovery approaches. The identified challenges are based on the observations made when comparing the presented approaches, cf. Section 5, and the status regarding the adoption in industry presented in Section 6. In total, we have identified ten challenges for future work, which are listed below.

7.1. Blending explicit domain knowledge and user feedback

Ten identified approaches (cf. Table 5) use either solely explicit domain knowledge or user feedback during the discovery phase. However, both types of domain knowledge are independently proven to be valuable in the context of learning a process model from event data. Therefore, approaches should not be limited to either user feedback or explicit domain knowledge and utilize both types.

7.2. Increased interaction

An interesting direction for future work is to combine different approaches to leverage diverse domain knowledge, rather than focusing only on limited and specific domain knowledge, as most existing approaches do. For example, combining incremental process discovery, i.e., a user incrementally selects trace variants, with an assisted model editor, where the user is guided by an algorithm when editing the model. Moreover, approaches might offer the user the option to provide feedback on various levels of detail, i.e., different options to integrate domain knowledge in vivo. For example, a user could provide feedback on the exact positioning of some process activities within a model and, at a later stage of the process discovery phase, only guide the algorithm in which trace variants should be added. In short, the goal should be to create overarching approaches that combine different ideas, each of which has individually proven useful in the context of process discovery.

7.3. Offering different interactivity modes

Existing approaches can be clearly categorized to one of the two introduced degrees of interactivity, i.e., automated and interactive/incremental (cf. Section 4.2.3). Six out of eight interactive/incremental approaches offer an auto-complete option. However, ideally, an approach offers both modes giving the user maximal flexibility to discover a process model. For example, imagine a discovery approach is executed in an automatic mode. During the automated discovery phase, the user gets constant feedback and observes an undesirable tendency in the model being constructed. Then, the user switches to an interactive/incremental mode, makes slight changes to the model, and switches back to automatic mode The described mode-switching requires that the user is able to interact at specific points during the discovery, i.e., break points. These breakpoints have to be chosen carefully, on the one hand, to give the user enough opportunities to interact, and on the other hand, not to be too fine-grained to avoid overburdening the user.

7.4. Scalable conformance checking

Many presented approaches use conformance checking techniques. Especially alignments (Adriansyah, 2014)– a state-of-the-art conformance checking technique that provide detailed diagnostics compared to other conformance checking techniques– are used in four identified approaches, i.e., A5, A7, A8, and A13. Alignments are used to relate a process model with an event log to, e.g., recommend edit options and repair a model. In general, many conformance checking techniques, especially alignments, are computationally complex (Carmona et al., 2018) and provide non-deterministic diagnostics, i.e., there are usually many optimal alignments (Adriansyah, 2014). However, in general, domain-knowledge-utilizing discovery approaches need a constant comparison between the model and the event log. Therefore, fast conformance checking techniques are critical to enabling interaction and fast suggestions from an interactive process discovery algorithm. In this context, research on the applicability of conformance approximation techniques, such as Bauer et al. (2020), Fani Sani et al. (2020), and Schuster et al. (2021b), within interactive process discovery is needed.

7.5. Minimizing the representational bias

As discussed in Section 5.3.4, the approaches A4, A9, and A11 use simple process model formalisms, i.e., DFGs and log-trees that both have limited expressiveness compared to, e.g., Petri nets and BPMN. Also, in Section 6, we highlighted the dominance of DFGs as the prominent process model formalism in commercial tools and pointed to (van der Aalst, 2019), discussing the limitations of DFGs. Also in (van der Aalst et al., 2012b), the authors mention the need for process model formalism that supports basic control flow structures. Further, four presented algorithms that use advanced process model formalism restrict the class of discoverable models. Such restrictions are generally made because discovery *algorithms can be designed more easily* if certain restrictions on the output process model formalism can be assumed. For example, approaches A5 and A13 work on process trees (i.e., sound, block-structured Workflow nets) or free-choice Workflow nets. Both block-structured and free-choice Workflow nets are important subclasses of Petri nets. However, the expressive power of these subclasses is limited compared to Petri nets. Note that many real-life processes tend to be non-free-choice and non-block-structured (van der Aalst, 2016, cf. Section 6.4). Further, common restrictions are, e.g., the absence of duplicate activity labels, i.e., multiple elements in a process model are labeled with the same activity, and silent labels, which are needed to model specific control flow patterns in various formalisms. Therefore, in future discovery approaches, the discoverable models' target class and potential restrictions should be carefully considered. We refer to (van der Aalst et al., 2012a) for an extensive discussion on the representational bias.

7.6. Enhanced process model visualizations

As we can observe in Table 5, many different process model formalisms are used to visualize process models, e.g., C-nets, Petri nets, process trees, DFGs, and BPMN. It is important to note that most of these modeling formalisms are hardly used in industrial practice. Of the five modeling formalisms mentioned, one could argue that BPMN (Dumas et al., 2013) and DFGs, as discussed in Section 6, are the most widely used in industry. Especially for interactive discovery approaches, it is of utmost importance to choose an appropriate process model visualization because a user has to quickly understand a process model in an interactive process discovery setting. The relevance of the challenge described here is supported by (Martin et al., 2021, Table 4). The authors identify non-standard visualizations of process mining outcomes as a challenge for the adaption of process mining in industry. Also in (van der Aalst et al., 2012b), the authors list the challenge of improving usability and understandability for non-experts as a key challenge. Note that this does not necessarily imply that also the underlying algorithms need to be specified for the chosen visualization. For example, a process tree may be visualized as a BPMN model to the user, but internally the algorithm operates on a process tree data structure.

7.7. Domain knowledge specification

Many approaches utilize explicit domain knowledge. However, little research has been done on supporting the user in specifying the domain knowledge in the context of process discovery. For instance, we are not aware of studies within process mining that analyze which formalisms are easy to understand by end-users and how a user can be assisted/guided in specifying explicit domain knowledge. In Martin et al. (2021), the authors present challenges regarding the application process mining in organizations. The authors identify insufficient analytical and technical skills of people applying process mining in industry. Therefore, we conclude that user support is of great importance in the specification of domain knowledge.

7.8. Domain knowledge prioritization

As discussed in Section 5.3.5, domain-knowledge-utilizing algorithms have to decide if the given domain knowledge and the given event data have conflicting information on the process to be discovered. An interesting direction for future work is to involve the user whenever such conflicts occur within the process discovery phase and let the user interactively decide which information, i.e., event data or domain knowledge, is prioritized per conflict. Additionally, dynamic and automatic strategies, i.e., not to statically always favor domain knowledge over event data or vice versa if conflicts occur, are an interesting direction for future work.

7.9. Software support

Even more than with conventional process discovery, sophisticated software implementing the interactive/incremental discovery approaches is of great importance for the acceptance and success of said techniques. Since conventional process discovery is fully-automated (cf. Fig. 1), the user's interaction with the algorithm is limited; hence, sophisticated software support is less critical for conventional process mining. However, there is a clear need for sophisticated software support for process discovery approaches that utilize user feedback, i.e., where user computer interaction is present. As discussed in Section 6, only approach A8 is available in a commercial tool (La Rosa et al., 2011), and only eight out of 13 approaches are realized in software tools.

7.10. Alternative perspectives

Most presented approaches focus solely on learning the control flow of a process. However, event data typically contain much more information, e.g., resource and time information. These data are often not exploited by process discovery algorithms, although process model formalisms like BPMN offer modeling elements to represent information beyond the control flow, e.g., organizational structures and data flow. For example, consider Fig. 4a. Especially in an interactive process discovery setting, information beyond the control flow in event data, e.g., resource and timing information, combined with the users' domain knowledge on the process under consideration could be advantageous for obtaining more extensive process models. The challenge is supported by findings presented in Martin et al. (2021). The authors identify the analysis of processes with the focus on resources as an opportunity for using process mining in industry.

The presented challenges and directions for future interactive, domain-knowledge-utilizing approaches show that many interesting open research questions exist, and much can still be achieved in this area. We hope that these identified challenges and opportunities will provide new impetus and ideas for further developments.

8. Conclusion

Process mining provides various methods, techniques, and tools to analyze operational processes in a data-driven manner systematically. Process discovery is a key discipline within process mining. Conventional process discovery deals with the automatic learning of a process model from event data. The low quality of models discovered with conventional process discovery algorithms and the presence of prior knowledge, i.e., domain knowledge, about the process to be discovered lead to domain-knowledge-utilizing process discovery algorithms.

This paper provided a systematic review of process discovery techniques that additionally utilize domain knowledge next to event data. First, we identified distinguishing features to categorize and classify domain-knowledge-utilizing process discovery approaches. Then, based on the distinguishing features, we compared and discussed thirteen identified approaches selected by objective criteria. Finally, based on the comparison and the discussion, we identified ten open challenges for future domain-knowledge-utilizing process discovery approaches. These challenges highlight the potential of using domain knowledge and user feedback within data-driven process discovery and demonstrate the need for further development in this area.

Declaration of Competing Interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: The third author, Wil M. P. van der Aalst, has an affiliation with the commercial software vendor Celonis SE. However, this affiliation did not influence the study in any way or affect objectivity.

Appendix A. Supplementary material

Supplementary data associated with this article can be found in the online version at doi:10.1016/j.compind.2022.103612.

References

Adriansyah, A., 2014. Aligning Observed and Modeled Behavior (Ph.D. thesis). Eindhoven University of Technology. DOI: 10.6100/IR770080.

Armas-Cervantes, A., van Beest, N.R.T.P., Rosa, M.L., Dumas, M., García-Bañuelos, L., 2017a. Interactive and incremental business process model repair. In: Proceedings of the On the Move to Meaningful Internet Systems. OTM 2017 Conferences – Confederated International Conferences: CoopIS, C&TC, and ODBASE 2017, Rhodes, Greece, 23–27 October 2017, Part I, vol. 10573 of Lecture Notes in Computer Science, Springer, pp. 53–74. DOI: 10.1007/978-3-319-69462-7_5.

Armas-Cervantes, A., van Beest, N.R.T.P., Rosa, M.L., Dumas, M., Raboczi, S., 2017b. Incremental and interactive business process model repair in apromore. In: Proceedings of the BPM Demo Track and BPM Dissertation Award co-located with 15th International Conference on Business Process Modeling (BPM 2017), Barcelona, Spain, 13 September 2017, vol. 1920 of CEUR Workshop Proceedings, CEUR-WS.org.URL: http://ceur-ws.org/Vol-1920/BPM_2017_paper_206.pdf.

Augusto, A., Conforti, R., Dumas, M., LaRosa, M., Maggi, F.M., Marrella, A., Mecella, M., Soo, A., 2018. Automated discovery of process models from event logs: review and benchmark. IEEE Trans. Knowl. Data Eng. 31 (4), 686–705. https://doi.org/10.1109/TKDE.2018.2841877

Bauer, M., van der Aa, H., Weidlich, M., 2020. Sampling and approximation techniques for efficient process conformance checking. Inf. Syst. https://doi.org/10.1016/j.is.2020.101666

Benevento, E., Dixit, P.M., Sani, M.F., Aloini, D., van der Aalst, W.M.P., 2019. Evaluating the effectiveness of interactive process discovery in healthcare: a case study. In: Proceedings of the Business Process Management Workshops – BPM 2019 International Workshops, Vienna, Austria, 1–6 September 2019, Revised Selected Papers, vol. 362 of Lecture Notes in Business Information Processing, Springer, pp. 508–519. DOI: 10.1007/978-3-030-37453-2_41.

Bogarín, A., Cerezo, R., Romero, C., 2018. A survey on educational process mining. WIREs Data Min. Knowl. Discov. 8 (1), e1230. https://doi.org/10.1002/widm.1230

Bottrighi, A., Canensi, L., Leonardi, G., Montani, S., Terenziani, P., 2016. Trace retrieval for business process operational support. Expert Syst. Appl. 55, 212–221. https://doi.org/10.1016/j.eswa.2015.12.002

Brocke, J.v., Simons, A., Niehaves, B., Niehaves, B., Reimer, K., Plattfaut, R., Cleven, A., 2009. Reconstructing the giant: on the importance of rigour in documenting the literature search process. In: Proceedings of the ECIS 2009, AIS Electronic Library (AISeL). URL: https://aisel.aisnet.org/ecis2009/161/.

Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P., 2014. Quality dimensions in process discovery: the importance of fitness, precision, generalization and simplicity. Int. J. Coop. Inf. Syst. 23 (01), 1440001. https://doi.org/10.1142/S0218843014400012

Canensi, L., Leonardi, G., Montani, S., Terenziani, P., 2017. Multi-level interactive medical process mining. In: Artificial Intelligence in Medicine. Springer International Publishing, Cham, pp. 256–260. https://doi.org/10.1007/978-3-319-59758-4_28

Carmona, J., van Dongen, B.F., Solti, A., Weidlich, M., 2018. Conformance Checking – Relating Processes and Models. Springerhttps://doi.org/10.1007/978-3-319-99414-7

Chinosi, M., Trombetta, A., 2012. BPMN: an introduction to the standard. Comput. Stand. Interfaces 34 (1), 124–134. https://doi.org/10.1016/j.csi.2011.06.002

Desel, J., Esparza, J., 2005. Free Choice Petri Nets, No. 40. Cambridge University Presshttps://doi.org/10.1017/CBO9780511526558

Dixit, P.M., Buijs, J.C.A.M., van der Aalst, W.M.P., 2018a. Prodigy: human-in-the-loop process discovery. In: Proceedings of the 12th International Conference on Research Challenges in Information Science (RCIS), pp. 1–12. doi: 10.1109/RCIS.2018.8406657.

Dixit, P.M., Verbeek, H.M.W., Buijs, J.C.A.M., van der Aalst, W.M.P., 2018b. Interactive data-driven process model construction. In: Proceedings of the Conceptual Modeling – 37th International Conference, ER 2018, Xi'an, China, 22–25 October 2018, vol. 11157 of Lecture Notes in Computer Science, Springer, pp. 251–265. DOI: 10.1007/978-3-030-00847-5_19.

Dixit, P.M., Buijs, J.C.A.M., van der Aalst, W.M.P., Hompes, B., Buurman, H., 2015. Enhancing process mining results using domain knowledge. In: Ceravolo, P., Rinderle-Ma, S., (Eds.), Proceedings of the 5th International Symposium on Data-driven Process Discovery and Analysis (SIMPDA 2015), Vienna, Austria, 9–11 December 2015, vol. 1527 of CEUR Workshop Proceedings, CEUR-WS.org, pp. 79–94. URL: http://ceur-ws.org/Vol-1527/paper6.pdf.

Dumas, M., Rosa, M.L., Mendling, J., Reijers, H.A., 2013. Fundamentals of Business Process Management. Springerhttps://doi.org/10.1007/978-3-642-33143-5

Dunzer, S., Stierle, M., Matzner, M., Baier, S., 2019. Conformance checking: a state-of-the-art literature review. In: Proceedings of the 11th International Conference on Subject-Oriented Business Process Management, pp. 1–10. doi: 10.1145/3329007.3329014.

Fahland, D., van der Aalst, W.M.P., 2012. Repairing process models to reflect reality. In: Barros, A., Gal, A., Kindler, E. (Eds.), Business Process Management. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 229–245. https://doi.org/10.1007/978-3-642-32885-5_19

Fahland, D., van der Aalst, W.M.P., 2015. Model repair – aligning process models to reality. Inf. Syst. 47, 220–243. https://doi.org/10.1016/j.is.2013.12.007

M. Fani Sani S.J. van Zelst W.M.P. vanderAalst Conformance checking approximation using subset selection and edit distance Advanced Information Systems Engineering 2020 Springer International Publishing Cham 234 251 doi: 10.1007/978-3-030-49435-3_15].

Ferilli, S., Esposito, F., 2013. A logic framework for incremental learning of process models. Fundam. Inform. 128, 413–443. https://doi.org/10.3233/FI-2013-951

Ferilli, S., 2020. Incremental declarative process mining with WoMan. In: Proceedings of the IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS), pp. 1–8. doi: 10.1109/EAIS48028.2020http://dx.doi.org/10.1109/EAIS48028.2020.9122700.9122700.

Friedrich, F., Mendling, J., Puhlmann, F., 2011. Process model generation from natural language text. In: Advanced Information Systems Engineering. Springer Berlin Heidelberg, pp. 482–496. https://doi.org/10.1007/978-3-642-21640-4_36

Goedertier, S., Martens, D., Vanthienen, J., Baesens, B., 2009. Robust process discovery with artificial negative events. J. Mach. Learn. Res. 10, 1305–1340. https://doi.org/10.5555/1577069.1577113

Greco, G., Guzzo, A., Lupia, F., Pontieri, L., 2015. Process discovery under precedence constraints. ACM Trans. Knowl. Discov. Data 9 (4), 32:1–32:39. https://doi.org/10.1145/2710020

Greco, G., Guzzo, A., Pontieri, L., 2012. Process discovery via precedence constraints. In: Raedt, L.D., Bessiere, C., Dubois, D., Doherty, P., Frasconi, P., Heintz, F., Lucas, P.J.F. (Eds.), Proceedings of the ECAI 2012 – 20th European Conference on and Applications. Including Prestigious Applications of and Applications (PAIS-2012) System Demonstrations Track, Montpellier, France, 27–31 August 2012, vol. 242 of Frontiers in and Applications IOS Press, pp. 366–371. doi: 10.3233/978-1-61499-098-7-366.

Gschwandtner, T., Aigner, W., Miksch, S., Gärtner, J., Kriglstein, S., Pohl, M., Suchy, N., 2014. Timecleanser: a visual analytics approach for data cleansing of time-oriented data,. In: Proceedings of the 14th International Conference on Knowledge Technologies and Data-Driven Business, i-KNOW '14, Association for Computing Machinery, New York, NY, USA. doi: 10.1145/2637748.2638423.

Hammori, M., Herbst, J., Kleiner, N., 2004. Interactive workflow mining. In: Proceedings of the Business Process Management: Second International Conference, BPM 2004, Potsdam, Germany, 17–18 June 2004, vol. 3080 of Lecture Notes in Computer Science, Springer, pp. 211–226. doi: 10.1007/978-3-540-25970-1_14.

Jans, M., Alles, M., Vasarhelyi, M., 2013. The case for process mining in auditing: sources of value added and areas of application. Int. J. Account. Inf. Syst. 14 (1), 1–20. https://doi.org/10.1016/j.accinf.2012.06.015

Kerremans, M., Srivastava, T., Choudhary, F., 2021. Market Guide for Process Mining, Technical Report, Gartner.

Kindler, E., Rubin, V., Schäfer, W., 2006. Incremental workflow mining based on document versioning information. In: Unifying the Software Process Spectrum. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 287–301. https://doi.org/10.1007/11608035_25

La Rosa, M., Reijers, H.A., van der Aalst, W.M.P., Dijkman, R.M., Mendling, J., Dumas, M., García-Bañuelos, L., 2011. Apromore: an advanced process model repository. Expert Syst. Appl. 38 (6), 7029–7040. https://doi.org/10.1016/j.eswa.2010.12.012

Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P., 2013. Discovering block-structured process models from event logs - a constructive approach. In: Application and Theory of Petri Nets and Concurrency. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 311–329. https://doi.org/10.1007/978-3-642-38697-8_17

Leemans, S.J.J., 2017. Robust Process Mining with Guarantees (Ph.D. thesis). Eindhoven University of Technology. URL: https://research.tue.nl/en/publications/robust-process-mining-with-guarantees.

Lu, X., Fahland, D., van den Biggelaar, F.J.H.M., van der Aalst, W.M.P., 2016. Handling duplicated tasks in process discovery by refining event labels. In: Business Process Management. Springer International Publishing, Cham, pp. 90–107. https://doi.org/10.1007/978-3-319-45348-4_6

Maggi, F.M., Mooij, A.J., van der Aalst, W.M.P., 2011. User-guided discovery of declarative process models. In: Proceedings of the IEEE Symposium on and Data Mining, CIDM 2011, Part of the IEEE Symposium Series on 2011, 11–15 April 2011, Paris, France, IEEE, pp. 192–199. DOI: 10.1109/CIDM.2011.5949297.

Martin, N., Martinez-Millana, A., Valdivieso, B., Fernández-Llatas, C., 2019. Interactive data cleaning for process mining: a case study of an outpatient clinic's appointment system. In: DiFrancescomarino, C., Dijkman, R., Zdun, U. (Eds.), Business Process Management Workshops. Springer International Publishing, Cham, pp. 532–544. https://doi.org/10.1007/978-3-030-37453-2_43

Martin, N., Fischer, D.A., Kerpedzhiev, G.D., Goel, K., Leemans, S.J.J., Röglinger, M., van der Aalst, W.M.P., Dumas, M., La Rosa, M., Wynn, M.T., 2021. Opportunities and challenges for process mining in organizations: results of a Delphi study. Bus. Inf. Syst. Eng. 63 (5), 511–527. https://doi.org/10.1007/s12599-021-00720-0

Nickerson, R.C., Muntermann, J., Varshney, U., 2010. Taxonomy development in information systems: a literature survey and problem statement. In: Proceedings of the ECIS 2010. AIS Electronic Library (AISeL). doi: 10.1016/j.is.2016.07.011.

Park, Minjeong, Song, Minseok, Baek, Tae Hyun, Son, SookYoung, Ha, Seung Jin, Cho, Sung Woo, 2015. Workload and Delay Analysis in Manufacturing Process Using Process Mining. In: Asia Pacific Business Process Management. AP-BPM 2015. Lecture Notes in Business Information Processing Springer, Cham. https://doi.org/10.1007/978-3-319-19509-4_11

Pérez-Alfonso, D., Fundora-Ramírez, O., Lazo-Cortés, M.S., Roche-Escobar, R., 2015. Recommendation of process discovery algorithms through event log classification. In: Carrasco-Ochoa, J.A., Martínez-Trinidad, J.F., Sossa-Azuela, J.H., OlveraLópez, J.A., Famili, F. (Eds.), Pattern Recognition. Springer International Publishing, pp. 3–12. https://doi.org/10.1007/978-3-319-19264-2_1

Pesic, M., van der Aalst, W.M.P., 2006. A declarative approach for flexible business processes management. In: Proceedings of the International Conference on Business Process Management, Springer, pp. 169–180. DOI: 10.1007/11837862_18.

Reinkemeyer, L., 2020. Process Mining in Action: Principles, Use Cases and Outlook. Springer Naturehttps://doi.org/10.1007/978-3-030-40172-6

Rembert, A.J., Omokpo, A., Mazzoleni, P., Goodwin, R., 2013. Process discovery using prior knowledge. In: Proceedings of the Service-Oriented Computing – 11th International Conference, ICSOC 2013, Berlin, Germany, 2–5 December 2013, vol. 8274 of Lecture Notes in Computer Science, Springer, pp. 328–342. DOI: 10.1007/978-3-642-45005-1_23.

Ribeiro, J., Carmona, J., Mísír, M., Sebag, M., 2014. A recommender system for process discovery. In: Sadiq, S., Soffer, P., Völzer, H. (Eds.), Business Process Management. Springer International Publishing, pp. 67–83. https://doi.org/10.1007/978-3-319-10172-9_5

Rojas, E., Munoz-Gama, J., Sepúlveda, M., Capurro, D., 2016. Process mining in healthcare: a literature review. J. Biomed. Inform. 61, 224–236. https://doi.org/10.1016/j.jbi.2016.04.007

Rozinat, A., de Jong, I.S.M., Günther, C.W., der Aalst, W.M.P.v., 2009. Process mining applied to the test process of wafer scanners in ASML. IEEE Trans. Syst. Man Cybern. Part C Appl. Rev. 39 (4), 474–479. https://doi.org/10.1109/TSMCC.2009.2014169

Sadeghianasl, S., ter Hofstede, A.H.M., Suriadi, S., Turkay, S., 2020. Collaborative and interactive detection and repair of activity labels in process event logs. In: Proceedings of the 2nd International Conference on Process Mining (ICPM), pp. 41–48. DOI: 10.1109/ICPM49681.2020.00017.

Sani, M.F., Berti, A., van Zelst, S.J., van der Aalst, W.M.P., 2019. Filtering toolkit: interactively filter event logs to improve the quality of discovered models. BPM 134–138.http://ceur-ws.org/Vol-2420/paperDT4.pdf.

Schuster, D., van Zelst, S.J., van der Aalst, W.M.P., 2020. Incremental discovery of hierarchical process models. In: Research Challenges in Information Science. Springer International Publishing, Cham, pp. 417–433. https://doi.org/10.1007/978-3-030-50316-1_25

Schuster, D., van Zelst, S.J., van der Aalst, W.M.P., 2021a. Cortado–an interactive tool for data-driven process discovery and modeling. In: Application and Theory of Petri Nets and Concurrency. Springer International Publishing, Cham, pp. 465–475. https://doi.org/10.1007/978-3-030-76983-3_23

Schuster, D., van Zelst, S., van der Aalst, W.M.P., 2021b. Alignment approximation for process trees. In: Process Mining Workshops. Springer International Publishing, Cham, pp. 247–259. https://doi.org/10.1007/978-3-030-72693-5_19

Suriadi, S., Andrews, R., terHofstede, A.H.M., Wynn, M.T., 2017. Event log imperfection patterns for process mining: towards a systematic approach to cleaning event logs. Inf. Syst. 64, 132–150. https://doi.org/10.1016/j.is.2016.07.011

Tax, N., Teinemaa, I., van Zelst, S.J., 2020. An interdisciplinary comparison of sequence modeling methods for next-element prediction. Softw. Syst. Model. 19 (6), 1345–1365. https://doi.org/10.1007/s10270-020-00789-3

Thiede, M., Fuerstenau, D., BezerraBarquet, A.P., 2018. How is process mining technology used by organizations? A systematic literature review of empirical studies. Bus. Process Manag. J. 24 (4), 900–922. https://doi.org/10.1108/BPMJ-06-2017-0148

van der Aalst, W., 2010. Process discovery: capturing the invisible. IEEE Comput. Intell. Mag. 5 (1), 28–41. https://doi.org/10.1109/MCI.2009.935307

van der Aalst, W., Buijs, J., van Dongen, B., 2012a. Towards improving the representational bias of process mining. In: Data-Driven Process Discovery and Analysis. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 39–54. https://doi.org/10.1007/978-3-642-34044-4_3

van der Aalst, W., Adriansyah, A., de Medeiros, A.K.A., Arcieri, F., Baier, T., Blickle, T., Bose, J.C., van den Brand, P., Brandtjen, R., Buijs, J., Burattin, A., Carmona, J., Castellanos, M., Claes, J., Cook, J., Costantini, N., Curbera, F., Damiani, E., de Leoni, M., Delias, P., van Dongen, B.F., Dumas, M., Dustdar, S., Fahland, D., Ferreira, D.R., Gaaloul, W., van Geffen, F., Goel, S., Günther, C., Guzzo, A., Harmon, P., terHofstede, A., Hoogland, J., Ingvaldsen, J.E., Kato, K., Kuhn, R., Kumar, A., LaRosa, M., Maggi, F., Malerba, D., Mans, R.S., Manuel, A., McCreesh, M., Mello, P., Mendling, J., Montali, M., Motahari-Nezhad, H.R., zurMuehlen, M., Munoz-Gama, J., Pontieri, L., Ribeiro, J., Rozinat, A., SeguelPérez, H., terHofstede, R., Sepúlveda, M., Sinur, J., Soffer, P., Song, M., Sperduti, A., Stilo, G., Stoel, C., Swenson, K., Talamo, M., Tan, W., Turner, C., Vanthienen, J., Varvaressos, G., Verbeek, E., Verdonk, M., Vigo, R., Wang, J., Weber, B., Weidlich, M., Weijters, T., Wen, L., Westergaard, M., Wynn, M., 2012b. Process mining manifesto. In: Business Process Management Workshops. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 169–194. https://doi.org/10.1007/978-3-642-28108-2_19

van der Aalst, W.M., van Hee, K.M., van der Werf, J.M., Verdonk, M., 2010. Auditing 2.0: using process mining to support Tomorrow's auditor. Computer 43 (3), 90–93. https://doi.org/10.1109/MC.2010.61

van der Aalst, W.M.P., 1996. Structural characterizations of sound workflow nets. Comput. Sci. Rep. 96 (23), 18–22.

van der Aalst, W.M.P., 1998. The application of Petri nets to workflow management. J. Circuits Syst. Comput. 8 (01), 21–66. https://doi.org/10.1142/S0218126698000043

van der Aalst, W.M.P., 2016. Data Science in Action, second ed. Springer Berlin Heidelberghttps://doi.org/10.1007/978-3-662-49851-4_1

van der Aalst, W.M.P., 2018. Process discovery from event data: relating models and logs through abstractions, WIREs. Data Min. Knowl. Discov. 8 (3). https://doi.org/10.1002/widm.1244

van der Aalst, W.M.P., van Hee, K.M., ter Hofstede, A.H.M., Sidorova, N., Verbeek, H.M.W., Voorhoeve, M., Wynn, M.T., 2011. Soundness of workflow nets: classification, decidability, and analysis. Form. Asp. Comput. 23 (3), 333–363. https://doi.org/10.1007/s00165-010-0161-4

van der Aalst, W.M.P., 2019. A practitioner's guide to process mining: limitations of the directly-follows graph. Procedia Computer Science, cENTERIS 2019 – International Conference on ENTERprise and Technologies/ProjMAN 2019 – International Conference on Project MANagement/HCist 2019 – International Conference on Health and Social Care and Technologies, CENTERIS/ProjMAN/HCist 2019, 164, pp. 321–328. doi: 10.1016/j.procs.2019.12.189.

van Dongen, B., 2020. BPI challenge 2020, 4TU. ResearchData. https://doi.org/10.4121/uuid:52fb97d4-4588-43c9-9d04-3604d4613b51

van Dongen, B.F., Alves de Medeiros, A.K., Wen, L., 2009. Process Mining: Overview and Outlook of Petri Net Discovery Algorithms. Springer Berlin Heidelberg, pp. 225–242. https://doi.org/10.1007/978-3-642-00899-3_13

van Zelst, S.J., Mannhardt, F., de Leoni, M., Koschmider, A., 2021. Event abstraction in process mining: literature review and taxonomy. Granul. Comput. 6 (3), 719–736. https://doi.org/10.1007/s41066-020-00226-2

Verbeek, E., Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P., 2010. Prom 6: the process mining toolkit. In: Proceedings of the Business Process Management 2010 Demonstration Track, Hoboken, NJ, USA, 14–16 September 2010, vol. 615 of CEUR Workshop Proceedings, CEUR-WS.org. URL: http://ceur-ws.org/Vol-615/paper13.pdf.

Webster, J., Watson, R.T., 2002. Analyzing the past to prepare for the future: writing a literature review. MIS Q. 26 (2) (xiii-xxiii).

Yahya, B.N., Bae, H., Sul, S.-o., Wu, J.-Z., 2013. Process discovery by synthesizing activity proximity and user's domain knowledge. In: Song, M., Wynn, M.T., Liu, J. (Eds.), Asia Pacific Business Process Management. Springer International Publishing, Cham, pp. 92–105. https://doi.org/10.1007/978-3-319-02922-1_7

Yang, W., Su, Q., 2014. Process mining for clinical pathway: literature review and future directions, in: Proceedings of the 11th International Conference on Service Systems and Service Management (ICSSSM), pp. 1–5. DOI: 10.1109/ICSSSM.2014.6943412.

Yürek, I., Birant, D., Birant, K.U., 2018. Interactive process miner: a new approach for process mining. Turk. J. Electr. Eng. Comput. Sci. 26 (3), 1314–1328. https://doi.org/10.3906/elk-1708-112

Daniel Schuster is a scientist at the Data Science & Artificial Intelligence department of the Fraunhofer Institute for Applied Information Technology FIT. In addition, he is a Ph.D. candidate under the supervision of Wil M. P. van der Aalst at the Chair for Process and Data Science of the RWTH Aachen University. His research interests include the field of process mining. He is particularly interested in interactive process discovery, i.e., incorporating human intelligence into data-driven process discovery.

Sebastiaan J. van Zelst is a senior scientist in process mining affiliated to the Fraunhofer Institute for Applied Information Technology and the RWTH Aachen University. His research interests include event data processing, real-time process monitoring, data-driven process optimization, and the application of hybrid intelligence in process mining. He has co-authored over 35 peer-reviewed conference contributions and over 10 peer-reviewed journal contributions in process mining. Personal website: https://sebastiaanvanzelst.com.

Wil M.P. van der Aalst is a full professor at RWTH Aachen University, leading the Process and Data Science (PADS) group. He is also the Chief Scientist at Celonis, part-time affiliated with the Fraunhofer FIT, and a member of the Board of Governors of Tilburg University. He also has unpaid professorship positions at Queensland University of Technology (since 2003) and the Technische Universiteit Eindhoven (TU/e). Currently, he is also a distinguished fellow of Fondazione Bruno Kessler (FBK) in Trento, deputy CEO of the Internet of Production (IoP) Cluster of Excellence, co-director of the RWTH Center for Artificial Intelligence. His research interests include process mining, Petri nets, business process management, workflow management, process modeling, and process analysis. Wil van der Aalst has published over 250 journal papers, 22 books (as author or editor), 550 refereed conference/workshop publications, and 80 book chapters. Personal website: http://www.padsweb.rwth-aachen.de/wvdaalst/.