Software Tools for Technology Transfer manuscript No.
(will be inserted by the editor)

Scientific Workflows for Process Mining: Building Blocks, Scenarios, and Implementation

Alfredo Bolt, Massimiliano de Leoni, Wil M.P. van der Aalst

Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands

Received: date / Revised version: date

Abstract. Over the last decade process mining emerged as a new analytical discipline able to answer a variety of questions based on event data. Event logs have a very particular structure; events have timestamps, refer to activities and resources, and need to be correlated to form process instances. Process mining results tend to be very different from classical data mining results, e.g., process discovery may yield end-to-end process models capturing different perspectives rather than decision trees or frequent patterns. A process-mining tool like ProM provides hundreds of different process mining techniques ranging from discovery and conformance checking to filtering and prediction. Typically, a combination of techniques is needed and, for every step, there are different techniques that may be very sensitive to parameter settings. Moreover, event logs may be huge and may need to be decomposed and distributed for analysis. These aspects make it very cumbersome to analyze event logs manually. Process mining should be repeatable and automated. Therefore, we propose a framework to support the analysis of process mining workflows. Existing scientific workflow systems and data mining tools are not tailored towards process mining and the artifacts used for analysis (process models and event logs). This paper structures the basic building blocks needed for process mining and describes various analysis scenarios. Based on these requirements we implemented *RapidProM*, a tool supporting scientific workflows for process mining. Examples illustrating the different scenarios are provided to show the feasibility of the approach.

1 Introduction

Scientific Workflow Management (SWFM) systems help users to design, compose, execute, archive, and share workflows that represent some type of analysis or experiment. Scientific workflows are often represented as directed graphs where the nodes represent "work" and the edges represent paths along which data and results can flow between nodes. Next to "classical" SWFM systems such as Taverna [18], Kepler [27], Galaxy [15], ClowdFlows [22], and jABC [33], one can also see the uptake of integrated environments for data mining, predictive analytics, business analytics, machine learning, text mining, reporting, etc. Notable examples are RapidMiner [17] and KNIME [4]. These can be viewed as SWFM systems tailored towards the needs of data scientists.

Traditional data-driven analysis techniques do not consider end-to-end processes. People are process models by hand (e.g., Petri nets, UML activity diagrams, or BPMN models), but this modeled behavior is seldom aligned with real-life event data. *Process mining* aims to bridge this gap by connecting end-to-end process models to the raw events that have been recorded.

Process-mining techniques enable the analysis of a wide variety of processes using event data. For example, event logs can be used to automatically learn a process model (e.g., a Petri net or BPMN model). Next to the automated discovery of the real underlying process, there are process-mining techniques to analyze bottlenecks, to uncover hidden inefficiencies, to check compliance, to explain deviations, to predict performance, and to guide users towards "better" processes. Hundreds of process-mining techniques are available and their value has been proven in many case studies. See for example the twenty *case studies* on the webpage of the IEEE Task Force on Process Mining [19]. The open source process mining framework *ProM* provides hundreds of plug-ins and has been downloaded over 100.000 times. The growing number of commercial *process mining tools* (Disco, Perceptive Process Mining, Celonis Process Mining, QPR ProcessAnalyzer, Software AG/ARIS PPM, Fujitsu In-

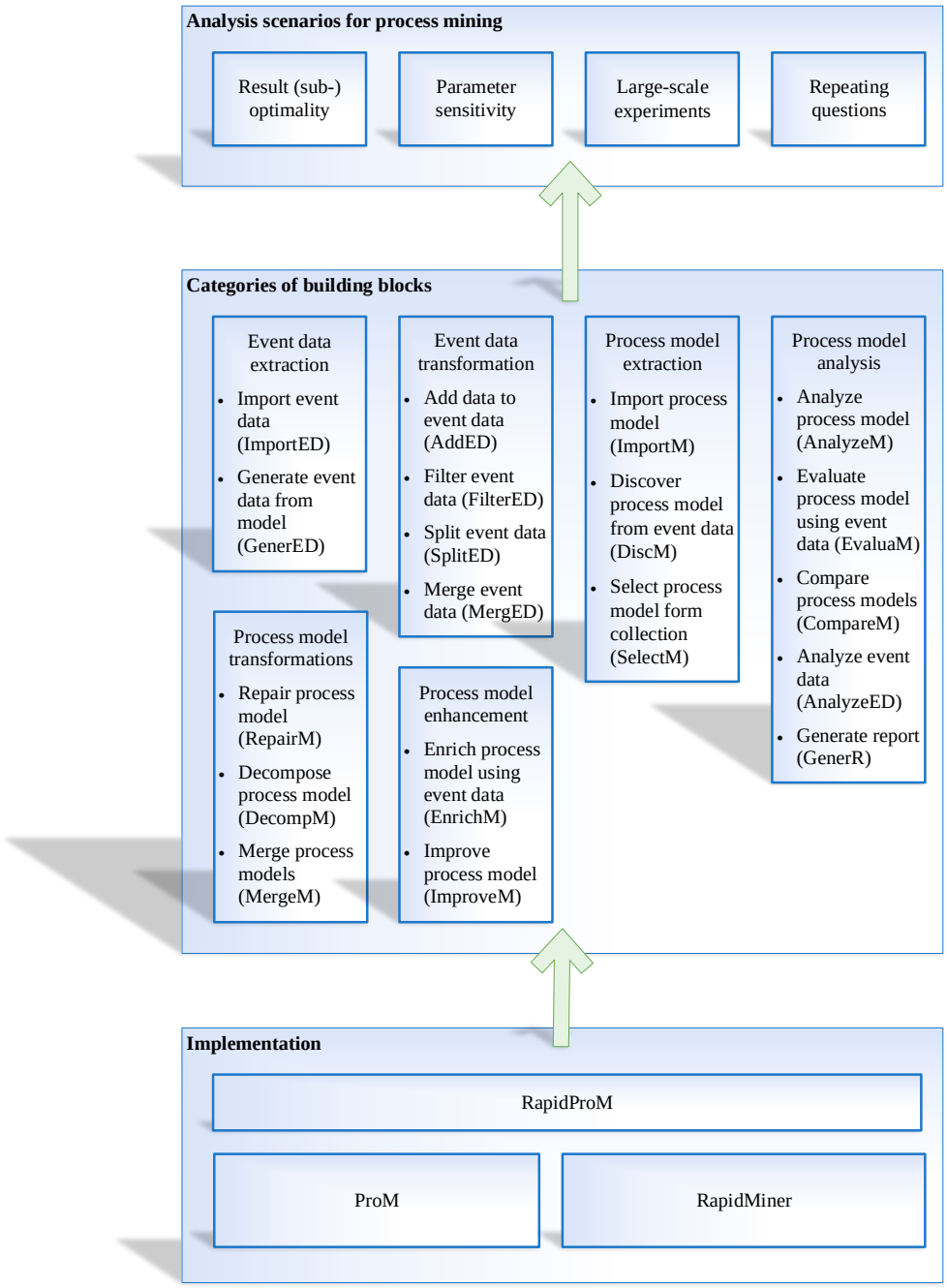


Fig. 1: Overview of the framework to support process mining workflows

terstage Automated Process Discovery, etc.) further illustrates the uptake of process mining.

For process mining typically many analysis steps need to be chained together. Existing process mining tools do not support such analysis workflows. As a result, analysis may be tedious and it is easy to make errors. Repeatability and provenance are jeopardized by manually executing more involved process mining workflows.

This paper is motivated by the observation that tool support for process mining workflows is missing. None of the process mining tools (ProM, Disco, Perceptive, Celonis, QPR, etc.) provides a facility to design and execute analysis workflows. None of the scientific workflow management systems including analytics suites like RapidMiner and KNIME support process mining. Yet, process models and event logs are very different from the artifacts typically considered. Therefore, we propose the *framework to support process mining workflows* depicted in Figure 1.

This paper considers four *analysis scenarios* where process mining workflows are essential:

- *Result (sub-)optimality*: Often different process mining techniques can be applied and upfront it is not clear which one is most suitable. By modeling the analysis workflow, one can just perform all candidate techniques on the data, evaluate the different analysis results, and pick the result with the highest quality (e.g., the process model best describing the observed behavior).
- *Parameter sensitivity*: Different parameter settings and alternative ways of filtering can have unexpected effects. Therefore, it is important to see how sensitive the results are (e.g., leaving out some data or changing a parameter setting a bit should not change the results dramatically). It is important to not simply show the analysis result without having some confidence indications.
- *Large-scale experiments*: Each year new process mining techniques become available and larger data sets need to be tackled. For example, novel discovery techniques need to be evaluated through massive testing and larger event logs need to be decomposed to make analysis feasible. Without automated workflow support, these experiments are tedious, error-prone and time consuming.
- *Repeating questions*: It is important to lower the threshold for process mining to let non-expert users approach it. Questions are often repetitive, e.g., the same analysis is done for a different period or a different group of cases. Process mining workflows facilitate recurring forms of analysis.

As shown in Figure 1 these scenarios build on process mining *building blocks* grouped into six categories:

- *Event data extraction*: Building blocks to extract data from systems or to create synthetic data.
- *Event data transformation*: Building blocks to preprocess data (e.g., splitting, merging, filtering, and enriching) before analysis.
- *Process model extraction*: Building blocks to obtain process models, e.g., through discovery or selection.
- *Process model analysis*: Building blocks to evaluate models, e.g., to check the internal consistency or to check conformance with respect to an event log.
- *Process model transformations*: Building blocks to repair, merge or decompose process models.
- *Process model enhancement*: Building blocks to enrich event logs with additional perspectives or to suggest process improvements.

Building blocks can be chained together to support specific analysis scenarios. The suggested approach has been implemented thereby building on the process mining framework *ProM* and the workflow and data mining capabilities of *RapidMiner*. The resulting tool is called *RapidProM* which supports process mining workflows. ProM was selected because it is open source and there is no other tool that supports as many process mining building blocks. RapidMiner was selected because it allows for extensions that can be offered through a marketplace. RapidProM is also offered as such an extension and the infrastructure allows us to mix process mining with traditional data mining approaches, text mining, reporting, and machine learning. Overall, RapidProM offers compressive support for any type of analysis involving event data and processes.

The remainder of this paper is organized as follows. Section 2 discusses related work and positions our framework. An initial set of process-mining building blocks is described in Section 3. These building blocks support the four analysis scenarios described in Section 4. The RapidProM implementation is presented in Section 5. Section 6 evaluates the approach by showing concrete examples. Finally, Section 7 concludes the paper.

2 Related Work

Over the last decade, *process mining* emerged as a new scientific discipline on the interface between process models and event data [36]. Conventional Business Process Management (BPM) [37,51] and Workflow Management (WfM) [25,41] approaches and tools are mostly model-driven with little consideration for event data. Data Mining (DM) [16], Business Intelligence (BI), and Machine Learning (ML) [29] focus on data without considering end-to-end process models. Process mining aims to bridge the gap between BPM and WfM on the one hand and DM, BI, and ML on the other hand. A wealth of *process discovery* [24,43,50] and *conformance checking* [1, 2,39] techniques has become available. For example, the process mining framework ProM [47] provides hundreds of plug-ins supporting different types of process mining (`www.processmining.org`).

This paper takes a *different* perspective on the gap between analytics and BPM/WfM. We propose to use workflow technology for process mining rather than the other way around. To this end, we focus on particular kinds of scientific workflows composed of process mining operators.

Differences between scientific and business workflows have been discussed in several papers [3]. Despite unification attempts (e.g., [31]) both domains have remained quite disparate due to differences in functional requirements, selected priorities, and disjoint communities.

Obviously, the work reported in this paper is closer to scientific workflows than business workflows (i.e., traditional BPM/WFM from the business domain). Numerous *Scientific Workflow Management (SWFM) systems* have been developed. Examples include Taverna [18], Kepler [27], Galaxy [15], ClowdFlows [22], jABC [33], Vistrails, Pegasus, Swift, e-BioFlow, VIEW, and many others. Some of the SWFM systems (e.g., Kepler and Galaxy) also provide repositories of models. The website `myExperiment.org` lists over 3500 workflows shared

by its members [14]. The diversity of the different approaches illustrates that the field is evolving in many different ways. We refer to the book [34] for an extensive introduction to SFWM.

An approach to mine process models for scientific workflows (including data and control dependencies) was presented in [53]. This approach uses "process mining for scientific workflows" rather than applying scientific workflow technology to process mining. The results in [53] can be used to recommend scientific workflow compositions based on actual usage. To our knowledge, *RapidProM* is the only approach supporting "scientific workflows for process mining". The demo paper [28] reported on the first implementation. In the meantime, *RapidProM* has been refactored based on various practical experiences.

There are many approaches that aim to analyze repositories of scientific workflows. In [52], the authors provide an extensible process library for analyzing jABC workflows empirically. In [12] graph clustering is used to discover subworkflows from a repository of workflows. Other analysis approaches include [13], [26], and [49].

Scientific workflows have been developed and adopted in various disciplines, including physics, astronomy, bioinformatics, neuroscience, earth science, economics, health, and social sciences. Various collections of reusable workflows have been proposed for all of these disciplines. For example, in [35] the authors describe workflows for quantitative data analysis in the social sciences.

The boundary between data analytics tools and scientific workflow management systems is not well-defined. Tools like RapidMiner [17] and KNIME [4] provide graphical workflow modeling and execution capabilities. Even the scripting in R [20] can be viewed as primitive workflow support. In this paper we build on RapidMiner as it allows us to mix process mining with data mining and other types of analytics (e.g., text mining). Earlier we developed extensions of ProM for chaining process mining plug-ins together, but these were merely prototypes. We also realized a prototype using and integration between KNIME and ProM. However, for reasons of usability, we opted for RapidMiner as a platform to expose process mining capabilities.

3 Definition of the process-mining building blocks

To create scientific workflows for process mining we need to define the building blocks, which are, then, connected with each other to create meaningful analysis scenarios. This section discusses a taxonomy and a repertoire of such building blocks inspired by the so-called "BPM use cases", which were presented in [37]. The use cases structure the BPM discipline and to provide a generic way of describing the usage of BPM techniques. The BPM use cases are characterized by two main aspects. Firstly, they are *abstract* as they are not linked to any specific technique or algorithm. Secondly, they represent logical units of work, i.e. they cannot be conceptually split while maintaining their generality. This does not imply that concrete techniques that implement BPM use cases cannot be composed by micro-steps, according to the implementation and design that was used.

Similarly, each process-mining building block for creating process-mining workflows represents a logical unit of work. The building blocks are conceptual in the sense that they are independent of the specific implementation and represent atomic operations. The process-mining building blocks can be chained, thus producing process-mining scientific workflows to answer a variety of process-mining questions.

Each process-mining building block takes a number of inputs and produces certain outputs. The inputs elements represent the set (or sets) of abstract objects required to perform the operation. The process-mining building block component represents the logical unit of work needed to process the inputs and produce the outputs. Inputs and outputs are indicated through circles whereas a process-mining building block is represented by a rectangle. Arcs are used to connect the blocks to the inputs and outputs.

Two process-mining building blocks a and b are chained if one or more outputs of a are used as an inputs in b. As mentioned, inputs and outputs are depicted by circles. The letter inside a circle specifies the type of the input or output. The following types of inputs and outputs are considered in this paper:

- *Process models*, which are a representation of the behavior of a process, are represented by letter *"M"*. Here we abstract from the notation used, e.g., Petri nets, Heuristics nest, BPMN models are concrete representation languages.
- *Event data sets*, which contain the recording of the execution of process instances within the information system(s), regardless of the format. They are represented by letter *"E"*. MXML and XES are standard formats to store events.
- *Information systems*, which supports the performance of processes at runtime. They are represented by the label *"S"*. Information systems may generate events used for analysis and process mining results (e.g., prediction) may influence the information system.
- *Sets of parameters* to configure the application of process-mining building blocks (e.g., thresholds, weights, ratios, etc.). They are represented by letter *"P"*.
- *Results* that are generated as outputs of a process-mining building blocks. This can be as simple as a number or more complex structures like a detailed report. In principle, the types enumerated above in this list (e.g., process models) can also be results. However, it is worth differentiating those specific types of outputs from results which are not process mining

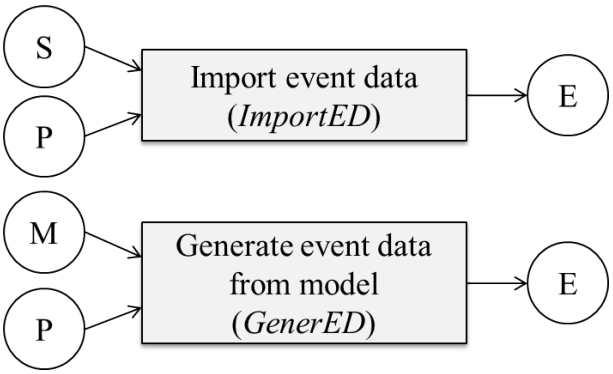


Fig. 2: Process-mining building blocks related to event data extraction.

specific (like a bar chart). Results are represented by letter *"R"*.
– *Additional Data Sets* that can be used as input for certain process-mining building blocks. These are represented by the letter *"D"*. Such an additional data set can be used to complement event data with context information (e.g., one can use weather or stock-market data to augment the event log with additional data).

The remainder of this section provides a taxonomy of process-mining building blocks grouped in six different categories. For each category, several building blocks are provided. They were selected because of their usefulness for the definition of many process-mining scientific workflows. The taxonomy is not intended to be exhaustive, there will be new process-mining building blocks as the discipline evolves.

3.1 Event data extraction

Event data are the *cornerstone* of process mining. In order to be used for analysis, event data has to be extracted and made available. All of the process-mining building blocks of this category can extract event data from different sources. Figure 2 shows some process-mining building blocks that belong to this category.

*Import event data (*ImportED*).* Information systems store event data in different format and media, from files in a hard drive to databases in the cloud. This building block represents the functionality of extracting event data from any of these sources. Some parameters can be set to drive the event-data extraction. For example, event data can be extracted from files in standard formats, such as XES[1], or from transactional databases.

[1] XES (Extensible Event Stream) is an XML-based standard for event logs `http://www.xes-standard.org`. It provides a standard format for the interchange of event log data between tools and application domains.

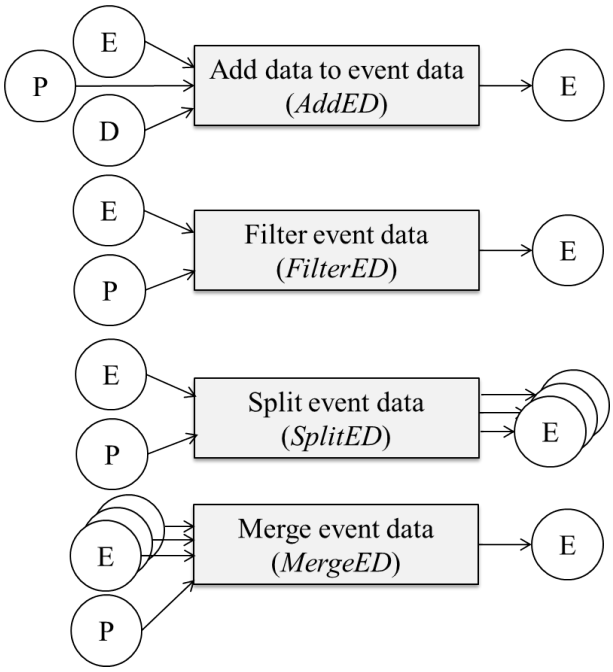


Fig. 3: Process-mining building blocks related to event data transformations

*Generate event data from model (*GenerED*).* In a number of cases, one wants to assess whether a certain technique returns the expected or desired output (i.e., synthetic event data). For this assessment, controlled experiments are necessary where input data is generated in a way that the expected output of the technique is clearly known. Given a process model M, this building block represents the functionality of generating event data that record the possible execution of instances of M. This is an important function for, e.g., testing a new discovery technique. Various simulators have been developed to support the generation of event data.

3.2 Event data transformation

Sometimes, event data sets are not sufficiently rich to enable certain process-mining analyses. In addition, certain data-set portions should be excluded, because they are are irrelevant, out of the scope of the analysis or, even, noise. Therefore, a number of event data transformations may be required before doing further analysis. This category comprises the building blocks to provide functionalities to perform the necessary event data transformations. Figure 3 the repertoire of process-mining building blocks that belong to this category.

*Add data to event data (*AddED*).* In order to perform a certain analysis or to improve the results, the event data can be augmented with additional data coming from different sources. For instance, if the process involves citizens, the event data can be augmented with data from

the municipality data source. If the level of performance of a process is suspected to be influenced by the weather, event data can incorporate weather data coming from a system storing such a kind of data. If the event data contains a ZIP code, then other data fields such as country or city can be added to the event data from external data sources. This building block represents the functionality of augmenting event data using external data, represented as a generic data set in the figure.

*Filter event data (*FilterED*).* Several reasons may exist to filter out part of the event data. For instance, the process behavior may exhibit *concept drifts* over time. In those situations, the analysis needs to focus on certain parts of the event data instead of all of it. One could filter the event data and use only those events that occurred, e.g., in year 2015. As a second example, the same process may run at different geographical locations. One may want to restrict the scope of the analysis to a specific location by filtering out the event data referring to different locations. This motivates the importance of being able to filter event data in various ways.

*Split event data (*SplitED*).* Sometimes, the organization generating the event data is interested in comparing the process' performances for different customers, offices, divisions, involved employees, etc. To perform such comparison, the event data needs to be split according to a certain criterion, e.g., according to organizational structures, and the analysis needs to be iterated over each portion of the event data. Finally, the results can be compared to highlight difference. Alternatively, the splitting of the data may be motivated by the size of the data. It may be intractable to analyze all data without decomposition or distribution. Many process-mining techniques are exponential in the number of different activities and linear in the size of the event log. If data is split in a proper way, the results of applying the techniques to the different portions can be fused into a single result. For instance, work [38] discusses how to split event data while preserving the correctness of results. This building block represents the functionality of splitting event data into overlapping or non-overlapping portions.

*Merge event data (*MergED*).* This process-mining building block is the dual of the previous: data sets from different information systems are merged into a single event data set. This process-mining building block can also tackle the typical problems of data fusion, such as redundancy and inconsistency.

3.3 Process model extraction

Process mining revolves around process models to represent the behavior of a process. This category is concerned with providing building blocks to mine a process model from event data as well as to select or extract it from a process-model collection. Figure 4 lists a number of process-mining building blocks belonging to this category.

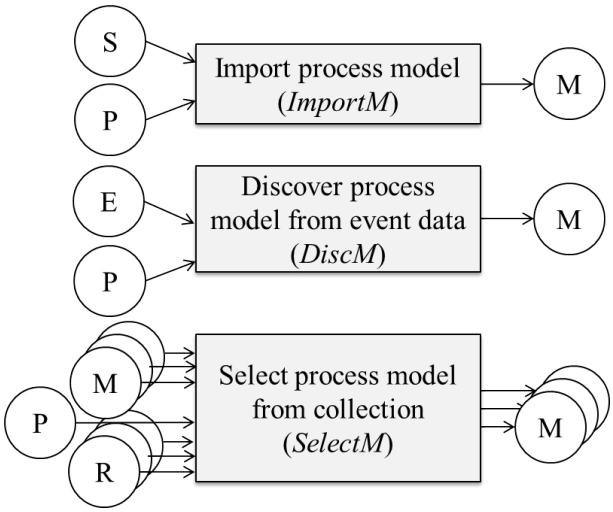


Fig. 4: Process-mining building blocks related to process model extraction

*Import process model (*ImportM*).* Process models can be stored in some media for later retrieval to conduct some analyses. This building block represents the functionality of loading a process model from some repository.

*Discover process model from event data (*DiscM*).* Process models can be manually *designed* to provide a normative definition for a process. These models are usually intuitive and understandable, but they might not describe accurately what happens in reality. Event data represent the "real behavior" of the process. Discovery techniques can be used to mine a process model on the basis of the behavior observed in the event data (cf. [36]). Here, we stay independent of the specific notations and algorithms. Examples of algorithms are the Alpha Miner [43], the Heuristics Miner [50] or, more recent techniques like the Inductive Miner [24]. This building block represents the functionality of discovering a process model from event data. This block, as many others, can receive a set of parameters as an input to customize the application of the algorithms.

*Select process model from collection (*SelectM*).* Organizations can be viewed as a collection of processes and resources that are interconnected and form a *process ecosystem*. This collection of processes can be managed and supported by different approaches, such as ARIS [30] or Apromore [23]. To conduct certain analyses, one needs to use some of these models and not the whole collection. In addition, one can give a criterion to retrieve a subset of the collection. This building block represents

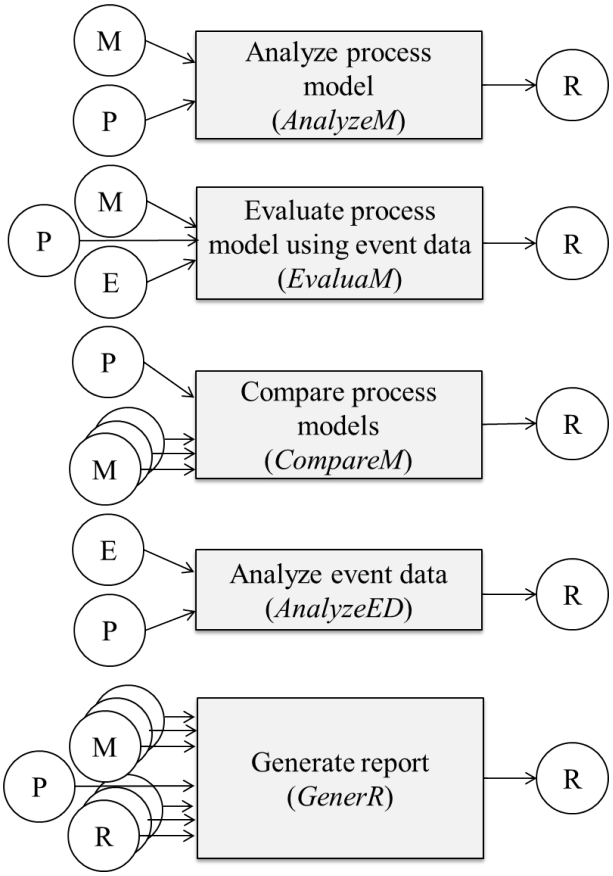


Fig. 5: Process-mining building blocks related to process model analysis

the functionality of selecting one or more process models from a process-model collection.

3.4 Process model analysis

Organizations normally use process models for the discussion, configuration and implementation of processes. In recent years, many process mining techniques are also using process models for analysis. This category groups process-mining building blocks that can analyze process models and provide analysis results. Figure 5 shows some process-mining building blocks that belong to this category.

*Analyze process model (*AnalyzeM*).* Process models may contain a number of structural problems. For instance, the model may exhibit undesired deadlocks, activities that are never enabled for execution, variables that used to drive decisions without previously taking on a value, etc. Several techniques have been designed to verify the soundness of process models against deadlocks and other problems [42]. This building block refers to design-time properties: the process model is analyzed without considering how the process instances are actually being executed. The checking of the conformance of the process model against real event data is covered by the next building block (*EvaluaM*). Undesired design-time properties happen for models designed by hand but also for models automatically mined from event data. Indeed, several discovery techniques do not guarantee to mine process models without structural problems. This building block provides functionalities for analyzing process models and detecting structural problems.

*Evaluate process model using event data (*EvaluaM*).* Besides structural analysis, process models can also be analyzed against event data. Compared with the previous building block (*AnalyzeM*), this block is not concerned with a design-time analysis. Conversely, it makes a-posteriori analysis where the adherence of the process model is checked with respect to the event data, namely how the process has actually been executed. In this way, the expected or normative behavior as represented by the process model is checked against the actual behavior as recorded in event data. In literature, this is referred to as *conformance checking* (cf. [36]). This can be used, for example, in fraud or anomaly detection. Replaying event data on process models has many possible uses: Aligning observed behavior with modeled behavior is key in many applications. For example, after aligning event data and model, one can use the *time* and *resource* information contained in the log for performance analysis. This can be used for *bottleneck* identification or to gather information for simulation analysis or predictive techniques. This building block represents the functionality of analyzing or evaluating process models using event data.

*Compare process models (*CompareM*).* Processes are not static as they dynamically evolve and adapt to the business context and requirements. For example, processes can behave differently over different years, or at different locations. Such differences or similarities can be captured through the comparison of the corresponding process models. For example, the degree of similarity can be calculated. Approaches that explicitly represent configuration or variation points [40] directly benefit from such comparisons. Building block *CompareM* is often used in combination with *SplitED* that splits the event data in sublogs and *DiscM* that discovers a model per sublog.

*Analyze event data (*AnalyzeED*).* Instead of directly creating a process model from event data, one can also first inspect the data and look at basic statistics. Moreover, it often helps to simply visualize the data. For example, one can create a so-called dotted chart [36] exploiting the temporal dimension of event data. Every event is plotted in a two dimensional space where one dimension represents the time (absolute or relative) and the other dimension may be based on the case, resource, activity or any other property of the event. The color of the dot can be used as a third dimension. See [21]

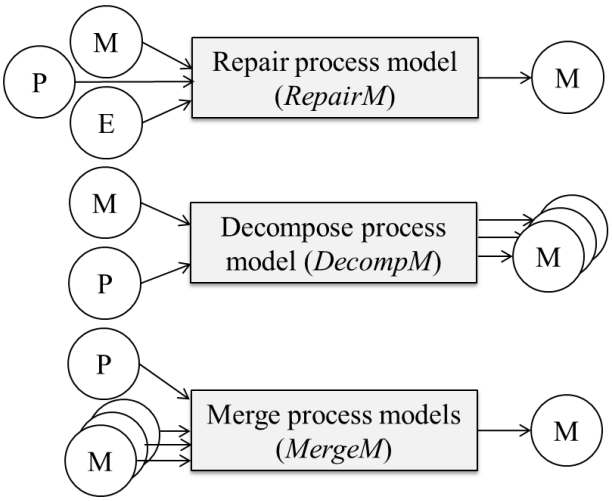


Fig. 6: Process-mining building blocks related to process model transformations

for other approaches combining visualization with other analytical techniques.

*Generate report (*GenerR*).* To consolidate process models and other results, one may create a structured report. The goal is not to create new analysis results, but to present the findings in an understandable and predictable manner. Generating standard reports helps to reduce the cognitive load and helps users to focus on the things that matter most.

3.5 Process model transformations

Process models can be designed or, alternatively, discovered from event data. Sometimes, these models need to be adjusted for follow-up analyses. This category groups process-mining building blocks that provide functionality to change the structure of a process model. Figure 6 shows some process-mining building blocks that belong to this category.

*Repair process model (*RepairM*).* Process models may need be repaired in case of consistency or conformance problems. Repairing can be regarded from two perspectives: repairing structural problems and repairing behavioral problems. The first case is related to the fact that models can contain undesired design-time properties such as deadlocks and livelocks (see also the *Analyze process model* building block discussed in Section 3.4). Repairing involves modifying the model to avoid those properties. Techniques for repairing behavioral problems focus on models that are structurally sound but that allow for undesired behavior or behavior that does not reflect reality. See also the *Evaluate process model using event data* building block discussed in Section 3.4, which is concerned with discovering the conformance problems.

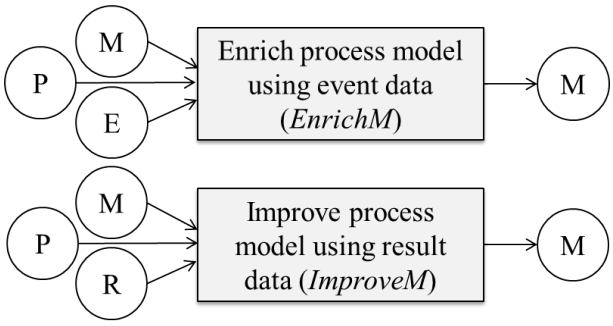


Fig. 7: Process-mining building blocks related to process model enhancement

This building block provides functionality for both types of repairing.

*Decompose process model (*DecompM*).* Processes running within organizations may be extremely large, in terms of activities, resources, data variables, etc. As mentioned, many techniques are exponential in the number of activities. The computation may be improved by splitting the models into fragments, analogously to what mentioned for splitting the event log. If the model is split according to certain criteria, the results can be somehow amalgamated and, hence, be meaningful for the entire model seen as a whole. For instance, the work on decomposed conformance checking [38] discusses how to split process model to make process mining possible with models with hundreds of elements (such as activities, resources, data variables), while preserving the correctness certain results (e.g., the fraction of deviating cases does not change because of decomposition). This block provides functionalities for splitting process models into smaller fragments.

*Merge process models (*MergeM*).* Process models may also be created from the intersection (i.e. the common behavior) or union of other models. This building block provides functionalities for merging process models into a single process model. When process discovery is decomposed, the resulting models need to be merged into a single model.

3.6 Process model enhancement

Process models just describing the control-flow are usually not the final result of process mining analysis. Process models can be enriched or improved using additional data in order to provide better insights about the real process behavior that it represents. This category groups process-mining building blocks that are used to enhance process models. Figure 7 shows a summary of the process-mining building blocks that belong to this category.

*Enrich process model using event data (*EnrichM*).* The backbone of any process models contains basic structural information relating to control-flow. However, the backbone can be enriched with additional perspectives derived from event data in to obtain better analysis results. For example, event frequency can be annotated in a process model in order to identify the most common paths followed by process instances. Timing information can also be used to enrich a process model in order to highlight bottlenecks or long waiting times. This enrichment does not have an effect on the structure of the process model. This building block represents the functionality of enriching process models with additional information contained in event data.

*Improve process model (*ImproveM*).* Besides being enriched with data, process models can also be improved. For example, performance data can be used to suggest structural modifications in order to improve the overall process performance. It is possible to automatically improve models using causal dependencies and observed performance. The impact of such modifications could be simulated in "what-if scenarios" using performance data obtained in previous steps. This building block represents the functionality of improving process models using data from other analysis results.

4 Analysis scenarios for process mining

This section reports generic analysis scenarios that are not domain-specific and, hence, that can be applied to different contexts. The analysis scenarios compose the basic process-mining building blocks and, hence, they remain independent of any specific operationalization of a technique. In fact, as mentioned before, the building blocks may employ different concrete techniques, e.g., there are dozens of process discovery techniques realizing instances of building block *DiscM* (Figure 4).

As depicted in Figure 1, we consider four analysis scenarios: (a) *result (sub-)optimality*, (b) *parameter sensitivity*, (c) *large-scale experiments*, and (d) *repeating questions*. These are described in the remainder of this section.

As discussed in this section and validated in Section 6, the same results could also be achieved without using scientific workflows. However, the results would require a tedious and error-prone work of repeating the same steps *ad nauseam*.

4.1 Result (sub-)optimality

This subsection discusses how process-mining building blocks can be used to mine optimal process model according to some optimality criteria. Often, in process discovery, optimality is difficult (or even impossible) to achieve. Often sub-optimal results are returned and it is no known what is "optimal".

Consider for example the process discovery task. The quality of a discovered process model is generally defined by four quality metrics [1, 2, 36, 39]:

- **Replay fitness** quantifies the ability of the process model to reproduce the execution of process instances as recorded in event data.
- **Simplicity** captures the degree of complexity of a process model, in terms of the numbers of activities, arcs, variables, gateways, etc.
- **Precision** quantifies the degree with which the model allows for too much behavior compared to what was observed in the event data.
- **Generalization** quantifies the degree with which the process model is capable to reproduce behavior that is not observed in the event data but that potentially should be allowed. This is linked to the fact that event data often are incomplete in the sense that only a fraction of the possible behaviors can be observed.

Traditionally, these values are normalized between 0 and 1, where 1 indicates the highest score and 0 the lowest.

The model of the highest value within a collection of (discovered) models is such that it can mediate among those criteria at best. Often, these criteria are in competing: higher score for one criterion may lower the score of a second criterion. For instance, in order to have a more precise model, it is necessary to sacrifice the behavior observed in the event data that is less frequent, partly hampering the replay-fitness score.

Later in this paper we will use an scoring criterion that is the geometric average of replay fitness and precision. This is merely an example to illustrate this analysis scenario. The geometric average of replay fitness and precision seems to be better than the arithmetic average since it is necessary to have a strong penalty if one of the criteria is low.

Figure 8 shows a suitable scientific workflow for mining a process model from event data that is sub-optimal with respect to the geometric average of fitness and precision. The optimization is done by finding the parameters that returns a sub-optimal model.

Event data is loaded from an information system and used n times as input for a discovery technique using different parameter values. The n resulting process models are evaluated using the original event data and the model that scores higher in the geometric average is returned. Please note that the result is likely to be sub-optimal: n arbitrary parameter values are chosen out of a much larger set of possibilities. If n is sufficiently large, the result is sufficiently close to the optimal. This scientific workflow is still independent of the specific algorithm used for discovery; as such, the parameter settings are also generic.

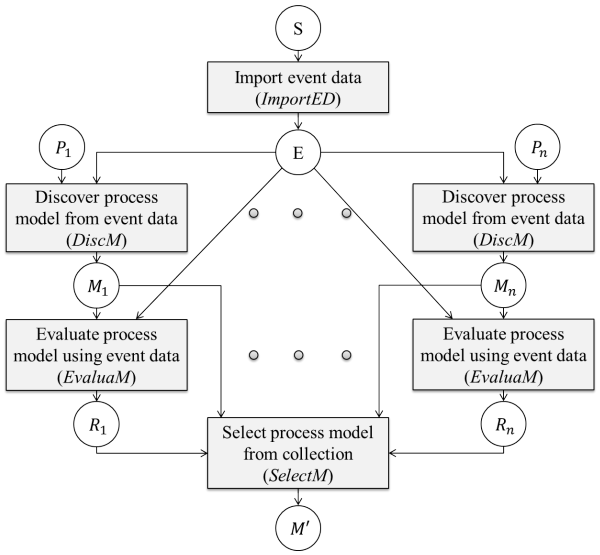
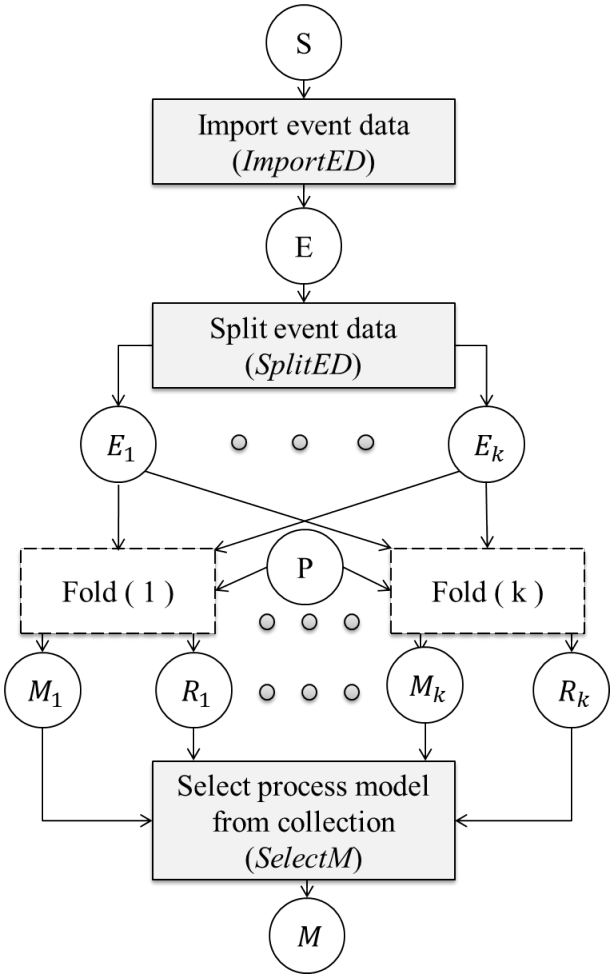


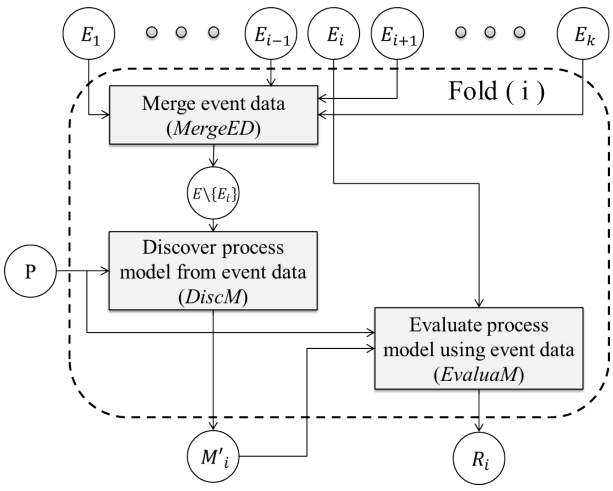
Fig. 8: Result (sub-)optimality in process model discovery: process-mining scientific workflow for mining an optimal model in terms of geometric average of replay fitness and precision

Figure 9a illustrates a scientific workflow that tries to account for generalization. For this purpose, a *k-fold cross validation* approach is used. In this approach, the process instances recorded in the event data are randomly split into k folds, through building block *Split event data* (SplitED). In each of the k times, a different fold is taken aside: the other $k-1$ folds are used for discovery and the "elected" fold is used for evaluation through conformance checking. This corresponds to block $Fold(i)$ with $1 \leq i \leq k$. Finally, through the process-mining building block *Select process model from collection* (SelectM), the model with the best geometric average is returned as output. Figure 9b enters inside the block $Fold(i)$ showing how fold E_i is used for evaluation and folds $E_1, \ldots, E_{i-1}, E_{i+1}, E_n$ are used for discovery (after being merged).

Scientific workflows can also be hierarchically defined: in turn, the discover process-mining building block (DiscM) in Figure 8 can be an entire scientific sub-workflow. The two scientific workflows shown in Figures 8 and 9 do not exclude each other. Process-mining building block *Discover process model from event data* (DiscM) can be replaced by the entire workflow in Figure 9a, thus including some generalization aspects in the search for a sub-optimal process model.



(a) Main workflow



(b) Process-mining sub-workflow for macro-block $Fold(i)$

Fig. 9: Process-mining main scientific workflow based on k-fold cross validation

4.2 Parameter sensitivity

Parameters are used by techniques to customize their behavior, e.g., adapting to the noise level in the event log. These parameters have different ways of affecting the results produced, depending on the specific implemen-

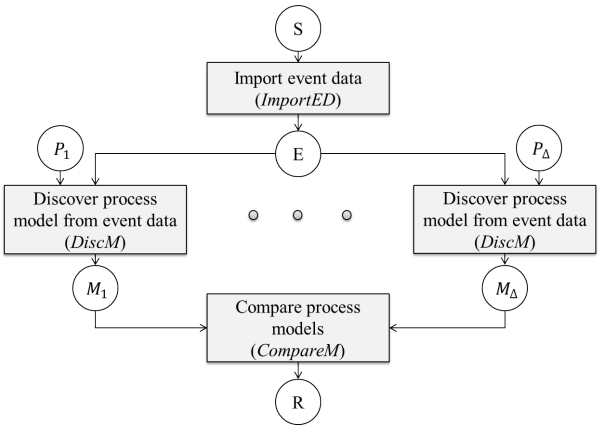


Fig. 10: Parameter sensitivity in process discovery techniques: process mining workflow for comparing the effects of different parameter values for a given discovery technique.

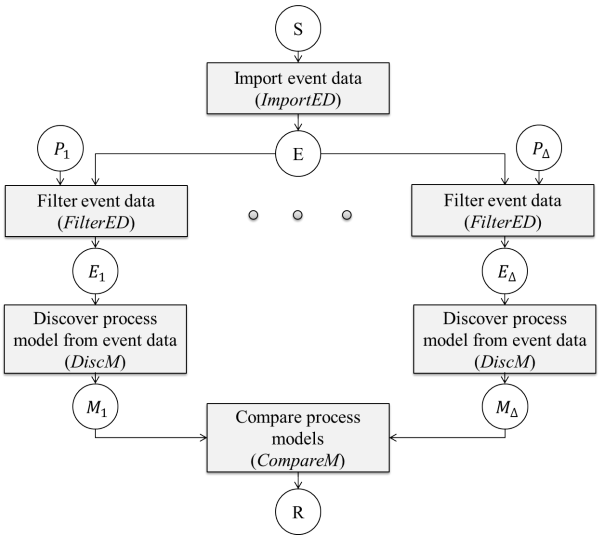


Fig. 11: Parameter sensitivity in event data filtering: process-mining scientific workflow for comparing the effect of different event-data filtering configurations on the discovered model.

tation of the technique or algorithm. Some parameters can have more *relevance* than others (i.e., they have a more substantial effect on the results). There are many ways to evaluate the sensitivity of a certain parameter for a given algorithm. Figure 10 shows an example of this analysis scenario. Now the parameter value is varied across the range. For each of the discovered models, the average of precision and generalization is computed. The results are finally plotted on a Cartesian coordinate system where the X-axis is associated with the potential parameter's values and the Y-axis is associated with the geometric average of precision and generalization.

Alternatively, the sensitivity analysis can also focus on the filtering part, while keeping the same configuration of parameter(s) for discovery. In other words, we can study how the discovered model is affected by different filtering, namely different values of the parameter(s) that customize the application of filtering.

Fig. 11 shows an example of this analysis scenario in the process mining domain, by using process-mining building block to analyze the differences and similarities of results obtained by discovery techniques from event data that was filtered using different parameter values. In this example, event data is loaded and filtered several times using different parameter settings, producing several filtered event data sets. Each of these filtered event data sets is input for the same discovery technique using the same configuration of parameter(s).

4.3 Large-scale experiments

Empirical evaluation is often needed (and certainly recommended) when testing new process mining algorithms. In case of process mining, many experiments need to be conducted in order to prove that these algorithms or techniques can be applied in reality, and that the results are as expected. This is due to the richness of the domain. Process models can have a wide variety of routing behaviors, timing behavior, and second-order dynamics (e.g., concept drift). Event logs can be large or small and contain infrequent behavior (sometimes called noise) or not. Hence, this type of evaluation has to be conducted on a large scale. The execution and evaluation of such large-scale experiment results is a tedious and time-consuming task: it requires intensive human assistance by configuring each experiment's run and waiting for the results at the end of each run.

This can be greatly improved by using process mining workflows, as only one initial configuration is required. There are many examples for this analysis scenario within the process mining domain. Two of them are presented next.

4.3.1 Assessment of discovery techniques through massive testing

When developing new process discovery techniques, several experiments have to be conducted in order to test the *robustness* of the approach. As mentioned, many discovery techniques use parameters that can affect the result produced. It is extremely time-consuming and error prone to assess the discovery techniques using several different combinations of parameter values and, at the same time, testing on a dozen of different event-data sets.

Figure 12 shows the result of a large-scale experiment using n event data sets and m different parameter settings that produces $n \times m$ resulting process models. In this example, the same discovery technique with differ-

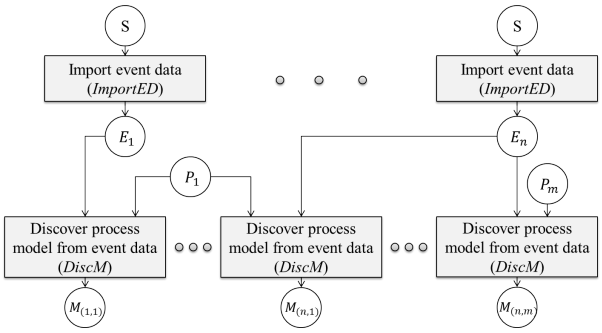


Fig. 12: Exhaustive testing of a discovery technique: Large-scale experiments using different types of event data and parameter combinations are needed to evaluate a discovery technique

ent parameters is used. However, one can consider the discovery algorithm to employ as an additional parameter. Therefore, the m different parameter settings can indicate m different discovery algorithms. After mining $n \times m$ models, the best model is considered.

4.3.2 Decomposed process discovery

Existing process mining techniques are often unable to handle "big event data" adequately. Decomposed process mining aims to solve this problem by decomposing the process mining problem into many smaller problems, which can be solved in less time and using less resources.

In decomposed process discovery, large event data sets are decomposed in sublogs, each of which refers to a subset of the process' activities. Once an appropriate decomposition is performed, the discovery can be applied to each cluster. The results in as many process models as the number of clusters; these models are finally merged to obtain a single process model. See for example the decomposed process mining technique described in [48] which presents an approach that clusters the event data, applies discovery techniques to each cluster, and merges the process models.

Figure 13 shows a process-mining workflow that splits the event data into n subsets, then uses a discovery algorithm to discover models for each of these subsets, and finally merges them into a single process model.

4.4 Repeating questions

Whereas the previous scenarios are aimed at (data) scientists, process mining workflows can also be used to lower the threshold for process mining. After the process mining workflow has been created and tested, the same analysis can be repeated easily using different subsets of data and different time-periods. Without workflow support this implies repeating the analysis steps manually or use *hardcoded* scripts that perform them over some

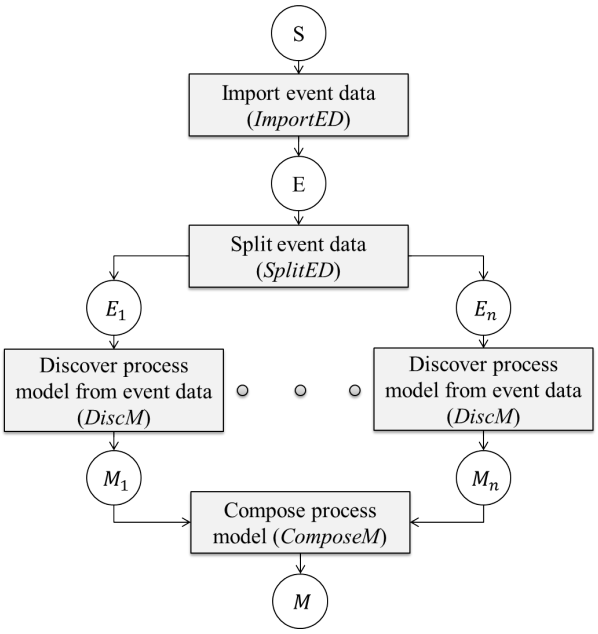


Fig. 13: Decomposed process discovery: a generic example using event data splitting, model composition and a specified discovery technique.

input data. The use of scientific workflows is clearly beneficial: the same workflow can be replayed many times using different inputs where no further configuration is required.

There are many examples for this analysis scenario within the process-mining domain. Two representative examples are described next.

4.4.1 Periodic benchmarking

Modern organizations make large investments to improve their own processes: better performance in terms of costs, time, or quality. In order to measure these improvements, organizations have to evaluate their performance periodically. This requires them to evaluate performance of the new time-period and compare it with the previous periods. Performance can improve or degrade in different time-periods. Obviously, the returned results require human judgments and, hence, cannot be fully automated by the scientific workflow.

Figure 14 shows an example of this analysis scenario using different process-mining building blocks. Let us assume that we want to compare period τ_k with period τ_{k-1}. For period τ_k, the entire event data is loaded and, then, is filtered so as to only keep portion E_{τ_k} that refers to the period τ_k only. Using portion E_{τ_k}, a process model M_{τ_k} is discovered. For period τ_{k-1}, the entire event data is loaded and, then, is filtered so as to only keep the portion $E_{\tau_{k-1}}$ that refers to the period τ_{k-1}. Finally, an evaluation is computed about the conformance between model M_{τ_k} and event-data portion E_{τ_k} and between M_{τ_k}

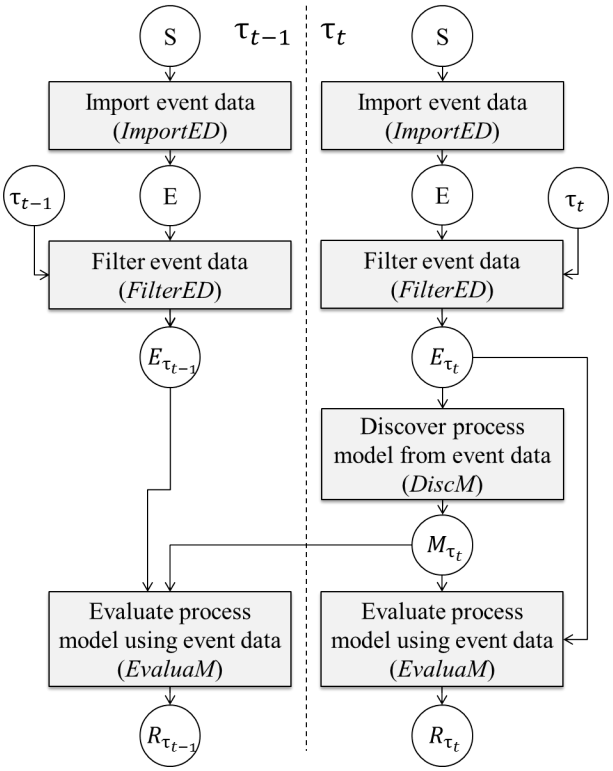


Fig. 14: Periodic performance benchmark: Process mining workflow for comparing the performance of the process in two different time-periods (t and $t-1$).

and $E_{\tau_{k-1}}$. Each evaluation will return valuable results, which are compared to find remarkable changes.

4.4.2 Report generation over collections of data sets

Scientific workflows are very handy when generating several reports for different portions of event data, e.g., different groups of patients or customers. Since the steps are the same and the only difference is concerned with using different portions of events, this can be easily automated, even when dozens of subsets need to be taken into consideration.

From this, it follows that this scenario shares common points with large-scale experiments. However, some differences exist. The report-generation scenario is characterized by a stable workflow with a defined set of parameters, whereas in the large-scale experiments scenario, parameters may change significantly in the different iterations. In addition to that, the input elements used in report-generation scenarios are similar and comparable event data sets. This can be explained by the desire that reports should have the same structure. In case of large-scale experiments, event data sets may be heterogenous. It is actually worthwhile repeating the experiments using diverse and dissimilar event data sets as input.

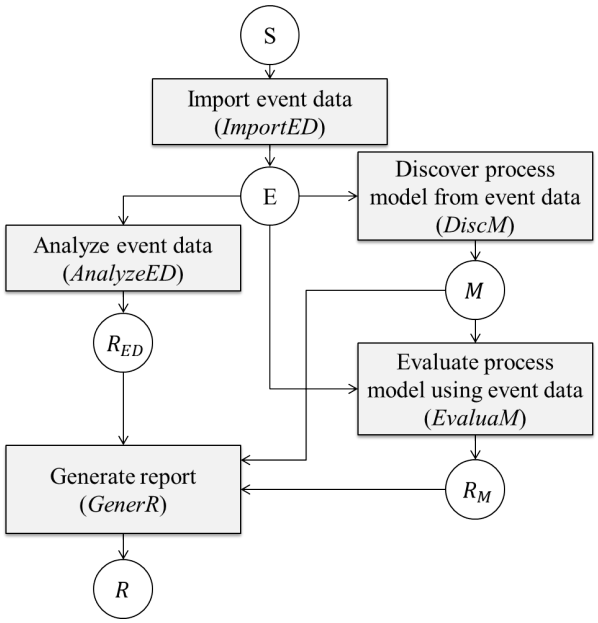


Fig. 15: Report Generation workflow

Figure 15 illustrates a potential scientific workflow to generate reports that contain process-mining results. For the sake of explanation, the process mining workflow is kept simple. The report is assumed to contain only three objects: the result R_{ED} of the analyze of the input event data, the discovered process model M and the results R_M of the evaluation of such a model against the input event data. Process-mining building block *Generate report* takes these three objects as input and combines them into a reporting document R.

5 Implementation

Our framework to support process mining workflows shown in Figure 1 is supported by *RapidProM*. RapidProM was implemented using ProM and RapidMiner. The building blocks defined in Section 3 have been concretely implemented in RapidProM. Most of the building blocks have been realized using RapidMiner-specific wrappers of plug-ins of the ProM Framework [47]. ProM is a framework that allows researchers to implement process mining algorithms in a standardized environment, which provides a number of facilities to support programmers. Nowadays, it has become the *de-facto* standard for process mining. ProM can be freely downloaded from `http://www.promtools.org`. The extension of RapidMiner to provide process-mining blocks for scientific workflows using ProM is also freely available. At the time of writing, RapidProM provides 37 process mining operators, including several process-discovery algorithms and filters as well as importers and exporters from/to different process-modeling notations.

The first version of RapidProM was presented during the BPM 2014 demo session [28]. This initial version successfully implemented basic process-mining functionalities and has been downloaded 4020 times since its release in July 2014 until April 2015 (on average, over 400 monthly downloads). However, process mining is a relatively new discipline, which is developing and evolving very fast. Therefore, various changes and extensions were needed to keep up with the state-of-the-art. The new version incorporates implementations of various new algorithms, which did not exist in the first version.

The RapidProM extension is hosted both at `http://www.rapidprom.org` and in the RapidProM extension manager server, which can be directly accessed through the RapidMiner Marketplace. After installation, the RapidMiner operators are available for use in any RapidMiner workflow. Figure 16 shows an example of a process-mining scientific workflow implemented using RapidProM.

Readers are referred to `http://www.rapidprom.org` for detailed installation, setup and troubleshooting instructions.

Table 1 shows the ProM import plugins implemented in RapidProM Version 2. These five operators are complemented with RapidMiner native operators to export visual results and data tables, in a way that most final results of process mining workflows can be exported and saved outside RapidMiner.

Operator Name	Operator Description
Read Log (path)	Imports an event log from a specified path
Read Log (file)	Takes a file object (usually obtained from a "loop files" operator) and transforms it to an Event Log
Read PNML	Imports a Petri Net from a specified path
Export Event Log	Exports an Event Log in different formats
Export PNML	Exports a Petri Net in PNML format

Table 1: Import/Export Operators

Table 2 shows a list of ProM Discovery plugins implemented in RapidProM as Discovery Operators. These nine operators (usually referred to as *miners*) are the most commonly used discovery techniques for process mining. These discovery operators produce different models using different techniques and parameters to fine-tune the resulting model.

Table 3 shows a list of ProM visualization plugins implemented in RapidProM as visualization operators. These four visualization plugins are accompanied by renderers that allow one to inspect both intermediate and final results during and after the execution of process mining workflows.

Operator Name	Operator Description
Alpha Miner	Discovers a Petri Net. Fast but results are not always reliable because of overfitting issues
ILP Miner	Discovers a Petri Net by solving ILP problems. Result have perfect fitness but generally poor precision. Slow on large Logs
Genetic Miner	Discovers a Heuristics Net using genetic algorithms. Depending on the parameter settings it can be slow or fast
Evolutionary Tree Miner	Discovers a Process Tree using a guided genetic algorithms based on model quality dimensions. Guarantees soundness but cannot represent all possible behavior due to its block-structured nature
Heuristics Miner	Discovers a Heuristics Net using a probabilistic approach. Good when dealing with noise. Fast
Inductive Miner	Discovers a Process Tree or Petri Net. Good when dealing with infrequent behavior and large Logs. Soundness is guaranteed
Social Network Miner	Discovers a Social Network from the Event Log resources. Different Social Networks can be obtained: similar task, handover of work, etc.
Transition System Miner	Discovers a Transition System using parameters to simplify the space-state exploration.
Fuzzy Miner	Discovers a Fuzzy Model. Good when dealing with unstructured behavior. Fast

Table 2: Discovery Operators

Operator Name	Operator Description
Dotted Chart	Shows the temporal distribution of events within traces
Inductive Visual Miner	Process exploration tool that shows an annotated interactive model for quick exploration of a Log
Animate Log in Fuzzy Instance	Shows an animated replay of a Log projected over a Fuzzy Instance
PomPom	Petri Net visualizer that emphasizes those parts of the process that correspond to high-frequent events in a given Log

Table 3: Visualization Operators

Table 4 shows a list of ProM conversion plugins implemented in RapidProM as conversion operators. These four conversion plugins are intended for converting models into other model formats. This way we improve the chances that a produced model can be used by other operators. For example, if a heuristics net is discovered

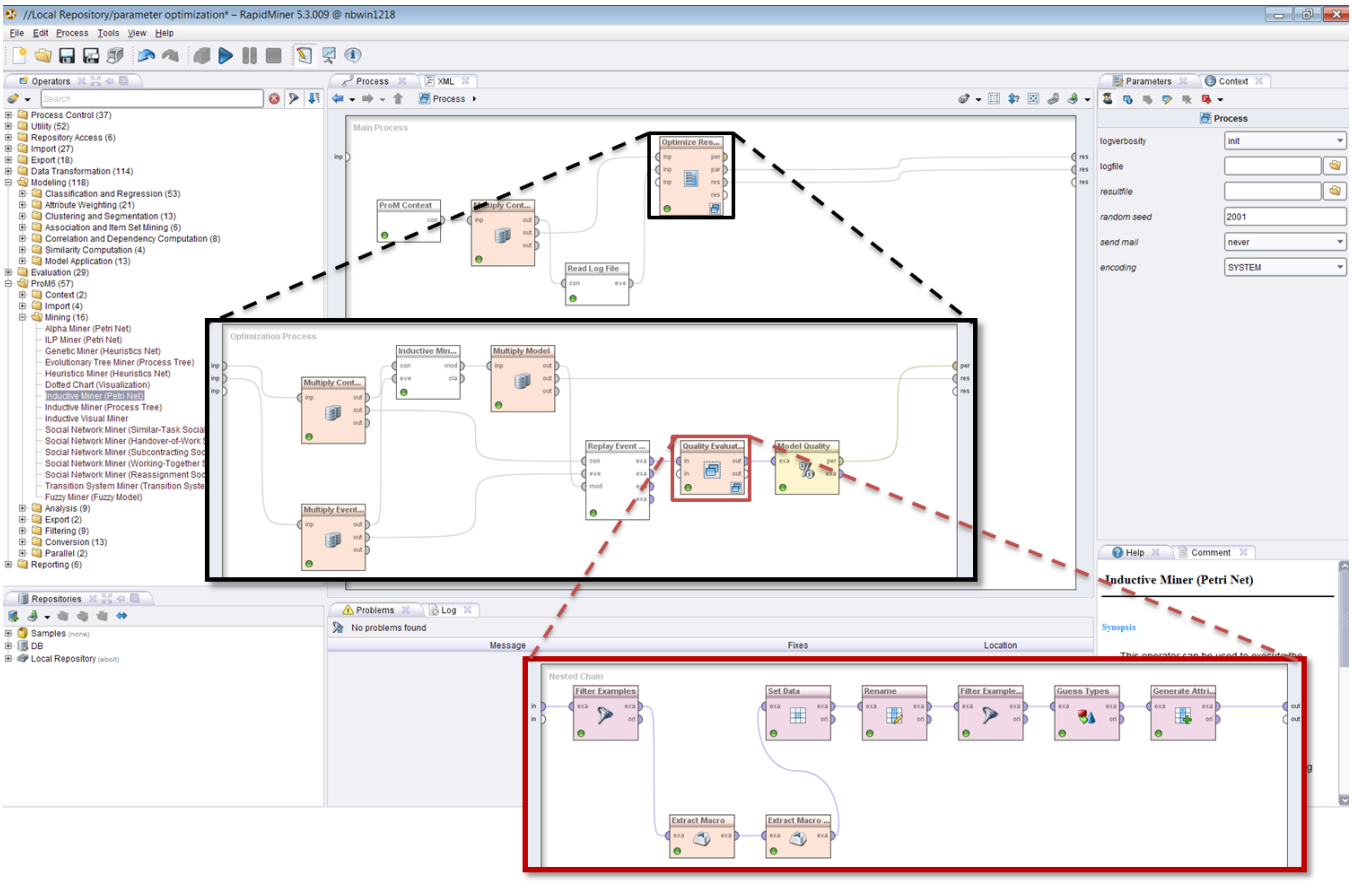


Fig. 16: Process Mining Workflows in RapidMiner through the RapidProM extension

from an Event Log using the Heuristics Miner, then the Replay Log on Petri Net (Conformance) operator cannot be executed unless a conversion to Petri Net is performed (which is supported).

Operator Name	Operator Description
Reachability Graph to Petri Net	Converts a Reachability Graph into a Petri Net
Petri Net to Reachability Graph	Converts a Petri Net into a Reachability Graph
Heuristics Net to Petri Net	Converts a Heuristics Net into a Petri Net
Process Tree to Petri Net	Converts a Process Tree into a Petri Net

Table 4: Conversion Operators

Table 5 shows a list of log processing operators implemented in RapidProM. Some of these eight operators use ProM functionalities to perform their tasks, but others were developed specifically for RapidProM, as the ProM framework generally does not use flat data tables to represent event data. These operators are used to modify an event log by adding attributes, events, or converting it to data tables, and vice versa.

Table 6 shows a list of ProM plugins implemented in RapidProM as analysis operators.

6 Evaluation

This section shows a number of instantiations of scientific workflows in RapidProM, highlighting the benefits of using scientific workflows for process mining. They are specific examples of the analysis scenarios discussed in Section 4

6.1 Evaluating result optimality

The first experiment is related to *Result Optimality*. In this experiment, we implemented a process mining workflow using RapidProM to extract the model that scores higher with respect to the geometric average of precision and replay fitness. For this experiment, we employed the *Inductive Miner - Infrequent* discovery technique [24] and used different values for the *noise threshold* parameter. This parameter is defined in a range of values between 0 and 1. This parameter allows for filtering out

Operator Name	Operator Description
Add Table Column to Event Log	Adds a single Data Table column as trace attribute to a given Event Log
Add Trace Attributes to Event Log	Adds all columns of a Data Table (except case id) as trace attributes to a given Event Log
Add Event Attributes to Event Log	Adds all columns of a Data Table (except case id and event id) as event attributes to a given Event Log
Add Events to Event Log	Adds Events to a given Event Log from selected columns on a Data Table
Merge Event Logs	Merges two Event Logs
Add Artificial Start and End Event	Adds an artificial Start Event to the beginning, and an artificial End Event to the ending of each trace
Event Log to ExampleSet	Converts an Event Log into a Data Table (ExampleSet)
ExampleSet to Event Log	Converts a Data Table (ExampleSet) into an Event Log

Table 5: Log Processing Operators

Operator Name	Operator Description
WOFLAN	Analyzes the soundness of a Petri Net
Select Fuzzy Instance	Selects the best fuzzy instance from a Fuzzy Model
Repair Model	Replays an Event Log in a Petri Net and repairs this net to improve fitness.
Reduce Silent Transitions	Reduces a Petri Net by removing invisible transitions (and places) that are not used
Feature Prediction	Produces predictions of business process features using decision trees
Replay Log on Petri Net (Performance)	Replays a Log on a Petri Net and generates performance metrics such as throughput time, waiting time, etc.
Replay Log on Petri Net (Conformance)	Replays a Log on a Petri Net and generates conformance metrics such as fitness

Table 6: Analysis Operators

infrequent behavior contained in event data in order to produce a simpler model: the lower the value is for this parameter (i.e., close to 0), the larger the fraction of behavior observed in the event data that the model allows. To measure fitness and precision, we employ the conformance-checking techniques reported in [1, 2]. All techniques are available as part of the RapidProM extension.

This experiment instantiates the analysis scenario described in Section 4.1 and depicted in Figure 8. The model obtained with the default value of the parameter is compared with the model that (almost) maximizes the geometric average of fitness and precision. To obtain this result, we designed a scientific workflow where several models are discovered with different values of the *noise threshold* parameter. Finally, the workflow selects the model with the highest value of the geometric average among those discovered. As input, we used an event-data log that records real-life executions of a process for road-traffic fine managements, which is employed by a local-police force in Italy [11]. This event data refers to 150370 process-instance executions and records the execution of around 560000 activities.

Figure 17b shows the model obtained through our scientific workflow, whereas Figure 17a illustrates the model generated using default parameters.

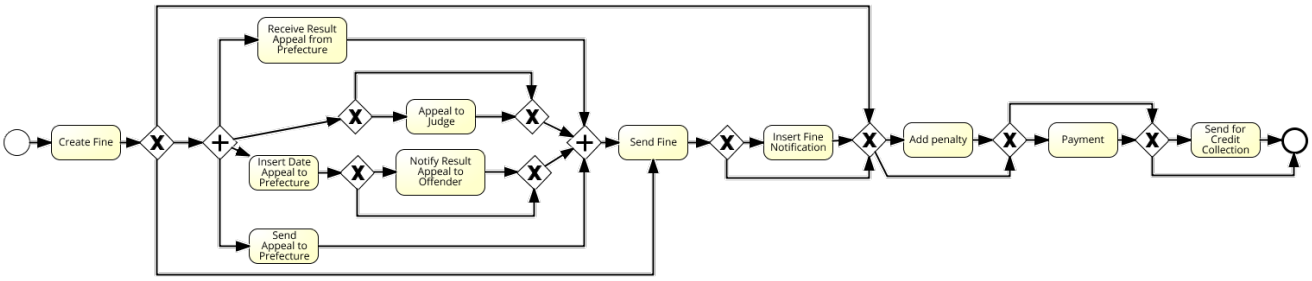
There are clear differences between the models. For example, in the default model, parallel behavior dominates the beginning of the process. Instead, the "optimal model" presents simpler choices. Another example concerns the final part of the model. In the default model, the latest process activities can be skipped through. However, in the optimal model, this is not possible. The optimal model has a replay fitness and precision of 0.921 and 0.903 respectively, with geometric average 0.912. It scores better than the model obtained through default parameters, where the replay fitness and precision is 1 and 0.548, respectively, with geometric average 0.708. The optimal model was generated with value 0.7 for the noise threshold parameter.

6.2 Evaluating parameter sensitivity

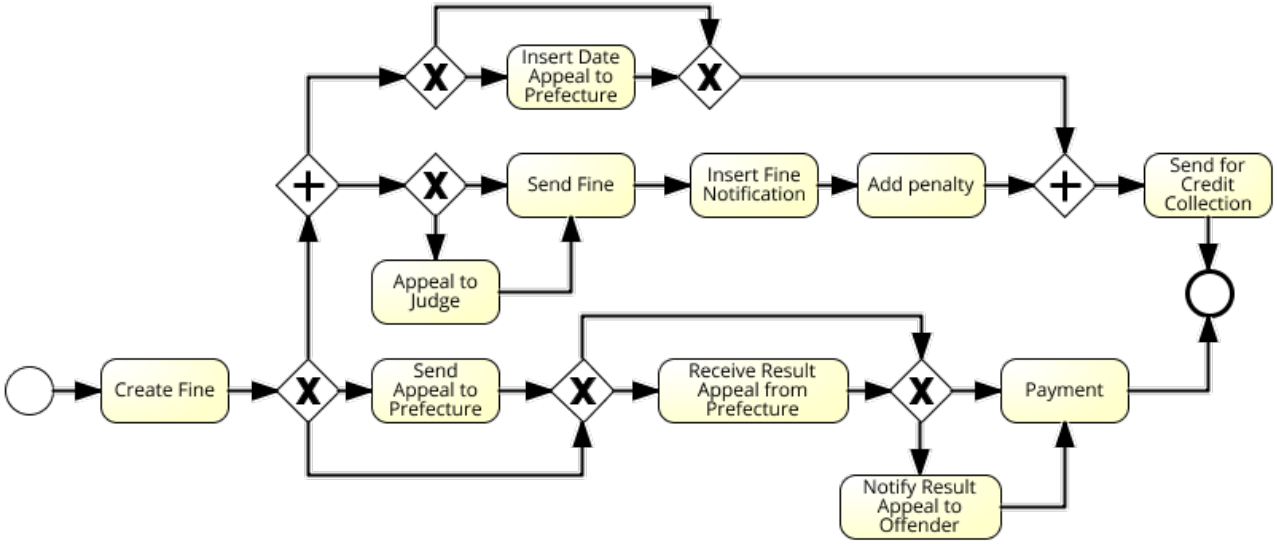
As second experiment illustrating the benefits of using scientific workflows for process mining, we conducted an analysis of the sensitivity of the *noise threshold* parameter of the *Inductive Miner - infrequent*. We used again the event data of the road-traffic fine management process also used in Section 6.1. This experiment operationalizes the analysis scenario discussed in Section 4.2 and depicted in Figure 10. In this experiment, we implemented a process mining workflow using RapidProM to explore the effect of this parameter in the final quality of the produced model. In order to do so, we discovered 41 models using different parameter values between 0 and 1 (i.e., a step-size 0.025) and evaluated their quality through the geometric average of replay fitness and precision used before.

Figure 18 shows the results of these evaluations, showing the variation of the geometric average for different values of the *noise threshold* parameter.

By analyzing the graph, the models with higher geometric average are produced when the parameter takes on a value between 0.675 and 0.875. The worst model is obtained when value 1 is assigned to the parameter.



(a) Model mined using the default value of the noise-threshold parameter, which is 0.2. The geometric average is 0.708



(b) Model mined using one of the best values of the noise-threshold parameter, which is 0.7. The geometric average is 0.912

Fig. 17: Comparison of process models that are mined with the default parameters and with the parameters that maximize the geometric average of replay fitness and precision. The process is concerned with road-traffic fine management and models are represented using the BPMN notation.

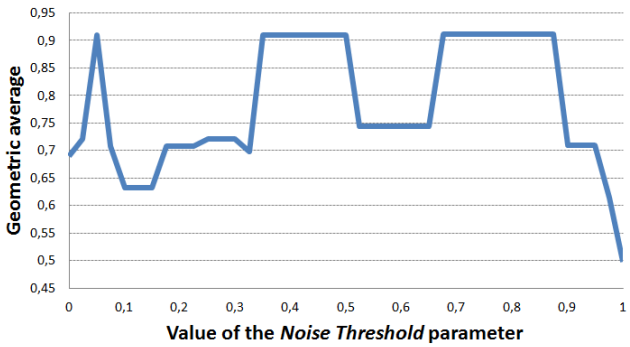


Fig. 18: Parameter sensitivity analysis: Variation of the geometric average of fitness and precision when varying the value of the *noise threshold* parameter.

6.3 Performing large scale experiments

As mentioned before, the use of scientific workflows is very beneficial for conducting large-scale experiments with many event logs. When assessing a certain process-mining technique one cannot rely on a single event log to draw conclusions.

For instance, here we want to study how the *noise threshold* parameter influences the quality of the discovered model, in term of geometric average of fitness and precision. In Section 4.2, the experiment was conducted using a single event log, but RapidProM allows us to do this for any number of event logs. To illustrate this, we use 11 real-life event logs and produce the corresponding process models using different parameter settings.

Table 7 shows the results of this evaluation, where each cell shows the geometric average of the replay fitness and the precision of the model obtained using a

Event data	nt=0	nt=0.1	nt=0.2	nt=0.3	nt=0.4	nt=0.5	nt=0.6	nt=0.7	nt=0.8	nt=0.9	nt=1	Average
BPI2012 [45]	0.314	0.730	0.430	0.450	0.508	0.474	0.675	0.683	0.674	0.679	0.644	0.569
BPI2013 [32]	0.847	0.826	0.778	0.863	0.458	0.458	0.458	0.458	0.458	0.458	0.453	0.592
BPI2014 [46]	0.566	0.720	0.708	0.613	0.616	0.654	0.626	0.414	0.530	0.527	0.490	0.588
Hospital [44]	0.153	0.111	0.546	0.473	0.338	0.172	0.280	0.342	0.392	0.515	0.517	0.349
Road Fines [11]	0.689	0.633	0.708	0.721	0.909	0.909	0.744	0.912	0.912	0.710	0.498	0.758
CoSeLoG 1 [5]	0.143	0.366	0.389	0.576	0.687	0.710	0.737	0.668	0.673	0.649	0.594	0.563
CoSeLoG 2 [6]	0.095	0.191	0.146	0.233	0.127	0.167	0.250	0.177	0.218	0.180	0.362	0.195
CoSeLoG 3 [7]	0.182	0.352	0.573	0.640	0.170	0.209	0.628	0.632	0.585	0.732	0.657	0.487
CoSeLoG 4 [8]	0.190	0.448	0.488	0.640	0.623	0.163	0.553	0.621	0.546	0.518	0.670	0.496
CoSeLoG 5 [9]	0.160	0.199	0.445	0.517	0.522	0.628	0.634	0.145	0.246	0.222	0.602	0.393
CoSeLoG R. [10]	0.520	0.860	0.838	0.869	0.859	0.377	0.868	0.868	0.883	0.861	0.656	0.769
Average	0.350	0.494	0.549	0.599	0.528	0.447	0.586	0.538	0.556	0.550	0.558	

Table 7: Summary of a few large-scale experimental results: Evaluating the geometric average of replay fitness and precision of models discovered with the Inductive Miner using different values of the noise threshold parameter (columns) and different real-life sets of event data (rows). We use *nt* to indicate the value of the *noise threshold* parameter of application of the algorithm.

specific parameter value (column) and event data (row). Every event log used in this experiment is publicly available through the Digital Object Identifiers (DOIs) of the included references. To use some of them for discovery, we had to conduct some pre-processing (depending on the specifics of the event data).

The hospital event data set [44] was extremely unstructured. To provide reasonable results and to allow for conformance checking using alignments, we filtered the event log to retain the 80% most frequent behavior before applying the mining algorithm. The same was done for the five CoSeLog event logs [5–9].

The actual results in Table 7 are not very relevant for this paper. It just shows that techniques can be evaluated on a large scale by using scientific workflows.

6.4 Automatic report generation

To illustrate the fourth analysis scenario we used event data related to the study behavior and actual performance of students of the Faculty of Mathematics and Computer Science at Eindhoven University of Technology (TU/e). TU/e provides video lectures for many courses to support students who are unable to attend face-to-face lectures for various reasons. The event data record the views of video lectures and the exam attempts of all TU/e courses.

First of all, students generate events when they watch lectures. It is known how long and when they watch a particular lecture of a particular course. These data can be preprocessed so that low-level events are collapsed into lecture views. Second, students generate events when they make exams and the result is added to the event.

For each course, we have generated a report that includes the results of the application of various data-mining and process-mining techniques. *This generation is automatic in the sense that the scientific workflow takes a list of courses as input and produces as many reports as the number of course in the list.*

The report contains three sections: course information, core statistics and advanced analysis.

Figure 19 shows a small part of the report generated for the course on Business Information Systems (2II05). In the first section, the report provides information about the course, the bachelor or master programs which it belongs to, as well as the information about the overall number of views of the course's video lectures. In the second section (only small fragment is shown), some basic distributions are calculated. For example, statistics are reported about the division per gender, nationality and final grade. The third section is devoted to process mining results. The results of applying conformance checking using the event data and the ideal process model where a student watches every video lecture and in the right order, namely he/she watches the i^{th} video lecture only after watching the $(i-1)^{th}$ video lecture. As expected, the results show a positive correlation between higher grades and higher compliance with the normative process just mentioned: The more a student watches all video lectures in the right order, the higher the corresponding grade will be. In addition to showing the conformance information, the report always embeds a dotted chart. The dotted chart is a similar to a Gannt chart, see building block *AnalyzeED*. The dotted chart shows the distribution of events for the different students over time. This way one can see the patterns and frequency with which students watch video lectures.

Note that reports like the one shown in Figure 19 are very informative for both professors and students. By using RapidProM we are able to automatically generate reports for all courses (after data conversion and modeling the desired process mining workflow).

7 Conclusions

This paper presented a framework for supporting the design and execution of process mining workflows. As

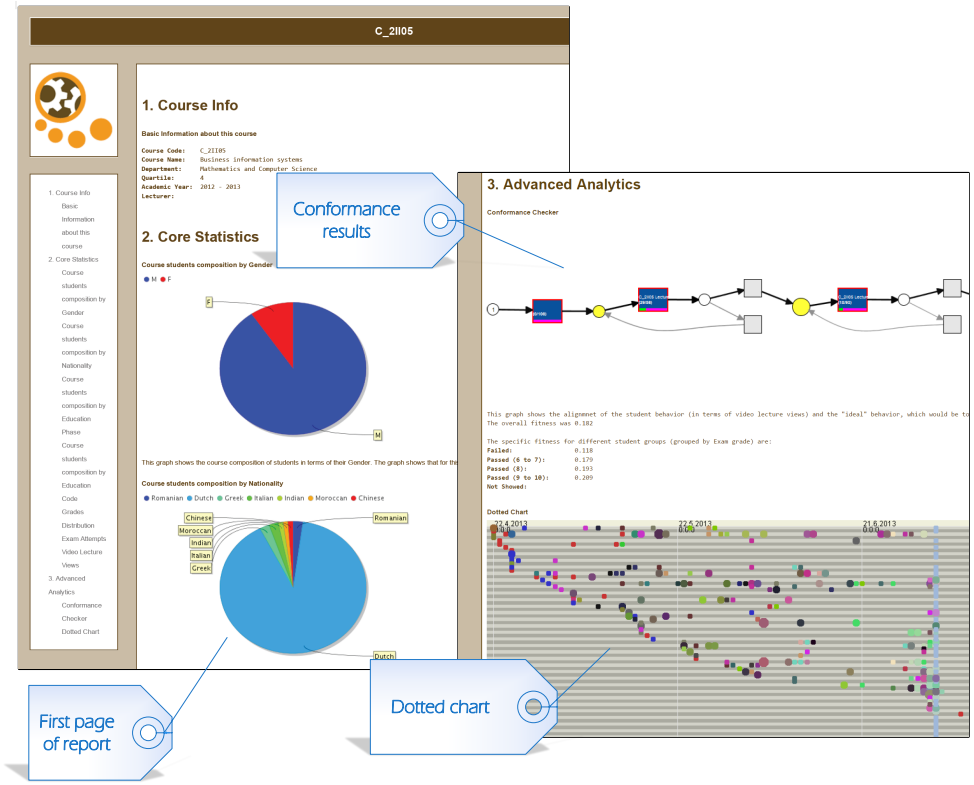


Fig. 19: Fragments of the automatically generated report using RapidProM

argued, scientific workflow systems are not tailored towards the analysis of processes based on models and logs. Tools like RapidMiner and KNIME can model analysis workflows but do not provide any process mining capabilities. The focus of these tools is mostly on traditional data mining and reporting capabilities that tend to use tabular data. Also more classical Scientific Workflow Management (SWFM) systems like Kepler and Taverna do not provide dedicated support for artifacts like process models and event logs. Process mining tools like ProM, Disco, Perceptive, Celonis, QPR, etc. do not provide any workflow support. The inability to model and execute process mining workflows was the primary motivation for developing the framework presented in this paper.

We proposed generic *process mining building blocks* grouped into six categories. These can be chained together to create process mining workflows. We identified four broader *analysis scenarios* and provided conceptual workflows for these. The whole approach is supported using *RapidProM* which is based on ProM and RapidMiner. RapidProM has been tested in various situations and in this paper we demonstrated this using concrete instances of the four analysis scenarios. RapidProM is freely available via `http://www.rapidprom.org` and the RapidMiner Market place.

Future work aims at extending the set of process mining building blocks and evaluating RapidProM in various case studies. We continue to apply RapidProM in all four areas described. Moreover, we would like to make standard workflows available via infrastructures like myExperiment and OpenML. We are also interested a further cross-fertilizations between process mining and other analysis techniques available in tools like RapidMiner and KNIME (text mining, clustering, predictive analytics, etc.).

Acknowledgements

The authors thank Ronny Mans for his seminal work on the initial version of RapidProM.

References

1. Arya Adriansyah, Jorge Munoz-Gama, Josep Carmona, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. Alignment based precision checking. In *Business Process Management Workshops*, volume 132, pages 137–149. Springer Berlin Heidelberg, 2013.
2. Arya Adriansyah, Jorge Munoz-Gama, Josep Carmona, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. Measuring precision of modeled behavior. *Information Systems and e-Business Management*, 13(1):37–67, 2015.

3. Roger Barga and Dennis Gannon. Scientific versus business workflows. In Ian J. Taylor, Ewa Deelman, Dennis B. Gannon, and Matthew Shields, editors, *Workflows for e-Science*, pages 9–16. Springer Verlag, Berlin, 2007.
4. Michael R. Berthold, Nicolas Cebron, Fabian Dill, Thomas R. Gabriel, Tobias Koetter, Thorsten Meinl, Peter Ohl, Christoph Sieb, Kilian Thiel, and Bernd Wiswedel. Knime: The konstanz information miner. In Christine Preisach, Hans Burkhardt, Lars Schmidt-Thieme, and Reinhold Decker, editors, *Data Analysis, Machine Learning and Applications*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 319–326. Springer Berlin Heidelberg, 2008.
5. Joos C. A. M. Buijs. Environmental permit application process (wabo), coselog project, municipality 1. `10.4121/uuid:c45dcbe9-557b-43ca-b6d0-10561e13dcb5`, 2014.
6. Joos C. A. M. Buijs. Environmental permit application process (wabo), coselog project, municipality 2. `10.4121/uuid:34b4f6f4-dbe0-4857-bf75-5b9e1138eb87`, 2014.
7. Joos C. A. M. Buijs. Environmental permit application process (wabo), coselog project, municipality 3. `10.4121/uuid:a8ed945d-2ad8-480e-8348-cf7f06c933b3`, 2014.
8. Joos C. A. M. Buijs. Environmental permit application process (wabo), coselog project, municipality 4. `10.4121/uuid:e8c3a53d-5301-4afb-9bcd-38e74171ca32`, 2014.
9. Joos C. A. M. Buijs. Environmental permit application process (wabo), coselog project, municipality 5. `10.4121/uuid:c399c768-d995-4086-adda-c0bc72ad02bc`, 2014.
10. Joos C. A. M. Buijs. Receipt phase of an environmental permit application process (wabo), coselog project. `10.4121/uuid:a07386a5-7be3-4367-9535-70bc9e77dbe6`, 2014.
11. Massimiliano de Leoni and Felix Mannhardt. Road traffic fine management process. `10.4121/uuid:270fd440-1057-4fb9-89a9-b699b47990f5`, 2015.
12. Claudia Diamantini, Domenico Potena, and Emanuele Storti. Mining usage patterns from a repository of scientific workflows. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, pages 152–157, New York, NY, USA, 2012. ACM.
13. Daniel Garijo, Pinar Alper, Khalid Belhajjame, Óscar Corcho, Yolanda Gil, and Carole A. Goble. Common motifs in scientific workflows: An empirical analysis. *Future Generation Comp. Syst.*, 36:338–351, 2014.
14. Carole A. Goble, Jiten Bhagat, Sergejs Aleksejevs, Don Cruickshank, Danius Michaelides, David Newman, Mark Borkum, Sean Bechhofer, Marco Roos, Peter Li, and David De Roure. myExperiment: A repository and social network for the sharing of bioinformatics workflows. *Nucleic Acids Research*, 38(suppl 2):W677–W682, 2010.
15. Jeremy Goecks, Anton Nekrutenko, James Taylor, and The Galaxy Team. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biology*, 11(8):R86, 2010.
16. David J. Hand, Padhraic Smyth, and Heikki Mannila. *Principles of Data Mining*. MIT Press, Cambridge, MA, USA, 2001.
17. Markus Hofmann and Ralf Klinkenberg. *RapidMiner: Data Mining Use Cases and Business Analytics Applications*. Chapman and Hall/CRC, 2013.
18. Duncan Hull, Katy Wolstencroft, Robert Stevens, Carole A. Goble, Matthew R. Pocock, Peter Li, and Tom Oinn. Taverna: A tool for building and running workflows of services. *Nucleic Acids Research*, 34:729–732, 2006.
19. IEEE Task Force on Process Mining. Process Mining Case Studies. `http://www.win.tue.nl/ieeetfpm/doku.php?id=shared:process_mining_case_studies`, 2013.
20. Ross Ihaka and Robert Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314, 1996.
21. Daniel Keim, Gennady Andrienko, Jean-Daniel Fekete, Carsten Grg, Jrn Kohlhammer, and Guy Melanon. Visual analytics: Definition, process, and challenges. In Andreas Kerren, JohnT. Stasko, Jean-Daniel Fekete, and Chris North, editors, *Information Visualization*, volume 4950 of *Lecture Notes in Computer Science*, pages 154–175. Springer Berlin Heidelberg, 2008.
22. Janez Kranjc, Vid Podpean, and Nada Lavra. Clowdflows: A cloud based scientific workflow platform. In PeterA. Flach, Tijl De Bie, and Nello Cristianini, editors, *Machine Learning and Knowledge Discovery in Databases*, volume 7524 of *Lecture Notes in Computer Science*, pages 816–819. Springer Berlin Heidelberg, 2012.
23. Marcello La Rosa, Hajo A. Reijers, Wil M. P. van der Aalst, Remco M. Dijkman, Jan Mendling, Marlon Dumas, and Luciano García-Bañuelos. Apromore: An advanced process model repository. *Expert Systems with Applications*, 38(6):7029–7040, 2011.
24. Sander J. J. Leemans, Dirk Fahland, and Wil M. P. van der Aalst. Discovering block-structured process models from event logs - a constructive approach. In José-Manuel Colom and Jörg Desel, editors, *Application and Theory of Petri Nets and Concurrency*, volume 7927 of *Lecture Notes in Computer Science*, pages 311–329. Springer Berlin Heidelberg, 2013.
25. Frank Leymann and Dieter Roller. *Production workflow - concepts and techniques*. Prentice Hall, 2000.
26. Richard Littauer, Karthik Ram, Bertram Ludäscher, William Michener, and Rebecca Koskela. Trends in use of scientific workflows: Insights from a public repository and recommendations for best practice. *IJDC*, 7(2):92–100, 2012.
27. Bertram Ludäscher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew B. Jones, Edward A. Lee, Jing Tao, and Yang Zhao. Scientific workflow management and the kepler system. *Concurrency and Computation: Practice and Experience*, 18(10):1039–1065, 2006.
28. Ronny S. Mans, Wil M. P. van der Aalst, and H. M. W. Verbeek. Supporting process mining workflows with RapidProM. In Lior Limonad and Barbara Weber, editors, *Proceedings of the BPM Demo Sessions 2014 Co-located with the 12th International Conference on Business Process Management (BPM)*, volume 1295 of *CEUR Workshop Proceedings*, pages 56–60. CEUR-WS.org, 2014.
29. Tom M. Mitchell. *Machine learning*. McGraw Hill series in computer science. McGraw-Hill, 1997.

30. August-Wilhelm Scheer and Markus Nüttgens. Aris architecture and reference models for business process management. In Wil M. P. van der Aalst, Jörg Desel, and Andreas Oberweis, editors, *Business Process Management*, volume 1806 of *Lecture Notes in Computer Science*, pages 376–389. Springer Berlin Heidelberg, 2000.
31. Mirko Sonntag, Dimka Karastoyanova, and Ewa Deelman. Bridging the gap between business and scientific workflows: Humans in the loop of scientific workflows. In *Sixth International Conference on e-Science, e-Science 2010, 7-10 December 2010, Brisbane, QLD, Australia*, pages 206–213, 2010.
32. Ward Steeman. Bpi challenge 2013. `10.4121/500573e6-accc-4b0c-9576-aa5468b10cee`, 2013.
33. Bernhard Steffen, Tiziana Margaria, Ralf Nagel, Sven Joerges, and Christian Kubczak. Model-driven development with the jABC. In Eyal Bin, Avi Ziv, and Shmuel Ur, editors, *Hardware and Software, Verification and Testing*, volume 4383 of *Lecture Notes in Computer Science*, pages 92–108. Springer Berlin Heidelberg, 2007.
34. Ian J. Taylor, Ewa Deelman, Dennis B. Gannon, and Matthew Shields. *Workflows for e-Science: Scientific Workflows for Grids*. Springer Verlag, Berlin, 2007.
35. Kenneth J. Turner and Paul S. Lambert. Workflows for quantitative data analysis in the social sciences. *International Journal on Software Tools for Technology Transfer*, pages 1–18, 2014.
36. Wil M. P. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer-Verlag Berlin Heidelberg, 1st edition, 2011.
37. Wil M. P. van der Aalst. A decade of business process management conferences: Personal reflections on a developing discipline. In Alistair Barros, Avigdor Gal, and Ekkart Kindler, editors, *Business Process Management*, volume 7481 of *Lecture Notes in Computer Science*, pages 1–16. Springer Berlin Heidelberg, 2012.
38. Wil M. P. van der Aalst. Decomposing process mining problems using passages. In Serge Haddad and Lucia Pomello, editors, *Application and Theory of Petri Nets*, volume 7347 of *Lecture Notes in Computer Science*, pages 72–91. Springer Berlin Heidelberg, 2012.
39. Wil M. P. van der Aalst, Arya Adriansyah, and Boudewijn F. van Dongen. Replaying history on process models for conformance checking and performance analysis. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(2):182–192, 2012.
40. Wil M. P. van der Aalst, Alexander Dreiling, Florian Gottschalk, Michael Rosemann, and Monique H. Jansen-Vullers. Configurable process models as a basis for reference modeling. In Christoph J. Bussler and Armin Haller, editors, *Business Process Management Workshops*, volume 3812 of *Lecture Notes in Computer Science*, pages 512–518. Springer Berlin Heidelberg, 2006.
41. Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, Bartek Kiepuszewski, and Alistair P. Barros. Workflow patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.
42. Wil M. P. van der Aalst, Kees M. van Hee, Arthur H. M. ter Hofstede, Natalia Sidorova, H. M. W. Verbeek, Marc Voorhoeve, and Moe T. Wynn. Soundness of workflow nets: classification, decidability, and analysis. *Formal Aspects of Computing*, 23(3):333–363, 2011.
43. Wil M. P. van der Aalst, Anton J. M. M. Weijters, and Laura Maruster. Workflow mining: discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, Sept 2004.
44. Boudewijn F. van Dongen. Real-life event logs - hospital log. `10.4121/uuid:d9769f3d-0ab0-4fb8-803b-0d1120ffcf54`, 2011.
45. Boudewijn F. van Dongen. Bpi challenge 2012. `10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f`, 2012.
46. Boudewijn F. van Dongen. Bpi challenge 2014. `10.4121/uuid:d5ccb355-ca67-480f-8739-289b9b593aaf`, 2014.
47. Boudewijn F. van Dongen, Ana Karla A. de Medeiros, H. M. W. Verbeek, Anton J. M. M. Weijters, and Wil M. P. van der Aalst. The ProM framework: A new era in process mining tool support. In Gianfranco Ciardo and Philippe Darondeau, editors, *Applications and Theory of Petri Nets*, volume 3536 of *Lecture Notes in Computer Science*, pages 444–454. Springer Berlin Heidelberg, 2005.
48. Wil M. P. vander Aalst. Decomposing petri nets for process mining: A generic approach. *Distributed and Parallel Databases*, 31(4):471–507, 2013.
49. Ingo H. C. Wassink, Paul E. van der Vet, Katy Wolstencroft, Pieter B. T. Neerincx, Marco Roos, Han Rauwerda, and Timo M. Breit. Analysing scientific workflows: Why workflows not only connect web services. In *2009 IEEE Congress on Services, Part I, SERVICES I 2009, Los Angeles, CA, USA, July 6-10, 2009*, pages 314–321, 2009.
50. Anton J. M. M. Weijters and Wil M. P. van der Aalst. Rediscovering workflow models from event-based data using Little Thumb. *Integrated Computer-Aided Engineering*, 10(2):151–162, 2003.
51. Mathias Weske. *Business Process Management - Concepts, Languages, Architectures, 2nd Edition*. Springer, 2012.
52. Alexander Wickert and Anna-Lena Lamprecht. jABC-stats: An extensible process library for the empirical analysis of jABC workflows. In Tiziana Margaria and Bernhard Steffen, editors, *Leveraging Applications of Formal Methods, Verification and Validation. Specialized Techniques and Applications*, volume 8803 of *Lecture Notes in Computer Science*, pages 449–463. Springer Berlin Heidelberg, 2014.
53. Reng Zeng, Xudong He, Jiafei Li, Zheng Liu, and Wil M. P. van der Aalst. A method to build and analyze scientific workflows from provenance through process mining. In *3rd Workshop on the Theory and Practice of Provenance, TaPP'11, Heraklion, Crete, Greece, June 20-21, 2011*, 2011.