Process Mining: Overview and Opportunities

WIL VAN DER AALST, Eindhoven University of Technology

Over the last decade, process mining emerged as a new research field that focuses on the analysis of processes using event data. Classical data mining techniques such as classification, clustering, regression, association rule learning, and sequence/episode mining do not focus on business process models and are often only used to analyze a specific step in the overall process. Process mining focuses on end-to-end processes and is possible because of the growing availability of event data and new process discovery and conformance checking techniques.

Process models are used for *analysis* (e.g., simulation and verification) and *enactment* by BPM/WFM systems. Previously, process models were typically made by hand without using event data. However, activities executed by people, machines, and software leave trails in so-called *event logs*. Process mining techniques use such logs to discover, analyze, and improve business processes.

Recently, the Task Force on Process Mining released the Process Mining Manifesto. This manifesto is supported by 53 organizations and 77 process mining experts contributed to it. The active involvement of end-users, tool vendors, consultants, analysts, and researchers illustrates the growing significance of process mining as a bridge between data mining and business process modeling. The practical relevance of process mining and the interesting scientific challenges make process mining one of the "hot" topics in Business Process Management (BPM). This article introduces process mining as a new research field and summarizes the guiding principles and challenges described in the manifesto.

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database Applications—*Data Mining*

General Terms: Management, Measurement, Performance

Additional Key Words and Phrases: Process mining, business intelligence, business process management, data mining

ACM Reference Format:
van der Aalst, W. 2012. Process mining: Overview and opportunities. ACM Trans. Manage. Inf. Syst. 3, 2, Article 7 (July 2012), 17 pages.
DOI = 10.1145/2229156.2229157 http://doi.acm.org/10.1145/2229156.2229157

1. INTRODUCTION

Process mining aims to discover, monitor, and improve real processes by extracting knowledge from event logs readily available in today's information systems [van der Aalst 2011]. Over the last decade there has been a spectacular growth of event data and process mining techniques have matured significantly. As a result, management trends related to process improvement and compliance can now benefit from process mining.

Starting point for process mining is an *event log*. Each event in such a log refers to an *activity* (i.e., a well-defined step in some process) and is related to a particular *case*

Author's address: W. van der Aalst, Department of Mathematics and Computer Science, Eindhoven University of Technology, PO Box 513, 5600 MB, Eindhoven, Netherlands; email: w.m.p.v.d.aalst@tue.nl.

DOI 10.1145/2229156.2229157 http://doi.acm.org/10.1145/2229156.2229157

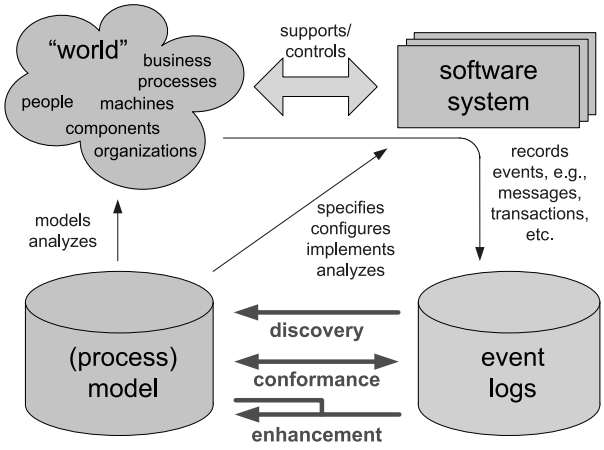

Fig. 1. The three basic types of process mining: (a) discovery, (b) conformance, and (c) enhancement.

(i.e., a *process instance*). The events belonging to a case are *ordered* and can be seen as one "run" of the process. Event logs may store additional information about events. In fact, whenever possible, process mining techniques use extra information such as the *resource* (i.e., person or device) executing or initiating the activity, the *timestamp* of the event, or *data elements* recorded with the event (e.g., the size of an order).

Event logs can be used to conduct three types of process mining, as shown in Figure 1 [van der Aalst 2011]. The first type of process mining is *discovery*. A discovery technique takes an event log and produces a model without using any a priori information. Process discovery is the most prominent process mining technique. For many organizations it is surprising to see that existing techniques are indeed able to discover real processes merely based on example behaviors stored in event logs. The second type of process mining is *conformance*. Here, an existing process model is compared with an event log of the same process. Conformance checking can be used to check if reality, as recorded in the log, conforms to the model and vice versa. The third type of process mining is *enhancement*. Here, the idea is to extend or improve an existing process model thereby using information about the actual process recorded in some event log. Whereas conformance checking measures the alignment between model and reality, this third type of process mining aims at changing or extending the a priori model. For instance, by using timestamps in the event log one can extend the model to show bottlenecks, service levels, and throughput times.

Unlike traditional *Business Process Management* (BPM) techniques that use hand-made models [Weske 2007], process mining is based on *facts*. Based on observed behavior recorded in event logs, intelligent techniques are used to extract knowledge. Therefore, we claim that process mining enables *evidence-based BPM*. Unlike existing analysis approaches, process mining is *process-centric* (and not data-centric), *truly intelligent* (learning from historic data), and *fact-based* (based on event data rather than opinions).

Process mining is related to data mining. Whereas classical data mining techniques are mostly data-centric [Hand et al. 2001], process mining is process-centric. Mainstream business process modeling techniques use notations such as the Business Process Modeling Notation (BPMN), UML activity diagrams, Event-driven Process Chains (EPC), and various types of Petri nets [Desel and Reisig 1998; van der Aalst and Stahl 2011; Weske 2007]. These notations can be used model process processes with concurrency, choice, iteration, etc.

This article not only introduces process mining as a new research field, but also familiarizes the reader with the *Process Mining Manifesto* [TFPM 2011] released by the Task Force on Process Mining in October 2011. The growing interest in log-based process analysis motivated the establishment of a Task Force on Process Mining in 2009. This manifesto aims to promote the topic of process mining. Moreover, by defining a set of guiding principles and listing important challenges, this manifesto hopes to serve as a guide for software developers, scientists, consultants, business managers, and end-users. The goal is to increase the maturity of process mining as a new tool to improve the (re)design, control, and support of operational business processes.

The remainder of this article is organized as follows. Section 2 introduces the notion of an event log, used as input for process mining. Section 3 shows how process models can be discovered from scratch using only raw event data. Section 4 discusses the second type of process mining: conformance checking. Section 5 elaborates on the third type of process mining: enhancement. The guiding principles and challenges listed in the manifesto are summarized in Section 6. Section 7 discusses tool support and shows some real-life examples. Section 8 concludes the article.

2. EVENT LOGS AS A STARTING POINT FOR PROCESS MINING

Digital event data is everywhere—in every sector, in every economy, in every organization, and in every home—and will continue to grow exponentially [Manyika et al. 2011]. The omnipresence of such data allows for new forms of process analysis, that is, based on observed facts rather than handmade models. Starting point for process mining is an *event log*.

To introduce the basic process mining concepts we use the event log shown in Figure 2 (log is taken from Chapter 5 of van der Aalst [2011]). This event log contains 1391 cases, that is, instances of some reimbursement process. There are 455 process instances following trace *acdeh*. Activities are represented by a single character: a = *register request*, b = *examine thoroughly*, c = *examine casually*, d = *check ticket*, e = *decide*, f = *reinitiate request*, g = *pay compensation*, and h = *reject request*. Hence, trace *acdeh* models a reimbursement request that was rejected after a registration, examination, check, and decision step. 455 cases followed this path consisting of five steps, that is, the first line in the table corresponds to $455 \times 5 = 2275$ events. The whole log consists of 7539 events.

Note that events can have all kinds of additional attributes (timestamps, transactional information, resource usage, etc.). Consider for example one of the 1391 "a" events. Such an event refers to the execution of "register request" for some reimbursement. The event may have a timestamp, for instance, "23-01-2012:8.38," and an attribute describing the resources involved. Moreover, data attributes of the reimbursement (e.g., name of customer or number of loyalty card) and data attributes of the registration event (e.g., the amount claimed or a booking reference) may have been recorded. All such attributes can be used by process mining techniques. However, the backbone of process mining is the control-flow perspective. Therefore, for simplicity, events in Figure 2 are described by their activity names only. However, it is important to realize that events can have various attributes, for instance, timestamps can be used for bottleneck analysis and resource attributes can be used for organizational mining (e.g., finding allocation rules).

3. DISCOVERY

This section introduces the notion of process discovery, that is, automatically construct models based on observed events.

#	trace
455	acdeh
191	abdeg
177	adceh
144	abdeh
111	acdeg
82	adceg
56	adbeh
47	acdefdbeh
38	adbeg
33	acdefbdeh
14	acdefbdeg
11	acdefdbeg
9	adcefcdeh
8	adcefdbeh
5	adcefbdeg
3	acdefbdefdbeg
2	adcefdbeg
2	adcefbdefbdeg
1	adcefdbefbdeh
1	adbefbdefdbeg
1	adcefdbefcdefdbeg
1391	

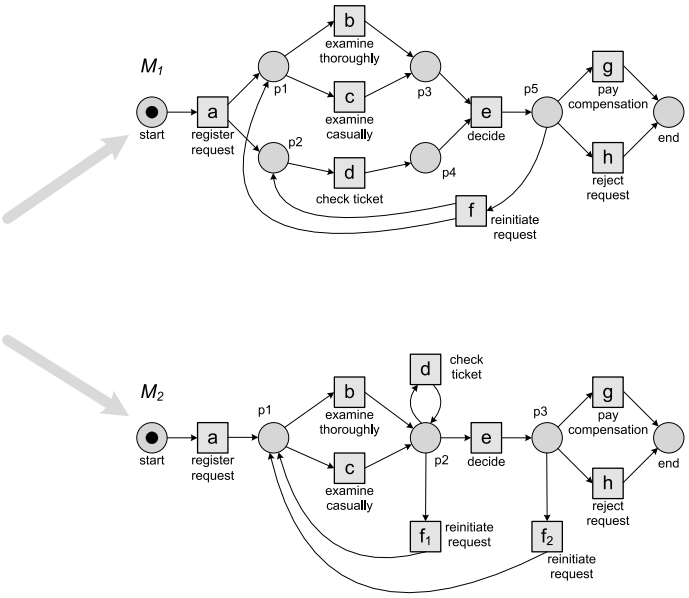

Fig. 2. One event log and two potential process models (M_1 and M_2) aiming to describe the observed behavior.

3.1 Applications of Process Discovery

Organizations use procedures to handle cases. Sometimes such procedures are enforced by the information system. However, in most cases, procedures are informal and may not have been documented at all. Moreover, even when procedures have been documented, reality may be very different. Therefore, it is important to discover the actual processes using event data. The discovered process models may be used

— for discussing problems among stakeholders (to reach consensus; it is important to have a shared view of the real processes),
— for generating process improvement ideas (seeing the actual process and its problems stimulates re-engineering efforts),
— for model enhancement (e.g., bottleneck analysis, see Section 5), and
— for configuring a WFM/BPM system (the discovered process model can serve as a template).

3.2 Learning Process Models From Event Logs

Process discovery techniques produce process models based on event logs such as the one shown in Figure 2. For example, the classical α-algorithm produces model M_1 for this log. This process model is represented as a *Petri net* [Desel and Reisig 1998; van der Aalst and Stahl 2011]. A Petri net consists of *places* (*start*, $p1$, $p2$, $p3$, $p4$, $p5$, and *end*) and *transitions* (a, b, c, d, e, f, g, and h). Transitions may be connected to places and places may be connected to transitions. It is not allowed to connect a place to a place or a transition to a transition.

The state of a Petri net, also referred to as *marking*, is defined by the distribution of *tokens* over places. A transition is *enabled* if each of its input places contains a token. For example, in M_1, transition a is enabled in the initial marking of M_1, because the only input place of a contains a token (black dot).

An enabled transition may *fire* thereby consuming a token from each of its input places and producing a token for each of its output places. Firing a in the initial marking corresponds to removing one token from *start* and producing two tokens (one for $p1$ and one for $p2$). After firing a, three transitions are enabled: b, c, and d. There is a non-deterministic choice between b and d. Firing b will disable c because the token is removed from the shared input place (and vice versa). Transition d is concurrent with b and c, that is, it can fire without disabling another transition. Transition e becomes enabled after d and b or c have occurred. Note that transition e in M_1 is only enabled if both input places ($p3$ and $p4$) contain a token. After executing e, three transitions become enabled: f, g, and h. These transitions are competing for the same token thus modeling a choice. When g or h is fired, the process ends with a token in place *end*. If f is fired, the process returns to the state just after executing a.

It is easy to check that all traces in the event log can be reproduced by M_1. This does not hold for the second process model in Figure 2. M_2 is able to reproduce traces such as *acdeh* (455 instances), *abdeg* (191 instances), and *acdefbdeh* (33 instances). Note that M_2 has two transitions corresponding to activity f. To refer to them they are named f_1 and f_2. M_2 also allows for behavior very different from what can be observed in the log, for instance, *abeg* and $abddddddf_1bddddeh$ are possible according to the model but do not appear in the log. There are also traces in the log that *cannot* be replayed by M_2, for instance, *adceh* (177 instances), *adceg* (82 instances), and *adcefcdeh* (9 instances) are not possible according to M_2.

The two process models in Figure 2 are visualized in terms of Petri nets. In fact, both models are so-called WF-nets [van der Aalst et al. 2011a]. A WF-net is a Petri net with one source place and one sink place such that all places and transitions are on a path from source to sink. Both models in Figure 2 have a source place named *start* and a sink place *end* and all nodes are on a path from *start* to *end*.

In general, the notation used to visualize the result may be very different from the representation used during the actual discovery process. All mainstream BPM notations (Petri nets, EPCs, BPMN, YAWL, UML activity diagrams, etc.) can be used to show discovered processes such as M_1 [van der Aalst 2011; Weske 2007].

3.3 Process Discovery Algorithms

Since the mid-nineties several groups have been working on techniques for automated process discovery based on event logs [Agrawal et al. 1998; Cook and Wolf 1998; Datta 1998; Greco et al. 2006; van der Aalst et al. 2004, 2007; van Dongen and van der Aalst 2004, 2005; Weijters and van der Aalst 2003]. Van der Aalst et al. [2003] give an overview of the early work in this domain. The idea to apply process mining in the context of workflow management systems was introduced in Agrawal et al. [1998]. In parallel, Datta [1998] looked at the discovery of business process models. Cook and Wolf [1998] investigated similar issues in the context of software engineering processes. Herbst [2000] was one of the first to tackle more complicated processes, for instance, processes containing duplicate tasks.

Most of the classical approaches have problems dealing with concurrency. The α-algorithm [van der Aalst et al. 2004] is an example of a simple technique that takes concurrency as a starting point. The α-algorithm scans the event log for particular patterns. For example, if activity a is followed by b but b is never followed by a, then it is assumed that there is a causal dependency between a and b. To reflect this

dependency, the corresponding Petri net should have a place connecting a to b. We use the notation, $a > b$ if and only if there is a trace $\sigma = \langle t_1, t_2, t_3, \ldots t_n \rangle$ in the log and an $i \in \{1, \ldots, n-1\}$ such that $t_i = a$ and $t_{i+1} = b$. $a \rightarrow b$ if and only if $a > b$ and $b \not> a$; $a\#b$ if and only if $a \not> b$ and $b \not> a$; and $a \| b$ if and only if $a > b$ and $b > a$. These four ordering relations are used to create places connecting the different transitions in the Petri net. The α-algorithm is simple and efficient, but has problems dealing with complicated routing constructs and noise (like most of the other approaches described in literature).

Region-based approaches are able to express more complex control-flow structures without underfitting. State-based regions were introduced in 1989 [Ehrenfeucht and Rozenberg 1989] and generalized in various ways [Cortadella et al. 1998]. In van der Aalst et al. [2010], van Dongen et al. [2007], Sole and Carmona [2010] it is shown how these state-based regions can be applied to process mining. In parallel, several authors applied language-based regions to process mining [Bergenthum et al. 2007; Werf et al. 2010]. The basic idea of these approaches is to discover places. Note that the addition of places limits the behavior of the Petri net. The idea is to add places that do not exclude any of the behavior seen in the event log.

For practical applications of process discovery it is essential that *noise* and *incompleteness* are handled well. Surprisingly, only few discovery algorithms focus on addressing these issues. Notable exceptions are heuristic mining [Weijters and van der Aalst 2003], fuzzy mining [Günther and van der Aalst 2007], and genetic process mining [Medeiros et al. 2007].

ProM's *heuristic miner* uses the algorithm described in Weijters and van der Aalst [2003] (see also Section 6.2 in van der Aalst [2011]). The algorithm first builds a dependency graph based on the frequencies of activities and the number of times one activity is followed by another activity. Based on predefined thresholds, dependencies are added to the dependency graph graph (or not). The dependency graph reveals the "backbone" of the process model. This backbone is used to discover the detailed split and join behavior of nodes. If an activity has multiple input arcs, then the heuristic miner analyzes the log to see whether the join is an AND-join, an XOR-join or an OR-join. In case of an OR-join, the detailed synchronization behavior is learned. If an activity has multiple output arcs, then the "split behavior" is learned in a similar fashion.

See Chapter 6 of van der Aalst [2011] for a more elaborate introduction to the various process discovery approaches described in literature.

4. CONFORMANCE

In recent years, powerful process mining techniques have been developed that can automatically construct a suitable process model given an event log. Whereas process discovery constructs a model without any a priori information (other than the event log), conformance checking uses a model and an event log as input. The model may have been made by hand or discovered through process discovery. For conformance checking, the modeled behavior and the observed behavior (i.e., event log) are compared.

4.1 Applications of Conformance Checking

Conformance checking techniques relate events in the log to activities in the model, For instance, events are mapped to transition firings in the Petri net. This way it is possible to compare the observed behavior in the event log and the modeled behavior. For example, one can quantify differences (e.g., "80% of the observed cases are possible according to the model") and diagnose deviations (e.g., "in reality activity x is often

skipped although the model does not allow for this"). Conformance checking can be used

— to check the quality of documented processes (asses whether they describe reality accurately),
— to identify deviating cases and understand what they have in common,
— to identify process fragments where most deviations occur,
— for auditing purposes,
— to judge the quality of a discovered process model,
— to guide evolutionary process discovery algorithms (e.g., genetic algorithms need to continuously evaluate the quality of newly created models using conformance checking), and
— as a starting point for model enhancement.

This list shows that conformance checking can be used for a variety of reasons ranging from evaluating a process discovery algorithm to auditing and compliance monitoring. Note that auditors need to validate information about organizations by determining whether they execute business processes within certain boundaries set by managers, governments, and other stakeholders. Clearly, event logs provide valuable input for this.

4.2 Diagnosing Differences Between Observed Behavior and Modeled Behavior

Typically, four quality dimensions for comparing model and log are considered: (a) *fitness*, (b) *simplicity*, (c) *precision*, and (d) *generalization* (Chapter 7 of van der Aalst [2011]).

A model with good *fitness* allows for most of the behavior seen in the event log. A model has perfect fitness if all traces in the log can be replayed by the model from beginning to end. Often fitness is described by a number between 0 (very poor fitness) and 1 (perfect fitness). Obviously, the *simplest* model that can explain the behavior seen in the log is the best model. This principle is known as Occam's Razor.

Fitness and simplicity alone are not sufficient to judge the quality of a discovered process model. For example, it is very easy to construct an extremely simple Petri net ("flower model") that is able to replay all traces in an event log (but also any other event log referring to the same set of activities). Similarly, it is often undesirable to have a model that only allows for the exact behavior seen in the event log. Remember that the log contains only example behavior and that many traces that are possible may not have been seen yet. (Note that in our simple example most traces are frequent, but often there are many "one-of-a-kind" traces.)

A model is *precise* if it does not allow for "too much" behavior. Clearly, the "flower model" lacks precision. A model that is not precise is "underfitting." Underfitting is the problem that the model over-generalizes the example behavior in the log (i.e., the model allows for behaviors very different from what was seen in the log).

A model should also *generalize* and not restrict behavior to just the examples seen in the log. A model that does not generalize sufficiently is "overfitting." Overfitting is the problem that a very specific model is generated whereas it is obvious that the log only holds example behavior (i.e., the model explains the particular sample log, but a next sample log of the same process should not produce a completely different process model).

The four quality dimensions for comparing model and log can be quantified in various ways [Adriansyah et al. 2011; Munoz-Gama and Carmona 2011; Rozinat and van der Aalst 2008; van der Aalst 2011; van der Aalst et al. 2012].

4.3 Conformance Checking Algorithms

Basically, there are three approaches to conformance checking.

The first approach is to create an abstraction of the behavior in the log and an abstraction of the behavior allowed by the model. An example is the notion of a *footprint* described in Section 7.3 of van der Aalst [2011]. A footprint is a matrix showing causal dependencies between activities. For example, the footprint of an event log may show that x is sometimes followed by y but never the other way around. If the footprint of the corresponding model shows that x is never followed by y or that y is sometimes followed by x, then the footprints of event log and model disagree on the ordering relation of x and y.

The second approach *replays* the event log on the model. A naive approach towards conformance checking would be to simply count the fraction of cases that can be "parsed completely" (i.e., the proportion of cases corresponding to firing sequences leading from the initial state to the final state). This approach cannot distinguish between an "almost fitting" case and a case that is completely unrelated to the modeled behavior. A better approach is to continue replaying the event log on the model even when transitions are not enabled. Simply "borrow tokens," force the transition to fire anyway, and record the problem. In the end, the number of "borrowed tokens" and the number of "tokens left behind" (not consumed) indicate the fitness level. See Rozinat and van der Aalst [2008] and Section 7.2 in van der Aalst [2011].

The third, and most advanced, approach is to compute an optimal *alignment* between each trace in the log and the most similar behavior in the model. Consider for example the following three alignments between the example log and model M_2:

$$\gamma_1 = \begin{array}{|c|c|c|c|c|}\hline a & c & d & e & h \\ \hline a & c & d & e & h \\ \hline \end{array} \quad \text{and} \quad \gamma_2 = \begin{array}{|c|c|c|c|c|}\hline a & d & c & e & h \\ \hline a & \gg & c & e & h \\ \hline \end{array} \quad \text{and} \quad \gamma_3 = \begin{array}{|c|c|c|c|c|c|c|c|c|}\hline a & d & c & e & f & d & b & e & h \\ \hline a & \gg & c & e & f_2 & \gg & b & e & h \\ \hline \end{array}.$$

γ_1 shows a perfect alignment: all moves of the trace in the event log (top part of alignment) can be followed by moves of the model (bottom part of alignment). γ_2 shows an optimal alignment for trace *adceh* in the event log and model M_2. The first move of the trace in the event log can be followed by the model (event a). However, in the second position of the alignment, we see a move of the trace in the event log which cannot be mimicked by the model. This move in just the log is denoted as $(d, \gg)$. γ_3 shows an optimal alignment for trace *adcefdbeh* in the event log and model M_2. Here, we encounter two situations where log and model cannot move together. Also note the move (f, f_2), that is, event f in the log corresponds to the execution of transition f_2. Alignments γ_2 and γ_3 clearly show the reasons for non-conformance between model and log. Such problems can easily be quantified as shown in van der Aalst et al. [2012] and Adriansyah et al. [2011].

Conformance can be viewed from two angles: (a) the model does not capture the real behavior ("the model is wrong") and (b) reality deviates from the desired model "the event log is wrong"). The first viewpoint is taken when the model is supposed to be *descriptive*, that is, capture or predict reality. The second viewpoint is taken when the model is *normative*, that is, used to influence or control reality.

5. ENHANCEMENT

It is also possible to extend or improve an existing process model using the event log. A non-fitting process model can be corrected using the diagnostics provided by the alignment of model and log. Moreover, event logs may contain information about resources, timestamps, and case data. For example, an event referring to activity "register request" and case "992564" may also have attributes describing the person that registered the request (e.g., "John"), the time of the event (e.g., "30-01-2012:14.55"), the age

of the customer (e.g., "45"), and the claimed amount (e.g., "650 euro"). After aligning model and log it is possible to replay the event log on the model. While replaying one can analyze these additional attributes.

For example, it is possible to analyze waiting times in-between activities. Simply measure the time difference between causally related events and compute basic statistics such as averages, variances, and confidence intervals. This way it is possible to identify the main bottlenecks [van der Aalst 2011].

Information about resources can be used to discover roles, that is, groups of people frequently executing related activities. Here, standard clustering techniques can be used. It is also possible to construct social networks based on the flow of work and analyze resource performance (e.g., the relation between workload and service times). See Song and van der Aalst [2008] for an overview of various process mining techniques analyzing the organizational perspective based on event logs.

Standard classification techniques can be used to analyze the decision points in the process model [Rozinat and van der Aalst 2006]. For example, activity *e* ("decide") has three possible outcomes ("pay," "reject," and "redo"). Using the data known about the case prior to the decision, we can construct a decision tree explaining the observed behavior.

Process mining is not restricted to offline analysis and can also be used for predictions and recommendations at runtime. For example, the completion time of a partially handled customer order can be predicted using a discovered process model with timing information [van der Aalst et al. 2011b].

6. PROCESS MINING MANIFESTO

The IEEE Task Force on Process Mining recently released a manifesto describing guiding principles and challenges [TFPM 2011]. The manifesto aims to increase the visibility of process mining as a new tool to improve the (re)design, control, and support of operational business processes. It is intended to guide software developers, scientists, consultants, and end-users. Before summarizing the manifesto, we briefly introduce the task force.

6.1 Task Force on Process Mining

The growing interest in log-based process analysis motivated the establishment of the IEEE Task Force on Process Mining. The goal of this task force is to promote the research, development, education, and understanding of process mining. The task force was established in 2009 in the context of the Data Mining Technical Committee of the Computational Intelligence Society of the IEEE. Members of the task force include representatives of more than a dozen commercial software vendors (e.g., Pallas Athena, Software AG, Futura Process Intelligence, HP, IBM, Fujitsu, Infosys, and Fluxicon), ten consultancy firms (e.g., Gartner and Deloitte) and over twenty universities.

Concrete objectives of the task force are: to make end-users, developers, consultants, managers, and researchers aware of the state-of-the-art in process mining, to promote the use of process mining techniques and tools, to stimulate new process mining applications, to play a role in standardization efforts for logging event data, to organize tutorials, special sessions, workshops, panels, and to publish articles, books, videos, and special issues of journals. For example, in 2010 the task force standardized *XES*[1], a standard logging format that is extensible and supported by the *OpenXES library*[2] and by tools such as ProM, XESame, Nitro, etc. See `http://www.win.tue.nl/ieeetfpm/` for recent activities of the task force.

[1] `www.xes-standard.org`
[2] `www.openxes.org`

Table I. Six Guiding Principles Listed in the Manifesto

GP1	**Event Data Should Be Treated as First-Class Citizens** Events should be *trustworthy*, that is, it should be safe to assume that the recorded events actually happened and that the attributes of events are correct. Event logs should be *complete*, that is, given a particular scope, no events may be missing. Any recorded event should have well-defined *semantics*. Moreover, the event data should be *safe* in the sense that privacy and security concerns are addressed when recording the event log.
GP2	**Log Extraction Should Be Driven by Questions** Without concrete questions it is very difficult to extract meaningful event data. Consider, for example, the thousands of tables in the database of an ERP system like SAP. Without questions one does not know where to start.
GP3	**Concurrency, Choice and Other Basic Control-Flow Constructs Should Be Supported** Basic workflow *patterns* supported by all mainstream languages (e.g., BPMN, EPCs, Petri nets, BPEL, and UML activity diagrams) are *sequence*, *parallel routing* (AND-splits/joins), *choice* (XOR-splits/joins), and *loops*. Obviously, these patterns should be supported by process mining techniques.
GP4	**Events Should Be Related to Model Elements** Conformance checking and enhancement heavily rely on the relationship between *elements in the model* and *events in the log*. This relationship may be used to "replay" the event log on the model. Replay can be used to reveal discrepancies between event log and model (e.g., some events in the log are not possible according to the model) and can be used to enrich the model with additional information extracted from the event log (e.g., bottlenecks are identified by using the timestamps in the event log).
GP5	**Models Should Be Treated as Purposeful Abstractions of Reality** A model derived from event data provides a *view on reality*. Such a view should serve as a purposeful abstraction of the behavior captured in the event log. Given an event log, there may be multiple views that are useful.
GP6	**Process Mining Should Be a Continuous Process** Given the dynamical nature of processes, it is not advisable to see process mining as a one-time activity. The goal should not be to create a fixed model, but to breathe life into process models such that users and analysts are encouraged to look at them on a daily basis.

6.2 Guiding Principles

As with any new technology, there are obvious mistakes that can be made when applying process mining in real-life settings. Therefore, the six guiding principles listed in Table I aim to prevent users/analysts from making such mistakes. As an example, consider Guiding Principle GP4: "Events Should Be Related to Model Elements." It is a misconception that process mining is limited to control-flow discovery, other perspectives such as the organizational perspective, the time perspective, and the data perspective are equally important. However, the control-flow perspective (i.e., the ordering of activities) serves as the layer connecting the different perspectives. Therefore, it is important to relate events in the log to activities in the model. Conformance checking and model enhancement heavily rely on this relationship. Using Figure 2, we showed for example alignment γ_3 which relates observed trace $adcefdbeh$ to firing sequence $acef_2beh$ in M_2. After relating events to model elements, it is possible to "replay" the event log on the model [van der Aalst 2011]. Replay may be used to reveal discrepancies between an event log and a model, for instance, some events in the log are not possible according to the model. Techniques for conformance checking can be used to quantify and diagnose such discrepancies. Timestamps in the event log can be used to analyze the temporal behavior during replay. Time differences between causally related activities can be used to add average/expected waiting times to the model. These examples illustrate the importance of guiding principle GP4; the

relation between events in the log and elements in the model serves as a starting point for different types of analysis.

6.3 Challenges

Process mining is an important tool for modern organizations that need to manage nontrivial operational processes. On the one hand, there is an incredible growth of event data. On the other hand, processes and information need to be aligned perfectly in order to meet requirements related to compliance, efficiency, and customer service. Despite the applicability of process mining there are still important challenges that need to be addressed; these illustrate that process mining is an emerging discipline. Table II lists the eleven challenges described in the manifesto [TFPM 2011].

As an example consider Challenge *C4*: "Dealing with Concept Drift." The term *concept drift* refers to the situation in which the process is changing while being analyzed [Bose et al. 2011]. For instance, in the beginning of the event log two activities may be concurrent whereas later in the log these activities become sequential. Processes may change due to periodic/seasonal changes (e.g., "in December there is more demand" or "on Friday afternoon there are fewer employees available") or due to changing conditions (e.g., "the market is getting more competitive"). Such changes impact processes and it is vital to detect and analyze them [Bose et al. 2011].

7. PROCESS MINING IN PRACTICE

Although the manifesto lists many open challenges, existing process mining techniques can easily be applied in practice. At TU/e (Eindhoven University of Technology) we have applied process mining in over 100 organizations. To help the reader to get started with process mining, we briefly discuss tool support and show two case studies taken from van der Aalst [2011].

7.1 Tool Support

The open-source tool *ProM* has been the de-facto standard for process mining during the last decade. Process discovery, conformance checking, social network analysis, organizational mining, decision mining, history-based prediction and recommendation, etc. are all supported by ProM [van der Aalst 2011; Verbeek et al. 2010]. For example, dozens of different process discovery algorithms are supported by ProM. The functionality of ProM is unprecedented, that is, there is no product offering a comparable set of process mining algorithms. However, the tool requires process mining expertise and is not supported by a commercial organization. Hence, it has the advantages and disadvantages common for open-source software.

Fortunately, there are also a growing number of commercially available software products offering process mining capabilities. Examples are: ARIS Process Performance Manager (Software AG), Comprehend (Open Connect), Discovery Analyst (StereoLOGIC), Flow (Fourspark), Futura Reflect (Futura Process Intelligence), Interstage Automated Process Discovery (Fujitsu), Process Discovery Focus (Iontas/Verint), ProcessAnalyzer (QPR), and Reflect|one (Pallas Athena).

All of the products mentioned support process discovery, that is, constructing a process model based on an event log. For example, Futura Reflect supports genetic process mining as described in Medeiros et al. [2007]. Some of the systems mentioned have difficulties discovering concurrency, for instance, ARIS Process Performance Manager, Flow, and Interstage Automated Process Discovery. All systems take the timestamps in the event log into account to be able to provide performance-related information, that is, flow times and bottlenecks can be discovered.

Table II. Some of the Most Important Process Mining Challenges Identified in the Manifesto

C1	**Finding, Merging, and Cleaning Event Data** When extracting event data suitable for process mining several challenges need to be addressed: data may be *distributed* over a variety of sources, event data may be *incomplete*, an event log may contain *outliers*, logs may contain events at *different level of granularity*, etc.
C2	**Dealing with Complex Event Logs Having Diverse Characteristics** Event logs may have very different characteristics. Some event logs may be extremely large making them difficult to handle whereas other event logs are so small that not enough data is available to make reliable conclusions.
C3	**Creating Representative Benchmarks** Good benchmarks consisting of example data sets and representative quality criteria are needed to compare and improve the various tools and algorithms.
C4	**Dealing with Concept Drift** The process may be changing while being analyzed. Understanding such concept drifts is of prime importance for the management of processes.
C5	**Improving the Representational Bias Used for Process Discovery** A more careful and refined selection of the representational bias is needed to ensure high-quality process mining results.
C6	**Balancing Between Quality Criteria such as Fitness, Simplicity, Precision, and Generalization** There are four competing quality dimensions: (a) fitness, (b) simplicity, (c) precision, and (d) generalization. The challenge is to find models that score good in all four dimensions.
C7	**Cross-Organizational Mining** There are various use cases where event logs of multiple organizations are available for analysis. Some organizations work together to handle process instances (e.g., supply chain partners) or organizations are executing essentially the same process while sharing experiences, knowledge, or a common infrastructure. However, traditional process mining techniques typically consider one event log in one organization.
C8	**Providing Operational Support** Process mining is not restricted to off-line analysis and can also be used for online operational support. Three operational support activities can be identified: *detect*, *predict*, and *recommend*.
C9	**Combining Process Mining with Other Types of Analysis** The challenge is to combine automated process mining techniques with other analysis approaches (optimization techniques, data mining, simulation, visual analytics, etc.) to extract more insights from event data.
C10	**Improving Usability for Nonexperts** The challenge is to hide the sophisticated process mining algorithms behind user-friendly interfaces that automatically set parameters and suggest suitable types of analysis.
C11	**Improving Understandability for Nonexperts** The user may have problems understanding the output or is tempted to infer incorrect conclusions. To avoid such problems, the results should be presented using a suitable representation and the trustworthiness of the results should always be clearly indicated.

None of the commercial software products provides comprehensive support for conformance checking, that is, the focus is on process discovery and performance measurement. However, ProM supports the different types of conformance checking described in Section 4.3.

Some of these products embed process mining functionality in a larger system, for instance, Pallas Athena embeds process mining in their BPM suite BPM|one. Other products aim at simplifying process mining using an intuitive user interface.

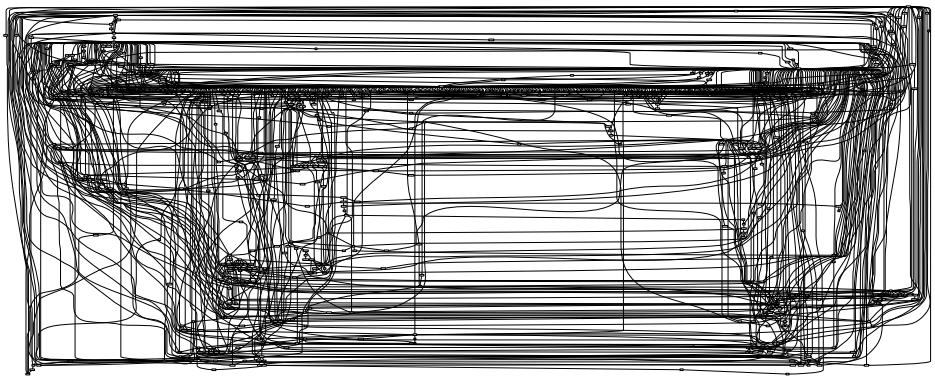

Fig. 3. Spaghetti process describing the diagnosis and treatment of 2765 patients in a Dutch hospital. The process model was constructed based on an event log containing 114,592 events. There are 619 different activities (taking event types into account) executed by 266 different individuals (doctors, nurses, etc.).

7.2 Discovering "Spaghetti Processes"

There is a continuum of processes ranging from highly structured processes (lasagna processes) to unstructured processes (spaghetti processes). Figure 3 shows why unstructured processes are often called "spaghetti processes." The model was obtained using ProM's heuristic miner [Weijters and van der Aalst 2003]. Hence, low frequent behavior has been filtered out. Nevertheless, the model is too difficult to comprehend. Note that this is not necessarily a problem of the discovery algorithm. Activities are only connected if they frequently followed one another in the event log. Hence, the complexity shown in Figure 3 reflects reality and is not caused by the discovery algorithm.

Figure 3 is an extreme example used to illustrate the characteristics of a typical spaghetti process. Given the data set it is not surprising that the process is unstructured; the 2765 patients did not form a homogeneous group and included individuals with very different medical problems. The process model in Figure 3 can be simplified dramatically by selecting a group of patients with similar problems or by selecting only the most frequent activities. Nevertheless, its complexity exemplifies some of the challenges mentioned in the manifesto (in particular C1, C2, C6, C10, and C11).

7.3 Analyzing "Lasagna Processes"

Processes in municipalities are typically lasagna processes. Figure 4 shows a so-called WOZ process discovered for a Dutch municipality. We applied the heuristic miner [Weijters and van der Aalst 2003] on an event log containing information about 745 objections against the so-called WOZ ("Waardering Onroerende Zaken," that is, Valuation of Real Estate) valuation. Dutch municipalities need to estimate the value of houses and apartments. The WOZ value is used as a basis for determining the real-estate property tax. The higher the WOZ value, the more tax the owner needs to pay. Therefore, Dutch municipalities need to handle many objections (i.e., appeals) of citizens that assert that the WOZ value is too high. Figure 4 shows the process of handling these objections within a particular municipality. The diagram is not intended to be readable; it is only included to show the contrast with Figure 3.

The discovered WF-net has a good fitness: 628 of the 745 cases can be replayed without encountering any problems. The fitness of the model and log at the event level is 0.98876214. This value is based on the approach described in van der Aalst [2011] and Rozinat and van der Aalst [2008]. The high value shows that almost all recorded events are explained by the model. Hence, the WOZ process is clearly a lasagna

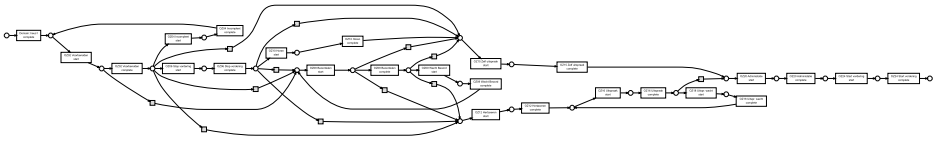

Fig. 4. WF-net discovered based on an event log of a Dutch municipality. The log contains events related to 745 objections against the so-called WOZ valuation. These 745 objections generated 9583 events. There are 13 activities. For 12 of these activities both start and complete events are recorded. Hence, the WF-net has 25 transitions.

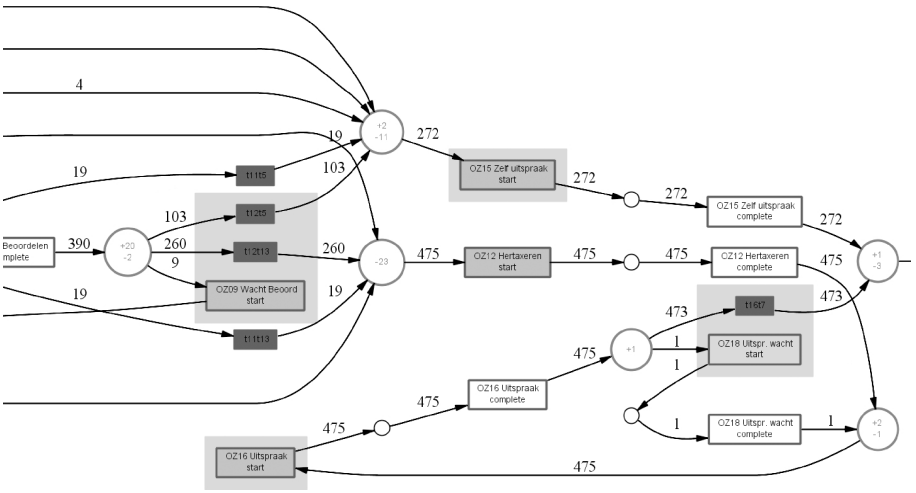

Fig. 5. Fragment of the WF-net annotated with diagnostics generated by ProM's conformance checker. The WF-net and event log fit well (fitness is 0.98876214). Nevertheless, several low-frequent deviations are discovered. For example. activity "OZ12 Hertaxeren" (re-evaluation of WOZ value) is started 23 times without being enabled according to the model.

process. Nevertheless, it was interesting for the municipality to see the deviations highlighted in the model. Figure 5 shows a fragment of the diagnostics provided by the ProM's conformance checker.

The municipality's log contains timestamps. Therefore, it is possible to replay the event log while taking the timestamps into account. ProM can visualize the phases of the process that take most time. For example, the place in-between "OZ16 Uitspraak start" (start of announcement of final judgment) and "OZ16 Uitspraak complete" (end of announcement of final judgment) was visited 436 times. The average time spent in this place is 7.84 days. This indicates that activity "OZ16 Uitspraak" (final judgment) takes about a week. It is also possible to simply select two activities and measure the time that passes in-between these activities. On average 202.73 days pass in-between the completion of activity "OZ02 Voorbereiden" (preparation) and the completion of "OZ16 Uitspraak" (final judgment). Such examples illustrate that process mining, unlike classical Business Intelligence (BI) tools, helps organizations to "look inside" their processes. This is in stark contrast with contemporary BI tools that typically focus on reporting and fancy looking dashboards.

8. CONCLUSION

This article introduced process mining as a new technology enabling evidence-based process analysis. We introduced the three basic types of process mining (discovery,

conformance, and enhancement) using a small example and used some larger examples to illustrate the applicability in real-life settings. Nevertheless, there are still many open scientific challenges and most end-user organizations are not yet aware of the potential of process mining. This triggered the development of the *Process Mining Manifesto* by an international task force involving 77 process mining experts representing 53 organizations. This manifesto can be obtained from `http://www.win.tue.nl/ieeetfpm/`. The reader interested in process mining is also referred to the recent book on process mining [van der Aalst 2011]. Also visit `www.processmining.org` for sample logs, videos, slides, articles, and software.

ACKNOWLEDGMENTS

The author would like to thank all who contributed to the Process Mining Manifesto: Arya Adriansyah, Ana Karla Alves de Medeiros, Franco Arcieri, Thomas Baier, Tobias Blickle, Jagadeesh Chandra Bose, Peter van den Brand, Ronald Brandtjen, Joos Buijs, Andrea Burattin, Josep Carmona, Malu Castellanos, Jan Claes, Jonathan Cook, Nicola Costantini, Francisco Curbera, Ernesto Damiani, Massimiliano de Leoni, Pavlos Delias, Boudewijn van Dongen, Marlon Dumas, Schahram Dustdar, Dirk Fahland, Diogo R. Ferreira, Walid Gaaloul , Frank van Geffen, Sukriti Goel, Christian Günther, Antonella Guzzo, Paul Harmon, Arthur ter Hofstede, John Hoogland, Jon Espen Ingvaldsen, Koki Kato, Rudolf Kuhn, Akhil Kumar, Marcello La Rosa, Fabrizio Maggi, Donato Malerba, Ronny Mans, Alberto Manuel, Martin McCreesh, Paola Mello, Jan Mendling, Marco Montali, Hamid Motahari Nezhad, Michael zur Muehlen, Jorge Munoz-Gama, Luigi Pontieri, Joel Ribeiro, Anne Rozinat, Hugo Seguel Pérez, Ricardo Seguel Pérez, Marcos Sepúlveda, Jim Sinur, Pnina Soffer, Minseok Song, Alessandro Sperduti, Giovanni Stilo, Casper Stoel, Keith Swenson, Maurizio Talamo, Wei Tan, Chris Turner, Jan Vanthienen, George Varvaressos, Eric Verbeek, Marc Verdonk, Roberto Vigo, Jianmin Wang, Barbara Weber, Matthias Weidlich, Ton Weijters, Lijie Wen, Michael Westergaard, and Moe Wynn.

REFERENCES

ADRIANSYAH, A., VAN DONGEN, B., AND VAN DER AALST, W. 2011. Conformance checking using cost-based fitness analysis. In *Proceedings of the IEEE International Enterprise Computing Conference (EDOC'11)*. C. Chi and P. Johnson Eds., IEEE Computer Society, 55–64.

AGRAWAL, R., GUNOPULOS, D., AND LEYMANN, F. 1998. Mining process models from workflow logs. In *Proceedings of the 6th International Conference on Extending Database Technology*. Lecture Notes in Computer Science, vol. 1377, Springer-Verlag, Berlin, 469–483.

BERGENTHUM, R., DESEL, J., LORENZ, R., AND MAUSER, S. 2007. Process mining based on regions of languages. In *Proceedings of the International Conference on Business Process Management (BPM'07)*. G. Alonso, P. Dadam, and M. Rosemann Eds., Lecture Notes in Computer Science, vol. 4714, Springer, 375–383.

BOSE, R. P. J. C., VAN DER AALST, W., ZLIOBAITE, I., AND PECHENIZKIY, M. 2011. Handling concept drift in process mining. In *Proceedings of the International Conference on Advanced Information Systems Engineering (CAISE'11)*. H. Mouratidis and C. Rolland Eds., Lecture Notes in Computer Science, vol. 6741, Springer, 391–405.

COOK, J. AND WOLF, A. 1998. Discovering models of software processes from event-based data. *ACM Trans. Softw. Engin. Method. 7*, 3, 215–249.

CORTADELLA, J., KISHINEVSKY, M., LAVAGNO, L., AND YAKOVLEV, A. 1998. Deriving Petri nets from finite transition systems. *IEEE Trans. Comput. 47*, 8, 859–882.

DATTA, A. 1998. Automating the discovery of as-is business process models: Probabilistic and algorithmic approaches. *Inf. Syst. Resear. 9*, 3, 275–301.

DESEL, J. AND REISIG, W. 1998. Place/Transition Nets. In Lectures on Petri Nets I: Basic Models, W. Reisig and G. Rozenberg Eds., Lecture Notes in Computer Science, vol. 1491, Springer-Verlag, Berlin, 122–173.

EHRENFEUCHT, A. AND ROZENBERG, G. 1989. Partial (set) 2-structures: Parts 1 Part 2. *Acta Informatica 27*, 4, 315–368.

GRECO, G., GUZZO, A., PONTIERI, L., AND SACCÀ, D. 2006. Discovering expressive process models by clustering log traces. *IEEE Trans. Knowl. Data Engin. 18*, 8, 1010–1027.

GÜNTHER, C. AND VAN DER AALST, W. 2007. Fuzzy mining: Adaptive process simplification based on multi-perspective metrics. In *Proceedings of the International Conference on Business Process Management*

(BPM'07). G. Alonso, P. Dadam, and M. Rosemann Eds., Lecture Notes in Computer Science, vol. 4714, Springer-Verlag, Berlin, 328–343.

HAND, D., MANNILA, H., AND SMYTH, P. 2001. *Principles of Data Mining*. MIT Press, Cambridge, MA.

HERBST, J. 2000. A machine learning approach to workflow management. In *Proceedings of the 11th European Conference on Machine Learning*. Lecture Notes in Computer Science, vol. 1810, Springer, 183–194.

MANYIKA, J., CHUI, M., BROWN, B., BUGHIN, J., DOBBS, R., ROXBURGH, C., AND BYERS, A. 2011. *Big Data: The Next Frontier for Innovation, Competition, and Productivity.* McKinsey Global Institute.

MEDEIROS, A., WEIJTERS, A., AND VAN DER AALST, W. 2007. Genetic process mining: An experimental evaluation. *Data Mining Knowl. Discov. 14*, 2, 245–304.

MUNOZ-GAMA, J. AND CARMONA, J. 2011. Enhancing precision in process conformance: Stability, confidence and severity. In *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining (CIDM'11)*. N. Chawla, I. King, and A. Sperduti Eds., IEEE.

ROZINAT, A. AND VAN DER AALST, W. 2006. Decision mining in ProM. In *Proceedings of the International Conference on Business Process Management (BPM'06)*. S. Dustdar, J. Fiadeiro, and A. Sheth Eds., Lecture Notes in Computer Science, vol. 4102, Springer, 420–425.

ROZINAT, A. AND VAN DER AALST, W. 2008. Conformance checking of processes based on monitoring real behavior. *Inf. Syst. 33*, 1, 64–95.

SOLE, M. AND CARMONA, J. 2010. Process mining from a basis of regions. In *Applications and Theory of Petri Nets 2010*. J. Lilius and W. Penczek Eds., Lecture Notes in Computer Science, vol. 6128, Springer 226–245.

SONG, M. AND VAN DER AALST, W. 2008. Towards comprehensive support for organizational mining. *Dec. Support Syst. 46*, 1, 300–317.

TFPM – IEEE TASK FORCE ON PROCESS MINING. 2011. Process mining manifesto. In *Proceedings of the BPM Workshops*. Lecture Notes in Business Information Processing Series, vol. 99, Springer.

VAN DER AALST, W. 2011. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer.

VAN DER AALST, W. AND STAHL, C. 2011. *Modeling Business Processes: A Petri Net Oriented Approach*. MIT Press, Cambridge, MA.

VAN DER AALST, W., VAN DONGEN, B., HERBST, J., MARUSTER, L., SCHIMM, G., AND WEIJTERS, A. 2003. Workflow mining: A survey of issues and approaches. *Data Knowl. Engin. 47,* 2, 237–267.

VAN DER AALST, W., WEIJTERS, A., AND MARUSTER, L. 2004. Workflow mining: Discovering process models from event logs. *IEEE Trans. Knowl. Data Engin. 16*, 9, 1128–1142.

VAN DER AALST, W., REIJERS, H., WEIJTERS, A., VAN DONGEN, B., MEDEIROS, A., SONG, M., AND VERBEEK, H. 2007. Business process mining: An industrial application. *Inf. Syst. 32*, 5, 713–732.

VAN DER AALST, W., RUBIN, V., VERBEEK, H., VAN DONGEN, B., KINDLER, E., AND GÜNTHER, C. 2010. Process mining: A two-step approach to balance between underfitting and overfitting. *Softw. Syst. Model. 9*, 1, 87–111.

VAN DER AALST, W., VAN HEE, K., HOFSTEDE, A., SIDOROVA, N., VERBEEK, H., VOORHOEVE, M., AND WYNN, M. 2011a. Soundness of workflow nets: Classification, decidability, and analysis. *Formal Asp. Comput. 23*, 3, 333–363.

VAN DER AALST, W., SCHONENBERG, M., AND SONG, M. 2011b. Time prediction based on process mining. *Inf. Syst. 36*, 2, 450–475.

VAN DER AALST, W., ADRIANSYAH, A., AND VAN DONGEN, B. 2012. Replaying history on process models for conformance checking and performance analysis. *WIREs Data Mining Knowl. Discov. 2*, 2, 182–192.

VAN DONGEN, B. AND VAN DER AALST, W. 2004. Multi-phase process mining: Building instance graphs. In *Proceedings of the International Conference on Conceptual Modeling (ER'04)*. P. Atzeni, W. Chu, H. Lu, S. Zhou, and T. Ling Eds., Lecture Notes in Computer Science, vol. 3288, Springer, 362–376.

VAN DONGEN, B. AND VAN DER AALST, W. 2005. Multi-phase mining: Aggregating instances graphs into EPCs and Petri nets. In *Proceedings of the 2nd International Workshop on Applications of Petri Nets to Coordination, Workflow and Business Process Management*. D. Marinescu Ed., 35–58.

VAN DONGEN, B., BUSI, N., PINNA, G., AND VAN DER AALST, W. 2007. An iterative algorithm for applying the theory of regions in process mining. In *Proceedings of the Workshop on Formal Approaches to Business Processes and Web Services (FABPWS'07)*. W. Reisig, K. Hee, and K. Wolf Eds., Publishing House of University of Podlasie, Siedlce, Poland, 36–55.

VERBEEK, H., BUIJS, J., VAN DONGEN, B., AND VAN DER AALST, W. 2010. ProM 6: The process mining toolkit. In *Proceedings of BPM Demonstration Track 2010*. M. L. Rosa Ed., CEUR Workshop Proceedings Series, vol. 615, 34–39.

WEIJTERS, A. AND VAN DER AALST, W. 2003. Rediscovering workflow models from event-based data using Little Thumb. *Integr. Comput.-Aid. Engin. 10*, 2, 151–162.

WERF, J., DONGEN, B. VAN, HURKENS, C., AND SEREBRENIK, A. 2010. Process discovery using integer linear programming. *Fundamenta Informaticae 94*, 387–412.

WESKE, M. 2007. *Business Process Management: Concepts, Languages, Architectures*. Springer.

Received October 2011; revised January 2012; accepted January 2012